Fondamenti d'Informatica: Simulazione d'esame

Barbara Re, Phd

Parte teorica
(1 punto per ogni domanda)

Domande Parte Teorica

- Che cosa s'intende per teoria della computabilità? Cosa è computabile? Chi computa? Come si computa?

Macchine, Linguaggi e Regole

- La teoria della computabilità tratta della definizione formale del concetto di calcolo meccanico
  - Le macchine calcolatrici per attuare le computazioni sono di fatto macchine "calcolatrici meccaniche"
  - Si sviluppa sino alle moderne soluzioni elettroniche
- Fondamentale è la definizione di linguaggi e regole con cui favorire la collaborazione delle macchine
  - Identificare un linguaggio astratto appropriato, con simboli opportuni per cui formulare problemi da rivolvere, per poterli poi proporre alla macchina che li deve computare
  - Identificare le leggi che la macchina deve seguire per svolgere i suoi calcoli

Cosa, Chi, e Come?

- Cosa è computabile?
  - Definizione del problema e di un alfabeto appropriato con cui formalizzare i contenuti
- Chi computa?
  - L'algoritmo che affronta il problema e la macchina che lo svolge
- Come si computa?
  - Le regole che algoritmo e macchina devono rispettare nel loro lavoro

Domande Parte Teorica

- Definire i concetti di: sottostringa, prefisso, suffisso e inversa di una stringa. Fornire alcuni esempi.

Sottostringhe, prefissi, suffissi

- Una stringa α è una **sottostringa** di β se β= γαρ per qualche scelta di stringhe γρ
- Una sottostringa α di una stringa β si chiama **prefisso** di β se β = αρ per qualche stringa ρ (dunque per γ =λ)
- αè **un prefisso proprio** se ρ ≠λ
- Una sottostringa α di una stringaβè un **suffisso** di βse β= γα per qualche stringa γ (dunque per ρ =λ)
  - αè **un suffisso proprio** se γ ≠λ
- Se $\alpha = a_1 a_2 \ldots a_n$, allora $a_n \ldots a_2 a_1$ è chiamata **l'inversa** di α e denotata α^{rev}

Domande Parte Teorica

- Che cosa s'intende per "Il calcolo non è probabilistico"?

Condizioni di un sistema di computazione

1. Un algoritmo è di lunghezza finita
2. Esiste un agente di calcolo (la macchina calcolatrice) che sviluppa la computazione eseguendo le istruzioni dell'algoritmo
3. L'agente di calcolo ha a disposizione una memoria, dove vengono immagazzinati i risultati intermedi del calcolo
4. Il calcolo avviene per passi discreti
5. **Il calcolo non è probabilistico**
6. Non deve esserci alcun limite finito alla lunghezz
7. Non deve esserci alcun limite finito alla quantità di memoria disponibile
8. Deve esserci un limite finito alla complessità delle istruzioni eseguibili dal dispositivo
9. Il numero di passi della computazione è finito ma non limitato
10. Sono ammesse computazioni senza fine

Il calcolo non è soggetto a fattori casuali, ma si svolge in modo assolutamente preciso e rigoroso (anche detto deterministico)

Domande Parte Teorica

- Che cosa s'intende per Macchine di Turing? Quali sono gli elementi di base? Quali le funzionalità?

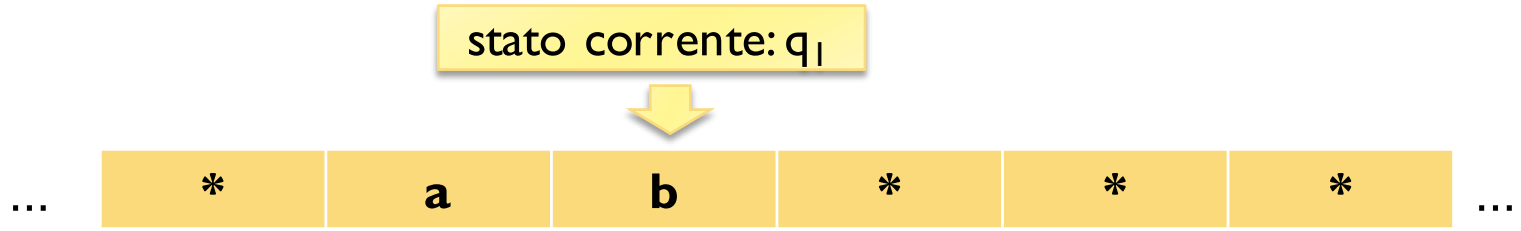
Elementi della MdT

- Un nastro di lunghezza infinita suddiviso in celle
  - Ogni cella contiene un simbolo dell'alfabeto oppure un simbolo bianco *
  - Il nastro contiene un numero finito di simboli tutto il resto del nastro contiene simboli bianchi

Una Memoria

- Una testina di **lettura** e **scrittura** che si muove lungo il nastro in entrambe le direzioni possibili
  - La testina ad ogni istante può indicare una sola cella del nastro
- Una unità di controllo a stati finiti, cioè controllata da una macchina a stati
  - Contiene il programma secondo cui viene eseguito il calcolo e mantiene lo stato della macchina

Una CPU



MdT è definita da una quintupla

- Una macchina di Turing M deterministica è una quintupla

$$(Q,\ A,\ \delta,\ q_0\ ,\ q_F)$$

- Q è l'insieme finito e non vuoto di stati della macchina
- A è un alfabeto, cui si aggiunge il carattere speciale detto **blank** (rappresentato da un asterisco) che rappresenta la situazione di cella non contenente alcun carattere
- $q_0 \in Q$ è lo stato iniziale
- $q_F \in Q$ è lo stato finale (possono essere definiti anche più di uno stato finale)
- δ è una **funzione di transizione** che fa evolvere la computazione della macchina da

$$\delta = (Q - q_F) \times (A \cup \{*\}) \rightarrow Q \times (A \cup \{*\}) \times \{dx, sx, i\}$$

Gli elementi (q, a, q', a', x) sono chiamati **regole di transizione** o **istruzioni di M**

Domande Parte Teorica

- Che cosa s'intende per Macchine di Turing Universale?

Macchina Universale

- Si dice **Macchina di Turing Universale (MTU)** una macchina di Turing capace di simulare le evoluzioni di ogni macchina di Turing

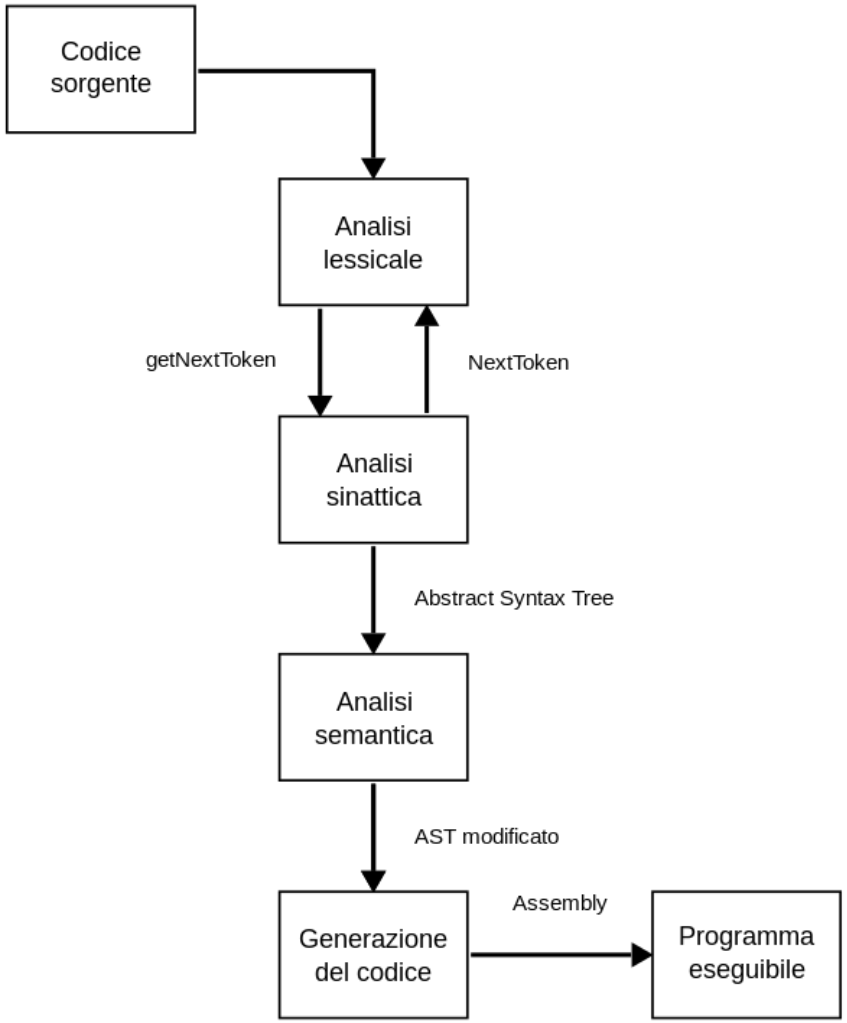
- La MTU universale è una MdT il cui input è composto dalla concatenazione di due elementi
  - La codifica della MdT da simulare - Stato iniziale, posizione iniziale della testina, regole di transizione
  - L'input di tale MdT - Nastro iniziale

- Rispetto alle semplici MT, la MTU è una **macchina calcolatrice programmabile**, mentre infatti le normali macchine di Turing eseguono un solo programma, che è "incorporato" nella tavola di transizione, la MTU assume in input il programma che deve eseguire

Domande Parte Teorica

- Quali sono le fasi tipiche della compilazione?

Fasi tipiche della compilazione

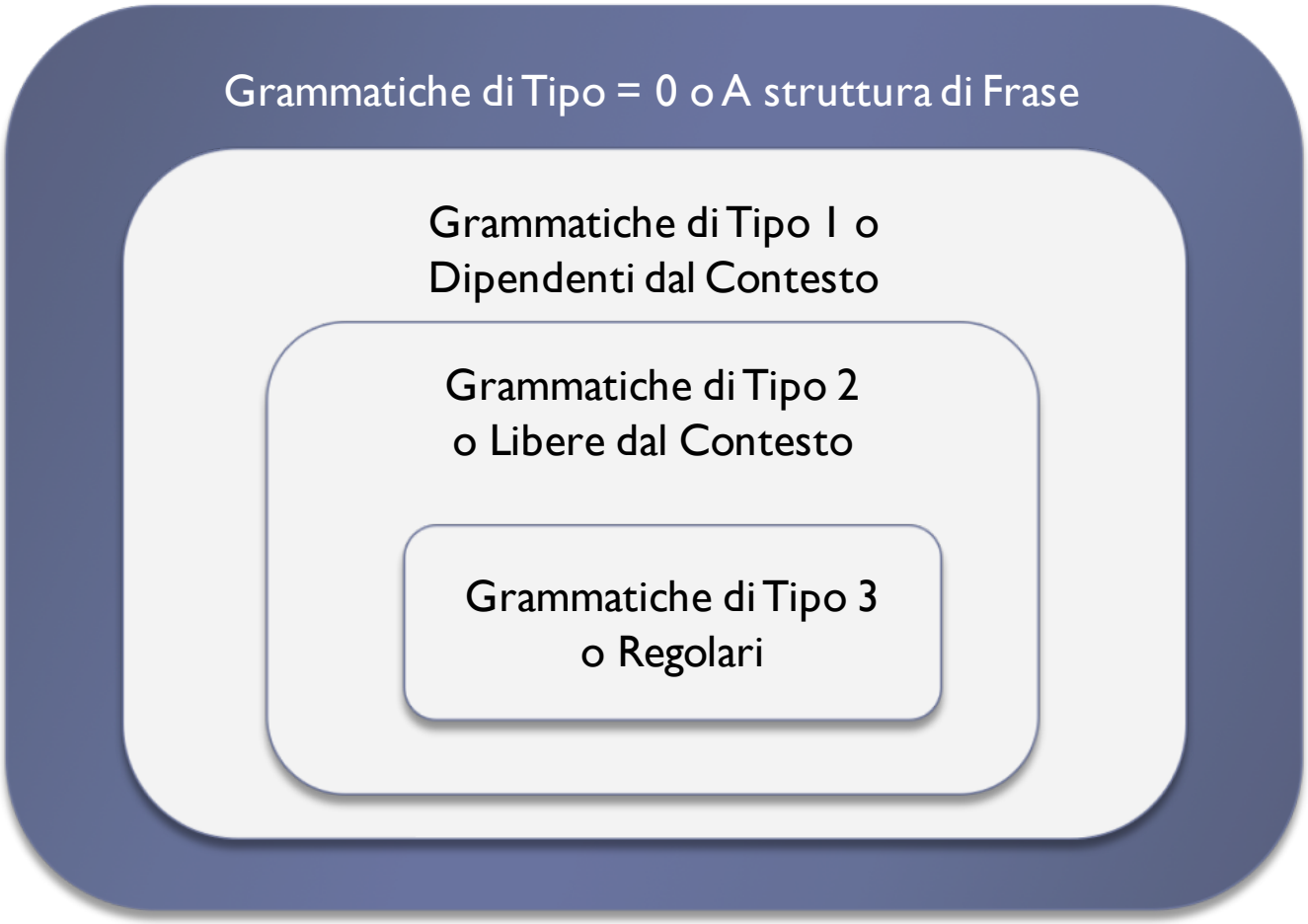


- **Analisi lessicale**
  - Attraverso un analizzatore lessicale il compilatore divide il codice sorgente in tanti pezzetti chiamati token. I token sono gli elementi minimi (non ulteriormente divisibili) di un linguaggio, ad esempio parole chiave (for, while), nomi di variabili (pippo), operatori (+, -, «)
- **Analisi sintattica**
  - L'analisi sintattica prende in ingresso la sequenza di token generata nella fase precedente ed esegue il controllo sintattico. Il controllo sintattico è effettuato attraverso una grammatica.
- **Analisi semantica**
  - L'analisi semantica si occupa di controllare il significato delle istruzioni presenti nel codice in ingresso. Controlli tipici di questa fase sono il type checking, ovvero il controllo di tipo, controllare che gli identificatori siano stati dichiarati prima di essere usati e così via. Come supporto a questa fase viene creata una tabella dei simboli che contiene informazioni su tutti gli elementi simbolici incontrati quali nome, scope, tipo (se presente) etc.

Domande Parte Teorica

- Presentare le varie tipologie di Grammatiche. Per ciascun tipo dare un esempio.

Tipologie di Grammatiche



Grammatiche di tipo 0 (o a struttura di frase)

- Le grammatiche di tipo 0, dette anche non limitate, definiscono la classe di linguaggi più ampia possibile
- In esse le produzioni sono del tipo più generale:

$$\alpha \rightarrow \beta\ , \quad \alpha \in V^* \circ V_N \circ V^*\ , \quad \beta \in V^* \quad \text{dove } V = (V_T \cup V_N)$$

- Si noti che queste grammatiche ammettono anche derivazioni che "accorciano" le forme di frase , come ad esempio quelle che si ottengono applicando le λ-produzioni
- I linguaggi generabili da grammatiche di tipo 0 si dicono linguaggi di tipo 0

Grammatiche di tipo 1 (o dipendenti dal contesto)

- Queste grammatiche, dette anche contestuali o context sensitive, ammettono qualunque regola di produzione che non riduca la lunghezza delle stringhe, cioè produzioni del tipo:

  $\alpha \rightarrow \beta, \quad \alpha \in V^* \circ V_N \circ V^*, \quad \beta \in V^+, \quad |\alpha| \leq |\beta|$

- I linguaggi generabili da grammatiche di tipo 1 si dicono linguaggi di tipo 1, o contestuali, o context sensitive

- Il termine "linguaggio contestuale", deriva dal fatto che, storicamente, questi linguaggi sono stati definiti da Chomsky come la classe dei linguaggi generabili da grammatiche aventi produzioni "contestuali" del tipo

  $\beta_1 A \beta_2 \rightarrow \beta_1 \gamma \beta_2, \; A \in V_N, \beta_1, \beta_2 \in V^*, \gamma \in V^+,$

  in cui si esprime il fatto che la produzione $A \rightarrow \gamma$ può essere applicata solo se A si trova nel contesto $\langle \beta 1, \beta 2 \rangle$.

Grammatiche di tipo 2 (o libere dal contesto)

- Queste grammatiche, dette anche non contestuali o context free, ammettono produzioni del tipo:

  $A \rightarrow \beta, \quad A \in V_N, \quad \beta \in V^+$

  cioè produzioni in cui ogni non terminale A può essere riscritto in una stringa β indipendentemente dal contesto in cui esso si trova

- I linguaggi generabili da grammatiche di tipo 2 vengono detti linguaggi di tipo 2 o non contestuali o context free

Grammatiche di tipo 3 (o regolari)

- Queste grammatiche, dette anche lineari destre o regolari, ammettono produzioni del tipo:

$A \to \delta, \quad A \in V_N, \quad \delta \in (V_T \circ V_N) \cup V_T$ (regolare destra)

$A \to \delta, \quad A \in V_N, \quad \delta \in (V_N \circ V_T) \cup V_T$ (regolare sinistra)

- I linguaggi generabili da grammatiche di tipo 3 vengono detti linguaggi di tipo 3 o regolari

Tipi di grammatiche

- Tipo 0
- $\alpha \rightarrow \beta\ , \quad \alpha \in V^* \circ V_N \circ V^*\ , \quad \beta \in V^*$ dove $V = (V_T \cup V_N)$

- Tipo 1
- $\alpha \rightarrow \beta, \quad \alpha \in V^* \circ V_N \circ V^*, \quad \beta \in V^+, \quad |\alpha| \leq |\beta|$

- Tipo 2
- $A \rightarrow \beta, \quad A \in V_N, \quad \beta \in V^+$

- Tipo 3
- $A \rightarrow \delta, \quad A \in V_N, \quad \delta \in (V_T \circ V_N) \cup V_T$ (regolare destra)

Domande Parte Teorica

- Con riferimento agli automi cosa s'intende per funzione di transizione?

La funzione di transizione

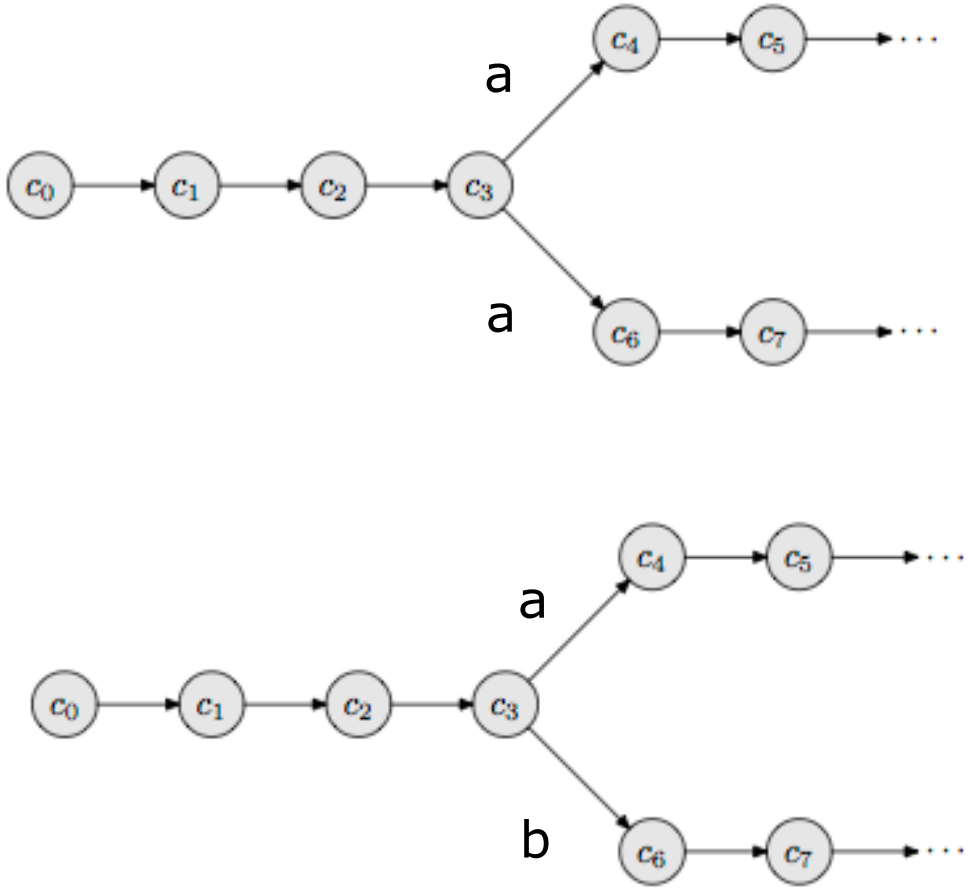
- La funzione di transizione, che è parte della definizione della struttura dell'automa, induce una relazione di transizione tra configurazioni, che associa ad una configurazione un'altra (o più di una) configurazione successiva
- L'applicazione della funzione di transizione ad una configurazione si dice **transizione** o **mossa** o **passo computazionale** dell'automa
- Per ogni automa A, date due configurazioni c_i, c_j di A, useremo nel seguito la notazione c_i |-- A-- c_j per indicare che c_i e c_j sono correlate dalla relazione di transizione, vale a dire che c_j deriva da ci per effetto dell'applicazione della funzione di transizione di A

Definizione di automa a stati finito deterministico

- Un automa a stati finiti deterministico (ASFD) è una quintupla $A = \langle \Sigma, Q, \delta, q_0, F \rangle$, dove
  - $\Sigma = \{a_1, ..., a_n\}$ è l'alfabeto di input,
  - $Q = \{q_0, ..., q_m\}$ è un insieme finito e non vuoto di stati,
  - $F \subseteq Q$ è un insieme di stati finali,
  - $q_0 \in Q$ è lo stato iniziale
  - $\delta : Q \times \Sigma \rightarrow Q$ è la funzione (totale) di transizione che ad ogni coppia ⟨stato, carattere in input⟩ associa uno stato successivo

Domande Parte Teorica

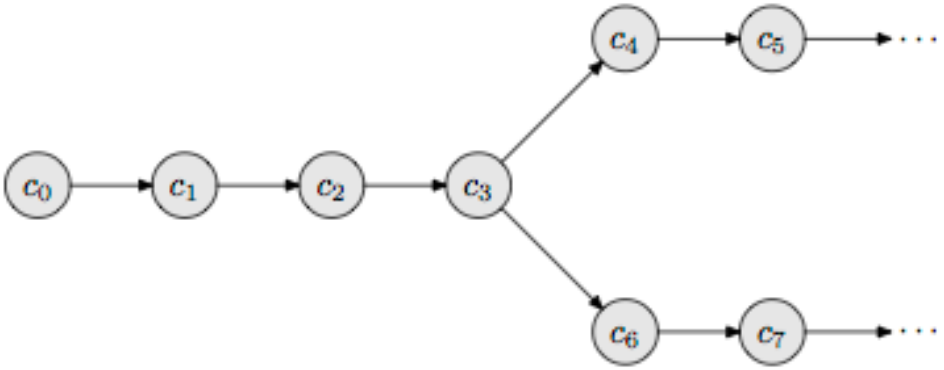
- Presentare la differenza tra automi deterministici e non deterministici. Fornire un esempio.


c_0
c_1
c_2
c_3
c_4
c_5
c_6
c_7
a
a
b

Automi deterministici e non deterministici

- Un automa è detto deterministico se ad ogni stringa di input associa una sola computazione, e quindi una singola sequenza di configurazioni

- Un automa è detto non deterministico se esso associa ad ogni stringa di input un numero qualunque, in generale maggiore di uno, di computazioni



Domande Parte Teorica

- Qualora possibile, come può essere trasformato un ASFND in un ASFD?

ASFD e ASFND

- Gli automi deterministici e quelli non deterministici hanno lo stesso potere espressivo!!!
- Questo significa che se un linguaggio può essere accettato da un ASFND allora esiste anche un ASFD che lo accetta, e viceversa
- Esiste un algoritmo noto come costruzione dei sottoinsiemi (subset construction) che serve per costruire, a partire da un ASFND dato, un ASFD equivalente che simula il non determinismo, ma è deterministico

Idea

- Partiamo da un insieme di stati DStates contenente solo uno stato iniziale non marcato
- Ad ogni passo selezioniamo uno stato non marcato da DStates e ne calcoliamo le transizioni uscenti aggiungendo a DStates, non marcati, eventuali nuovi stati trovati
- Prima o poi, dato che il numero di stati possibili è non ci saranno più stati non marcati da considerare e l'algoritmo terminerà

Algoritmo di traduzione

LINGUAGGIO: Pseudocodice.
INPUT: Un automa non deterministico N.
OUTPUT: Un automa deterministico equivalente D.

Sia $\{s_0\}$ l'unico stato non marcato di $DStates$;
while c'e' uno stato non marcato T in $DStates$ **do**
begin
marca T;
for each simbolo di input $x \in \Sigma$ **do**
begin
$U := \overline{move}(T, x)$;
if U non e' in $DStates$ **then**
begin
aggiungi U, non marcato, a $DStates$;
end
$Dtran(T, x) := U$;
end
end

$$\overline{move}: (\wp(S) \times \Sigma) \longrightarrow \wp(S)$$

$$\overline{move}(T, x) = \bigcup_{s \in T} move(s, x)$$

lo stato iniziale di D e' $\{s_0\}$
gli stati finali di D sono tutti quelli che contengono almeno uno stato finale di N

Domande Parte Teorica

- Con riferimento agli automi che cosa s'intende per configurazione?

La configurazione

- Una configurazione rappresenta l'insieme delle informazioni che determinano, in un certo istante, il comportamento futuro dell'automa

- Una configurazione è composta dalle seguenti informazioni

1. Stato interno dell'automa
2. Contenuto di tutti i nastri di memoria
3. Posizione di tutte le testine sui nastri

- Esiste una sola configurazione, detta **configurazione iniziale**, che rappresenta la situazione complessiva in cui si trova l'automa all'inizio, nel momento in cui la stringa di input gli viene sottoposta

Domande Parte Teorica

- Distinguere quando un problema è decidibile e quando è semi-decidibile.

Decidibile e Semi-decidibile

- Un problema decisionale (o un linguaggio) è detto **decidibile** se esiste un algoritmo (in particolare, un automa) che per ogni istanza del problema risponde correttamente vero oppure falso
  - In caso contrario il problema è detto **indecidibile**
- Un problema decisionale (o un linguaggio) è detto **semi-decidibile** se esiste un algoritmo (in particolare, un automa) che per tutte e sole le istanze positive del problema risponde correttamente vero.

I linguaggi di tipo 1, 2 e 3 sono decidibili, mentre quelli di tipo 0 sono semi-decidibili

Domande Parte Teorica

- Che tipo di grammatiche sono supportate dagli automi a stati finiti?

Anticipiamo qualcosa!

- Per i linguaggi di tipo 3 esistono dispositivi di riconoscimento che operano con memoria costante e tempo lineare
  - Automi a stati finiti
- Per i linguaggi strettamente di tipo 2, il riconoscimento può avvenire sempre in tempo lineare ma con dispositivi di tipo non deterministico6 dotati di una memoria a pila
  - Automi a pila non deterministici
- Per i linguaggi strettamente di tipo 1, l'esigenza di tempo e di memoria è ancora maggiore. Per essi si può mostrare che il riconoscimento può essere effettuato con una macchina non deterministica che fa uso di una quantità di memoria che cresce linearmente con la lunghezza della stringa da esaminare
  - Macchina di Turing non deterministica "linear bounded", o Automa lineare
- Per i linguaggi strettamente di tipo 0, si farà riferimento a un dispositivo di calcolo costituito da un automa che opera con quantità di tempo e di memoria potenzialmente illimitate
  - Macchina di Turing