PARTE 2 - TEORIA DELLA COMPUTABILITÀ

• LA MACCHINA DI TURING (TM):

E' un modello computazionale introdotto da Alan Turing nel 1936.
Consiste in una 7-pla del tipo $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$, dove:

- Q := insieme degli stati interni (finito);
- Σ := alfabeto di input, non contenente il blank ($\sqcup \notin \Sigma$);
- Γ := alfabeto del nastro ($\Sigma \subseteq \Gamma$ e $\sqcup \in \Gamma$);
- $\delta : Q \times \Gamma \longmapsto Q \times \Gamma \times \{L, R\}$
  $(q, \sigma) \longrightarrow (q', \sigma', m)$, con q' = nuovo stato
  σ' = nuovo simbolo scritto
  m = mossa della testina
- q_0 := start state;
- q_A := stato di accettazione;
- q_R := stato di rifiuto ($q_A \neq q_R$ SEMPRE!).

• CONFIGURAZIONE DI UNA TM:

La configurazione è data da: STATO + CONTENUTO DEL NASTRO + POSIZIONE DELLA TESTINA

Si rappresenta con una stringa che dice il contenuto del nastro e lo stato nella posizione precedente alla cella puntata dalla testina:

$C = [uqv]$, $u, v \in \Gamma^*$ e $q \in Q$ $\longrightarrow$ Sul nastro c'è "uv", siamo nello stato q e la testina punta a v.

• TRANSIZIONI (TRA CONFIGURAZIONI):

C_1 porta a C_2 se la TM può legalmente andare da C_1 a C_2 con un SINGOLO PASSO.

Sia $C_1 = [uaq_ibv]$, con $u, v \in \Gamma^*$, $a, b \in \Gamma$, $q_i \in Q$;

Allora abbiamo le seguenti mosse:

- **LEFT - MOVE**:

$C_2 = [u q_j a c v]$, $c \in \Gamma$, $q_j \in Q$

Ci arrivo tramite: $\delta(q_i, b) = (q_j, c, L)$

* CASO PARTICOLARE:

se sono all'inizio del nastro, non posso andare più a sinistra di così, quindi resto dove sono:

se $C_1 = [q_i b v] \Rightarrow C_2 = [q_j c v]$

- **RIGHT - MOVE**:

$C_2 = [u a c q_j v]$, $c \in \Gamma$, $q_j \in Q$

Ci arrivo tramite: $\delta(q_i, b) = (q_j, c, R)$

* CASO PARTICOLARE:

se a destra ho un blank, scrivo ciò che devo e proseguo:

se $C_1 = [u a q_i] \equiv [u a q_i \sqcup] \Rightarrow C_2 = [u a c q_j]$

- **START, ACCEPT, REJECT**:

Se $w = w_1 \dots w_n$ è la stringa di input $\Rightarrow C_0 = [q_0 w_1 \dots w_n]$

* ATTENZIONE: per accettare o rifiutare NON devo terminare l'input!

Mi basta entrare in una configurazione con q_A o q_R.

"M accetta l'input w" se $\exists$ sequenza di configurazioni $C_0 C_1 \dots C_k$ tale che:

(1) $C_0 = [q_0 w]$;

(2) C_i genera legalmente C_{i+1} , $\forall i = 1, \dots, k-1$;

(3) $q_A \in C_k$;

"M rifiuta l'input w": se (3) $q_R \in C_k$.

- **Linguaggio Ricorsivamente Enumerabile (r.e.)**:

Un linguaggio L è RICORSIVAMENTE ENUMERABILE (r.e.) se è riconoscibile da una TM, cioè se $\exists$ TM M tale che $\mathcal{L}(M) = L$.

* **NOTA**: Sia A r.e. e sia M tale che $\mathcal{L}(M) = A$. Se $w \notin A$, non si può dire che M RIFIUTA w! In realtà, potrebbe anche NON TERMINARE!

- **Linguaggio Decidibile**:

Un linguaggio $\mathcal{L}$ è DECIDIBILE (o "RICORSIVO") se $\exists$ TM M tale che, $\forall w \in \Sigma^*$: $M(w)$ accetta se $w \in L$ oppure $M(w)$ rifiuta se $w \notin L$.

- **Lemma**: $\boxed{\text{Se } A \text{ è DECIDIBILE} \Rightarrow A \text{ è r.e.}}$

* Il VICEVERSA non è vero!

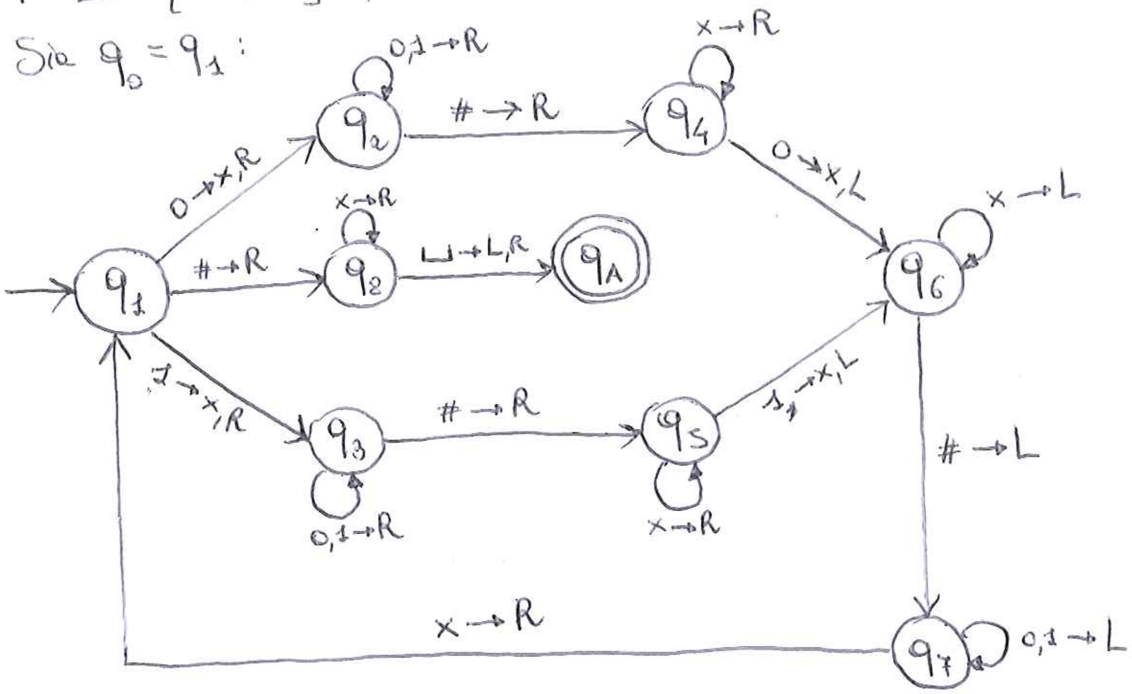
Esercizio: Dimostrare che $B = \{w\#w \mid w \in \{0,1\}^*\}$ è DECIDIBILE.

Avevamo già visto che B non è CFL.

La TM sarà: $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$:

$\Gamma = \Sigma \cup \{\sqcup, x\}$, con x che serve per MARCARE i caratteri già considerati.

Sia $q_0 = q_1$:



La Tabella delle Transizioni:

	0	1	#	x	⊔
q_1	$q_2 \times R$	$q_3 \times R$	$q_8 \# R$	$q_R \times R$	$q_R ⊔ R$
q_2	$q_2\, 0\, R$	$q_2\, 1\, R$	$q_4 \# R$	$q_R \times R$	$q_R ⊔ R$
...	...	...	...	...	...
q_8	$q_R\, 0\, R$	$q_R\, 1\, R$	$q_R \# R$	$q_8 \times R$	$q_A ⊔ R$

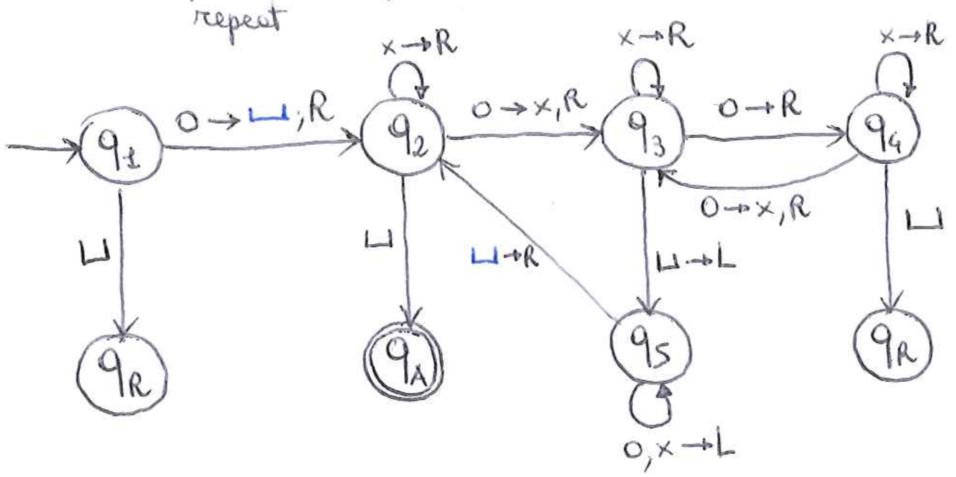
* **NOTA**: costruire una TM vuol dire trovare un ALGORITMO che risolve il problema.

• Esercizio:

Dimostrare che $C = \{ 0^{2^n} \mid n \geq 0 \}$ è DECIDIBILE.

Divido ogni volta a metà il # di 0, uno sì ed uno no. Se devo cancellare uno 0, ma c'è un ⊔, rifiuto, poiché vuol dire che sono dispari.

ALGORITMO:
- if there is no 0, reject ($\varepsilon \notin C$)
- if there is just one 0, accept
- if there is and odd number of 0's, reject
- cancel half of the 0's
- repeat



* **NOTA**: sostituisco il 1° zero con un blank ⊔ per poter riconoscere l'inizio del nastro, altrimenti in q_5 sarei andato in un loop infinito!

• Esercizio:

Dimostrare che $A = \{ a^i b^j c^k \mid i,j,k \geq 1 \wedge i \cdot j = k \}$ è DECIDIBILE.

Si marca una "a", poi si marcano le "b" e per ogni "b" si marca una "c". Poi si demarcano le "b" e si riparte dalle "a" successive.

M = "On input string w:
1. Scan w from left to right and reject if $w \notin \mathcal{L}(a^+b^+c^+)$;
2. return the head to the leftmost symbol of w;
3. cross an "a" and scan to the right until a "b" occurs;
4. shuttle between "b" and "c", crossing a "c" for every "b", until all "b" are crossed;
5. restore the crossed "b"'s;
6. repeat from 3. If all "a" are crossed: if all "c" have been crossed, then accept; otherwise, reject."

• Esercizio: PROBLEMA DELLA DISTINZIONE DEGLI ELEMENTI

$E = \{ \#x_1\#x_2 \dots \#x_l \mid l \geq 0 \wedge x_i \in \{0,1\}^* \wedge x_i \neq x_j, \text{ if } i \neq j \}$. Dim. che è decidibile.

Marco gli # 2 alla volta e confronto le stringhe: se sono uguali, RIFIUTO; altrimenti, se dopo aver marcato il 1° # non riesco a marcare esattamente il 2° #, ACCETTO.

M = "On input string w:
1. If the leftmost input symbol is ⊔, then accept;
2. If the leftmost input symbol is not #, then reject;
3. Place a mark $\overline{\#}$ on the # at the left;
4. Scan to the right and place a mark on first # occurred.
   4.1 If no such # is found, then ACCEPT;
5. Compare the strings having $\overline{\#}$ on the left: if they are equal, then REJECT;
6. Remove the mark on the ~~left~~ rightmost $\overline{\#}$, and place a mark on the next # to the right. If no # is found, remove the mark from the leftmost $\overline{\#}$ and place it on the next # to the right;
7. Repeat from 4."

• NTM : Nondeterministic TM

È una TM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ ma con una DIVERSA δ:

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

porta in un SOTTOINSIEME DI STATI.

- CONFIGURAZIONE ACCETTANTE:

Una NTM N accetta $w \in \Sigma^*$ se $\exists$ sequenza $C_0, C_1, \dots, C_m$ tale che:

(1) $C_0 = [q_0 w]$;

(2) $\forall K \geq 0$, $\exists$ transizione lecita in δ che porta C_K in C_{K+1};

(3) C_m è accettante, cioè $q_A \in C_m$.

• Decide un linguaggio:

Una NTM N decide un linguaggio $A \iff \mathcal{L}(N) = A$ ed OGNI possibile COMPUTATIONAL PATH TERMINA in un numero finito di passi.

• TEOREMA: $\boxed{\text{Ogni NTM } N \text{ ha una DTM } D \text{ equivalente!}}$

Idea: $w \in \Sigma^*$ è accettata da N se ALMENO 1 ramo di computazione giunge in q_A.

D potrebbe ESPLORARE DETERMINISTICAMENTE l'ALBERO computazionale:

•) depth-first search: in profondità;

•) breadth-first search: IN AMPIEZZA → D funziona solo in AMPIEZZA, poiché potrebbero esserci degli endless-loop e potrei non riuscire a risalire da un ramo di computazione.

• D è una TM con 3 NASTRI:

(1) nastro di input (read-only);

(2) nastro di lavoro;

(3) breadth-first search state.

Dato un nodo, definiamo con $b :=$ max degree (node), cioè il massimo numero di figli di quel nodo. Di certo: $b \leq |Q| \times |\Gamma| \times |3|$

$\Gamma_b = \{1, 2, \dots, b\}$ → ogni simbolo rappresenta un percorso di un ramo da poter seguire dal nodo in esame;

Per esempio: "2314" vuol dire → Dalla radice segui il 2° arco, poi il 3°, poi il 1° ed infine il 4°.

Dunque, l'algoritmo di D è il seguente:

D = "On input w on tape 1, and blank on tape 2 and 3:

1. Copy input from tape 1 to tape 2;
2. use tape 2 to simulate the steps described on tape 3; ε on tape 3 means to stay on the root;
   - 2.1 if q_A is even entered, then ACCEPT;
   - 2.2 if q_R is even entered, then GOTO step 3.
   - 2.3 if all steps on tape 3 have been simulated, then GOTO step 3.
3. read the string on tape 3 and replace it with the next string in the STRING ORDER in Γ_b;
4. GOTO step 1."

• **COROLLARIO**: $\mathcal{L}$ è r.e. $\Longleftrightarrow$ $\exists$ NTM N, tale che $\mathcal{L}(N) = \mathcal{L}$

• **TEOREMA**: $\mathcal{L}$ è decidibile $\Longleftrightarrow$ $\exists$ NTM N che "decide" $\mathcal{L}$, cioè ogni computational path termina.

È possibile perché VISITO IN AMPIEZZA e sono sempre a conoscenza se ci sono altri nodi ad un livello più basso!

• **ENUMERATOR**

È una TM con un NASTRO SPECIALE P ed uno STATO SPECIALE q_P; ogni volta che entra nello stato q_P, è come se "stampasse" la stringa su P.

* L'ENUMERATOR genera stringhe, non decide!

Se non si ferma mai, è potenzialmente in grado di generare infinite stringhe.

• **TEOREMA**: $\boxed{\text{Un linguaggio } A \text{ è r.e.} \iff \exists \text{ enumerator } E \text{ che enumera } A}$

$\Leftarrow$) Sia $A \subseteq \Sigma^*$ e $w \in \Sigma^*$, la TM M che riconosce A è fatta così:

M = "On input w:

1. Simulate E. Every time E enters in q_p, then compare input w with the string on tape P of E;
2. If they're equal, then ACCEPT; otherwise GOTO 1."

Se $w \notin A$, non è detto che M termini $\Rightarrow$ A è r.e. !

$\Rightarrow$) L'enumeratore sarà fatto così:

E = "Ignore the input;

1. for $i = 1$ to $+\infty$:
   2. Simulate M for i steps on the first i strings of Σ^*, supposing that an order is established on Σ^*;
   3. If any computation accepts, then copy the corresponding string on tape P and enter q_p."

• Se $w \in A \Rightarrow w \in \Sigma^* \Rightarrow \exists k$: $w = s_k$ in $\Sigma^* = \{s_1, s_2, \dots, s_k\}$ ordinato.
$w \in A \Rightarrow M(w)$ termina in un numero finito di passi, diciamolo h.
Sia $\ell := \max(k, h)$. Quando $i = \ell$, si eseguono le computazioni $M(s_1), M(s_2), \dots, M(s_\ell)$; essendo ℓ definito in quel modo, c'è di sicuro $M(s_k) = M(w)$, che è accettante $\Rightarrow$ E la mette sul nostro P e la stampa.
Quindi, se $w \in A \Rightarrow E$ genera w.

• Se E genera w, vuol dire che $M(w)$ ha accettato $\Rightarrow$ $w \in A$, poiché $L(M) = A$.
ok!!!

* **NOTA**: l'enumeratore E genera ogni stringa di Σ^*, e quindi di A, più volte; anzi, INFINITE VOLTE !

La Tesi di Church - Turing

Qualunque ALGORITMO, comunque sia fatto, può essere equivalentemente implementato su una TURING MACHINE o con il λ-CALCULUS.

Per capire quali PROBLEMI possono essere risolti, facciamo coincidere il problema con un LINGUAGGIO:

PROBLEM: il numero n è potenza di 2?

$\hookrightarrow$ LANGUAGE: $A = \{ 0^{2^n} \mid n \geq 0 \}$

Abbiamo già visto che A è DECIDIBILE $\Rightarrow \exists$ TM M che decide A.

Applico i seguenti passi:

(1) costruisco la rappresentazione unaria di n: $n = \langle n \rangle_1 = 0^n$

(2) eseguo $M(0^n)$: $\begin{cases} M \text{ accetta} \Rightarrow n \text{ è potenza di 2} & \text{: istanza-sì} \\ M \text{ rifiuta} \Rightarrow n \text{ non è potenza di 2} & \text{: istanza-no} \end{cases}$

Il LINGUAGGIO è decidibile $\Longleftrightarrow$ il PROBLEMA è risolvibile

* L'implicazione "$\Leftarrow$" è data dalla TESI DI CHURCH-TURING!

• **Esempio: POLINOMIO UNIVARIATO**

PROBLEMA: un dato polinomio univariato ha una RADICE INTERA?

$\downarrow$

LINGUAGGIO: $D_1 = \{ \langle p \rangle \mid p \text{ è un polinomio univariato con 1 radice intera} \}$

• $\boxed{D_1 \text{ è r.e.}}$:

M_1 = "On input $\langle p \rangle$:

1. for $x = 0, +1, -1, +2, -2, +3, -3, \ldots$
2. evaluate $p(x)$
3. if $p(x) = 0$, then accept."

* M_1 NON TERMINA MAI sulle istanze-no!

(Cioè se p non ha una radice intera)

- ## 10° PROBLEMA DI HILBERT: polinomio multivariato

Un polinomio MULTIVARIATO ha 1 radice intera?

$D_n = \{ \langle p \rangle \mid p$ è un polinomio in n variabili con 1 radice intera $\}$

* $\boxed{D_n \text{ è r.e.}}$ → Stesse TM (come M_1), ma con cicli for annidati per ogni variabile.

- **TEOREMA**: Ogni radice intera r di un polinomio UNIVARIATO p soddisfa la relazione: $|r| \le k \cdot \frac{C_{max}}{C_1}$, dove k = numero di termini di p, C_{max} = valore assoluto del coefficiente massimo tra quelli di p, C_1 è il valore assoluto del coefficiente del termine di grado più alto.

Dunque, ho un limite superiore per le radici intere di polinomi UNIVARIATI; so quando fermarmi, senza andare all'infinito. Quindi:

- **TEOREMA**: $\boxed{D_1 \text{ è DECIDIBILE!}}$

Nel 1970 è stato dimostrato che D_n non è decidibile, quindi il 10° Problema di Hilbert NON È RISOLVIBILE!

- **Teorema: GRAFO CONNESSO**:

Un grafo NON DIRETTO G è CONNESSO? $A = \{ \langle G \rangle \mid G$ è un grafo connesso, non diretto $\}$

$\boxed{A \text{ è decidibile.}}$

M = "On input $\langle G \rangle$:
1. Select the first node of G and mark it;
2. Repeat until no new nodes are marked:
   3. for each node of G, mark it if it is adjacent to an already marked node;
4. Scan all nodes: if all nodes are marked, then accept; oth. reject."

- **PROBLEMA DI ACCETTAZIONE PER I DFA:** $\boxed{A_{DFA} \text{ è DECIDIBILE}}$

$A_{DFA} = \{ <B,w> \mid B \text{ is a DFA that accepts string } w \}$.

M = "On input <B,w>:
1. Simulate B on input w;
2. If B(w) accepts, then accept; otherwise, reject."

<B> potrebbe essere: list for Q, list for Σ, list for δ, q_0, list for F.

- **Teorema:** $\boxed{A_{NFA} = \{ <B,w> \mid B \text{ is an NFA accepting } w \} \text{ è DECIDIBILE}}$

N = "On input <B,w>:
1. Convert B in an equivalent DFA C;
2. Run the M (for A_{DFA}) on input <C,w>;
3. Accept or reject accordingly to M."

- **Teorema:** $\boxed{A_{REX} = \{ <R,w> \mid R \text{ is a REX that generates } w \} \text{ è DECIDIBILE.}}$

P = "On input <R,w>:
1. Convert R into an equivalent NFA A;
2. Run N on input <A,w>;
3. Accept or reject, accordingly to N(<A,w>)."

- **EMPTINESS PROBLEM for DFA:**

L'insieme dei DFA che non accettano alcune stringhe:

$$E_{DFA} = \{ <A> \mid A \text{ is a DFA such that } L(A) = \emptyset \}$$

$\boxed{E_{DFA} \text{ è decidibile}}$

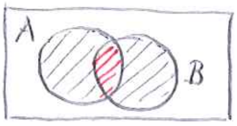
T = "On input <A>:
1. mark state q_0 of the DFA A;
2. repeat the following, until no new state is marked:
   3. mark any state that has a transition coming into it from a state already marked;
4. If no accepting state is marked, then accept; otherwise, reject."

- **EQUALITY PROBLEM for DFA:**

Dati 2 DFA, riconoscono lo stesso linguaggio?

$$EQ_{DFA} = \{ <A,B> \mid A, B \text{ are DFA's such that } \mathcal{L}(A) = \mathcal{L}(B) \}$$

* $\boxed{EQ_{DFA} \text{ è DECIDIBILE.}}$



Costruiamo la differenza simmetrica $\mathcal{L}(A) \Delta \mathcal{L}(B)$

$$\mathcal{L}(C) = \mathcal{L}(A) \Delta \mathcal{L}(B) = \left(\mathcal{L}(A) \cap \mathcal{L}(B)^c\right) \cup \left(\mathcal{L}(A)^c \cap \mathcal{L}(B)\right)$$

$$\mathcal{L}(A) = \mathcal{L}(B) \iff \mathcal{L}(C) = \emptyset$$

$\mathcal{L}(A)$ ed $\mathcal{L}(B)$ sono linguaggi REGOLARI $\Rightarrow$ chiusi rispetto a $\cap$, ^{c}, $\cup$.

$\Rightarrow \mathcal{L}(C)$ è regolare $\Rightarrow \exists$ DFA C tale che: $\mathcal{L}(C) = \mathcal{L}(A) \Delta \mathcal{L}(B)$.

Eseguiamo il TEST DEL VUOTO su C:

F = "On input $<A,B>$:
1. Construct a DFA C that recognizes $\mathcal{L}(A) \Delta \mathcal{L}(B)$;
2. Run TM T (for E_{DFA}) on input $<C>$;
3. Accept or Reject, accordingly to $T(<C>)$".

- **TEOREMA:** $\boxed{A_{CFG} = \{ <\mathcal{G}, w> \mid \mathcal{G} \text{ is a CFG that generates } w \} \text{ è DECIDIBILE}}$

Consideriamo $\mathcal{G}$ in Chomsky Normal Form: $\begin{cases} A \to BC \\ A \to a \\ S \to \varepsilon \end{cases}$

Lemma: In una CFG $\mathcal{G}$ in CNF, una derivazione per w con $|w| = n > 0$, ha lunghezza esattamente pari a $2n - 1$.
Dimostrabile per induzione.

S = "On input $<\mathcal{G}, w>$:
1. Convert $\mathcal{G}$ in an equivalent CFG $\mathcal{G}'$ in CNF;
2. if $w = \varepsilon$, then accept if $S \to \varepsilon$ is in $\mathcal{G}'$; oth. reject;
3. here $|w| > 0$: list all derivations with $2|w| - 1$ steps;
4. If any derivation generates w, then accept; oth. reject."

• **TEOREMA**: $\boxed{E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG such that } L(G) = \emptyset \} \text{ è DECIDIBILE.}}$

Dato $A \in V$, cerchiamo di capire se $A \Rightarrow^* w$, $w \in \Sigma^*$. Una volta risolto ciò, ci basterà capire se $S \Rightarrow^* w$.

R = "On input $\langle G \rangle$:
1. mark all terminal symbols of G;
2. repeat, until non new variable get marked:
   3. mark any variable A, where G has a rule $A \to u_1 \dots u_k$, where all u_i are already marked;
4. If S is marked, then reject; oth. accept."

Se S è marcato RIFIUTO, poiché vuol dire che $\exists\, w \in \Sigma^*$ t.che $S \Rightarrow^* w$ e quindi $L(G) \neq \emptyset$.

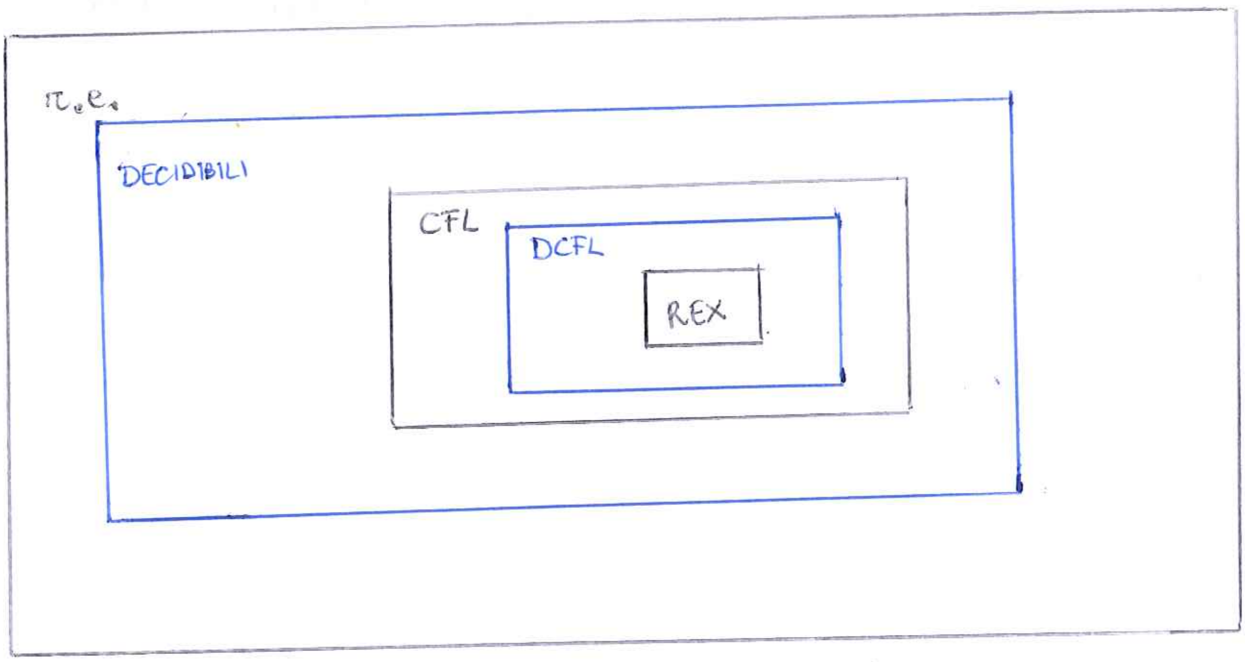
• **TEOREMA**: $\boxed{A_{PDA} = \{ \langle P, w \rangle \mid P \text{ is a PDA accepting } w \} \text{ è DECIDIBILE}}$

(1) $\forall$ CFG G, $\exists$ PDA P tale che $L(G) = L(P)$
(2) A_{CFG} è decidibile

$\Rightarrow A_{PDA}$ è decidibile!

ok!!!

LINGUAGGI



* I linguaggi DECIDIBILI sono ovviamente un sottoinsieme dei linguaggi ricorsivamente enumerabili.

• **TEOREMA:** $\boxed{\text{Ogni linguaggio CFL è DECIDIBILE.}}$

* WRONG idea: CFL → PDA → NFA ⇒ CFL ⊆ r.e, ma NON decidibile! (lo dobbiamo dimostrare)

Proof: Sia A un CFL ⇒ ∃ CFG G tale che: $L(G) = A$

Ma, esiste una TM S che DECIDE A_{CFG}! ⇒ CFL ⊆ decidable language

Costruiamo un decisore M_G:

M_G = "On input w:
1. Put <G> before <w> on the tape;
2. Run S (for A_{CFG}) on input <G,w>;
3. Accept or reject, accordingly to S(<G,w>)".

M_G decide A ⇒ I CFL sono decidibili! ok!!!

RIEPILOGO

	DFA	NFA	REX	CFG	PDA
ACCEPTANCE	A_{DFA}	A_{NFA}	A_{REX}	A_{CFG}	A_{PDA}
EMPTINESS	E_{DFA}	E_{NFA}	E_{REX}	E_{CFG}	E_{PDA}
EQUALITY	EQ_{DFA}	EQ_{NFA}	EQ_{REX}	EQ_{CFG}	EQ_{PDA}

* EQ_{CFG}, e di conseguenza anche EQ_{PDA}, è r.e., ma NON DECIDIBILE.

Infatti, per EQ_{DFA} si usava la differenza simmetrica:

$$L(A) \triangle L(B) = (L(A) \cap L(B)^c) \cup (L(A)^c \cap L(B))$$

e funzionava poiché i linguaggi REGOLARI sono chiusi rispetto a $\cup, \cap, {}^c$!

MA, i CFL non sono chiusi rispetto a $\cap$ e ^{c}!

- **UTM : Universal Turing Machine**

Una UTM prende in input una TM e un input w, e simula $M(w)$. È importante poiché è la 1ª macchina che lavora con un PROGRAMMA immagazzinato in memoria insieme ai dati ("STORED - PROGRAM COMPUTER").

U = "On input $\langle M, w \rangle$, where M is a TM and w is a string:
1. Simulate M on input w;
2. If $M(w)$ accepts, then accept;
3. If $M(w)$ rejects, then reject."

- **PROBLEMA dell' ACCETTAZIONE delle TM's**

$$\boxed{A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM accepting } w \}} \text{ è r.e.}$$

Una UTM U su input $\langle M, w \rangle \longrightarrow U(\langle M, w \rangle) = \begin{cases} \text{accept if } M(w) \text{ accepts} \\ \text{reject if } M(w) \text{ rejects} \\ \text{endless-loop if } M(w) \text{ does not halt.} \end{cases}$

$\langle M, w \rangle \in A_{TM} \iff M(w) \text{ accepts} \iff U(\langle M, w \rangle) \text{ accepts.}$

$\Rightarrow$ A_{TM} è r.e. !

* **NOTA**: Tuttavia, $\langle M, w \rangle \notin A_{TM} \not\Rightarrow U(\langle M, w \rangle)$ rejects, poiché $M(w)$ potrebbe non terminare.

Sembra che A_{TM} sia non DECIDIBILE (ed è così!).

- **METODO DI DIAGONALIZZAZIONE DI CANTOR :**

Due insiemi, finiti o infiniti, hanno la stessa cardinalità se $\exists$ BIEZIONE tra di loro.

Insiemi NUMERABILI (COUNTABLE) : $\mathbb{N}, 2\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \ldots$

$\boxed{\mathbb{R} \text{ non è numerabile}}$

Per assurdo, esista una biezione tra l'intervallo $[0,1] \subseteq \mathbb{R}$ ed i naturali $\mathbb{N}$.

$[0,1] \xleftrightarrow{bij} \mathbb{N}$. Supponiamo di aver creato una corrispondenza:

$\mathbb{N}$ $\qquad$ $[0,1]$

$r_1 \longleftrightarrow 0,\underline{0}135422\ldots$

$r_2 \longleftrightarrow 0,0\underline{3}54217\ldots$

$r_3 \longleftrightarrow 0,02\underline{1}3457\ldots$

$\vdots \qquad \vdots$

Costruiamo un nuovo numero in $(0,1)$ che ha come K-esima cifra dopo la virgola una cifra diversa dalla K-esima di r_k:

$\boxed{0,142\ldots}$ $\Rightarrow$ È diverso da tutti gli altri, ma $\in [0,1)$ e non ha una corrispondenza

$\rightarrow$ ASSURDO $\Rightarrow [0,1)$ non è numerabile $\Rightarrow \mathbb{R}$ NON è NUMERABILE!

• **TEOREMA 1:** L'insieme di tutte le TM's è NUMERABILE.

Alfabeto Σ; Σ^* ORDINATO, ma un ordinamento è una biezione con $\mathbb{N}$.

$\Rightarrow \exists\ \Sigma^* \xleftrightarrow{bij} \mathbb{N}$. Con un alfabeto Σ posso codificare tutte le TM's.

$\Rightarrow \{\langle M\rangle \mid M \text{ is a TM}\} \subseteq \Sigma^* \xleftrightarrow{bij} \mathbb{N}$ $\Rightarrow$ L'insieme di tutte le TM's è numerabile!

• **TEOREMA 2:** $BIN = \{ \underbrace{w_1 w_2 \ldots}_{\infty} \mid w_i \in \{0,1\} \}$ è NON NUMERABILE.

Assumo per assurdo che lo sia. Utilizzo il METODO DI DIAGONALIZZAZIONE di CANTOR per costruire un numero binario diverso dagli altri $\rightarrow$ ASSURDO $\rightarrow$ BIN non è numerabile!

• **TEOREMA 3:** L'insieme di tutti i linguaggi su un alfabeto Σ
$Lang = \{ L \mid L \text{ is a language over } \Sigma \}$ è NON numerabile

Ordino in STRING ORDER $\Sigma^* = \{\sigma_1, \sigma_2, \sigma_3, \ldots\}$

Esiste la BIEZIONE $BIN \xleftrightarrow{bij} Lang$ poiché considero la sequenza caratteristica del linguaggio: $\underset{BIN}{\underset{\cap}{w_1 w_2 \ldots}}$, dove $w_i = \begin{cases} 0 & \text{se } \sigma_i \notin L \\ 1 & \text{se } \sigma_i \in L \end{cases}$

Poiché BIN è non numerabile $\Rightarrow$ Lang NON è NUMERABILE!

TEOREMA: $\boxed{\text{Esistono infiniti linguaggi su un dato alfabeto } \Sigma \text{ che NON sono r.e.}}$

Ogni TM può riconoscere 1 UNICO linguaggio su Σ.

$\{\langle M\rangle \mid M \text{ is a TM}\}$ è NUMERABILE, ma $Lang = \{L \mid L \text{ is a language over } \Sigma\}$ NO.

$\Rightarrow$ Le TM's riconoscono solo un'infinità numerabile di linguaggi su Σ.

$\rightarrow$ Ci sono infiniti linguaggi su Σ che NON sono r.e. !

• **TEOREMA:** $\boxed{A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM accepting } w\} \text{ è NON DECIDIBILE.}}$

Proof: Per assurdo, sia A_{TM} decidibile $\Rightarrow \exists$ DECISORE H tale che:

$$H(\langle M,w\rangle) = \begin{cases} \text{accept if } M(w) \text{ accepts} \\ \text{reject if } M(w) \text{ does not accept} \end{cases}$$

Costruiamo una TM D' che inverte le condizioni di accettazione di H:

$$D'(\langle M,w\rangle) = \begin{cases} \text{accept if } M(w) \text{ does not accept} \\ \text{reject if } M(w) \text{ accepts} \end{cases}$$

Costruiamo una TM D, che su input $\langle M\rangle$ costruisce la stringa "$\langle M, \langle M\rangle\rangle$" e simula $D'(\langle M, \langle M\rangle\rangle)$:

$$D'(\langle M, \langle M\rangle\rangle) = \begin{cases} \text{accept if } M(\langle M\rangle) \text{ does not accept} \\ \text{reject if } M(\langle M\rangle) \text{ accepts} \end{cases} = D(\langle M\rangle)$$

- se $D(\langle D\rangle)$ accetta, per costruzione $\Rightarrow D(\langle D\rangle)$ non accetta
- se $D(\langle D\rangle)$ rifiuta, per costruzione $\Rightarrow D(\langle D\rangle)$ accetta

$\}$ ↯ ASSURDO

- se $D(\langle D\rangle)$ non termina mai $\Rightarrow D'(\langle D, \langle D\rangle\rangle)$ non termina mai $\Rightarrow$ $\Rightarrow H(\langle D\rangle)$ non termina mai $\rightarrow$ MA H è un DECISORE ! $\rightarrow$ ↯ ASSURDO.

$\Rightarrow$ A_{TM} NON è DECIDIBILE !!!

■ ok!!!

• **TEOREMA:** $\boxed{\text{Un linguaggio } A \text{ è DECIDIBILE} \iff A^c \text{ è r.e. ed } A \text{ è r.e.}}$

$\Rightarrow$) Se A è decidibile, basta complementare la TM che decide A per decidere $A^c \Rightarrow$ sia A che A^c sono r.e., essendo decidibili.

$\Leftarrow$) Se A e A^c sono r.e. $\Rightarrow \exists$ TM's M_1 ed M_2 : $\mathcal{L}(M_1) = A$ e $\mathcal{L}(M_2) = A^c$

Decisore N:

N = "On input w:
1. Run $M_1(w)$ and $M_2(w)$ in PARALLEL;
2. If $M_1(w)$ accepts, then accept;
3. If $M_2(w)$ accepts, then reject."

$w \in A$ oppure $w \in A^c$ SEMPRE! $N(w)$ accetta $\iff w \in A$
$N(w)$ rifiuta $\iff w \in A^c$

$\to$ N DECIDE sempre! $\Rightarrow$ A è DECIDIBILE! ■ ok!!!

* **NOTA:** Poiché $\mathcal{A}_{TM}$ è non decidibile $\Rightarrow$ $\mathcal{A}_{TM}^c$ non è r.e.

$\mathcal{A}_{TM}^c = \{ \langle M, w \rangle \mid M \text{ is a TM that does not accept } w \}$

$\hookrightarrow$ può anche non terminare, perciò $\mathcal{A}_{TM}^c$ non è r.e.

• **PROBLEMA della FERMATA delle TM's:**

$\boxed{Halt_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \} \text{ è NON DECIDIBILE}}$

IDEA: $\mathcal{A}_{TM}$ "$\leq$" $Halt_{TM}$.

Per assurdo, sia $Halt_{TM}$ decidibile $\Rightarrow \exists$ decisore R tale che:

$$R(\langle M, w \rangle) = \begin{cases} \text{accept if } M(w) \text{ halts} \\ \text{reject if } M(w) \text{ does not halt} \end{cases}$$

Costruiamo un decisore N per A_{TM}, basato su R:

N = "On input <M,w>:
1. Run R on input <M,w>;
2. If R(<M,w>) rejects, then reject;
3. If R(<M,w>) accepts, then run M on input w;
4. Accept or reject, accordingly to M(w)."

$N(<M,w>)$ accetta $\Leftrightarrow$ $M(w)$ accetta
$N(<M,w>)$ rifiuta $\Leftrightarrow$ $M(w)$ non accetta
$\Big\} \rightarrow$ N decide A_{TM} $\rightarrow$ $\lightning$ ASSURDO

ok!!

$\Rightarrow$ $Halt_{TM}$ è <u>non</u> decidibile!

• **TEOREMA:** $\boxed{\mathcal{E}_{TM} = \{<M> \mid M \text{ is a TM such that } \mathcal{L}(M) = \emptyset\} \text{ è NON DECIDIBILE}}$

A_{TM} "$\leq$" $\mathcal{E}_{TM}$. Costruiamo una <u>RIDUZIONE</u>.

Sia per assurdo $\mathcal{E}_{TM}$ decidibile $\Rightarrow$ $\exists$ decisore R per $\mathcal{E}_{TM}$.

Data l'istanza <M,w> per A_{TM}, costruiamo M':

M' = "On input z:
1. If $z \neq w$, then reject;
2. If $z = w$, then run M on input w;
3. Accept or reject, accordingly to M(w)."

Abbiamo 2 possibilità:
- $\mathcal{L}(M') = \{w\}$ $\Leftrightarrow$ $M'(z)$ accepts $\Leftrightarrow$ $M(w)$ accepts $\Leftrightarrow$ $R(<M'>)$ rejects
- $\mathcal{L}(M') = \emptyset$ $\Leftrightarrow$ $M'(z)$ does not accept $\Leftrightarrow$ $M(w)$ does not accept $\Leftrightarrow$ $R(<M'>)$ accepts.

Decisore S per A_{TM}:

S = "On input <M,w>:
1. Build <M'> from M and w;
2. Run R on input <M'>;
3. If R(<M'>) accepts, then reject;
4. If R(<M'>) rejects, then accept."

$\rightarrow$ S decide A_{TM}
$\downarrow$
$\lightning$ ASSURDO
$\Downarrow$
$\mathcal{E}_{TM}$ è <u>NON</u> DECIDIBILE!

ok!!

• **TEOREMA:** $REGULAR_{TM} = \{ \langle M \rangle \mid M \text{ is a TM such that } L(M) \text{ is regular} \}$ è <u>NON</u> decidibile

Lo sia per assurdo $\Rightarrow \exists$ decisore R per $REGULAR_{TM}$. Consideriamo il decisore S:

S = "On input $\langle M, w \rangle$:

1. Build the encoding of the following TM M_2:

   M_2 = "On input x:
   1. if x has the form $0^n 1^n$, then accept;
   2. otherwise, run M on input w and accept if $M(w)$ accepts."

2. Run R on input $\langle M_2 \rangle$;
3. If $R(\langle M_2 \rangle)$ accepts, then accept; otherwise reject."

* **NOTA**: M_2 <u>non</u> viene eseguita! Serve solo la codifica.

$$L(M_2) = \begin{cases} \{0^n 1^n \mid n \geq 0\} & \text{se } M(w) \text{ non accetta} \rightarrow \text{NON REGOLARE} \\ \Sigma^* & \text{se } M(w) \text{ accetta} \rightarrow \text{REGOLARE} \end{cases}$$

• Se $M(w)$ non accetta $\Rightarrow \langle M, w \rangle \notin A_{TM}$
• Se $M(w)$ accetta $\Rightarrow \langle M, w \rangle \in A_{TM}$

$\}$ S decide A_{TM} $\rightarrow$ ASSURDO!

$\Rightarrow$ $REGULAR_{TM}$ è <u>non</u> decidibile!

• **TEOREMA:** $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TM's s.t. } L(M_1) \equiv L(M_2) \}$ è <u>non</u> decidibile.

RIDUZIONE E_{TM} "$\leq$" EQ_{TM}.

Se uno dei 2 linguaggi $L(M_1)$ o $L(M_2)$ fosse <u>VUOTO</u>, fare il test di uguaglianza equivale a fare il test del vuoto.

Per Assurdo, sia EQ_{TM} decidibile $\Rightarrow \exists$ TM R t.che R decide EQ_{TM}.

Costruiamo un decisore S per E_{TM}:

$S =$ "On input $<M>$:

1. Build an encoding $<M_3>$ of a TM such that $L(M_3) = \emptyset$;
2. Run R on input $<M, M_3>$;
3. If $R(<M, M_3>)$ accepts, then accept; otherwise reject."

S accepts $\Leftrightarrow R(<M, M_3>)$ accepts $\Leftrightarrow L(M) = \emptyset$

S rejects $\Leftrightarrow R(<M, M_3>)$ rejects $\Leftrightarrow L(M) \neq \emptyset$

$\Big\}$ S decide E_{TM}

$\Downarrow$

$\nexists$ ASSURDO

$\Rightarrow EQ_{TM}$ è NON decidibile!

$\blacksquare$ di!!!