1 Automi — **10 Points**

- Sia $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ un DFA, e si supponga che per ogni $a \in \Sigma$ si abbia $\delta(q_0, a) = \delta(q_f, a)$.
  - Si mostri che per ogni $w \neq \varepsilon$ si ha $\delta^*(q_0, w) = \delta^*(q_f, w)$, dove δ^* è la funzione di transizione estesa.
  - Si mostri che se x è una stringa non vuota in $L(A)$, allora x^k appartiene ad $L(A)$ per ogni $k > 0$.
- Definire la nozione di Push-Down Automaton (PDA). Dimostrare che un linguaggio è acontestuale se e solo se esiste un PDA che lo riconosce.

$\delta^*(q_0, \omega) = q_K$ se $\exists (q_1, q_2 \dots q_K)$ t.c. $\delta(q_0, w_1) = q_1 \wedge \dots \delta(q_{K-1}, w_K) = q_K$

$\omega = w_1 w_2 \cdots w_K$

ma $\delta(q_0, w_1) = q_1 \Rightarrow \delta(q_F, w_1) = q_1 \Rightarrow \delta^*(q_F, \omega) = q_K$.

Se $x \in L(A) \Rightarrow \delta^*(q_0, x) = q_F$ ma $\delta^*(q_F, x) = q_F$

↳ $(q_0, x) \vdash^* (q_F, \varepsilon)$ ↳ $(q_F, x) \vdash^* (q_F, \varepsilon)$

$(q_0, xx) \vdash^* (q_F, x)$

↳ $(q_0, x^K) \vdash^* (q_F, x^{K-1}) \vdash^* (q_F, \varepsilon) \Rightarrow x^K \in A$

Bisogna ora dimostrare l'equivalenza fra PDA e CFG.

[⇒]: Sia $G = (V, \Sigma, R, S)$ una grammatica, definisco $P_G = (Q, \Sigma, (\Sigma \cup \Gamma), \delta, q_0, \{q_F\})$ t.c.

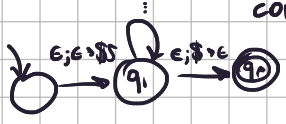
1) $\delta(q_0, \epsilon, \epsilon) = (q_1, \$S)$

2) $\delta(q_1, \epsilon, V_i) = (q_1, w)$ se $V_i \to w \in R$

3) $\delta(q_1, a, a) = (q_1, \epsilon)$

4) $\delta(q_1, \epsilon, \$) = (q_F, \epsilon)$

in tal modo il PDA deriva ogni possibile stringa di G e la scrive nello stack, se nell'input c'e' $\omega \in L(G)$, P_G "ripulisce" lo stack con la regola (3) ed accetta.



[⇐]: Sia $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_F\})$ un PDA, definisco G_P una grammatica con le seguenti regole:

- Se $\delta(p, a, x) = (s, xt) \wedge \delta(r, b, xu) = (q, x) \Rightarrow A_{pq} \to a A_{sr} b$
- $\forall p \in Q : A_{pp} \to \epsilon$
- $A_{pq} \to A_{pr} A_{rq} \quad \forall r, p, q \in Q$

Oss: $A_{pq} \to x \Leftrightarrow x$ porta P da p a q Senza Alterare Lo Stack. (SALS)

Se $A_{pq} \to x$ e $x = a A_{rs} b$ per costruzione $A_{rs} = t$ porta da r a s SALS $\Rightarrow$

$x = a t b$ porta da p a q SALS.

Se $A_{pq} \to A_{pr} A_{rq}$ allora $\begin{cases} A_{pr} \to u \\ A_{rq} \to v \end{cases}$ porta da p a r // r a q $\Rightarrow x = uv$ porta da p a q SALS

Se x porta da p a q SALS allora (2 opzioni):

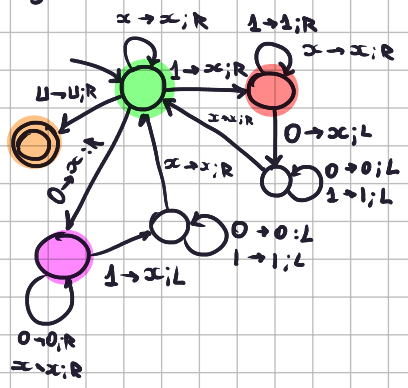
- nel primo passo si aggiunge u allo stack, all'ultimo si toglie, $A_{pq} = a A_{rs} b$ ma per ipotesi induttiva $A_{rs} \to t$ che porta da r a s SALS $\Rightarrow A_{pq} \to a t b = x$ che porta da p a q SALS.
- $A_{pq} \to x t$ e $A_{pr} \to x$, $A_{rs} \to t \Rightarrow$ in r lo stack viene svuotato $\Rightarrow A_{pq} \to x t$ e $x t$ porta da p a q SALS. ∎

2 Calcolabilità — **10 Points**

- Descrivere formalmente una macchina di Turing che riconosca il linguaggio delle stringhe binarie contenenti lo stesso numero di 0 e di 1. Dimostrare la correttezza della soluzione proposta.
- Sia L un linguaggio arbitrario. Dimostrare che L è decidibile se e solo se L è Turing-riconoscibile e coTuring-riconoscibile.

$L = \{x \in \{0,1\}^* \text{ t.c. } \omega_0(x) = \omega_1(x)\}$ La TM in questione si comporta' cosi':

per ogni 1 letto cerca uno 0 e cancella entrambi. nastro vuoto $\Leftrightarrow \omega \in L$



: leggo prossimo numero

: cerco 0 dopo aver letto 1

: cerco 1 dopo " " 0

: letto tanti zeri quanti 1

[⇒]: Se L e' decidibile, e $L(M) = L$, definisco M' t.c.

- Su input ω esegue $M(\omega)$, e se accetta, M' rifiuta, e viceversa
- M decide $L \Rightarrow$ M' decide $\bar{L} \Rightarrow \bar{L}$ e' riconoscibile $\Rightarrow L$ e' coriconoscibile.

[⇐]: Se L e $\bar{L}$ sono riconoscibili e $L(M) = L \wedge L(M') = \bar{L}$ definisco N:

- N su input ω esegue NON DETERMINISTICAMENTE $M(\omega)$ e $M'(\omega)$
- Se M accetta N accetta. Se M' rifiuta N accetta
- $\omega \in L \vee \bar{L} \Rightarrow$ M' o M accettano $\Rightarrow$ N decide L. ∎

3 Complessità — **10 Points**

- Determinare se il seguente problema è in NL: Dato un grafo non orientato $G = (V, E)$, due vertici $s, t \in V$, ed un intero $k \in \mathbb{N}$, stabilire se esiste un cammino da s a t di lunghezza esattamente k.
- Definire le classi di complessità P ed NP, e la nozione di NP-completezza. Mostrare che se P = NP, allora la classe NP coincide con la classe di problemi NP-completi.

PATH e' in NL, e l'algoritmo cerca un nodo di mezzo u per cui esistono i cammini $s \to u$ e $u \to t$. Si può considerare nell'algoritmo un contatore e stabilire se l'eventuale cammino e' lungo K.

- il contatore i e' limitato dal numero di nodi, che n
- per rappresentare i bastano $\log_2(n)$ celle.

Quindi il problema e' in NL. Ovviamente questa risposta e' errata perche' il problema presentato e' NP-Completo.

- $P = \bigcup_K Dtime(n^k)$
- $NP = \bigcup_K Ntime(n^k)$
- A e' NP-completo se $\begin{cases} \forall L \in NP \quad L \leq_m^P A \\ A \in NP \end{cases}$

Teorema di Ladner: $P = NP \Rightarrow$ NP-completi $= NP$

Un problema e' NP-completo se e' in NP ed ogni linguaggio in NP si riduce ad esso.

Voglio mostrare che se $A \in NP$ e $B \in NP$, $A \leq_m^P B$.

$A = L(M)$ (=P) $\quad B = L(M')$ (=P)

$\exists R$ t.c. $x \in A \iff R(x) \in B$

- R su input x esegue $M(x)$
- Se accetta scrive in output $\omega \in B$
- Se rifiuta // // $\omega \notin B$