Espressioni Regolari

> 🖊 Espressioni Regolari
>
> Dato un alfabeto $\sum$, si definisce espressione regolare di $\sum$ una stringa R rappresentante un linguaggio $L(R) \subseteq \sum^*$.
> Quindi ogni espressione regolare R è in verità il linguaggio $L(R)$ ad essa associata.
> Le **espressioni regolari (regex)** non sono solo delle stringhe con simboli speciali, ma in **teoria della computazione** sono una notazione compatta per descrivere un **linguaggio formale**, cioè un insieme di stringhe costruite su un alfabeto Σ.
>
> Ogni espressione regolare R corrisponde ad un linguaggio $L(R)$.

Costruzione delle espressioni regolari

L'insieme delle espressioni regolari su un alfabeto Σ, indicato con $re(\Sigma)$, si definisce induttivamente:

Casi base

- $\emptyset \in re(\Sigma)$
  - Linguaggio vuoto: $L(\emptyset) = \{\}$

- $\varepsilon \in re(\Sigma)$
  - Linguaggio contenente solo la stringa vuota: $L(\varepsilon) = \varepsilon$

- $a \in re(\Sigma)$ con $a \in \Sigma$
  - Linguaggio con la stringa "a": $L(a) = a$

Costruzioni induttive (operazioni sui linguaggi)

Se $R1, R2 \in re(\Sigma)$, allora:

- ***Unione***: $R1 \cup R2 \in re(\Sigma)$
  - Linguaggio: $L(R1 \cup R2) = L(R1) \cup L(R2)$ e significa "**o R1 o R2**"

- ***Concatenazione***: $R1 \circ R2 \in re(\Sigma)$
  - Linguaggio: $L(R1 \circ R2) = xy \mid x \in L(R1), y \in L(R2)$ e significa "**prima R1, poi R2**"
    Se $R \in re(\Sigma)$, allora:

- ***Star (Kleene star)***: $R^* \in re(\Sigma)$
  - Linguaggio: $L(R*) = \varepsilon, w, ww, www, \ldots \mid w \in L(R)$ e significa "**zero o più ripetizioni di R**"

- ***Plus (Kleene plus)***: $R^+ \in re(\Sigma)$
  - Linguaggio: $L(R^+) = L(R) \circ L(R^*)$ e significa "**una o più ripetizioni di R**"

Classi dei linguaggi descritti da esp. reg.

Dato un alfabeto Σ, definiamo come classe dei linguaggi di Σ descritti da un'espressione regolare il seguente insieme:

$$L(re) = L \subseteq \Sigma^* | \ \exists \ R \in re(\Sigma) \ t.c. \ L = L(R)$$

La conversione da espresione regolare a NFA, ($\mathcal{L}(re) \subseteq \mathcal{L}(NFA)$)

Si procede dimostrando per inclusione, quindi che se per ogni sotto-componente vale una determinata proprietà, allora essa varrà anche per ogni componente formato da tali sotto-componenti.

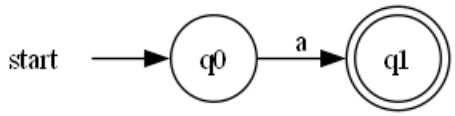
- *Caso Base*:
  - Se $R = \emptyset \in re(\sum)$, definiamo il NFA $N_\emptyset = (\{q_0\}, \sum, \delta, q_0, \emptyset)$ ossia:

    Per cui si ha che $w \in L(R) \iff w \in L(N_\emptyset)$, quindi che $L(R) = L(N_\emptyset) \in \mathcal{L}(NFA)$
  - Se $R = \epsilon \in re(\sum)$, definiamo il NFA $N_\emptyset = (\{q_0\}, \sum, \delta, q_0, \{q_0\})$ ossia:

    Per cui si ha che $w \in L(R) \iff w \in L(N_\epsilon)$, quindi che $L(R) = L(N_\emptyset) \in \mathcal{L}(NFA)$
  - Se $R = a \in re(\sum)$ con $a \in \sum$, definiamo il NFA $N_a = (\{q_0, q_1\}, \sum, \delta, q_0, \{q_1\})$ dove per δ è definita solo la coppia $\delta(q_0, a)$, ovvero:

    Per cui si ha che $w \in L(R) \iff w \in L(N_a)$, quindi che $L(R) = L(N_a) \in \mathcal{L}(NFA)$

- *Ipotesi Induttiva:*
  - Date due stringhe $R_1, R_2 \in re(\sum)$, assumiamo che $\exists\ NFA\ N_1, N_2 | L(R_1) = L(N_1), L(N_2) = L(N_2)$, dunque che $L(R_1), L(R_2) \in (L)(NFA)$

- *Passo Induttivo*:
  - Se $R = R_1 \cup R_2$, tramite ***Chiusura dell'UNIONE in REG***, si ottiene:

    $$L(R) = L(R_1) \cup L(R_2) = L(N_1) \cup L(N_2) \in REG = \mathcal{L}(NFA)$$

  - Se $R = R_1 \circ R_2$, tramite ***Chiusura della CONCATENAZIONE in REG***, si ottiene:

    $$L(R) = L(R_1) \circ L(R_2) = L(N_1) \circ L(N_2) \in REG = \mathcal{L}(NFA)$$

  - Se $R = {R_1}^*$, tramite ***Chiusura di PLUS in REG***, si ottiene:

    $$L(R) = L(R_1)^* = L(N_1)^* \in REG = \mathcal{L}(NFA)$$

□

NFA GENERALIZZATI

Generalized NFA (GNFA)

Un ***Generalized NFA (GNFA)*** è una quintupla formata da $(Q, \sum, \delta, q_{start}, q_{accept})$ dove:

- Q è l'insieme finito degli stai dell'automa dove $|Q| >= 2$
- $\sum$ è l'alfabeto dell'automa
- $q_{start} \in Q$ è l'unico stato iniziale dell'automa
- $q_{accept} \in Q$ è l'unico stato accettante dell'automa
- $\delta : (Q - \{q_{accep}\}) \times (Q - \{q_{start}\}) \longrightarrow re(\sum)$ è la *funzione di transizione* degli stati dell'automa, implicando che :
  - Lo stato q_{start} abbia solo transizioni *uscenti*
  - Lo stato q_{accapt} abbia solo transizioni *entranti*
  - Tra tutte le *possibili coppie di stati* $q, q' \in Q$ vi sia una transizione $q \longrightarrow q'$ ed una transizione $q' \longrightarrow q$
  - le "etichette" delle transizioni sono delle **espressioni regolari**

Stringa accettata da un GNFA

Sia $G := (Q, \sum, \delta, q_{start}, q_{accept})$ un GNFA. Data una stringa $w := w_0 \dots w_k \in \sum^*$ dove $w_0 \dots w_k \in \sum^*$(sono delle sottostringhe), si dice che w è una stringa ***accettata da G*** se esiste una sequenza di stati $r_0, r_1, \dots, r_{k+1} \in Q$ tali che:

- $r_0 = q_{start}$
- $\forall i \in [0, k] w_i \in L(\delta(r_i, r_{i+1}))$
- $r_{k+1} = q_{accept}$

Una transazine con "etichetta" identificata con $\emptyset$ è una ***transazione inutilizzabile*** in quanto $L(\emptyset) = \emptyset$

DEF: dato un alfabeto $\sum$, definiamo come ***classe dei linguaggi di*** $\sum$ ***riconosciuti da un GNFA*** il seguente insieme:

$$\mathcal{L}(GNFA) = \{L \subseteq \sum^* \mid \exists\, GNFA\ G\ t.c\ L = L(G)\}$$

La conversione da DFA a GNFA, ($\mathcal{L}(DFA) \subseteq \mathcal{L}(GNFA)$)

Dimostrazione:

- Dato $L \in \mathcal{L}(DFA)$, sia $D := (Q, \sum, \delta, q_0, F)$ il DFA tale che $L(D) = L$
- Si considera quindi il GNFA $G := (Q', \sum, \delta', q_{start}, q_{accept})$ costruito tramite D stesso:
  - $Q' = Q \cup \{q_{accept}, q_{start}\}$
  - $\delta'(q_{start,q_0}) = \epsilon$
  - $\forall q \in F \ \delta'(q, q_{accept}) = \epsilon$
  - Per ogni transizione che ha etichetta multipla in D, implica che in G esiste una transazione equivalente con etichetta corrispondente all'unionedi tali etichette multiple.
  - Per ogni coppia di stati in cui non esiste una transazione entrante o uscente i D, viene aggiunta una transazione con etichetta $\emptyset$.
- Adesso, per costruzione di G si ha:

$$w \in L = L(D) \implies L(G)$$

Implicando quindi che $L(D) \in \mathcal{L}(DFA)$ e quindi: $\mathcal{L} \subseteq \mathcal{L}(GNFA)$. □

Convertire G in un espressione regolare

🖊 **Algoritmo Convert(G):**

Supponiamo di voler convertire un **GNFA** GGG in un'espressione regolare.

- Sia k il numero di stati in G.
- **Se k=2**
  (cioè $G$$ ha solo lo stato iniziale e quello finale)
  allora l'arco tra i due stati è etichettato con una **espressione regolare** F; quell'etichetta è il **risultato finale**.
  $\longrightarrow$ **Output:** R.
- **Se k>2k**
  allora **scegliamo uno stato** q_r da eliminare
  Costruiamo un nuovo automa G' senza quello stato:

  $$G' = (Q', \Sigma, \delta', q_{start}, q_{accept})$$

  Dove $Q' = Q \setminus \{q_r\}$

Per ogni coppia di stati (p, q) **diversi da** q_r:

$$R'(p, q) = R(p, q) \ \cup \ [R(p, q_r) \cdot (R(q_r, q_r))^* \cdot R(q_r, q)]$$

(cioè: aggiungiamo ai cammini diretti da p a q anche quelli che passano per lo stato eliminato).

Il nuovo GNFA G' ha dunque archi aggiornati secondo questa formula.

Poi:

$$Convert(G) = Convert(G')$$

Spiegazione

In parole semplici, ogni volta che eliminiamo uno stato q_r
"ricuciamo" i cammini che passavano per esso, costruendo per ogni coppia di stati una nuova etichetta che rappresenta:

- **il vecchio cammino diretto** $R(p, q)$
- **più tutti i cammini che passano** per q_r
  (entrare in q_r, fare eventualmente dei giri, e poi uscire).

Per concludere, dobbiamo dimostrare che **Convert(G)** è equivalente a G, cioè:

$$L(Convert(G)) = L(G)$$

- *Caso Base:*
  - Se $k = 2$, è ovviamente vero: l'espressione regolare sull'arco fra $start$ e $accept$ escrive tutte le stringhe accettate da G.

- *Caso Induttivo:*
  - Supponiamo che sia vero per tutti i **GNFA** con $k - 1$ stati.
    Dimostriamo che vale anche per k stati.
    Rimuoviamo uno statoq_r da G e otteniamo G'.
    Per ipotesi induttiva, $Convert(G')$ produce un'espressione regolare equivalente a G'.

    Ora, se una stringa è accettata da , allora esiste un cammino:

    $$q_{start} \rightarrow \ldots \rightarrow q_r \rightarrow \ldots q_{accept}$$

    che può:

    - **non passare** per q_r (in tal caso la stringa è accettata già da G');
    - oppure **passare** per q_r una o più volte,
      in tal caso è rappresentata dal termine:

      $$R(p, q_r)(R(q_r, q_r)) * R(q, q)$$

      che è incluso nella definizione di $R'(p, q)$.

    Dunque e riconoscono lo stesso linguaggio:

    $$L(G) = L(G\prime)$$

    Applicando l'ipotesi induttiva:

    $$L(G) = L(G') = L(Convert(G'))$$

    e quindi:

    $$L(Convert(G)) = L(G)$$

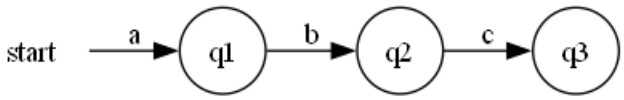
□

Spiegazione in parole semplici

Si sta dimostrando che **l'eliminazione di uno stato alla volta** non cambia il linguaggio riconosciuto dal GNFA.
Ogni volta che elimini uno stato, "si ricreano" tutte le possibili strade che passavano da quello stato sotto forma di **espressione regolare equivalente**.
Alla fine, quando rimangono solo due stati, l'etichetta che li collega racchiude **tutti i percorsi possibili** che portano da start a accept: cioè l'intero linguaggio riconosciuto dall'automa.

Esempio:

Trovare l'espressione regolare associata ad a



Rimuoviamo lo stato intermedio (q_2).
Applichiamo la formula:

$$R(start, q_3) = R(start, q_3) \cup [R(start, q_2) \cdot (R(q_2, q_2)) * \cdot R(q_2, q_3)]$$

Poiché $R(start, q_3) = \emptyset$, $R(2, 2) = \emptyset$, otteniamo:

$$R(start, q_3) = a \cdot (\emptyset) * \cdot c = ac$$

Quindi l'espressione regolare corrispondente è ***ac***.

Conversione da GNFA ad Espressione Regolare ($\mathcal{L}(GNFA) \in \mathcal{L}(re)$)

Dimostrazione:

- Dato $L \in \mathcal{L}(GNFA)$, sia $G := (Q, \sum, \delta, q_{accept}, q_{start})$ il GNFA tale che $L(G) = L$
- Dato il GNFA G' ottenuto applicando l'algoritmo di riduzione `convert(G)`, sia $R \in re(\Sigma)$ l'espressione regolare tale che $R = \delta'(q_{accept}, q_{start})$. Dato che l'unica transizione di G' ed essendo G' equivalente a G, ne consegue che:

  $$L = L(G) = L(G') = L(R) \in re(\Sigma)$$

  e quindi $\mathcal{L}(GNFA) \subseteq \mathcal{L}(re)$. □

Teorema: Equivalenza tra le espressioni e i linguaggi regolari

> **Definizione**
>
> Date le due classi dei linguaggi $\mathcal{L}(re)$ e REG, si ha: $\mathcal{L}(re) = REG$

Dimostrazione:

- **Prima implicazione:**
  - Grazie alla Conversione da espressione regolar a NFA si ottiene $\mathcal{L}(re) \subseteq \mathcal{L}(NFA) = REG$
  - Inoltre, dato che un NFA è un GNFA, allora si può dire che: $\mathcal{L}(NFA) \subseteq \mathcal{L}(GNFA)$
- **Seconda implicazione:**
  - Grazie alla dimostrazione per passare da un DFA ad un GNFA e la conversione da GNFA ad espressione regolare, si ha: $REG = \mathcal{L}(DFA) \subseteq \mathcal{L}(GNFA) \subseteq \mathcal{L}(re)$ □

Si ha quindi che dato un alfabeto Σ , si ha: $REG := \mathcal{L}(DFA) = \mathcal{L}(NFA) = \mathcal{L}(GNFA) = \mathcal{L}(re)$, ossia che ***per ogni linguaggio regolare L esistono un DFA, NFA e GNFA che lo riconoscono ed un espressione regolare che lo descrive***.