Leggi algebriche per i linguaggi

- $L \cup M = M \cup L$.
  L'unione è *commutativa*.
- $(L \cup M) \cup N = L \cup (M \cup N)$.
  L'unione è *associativa*.
- $(LM)N = L(MN)$.
  La concatenazione è *associativa*.

Nota: La concatenazione non è commutativa, *cioè*, esistono L e M tali che $LM \neq ML$.

- $\emptyset \cup L = L \cup \emptyset = L$.
  $\emptyset$ è *l'identità* per l'unione.
- $\{\epsilon\}L = L\{\epsilon\} = L$.
  $\{\epsilon\}$ è l'identità *sinistra* e *destra* per la concatenazione.
- $\emptyset L = L\emptyset = \emptyset$.
  $\emptyset$ è l'annichilatore *sinistro* e *destro* per la concatenazione.

- $L(M \cup N) = LM \cup LN$.
  La concatenazione è *distributiva a sinistra* sull'unione.
- $(M \cup N)L = ML \cup NL$.
  La concatenazione è *distributiva a destra* sull'unione.
- $L \cup L = L$.
  L'unione è *idempotente*.
- $\emptyset^* = \{\epsilon\}, \ \{\epsilon\}^* = \{\epsilon\}$.
- $L^+ = LL^* = L^*L, \ L^* = L^+ \cup \{\epsilon\}$

- $(L^*)^* = L^*$. La chiusura è *idempotente*.

Dimostrazione:

$$w \in (L^*)^* \iff w \in \bigcup_{i=0}^{\infty}\Big(\bigcup_{j=0}^{\infty} L^j\Big)^i$$

$$\iff \exists k, m \in \mathrm{N} : w \in (L^m)^k$$

$$\iff \exists p \in \mathrm{N} : w \in L^p$$

$$\iff w \in \bigcup_{i=0}^{\infty} L^i$$

$$\iff w \in L^*$$

Espressioni Regolari in UNIX

Le espressioni regolari fanno parte del sistema operativo UNIX fin dall'inizio e sono usate in vari comandi che elaborano testi (ad esempio `grep, sed, lex, vi, awk, ad, ex, pg`) per:

- elencare/eliminare righe che contengono un'espressione regolare
- sostituire un'espressione regolare (find/replace)
- ...

La sintassi è leggermente diversa da quella vista finora.

Espressioni Regolari in UNIX

- insiemi di caratteri: pattern elementari che specificano la presenza un carattere appartenente ad un certo insieme.
- ancore: legano il pattern a comparire una posizione specifica della riga (es. inizio, fine).
- gruppi: "racchiudono" l'espressione regolare che può quindi essere riferita come una singola entità.
- modificatori: specificano ripetizioni dell'espressione che precede il modificatore stesso.

Espressioni Regolari in UNIX: sintassi

- `[abc]` match any of the characters enclosed
- `[a-d]` match any character in the enclosed range
- `[set]` match any character not in the following set
- `.`match any single character except `<newline>`
- `[:alpha:]` some predefined character sets
- `[:digit:]`
- `\` treat the next char literally. Normally used to escape special characters such as "." and "*"

Proprietà dei Linguaggi regolari

- **Pumping Lemma.**
  Ogni linguaggio regolare soddisfa il pumping lemma. Se qualcuno vi presenta un falso linguaggio regolare, l'uso del pumping lemma mostrerà una contraddizione.
- **Proprietà di chiusura.**
  Come costruire automi da componenti usando delle operazioni, ad esempio dati L e M possiamo costruire un automa per $L \cap M$.
- **Proprietà di decisione.**
  Analisi computazionale di automi, cioè quanto costa controllare varie proprietà, come l'equivalenza di due automi.
- **Tecniche di minimizzazione.**
  Possiamo risparmiare costruendo automi più piccoli.

Il Pumping Lemma, informalmente

- Supponiamo che $L_{01} = \{0^n 1^n : n \geq 1\}$ sia regolare.
- Allora deve essere accettato da un qualche DFA A, con, ad esempio, k stati.
- Supponiamo che A legga 0^k. Avrà le seguenti transizioni:

$$\begin{array}{ll} \epsilon & p_0 \\ 0 & p_1 \\ 00 & p_2 \\ \ldots & \ldots \\ 0^k & p_k \end{array}$$

  $\Rightarrow \exists i < j : p_i = p_j$
- Chiamiamo q questo stato.

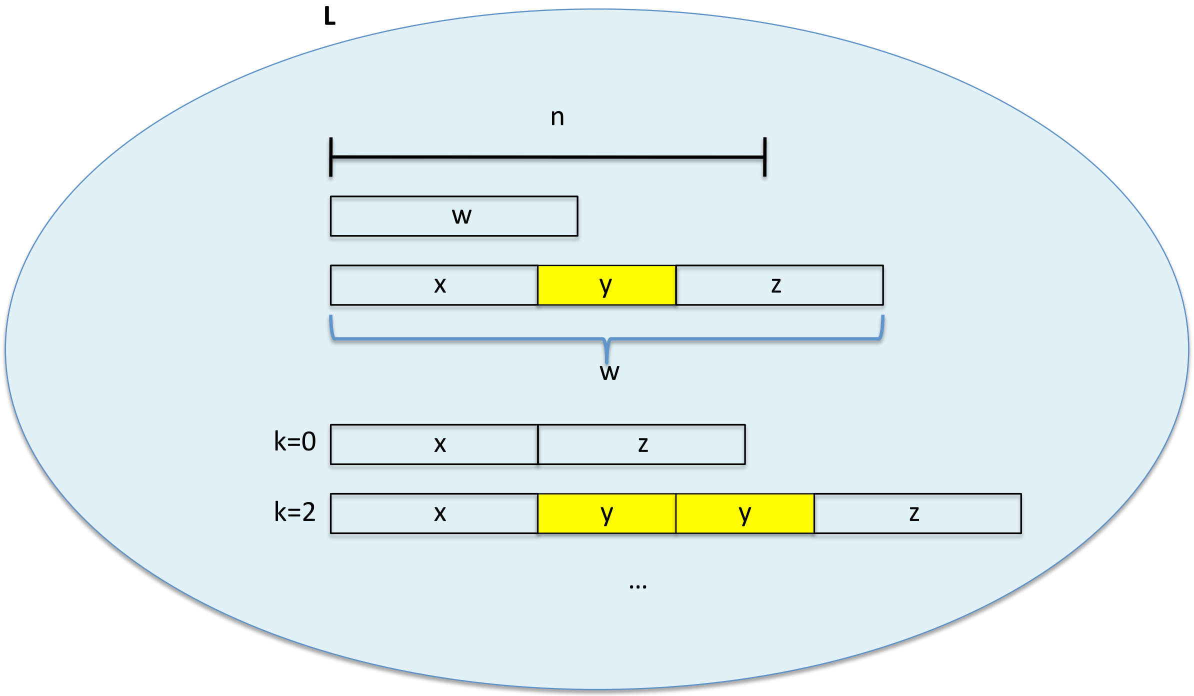
- Adesso possiamo ingannare A:
  - Se $\hat{\delta}(q, 1^i) \in F$ l'automa accetterà, sbagliando, $0^j 1^i$.
  - Se $\hat{\delta}(q, 1^i) \notin F$ l'automa rifiuterà, sbagliando, $0^i 1^i$.
- Quindi L_{01} non può essere regolare.

Teorema 4.1: Il Pumping Lemma per Linguaggi Regolari

Se L è un linguaggio regolare, per il Pumping Lemma, allora
Allora $\exists n, \forall w \in L : |w| \geq n \Rightarrow w = xyz$ tale che:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. $\forall k \geq 0, \; xy^k z \in L$

Intuitivamente



Intuitivamente (2)

- Esiste una costante n dipendente dal linguaggio L tale che tutte le stringhe di lunghezza $\geq n$ possono essere scomposte in un dato modo
- È sempre possibile scegliere una stringa *non vuota* y da replicare, ovvero **cancellare** o **ripetere** k volte, pur rimanendo all'interno del linguaggio L

Ovvero un cammino più lungo di n deve contenere un ciclo ed è il ciclo a pompare.

Dimostrazione

- Supponiamo che L sia regolare.
- Allora L è riconosciuto da un DFA A con, ad esempio, n stati $Q = \{q_0, ..., q_{n-1}\}$.
- Prendiamo come costante il valore n, e consideriamo una generica stringa $w \in L$ più lunga di n. Avremo quindi $w = a_1 a_2 \ldots a_m \in L$ con $m \geq n$.

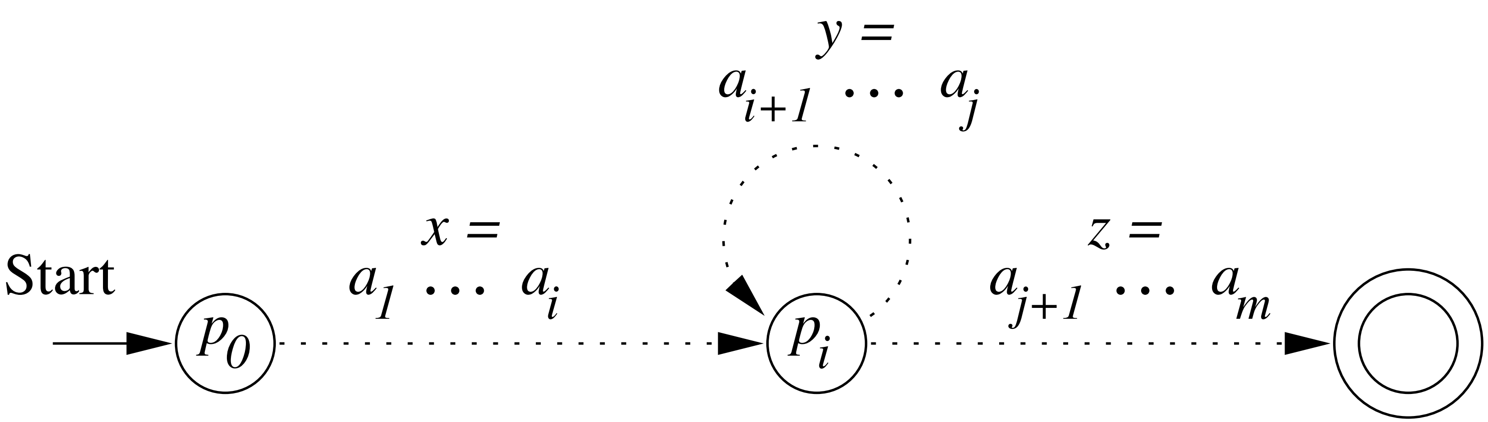
Dimostrazione (2)

Chiamiamo p_i, per $i \in \{0, ..., m\}$, lo stato in cui si trova l'automa A dopo avere esaminato $a_1 a_2 ... a_i$ a partire dallo stato iniziale q_0. Formalmente, utilizzando la funzione di transizione estesa:

- $p_0 = \hat{\delta}(q_0, \epsilon) = q_0$
- $p_i = \hat{\delta}(q_0, a_1 a_2 \cdots a_i)$.
- Dato che ci sono n stati distinti, gli $n + 1$ stati p_i non possono essere tutti distinti: $\Rightarrow \exists i < j : p_i = p_j$

Ora $w = xyz$, dove

1. $x = a_1 a_2 \cdots a_i$ (x porta a p_i la prima volta)
2. $y = a_{i+1} a_{i+2} \cdots a_j$ (y porta da p_i a p_i, dato che p_i e p_j coincidono)
3. $z = a_{j+1} a_{j+2} \ldots a_m$ (z conclude w)



Dimostrazione (4)

Notiamo che

- x può essere vuota (per $i = 0$) e anche z può essere vuota (per $j = n = m$). Invece
- $y \neq \epsilon$: la stringa y non è vuota, dato che $i < j$
- $|xy| \leq n$ dato che gli stati $p_0, \ldots, p_{j-1}$ sono tutti distinti (basta considerare il minimo indice che si ripete)

Data la forma dell'automa, è chiaro che, eseguendo $k \geq 0$ cicli in p_i, l'automa accetta ogni stringa xy^kz.

- per $k = 0$, l'automa passa dallo stato iniziale $q_0 = p_0$ a $p_i = p_j$ su input x. Allora passa da p_i allo stato accettante con input z. Quindi accetta xz.
- per $k > 0$, A va da q_0 a p_i su x, cicla su p_i per k volte su input y^k e passa allo stato accettante per z e accetta xy^kz

Quindi per $k \geq 0$, abbiamo che $xy^kz \in L(A)$

PL: una condizione necessaria per la regolarità

Il pumping lemma fornisce una condizione necessaria affinché un linguaggio sia regolare. Ovvero:

- L è regolare $\Rightarrow$ L soddisfa il Pumping Lemma
- L **non** soddisfa il Pumping Lemma $\Rightarrow$ L **non** è regolare

Il Pumping Lemma non dice che **solo** i linguaggi regolari possono godere della proprietà.

Dimostrare che un linguaggio non è regolare con il P.L.

L **non** soddisfa il Pumping Lemma $\Rightarrow$ L **non** è regolare

Non soddisfare il Pumping Lemma significa invertire l'implicazione, utilizzando il fatto che $A \Rightarrow B$ equivale a $\overline{B} \Rightarrow \overline{A}$. Con un po' di manipolazione algebrica possiamo passare quindi dalla formula:

$$\text{L reg.} \Rightarrow \left((\exists n\ \forall w \in L |w| \geq n \Rightarrow \left(\exists x, y, z\ t.c. \begin{cases} w = xyz \\ |xy| \leq n \\ y \neq \epsilon \end{cases} \wedge\ \forall k : xy^k z \in L \right) \right)$$

alla formula

$$\left(\forall n\ \exists w \in L |w| \geq n \wedge \left(\forall x, y, z\ t.c. \begin{cases} w = xyz \\ y \neq \epsilon \\ |xy| \leq n \end{cases} \Rightarrow\ \exists k : xy^k z \notin L \right) \right) \Rightarrow \overline{\text{L reg.}}$$

Esempio

- Sia $L_{01} = \{0^n 1^n\}$ il linguaggio delle stringhe formate da un certo numero di 0, seguiti dallo stesso numero di 1.
- Supponiamo che L_{01} sia regolare. Allora $w = 0^n 1^n \in L$ la stringa per n (infatti $|w| = 2n \geq n$)
- Per il pumping lemma, $w = xyz$, $|xy| \leq n$, $y \neq \epsilon$ e $xy^k z \in L_{01}$

$$w = \underbrace{000...}_{x}\underbrace{...000}_{y}\underbrace{0111...11}_{z} \quad \begin{cases} x = 0^i & \\ y = 0^h & h \geq 1 \wedge i + h \leq n \\ z = 0^j 1^n & i + h + j = n \end{cases}$$

- Valgono (1) e (2), ma
- non vale (3): consideriamo $xy^0 z = xz = 0^{i+j} 1^n$ ha meno 0 che 1: xz non sta nel linguaggio.
- Ne segue che L **non** è regolare.

Esempio (cont.)

Anche non considerando l'ipotesi $|xy| \leq n$, potevamo anche scegliere

- $y = 0^h 1^j$ ($x = 0^{n-h}$, $z = 1^{n-j}$): è chiaro che ripetendo la stringa k volte, gli 0 e gli 1 vengono mescolati; quindi la stringa ottenuta non sta nel linguaggio 1
- $y = 1^h$ è formata solo da 1: basta considerare xz ha meno 0 che 1 e non sta nel linguaggio

Esempio

- Sia L_{eq} il linguaggio delle stringhe con ugual numero di zeri e di uni.
- Supponiamo che L_{eq} sia regolare. Allora $w = 0^n1^n \in L_{eq}$.
- Per il pumping lemma, $w = xyz$, $|xy| \leq n$, $y \neq \epsilon$ e $xy^kz \in L_{eq}$

$$w = \underbrace{000\cdots}_{x}\underbrace{\cdots 0}_{y}\underbrace{0111\cdots 11}_{z}$$

- In particolare, $xz \in L_{eq}$, ma xz ha meno zeri di uni.
- In alternativa possiamo utilizzare la chiusura dei linguaggi regolari rispetto all'intersezione (vedi dopo) e procedere così:
  - Supponiamo che L_{eq} sia regolare
  - $\mathbf{0}^*\mathbf{1}^*$ sappiamo che è regolare
  - allora $L_{eq} \cap \mathbf{0}^*\mathbf{1}^* = L_{01}$ è regolare, ma questo non lo è (lo abbiamo dimostrato).
  - Quindi anche L_{eq} non è regolare.