Tutorato di Automi e Linguaggi Formali

Homework 4: Pumping Lemma, Linguaggi non Regolari, Scelta della parola per dimostrare che L non è regolare

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

Tutorato 4 - 31-03-2025

1 Scelta Strategica della Parola

Esercizio 1. Per ciascuno dei seguenti linguaggi, determinare quale sarebbe la scelta ottimale della stringa w da utilizzare per applicare il Pumping Lemma e dimostrare la non regolarità. Giustificare la scelta in base alle proprietà specifiche di ciascun linguaggio:

a) $L_1 = \{a^n b^n \mid n \geq 0\}$

b) $L_2 = \{a^n b^m \mid n > m \geq 0\}$

c) $L_3 = \{a^n \mid n \text{ è un numero primo}\}$

d) $L_4 = \{a^{n^2} \mid n \geq 0\}$

e) $L_5 = \{ww \mid w \in \{a, b\}^*\}$ (stringhe che sono duplicazioni di una sottostringa)

Importante: Per ogni linguaggio, fornire la forma generale della stringa scelta e spiegare perché questa scelta permette di dimostrare efficacemente la non regolarità.

Soluzione. Per dimostrare la non regolarità attraverso il Pumping Lemma, dobbiamo scegliere una stringa $w \in L$ tale che, per qualsiasi decomposizione $w = xyz$ con $|xy| \leq p$ e $|y| > 0$, esista un valore di $i \geq 0$ per cui $xy^i z \notin L$. Analizziamo ciascun linguaggio:
a) $L_1 = \{a^n b^n \mid n \geq 0\}$
Scelta ottimale: $w = a^p b^p$
Giustificazione:

- Questa stringa ha lunghezza $2p \geq p$, quindi soddisfa il requisito $|w| \geq p$.
- Poiché $|xy| \leq p$, la decomposizione $w = xyz$ deve avere xy interamente contenuto nel blocco di a (i primi p caratteri).

- Quindi $y = a^k$ per qualche $k > 0$.
- Scegliendo $i = 0$, otteniamo $xy^0z = xz$ che ha meno a che b, cioè $xz = a^{p-k}b^p$ con $p - k < p$.
- Di conseguenza, $xz \notin L_1$ poiché in L_1 il numero di a deve uguagliare il numero di b.

Questa scelta è ottimale perché:

- Forza y a contenere solo caratteri a, creando uno squilibrio quando viene pompato
- È la stringa più semplice possibile che consente di applicare il Pumping Lemma
- Qualunque valore di p scelto dall'avversario, possiamo sempre costruire a^pb^p

b) $L_2 = \{a^nb^m \mid n > m \geq 0\}$
Scelta ottimale: $w = a^{p+1}b^p$
Giustificazione:

- Questa stringa ha $n = p + 1$ e $m = p$, quindi $n > m$ e $w \in L_2$.
- La lunghezza è $|w| = 2p + 1 \geq p$, soddisfacendo il requisito.
- Poiché $|xy| \leq p$, e ci sono $p + 1$ caratteri a all'inizio, y deve contenere solo caratteri a.
- Scegliendo $i = 0$, rimuoviamo almeno un carattere a, ottenendo $xz = a^{p+1-k}b^p$ con $k > 0$.
- Se $k = 1$, abbiamo $xz = a^pb^p$, e quindi $n = m$, il che implica $xz \notin L_2$.
- Se $k > 1$, abbiamo $n < m$, che è anche fuori da L_2.

Questa scelta è ottimale perché:

- Ha esattamente un a in più rispetto ai b, rappresentando il "caso limite" di L_2
- Quando si rimuove anche solo un a (con $i = 0$), la stringa esce immediatamente da L_2
- La proprietà $n > m$ viene violata con la minima alterazione possibile

c) $L_3 = \{a^n \mid n$ **è un numero primo**$\}$
Scelta ottimale: $w = a^q$ dove q è un numero primo con $q > p$
Giustificazione:

- Scegliamo un qualsiasi numero primo $q > p$, così che $|w| = q \geq p$.
- Dato che w contiene solo a, qualunque decomposizione $w = xyz$ avrà $y = a^k$ per qualche $k > 0$.
- Scegliendo $i = 2$, otteniamo $xy^2z = a^{q+k}$.
- Ora dobbiamo dimostrare che $q + k$ non è un numero primo:

- Se $k = 1$, allora $q + k = q + 1 = q + 1$ che è pari (eccetto per $q = 2$), quindi non primo.
- Se $k > 1$, allora $q + k$ è divisibile per $\gcd(q + k, k)$, che potrebbe essere > 1.
- In alternativa, possiamo scegliere i tale che $q + k(i - 1)$ sia divisibile per un numero diverso da 1 e sé stesso, rendendolo non primo.

Questa scelta è ottimale perché:

- Sfrutta le proprietà algebriche dei numeri primi
- Qualunque sia p, possiamo sempre trovare un numero primo $q > p$ (ci sono infiniti numeri primi)
- Spesso possiamo trovare un valore di i che porta a un numero composto

d) $L_4 = \{a^{n^2} \mid n \geq 0\}$
Scelta ottimale: $w = a^{p^2}$
Giustificazione:

- Scegliamo $w = a^{p^2}$, che appartiene a L_4 con $n = p$.
- La lunghezza è $|w| = p^2 \geq p$ per ogni $p \geq 1$.
- Qualsiasi decomposizione $w = xyz$ avrà $y = a^k$ per qualche $k > 0$.
- Scegliendo $i = 2$, otteniamo $xy^2z = a^{p^2+k}$.
- Dobbiamo dimostrare che non esiste alcun intero m tale che $p^2 + k = m^2$:
  - Per ogni k con $1 \leq k \leq p$, vale $p^2 < p^2 + k < p^2 + 2p + 1 = (p + 1)^2$.
  - Quindi $p^2 + k$ si trova strettamente tra due quadrati consecutivi p^2 e $(p + 1)^2$.
  - Poiché $p^2 + k$ non è un quadrato perfetto, $xy^2z \notin L_4$.

Questa scelta è ottimale perché:

- Sfrutta il fatto che i quadrati perfetti hanno una "distanza" crescente tra loro
- Assicura che pompando qualsiasi sottostringa, si ottiene un numero di a che non è un quadrato perfetto
- È la stringa più semplice possibile per dimostrare la non regolarità di L_4

e) $L_5 = \{ww \mid w \in \{a, b\}^*\}$ **(stringhe che sono duplicazioni di una sottostringa)**
Scelta ottimale: $w = a^p b^p a^p b^p$
Giustificazione:

- Questa stringa è in L_5 poiché è la duplicazione di $a^p b^p$.
- La lunghezza è $|w| = 4p \geq p$ per ogni $p \geq 1$.
- Poiché $|xy| \leq p$, la sottostringa xy deve essere interamente contenuta nella prima parte a^p della stringa.

- Quindi $y = a^k$ per qualche $k > 0$.
- Scegliendo $i = 2$, otteniamo $xy^2z = a^{p+k}b^pa^pb^p$.
- Questa stringa non può essere scritta come duplicazione di alcuna sottostringa:
  - La prima metà avrebbe lunghezza $2p + k/2$, che non è un intero se k è dispari.
  - Anche se k è pari, le due metà sarebbero $a^{p+k/2}b^pa^{k/2}$ e $a^{p-k/2}b^p$, che non sono uguali.

Questa scelta è ottimale perché:

- Forza il pompaggio nella prima parte della stringa, creando uno squilibrio
- È strutturata in modo da essere chiaramente una duplicazione prima del pompaggio
- Qualsiasi pompaggio distrugge la proprietà di duplicazione

Esercizio 2. Sia $\Sigma = \{1, \#\}$ e sia

$$Y = \{w \mid w = x_1\#x_2\#\cdots\#x_k \text{ per } k \geq 0, \text{ dove ogni } x_i \in 1^*, \text{ e } x_i \neq x_j \text{ per } i \neq j\}$$

a) Proporre una stringa $w \in Y$ ottimale per applicare il Pumping Lemma

b) Spiegare perché questa scelta è strategica rispetto ad altre possibili

c) Dimostrare formalmente che Y non è regolare utilizzando la stringa scelta

d) Discutere come la proprietà "tutte sezioni diverse" renda impossibile l'utilizzo di memoria finita

Suggerimento: Considerare una struttura che forzi qualsiasi automa a memorizzare un numero arbitrario di informazioni.

Soluzione. a) Stringa ottimale per il Pumping Lemma

La stringa ottimale da scegliere è:

$$w = 1^1\#1^2\#1^3\#\ldots\#1^p\#1^{p+1}$$

Questa stringa consiste di $p + 1$ sezioni di 1 separate da #, dove la i-esima sezione contiene esattamente i ripetizioni del simbolo 1.

b) Giustificazione della scelta strategica

Questa scelta è strategica per i seguenti motivi:

- Appartiene a Y poiché ogni sezione $x_i = 1^i$ è diversa dalle altre (hanno tutte lunghezze diverse)
- Ha lunghezza $\sum_{i=1}^{p+1} i + p = \frac{(p+1)(p+2)}{2} + p > p$ per ogni $p \geq 1$
- Forza il Pumping Lemma a generare una contraddizione chiara:
  - Se pompiamo una sezione di 1, generiamo una sezione ripetuta, violando la definizione di Y

- Se pompiamo un # con 1 adiacenti, alteriamo la struttura delle sezioni

- La scelta di usare lunghezze crescenti $(1, 2, 3, \ldots)$ rende più facile dimostrare la non regolarità rispetto ad altre scelte, come sezioni di lunghezza arbitraria o casuale

c) Dimostrazione formale della non regolarità di Y

Assumiamo per assurdo che Y sia regolare. Allora per il Pumping Lemma esiste una costante $p > 0$ tale che ogni stringa $w \in Y$ con $|w| \geq p$ può essere scritta come $w = xyz$ con le seguenti proprietà:

1. $|xy| \leq p$
2. $|y| > 0$
3. Per ogni $i \geq 0$, $xy^iz \in Y$

Consideriamo la stringa $w = 1^1\#1^2\#1^3\#\ldots\#1^p\#1^{p+1} \in Y$. Poiché $|w| > p$, possiamo applicare il Pumping Lemma.

Poiché $|xy| \leq p$, la sottostringa xy deve essere contenuta entro i primi p caratteri di w. Dato che le prime p sezioni hanno lunghezze crescenti e sono separate da #, xy può rientrare in uno dei seguenti casi:

Caso 1: y contiene solo caratteri 1 all'interno di una sezione. Se $y = 1^m$ per qualche $m > 0$ e fa parte della sezione 1^j, allora con $i = 2$ otteniamo:

$$xy^2z = 1^1\#1^2\#\ldots\#1^{j-1}\#1^{j+m}\#\ldots\#1^p\#1^{p+1}$$

Questo porta a una sezione di 1^{j+m} che potrebbe corrispondere in lunghezza a una sezione successiva (se $j + m \leq p + 1$), violando la condizione che tutte le sezioni devono essere diverse.

Caso 2: y contiene il simbolo # più eventuali caratteri 1 adiacenti. Con $i = 0$, rimuoviamo il simbolo #, unendo due sezioni adiacenti e creando una sezione più lunga che non era presente nella stringa originale. Questo viola ancora la proprietà fondamentale di Y.

Caso 3: y contiene più simboli #. Con $i = 2$, duplicheremmo alcune sezioni, violando nuovamente la condizione di unicità.

In tutti i casi possibili, esiste un valore di i tale che $xy^iz \notin Y$, contraddicendo l'ipotesi che Y sia regolare. Quindi, Y non è regolare.

d) Impossibilità di utilizzare memoria finita

La proprietà "tutte sezioni diverse" rende impossibile il riconoscimento di Y con memoria finita per le seguenti ragioni:

- Un automa a stati finiti avrebbe bisogno di memorizzare tutte le sezioni già incontrate per confrontarle con quelle successive

- Poiché il numero di sezioni diverse può essere arbitrariamente grande, e ogni sezione può avere lunghezza arbitraria, l'automa dovrebbe memorizzare una quantità non limitata di informazioni

- Con un numero finito di stati, un DFA può memorizzare solo una quantità limitata di informazioni sul suo input passato

- Per ogni DFA con n stati, possiamo costruire una stringa con $n+1$ sezioni tutte diverse, che l'automa non può distinguere correttamente da una stringa con sezioni ripetute
- In termini formali, questo fenomeno rappresenta il "pumping" di stati: dopo aver letto un certo numero di sezioni, l'automa deve ripetere alcuni stati, perdendo l'informazione necessaria per verificare che tutte le sezioni siano diverse

Questa è una dimostrazione intuitiva del perché il linguaggio Y richiede una forma di memoria illimitata, come quella di una pila o di un contatore non limitato, che non può essere implementata con un automa a stati finiti.

Esercizio 3. Sia $\Sigma = \{0, 1, +, =\}$ e

$$ADD = \{x = y + z \mid x, y, z \text{ sono numeri binari, e } x \text{ è la somma di } y \text{ e } z\}$$

a) Proporre almeno tre possibili stringhe candidate per dimostrare che ADD non è regolare

b) Per ciascuna, analizzare vantaggi e svantaggi nell'applicazione del Pumping Lemma

c) Scegliere la stringa ottimale e presentare la dimostrazione completa

d) Specificare perché è necessario "pompare" parti specifiche della stringa scelta

Suggerimento: Considerare stringhe dove il "pompaggio" altera le proprietà aritmetiche in modo evidente.

Soluzione. **a) Tre stringhe candidate per dimostrare che ADD non è regolare**

Candidato 1: $w_1 = 1^p = 1^p + 0$

- Rappresenta l'equazione $2^p - 1 = 2^p - 1 + 0$
- In notazione binaria: una sequenza di p cifre 1 uguali a se stessi più zero

Candidato 2: $w_2 = 10^p = 10^p + 0$

- Rappresenta l'equazione $2^p = 2^p + 0$
- In notazione binaria: 1 seguito da p zeri, uguale a se stesso più zero

Candidato 3: $w_3 = 10^p = 1 + 0^p 1$

- Rappresenta l'equazione $2^p = 1 + (2^p - 1)$
- In notazione binaria: 1 seguito da p zeri, uguale a 1 più una sequenza di $p-1$ zeri seguiti da 1

b) Analisi dei vantaggi e svantaggi di ciascuna stringa

Analisi di $w_1 = 1^p = 1^p + 0$:

- *Vantaggi*:
  - Struttura semplice con parti chiaramente identificabili

- Pompando le cifre 1 si altera facilmente il valore numerico
- La presenza di tutti 1 a sinistra e al centro facilita l'applicazione del Pumping Lemma

- *Svantaggi*:
  - Il riporto nei numeri binari può rendere complessa la dimostrazione dell'invalidità dopo il pompaggio
  - Se y contiene simboli da entrambe le parti dell'equazione, la dimostrazione può diventare più complessa

Analisi di $w_2 = 10^p = 10^p + 0$:

- *Vantaggi*:
  - La potenza di 2 è rappresentata chiaramente
  - Pompare gli zeri modifica in modo semplice e prevedibile il valore
  - Non ci sono riporti da gestire quando si pompano solo zeri
- *Svantaggi*:
  - Con $|xy| \leq p$, potremmo non riuscire a "toccare" il bit più significativo (il 1 iniziale)
  - Se y contiene solo zeri, il pompaggio può non alterare l'uguaglianza in alcuni casi

Analisi di $w_3 = 10^p = 1 + 0^p1$:

- *Vantaggi*:
  - L'equazione rappresenta una somma non banale
  - Pompando zeri in una delle parti si altera facilmente l'uguaglianza
  - La struttura asimmetrica rende più facile dimostrare che il pompaggio viola l'uguaglianza
- *Svantaggi*:
  - La struttura è più complessa da analizzare
  - Pompare zeri in diverse posizioni può richiedere casi di analisi separati
  - La dimostrazione può richiedere una maggiore attenzione ai dettagli aritmetici

c) Scelta della stringa ottimale e dimostrazione completa

Scegliamo $w_2 = 10^p = 10^p + 0$ come stringa ottimale, poiché offre il miglior equilibrio tra semplicità e efficacia dimostrativa.

Dimostrazione formale che ADD non è regolare:

Assumiamo per assurdo che ADD sia regolare. Allora esiste una costante $p > 0$ tale che ogni stringa $w \in ADD$ con $|w| \geq p$ può essere scritta come $w = xyz$ con:

1. $|xy| \leq p$
2. $|y| > 0$
3. Per ogni $i \geq 0$, $xy^i z \in ADD$

Consideriamo la stringa $w = 10^p = 10^p + 0 \in ADD$, che rappresenta l'equazione $2^p = 2^p + 0$.

Poiché $|xy| \leq p$ e la stringa inizia con $10^p =$, la decomposizione xyz deve avere xy interamente contenuto nella prima parte $10^p =$ (potrebbe includere il simbolo $=$). Analizziamo i possibili casi per y:

Caso 1: y contiene solo caratteri 0 nella parte 10^p. Sia $y = 0^k$ per qualche $k > 0$. Scegliendo $i = 2$, otteniamo:

$$xy^2z = 10^{p+k} = 10^p + 0$$

Questo rappresenta $2^{p+k} = 2^p + 0$, che è falso poiché $2^{p+k} > 2^p$ per ogni $k > 0$.

Caso 2: y contiene il simbolo 1 all'inizio della stringa. Sia $y = 10^j$ per qualche $j \geq 0$. Scegliendo $i = 0$, otteniamo:

$$xy^0z = 0^{p-j} = 10^p + 0$$

Questo rappresenta $2^{p-j} = 2^p + 0$ (o semplicemente $0 = 2^p + 0$ se $j = p$), che è falso.

Caso 3: y contiene il simbolo $=$. Scegliendo $i = 0$, eliminiamo il simbolo $=$, producendo una stringa che non è più nella forma $x = y + z$ e quindi non appartiene a ADD.

In tutti i casi possibili, esiste un valore di i per cui $xy^i z \notin ADD$, contraddicendo l'ipotesi che ADD sia regolare. Quindi, ADD non è regolare.

d) Necessità di pompare parti specifiche

È necessario "pompare" parti specifiche della stringa scelta per i seguenti motivi:

- *Pompare zeri nella rappresentazione binaria*:
  - Modificare il numero di zeri nella rappresentazione binaria cambia il valore del numero in modo prevedibile e matematicamente rigoroso
  - Aggiungere zeri dopo il bit più significativo moltiplica il valore per potenze di 2
  - Questo effetto è cruciale per dimostrare che l'uguaglianza aritmetica viene violata
- *Pompare il simbolo di uguaglianza o operazione*:
  - Rimuovere o duplicare simboli come $=$ o $+$ distrugge completamente la struttura sintattica dell'espressione
  - Una stringa senza il simbolo $=$ non può appartenere a ADD per definizione
- *Pompare cifre in posizioni diverse dell'equazione*:
  - Pompare cifre a sinistra dell'uguaglianza senza modificare la parte destra (o viceversa) altera l'equilibrio dell'equazione

- Questo è il principio fondamentale che dimostra perché una macchina a stati finiti non può verificare l'uguaglianza aritmetica: non può "bilanciare" i cambiamenti tra le parti dell'equazione

In conclusione, il linguaggio ADD richiede la capacità di verificare operazioni aritmetiche, che implica il confronto e la memorizzazione di valori potenzialmente illimitati. Questo va oltre le capacità di un automa a stati finiti e dimostra formalmente perché ADD non è regolare.

2 Strategie di Dimostrazione

Esercizio 4. Per i seguenti linguaggi, applicare il Pumping Lemma nella forma del "gioco" tra Avversario (che difende la regolarità) e Dimostratore (che nega la regolarità):

a) $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e se } i = 1 \text{ allora } j = k\}$

b) $L = \{w \in \{a, b, c\}^* \mid \text{ogni } a \text{ in } w \text{ è seguita da almeno una } b \text{ e una } c \text{ (in qualsiasi ordine)}\}$

c) $L = \{w \in \{a, b\}^* \mid w \text{ contiene lo stesso numero di occorrenze dei pattern } ab \text{ e } ba\}$

Per ogni linguaggio:

- Descrivere la strategia per la scelta della stringa w
- Analizzare i possibili modi in cui l'Avversario può decomporre $w = xyz$
- Mostrare come il Dimostratore può sempre trovare un valore i tale che $xy^iz \notin L$

Soluzione. Il gioco del Pumping Lemma si struttura come segue:

- L'Avversario sceglie una costante $p > 0$ (la pumping length)
- Il Dimostratore seleziona una stringa $w \in L$ con $|w| \geq p$
- L'Avversario suddivide $w = xyz$ con $|xy| \leq p$ e $|y| > 0$
- Il Dimostratore deve trovare un valore $i \geq 0$ tale che $xy^iz \notin L$

a) $L = \{a^i b^j c^k \mid i, j, k \geq 0$ **e se** $i = 1$ **allora** $j = k\}$
Strategia per la scelta della stringa w:
Il Dimostratore sceglie $w = ab^pc^p$, che appartiene a L poiché $i = 1$, $j = k = p$, rispettando la condizione che se $i = 1$ allora $j = k$.
Analisi delle possibili decomposizioni dell'Avversario:
L'Avversario deve scegliere $w = xyz$ con $|xy| \leq p$ e $|y| > 0$. Considerando che $w = ab^pc^p$, abbiamo:

- *Caso 1*: y contiene solo il carattere a. Qui $y = a$ e $x = \varepsilon$ (stringa vuota).
- *Caso 2*: y contiene solo caratteri b. Qui $y = b^m$ per qualche $1 \leq m \leq p$, e x contiene l'unico carattere a più eventualmente alcuni b.

- *Caso 3*: y contiene a e alcuni b. Qui $y = ab^m$ per qualche $0 \leq m < p$, e $x = \varepsilon$.

Strategia vincente del Dimostratore:

- *Caso 1*: Se $y = a$, il Dimostratore sceglie $i = 2$. Otteniamo $xy^2z = a^2b^pc^p$, che non è in L perché $i = 2$ (non è uguale a 1), ma la condizione speciale "se $i = 1$ allora $j = k$" non si applica qui. La stringa rimane in L.

  Questo caso è problematico! Proviamo una stringa diversa: $w = a^2b^{p+1}c^p$. Qui $i = 2$, $j = p + 1$, $k = p$, quindi $j \neq k$ ma la stringa è in L perché $i \neq 1$.

  Se l'Avversario sceglie $y = a$, il Dimostratore seleziona $i = 0$. Otteniamo $xy^0z = ab^{p+1}c^p$, che non è in L perché ora $i = 1$ ma $j = p + 1 \neq p = k$.

Analisi rivista per $w = a^2b^{p+1}c^p$:

- *Caso 1*: y contiene solo caratteri a.
  - Se $y = a$, con $i = 0$ otteniamo $xy^0z = ab^{p+1}c^p \notin L$ (come mostrato sopra).
  - Se $y = a^2$, con $i = 0$ otteniamo $xy^0z = b^{p+1}c^p \notin L$ perché manca il prefisso a^i.
- *Caso 2*: y contiene caratteri a e b. Con $i = 0$, rimuoviamo parte del prefisso, alterando la relazione tra i, j e k, portando a una stringa non in L.
- *Caso 3*: y contiene solo caratteri b. Con $i = 2$, aumentiamo il numero di b senza modificare a o c, ottenendo $a^2b^{p+1+m}c^p$ che resta in L (poiché $i = 2$, la condizione speciale non si applica).

Questo linguaggio presenta un caso interessante! Proviamo una terza stringa: $w = ab^pc^{p+1}$.

Con questa stringa, $i = 1$, $j = p$, $k = p + 1$, quindi $j \neq k$ e $w \notin L$. Correggiamo: $w = ab^pc^p$.

Se y contiene solo b, con $i = 0$ otteniamo $xy^0z = ab^{p-m}c^p \notin L$ perché ora $j = p - m < p = k$, violando la condizione che se $i = 1$ allora $j = k$.

b) $L = \{w \in \{a, b, c\}^* \mid$ ogni a in w è seguita da almeno una b e una c (in qualsiasi ordine)

Strategia per la scelta della stringa w:

Il Dimostratore sceglie $w = (abc)^p$, che appartiene a L poiché ogni a è seguita da b e c.

Analisi delle possibili decomposizioni dell'Avversario:

L'Avversario deve scegliere $w = xyz$ con $|xy| \leq p$ e $|y| > 0$. Considerando la struttura ripetitiva di w:

- *Caso 1*: y è contenuto all'interno di uno o più gruppi (abc) completi.
- *Caso 2*: y attraversa confini di gruppi, ad esempio contiene la fine di un gruppo e l'inizio di un altro.

Strategia vincente del Dimostratore:

- *Caso 1*: Se $y = (abc)^m$ per qualche $m > 0$, con qualsiasi valore di i, otteniamo $xy^iz = (abc)^{p+(i-1)m}$ che rimane in L. Questo non aiuta il Dimostratore.

  Consideriamo invece y che contiene un carattere a ma non tutti i b e c che lo seguono.

- *Caso 2*: Se y contiene a ma non il successivo b o c, il Dimostratore sceglie $i = 0$. Rimuovendo y, otteniamo una stringa dove un carattere a non è seguito da almeno una b e una c, quindi $xy^0z \notin L$.

- *Caso 3*: Se y contiene un solo carattere a e almeno uno ma non tutti i b e c che lo seguono, il Dimostratore sceglie $i = 2$. Duplicando y, creiamo una sequenza dove un carattere a non è seguito dal completo insieme di b e c richiesti, quindi $xy^2z \notin L$.

La strategia migliore: Il Dimostratore sceglie una stringa più specifica, $w = a(bc)^p$, dove c'è un singolo a seguito da p coppie di bc.

Con questa scelta, se l'Avversario seleziona y che include a ma non tutte le bc che lo seguono, il Dimostratore può scegliere $i = 0$ per rimuovere l'unico a, ottenendo una stringa senza a, che banalmente appartiene a L.

Un'opzione più efficace: $w = (abc)^{p-1}a(bc)^{p-1}ab$. Con questa stringa, l'ultima a è seguita solo da b, non da c, quindi $w \notin L$.

Corretto, la strategia deve essere rivista. Optiamo per: $w = (abc)^p$.

Se l'Avversario sceglie y che include un carattere a ma non il completo bc che lo segue, il Dimostratore seleziona $i = 0$, ottenendo una stringa dove quell'a non è più seguito da b e c, quindi $xy^0z \notin L$.

c) $L = \{w \in \{a, b\}^* \mid w$ **contiene lo stesso numero di occorrenze dei pattern** ab **e** $ba\}$

Strategia per la scelta della stringa w:

Il Dimostratore sceglie $w = (ab)^p$, che appartiene a L poiché:

- Numero di occorrenze di ab: p

- Numero di occorrenze di ba: 0

Questa non è una buona scelta perché non ci sono occorrenze di ba.

Scegliamo invece: $w = (ab)^p(ba)^p$, che contiene esattamente p occorrenze di ab e p occorrenze di ba.

Analisi delle possibili decomposizioni dell'Avversario:

Dato che $|xy| \leq p$ e w inizia con $(ab)^p$, la decomposizione xyz avrà xy interamente contenuto nella prima parte $(ab)^p$. Possiamo avere:

- *Caso 1*: $y = (ab)^m$ per qualche $1 \leq m \leq \lfloor p/2 \rfloor$.

- *Caso 2*: y include un'occorrenza incompleta, come $y = b(ab)^{m-1}a$ o $y = b(ab)^m$ o $y = (ab)^m a$.

- *Caso 3*: y è completamente all'interno di un'occorrenza, come $y = a$ o $y = b$.

Strategia vincente del Dimostratore:

- *Caso 1*: Se $y = (ab)^m$, il Dimostratore sceglie $i = 2$. Con $xy^2z = (ab)^{p+m}(ba)^p$, abbiamo $p + m$ occorrenze di ab e p occorrenze di ba, quindi $xy^2z \notin L$.

- *Caso 2*: Se y è un frammento che rompe il pattern, come $y = a$ o $y = b$, il Dimostratore sceglie $i = 0$ o $i = 2$ a seconda del frammento. Questo altera la struttura delle occorrenze, creando uno squilibrio tra ab e ba.

  Ad esempio, se $y = a$, con $i = 0$ otteniamo una stringa con occorrenze di bb che alterano il conteggio dei pattern.

Per il caso specifico di $w = (ab)^p(ba)^p$:

- Se $y = (ab)^m$ per qualche $m > 0$, con $i = 2$ otteniamo $xy^2z = (ab)^{p+m}(ba)^p$ con $p + m$ occorrenze di ab e p occorrenze di ba. Quindi $xy^2z \notin L$.

- Se $y = a$ all'inizio di una coppia ab, con $i = 0$ rimuoviamo questa a, creando un pattern bab che contiene un'occorrenza di ba ma elimina un'occorrenza di ab. Il risultato sarà $p - 1$ occorrenze di ab e p occorrenze di ba, quindi $xy^0z \notin L$.

- Se $y = b$ alla fine di una coppia ab, con $i = 0$ rimuoviamo questa b, creando un pattern aa che elimina un'occorrenza di ab senza modificare le occorrenze di ba. Il risultato non sarà in L.

In tutti questi casi, il Dimostratore può trovare un valore di i tale che $xy^iz \notin L$, dimostrando che L non è regolare.

Esercizio 5. Per ciascuno dei seguenti linguaggi, stabilire se è regolare o non regolare. Se è regolare, fornire un'espressione regolare o un automa. Se non è regolare, dimostrarlo utilizzando il Pumping Lemma con particolare attenzione alla scelta della stringa:

a) $L = \{w \in \{a, b\}^* \mid w \text{ contiene almeno un } a \text{ e un } b, \text{ e il primo } a \text{ appare prima del primo } b\}$

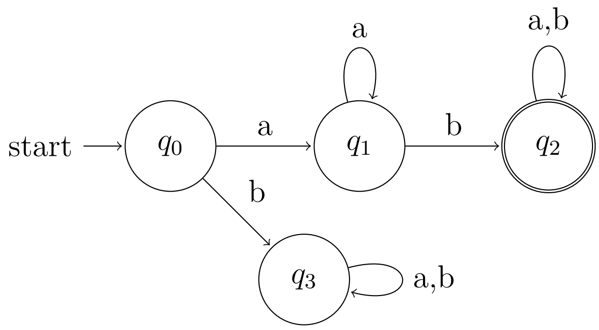
b) $L = \{w \in \{a, b\}^* \mid \text{la terza lettera da destra è } a\}$

c) $L = \{w \in \{a, b\}^* \mid w \text{ contiene un ugual numero di sottostringhe } aa \text{ e } bb\}$

d) $L = \{a^ib^j \mid i, j \geq 0 \text{ e } i \neq 2j\}$

Soluzione. a) $L = \{w \in \{a, b\}^* \mid w$ **contiene almeno un** a **e un** b, **e il primo** a **appare prima**

Questo linguaggio è regolare. Possiamo costruire un automa a stati finiti che lo riconosce:



L'automa funziona come segue:

- q_0: stato iniziale, nessun simbolo letto

- q_1: almeno un a è stato letto, ma nessun b

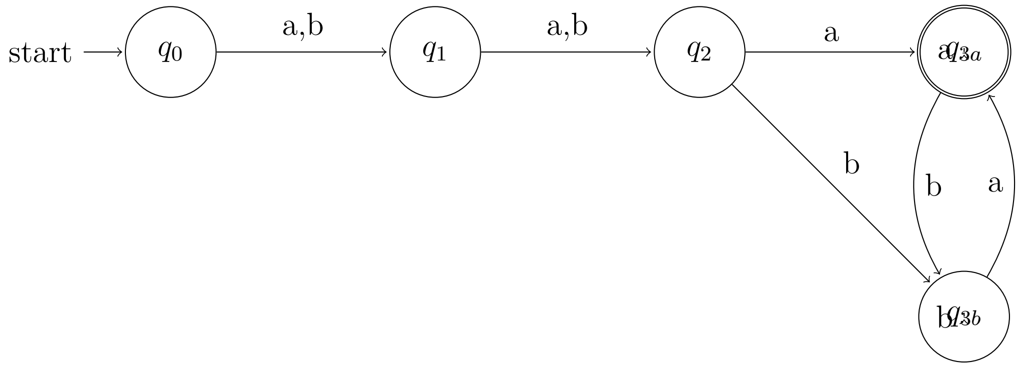
- q_2: almeno un a e un b sono stati letti, e il primo a appare prima del primo b

- q_3: stato "trappola" quando il primo b appare prima di qualsiasi a

Espressione regolare equivalente: $a^+b\{a,b\}^*$

b) $L = \{w \in \{a,b\}^* \mid$ **la terza lettera da destra è** $a\}$

Questo linguaggio è regolare. Possiamo costruire un DFA che tiene traccia solo degli ultimi tre caratteri letti:



L'automa tiene traccia delle ultime tre lettere lette:

- q_0: nessuna lettera letta
- q_1: una lettera letta
- q_2: due lettere lette
- q_{3a}: tre o più lettere lette, con la terza lettera da destra che è a
- q_{3b}: tre o più lettere lette, con la terza lettera da destra che è b

Espressione regolare equivalente: $\{a,b\}^*a\{a,b\}\{a,b\}$

c) $L = \{w \in \{a,b\}^* \mid w$ **contiene un ugual numero di sottostringhe** aa **e** $bb\}$

Questo linguaggio non è regolare. Dimostriamolo utilizzando il Pumping Lemma.

Scelta della stringa: $w = a^{2p}b^{2p}$

Questa stringa contiene:

- Numero di sottostringhe aa: $2p-1$ (tutte le coppie di a consecutive)
- Numero di sottostringhe bb: $2p-1$ (tutte le coppie di b consecutive)

Quindi $w \in L$ poiché ha lo stesso numero di sottostringhe aa e bb.

Applicazione del Pumping Lemma:

Assumiamo che L sia regolare. Allora esiste una costante $p > 0$ tale che ogni stringa $w \in L$ con $|w| \geq p$ può essere decomposta come $w = xyz$ con $|xy| \leq p$, $|y| > 0$, e per ogni $i \geq 0$, $xy^iz \in L$.

Per la stringa $w = a^{2p}b^{2p}$, poiché $|xy| \leq p$, la decomposizione xyz deve avere xy interamente contenuto nella prima parte a^{2p}. Quindi $y = a^k$ per qualche $k > 0$.

Scegliendo $i = 2$, otteniamo $xy^2z = a^{2p+k}b^{2p}$. Questa stringa contiene:

- Numero di sottostringhe aa: $(2p+k-1)$ (aumentato di k)
- Numero di sottostringhe bb: $(2p-1)$ (invariato)

Poiché $(2p + k - 1) > (2p - 1)$ per ogni $k > 0$, la stringa xy^2z contiene un numero diverso di sottostringhe aa e bb, quindi $xy^2z \notin L$.

Questo contraddice l'ipotesi che L sia regolare, quindi L non è regolare.

d) $L = \{a^i b^j \mid i, j \geq 0 \textbf{ e } i \neq 2j\}$

Questo linguaggio non è regolare. Dimostriamolo utilizzando il Pumping Lemma.

Scelta della stringa: $w = a^{2p+1}b^p$

Questa stringa ha $i = 2p + 1$ e $j = p$, quindi $i = 2j + 1 \neq 2j$, e $w \in L$.

Applicazione del Pumping Lemma:

Assumiamo che L sia regolare. Allora esiste una costante $p > 0$ tale che ogni stringa $w \in L$ con $|w| \geq p$ può essere decomposta come $w = xyz$ con $|xy| \leq p$, $|y| > 0$, e per ogni $i \geq 0$, $xy^iz \in L$.

Per la stringa $w = a^{2p+1}b^p$, poiché $|xy| \leq p$, la decomposizione xyz deve avere xy interamente contenuto nella prima parte a^{2p+1}. Quindi $y = a^k$ per qualche $k > 0$.

Scegliendo $i = 0$, otteniamo $xy^0z = a^{2p+1-k}b^p$. Analizziamo quando questa stringa non appartiene a L:

Per $xy^0z \notin L$, deve valere $(2p + 1 - k) = 2p$, cioè $k = 1$.

Con $k = 1$, abbiamo $xy^0z = a^{2p}b^p$, e poiché $2p = 2(p)$, questa stringa non appartiene a L.

Quindi, con la decomposizione che ha $y = a$, e scegliendo $i = 0$, otteniamo $xy^0z \notin L$, contraddicendo l'ipotesi che L sia regolare.

Pertanto, L non è regolare.