Appunti LFT

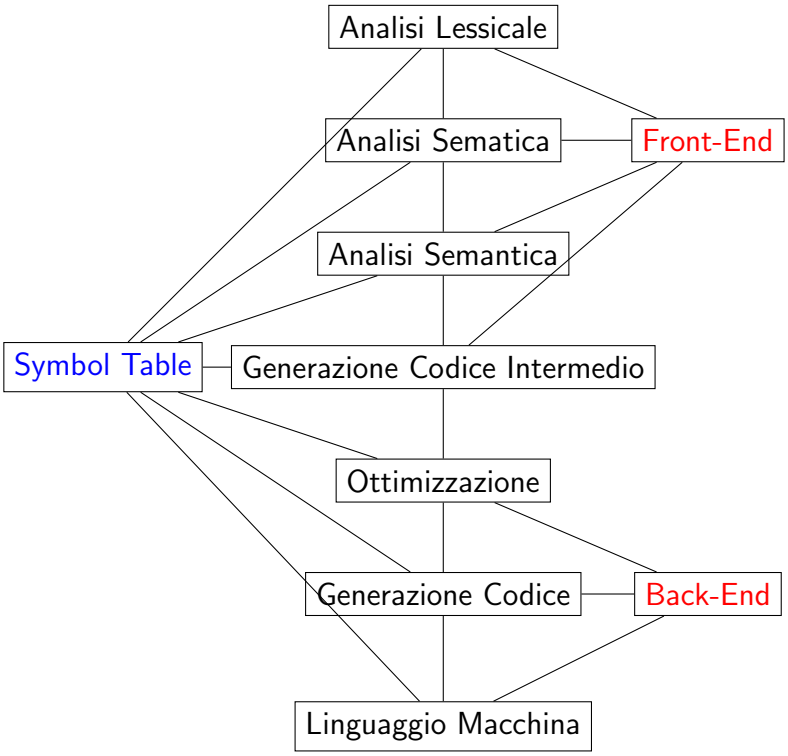
Lorenzo Tabasso

Aggiornato al 1 agosto 2017

Indice

Capitolo 1

Introduzione



Schema generale di un compilatore-interprete

1.1 Token

Coppia (nome token, valore attributo)

- **Nome Token**: simbolo astratto che rappresenta un'unita' lessicale (una parola chiave, un identicatore, ecc.).

1.2 Pattern

Descrizione della forma che i lessemi di un'unità lessicale possono avere.

1.3 Lessema

Sequenza di caratteri del programma sorgente che rispetta il pattern di un token.

Capitolo 2

Automi: metodo e follia

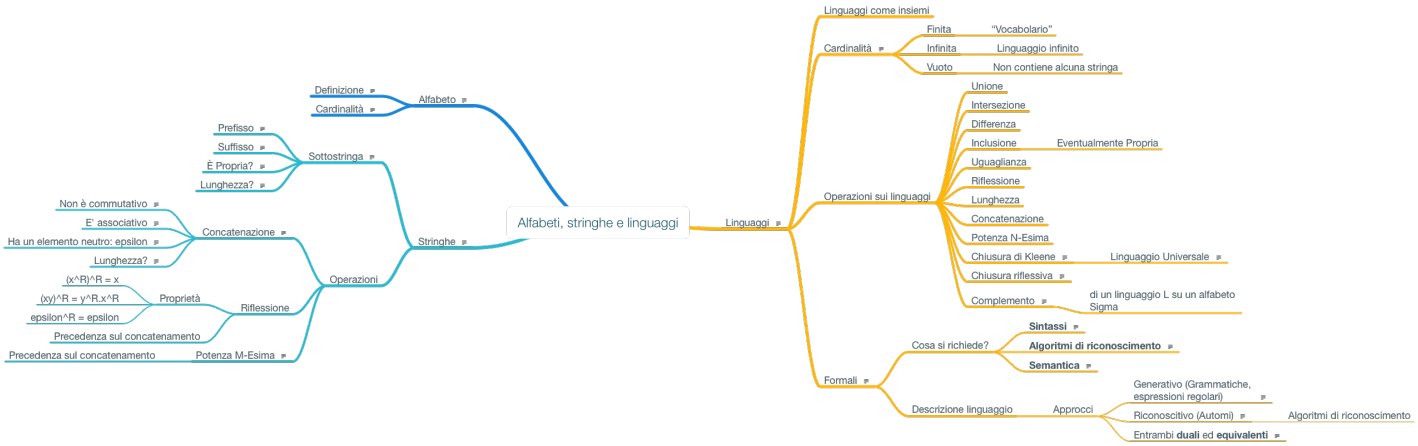


Figura 2.1: Mappa mentale degli argomenti del capitolo

2.1 Concetti Base

In questo paragrafo, introdurremo i concetti base della teoria degli automi

2.1.1 Alfabeto

Un alfabeto è un insieme finito e non vuoto di simboli (anche detti *caratteri*), si indica convenzionamene con il simbolo Σ. Tra gli alfabeti più comuni citiamo:

1. $\Sigma = \{0, 1\}$ l'alfabeto binario
2. $\Sigma = \{a, b, ..., z\}$ l'insieme di tutte le lettere minuscole
3. $\Sigma = \{\alpha, \beta, \gamma, \delta\}$ l'insieme delle prime quattro lettere minuscole dell'alfabeto greco

Cardinalità di un alfabeto

La cardinalità di un alfabeto è il numero di simboli dell'alfabeto. Se Σ denota l'alfabeto, $|\Sigma|$ denota la sua cardinalità. Di seguito alcuni esempi:

1. $|\Sigma| = |\{0, 1\}| = 2$
2. $|\Sigma| = |\{a, b, ..., z\}| = 27$
3. $|\Sigma| = |\{\alpha, \beta, \gamma, \delta\}| = 4$

2.1.2 Stringa

Una stringa (o *parola*) è una sequenza finita di simboli scelti da un alfabeto.
Ad esempio:

1. **aabb, cac, cba, abba** sono quattro stringhe sull'alfabeto {a,b,c}
2. **01101, 111** sono due stringhe sull'alfabeto {0,1}

Stringa Vuota

La *stringa vuota* è una stringa composta da 0 simboli, questa stringa indicata con ε è una stringa che può essere scelta da un qualunque alfabeto.

Lunghezza di una stringa

La lunghezza di una stringa è il numero di simboli della stringa stessa. Se x denota la stringa, $|x|$ denota la sua lunghezza. Ad esempio:

1. $|aabb| = 4$
2. $|cab| = 3$
3. $|101101| = 6$

Attenzione: $|\varepsilon| = 0$, la lunghezza della stringa vuota è sempre 0!

Stringhe uguali e diverse

Due stringhe della stessa lunghezza sono **uguali** se e solo se **i loro caratteri letti da sinistra verso destra coincidono**. Formalmente:

$$\text{Siano } x = a_1...a_n \text{ e } y = b_1...b_m$$
$$x = y \Leftrightarrow n = m \text{ and } \forall\ 1 \leq i \leq n \quad a_i = b_i$$

Contrariamente, se l'ordine non coincide, sono **diverse** tra loro.

Potenze di un alfabeto

Se Σ è un alfabeto, possiamo esprimere l'insieme di tutte le stringhe di una certa lunghezza su tale alfabeto usando una notazione esponenziale. Definiamo Σ^k come l'insieme di stringhe di lunghezza k, con simboli tratti da Σ. Ad esempio:

1. $\Sigma = \{0, 1\}$
   (a) $\Sigma^1 = \{0, 1\}$
   (b) $\Sigma^2 = \{00, 01, 10, 11\}$
   (c) $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Notare la **differenza tra** Σ **e** Σ^1, il primo è un alfabeto e i suoi membri sono i simboli { 0,1 }, mentre il secondo è un insieme di stringhe di lunghezza 1 su quell'alfabeto!

L'insieme di tutte le stringhe su un alfabeto Σ viene indicato con Σ^* (**Kleene Star**) e contiene anche ε, per escluderla si può usare Σ^+ (**Kleene Plus**), più chiaramente:

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup ...$
- $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Vedremo tutto ciò in maniera meglio approfondita più avanti.

Concatenazione di stringhe

siano x e y due stringhe, allora xy denota la loro concatenazione (o alternativamente anche $x.y$), vale a dire la stringa formata da una copia di x seguita da una copia di y. Più precisamente se x è la stringa composta da da i simboli $x = a_1a_2...a_i$ e y è la stringa composta da da j simboli $y = b_1b_2...b_j$, allora xy è la stringa di lunghezza $i + j$: $xy = a_1a_2...a_ib_1b_2...b_j$.
Per esempio:

1. $x = 01101$ $y = 110$
   (a) $xy = 01101110$
   (b) $xy = 11001101$
2. nano.tecnologie = nanotecnologie
3. tele.visione = televisione

Attenzione però:

1. ε è **l'identità** della concatenazione (detto anche *elemento neutro*), infatti, data una qualsiasi stringa w si ottiene $\varepsilon w = w\varepsilon = w$
2. Il concatenamento **non è commutativo**, infatti $xy \neq yx$
   (a) (tele.visione = televisione) $\neq$ (visione.tele = visionetele)
3. Il concatenamento **è associativo**, infatti $x(yz) = (xy)z$
   (a) nano.(tecno.logie) = nano.tecnologie = nanotecnologie
   (b) (nano.tecno).logie = nanotecno.logie = nanotecnologie

Sottostringa

La stringa y è una sottostringa della stringa x se esistono delle stringhe u e v tali che x = uyv. Per esempio, le sottostrighe di *abbc* sono ε, a, b, c, ab, bb, bc, abb, bbc, abbc.

Prefisso

La stringa y è un prefisso della stringa x se esiste una stringa v tale che x = yv. Un prefisso è una sottostringa in cui u = ε. Per esempio, i prefissi di *abbc* sono {ε, a, ab, abb, abbc}.

Suffisso

la stringa y è un suffisso della stringa x se esiste una stringa u tale che x = uy. Un suffisso è una sottostringa in cui v = ε. Per esempio, i suffissi di *abbc* sono {ε, c, bc, bbc, abbc}.

Sottostringa Propria

Una sottostringa (prefisso, suffisso) di una stringa e' **propria** se non coincide con ε o con la stringa stessa. Esempi:

- Le sottostrighe proprie di *abbc* sono {a, b, c, ab, bb, bc, abb, bbc},
- i prefissi propri di *abbc* sono {a, ab, abb}
- I suffissi propri di *abbc* sono {c, bc, bbc}

Lunghezza della sottostringa

se $|x| \geq k$ indichiamo con k : x il prefisso di x di lunghezza k (inizio di lunghezza k di x). Ad esempio:

- 2 : abbc = ab
- 3 : abbc = abb

2.1.3 Operazioni su stringhe

Riflessione

la riflessione di una stringa è la stringa ottenuta scrivendo i caratteri in ordine inverso. x^R denota la riflessione della stringa x.

$$(a_1...a_n)^R = a_n...a_1$$
$$(abbc)^R = cbba$$

la riflessione gode inoltre delle seguenti proprietà:

1. $(x^R)^R = x$
2. La riflessione della concatenazione di due stringhe e' la concatenazione inversa delle loro riflessioni, $(xy)^R = y^R x^R$.
3. La riflessione della stringa vuota e' la stringa vuota: $\varepsilon^R = \varepsilon$, e vale anche per $a^R = a^R$, se $a \in \Sigma$.
4. La riflessione ha **precedenza sul concatenamento**, $abbc^R = abbc$

Potenza m-esima

della stringa x è il concatenamento di x con se stessa m volte. x^m denota potenza m-esima di x.

$$x^0 = \varepsilon$$
$$x^m = x^m - 1x \qquad m > 0$$

Esempi:

- $(abbc)^3 = abbcabbcabbc$
- $(abbc)^6 = abbcabbcabbcabbcabbcabbc$
- $(aa)^2 = aaaa$

La potenza ha **precedenza sul concatenamento**: $abbc^3 = abbccc$

- $((ab)^R)^3 = (ba)^3 = bababa$
- $((ab)^3)^R = (ababab)^R = bababa$

2.1.4 Linguaggio

Un linguaggio su un alfabeto è un insieme di stringhe su quell'alfabeto. Le stringhe o parole di un linguaggio vengono anche chiamate *frasi*. Esempi:

- aabb, cac, cba, abba è un linguaggio sull'alfabeto a, b, c
- l'insieme dei numeri scritti in binario e' un linguaggio sull'alfabeto 0, 1
- l'insieme delle stringhe palindrome contenenti solo i simboli a, b, c e' un linguaggio sull'alfabeto a, b, c

N.B. il primo ed il terzo linguaggio hanno lo stesso alfabeto.

Dato un alfabeto quanti linguaggi si possono definire su di esso? **Infiniti**.

Linguaggi come insiemi

Un linguaggio può essere definito mediante un descrittore di insiemi: $\{w \mid \text{enunciato su } w\}$.
Questa espressione va letta come "l'insieme delle parole w tali che vale l'enunciato su w scritto a destra di | ". Certe volte w viene sostituito da un'espressione con parametri secondo l'uso della teoria degli insiemi, come ad esempio

$$\{0^n1^n \mid n \geq 1\}, \text{ oppure } \{0^n1^m \mid 0 \leq n \leq m\}$$

Cardinalità dei linguaggi

La cardinalità' di un linguaggio e' il numero delle sue stringhe. Se L denota un linguaggio, $\mid L \mid$ denota la sua cardinalità. Esempio:

- $\mid \{aabb, cac, cba, abba\} \mid = 4$
- $\mid$ Insieme dei numeri binari $\mid = \infty$

inoltre,

1. Un linguaggio è **finito** se la sua cardinalità è **finita**. Allora, esso è anche detto *vocabolario*.
2. Un linguaggio è **infinito** se la sua cardinalità è **infinita**.
3. Il **linguaggio vuoto** (denotato da Φ) è il linguaggio che non contiene alcuna stringa. $\mid \Phi \mid = 0$

2.1.5 Operazioni sui linguaggi

Unione

L'unione $L_1 \cup L_2$ dei linguaggi L_1 ed L_2 è l'insieme delle stringhe che appartengono a L_1 oppure a L_2.

$$L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$$

Intersezione

L'intersezione $L_1 \cap L_2$ dei linguaggi L_1 ed L_2 è l'insieme delle stringhe che appartengono sia a L_1 che a L_2.

$$L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$$

Differenza

La differenza $L_1 - L_2$ del linguaggio L_1 meno L_2 è l'insieme delle stringhe di L_1 che non appartengono a L_2.

$$L_1 - L_2 = \{x \mid x \in L_1 \text{ and } x \notin L_2\}$$

Incluso

Il linguaggio L_1 è incluso nel linguaggio L_2 (in notazione $L_1 \subseteq L_2$) se tutte le stringhe appartenenti a L_1 appartengono anche a L_2.

Inclusone propria

Il linguaggio L_1 è propriamente incluso nel linguaggio L_2 (in notazione $L_1 \subset L_2$) se tutte le stringhe di L_1 appartengono a L_2, e almeno una stringa di L_2 non appartiene a L_1.

Linguaggi uguali

Due linguaggi sono uguali se contengono lo stesso numero di stringhe.

$$L_1 = L_2 \Leftrightarrow L_1 \subseteq L_2 \text{ and } L_2 \subseteq L_1$$
$$L_1 = L_2 \Leftrightarrow L_1 - L_2 = L_2 - L_1 = \Phi$$

Riflessione

La riflessione del linguaggio L (in notazione L^R) è l'insieme delle stringhe riflesse di L.

$$L^R = \{x \mid x = y^R \text{ and } y \in L\}$$

Inizi di lunghezza k

L'insieme degli inizi di lunghezza k del linguaggio L (in notazione $k : L$) è l'insieme degli inizi di lunghezza k delle stringhe di L.

$$k : L = \{k : x \mid x \in L \text{ and } \mid x \mid \geq k\}$$

Concatenamento

Il concatenamento dei linguaggi L_1 ed L_2 (in notazione L_1L_2) è l'insieme ottenuto concatenando in tutti i modi possibili le stringhe di L_1 con le stringhe di L_2.

$$L_1L_2 = \{x \mid x = yz \text{ and } y \in L_1 \text{ and } z \in L_2\}$$
$$L\Phi = \Phi = \Phi L$$
$$L\{\varepsilon\} = L = \{\varepsilon\}L$$

Potenza m-esima

La potenza m-esima del linguaggio L (in notazione L^m) è il concatenamento di L con sè stesso m volte.

- $L^0 = \{\varepsilon\}$
- $L^m = L^{m-1}L$
- $\Phi^0 = \{\varepsilon\}$

In generale, si ha: $\{x \mid x = y^m \text{ and } y \in L\} \subseteq L^m$

Chiusura di Kleene (Kleene Star)

La chiusura di Kleene (o chiusura rispetto al concatenamento) del linguaggio L (notazione L^*) è l'unione di tutte le potenze di L.

$$L^* = \bigcup_{n=0}^{\infty} L^n = \{\varepsilon\} \cup L^1 \cup L^2 \cup ...$$

Essa gode delle seguenti proprietà:

1. Monotonicità $L \subseteq L^*$
2. Chiusura rispetto al concatenamento $(x \in L^*)$ and $(y \in L^*) \Rightarrow xy \in L^*$
3. Idempotenza $(L^*)^* = L^*$
4. Commutatività della riflessione con la chiusura di Kleene $(L^*)^R = (L^R)^*$
5. Commutatività della riflessione con la potenza $(L^m)^R = (L...L)^R = L^R...L^R = (L^R)^m$
6. $\Phi^* = \{\varepsilon\}$
7. $\{\varepsilon\}^* = \{\varepsilon\}$

Chiusura non riflessiva (Kleene Plus)

La chiusura non riflessiva rispetto al concatenamento del linguaggio L (notazione L^+) è l'unione di tutte le potenze positive di L.

$$L^+ = \bigcup_{n=1}^{\infty} L^n = L^1 \cup L^2 \cup ...$$

Essa gode delle seguenti proprietà:

1. $L^* = L^+ \cup \{\varepsilon\}$
2. $L^+ \subseteq L^*$
3. $L^+ = L^*L = LL^*$
4. $\varepsilon \in L^+ \Leftrightarrow \varepsilon \in L$

Linguaggio universale

Il linguaggio universale (o monoide libero di un alfabeto Σ) è la sua chiusura di Kleene:

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Contiene stringhe di lunghezza finita, ma illimitate. Inoltre, ogni lunguaggio su un alfabeto Σ è un sottoinsieme di Σ^*

Complemento

Il complemento di un linguaggio L su un alfabeto Σ rispetto a un alfabeto Δ (notazione $(\neg L)_\Delta$) è la differenza fra Δ^* ed L

$$\neg L_\Delta = \Delta^* - L$$

$\neg L$ indicherà il complemento fatto rispetto all'alfabeto su cui L è definito. Esempi:

- $\neg\{ab, ba\}_{a,b} = \{\varepsilon, a, b, aa, bb, aaa, ...\}$
- $\neg\{ab, ba\}_{a,b,c} = \{\varepsilon, a, b, aa, bb, aaa, ...\} \cup \{\text{Stringhe contenenti almeno un c}\}$

2.1.6 Tipi di linguaggi

Linguaggio Formale

I linguaggi "formali", in senso lato, sono linguaggi in cui l'insieme delle stringhe che li costituiscono è definibile in modo rigoroso e formale.

> *Per linguaggio formale, in matematica, logica, informatica e linguistica, si intende un insieme di stringhe di lunghezza finita costruite sopra un alfabeto finito, cioè sopra un insieme finito di oggetti tendenzialmente semplici che vengono chiamati caratteri, simboli o lettere.*
>
> *Wikipedia IT*

> *In mathematics, computer science, and linguistics, a formal language is a set of strings of symbols together with a set of rules that are specific to it.*
>
> *Wikipedia EN*

Cosa si richiede ad un linguaggio formale?

1. Struttura delle frasi descritta in modo chiaro e comprensibile (sintassi).
2. Possibilità di definire algoritmi di riconoscimento.
3. Possibilità di associare regole per definire il significato delle frasi (semantica).

La semplice notazione insiemistica non è sufficiente. Bisogna ricorrere a formalismi più specifici.

1. Formalismi **generativi** (*grammatiche, espr. regolari*): permettono di capire se una frase appartiene a un dato linguaggio attraverso la descrizione della sua struttura.
2. Formalismi **riconoscitivi** (*automi*): forniscono algoritmi per decidere se una frase appartiene o no al linguaggio.

Entrambi gli approcci sono *duali* ed *equivalenti*, si può infatti passare da un algoritmo di generazione ad uno di riconoscimento in modo meccanico.

Capitolo 3

Automi a stati finiti

3.1 Introduzione informale

Un automa è un modello teorico di un sistema hardware/software utilizzato a scopo di verifica riguardo alla compatibilità di un input in un certo algoritmo. Anche se questa definizione può sembrare molto criptica, vedremo più avanti di approfondire meglio l'argomento.

3.2 Automi a stati finiti deterministici - DFA

Il termine *deterministico* sta a indicare che **per ogni input esiste una sola transizione** verso un'altro stato. Spesso questo tipo di automa è abbreviato con la sigla **DFA** (*Deterministic Finite Automaton*).

3.2.1 Definizione formale

Un DFA A è una quintupla di 5 elementi:

$$A = (Q, \Sigma, \delta, q_0, F)$$

1. Q $=$ è un insieme finito di stati,
2. $\Sigma =$ è l'alfabeto finito di input,
3. $\delta = Q \times \Sigma \to Q$ è la funzione di transizione,

   la quale prende in input uno stato e un simbolo, e restituisce uno stato. Si potrebbe vedere in "pseudocodice alla C" come:

```
status delta(status q, symbol w) {
  status result;
  ...
  return result;
}
```

4. q_0 è lo stato iniziale ($q_0 \in Q$),
5. $F \subseteq Q$ è l'insieme di stati di accettazione.($F \subseteq Q$).

3.2.2 Elaborazione di stringhe di un DFA

La prima cosa che bisogna capire di un DFA è come decide se "accettare" o no una sequenza di simboli in input. Il "linguaggio" di un DFA è l'insieme di tutte le stringhe che esso accetta.

3.2.3 Notazioni differenti

Esistono diversi tipi di notazioni per un DFA, tutte tra loro equivalenti.

1. **Definizione della quintupla e delle funzioni di transizione**
2. **Diagrammi di transizione**
   (a) Per ogni stato in Q, esiste un nodo
   (b) Per ogni stato q in Q e per ogni simbolo di input a in Σ sia $\delta(q, a)$ allora, il diagramma ha un arco dal nodo q al nodo p etichettato a, se vi sono altri simboli che descrivono la stessa transizione da q a p, si aggiungono i rispettivi archi annotati tra i due stati.
   (c) Si aggiunge una freccia etichettata *Start* che entra nel nodo q_0, che indicherà lo stato iniziale dell'automa.
   (d) Gli stati appartenenti a F (cioè quelli finali) sono etichettati da un doppio cerchio.

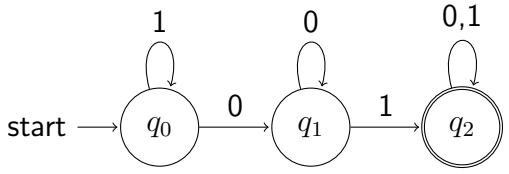


Figura 3.1: Diagramma di transizione per il DFA che accetta tutte le stringhe contenenti 01.

3. **Tabelle di transizione**
   La tabella di transizione è una comune rappresentazione tabellare di una funzione come δ, che ha due argomenti e restituisce un valore. Le righe della tabella corrispondono agli stati, le colonne all'input. All'incrocio della riga si ottiene il risultato dell'operazione δ, con l'input e lo stato nelle rispettive righe/colonne.

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

3.2.4 Funzione di transizione estesa ($\hat{\delta}$)

Abbiamo visto in modo informale che un DFA definisce un linguaggio: l'insieme di tutte le sringhe he producono una sequenza di transizioni di salti dallo stato iniziale a uno stato accettante.Rispetto al diagramma di transizione il linguaggio di un DFA è l'insieme delle etichette lungo i cammini che conducono dallo stato iniziale a un qualunque stato accettante.
A questo punto, per precisare la nozione di linguaggio di un DFA, introduciamo la definizione di **funzione di transizione estesa**, (in notazione $\hat{\delta}$), che descrive cosa succede quando partiamo da uno stato e seguiamo una sequenza di input. Questa funzione di transizione prende uno stato q e una stringa w e restituisce uno stato, ed è definita induttivamente come segue:

BASE $\hat{\delta}(q, \varepsilon) = q$. In altre parole, se ci troviamo in uno stato q e non leggiamo nessun input, allora rimaniamo in q.

INDUZIONE supponiamo che w sia una stringa della forma xa, ossia a è l'ultimo simbolo di w e x è la stringa che contiene tutti i simboli eccetto l'ultimo. Per esempio, $w = 1101$ si scompone in $x = 110$ e $a = 1$ allora, per induzione

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$$

Può sembrare contorta, ma in realtà è molto semplice. Per computare $\hat{\delta}(q, w)$ calcoliamo prima $\hat{\delta}(q, x)$, lo stato in cui si trova l'automa dopo aver elaborato tutti i simboli di w eccetto l'ultimo. Supponiamo ora che questo stato sia p, ossia $\hat{\delta}(q, x) = p$, allora $\hat{\delta}(q, w)$ è quanto si ottiene compiendo una transizione dallo stato p sull'input a, l'ultimo simbolo di w. In altre parole $\hat{\delta}(q, w) = \delta(p, a)$.

3.2.5 Linguaggio di un DFA

Dato un DFA $A = (Q, \Sigma, \delta, q_0, F)$, definiamo il linguaggio di A, in notazione $L(A)$ come

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \text{ è in } F\}$$

Ovvero, l'insieme delle stringhe w che portano dallo stato iniziale $q_=$ a uno degli stati accettanti. Se L è uguale a $L(A)$ per un DFA A, allora diciamo che L è un **linguaggio regolare**.

3.3 Automi a stati finiti non deterministici - NFA

Il termine *non-deterministico* sta a indicare che **per ogni input possono esiste più di una transizione** verso altri stati. Spesso questo tipo di automa è abbreviato con la sigla **NFA** (*Non-deterministic Finite Automaton*).
Infatti, tra gli NFA e i DFA, l'unica cosa che cambia (e lo vedremo dalla definizione formale) è la funzione δ, poiché nel caso dei NFA riceve in input uno stato e un simbolo, ma ha come valore di output un **sottoinsieme di stati**.

3.3.1 Definizione formale

Un NFA A è una quintupla di 5 elementi:

$$A = (Q, \Sigma, \delta, q_0, F)$$

1. Q $=$ è un insieme finito di stati,
2. $\Sigma =$ è l'alfabeto finito di input,
3. $\delta = Q \times \Sigma \to Q$ è la funzione di transizione,

   la quale prende in input uno stato e un simbolo, e restituisce un **sottoinsieme di** Q.
4. q_0 è lo stato iniziale ($q_0 \in Q$),
5. $F \subseteq Q$ è l'insieme di stati di accettazione.($F \subseteq Q$).

3.3.2 Funzione di transizione estesa ($\hat{\delta}$)

Come per i DFA, anche gli NFA hanno una funzione di transizione estesa, ma, dato che questa definizione si basa sulla relativa definizione di δ nell'automa, allora anche $\hat{\delta}$ cambia.
Negli NFA, la funzione di transizione $\hat{\delta}$ prende come argomenti uno stato q e una stringa w e restituisce **l'insieme degli stati in cui si trova l'NFA quanto parte dallo stato stato q e dalla stringa** w. Essa è definita induttivamente come segue:

BASE $\hat{\delta}(q, \varepsilon) = \{q\}$. Se nessun simbolo di input è stato letto, ci troviamo nel solo stato da cui siamo partiti.

INDUZIONE supponiamo che w sia una stringa della forma $w = xa$, dove a è l'ultimo simbolo di w e x è la parte restante. Supponiamo altresì che $\hat{\delta}(q, x) = \{p_1, p_2, ..., p_k\}$. Sia

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, ..., r_m\}$$

allora $\hat{\delta}(q, w) = \{r_1, r_2, ..., r_m\}$. In poche parole, computiamo $\hat{\delta}(q, w)$ calcolando inizialmente $\hat{\delta}(q, w)$ e poi seguendo le transizioni etichettate "a" da tutti questi stati.