

**SEQUENZE**  $\Rightarrow S_0, S_1, \dots, S_n$  "caso"  $S_0 = c, S_1 = i, S_2 = o, S_3 = o$

Per sequenza si intende una collezione finita o infinita di elementi.  
(a volte si suddivide sequenze: finiti, successione: infiniti). Possono essere

**CRESCENTE**  $\Rightarrow$  Dati  $S_0, \dots, S_n$   $S_i < S_{i+1} \forall i$

**DECRESCENTE**  $\Rightarrow$  Dati  $S_0, \dots, S_n$   $S_i > S_{i+1} \forall i$

**NON-DECRESCENTE**  $\Rightarrow$  Dati  $S_0, \dots, S_n$   $S_i \leq S_{i+1} \forall i$

$$S(n) = \frac{1}{n}, \quad S'(n) = \frac{1}{2n} \quad S(n) = \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \dots \quad S'(n) = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$$

**SOTTO-SEQ.**  $\Rightarrow$  Dato una sequenza, la sotto seq. è formata da elem. della sequenza.

N.B. Contiene ordine e ripetizioni.

## STRINGHE

Le stringhe sono once sequenze di caratteri con delle caratteristiche speciali.

**STRINGA NULL**  $\Rightarrow \lambda$  o  $\epsilon$  è la stringa vuota.

**LUNGHEZZA**  $\Rightarrow$  once lunghez.

**CONCATENAZ.**  $\Rightarrow$  Si possono concatenare le stringhe: "ab" + "b" = abb

**SOTTO-STRINGA**  $\Rightarrow \beta$  sott-str  $\Leftrightarrow \exists y = \alpha \beta$

**ALFABETO**  $\Rightarrow \Sigma = \{a, b, c, \dots\}$

Per alfabeto si intende l'insieme di elementi che possono essere usati in un linguaggio. Solitamente sono finiti

**PAROLE/STRINGHE**  $\Rightarrow \Sigma^* = \{aaca, abba, \dots\} \Leftarrow$  finite  $\Sigma^\infty = \{aaca, \dots\} \Leftarrow$  infinite

Le singole parole sono infinite e sono formate a partire dagli elementi dell'alfabeto.

**SOTTO-STRINGA**  $\Rightarrow$  Dato una stringa  $\alpha \in \Sigma^*$ , la sua sotto-stringa è la stringa  $\beta$  tale che  $\alpha = y \beta y$  ovvero è contenuta in essa.

**OCCORRENZA**  $\Rightarrow$  Le occorrenze sono il numero di volte che una sotto-stringa appare in una stringa.

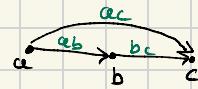
**INVERSA**  $\alpha = aca \quad \alpha^R = eac$   $\Rightarrow$  L'inversa di una stringa è una funzione biettiva contenente la stringa invertita.

RELAZIONI >  $R \subseteq A \times B$

Per relazioni si intendono dei sotto-insiemi rispetto al prodotto cartesiano. Esse possono avere diverse proprietà:

- TRANSITIVA

Se  $(a,b) \in R$  e  $(b,c) \in R$ , allora anche  $(a,c) \in R$  o  
ovvero la composizione.



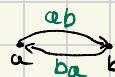
- RIFLESSIVA

Sono riflessive - le relazioni che contengono le relazioni con loro stessi  $\forall a, b \in R \quad (a, a) \in R \wedge (b, b) \in R$



- SIMMETRICA

Le relazioni riflessive sono quelle per cui per ogni coppia, la sua inversa è presente



- ANTI-SIMMETRICA

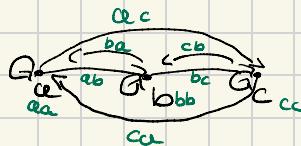
Sono anti-simmetriche le relazioni formate da coppie ordinate per cui l'inversa è diversa dalla relazione di partenza.



RELAZIONI DI EQUIVALENZA >  $\langle A, E \rangle$

Per relazioni di equivalenza si intendono delle relazioni che sono

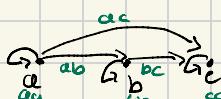
- Riflessive
- Transitiva
- Simmetriche



RELAZIONI D'ORDINE >  $\langle R, \leq \rangle$

Sono relazioni d'ordine le relazioni

- anti-simmetriche
- transitive
- Riflessive



COMPOSIZIONE >

Dette 2 relazioni d'ordine  $R: x \rightarrow y$   $S: y \rightarrow z$

allora la loro composizione  $S \circ R: x \rightarrow z$

è data dagli elementi del primo insieme, legati agli ultimi tramite l'insieme comune

che possono essere

- Totali: Se tutti i suoi elementi
- L'idea: sono comparabili
- Totale:  $(b, c) \notin R$
- Non totale

ESEMPIO:  $R = \{(1,2), (3,4), (5,6)\}$

$$S = \{(2,2), (4,4), (4,8), (6,6)\}$$

$$S \circ R = \{(1,2), (3,2), (3,8), (5,6)\}$$

CLASSI DI EQUIVALENZA >  $a \in X \Rightarrow [a] = \{x \in X \mid xRa\}$

Dato un elemento  $a \in X$ , le sue classi di equivalenza su  $a$  è  $[a]$  è l'insieme di elementi che sono in relazione col dato elemento.  
L'insieme di tutte le classi di equivalenza viene detta partizione

## CIRCUITI >

Tipologie di circuiti

COMBINATORI => Circuiti che NON hanno retro-azione.

SEQUENZIALI => Circuiti che HANNO retro-azioni

## ALGEBRA DEI CIRCUITI >

È possibile definire un'algebra del funzionamento sui circuiti combinatori usando degli operatori:

$$\text{AND} \Rightarrow x \wedge y \Rightarrow$$

$$\text{OR} \Rightarrow x \vee y \Rightarrow$$

$$\text{NOT} \Rightarrow \bar{x} \Rightarrow$$

$$\begin{array}{c|c|c} x & y & x \wedge y \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$$

$$\begin{array}{c|c|c} x & y & x \vee y \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}$$

$$\begin{array}{c|c} x & \bar{x} \\ \hline 0 & 1 \\ 1 & 0 \end{array}$$

### PRIORITÀ

- NOT

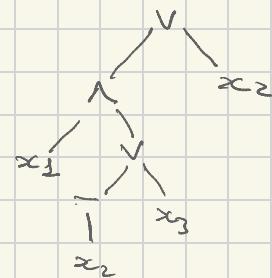
- AND/OR

## CALCOLO -> ALGEBRA >

Per passare dal circuito combinatorio alla sua algebrica bisita guardarlo e convertirlo



$$\text{ESERCIZIO} \rightarrow (x_1 \wedge (\bar{x}_2 \vee x_3)) \vee x_2$$

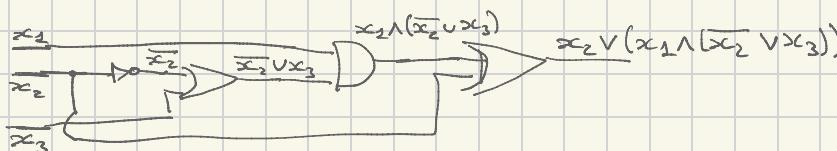


## ALGEBRA -> CIRCUITO >

Per passare dall'algebra al circuito, cominciare sempre prendendo in considerazione prime le

- parentesi
- NOT
- IL RESTO

Se è presente un operatore, dividere il ramo a metà



CIRCUITI EQUIVALENTI  $\Rightarrow$  Due circuiti si dicono equivalenti se hanno le stesse tabelle di verità.

ALGEBRA BOOLEANA  $\Rightarrow$  L'algebra booleana consente di definire diverse operazioni: generali  $\cdot \equiv \wedge$ ,  $+ \equiv \vee$ ,  $' \equiv \neg$

DUALE  $\Rightarrow$

Dato un'espressione in algebra booleana, allora la sua duale è ottenuta facendo

- Scambio di congiunzione e disgiunzione  $+ = \cdot$ ,  $\cdot = +$
- Scambio di 1 e 0  $1=0$ ,  $0=1$

Che conserva lo stesso significato e valenza dell'originale

### FUNZIONI BOOLEANE $\Rightarrow$

Per funzioni booleane si intendono delle funzioni che hanno:

- $Z_2^n$ : Ovvero una n-upla di elementi tra 0,1 IN  $(z_1, z_2, z_3) \rightarrow (z_1 + z_2) \cdot z_3$
- $Z_2$ : Un numero compreso tra 0,1 OUT il 3 di  $Z_2^3$

Dalle funzioni booleane è possibile estrarre la tabella di verità del circuito corrispondente.

TAB VERTÀ  $\rightarrow$  FORMULA  $\rightarrow$  CIRCUITO  $\Rightarrow$

È possibile definire le formule partendo dalla tabella di verità:

- Per ogni riga che ritorna 1
  - Montare le formule corrispondenti
- Mettere insieme le formule in disgiunzione

N.B.: Le singole righe sono chiamate minterm

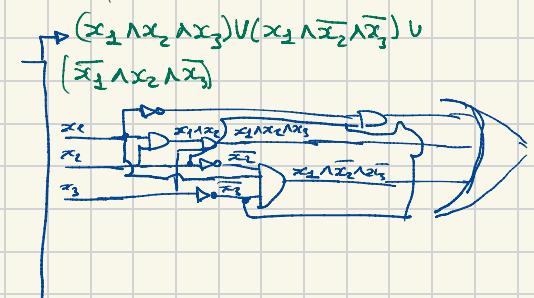
ESEMPPIO  $\Rightarrow$  Dato la seg. Tabella di verità, montare funz. bool e circuito

$$\begin{array}{cccc} x_1 & x_2 & x_3 & f(x_1, x_2, x_3) \\ \hline 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}$$

$$\rightarrow x_1 \wedge x_2 \wedge x_3$$

$$\rightarrow x_1 \wedge \bar{x}_2 \wedge \bar{x}_3$$

$$\rightarrow \bar{x}_1 \wedge x_2 \wedge \bar{x}_3$$



FDI • 22-2023-11-30

DISGIUNZIONE NORMAL FORM  $\Rightarrow f() \rightarrow (\overbrace{x_1 \wedge x_2 \dots}^{\text{minterm}}) \vee (\overbrace{x_1 \wedge x_2 \dots}^{\text{minterm}})$

La disgiuntive Normal Form o DNF è uno dei modi principali di rappresentare i circuiti come disgiunzione di congiunzioni / minterm

MINTERM  $\Rightarrow (x_1 \wedge x_2 \wedge x_3 \dots)$

MAXTERM  $\Rightarrow (x_1 \vee x_2 \vee x_3 \dots)$

CONGIUNZIONE NORMAL FORM  $\Rightarrow$  Matrice alternativa per def. le funzioni booleane come congiunzione di maxterm (forse non in esame!)

GATE  $\Rightarrow$  AND:  $Z_2^2 \rightarrow Z_2$ , NOT:  $Z_2 \rightarrow Z_2$ , OR:  $Z_2^2 \rightarrow Z_2$

Per gate o porte si intende un'operatore che consente di prendere in input 1-2 elementi e ritornare in output 1 elemento.

GATE PUÒ COMPLETARE  $\Rightarrow$

Dato un insieme di porte  $G = \{g_1, g_2, \dots\}$  esso viene detto funzionalmente completo se è possibile mappare qualsiasi combinazione usando le  $g \in G$

Un esempio di base completa è  $G = \{\text{AND}, \text{OR}, \text{NOT}\}$

Inoltre, l'insieme di gate può essere

MINIMO  $\Rightarrow$  Non esistono altri sotto-insiemi del dato insieme di gate completi

il minimo è unico

MINIMALE  $\Rightarrow$  Non ci sono altri elementi sotto di quello

V V  
2 minimali  
possono  
coesistere

ESEMPIO  $\Rightarrow G = \{\text{AND}, \text{NOT}, \text{OR}\}$  non è NE minimo né minimale.

$G' = \{\text{AND}, \text{NOT}\}$ ,  $G'' = \{\text{NOT}, \text{OR}\}$  è minimo perché grazie a De Morgan è possibile descrivere gli AND con OR e vice versa  $\Rightarrow a \wedge b = \bar{a} \vee \bar{b} = a \wedge b = \bar{a} \vee \bar{b}$

$$\overline{a \vee b} = \overline{\bar{a} \wedge \bar{b}} \equiv a \wedge b = \bar{a} \wedge \bar{b}$$

$G''' = \{\text{NAND}\}$  è funz. completa.

$$\text{NAND} = a \uparrow b$$

$$\text{NOR} = a \downarrow b$$

## CIRCUITI ADDIZIONALI &gt;

Per poter costruire l'ALU, è necessario creare dei circuiti dedicati alle sue mansioni di aggiunta, confronto ecc.

Per quanto riguarda l'aggiunta è possibile definire:

## HALF ADDER &gt;

L'Half Adder è un circuito che accetta:

- $Z_2^1$  (0,1) in input, come addendi
- $Z_2^2$  (0,1) come output, contenente:
  - Il risultato dell'addizione
  - Il riporto/carry dell'addizione.



x	y	c	s
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0



x	y	z	c	s
1	1	1	+ 1	0
1	1	0	1	1
1	0	1	1	1
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

## FULL ADDER &gt;

Il Full Adder è una versione modificata del circuito che accetta:

- $Z_2^3$  in input, ovvero
  - I due addendi  $\in \{0,1\}$
  - Il riporto in ingresso
- $Z_2^2$  in output, contenente
  - Il risultato
  - Il riporto

Dalla congiunzione dei circuiti di Full Adder e Half Adder è possibile costruire circuiti in grado di calcolare la somma degli n elementi.

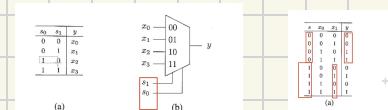
DECODER > 

Per decoder si intende un circuito che consente di prendere 1 input e mapparlo in 1 degli N output. In base al valore degli elementi in W refer. agli output.

## MULTIPLEXER &gt;

Il multiplexer, in maniera inversa, consente di mappare N input in 1 output

In base a cosa è selezionato sui S



a. A 3-bit decoder	Inputs								Outputs							
	x2	x1	x0	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0					
0	0	0	0	0	0	0	0	0	0	0	0					
0	0	0	1	0	0	0	0	0	0	0	1					
0	0	1	0	0	0	0	0	0	0	1	0					
0	0	1	1	0	0	0	0	0	1	1	1					
0	1	0	0	0	0	0	0	1	0	0	0					
0	1	0	1	0	0	0	1	0	0	1	0					
0	1	1	0	0	0	1	0	0	0	0	0					
0	1	1	1	0	0	1	0	1	0	1	1					
1	0	0	0	0	0	0	0	0	0	0	0					
1	0	0	1	0	0	0	0	0	0	1	0					
1	0	1	0	0	0	0	0	1	0	0	0					
1	0	1	1	0	0	0	1	0	0	1	1					
1	1	0	0	0	0	1	0	0	0	0	0					
1	1	0	1	0	0	1	0	0	0	1	0					
1	1	1	0	0	0	1	0	1	0	1	1					
1	1	1	1	0	0	1	0	1	0	1	1					

b. The truth table for a 3-bit decoder

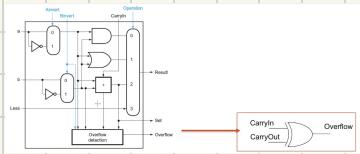
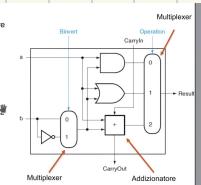


## AU IN PLSC-V

Audendo le operazioni elementari, è possibile definire l'AU come un'unità contenente porte AND, OR, ADDER. Le quali vengono calcolate a fronte degli input:

- I 2 elementi su cui effettuare l'operazione.
- Il selettore di operazione
- Un eventuale carry in.
- Un bi-invert, usato per invertire il valore del 2° input in caso di implementazione di sottrazione
- Un A-invert analogamente in out
- Un Less, contenente come risultato se  $a < b$ , controllato come  $a - b < 0$

- Dati  $a$  e  $b$  è in grado di calcolare le funzioni precedenti con o
- Un ingresso di controllo aggiuntivo (Binvert) seleziona l'operazione desiderata
- Serve, ad esempio, per le sottrazioni ( $a - b$ )
- Ricordare il complemento a  $\frac{1}{2}$
- Con l'uso di
- $\text{CarryIn} = 1$  per il bit meno significativo



Ritirerà come output:

- Il risultato dell'operazione
- Un eventuale carry out (per la somma)
- Un set, usato dall'output come ingresso per il less

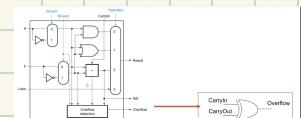
Per scalare è possibile uscire le parti dell'AU a 1bit in cascata

## CLOCK

Il tempo di clock è un circuito che consente di emettere delle onde che servono a sincronizzare le istruzioni dei vari sistemi.

## DELAY

Il delay è un componente che trasmette l'istruzione nell'onda quella successiva.



MACCHINE A STATI FINITI  $\rightarrow M(I, O, S, f, g, \sigma_0)$

Una macchina a stati finiti è un elemento  $M$  composto da:

- Un insieme di input  $I$  finiti
- Un insieme di output  $O$  finiti
- Un insieme di stati  $S$  finiti
- Uno stato di partenza  $\sigma_0 \in S$
- Una funzione di cambio stato  $f: S \times I \rightarrow S$   
che manda le coppie formate da:

- Stato corrente  $\in S$
- Input  $\in I$

In uno stato  $\in S$

- Una funzione di output  $f: S \times I \rightarrow O$

che mappa:

- L'input  $\in I$
- Lo stato corrente  $\in S$

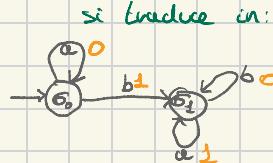
In un elemento in uscita  $\in O$

Pot essere rappresentata con tabella

Example 12.1.5 Let  $I = \{a, b\}$ ,  $O = \{0, 1\}$ , and  $S = \{\sigma_0, \sigma_1\}$ . Define the pair of functions  $f: S \times I \rightarrow S$  and  $g: S \times I \rightarrow O$  by the rules given in Table 12.1.1.

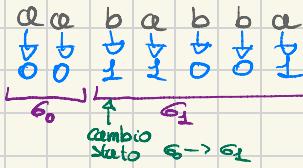
		$f$		$g$	
		$a$	$b$	$a$	$b$
$S$	$I$				
	$\sigma_0$	$\sigma_0$	$\sigma_1$	0	1
$\sigma_1$	$a$	$\sigma_1$	$\sigma_1$	1	0

Then  $M = (I, O, S, f, g, \sigma_0)$  is a finite-state machine.



Da essa è possibile generare diversi output a seconda degli elementi in input.

ESEMPIO DATA MACCHINA SU > "Ottenere output per 'aababbba'"



ESEMPIO 2: FULL ADDER >

È possibile rappresentare un full adder con una macchina a stati

① DEFINIRE I

I saranno tutte le combinaz. sommabili:  $I = \{00, 01, 10, 11\}$

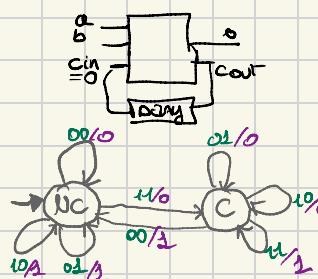
② DEFINIRE O

O sarà il risultato della somma  $O = \{0, 1\}$

③ DEFINIRE S

Gli stati saranno se c'è riporto o no  $S = \{C, NC\}$ .  $\sigma_0 = 0$

per disegno  
con concetti



## AUTOMI A STATI FINITI &gt;

Per automi a stati finiti si intende una categoria di macchine a stati finiti che possiede certe caratteristiche:

- 1° insieme di uscite  $O = \{0, 1\}$
- Lo stato corrente determina l'output. Inoltre:
  - Se dello stato esce come output 0, allora è accettante
  - Se dello stato esce come output 1, allora è non accettante

ESEMPIO > "Disegnare la macchina a stati finiti per la seguente tabella"

S	f		g		
	I	a	b	a	b
s <sub>0</sub>	a <sub>1</sub>	a <sub>0</sub>	1	0	
s <sub>1</sub>	a <sub>1</sub>	a <sub>0</sub>	1	0	
s <sub>2</sub>	a <sub>2</sub>	a <sub>0</sub>	1	0	



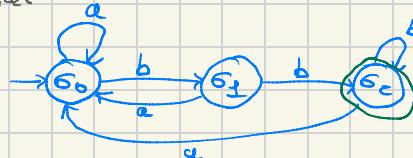
L'automa a stati finiti può quindi essere espresso in forma compatta come  $M(I, S, A, f, \sigma)$  in cui:

- $I \rightarrow$  Insieme di input
- $S \rightarrow$  Insieme di stati
- $A \rightarrow$  Un sotto-insieme di stati contenente gli stati accettanti
- $f: S \times I \rightarrow S \rightarrow$  funzione di transizione
- $\sigma \rightarrow$  Stato iniziale.

ESEMPIO 2 > "Progettare l'automa a partire dalla Tabella".

$$I = \{a, b\}, S = \{s_0, s_1, s_2\}$$

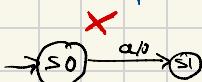
S	f		
	I	a	b
s <sub>0</sub>	a <sub>0</sub>	a <sub>1</sub>	
s <sub>1</sub>	a <sub>0</sub>	a <sub>2</sub>	
s <sub>2</sub>	a <sub>0</sub>	a <sub>2</sub>	



$$A = \{s_2\}, \sigma = s_0$$

NOTA BENE

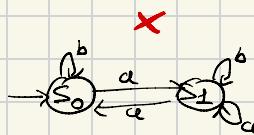
Ogni macchina / automa a stati finiti deve rispettare la propria definizione di funzione per cui:



$$S = \{s_0, s_1, s_2\}, I = \{a, b\}$$

$$O = \{0, 1\}$$

Non è una macchina a stati finiti perché  $f(s_0, b)$  non è mappata  
perché  $f(s_0, b) \neq f(s_1, a)$   
 $f(s_0, b) \neq f(s_2, a)$



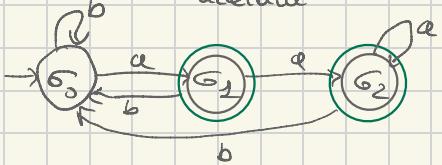
Non è una macchina a stati finiti perché  $f$  non è funzionale ovvero  $f(s_1, a) = \{s_0, s_2\}$  NO!

AUTOMI A STATI FINITI ➤

LINGUAGGI ACCETTATI ➤

Per linguaggio accettato si intende l'insieme di tutte le stringhe per cui l'automa restituisce uno stato accettato.

ESEMPIO ➤ "Data il seguente automa, controllare che la stringa  $w$  sia accettata"

 $w = abaa$ 

AUTOMI EQUIVALENTI ➤ Due automi si dicono equivalenti se a fronte dello stesso input producono lo stesso output

LINGUAGGIO ➤  $\Sigma, \Sigma^* \models L \subseteq \Sigma^*$ 

Dato un alfabeto  $\Sigma$ , dal quale si forma l'insieme infinito delle stringhe finite  $\Sigma^*$ , allora il linguaggio  $L$  è un sotto-insieme di  $\Sigma^*$

GRAMMATICA ➤  $G = (N, T, P, S)$ 

Dato un linguaggio, per grammatica si intende una quadrupla  $G(N, T, P, S)$

- $N$ , È l'alfabeto delle stringhe non terminali, ovvero non accettabili come fine.
- $T$ , È l'alfabeto di stringhe terminali, ovvero accettabili come fine.
- $P$ , Detto insieme di Produzioni, ovvero la relazione tra  $(N \cup T)^*$   $\xrightarrow{P} (N \cup T)^*$

Ovvero che mappa dall'insieme di str. non-terminali  $N$  esclusa stringa vuota all'insieme di stringhe miste fra terminali e non terminali.

NB:  $(N \cup T)^* - T^*$ , oltre a dire che non comprende  $\epsilon$ , vuol anche dire che una volta ottenuta una stringa terminale, ovvero che contenga solo terminali, essa non può più essere mappata nelle produzioni.

~~(ab) < c~~ ~~exa~~  
et

-  $S \rightarrow$  Un simbolo non-terminali di parfenza.

ESEMPIO ➤ "Definire la grammatica per definire i num. binari"

$$N = \{S\}, T = \{0, 1\}, S = S$$

es. 1011

$$P = \{ S \rightarrow 00, S \rightarrow 01, S \rightarrow 1, S \rightarrow 0 \}$$

$$S \rightarrow S \times 2 \quad S \rightarrow 00 \quad S \rightarrow 1$$

$$011 \rightarrow 0011 \rightarrow 1011$$

LINGUAGGIO GENERATO ➤ Per linguaggio generato si intende il linguaggio formato a partire dalla grammatica in cui ogni sua stringa è formata a partire dal simbolo di parfenza della grammatica.