

PROCESSI DI SEMI-THUE

Introdurremo adesso un'altra nozione, quella di processo di semi-Thue che ci servirà successivamente ad introdurre la nozione di grammatica.

Alcuni risultati comuni ai processi di semi-Thue ci permetteranno di dimostrare facilmente il teorema della forma normale di Kleene (che abbiamo già utilizzato per dimostrare l'equivalenza tra la WHILE-calcolabilità e la calcolabilità in \mathcal{S}).

I processi che verranno adesso esaminati riguardano la sostituzione di sottostringhe all'interno di una stringa.

Definizione

Si dice produzione di Semi-Thue (o anche semplicemente produzione o ancora regola di riscrittura) l'espressione

$$g \rightarrow \bar{g}$$

dove g e \bar{g} sono parole su un qualche alfabeto A .

Sia ora P la produzione $g \rightarrow \bar{g}$,

la scrittura $u \xRightarrow{P} v$ indica che

v è ottenuta da u rimpiazzando g con \bar{g} .

Cioè si ha: $u = rgs$ e $v = r\bar{g}s$

con $r, s \in A^*$ (si ammette quindi la possibilità che $r, s = \epsilon$). 1.

Definizione

Si chiama processo di semi-Thue Π un insieme finito di regole di riscrittura P .

Scriveremo $u \underset{\Pi}{\Rightarrow} v$ per indicare che esiste qualche P in Π tale che $u \underset{P}{\Rightarrow} v$.

Scriveremo $u \overset{*}{\underset{\Pi}{\Rightarrow}} v$ per indicare che esiste una sequenza u_1, u_2, \dots, u_n tale che

$$u = u_1 \underset{\Pi}{\Rightarrow} u_2 \underset{\Pi}{\Rightarrow} \dots \underset{\Pi}{\Rightarrow} u_n = v$$

Tale sequenza u_1, u_2, \dots, u_n verrà chiamata derivazione di v da u .

Definizione

La regola di riscrittura $\bar{g} \rightarrow g$ si dice inverso della regola di riscrittura $g \rightarrow \bar{g}$.

Definizione

Un processo di semi-Thue si chiama processo di Thue nel caso in cui se contiene una regola di riscrittura allora contiene anche la sua inversa.

Esempi

1) Sia $\Pi = \{aa \rightarrow b, bb \rightarrow a, abb \rightarrow ba\}$

Partiamo dalla parola abba

Si ha:

$$\underline{abba} \Rightarrow \underline{baa} \Rightarrow bb \Rightarrow a$$

quindi abbiamo che:

$$abba \xRightarrow{*} a$$

e la sequenza delle quattro stringhe

abba, baa, bb, a

è una derivazione di a da abba.

Ma abbiamo ancora che

$$\underline{abba} \Rightarrow \underline{aaa} \Rightarrow ba$$

$$\text{e } \underline{abba} \Rightarrow \underline{aaa} \Rightarrow ab$$

quindi anche ba ed ab possono essere derivate da abba rispettivamente mediante le sequenze

abba, aaa, ba

e abba, aaa, ab.

Mostriamo adesso che è possibile simulare, in un senso che verrà specificato nel seguito, una Macchina di Turing non deterministica mediante dei processi di Semi-Thue.

Sia M una macchina di Turing non deterministica con alfabeto $A = \{s_1, \dots, s_k\}$ e stati q_1, q_2, \dots, q_n .

Vogliamo adesso definire un processo di Semi-Thue $\Sigma(M)$ sull'alfabeto:

$s_0, \overbrace{s_1, \dots, s_k}, \overbrace{q_0, q_1, \dots, q_n}, q_{n+1}, h$

in modo da farli "simulare" M .

Poiché ^{rappresentato} ciascun passo del calcolo di M è completamente dalla sua configurazione per avere tutta questa informazione basta considerare delle opportune codifiche mediante parole sull'alfabeto di $\Sigma(M)$.

Una parola che rappresenta una configurazione di M verrà costruita nel modo seguente:

- Prendiamo la stringa scritta sul nastro della M.d.T M .
- Inseriamo in tale stringa il simbolo che indica lo stato in cui si trova M , immediatamente prima del simbolo esaminato.
- Indichiamo l'inizio e la fine della parola mediante il simbolo appiuntivo h .

Tale stringa verrà chiamata parola di POST.

Se aggiungiamo a destra ed a sinistra della parola scritta nel nastro della MDT dei Blank, rappresentati dal simbolo s_0 e racchiudiamo il tutto sempre tra due simboli h avremo delle differenti parole di Post che però rappresentano la stessa configurazione.

UNA PAROLA DI POST È QUINDI UNA STRINGA $h u q_i v h$ ($0 \leq i \leq n+1$) con $u, v \in \{s_0, s_1, \dots, s_k\}^*$

Non abbiamo ancora detto che ruolo svolge il nuovo stato aggiuntivo. Anticipiamo due (o meglio, indicazione) svolgerà il ruolo di un comando di fine.

Vedremo ciò in modo più preciso mostrando come associare delle regole di riscrittura alle quaduple di M .

1. PER CIASCUNA QUADRUPLA DEL TIPO $q_i s_j s_k q_\ell$ METTIAMO IN $\Sigma(M)$ LA REGOLA DI RISCrittURA:

$$q_i s_j \rightarrow q_\ell s_k$$

osserva s_j stando		osserva s_k stando
nello stato q_i		nello stato q_ℓ

2. PER CIASCUNA QUADRUPLA DEL TIPO $q_i s_j R q_\ell$ METTIAMO IN $\Sigma(M)$ LE REGOLE DI RISCrittURA:

$$q_i s_j s_k \rightarrow s_j q_\ell s_k \quad (\text{si è spostata a destra di un passo passando ad osservare il simbolo } s_k)$$

$$\text{e } q_i s_j h \rightarrow s_j q_\ell s_0 h \quad (\text{deve spostarsi a destra andando a finire sul blank immediatamente a destra della stringa; ricordarsi che } h \text{ indica "fine stringa"})$$

3. Per ciascuna quadrupla del tipo $q_i s_j L q_e$ poniamo in $\Sigma(M)$ le produzioni:

$$s_k q_i s_j \rightarrow q_e s_k s_j$$

$$h q_i s_j \rightarrow h q_e s_0 s_j$$

(stessi commenti del punto 2 con l'inversione $R \rightarrow L$)

4. Ogni volta che $q_i s_j$ ($i=1, \dots, n$; $j=0, 1, \dots, k$) NON sono simboli iniziali di una quadrupla di M , mettiamo in $\Sigma(M)$ la produzione:

$$q_i s_j \rightarrow q_{n+1} s_j$$

5. Mettiamo infine in $\Sigma(M)$ le regole di riscrittura.

$$q_{n+1} s_i \rightarrow q_{n+1} \quad \text{per } i=0, 1, \dots, k$$

$$q_{n+1} h \rightarrow q_0 h$$

$$s_i q_0 \rightarrow q_0 \quad \text{per } i=0, 1, \dots, k.$$

PROPOSIZIONE

Sia M una MdT deterministica e sia w una parola di Post sull'alfabeto di $\Sigma(M)$.

Allora:

1) esiste al più una parola z tale che:

$$w \xRightarrow{\Sigma(M)} z$$

2) se una tale parola z esiste allora z è una parola di Post.

DIMOSTRAZIONE

Prima di passare alla dimostrazione osserviamo che, oltre al suo interesse intrinseco, la presente proposizione mostra come la maniera utilizzata di associare regole di riscrittura a quadruple di MdT "conserva il determinismo".

La dimostrazione, per così dire, si limita ad una verifica che le regole di riscrittura in $\Sigma(M)$ non permettono di ottenere più di una parola (e che questa continua ad essere una parola di Post).

Poiché, per ipotesi, w è una parola di Post si ha che:

$$w = huq_i v h \quad (0 \leq i \leq n+1)$$

Distinguiamo tre casi:

1. q_i è uno stato di M , cioè $\underline{1 \leq i \leq n}$

- se $v = \epsilon$ allora non c'è nessuna regola di riscrittura che si applica perché le regole applicabili per $v = \epsilon$

richiedono $i = 0$ ($s_i q_0 \rightarrow q_0$)

$i = n+1$ ($q_{n+1} h \rightarrow q_0 h$)

- se v inizia col simbolo s_j sono possibili due sottocasi:

- ESISTE UNA QUADRUPLA che inizia con $q_i s_j$ (UNICA perché la Macchina è deterministica) ed allora esiste un'unica regola di riscrittura applicando la quale si ottiene una parola di Post.

- NON ESISTE UNA TALE QUADRUPLA ed allora è applicabile solo la regola di riscrittura

$$q_i s_j \rightarrow q_{n+1} s_j$$

che produce anch'essa, applicata a w , una parola di Post.

2. caso $i = n+1$ ($q_i = q_{n+1}$)

- se $v = \epsilon$

$$w = huq_{n+1}vh$$

è applicabile solo la: $q_{n+1}h \rightarrow q_0h$
che dà una parola di Post.

- se v inizia con s_j l'unica produzione applicabile è

$$q_{n+1}s_j \rightarrow q_{n+1}$$

che dà ancora una parola di Post

$$w = huq_0vh$$

3. caso $i = 0$

(l'unica produzione con q_0 è

$$s_jq_0 \rightarrow q_0$$

e quindi dobbiamo osservare la stringa a sinistra di q)

- se $u = \epsilon$

non è applicabile nessuna regola di riscrittura.

- se u termina con s_j , l'unica regola di riscrittura applicabile produce una parola di Post.

SOMMARIO DELLE REGOLE DI RISCrittURA presenti in $\Sigma(M)$

$$q_i s_j \rightarrow q_e s_k$$

R

$$q_i s_j s_k \rightarrow s_j q_e s_k$$

$$q_i s_j h \rightarrow s_j q_e s_o h$$

L

$$s_k q_i s_j \rightarrow q_e s_k s_j$$

$$h q_i s_j \rightarrow h q_e s_o s_j$$

$$q_i s_j \rightarrow q_{n+1} s_j$$

$$q_{n+1} s_i \rightarrow q_{n+1}$$

$$q_{n+1} h \rightarrow q_o h$$

$$s_i q_o \rightarrow q_o$$

non corrispondono
a quadruple;
le ultime tre
hanno lo scopo
di cancellare
tutti i simboli
tranne h . (e q_o)

PROPOSIZIONE

Sia M una MdT non deterministica.

Allora per ogni stringa u dell'alfabeto di M ,

M accetta u

se e solo se

$$hq_1 s_0 u h \xRightarrow[\Sigma(M)]{*} hq_0 h$$

Dimostrazione

Sia s_1, \dots, s_k l'alfabeto di M .

Supponiamo che M accetti u . Allora se M inizia con la configurazione:

$$\begin{array}{c} s_0 u \\ \uparrow \\ q_1 \end{array}$$

allo fine si troverà in uno stato q_i mentre esamina un simbolo s_k e NON vi è nessuna quadruple di M che inizia con $q_i s_k$.

Vediamo adesso cosa succede, in corrispondenza di ciò, in $\Sigma(M)$:

$$hq_1 s_0 u h \xRightarrow{*} h v \overbrace{q_i s_k} w h \Rightarrow h v q_{n+1} s_k w h \xRightarrow{*}$$

In più passi applicherò tutte le regole di scrittura che corrispondono alle quadruple...

applica la regola di scrittura $q_i s_k \rightarrow q_{n+1} s_k$ (che non corrisponde a quadruple)

$$\xRightarrow{*} h v q_{n+1} h \Rightarrow h v q_0 h \xRightarrow{*} h q_0 h .$$

applica la regola $q_{n+1} s_i \rightarrow q_{n+1}$ fino a cancellare $s_k w$

applica la regola $q_{n+1} h \rightarrow q_0 h$

applica in più passi la regola $s_i q_0 \rightarrow q_0$

Ammettiamo adesso che M non accetti u .

Allora, partendo dalla configurazione

$S_0 u$
 \uparrow
 q_1

, M non si fermerà mai.

Sia ora w_1 la parola di Post che corrisponde alla configurazione iniziale:

$$w_1 = h q_1 S_0 u h$$

Supponiamo adesso che sia:

$$w_1 \xRightarrow{\Sigma(M)} w_2 \xRightarrow{\Sigma(M)} w_3 \xRightarrow{\Sigma(M)} \dots$$

a partire da w_1 , tutte le stringhe successive w_i dovranno contenere un simbolo q_i ($1 \leq i \leq n$) perché avendo sempre delle regole di riscrittura che (simulando il funzionamento della MdT M) fanno passare da stringhe contenenti qualche stato q_i a stringhe contenenti altri stati q_j ($1 \leq i, j \leq n$), cioè non verranno mai attivate le regole di riscrittura che contengono q_{n+1} e q_0 , che per come abbiamo ~~costruito~~ costruito le regole di riscrittura, ~~però~~ entrano in funzione solo dopo che la MdT M corrispondente si è fermata. Non si potrà mai arrivare ad una parola di Post del tipo $h q_0 h$.

Osservazioni sulla dimostrazione precedente

La dimostrazione della proposizione precedente si basa sullo schema seguente:

Vogliamo mostrare che e^-

$$A \iff B$$

dove

$$A = M \text{ accetta } u$$

$$B = \langle q_1 s_0 u \rangle \xrightarrow[\Sigma(M)^*]{*} \langle q_0 \rangle$$

Allora dovremmo mostrare:

$$A \Rightarrow B \quad (1)$$

$$\text{e } B \Rightarrow A \quad (2)$$

In realtà abbiamo mostrato la (1) e

$$\neg A \Rightarrow \neg B \quad (2')$$

ma questo è lecito perché la (2') è equivalente alla (2).

Abbiamo già introdotto il concetto di ^(di riscrittura) regola
inversa di un'altra.

Sia adesso $\Omega(M)$ il processo di
semi-Thue formato dalle inverse di
tutte le regole di riscrittura di $\Sigma(M)$.

Allora possiamo riformulare la
proposizione precedente nel modo
seguente:

Sia M una MDT non deterministica.
Per ogni stringa u nell'alfabeto di
 M , si ha che:

M accetta u se e solo se $h q_0 h \xRightarrow[\Omega(M)]{*} h q_1 s_0 u h$

GRAMMATICHE

Una grammatica a struttura di frase (detta anche, semplicemente, grammatica) è un processo di semi-Thue nel quale le lettere dell'alfabeto sono distinte a seconda delle loro funzioni.

La prima distinzione è tra variabili e simboli terminali.

Inoltre tra le variabili viene ulteriormente distinto un simbolo, il simbolo d'avvio.

Per distinguere graficamente i vari simboli useremo le minuscole per i simboli terminali e le maiuscole per le variabili.

Il simbolo di avvio sarà indicato con S .

Sia T una grammatica con simbolo d'avvio S , V insieme delle variabili e T insieme dei simboli terminali.

Definiamo LINGUAGGIO GENERATO DALLA GRAMMATICA T l'insieme di tutte le stringhe su T che si ottengono a partire dal simbolo S applicando un numero finito di volte le regole di riscrittura di T . (u simboli:

$$L(T) = \{u \in T^* \mid S \xrightarrow[T]{*} u\}$$

Studiamo adesso la relazione che intercorre tra i linguaggi generati da una grammatica e quelli accettati dalle Macchine di Turing.

Abbiamo che :

TEOREMA

Se un linguaggio U è accettato da una MdT non-deterministica allora esiste una grammatica T tale che $U = L(T)$.

Dimostrazione

Sia $U \subseteq T^*$ e sia M una MdT non deterministica che accetta U .

Costruiamo opportunamente una grammatica T .

La MdT M abbia gli stati q_1, \dots, q_n .

L'alfabeto di T consista di $s_0, q_0, q_1, \dots, q_n, q_{n+1}, h$ più le lettere dell'alfabeto di M .

- Prendiamo come simboli terminali di T gli elementi di T ,
- come variabili di T tutti gli altri simboli più i due simboli addizionali S e q .
- S sarà il simbolo di avvio di T .

Le regole di riscrittura di T saranno :

- tutte le regole di $\Omega(M)$
- $S \rightarrow hq_0h$
- $hq_1s_0 \rightarrow q$
- $qs \rightarrow sq$ per ogni $s \in T$
- $qh \rightarrow \emptyset$

Assumiamo adesso che M accetti $u \in T^*$,
 per una delle proposizioni precedentemente dimostrate,
 si ha che deve essere

$$hq_0h \xRightarrow[\Sigma(M)]{*} hq_1s_0uh$$

usando le regole di riscrittura aggiunte
 abbiamo che :

$$S \xRightarrow[\pi]{*} hq_0h \xRightarrow[\pi]{*} hq_1s_0uh \xRightarrow[\pi]{*} quh \xRightarrow[\pi]{*} uqh \Rightarrow u$$

regole
regole
regole
regole
regole

$S \rightarrow hq_0h$
 $h \pi \Sigma(M)$
 $hq_1s_0 \rightarrow q$
 $qs \rightarrow sq$
 $qh \rightarrow \epsilon$

Abbiamo quindi che :

$$S \xRightarrow[\pi]{*} u$$

e possiamo concludere che $u \in L(\pi)$.

Viceversa assumiamo che $u \in L(\pi)$

Allora avremo che : $u \in T^*$ ed \bar{e} :

$$S \xRightarrow[\pi]{*} u$$

Cerchiamo adesso di evidenziare alcuni passi intermedi.

$$S \xRightarrow[\pi]{*} u$$

Sicuramente avremo che

(1) $S \xRightarrow[\pi]{} hq_0h$ (perché $S \rightarrow hq_0h$ è l'unica regola che introduce un simbolo di tipo q , ed anche l'unica applicabile ad S)

(2) $hq_0h \xRightarrow{\pi} vqhz \Rightarrow vz = u$ (perché $qh \rightarrow \epsilon$ è l'unica regola che cancella un simbolo di tipo q)

Combinando la (1) e la (2) - dopo un certo numero di passi - avremo che:

$$S \Rightarrow hq_0h \xRightarrow{*} vqhz \Rightarrow vz = u$$

Osserviamo ancora che q viene introdotto solo dalle regole $hq_1s_0 \rightarrow q$ e che può avvenire $qs \rightarrow sq$.

Quindi avremo che:

$$S \Rightarrow hq_0h \xRightarrow{*} x \overbrace{hq_1s_0}^v y hz \Rightarrow xqyhz \xRightarrow{*} xy \underbrace{qhz}_v \Rightarrow xy \underbrace{z}_v = u$$

\uparrow applicaz. regola $hq_1s_0 \rightarrow q$ \uparrow applicaz. della regola $qs \rightarrow sq$
 che spezza v in due pezzi x ed y .
 ($v = xy$)

Osservazione:

Tutti questi passaggi li abbiamo ricostruito andando a ritroso. Vediamo se ci sono restrizioni da fare.

Dobbiamo poter effettuare la derivazione:

$$hq_0h \xRightarrow{*} xhq_1s_0yh z$$

Questa derivazione, inoltre, deve essere effettuata necessariamente utilizzando regole di riscrittura di $\Omega(M)$ perché le nuove regole aggiunte non sono applicabili.

Poiché sappiamo che le regole di riscrittura di $\Omega(M)$ (con come quelle di $\Sigma(M)$) trasformano parole di Post in parole di Post dobbiamo richiedere che $xhq_1s_0yh z$ sia una parola di Post e quindi che sia $x=z=0$ e $u=y$.

Possiamo allora concludere che

$$hq_0h \xRightarrow[\Omega(M)]{*} hq_1s_0uh$$

In base alla proposizione precedente allora M accetta u .

Osservazioni

Che cosa abbiamo fatto fino ad ora:

- 1) Abbiamo associato alle quadruple di una MdT delle regole di riscrittura che simulano il comportamento della MdT.
- 2) Aggiungendo delle opportune regole di riscrittura ulteriori abbiamo poi ricondotto la parola di Post corrispondente alla configurazione del nastro quando la MdT si è fermata ad una forma standard ($h q_0 h$).

Tutto questo è stato fatto nel contesto dei processi di Semi-Thue.

Ciò che ci farebbe piacere avere, in realtà, è la possibilità di aver prodotte dal processo proprio quelle stringhe che sono accettate dalla MdT. Per potere fare ciò abbiamo bisogno di due cose:

- poter distinguere tra tutti i simboli quelli che appartengono all'alfabeto della MdT.
- poter simulare "al rovescio" il comportamento della MdT.

Il primo punto lo abbiamo realizzato mediante la definizione di grammatica, il secondo mediante l'introduzione delle regole di riscrittura inverse.

Vogliamo stabilire adesso qualche proprietà dei sistemi introdotti.

Per trovare proprietà di tipo "numerico" finiamo rigorosamente l'alfabeto di T e l'ordine dei simboli.

Abbiamo che:

$$\text{Alfabeto di } T: A = \{s_1, \dots, s_n, v_1, \dots, v_k\}$$

dove $\{s_1, \dots, s_n\} = T$ insieme dei simboli terminali

e $\{v_1, \dots, v_k\} = V$ insieme delle variabili

$v_1 = S$ simbolo di avvio

Le stringhe sull'alfabeto A , ordinato come precede, si possono quindi considerare come rappresentazioni di numeri in base $n+k$.

LEMMA: Il predicato $u \Rightarrow_T v$ è ricorsivo primitivo.

Dimostrazione:

Le produzioni di T siano $g_i \rightarrow h_i$ ($i=1, 2, \dots, \ell$)

Per ogni i possiamo scrivere:

$$\text{PROD}_i(u, v) \iff (\exists r, s)_{\leq n} \left[u = \text{CONCAT}(r, g_i, s) \ \& \ v = \text{CONCAT}(r, h_i, s) \right]$$

Poiché sappiamo già che CONCAT è ric. primitivo possiamo concludere che anche PROD_i lo è per ogni i .

Poiché $u \Rightarrow_T v \iff \text{PROD}_1(u, v) \vee \dots \vee \text{PROD}_\ell(u, v)$,

abbiamo dimostrato il lemma.

Introduciamo adesso il predicato $DERIV(u, y)$ per formalizzare la nozione che u è derivabile da S in T mediante la successione u_1, \dots, u_m , avendo che $y = [u_1, \dots, u_m, 1]$.

L'ultimo 1 è stato posto in y per permettere di individuare u_m anche quando $u_m = u = 0$. (si ricordino le codifiche di Gödel)

Ricordiamo che $S = V_1$ si trova al posto $(n+1)$ -esimo e che pertanto questo è il suo valore numerico nella base $n+k$.

Come può esprimersi il predicato $DERIV$?

Diciamo che u è derivabile mediante y se esiste un $m (\leq y)$ tale che la lunghezza di y è $m+1$, il primo termine della sequenza è S (e quindi ha valore $n+1$), l' m -esimo termine è u e qualsiasi indice inferiore ad m

è tale che $u_j \Rightarrow u_{j+1}$, cioè $(y)_j \Rightarrow (y)_{j+1}$ è ottenibile con una regola di riscrittura di T . (oppure è uguale a 0)

Le formule:

$$DERIV(u, y) \Leftrightarrow (\exists m)_{\leq y} (m+1 = Lt(y) \ \& \ (y)_1 = n+1 \ \& \ (y)_m = u \ \& \ (y)_{m+1} = 1 \ \& \ (\forall i)_{< m} \{j=0 \vee [(y)_j \Rightarrow_{T_i} (y)_{j+1}]\})$$

Possiamo subito concludere che $DERIV$ è ricorsivo primitivo perché tutte le operazioni usate lo sono e lo è anche il predicato \Rightarrow , come si è appena mostrato.

(\Rightarrow è immediatamente ricorsivo)

Data la definizione di DERIV possiamo scrivere

$$S \stackrel{*}{\Rightarrow}_{T'} u \iff \exists y \text{ DERIV}(u, y)$$

Ma la relazione precedente ci dice ancora che u è derivabile da S in T' se e solo se la funzione $\min_y \text{DERIV}(u, y)$ è definita:

$$S \stackrel{*}{\Rightarrow}_{T'} u \iff \min_y \text{DERIV}(u, y) \downarrow \quad (*)$$

Ma $\min_y \text{DERIV}(u, y)$ è parzialmente calcolabile (essendo DERIV un predicato calcolabile) e quindi:

la relazione (*) ci dice che $\{u \mid S \stackrel{*}{\Rightarrow}_{T'} u\}$ è r.e.

Possiamo già concludere che $L(T')$ è r.e. NO, ma:

Sappiamo ancora che $L(T') = T^* \cap \{u \mid S \stackrel{*}{\Rightarrow}_{T'} u\}$

per cui possiamo concludere che $L(T')$ essendo intersezione di due linguaggi r.e. è anch'esso r.e.

Quindi:

SE UN LINGUAGGIO L È GENERATO DA UNA GRAMMATICA, ALLORA È RICORSIVAMENTE ENUMERABILE.

Mettiamo adesso insieme alcuni risultati già trovati :

Il linguaggio L è r.e.



Il linguaggio L è accettato da una MdT (determ.)



Il linguaggio L è accettato da una MdT NON-det.



Il linguaggio L è generato da una grammatica



Il linguaggio L è r.e.

Abbiamo chiuso il ciclo e mostrato che tutti gli enunciati precedenti sono equivalenti -

In particolare, notiamo le due equivalenze :

UN LINGUAGGIO È R.E. \iff L È GEN. DA UNA GRAMM.

L È ACC. DA MdT DET. \iff L È ACC. DA MdT NON-DET.