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

$$g \to \bar{g}$$

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

Sia ora P la produzione $g \to \bar{g}$, la scrittura $u \underset{P}{\Rightarrow} v$ 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 $r, s = 0$).

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

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

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

Definizione

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

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 $\Pi = \{aa \to b,\ bb \to a,\ abb \to ba\}$

Partiamo dalla parola abba

Si ha:

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

quindi abbiamo che:

$$abba \overset{*}{\Rightarrow} a$$

e la sequenza delle quattro stringhe

abba, baa, bb, a

è una derivazione di a da abba.

Ma abbiamo ancora che

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

e $abba \Rightarrow \underbrace{aaa} \Rightarrow ab$

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:

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

in modo da fargli "simulare" M.

Poiché ciascun passo del calcolo di M è completamente rappresentato 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. di 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 aggiuntivo h.

Tale stringa verrà chiamata <u>parola di POST</u>.

Se aggiungiamo a destra ed a sinistra della parola scritta sul nastro della MdT dei Blanks, 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

$h\,u\,q_i\,v\,h$ $(0 \le i \le n+1)$ con $u \in v \in \{s_0, s_1, \ldots, s_k\}^*$

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

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

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

$$q_i s_j \to q_\ell s_k$$

osserva s_j stando nello stato q_i ⋮ osserva s_k stando 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:

$q_i s_j s_k \to s_j q_\ell s_k$ (si è spostata a destra di un passo passando ad osservare il simbolo s_k)

e $q_i s_j h \to s_j q_\ell s_0 h$ (deve spostarsi a destra andando a finire sul blank immediatamente a destra della stringa: ricordarsi che h indica "fine stringa")

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

$$s_k q_i s_j \to q_\ell s_k s_j$$

$$h q_i s_j \to h q_\ell s_0 s_j$$

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

4. Ogniqualvolta 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:

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

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

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

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

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

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:

$$w \underset{\Sigma(M)}{\Rightarrow} z$$

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 casi, 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:

$$w = huq_i vh \qquad (0 \leq i \leq n+1)$$

Distinguiamo tre casi

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

   - se $v = 0$ allora non c'è nessuna regola di riscrittura che si applica perché le regole applicabili per $v=0$ richiedono $i = 0$ $(s_i q_0 \to q_0)$
     $i = n+1$ $(q_{n+1} h \to q_0 h)$

   - 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

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

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

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

$w = h u q_{n+1} v h$

- se $v = 0$

è applicabile solo la: $q_{n+1} h \to q_0 h$

che dà una parola di Post.

- se v inizia con s_j l'unica produzione applicabile è $q_{n+1} s_j \to q_{n+1}$

che dà ancora una parola di Post

$w = h u q_0 v h$

3. caso $i = 0$ (l'unica produzione con q_0 è $s_j q_0 \to q_0$ e quindi dobbiamo osservare la stringa a sinistra di q)

- Se $u = 0$

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 $\Sigma(M)$

$q_i s_j \to q_\ell s_k$

R

$q_i s_j s_k \to s_j q_\ell s_k$

$q_i s_j h \to s_j q_\ell s_0 h$

L

$s_k q_i s_j \to q_\ell s_k s_j$

$h q_i s_j \to h q_\ell s_0 s_j$

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

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

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

$s_i q_0 \to q_0$

non corrispondono a quadruple; le ultime tre hanno lo scopo di cancellare tutti i simboli tranne h (e q_0)

PROPOSIZIONE

Sia M una MdT non deterministica.
Allora per ogni stringa u dell'alfabeto di M,

M accetta u se e solo se $h q_1 s_0 u h \overset{*}{\underset{\Sigma(M)}{\Rightarrow}} h q_0 h$

Dimostrazione

Sia $s_1, \dots, s_k$ l'alfabeto di M.
Supponiamo che M accetti u. Allora se M inizia con la configurazione:

$$\begin{matrix} s_0 u \\ \uparrow \\ q_1 \end{matrix}$$

alla fine si troverà in uno stato q_i mentre esamina un simbolo s_k e NON vi è nessuna quadrupla di M che inizia con $q_i s_k$.
Vediamo adesso cosa succede, in corrispondenza di ciò, in $\Sigma(M)$:

$$h q_1 s_0 u h \overset{*}{\Rightarrow} h v \overbrace{q_i s_k} w h \Rightarrow h v q_{n+1} s_k w h \overset{*}{\Rightarrow}$$

In più passi applicherà tutte le regole di riscrittura che corrispondono alle quadruple...

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

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

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

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

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

Ammettiamo adesso che M non accetti u.

Allora, partendo dalla configurazione

$$\begin{matrix} s_0 u \\ \uparrow \\ q_1 \end{matrix}$$

, 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:

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

a partire da w_1, tutte le stringhe successive w_j dovranno contenere un simbolo q_i $(1 \le i \le n)$ perché avremo 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 $(1 \le i, j \le n)$, cioè non verranno mai attivate le regole di riscrittura che contengono q_{n+1} e q_0, che per come abbiamo costruito le regole di riscrittura, 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 è

$$A \iff B$$

dove

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

$$B = h q_1 s_0 u h \overset{*}{\underset{\Sigma(M)}{\Longrightarrow}} h q_0 h$$

Allora dovremmo mostrare:

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

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

In realtà abbiamo mostrato la (1) e

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

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

Abbiamo già introdotto il concetto di (di riscrittura) regola inverso 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 $h q_0 h \overset{*}{\underset{\Omega(M)}{\Longrightarrow}} h q_1 s_0 u h$.

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:

$$L(T) = \{u \in T^* \mid S \overset{*}{\underset{T}{\Longrightarrow}} u\}$$

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' coincide 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 $\Omega(M)$
- $S \to hq_0h$
- $hq_1S_0 \to q$
- $qs \to sq$ per ogni $s \in T$
- $qh \to 0$

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

$$h q_0 h \overset{*}{\underset{\Sigma(M)}{\Longrightarrow}} h q_1 s_0 u h$$

usando le regole di riscrittura aggiunte abbiamo che:

$$S \underset{T'}{\Longrightarrow} h q_0 h \overset{*}{\underset{T'}{\Longrightarrow}} h q_1 s_0 u h \underset{T'}{\Longrightarrow} q u h \overset{*}{\underset{T'}{\Longrightarrow}} u q h \Rightarrow u$$

- $S \to h q_0 h$: regola
- per $\Sigma(M)$
- $h q_1 s_0 \to q$: regola
- $q s \to s q$: regola
- $qh \to \circ$: regola

Abbiamo quindi che:

$$S \overset{*}{\underset{T'}{\Longrightarrow}} u$$

e possiamo concludere che $u \in L(T')$.

Viceversa ammettiamo che sia $u \in L(T')$

Allora avremo che: $u \in T^*$ ed è:

$$S \overset{*}{\underset{T'}{\Longrightarrow}} u$$

Cerchiamo adesso di evidenziare alcuni passi intermedi.

$$S \overset{*}{\underset{T}{\Longrightarrow}} u$$

Sicuramente avremo che

(1) $S \underset{T}{\Longrightarrow} hq_0h$ (perché $S \to hq_0h$ è l'unica regola che introduce un simbolo di tipo q, ed anche l'unica applicabile ad S)

(2) $vqhz \Longrightarrow vz = u$ (perché $qh \to 0$ 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:

$$S \Longrightarrow hq_0h \overset{*}{\Longrightarrow} vqhz \Longrightarrow vz = u$$

Osserviamo ancora che q viene introdotto solo dalla regola $hq_1s_0 \to q$ e che può aversi $qs \to sq$.

Quindi avremo che:

$$S \Longrightarrow hq_0h \overset{*}{\Longrightarrow} x\overbrace{hq_1s_0}yhz \Longrightarrow xqyhz \overset{*}{\Longrightarrow} xyqhz \Longrightarrow xyz = u$$

v

↑ applicaz. regola $hq_1s_0 \to q$

↑ applicazione della regola $qs \to sq$ 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:

$$hq_0h \overset{*}{\Rightarrow} xhq_1s_0yhz$$

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 $\Omega(M)$ (così come quelle di $\Sigma(M)$) trasformano parole di Post in parole di Post dobbiamo richiedere che xhq_1s_0yhz sia una parola di Post e quindi che sia $x = z = 0$ e $u = y$

Possiamo allora concludere che

$$hq_0h \underset{\Omega(M)}{\overset{*}{\Rightarrow}} hq_1s_0uh$$

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 (hq_0h).

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" fissiamo rigorosamente l'alfabeto di T' e l'ordine dei simboli.

Abbiamo che:

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 prima, si possono quindi considerare come rappresentazioni di numeri in base $n+k$.

LEMMA: Il predicato $u \underset{T'}{\Rightarrow} v$ è ricorsivo primitivo.

Dimostrazione:
Le produzioni di T' siano $g_i \to h_i$ $(i = 1, 2, \dots, l)$
Per ogni i possiamo scrivere:

$$PROD_i(u,v) \Leftrightarrow (\exists r,s)_{\leq u} \left[u = CONCAT_{n+k}(r, g_i, s) \;\&\; v = CONCAT_{n+k}(r, h_i, s) \right]$$

Poiché sappiamo già che CONCAT è ric. primitiva possiamo concludere che anche $PROD_i$ lo è per ogni i.
Poiché $u \underset{T'}{\Rightarrow} v \Leftrightarrow PROD_1(u,v) \vee \dots \vee PROD_l(u,v)$,
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 l'$(m+1)$esimo è 1 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 risoluzione di T. (oppure è uguale a 0)

Le formule:

$$\text{DERIV}(u, y) \Leftrightarrow (\exists m)_{\leq y} \Big(m+1 = Lt(y) \;\&\; (y)_1 = n+1 \;\&\; (y)_m = u \;\&\; (y)_{m+1} = 1 \;\&\; (\forall j)_{<m} \big\{ j = 0 \vee [(y)_j \underset{T}{\Rightarrow} (y)_{j+1}] \big\} \Big)$$

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 decidibile da ...)

Data la definizione di DERIV possiamo scrivere

$$S \overset{*}{\underset{T}{\Rightarrow}} u \iff \exists y \; DERIV(u, y)$$

Ma la relazione precedente ci dice ancora che u è derivabile da S in T se e solo se la funzione $\min_y DERIV(u,y)$ è definita:

$$S \overset{*}{\underset{T}{\Rightarrow}} u \iff \min_y DERIV(u,y) \downarrow \qquad (*)$$

Ma $\min_y DERIV(u,y)$ è parzialmente calcolabile (essendo DERIV un predicato calcolabile) e quindi la relazione $(*)$ ci dice che $\{u \mid S \overset{*}{\underset{T}{\Rightarrow}} u\}$ è r.e.

Possiamo già concludere che $L(T)$ è r.e. NO, ma:

Sappiamo ancora che $L(T) = T^* \cap \{u \mid S \overset{*}{\underset{T}{\Rightarrow}} u\}$

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 L È R.E. ⟺ L È GEN. DA UNA GRAMM.

L È ACC. DA MdT DET. ⟺ L È ACC. DA MdT NON-DET.

Il prossimo risultato è più importante di quanto la semplicità della sua dimostrazione possa far pensare.
Tale semplicità è dovuta all'uso combinato di teoria della ricorsività e teoria delle grammatiche. Una dimostrazione puramente "ricorsiva" sarebbe stata più complessa.

TEOREMA
Sia U un insieme r.e. di numeri. Allora esiste un predicato ricorsivo primitivo $R(x,t)$ tale che

$$U = \{x \mid \exists t\ R(x,t)\}$$

Dimostrazione

Guardiamo ad U come ad un linguaggio r.e. su $\{s_1\}$
Sappiamo che U è r.e. se e solo se esiste una gramm. T' tale che $U = L(T')$.
Deve pertanto esistere una grammatica T' con il singolo simbolo terminale s_1 che genera U.
L'alfabeto di T' sia $s_1, V_1, \ldots, V_k$.
La stringa che rappresenta x in base 1, $s_1^{[x]}$, rappresenta, in base $k+1$, il numero $f(x)$ definito da:

$$f(x) = \begin{cases} 0 & \text{se } x = 0 \\ \sum_{i=0}^{x-1} (k+1)^i & \text{altrimenti.} \end{cases}$$

$f(x)$ è quindi ricorsivo primitivo.
Ora $x \in U$ se esiste una sua deriv. in T', cioè:

$$x \in U \iff (\exists y)\, DERIV(f(x), y) \qquad R'(x,y)$$

($f(x)$: suo "nome" numerico in T')

Ma allora il teorema è dimostrato perché DERIV$(f, \ )$ come composizione di "oggetti" ricorsivi primitivi è anch'esso ricorsivo primitivo.

TEOREMA.

Sia S un insieme non vuoto r.e. Allora esiste una funzione ricorsiva primitiva $f(x)$ tale che:

$$S = \{ f(n) \mid n \in \mathbb{N} \} = \{ f(0), f(1), \dots \}$$

Cioè, S è il codominio di f.

Dimostrazione.

Per il teorema precedente $S = \{ x \mid \exists t \; R(x,t) \}$ (*) dove R è un predicato ricorsivo primitivo.

Sia adesso x_0 un elemento fissato di S, per esempio, il più piccolo.

Poniamo

$$f(u) = \begin{cases} l(u) & \text{se } R(l(u), r(u)) \\ x_0 & \text{altrimenti.} \end{cases}$$

La funzione f è ricorsiva primitiva.

- Mostriamo adesso che ogni valore di $f(u)$ è in S.
  - x_0 appartiene ad S per definizione.
  - il valore $l(u)$ viene assunto da f quando esiste un t, in questo caso $r(u)$, che si trova nella relazione R con $l(u)$, ma in questo caso, per la relazione (*), $l(u) \in S$.
- Mostriamo adesso l'inverso. Se $x \in S$, allora $R(x, t_0)$ è vera per qualche t_0.
  Calcoliamo $f(\langle x, t_0 \rangle)$, dove $\langle x, t_0 \rangle$ è la solita codifica
  Si ha: $f(\langle x, t_0 \rangle) = l(\langle x, t_0 \rangle) = x$
  Abbiamo allora effettivamente che se un elemento x appartiene ad S allora è il valore assunto da f quando è calcolata in $\langle x, t_0 \rangle$. Ed il teorema è dimostrato.

TEOREMA

Sia $f(x)$ una funzione parzialmente calcolabile e sia $S = \{f(x) \mid f(x)\downarrow\}$, allora S è r.e.

(Cioè se S è il codominio di una funzione part. calcolabile allora S è ricorsiv. enumerabile).

Dimostrazione.

Poniamo:

$$g(x) = \begin{cases} 0 & \text{se } x \in S \\ \text{non definita} & \text{altrimenti} \end{cases}$$

Allora $S = \{x \mid g(x) \text{ è definita}\}$ (*)

Ma la (*), nel caso in cui la g è parzialmente calcolabile è proprio la definizione di insieme r.e.

Per dimostrare il teorema è quindi sufficiente mostrare che g è parzialmente calcolabile.

Sia, allora, P un programma che calcola f (esiste per l'ipotesi che f è part. calcolabile) e sia $\#(P) = p$.

g è calcolata dal programma che segue:

1. [A] IF $\nu STP^{(1)}(Z, p, T)$ GOTO B
2. V ← f(Z)
3. IF V = X GOTO E
4. [B] Z ← Z + 1
5. IF Z ≤ T GOTO A
6. T ← T + 1
7. Z ← 0
8. GOTO A

- L'istruzione N° 1 controlla se il programma N° p (cioè quello che calcola f) si ferma dopo non più di T passi sulla variabile Z.
- Se non si ferma allora viene rinviato all'istruzione 4 che aumenta di una unità la variabile Z e nel caso in cui Z è minore o eguale a T rinvia ad A
- Se Z > T allora viene aumentato il valore di T di una unità, riazzerata la variabile Z ed il processo ricomincia.

È CHIARO CHE LO SCOPO CHE SI PREFIGGE TUTTO QUESTO MARCHINGEGNO È SOLO QUELLO DI FARE ATTIVARE LA MACRO $V \leftarrow f(Z)$ SOLO QUANDO LA $f(Z)$ È DEFINITA.

A QUESTO PUNTO IL NUCLEO DEL PROGRAMMA

$V \leftarrow f(Z)$
IF $V = X$ GOTO E

SEMPLICEMENTE CONFRONTA IL VALORE $f(Z)$ CON QUELLO DI X E SE I VALORI SONO EGUALI IL PROGRAMMA SI FERMA.
LA VARIABILE DI USCITA Y SU CUI NON SI È INTERVENUTI DARÀ IL VALORE 0.

I due teoremi precedenti ci permettono di mostrare l'equivalenza tra gli enunciati che seguono:

L'INSIEME S È RICORSIVAMENTE ENUMERABILE

⇓ per il penultimo teorema dimostrato

S È IL CODOMINIO DI UNA FUNZIONE RICORSIVA PRIMITIVA

⇓ ovvio

S È IL CODOMINIO DI UNA FUNZIONE RICORSIVA

⇓ ovvio

S È IL CODOMINIO DI UNA FUNZIONE RICORSIVA PARZIALE

⇓ per l'ultimo teorema dimostrato.

S È RICORSIVAMENTE ENUMERABILE

Gli enunciati 2, 3, 4 sono quindi caratterizzazioni degli insiemi ricorsivamente enumerabili.

Presentiamo ancora una proposizione la cui dimostrazione è immediata e che useremo successivamente.

Proposizione

Sia $g(x)$ una funzione parzialmente calcolabile e sia $S = \{ \langle x, y \rangle \mid y = g(x) \}$ allora S è r.e.

Dimostrazione

S non è altro che l'insieme dei valori assunti dalla funzione di codifica "coppia" $\langle , \rangle$ calcolata su $(x, g(x))$.

Chiamiamo f tale funzione : $f(x) = \langle x, g(x) \rangle$

Abbiamo allora che

$$S = \{ f(x) \mid x \in N \}$$

con f parzialmente calcolabile (perché lo è sia $\langle , \rangle$ che g).

In base ad uno dei teoremi precedenti S è, allora, ricorsivamente enumerabile.

TEOREMA (della forma normale di Kleene per funz. di una var.)
Sia f una funzione parzialmente calcolabile di una variabile, allora esiste un predicato ricorsivo primitivo $R(x, y)$ tale che:

$$f(x) = l(\min_y R(x, y))$$

Dimostrazione

Sia $$S = \{\langle x, y\rangle \mid y = f(x)\} \qquad (*)$$

Sappiamo già che se $f(x)$ è parzialmente calcolabile allora un insieme S della forma $(*)$ è ricorsivamente enumerabile.
Ancora, sappiamo che un insieme r.e. si può sempre esprimere come:

$$S = \{x \mid \exists t\ Q(x, t)\}$$

con Q predicato ricorsivo primitivo.

Nel nostro caso si ha quindi che:

$$y = f(x) \iff \langle x, y\rangle \in S \iff (\exists t) Q(\langle x, y\rangle, t).$$

Vogliamo adesso mostrare che:

$$f(x) = l\left(\min_y Q(\langle x, l(y)\rangle, r(y))\right)$$

Supponiamo che $\min_y Q(\langle u, \ell(y)\rangle, r(y))$ sia definito e sia:

$$y_0 = \min_y Q(\langle u, \ell(y)\rangle, r(y))$$

ponendo $v = \ell(y_0)$

$t = r(y_0)$

possiamo scrivere

$Q(\langle u, v\rangle, t)$

Ricordiamo che, per definizione,

(*) $\quad y = f(u) \iff \exists t\ Q(\langle u, y\rangle, t)$

nel nostro caso si ha che $\exists t, t = r(y_0)$ tale che $Q(\langle u, y\rangle, t)$ con $y = v = \ell(y_0)$, quindi si ha che:

$$f(u) = \ell(y_0) = \ell\left(\min_y Q(\langle u, \ell(y)\rangle, r(y))\right)$$

Supponiamo adesso che $\min_y Q(\langle u, \ell(y)\rangle, r(y))$ non sia definito.

Questo vuol dire che $Q(\langle u, v\rangle, t)$ non è mai verificata per nessun v, t ed allora non esiste nessun valore v tale che $f(u) = v$ e quindi $f(u)$ non è definito (si usa ancora una volta la definizione (*)).

Abbiamo così dimostrato il teorema -

Vogliamo adesso estendere una rappresentazione come quella precedente a funzioni parzialmente calcolabili di n variabili.

È FACILE PREVEDERE CHE IL MECCANISMO CHE USEREMO SARÀ UN TRUCCO DI CODIFICA CHE CI PERMETTERÀ DI LEGGERE LA FUNZIONE DI PIÙ VARIABILI COME FUNZIONE DI UNA SOLA VARIABILE. DOPO DI CHE POTREMO APPLICARE IMMEDIATAMENTE IL TEOREMA PRECEDENTE.

TEOREMA (DELLA FORMA NORMALE DI KLEENE)

Sia $f(x_1,\dots,x_n)$ una funzione parzialmente calcolabile di n variabili.
Allora esiste un predicato ricorsivo primitivo $R(x_1,\dots,x_n,y)$, di $n+1$ variabili tale che:

$$f(x_1,\dots,x_n) = \ell\left(\min_y R(x_1,\dots,x_n,y)\right)$$

Dimostrazione.

Poniamo $x = [x_1,\dots,x_n]$ (codifica di Gödel)

e $g(x) = f((x)_1,\dots,(x)_n)$

g è, ovviamente, parzialmente calcolabile per cui si ha che:

$$g(x) = \ell\left(\min_y Q(x,y)\right)$$ (applicazione del teorema precedente)

Ma la relazione precedente si può scrivere come:

$$f(x_1,\dots,x_n) = g([x_1,\dots,x_n]) = \ell\left(\min_y Q([x_1,\dots,x_n],y)\right)$$

ed il teorema è dimostrato.

Possiamo adesso trarre alcune conclusioni:

1. LA DIMOSTRAZIONE DELL'EQUIVALENZA TRA IL LINGUAGGIO $\mathcal{S}$ ED IL LINGUAGGIO W È COSÌ COMPLETATA
(e, di conseguenza, quella tra W e tutte le altre versioni della calcolabilità equivalenti ad $\mathcal{S}$).

2. È ANCHE DIMOSTRATA L'EQUIVALENZA TRA LE FUNZIONI μ-ricorsive E LE FUNZIONI CALCOLABILI IN $\mathcal{S}$.

Infatti, dato il teorema della forma normale, possiamo concludere che una funzione parzialmente calcolabile può essere ottenuta dalle funzioni iniziali mediante un numero finito di applicazioni delle operazioni di composizione, ricorsione e minimalizzazione (quest'ultima, a rigore, basta <u>una</u> <u>sola</u> volta)

(Avevamo già mostrato, a suo tempo, la calcolabilità delle funzioni ricorsive primitive e dell'operatore di minimalizzazione -)

Ricordiamo qui brevemente come si mostra la calcolabilità dell'operatore di minimalizzazione:

Sia $P(x_1, \ldots, x_n, y)$ un predicato calcolabile

Allora $g(x_1, \ldots, x_n) = \min_y P(x_1, \ldots, x_n, y)$

è una funzione parzialmente calcolabile.

g, infatti, è calcolata dal programma:

```
[A]  IF  P(X1, ..., Xn, Y)  GOTO E
     Y ← Y + 1
     GOTO  A
```

Funzioni μ-ricorsive:

- Funzioni iniziali
- composizione
- ricorsione
- minimalizzazione NON LIMITATA

} Ricorrive prim.

Il Teorema della forma normale di Kleene ci permette di dire immediatamente che le funzioni calcolabili sono μ-ricorrive.

Il viceversa è conseguenza della calcolabilità delle funzioni iniziali e delle operazioni di chiusura.

QUINDI

la classe delle funzioni μ-ricorsive parziali coincide con la classe delle funzioni parzialmente calcolabili in $\mathcal{S}$.