# Teoremi di Gödel

Alessandro Berarducci

Versione molto preliminare, solo sulla parte tecnica, 1 giugno 2001, perdonate gli errori ...

## 1 Algoritmi e funzioni calcolabili

### 1.1 Funzioni calcolabili

Una funzione $f$ si dice *calcolabile* (o "algoritmicamente calcolabile") se esiste un *algoritmo* che dato in ingresso $x \in dom(f)$ fornisce in uscita il valore $y = f(x)$.

Questa definizione ha bisogno di varie precisazioni. Innanzitutto la nozione di algoritmo è data per intuitivamente nota e non verrà definita. L'idea è che un algoritmo è un insieme finito di istruzioni che possono essere eseguite in modo del tutto meccanico senza che sia richiesta alcuna "creatività" da parte di chi esegue le istruzioni (che quindi può essere una macchina).

Poichè un algoritmo deve necessariamente operare su *rappresentazioni finite*, affinchè la definizione abbia senso assumiamo gli elementi del dominio e codominio di $f$ siano rappresentati da stringhe finite di simboli presi da un alfabeto finito $\Sigma$. Questo implica che il dominio e il codominio di una funzione calcolabile è necessariamente un insieme finito o numerabile. Esempi di domini e codomini ammissibili sono i numeri naturali $\mathbf{N}$ con la usuale rappresentazione in notazione decimale[1], oppure i razionali $\mathbf{Q}$ dove $q \in \mathbf{Q}$ è rappresentato dando il numeratore e denominatore della frazione ridotta ai minimi termini che corrisponde a $q$ [2]. Non considereremo invece funzioni su numeri reali in quanto non è possibile fissare una rappresentazione finita

[1] Un'altra notazione spesso usata in queste note è quella di rappresentare $n \in \mathbf{N}$ con l'espressione $S(S(\ldots S(0)))$ dove il simbolo $S$ (successore) è ripetuto $n$ volte.

[2] Ovviamente non è necessario che il dominio e codominio siano lo stesso insieme: possiamo ad esempio considerare funzioni da $\mathbf{N}^5$ ad $\mathbf{Q}$.

per i numeri reali[3].

A priori potrebbe capitare che una funzione sia calcolabile rispetto ad una data rappresentazione degli elementi del dominio e codominio e non calcolabile rispetto ad un'altra rappresentazione. Se però consideriamo due rappresentazioni tali che esiste un algoritmo per passare dall'una all'altra, ad esempio la rappresentazione in base decimale e quella in base due per i numeri naturali, allora è chiaro che una qualsiasi funzione è calcolabile rispetto alla prima rappresentazione se e solo se lo è rispetto alla seconda[4].

In genere tutte le rappresentazioni comunemente considerate hanno la proprietà che si può passare in modo algoritmico dall'una all'altra rappresentazione. Possiamo quindi parlare di funzioni calcolabili lasciando implicita la scelta della particolare rappresentazione usata.

Esempi classici di funzioni calcolabili sono i seguenti:

1) La funzione di divisione con resto sui numeri naturali (possiamo usare l'algoritmo delle sottrazioni successive).

2) La funzione da $\mathbf{N}^2$ a $\mathbf{N}$ che dati due numeri $x$ e $y$ fornisce il massimo comun divisore di $x$ ed $y$ (usiamo l'algoritmo di Euclide).

3) La funzione da $\mathbf{N}$ all'insieme di due elementi $\{SI, NO\}$ che dato un numero $x \in \mathbf{N}$ stabilisce se $x$ è primo (dato $x$ calcoliamo il resto della divisione di $x$ per tutti i numeri maggiori di uno e più piccoli di $x$ e diamo risposta $SI$ se tutti i resti sono diversi da zero).

4) La funzione da $\mathbf{N}$ ad $\mathbf{N}^{<\omega}$ (sequenze finite di elementi di $\mathbf{N}$) che dato un numero fornisce la sua scomposizione in fattori primi.

Chiaramente può esistere più di un algoritmo per calcolare la stessa funzione, alcuni più efficienti altri meno [5].

[3] I reali possono essere rappresentati tramite i loro sviluppi decimali, ma in generale sono necessarie infinite cifre dopo la virgola, e quindi non si tratta di una rappresentazione finita. Sono stati fatti vari tentativi di estendere il concetto di funzione calcolabile ai numeri reali, ma non entreremo in questo argomento.

[4] Ad esempio se ho un algoritmo per scomporre in fattori primi in numero naturale scritto in base due, posso facilmente ottenere un secondo algoritmo per scomporre in fattori primi un numero scritto in base dieci. Il secondo algoritmo prende in ingresso un numero in base 10, ne trova la rappresentazione binaria, applica il primo algoritmo per scomporlo, e infine riscrive il risultato in base 10.

[5] La teoria della complessità si occupa del trovare algoritmi efficienti. La teoria della ricorsività, di cui ci occuperemo, si interessa solo all'esistenza di algoritmi indipendentemente dalla loro efficienza.

## 1.2 Funzioni calcolabili parziali

Alcuni algoritmi potrebbero non arrestarsi mai per alcuni valori in ingresso. Consideriamo il seguente esempio.

**Esempio 1.1** E' facile progettare un algoritmo che, ricevendo in ingresso una equazione polinomiale a coefficienti interi, passi in rassegna uno dopo l'altro tutti i possibili valori interi delle variabili arrestandosi qualora venga trovata una soluzione dell'equazione e fornendo in uscita la soluzione. Se l'equazione non ha soluzione, l'algoritmo continuerà la ricerca senza arrestarsi mai.

Sarebbe bello progettare un algoritmo che non solo trovi una soluzione quando essa esiste, ma sia anche in grado di determinare quando essa non esiste anziché andare avanti a cercarla senza fine. Ciò però non è possibile. Nel 1970 Y. Matijaseviç ha infatti dimostrato (assumendo la tesi di Church) che non esiste un metodo algoritmico per stabilire se una equazione polinomiale a coefficienti interi (in più variabili) ammetta una soluzione in numeri interi.

Diamo ora un esempio di un algoritmo di cui non è noto se la sua esecuzione termini per ogni valore in ingresso.

**Esempio 1.2** Due numeri primi $p, q \in \mathbf{N}$ si dicono *gemelli* se $q - p = 2$. Mentre si sa che esistono infiniti numeri primi, non è noto se esistano infiniti numeri primi gemelli.

È facile costruire un algoritmo che dato $x \in \mathbf{N}$, trova, se esistono, due numeri primi gemelli $p, q \in N$ maggiori di $x$. L'algoritmo enumera uno dopo l'altro tutti i numeri dispari maggiori di $x$ finchè trova, se esistono, due numeri dispari consecutivi che sono entrambi primi.

Se esistono infiniti numeri primi gemelli (cosa che a tutt'oggi è ignota) l'algoritmo si ferma sempre. Altrimenti si fermerà solo per valori $x$ in ingresso non più grandi del più grande primo gemello.

Ad un algoritmo che non termina per alcuni valori in ingresso risulta naturale associare una "funzione parziale" nel senso seguente.

Una *funzione parziale* da $A$ a $B$ è una funzione da $A$ a $B \cup \{\perp\}$ (unione disgiunta), dove $f(a) = \perp$ significa che $f$ è indefinita su $a$.

Scriviamo $f\colon A \rightsquigarrow B$ per indicare che $f$ è una funzione parziale da $A$ a $B$, cioè $f\colon A \to B \cup \{\perp\}$. Per funzioni parziali $f\colon A \rightsquigarrow B$ esiste una distinzione tra l'insieme $A$ dei possibili argomenti, e il dominio $dom(f) \subseteq A$ consistente

di quegli argomenti $a \in A$ tali che $f(n) \neq \perp$. Ammettiamo come casi limite di funzioni parziali le funzioni totali, quelle cioè per cui $dom(f) = A$. Un altro caso limite di funzione parziale è la funzione indefinita per ogni input.

Possiamo associare ad ogni algoritmo $M$ una funzione parziale $f$ nel modo seguente. Supponiamo che l'algoritmo $M$ prenda in ingresso elementi di un insieme $A$ (in una fissata rappresentazione) e fornisca in uscita, nei casi in cui la computazione termina dopo un numero finito di passi, elementi di un insieme $B$ (in una fissata rappresentazione).

La funzione calcolata da $M$ è la funzione parziale $f_M: A \rightsquigarrow B$ tale che:

$f_M(a) = b$ se $M$ con input $a$ fornisce in uscita $b$,

$f_M(a) = \perp$ se $M$ con input $a$ non si ferma mai.

Una funzione parziale $f: X \rightsquigarrow Y$ si dice *calcolabile parziale* se esiste un algoritmo $M$ tale che $f = f_M$.

Qualche autore parla di *semi-algoritmi* per designare quegli algoritmi che potrebbero non fermarsi per alcuni input.

# 2 Macchine di Turing e tesi di Church

## 2.1 Macchine di Turing

Un tentativo di formalizzare il concetto di funzione calcolabile è stato compiuto da Alan Turing nel 1936. Una macchina di Turing può essere informalmente descritta come un calcolatore ideale fornito di due tipi di memoria: interna ed esterna. Ad ogni dato momento entrambi i tipi di memoria contengono una quantità finita di informazione, ma mentre la memoria esterna è potenzialmente illimitata (nel senso che il calcolatore può richiedere in ogni momento una sua espansione), la memoria interna ha una fissata capacità massima che non può essere superata nell'intero processo di elaborazione, cosicché essa può assumere solamente un insieme finito $Q$ di stati. La memoria esterna è rappresentata da un nastro suddiviso in celle, dove ogni cella può contenere una quantità limitata di informazione, che senza perdita di generalità può essere rappresentata da un simbolo preso da un alfabeto finito $\Sigma$. La macchina è inoltre fornita di un puntatore che può scandire una cella alla volta della memoria esterna, e di un "programma" che determina il corso della elaborazione nel modo seguente. In funzione del contenuto della memoria interna e della parte della memoria esterna scandita dal puntatore, il programma specifica: 1) come deve essere modificato il contenuto della cella del nastro scandita in quel momento; 2) dove si deve spostare il puntatore (deve essere una cella contigua); 3) come deve essere alterata la

memoria interna. A partire dalla configurazione iniziale della memoria la macchina esegue ripetutamente le operazioni specificate dal programma arrestandosi se, e quando, viene raggiunto un determinato stato della memoria interna, detto stato di arresto.

Diamo ora le definizioni formali. Una macchina di Turing è costituita da:

0. Un *alfabeto* finito $\Sigma$ contenente almeno i simboli $0, 1, \#$ (dove $\#$ è un simbolo speciale per lo spazio bianco).

1. Un *nastro* infinito diviso in celle. Ogni cella contiene uno dei simboli di $\Sigma$. In ogni dato momento il nastro conterrà solamente un numero finito di simboli diversi da $\#$.

2. Un insieme finito $Q$ di *stati*. Assumiamo che tra gli stati $q \in Q$ vi siano due stati speciali di INIZIO $\in Q$ e di ARRESTO $\in Q$

3. Un *puntatore* del nastro che ad ogni dato momento si trova in uno degli stati di $Q$ e scandisce una cella del nastro. Esso può spostarsi lungo le celle del nastro, cambiare il contenuto delle celle stesse, e cambiare stato secondo delle regole che vedremo. Ad ogni movimento il puntatore scrive un simbolo di $\Sigma$ nella cella che viene scandita, sostituendo ciò che vi era scritto, e poi si sposta di una cella a sinistra o a destra.

4. Le regole che governano il comportamento del puntatore sono date da un *programma*. Esso consiste di una *funzione di transizione* $\delta: Q \times \Sigma \to Q \times \Sigma \times \{S, D\}$ (dove $S, D$ stanno per sinistra e destra)[6]. Supponiamo che in un certo momento della computazione il puntatore si trovi nello stato $q \in Q$ e stia scandendo una cella contenente il simbolo $\alpha \in \Sigma$. Sia $(q', \alpha', i) = \delta(q, \alpha)$ (dove $i \in \{S, D\}$). In questa situazione il puntatore compie le seguenti azioni: scrive il simbolo $\alpha'$ sulla cella che in quel momento sta scandendo (cancellando il simbolo $\alpha$ che vi si trovava), si porta nello stato $q'$, e se $q' \neq ARRESTO$ si sposta di una cella a sinistra o a destra a seconda che $i = S, F$. Nel caso $q' = ARRESTO$ la macchina di Turing si arresta.

## 2.2 Funzione parziale calcolata da una macchina di Turing.

Fissiamo $k \in \mathbf{N}$. Una macchina di Turing $M$ con un alfabeto di input-output contenente i simboli $0, 1$ calcola una funzione parziale $f_M^k: \mathbf{N}^k \rightsquigarrow \mathbf{N}$ nel modo seguente.

Rappresentiamo $x \in \mathbf{N}$ con la parola $11 \ldots 1$ ($x$ volte 1).

[6] Poichè $Q \times \Sigma$ è un insieme finito, la funzione di transizione può essere data sotto forma di una tabella finita

Se $k > 1$ rappresentiamo $(x_1, x_2, \ldots, x_k) \in \mathbf{N}^k$ con la parola che contiene $x_1$ volte il simbolo 1, seguita da uno 0, seguita dalla rappresentazione di $(x_2, \ldots, x_k)$.

Esempio: $(4, 3, 5)$ è rappresentata dalla parola 11110111011111.

Diciamo che $f_M^k(x_1, \ldots, x_k) = y$ se e solo se, facendo partire la macchina $M$ nello stato INIZIO con la parola che rappresenta $(x_1, \ldots, x_n)$ scritta inizialmente sul nastro e con il puntatore all'inizio della parola, abbiamo che dopo un certo numero finito di passi la macchina $M$ si arresta con la parola rappresentante $y$ scritta sul nastro e il puntatore all'inizio della parola.

Negli altri casi definiamo $f_M^k(x_1, \ldots, x_k) = \perp$[7].

Chiaramente possiamo usare una macchina di Turing anche per calcolare funzioni parziali non numeriche $f: A \rightsquigarrow B$. Basta fissare una rappresentazione degli elementi di $A$ e $B$ come parole sull'alfabeto $\Sigma$.

## 2.3 Tesi di Church

Nella seconda metà degli anni trenta vari autori, tra cui Turing, Post, Church e Kleene, hanno indipendentemente cercato di rendere rigoroso il concetto di funzione calcolabile (si tenga presente che non esistevano ancora i calcolatori). A tal fine Turing ha introdotto il concetto di "macchina di Turing" (una sorta di calcolatore astratto il cui funzionamento può essere facilmente simulato con carta e penna). Una funzione si dice Turing-calcolabile se può essere calcolata da una macchina di Turing. Similmente Kleene ha definito le funzioni "$\mu$-ricorsive", e Church le funzioni "$\lambda$-calcolabili". È stato successivamente dimostrato che le funzioni Turing calcolabili, $\mu$-ricorsive, e $\lambda$-calcolabili coincidono. Questi tre concetti individuano dunque una unica classe di funzioni, dette *funzioni ricorsive parziali*. La coincidenza di questi diversi tentativi di rendere rigoroso il concetto di funzione calcolabile ha portato all'accettazione della:

**Tesi di Church** Le funzioni algoritmicamente calcolabili parziali coincidono con le funzioni parziali Turing calcolabili.

Questa è una tesi che può essere giustificata e motivata in vari modi ma che ovviamente non può essere "dimostrata" in quanto il concetto di algoritmo è intuitivo e informale.

[7] È facile vedere che possiamo sempre modificare una macchina di Turing in modo da ottenerne un'altra che ha esattamente lo stesso comportamento della prima eccetto che, nel caso la prima si ferma senza fornire in uscita la rappresentazione di un intero, la seconda non si ferma mai.

L'accettazione della Tesi di Church rende possibile dimostrare la non-esistenza di algoritmi per calcolare determinate funzioni o risolvere determinati problemi.

# 3 Insiemi decidibili e semidecidibili

Consideriamo per semplicità sottoinsiemi di $\mathbf{N}$ sebbene le seguenti definizioni si applichino a sottoinsiemi di $\mathbf{N}^k$ ed anche ad insiemi non numerici, ad esempio sottoinsiemi di $\Sigma^*$, ovvero insiemi i cui elementi siano rappresentati da stringhe finite di simboli da un certo alfabeto finito $\Sigma$.

Un insieme $A \subseteq \mathbf{N}$ si dice *decidibile* (come sottoinsieme di $\mathbf{N}$) se esiste un algoritmo che, dato in ingresso $x \in \mathbf{N}$ (in una certa notazione fissata) stabilisce se $x$ appartiene o non appartiene ad $A$ (ad esempio fornendo in uscita 1 o 0 a seconda che $x$ appartenga o non appartenga ad $A$).

In altre parole $A$ è decidibile se la sua funzione caratteristica $\chi_A: U \to \{0,1\}$ (definita da $\chi_A(x) = 1$ se $x \in A$, $\chi_A(x) = 0$ se $x \notin A$) è calcolabile.

Esempio: l'insieme dei numeri primi (come sottoinsieme dei numeri naturali) è decidibile. Dato $x \in \mathbf{N}$ possiamo stabilire se $x$ è primo o no esaminando tutti i possibili divisori di $x$ minori di $x$.

Un insieme $A \subseteq \mathbf{N}$ si dice *semidecidibile* se esiste un algoritmo tale che, se viene dato in ingresso un elemento $x \in A$, l'esecuzione dell'algoritmo termina dopo un numero finito di passi, mentre se in ingresso viene dato un elemento $x \in U - A$, l'esecuzione dell'algoritmo non termina mai.

Segue dalle definizioni che un insieme è semidecidibile se e solo se è il dominio di una funzione calcolabile parziale.

Chiaramente $\mathbf{N}$ e l'insieme vuoto e l'insieme sono (sia decidibili che) semidecidibili (come sottoinsiemi di $\mathbf{N}$). Basta considerare un algoritmo che non si ferma per alcun input ed uno che si ferma per ogni input. Abbiamo:

**Proposizione 3.1** *Ogni insieme decidibile è anche semidecidibile.*

Dim. Sia $A \subseteq \mathbf{N}$ decidibile. Ne segue, per definizione, che esiste un algoritmo $M_1$ che dato $x \in \mathbf{N}$ stabilisce se $x$ appartiene o no ad $A$ (dando come uscita 1 o 0 rispettivamente). Consideriamo ora un algoritmo ausiliario $M_2$ che avendo in ingresso 1 si ferma subito, ed avendo in ingresso 0 comincia un calcolo senza fine. Sia $M_3$ l'algoritmo che si ottiene componendo $M_1$ ed $M_2$ in modo tale che l'uscita di $M_1$ viene fornita in entrata ad $M_2$. Ne segue che se $x \in A$, l'algoritmo $M_3$ si ferma dopo un numero finito di passi, e se

$x \in \mathbf{N} - A$, l'algoritmo $M_3$ non si ferma mai. Quindi $A$ è semidecidibile. QED

**Teorema 3.2** *(Teorema di Post) Supponiamo che sia $A$ che il suo complemento $\mathbf{N} - A$ siano semidecidibili. Allora $A$ è decidibile.*

Dim. Dato $x \in \mathbf{N}$ possiamo dare $x$ come ingresso ai due semi-algoritmi per $A$ e $\mathbf{N} - A$ e eseguirli simultaneamente in modo alternato (eseguendo cioè un passo dell'uno ed uno dell'altro). Poichè $x$ appartiene ad $A$ oppure ad $\mathbf{N} - A$, uno dei due semi-algoritmi si ferma dopo un numero finito di passi. Se è il primo a fermarsi, abbiamo stabilito che $x \in A$, se invece si ferma il secondo, $x \notin A$. In questo modo abbiamo definito un algoritmo per stabilire se $x$ appartiene o no ad $A$ e quindi $A$ è decidibile. QED

Ricordiamo che un insieme $A$ è *numerabile* se è vuoto oppure esiste una lista $a_0, a_1, a_2, \ldots$ indiciata dai numeri naturali che esaurisce tutti e soli gli elementi di $A$ (non è detto che nella lista non ci siano ripetizioni, e non escludiamo il caso in cui $A$ sia finito).

Un insieme $A \subseteq \mathbf{N}$ si dice *algoritmicamente enumerabile* se $A$ è vuoto oppure esiste una enumerazione $a_0, a_1, a_2, \ldots$ degli elementi di $A$ che è effettiva, nel senso che la funzione $n \mapsto a_n$ (da $\mathbf{N}$ ad $\mathbf{N}$) è calcolabile.

Quindi un insieme non-vuoto $A$ è algoritmicamente enumerabile se e solo se è l'immagine di una funzione calcolabile con dominio $\mathbf{N}$.

Esiste una importante differenza tra gli insiemi numerabili e quelli algoritmicamente numerabili: mentre ogni sottoinsieme di un insieme numerabilie è numerabile, un sottoinsieme di un insieme algoritmicamente enumerabile non è necessariamente algoritmicamente enumerabile. Ad esempio $\mathbf{N}$ è algoritmicamente enumerabile, ma vedremo (assumendo la "tesi di Church") che $\mathbf{N}$ possiede sottoinsiemi non algoritmicamente enumerabili.

**Teorema 3.3** *Un insieme $A \subseteq \mathbf{N}$ è semidecidibile se e solo se è algoritmicamente enumerabile.*

Dim. Se $A$ è vuoto è sia semidecidibile che algoritmicamente enumerabile. Assumiamo che $A$ non sia vuoto.

Supponiamo che $A$ sia algoritmicamente enumerabile e fissiamo una funzione calcolabile $f\colon \mathbf{N} \to A$ suriettiva. Definiamo ora un algoritmo $H$ nel modo seguente. Avendo $x \in \mathbf{N}$ in ingresso $H$ comincia a enumerare uno

dopo l'altro $f(0), f(1), f(2), \ldots$ (usando un algoritmo per calcolare $f$) e si ferma solamente se nel corso di questa enumerazione trova un elemento $f(n)$ uguale ad $x$. È chiaro che l'algoritmo $H$ così definito si ferma su input $x \in \mathbf{N}$ se e solo se $x \in A$. Quindi $A$ è semidecidibile.

Il viceversa è più complicato e richiede di fissare una funzione calcolabile suriettiva $f: \mathbf{N} \to \mathbf{N} \times \mathbf{N}$ (si usi una codifica delle coppie di numeri). Supponiamo che $A$ sia semidecidibile. Esiste quindi un algoritmo $H_1$ che si ferma su input $x \in \mathbf{N}$ se e solo se $x \in A$. Fissiamo un elemento $a \in A$ e definiamo una funzione calcolabile $g: \mathbf{N} \to \mathbf{N}$ nel modo seguente: per calcolare $g(n)$ calcoliamo dapprima $f(n) = (a_n, b_n) \in \mathbf{N} \times \mathbf{N}$. Ora poniamo $g(n) = a_n$ se l'algoritmo $H_1$ su input $a_n$ si ferma in al più $b_n$ passi (questo può essere controllato algoritmicamente semplicemente eseguendo $H_1$ per al più $b_n$ passi), e $g(n) = a$ nel caso contrario. È chiaro che i valori di $g$ appartengono tutti ad $A$. Resta da vedere che ogni elemento $u \in A$ è nell'insieme dei valori di $g$. Fissiamo dunque $u \in A$. Ne segue che $H_1$ applicato ad $u$ si ferma dopo un certo numero di passi, diciamo $m$ passi. Poichè $f: \mathbf{N} \to \mathbf{N} \times \mathbf{N}$ è suriettiva, esiste $n \in \mathbf{N}$ tale che $f(n) = (u, m)$. Per definizione di $g$, $g(n) = u$. Quindi $g: \mathbf{N} \to A$ è suriettiva (e calcolabile) e $A$ è algoritmicamente enumerabile. QED

Esercizio: l'unione, l'intersezione, e il prodotto cartesiano di insiemi algoritmicamente enumerabili è algoritmicamente enumerabile. Tutto ciò è vero a fortiori per gli insiemi decidibili che però hanno una proprietà in più: il complemento di un insieme decidibile è decidibile. Quindi i sottoinsiemi decidibili di $\mathbf{N}$ formano un algebra di Boole (rispetto a unioni, intersezioni e complementi).

Esercizio: Dimostrare il teorema precedente per insiemi non numerici $A \subseteq \Sigma^*$. Cosa prende il posto della funzione calcolabile suriettiva $f: \mathbf{N} \to \mathbf{N} \times \mathbf{N}$?

Esercizio: Siano $A, B, C$ tre insiemi algoritmicamente enumerabili disgiunti la cui unione sia $\mathbf{N}$. Dimostrare che $A, B, C$ sono decidibili.

## 3.1 Enumerabilità algoritmica dei teoremi

Il teorema che segue fornisce l'esempio più importante di un insieme semidecidibile che non è in generale decidibile.

**Teorema 3.4** *L'insieme dei teoremi di una teoria formale con un numero*

*finito di assiomi*[8] *è semidecidibile.*

Dim. Supponiamo di avere una teoria $T$ con un insieme finito di assiomi $Ax(T) \subseteq \Sigma^*$ dove $\Sigma$ è un alfabeto finito. Non è importante che $T$ sia una teoria del primo ordine, l'unica cosa che conta è che abbiamo un insieme finito di regole di inferenza (o di schemi di regole) che a partire dagli assiomi permettono di dedurre nuovi teoremi[9]. Sia $Th(T) \subseteq \Sigma^*$ l'insieme dei teoremi che si possono dedurre dagli assiomi usando un numero finito di volte le regole di inferenza. Mostreremo che $Th(T)$ è semidecidibile (come sottoinsieme di $\Sigma^*$).

A tal fine definiremo una procedura algoritmica che dato $x \in \Sigma^*$ cerca di dedurre $x$ dagli assiomi applicando in modo sistematico le regole di inferenza in tutti i modi possibili. Questo può essere fatto nel seguente modo: dapprima controlliamo se $x$ compare nella lista degli assiomi. Visto che gli assiomi sono in numero finito questo controllo termina dopo un numero finito di passi. Nel caso $x$ compare nella lista degli assiomi ci fermiamo (in questo caso $x$ è certamente un teorema). Se invece $x$ non è un assioma generiamo l'insieme $T_1$ di tutte quelle espressioni che si possono dedurre dagli assiomi usando al più una volta le regole di inferenza. Chiaramente $T_1$ è un insieme finito ed è possibile generarlo in modo algoritmico in un numero finito di passi. Ora controlliamo se $x$ appartiene a $T_1$. Se ci appartiene ci fermiamo, se no ripetiamo il procedimento ma partendo da $T_1$ invece che dall'insieme degli assiomi. In questo modo generiamo l'insieme $T_2$ di tutti i teoremi che si possono ottenere dagli assiomi usando al più due volte le regole di inferenza. Se $x$ è in $T_2$ ci fermiamo, se no ripetiamo il procedimento partendo da $T_2$ invece che da $T_1$ e così via.

È chiaro che se $x$ è un teorema, prima o poi seguendo questa procedura una corretta derivazione di $x$ dagli assiomi verrà trovata. Se però $x$ non è un teorema, l'algoritmo che abbiamo dato non è in grado di accorgersene e andrà avanti a cercare possibili derivazioni di $x$ di lunghezza sempre maggiore senza mai fermarsi. Questo mostra che l'insieme dei teoremi è in effetti semidecidibile[10]. QED

[8] Questa ipotesi si può indebolire: basta avere un insieme decidibile o semidecidibile di assiomi.

[9] Assumiamo che ci sia un modo algoritmico per passare dalle premesse di una regola di inferenza alla sua conclusione, escludiamo quindi regole di inferenza che abbiano bisogno di infinite premesse

[10] Ovviamente l'argomentazione data non dimostra che l'insieme dei teoremi non è decidibile in quanto a priori potrebbero esserci algoritmi più ingegnosi di quello che abbiamo dato.

Chiaramente l'algoritmo che abbiamo sopra dato non è molto ingegnoso e certamente non corrisponde a ciò che un matematico farebbe nel tentare di stabilire se una certa espressione è un teorema di una certa teoria formale. Più verosimilmente il matematico farebbe la cosa seguente: prima cercherebbe di dimostrare $x$. Se dopo un ragionevole lasso di tempo questo tentativo fallisse, il matematico comincerebbe a dubitare della validità di $x$ e cercherebbe di dare una dimostrazione della negazione di $x$. Se dopo vari tentativi anche questo tentativo non riuscisse, egli tornerebbe a cercare di dimostrare $x$ percorrendo strade prima inesplorate. In questo modo, alternando la sua attenzione tra $x$ e la sua negazione, il matematico può sperare di riuscire a trovare, avendo abbastanza tempo a disposizione, una dimostrazione di $x$ o della negazione di $x$. Nel caso di teorie incomplete è tuttavia possibile che nè $x$ nè la sua negazione siano derivabili dagli assiomi[11] e in questo caso questa procedura fallisce. Chiaramente questa procedura ricorda molto da vicino il teorema di Post.

**Teorema 3.5** *Sia $T$ una teoria formale* coerente e completa *con un numero finito di assiomi*[12]*. Allora i teoremi di $T$ sono un insieme decidibile.*

Dim. Per teorie $T$ coerenti e complete per ogni enunciato $x$ nel linguaggio della teoria, o $x$ o la sua negazione è dimostrabile in $T$ ma non entrambi. Ne segue che la ricerca alternata sopra delineata (consistente nel cercare alternativamente dimostrazioni di $x$ e della sua negazione) se organizzata in modo sistematico, definisce un algoritmo che riesce a dimostrare qualsiasi dato enunciato del linguaggio formale o la sua negazione. Questo fornisce un algoritmo per stabilire se $x$ è un teorema. QED

Una dimosrazione alternativa si basa sul teorema di Post: l'insieme dei teoremi è semidecidibile, l'insieme degli enunciati refutabili (ciò gli enunciati la cui negazione è un teorema) è per lo stesso motivo semidecidibile. Per teorie coerenti e complete questi due insiemi sono l'uno il complemento dell'altro all'interno dell'insieme di tutti gli enunciati ben formati. Poichè gli enunciati ben formati sono un sottoinsieme decidibile di $\Sigma^*$, una semplica applicazione del teorema di Post garantisce che entrambi questi insiemi sono decidibili.

[11] Questo capita ad esempio nella teoria dei gruppi: usando gli assiomi dei gruppi non si riesce a dimostrare nè la legge commutativa nè la sua negazione in quanto esistono sia gruppi commutativi che non.

[12] Anche qui si può indebolire l'ipotesi: basta che l'insieme degli assiomi sia algoritmicamente enumerabile

Resta da vedere se esistono interessanti teorie coerenti e complete complete. Gödel ha dimostrato che non è possibile avere una teoria coerente e completa nemmeno per quella parte della matematica che si occupa delle proprietà "elementari" (cioè esprimibili al primo ordine) dei numeri naturali. Tuttavia Tarski ha mostrato che esiste una teoria completa per le proprietà elementari dei numeri reali.

# 4 Funzioni primitive ricorsive

In una delle sue tante formulazioni equivalenti, la tesi di Church afferma che la classe delle funzioni calcolabili coincide con una classe di funzioni introdotta da Kleene, e nota come classe delle "funzioni $\mu$-ricorsive", o "funzioni ricorsive generali" (che si può dimostrare coincidere con la classe delle funzioni calcolabili su una macchina di Turing). Prima di definire tale classe introdurremo la classe più piccola, ma già molto comprensiva, delle "funzioni ricorsive primitive". Tali funzioni costituiscono un sottoinsieme dell'insieme delle funzioni numeriche calcolabili. Cominciamo con un esempio. La funzione fattoriale è definita da:

$0! = 1$,

$(n+1)! = (n+1) \cdot n!$.

Questa definizione è "ricorsiva" in quanto il simbolo della funzione fattoriale compare sia a sinistra che a destra di una delle uguaglianze che lo definiscono. La prima equazione definisce il fattoriale di zero. La seconda mostra come calcolare il fattoriale di $n+1$ supponendo di aver già precedentemente calcolato il fattoriale di $n$.

Questo genere di definizioni, in cui una funzione $f$ viene definita dando il valore di $f(0)$ e definendo $f(n+1)$ in termini di $n$ ed $f(n)$ (tramite funzioni ausiliarie note), vengono dette "definizioni primitive ricorsive". Generalizzando al caso di funzioni di più variabili diciamo che una funzione $f\colon \mathbf{N}^{n+1} \to \mathbf{N}$ si dice ottenuta per **ricorsione primitiva** da $g\colon \mathbf{N}^n \to \mathbf{N}$ e da $h\colon \mathbf{N}^{n+2} \to \mathbf{N}$ se per ogni $x_1, \ldots, x_n, y \in \mathbf{N}$,

$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$,

$f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$.

Il valore di $f(x_1, \ldots, x_n, 0)$ è dato dalla prima equazione per qualunque valore dei **parametri** $x_1, \ldots, x_n$, e supponendo di conoscere il valore di $f(x_1, \ldots, x_n, y)$ possiamo ottenere per mezzo della seconda equazione il valore di $f(x_1, \ldots, x_n, y+1)$.

Nel caso senza parametri ($n = 0$) la funzione $g$ di riduce a una costante $k$

e lo schema di ricursione diventa $f(0) = k$ e $f(y+1) = h(y, f(y))$. Nel caso in cui $h$ dipenda solo dal secondo argomento nel senso che $h(y, z) = g(z)$ per una certa $g$, otteniamo $f(y+1) = g(f(y))$ dove $f(y+1)$ dipende solo da $f(y)$ e non anche da $y$. In questo caso $f(y)$ non è niente altro che l'**iterazione** $y$ volte della funzione $g$, cioè $f(y) = g(g(\ldots g(k))$ dove $g$ viene ripetuta $y$ volte e $k$ è il valore iniziale $f(0)$. In presenza di parametri $\vec{x} = (x_1, \ldots, x_n)$ l'iterazione diventa $f(\vec{x}, 0) = g(\vec{x})$ e $f(\vec{x}, y+1) = g(\vec{x}, f(\vec{x}, y))$.

Una funzione $f: \mathbf{N}^k \to \mathbf{N}$ si dice ottenuta per **composizione** a partire dalle funzioni $g, h_1, \ldots, h_n$ se per ogni $x_1, \ldots, x_k \in \mathbf{N}$,

$f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_n(x_1, \ldots, x_k))$.

Le **funzioni primitive ricorsive** sono, per definizione, la più piccola classe di funzioni numeriche chiusa per composizione e ricursione primitiva che contiene le seguenti **funzioni iniziali**:

1) La funzione zero: $Z(x) = 0$,

2) la funzione successore: $S(x) = x + 1$,

3) le funzioni proiezione: $U_i^n(x_1, \ldots, x_n) = x_i$ (assumiamo $1 \leq i \leq n$).

È facile verificare che la somma è primitiva ricorsiva. Infatti $x + y$ può essere definita iterando $y$ volte il successore a partire da $x$: $x + 0 = x, x + s(y) = s(x + y)$.

Analogamente il prodotto può essere definito iterando la somma, e l'esponenziale iterando il prodotto.

La funzione predecessore (con la convenzione che il predecessore di zero è zero) è primitiva ricorsiva in quanto può essere definita da $p(0) = 0, p(S(x)) = x$. Per far rientrare la seconda uguaglianza nello schema definitorio $p(S(x)) = h(x, p(x))$ basta scegliere come $h$ la funzione $U_1^2$.

La funzione "sottrazione con il punto", che coincide con la sottrazione dove questa fornisce un risultato non negativo ed è zero negli altri casi, è primitiva ricorsiva in quanto può essere definita iterando il predecessore: $x \dot{-} 0 = x, x \dot{-} S(y) = p(x \dot{-} y)$. Analogamente si può definire la "divisione sui naturali" iterando la sottrazione con il punto.

## Predicati primitivi ricorsivi

Un insieme $P \subseteq \mathbf{N}^k$ si dice **primitivo ricorsivo** se la sua funzione caratteristica $\chi_P$ (definita da $\chi_P(\vec{a}) = 0$ se $\vec{a} \notin P$ e $\chi_P(\vec{a}) = 1$ se $\vec{a} \in P$), è primitiva ricorsiva. L'insieme $P \subseteq \mathbf{N}^k$ può anche essere pensato come una relazione (o "predicato") a $k$-argomenti su $\mathbf{N}$ e scriviamo in tal caso $P(a_1, \ldots, a_n)$ anziché $(a_1, \ldots, a_n) \in P$.

Non è del tutto banale mostrare che il **predicato di uguaglianza** $x = y$ **è primitivo ricorsivo**, e cioè che lo è la funzione $\chi_=: \mathbf{N}^2 \to \mathbf{N}$ che vale uno o zero a seconda che il primo argomento sia uguale al secondo. Abbiamo bisogno di alcuni risultati preliminari. Innanzitutto il predicato di uguaglianza a zero "$x = 0$" è primitivo ricorsivo in quanto la sua funzione caratteristica $\chi_{=0}$ può essere definita da: $\chi_{=0}(0) = 1$, $\chi_{=0}(S(x)) = 0$. La disuguaglianza $x \leq y$ si può esprimere dicendo che $x \dot{-} y$ è uguale a zero. Ne segue che **la relazione $\leq$ è primitiva ricorsiva** in quanto la sua funzione caratteristica si ottiene componendo la funzione differenza con la funzione caratteristica del predicato di uguaglianza a zero. Definiamo $x \geq y$ come $y \leq x$. Anche $\geq$ è una relazione primitiva ricorsiva poiché la sua funzione caratteristica può essere definita dalla funzione caratteristica di $\leq$ e dalle proiezioni $U_1^2$ e $U_2^2$ che opportunamente composte ci permettono di permutare le variabili: $\chi_\geq(x, y) = \chi_\leq(U_2^2(x, y), U_1^2(x, y))$. Per mostrare che il predicato di uguaglianza è primitivo ricorsivo esprimiamo $x = y$ come la congiunzione di $x \leq y$ e $x \geq y$. Passando alle funzioni caratteristiche la congiunzione diventa il prodotto e otteniamo: $\chi_=(x, y) = \chi_\leq(x, y) \cdot \chi_\geq(y, x)$, da cui la nostra tesi.

Se $P \subset \mathbf{N}^n$ è un predicato primitivo ricorsivo e $f: \mathbf{N}^k \to \mathbf{N}$ è una funzione primitiva ricorsiva, per composizione l'insieme
$\{(x_1, \ldots, x_k, y_1, \ldots, y_{n-1}) \mid P(g(x_1, \ldots, x_k), y_1, \ldots, y_{n-1})\}$ è primitivo ricorsivo. In particolare prendendo come $P$ il predicato di uguaglianza, vediamo che se $f: \mathbf{N}^k \to \mathbf{N}$ è una funzione primitiva ricorsiva il suo **grafico** $\text{Grafo}(f) = \{(x_1, \ldots, x_k, y) \mid f(x_1, \ldots, x_k) = y\}$ è un insieme primitivo ricorsivo.

Il viceversa non è però vero: se una funzione $f: \mathbf{N}^k \to \mathbf{N}$ ha come grafico un insieme primitivo ricorsivo $P \subset \mathbf{N}^{k+1}$, non è detto che $f$ sia primitiva ricorsiva. Vedremo però che se $f$ oltre ad avere un grafo primitivo ricorsivo è maggiorata da una funzione primitiva ricorsiva, allora è essa stessa primitiva ricorsiva.

## Connettivi booleani e quantificatori limitati

**Proposizione 4.1** *I predicati primitivi ricorsivi sono chiusi per connettivi booleani.*

Dim. La funzione caratteristica della congiunzione di due predicati è il prodotto delle rispettive funzioni caratteristiche. La funzione caratteristica della negazione di un predicato si ottiene componendo la funzione $1 \dot{-} x$

con la funzione caratteristica del predicato stesso. Gli altri connettivi si ottengono dalla congiunzione e dalla negazione. QED

Per la proposizione appena dimostrata i sottoinsiemi primitivi ricorsivi di $\mathbf{N}^k$ costituiscono un'algebra di Boole, ovvero sono chiusi per intersezioni finite, unioni finite e complementi.

In generale i predicati primitivi ricorsivi non sono chiusi rispetto ai quantificatori $\forall x$ ed $\exists x$. Se $\{(x, \vec{y}) \mid P(x, \vec{y})\}$ è primitivo ricorsivo, non è detto che $\{\vec{y} \mid \forall x P(x, \vec{y})\}$ e $\{\vec{y} \mid \exists x P(x, \vec{y})\}$ siano primitivi ricorsivi. Questo non dovrebbe stupire in quanto mentre si ha sempre un metodo finito per stabilire se un predicato primitivo ricorsivo è verificato per un dato valore degli argomenti, non esiste in generale un criterio finito per stabilire se un dato $\vec{y}$ verifica $\forall x P(x, \vec{y})$ o $\exists x P(x, \vec{y})$. Il seguente teorema mostra che la situazione cambia se limitiamo i quantificatori in modo che la variabile quantificata vari su un insieme finito.

**Teorema 4.2** *Se $P$ è un predicato primitivo ricorsivo di $n+1$ variabili, allora lo è anche il predicato $R$ definito da $R(x_1, \ldots, x_n, z) \leftrightarrow \forall y \leq z P(x_1, \ldots, x_n, y)$. Analogamente con $\exists y \leq z$ al posto di $\forall y \leq z$. In breve, i predicati primitivi ricorsivi sono chiusi per quantificazione limitata.*

Dim. Passando alle funzioni caratteristiche il quantificatore universale limitato $\forall x \leq z$ diventa una produttoria:

$\chi_R(x_1, \ldots, x_n, z) = \Pi_{y=0}^{z} \chi_P(x_1, \ldots, x_n, y)$.

Mostriamo dunque che le funzioni primitive ricorsive sono chiuse per produttorie $\Pi_{y=0}^{z}$. Sia dunque $g$ una funzione primitiva e consideriamo la funzione $f(\vec{x}, z) = \Pi_{y=0}^{z} g(\vec{x}, y)$. Possiamo dare una definizione primitiva ricorsiva di $f$ nel modo sequente:

$f(\vec{x}, 0) = g(\vec{x}, 0)$,

$f(\vec{x}, z+1) = f(\vec{x}, z) \cdot g(\vec{x}, z+1)$.

Il caso del quantificatore esistenziale limitato si riconduce al caso del quantificatore universale limitato tramite la negazione: $\exists x \leq t \varphi \leftrightarrow \neg \forall x \leq t \neg \varphi$. QED

**Corollario 4.3** *Il predicato "$x$ è primo" è primitivo ricorsivo.*

Dim. $x$ è primo se e solo se $\forall u, v \leq x (x = u \cdot v \rightarrow u = 1 \vee v = 1)$. Basta ora usare il fatto che il predicato di uguaglianza è primitivo ricorsivo, la

moltiplicazione è primitiva ricorsiva, e i predicati primitivi ricorsivi sono chiusi per connettivi booleani e quantificazioni limitate. QED

Mostriamo ora che i predicati primitivi ricorsivi sono chiusi ripetto alle definizioni per casi.

**Lemma 4.4** *Sia $P(\vec{x})$ un predicato primitivo ricorsivo e definiamo*
$f(\vec{x}) = g(\vec{x})$ *se* $P(\vec{x})$,
$f(\vec{x}) = h(\vec{x})$ *se* $\neg P(\vec{x})$.
*Allora se $g, h$ sono funzioni primitive ricorsive, anche $f$ lo è.*

Dim. $f(\vec{x}) = \chi_P(\vec{x}) \cdot g(\vec{x}) + \chi_{\neg P}(\vec{x}) \cdot h(\vec{x})$. QED

Il lemma precedente si generalizza a più di due casi: basta applicare il lemma ripetutamente. Indicheremo la funzione $f$ del precedente lemma con la notazione:
$f(\vec{x}) =$ se $P(x)$, allora $f(\vec{x})$, altrimenti $g(\vec{x})$.

## Minimalizzazione limitata

Introduciamo la notazione $\min x \leq z.P(x)$ per indicare il minimo $x \leq z$ che soddisfa il predicato $P(x)$ se un tale $x$ esiste, e conveniamo che $\min x \leq z.P(x)$ sia uguale a $z$ se tale $x$ non esiste. (Oltre ad $x$ il predicato $P$ può contenere altre variabili che servono come parametri.)

**Lemma 4.5** *Sia $f$ definita da $f(x_1, \ldots, x_k, z) = \min y \leq z.P(x_1, \ldots, x_n, y)$. Se $P$ è un predicato primitivo ricorsivo, allora $f$ è primitiva ricorsiva.*

Dim. Possiamo dare una definizione primitiva ricorsiva di $f$ tramite le equazioni:
$f(x_1, \ldots, x_n, 0) = 0$,
$f(x_1, \ldots, x_n, z+1) =$ se $\exists y \leq z P(x_1, \ldots, x_n, y)$, allora $f(x_1, \ldots, x_n, z)$, altrimenti $z+1$. QED

Oltre a minimalizzazioni limitate della forma $\min y \leq z.P$ possiamo ammettere minimalizzazioni della forma $\min y \leq g(\vec{x}, z).P$ dove $g$ è primitiva ricorsiva: basta comporre con $g$.

## La successione dei primi è primitiva ricorsiva

Sia $p(i)$ l'$i+1$-esimo numero primo, cioè $p(0) = 2, p(1) = 3, p(2) = 5, p(3) = 7, p(4) = 11, \ldots$.

**Proposizione 4.6** *La funzione $i \mapsto p(i)$ è primitiva ricorsiva.*

Dim. Dato $x$, per il teorema di Euclide esiste un numero primo $q$ con $x < q \leq x! + 1$. (Infatti sia $q$ un divisore primo di $x! + 1$. Se $q$ fosse minore o uguale a $x$, allora dividerebbe $x!$. Ma questa è una contraddizione perchè un numero diverso da 1 non può dividere al tempo stesso $x!$ e $x! + 1$.) Sia $q(x)$ il minimo numero primo maggiore di $x$. Poiché possiamo limitare $q(x)$ con $x! + 1$, la funzione $x \mapsto q(x)$ è primitiva ricorsiva in quanto è definibile per minimalizzazione limitata a partire dal predicato primitivo ricorsivo "$x$ è primo" e dalla funzione primitiva ricorsiva $x! + 1$. Possiamo ora definire in modo primitivo ricorsivo $p(0) = 2, p(i+1) = q(p(i))$. QED

## Codifica di successioni tramite la scomposizione in primi

Sia $p(i)$ l'$i+1$-esimo numero primo. Fissato $n \in \mathbf{N}$, la funzione $(x_0, \ldots, x_n) \mapsto \Pi_{i=0}^{n} p(i)^{x_i+1}$ da $\mathbf{N}^{n+1}$ a $\mathbf{N}$ è iniettiva (per l'unicità della scomposizione in primi) e primitiva ricorsiva. In particolare abbiamo le funzioni iniettive e primitive ricorsive $(x, y) \mapsto 2^{x+1}3^{y+1}$ da $\mathbf{N}^2$ ad $\mathbf{N}$ e $(x, y, z) \mapsto 2^{x+1}3^{y+1}5^{z+1}$ da $\mathbf{N}^3$ ad $\mathbf{N}$. Possiamo usare queste funzioni per codificare una coppia, una tripla, e più in generale una successione finita di numeri naturali, con un singolo numero naturale: $(a_0, a_1, \ldots, a_n)$ è codificata da $\Pi_{i=0}^{n} p(i)^{a_i+1}$ che indichiamo per brevità con $\langle a_0, a_1 \ldots, a_{n+1} \rangle$. Ad esempio $\langle 3, 1, 0 \rangle = 2^4 \cdot 3^2 \cdot 5^1 = 360$. Conveniamo che il numero 1 codifichi la successione vuota di lunghezza zero.

Una codifica leggermente più semplice si ottiene decrementando tutti gli esponenti di uno, associando ad esempio $(3, 1, 0)$ con $2^3 \cdot 3^1 \cdot 5^0$, ma lo svantaggio è che la tripla $(3, 1, 0)$ avrebbe la stessa codifica della coppia $(3, 1)$.

Si osservi che alcuni numeri, come ad esempio $2^3 \cdot 5^7$ non codificano alcuna successione. Un numero codifica una successione se e solo se i numeri primi che lo dividono costituiscono un segmento iniziale dei numeri primi (nel senso che se c'è un primo ci devono essere quelli minori di lui).

**Proposizione 4.7** *1. L'insieme dei numeri che codificano una successione è primitivo ricorsivo.*

2. *Esiste una funzione Lunghezza*: $\mathbf{N} \to \mathbf{N}$ *primitiva ricorsiva tale che se $s$ codifica una successione di lunghezza $n$, allora Lunghezza$(s) = n$ (non ha importanza quale valore abbia la funzione su argomenti che non codificano una successione).*

3. *Esiste una funzione Estrai*: $\mathbf{N}^2 \to \mathbf{N}$ *primitiva ricorsiva, tale che se $s$ codifica una successione di lunghezza $n$ e $i < n$ allora Estrai$(i, s)$ è l'$(i+1)$-esimo elemento della successione.*

4. *Esiste una funzione Cat*: $\mathbf{N}^2 \to \mathbf{N}$ *che dati come argomenti due numeri che codificano due successioni fornisce come valore il numero che codifica la concatenazione delle due successioni: Cat$(\langle a_1, \ldots, a_n\rangle, \langle b_1, \ldots, b_m\rangle) = \langle a_1, \ldots, a_n, b_1, \ldots, b_m\rangle$. Conveniamo che una successione non venga alterata concatenandola con la successione vuota.*

Dim. 1. Il predicato $x|y$ ($x$ divide $y$) può essere espresso con quantificatori limitati dalla formula $\exists z \leq x(x \odot z = y)$. Ora $s$ codifica una successione se e solo se $\forall i \leq s(p(i+1) \mid s \to p(i) \mid s)$. Ne consegue "$s$ codifica una successione" è esprimibile con quantificatori limitati a partire da predicati e funzioni primitive ricorsive, ed è pertanto un predicato primitivo ricorsivo.

2. Se una successione è codificata da $s$, la sua lunghezza è il minimo $i$ tale che $p(i)$ non divide $s$. Possiamo maggiorare tale $i$ con $s$, e quindi definire la funzione lunghezza per minimalizzazione limitata.

3. Possiamo definire per minimalizzazione limitata Estrai$(i, s)$ come il minimo $a \leq s$ tale che $p(i)^{a+2}$ non divide $s$.

4. Date due successioni codificate da $s$ e $r$, la loro concatenazione è codificata dal minimo $t$ tale che $\forall i <$ Lunghezza$(s)$ Estrai$(i, t) =$ Estrai$(i, s)$ e $\forall j <$ Lunghezza$(r)$ Estrai(Lunghezza$(s) + j, t) =$ Estrai$(j, r)$. Per concludere basta dare una limitazione primitiva ricorsiva a $t$. A tal fine osserviamo che una successione di lunghezza $m$ è codificata da un numero $\leq p(m)^{m \cdot N}$ dove $N$ è il massimo degli elementi della successione. Ne segue che possiamo limitare $t$ con $p(T)^{T \cdot T}$ dove $T = s + r$. QED

## Ricursione sul decorso dei valori

**Teorema 4.8** *Data una funzione $h$*: $\mathbf{N}^2 \to \mathbf{N}$ *sia $f$*: $\mathbf{N} \to \mathbf{N}$ *l'unica funzione tale che per ogni $x$, $f(x) = h(x, \langle f(0), \ldots, f(x-1)\rangle)$. In particolare*

$f(0) = h(0, \langle\rangle)$ *dove* $\langle\rangle = 1$ *codifica la succesione vuota. Se* $h$ *è primitiva ricorsiva, anche* $f$ *lo è. Un risultato analogo vale per funzioni di più argomenti alcuni dei quali giocano il ruolo di parametri. In questo caso l'equazione che definisce* $f$ *diventa:* $f(\vec{y}, x) = h(\vec{y}, x, \langle f(\vec{y}, 0), \ldots, f(\vec{y}, x-1)\rangle)$

Dim. Per semplicità di notazione consideriamo il caso senza parametri, e introduciamo la funzione ausiliaria definita da:

$f^{\#}(x) = \langle f(0), \ldots, f(x)\rangle = \pi_{i \leq x} p(i)^{f(i)+1}$

La funzione $f$ si ottiene in modo primitivo ricorsivo da $f^{\#}$ estraendo l'ultima componente, basta quindi mostrare che $f^{\#}$ è primitiva ricorsiva. Sia $a = \langle f(0)\rangle)$. Valgono le identità $f^{\#}(0) = a$ e $f^{\#}(x+1) = \mathrm{Cat}(f^{\#}(x), f(x+1)) = \mathrm{Cat}(f^{\#}(x), h(x+1, f^{\#}(x))$. Ne consegue che $f^{\#}$ è primitiva ricorsiva. QED

Usando la ricursione sul decorso dei valori si possono ottenere funzioni che dipendono da un qualsiasi numero di valori precedenti scelti in modo primitivo ricorsivo.

## 5 Funzioni ricorsive generali

Dato un predicato $P \subset \mathbf{N}^k$, possiamo definire una funzione parziale $f\colon \mathbf{N}^k \to \mathbf{N}$ stipulando che per ogni $x \in \mathbf{N}^k$, $f(x) = \min y P(x, y)$ se tale $y$ esiste, e $f(x)$ è indefinita altrimenti. Se $P$ è un predicato algoritmicamente calcolabile, $f$ è una funzione calcolabile parziale: infatti per calcolare $f(x)$ basta calcolare in sequenza i valori di verità di $P(x,0), P(x,1), P(x,2), \ldots$ (usando un algoritmo per calcolare $P$), fermandosi se si trova un $y$ tale che vale $P(x,y)$, e ponendo in tal caso $f(x) = y$. Se un tale $y$ non esiste il calcolo dei successivi valori del predicato va avanti all'infinito.

L'operatore di minimalizzazione appena definito porta da predicati (che possiamo identificare con le funzioni totali a valori $0, 1$) a funzioni parziali. È possibile estendere tale operatore ad un operatore $\mu$ da funzioni parziali a funzioni parziali nel modo seguente. Data una funzione parziale $g\colon \mathbf{N}^{k+1} \rightsquigarrow \mathbf{N}$ definiamo una funzione parziale $f\colon \mathbf{N}^k \rightsquigarrow \mathbf{N}$, $f(\vec{x}) = \mu y g(\vec{x}, y) = 1$ nel modo seguente (il valore 1 è stato scelto perchè, nel caso delle funzioni caratteristiche dei predicati, 1 corrisponde al valore "vero"):

$f(\vec{x}) = y$ se $g(\vec{x}, y) = 1 \wedge \forall i < y[g(\vec{x}, i) \neq 1 \wedge g(\vec{x}, i) \neq \perp]$.

$f(\vec{x}) = \perp$ se non esiste alcun $y$ tale che $g(\vec{x}, y) = 1 \wedge \forall i < y\ g(\vec{x}, i) \neq 1$ e $g(\vec{x}, i) \neq \perp$.

La funzione $f$ così definita si dice ottenuta per minimalizzazione da $g$. Dimostriamo ora che se $g$ è una funzione calcolabile parziale, allora $f(\vec{x}) = \mu y.g(\vec{x}, y) = 1$ è una funzione calcolabile parziale. Per trovare il valore di $f(\vec{x})$ possiamo calcolare in sequenza $g(\vec{x}, 0)$, $g(\vec{x}, 1), g(\vec{x}, 2), \ldots$ (usando l'algoritmo per $g$) fermandoci se e quando troviamo un $y$ tale che $g(\vec{x}, y) = 1$. Tale $y$ se esiste è il valore di $f(\vec{x})$. Si noti che questo algoritmo prima di iniziare a calcolare $g(\vec{x}, y)$ aspetta che il calcolo di $g(\vec{x}, 0), g(\vec{x}, 1), \ldots, g(\vec{x}, y-1)$ sia stato portato a termine. Quindi l'algoritmo non termina esattamente in quei casi in cui abbiamo posto $f(\vec{x}) = \perp$.

Abbiamo così dimostrato:

**Proposizione 5.1** *L'operatore di minimalizzazione $\mu$ porta da funzioni parziali calcolabili a funzioni parziali calcolabili.*

Un simile risultato è vero anche per gli operatori di composizione e ricursione primitiva se facciamo alcune ovvie convenzioni sui domini di definitione delle composizioni di funzioni parziali e delle funzioni ottenute per ricursione primitiva da funzioni parziali. Per quanto riguarda la composizione $f(\vec{x}) = g(h_1(\vec{x}), \ldots, h_k(\vec{x}))$ conveniamo che $f(\vec{x})$ è definita (per un dato $\vec{x}$) se lo sono tutte le $h_i(\vec{x})$ e se inoltre $g$ è definita sui valori delle $h_i(\vec{x})$. Questa non è l'unica convenzione possibile ma è la più semplice. Con questa convenzione una funzione composta della forma $f(x) = U_1^2(h_1(x), h_2(x))$ risulta indefinita ogniqualvolta $h_2(x)$ è indefinita, anche se $U_1^2: \mathbf{N}^2 \to \mathbf{N}$ dipende solo dal primo argomento. Per la ricursione primitiva si fa una convenzione analoga. L'idea generale è che

*una funzione parziale $f: \mathbf{N}^k \to \mathbf{N} \cup \{\perp\}$ si estende in modo naturale ad una funzione $f: (\mathbf{N} \cup \{\perp\})^k \to \mathbf{N} \cup \{\perp\}$ usando la convenzione che il risultato è $\perp$ se uno degli argomenti è uguale a $\perp$.*

Alla luce di questa convenzione consideriamo di nuovo la ricursione primitiva

$f(\vec{x}, 0) = g(\vec{x})$,

$f(\vec{x}, y+1) = h(\vec{x}, y, f(\vec{x}, y))$

dove però ora tutte le uguaglianze vanno considerate come uguaglianze tra funzioni da $(\mathbf{N} \cup \{\perp\})^k$ a $\mathbf{N} \cup \{\perp\}$. Ne segue che se $f(\vec{x}, y) = \perp$, allora $f(\vec{x}, y+1) = h(\vec{x}, y, \perp) = \perp$. Quindi in generale se $f(\vec{x}, i) = \perp$, allora $\forall k f(\vec{x}, i+k) = \perp$. È facile verificare che:

**Proposizione 5.2** *Se $g$ ed $h$ sono funzioni parziali calcolabili, e se $f$ è definita per ricursione primitiva da $g$ ed $h$, anche $f$ è parziale calcolabile.*

Dim. Per calcolare $b = f(\vec{x}, y)$ si calcolino in successione $b_0 = g(\vec{x})$, $b_1 = h(\vec{x}, 1, b_0)$, $b_2 = h(\vec{x}, 2, b_1)$, $b_3 = h(\vec{x}, 3, b_2)$, etc. (usando gli algoritmi per calcolare $g$ ed $h$), fino a che si arriva a $b = b_y = f(\vec{x}, b_{y-1})$. Se uno dei $b_i$ con $i < y$ non è definito il calcolo non termina mai in accordo con il fatto che in questi casi $f(\vec{x}, y) = \perp$. QED

**Definizione 5.3** Le funzioni **$\mu$-ricorsive** sono la più piccola classe di funzioni numeriche parziali contenente le funzioni primitive ricorsive iniziali e chiusa per composizione, ricursione primitiva e minimalizzazione (come operatori da funzioni parziali a funzioni parziali).

Il teorema di forma normale di Kleene stabilisce che ogni funzione $\mu$-ricorsiva $f: \mathbf{N}^k \rightsquigarrow \mathbf{N}$ si può scrivere nella forma $f(\vec{x}) = U(\min y P(\vec{x}, y))$, dove $U$ è una funzione primitiva ricorsiva e $P$ è un predicato primitivo ricorsivo. In particolare questo risultato dice che $f$ si può definire usando una sola volta l'operatore di minimalizzazione.

Si può dimostrare che le funzioni $\mu$-ricorsive coincidono con le funzioni Turing calcolabili parziali. Useremo il termine *funzioni ricorsive parziali* per indicare questa comune classe di funzioni. La tesi di Church afferma che una funzione parziale $f: \mathbf{N}^k \rightsquigarrow \mathbf{N}$ è calcolabile se e solo se è parziale ricorsiva.

Una funzione ricorsiva generale che risulta definita per ogni valore dei suoi argomenti viene detta **ricorsiva totale**.

# 6 Insiemi ricorsivi e ricorsivamente enumerabili

Un insieme $A \subseteq \mathbf{N}^k$ è *ricorsivo* se la sua funzione caratteristica è parziale ricorsiva (ovviamente essendo una funzione caratteristica è sicuramente totale).

Un insieme $A \subseteq \mathbf{N}^k$ è *semi-ricorsivo* se è il dominio di una funzione parziale ricorsiva $f: \mathbf{N}^k \rightsquigarrow \mathbf{N}$.

Un insieme $A \subseteq \mathbf{N}$ è *ricorsivamente enumerabile* se è l'immagine di una funzione parziale ricorsiva $f: \mathbf{N} \rightsquigarrow \mathbf{N}$. Nel caso di sottoinsiemi di $\mathbf{N}^k$ possiamo adoperare una codifica ricorsiva totale biunivoca $c: \mathbf{N}^k \to \mathbf{N}$ per ricondurci al caso di sottoinsiemi di $\mathbf{N}$. Diremo cioè che $A \subseteq \mathbf{N}^k$ è ricorsivamente enumerabile se $\{c(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in A\}$ è ricorsivamente enumerabile.

Si può dimostrare che ogni insieme ricorsivo è semi-ricorsivo e che un sottoinsieme di $\mathbf{N}^k$ è ricorsivamente enumerabile se e solo se è semi-ricorsivo.

Se assumiamo la tesi di Church, decidibile = ricorsivo, e semidecidibile = semi-ricorsivo (= algoritmicamente enumerabile = ricorsivamente enumerabile).

# 7 Aritmetizzazione della sintassi e predicato di dimostrabilità

Faremo uso dei risultati sulla codifica delle successioni sviluppate nella sezione sulle funzioni primitive ricorsive.

Sia $L$ un linguaggio del primo ordine con un numero finito di simboli di funzione, relazione e costante e associamo ad ogni tale simbolo $s$ un numero naturale $\#(s)$. Associamo poi ai simboli logici $\neg, \vee, \wedge, \rightarrow, \forall, \exists, =$ altri numeri naturali $\#(\neg), \#(\vee), \#(\wedge), \#(\rightarrow), \#(\forall), \#(\exists), \#(=)$ diversi tra loro. Consideriamo infine un nuovo numero naturale che indichiamo con $\#(v)$.

**Definizione 7.1** Associamo ad ogni $L$-termine $t$ un numero naturale $\ulcorner t \urcorner$ nel modo seguente.

1. Associamo alla variabile $v_i$ il numero $\ulcorner v_i \urcorner = \langle \#(v), i \rangle$.
2. Se $c$ è un simbolo di costante di $L$, $\ulcorner c \urcorner = \langle \#(c) \rangle$.
3. Se $f$ è un simbolo di funzione $n$-aria di $L$, e $t$ è un $L$-termine della forma $f(t_1, \ldots, t_n)$, $\ulcorner t \urcorner = \langle \ulcorner f \urcorner, \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner \rangle$.

**Definizione 7.2** Associamo ad ogni $L$-formula $\varphi$ un numero naturale $\ulcorner \varphi \urcorner$ nel modo seguente.

1. Se $t_1, t_2$ sono $L$-termini, $\ulcorner t_1 = t_2 \urcorner = \langle \#(=), \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle$.
2. Se $R$ è un simbolo di relazione $n$-aria di $L$, e $t_1, \ldots, t_n$ sono $L$-termini, allora $\ulcorner R(t_1, \ldots, t_n) \urcorner = \langle \ulcorner R \urcorner, \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner \rangle$.
3. Se $\alpha, \beta$ sono $L$-formule, allora $\ulcorner \neg\alpha \urcorner = \langle \ulcorner \neg \urcorner, \ulcorner \alpha \urcorner \rangle$, $\ulcorner (\alpha \vee \beta) \urcorner = \langle \ulcorner \vee \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner (\alpha \wedge \beta) \urcorner = \langle \ulcorner \wedge \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner (\alpha \rightarrow \beta) \urcorner = \langle \ulcorner \rightarrow \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner \forall v_i \alpha \urcorner = \langle \ulcorner \forall \urcorner, \ulcorner v_i \urcorner, \ulcorner \alpha \urcorner \rangle$, $\ulcorner \exists v_i \alpha \urcorner = \langle \ulcorner \exists \urcorner, \ulcorner v_i \urcorner, \ulcorner \alpha \urcorner \rangle$.

**Lemma 7.3** *L'insieme* $\{ \ulcorner t \urcorner \mid t$ *è un* $L$*-termine* $\}$ *è primitivo ricorsivo.*

Dim. Supponiamo per semplicità che il linguaggio $L$ consista di un simbolo di funzione binaria $f$, un simbolo di constante $c$, e un simbolo di relazione binaria $R$.

Abbiamo: $t$ è un $L$-termine se e solo se $t$ è la costante $c$ oppure $t$ è una variabile $v_i$ è una variabile, oppure $t = f(t_1, t_2)$ dove $t_1$ e $t_2$ sono $L$-termini.

Quindi:

> $n$ codifica un $L$-termine se e solo se $n = \lceil c \rceil$, oppure esiste $i < n$ tale che $n = \langle \lceil v \rceil, i \rangle$, oppure esistono $u, v < n$ tali che $n = \langle \lceil f \rceil, u, v \rangle$ e $u, v$ codificano $L$-termini.

In questo modo abbiamo definito il valore di verità di "$n$ codifica un $L$-termine" in termini dei valori di verità degli enunciati "$u$ codifica un $L$-termine", dove $u < n$, usando funzioni ausiliarie primitive ricorsive. Da ciò segue facilmente che l'insieme delle codifiche degli $L$-termini è primitivo ricorsivo. Per i dettagli si può procedere nel modo seguente. Dobbiamo mostrare che la funzione caratteristica degli $L$-termini, ovvero la funzione $T\colon \mathbf{N} \to \mathbf{N}$ che vale 1 sulle codifiche di $L$-termini e 0 altrimenti, è primitiva ricorsiva. A tal fine mostreremo che $T$ può essere definita per ricursione sul decorso dei valori, ovvero $T(n) = h(n, \langle T(0), \ldots, T(n-1) \rangle)$ per una opportuna funzione primitiva ricorsiva $h\colon \mathbf{N}^2 \to \mathbf{N}$. Basterà definire $h$ nel modo seguente:

**i)** $h(n, s) = 1$ se $n = \lceil c \rceil$ oppure $\exists i < n\colon n = \langle \lceil v \rceil, i \rangle$ oppure "$s$ codifica una successione di lunghezza $n$" e $\exists u, v < n\colon n = \langle \lceil f \rceil, u, v \rangle$ e Estrai$(u, s) =$ 1 e Estrai$(v, s) = 1$;

**ii)** $h(n, s) = 0$ altrimenti.

La funzione $h$ così definita è primitiva ricorsiva in quanto definita per casi a partire da predicati primitivi ricorsivi. QED

**Lemma 7.4** *L'insieme* $\{ \lceil \phi \rceil \mid \phi$ *è una L-formula* $\}$ *è primitivo ricorsivo.*

Dim. La dimostrazione è del tutto simile a quella del precedente lemma, e si basa sul fatto che possiamo definire il valore di verità dell'enunciato "$n$ codifica una $L$-formula" in termini del valore di verità degli enunciati "$a$ codifica una $L$-formula", dove $a < n$, usando funzioni ausiliarie primitive ricorsive (tra cui la funzione caratteristica degli $L$-termini).

Supponiamo per semplicità che $L$ contenga un simbolo di relazione binaria $R$ e nessun altro simbolo di relazione. Abbiamo:

"$n$ codifica una $L$-formula" se e solo se $\exists u, v < n$ tali che "$u, v$ codificano $L$-termini" e $(n = \langle \#(=), u, v\rangle$ o $n = \langle \#(R), u, v\rangle)$, oppure $\exists a, b < n$ tali che "$a, b$ codificano $L$-formule" e $(n = \langle \#(\neg), a\rangle$ o $n = \langle \#(\vee), a, b\rangle$ o $n = \langle \#(\wedge), a, b\rangle$ o $n = \langle \#(\rightarrow), a, b\rangle$ o $\exists i < n : n = \langle \#(\forall), \langle \#(v), i\rangle, a\rangle$ o $\exists i < n : n = \langle \#(\exists), \langle \#(v), i\rangle, a\rangle)$.

Possiamo ora procedere come nel lemma precedente ottenendo una definizione della funzione caratteristica delle $L$-formule per ricursione sul decorso dei valori. QED

**Lemma 7.5** *Esiste una funzione primitiva ricorsiva* $\mathbf{sub}: \mathbf{N}^3 \rightarrow \mathbf{N}$ *tale che* $\mathbf{sub}(\ulcorner\phi\urcorner, i, \ulcorner t\urcorner) = \ulcorner\phi[t/v_i]\urcorner$ *per ogni $L$-formula $\phi$ e ogni $L$-termine $t$. (Ricordiamo che $\phi[t/v_i]$ è il risultato della sostituzione in $\phi$ di tutte le occorrenze libere di $v_i$ con il termine $t$.)*

Dim. Basterà fare in modo che **sub** soddisfi le clausole che seguono, dove $f$ è un simbolo di funzione $n$-aria di $L$, $R$ è un simbolo di relazione $n$-aria di $L$, $t_1, \ldots, t_n$ sono $L$-termine, $\alpha, \beta$ sono $L$-formule. (Dalle clausole segue che **sub** effettua la sostituzione di $v_i$ con $t$ non solo all'interno di una formula ma anche anche all'interno di un termine.)

$$
\begin{aligned}
\mathbf{sub}(\ulcorner v_i\urcorner, , i, y) &= y \\
\mathbf{sub}(\ulcorner v_i\urcorner, j, y) &= \ulcorner v_i\urcorner \text{ se } i \neq j \\
\mathbf{sub}(\ulcorner f(t_1, \ldots, t_n)\urcorner, i, y) &= \langle \ulcorner f\urcorner, \mathbf{sub}(\ulcorner t_1\urcorner, i, y), \ldots, \mathbf{sub}(\ulcorner t_n\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner R(t_1, \ldots, t_n)\urcorner, i, y) &= \langle \ulcorner R\urcorner, \mathbf{sub}(\ulcorner t_1\urcorner, i, y), \ldots, \mathbf{sub}(\ulcorner t_n\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner \neg\alpha\urcorner, i, y) &= \langle \ulcorner\neg\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner (\alpha \vee \beta)\urcorner, i, y) &= \langle \ulcorner\vee\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y), \mathbf{sub}(\ulcorner\beta\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner (\alpha \wedge \beta)\urcorner, i, y) &= \langle \ulcorner\wedge\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y), \mathbf{sub}(\ulcorner\beta\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner (\alpha \rightarrow \beta)\urcorner, i, y) &= \langle \ulcorner\rightarrow\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y), \mathbf{sub}(\ulcorner\beta\urcorner, i, y)\rangle \\
\mathbf{sub}(\ulcorner \forall v_i\alpha\urcorner, i, y) &= \langle \ulcorner\forall\urcorner, \ulcorner v_i\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y)\rangle \text{ se } i \neq j \\
\mathbf{sub}(\ulcorner \forall v_i\alpha\urcorner, i, y) &= \ulcorner\forall v_i\alpha\urcorner \\
\mathbf{sub}(\ulcorner \exists v_i\alpha\urcorner, i, y) &= \langle \ulcorner\exists\urcorner, \ulcorner v_i\urcorner, \mathbf{sub}(\ulcorner\alpha\urcorner, i, y)\rangle \text{ se } i \neq j \\
\mathbf{sub}(\ulcorner \exists v_i\alpha\urcorner, i, y) &= \ulcorner\exists v_i\alpha\urcorner
\end{aligned}
$$

È possibile dare una definizione di **sub** per ricursione sul decorso dei valori della forma $\mathbf{sub}(n, i, y) = h(n, i, \langle \mathbf{sub}(0, i, y), \ldots, \mathbf{sub}(n-1, i, y)\rangle)$ per una opportuna funzione primitiva ricorsiva $h(n, s)$. La funzione $h$ è definita per casi (tanti casi quante le clausole sopra date). A titolo di esempio diamo il caso relativo a $n = \ulcorner(\alpha \wedge \beta)\urcorner$:

se $\exists a, b < n$ tali che $n = \langle \ulcorner \wedge \urcorner, a, b \rangle$, allora $h(n, i, s) = \langle \ulcorner \wedge \urcorner, \text{Estrai}(a, s), \text{Estrai}(b, s) \rangle$.

QED

**Osservazione 7.6** È possibile dare una definizione più semplice di **sub** se anzichè richiedere $\mathbf{sub}(\ulcorner \phi \urcorner, i, \ulcorner t \urcorner) = \ulcorner \phi[t/v_i] \urcorner$ richiediamo solamente che $\mathbf{sub}(\ulcorner \phi \urcorner, i, \ulcorner t \urcorner)$ sia la codifica non della formula $\phi[t/v_i]$ ma della formula a lei equivalente $\forall v_i(t = v_i \to \phi)$. Questo è in effetti l'approccio che abbiamo seguito a lezione.

**Corollario 7.7** *I seguenti insiemi sono primitivi ricorsivi.*

1. *L'insieme delle coppie $(\ulcorner \phi \urcorner, i)$ tali che $\phi$ è una L-formula e $v_i$ occorre libera in $\phi$.*
2. *L'insieme dei numeri di Gödel delle L-formule chiuse.*
3. *L'insieme delle coppie $(\ulcorner t \urcorner, i)$ tali che $t$ è un L-termine e la variabile $v_i$ occorre in $t$.*
4. *L'insieme delle terne $(\ulcorner \phi \urcorner, i, \ulcorner t \urcorner)$ tali che $\phi$ è una L-formula, $t$ è un L-termine, e $t$ è sostituibile per $v_i$ in $\phi$.*

Dim. Sia $\phi$ una $L$-formula. Chiaramente una variabile $v_j$ con un indice $j$ maggiore del numero di Gödel di $\phi$ non occorre in $\phi$ (nè libera nè legata), e quindi se la sostituiamo al posto di una variabile che occorre libera in $\phi$ otteniamo una formula contenente $v_j$ e pertanto diversa da $\phi$. Ne segue che, ponendo $j = \ulcorner \phi \urcorner + 1$, $v_i$ occorre libera in $\phi$, se e solo se $\phi \neq \phi[v_j/v_i]$. Quindi l'insieme del punto 1 è definibile in modo primitivo ricorsivo come l'insieme delle coppie $(a, i)$ tali che $a$ codifica una $L$-formula e $\mathbf{sub}(a, i, \langle \ulcorner v \urcorner, a+1 \rangle) \neq a$. Per il punto 2. basta osservare che $a$ codifica una formula chiusa se per ogni $i \leq a$ si ha $\mathbf{sub}(a, i, \langle \ulcorner v \urcorner, a + 1 \rangle) = a$. Il punto 3. è analogo al punto 1. Per l'ultimo punto osserviamo che $t$ è sostituibile al posto di $v_i$ in $\phi$ se e solo se ogni variabile $v_j$ occorrente in $t$ è sostituibile al posto di $v_i$ in $\phi$, e questo avviene se e solo se $\phi[v_j/v_i][v_i/v_j] = \phi[v_i/v_j]$, ovvero sostituendo le occorrenze libere di $v_i$ con $v_j$, e poi quelle di $v_j$ con $v_i$ nella formula risultante, si ottiene lo stesso risultato che sostituendo direttamente quelle di $v_j$ con $v_i$. Questa condizione è esprimibile in modo primitivo ricorsivo tramite la funzione **sub**. QED

**Teorema 7.8** *Sia $T$ una $L$-teoria tale che l'insieme $\{\ulcorner\phi\urcorner \mid \phi$ è un assioma di $T\}$ è primitivo ricorsivo. Allora esiste una relazione binaria primitiva ricorsiva* $\mathbf{Prov}_T$ *tale che $\phi$ è un teorema di $T$ se e solo se $\exists x \in \mathbf{N}: \mathbf{Prov}_T(x, \ulcorner\phi\urcorner)\}$.*

Dim. Il fatto che una data formula sia un teorema di $T$ non dipende dal sistema dimostrativo scelto (a condizione di considerare solo sistemi dimostrativi per i quali vale il teorema di completezza di Gödel). Tuttavia per procedere nella nostra dimostrazione dobbiamo fissare uno specifico sistema dimostrativo, in quanto il predicato $\mathbf{Prov}_T$ dipenderà da questa scelta avendosi:

$\mathbf{Prov}_T(x, \ulcorner\phi\urcorner)$ se e solo se $x$ codifica una dimostrazione di $\phi$

Per fissare le idee consideriamo il sistema dimostrativo nello stile Hilbert-Frege descritto nell'articolo di Barwise nell'Handbook of Mathematical Logic. Una dimostrazione consisterà allora di una successione finita di formule di cui ognuna è un assioma logico, un assioma di $T$, oppure segue da formule precedenti tramite opportune regole di inferenza. Le regole sono le seguenti: da $A$ e $A \to B$ posso dedurre $B$. Da $A \to B(x)$ posso dedurre $A \to \forall x B(x)$ a condizione che $x$ non sia libera in $A$. Da $B(x) \to A$ posso dedurre $\exists x B(x) \to A$ a condizione che $x$ non sia libera in $A$. Possiamo dedurre senza bisogno di alcuna premessa tutte le tautologie e gli assiomi dell'uguaglianza (riflessività, simmetria, transitività, e sostitutività espressa dallo schema: $x = y \wedge \phi(x) \to \phi(y)$). Se $t$ è un termine sostituibile al posto di $x$ in $A$, posso dedurre senza premesse le formule della forma $\forall x A(x) \to A(t)$ (cioè $\forall x A \to A[t/x]$) e quelle della forma $A(t) \to \exists x A(x)$.

Definiamo $\mathbf{Prov}_T(x, n)$ in modo primitivo ricorsivo come congiunzione delle seguenti condizioni:

1. Esiste $k \leq x$ tale che $x$ codifica una successione di lunghezza $k$ e $\text{Estrai}(k, x) = n$.

2. Per ogni $i < \text{Lunghezza}(x)$ $\text{Estrai}(i, x)$ codifica una formula.

3. Per ogni $i < \text{Lunghezza}(x)$ vale uno dei casi seguenti casi:

   (a) $\exists a, i, s < n$ tale che "$a$ codifica una $L$-formula", "$s$ codifica un $L$-termine sostituibile al posto di $v_i$ nella formula codificata da $a$", e $n = \langle \#(\forall)\langle \#(v), i\rangle, \langle \#(\to), a, \mathbf{sub}(a, i, s)\rangle\rangle$.

(b) $\exists j, l < i$ ed esistono $a, b \leq x$ tali che $a, b$ codificano formule, e Estrai$(j, x) = \langle \#(\rightarrow), a, b \rangle$ e Estrai$(l, x) = a$ e Estrai$(i, x) = b$.

(c) Estrai$(i, x)$ codifica un assioma di $T$.

(d) Gli altri casi sono simili e lasciati al lettore.

QED

# 8 Ogni funzione calcolabile è rappresentabile nella teoria $Q$ di Robinson

Ci proponiamo di dimostrare che se $f: \mathbf{N}^k \rightsquigarrow \mathbf{N}$ è una funzione ricorsiva generale, allora il grafico di $f$ è definibile nella struttura $(\mathbf{N}, 0, 1, S, +, \cdot)$ da una formula del primo ordine $\phi(\vec{x}, y)$ in $k+1$ variabili libere. Dimostreremo in realtà il risultato più forte che il grafico di $f$ è binumerabile nella teoria $Q$ di Robinson (vedi Teorema **??**).

La teoria $Q$ di Robinson assiomatizza alcune delle proprietà dei numeri naturali, ma a differenza della aritmetica di Peano non ha alcun assioma di induzione. Come conseguenza $Q$ è una teoria molto debole. Gli assiomi di $Q$ sono i seguenti (formulati in un linguaggio che contiene i simboli $0, S, +, \cdot$ per zero, successore, somma e prodotto):

Q1. $S(x) = S(y) \rightarrow x = y$,
Q2. $0 \neq S(x)$,
Q3. $x \neq 0 \rightarrow \exists y(x = S(y))$,
Q4. $x + 0 = x$,
Q5. $x + S(y) = S(x + y)$,
Q6. $x \cdot 0 = 0$,
Q7. $x \cdot S(y) = x \cdot y + x$.

Chiaramente le proprietà espresse dagli assiomi di $Q$ sono verificate nei numeri naturali con le solite operazioni aritmetiche di zero, successore, somma e prodotto. Quindi $\mathbf{N}$ è un modello di $Q$ (e scriviamo $\mathbf{N} \models Q$), ma non è certo l'unico modello. Gli assiomi di $Q$ sono infatti troppo deboli per caratterizzare $\mathbf{N}$.

Un'altro modello si può ottenere nel modo seguente. Sia $\mathbf{Z}[x]$ l'anello dei polinomi in una variabile a coefficienti in $\mathbf{Z}$. Definiamo un ordine totale su $\mathbf{Z}[x]$ stabilendo che un polinomio $p(x) \in \mathbf{Z}[x]$ è positivo se $\lim_{x \to \infty} p(x) > 0$

[13]. Sia $\mathbf{Z}[x]^+$ l'insieme dei polinomi non-negativi dell'anello $\mathbf{Z}[x]$: $p(x) \in \mathbf{Z}[x]^+$ se e solo se $p(x) \geq 0$ (cioè $p(x) > 0$ o $p(x) = 0$). Consideriamo le solite operazioni di somma e prodotto tra polinomi. È facile verificare che i polinomi non-negativi sono chiusi per somme e prodotti ed quindi anche per l'operazione successore (sommare il polinomio costante 1), inoltre $\mathbf{Z}[x]^+$ con queste operazioni verifica tutti gli assiomi di $Q$. Ad esempio il secondo assioma è soddisfatto in quanto il polinomio costante zero non è il successore di alcun polinomio non-negativo.

L'interesse per la teoria $Q$ è motivato dal fatto che $Q$ è in grado di verificare la correttezza o la falsità di ogni "calcolo numerico". Ad esempio $Q$ dimostra tutti i casi particolari della legge di commutatività della moltiplicazione: $2 \cdot 3 = 3 \cdot 2, 3 \cdot 7 = 7 \cdot 3$, etc. Tuttavia, a causa della mancanza di assiomi di induzione, $Q$ non è in grado di "generalizzare" e dimostrare la validità universale della legge commutativa: $\forall x, y(x \cdot y = y \cdot x)$. Vedremo però che $Q$ dimostra $\forall x, y \leq 10(x \cdot y = y \cdot x)$. Più in generale $Q$ è in grado di stabilire la verità o la falsità di qualsiasi formula aritmetica che contenga solo quantificatori "limitati". Il motivo è che la verità di tali formule si riconduce al controllo di un numero finito di casi particolari.

## Alcune formule dimostrabili in $Q$

Ogni termine chiuso $t$ di $Q$ interpretato in $\mathbf{N}$ denota un numero naturale. Ad esempio $[S(0) + S(0)]^{\mathbf{N}} = 2$. Se $n \in \mathbf{N}$ il termine $S(S(S(\ldots S(0))))$ ($n$ volte $S$) viene detto il **numerale** di $n$ e indicato con $\overline{n}$.

**Lemma 8.1** *$\forall a, b, c \in \mathbf{N}$, se $a + b = c$, allora $Q \vdash \overline{a} + \overline{b} = \overline{c}$.*

Dim. Per induzione su $b$ (l'induzione avviene nella metateoria non all'interno di $Q$, che peraltro non ha assiomi di induzione).

Se $b = 0$ basta usare il fatto che $Q \vdash \forall x(x + 0 = x)$.

Se $b > 0$ e $a + b = c$, allora $a + (b - 1) = c - 1$ e per ipotesi induttiva $Q \vdash \overline{a} + \overline{b-1} = \overline{c-1}$. Inoltre $Q \vdash x + S(y) = S(x + y)$. Prendendo $x = \overline{a}, y = \overline{b-1}$, otteniamo $Q \vdash \overline{a} + \overline{b} = \overline{c}$. QED

**Lemma 8.2** *$\forall a, b, c \in \mathbf{N}$, se $a \cdot b = c$, allora $Q \vdash \overline{a} \cdot \overline{b} = \overline{c}$.*

[13] Equivalentemente un polinomio $a_0 + a_1 x + \ldots + a_n x^n$, con $a_n \neq 0$, è positivo se e solo se $a_n > 0$

Dim. Per induzione su $b$ (nella metateoria) usando il lemma precedente. Per la base dell'induzione usiamo il fatto che $Q \vdash \forall x(x \cdot 0 = 0)$. Per il passo induttivo usiamo $Q \vdash x \cdot S(y) = x \cdot y + x$. QED

**Corollario 8.3** *Per ogni termine chiuso $t$ di $L(Q)$ esiste $n \in \mathbf{N}$ tale che $Q \vdash t = \overline{n}$.*

Dim. Per induzione sulla lunghezza di $t$.

Se $t$ è la costante 0, allora prendiamo $n = 0$.

Se $t = t_1 + t_2$, allora per ipotesi induttiva esistono $a, b \in \mathbf{N}$ tali che $Q \vdash t_1 = \overline{a}$ e $Q \vdash t_2 = \overline{b}$. Sia $c = a+b$. Per i lemmi precedenti $Q \vdash \overline{a}+\overline{b} = \overline{c}$. Ne segue che $Q \vdash t_1 + t_2 = \overline{c}$, cioè $Q \vdash t = \overline{c}$.

Se $t = t_1 \cdot t_2$ procediamo analogamente. QED

Abbiamo così dimostrato che $Q$ effettua correttamente il calcolo della addizione e moltiplicazione di numeri naturali ed è in grado di dimostrare che ogni termine chiuso corrisponde ad un numero naturale.

**Lemma 8.4** $\forall a, b \in \mathbf{N}$*, se $a \neq b$, allora $Q \vdash \overline{a} \neq \overline{b}$.*

Dim. Possiamo assumere $a < b$. La dimostrazione è per induzione su $a$.

Se $0 = a < b$, il risultato segue dal fatto che $Q \vdash \forall x.0 \neq S(x)$.

Se $0 < a < b$, allora $a-1 \neq b-1$ e per ipotesi induttiva $Q \vdash \overline{a-1} \neq \overline{b-1}$. Inoltre $Q \vdash \forall x, y.S(x) = S(y) \rightarrow x = y$, e quindi $Q \vdash \forall x, y.x \neq y \rightarrow S(x) \neq S(y)$. Prendendo $x = \overline{a-1}$, $y = \overline{b-1}$ otteniamo $Q \vdash \overline{a} \neq \overline{b}$. QED

**Lemma 8.5** $\forall a, b \in \mathbf{N}$*, se $a = b$, allora $Q \vdash \overline{a} = \overline{b}$.*

Dim. Segue da $Q \vdash \forall x.x = x$. QED

**Lemma 8.6** *Se $t_1$ e $t_2$ sono termini chiusi di $L(Q)$ allora $Q \vdash t_1 = t_2$ oppure $Q \vdash t_1 \neq t_2$.*

Dim. Siano $a, b \in \mathbf{N}$ tali che $Q \vdash t_1 = \overline{a}$ e $Q \vdash t_2 = \overline{b}$. Se $a \neq b$, allora $Q \vdash \overline{a} \neq \overline{b}$ e quindi $Q \vdash t_1 \neq t_2$. Analogamente se $a = b$, $Q \vdash t_1 = t_2$. QED

## Numeri non-standard

Dato un modello $M$ di $Q$ consideriamo il sottoinsieme $M' \subseteq M$ di quegli elementi di $M$ che sono l'interpretazione di qualche numerale $\overline{n}$, ovvero $M' = \{x \in M \mid \exists n \in \mathbf{N}.x = \overline{n}^M\}$.

Gli elementi di $M'$ sono chiamati **numeri standard** di $M$, mentre gli elementi di $M - M'$ sono chiamati numeri **non-standard** di $M$.

I numeri standard costituiscono quindi il più piccolo sottoinsieme di $M$ contenente lo 0 e chiuso per la funzione successore $S$ (valutata in $M$). Dai risultati fin qui mostrati segue che l'addizione e la moltimplicazione di numeri standard si comporta come la usuale addizione e moltiplicazione di numeri naturali e quindi $M'$ è una sottostruttura di $M$ isomorfa ad $\mathbf{N}$ con le usuali operazioni di addizione e moltiplicazione. Inoltre esiste un unico isomorfismo da $\mathbf{N}$ ad $M'$ dato dalla mappa che manda $n \in \mathbf{N}$ in $\overline{n}^M$. L'unicità segue dal fatto che un qualsiasi omomorfismo di $\mathbf{N}$ in $M$ deve mandare lo 0 nello 0 e preservare il successore $S$. Ne segue che $\mathbf{N}$ è contenuto (o meglio immergibile) in ogni modello $M$ di $Q$ e che $M$ è isomorfo ad $\mathbf{N}$ se e solo se $M$ non ha numeri non-standard.

Esempio: $\mathbf{N}$ è contenuto nel modello $\mathbf{Z}[x]^+$ e i numeri non-standard sono esattamente i polinomi non-costanti.

## La relazione $\leq$ in modelli di $Q$

Il linguaggio di $Q$ non comprende il simbolo $\leq$. Tuttavia possiamo definire $x \leq y$ come $\exists z.z + x = y$. Con questa definizione abbiamo:

**Corollario 8.7** $\forall n \in \mathbf{N}$, $Q \vdash \forall x\ (\ x \leq \overline{n} \leftrightarrow x = \overline{0} \vee x = \overline{1} \vee \ldots \vee x = \overline{n})$.

Dim. Induzione su $n$. Il verso da destra a sinistra è facile. Mostriamo l'altro verso.

Caso base: Sia $n = 0$. Se $x \leq 0$, allora $z + x = 0$ per qualche $z$. Mostreremo che $x = 0$. Se per assurdo $x \neq 0$, allora $x = S(u)$ per qualche $u$, e $z + x = z + S(u) = S(z + u) = 0$ contraddicendo il fatto che 0 non è il successore di alcun elemento.

Passo induttivo: Sia $n > 0$. Ragioniamo in $Q$. Supponiamo $x \leq \overline{n}$. Sia $z$ tale che $z + x = \overline{n}$. Se $x$ è diverso da 0, per Q3 possiamo scrivere $x = S(y)$. Per Q5 $z + y = \overline{n-1}$. Quindi $y \leq \overline{n-1}$. Per ipotesi induttiva $y = 0 \vee \ldots \vee y = \overline{n-1}$. Quindi $x = \overline{1} \vee \ldots \vee x = \overline{n}$. QED

Il risultato appena ottenuto implica che in un qualsiasi modello $M$ di $Q$ un elemento $a \in M$ che è minore di un numero standard $\overline{n}^M$ (cioè $M \models a \leq$

$\overline{n}$) è necessariamente un numero standard, e inoltre è un numero standard della forma $\overline{m}^M$ per qualche $m \leq n$. In altre parole l'isomorfismo tra $\mathbf{N}$ e i numeri standard di $M$ preserva non solo $0, S, +$ e $\cdot$, ma anche la relazione $\leq$.

Per abuso di linguaggio identificheremo talvolta $\mathbf{N}$ con i numeri standard di $M$ e scriveremo quindi $n$ anzichè $\overline{n}^M$.

**Corollario 8.8** *Sia $n \in \mathbf{N}$. Sono equivalenti:*
*1)* $\forall a \leq n Q \vdash \varphi(\overline{a})$,
*2)* $Q \vdash \forall x \leq \overline{n} \phi(x)$.

**Lemma 8.9** $\forall a, b \in \mathbf{N}$, $Q \vdash \overline{a} \leq \overline{b}$ *oppure* $Q \vdash \neg(\overline{a} \leq \overline{b})$.

Dim. Se $a \leq b$, allora $Q \vdash \overline{a} = \overline{0} \vee \overline{a} = \overline{1} \vee \ldots \vee \overline{a} = \overline{b}$ (in quanto la disgiunzione contiene $\overline{a} = \overline{a}$) e quindi $Q \vdash \overline{a} \leq \overline{b}$.

Se $a > b$, allora $Q \vdash \forall x \ . \ x = 0 \vee \ldots \vee x = \overline{b} \rightarrow x \neq \overline{a}$ (poichè $n \neq m$ implica $Q \vdash \overline{n} \neq \overline{m}$), da cui $Q \vdash \forall x . x \leq \overline{b} \rightarrow x \neq \overline{a}$ e quindi $Q \vdash \neg(\overline{a} \leq \overline{b})$. QED

**Lemma 8.10** $\forall b \in \mathbf{N} Q \vdash \forall x (x \leq \overline{b} \vee \overline{b} < x)$ [14].

Dim. Per induzione su $b$. Ragionando in $Q$ fissiamo $x$ e mostriamo $x \leq \overline{b}$ o $x \geq \overline{b}$. Per l'assioma $Q4$ $0 \leq x$. Supponiamo dunque $b > 0$. Per ipotesi induttiva $x \leq \overline{b-1}$ o $x \geq \overline{b-1}$. Nel primo caso per il Lemma **??** $x \leq \overline{b}$. Nel secondo caso sia $z$ tale che $z + \overline{b-1} = x$. Se $z = 0$, $x = \overline{b-1} \leq \overline{b}$. Se $z = S(y)$, allora $y + \overline{b} = x$ e quindi $\overline{b} \leq x$. In ogni caso $x \leq \overline{b}$ o $x \geq \overline{b}$. QED

In un modello non-standard $M$ di $Q$ la relazione $\leq$ non è mai un buon ordine (in quanto se $x \in M$ è non-standard, allora $M$ contiene la catena discendente infinita $x, x-1, x-2, x-3, \ldots$). In alcuni modelli di $Q$ la relazione $\leq$ non è neppure un ordine totale. Tuttavia abbiamo:

**Lemma 8.11** *Sia $M \models Q$ e sia $A \subseteq M$ un sottoinsieme di $M$ contenente un numero standard $n$. Allora $A$ ha un minimo elemento $m$, ovvero esiste $m \in A$ tale che $\forall a \in A \ M \models m \leq a$.*

[14] Dove $x < y$ sta per $x \leq y \wedge x \neq y$.

Dim. L'insieme $A' = \{x \in A \mid M \models x \leq n\}$ è un insieme che contiene solo numeri standard minori di $n$ e quindi in particolare è un insieme finito e totalmente ordinato (in quanto tra numeri standard la relazione $\leq$ è un ordine totale). Ne segue che $A'$ ha un minimo elemento $a \in A'$. Tale $a$ è anche un minimo di $A$ perchè se $x \in A - A'$ , allora $\neg(x \leq n)$ (dove $\leq$ è calcolato in $M$) e dato che $n$ è standard per la proposizione precedente $n < x$, da cui $a < x$. QED

## Le formule con quantificatori limitati sono decidibili in $Q$

Un enunciato $\varphi$ si dice **decidibile** in $Q$ (o $Q$-decidibile) se $Q \vdash \varphi$ oppure $Q \vdash \neg\varphi$. Una formula $\varphi(x_1, \ldots, x_n)$ si dice **$Q$-determinata** se ogni sua istanza numerica è decidibile, cioè $\forall a_1, \ldots, a_n \in \mathbf{N}$, $Q \vdash \varphi(\overline{a_1}, \ldots, \overline{a_n})$ oppure $Q \vdash \neg\varphi(\overline{a_1}, \ldots, \overline{a_n})$. Ovviamente se $\phi$ è un enunciato ($n = 0$) allora $\phi$ è $Q$-determinato se e solo se è $Q$-decidibile.

Da un punto di vista semantico (cioè dei modelli) dire che $\varphi(x_1, \ldots, x_n)$ è $Q$-determinata significa che $\forall a_1, \ldots, a_n \in \mathbf{N}$ l'enunciato $\varphi(\overline{a_1}, \ldots, \overline{a_n})$ ha lo stesso valore di verità in tutti i modelli di $Q$, cioè e vero in tutti i modelli di $Q$ (e in particolare in $\mathbf{N}$) oppure è falso in tutti i modelli di $Q$.

Da questa caratterizzazione è immediato vedere che una combinazione booleana di formule $Q$-determinate è $Q$-determinata. Ad esempio supponiamo che $\varphi(x_1, \ldots, x_n)$ e $\phi(x_1, \ldots, x_n)$ siano $Q$-determinate e mostriamo che la disgiunzione $\varphi(x_1, \ldots, x_n) \vee \phi(x_1, \ldots, x_n)$ è $Q$-determinata. Siano $a_1, \ldots, a_n \in \mathbf{N}$. Si hanno quattro possibili casi.

1) $Q \vdash \varphi(\overline{a_1}, \ldots, \overline{a_n})$ e $Q \vdash \phi(\overline{a_1}, \ldots, \overline{a_n})$.
2) $Q \vdash \varphi(\overline{a_1}, \ldots, \overline{a_n})$ e $Q \vdash \neg\phi(\overline{a_1}, \ldots, \overline{a_n})$.
3) $Q \vdash \neg\varphi(\overline{a_1}, \ldots, \overline{a_n})$ e $Q \vdash \phi(\overline{a_1}, \ldots, \overline{a_n})$.
4) $Q \vdash \neg\varphi(\overline{a_1}, \ldots, \overline{a_n})$ e $Q \vdash \neg\phi(\overline{a_1}, \ldots, \overline{a_n})$.

I casi in cui $Q$ non dimostra nessuna delle formule in questione nè la loro negazione è esculso perchè $\varphi$ e $\phi$ sono $Q$-determinate. Nel caso 4 abbiamo per inferenza tautologica $Q \vdash \neg(\varphi(\overline{a_1}, \ldots, \overline{a_n}) \vee \phi(\overline{a_1}, \ldots, \overline{a_n}))$. Negli altri tre casi $Q \vdash \varphi(\overline{a_1}, \ldots, \overline{a_n}) \vee \phi(\overline{a_1}, \ldots, \overline{a_n})$. Quindi $\varphi \vee \phi$ è $Q$-determinata.

Sebbene il linguaggio di $Q$ non abbia il simbolo $\leq$, possiamo definire $\leq$ come sopra e introdurre come al solito i quantificatori limitati $\forall x \leq t$ e $\exists x \leq t$ (dove $t$ è un termine non centenente la $x$) vengono tramite le abbreviazioni:

$\forall x \leq t\varphi$ sta per $\forall x(x \leq t \to \varphi)$
$\exists x \leq t\varphi$ sta per $\exists x(x \leq t \wedge \varphi)$

**Lemma 8.12** *Se $\varphi$ è $Q$-determinata e $t$ è un termine non contenente la $x$, allora le formule $\forall x \leq t\varphi$ e $\exists x \leq t\varphi$ sono $Q$-determinate.*

Dim. Siano $y_1, \ldots, y_n$ le variabili libere di $\forall x \leq t\varphi$. Allora $\forall x \leq t\varphi$ può essere scritta nella forma $\forall x \leq t(y_1, \ldots, y_n)\varphi(x, y_1, \ldots, y_n)$. Siano $a_1, \ldots, a_n \in \mathbf{N}$. Dobbiamo mostrare che l'enunciato $\forall x \leq t(\overline{a_1}, \ldots, \overline{a_n})\varphi(x, \overline{a_1}, \ldots, \overline{a_n})$ è decidibile in $Q$. Sia $b \in \mathbf{N}$ tale che $Q \vdash \overline{b} = t(\overline{a_1}, \ldots, \overline{a_n})$. Basta allora mostrare che l'enunciato $\forall x \leq \overline{b}\varphi(x, \overline{a_1}, \ldots, \overline{a_n})$ è decidibile in $Q$ (in pratica con questo passaggio ci siamo ricondotti al caso in cui $t$ è una variabile). Poichè $Q \vdash \forall x. x \leq \overline{b} \leftrightarrow x = \overline{0} \vee x = \overline{1} \vee \ldots \vee x = \overline{b}$, l'enunciato $\forall x \leq \overline{b}\varphi(x, \overline{a_1}, \ldots, \overline{a_n})$ è equivalente, in $Q$, all'enunciato $\varphi(\overline{0}, \overline{a_1}, \ldots, \overline{a_n}) \wedge \varphi(\overline{1}, \overline{a_1}, \ldots, \overline{a_n}) \wedge \ldots \wedge \varphi(\overline{b}, \overline{a_1}, \ldots, \overline{a_n})$ e quest'ultimo è decidibile in $Q$ in quanto combinazione booleana di enunciati decidibili in $Q$.

Il caso del quantificatore $\exists x \leq t$ si tratta analogamente considerando la disgiunzione $\varphi(\overline{0}, \overline{a_1}, \ldots, \overline{a_n}) \vee \varphi(\overline{1}, \overline{a_1}, \ldots, \overline{a_n}) \vee \ldots \vee \varphi(\overline{b}, \overline{a_1}, \ldots, \overline{a_n})$. QED

Una formula $\Delta_0$, è una formula senza quantificatori o una formula i cui quantificatori siano tutti limitati.

**Corollario 8.13** *Ogni formula $\Delta_0$ è $Q$-determinata.*

Per ogni enunciato $\phi \in \Delta_0$ abbiamo quindi:
1) se $\mathbf{N} \models \phi$, allora $Q \vdash \phi$,
2) se $\mathbf{N} \models \neg\phi$, allora $Q \vdash \neg\phi$.

Ne segue, ad esempio, che $Q \vdash \forall x \forall y \leq \overline{15}(x + y = y + x)$. D'altra parte si può mostrare che $Q \nvdash \forall x, y(x + y = y + x)$ in quanto ci sono modelli $M$ di $Q$ in cui la addizione non è commutativa. Naturalmente il fallimento della commutatività si può verificare solo per elementi non-standard di $M$ in quanto gli elementi standard di qualsiasi modello di $Q$ commutano (in quanto formano una struttura isomorfa ad $\mathbf{N}$).

Una spiegazione intuitiva del fatto che gli enunciati $\Delta_0$ sono decidibili in $Q$ sta nel fatto che un enunciato $\phi \in \Delta_0$ è vero in un modello $M$ di $Q$ se e solo se è vero nella sottostruttura $M'$ di $M$ costituita dai numeri standard (dimostrarlo come esercizio per induzione sulla lunghezza della formula usando la semantica di Tarski). Poichè $M'$ è isomorfo ad $\mathbf{N}$ ne segue che $\phi$ è vera in $M$ se e solo se è vera in $\mathbf{N}$ e per l'arbitrarietà di $M$ possiamo concludere che $\phi$ ha lo stesso valore di verità in tutti i modelli di $Q$ e quindi è decidibile in $Q$.

**Definizione 8.14** Le formule $\Sigma_1$ sono la più piccola classe di formule contenenti le $\Delta_0$ e chiusa per congiunzioni, disgiunzioni, quantificatori universali, e quantificatori esistenziali anche illimitati. Non richiediamo la chiusura per negazioni.

**Lemma 8.15** *Se $\phi$ è un enunciato $\Sigma_1$ e $\mathbf{N} \models \phi$, allora $Q \vdash \phi$.*

Dim. Per induzione sulla complessità di $\phi$. Il caso interessante è quando $\phi$ inizia con un quantificatore esistenziale illimitato ed è quindi della forma $\exists x \psi(x)$. Supponiamo che tale enunciato sia vero in $\mathbf{N}$ e sia $a \in \mathbf{N}$ tale che $\mathbf{N} \models \psi(\overline{a})$. Per induzione $Q \vdash \phi(\overline{a})$, da cui segue che $Q$ dimostra $\exists x \psi(x)$. QED

Riassumendo abbiamo:

1) Gli enunciati $\Delta_0$ veri in $\mathbf{N}$ sono dimostrabili in $Q$.

2) Gli enunciati $\Delta_0$ falsi in $\mathbf{N}$ sono refutabili in $Q$ (cioè è dimostrabile la negazione).

3) Gli enunciati $\Sigma_1$ veri in $\mathbf{N}$ sono dimostrabili in $Q$.

4) Gli enunciati $\Sigma_1$ falsi in $\mathbf{N}$ non sono dimostrabili in $Q$ (poichè $\mathbf{N} \models Q$).

Non è però detto che un enunciato $\Sigma_1$ falso in $\mathbf{N}$ sia refutabile in $Q$, ad esempio $\exists x, y. x + y \neq y + x$ è una formula $\Sigma_1$ falsa in $\mathbf{N}$ ma non refutabile in $Q$. Quindi esistono formule $\Sigma_1$ non $Q$-determinate.

## Rappresentabilità in $Q$ delle funzioni primitive ricorsive

Un insieme $A \subseteq \mathbf{N}^k$ è **binumerabile** in $Q$ se esiste una formula $\varphi_A(x_1, \ldots, x_k)$ [15] tale che $\forall a_1, \ldots, a_k \in \mathbf{N}$,

1) se $(a_1, \ldots, a_k) \in A$, allora $Q \vdash \varphi_A(\overline{a_1}, \ldots, \overline{a_k})$,

2) se $(a_1, \ldots, a_k) \notin A$, allora $Q \vdash \neg\varphi_A(\overline{a_1}, \ldots, \overline{a_k})$.

**Proposizione 8.16** *Ogni insieme $\Delta_0$-definibile è binumerabile in $Q$.*

Dim. Segue dal fatto che ogni formula $\Delta_0$ è $Q$-determinata. QED

Una funzione totale $f: \mathbf{N}^k \to \mathbf{N}$ è binumerabile in $Q$, se il suo grafo è binumerabile in $Q$, cioè esiste una formula $\varphi_f(x_1, \ldots, x_k, y)$ tale che $\forall a_1, \ldots, a_k \in \mathbf{N}$:

1) se $f(a_1, \ldots, a_k) = b$, allora $Q \vdash \varphi_f(\overline{a_1}, \ldots, \overline{a_k}, \overline{b})$,

[15] Un insieme può essere binumerato da tante formule non $Q$-equivalenti, quindi la notazione $\varphi_A$ è ambigua.

2) se $f(a_1, \ldots, a_k) \neq b$, allora $Q \vdash \neg\varphi_f(\overline{a_1}, \ldots, \overline{a_k}, \overline{b})$.

Se valgono queste due condizioni diciamo che $\varphi_f$ binumera $f$.

Se oltre ad (1) e (2) vale la condizione

3) $Q \vdash \exists! y \varphi_f(\overline{a_1}, \ldots, \overline{a_k}, y)$

allora diciamo che $f$ è **binumerata funzionalente** da $\varphi_f$. Ciò equivale a richiedere che $Q \vdash \forall y(\varphi_f(\overline{a_1}, \ldots, \overline{a_k}, y) \iff y = \overline{b})$ dove $b = f(a_1, \ldots, a_k)$.

Mostreremo che il grafico della funzione esponenziale, ovvero l'insieme $\{(x, y, z) \mid x^y = z\}$, è binumerabile funzionalmente in $Q$, e inoltre è possibile binumerarlo con una formula $\Sigma_1^0$. Lo stesso vale più in generale per una funzione primitiva ricorsiva o anche ricorsiva generale.

A tal fine abbiamo bisogno di un modo alternativo di codificare le successioni, non basato sulla scomposizione in fattori primi (che richiede l'esponenziazione) ma basato invece sul teorema cinese dei resti.

**Teorema 8.17** *(Teorema cinese dei resti) Siano $a_0, a_1, \ldots, a_n$ numeri a due a due relativamente primi. Siano $b_0, b_1, \ldots, b_n$ numeri arbitrari. Allora esiste un numero $x$ che per ogni $i \leq n$:*

$x \equiv b_i \pmod{a_i}$

*(Il numero $x$ è univocamente determinato modulo il prodotto degli $a_i$.)*

Dim. Assumiamo dapprima che uno dei $b_i$ sia 1 e tutti gli altri siano 0. Dato $i \leq n$ cerchiamo dunque un $x_i$tale che

$x_i \equiv 1 \pmod{a_i}$

$x_i \equiv 0 \pmod{a_j}\ \forall j \neq i$.

Poichè $a_i$ e $\Pi_{j \neq i} a_j$ sono relativamente primi, esistono $\alpha, \beta$ tali che $\alpha \cdot a_i + \beta \cdot \Pi_{j \neq i} a_j = 1$. Basta ora porre $x_i = \beta \cdot \Pi_{j \neq i} a_j$. Per risolvere il sistema originale basta scegliere $x = b_0 x_0 + b_1 x_1 + \ldots + b_n x_n$. QED

Per poter applicare per i nostri scopi il teorema cinese dei resti ci occorre un modo per poter generare con facilità una successione di numeri primi tra loro $a_0, \ldots, a_n$. Il seguente lemma mostra che esistono progressioni aritmetiche arbitrariamente lunghe composte di numeri arbitrariamente grandi e a due a due relativamente primi.

**Lemma 8.18** *Per ogni $n, x$ esiste $d > x$ tale che la progressione aritmetica $d + 1, 2d + 1, 3d + 1, \ldots, nd + 1$ è composta da numeri relativamente primi.*

Dim. È sufficiente prendere $d = y!$ dove $y > \max(n, x)$. Se infatti un numero primi $p$ dividesse due elementi della successione $d+1, 2d+1, \ldots, nd+$

1, diciamo $p|(i+1)d+1$ e $p|(j+1)d+1$ con $i > j$, allora $p$ dividerebbe la loro differenza $(i-j)d$ e quindi $p|(i-j)$ oppure $p|d$. Ma $p|d$ conduce subito ad un assurdo in quanto avendosi anche $p|(i+1)d+1$ ne conseguirebbe $p|1$. D'altra parte se $p|(i-j)$ allora, visto che $i-j \leq n < y$, ne conseguirebbe $p|y$! cioè di nuovo $p|d$ che è assurdo. QED

Sia $re(x,y)$ il resto della divisione di $x$ per $y$, cioè $re(x,y) = r$ se e solo se $\exists q \leq x(x = qy + r \wedge 0 \leq r < y)$. La funzione $\beta$ di Gödel, $\beta: \mathbf{N}^3 \to \mathbf{N}$, è definita da: $\beta(c,d,i) = re(c,(i+1)d+1)$. Osserviamo che il grafo della funzione $\beta$ è $\Delta_0^{\mathbf{N}}$: $\beta(c,d,i) = r \leftrightarrow \exists q \leq c(c = q[(i+1)d+1] + r \wedge 0 \leq r < (i+1)d+1)$. La proprietà fondamentale della funzione $\beta$ di Gödel è espressa dal seguente:

**Lemma 8.19** *Per ogni $n$ ed ogni seguenza di numeri $b_0, b_1, \ldots, b_n$ esistono $c, d$ tali che $\beta(c,d,0) = b_0$, $\beta(c,d,1) = b_1, \ldots, \beta(c,d,n) = b_n$.*

Dim. Dati $b_0, \ldots, b_n$ scegliamo $d$ in modo tale che i numeri $d+1, 2d+1, \ldots, nd+1$ siano relativamente primi e $d$ sia più grande di tutti i $b_i$. Ora scegliamo $c$ in modo che soddisfi le congruenze $c \equiv b_i \pmod{(i+1)d+1}$ per $i < n$. Ne segue che per ogni $i < n$, $b_i$ è il resto della divisione di $c$ per $(i+1)d+1$, ovvero $b_i = \beta(c,d,i)$. QED

In virtù del precedente lemma possiamo pensare alla coppia $c, d$ come ad una codifica della successione $b_0, \ldots, b_n$. La funzione $\beta$ può essere interpretata nel modo seguente: $\beta(c,d,i) =$ l'$i$-esimo elemento della successione codificata da $(c,d)$. Il lemma ci garantisce che ogni successione ha (almeno) una codifica. Volendo potremmo usare una opportuna "funzione coppia" per codificare $(c,d)$ con un singolo numero. In tal modo giungeremmo ad un metodo per codificare le sequenze con un singolo numero anzichè due, ma questo non è necessario per i nostri scopi.

La funzione $\beta$ può essere usata in alternativa alla Estrai vista precedentemente. Essendo $\Delta_0$-definibile, il grafico della funzione $\beta$ è $Q$-determinato.

**Lemma 8.20** *La funzione $\beta$ di Gödel è binumerabile funzionalmente in $Q$.*

Dim. Facile esercizio. QED

**Teorema 8.21** *Ogni funzione primitiva ricorsiva è binumerabile funzionalmente in $Q$ (da una formula $\Sigma_1^0$).*

Dim. Basta mostrare che le funzioni binumerabili funzionalmente in $Q$ (da una formula $\Sigma_1^{\mathbf{N}}$) contengono le funzioni primitive ricorsive iniziali, e sono chiuse per composizione e ricursione primitiva.

Lasciamo al lettore la verifica per le funzioni iniziali.

La chiusura per composizione è semplice: supponiamo ad esempio che $f(x) = h(g_1(x), g_2(x))$. Allora $f(x) = y$ se e solo se $\exists a, b, c(a = g_1(x) \wedge b = g_2(x) \wedge y = h(a, b))$, da cui sostituendo le uguaglianze che coinvolgono $g_1, g_2, h$ con le formule che binumerano funzionalmente tali funzioni, otteniamo la formula che binumera funzionalmente $f$.

Chiusura per ricursione primitiva.

Sia $f$ definita per ricursione primitiva da $g$ e da $h$:

$f(\vec{x}, 0) = g(\vec{x})$,

$f(\vec{x}, y + 1) = h(\vec{x}, y, f(\vec{x}, y))$.

Fissati $\vec{x}, y$ si ha che $f(\vec{x}, y) = z$ se e solo se esiste una successione di numeri $a_0, a_1, \ldots, a_y$ tale che:

1) $a_0 = g(\vec{x})$,

2) $a_y = z$,

3) $\forall i < y \; a_{i+1} = h(\vec{x}, y, a_i)$.

Per i risultati sulla funzione $\beta$ di Gödel esistono due numeri $c, d$ che codificano l'intera successione $a_0, a_1, \ldots, a_y$ nel senso che $\beta(c, d, 0) = a_0, \beta(c, d, 1) = a_1, \ldots, \beta(c, d, y) = a_y$. Abbiamo ora l'equivalenza: $f(\vec{x}, y) = z$ se e solo se $\exists c, d$ :

1) $\beta(c, d, 0) = g(\vec{x})$,

2) $\beta(c, d, y) = z$,

3) $\forall i < y \beta(c, d, i + 1) = h(\vec{x}, y, \beta(c, d, i))$.

Usando il fatto che la funzione $\beta$ e le funzioni $g$ ed $h$ sono binumerabili funzionalmente in $Q$, si può verificare che l'espressione così ottenuta per $f$ (rimpiazzando opportunamente $\beta, g, h$ con le rispettive formule) binumera funzionalmente $f$ in $Q$. Si osservi che $c, d$ non sono univocamente determinati da $\vec{x}, y$, ma i numeri $\beta(c, d, i)$ $(i = 0, \ldots, y)$ sono univocamente determinati. QED

Possiamo rafforzare il teorema come segue:

**Teorema 8.22** *Ogni funzione ricorsiva totale è binumerabile funzionalmente in $Q$ (da una formula $\Sigma_1^0$).*

Dim. Basta integrare la dimostrazione precedente nel modo seguente:

Chiusura per minimalizzazione

Sia $f(\vec{x})$ il minimo $y$ tale che $g(\vec{x},y)=0$ (supponeniamo che esista). Osserviamo che $f(\vec{x})=y$ se e solo se $g(\vec{x},y)=0 \wedge \forall i<y \exists v \neq 0\ g(\vec{x},i)=v$. Assumiamo che $g$ sia funzionalmente binumerabile in $Q$ dalla formula $G(\vec{x},y)$. Allora $f$ è funzionalmente binumerabile in $Q$ dalla formula $G(\vec{x},y,0) \wedge \forall i < y \exists v \neq 0\ G(\vec{x},i,v)$. (Verificare come esercizio.) QED

Poiché $\mathbf{N}$ è un modello di $Q$ otteniamo:

**Corollario 8.23** *Ogni funzione ricorsiva totale è binumerabile funzionalmente in $Q$ (da una formula $\Sigma_1^0$).*

# 9 Teoremi di incompletezza

Supponiamo che $L$ contenga un simbolo di funzione unario $s$ e un simbolo di costante 0.

**Lemma 9.1** *Esiste una funzione primitiva ricorsiva* $\mathbf{num}\colon \mathbf{N} \to \mathbf{N}$ *tale che per ogni* $n \in \mathbf{N}$, $\mathbf{num}(n) = \ulcorner s^n(0) \urcorner$, *dove il termine* $s^n(0)$ *è definito da* $s^0(0)=0$ *e* $s^{n+1}(0) = s(s^n(0))$.

Dim. $\mathbf{num}(0) = \ulcorner 0 \urcorner$, $\mathbf{num}(n+1) = \langle \ulcorner s \urcorner, \mathbf{num}(n) \rangle$. QED

Se $\ulcorner \alpha \urcorner = n \in \mathbf{N}$, indichiamo con $\overline{\ulcorner \alpha \urcorner}$ il termine $s^n(0)$ (la cui interpretazione nella struttura $\mathbf{N}$ è proprio il numero $n$). Data una formula $\alpha$ e un termine $t$, scriviamo $\alpha(t)$ per $\alpha[t/v_0]$.

**Lemma 9.2** *Esiste una funzione primitiva ricorsiva* $D\colon \mathbf{N} \to \mathbf{N}$ *tale che per ogni formula* $\alpha$, $D(\ulcorner \alpha \urcorner) = \ulcorner \alpha(\overline{\ulcorner \alpha \urcorner}) \urcorner$.

Dim. $D(x) = \mathbf{sub}(x, 0, \mathbf{num}(x))$. QED

**Lemma 9.3** *Sia* $\alpha(x)$ *una formula nella variabile libera* $x$. *Allora esiste una formula* $\beta$ *tale che* $Q$ *dimostra l'equivalenza* $\beta \leftrightarrow \alpha(\overline{\ulcorner \beta \urcorner})$.

Dim. Sia $\delta(x,y)$ la formula che binumera funzionalmente $D$ in $Q$. Poniamo $\beta = \gamma(\overline{\ulcorner \gamma \urcorner})$ dove $\gamma = \gamma(v_0)$ è la formula $\forall y(\delta(v_0,y) \to \alpha(y))$. Allora in $Q$ si ha: $\gamma(\overline{\ulcorner \gamma \urcorner})$ se e solo se $\forall y(\delta(\overline{\ulcorner \gamma \urcorner}, y) \to \alpha(y))$, e visto che l'unico $y$ che verifica $\delta(\overline{\ulcorner \gamma \urcorner}, y)$ è $\overline{\ulcorner \gamma(\overline{\ulcorner \gamma \urcorner}) \urcorner}$, questo vale se e solo se $\alpha(\overline{\ulcorner \gamma(\overline{\ulcorner \gamma \urcorner}) \urcorner})$. QED

**Teorema 9.4** *La teoria $PA$ (aritmetica di Peano del primo ordine) è incompleta.*

Dim. Poiché le codifiche degli assiomi di $PA$ sono un insieme primitivo ricorsivo, per il Teorema **??** esiste un insieme primitivo ricorsivo **Prov** tale che $PA \vdash \varphi$ se e solo se $\exists n : (n, \lceil\varphi\rceil) \in \mathbf{Prov}$. Sia $P(x, y)$ una formula che binumera **Prov** in $Q$. Sia $\mathrm{Teo}(x)$ la formula $\exists d P(d, x)$. Sia $G$ un enunciato tale che $Q$ dimostra $G \leftrightarrow \neg\mathrm{Teo}(\overline{\lceil G\rceil})$. Interpretatando questa equivalenza nel modello $\mathbf{N}$, si ha in particolare che $G$ è vera (in $\mathbf{N}$) se e solo se $G$ non è dimostrabile in $PA$. Mostriamo che nè $G$ nè la sua negazione è dimostrabile in $PA$.

Se per assurdo $PA \vdash G$, allora esiste $n$ tale che $(n, \lceil G\rceil) \in \mathbf{Prov}$. Fissato un tale $n$ si ha $Q \vdash P(n, \overline{\lceil G\rceil})$. Quindi $Q \vdash \exists d P(d, \overline{\lceil G\rceil})$, ovvero $Q \vdash \mathrm{Teo}(\overline{\lceil G\rceil})$. Ma allora per la scelta di $G$, $Q \vdash \neg G$, e poiché $PA$ contiene $Q$, $PA \vdash \neg G$. Questo è impossibile, in quanto $PA$ è coerente.

Abbiamo quindi dimostrato che $G$ non è dimostrabile in $PA$, e per la scelta di $G$ ne segue che $G$ è vera (in $\mathbf{N}$). Ma allora nemmeno la negazione di $G$ è dimostrabile in $PA$ in quanto $PA$ dimostra solo cose vere in $\mathbf{N}$ (essendo $\mathbf{N}$ un suo modello). QED

Con essenzialmente la stessa dimostrazione si ha:

**Teorema 9.5** *Sia $T$ una teoria coerente, contenente $Q$, avente $\mathbf{N}$ tra i suoi modelli, e tale che l'insieme dei numeri di Gödel dei suoi assiomi è ricorsivo totale. Allora $T$ è incompleta.*

**Definizione 9.6** Sia $T$ una teoria nel linguaggio dell'aritmetica. $T$ è $\omega$-coerente se non è possibile che $T$ dimostri un enunciato della forma $\exists x \varphi(x)$ e al tempo stesso, per ogni $n$, $T \vdash \neg\varphi(\overline{n})$.

**Osservazione 9.7** Se $T$ ha come modello $\mathbf{N}$, $T$ è $\omega$-coerente. Se $T$ è $\omega$-coerente, allora $T$ è coerente (affinché una teoria sia coerente basta che vi sia un enunciato non dimostrabile).

**Lemma 9.8** *Sia $T$ una teoria $\omega$-coerente contenente $Q$ e sia $\sigma$ un enunciato $\Sigma^0_1$. Allora $T \vdash \sigma$ se e solo se $\sigma$ è vero nel modello $\mathbf{N}$.*

Dim. Ci limitiamo a considerare il caso in cui $\sigma$ è della forma $\exists x \theta(x)$ con $\theta \in \Delta_0$. Se $\sigma$ è vero in $\mathbf{N}$, allora per qualche $n \in \mathbf{N}$, $\theta(\overline{n})$ è vero in $\mathbf{N}$, e

quindi dimostrabile in $Q$ (essendo $\theta \in \Delta_0$). Ne segue che $\theta(\overline{n})$ è dimostrabile in $T$, e quindi anche $\exists x \theta(x)$ lo è.

Viceversa se $T \vdash \sigma$ e se per assurdo $\sigma$ fosse falso in $\mathbf{N}$, allora per ogni $n$ varrebbe in $\mathbf{N}$ l'enunciato $\neg\theta(\overline{n})$ e pertanto questo enunciato sarebbe dimostrabile in $Q$ (essendo $\Delta_0$) e quindi in $T$, contraddicendo la $\omega$-coerenza. QED

**Teorema 9.9** *Sia $T$ una teoria $\omega$-coerente, contenente $Q$, e tale che l'insieme dei numeri di Gödel dei suoi assiomi è ricorsivo totale. Allora $T$ è incompleta.*

Dim. Poiché le codifiche degli assiomi di $T$ sono un insieme ricorsivo, esiste un insieme ricorsivo **Prov** tale che $T \vdash \varphi$ se e solo se $\exists n : (n, \ulcorner\varphi\urcorner) \in \mathbf{Prov}$. Sia $P(x,y)$ una formula che binumera **Prov** in $Q$. Sia $\mathrm{Teo}(x)$ la formula $\exists d P(d,x)$. Sia $G$ un enunciato tale che $Q$ dimostra $G \leftrightarrow \neg\mathrm{Teo}(\overline{\ulcorner G\urcorner})$.

Se per assurdo $T \vdash G$, allora esiste $n$ tale che $(n, \ulcorner G\urcorner) \in \mathbf{Prov}$. Fissato un tale $n$ si ha $Q \vdash P(n, \overline{\ulcorner G\urcorner})$. Quindi $Q \vdash \exists d P(d, \overline{\ulcorner G\urcorner})$, ovvero $Q \vdash \mathrm{Teo}(\overline{\ulcorner G\urcorner})$. Ma allora per la scelta di $G$, $Q \vdash \neg G$, e poiché $T$ contiene $Q$, $T \vdash \neg G$. Questo è impossibile, in quanto $T$ è coerente.

Abbiamo quindi dimostrato che $G$ non è dimostrabile in $T$, e pertanto per ogni $n$ si ha $(n, \ulcorner G\urcorner) \notin \mathbf{Prov}$, da cui discende $\neg P(\overline{n}, \overline{\ulcorner G\urcorner})$ è dimostrabile in $Q$ e quindi in $T$. Ma allora per la $\omega$-coerenza $T \not\vdash \exists x P(x, \overline{\ulcorner G\urcorner})$, e per definizione di $G$, $T \not\vdash \neg G$. QED

**Teorema 9.10** *Sia $T$ una teoria $\omega$-coerente, contenente $Q$, e tale che l'insieme dei numeri di Gödel dei suoi assiomi è ricorsivo totale. Allora $T$ è indecidibile (nel senso che l'insieme dei suoi teoremi non è ricorsivo). In particolare $PA$ è indecidibile.*

Dim. Sia $K_0 \subset \mathbf{N}$ un insieme ricorsivamente enumerabile e non ricorsivo. Essendo ricorsivamente enumerabile $K_0$ è definibile in $\mathbf{N}$ da una formula $\sigma(x) \in \Sigma_1^0$ in una variabile libera. Per ogni $n \in \mathbf{N}$, $n \in K_0$ se e solo se $\sigma(\overline{n})$ è vera in $\mathbf{N}$, e per la $\omega$-coerenza di $T$, questo avviene se e solo se $T \vdash \sigma(\overline{n})$. Ne segue che se avessimo un algoritmo per stabilire se un enunciato è un teorema di $T$, ne avremmo uno per stabilire se un numero appartiene a $K_0$. QED

**Osservazione 9.11** La incompletezza di $PA$ segue anche direttamente dalla sua indecidibilità, in quanto una teoria completa con un insieme di assiomi ricorsivo è decidibile.

Rosser ha dimostrato che nei teoremi sopra enunciati, l'ipotesi di $\omega$ coerenza di può indebolire nell'ipotesi di semplice coerenza.

Si può dimostrare che la formula $G$ nella dimostrazione del Teorema **??** equivale in $PA$ alla formula $Con(PA)$, definita da $\exists x \neg Teo(x)$, che esprime la coerenza di $PA$. Quindi $PA$ non dimostra la sua propria coerenza.

**Teorema 9.12** $PA \nvdash Con(PA)$.

Dim.(Cenni) Per dimostrare, nella teoria $PA$, l'equivalenza $G \leftrightarrow Con(PA)$, notiamo che l'implicazione da sinistra a destra è ovvia in quanto $G$ implica $\neg Teo(\lceil G \rceil)$, che ovviamente implica $\neg \exists x Teo(x)$. L'implicazione inversa $Con(PA) \rightarrow G$ si ottiene formalizzando in $PA$ la prima parte della dimostrazione del Teorema **??**, quella cioè in cui si mostra $PA \nvdash G$. In tale dimostrazione si usa solo la coerenza di $PA$ (e non che $\mathbf{N}$ sia un modello). Formalizzando il ragionamento si ottiene una dimostrazione, in $PA$, dell'implicazione $Con(PA) \rightarrow \neg Teo(\lceil G \rceil)$. Poiché $\neg Teo(\lceil G \rceil)$ equivale dimostrabilmente a $G$, si ha $PA \vdash Con(PA) \rightarrow G$. QED