Appunti del corso di Elementi di Logica, Calcolabilità e Complessità

A. Berarducci

Versione preliminare, 20 Maggio 2002

Sommario

Questi appunti sono ad uso esclusivo degli studenti del corso e sono ancora INCOMPLETI e da perfezionare. Gli argomenti di calcolabilità si possono trovare in forma più esauriente nel libro di N.J.Cutland, *Computability, An introduction to recursive function theory*, Cambridge University Press 1980.

Indice

1 Macchine a registri

Una macchina a registri è un modello idealizzato di calcolatore proposto da Shepherdson e Sturgis nel 1963. Ci baseremo su tale modello per dare una definizione di funzione calcolabile (le prime definizioni di funzione calcolabile risalgono agli anni 1930-40). A differenza di un calcolatore reale una macchina a registri ha una capacità di memoria potenzialmente illimitata (la memoria di un calcolatore è un dispositivo che ha la funzione di immagazzinare dell'informazione opportunamente codificata). Ogni programma per macchina a registri può essere realizzato anche su un calcolatore reale (e viceversa), ma su quest'ultimo non sarà possibile elaborare dati che superino delle fissate capacità di memoria.

1.1 Registri e configurazione di memoria

Una macchina a registri ha infiniti registri di memoria $R_0, R_1, R_2, \ldots$. In ogni momento del calcolo ogni registro R_i contiene un numero naturale a_i, ma solo un numero finito di essi contengono un numero diverso da zero. La successione dei numeri naturali $(a_0, a_1, a_2, \ldots)$ contenuti nei vari registri in un dato momento viene detta configurazione di memoria.

1.2 Contatore di programma

Una macchina a registri ha anche uno speciale registro chiamato contatore di programma. Il contatore di programma contiene un numero che indica quale sia la prossima istruzione da eseguire. All'inizio del calcolo il contatore contiene il numero 1.

1.3 Stato

Lo stato di una macchina a registri in un dato momento è dato dal contenuto dei registri di memoria e dal contenuto del contatore di programma.

1.4 Istruzioni e programmi

Un programma P per macchina a registri è una lista finita $(I_1, \ldots, I_s)$ di istruzioni. Le istruzioni possono essere dei seguenti tipi.

- $R_n := 0$,
- $R_n := R_n + 1$,
- $R_n := R_m$,
- if $R_n = R_m$ go to q.

Le istruzioni dei primi tre tipi sono dette **istruzioni di assegnazione**, le istruzioni del quarto tipo sono dette **istruzioni di salto condizionato**. Se in un dato momento il contatore di programma contiene il numero i, con $1 \leq i \leq s$, viene eseguita l'istruzione I_i, altrimenti la macchina si arresta. Una istruzione della forma "if $R_n = R_m$ go to $s + 1$", dove s è la lunghezza del programma, equivale quindi ad un comando di arresto.

1.5 Semantica delle istruzioni

L'esecuzione di una istruzione all'interno di un programma comporta un cambiamento dello stato, ovvero della configurazione di memoria e del contatore di programma. Consideriamo innanzitutto il cambiamento della configurazione di memoria:

- L'esecuzione di $R_n := 0$ memorizza il numero 0 nel registro R_n (cancellando i dati precedentemente contenuti in tale registro e lasciando invariato il contenuto degli altri registri).
- L'esecuzione di $R_n := R_n + 1$ incrementa di uno il contenuto del registro R_n.
- L'esecuzione di $R_n := R_m$ memorizza nel registro R_n il contenuto del registro R_m.
- L'esecuzione di una istruzione di salto condizionato non modifica la configuazione di memoria.

Consideriamo ora il cambiamento del contatore di programma:

- L'esecuzione di una istruzioni di assegnazione comporta un incremento di 1 del contenuto del contatore di programma.

- L'esecuzione di una istruzione della forma "if $R_n = R_m$ go to q" modifica il contenuto del contatore di programma come segue. Il contenuto del contatore di programma viene posto uguale a q se i contenuti dei registri R_n ed R_m sono uguali, altrimenti il contatore viene incrementato di 1.

1.6 Semantica dei programmi

1.6.1 Funzioni parziali

1.1 Definizione. Data una relazione $f \subseteq A \times B$ definiamo il suo dominio come l'insieme $\mathrm{dom}(f) = \{a \in A \mid \exists b \in B\ (a,b) \in f\}$, e la sua immagine come l'insieme $\mathrm{Im}(f) = \{b \in B \mid \exists a \in A\ (a,b) \in f\}$. Se per ogni $a \in \mathrm{dom}(f)$ esiste un unico $b \in B$ tale che $(a,b) \in f$, allora diciamo che f è una funzione parziale da A a B, e scriviamo $f\colon A \to B$ per indicare questo fatto. Se inoltre $\mathrm{dom}(f) = A$ diciamo che f è una funzione totale da A a B. Data una funzione parziale $f\colon A \to B$ scriviamo $f(a) = b$ se $(a,b) \in f$. Inoltre, dato un elemento $a \in A$, scriviamo $f(a) \downarrow$ se $a \in dom(f)$, e $f(a) \uparrow$ altrimenti. Due funzioni parziali $f\colon A \to B$ e $g\colon A \to B$ sono uguali se hanno lo stesso dominio $A' \subseteq A$ e se per ogni $a \in A'$ si ha $f(a) = g(a)$.

1.2 Definizione. Date due funzioni parziali $f\colon A \to B$ e $g\colon B \to C$, la loro composizione $g \circ f$ è la funzione parziale $h\colon A \to C$ definita come segue: $h(a) = c$ se e solo se esiste $b \in B$ con $f(a) = b$ e $g(b) = c$. Scriviamo $h(x) = g(f(x))$ per dire che h è la composizione $g \circ f$. Fissato un valore di x, abbiamo $h(x) \downarrow$ se e solo se $x \in \mathrm{dom}(f)$ e $f(x) \in \mathrm{dom}(g)$.

1.3 Esempio. Se $f(x) \uparrow$, allora $(0 \cdot f(x)) \uparrow$.

1.6.2 Trasformazione di memoria determinata da un programma

1.4 Definizione. Un **calcolo** del programma P è una successione finita o infinita $(s_0, s_1, s_2, \ldots)$ di stati. A partire da un dato stato viene eseguita l'istruzione indicata dal contatore di programma. L'esecuzione dell'istruzione determina il successivo stato del calcolo secondo le regole sopra date. Il calcolo ha termine se e quando si raggiunge uno **stato di arresto**, ovvero uno stato in cui il contatore di programma contiene un numero che non corrisponde ad alcuna istruzione del programma.

1.5 Definizione. Indichiamo con $\mathbb{N}^\infty$ l'insieme di tutte le possibili configurazioni di memoria, ovvero di tutte le successioni infinite di numeri naturali che sono zero da un certo punto in poi. Un programma P definisce una funzione parziale

$$\mathbf{P}\colon \mathbb{N}^\infty \to \mathbb{N}^\infty$$

come segue: $\mathbf{P}(c) = c'$ se il calcolo del programma P a partire dallo stato $s = (c, 1)$ avente configurazione di memoria $c \in \mathbb{N}^\infty$ e contatore di programma 1, termina dopo un numero finito di passi in uno stato di arresto avente c' come configurazione di memoria. Se non viene mai raggiunto uno stato di arresto $\mathbf{P}(c) \uparrow$.

1.6 Esempio. Il seguente programma scambia il contenuto dei registri R_1 ed R_2.

1. $R_3 := R_1$;
2. $R_1 := R_2$;
3. $R_2 := R_3$.

1.6.3 Nomi dei registri

Nello scrivere un programma risulta comodo riferirsi ai registri di memoria attraverso dei nomi simbolici, scrivendo ad esempio $x, y, z, \ldots$, anziché $R_1, R_2, R_3, \ldots$. Il programma dell'esempio precedente assume allora la forma:

1. $z := x$;
2. $x := y$;
3. $y := z$.

dove x, y, z sono nomi di registri (da non confondersi con il contenuto dei registri stessi).

1.7 Funzioni calcolabili

1.7 Definizione. Supponiamo di voler scrivere un programma P per macchina a registri per calcolare una funzione da $\mathbb{N}^m$ a $\mathbb{N}^n$. Distinguiamo a tal fine i registri di memoria usati dal programma in tre gruppi.

1. **Registri di input**: contengono, all'inizio del calcolo, i dati di ingresso $(a_1, \ldots, a_m) \in \mathbb{N}^m$ (non ci occupiamo del problema di come vengano forniti alla macchina tali dati: possiamo ad esempio immaginare che vengano digitati da una apposita tastiera).
2. **Registri di output**: conterrano alla fine del calcolo, se questo ha termine, i risultati finali $(b_1, \ldots, b_n) \in \mathbb{N}^n$ a cui siamo interessati (non ci occupiamo del problema di come vengano letti o stampati tali risultati).
3. **Registri di lavoro**: consistono di tutti gli altri registri che vengono usati dal programma e contengono informazioni utili al suo funzionamento. In genere supporremo che all'inizio del calcolo tutti i registri di lavoro contengano il numero zero.

La funzione parziale da $\mathbb{N}^m$ a $\mathbb{N}^n$ calcolata dal programma P, rispetto ad una fissata scelta di m registri di input $x_1, \ldots, x_m$ ed n registri di output $y_1, \ldots, y_n$, viene indicata con la notazione

$$(y_1, \ldots, y_n) := P(x_1, \ldots, x_m) \tag{1}$$

ed è definita dalla composizione

$$\mathbb{N}^m \xrightarrow{\text{Input}_{x_1,\ldots,x_m}} \mathbb{N}^\infty \xrightarrow{\mathbf{P}} \mathbb{N}^\infty \xrightarrow{\text{Output}_{y_1,\ldots,y_n}} \mathbb{N}^n, \tag{2}$$

dove $\text{Input}x_1, \ldots, x_m \colon \mathbb{N}^m \to \mathbb{N}^\infty$ è la funzione che immagazzina i dati di ingresso negli m registri di input, $\mathbf{P} \colon \mathbb{N}^\infty \to \mathbb{N}^\infty$ è la funzione parziale descritta nella Definizione 1.5, e $\text{Output}y_1, \ldots, y_n \colon \mathbb{N}^\infty \to \mathbb{N}^n$ è la funzione che legge dalla configurazione di memoria i dati contenuti negli n registri di output.

1.8 Esempio. Supponendo che x_0, x_1, x_2 siano i primi tre registri di memoria R_0, R_1, R_2, $\text{Input}x_0, x_1, x_2(3,5,4) = (3,5,4,0,0,0,\ldots)$.

Supponendo che y_1, y_2 siano i registri R_0 ed R_3 rispettivamente, abbiamo $\text{Output}y_1, y_2(3,2,7,9,8,0,0,\ldots) = (3,9)$.

1.9 Definizione. Una funzione parziale $f \colon \mathbb{N}^m \to \mathbb{N}^n$ è calcolabile se e solo se coincide con la funzione calcolata da un programma P, rispetto ad una data scelta di m registri di input ed n registri di output.

Nel seguito quando scriveremo un programma per calcolare una funzione da $\mathbb{N}^m$ ad $\mathbb{N}^n$ lo accompagneremo da un commento che dichiara quali siano i registri di input e output (in un vero linguaggio di programmazione tale dichiarazione farebbe parte integrante del programma).

1.10 Esercizio. Si scrivano dei programmi per calcolare la somma e il prodotto di due numeri.

1.11 Esercizio. La funzione $pred(x) = x \dot{-} 1$ (predecessore sugli interi non negativi) è definita da: $pred(0) = 0$ e $pred(n+1) = n$. Si scriva un programma per calcolare $pred$.

Suggerimento: dati due registri a, b si definisca la macroistruzione $(a, b) := (a+1, a)$ come la successione di istruzioni $b := a$; $a := a + 1$. Inizialmente si assegnino ad a, b i valori $0, 0$. Dato un numero x di cui si vuole calcolare il predecessore, si iteri la macroistruzione fino a quando a contiene il numero x. A quel punto b contiene il predecessore di x.

1.8 Concatenazione di programmi

1.12 Proposizione. *Dati due programmi $P = (I_1, \ldots, I_s)$ e $Q = (J_1, \ldots, J_k)$, è possibile costruire un programma $P;Q$, detto la concatenazione di P e Q, il cui effetto è il seguente. Se P e Q definiscono le funzioni parziali sulle configurazioni di memoria $\mathbf{P} \colon \mathbb{N}^\infty \to \mathbb{N}^\infty$, e $\mathbf{Q} \colon \mathbb{N}^\infty \to \mathbb{N}^\infty$, allora $P;Q$ definisce la funzione parziale $\mathbf{Q} \circ \mathbf{P} \colon \mathbb{N}^\infty \to \mathbb{N}^\infty$. In altre parole esequire $P;Q$ equivale ad esequire prima P e poi Q.*

Dimostrazione. Basta concatenare le istruzioni di P e Q ricalcolando gli indirizzi di ritorno delle istruzioni di salto condizionato per tener conto che la numerazione delle istruzioni è cambiata. Più precisamente si definisce $P;Q$ come segue. Se nè P nè Q contengono istruzioni di salto condizionato, allora $P;Q$ si

ottiene semplicemente giustapponendo le istruzioni di Q a quelle di P, ovvero $P;Q = (I_1, \ldots, I_s, J_1, \ldots, J_k)$. In presenza di istruzioni di tipo "if $R_m = R_n$ go to q", si pone $P;Q = (I_1^*, \ldots, I_s^*, J_1^*, \ldots, J_k^*)$ dove I_i^* e J_i^* sono definite come segue. Se I_i e J_i non sono istruzioni di salto condizionato allora $I_i^* = I_i$ e $J_i^* = J_i$. Se J_i è della forma "if $R_m = R_n$ go to q", allora J_i^* è l'istruzione "if $R_m = R_n$ go to $q + s$" (s è il numero di istruzioni di P). Se I_i è della forma "if $R_m = R_n$ go to q", allora $I_i^* = I_i$ eccetto che nel caso in cui $q > s$, nel qual caso I_i^* è l'istruzione "if $R_m = R_n$ go to $s + 1$" (per fare in modo che al termine della esecuzione di P, il calcolo riparta dalla prima istruzione di Q). □

1.9 Programmi con istruzioni etichettate

Per evitare di dover ricalcolare gli indirizzi di rimando delle istruzioni di salto condizionato quando si concatenano due programmi, conviene etichettare le istruzioni e usare istruzioni di salto condizionato del tipo "se $R_i = R_j$ vai a q", dove q è l'etichetta dell'istruzione a cui si rimanda, anziché l'indice numerico che identifica la sua posizione nel programma. Nel seguito useremo liberamente la convenzione delle etichette (che possono essere anche essere dei numeri).

1.10 Diagrammi di flusso

Un altro modo, sostanzialmente equivalente alle etichette, è di usare delle frecce per indicare la prossima istruzione da eseguire. Questa idea dà luogo alla seguente definizione.

1.13 Definizione. Un diagramma di flusso è un modo di rappresentare in forma grafica un programma. Anziché rappresentare un programma come una successione $P = (I_1, \ldots, I_s)$ di istruzioni, si disegna un grafo in cui ogni nodo, eccetto due nodi speciali di INIZIO e FINE, è etichettato da una istruzione di assegnazione (della forma "$R_m := 0$" oppure "$R_n := R_n + 1$" oppure "$R_m := R_m$") o con una istruzione di controllo (della forma "$R_m = R_n$?"). I nodi sono collegati da frecce nel modo seguente. Se un nodo contiene una istruzione di assegnazione, allora da quel nodo parte una freccia che punta al nodo contenente la prossima istruzione da eseguire. Da un nodo contenente una istruzione di controllo della forma "$R_m = R_n$?" partono due freccie etichettate SI e NO, dove la freccia SI punta al nodo contenente la prossima istruzione da eseguire nel caso la condizione di controllo sia verificata (ovvero se in quel momento i registri R_m ed R_n contengono lo stesso numero), e la freccia NO punta al nodo contenente la prossima istruzione da eseguire nel caso la condizione di controllo non sia verificata. (Le condizioni di controllo giocano lo stesso ruolo delle istruzioni di salto condizionato "if $R_m = R_n$ go to q" precedentemente viste.) Dal nodo INIZIO parte una freccia che indica la prima istruzione da eseguire. Se seguendo le freccie si arriva al nodo FINE il calcolo termina. La semantica dei diagrammi di flusso si definisce in modo analogo alla semantica dei programmi. Ogni programma $P = (I_1, \ldots, I_s)$ può essere rappresentato in forma di diagramma di flusso. Viceversa si può dimostrare che per ogni

diagramma di flusso si può scrivere un programma a lui equivalente (nel senso che definisce la stessa trasformazione di memoria).

1.14 Esercizio. 1. Si scriva un programma per calcolare la sottrazione tra due numeri naturali (definita uguale a zero se il numero da sottrarre è maggiore del numero da cui si sottrae).

2. Si scriva un programma per calcolare il massimo tra due numeri.

3. Consideriamo delle istruzioni di salto condizionato che controllino delle disuguaglianze anzichè delle uguaglianze (del tipo "se $R_i \geq R_j$ vai a q"). Si mostri che tali istruzioni possono essere simulate usando le usuali istruzioni di salto condizionato sulle uguaglianze (suggerimento: $a \geq b$ equivale a $\max(a, b) = a$).

4. Si scriva un programma per calcolare il resto e il quoziente di una divisione e se ne dimostri la correttezza.

5. Si scriva un programma per calcolare il massimo comun divisore tra due numeri e se ne dimostri la correttezza.

6. Si scriva un programma per calcolare la funzione fattoriale.

1.11 Uso di funzioni predefinite

1.15 Definizione. All'interno di un programma o di un diagramma di flusso, oltre alle istruzioni base, possiamo usare delle **macroistruzioni** (o **procedure**) che corrispondono a programmi precedentemente definiti. In altre parole, se abbiamo già definito un programma P per calcolare una certa funzione parziale $f\colon \mathbb{N}^m \to \mathbb{N}^n$ usando i registri di input $x_1, \ldots, x_n$ e i registri di output $y_1, \ldots, y_m$, possiamo utilizzare all'interno di un diagramma di flusso la macroistruzione $(y_1, \ldots, y_m) := f(x_1, \ldots, x_n)$ la cui esecuzione ha il seguente effetto: supponendo che $a_1, \ldots, a_m$ siano i contenuti dei registri $x_1, \ldots, x_n$ prima della esecuzione della macroistruzione, allora dopo l'esecuzione della macroistruzione i contenuti dei registri $y_1, \ldots, y_m$ sono dati dal vettore di valori $(b_1, \ldots, b_n) = f(a_1, \ldots, a_m)$, a condizione che $f(a_1, \ldots, a_m) \downarrow$, mentre in caso contrario l'esecuzione dell'istruzione porta ad un calcolo non terminante.

La legittimità dell'uso delle macroistruzioni è espressa dalla seguente proposizione.

1.16 Proposizione. *Un programma che fa uso di macroistruzioni* $(y_1, \ldots, y_m) := f(x_1, \ldots, x_n)$*, dove* $f\colon \mathbb{N}^m \to \mathbb{N}^n$ *è una funzione parziale calcolata da un programma precedentemente definito, può essere trasformato in un programma che non fa uso di macroistruzioni.*

Dimostrazione. Basta sostanzialmente sostituire la macroistruzione con la successione delle istruzioni che la definiscono, prendendosi cura di ridefinire in modo appropriato gli indirizzi a cui rimandano le istruzioni di salto condizionato (come

nella dimostrazione della Propositione 1.12). È però necessario prendere alcune cautele nella scelta dei registri di lavoro. Se infatti la successione di istruzioni che definisce una data macroistruzione usasse come registri di lavoro gli stessi registri che vengono usati nel resto del programma per altri scopi, si potrebbero ovviamente avere degli effetti collaterali indesiderati. A ciò si può porre rimedio cambiando se necessario i registri di lavoro della macroistruzione. □

1.17 Esempio. Sia $f\colon \mathbb{N}^n \to \mathbb{N}$ una funzione definita da un polinomio a coefficienti interi non negativi. Allora f è calcolabile e possiamo quindi usare la macroistruzione $y := f(x_1, \ldots, x_n)$.

2 Funzioni ricorsive parziali

Il nostro prossimo obbiettivo è di dare una caratterizzazione delle funzioni numeriche calcolabili senza far riferimento alle macchine a registri (o ad altr modelli idealizzati di calcolatore). Per funzione numerica intendiamo una funzione parziale da $\mathbb{N}^n$ ad $\mathbb{N}$, per un certo n. Dimostreremo che le funzioni numeriche calcolabili coincidono cone le funzioni che possono essere generate, a partire da certe funzioni iniziali, applicando degli opportuni "operatori". Un operatore è una funzione che porta funzioni parziali in funzioni parziali. Per generare tutte le funzioni numeriche calcolabili ci bastano tre tipi di operatori: la composizione, la ricursione primitiva, e la minimalizzazione. Chi abbia qualche familiarità con la programmazione in linguaggi tipo FORTRAM o PASCAL noterà la somiglianza tra la ricursione primitiva e i costrutti della forma "for $i = 1$ to n, do Q", e tra la minimalizzazione e i costrutti della forma "while P do Q".

2.1 Funzione ricorsiva parziale associata ad un programma

2.1 Definizione. Sia $n \in \mathbb{N}$ e sia P un programma. La funzione parziale $\varphi_P^n\colon \mathbb{N}^n \to \mathbb{N}$ definita da P e da n è la funzione parziale calcolata dal programma P usando come registri di input $R_1, \ldots, R_n$ e come registro di output R_0 (vedi Definizione 1.9). Più esplicitamente:

1. $\varphi_P^n(a_1, \ldots, a_n) = b$ se eseguendo il programma P a partire dallo stato iniziale in cui i primi n registri $R_1, \ldots, R_n$ contengono $a_1, \ldots, a_n$, gli altri registri contengono 0, e il contatore di programma contiene 1, si determina un calcolo che si arresta dopo un numero finito di passi con b nel registro R_0.

2. $\varphi_P^n(a_1, \ldots, a_n) \uparrow$ (indefinita) se partendo dallo stato iniziale sopra definito si determina un calcolo che non si arresta mai.

2.2 Definizione. Sia $f\colon \mathbb{N}^n \to \mathbb{N}$ una funzione parziale. Diciamo che f è ricorsiva parziale (o calcolabile parziale) se e solo se esiste un programma P tale che $f = \varphi_P^n$. Se inoltre $\mathrm{dom}(f) = \mathbb{N}^n$ diciamo che f è ricorsiva totale (o calcolabile totale).

Abbiamo quindi: Funzioni ricorsive totali = Funzioni ricorsive parziali $\cap$ Funzioni totali.

2.2 Composizione di funzioni ricorsive parziali

2.3 Proposizione. *La composizione di funzioni ricorsive parziali è ricorsiva parziale. Più precisamente sia* $f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_k(\vec{x}))$*, dove si intende che* $f(\vec{x})$ *è definita se lo sono tutte le* $g_i(\vec{x})$ *e se* $(g_1(\vec{x}), \ldots, g_k(\vec{x})) \in dom(h)$*. Se* $h, g_1, \ldots, g_k$ *sono funzioni ricorsive parziali, anche* f *è una funzione ricorsiva parziale.*

Dimostrazione. Il seguente programma calcola f, facendo uso di macroistruzioni per calcolare $h, g_1, \ldots, g_k$.

- Input $x_1, \ldots, x_n$
  (questo è un commento per dire che usiamo $x_1, \ldots, x_n$ come registri di input).
- $z_1 := g_1(x_1, \ldots, x_n); \ldots; z_k := g_k(x_1, \ldots, x_n);$
- $y := h(z_1, \ldots, z_k);$
- Output y.

□

2.4 Esempio.
1. $f(x, y, z) = x + y + z$ è calcolabile in quanto si ottiene componendo con se stessa la funzione calcolabile $x + y$.
2. Se $(x, y) \mapsto f(x, y)$ è calcolabile, allora per ogni fissato b, $x \mapsto f(x, b)$ è calcolabile.

2.3 Ricursione primitiva

2.5 Definizione. Date due funzioni parziali $h\colon \mathbb{N}^{n+2} \to \mathbb{N}, g\colon \mathbb{N}^n \to \mathbb{N}$ definiamo una funzione parziale $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ nel modo seguente.

- $f(\vec{x}, 0) = g(\vec{x})$,
- $f(\vec{x}, y + 1) = h(\vec{x}, y, f(\vec{x}, y))$.

dove

- $(\vec{x}, 0) \in \mathrm{dom}(f)$ sse $\vec{x} \in \mathrm{dom}(g)$,
- $(\vec{x}, y + 1) \in \mathrm{dom}(f)$ sse $(\vec{x}, y) \in \mathrm{dom}(f)$ e $(\vec{x}, y, f(\vec{x}, y)) \in \mathrm{dom}(h)$.

Diciamo che f è ottenuta per ricursione primitiva da h e g. Si noti che se h, g sono funzioni totali anche f lo è. Ammettiamo il caso $n = 0$ in cui le variabili $\vec{x}$ sono mancanti. In tal caso g non è una funzione ma un numero $g \in \mathbb{N}$.

2.6 Teorema. *Se h, g sono calcolabili, anche f è calcolabile.*

Dimostrazione. Il seguente programma calcola f usando dei sottoprogrammi per calcolare h e g.

1. Input $x_1, \ldots, x_n, y$;
2. $z := g(\vec{x})$;
3. $k := 0$;
4. se $k = y$ vai al punto 8 (altrimenti vai al prossimo punto);
5. $z := h(\vec{x}, y, z)$;
6. $k := k + 1$;
7. vai al punto 4;
8. Output z.

□

2.3.1 Funzioni primitive ricorsive

2.7 Definizione. Una funzione è primitiva ricorsiva se si ottiene, a partire dalle seguenti funzioni iniziali, per composizione e ricursione primitiva applicate a funzioni precedentemente ottenute.

Funzioni iniziali:

1. La funzione costante zero $c_0 \colon \mathbb{N} \to \mathbb{N}$.
2. La funzione successore $s \colon \mathbb{N} \to \mathbb{N}$.
3. La funzione $U_i^n \colon \mathbb{N}^n \to \mathbb{N}$ che manda $(x_1, \ldots, x_n)$ in x_i.

2.8 Esercizio. Le seguenti funzioni sono primitive ricorsive. Somma, prodotto, sottrazione (definita uguale a zero se il numero da sottrare è più grande del numero da cui si sottrae), esponenziazione, quoziente e resto di una divisione, $|x = y|$, fattoriale, minimo, massimo, massimo comun divisore, minimo comune multiplo.

2.9 Fatto. Le funzioni primitive ricorsive sono un sottoinsieme proprio dell'insieme delle funzioni ricorsive totali.

2.4 Minimalizzazione

La minimalizzazione è un operatore μ che porta funzioni (ricorsive) parziali in funzioni (ricorsive) parziali. Abbiamo già visto un esempio di un simile operatore: la ricursione primitiva. A differenza della ricursione primitiva, la minimalizzazione può generare funzioni non totali anche quando viene applicata a funzioni totali. Definiamo dapprima la minimalizzazione applicata a funzioni totali, per poi vedere il caso generale.

2.10 Proposizione. *Sia* $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ *una funzione totale e sia* $f(\vec{x}) = \mu z(h(\vec{x}, z) = 0)$ *la funzione parziale definita come segue:*

$$f(\vec{x}) = \begin{cases} \min\{z \mid h(\vec{x}, z) = 0\} & \text{se } \exists z(h(\vec{x}, z) = 0) \\ \uparrow & \text{altrimenti} \end{cases}$$

Se h *è ricorsiva (totale), allora* f *è parziale ricorsiva.*

Dimostrazione. Il seguente programma calcola f, usando un sottoprogramma per calcolare h.

1. Input $\vec{x}$;
2. $z := 0$;
3. $u := h(\vec{x}, z)$;
4. se $u = 0$ vai al punto 7;
5. $z := z + 1$;
6. vai al punto 3;
7. Output z.

La totalità di h garantisce che il punto 3 abbia sempre termine e che quindi fino a che non si trovi il valore di z desiderato si continui ad incrementare z di 1. □

2.11 Corollario. *Sia* $h\colon \mathbb{N} \to \mathbb{N}$ *una funzione ricorsiva totale invertibile. Allora l'inversa* $h^{-1}\mathbb{N} \to \mathbb{N}$ *è ricorsiva totale.*

Dimostrazione. Basta osservare che $h^{-1}(y) = \mu z(h(z) = y)$. Questo rientra nello schema della minimalizzazione scrivendo "$h(z) = y$" nella forma "$|h(z) - y| = 0$" (è facile vedere che $|u - v|$ è una funzione ricorsiva, anzi addirittura primitiva ricorsiva). □

Consideriamo ora la definizione dell'operatore di minimalizzazione μ quando venga applicato a funzioni parziali.

2.12 Proposizione. *Sia* $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ *una funzione parziale e sia* $f(\vec{x}, y) = \mu z(h(\vec{x}, z) = 0)$ *la funzione parziale definita come segue:*

$$f(\vec{x}, y) = \begin{cases} \min\{z \mid h(\vec{x}, z) = y \wedge \forall u < z h(\vec{x}, u) \downarrow\} & \text{se tale } z \text{ esiste} \\ \uparrow & \text{altrimenti} \end{cases}$$

Se h *è parziale ricorsiva, allora* f *è parziale ricorsiva.*

Dimostrazione. Il programma per calcolare f è esattamente come nella Proposizione 2.10. Nelle presenti ipotesi h potrebbe non essere totale e quindi il programma potrebbe entrare in una condizione di non terminazione senza che z continui ad essere incrementato. Affinché il programma termini deve quindi accadere non soltanto che esista z tale che $h(\vec{x}, z) = y$, ma anche che il programma non entri in una condizione di non terminazione prima di trovare il minimo tale z. Ciò equivale a dire che deve essere verificata la condizione $\forall u < z h(\vec{x}, u) \downarrow$. □

2.4.1 Funzioni μ-ricorsive di Kleene

2.13 Definizione. Una funzione parziale $f : \mathbb{N}^n \to \mathbb{N}$ è μ-ricorsiva se si ottiene, a partire dalle funzioni iniziali della Definizione 2.7, per composizione, ricursione primitiva e minimalizzazione, applicate a funzioni precedentemente ottenute.

2.14 Teorema. *La classe delle funzioni μ-ricorsive è inclusa nella classe delle funzioni calcolabili da un programma per macchine a registri.*

Dimostrazione. La classe delle funzioni calcolabili con una macchina a registri contiene le funzioni iniziali della Definizione 2.7 e abbiamo visto che gli operatori di composizione, ricursione primitiva e minimalizzazione non fanno uscire da tale classe. □

Vedremo nel seguito che vale anche il viceversa: una funzione parziale $f : \mathbb{N}^n \to \mathbb{N}$ è μ-ricorsiva se e solo se è calcolabile da un programma per macchine a registri.

2.5 Altri operatori ricorsivi

Oltre alla composizione, alla ricursione primitiva, e alla minimalizzazione, ci sono molti altri operatori che applicati a funzioni ricorsive parziali generano funzioni ricorsive parziali. In questa sezione ne consideriamo alcuni particolarmente utili.

2.5.1 Sommatorie e produttorie limitate

2.15 Esercizio. Definiamo

1. $f_1(\vec{x}, y) = \Sigma_{z<y} h(\vec{x}, y)$.
2. $f_2(\vec{x}, y) = \Pi_{z<y} h(\vec{x}, y)$.

Se h è totale ricorsiva, anche f_1, f_2 lo sono. Se h è primitiva ricorsiva, anche f_1, f_2 lo sono.

2.5.2 Minimalizzazione limitata

2.16 Proposizione. *Sia* $f(\vec{x}, y) = \mu z < y(h(\vec{x}, z) = 0)$ *la funzione definita come segue:*

$$f(\vec{x}, y) = \begin{cases} \min\{z \mid z < y \wedge h(\vec{x}, z) = 0\} & \textit{se tale } z \textit{ esiste} \\ y & \textit{altrimenti} \end{cases}$$

Se h *è totale ricorsiva, anche* f *è totale ricorsiva. Se* h *è primitiva ricorsiva, anche* h *è primitiva ricorsiva.*

Dimostrazione. Basta osservare che $f(\vec{x}, y) = \Sigma_{v<y}\Pi_{u<v}\mathrm{sg}(h(\vec{x}, u))$ dove sg è la funzione ricorsiva primitiva definita da $\mathrm{sg}(0) = 0, \mathrm{sg}(x+1) = 1$. □

2.5.3 Ricursione sul decorso dei valori

2.17 Teorema. *Data una funzione* $h\colon \mathbb{N}^2 \to \mathbb{N}$ *sia* $f\colon \mathbb{N} \to \mathbb{N}$ *l'unica funzione tale che per ogni* x, $f(x) = h(x, \langle f(0), \ldots, f(x-1)\rangle)$, *dove* $\langle f(0), \ldots, f(x-1)\rangle \in \mathbb{N}$ *indica la codifica della successione. 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:* $f(\vec{y}, x) = h(\vec{y}, x, \langle f(\vec{y}, 0), \ldots, f(\vec{y}, x-1)\rangle)$

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

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

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. □

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.

3 Tesi di Church

Ci sono due nozioni di funzione calcolabile. Una nozione informale secondo la quale una funzione è calcolabile se esiste un'algoritmo per calcolarla (lasciando indefinito il concetto intuitivo di algoritmo), e una nozione formale secondo la quale una funzione è calcolabile se esiste un programma per macchine a registri per calcolarla. La tesi di Church afferma che la nozione informale coincide con la nozione formale: tutte le funzioni intuitivamente calcolabili sono calcolabili con una macchina a registri (Church faceva riferimento alle macchine di Turing, ma si può dimostrare la loro equivalenza con le macchine a registri). La tesi di Church non è dimostrabile formalmente in quanto appunto collega una nozione formale con una nozione informale. La fiducia nella verità della tesi si basa

sull'esperienza e sul fatto che diverse formalizzazioni della nozione di funzione calcolabile si sono dimostrate equivalenti.

Facendo appello alla tesi di Church possiamo ad esempio dimostrare:

3.1 Proposizione. *Sia $f(n)$ l'n-esima cifra dello sviluppo decimale di π. La funzione f è calcolabile con una macchina a registri.*

Dimostrazione. Usiamo la serie di Hutton per π. Sia

$$h_n = \frac{(n!2^n)^2}{(2n+1)!}\left(\frac{12}{5}\left(\frac{1}{10}\right)^n + \frac{14}{25}\left(\frac{1}{50}\right)^n\right).$$

Sia $s_k = \Sigma_{n\leq k} h_n$. Si ha: $s_k < \pi < s_k + \left(\frac{1}{10}\right)^k$. Questo ci fornisce un algoritmo per trovare le prime k cifre decimali di π e quindi per calcolare f (C'è qualche piccola difficoltà dovuta alla non unicità dello sviluppo decimale nel caso di sviluppi che finiscono con tutti nove da un certo punto in poi, ad esempio $0,9999\ldots = 1$. Tuttavia siccome π non è razionale, in particolare il suo sviluppo decimale non termina con tutti nove, e possiamo quindi trovare $n > k$ tale che lo sviluppo di s_n non termina con tutti nove. Le prime k cifre dello sviluppo di π saranno allora uguali a quelle di s_n). Dalla tesi di Church segue che esiste un programma per macchine a registri per calcolare f. □

Per dimostrare la proposizione senza fare appello alla tesi di Church dobbiamo scrivere esplicitamente il programma per macchine a registri che calcola f.

4 Predicati decidibili

La nozione di "predicato" o "relazione" può essere ricondotta alla nozione di insieme nel modo seguente. Dato un sottoinsieme $M \subseteq \mathbb{N}^m$ scriviamo $M(x_1,\ldots,x_m)$ se e solo se $(x_1,\ldots,x_m) \in M$. Se vale $M(x_1,\ldots,x_m)$ diciamo che $(x_1,\ldots,x_m)$ verifica il predicato (o relazione) M.

4.1 Definizione. Un insieme $A \subseteq \mathbb{N}^k$ è decidibile se la sua funzione caratteristica $\chi_A\colon \mathbb{N}^k \to \{0,1\}$ è calcolabile (χ_A vale 1 su A e zero su $\mathbb{N}^k - A$). Un predicato è decidibile se l'insieme degli elementi che lo verificano è decidibile. Useremo "ricorsivo" come sinonimo di "decidibile".

4.2 Proposizione. *Se $f\colon \mathbb{N}^k \to \mathbb{N}$ è una funzione ricorsiva totale, allora il suo grafico $\{(x_1,\ldots,x_k,y) \mid f(x_1,\ldots,x_k) = y\}$ è decidibile.*

Dimostrazione. La funzione caratteristica del grafico di f è data dalla funzione calcolabile $1 \dot{-} |y - f(x_1,\ldots,x_k)|$. □

4.1 Definizioni per casi

4.3 Teorema. *Definiamo*

$$g(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{se } M_1(\vec{x}) \\ f_2(\vec{x}) & \text{se } M_2(\vec{x}) \end{cases}$$

Se M_1, M_2 sono predicati ricorsivi, e f_1, f_2 sono funzioni calcolabili totali, allora g è una funzione calcolabile totale.

Dimostrazione. $g(\vec{x}) = \chi_{M_1}(\vec{x}) f_1(\vec{x}) + \chi_{M_2}(\vec{x}) f_2(\vec{x})$. □

4.2 Algebra di Boole degli insiemi decidibili

4.4 Proposizione. *L'unione o l'intersezione di due sottoinsiemi decidibili A, B di $\mathbb{N}^k$ è decidibile. Il complemento $\sim A = \mathbb{N}^k - A$ di un sottoinsieme decidibile A di $\mathbb{N}^k$ è decidibile.*

Dimostrazione. $\chi_{A\cap B}(\vec{x}) = \min(\chi_A(\vec{x}), \chi_B(\vec{x}))$, $\chi_{A\cup B}(\vec{x}) = \max(\chi_A(x), \chi_B(\vec{x}))$, $\chi_{\sim A}(\vec{x}) = 1 - \chi_A(\vec{x})$. □

Una forma equivalente della proposizione è la seguente.

4.5 Proposizione. *Dati due predicati decidibili $A(\vec{x})$ e $B(\vec{x})$ (su $\mathbb{N}^n$), la loro congiunzione $A(\vec{x}) \wedge B(\vec{x})$ è un predicato decidibile. Similamente la disgiunzione $A(\vec{x}) \vee B(\vec{x})$ è decidibile, e la negazione $\neg A(\vec{x})$ è decidibile.*

4.3 Quantificatori limitati

4.6 Proposizione. *Definiamo*

- $M_1(\vec{x}, y) \equiv \forall z < y R(\vec{x}, z)$
- $M_2(\vec{x}, y) \equiv \exists z < y R(\vec{x}, z)$

Se R è un predicato decidibile, anche M_1 ed M_2 sono predicati decidibili.

Dimostrazione. Per il primo punto osserviamo che $\chi_{M_1}(\vec{x}, y) = \Pi_{z<y}\chi_R(\vec{x}, z)$. Per il secondo punto definiamo $\mathrm{not}(x) = 1 - x$. Usando l'equivalenza $\exists z < y R(\vec{x}, z) \iff \neg\forall z < y \neg R(\vec{x}, y)$ otteniamo $\chi_{M_2}(\vec{x}, y) = \mathrm{not}\Pi_{z<y}\mathrm{not}\chi_R(\vec{x}, z)$. □

5 Codifiche

Definiamo il concetto di funzione calcolabile su domini diversi da $\mathbb{N}$.

5.1 Definizione. Una **codifica** di un insieme numerabile D è una funzione biunivoca $\alpha\colon D \to \mathbb{N}$, o più in generale una funzione iniettiva $\alpha\colon D \to \mathbb{N}$ la cui immagine sia un sottoinsieme ricorsivo di $\mathbb{N}$. Sia $f\colon D \to D$ una funzione parziale. f è **calcolabile rispetto alla codifica biunivoca** $\alpha\colon D \to \mathbb{N}$ se $\alpha \circ f \circ \alpha^{-1}\colon \mathbb{N} \to \mathbb{N}$ è una funzione ricorsiva parziale.

5.2 Esempio. Codifichiamo $\mathbb{Z}$ associando ad ogni $n \geq 0$ il numero $\alpha(n) = 2n$ e ad ogni $n < 0$ il numero $\alpha(n) = -2n - 1$. Rispetto a questa codifica, le funzioni di somma e prodotto su $\mathbb{Z}$ sono calcolabili, e il predicato $x \geq 0$ è calcolabile.

5.1 Codifica delle stringhe

Un alfabeto è un insieme di simboli. Sia $\Sigma = \{a_1, \ldots, a_k\}$ un alfabeto finito. Sia $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$ l'insieme delle stringhe (successioni finite) su Σ. Indichiamo con $\lambda \in \Sigma^*$ la stringa vuota. Diamo una codifica $\alpha \colon \Sigma^* \to \mathbb{N}$ come segue.

- $\alpha(\lambda) = 0$.
- $\alpha(a_{r_0} \ldots a_{r_m}) = r_0 + r_1 k + \ldots + r_m k^m$.

Possiamo parlare di funzioni calcolabili su stringhe riferendoci a questa codifica.

5.2 Codifica delle coppie

5.3 Definizione. La funzione Coppia: $\mathbb{N}^2 \to \mathbb{N}$ definita da $\text{Coppia}(x, y) = 2^x(1 + 2y) - 1$ è una corrispondenza biunivoca tra $\mathbb{N}^2$ ed $\mathbb{N}$. Inoltre Coppia e le due funzioni inverse sono primitive ricorsive.

5.4 Definizione. La funzione Tripla: $\mathbb{N}^3 \to \mathbb{N}$ definita da $\text{Tripla}(x, y, z) = \text{Coppia}(x, \text{Coppia}(y, z))$ è una corrispondenza biunivoca tra $\mathbb{N}^3$ ed $\mathbb{N}$. Inoltre Tripla e le sue tre funzioni inverse sono primitive ricorsive.

5.3 Codifica delle successioni a supporto finito

Sia $p(x) =$ lo $x + 1$-esimo numero primo. (Quindi $p(0) = 2, p(1) = 3, p(2) = 5, \ldots$.)

5.5 Proposizione. *Il predicato "x è primo" è primitivo ricorsivo. La funzione $x \mapsto p(x)$ è ricorsiva primitiva.*

Dimostrazione. Il predicato "x è primo" è primitivo ricorsivo in quanto può essere espresso a partire da predicati primitivi ricorsivi usando i connettivi booleani e i quantificatori limitati nel modo seguente: $\forall u, v \leq x(uv = x \to u = 1 \vee v = 1)$. Definiamo per ricursione primitiva $p(0) = 2$, $p(n+1) = H(p(n))$ dove $H(x) = \mu y(y \text{ è primo} \wedge y > x)$. □

5.6 Definizione. Sia $(a_i \mid i \in \mathbb{N})$ una successione di numeri naturali. Diciamo che la successione ha supporto finito l'insieme degli i tali che $a_i \neq 0$ è un insieme finito.

5.7 Definizione. Diamo una corrispondenza biunivoca tra numeri naturali e successioni a supporto finito come segue. Alla successione $(a_i \mid i \in \mathbb{N})$ corrisponde il numero $c = \Pi_{i \in \mathbb{N}} p(i)^{a_i}$, detto la codifica della successione. La produttoria è ben definita in quanto tutti i fattori eccetto un numero finito sono uguali ad 1.

Non ha senso chiedersi se la codifica sopra data sia μ-ricorsiva in quanto si tratta di una funzione da $\mathbb{N}^\infty$ ad $\mathbb{N}$. Possiamo però chiederci se la funzione che estrae l'i-esimo elemento da una successione è μ-ricorsiva, rispetto alla codifica data. Più precisamente abbiamo:

5.8 Proposizione. *Sia $(x)_i$ l'esponente di p_i nella scomposizione in fattori primi di x. In altre parole, se $x = \Pi_i p(i)^{a_i}$ codifica la successione a supporto finito $(a_i \mid i \in \mathbb{N})$, allora $(x)_i = a_i$ estrae l'i+1-esimo elemento della successione. La funzione $(x, i) \mapsto (x)_i$, da $\mathbb{N}^2$ ad $\mathbb{N}$, è primitiva ricorsiva.*

Dimostrazione. Possiamo definire per minimalizzazione limitata $(x)_i = \mu u < x \neg(p(i)^{u+1} \mid x)$. □

5.4 Codifica degli insiemi finiti

5.9 Definizione. Diamo una corrispondenza biunivoca tra numeri naturali e insiemi finiti di numeri naturali. All'insieme vuoto associamo 0. Ad un insieme di n elementi $\{a_1, \ldots, a_n\}$ associamo $2^{a_1} + \ldots + 2^{a_n}$.

5.5 Codifica delle successioni finite

5.10 Definizione. Diamo una corrispondenza biunivoca tra numeri naturali e successioni finite di numeri naturali. Ad una successione finita $(a_1, \ldots, a_n)$ di numeri naturali, associamo un numero naturale $\langle a_1, \ldots, a_n \rangle \in \mathbb{N}$, detto la codifica della successione, definito come segue.

- $\langle\rangle = 0$;
- $\langle a_1 \rangle = 2^{a_1}$;
- $\langle a_1, a_2, \ldots, a_n \rangle = 2^{a_1}(1 + 2(\langle a_2, \ldots, a_n \rangle)$

In altre parole: $\langle a_1, a_2, a_3, \ldots, a_n \rangle = 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \ldots + 2^{a_1+a_2+\ldots+a_n+n-1}$.

La biunivocità della codifica si dimostra osservando che $\langle a_1, a_2, a_3, \ldots, a_n \rangle$ è il numero che, nella rappresentazione in base due, inizia con a_0 cifre uguali a zero, seguite dalla cifra 1, seguite da a_1 cifre uguali a zero, seguita da 1, e così via.

5.11 Proposizione. *Esistono funzioni ricorsive primitive* $\mathrm{lh} : \mathbb{N} \to \mathbb{N}$, $\mathrm{Cat} : \mathbb{N}^2 \to \mathbb{N}$, $\pi : \mathbb{N}^2 \to \mathbb{N}$ *tali che:*

- $\mathrm{lh}(\langle a_1, \ldots, a_n \rangle) = n$;
- $\mathrm{Cat}(\langle a_1, \ldots, a_n \rangle, \langle b_1, \ldots, b_k \rangle) = \langle a_1, \ldots, a_n, b_1, \ldots, b_k \rangle$;
- $\pi(i, \langle a_1, \ldots, a_n \rangle) = a_i$ *se* $1 \leq i \leq n$.

Dimostrazione. Definiamo:

- $lh(2^x(1+2y)) = 1 + lh(y)$, $lh(2^x) = 1$, $lh(0) = 0$;
- $Cat(a,b) = a + 2^{lh(a)}b$;
- $\pi(0,n) = 0$, $\pi(1,0) = 0$, $\pi(1, 2^x(1+2y)) = x$, $\pi(n+2, 2^x(1+2y)) = \pi(n+1, y)$.

Lasciamo al lettore la verifica che tali funzioni sono primitive ricorsive. □

Talvolta, dato un insieme numerabile D, risulta più facile trovare una funzione iniettiva da D ad $\mathbb{N}$, che una biunivoca. La seguente proposizione può risultare utile per trovare codifiche biunivoche.

5.12 Esercizio. Sia $A \subset \mathbb{N}$ un insieme infinito ricorsivo. Allora esiste una funzione ricorsiva biunivoca $f\colon A \to \mathbb{N}$.

6 Predicati semidecidibili

6.1 Definizione. Diciamo che $M \subseteq \mathbb{N}^m$ è semidecidibile, o ricorsivamente enumerabile, se e solo se esiste un predicato decidibile $R \subseteq \mathbb{N}^{m+1}$ tale che $M(\vec{x}) \equiv \exists y R(\vec{x}, y)$.

6.2 Proposizione. *Se M è decidibile, è anche semidecidibile.*

6.3 Proposizione. *Se $M \subseteq \mathbb{N}^m$ è un insieme semidecidibile, allora M è il dominio di una funzione ricorsiva parziale $f\colon \mathbb{N}^m \to \mathbb{N}$.*

Dimostrazione. Sia $M(\vec{x}) \equiv \exists y R(\vec{x}, y)$ con R decidibile e definiamo $f(\vec{x}) = \mu y R(\vec{x}, y)$. □

Dimostreremo nel seguito che vale anche il viceversa: un insieme è semidecidibile se e solo se è il dominio di una funzione ricorsiva parziale.

6.1 Operazioni logiche tra predicati semidecidibili

6.4 Teorema. *Se $M(\vec{x}, y_1, \ldots, y_k)$ è un predicato semidecidibile, allora lo è anche $\exists y_1, \ldots, y_k M(\vec{x}, y_1, \ldots, y_k)$.*

Dimostrazione. Essendo semidecidibile $M(\vec{x}, \vec{y})$ può essere scritto nella forma $\exists z R(\vec{x}, y_1, \ldots, y_k, z)$, dove R è decidibile. Dobbiamo mostrare la semidecidibilità del predicato $\exists y_1, \ldots, y_k \exists z R(\vec{x}, y_1, \ldots, y_k, z)$. A tal fine basta scriverlo nella forma equivalente $\exists t(\exists y_1 < t \ldots \exists y_k < t \exists z < t R(\vec{x}, y, z))$ osservando che la parte in parentesi è decidibile. □

6.5 Teorema. *L'unione e l'intersezione di due insiemi semidecidibili $R, S \subseteq \mathbb{N}^m$, è semidecidibile. Di conseguenza la congiunzione e la disgiunzione di due predicati semidecidibili è semidecidibile.*

6.2 Teorema di Post

In contrasto con quanto avviene per i predicati decidibili, il complemento di un predicato semidecidibile non è in generale semidecidibile.

6.6 Teorema. *Sia $A \subseteq \mathbb{N}^m$. Supponiamo che sia A che il suo complemento $\mathbb{N}^m - A$ siano semidecidibili. Allora A è decidibile.*

Dimostrazione. Possiamo scrivere $A(\vec{x}) \equiv \exists y R(\vec{x}, y)$ ed $\neg A(\vec{x}) \equiv \exists y S(\vec{x}, y)$ con R, S decidibili. Sia f la funzione totale ricorsiva $f(\vec{x}) = \mu y(R(\vec{x}, y) \vee S(\vec{x}, y))$. Abbiamo $A(\vec{x}) \equiv R(\vec{x}, f(\vec{x}))$, e quindi A è decidibile. □

6.7 Osservazione. Vale anche il viceversa: se A è decidibile allora sia A che il suo complemento sono semidecidibili (in quanto sono addirittura decidibili).

6.3 Quantificatori limitati su predicati semidecidibili

6.8 Proposizione. *Se $R(\vec{x}, z)$ è un predicato semidecidibile, anche i predicati $\forall z < y R(\vec{x}, z)$ e $\exists z < y R(\vec{x}, z)$ sono semidecidibili.*

Dimostrazione. Sia $R(\vec{x}, z) \equiv \exists t P(\vec{x}, z, t)$, con P ricorsivo.

Dobbiamo mostrare che il predicato $\forall z < y \exists t P(\vec{x}, z, t)$ è semidecidibile. Basta a tal fine scriverlo nella forma equivalente $\exists s(\forall z < y \exists t < s P(\vec{x}, z, t))$, e osservare che la parte in parentesi è decidibile.

Dobbiamo ora mostrare che il predicato $\exists z < y \exists t R(\vec{x}, z, t)$ è semidecidibile. Basta a tal fine scriverlo nella forma equivalente $\exists z \exists t(z < y \wedge R(\vec{x}, z, t))$ e usare il Teorema 6.4 □

Riassumendo: sia i predicati decidibili che quelli semidecidibili sono stabili per congiunzioni, disgiunzioni e quantificatori limitati. I predicati decidibili sono stabili per negazioni. I predicati semidecidibili sono stabili per quantificazione esistenziale.

6.4 Enumerazioni ricorsive degli insiemi semidecidibili

IL seguente teorema giustifica il nome "ricorsivamente enumerabile" come sinonimo di "semidecidibile".

6.9 Teorema. *$A \subseteq \mathbb{N}$ è semidecidibile se e solo se è vuoto oppure è della forma $A = \{f(n) \mid n \in \mathbb{N}\}$ per qualche funzione ricorsiva totale $f : \mathbb{N} \to \mathbb{N}$.*

Dimostrazione. Se A è vuoto è sia semidecidibile che ricorsivamente enumerabile, quindi assumiamo A non vuoto e fissiamo un elemento $a \in A$.

Sia A semidecidibile. Allora $A = \{x \mid \exists y R(x, y)\}$ per qualche predicato ricorsivo R. La funzione binaria

$$g(x, y) = \begin{cases} x & \text{se } R(x, y) \\ a & \text{altrimenti} \end{cases}$$

è calcolabile totale e ha come immagine A. Per finire definiamo una funzione calcolabile totale unaria f che ha la stessa immagine. Basta porre $f(x) = g((x)_0, (x)_1))$, dove $(x)_0$ e $(x)_1$ sono gli esponenti di 2 e 3 nella scomposizione in primi di x (vedi Definizione 5.7).

Per il verso opposto sia $A = Im(f)$ con $f\colon \mathbb{N} \to \mathbb{N}$ calcolabile totale. Allora $y \in A$ se e solo se $\exists x(f(x) = y)$. Il predicato in parentesi è decidibile, quindi A è semidecidibile. □

6.5 Enumerazioni ricorsive crescenti degli insiemi decidibili

6.10 Teorema. *Sia $A \subseteq \mathbb{N}$ un insieme infinito. A è ricorsivo se e solo se esiste una funzione ricorsiva totale crescente $f\colon \mathbb{N} \to \mathbb{N}$ tale che $A = Im(f)$.*

Dimostrazione. Supponiamo che A sia ricorsivo e definiamo $f(0) = \min A$, $f(n+1) = \min x(x \in A \wedge x > f(n))$. Allora f è ricorsiva ed ha A come immagine.

Viceversa supponiamo $A = Im(f)$ con f crescente. Allora $y \in A$ se e solo se $\exists n \leq y(f(n) = y)$. Ne segue che A è ricorsivo. □

6.11 Teorema. *Ogni insieme ricorsivamente enumerabile infinito $A \subseteq \mathbb{N}$ ha un sottoinsieme infinito ricorsivo.*

Dimostrazione. Sia $A = Im(f)$. Definiamo $g(0) = f(0), g(n+1) = f(k(n))$, dove $k(n) = \mu y(f(y) > g(n))$. Poiché g è crescente, l'insieme $Im(g) \subseteq A$ è ricorsivo. □

7 Codifica dei programmi

Fissiamo una codifica Coppia: $\mathbb{N}^2 \to \mathbb{N}$ delle coppie, e una codifica Tripla: $\mathbb{N}^3 \to \mathbb{N}$ delle triple.

7.1 Definizione. Ogni istruzione per macchina a registri può essere codificata con un numero naturale come segue.

Istruzione	Codifica
$R_n := 0$	$4n$
$R_n := R_{n+1}$	$4n + 1$
$R_m := R_n$	$4\text{Coppia}(m, n) + 2$
if $R_m = R_n$ go to q	$4\text{Tripla}(m, n, q) + 3$

Un programma $P = (I_1, I_2, I_3, \ldots, I_n)$ è codificato da $\langle a_1, a_2, a_3, \ldots, a_n\rangle = 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \ldots + 2^{a_1+a_2+\ldots+a_n+n-1}$ (vedi Definizione 5.10), dove $a_i \in \mathbb{N}$ è la codifica dell'istruzione I_i.

La codifica dei programmi sopra descritta è una corrispondenza biunivoca tra programmi e numeri naturali.

7.2 Definizione. Indichiamo con P_e il programma codificato da e. Dato $n \in \mathbb{N}$, sia $\phi_e^n : \mathbb{N}^n \to \mathbb{N}$ la funzione parziale di n argomenti calcolata dal programma P_e usando come registri di input $R_1, \ldots, R_n$ e come registro di output R_0. Sia $\phi_e = \phi_e^1$

7.1 Una funzione non calcolabile

7.3 Teorema. *Sia*

$$f(n) = \begin{cases} \phi_n(n) + 1 & \textit{se } \phi_n(n) \downarrow \\ 0 & \textit{altrimenti} \end{cases}$$

La funzione f non è calcolabile.

Dimostrazione. Se f è calcolabile esiste $e \in \mathbb{N}$ tale che $f = \phi_e$. Per come è definita chiaramente f è una funzione totale. Quindi $\phi_e(e)$ è definita. Di nuovo per la definizione di f abbiamo allora $f(e) = \phi_e(e) + 1$. Questo contraddice $f = \phi_e$. □

Vedremo, come conseguenza del "teorema della funzione universale", che la funzione $n \mapsto \phi_n(n)$ è una funzione calcolabile parziale. Quindi il motivo per cui la funzione sopra definita non è calcolabile deve risiedere nel fatto che il predicato $\phi_n(n) \downarrow$ che discrimina i due casi non è decidibile.

8 Programmi universali

8.1 Forma normale di Kleene

8.1 Definizione. Scriviamo $\phi_e(x_1, \ldots, x_n) \downarrow_{\leq t}$ se e solo se il calcolo di P_e su input $x_1, \ldots, x_n$ termina in al più t passi. Scriviamo "$\phi_e(x_1, \ldots, x_n) \downarrow_{\leq t} = y$" se termina dopo al più t passi con output y (assumiamo che i registri di input siano $R_1, \ldots, R_n$ e il registro di output sia R_0). Usiamo $\phi_e(x_1, \ldots, x_n) \downarrow y$ come notazione equivalente a $\phi_e(x_1, \ldots, x_n) = y$.

8.2 Teorema. *(Kleene) I seguenti insiemi sono primitivi ricorsivi.*

1. $\{(e, x_1, \ldots, x_n, y, t) \mid \phi_e(x_1, \ldots x_n) \downarrow_{\leq t} = y\}$;
2. $\{(e, x_1, \ldots, x_n, t) \mid \phi_e(x_1, \ldots x_n) \downarrow_{\leq t}\}$.

Per dimostrare la ricorsività degli insiemi sopra dati facciamo uso della tesi di Church. Un algoritmo per risolvere il problema è il seguente: si esegua il programma P_e per t passi e si determini se dopo t passi il calcolo è terminato. In caso affermativo si determini il valore di uscita. Daremo nel seguito una dimostrazione che non fa uso della tesi di Church e che fornirà l'ulteriore informazione che gli insiemi dati sono primitivi ricorsivi.

8.3 Teorema. *(Forma normale di Kleene) Sia $n \in \mathbb{N}$. Esiste una funzione primitiva ricorsiva U e un predicato primitivo ricorsivo T^n tale che*

1. $\phi_e^n(x_1, \ldots, x_n) \downarrow$ *sse* $\exists z T^n(e, x_1, \ldots, x_n, z)$.

2. $\phi_e^n(x_1, \ldots, x_n) = U(\mu z T^n(e, x_1, \ldots, x_n, z)$.

Dimostrazione. Definiamo $T^n(e, x_1, \ldots, x_n, z)$ come $\phi_e(x_1, \ldots, x_n) \downarrow_{(z)_1} = (z)_0$, e $U(z) = (z)_0$. □

Il predicato $T^n(e, x_1, \ldots, x_n, z)$ è detto predicato di Turing, ed esprime il fatto che z codifica l'output e il tempo di calcolo del programma P_e su input $x_1, \ldots, x_n$ (quindi se z esiste la macchina si arresta). La funzione U estrae l'output da tale codifica.

8.4 Corollario. *Un insieme $A \subseteq \mathbb{N}^n$ è semidecidibile se e solo se è il dominio di una funzione ricorsiva parziale.*

Dimostrazione. Dal primo punto del Teorema 8.3 segue, fissando e, che il dominio di una funzione parziale ricorsiva è semidecidibile.

Viceversa abbiamo già visto che ogni predicato semidecidibile $\exists z R(\vec{x}, z)$ (con R decidibile) è il dominio di una funzione parziale ricorsiva $f(\vec{x}) = \mu z R(\vec{x}, z)$. □

8.5 Corollario. *Sia $W_x = \mathrm{dom}(\phi_x^1)$. Allora la successione $W_0, W_1, W_2, \ldots$ enumera tutti e soli i sottoinsiemi semidecidibili di $\mathbb{N}$.*

8.6 Corollario. *Una funzione parziale $f\colon \mathbb{N}^k \to \mathbb{N}$ è calcolabile con una macchina a registri se e solo se è μ-ricorsiva.*

Dimostrazione. Sia f calcolabile da una macchina a registri. Sia $e \in \mathbb{N}$ il codice della macchina. Allora $f(\vec{x}) = \phi_e(\vec{x}) = U(\mu z T^n(e, \vec{x}, z))$. Essendo U, T^n primitivi ricorsivi, ne segue che f è μ-ricorsiva. Il verso opposto è il Teorema 2.14 □

8.7 Osservazione. La dimostrazione del corollario mostra che ogni funzione μ-ricorsiva può essere definita usando una sola volta l'operatore di minializzazione.

Fissato $e \in \mathbb{N}$, la funzione $x \mapsto \phi_e(x)$ è ovviamente calcolabile, essendo la funzione calcolata dal programma con codice e. Il seguente risultato mostra che $(e, x) \mapsto \phi_e(x)$ è anch'essa calcolabile. Usando la tesi di Church possiamo ragionare nel modo seguente. Per calcolare tale funzione su input (e, x) si decodifichi e per trovare il programma P_e corrispondente, e poi si calcoli P_e su input x.

8.8 Corollario. *(Funzione universale) La funzione parziale $\psi_U^n\colon \mathbb{N}^{n+1} \to \mathbb{N}$, definita da $\psi_U^n(e, x_1, \ldots, x_n) = \phi_e^n(x_1, \ldots, x_n)$, è ricorsiva parziale.*

Dimostrazione. $\psi_U^n(e, x_1, \ldots, x_n) = U(\mu z T^n(e, x_1, \ldots, x_n, z)$. □

8.9 Esercizio. (non esistenza di un polinomio universale) Fissiamo una codifica $\alpha\colon \mathbb{Z}[x] \to \mathbb{N}$ dei polinomi. Sia $h\colon \mathbb{N}^2 \to \mathbb{N}$ definita da $h(e, x) = p(x)$, dove p è il polinomio con codice e. Si mostri che h non può essere una funzione polinomiale.

8.10 Esercizio. (non esistenza di una funzione primitiva ricorsiva universale) Fissiamo una numerazione $\alpha\colon \mathbb{N} \to F$ delle funzioni primitive ricorsive di un argomento. Sia $h\colon \mathbb{N}^2 \to \mathbb{N}$ definita da $h(e,x) = q(x)$, dove q è la funzione primitiva ricorsiva con codice e. Si mostri che q non può essere una funzione primitiva ricorsiva.

Per dimostrare il Teorema 8.2 abbiamo bisogno di codificare la successione degli stati che costituisce il calcolo di una macchina a registri. Ciò viene fatto nelle sezioni seguenti.

8.2 Dimostrazione del teorema di Kleene

8.11 Definizione. La configurazione di memoria di una macchina a regitri in un dato istante è codificata dal numero $c = \Pi_{i=0}^{\infty} p(i)^{r_i}$, dove $r_i \in \mathbb{N}$ è il contenuto del registro R_i. Affinché la produttoria abbia senso si assume che al più un numero finito di registri contenga un numero diverso da zero.

8.12 Lemma. *Sia I una istruzione per macchina a registri. Definiamo $[I]\colon \mathbb{N} \to \mathbb{N}$ nel modo seguente. Se $c \in \mathbb{N}$ codifica una configurazione di memoria, $[I](c) \in \mathbb{N}$ codifica la nuova configurazione di memoria dopo l'esecuzione di I. Sia $\lceil I \rceil \in \mathbb{N}$ la codifica dell'istruzione I. La funzione $Mem\colon \mathbb{N}^2 \to \mathbb{N}$ definita da $Mem(\lceil I \rceil, x) = [I](x)$ è primitiva ricorsiva.*

Dimostrazione. Definiamo $\mathrm{Mem}(i,x)$ per casi, ricordando la Definizione 7.1.

Se $i = 4n$ (i codifica $R_n := 0$), allora $\mathrm{Mem}(i,x) = \frac{x}{p_0^{(x)_0}} \cdot p_0^{(x)_n}$. Osserviamo che n è una funzione primitiva ricorsiva di i ($n = i/4$).

Se $i = 4n+1$ (i codifica $R_n := R_{n+1}$), $\mathrm{Mem}(i,x) = x \cdot p_n$. Osserviamo che n è una funzione primitiva ricorsiva di i.

Se $i = 4\mathrm{Coppia}(m,n) + 2$ (i codifica $R_m := R_n$), $\mathrm{Mem}(i,x) = \frac{x}{p_m^{(x)_m}} \cdot p_m^{(x)_n}$. Osserviamo che m, n sono funzioni primitive ricorsive di i.

Se $i = 4\mathrm{Tripla}(m,n,q)+3$ (i codifica if $R_m = R_n$ go to q), allora $\mathrm{Mem}(i,x) = x$.

I quattro casi possono essere distinti da predicati primitivi ricorsivi $P_0, \ldots, P_3$ (dove $P_k(i) \iff i$ è congruo ad k modulo 4), e le funzioni coinvolte nei vari casi sono primitive ricorsive. Possiamo concludere che Mem è primitiva ricorsiva. □

8.13 Definizione. Ricordiamo che lo stato di una macchina a registri in un dato istante è dato dalla configurazione di memoria e dal contenuto del contatore di programma. Codifichiamo lo stato con il numero $\mathrm{Coppia}(c,j) \in \mathbb{N}$, dove c codifica la configurazione di memoria e j è il contenuto del contatore di programma.

8.14 Lemma. *Il predicato "s codifica uno stato di arresto del programma P_e" è un predicato primitivo ricorsivo nelle variabili e, s.*

Dimostrazione. Dato $s = \mathrm{Coppia}(c,j)$, possiamo ottenere c, j come funzioni primitive di s (ad esempio $j = \mu x(\exists c', j' \leq s(s = \mathrm{Coppia}(c', j') \wedge x = j'))$). Ora

basta notare che s codifica uno stato di arresto se e solo se $j = 0$ o $j \geq lh(e)$, dove $lh(e)$ è la lunghezza del programma P_e. □

8.15 Lemma. *Sia I una istruzione per macchina a registri. Definiamo Next: $\mathbb{N}^2 \to \mathbb{N}$ nel modo seguente. Se $s = Coppia(c, j) \in \mathbb{N}$ codifica uno stato non di arresto, $s' = Next(e, s) \in \mathbb{N}$ è la codifica del nuovo stato dopo l'esecuzione della j-esima istruzione del programma P_e. Se s codifica uno stato di arresto $Next(e, s) = s$. La funzione Next è primitiva ricorsiva.*

Dimostrazione. Possiamo dare una definizione primitiva ricorsiva di $\text{Next}(e, s)$ nel modo seguente. Osserviamo che, ponendo $s = \text{Coppia}(c, j)$, possiamo ricavare c, j come funzioni ricorsive di s. Quindi nella definizione di $\text{Next}(e, s)$ possiamo liberamente usare c, j. Distinguamo i seguenti casi.

Se $j = 0$ o $j > lh(e)$ (cioè s codifica uno stato di arresto) allora $\text{Next}(e, s) = s$.

Se invece $1 \leq j \leq lh(e)$, distinguiamo i seguenti sottocasi, ricordando che $\pi(j, e)$ estrae il codice della j-esima istruzione dal programma P_e.

- Se $\pi(j, e) = 4n$ (cioè se la j-esima istruzione è $R_n := 0$), allora $\text{Next}(e, s) = \text{Coppia}([R_n := 0](c), j+1)$ (per il Lemma 8.12 = $\text{Coppia}(\text{Mem}(\pi(j, e), c), j+1)$, una funzione primitiva ricorsiva di e, s);
- Se $\pi(j, e) = 4n+1$ (la j-esima istruzione è $R_n := R_n+1$), allora $\text{Next}(e, s) = \text{Coppia}([R_n := R_m](c), j + 1)$;
- Se $\pi(j, e) = 4\text{Coppia}(m, n) + 2$ (la j-esima istruzione è $R_n := R_m$), allora $\text{Next}(e, s) = \text{Coppia}([R_n := R_m](c), j + 1)$;
- Se $\pi(j, e) = 4\text{Tripla}(m, n, q) + 3$ (la j-esima istruzione è if $R_n = R_m$ go to q), allora

$$\text{Next}(e, s) = \begin{cases} \text{Coppia}(c, q) & \text{se } (c)_m = (c)_m \\ \text{Coppia}(c, j + 1) & \text{altrimenti} \end{cases}$$

I predicati che distinguono i casi sono predicati primitivi ricorsivi in e, s. Le funzioni coinvolte nei vari casi sono anch'esse primitive ricorsive in e, s. Ne concludiamo che Next è primitiva ricorsiva. □

8.16 Lemma. *Sia $Input^n : \mathbb{N}^n \to \mathbb{N}$ la funzione così definita. $Input^n(a_1, \ldots, a_n) = s$ se e solo se s codifica lo stato iniziale di una macchina registri su input $a_1, \ldots, a_n$, ovvero lo stato in cui il contatore di programma contiene 1, i registri di memoria $R_1, \ldots, R_n$ contengono rispettivamente i numeri $a_1, \ldots, a_n$, e gli altri registri contengono 0. La funzione Input è primitiva ricorsiva.*

Dimostrazione. Per n fissato, la funzione $h(a_1, \ldots, a_n) = \langle a_1, \ldots, a_n \rangle$ è primitiva ricorsiva. Definiamo $\text{Input}^n(a_1, \ldots, a_n) = \text{Coppia}(h(a_1, \ldots, a_n), 1)$. □

8.17 Lemma. *Sia $n \in \mathbb{N}$ e sia $Stato^n : \mathbb{N}^{n+2} \to \mathbb{N}$ la funzione definita nel modo seguente. $Stato(e, x_1, \ldots, x_n, t) = s'$ se s' codifica lo stato in cui si trova, dopo t passi di calcolo, una macchina con programma P_e che viene fatta partire con input $x_1, \ldots, x_n$. La funzione Stato è primitiva ricorsiva.*

Dimostrazione. . Definiamo per ricursione primitiva $\text{Stato}(e, x_1, \ldots, x_n, 0) = \text{Input}(x_1, \ldots, x_n)$, $\text{Stato}(e, x_1, \ldots, x_n, t+1) = \text{Next}(e, \text{Stato}(e, x_1, \ldots, x_n, t))$. □

8.18 Lemma. *Sia Output*: $\mathbb{N} \to \mathbb{N}$ *la funzione così definita.* $\mathit{Output}(s) = n$ *se e solo se* s *codifica uno stato contenente* n *nel registro di output* R_0*. La funzione Output è primitiva ricorsiva.*

Dimostrazione. $\text{Output}(s) = n$ se e solo se $s = \text{Coppia}(c, i)$ e $n = (c)_0$. □

8.19 Teorema. *Il predicato* $\phi_e(x_1, \ldots, x_n) \downarrow_{\leq t}$ *è un predicato primitivo ricorsivo nelle variabili* $e, x_1, \ldots, x_n, t$*.*

Dimostrazione. Tale predicato equivale a: "$\text{Stato}(e, x_1, \ldots, x_n, t)$ codifica uno stato di arresto". □

8.20 Teorema. *Il predicato* $\phi_e(x_1, \ldots, x_n) \downarrow_{\leq t} = y$ *è un predicato primitivo ricorsivo nelle variabili* $e, x_1, \ldots, x_n, t, y$*.*

Dimostrazione. Tale predicato equivale alla congiunzione $\phi_e(x_1, \ldots, x_n) \downarrow_{\leq t}$ e $y = \text{Output}(\text{Stato}(e, x_1, \ldots, x_n, t))$. □

9 Il problema della fermata

9.1 Definizione. Sia $K_0 = \{x \mid \phi_x(x) \downarrow\}$ e $K = \{(x, y) \mid \phi_x(y) \downarrow\}$.

9.2 Proposizione. *K e K_0 sono semidecidibili.*

Dimostrazione. $(x, y) \in K$ se e solo se $\exists t(\phi_x(x) \downarrow_{\leq t})$, e poiché il predicato in parentesi è decidibile otteniamo la semidecidibilità di K. Inoltre $x \in K_0$ se e solo se $(x, x) \in K$, quindi anche K_0 è semidecidibile. □

9.3 Teorema. *(Indecidibilità del problema della fermata) K e K_0 non sono decidibili.*

Dimostrazione. Sia $W_x = \text{dom}(\phi_x^1) \subseteq \mathbb{N}$. Si osservi che $x \in K_0 \iff x \in W_x$. Se K_0 fosse decidibile lo sarebbe anche il suo complemento, e quindi il predicato $x \notin W_x$ sarebbe decidibile. Ma allora a maggior ragione $\{x \mid x \notin W_x\}$ sarebbe semidecidibile e coinciderebbe dunque con W_e per qualche e. Avremmo allora $x \notin W_x \iff x \in W_e$. Ponendo $x = e$ si raggiunge un assurdo. Quindi K_0 non è decidibile.

D'altra parte se K fosse decidibile lo sarebbe anche K_0 in quanto $x \in K_0$ se e solo se $(x, x) \in K$. □

Un problema strettamente connesso al problema della fermata è il problema della totalità.

9.4 Teorema. *L'insieme* $TOT = \{x \mid \phi_x$ *è totale* $\}$ *non è decidibile.*

Dimostrazione. Definiamo per casi

$$f(n) = \begin{cases} \phi_n(n) + 1 & \text{se } \phi_n \text{ è totale} \\ 0 & \text{altrimenti} \end{cases}$$

Si osservi che f è totale. Inoltre f non è calcolabile, perchè se $f = \phi_e$ otterremmo $f(e) = f(e) + 1$. D'altra parte, per il teorema della funzione universale, $n \mapsto \phi_n(n)$ è calcolabile. Ne segue che il predicato che distingue i due casi non è decidibile. □

Possiamo rafforzare il teorema:

9.5 Teorema. *L'insieme* $TOT = \{x \mid \phi_x \text{ è totale } \}$ *non è semidecidibile.*

Dimostrazione. Se TOT fosse semidecidibile sarebbe l'immagine di una funzione totale ricorsiva $h\colon \mathbb{N} \to \mathbb{N}$. Sia $f(x) = \phi_{h(x)}(x) + 1$. Visto che $h(x) \in TOT$, f è totale. Inoltre per il teorema della funzione universale f è ricorsiva. Ne segue che $f = \phi_e$ con $e \in TOT = \mathrm{Im}(h)$. Sia $e = h(n)$. Per definizione di f abbiamo $f(n) = \phi_{h(n)}(n) + 1$, ma d'altra parte per definizione di n abbiamo $f(n) = \phi_{h(n)}$, da cui l'assurdo $f(n) = f(n) + 1$. (Si noti che questa uguaglianza non produrrebbe una contraddizione se f non fosse definita in n.) □

Osserviamo che mentre i predicati semidecidibili si possono scrivere nella forma $\exists y R(\vec{x}, y)$ con R decidibile, per il predicato $x \in TOT$ serve un quantificatore in più: $x \in TOT$ se e solo se $\forall u \exists t(\phi_x(u) \downarrow_{\leq t})$, dove il predicato in parentesi è decidibile.

10 Riducibilità molti-uno

10.1 Definizione. Siano $A \subseteq \mathbb{N}^m, B \subseteq \mathbb{N}^n$. Diciamo $A \leq_m B$ (A si riduce a B tramite una riduzione molti-uno) se esiste $f\colon \mathbb{N}^m \to \mathbb{N}^n$ calcolabile totale (f è calcolabile se lo sono tutte le sue n funzioni componenti $f_i\colon \mathbb{N}^m \to \mathbb{N}$), tale che $\forall x \in \mathbb{N}^m$ si ha $x \in A$ sse $f(x) \in B$.

10.2 Teorema. *Se* $A \leq_m B$ *e* B *è decidibile, allora* A *è decidibile.*

Dimostrazione. $\chi_A(x) = \chi_B(fx)$. □

Intuitivamente, se $A \leq_m B$, allora B è almeno altrettanto complicato di A.

10.3 Corollario. *Se* $A \leq_m B$ *e* A *è indecidibile, allora* B *è decidibile. In particolare se* $K_0 \leq_m B$*, allora* B *è indecidibile.*

10.4 Esempio. $K_0 \leq_m K$.

Dimostrazione. Sia $g\colon \mathbb{N} \to \mathbb{N}^2$, $x \mapsto (x, x)$. Abbiamo $x \in K_0$ sse $g(x) \in K$. □

Quindi la indecidibilità di K segue da quella di K_0.

10.5 Osservazione. Un difetto della riducibilità molti-uno, è che in generale un insieme non si riduce molti-uno al suo complemento, nonostante a livello intuitivo se abbiamo un metodo per decidere l'appartenenza ad un insieme abbiamo anche un metodo per decidere l'appartenenza al suo complemento. Esistono altri tipi di riduzione tra problemi che non hanno questo difetto (riduzione di Turing).

10.6 Definizione. Un insieme B è r.e.-completo, se è ricorsivamente enumerabile (= semidecidibile) e se per ogni insieme ricorsivamente enumerabile A si ha $A \leq_m B$.

Vedremo che esistono insiemi r.e.-completi: K e K_0 sono due esempi. Per dimostrarlo avremo bisogno del "teorema s.m.n.".

11 Il teorema s-m-n e la completezza del problema della fermata

Il teorema s-m-n è un utile strumento per dimostrare la riducibilità molti-uno tra vari problemi.

11.1 Teorema. *Esiste una funzione totale ricorsiva* $s\colon \mathbb{N}^2 \to \mathbb{N}$ *tale che, per ogni* x, y,

$$\phi_e^2(x,y) = \phi^1_{s(e,x)}(y)$$

Più in generale, dati m, n, *esiste una funzione totale ricorsiva* $s\colon \mathbb{N}^{m+1} \to \mathbb{N}$ *tale che*

$$\phi_e^{m+n}(\vec{x},\vec{y}) = \phi^n_{s(e,\vec{x})}(\vec{y})$$

dove $\vec{x} = (x_1,\ldots,x_m) \in \mathbb{N}^m$ *e* $\vec{y} = (y_1,\ldots,y_n) \in \mathbb{N}^n$.

(In effetti s *si può scegliere addirittura primitiva ricorsiva.)*

Dimostrazione. Dimostriamo il teorema nel caso $m = n = 1$, il caso generale è analogo. É chiaro che, fissato x, la funzione di un argomento $y \mapsto \phi_e^2(x,y)$ è calcolabile, e quindi esisterà un programma P_a, dipendente da x e da e, che calcola tale funzione. Il teorema asserisce che a si può trovare come funzione totale ricorsiva di x ed e.

Veniamo ai dettagli. Ricordiamo che ϕ_e^2 è la funzione definita dal programma P_e usando come registri di input R_1 ed R_2 e $\phi_a^1(x)$ è la funzione definita dal programma P_a usando come registro di input R_1 (in entrambi i casi il registro di output è R_0). Definiamo $s(e,x)$ come la codifica del programma seguente:

- Input R_1 (questo è un commento, non fa parte del programma);
- $R_2 := R_1$;
- $R_1 := x$;
- P_e

dove P_e è il programma con codice e e $R_1 := x$ è una macroistruzione il cui effetto è di memorizzare il numero x nel registro R_1.

La codifica di questo programma è data da una funzione primitiva ricorsiva applicata alle codifiche delle macroistruzioni che lo compongono. Per finire basta osservare che la codifica della macroistruzione $R_1 := x$ è una funzione ricorsiva primitiva di x. (Per $x > 0$ possiamo definire induttivamente $R_1 := x$ come $R_1 := x - 1; R_1 := R_1 + 1$.) □

11.2 Corollario. *Data una funzione parziale ricorsiva $f(\vec{x}, \vec{y})$ esiste una funzione totale ricorsiva $h(\vec{x})$ tale che $f(\vec{x}, \vec{y}) = \phi_{h(\vec{x})}(\vec{y})$.*

Dimostrazione. Sia $f(\vec{x}, \vec{y}) = \phi_e^{m+n}(\vec{x}, \vec{y})$ e sia $h(\vec{x}) = s(e, \vec{x})$. □

11.3 Teorema. *(Completezza del problema della fermata) Sia A un insieme semidecidibile. Allora $A \leq_m K_0$.*

Dimostrazione. Abbiamo $x \in A \iff \exists y R(x, y)$, per un certo predicato ricorsivo R. Definiamo la funzione parziale ricorsiva $f(x, y) = \mu y R(x, y)$. Allora $f(x, y) \downarrow$ (per ogni y) se e solo se $x \in A$. Per il teorema s-m-n esiste una funzione ricorsiva totale h tale che $f(x, y) = \phi_{h(x)}(y)$. Ne segue che $x \in A$ se e solo se $\phi_{h(x)}(h(x)) \downarrow$, e cioè se e solo se $h(x) \in K_0$. □

12 Equivalenza tra programmi

12.1 Teorema. *L'insieme $\{x \mid \phi_x = 0\}$ è indecidibile.*

Dimostrazione. Definiamo

$$f(x, y) = \begin{cases} 0 & \text{se } \phi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema della funzione universale $x \mapsto \phi_x(x)$ è calcolabile. Per il teorema s-m-n esiste una funzione ricorsiva totale $k \colon \mathbb{N} \to \mathbb{N}$ tale che $f(x, y) = \phi_{k(x)}(y)$. Abbiamo $\phi_x(x) \downarrow$ sse $\phi_{k(x)} = 0$. Quindi la funzione k è una riduzione di K_0 a $\{x \mid \phi_x = 0\}$, mostrando che quest'ultimo è indecidibile. □

12.2 Corollario. *L'insieme $\{(x, y) \mid \phi_x = \phi_y\}$ è indecidibile.*

Dimostrazione. Sia e tale che ϕ_e è la funzione costante zero. Allora $x \in \mathbb{K}_0 \iff (x, e) \in \{(x, y) \mid \phi_x = \phi_y\}$. Pertanto la funzione $h(x) = (x, e)$ riduce K_0 all'insieme dato. □

12.3 Definizione. Sia $A \subseteq \mathbb{N}$. Diciamo che A è estensionale se $\phi_x = \phi_y$ e $x \in A$ implica $y \in A$.

Ogni insieme estensionale determina una classe di funzioni calcolabili e viceversa.

12.1 Teorema di Rice

12.4 Teorema. *(Rice) Sia $A \subseteq \mathbb{N}$ un insieme estensionale diverso da $\emptyset$ e da $\mathbb{N}$. Allora A è indecidibile.*

Dimostrazione. Sia P_e un programma per la funzione sempre indefinita. Distinguiamo due casi seconda che $e \in A$ o $e \notin A$.

Primo caso. Supponiamo $e \notin A$. Scegliamo $a \in A$ e dia $g(x) = \phi_a(x)$. Definiamo

$$f(x,y) = \begin{cases} g(y) & \text{se } \phi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il teorema s-m-n (e il teorema della funzione universale) esiste una funzione calcolabile totale $k\colon \mathbb{N} \to \mathbb{N}$ tale che $f(x,y) = \phi_{k(x)}(y)$. Abbiamo $\phi_x(x) \downarrow$ sse $\phi_{k(x)} = g$. Ne segue che $\phi_x(x) \downarrow$ sse $k(x) \in A$. Abbiamo così ridotto K_0 ad A, e quindi A è indecidibile.

Secondo caso. Supponiamo $e \in A$. Sia B il complemento di A. Ragionando come nel primo caso mostriamo che B è indecidibile. Ma allora anche A è indecidibile. □

13 Il secondo teorema di punto fisso

Vedi Cutland.

14 Gerarchia aritmetica

14.1 Insiemi aritmetici

14.1 Definizione. Un predicato $P(\vec{x})$ è aritmetico se e solo se equivale ad un predicato della forma $Q_1 y_1 \ldots Q_n y_n R(\vec{x}, y_1, \ldots, y_n)$, dove ogni Q_i è un quantificatore universale od esistenziale ed R è un predicato ricorsivo.

Possiamo classificare la complessità dei predicati aritmetici contando il numero dei quantificatori davanti ad un predicato ricorsivo. Più precisamente conviene contare il numero di alternanze tra quantificatori universali ed esistenziali, in quanto è facile vedere che un blocco di quantificatori successivi dello stesso tipo (tutti universali o tutti esistenziali) può essere sostituito da un solo quantificatore di quel tipo usando una codifica delle successioni finite. Ad esempio $\forall y_1 \forall y_2 R(\vec{x}, y_1, y_2)$ equivale a $\forall y R(\vec{x}, (y)_1, (y)_2)$, dove $(y)_i$ è l'esponente di p_i nella scomposizione in primi di y (possiamo pensare ad $(y)_i$ come all'i-esimo elemento della successione codificata da y). Diamo quindi la definizione seguente:

14.2 Definizione. Un predicato $P(\vec{x})$ è Σ^0_n se e solo se si può ottenere da un predicato ricorsivo premettendo ad esso n quantificatori (o blocchi di quantificatori dello stesso tipo) di cui il primo, quello più esterno, esistenziale. I predicati Π^0_n sono definiti in modo analogo, ma con il primo quantificatore universale.

Analoghe definizioni si applicano agli insiemi, identificando un predicato con l'insieme degli elementi che lo verificano.

14.3 Osservazione. Gli insiemi ricorsivamente enumerabili sono esattamente quelli Σ^0_1. Gli insiemi Π^n_0 sono i complementi degli insemi Σ^0_n. L'intersezione $\Pi^0_1 \cap \Sigma^0_1$ è costituita dagli insiemi ricorsivi. Valgono le inclusioni $\Sigma^0_n \cup \Pi^0_n \subseteq \Sigma^0_{n+1} \cap \Pi^0_{n+1}$. L'unione $\bigcup_n \Sigma^0_n = \bigcup_n \Pi^0_n$ coincide con la classe di tutti gli insiemi aritmetici.

14.2 Insiemi definibili nella struttura $(\mathbb{N}; +, \cdot)$

Indichiamo con $(\mathbb{N}; +, \cdot)$ la struttura dei numeri naturali con le funzioni di addizione e moltiplicazione.

14.4 Definizione. Un predicato definibile (al primo ordine) nella struttura $(\mathbb{N}; +, \cdot)$ è un predicato $P(x_1, \ldots, x_n)$ che può essere espresso da una formula in cui possono comparire esclusivamente i simboli $+, \cdot$, gli elementi di $\mathbb{N}$, le parentesi, le variabili, i connettivi booleani, e il quantificatore universale $(\exists x)$ ed esistenziale $(\forall x)$, dove tutte le variabili denotano numeri naturali. Analoghe definizioni valgono per altre strutture al posto di $(\mathbb{N}; +, \cdot)$.

14.5 Esempio. La relazione $\leq$ è definibile in $(\mathbb{N}; +, \cdot)$. Infatti $x \leq y$ se e solo se $\exists z(x + z = y)$. Quindi i predicati definibili in $(\mathbb{N}; +, \cdot)$ coincidono con quelli definibili in $(\mathbb{N}, +, \cdot, \leq)$.

14.6 Esempio. Il predicato "x è primo" è definibile tramite la formula "$\forall u, v(x = u \cdot v \to u = 1 \vee v = 1)$".

14.7 Definizione. Un insieme è definibile se è della forma $\{(a_1, \ldots, a_n) \mid P(a_1, \ldots, a_n)\}$, dove P è un predicato definibile. Una funzione è definibile se il suo grafico è definibile.

L'esempio mostra che l'insieme dei primi è definibile in $(\mathbb{N}; +, \cdot)$. Mostreremo che ogni predicato ricorsivo è definibile in $(\mathbb{N}; +, \cdot)$. Ne seguirà che gli insiemi definibili in tale struttura coincidono con i predicati aritmetici.

14.3 Funzione β di Gödel

Per mostrare che ogni predicato ricorsivo è definibile in $(\mathbb{N}; +, \cdot)$, avremo bisogno di un modo di codificare le successioni che sia definibile in questa struttura. La codifica basata sulla scomposizione in primi è per il momento inutilizzabile in quanto richiede l'esponenziazione che ancora non abbiamo mostrato essere definibile. Una codifica alternativa è basata sul seguente:

14.8 Teorema. *(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.)

Dimostrazione. 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 α, β 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 $x = b_0x_0 + b_1x_1 + \ldots + b_nx_n$. □

Per l'applicazione a cui siamo interessati i moduli a_i saranno forniti dal seguente lemma.

14.9 Lemma. *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.*

Dimostrazione. È sufficiente prendere $d = y!$ dove $y > \max(n, x)$. Se infatti un numero primo 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. □

14.10 Definizione. 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 β di Gödel, $\beta\colon \mathbb{N}^3 \to \mathbb{N}$, è definita da: $\beta(c, d, i) = re(c, (i+1)d+1)$.

14.11 Osservazione. Il grafico della funzione β è definabile in $(\mathbb{N}; +, \cdot)$. Infatti $\beta(c, d, i) = r \leftrightarrow \exists q(c = q[(i+1)d+1] + r \wedge 0 \leq r < (i+1)d+1)$. Si noti che il quantificatore esistenziale $\exists q$ può in effetti essere equivalentemente sostituito dal quantificatore limitato $\exists q \leq c$. Pertanto il grafico della funzione β è definibile da una formula con quantificatori limitati. Questa osservazione risulterà utile nel seguito.

La proprietà fondamentale della funzione β di Gödel è espressa dal seguente:

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

Dimostrazione. 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 $i < n$. Poiché $b_i < (i+1)d+1$, ne segue che b_i è il resto della divisione di c per $(i+1)d+1$, ovvero $b_i = \beta(c, d, i)$. □

La funzione β può essere interpretata nel modo seguente: $\beta(c, d, i) =$ l'i-esimo elemento della successione codificata da (c, d).

14.4 Gli insiemi ricorsivamente enumerabili sono definibili in $(\mathbb{N}; +, \cdot)$

14.13 Teorema. *Ogni funzione ricorsiva parziale è definibile in* $(\mathbb{N}; +, \cdot)$.

Dimostrazione. Ricordiamo che le funzioni ricorsive parziali sono la più piccola classe di funzioni parziali contenente alcune funzioni iniziali (la funzione costante zero, il successore, e le funzioni proiezione), e chiusa rispetto alla composizione, alle definizioni per ricursione primitiva, e alla minimalizzazione.

Lasciamo al lettore la verifica che le funzioni iniziali sono definibili (nella struttura $(\mathbb{N}; +, \cdot)$).

Mostriamo che le funzioni definibili sono chiuse per composizione: 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 definiscono tali funzioni, otteniamo una formula per 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 β 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 β e le funzioni g ed h sono definibili, si ottiene per sostituzione una formula che definisce f.

Chiusura per minimalizzazione

Sia $f(\vec{x}) = \mu y g(\vec{x}, y) = 0$. Osserviamo che $f(\vec{x}) = y$ se e solo se $g(\vec{x}, y) = 0 \wedge \forall i < y \exists v \neq 0 \; g(\vec{x}, i) = v$. Assumendo che g sia definibile, da quest'ultima espressione ricaviamo una formula che definisce il grafico di f. □

14.14 Definizione. Una formula aritmetica si dice Δ_0 o limitata, se tutti i quantificatori che vi compaiono sono limitati. Una formula aritmetica si dice $\vec{\exists}\Delta^0$, se è ottenuta premettendo un certo numero di quantificatori esistenziali davanti ad una formula Δ_0.

Analizzando la dimostrazione del teorema precedente si ottiene:

14.15 Osservazione. Ogni funzione ricorsiva parziale è definibile in $(\mathbb{N}; +, \cdot)$ da una formula $\vec{\exists}\Delta_0$.

14.16 Corollario. *Ogni insieme ricorsivamente enumerabile è definibile in* $(\mathbb{N}; +, \cdot)$. *Inoltre la formula che lo definisce può essere presa di complessità* $\vec{\exists}\Delta_0$.

Dimostrazione. Sia $A \subseteq \mathbb{N}^k$ ricorsivamente enumerabile. Allora $A = dom(g)$ per una certa g ricorsiva parziale. Abbiamo $\vec{x} \in A$ se e solo se $\exists y \in \mathbb{N}(g(\vec{x}) = y)$. Per i risultati precedenti il predicato $g(\vec{x}) = y$ è definibile in $(\mathbb{N}; +, \cdot)$ da una formula $\vec{\exists}\Delta_0$. Aggiungendo il quantificatore $\exists y$ la formula rimane $\vec{\Delta}_0$, quindi anche A è definibile da una formula $\vec{\Delta}_0$. □

14.17 Corollario. *Gli insiemi ricorsivamente enumerabili sono esattamente quegli insiemi definibili in* $(\mathbb{N}; +, \cdot)$ *che possono essere definiti da una formula* $\vec{\exists}\Delta_0$.

Dimostrazione. Avendo già dimostrato la parte difficile, rimane solo da verificare che ogni insieme definibile da una formula $\vec{\exists}\Delta_0$ è ricorsivamente enumerabile. Osserviamo che gli insiemi definibili da una formula Δ_0 sono ricorsivi (addirittura primitivi ricorsivi) essendo ottenuti da predicati ricorsivi (i grafici di $+$ e $\cdot$) per mezzo di connettivi booleani e quantificatori limitati. Per concludere basta osservare che i predicati ricorsivamente enumerabili sono stabili per quantificazione esistenziale. □

14.18 Osservazione. Indicando con $\Delta_0^{\mathbb{N}}$ la classe dei predicati definibili in $(\mathbb{N}; +, \cdot)$ da una formula Δ_0, si hanno le inclusioni $\Delta_0^{\mathbb{N}} \subset$ Primitivi ricorsivi $\subset$ Ricorsivi. Tali inclusioni sono strette. Valgono però le uguaglianze $\exists\Delta_0^{\mathbb{N}} =$ $\exists$Primitivi ricorsivi $=$ $\exists$Ricorsivi, in quanto ognuna di queste classi coincide con la classe dei predicati ricorsivamente enumerabili.

15 Teorie formali per l'aritmetica

15.1 L'aritmetica di Peano del secondo ordine

L'aritmetica di Peano PA^2 (l'indice 2 sta ad indicare che si tratta di una teoria del "secondo ordine") è una teoria formale assiomatizzata dai seguenti assiomi.

P1. $\forall xy(S(x) = S(y) \to x = y)$

P2. $\forall x(0 \neq S(x))$

P3 (Induzione). $\forall P(P(0) \wedge \forall x(P(x) \to P(Sx)) \to \forall y P(y))$.

15.1 Definizione. Un modello di PA^2 è una struttura $(M; S^M, 0^M)$ dove M è un insieme non vuoto, 0^M è un elemento di M, S^M è una funzione (totale) da M ad M, e gli assiomi sono soddisfatti interpretando i simboli 0 e S come l'elemento 0^M e la funzione S^M, e convenendo che le variabili x, y variano nell'insieme M, mentre P varia nell'insieme delle parti di M (scriviamo $P(x)$ se $x \in P$).

15.2 Osservazione. I numeri naturali con lo zero e il successore sono un modello di PA^2. Vedremo che PA^2 definisce la struttura dei numeri naturali a meno di isomorfismo, nel senso che tutti i modelli di PA^2 sono isomorfi tra loro.

15.3 Osservazione. Intuitivamente l'assioma di induzione garantisce che tutti gli elementi di un modello di PA^2 si ottengono a partire da 0 applicando la funzione successore un numero finito di volte. Infatti l'insieme P degli elementi che così si ottengono contiene 0 ed è ovviamente chiuso per successore, e pertanto in base all'assioma di induzione contiene ogni elemento del modello. Questa spiegazione ha valore euristico ma è informale, in quanto assume a livello intuitivo la nozione "numero finito di volte", ovvero sostanzialmente la nozione di numero naturale, che è proprio ciò che gli assiomi si prefiggono di definire.

15.4 Osservazione. P1, P2 sono formule "del primo ordine" in quanto le variabili quantificate variano sul dominio del modello, mentre P3 è una formula "del secondo ordine" in quanto la variabile quantificata P assume valori nell'insieme delle parti del modello. In generale una formula del secondo ordine può contenere variabili che assumono come valori relazioni n-arie sul dominio del modello. Chiaramente la distizione tra primo e secondo ordine dipende dalle nostre convenzioni sulla interpretazione delle formule.

15.5 Osservazione. Non c'è bisogno di aggiungere alla teoria PA^2 assiomi riguardandi la somma, il prodotto, o la relazione d'ordine $\leq$, in quanto queste operazioni possono essere definite da opportune formule (del secondo ordine) a partire da zero e successore. Ad esempio $x \leq y$ può essere definito come $\forall P(P(x) \wedge \forall u(P(u) \to P(S(u))) \to P(y))$. La relazione $x + y = z$ può essere definita da $\forall R(R(x, 0, x) \wedge \forall uv(R(x, u, v) \to R(x, Su, Sv)) \to R(x, y, z))$, dove R varia tra le relazioni ternarie. Lasciamo come esercizio la definizione di $x \cdot y = z$.

Mostriamo che, dato un qualsiasi modello $(M; S^M, 0^M)$ di PA^2, è possibile giustificare l'uso di definizioni ricorsive su M.

15.6 Teorema. *(Definizioni ricorsive) Sia $(M; S^M, 0^M)$ un modello dell'aritmetica di Peano. Sia N un insieme non vuoto, sia h una funzione $h\colon N \to N$ e sia a un elemento di N. Esiste una e una sola funzione $F\colon M \to N$ tale che $F(0^M) = a$ ed $F(S^M x) = h(F(x))$.*

Dimostrazione. La dimostrazione avviene all'interno di una opportuna metateoria insiemistica, che in questa trattazione viene assunta a livello intuitivo ma che potrebbe essere formalizzata. La funzione F viene definita (identificando una funzione con il suo grafico) come l'intersezione della classe di tutte le relazioni binarie $R \subseteq M \times N$ che godono delle seguenti proprietà: 1) $0^M R a$; 2) se xRy, allora $S^M(x) R\, h(y)$. Osserviamo che esiste almeno una relazione con queste proprietà: basta considerare la relazione che vale tra ogni elemento di M e ogni elemento di N. Ha senso dunque considerare l'intersezione $F \subseteq M \times N$ di tutte queste relazioni. Si verifica che F è essa stessa una relazione che gode delle stesse proprietà, ed è ovviamente la più piccola tale relazione. Mostriamo che F è una funzione, ovvero che per ogni $x \in M$ esiste uno ed un solo $y \in N$ tale che xFy (fatto ciò potremo scrivere $Fx = y$). Consideriamo a tal fine l'insieme $P \subseteq M$ di tutti gli $x \in M$ per i quali esiste un $y \in N$ tale che xFy. Tale insieme contiene 0^M in quanto $0^M F a$. Inoltre P è chiuso per successore in quanto se xFy allora $S^M(x) F\, h(y)$. Siccome M verifica l'assioma di induzione, possiamo

concluderne che $P = M$. Abbiamo così mostrato che per ogni $x \in M$ esiste almeno un $y \in N$ per cui xFy.

Analogamente si mostra l'unicità di y, cioè il fatto che F è una funzione. Basta considerare questa volta l'insieme $Q \subseteq M$ di tutti gli $x \in M$ tali che $\forall y, z \in N(xFy \wedge xFz \to y = z)$ e mostrare che $Q = M$. Basta dunque verificare che Q contiene 0^M ed è chiuso per successore. Sappiamo che $0^M Fa$. Se avessimo $0^M Fy$ anche per qualche $y \neq a$, togliendo la coppia $(0^N, y)$ da F si otterrebbe una relazione $R \subseteq F$ che continua a verificare le proprietà 1) e 2), contraddicendo la minimalità di F. Quindi $0^M \in Q$. Supponiamo ora che $x \in Q$ e consideriamo l'unico $y \in N$ tale che xFy. Abbiamo allora $S^M(x)Fh(y)$. Se ci fosse un elemento $b \neq h(y)$ tale che $S^M(x)Fb$, togliendo la coppia $(S^M x, b)$ da F otterremmo di nuovo una relazione che verifica 1) e 2) contraddicendo la minimalità di F. Abbiamo così mostrato che F è effettivamente una funzione da M ad N.

Per mostrare l'unicità di F consideriamo un'altra funzione $G\colon M \to N$ tale che $G(0^M) = a$ e $G(S^M x) = h(Gx)$. Sia $P \subseteq M$ l'insieme di tutti gli $x \in M$ tali che $Fx = Gx$. Chiaramente $0^M \in P$, e inoltre se $x \in P$ si ha $G(S^M x) = h(Gx) = h(Fx) = F(S^M x)$, e quindi $S^M x \in P$. Di nuovo per l'assiome di induzione possiamo concludere $P = M$, e pertanto $G = F$. □

15.7 Teorema. *(Dedekind) Tutti i modelli dell'aritmetica di Peano (assiomatizzata da $P1, P2, P3$) sono isomorfi tra loro.*

Dimostrazione. Siano $(M; S^M, 0^M)$ ed $(N; S^N, 0^N)$ due modelli. Poichè M è un modello, per il Teorema 15.6 esiste una funzione $f\colon M \to N$ che manda lo zero di M nello zero di N e preserva il successore, nel senso che $f(S^M x) = S^N(fx)$. Analogamente esiste $g\colon N \to M$ nel verso opposto con analoghe proprietà. La composizione $g \circ f\colon M \to M$ manda anche essa lo zero in zero e preserva il successore, e pertanto per la proprietà di unicità espressa dal Teorema 15.6, coincide con l'identità su M (in quanto anche l'identità manda zero in zero e preserva il successore). Analogamente $f \circ g$ è l'identità su N, e quindi f, g sono l'una l'inversa dell'altra, da cui segue che sono entrambe isomorfismi. □

15.8 Osservazione. Il teorema precedente mostra che se esiste un modello esso è unico esso è unico a meno di isomorfismo, ma non mostra l'esistenza di un modello. L'esistenza può essere dimostrata all'interno di una opportuna teoria assiomatica degli insiemi, quale la teoria di Zermelo Fraenkel, facendo uso del cosiddetto "assioma dell'infinito".

15.2 La teoria Q di Robinson

La teoria Q di Robinson assiomatizza alcune delle proprietà dei numeri naturali, ma non ha assiomi di induzione. 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. $S(x) = S(y) \to x = y$,

Q2. $0 \neq S(x)$,

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

In questi assiomi i quantificatori sono stati sottointesi: le variabili x, y si intendono quantificate universalmente. Le proprietà espresse dagli assiomi di Q sono verificate nei numeri naturali con le solite operazioni aritmetiche di zero, successore, somma e prodotto. Esistono però altri modelli di Q non isomorfi al modello $\mathbb{N}$ dei numeri naturali.

15.9 Esempio. Un'altro modello di Q si può ottenere nel modo seguente. Sia $\mathbb{Z}[x]$ l'anello dei polinomi in una variabile a coefficienti in $\mathbb{Z}$. Definiamo un ordine totale su $\mathbb{Z}[x]$ stabilendo che un polinomio $p(x) \in \mathbb{Z}[x]$ è positivo se $\lim_{x\to\infty} p(x) > 0$. Equivalentemente un polinomio $a_0 + a_1x + \ldots + a_nx^n$, con $a_n \neq 0$, è positivo se e solo se $a_n > 0$. Sia $\mathbb{Z}[x]^+$ l'insieme dei polinomi non negativi (positivi o zero) dell'anello $\mathbb{Z}[x]$. 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 $\mathbb{Z}[x]^+$ 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.

La definizione formale di modello è analoga alla Definizione 15.1:

15.10 Definizione. Un modello di Q è una struttura $(M; +^M, \cdot^M, S^M, 0^M)$ dove M è un insieme non vuoto, $+^M$ e $\cdot^M$ sono funzioni binarie (totali) su M, S^M è una funzione unaria, e 0^M è un elemento di M, dove si richiede che gli assiomi di Q siano verificati interpretando i simboli $+, \cdot, S, 0$ con le corrispondenti funzioni di M (il simbolo $=$ è interpretato come la relazione di uguaglianza), e stipulando che le variabili assumano valori tra gli elementi di M.

15.3 L'aritmetica di Peano del primo ordine

15.11 Definizione. L'aritmetica di Peano del primo ordine, che indicheremo con PA, o con PA^1 per ricordarci che si tratta di una teoria del primo ordine, è una teoria formulata nello stesso linguaggio $L = \{+, \cdot, S, 0\}$ della teoria di Robinson Q, e che possiede, oltre agli assiomi di Q, il cosiddetto schema di induzione. Tale schema è una lista infinità di assiomi, uno per ogni coppia (ϕ, x) dove ϕ è una formula del primo ordine di L, e x è una variabile libera di ϕ (la variabile su cui facciamo l'induzione). Alla coppia (ϕ, x) associamo l'assioma $Ind_{\phi,x}$ definito come $\phi(0) \wedge \forall x(\phi(x) \to \phi(Sx)) \to \forall y\phi(y)$, dove la notazione $\phi(t)$, indica la formula ottenuta sostituendo t al posto delle occorrenze libere di x in ϕ. La formula ϕ potrebbe contenere altre variabili libere oltre alla x, nel qual caso si intende che le rimanenti variabili siano quantificate universalmente nel corrispondente assioma $Ind_{\phi,x}$. Ad esempio se $\phi = \phi(x, z)$ ha come variabili libere x e z, l'assioma $Ind_{\phi,x}$ è $\forall z[\phi(0, z) \wedge \forall x(\phi(x, z) \to \phi(Sx, z)) \to \forall y\phi(y, z)]$,

dove l'induzione si fa su x e l'altra variabile libera z si comporta come un parametro.

15.12 Osservazione. La differenza tra l'assioma di induzione del secondo ordine $\forall P(P(0) \wedge \forall x(P(x) \rightarrow P(Sx)) \rightarrow \forall y P(y))$, e lo schema di assiomi di induzione, è che nel primo caso la proprietà di induzione viene chiesta per ogni possibile proprietà P, mentre nel secondo caso viene chiesta solo per quelle proprietà $P(x)$ che sono definibili da una formula (possibilmente con parametri). Pur non possedendo l'induzione nella sua piena forza, PA^1 è tuttavia una teoria molto potente: quando facciamo una dimostrazione per induzione, quasi sempre la proprietà su cui facciamo l'induzione è di fatto definibile da una formula.

Il concetto di modello per PA^1 si definisce in modo analogo a quanto visto in precendenza per le altre terie. Ogni modello di PA^1 è anche un modello di Q ma non viceversa. Ci sono modelli di Q che non verificano lo schema di induzione.

15.13 Esercizio. Il modello $\mathbb{Z}[x]^+$ di Q non è modello di PA^1. Suggerimento: si consideri l'insieme degli elementi che sono divisibili per due o il cui successore è divisibile per due.

15.14 Osservazione. Il motivo per considerare PA^1 anziché la più naturale PA^2, è che per la logica del primo ordine vale il teorema di completezza: un enunciato ϕ del primo ordine è vero in tutti i modelli di una teoria T del primo ordine se e solo se esiste una dimostrazione formale di ϕ da T.

15.4 Conseguenza logica

Ricordiamo la definizione di conseguenza logica.

15.15 Definizione. (Conseguenza logica) Un enunciato ϕ è conseguenza logica di una teoria T se è vero in tutti i modelli di T. Il concetto di conseguenza logica dipende ovviamente dalla definizione del concetto di modello.

Dimostrare ϕ da T significa dimostrare che ϕ è conseguenza logica di T.

15.16 Esempio. L'enunciato $\forall x \exists y(2y = x \vee 2y = x + 1)$ (dove abbiamo usato le abbreviazioni $1 = S(0)$ e $2 = S(1)$), non è conseguenza logica di Q in quanto nel modello $\mathbb{Z}[x]^+$ dell'Esempio 15.9 non è vero che ogni elemento è divisibile per due o il suo successore lo è (la variabile x di $\mathbb{Z}[x]^+$ non è il doppio di alcun polinomio, e neppure $x + 1$ lo è).

15.17 Osservazione. È possibile trovare modelli di Q in cui non vale la legge commutativa della somma e del prodotto. Quindi $\forall xy(x + y = y + x)$ non è conseguenza logica di Q.

15.18 Esercizio. Si dimostri in PA^1 la commutatività dell'addizione e della moltiplicazione.

15.5 Dimostrazioni formali

15.19 Fatto. (Dimostrazioni formali) Il teorema di completezza per il calcolo dei predicati, afferma che, nel caso di teorie ed enunciati del primo ordine, è possibile dare esplicitamente un insieme finito di regole di inferenza logiche che è sufficiente per dimostrare tutte le conseguenze logiche di una qualsiasi teoria. Una possibile scelta delle regole logiche è data dalla Definizione 15.21. Una dimostrazione condotta utilizzando esclusivamente le regole fissate viene detta dimostrazione formale. L'adeguatezza delle regole della Definizione 15.21 viene dimostrata all'interno di una opportuna metateoria insiemistica.

15.20 Osservazione. Il teorema di completezza non si applica alla teoria PA^2, essendo questa una teoria del secondo ordine. Per tale teoria abbiamo definito il concetto di modello e quindi la nozione di conseguenza logica, ma non è possibile dare una corrispondente nozione di dimostrazione formale tale che tutte le conseguenze logiche di PA^2 abbiano una dimostrazione formale.

15.21 Definizione. (Regole di inferenza logiche) Scriviamo $\Gamma \vdash \phi$ per esprimere il fatto che la tesi ϕ è deducibile dall'insieme di ipotesi Γ. Scriviamo Γ, ϕ per l'insieme $\Gamma \cup \{\phi\}$, dove Γ è un insieme di formule e ϕ è una formula. Scriviamo $\phi(t/x)$ per il risultato della sostituzione del termine t al posto di tutte le occorrenze libere della variabile x nella formula ϕ.

- (Ax) $\phi \vdash \phi$
- (Wk) $\dfrac{\Gamma \vdash \phi}{\Gamma, \alpha \vdash \phi}$
- ($\vdash \wedge$) $\dfrac{\Gamma \vdash \alpha \qquad \Gamma \vdash \beta}{\Gamma \vdash \alpha \wedge \beta}$
- ($\wedge \vdash$) $\dfrac{\Gamma, \alpha \vdash \gamma}{\Gamma, \alpha \wedge \beta \vdash \gamma}$ $\qquad$ ($\wedge \vdash$) $\dfrac{\Gamma, \beta \vdash \gamma}{\Gamma, \alpha \wedge \beta \vdash \gamma}$
- ($\vee \vdash$) $\dfrac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha \vee \beta}$ $\qquad$ ($\vee \vdash$) $\dfrac{\Gamma \vdash \beta}{\Gamma \vdash \alpha \vee \beta}$
- ($\vdash \vee$) $\dfrac{\Gamma, \alpha \vdash \gamma \qquad \Gamma, \beta \vdash \gamma}{\Gamma, \alpha \vee \beta \vdash \gamma}$
- ($\vdash \rightarrow$) $\dfrac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta}$ $\qquad$ ($\rightarrow / e$) $\dfrac{\Gamma \vdash \alpha \qquad \Gamma \vdash \alpha \rightarrow \beta}{\Gamma \vdash \beta}$
- ($\vdash \perp$) $\dfrac{\Gamma \vdash \alpha \qquad \Gamma \vdash \neg\alpha}{\Gamma \vdash \perp}$ $\qquad$ ($\perp$) $\dfrac{\Gamma \vdash \perp}{\Gamma \vdash \alpha}$
- ($\neg \vdash$) $\dfrac{\Gamma \vdash \alpha}{\Gamma, \neg\alpha \vdash \perp}$ $\qquad$ ($\vdash \neg$) $\dfrac{\Gamma, \alpha \vdash \perp}{\Gamma \vdash \neg\alpha}$
- (RAA) $\dfrac{\Gamma, \neg\alpha \vdash \perp}{\Gamma \vdash \alpha}$

- $(\vdash \forall)\dfrac{\Gamma \vdash \phi(y/x)}{\Gamma \vdash \forall x\phi}$ dove y è una variabile che non occorre libera in Γ.
- $(\forall/e)\dfrac{\Gamma \vdash \forall x\phi}{\Gamma \vdash \phi(t/x)}$ dove t è un qualsiasi termine sostituibile per x in ϕ.
- $(\vdash \exists)\dfrac{\Gamma \vdash \phi(t/x)}{\Gamma \vdash \exists x\phi}$ dove t è un qualsiasi termine sostituibile per x in ϕ.
- $(\exists \vdash)\dfrac{\Gamma, \phi(y/x) \vdash \gamma}{\Gamma, \exists x\phi \vdash \gamma}$ dove y è una variabile che non occorre libera in Γ o in γ.
- $(\vdash =)\Gamma \vdash x = x$
- $(=/e)\dfrac{\Gamma \vdash \phi(t/x) \qquad \Gamma \vdash t = t'}{\Gamma \vdash \phi(t'/x)}$
  dove t, t' sono termini sostituibili per x in ϕ.

Se T è una teoria, dare una dimostrazione formale di un enunciato ϕ da T, significa trovare un sottoinsieme Γ degli assiomi di T e dedurre $\Gamma \vdash \phi$ usando le regole sopra date.

15.22 Osservazione. Non bisogna confondere il concetto di dimostrazione rigorosa con il concetto di dimostrazione formale. Una dimostrazione può essere rigorosa pur non essendo formale. Nella prossima sezione mostreremo che certi enunciati sono veri in tutti i modelli della teoria Q dandono delle dimostrazioni rigorose seppure non formali.

15.6 Alcune formule dimostrabili in Q

Dato $n \in \mathbb{N}$, il termine $S(S(S(\ldots S(0))))$ (n volte S) viene detto il *numerale* di n e indicato con $\overline{n}$ o con $S^n 0$.

15.23 Lemma. *$\forall a, b, c \in \mathbb{N}$, se $a + b = c$, allora $Q \vdash \overline{a} + \overline{b} = \overline{c}$ (nel senso che l'enunciato $\overline{a} + \overline{b} = \overline{c}$ è vero in tutti i modelli di Q).*

Quindi as esempio Q dimostra $S(S(0)) + S(S(S(0))) = S(S(S(S(0)))))$.

Dimostrazione. 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 $Q \vdash x + S(y) = S(x + y)$ (dove sottointendiamo $\forall xy$). Prendendo $x = \overline{a}, y = \overline{b-1}$, otteniamo $Q \vdash \overline{a} + \overline{b} = \overline{c}$. □

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

Dimostrazione. 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 $Q \vdash x \cdot S(y) = x \cdot y + x$. □

15.25 Definizione. I termini del linguaggio $L = L(Q) = \{+, \cdot, S, 0\}$ sono le espressioni che è possibile costruire usando le variabili e i simboli di L, correttamente parentesizzate. Ad esempio $(S(x) + S(0)) \cdot 0$ è un termine di L, o L-termine.

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

Ad esempio Q dimostra che $(S(0) + S(0)) \cdot (S(0) + S(0)) = S(S(S(S(0))))$.

Dimostrazione. 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 $a, b \in \mathbb{N}$ 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. □

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.

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

Dimostrazione. Segue dal fatto che Q, così come ogni altra teoria, dimostra $\forall x(x = x)$. □

Abbiamo fino ad ora dimostrato solamente uguaglianze. Veniamo ora alle disuguaglianze.

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

Ad esempio Q dimostra $S(0) \neq S(S(0))$.

Dimostrazione. 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 $Q \vdash \overline{a} \neq \overline{b}$. □

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

Dimostrazione. Siano $a, b \in \mathbb{N}$ 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 $a = b$, $Q \vdash t_1 = t_2$. □

15.7 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 $M' = \{x \in M \mid \exists n \in \mathbb{N}\ x = \overline{n}^M\}$, dove $\overline{n}^M = S^M(S^M(\ldots S^M(0^M)))$ (con n occorrenze di S^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 il più piccolo sottoinsieme di M contenente lo 0 di M e chiuso per la funzione successore S di M. Ne segue che se M è modello di PA^2 tutti i suoi elementi sono standard: basta applicare l'assioma di induzione all'insieme degli elementi standard. Siccome l'insieme dei numeri standard non è necessariamente definibile da una formula del primo ordine, lo stesso ragionamento non è fattibile in PA^1.

Dai risultati fin qui mostrati per Q segue che l'addizione e la moltiplicazione di numeri standard si comporta come la usuale addizione e moltiplicazione di numeri naturali, nel senso che il sottoinsieme $M' \subseteq M$ dei numeri standard costituisce una sottostruttura di M isomorfa ad $\mathbb{N}$ con le usuali operazioni di addizione e moltiplicazione. Inoltre esiste un unico isomorfismo da $\mathbb{N}$ ad M' dato dalla mappa che manda $n \in \mathbb{N}$ in $\overline{n}^M$. L'unicità segue dal fatto che un qualsiasi omomorfismo di $\mathbb{N}$ in M deve mandare 0 in 0^M e preservare il successore S. Ne segue che $\mathbb{N}$ è immergibile in ogni modello M di Q, e che M è isomorfo ad $\mathbb{N}$ se e solo se M non ha numeri non-standard.

15.30 Esempio. Nel modello $\mathbb{Z}[x]^+$ di Q i numeri non-standard sono esattamente i polinomi non-costanti.

15.8 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:

15.31 Lemma. $\forall n \in \mathbb{N}$, $Q \vdash \forall x\ (\ x \leq \overline{n} \leftrightarrow x = \overline{0} \vee x = \overline{1} \vee \ldots \vee x = \overline{n})$.

Dimostrazione. Induzione su n. Fissiamo un modello M di Q ed un elemento x di M e mostriamo che in M vale la doppia implicazione $x \leq \overline{n} \leftrightarrow x = \overline{0} \vee x = \overline{1} \vee \ldots \vee x = \overline{n})$.

L'implicazione da destra a sinistra segue dalla definizione di $\leq$ e dal Lemma 15.23. Mostriamo l'altra implicazione.

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 e Q1 $z + y = \overline{n-1}$. Quindi $y \leq \overline{n-1}$. Per ipotesi induttiva $y = \overline{0} \vee \ldots \vee y = \overline{n-1}$. Quindi $x = \overline{1} \vee \ldots \vee x = \overline{n}$. □

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 $\overline{n}^M$ 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 $\mathbb{N}$ e i numeri standard di M preserva non solo $0, S, +$ e $\cdot$, ma anche la relazione $\leq$.

15.32 Corollario. $\forall n \in \mathbb{N}\ Q \vdash \forall x. x \leq \overline{n+1} \longleftrightarrow x \leq \overline{n} \vee x = \overline{n+1}$.

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

15.33 Corollario. *Sia $n \in \mathbb{N}$. Sono equivalenti:*

1. $\forall a \leq n Q \vdash \varphi(\overline{a})$,
2. $Q \vdash \forall x \leq \overline{n} \phi(x)$.

15.34 Lemma. $\forall a, b \in \mathbb{N}$, $Q \vdash \overline{a} \leq \overline{b}$ *oppure* $Q \vdash \neg(\overline{a} \leq \overline{b})$. *Il primo caso capita se $a \leq b$. Il secondo se $a > b$.*

Dimostrazione. Se $a \leq b$, allora $Q \vdash \overline{a} = 0 \vee \overline{a} = \overline{1} \vee \ldots \vee \overline{a} = \overline{b}$ (in quanto la disgiunzione contiene $\overline{a} = \overline{a}$) e quindi $Q \vdash \overline{a} \leq \overline{b}$ per il Lemma 15.31.

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 $Q \vdash \neg(\overline{a} \leq \overline{b})$. □

Conveniamo che $x < y$ stia per $x \leq y \wedge x \neq y$.

15.35 Lemma. $\forall b \in \mathbb{N}\ Q \vdash \forall x(x \leq \overline{b} \vee \overline{b} < x)$.

Dimostrazione. Per induzione su b. Ragionando in Q fissiamo x e mostriamo $x \leq \overline{b}$ o $x \geq \overline{b}$. Per l'assioma $Q4$ $x + 0 = x$ e quindi $0 \leq x$. Supponiamo dunque $b > 0$. Per ipotesi induttiva $x \leq \overline{b-1}$ o $x \geq \overline{b-1}$. Nel primo caso per il Lemma 15.31 $x \leq \overline{b}$. 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 $x \geq \overline{b}$. □

In un modello non-standard M di Q la relazione $\leq$ non è un buon ordine, nel senso che contiene delle successioni discendent infinite $x, x-1, x-2, x-3, \ldots$, dove $x-1$ indica un elemento y tale che $S(y) = x$, $x-2$ un elemento z tale che $S(S(z)) = x$ e così via. In alcuni modelli di Q la relazione $\leq$ non è neppure un ordine totale. Tuttavia abbiamo:

15.36 Lemma. *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 $\forall a \in A\ M \models m \leq a$.*

Dimostrazione. 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 $a < x$. □

15.9 Le formule con quantificatori limitati sono decidibili in Q

Un enunciato si dice **indipendente** da Q se $Q \nvdash \phi$ e $Q \nvdash \neg\phi$. Ad esempio l'enunciato $\forall x \exists y(2y = x \vee 2y = x + 1)$ è indipendente da Q. Un enunciato φ si dice **decidibile** in Q (o Q-decidibile) se $Q \vdash \varphi$ oppure $Q \vdash \neg\varphi$. Quindi un enunciato è decidibile se non è indipendente. Questa nozione non va confusa con la decidibilità nel senso della teoria della calcolabilità, la quale si riferisce ad insiemi o predicati, mentre la decidibilità in Q si riferisce a singoli enunciati.

Da un punto di vista semantico (cioè dei modelli) dire che φ è indipendente da Q significa che ϕ è vero in certi modelli di Q e falso in altri. Dirre che ϕ è decidibile in Q significa che φ 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 $\mathbb{N}$) oppure è falso in tutti i modelli di Q.

Da questa caratterizzazione è immediato vedere:

15.37 Lemma. *Una combinazione booleana di enunciati decidibili in Q è decidibile in Q.*

Dimostrazione. Ad esempio supponiamo che φ e ϕ siano decidibili in Q e mostriamo che la disgiunzione $\varphi \vee \phi$ è decidibile in Q. Si hanno quattro possibili casi.

1) $Q \vdash \varphi$ e $Q \vdash \phi$.
2) $Q \vdash \varphi$ e $Q \vdash \neg\phi$.
3) $Q \vdash \neg\varphi$ e $Q \vdash \phi$.
4) $Q \vdash \neg\varphi$ e $Q \vdash \neg\phi$.

I casi in cui Q non dimostra nessuna delle formule in questione nè la loro negazione è escluso perchè φ e ϕ sono decidibili in Q. Nel caso 4 abbiamo $Q \vdash \neg(\varphi \vee \phi)$. Negli altri tre casi $Q \vdash \varphi \vee \phi$. Quindi $\varphi \vee \phi$ è decidibile in Q. □

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

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

15.39 Lemma. *Se $\varphi(\overline{n})$ è decidibile in Q per ogni $n \in \mathbb{N}$, e se t è un termine chiuso, allora gli enunciati $\forall x \leq t\varphi(x)$ e $\exists x \leq t\varphi(x)$ sono decidibili in Q.*

Dimostrazione. Sia $b \in \mathbb{N}$ tale che $Q \vdash \overline{b} = t$. Basta allora mostrare che l'enunciato $\forall x \leq \overline{b}\varphi(x, \overline{a_1}, \ldots, \overline{a_n})$ è decidibile in Q. 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 $\forall x \leq \overline{b}\varphi(x)$ è equivalente, in Q, all'enunciato $\varphi(\overline{0}) \wedge \varphi(\overline{1}) \wedge \ldots \wedge \varphi(\overline{b})$ 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 $\varphi(\overline{0}) \vee \varphi(\overline{1}) \vee \ldots \vee \varphi(\overline{b})$. □

Ricordiamo che una formula Δ_0, è una formula senza quantificatori o una formula i cui quantificatori siano tutti limitati.

15.40 Corollario. *Ogni enunciato Δ_0 è decidibile in Q.*

Dimostrazione. Segue dai risultati precedenti per induzione sul numero dei connettivi logici della formula. □

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

Una spiegazione intuitiva del fatto che gli enunciati Δ_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). Poichè M' è isomorfo ad $\mathbb{N}$ ne segue che ϕ è vera in M se e solo se è vera in $\mathbb{N}$ e per l'arbitrarietà di M possiamo concludere che ϕ ha lo stesso valore di verità in tutti i modelli di Q e quindi è decidibile in Q.

Ricordiamo che una formula si dice $\vec{\exists}\Delta^0$, se è ottenuta premettendo un certo numero di quantificatori esistenziali davanti ad una formula Δ_0.

15.41 Lemma. *Se ϕ è un enunciato $\vec{\exists}\Delta^0$ e $\mathbb{N} \models \phi$, allora $Q \vdash \phi$.*

Dimostrazione. Supponiamo che ϕ sia della forma $\exists x_1 \ldots x_n \psi(x_1, \ldots, x_n)$ con $\psi \in \Delta_0$. Assumiamo che ϕ sia vero in $\mathbb{N}$ e siano $a_1, \ldots, a_n \in \mathbb{N}$ tali che $\mathbb{N} \models \psi(\overline{a_1}, \ldots, \overline{a_n})$. Per la decidibilità delle formule Δ_0, $Q \vdash \psi(\overline{a_1}, \ldots, \overline{a_n})$, da cui segue $Q \vdash \exists x_1 \ldots x_n \psi(x_1, \ldots, x_n)$. □

Riassumendo abbiamo:
1) Gli enunciati Δ_0 veri in $\mathbb{N}$ sono dimostrabili in Q.
2) Gli enunciati Δ_0 falsi in $\mathbb{N}$ sono refutabili in Q (cioè è dimostrabile la negazione).
3) Gli enunciati $\vec{\exists}\Delta_0$ veri in $\mathbb{N}$ sono dimostrabili in Q.
4) Gli enunciati $\vec{\exists}\Delta_0$ falsi in $\mathbb{N}$ non sono dimostrabili in Q (poichè $\mathbb{N} \models Q$).

Non è però detto che un enunciato $\vec{\exists}\Delta_0$ falso in $\mathbb{N}$ sia refutabile in Q, ad esempio $\exists x, y.x + y \neq y + x$ è una formula $\vec{\exists}\Delta_0$ falsa in $\mathbb{N}$ ma non refutabile in Q. Quindi esistono formule $\vec{\exists}\Delta_0$ non decidibili in Q.

15.10 Rappresentabilità in Q delle funzioni ricorsive

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

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 $Q \vdash \neg\varphi_A(\overline{a_1}, \ldots, \overline{a_k})$.

La notazione φ_A è leggermente ambigua in quanto un insieme A può essere binumerato da tante formule non equivalenti in Q.

15.43 Proposizione. *Ogni insieme Δ_0-definibile è binumerabile in Q.*

Dimostrazione. Segue dal fatto che ogni formula Δ_0 è decidibile in Q. □

15.44 Definizione. Una funzione totale $f\colon \mathbb{N}^k \to \mathbb{N}$ 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 $\forall a_1, \ldots, a_k \in \mathbb{N}$:

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

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 φ_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 φ_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 $b = f(a_1, \ldots, a_k)$.

Il seguente teorema mostra che ogni relazione semidecidibile contiene il grafico di una funzione.

15.45 Teorema. *(Selezione) Sia $R \subseteq \mathbb{N}^2$ un predicato semidecidibile. Esiste una funzione ricorsiva parziale f tale che per ogni $x \in \mathbb{N}$ se esiste un y tale che $R(x, y)$, allora si ha $R(x, f(x))$.*

Dimostrazione. Definiamo $u = \langle y, z\rangle$ se e solo se $2u = (y + z)(y + z + 1) + x$. La corrispondenza $(y, z) \mapsto \langle y, z\rangle$ è una funzione biunivoca da $\mathbb{N}^2$ ad $\mathbb{N}$. Fissiamo un predicato decidibile $D(x, y, z)$ tale che $R(x, y)$ se e solo se $\exists z D(x, y, z)$. Basta definire $f(x) = \pi_1(\mu\langle y, z\rangle D(x, y, z))$ dove π_1 estrae la componente y della coppia $\langle y, z\rangle$ e $\mu\langle y, z\rangle D(x, y, z)$ è una abbreviazione per $\mu u \exists y, z \leq u(u = \langle y, z\rangle \wedge D(x, y, z))$. □

15.46 Teorema. *Ogni funzione ricorsiva totale $f\colon \mathbb{N}^k \to \mathbb{N}$ è binumerata funzionalmente in Q. Inoltre la formula binumerante può essere scelta di complessità $\exists\Delta_0$.*

Diamo due dimostrazioni. Della prima forniamo solo un cenno.

Dimostrazione. (Cenno) Analizzando i dettagli della dimostrazione del Teorema 14.13, si vede che la stessa dimostrazione funziona sostituendo "definibile in $\mathbb{N}$" con "binumerabile in Q". □

La seconda dimostrazione usa le idee del teorema di selezione.

Dimostrazione. Per semplicità supponiamo che $k = 1$, cioè che f sia una funzione di un argomento. Il grafico di $f\colon \mathbb{N} \to \mathbb{N}$ è ricorsivamente enumerabile, quindi è definibile (nel modello standard $\mathbb{N}$) da una formula $\rho(x, y)$ della forma $\exists z \phi(x, y, z)$ con $\phi \in \Delta_0$, nel senso che $fa = b$ se e solo se $\mathbb{N} \models \exists z \phi(\overline{a}, \overline{b}, z)$.

Fissiamo una tale formula. Poichè le formule di complessità $\exists\Delta_0$ che sono vere nel modello standard $\mathbb{N}$ sono dimostrabili in Q, questa condizione implica $Q \vdash \exists z \phi(\overline{a}, \overline{b}, z)$ (vale anche l'implicazione inversa essendo Q un modello di $\mathbb{N}$, ma questo non ci garantisce la binumerabilità, nè tantomeno la binumerabilità funzionale). Non è detto che una qualsiasi formula che definisce (in $\mathbb{N}$) il grafico di f binumeri funzionalemente f in Q. L'idea allora è di applicare il Teorema di Selezione 15.45 alla relazione definita dalla formula $\rho(x, y)$. In generale il teorema di selezione permette di trovare, all'interno di una relazione, il grafico di una funzione. Se viene applicato alla relazione $\rho(x, y)$, che già definisce (in $\mathbb{N}$) il grafico di una funzione, esso fornirà una nuova formula $\theta(x, y)$ che definisce lo stesso grafico, con il vantaggio che per la nuova formula si potrà dimostrare la binumerabilità funzionale in Q.

Veniamo ai dettagli. Poniamo $u = \langle y, z \rangle$ se e solo se $2u = (y + z)(y + z + 1) + x$. La corrispondenza $(y, z) \mapsto \langle y, z \rangle$ è una funzione biunivoca da $\mathbb{N}^2$ ad $\mathbb{N}$. Definiamo $\theta(x, y)$ come la formula $\exists u \exists z [u = \langle y, z \rangle \wedge \phi(x, y, z) \wedge \forall u', y', z' < u (u' = \langle y', z' \rangle \rightarrow \neg \phi(x, y', z'))]$. Mostriamo che la formula $\theta(x, y)$ binumera funzionalmente f in Q. Supponiamo a tale fine che $fa = b$. Qualsiasi coppia $(u, v) \in \mathbb{N}^2$ tale che $\mathbb{N} \models \phi(a, u, v)$ deve avere $u = b$ (perché $\exists z \phi(x, y, z)$ definisce il grafico di f). Tra tutte queste coppie sia (b, c) quella che ha la minima codifica. Poiché $\mathbb{N} \models \phi(a, b, c)$ si ha $Q \vdash \phi(\overline{a}, \overline{b}, \overline{c})$ (essendo $\phi \in \Delta_0$). Per la minimalità di $\langle b, c \rangle$, e per il fatto che Q dimostra tutti gli enunciati limitati veri, $Q \vdash \forall u, b', c' < \overline{\langle b, c \rangle} (u = \langle b', c' \rangle \rightarrow \neg \phi(\overline{a}, b', c'))$. Ne segue che $Q \vdash \theta(\overline{a}, \overline{b})$. Resta da dimostrare che $Q \vdash \forall y (\theta(\overline{a}, y) \rightarrow y = \overline{b})$. Ragionando in Q, supponiamo che valga $\theta(\overline{a}, y)$ per un certo y (in un modello di Q tale y potrebbe essere non standard). Esiste allora z tale che $\phi(\overline{a}, y, z)$ e per ogni y', z' tali che $\langle y', z' \rangle < \langle y, z \rangle$ si ha $\neg \phi(\overline{a}, y', z')$. In particolare, visto che vale $\phi(\overline{a}, \overline{b}, \overline{c})$, non può essere che $\overline{\langle b, c \rangle} < \langle y, z \rangle$, e pertanto deve essere $\langle y, z \rangle \leq \overline{\langle b, c \rangle}$. Non può valere la disuguaglianza stretta in quanto abbiamo $\forall u, b', c' < \overline{\langle b, c \rangle} (u = \langle b', c' \rangle \rightarrow \neg \phi(\overline{a}, b', c'))$ (avendo dimostrato questa formula in Q). Pertanto vale l'uguaglianza, e ne segue $y = \overline{b}$. □

16 Teoremi di incompletezza di Gödel

16.1 Aritmetizzazione della sintassi e predicato di dimostrabilità

Sia L un linguaggio del primo ordine con un numero finito di simboli di funzione, relazione e costante (ad esempio $L = \{+, \cdot, S, 0\}$) e associamo ad ogni simbolo s di L 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 $\#(v)$.

16.1 Definizione. Fissiamo una codifica delle successioni di numeri naturali e scriviamo $\langle a_1, \ldots, a_n \rangle$ per indicare la codifica della successione $a_1, \ldots, a_n$. Associamo ad ogni L-termine t un numero naturale $\lceil t \rceil$ 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 $f(t_1, \ldots, t_n)$, $\ulcorner t \urcorner = \langle \ulcorner f \urcorner, \ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner \rangle$.

16.2 Definizione. Associamo ad ogni L-formula φ 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 α, β sono L-formule, allora $\ulcorner \neg\alpha \urcorner = \langle \ulcorner \neg \urcorner, \ulcorner \alpha \urcorner \rangle$, $\ulcorner (\alpha \vee \beta) \urcorner = \langle \ulcorner \vee \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner (\alpha \wedge \beta) \urcorner = \langle \ulcorner \wedge \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner (\alpha \to \beta) \urcorner = \langle \ulcorner \to \urcorner, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle$, $\ulcorner \forall v_i \alpha \urcorner = \langle \ulcorner \forall \urcorner, \ulcorner v_i \urcorner, \ulcorner \alpha \urcorner \rangle$, $\ulcorner \exists v_i \alpha \urcorner = \langle \ulcorner \exists \urcorner, \ulcorner v_i \urcorner, \ulcorner \alpha \urcorner \rangle$.

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

Dimostrazione. 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 $n = \ulcorner c \urcorner$, oppure esiste $i < n$ tale che $n = \langle \ulcorner v \urcorner, i \rangle$, oppure esistono $u, v < n$ tali che $n = \langle \ulcorner f \urcorner, u, v \rangle$ 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 $T\colon \mathbb{N} \to \mathbb{N}$ 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 $h\colon \mathbb{N}^2 \to \mathbb{N}$. Basterà definire h nel modo seguente:

i) $h(n, s) = 1$ se $n = \ulcorner c \urcorner$ oppure $\exists i < n\colon n = \langle \ulcorner v \urcorner, i \rangle$ oppure "s codifica una successione di lunghezza n" e $\exists u, v < n\colon n = \langle \ulcorner f \urcorner, u, v \rangle$ 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. □

16.4 Lemma. *L'insieme* $\{\ulcorner\phi\urcorner \mid \phi$ *è una L-formula* $\}$ *è primitivo ricorsivo.*

Dimostrazione. 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 $n = \langle \#(\to), a, b\rangle$ 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. □

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

Dimostrazione. Basterà fare in modo che **sub** soddisfi le clausole che seguono, dove abbiamo indicato con f un generico simbolo di funzione n-aria di L, con R un generico simbolo di relazione n-aria di L, con $t_1, \ldots, t_n$ degli L-termini, e con α, β delle L-formule.

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

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). □

16.6 Osservazione. 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 $\forall v_i(t = v_i \to \phi)$.

16.7 Teorema. *Sia T una L-teoria (consideriamo solo teorie del primo ordine) tale che l'insieme $\{\ulcorner\phi\urcorner \mid \phi$ è un assioma di $T\}$ è ricorsivo. Allora l'insieme delle conseguenze logiche di T (o più precisamente delle loro codifiche) è ricorsivamente enumerabile.*

Dimostrazione. Basta mostrare che esiste un insieme ricorsivo $\mathbf{Prov}_T \subseteq \mathbb{N}^2$ tale che ϕ è un teorema di T se e solo se $\exists n \in \mathbb{N}$: $(x, \ulcorner\phi\urcorner) \in \mathbf{Prov}_T$.È possibile codificare con dei numeri le dimostrazioni formali. Definiamo $\mathbf{Prov}_T$ come l'insieme di tutte le coppie $(n, \ulcorner\phi\urcorner)$ tali che n codifica una dimostrazione formale di ϕ da T. L'insieme così ottenuto è ricorsivo a condizione che gli assiomi di T costituiscano un insieme ricorsivo. □

16.8 Osservazione. Il teorema vale anche indebolendo le ipotesi: basta che l'insieme delle codifiche degli assiomi si ricorsivamente enumerabile.

Ricordiamo che una teoria è completa se per ogni enunciato ϕ si ha $T \vdash \phi$ oppure $T \vdash \neg\phi$.

16.9 Corollario. *Sia T una teoria completa con un insieme ricorsivo di assiomi. Allora T è decidibile, ovvero l'insieme $\{\ulcorner\phi\urcorner \mid T \vdash \phi\}$ è ricorsivo.*

Dimostrazione. L'insieme in questione è ricorsivamente enumerabile. In virtù della completezza il suo complemento (all'interno dell'insieme ricorsivo di tutte le formule) consiste delle formule ϕ tali che $\neg\phi$ è un teorema di T. Questo secondo insieme è anch'esso ricorsivamente enumerabile, e per il teorema di Post concludiamo che sono entrambi ricorsivi. □

16.2 Enunciati indimostrabili

Consideriamo il linguaggio $L = \{+, \cdot, S, 0\}$.

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

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

Se $\ulcorner\alpha\urcorner = n \in \mathbb{N}$, indichiamo con $\overline{\ulcorner\alpha\urcorner}$ il termine $S^n(0)$. Data una formula α e un termine t, scriviamo $\alpha(t)$ per $\alpha[t/v_0]$.

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

Dimostrazione. $D(x) = \mathbf{sub}(x, 0, \mathbf{num}(x))$. □

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

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

16.13 Teorema. *La teoria PA (aritmetica di Peano del primo ordine) è incompleta, ovvero esistono enunciati ϕ tali che $PA \nvdash \phi$ e $PA \nvdash \neg\phi$.*

Dimostrazione. Poiché le codifiche degli assiomi di PA sono un insieme primitivo ricorsivo 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 $\mathbb{N}$, si ha in particolare che G è vera (in $\mathbb{N}$) 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 pertanto l'enunciato $\neg\mathrm{Teo}(\overline{\lceil G\rceil})$ è vero (in $\mathbb{N}$) in quanto esprime proprio la non dimostrabilità di G. Ma G equivale in tutti i modelli di Q a tale enunciato, e quindi G è vero in $\mathbb{N}$. Ma allora nemmeno la negazione di G è dimostrabile in PA in quanto PA dimostra solo cose vere in $\mathbb{N}$ (essendo $\mathbb{N}$ un suo modello). □

Con essenzialmente la stessa dimostrazione si ha:

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

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

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

16.17 Lemma. *Sia T una teoria ω-coerente contenente Q e sia σ un enunciato $\vec{\exists}\Delta_0$. Allora $T \vdash \sigma$ se e solo se σ è vero nel modello $\mathbb{N}$.*

Dimostrazione. Consideriamo il caso in cui σ è della forma $\exists x\theta(x)$ con $\theta \in \Delta_0$. Se σ è vero in $\mathbb{N}$, allora per qualche $n \in \mathbb{N}$, $\theta(\overline{n})$ è vero in $\mathbb{N}$, 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 σ fosse falso in $\mathbb{N}$, allora per ogni n varrebbe in $\mathbb{N}$ l'enunciato $\neg\theta(\overline{n})$ e pertanto questo enunciato sarebbe dimostrabile in Q (essendo Δ_0) e quindi in T, contraddicendo la ω-coerenza. □

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

Dimostrazione. 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 $\exists n : (n, \ulcorner\varphi\urcorner) \in$ **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 $(n, \ulcorner G\urcorner) \in$ **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 $(n, \ulcorner G\urcorner) \notin$ **Prov**, da cui discende che $\neg P(\overline{n}, \overline{\ulcorner G\urcorner})$ è dimostrabile in Q e quindi in T. Ma allora per la ω-coerenza $T \nvdash \exists x P(x, \overline{\ulcorner G\urcorner})$, e per definizione di G, $T \nvdash \neg G$. □

16.19 Teorema. *Sia T una teoria ω-coerente, contenente Q, e tale che l'insieme dei numeri di Gödel dei suoi assiomi è ricorsivo. Allora T è indecidibile (nel senso che l'insieme dei suoi teoremi non è ricorsivo). In particolare sia Q che PA sono indecidibili.*

Dimostrazione. Sia $K_0 \subset \mathbb{N}$ un insieme ricorsivamente enumerabile e non ricorsivo. Essendo ricorsivamente enumerabile K_0 è definibile in $\mathbb{N}$ da una formula $\sigma(x) \in \vec{\exists}\Delta_0$ in una variabile libera. Per ogni $n \in \mathbb{N}$, $n \in K_0$ se e solo se $\sigma(\overline{n})$ è vera in $\mathbb{N}$, e per la ω-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. □

16.20 Osservazione. 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 ω coerenza di può indebolire nell'ipotesi di semplice coerenza.

Si può dimostrare che la formula G nella dimostrazione del Teorema 16.13 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.

16.21 Teorema. $PA \nvdash Con(PA)$.

Dimostrazione. (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 $\neg Teo(\ulcorner G\urcorner)$, che ovviamente implica $\exists x \neg Teo(x)$. L'implicazione inversa $Con(PA) \rightarrow G$ si ottiene formalizzando in PA la prima parte della dimostrazione del Teorema 16.13, quella cioè in cui si mostra $PA \nvdash G$. In tale dimostrazione si usa solo la coerenza di PA (e non che $\mathbb{N}$ sia un modello).

Formalizzando il ragionamento si ottiene una dimostrazione, in PA, dell'implicazione $Con(PA) \to \neg Teo(\lceil G \rceil)$. Poiché $\neg Teo(\lceil G \rceil)$ equivale dimostrabilmente a G, si ha $PA \vdash Con(PA) \to G$. □

16.22 Teorema. *(Church) Il calcolo dei predicati è indecidibile.*

Dimostrazione. Si ha $Q \vdash \phi$ se e solo se $\vdash \bigwedge Q \to \phi$, dove $\bigwedge Q$ indica la congiunzione di tutti gli assiomi di Q (che sono in numero finito). Ne segue che se avessimo un algoritmo per stabilire se una formula è logicamente valida, applicandolo alla formula $\bigwedge Q \to \phi$ otterremmo un algoritmo per stabilire se una formula ϕ è teorema di Q, che abbiamo visto essere impossibile. □