UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Matematica e Informatica

Corso di Laurea Triennale in Matematica

Tesi di Laurea

AUTOIMMERSIONI DI ORDINI LINEARI E GRADI DI TURING

Relatore:
Prof. ALBERTO MARCONE

Laureando:
ANDREA CETTOLO

ANNO ACCADEMICO 2012-2013

Indice

Introduzione

Il campo di ricerca che viene comunemente chiamato Matematica Computabile consiste nell'analisi degli aspetti effettivi dei risultati ottenuti nelle altre aree della Matematica.

Mentre le radici e le motivazioni storiche della Matematica Computabile possono essere ricondotte alle filosofie dei matematici costruttivisti, essa non può essere considerata interna alla Matematica Costruttiva (o Contenutistica). Per compiere tale indagine vengono infatti usati gli strumenti e le tecniche proprie della moderna Teoria della Computabilità la quale si sviluppa strettamente all'interno della Logica Classica.

Nonostante ciò, molti risultati della Matematica Computabile hanno un loro interesse in termini costruttivi. È il caso di questa breve trattazione, il cui obiettivo è quello di analizzare alcuni aspetti computabili di un classico Teorema di Dushnik-Miller ([DM40]), ed i cui risultati principali consistono di vere e proprie costruzioni effettive. In particolare costruiremo ordini lineari e studieremo la complessità computazionale delle loro autoimmersioni, ovvero di funzioni che agiscono su di essi e ne rispettano l'ordine.

Nel corso della trattazione si è cercato di fornire il maggior numero possibile di nozioni necessarie alla comprensione dei risultati più importanti. Per questo motivo il lavoro è organizzato in maniera tale da presentare, con l'avanzare dei capitoli, costruzioni via via più raffinate, le quali richiederanno nozioni sempre più avanzate di Teoria della Computabilità. Per tutta la parte riguardante le nozioni teoriche di Teoria della Computabilità è stato usato il testo di R.I. Soare ([So87]) come riferimento principale.

Nel primo capitolo si introducono i semplici concetti di ordine lineare, di ordine lineare denso, discreto, di intervallo e vengono esibiti alcuni esempi elementari. Viene quindi data l'importante definizione di autoimmersione. La seconda sezione di questo capitolo presenta invece i classici risultati ottenuti da Dushnik e Miller sugli ordini lineari. Tra questi, particolare importanza in questo lavoro gioca il Teorema di Dushnik-Miller, il quale garantisce l'esistenza di autoimmersioni non banali per ogni ordine lineare infinito e numerabile. Vedremo inoltre come questo risultato fallisce nel caso di ordini più che numerabili.

Il secondo capitolo inizia con un breve richiamo alle nozioni basilari di Teoria della Computabilità, come, per esempio, quelli di funzione ricorsiva, computabile e di stadio di computazione. Sono inoltre presentati, senza dimostrazione, alcuni risultati tecnici fondamentali come il Teorema di forma normale di Kleene, il Teorema di enumerazione e il Teorema s-m-n. Nella seconda sezione si definiscono gli insiemi c.e. e il concetto di m-riducibiltà. Per completezza, vengono inoltre presentati due classici risultati: la soluzione del "problema della fermata" e il Teorema di punto fisso.

Il primo risultato sugli ordini lineari computabili è presentato nella terza sezione di questo capitolo. Infatti dopo un semplice esempio, utile a prendere confidenza con questo tipo di costruzioni, viene affrontato il problema dell'effettivizzazione del Teorema di Dushnik-Miller. Ovvero di determinare se ogni ordine lineare infinito e computabile ammetta autoimmersioni non banali anch'esse computabili. Vedremo, con il Teorema di Hay-Rosenstein (presentato in [Ro82]), che è possibile costruire un controesempio a questa affermazione.

L'impossibilità di trovare sempre autoimmersioni computabili apre la strada al problema di determinare quale sia la complessità necessaria a calcolarle. Per questo, nel terzo capitolo, si esegue un lavoro di relativizzazione dei concetti visti nel capitolo 2. In particolare si definisce cosa si intende per funzioni e insiemi A-computabili. Questi nuovi concetti permettono di introdurre il nozione di Turing-riducibilità ($\leq_T$), la quale determina una relazione di equivalenza tra i sottoinsiemi di $\mathbb{N}$ ed i cui elementi vengono chiamati gradi di Turing. Inoltre la relativizzazione del "problema della fermata" porta all'importantissima definizione di jump di un insieme A (che si indica con A'), le cui proprietà sono analizzate nel Teorema del jump presentato alla fine della seconda sezione.

La caratteristica fondamentale del jump è quella di possedere una complessità computazionale strettamente maggiore a quella dell'insieme di partenza. Il Teorema di Downey-Lempp ([DL99]) presentato nella terza sezione utilizza proprio questa caratteristica per migliorare quello di Hay-Rosenstein visto nel primo capitolo. Anche in questo caso si effettua una costruzione di un ordine lineare computabile, le cui autoimmersioni non banali siano in grado di computare il jump di $\emptyset$ (che rappresenta la forza computazionale degli insiemi computabili). Questo risultato afferma in particolare che, per riuscire a calcolare sempre almeno un'autoimmersione non banale dobbiamo necessariamente essere in grado di computare $\emptyset'$, determinando quindi un primo limite inferiore.

Il capitolo 4 ha due obiettivi principali (entrambi esposti in [DJM06]): garantire un limite superiore alla complessità necessaria a computare le autoimmersioni, e migliorare il Teorema di Downey-Lempp. Per questo motivo le prime due sezioni del quarto capitolo sono dedicate ad introdurre nuove definizioni teoriche come quelle di insieme aritmetico e di grado PA. Nella prima sezione è di fondamentale importanza il Teorema di Post, che pone in relazione le nozioni di jump e di insieme aritmetico, mentre la seconda sezione si occupa dei gradi PA e del loro significato. Acquisiti tali concetti, nella terza sezione si dimostra che $\emptyset''$ (l'operazione di jump applicata due volte su $\emptyset$) è in grado di calcolare sempre almeno un'autoimmersione non banale di ogni ordine lineare computabile infinito. Infine la quarta sezione presenta la costruzione, dovuta a Downey-Jockusch-Miller, di un ordine lineare computabile infinito, le cui autoimmersioni non banali hanno una complessità computazionale strettamente maggiore a $\emptyset'$ (diremo di grado PA su $\mathbf{0}'$).

I concetti di jump e grado PA non coincidono. In particolare esistono insiemi di grado PA su $\mathbf{0}'$ che non computano $\emptyset''$. Rimane quindi aperto il problema di determinare, se esiste, la minima complessità computazionale in grado di calcolare autoimmersioni non banali. Una possibile soluzione è quella di costruire un ordine lineare le cui autoimmersioni non banali siano in grado di calcolare $\emptyset''$. L'altra soluzione consiste nel dimostrare l'esistenza di un grado PA su $\mathbf{0}'$, strettamente minore di $\mathbf{0}''$, capace di calcolare un'autoimmersione non banale di ogni ordine lineare computabile e che è minimale rispetto a questa proprietà. Attualmente non è ancora stata data una risposta definitiva al problema.

Capitolo 1

Definizioni e risultati preliminari

Nella prima sezione di questo capitolo verranno richiamate brevemente alcune nozioni basilari sugli ordini. Nella seconda sezione saranno esposti i risultati classici ottenuti da Dushnik e Miller ([DM40]) sulle autoimmersioni di ordini lineari.

1.1 Definizioni

Definizione 1.1. Un ordine lineare su un insieme A è una relazione binaria R (su A) soddisfacente le seguenti condizioni:

1. se $(x, y) \in R$ e $(y, z) \in R$ allora $(x, z) \in R$,
2. se $x \notin y$ allora $(x, y) \in R$ oppure $(y, x) \in R$ (esattamente uno),
3. per ogni $x \in A$ $(x, x) \notin R$.

D'ora in avanti indicheremo le relazioni d'ordine con il simbolo $<$ e scriveremo $x < y$ al posto di $(x, y) \in R$. Inoltre utilizzeremo la notazione $x \leq y$ per indicare $x < y \vee x = y$.

Definizione 1.2. Dato un insieme linearmente ordinato $(A, \leq)$ e $I \subseteq A$, diremo che I è un intervallo di A se, per ogni $x, y \in I$ con $x \leq y$, vale

$$\forall z\ (x \leq z \leq y \rightarrow z \in I). \tag{1.1}$$

Osservazione 1.3. Se $(A, \leq)$ è un ordine lineare e $x, y \in A$, gli insiemi $\{z : x < z < y\}$, $\{z : x \leq z < y\}$, $\{z : x < z \leq y\}$, $\{z : x \leq z \leq y\}$ sono tutti intervalli di A. Nel seguito per indicare tali intervalli useremo rispettivamente le seguenti notazioni (x, y), $[x, y)$, $(x, y]$, $[x, y]$. Se $x \in A$ gli insiemi $\{z : z < x\}$, $\{z : z \leq x\}$, $\{z : x < z\}$, $\{z : x \leq z\}$ sono anch'essi intervalli di A (che indicheremo rispettivamente con $(-\infty, x)$, $(-\infty, x]$, $(x, +\infty)$, $[x, +\infty)$).

Definizione 1.4. Un ordine lineare $(A, \leq)$ si dice discreto se valgono le due seguenti condizioni:

- $\exists z\ (z < x) \rightarrow \exists t < x\ \forall y < x\ (y \leq t)$,
- $\exists z\ (z > x) \rightarrow \exists t > x\ \forall y > x\ (y \geq t)$.

Definizione 1.5. Un ordine lineare $(A, \leq)$ si dice denso se

$$\forall x, y\ (x < y \rightarrow \exists z\ (x < z < y)). \tag{1.2}$$

Esempio 1.6. Classici esempi di ordini lineari sono dati dagli insiemi numerici $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ e dai loro ordini naturali. L'ordine dei numeri naturali viene denotato con la lettera ω. L'insieme $\{x \in \mathbb{Z} : x \leq 0\}$ con l'ordine ereditato dai numeri interi si indica con ω^*.

Definizione 1.7. Siano $(A, \leq_A)$ e $(B, \leq_B)$ due ordini lineari disgiunti. Definiamo l'ordine lineare $A + B = (A \cup B, \leq)$ nel seguente modo: $x < y$ se e solo se vale una delle seguenti tre condizioni:

- $x \in A$ e $y \in B$,
- $x, y \in A$ e $x <_A y$,
- $x, y \in B$ e $x <_B y$.

Definizione 1.8. Un ordine lineare $(A, \leq)$ è un buon ordine se ogni suo sottoinsieme non vuoto ha un elemento minimo o, equivalentemente, se non esistono catene discendenti infinite.

Esempio 1.9. Nella Teoria degli Insiemi la relazione di inclusione $\subset$ (equivalentemente la relazione di appartenenza $\in$) sulla classe degli ordinali determina un ordine lineare che è anche un buon ordine.

Definizione 1.10. Siano $(A, \leq_A)$, $(B, \leq_B)$ due ordini lineari e sia $f : A \rightarrow B$ funzione. Diremo che f preserva (o rispetta) l'ordine se

$$\forall x, y \in A\ (x <_A y \iff f(x) <_B f(y)). \tag{1.3}$$

Una funzione con tali caratteristiche viene anche chiamata immersione. Se f è una biezione diremo che $(A, \leq_A)$ e $(B, \leq_B)$ sono isomorfi.

Osservazione 1.11. È immediato verificare che ogni immersione è iniettiva.

Esempio 1.12. Se consideriamo ω e ω^* come insiemi disgiunti, possiamo definire l'ordine $\omega+\omega^*$. È facile convincersi che tale insieme è numerabile, discreto, ha massimo e minimo e non è un buon ordine.

Nel seguito, denoteremo con 1 un generico ordine composto da un singolo elemento. Dato quindi un ordine A, sono ben definiti, a meno di isomorfismi, gli insiemi $A+1$, $1+A$ (in cui si intende sempre $1 \cap A = \emptyset$).

In seguito verrà utilizzato spesso il seguente importante risultato:

Lemma 1.13. *Ogni ordine lineare numerabile è immergibile in* $\mathbb{Q}$.

Dimostrazione. Sia $A = \{a_i : i \in \mathbb{N}\}$ ordine lineare numerabile. Definiamo induttivamente una successione di funzioni $(\pi_i)_{i\in\mathbb{N}}$ tale che per ogni $i \in \mathbb{N}$ valgano: (1) $\pi_i : \{a_0, \dots, a_i\} \to \mathbb{Q}$, (2) $\pi_i \subset \pi_{i+1}$, (3) π_i rispetta l'ordine. Supponiamo quindi di aver definito π_i. Dato che π_{i+1} deve estendere π_i, dobbiamo solamente determinare $\pi_{i+1}(a_{i+1})$. Si possono verificare i seguenti casi:

1. $a_{i+1} <_A a_k$ per ogni $k = 0, \dots, i$. In tal caso scegliamo $b \in \mathbb{Q}$ tale che $b < \pi_i(a_k)$ per ogni $k = 0, \dots, i$ e poniamo $\pi_{i+1}(a_{i+1}) = b$.

2. $a_{i+1} >_A a_k$ per ogni $k = 0, \dots, i$. Analogamente ad (1) scegliamo $b \in \mathbb{Q}$ tale che $b > \pi_i(a_k)$ per ogni $k = 0, \dots, i$ e poniamo $\pi_{i+1}(a_{i+1}) = b$.

3. Esistono $0 \le l, k \le i$ tali che $a_k <_A a_{i+1} <_A a_l$ e per ogni $j \in \{0, \dots, i\}$ se $a_k <_A a_j$ allora $a_l \le a_j$ (ovvero i due elementi sono consecutivi). In questo caso scegliamo $b \in \mathbb{Q}$ tale che $\pi_i(a_k) < b < \pi_i(a_l)$ (che esiste perché $\mathbb{Q}$ è denso). È immediato verificare che $\pi = \cup_{i\in\mathbb{N}}\pi_i$ è un'immersione di A in $\mathbb{Q}$.

□

Osservazione 1.14. Si può dimostrare (si veda [Ro82, Corollary 2.9]) che ogni ordine lineare numerabile e denso è isomorfo ad uno dei seguenti: $\mathbb{Q}$, $1+\mathbb{Q}$, $\mathbb{Q}+1$, $1+\mathbb{Q}+1$.

Osservazione 1.15. Un importante risultato in Teoria degli Insiemi è che ogni insieme ben ordinato è isomorfo ad un unico ordinale. È quindi equivalente parlare di buoni ordini o di ordinali.

Definizione 1.16. Dato un ordine lineare $(A, \le)$, una sua autoimmersione è una funzione $f : A \to A$ che rispetta l'ordine.

Osserviamo che ogni ordine lineare ammette almeno l'identità come autoimmersione. Le autoimmersioni diverse dall'identità vengono anche dette non banali.

1.2 Risultati di Dushnik-Miller

In questa sezione vengono affrontati essenzialmente tre problemi:

- Determinare quando un ordine lineare è isomorfo ad un suo sottoinsieme proprio.
- Determinare quando un insieme linearmente ordinato ammette un'autoimmersione non banale.
- Dato un ordine lineare A tale che per ogni sua autoimmersione f vale $f(x) \geq x$ per ogni $x \in A$, determinare se A è un buon ordine.

È evidente che un ordine lineare isomorfo ad un suo sottoinsieme proprio ammette un'autoimmersione non banale (basta infatti considerare l'isomorfismo in questione). La proposizione che segue dimostra l'implicazione opposta.

Proposizione 1.17. *Ogni ordine lineare che ammette un'autoimmersione non banale è isomorfo ad un suo sottoinsieme proprio.*

Dimostrazione. Sia A un insieme linearmente ordinato e sia $f : A \to A$ una sua autoimmersione diversa dall'identità. Allora deve esistere $x \in A$ tale che $f(x) \neq x$. Supponiamo che $f(x) > x$. Possiamo allora definire $g : A \to A$ nel seguente modo:

$$g(y) := \begin{cases} y & \text{se } y < x, \\ f(y) & \text{se } y \geq x. \end{cases} \tag{1.4}$$

Tale g rispetta l'ordine e non è suriettiva (infatti $x \notin \operatorname{ran}(g)$). A è pertanto isomorfo a $\operatorname{ran}(g)$ che è un suo sottoinsieme proprio. Nel caso in cui $f(x) < x$ si procede in maniera analoga. $\square$

La proposizione precedente sancisce quindi l'equivalenza dei primi due problemi posti ad inizio sezione. Il seguente teorema risolve il secondo (e perciò anche il primo) problema nel caso numerabile.

Teorema 1.18 (Teorema di Dushnik-Miller)**.** *Ogni ordine lineare numerabile infinito ammette un'autoimmersione non banale.*

Dimostrazione. Sia $(A, \leq)$ un ordine lineare numerabile infinito. Definiamo su A una relazione d'equivalenza $\equiv$ nel seguente modo:

$$x \equiv y \iff [x, y] \cup [y, x] \text{ è finito.} \tag{1.5}$$

Gli elementi del quoziente $\{[x] : x \in A\}$ vengono chiamati blocchi. È facile convincersi che ogni blocco è un intervallo in A. Possono verificarsi due casi:
Caso 1. Esiste un blocco infinito.

Sia x tale che $[x]$ è infinito. Per come è stata definita la relazione d'equivalenza, l'ordine su $[x]$ deve essere discreto ed inoltre per ogni $y, z \in [x]$ con $y < z$ l'intervallo $[y, z]$ è necessariamente finito. Quindi l'insieme $[x]$ è isomorfo a uno tra ω, ω^* o $\mathbb{Z}$. Supponiamo per esempio che $[x]$ abbia tipo d'ordine ω. Dato che ogni elemento di A che non appartiene ad $[x]$ precede o segue tutti gli elementi di $[x]$, possiamo scrivere A come unione di tre insiemi disgiunti A_1, A_2, A_3 tali che: A_1 consiste degli elementi di A che precedono tutti gli elementi di $[x]$, A_3 di quelli che seguono tutti gli elementi di $[x]$ e $A_2 = [x]$. Possiamo ora definire una funzione $f : A \to A$ nel seguente modo:

- se $y \in A_1$ o $y \in A_3$ poniamo $f(y) = y$,
- se $y \in A_2$ allora $f(y)$ è il successore di y in $[x]$.

È evidente che f è un'autoimmersione non banale di A.

Nel caso in cui $[x]$ sia isomorfo a $\mathbb{Z}$ si ripete la medesima costruzione. Se $[x]$ ha tipo d'ordine ω^*, occorre solamente cambiare la definizione di f in A_2 ponendo $f(y)$ pari al predecessore di y.
Caso 2. Ogni blocco è finito.

Consideriamo $\overline{A}$, il sottoinsieme ordinato di A consistente di tutti i primi elementi delle classi di equivalenza di A, e siano $x, y \in \overline{A}$ distinti. Allora deve esistere $z \in \overline{A}$ tale che $x < z < y$. Altrimenti avremmo chiaramente $x \equiv y$, contro l'ipotesi che x, y siano elementi distinti di $\overline{A}$. Pertanto $\overline{A}$ è denso. Per l'Osservazione 1.14 $\overline{A}$ è isomorfo a $\mathbb{Q}, 1 + \mathbb{Q}, \mathbb{Q} + 1$ o $1 + \mathbb{Q} + 1$. Esiste quindi un'immersione $g : \mathbb{Q} \to \overline{A}$ $(\subseteq A)$. Per il Lemma 1.13 A è immergibile in $\mathbb{Q}$. Inoltre, sempre per l'Osservazione 1.14, l'intervallo $(0, 1)$ $(\subset \mathbb{Q})$ è isomorfo a $\mathbb{Q}$. Perciò esiste un'immersione $f : A \to (0, 1)$. Allora la funzione $g \circ f : A \to A$ rispetta l'ordine e non è suriettiva (per esempio l'elemento $g(0) \in A$ non ha controimmagine) ed è quindi un'autoimmersione non banale di A. □

Analogamente a quanto succede per i primi due problemi, il prossimo teorema garantisce una risposta positiva al terzo (sempre nel caso numerabile).

Per le prossime dimostrazioni faremo uso di alcuni tipici concetti della Teoria deglli insiemi.

Teorema 1.19. *Se A è un ordine lineare numerabile tale che ogni sua autoimmersione f ha la proprietà che $f(x) \geq x$ per ogni $x \in A$, allora A è un buon ordine.*

Dimostrazione. Osserviamo prima di tutto che A non può contenere nessun sottoinsieme denso $\overline{A}$. Supponiamo infatti per assurdo che $\overline{A} \subseteq A$ sia denso e fissiamo $y \in \overline{A}$. L'insieme $B = (-\infty, y) \cap \overline{A}$ è numerabile e denso. Per l'Osservazione 1.14 ed il Lemma 1.13, A è isomorfo ad un sottoinsieme B' di B. Perciò esiste un'autoimmersione f di A che ha come immagine B'. Dato che $f(y) \in B'$ e tutti gli elementi di B' precedono y, abbiamo che $f(y) < y$ contro l'ipotesi del teorema.

Definiamo ora una relazione di equivalenza sull'insieme A nel seguente modo:

$$x \equiv y \iff [x, y] \cup [y, x] \text{ è ben ordinato.} \tag{1.6}$$

Facciamo alcune facili osservazioni sulle classi di equivalenza:

- Ogni classe di equivalenza è un intervallo in A.
- Ogni intervallo chiuso o aperto di una classe di equivalenza è un insieme ben ordinato.
- Se $[x]$ è una tale classe e $y \in [x]$, ogni insieme del tipo $(y, +\infty) \cap [x]$ è ben ordinato.
- Se $[x]$ possiede un elemento minimo, $[x]$ e ogni insieme del tipo $(-\infty, y) \cap [x]$ sono ben ordinati.

Dimostriamo ora che ogni classe di equivalenza ha elemento minimo.

Supponiamo che esista una classe di equivalenza senza elemento minimo. Dimostreremo che questa ipotesi porta ad una contraddizione. Sia $[x]$ tale classe. Possiamo quindi trovare in $[x]$ una sequenza di elementi $x_1 > x_2 > x_3 > \cdots > x_n > \ldots$ tale che per ogni $y \in [x]$ esiste un n tale che $x_n < y$. Osserviamo che per ogni $i > 0$ l'intervallo $[x_{i+1}, x_i)$ è un insieme ben ordinato e pertanto isomorfo ad un ordinale β_i. Consideriamo la successione di ordinali $\beta_1, \beta_2, \ldots, \beta_n, \ldots$. Può esserci solo un numero finito di indici k con la proprietà che per ogni $n > k$ $\beta_n < \beta_k$, altrimenti otterremmo una catena discendente infinita di ordinali, sia r massimo tra questi. Allora per ogni $q > r$ esiste un indice $n > q$ tale che $\beta_q \leq \beta_n$. Iterando questa osservazione abbiamo che per ogni $q > r$ esistono infiniti indici $n > q$ tali che $\beta_q \leq \beta_n$. È quindi possibile definire una successione crescente di indici $q_{r+1} < q_{r+2} < q_{r+3} < \ldots$ tale che per ogni $i > r$ si abbia $i < q_i$ ed esista un'immersione $\gamma_i : [x_{i+1}, x_i) \to [x_{q_i+1}, x_{q_i})$. Spezziamo quindi A nell'unione di tre sottoinsiemi A_1, A_2, A_3, dove $A_2 = [x] \cap (-\infty, x_{r+1})$, A_1 consiste degli elementi di A che precedono tutti quelli di A_2 e $A_3 = [x_{r+1}, +\infty)$. Definiamo ora la seguente funzione f su A:

- se $y \in A_1$ o $y \in A_3$ poniamo $f(y) = y$,

- se $y \in A_2$, esisterà un indice $i > r$ con $y \in [x_i + 1, x_i)$.Poniamo allora $f(y) = \gamma_i(y)$.

Questa funzione f è chiaramente un'autoimmersione di A con la proprietà che per ogni $y \in A_2$ si ha $f(y) < y$. Ma questo è impossibile per l'ipotesi del teorema. Ogni classe d'equivalenza è quindi un buon ordine.

Per poter concludere la dimostrazione proviamo che esiste un'unica classe di equivalenza.

Supponiamo che A contenga due classi di equivalenza $[x] \neq [y]$ dove, senza perdita di generalità, supponiamo che x e y siano i primi elementi delle rispettive classi e che $x < y$. Se non esistesse nessun elemento z che separa le due classi avremmo che l'insieme $D = [x] \cup [y]$ sarebbe un intervallo ben ordinato e quindi $x \equiv y$, contro l'ipotesi. Perciò esiste una classe $[z]$ (con z elemento minimo della classe), distinta dalle altre due, con $x < z < y$. Iterando l'osservazione abbiamo che l'insieme $A' \subseteq A$ composto dagli elementi minimi delle classi di equivalenza è un insieme denso. Ma ciò è assurdo per quanto visto ad inizio dimostrazione, quindi A ammette un'unica classe. L'insieme A coincide quindi con l'unico blocco presente ed è pertanto un buon ordine. $\square$

Nel caso di ordini lineari più che numerabili i due risultati ottenuti sopra non valgono. Per dimostrare questo fatto abbiamo bisogno del seguente lemma ([Ro82, Lemma 9.3]).

Lemma 1.20. *L'insieme* $\{f : \mathbb{R} \to \mathbb{R} :$ *f è strettamente crescente*$\}$ *ha cardinalità* $2^{\aleph_0}$.

Dimostrazione. Sia $f : \mathbb{R} \to \mathbb{R}$ strettamente crescente. Diciamo che f ha una discontinuità in $x \in \mathbb{R}$ se $\sup_{y<x} f(y) < \inf_{y>x} f(y)$.

Vogliamo dimostrare che f ammette al più una quantità numerabile di discontinuità. Infatti se f è discontinua in $x \in \mathbb{R}$ esiste un razionale r_x con $\sup_{y<x} f(y) < r_x < \inf_{y>x} f(y)$. Inoltre, dato che f è strettamente crescente, se f è discontinua in $y > x$ allora sarà necessariamente $r_y > r_x$. L'applicazione $x \mapsto r_x$ è quindi iniettiva, perciò il numero di discontinuità è al più numerabile.

Consideriamo ora $g : \mathbb{Q} \to \mathbb{Q}$ strettamente crescente. Se vogliamo estenderla su $\mathbb{R}$ (mantenendola strettamente crescente) è chiaro che nei punti $x \in \mathbb{R} \setminus \mathbb{Q}$ in cui $\sup\{f(y) : y \in \mathbb{Q} \wedge y < x\} = \inf\{f(y) : y \in \mathbb{Q} \wedge y > x\}$ la scelta è obbligata. Negli altri dobbiamo solo mantenere la condizione $\sup\{f(y) : y \in \mathbb{Q} \wedge y < x\} \leq f(x) \leq \inf\{f(y) : y \in \mathbb{Q} \wedge y > x\}$.

Per definire una funzione strettamente crescente $f : \mathbb{R} \to \mathbb{R}$ è pertanto sufficiente definirla su $\mathbb{Q}$ e nei punti di discontinuità (al più numerabili). Quindi il numero di scelte possibili è $(2^{\aleph_0})^{\aleph_0} = 2^{\aleph_0}$. $\square$

Teorema 1.21. *L'insieme dei numeri reali* $\mathbb{R}$ *contiene un insieme* E*, di cardinalità più che numerabile, che non ammette autoimmersioni non banali.*

Dimostrazione. Per il lemma precedente esistono $2^{\aleph_0}$ autoimmersioni di $\mathbb{R}$. Se denotiamo con $\mathfrak{c}$ il cardinale corrispondente al continuo, possiamo ben ordinare l'insieme delle autoimmersioni non banali in una successione del tipo $f_0, f_1, \dots, f_\alpha, \dots$ con $\alpha < \mathfrak{c}$.

Ora osserviamo che, fissata una di queste autoimmersioni, l'insieme dei suoi punti fissi non può essere denso ovunque in $\mathbb{R}$. Altrimenti, scelto $x \in \mathbb{R}$ con $x < f(x)$ (o $f(x) < x$), esisterebbe un punto fisso $y \in [x, f(x)]$ (o $[f(x), x]$), assurdo perche f rispetta l'ordine. Pertanto se f è una tale trasformazione esistono $2^{\aleph_0}$ punti y tali che $f(y) \neq y$.

Definiamo allora $\mathfrak{c}$ coppie $x_\beta \neq y_\beta$ con $\beta < \mathfrak{c}$ nel seguente modo: supposto di aver definito x_β, y_β per ogni $\beta < \alpha < \mathfrak{c}$, scegliamo $x_\alpha, y_\alpha \in \mathbb{R} \setminus \bigcup_{\beta<\alpha} \{x_\beta, y_\beta\}$ tali che $f_\alpha(x_\alpha) = y_\alpha \neq x_\alpha$. Queste scelte sono sempre possibili per ogni $\alpha < \mathfrak{c}$ in quanto la cardinalità di $\bigcup_{\beta<\alpha} \{x_\beta, y_\beta\}$ è minore di $\mathfrak{c}$.

Definiamo ora $E = \{x_\alpha : \alpha < \mathfrak{c}\}$. Notiamo prima di tutto che E è un insieme denso in $\mathbb{R}$. Infatti scelto un qualsiasi intervallo $[a, b]$ in $\mathbb{R}$ è facile costruire una funzione strettamente crescente non banale $g : \mathbb{R} \to \mathbb{R}$ che sia l'identità fuori $[a, b]$ (si può per esempio porre $g(x) = x/2 + a/2$ per ogni $x \in [a, b]$). Scelti quindi due elementi $x, y \in E$ con $x < y$ esisterà un $\beta < \mathfrak{c}$ tale che $x < x_\beta < y$.

Vogliamo ora dimostrare che tale insieme E non ammette autoimmersioni non banali. Infatti supponiamo che $\tau : E \to E$ sia una trasformazione strettamente crescente diversa dall'identità. È facile convincersi che, data la densità di E, esiste un'estensione strettamente crescente di τ in $\mathbb{R}$. Scelto infatti $x \in \mathbb{R} \setminus E$ basta porre $\tau'(x) = \sup\{\tau(y) : y \in E \wedge y < x\}$. Pertanto esiste un α tale che f_α e τ assumono gli stessi valori in E. Ma ciò è impossibile dato che $f_\alpha(x_\alpha) = y_\alpha \notin E$. □

Osservazione 1.22. L'insieme E costruito nella dimostrazione del teorema precedente non è ben ordinato (è denso). Inoltre se f è un'autoimmersione di E allora per ogni $x \in E$ si ha $f(x) \geq x$ (infatti f è necessariamente l'identità). Pertanto l'insieme E risolve negativamente il terzo dei problemi posti all'inizio della sezione nel caso di ordini più che numerabili.

Capitolo 2

Autoimmersioni non computabili

In questo capitolo si comincia a parlare di Teoria della computabilità. Le prime due sezioni (il cui riferimento principale è [So87, Chapters 1,2]) avranno uno scopo introduttivo. La terza propone un notevole risultato dovuto a Hay e Rosenstein (si veda [Ro82, Theorem 16.49]) sull'esistenza di autoimmersioni non computabili.

2.1 Funzioni computabili

In questa trattazione verranno considerate note le nozioni di funzione ricorsiva primitiva, ricorsiva parziale e di macchina a registri illimitati (URM). Diremo che una funzione è calcolabile se esiste un programma per URM che la calcola.

Mediante l'utilizzo di codifiche dei programmi per URM si giunge a dedurre l'equivalenza delle nozioni di funzione ricorsiva parziale e funzione calcolabile, ovvero a determinare il seguente risultato:

Teorema 2.1. $f : A(\subseteq \mathbb{N}^k) \to \mathbb{N}$ *è ricorsiva parziale se e solo se è calcolabile.*

Chiameremo computabili le funzioni di questo tipo. Indicheremo con P_e l'e-simo programma (per URM). Con $\varphi_e^{(n)}$ indicheremo la funzione n-aria calcolata dal programma P_e (ometteremo di indicare l'arietà nel caso $n = 1$). Scriveremo inoltre $\varphi_e^{(n)}(x_1, \dots, x_n) \downarrow$ se $(x_1, \dots, x_n) \in \mathrm{dom}(\varphi_e^{(n)})$ e $\varphi_e^{(n)}(x_1, \dots, x_n) \uparrow$ altrimenti. Utilizzeremo spesso $\vec{x}$ al posto della n-upla $(x_1, \dots, x_n)$).

Dato un insieme $A \subseteq \mathbb{N}^k$ indicheremo con $\chi_A : \mathbb{N}^k \to \mathbb{N}$ la funzione totale tale che $\chi_A(\vec{x}) = 1$ se $\vec{x} \in A$ e $\chi_A(\vec{x}) = 0$ se $\vec{x} \notin A$. Questa verrà chiamata funzione caratteristica dell'insieme A. Mediante la definizione di funzione caratteristica possiamo estendere agli insiemi (e quindi alle relazioni) la nozione di computabilità. Un insieme risulterà computabile se la sua funzione caratteristica è computabile.

A meno che non venga indicato diversamente tratteremo sempre funzioni parziali $\varphi : A(\subseteq \mathbb{N}^{(k)}) \to \mathbb{N}$, elementi $x \in \mathbb{N}^k$ e insiemi $A \subseteq \mathbb{N}^k$.

Elenchiamo ora una serie di risultati preliminari che verranno utilizzati nel seguito.

Lemma 2.2 (Padding Lemma). *Se φ è una funzione computabile n-aria, esistono infiniti indici x tali che $\varphi = \varphi_x^{(n)}$.*

Dimostrazione. È una diretta conseguenza del fatto che, dato un programma, possiamo aggiungere istruzioni che non alterano il comportamento della funzione calcolata. □

Teorema 2.3 (Teorema di forma normale di Kleene). *Per ogni $n \geq 1$ esiste un predicato primitivo ricorsivo $T \subseteq \mathbb{N}^{n+2}$ e una funzione unaria primitiva ricorsiva U tale che*

$$\forall e \in \mathbb{N}\ \forall \vec{x} \in \mathbb{N}^n\ \varphi_e^{(n)}(\vec{x}) = U(\mu y\ T(e, \vec{x}, y)) \tag{2.1}$$

Il predicato di Kleene $T(e, x, y)$ afferma che la computazione di P_e sull'input $\vec{x}$ termina ed è codificata da y.

Teorema 2.4 (Teorema di enumerazione). *Per ogni $n \geq 1$ esiste una funzione computabile $n+1$-aria φ tale che*

$$\forall e \in \mathbb{N}\ \forall \vec{x} \in \mathbb{N}^n\ \varphi_e^{(n)}(\vec{x})) = \varphi(e, \vec{x}) \tag{2.2}$$

Teorema 2.5 (Teorema s-m-n). *Per ogni $n, m \geq 1$ esiste una funzione computabile totale iniettiva $m+1$-aria s_n^m tale che*

$$\forall x \in \mathbb{N}\ \forall \vec{y} \in \mathbb{N}^m\ \forall \vec{z} \in \mathbb{N}^n\ \varphi^{(n)}_{s_n^m(x,\vec{y})}(\vec{z}) = \varphi_x^{(m+n)}(\vec{y}, \vec{z}) \tag{2.3}$$

Osservazione 2.6. La funzione computabile $\langle x, y\rangle = \frac{(x+y)(x+y+1)}{2} + x$ definisce una biiezione tra $\mathbb{N}^2$ e $\mathbb{N}$. Le sue due funzioni inverse sono anch'esse computabili e le chiameremo π_1, π_2.

A partire da queste funzioni è facile definire biiezioni tra $\mathbb{N}^n$ e $\mathbb{N}$ con $n \geq 1$. Le funzioni inverse le indicheremo con $\pi_k^{(n)}$ per $1 \leq k \leq n$. In questo modo è possibile, nella maggior parte dei casi, restringersi a trattare funzioni ed insiemi definiti in $\mathbb{N}$.

Mediante il predicato di Kleene possiamo definire la nozione di s-esimo stadio di computazione nel seguente modo: $\varphi_{e,s}(x) = U(\mu y\ (y \leq s\ \wedge\ T(e, x, y)))$.

Diamo alcuni risultati riguardanti gli stadi di computazione.

Lemma 2.7. *Gli insiemi* $A = \{\langle e, x, s\rangle : \varphi_{e,s}(x) \downarrow\}$ *e* $B = \{\langle e, x, s, z\rangle : \varphi_{e,s}(x) = z\}$ *sono computabili.*

Dimostrazione. Dato $\langle e, x, s\rangle$ è sufficiente esaminare tutti gli $y \leq s$ cercando il minimo tale che $T(e, x, y)$. Se lo trovo allora $\langle e, x, s\rangle \in A$ altrimenti $\langle e, x, s\rangle \notin A$. Nel caso dell'insieme B si procede allo stesso modo, verificando anche che $U(y) = z$. □

Lemma 2.8. *1. Se* $\varphi_{e,s}(x) = z$ *allora* $e, x, s < z$.

2. Per ogni s *esiste al più un* $\langle e, x, z\rangle$ *tale che* $\varphi_{e,s}(x) = z$ *e* $\varphi_{e,s-1}(x) \uparrow$.

Dimostrazione. (1) Se $T(e, x, y)$ allora abbiamo $e, x < y$. Questo dipende dal fatto che y codifica la computazione di P_e su input x. Inoltre abbiamo che $U(y) < y$, pertanto $z < y < s$.

(2) Siano $\varphi_{e,s}(x) = z$ e $\varphi_{e',s}(x') = z'$ con $\varphi_{e,s-1}(x) \uparrow$ e $\varphi_{e',s-1}(x') \uparrow$. Supponiamo inoltre $y, y' < s$ con $T(e, x, y)$ e $T(e', x', y')$. Allora deve essere necessariamente $y = y' = s - 1$. Perciò $z = z'$, $e = e'$ e $x = x'$. □

Nel seguito scriveremo W_e al posto di $\mathrm{dom}(\varphi_e)$ e $W_{e,s}$ per $\mathrm{dom}(\varphi_{e,s})$.

Osservazione 2.9. Abbiamo evidentemente che se $\varphi_{e,s}(x) \downarrow$ allora $\varphi_e(x) \downarrow$. Inoltre se $\varphi_e(x) \downarrow$ allora esiste s tale che $\varphi_{e,s}(x) \downarrow$. Vale quindi la relazione $W_e = \bigcup_s W_{e,s}$.

Per il lemma precedente se $x \in W_{e,s}$ allora $x < s$, quindi ogni $W_{e,s}$ è finito. In particolare per ogni s si ha $|W_{e,s} \setminus W_{e,s-1}| \leq 1$.

Come ultimo risultato della sezione vediamo come sia possibile enumerare in maniera effettiva gli insiemi computabili infiniti.

Proposizione 2.10. *Sia* A *insieme infinito. Allora* A *è computabile se e solo se è immagine di una funzione computabile totale e strettamente crescente.*

Dimostrazione. ($\rightarrow$) Definiamo ricorsivamente la seguente funzione computabile:

- $f(0) = \mu x\ (x \in A)$,
- $f(n+1) = \mu x\ (x \in A \land x > f(n))$.

f è totale per l'infinitezza di A e strettamente crescente.

($\leftarrow$) Sia f computabile, totale e strettamente crescente tale che $\mathrm{ran}(f) = A$. Abbiamo che $n \in A$ se e solo se n viene assunto come valore da f nei primi $n + 1$ input. Questa è una verifica finita e quindi computabile. □

2.2 Insiemi c.e. e il problema della fermata

Definizione 2.11. Un insieme A si dice computabilmente enumerabile (c.e.) se è il dominio di una funzione computabile, ovvero se esiste e tale che $A = W_e$.

Osservazione 2.12. Dato un insieme computabile A possiamo definire la funzione

$$\varphi(x) := \begin{cases} 0 & \text{se } x \in A, \\ \uparrow & \text{se } x \notin A. \end{cases} \tag{2.4}$$

φ è computabile e $\mathrm{dom}(\varphi) = A$, quindi ogni insieme computabile è computabilmente enumerabile.

È naturale porsi il quesito se esistano insiemi c.e. ma non computabili. La seguente proposizione risponde a tale domanda.

Proposizione 2.13 (Halting problem)**.** *L'insieme $K = \{x : \varphi_x(x)\downarrow\}$ è c.e. ma non è computabile.*

Dimostrazione. Per il Teorema di enumerazione la funzione $\psi(x) := \varphi_x(x)$ è computabile. Dato che evidentemente $K = \mathrm{dom}(\psi)$, K è computabilmente enumerabile. Supponiamo ora per assurdo che K sia computabile. Possiamo quindi definire la seguente funzione computabile

$$f(x) := \begin{cases} \varphi_x(x) + 1 & \text{se } x \in K, \\ 0 & \text{se } x \notin K. \end{cases} \tag{2.5}$$

Sia e tale che $f = \varphi_e$. Osserviamo che non può essere $\varphi_e(e)\uparrow$ in quanto f è totale. Perciò abbiamo necessariamente $e \in K$, quindi $f(e) = \varphi_e(e) + 1 \neq \varphi_e(e)$, assurdo. □

Osservazione 2.14. Anche l'insieme $K_0 = \{\langle x, y\rangle : \varphi_x(y)\downarrow\}$ è c.e. ma non computabile. La dimostrazione è analoga a quella vista per l'insieme K.

Nel seguito vedremo che K e K_0 sono insiemi c.e. particolarmente importanti. La loro classificazione prende storicamente il nome di "problema della fermata".

Diamo ora una definizione che concerne la confrontabilità tra insiemi.

Definizione 2.15. Siano $A, B \subseteq \mathbb{N}$, diremo che

- A è molti-a-uno riducibile a B ($A \leq_m B$) se esiste f totale e computabile tale che $\forall x(x \in A \iff f(x) \in B)$.
- A è uno-a-uno riducibile a B ($A \leq_1 B$) se esiste f totale computabile ed iniettiva tale che $\forall x(x \in A \iff f(x) \in B)$.

Osservazione 2.16. È evidente che se $A \leq_1 B$ allora $A \leq_m B$. Se indichiamo con $\overline{A}, \overline{B}$ i complementari di A e B, abbiamo che se $A \leq_* B$ allora $\overline{A} \leq_* \overline{B}$ (con $* \in \{1, m\}$).

La funzione identica garantisce banalmente la riflessività di $\leq_*$. Utilizzando la composizione tra funzioni, è facile verificare anche la transitività di $\leq_*$.

Nei casi in cui avremo $A \leq_* B$ e $B \leq_* A$ scriveremo $A \equiv_* B$, A e B verranno detti *-equivalenti. In questo modo $\equiv_*$ definisce una relazione d'equivalenza tra i sottoinsiemi di $\mathbb{N}$.

Lemma 2.17. $K \equiv_1 K_0$

Dimostrazione. La riduzione $K \leq_1 K_0$ è evidente e si ottiene considerando $f(x) := \langle x, x \rangle$.

Per ottenere la riduzione $K_0 \leq_1 K$ definiamo la seguente funzione computabile $\psi(x, y, z) := \varphi_x(y)$. Per il teorema s-m-n esiste f totale computabile e iniettiva tale che $\psi(x, y, z) = \varphi_{f(x,y)}(z)$. La funzione $g(n) := f(\pi_1(n), \pi_2(n))$ determina la riduzione. Infatti, scelto $\langle x, y \rangle \in K_0$ abbiamo che $\varphi_{f(x,y)}$ è totale (e costante), perciò $f(x, y) \in K$. Se invece $\langle x, y \rangle \notin K_0$ la funzione $\varphi_{f(x,y)}$ diverge sempre, pertanto $f(x, y) \notin K$. □

Vediamo ora alcuni risultati sulla riducibilità.

Proposizione 2.18. *Dati $A, B \subseteq \mathbb{N}$,*

1. *se B è computabile e $A \leq_m B$ allora A è computabile,*
2. *se A è computabile e $B \neq \emptyset$ e $B \neq \mathbb{N}$ allora $A \leq_m B$.*

Dimostrazione. (1) Se f testimonia $A \leq_m B$ abbiamo $\chi_A = \chi_B \circ f$, che è computabile.
(2) Fissiamo $n_0 \in B$ e $n_1 \notin B$ e definiamo

$$f(x) := \begin{cases} n_0 & \text{se } x \in A, \\ n_1 & \text{se } x \notin A. \end{cases} \tag{2.6}$$

Allora f testimonia $A \leq_m B$. □

Proposizione 2.19. *Se B è c.e. e $A \leq_m B$ allora anche A è c.e.*

Dimostrazione. Sia ψ computabile tale che $B = \mathrm{dom}(\psi)$ e f testimone di $A \leq_m B$. Allora si ha che $A = \mathrm{dom}(\psi \circ f)$. □

Sapendo che K è computabilmente enumerabile, possiamo classificare altri insiemi compiendo delle riduzioni.

Esempio 2.20. L'insieme $Tot := \{x : \varphi_x \text{ è totale}\}$ non è computabile in quanto $K \leq_1 Tot$. Per verificarlo definiamo $\psi(x,y) := \varphi_x(x)$, computabile per il teorema di enumerazione. Per il teorema s-m-n esiste f computabile totale iniettiva tale che $\psi(x,y) = \varphi_{f(x)}(y)$. È facile convincersi che $x \in K \iff f(x) \in Tot$. Infatti se $x \in K$ allora $\varphi_{f(x)}(y)$ è costante e totale, perciò $f(x) \in Tot$. Se $x \notin K$ allora $\varphi_{f(x)}(y)$ non converge mai (in particolare non è totale) quindi $f(x) \notin Tot$.

La stessa funzione dimostra inoltre che $K \leq_1 \{x : W_x \neq \emptyset\}$.

Definizione 2.21. A si dice 1-completo se A è c.e. e per ogni insieme B computabilmente enumerabile si ha $B \leq_1 A$.

Proposizione 2.22. *K e K_0 sono 1-completi.*

Dimostrazione. Dato che K e K_0 sono 1-equivalenti è sufficiente dimostrare il risultato solamente per uno dei due. Sia quindi B c.e. e verifichiamo che $B \leq_1 K_0$. Per definizione esisterà un e tale che $B = W_e$. Definiamo allora la seguente funzione calcolabile $f(y) = \langle e, y\rangle$. Allora abbiamo chiaramente $y \in W_e \iff \varphi_e(y)\downarrow \iff \langle e,y\rangle \in K_0$. □

Definizione 2.23.
- Diremo che $A \subseteq \mathbb{N}$ è una proiezione di $R \subseteq \mathbb{N}^2$ se $A = \{x : \exists y (x,y) \in R\}$.
- Diremo che A è Σ_1 se è proiezione di un insieme computabile.

Vediamo ora un'importante caratterizzazione degli insiemi c.e.

Teorema 2.24 (Teorema di forma normale per insiemi c.e.)**.** *A è c.e. se e solo se A è Σ_1.*

Dimostrazione. ($\rightarrow$) Sia e tale che $A = W_e$, allora $x \in A \iff \exists y T(e,x,y)$. Se poniamo $R = \{(x,y) : T(e,x,y)\}$, abbiamo che R è ovviamente computabile e A è proiezione di R.

($\leftarrow$) Sia A proiezione di R computabile, allora A è dominio della seguente funzione computabile $\psi(x) = \mu y\ ((x,y) \in R)$. □

Corollario 2.25 (Teorema di contrazione)**.** *Sia $R \subseteq \mathbb{N}^{n+1}$ computabile, allora l'insieme $A = \{x : \exists y_1 \ldots \exists y_n\ (x, y_1, \ldots, y_n) \in R\}$ è Σ_1.*

Dimostrazione. Definiamo $S := \{(x,z) : (x, \pi_1^{(n)}(z), \ldots, \pi_n^{(n)}(z)) \in R\}$. S è computabile e $A = \{x : \exists z (x,z) \in S\}$. □

Corollario 2.26. *La proiezione di un insieme c.e. è ancora c.e.*

Esempio 2.27. Alcuni esempi di proiezioni di insiemi computabili sono: $K = \{e : \exists s\ \varphi_{e,s}(e)\downarrow\}$, $K_0 = \{\langle e, x\rangle : \exists s\ \varphi_{e,s}(x)\downarrow\}$, $K_1 = \{e : W_e \neq \emptyset\} = \{e : \exists s\ \exists x\ \varphi_{e,s}(x)\downarrow\}$, $\mathrm{ran}(\varphi_e) = \{y : \exists s\ \exists x\ \varphi_{e,s}(x) = y\}$.

Definizione 2.28. Diremo che A è Π_1 se è il complementare di un Σ_1. Se A è Σ_1 e Π_1 diremo che è Δ_1.

Osservazione 2.29. Supponiamo che A sia Π_1. Se indichiamo con $\overline{A}$ il suo complementare, esiste un $R \subseteq \mathbb{N}^2$ tale che $\overline{A} = \{x : \exists y\ (x, y) \in R\}$. Abbiamo quindi che $A = \{x : \forall y\ (x, y) \in \overline{R}\}$.

Teorema 2.30 (Teorema del complemento)**.** *A è computabile se e solo se A è Δ_1.*

Dimostrazione. ($\rightarrow$) Se A è computabile, anche $\overline{A}$ lo è. Quindi sono entrambi c.e., perciò A è Δ_1.

($\leftarrow$) Siano $A = W_e$ e $\overline{A} = W_i$. Sia $f(x) := \mu s\ (x \in W_{e,s} \vee x \in W_{i,s})$ totale e computabile. Allora si ha $x \in A \iff x \in W_{e,f(x)}$ che è una condizione computabile. □

Osservazione 2.31. Diretta conseguenza del precedente teorema è che $\overline{K}$ non è c.e. Infatti se lo fosse avremmo che K sarebbe computabile.

2.3 Teorema di Hay-Rosenstein

Anche in questa sezione lavoreremo rigorosamente all'interno dei numeri naturali. Ogni insieme, relazione (ordine), elemento, funzione sarà definito tramite $\mathbb{N}$.

Definizione 2.32. Un ordine lineare $(A, \leq_A)$ è computabile se $A \subseteq \mathbb{N}$ e $\leq_A \subseteq \mathbb{N}^2$ sono insiemi computabili.

Spesso in Teoria della Computabilità per dimostrare l'esistenza di insiemi (in questo caso ordini lineari) con determinate proprietà si eseguono delle costruzioni a stadi. Ad ogni stadio vengono impartite delle istruzioni da compiere per avanzare nella costruzione. Se le istruzioni sono computabili allora l'intera costruzione finale risulterà computabile.

Come esempio vediamo, prima di tutto, un semplice teorema che risulta essere l'effettivizzazione del seguente,

Teorema 2.33. *Ogni ordine parziale ha un'estensione lineare.*

Per ordine parziale si intende un ordine in cui ci possano essere degli elementi inconfrontabili tra loro. Ciò che si ottiene è:

Teorema 2.34. *Se $(A, \leq_A)$ è un ordine parziale computabile allora esiste $\trianglelefteq$, ordine su A, computabile e totale che estende $\leq_A$.*

Dimostrazione. Se A è un insieme finito anche $\leq_A$ lo è. É facile convincersi che è possibile ampliare $\leq_A$ fino a renderlo totale. L'ordine risultante $\trianglelefteq$ sarà anche lui finito e quindi computabile.

Supponiamo ora che A sia infinito. Sia $\{a_k : k \in \mathbb{N}\}$ un'enumerazione computabile di A. Definiremo $\trianglelefteq$ a stadi. Ad ogni stadio s costruiremo $\trianglelefteq_s$, in modo tale che $\trianglelefteq_s$ sia un ordine lineare che estende $\leq_A$ su $\{a_k : k < s\}$. Inoltre faremo in modo che $\trianglelefteq_s \subseteq \trianglelefteq_{s+1}$.

Stadio 0. Poniamo $\trianglelefteq_0 = \emptyset$.

Stadio s+1. Supponiamo di aver definito $\trianglelefteq_s$ che ordina linearmente $\{a_k : k < s\}$ come $x_0 \trianglelefteq_s x_1 \trianglelefteq_s \cdots \trianglelefteq_s x_{s-1}$. Dobbiamo decidere dove porre a_s. Operiamo quindi nel seguente modo:

- se non esiste $i < s$ con $x_i \leq_A a_s$ poniamo $a_s \trianglelefteq_{s+1} x_i$ per ogni $i < s$,

- altrimenti sia $i_0 < s$ massimo tale che $x_{i_0} \leq_A a_s$. Poniamo allora $x_i \trianglelefteq_{s+1} a_s$ per ogni $i \leq i_0$ e $a_s \trianglelefteq_{s+1} x_i$ per ogni i con $i_0 < i < s$.

Definiamo infine $\trianglelefteq = \bigcup_s \trianglelefteq_s$. Dato che $(A, \leq_A)$ è computabile, la costruzione dei vari $\trianglelefteq_s$ è computabile. Questo implica che anche $\trianglelefteq$ è computabile, infatti per confrontare due elementi, diciamo a_k e a_h, è sufficiente costruire $\trianglelefteq_{max(h,k)+1}$. Con un semplice ragionamento induttivo ci si convince anche che $\trianglelefteq$ è lineare. □

Il Teorema di Dushnik-Miller visto nel primo capitolo garantisce l'esistenza di autoimmersioni non banali nel caso di ordini numerabili. È lecito quindi chiedersi se debba valere anche la sua effettivizzazione, ovvero se un ordine computabile ammetta necessariamente delle autoimmersioni non banali anch'esse computabili. Il teorema [Ro82, Theorem 16.49] che segue risponde negativamente alla domanda.

Teorema 2.35 (Teorema di Hay-Rosenstein)**.** *Esiste un ordine lineare computabile ed infinito $(A, \trianglelefteq)$ tale che ogni sua autoimmersione non banale non è computabile.*

Dimostrazione. L'obiettivo della costruzione che si intende effettuare è quello di definire un ordine $\trianglelefteq$ isomorfo a ω. L'insieme A verrà costruito per stadi. Ad ogni stadio s avremo un insieme A_s finito e un ordine $\trianglelefteq_s$ su di esso. Faremo in modo che per ogni s $A_s \subseteq A_{s+1}$ e $\trianglelefteq_s \subseteq \trianglelefteq_{s+1}$, e ad ogni s $A_s \setminus \bigcup_{t<s} A_t$ contenga un elemento m_s che sarà il massimo in $\trianglelefteq_s$.

L'ordine finale $\trianglelefteq = \bigcup_s \trianglelefteq_s$ dovrà inoltre soddisfare le seguenti due classi di condizioni (per ogni e ed s):

- (R_e) φ_e non è diversa dall'identità e rispetta $\trianglelefteq$.
- (N_s) m_s ha finiti predecessori in $\trianglelefteq$.

È importante capire quale sarà la strategia che verrà usata per soddisfare le condizioni R_e. Supponiamo che $\trianglelefteq$ sia isomorfo ad ω e che φ_e rispetti l'ordine ma non sia l'identità. Allora per ogni x si ha $x \trianglelefteq \varphi_e(x)$, altrimenti esisterebbe un y tale che $y \triangleright \varphi_e(y) \triangleright \varphi_e^{(2)}(y) \triangleright \ldots$ ma ciò è impossibile in un buon ordine come $\trianglelefteq$. Dato che φ_e non è l'identità, esisterà un x per il quale $x \triangleleft \varphi_e(x) \triangleleft \varphi_e^{(2)}(x) \triangleleft \ldots$. La strategia sarà allora quella di attendere lo stadio s in cui $\varphi_{e,s}(x)\downarrow$ e $\varphi_{e,s}^{(2)}(x)\downarrow$ ed a tale stadio aggiungere tra x e $\varphi_e(x)$ $k-1$ nuovi elementi dove $k = |[\varphi_{e,s}(x), \varphi_{e,s}^{(2)}(x)]|$. Agendo in tal modo si otterrà $|[x, \varphi_{e,s}(x)]| \geq k+1$. Impedendo poi che in stadi successivi i nuovi elementi vengano inseriti tra $\varphi_{e,s}(x)$ e $\varphi_{e,s}^{(2)}(x)$ si impedirà, per evidenti ragioni di cardinalità, che φ_e rispetti l'ordine $\trianglelefteq$.

Vediamo ora i dettagli della costruzione. Con $r(e,s)$ indicheremo la restrizione imposta dalla condizione R_e allo stadio s.

- *Stadio 0* Poniamo $A_0 = \{0\}$, $m_0 = 0$ e $\forall e\ r(e,0) = m_0$. Necessariamente abbiamo $\trianglelefteq_0 = \{(0,0)\}$.
- *Stadio s+1* Supponiamo di aver definito A_s, m_s e $r(e,s)$ per ogni e. Diremo che i richiede attenzione se soddisfa:
  $i \leq s\ \wedge\ r(i,s) = m_0\ \wedge\ \exists x \in A_s\ [\varphi_{i,s}(x)\downarrow \wedge\ \varphi_{i,s}^{(2)}(x)\downarrow \wedge\ (x \triangleleft \varphi_{i,s}(x)) \wedge$ $(\varphi_{i,s}(x) \triangleleft \varphi_{i,s}^{(2)}(x)) \wedge\ (\forall e \leq i\ m_e \trianglelefteq x) \wedge\ (\forall e < i\ r(e,s) \trianglelefteq x)]$.

  Se nessun i richiede attenzione, poniamo $m_{s+1} = \min\{m : m \notin A_s\}$, $A_{s+1} = A_s \cup \{m_{s+1}\}$ e $\forall e\ r(e,s+1) = r(e,s)$. Definiamo inoltre $\trianglelefteq_{s+1} = \trianglelefteq_s \cup \{(n, m_{s+1}) : n \in A_{s+1}\}$.

  Se qualche i richiede attenzione, fissiamo i minimo tra questi. Allora i riceve attenzione. Sia inoltre $x \in A_s$ minimo (in $\mathbb{N}$) tra quelli che soddisfano le condizioni per i. Chiamiamo $x' = \varphi_{i,s}(x)$, $x'' = \varphi_{i,s}(x')$ e $k = |[x', x'']|$ (intervallo in A_s). Scegliamo allora i primi k numeri non ancora in A_s, diciamo $n_0, \ldots, n_{k-1}$ e poniamo $m_{s+1} = n_{k-1}$, $A_{s+1} = A_s \cup \{n_0, \ldots, n_{k-1}\}$. Estendiamo quindi $\trianglelefteq_s$ inserendo la catena $n_0 \triangleleft_{s+1} \triangleleft n_1 \cdots \triangleleft_{s+1} n_{k-2}$ immediatamente alla destra di x e ponendo $y \triangleleft_{s+1} m_{s+1}$ per ogni $y \in A_{s+1}$. Definiamo infine $r(i,s+1) = x''$, $\forall e < i\ r(e,s+1) = r(e,s)$ e $\forall e > i\ r(e,s+1) = m_0$.

È facile verificare, utilizzando quello che si è visto nelle precedenti due sezioni, che ad ogni stadio le operazioni da compiere sono computabili. L'ordine lineare

$(A, \trianglelefteq)$ risulta quindi computabile. Verifichiamo ora con una serie di risultati che la costruzione ha le proprietà richieste.

(1) Ogni i riceve attenzione un numero finito di volte. Proviamolo per induzione su i. Supponiamo che valga per ogni $e < i$. Sia quindi s uno stadio tale che nessun $e < i$ riceva più attenzione per stadi $t \geq s$. Allora se i riceve attenzione in uno stadio $s' \geq s$, avremo $\forall t \geq s'$ $r(i,t) = r(i,s') \neq m_0$ in quanto ogni $e < i$ non riceve più attenzione. Quindi i riceve, dopo lo stadio s, attenzione al più ancora una volta. Pertanto i riceverà attenzione un numero finito di volte.

(2) $(A, \trianglelefteq)$ ha tipo d'ordine ω. Basta dimostrare che ogni $n \in A$ ha finiti predecessori. Se $n \in A_s \setminus \bigcup_{t<s} A_t$, allora $n \trianglelefteq m_s$. È quindi sufficiente verificare che la condizione N_s sia soddisfatta. Sia $z \triangleleft m_s$. Allora o $z \in A_s$ (insieme finito) oppure z entra quando qualche $i < s$ riceve attenzione. Per il punto precedente questo avviene un numero finito di volte ed ogni volta si aggiunge un numero finito di elementi.

(3) Ogni R_e è soddisfatta. Supponiamo per assurdo che $\varphi_e : A \to A$ non sia l'identità e rispetti $\trianglelefteq$. Per il punto (2) esisterà un y per cui $y \triangleleft \varphi_e(y)$. Iterando si ha che $\forall n$ $\varphi_e^{(n)}(y) \triangleleft \varphi_e^{(n+1)}(y)$. Sia s tale che nessun $i < e$ riceva attenzione negli stadi futuri. Fissiamo n tale che $x = \varphi_e^{(n)}(y)$ soddisfi $\forall i \leq e$ $m_i \trianglelefteq x$ e $\forall i < e$ $r(i,s) \trianglelefteq x$. Osserviamo che un n che soddisfa queste condizione esiste in quanto $x \triangleleft \varphi_e(x) \triangleleft \varphi_e^{(2)}(x) \triangleleft \ldots$. Sia ora $t \geq s$ stadio tale che $x \in A_t$, $\varphi_{e,t}(x)\downarrow$, $\varphi_{e,t}^{(2)}(x)\downarrow$ e $t \geq e$. A questo punto, se $r(e,t) \neq m_0$ allora e ha già ricevuto attenzione ad uno stadio precedente e non riceverà più attenzione. Perciò esisterà un x' per il quale $|[x', \varphi_e(x')]| > |[\varphi_e(x'), \varphi_e^{(2)}(x')]|$, ma ciò è assurdo per l'ipotesi su φ_e. Se invece $r(e,t) = m_0$, allora e riceve attenzione esattamente allo stadio t e per qualche $x' \in A_t$ vale la disuguaglianza di cardinalità. Vale in A_t ma anche in A, in quanto nessuna condizione che inserisce elementi sotto $\varphi_e^{(2)}(x')$ agisce più. Anche questa conclusione è assurda, quindi φ_e non può rispettare $\trianglelefteq$. $\square$

Capitolo 3

Un limite inferiore: il Teorema di Downey-Lempp

Abbiamo dimostrato che le procedure computabili non sono in grado di calcolare autoimmersioni di ogni ordine lineare. Sorge allora in maniera naturale il problema di determinare, se esiste, la minima complessità computazionale in grado di calcolare autoimmersioni non banali. Il Teorema di Downey-Lempp ([DL99]), oltre a migliorare quello di Hay-Rosenstein, garantisce un limite inferiore a tale complessità. Per presentare questo risultato vengono introdotte, nelle seguenti due sezioni, alcune nozioni leggermente più avanzate di quelle viste nel precedente capitolo (il riferimento è [So87, Chapter 3]).

3.1 Turing-riducibilità

Vogliamo introdurre una nuova relazione binaria tra i sottoinsiemi di $\mathbb{N}$. L'idea intuitiva da cui si vuole partire è la seguente: dati A,$B \subseteq \mathbb{N}$ diremo che B si Turing-riduce ad A ($B \leq_T A$) se si è in grado di computare χ_B avendo a disposizione χ_A. Per formalizzare questo concetto sarà necessario relativizzare i concetti basilari di funzioni ricorsive e calcolabili. Fissiamo $A \subseteq \mathbb{N}$,

1. definiamo le funzioni A-ricorsive come la più piccola classe di funzioni contenente la funzione nulla ($x \mapsto 0$), la funzione successore ($x \mapsto x+1$), le proiezioni di arietà qualsiasi e χ_A e che è chiusa per le operazioni di composizione, ricorsione primitiva e minimizzazione.

2. aggiungiamo alle istruzioni per URM l'operazione $O(n)$ per ogni $n \in \mathbb{N}$, la quale analizza il contenuto r_n del registro n-esimo e lo sostituisce con $\chi_A(r_n)$. Le istruzioni del tipo $O(n)$ vengono anche chiamate "oracolo".

I programmi per URM con l'aggiunta dell'oracolo $O(n)$ si possono codificare in maniera ricorsiva (in stretta analogia a quello che si fa per i programmi senza questa istruzione). Denoteremo quindi con $\hat{P}_e$ il programma con oracolo di indice e. Osserviamo che la codifica di questi programmi è indipendente dall'insieme collegato all'oracolo (ovvero dalla funzione caratteristica calcolata da O).

Definizione 3.1. Diremo che la funzione parziale $\psi : \mathbb{N}^k \to \mathbb{N}$ è A-calcolabile, e scriveremo $\psi \leq_T A$, se esiste un programma $\hat{P}_e$ tale che, collegando A con l'oracolo, calcola esattamente ψ. In tal caso scriveremo anche φ_e^A al posto di ψ.

Analogamente a quanto visto all'inizio del capitolo precedente, anche in questo caso vale l'equivalenza tra le due definizioni.

Teorema 3.2. *Una funzione ψ è A-ricorsiva se e solo se è A-calcolabile. In tal caso diremo che ψ è A-computabile.*

Ora possiamo formalizzare la nozione di Turing-riducibilità.

Definizione 3.3. Dati A,$B \subseteq \mathbb{N}$, diremo che B è Turing-riducibile ad A ($B \leq_T A$) se e solo se $\chi_B \leq_T A$. Diremo anche che B è A-computabile.

Nel caso in cui si abbia $A \leq_T B$ e $B \leq_T A$ scriveremo $A \equiv_T B$ e diremo che A e B sono Turing-equivalenti.

Lemma 3.4. *Le funzioni computabili sono A-computabili per ogni $A \subseteq \mathbb{N}$.*

Dimostrazione. Ogni funzione computabile può essere calcolata da un programma che non usa l'istruzione $O(n)$. Questo programma calcolerà la medesima funzione indipendentemente dall'insieme che si collega all'oracolo. $\square$

Osservazione 3.5. La Turing-riducibilità è più debole della m-riducibilità vista nel capitolo precedente. Infatti supponiamo $A, B \subseteq \mathbb{N}$ con $B \leq_m A$. Sia f computabile testimone dellla m-riduzione, allora $\chi_B = \chi_A \circ f$ è A-computabile. Quindi $B \leq_T A$. Inoltre è evidente che per ogni insieme A vale $A \leq_T \overline{A}$ (quindi $A \equiv_T \overline{A}$) mentre questo non vale per la m-riducibilità. Basti pensare al caso $A = K$.

Osservazione 3.6. Se A è un insieme computabile, le funzioni A-computabili risulteranno esattamente quelle computabili. All'interno di ogni programma possiamo infatti sostituire l'oracolo con una serie di istruzioni che calcoli la funzione caratteristica di A.

La maggior parte dei risultati sulle funzioni computabili ha un naturale corrispettivo riguardante le funzioni A-computabili. Vediamo due esempi.

Teorema 3.7 (Teorema di enumerazione relativizzato)**.** *Esiste z tale che per ogni $A \subseteq \mathbb{N}$ vale*

$$\forall x, y \in \mathbb{N}\ \varphi_z^A(x, y) = \varphi_x^A(y). \tag{3.1}$$

Teorema 3.8 (Teorema s-m-n relativizzato)**.** *Per $n, m \geq 1$ esiste una funzione computabile iniettiva s_n^m tale che per ogni $A \subseteq \mathbb{N}$ vale*

$$\forall x \in \mathbb{N}\ \forall \vec{y} \in \mathbb{N}^m\ \forall \vec{z} \in \mathbb{N}^n\ \varphi_{s_n^m(x,\vec{y})}^A(\vec{z}) = \varphi_x^A(\vec{y}, \vec{z}). \tag{3.2}$$

Il predicato di Kleene può essere relativizzato facilmente. Dato un insieme A, $T(A; e, x, y)$ indicherà che y è la codifica della computazione della macchina $\hat{P}_e$ su input x quando A è collegato all'oracolo.

Osservazione 3.9. Fissato $A \subseteq \mathbb{N}$, l'insieme $\{(e, x, y) : T(A; e, x, y)\}$ è A-computabile.

Possiamo introdurre una funzione $M(y) := \max\{r :$ un'istruzione $O(n)$ viene eseguita in y quando il contenuto dell'n-esimo registro è $r\} \cup \{0\}$. Tale funzione compie decodifiche di computazioni già effettuate, quindi è computabile. Essa viene utilizzata per definire la seguente funzione.

Definizione 3.10. Fissato $A \subseteq \mathbb{N}$, definiamo la funzione uso nel seguente modo $u(A; e, x) = M(\mu y\ T(A; e, x, y)) + 1$.

In questo nuovo contesto si modifica lievemente il concetto di s-esimo stadio di computazione. Definiremo:

$$\varphi_{e,s}^A(x) := \begin{cases} U(\mu y\ (y < s \wedge T(A, e, x, y))) & \text{se } u(A; e, x) < s, \\ \uparrow & \text{altrimenti.} \end{cases} \tag{3.3}$$

Pertanto quando $\varphi_{e,s}^A(x) \downarrow$ si richiede anche che la computazione consulti l'oracolo su valori minori di s.

Analogamente a quanto accade nel caso non relativizzato, vale il seguente:

Lemma 3.11. *Gli insiemi $\{(e, x, s) : \varphi_{e,s}^A(x) \downarrow\}$, $\{(e, x, s, z) : \varphi_{e,s}^A(x) = z\}$ sono A-computabili.*

Possiamo anche definire una funzione uso che tenga conto degli stadi di computazione:

$$u(A; e, x, s) := \begin{cases} u(A; e, x) & \text{se } \varphi_{e,s}^A(x) \downarrow, \\ 0 & \text{se } \varphi_{e,s}^A(x) \uparrow. \end{cases} \tag{3.4}$$

In tal caso u risulterà totale e A-computabile.

Indicheremo con $2^{<\mathbb{N}}$ l'insieme di tutte le successioni finite (stringhe) di 0 e 1. Se $\sigma \in 2^{<\mathbb{N}}$ con $lh(\sigma)$ intenderemo la sua lunghezza. Dato $A \subseteq \mathbb{N}$ diremo che $\sigma \subseteq A$ se σ è una sequenza iniziale di χ_A, cioè se $\forall\, x < lh(\sigma)\ \sigma(x) = \chi_A(x)$. Indicheremo infine con $A|x$ e $\sigma|x$ le restrizioni di A e σ agli $y < x$. Con queste notazioni possiamo definire :

$$\varphi_e^\sigma(x) := \begin{cases} \varphi_e^A(x) & \text{se esiste } A \text{ tale che } \sigma \subset A \text{ e } u(A;e,x) \leq lh(\sigma), \\ \uparrow & \text{altrimenti.} \end{cases} \tag{3.5}$$

Osservazione 3.12. La definizione di $\varphi_e^\sigma(x)$ non dipende dall'insieme A scelto come estensione di σ. Se B è tale che $\sigma \subseteq B$ e $u(B;e,x) \leq (\sigma)$ allora abbiamo che $\varphi_e^A(x) = \varphi_e^B(x)$. Infatti A e B coincideranno sui valori richiesti dall'oracolo durante la computazione.

Estendendo questa definizione nel modo ovvio si introduce anche $\varphi_{e,s}^\sigma(x)$. Vediamo adesso alcuni risultati di immediata verifica riguardanti queste nozioni.

Lemma 3.13. *Sia $A \subseteq \mathbb{N}$ e $\varphi_{e,s}^A(x) = y$. Allora valgono:*

1. $x, y, e < s$ *e* $u(A;e,x) \leq s$.
2. $\forall t \geq s\ \varphi_{e,t}^A(x) = y$.
3. *Se* $\sigma = A|u(A;e,x)$ *allora* $\varphi_{e,s}^\sigma(x) = y$.

Lemma 3.14. *Gli insiemi* $\{\langle e,\sigma,x,s\rangle : \varphi_{e,s}^\sigma(x)\downarrow\}$, $\{\langle e,\sigma,x,s,y\rangle : \varphi_{e,s}^\sigma(x) = y\}$ *sono computabili.*

Dimostrazione. Deriva dal fatto che uso σ, insieme finito e computabile, per rispondere all'oracolo e termino la computazione se giungo a convergenza oppure supero lo stadio s oppure l'oracolo viene consultato su valori maggiori di $\min\{\mathrm{lh}(\sigma), s\}$. □

Lemma 3.15. *Gli insiemi* $\{\langle e,\sigma,x\rangle : \varphi_e^\sigma(x)\downarrow\}$ *e* $\{\langle e,\sigma,x,y\rangle : \varphi_e^\sigma(x) = y\}$ *sono computabilmente enumerabili.*

Dimostrazione. Basta osservare che $\varphi_e^\sigma(x)\downarrow$ se e solo se $\exists s \varphi_{e,s}(x)\downarrow$. □

Teorema 3.16 (Principio dell'uso)**.** *Valgono le seguenti implicazioni:*

1. $\varphi_e^A(x) = y \iff \exists s\ \exists \sigma \subset A\ \varphi_{e,s}^\sigma(x) = y$.
2. $\varphi_{e,s}^\sigma = y \to \forall t \geq s\ \forall \tau \supset \sigma\ \varphi_{e,t}^\tau(x) = y$.
3. $\varphi_e^\sigma = y \to \forall A \supset \sigma\ \varphi_e^A(x) = y$.

3.2 Gradi di Turing e teorema del jump

La Turing-riducibilità è banalmente riflessiva. Inoltre non è difficile convincersi della sua transitività. Infatti se si ha $B \leq_T C$ allora ogni funzione B-computabile è anche C-computabile. Pertanto la Turing-equivalenza ($\equiv_T$) definisce una relazione di equivalenza tra i sottoinsiemi di $\mathbb{N}$.

Per indicare il livello di complessità di un insieme A è quindi naturale porre la seguente definizione:

Definizione 3.17. Chiameremo grado di Turing (o grado di indecibilità) di A la classe di equivalenza $\deg(A) = \{B : B \equiv_T A\}$.

I gradi di Turing sono quindi esattamente gli elementi dell'insieme $\mathbf{D} := P(\mathbb{N})/\equiv_T$. Indicheremo con lettere minuscole in grassetto gli elementi di tale insieme.

Osservazione 3.18. Per ogni $A \subseteq \mathbb{N}$ $\deg(A) = \{B : B \equiv_T A\}$ ha cardinalità numerabile. Questo dipende dalla numerabilità dei programmi e (quindi) delle funzioni A-computabili. Perciò $\mathbf{D}$ ha cardinalità $2^{\aleph_0}$.

Osservazione 3.19. È chiaro che in $\mathbf{D}$ possiamo definire un ordine parziale definendo

$$\mathbf{a} \leq \mathbf{b} \iff \forall A \in \mathbf{a}\ \forall B \in \mathbf{b}\ A \leq_T B. \tag{3.6}$$

Grazie alla transitività di $\leq_T$ questa condizione è equivalente a

$$\exists A \in \mathbf{a}\ \exists B \in \mathbf{b}\ A \leq_T B. \tag{3.7}$$

Introduciamo un'importante operazione tra i sottoinsiemi di $\mathbb{N}$.

Definizione 3.20. Dati $A, B \in \mathbb{N}$, definiamo $A \oplus B := \{2x : x \in A\} \cup \{2x+1 : x \in B\}$.

Osservazione 3.21. È facile vedere che $A \oplus B$ è il sup rispetto alla relazione $\leq_T$ dell'insieme $\{C : A \leq_T C \wedge B \leq_T C\}$. Infatti da $A \oplus B$ si computa facilmente sia A che B. Inoltre se C permette di computare χ_A e χ_B allora si ottiene $\chi_{A\oplus B}$ nel seguente modo:

$$\chi_{A\oplus B}(x) := \begin{cases} \chi_A(x/2) & \text{se } x \text{ è pari,} \\ \chi_B((x-1)/2) & \text{se } x \text{ è dispari.} \end{cases} \tag{3.8}$$

Il seguente lemma permette di estendere l'operazione appena introdotta ai gradi di Turing.

Lemma 3.22. *Siano $A, B, C, D \subseteq \mathbb{N}$ tali che $A \equiv_T C$ e $B \equiv_T D$. Allora $A \oplus B \equiv_T C \oplus D$.*

Dimostrazione. Dall'insieme $A \oplus B$ si possono computare A e B. Dato che $A \equiv_T C$, da A si ottiene C e, per la stessa ragione, da B si ottiene D. Per quanto si è visto da C e D si computa $C \oplus D$. Perciò $A \oplus B \leq_T C \oplus D$. Per simmetria vale anche $C \oplus D \leq_T A \oplus B$. □

Dati $\mathbf{a}, \mathbf{b} \in \mathbf{D}$, definiamo l'operazione $\mathbf{a} \cup \mathbf{b} := \deg(A \oplus B)$. Per il lemma precedente la definizione è ben posta.

Osservazione 3.23. Come già detto $\leq$ definisce un ordine parziale su $\mathbf{D}$ e l'operazione $\cup$ associa a due gradi di Turing il loro sup rispetto a tale ordine. Gli ordini parziali accompagnati da operazioni con questa proprietà vengono chiamati semireticoli superiori.

Relativizziamo ora il concetto di insieme c.e. Dati $A \subseteq \mathbb{N}$ e $\sigma \in 2^{<\mathbb{N}}$, useremo le seguenti notazioni: $W_e^A = \mathrm{dom}(\varphi_e^A)$, $W_{e,s}^A = \mathrm{dom}(\varphi_{e,s}^A)$, $W_e^\sigma = \mathrm{dom}(\varphi_e^\sigma)$, $W_{e,s}^\sigma = \mathrm{dom}(\varphi_{e,s}^\sigma)$.

In modo da avere la stessa codifica, d'ora in poi scriveremo φ_e al posto di $\varphi_e^\emptyset$.

Definizione 3.24. Dati $A, B \subseteq \mathbb{N}$ diremo che B è computabilmente enumerabile in A (B è A-c.e.) se esiste e tale che $B = W_e^A$.

Diremo inoltre che B è Σ_1^A se B è la proiezione di un insieme A-computabile.

Valgono anche in questo caso i due seguenti risultati.

Teorema 3.25 (Teorema del complemento)**.** *$B \leq_T A$ se e solo se B e $\overline{B}$ sono B-c.e. .*

Dimostrazione. ($\rightarrow$) Basta far divergere in modo opportuno le computazioni di χ_B.

($\leftarrow$) Come nel caso non relativizzato, si fanno partire in parallelo le due computazioni. □

Proposizione 3.26. *B è A-c.e. se e solo se B è Σ_1^A.*

Dimostrazione. ($\rightarrow$) Se $B = W_e^A$ allora per il principio dell'uso abbiamo la seguente equivalenza: $x \in B \iff \exists s\, \exists \sigma\, (\sigma \subset A \wedge x \in W_{e,s}^\sigma)$. La seconda corrisponde alla proiezione di una condizione A-computabile.

($\leftarrow$) Se B è del tipo $\{x : \exists y (x, y) \in C\}$ con C insieme A-computabile, allora B è dominio della seguente funzione: $\varphi(x) := \mu z\ (x, z) \in C$. Tale funzione è A-computabile quindi B è A-c.e. □

Diremo che il grado $\mathbf{a} \in \mathbf{D}$ è c.e. se esiste $A \in \mathbf{a}$ che è c.e. Dati due gradi $\mathbf{a}, \mathbf{b} \in \mathbf{D}$, diremo che $\mathbf{a}$ è c.e. in $\mathbf{b}$ se esistono $A \in \mathbf{a}$ e $B \in \mathbf{b}$ tali che A è B-c.e. Indicheremo con $\mathbf{R}$ il sottoinsieme di $\mathbf{D}$ composto dai gradi c.e. ovvero $R = \{\mathbf{a} : \mathbf{a} \text{ è c.e.}\}$.

Osservazione 3.27. $\mathbf{R}$ ha cardinalità $\leq \aleph_0$ in quanto gli insiemi c.e. sono in quantità numerabile. Sappiamo che $\deg(\emptyset)$ e $\deg(K)$ sono due gradi di Turing distinti e c.e., perciò $|\mathbf{R}| \geq 2$.

In realtà si può dimostrare che $\mathbf{R}$ ha cardinalità esattamente $\aleph_0$ e che ogni ordine lineare numerabile può essere immerso in $(\mathbf{R}, \leq)$.

Osservazione 3.28. Dati due insiemi c.e. A e B è facile definire una funzione computabile che converga esattamente su $A \oplus B$. Pertanto l'operazione $\cup$ può essere ristretta in maniera corretta all'insieme $\mathbf{R}$. In tal modo anche la terna $(\mathbf{R}, \leq, \cup)$ risulta un semireticolo superiore.

Definizione 3.29. Sia $A \subseteq \mathbb{N}$, chiameremo il jump di A l'insieme $K^A = \{x : \varphi_x^A(x)\downarrow\}$.

Per indicare il jump di A useremo anche la notazione A'. L'operazione di costruzione del jump può essere iterata. Definiremo induttivamente $A^0 := A$ e $A^{n+1} := (A^n)'$.

D'ora in avanti, con un abuso di notazione, identificheremo $K = \{x : \varphi_x(x)\}$ con $\emptyset'$ nonostante siano a tutti gli effetti insieme distinti. La differenza è determinata da una differente codifica dei programmi. È chiaro però che essi possiedono la stessa complessità computazionale e che ogni risultato ottenuto nel precedente capitolo per K vale anche per $\emptyset'$.

Osservazione 3.30. Il jump di un insieme A si può in alternativa definire come $K_0^A := \{\langle x, y\rangle : \varphi_x^A(y)\downarrow\}$. Infatti, in maniera del tutto simile a quanto visto per K e K_0, si può dimostrare che $K^A \equiv_1 K_0^A$ e (quindi) che $K^A \equiv_T K_0^A$.

Il seguente importante teorema determina una serie di proprietà riguardanti il jump.

Teorema 3.31 (Teorema del jump)**.** *Dati $A, B, C \subseteq \mathbb{N}$. Valgono le seguenti affermazioni:*

1. *A' è A-c.e.*
2. *$A' \not\leq_T A$.*
3. *B è A-c.e. se e solo se $B \leq_1 A'$.*
4. *Se A è B-c.e. e $B \leq_T C$ allora A è C-c.e.*
5. *$B \leq_T A$ se e solo se $B' \leq_1 A'$.*
6. *Se $B \equiv_T A$ allora $B' \equiv_1 A'$ e $B' \equiv_T A'$.*
7. *A è B-c.e. se e solo se A è $\overline{B}$-c.e.*

Dimostrazione. (1) Basta dimostrare che A è Σ_1^A, ma per definizione si ha $x \in A' \iff \exists s\ \varphi_{x,s}^A(x)\downarrow$.

(2) In questo caso si ripete esattamente la dimostrazione utilizzata per il problema della fermata.

(3) ($\rightarrow$) Sia B c.e. Allora esiste e tale che $B = W_e^A$. La funzione $f(x) = \langle e, x\rangle$ testimonia che $B \leq_1 K_0^A$, quindi $B \leq_1 A'$. ($\leftarrow$) Sia $B \leq_1 A'$ e sia f, computabile e totale, testimone di tale riduzione. Allora B è dominio di $\psi(x) := \varphi_{f(x)}^A(f(x))$.

(4) Deriva dal fatto che se abbiamo $B \leq_T C$, ogni funzione B-computabile è anche C-computabile (in particolare lo sarà χ_A).

(5) ($\rightarrow$) Sia $B \leq_T A$. Dato che, per (1), B' è B-c.e. per (4) si ha che B' è A-c.e. Per (3) $B' \leq_1 A'$. ($\leftarrow$) Supponiamo ora $B' \leq_1 A'$. Si ha banalmente che $B, \overline{B}$ sono B-c.e. Quindi, per (3), abbiamo che $B \leq_1 B' \leq_1 A'$ e $\overline{B} \leq_1 B' \leq_1 A'$. Sempre per (3) otteniamo che B e $\overline{B}$ sono entrambi A-c.e. Per il Teorema del complemento B è A-computabile.

(6) Ovvio corollario di (5).

(7) Deriva da (4) e dall'immediata osservazione che $B \equiv_T \overline{B}$. □

Il Teorema del jump permette di estendere l'operazione di jump sui gradi di Turing. Dato quindi $\mathbf{a} \in \mathbf{D}$ e $A \in \mathbf{a}$ porremo $\mathbf{a}' := \deg A'$. Infatti per (6) la definizione è ben posta. Per (2) abbiamo immediatamente che $\mathbf{a} < \mathbf{a}'$ e che $\mathbf{a}'$ è $\mathbf{a}$-c.e.

Indicheremo con $\mathbf{0}$ e $\mathbf{0^{(n)}}$ i gradi $\deg(\emptyset)$ e $\deg(\emptyset^{(n)})$.

3.3 Teorema di Downey-Lempp

In questa sezione usiamo le nozioni introdotte nelle precedenti due per ottenere alcuni nuovi risultati sugli ordini lineari. La prima cosa che possiamo ottenere è una relativizzazione del Teorema di Hay-Rosenstein visto nel secondo capitolo.

Teorema 3.32 (Teorema di Hay-Rosenstein relativizzato)**.** *Per ogni $A \subseteq \mathbb{N}$, esiste un ordine lineare infinito A-computabile tale che ogni sua autoimmersione non banale non è A-computabile.*

Dimostrazione. La costruzione è identica a quella vista per il teorema non relativizzato. Le nuove condizioni di tipo R_e sono "φ_e^A non è diversa dall'identità e rispetta l'ordine".

L'unica differenza è che ora si utilizza A come oracolo ad ogni stadio. La strategia rimane la medesima (modulo sostituire φ_e con φ_e^A in tutte le condizioni poste). □

Per il prossimo importante teorema abbiamo bisogno di introdurre alcune notazioni. Data una funzione totale $f(x,s)$ diremo che essa ammette limite se

$$\forall x\ \exists t\ \forall s > t\ f(x,s) = f(x,t). \tag{3.9}$$

In tal caso definiremo la funzione limite come $\lim_s f(x,s) := f(x)$.

Fissato $A \subseteq \mathbb{N}$, lavoreremo con la seguente funzione unaria

$$c^A(x) := \mu s(A'|(x+1) = A'_s|(x+1)) \tag{3.10}$$

dove si intende $x \in A'_s \iff \varphi^A_{x,s}(x)\downarrow$.

Utilizzeremo anche una sua approssimazione binaria definita nel seguente modo: $\overline{c}^A(x,t) := \mu s(A'_t|(x+1) = A'_s|(x+1))$.

Osservazione 3.33. La funzione c^A è A'-computabile. Osserviamo che c^A è debolmente crescente in quanto se s è tale che $A'|(x+1) = A'_s|(x+1)$ allora si ha $A'|(y+1) = A'_s|(y+1)$ per ogni $y < x$. Inoltre c^A è illimitata. Questo dipende direttamente dal fatto che A' è un insieme infinito (altrimenti sarebbe computabile).

La funzione $\overline{c}^A(x,t)$ è A-computabile, ammette limite e si ha $\lim_t \overline{c}^A(x,t) = c^A(x)$. Per ogni x e s anche $\overline{c}^A(\cdot,s)$ e $\overline{c}^A(x,\cdot)$ sono non decrescenti.

Dato un ordine lineare $(A,<)$ e $x,y \in A$, con $d_A(x,y)$ intenderemo la distanza tra x e y in A. Più precisamente $d_A(x,y) := |[x,y] \cup [y,x]|$ (intesi come intervalli in A).

Teorema 3.34. *Per ogni $A \subseteq \mathbb{N}$ esiste un ordine lineare infinito A-computabile, tale che ogni sua autoimmersione i ha la proprietà $c^A \leq_T i \oplus A$.*

Dimostrazione. Fissiamo $A \subseteq \mathbb{N}$. Costruiremo un ordine lineare A-computabile $(L,\trianglelefteq)$, isomorfo ad ω e tale che $L = \mathbb{N}$, a stadi. Ad ogni stadio s definiremo L_s e $\trianglelefteq_s$ utilizzando A come oracolo. Faremo in modo che per ogni s, $L_s \subseteq L_{s+1}$ e $\triangleleft_s \subseteq \triangleleft_{s+1}$. Porremo infine $L = \bigcup_s L_s$ e $\trianglelefteq = \bigcup_s \trianglelefteq_s$.

Allo stadio zero inizieremo con l'ordine $L_0 := 0 \triangleleft_0 2 \triangleleft_0 4 \triangleleft_0 \ldots$ composto da tutti i numeri pari. La strategia consisterà nel definire L in modo tale che esista una funzione $e : \mathbb{N} \to L_0$ soddisfacente le due seguenti condizioni per ogni $x \in \mathbb{N}$:

- (1_x) $\forall n_0, n_1 < c^A(x)(e(x) \triangleleft n_0 \triangleleft n_1 \to d_L(e(x), n_0) > d_L(n_0, n_1))$.
- (2_x) $e(x+1) = \mu y \in L_0(\forall n(n < c^A(x) \to n \triangleleft y))$.

Verifichiamo prima di tutto che quella scelta è una buona strategia.

Sia L un ordine lineare A-computabile, isomorfo ad ω e tale che esiste una funzione $e : \mathbb{N} \to L_0$ per la quale valgono le condizioni (1_x) e (2_x) per ogni

x. Verifichiamo che allora ogni sua autoimmersione non banale i è tale che $c^A \leq_T i \oplus A$.

Sia infatti i autoimmersione non banale di L. Per le condizioni del tipo (2_x) e dal fatto che c^A è debolmente crescente e illimitata, abbiamo che anche la funzione e è debolmente crescente e illimitata. L ha tipo d'ordine ω, esisterà quindi un $x_0 \in \mathbb{N}$ tale che $\forall x \geq x_0\ e(x) \lhd i(e(x))$. Dato che i mappa l'intervallo $[e(x_0), i(e(x_0))]$ in $[i(e(x_0)), i^2(e(x_0))]$ si ha che $d_L(e(x_0), i(e(x_0))) \leq d_L(i(e(x_0)), i^2(e(x_0)))$. Per la condizione (1_{x_0}) abbiamo necessariamente che almeno uno tra $i(e(x_0))$ e $i^2(e(x_0))$ è maggiore di $c^A(x_0)$. Se poniamo $t = max\{i(e(x_0)), i^2(e(x_0))\}$, abbiamo che $c^A(x_0) = \overline{c}^A(x_0, t)$, che è A-computabile.

Pertanto dall'ordine L (quindi da A), da $e(x_0)$ e da i abbiamo computato $c^A(x_0)$. Osserviamo che da $c^A(x_0)$ possiamo computare (con A come oracolo) $c^A(x)$ per ogni $x < x_0$. Infatti si ha $c^A(x) = \overline{c}^A(x, c^A(x_0))$ per ogni $x < x_0$. Ora, dalla condizione (2_{x_0}) possiamo computare $e(x_0 + 1)$ utilizzando L e $c^A(x_0)$. Pertanto, iterando il ragionamento, siamo in grado di computare c^A a partire da $e(x_0)$, i e A. Perciò $c^A \leq_T i \oplus A$.

Concludiamo la dimostrazione del teorema con la costruzione dell'ordine lineare. Avremo bisogno di una funzione binaria $\overline{e}(x, s) : \mathbb{N}^2 \rightarrow L_0$ che definiremo durante la costruzione e che risulterà essere un'approssimazione della funzione e.

L'obiettivo della costruzione è quello di fare in modo che ad ogni stadio s risultino soddisfatte le due seguenti condizioni per ogni $x \in \mathbb{N}$:

- $(1_{x,s})$ $\forall n_0, n_1 \in L_s\ (\ (n_0, n_1 < \overline{c}^A(x, s) \ \wedge\ \overline{e}(x, s) \lhd_s n_0 \lhd_s n_1) \rightarrow d_{L_s}(\overline{e}(x, s), n_0) > d_{L_s}(n_0, n_1)\ )$

- $(2_{x,s})$ $\overline{e}(x + 1, s) = \mu y \in L_0\ (\forall n(n < \overline{c}^A(x, s) \rightarrow n \lhd_s y))$

L'idea è quella di operare allo stadio s solo quando un x entra in A'. In tal caso verrà aggiunto il giusto numero di nuovi elementi a fianco di $\overline{e}(x, s)$ in modo tale da mantenere verificata la condizione $(1_{x,s})$. Ad ogni stadio s le condizioni del tipo $(2_{x,s})$ verranno usate come definizioni della funzione $\overline{e}(x, s)$ per ogni x.

Vediamo ora i dettagli della costruzione:

- *Stadio 0* Poniamo $(L_0, \trianglelefteq_0) := 0 \lhd_0 2 \lhd_0 4 \lhd_0 \dots$, $\overline{e}(0, 0) := 0$ e $\forall x\ \overline{e}(x + 1, 0) := \mu y \in L_0(\forall n(n < \overline{c}^A(x, 0) \rightarrow n \lhd_0 y))$.

- *Stadio s+1* Supponiamo di aver definito $(L_s, \lhd_s)$ in modo tale che valgano $(1_{x,s})$ e $(2_{x,s})$ per ogni $x \in \mathbb{N}$. Se siamo nel caso che $\forall x \leq s + 1\ x \notin W^A_{x,s+1} \setminus W^A_{x,s+1}$ allora poniamo $L_{s+1} := L_s$ e $\trianglelefteq_{s+1} := \trianglelefteq_s$ e $\forall x\ \overline{e}(x, s+1) := \overline{e}(x, s)$.

Se invece esiste (in tal caso unico) $x \leq s+1$ tale che $x \in W^A_{x,s+1} \setminus W^A_{x,s}$ allora definiamo $\overline{e}(0,s+1) := 0$ e $\forall x\ \overline{e}(x+1,s+1) := \mu y \in L_0(\forall n(n < \overline{c}^A(x,s+1) \to n \lhd_s y))$. Per definire L_{s+1} consideriamo l'insieme $D := \{z \in L_s : z < \overline{c}^A(x,s+1) \wedge z \rhd_s \overline{e}(x,s+1)\}$. Chiamiamo $l := \max\{d_{L_s}(z_0,z_1) : z_0,z_1 \in D\}$ e siano $n_0 < n_1 < \cdots < n_{l-1}$ i primi $l-1$ elementi di $\mathbb{N} \setminus L_s$ maggiori di $\overline{c}^A(x,s+1)$. Estendiamo allora $\leq_s$ ponendo la catena $n_0 \lhd_{s+1} n_1 \lhd_{s+1} \cdots \lhd_{s+1} n_{l-1}$ immediatamente sopra $\overline{e}(x,s+1)$. Inseriamo inoltre tutti gli elementi $y \leq s+1$ non ancora in L_s immediatamente sotto $\overline{e}(x,s+1)$.

Come è gia stato detto, poniamo $L := \bigcup_s L_s$ e $\unlhd := \bigcup_s \unlhd_s$.

Verifichiamo ora alcuni fatti riguardanti la costruzione per assicurarci che l'ordine abbia le proprietà volute.

(1) *L'ordine* $(L,\unlhd)$ *è A-computabile, ha tipo d'ordine* ω *e* $L = \mathbb{N}$. Ogni volta che un elemento x entra in A' ad uno stadio s, vengono inseriti in L tutti gli elementi $y \leq s$. Dato che A' è infinito, esistono infiniti stadi in cui vengono inseriti elementi. Perciò $L = \mathbb{N}$.

Per dimostrare che il suo tipo d'ordine è ω basta verificare che ogni elemento x ha finiti predecessori. Osserviamo prima di tutto che per ogni x e s le funzioni $\overline{e}(\cdot,s)$ e $\overline{e}(x,\cdot)$ sono debolmente crescenti (per costruzione perchè $\overline{c}^A(\cdot,s)$ e $\overline{c}^A(x,\cdot)$ lo sono). Osserviamo poi che in L vengono inseriti elementi sotto x ad un dato stadio t, se e solo se entra in A' un y tale che $\overline{e}(y,t) \unlhd_t x$. Sia $t > x$ uno stadio in cui entra un elemento m in A'. Indichiamo con k il primo elemento maggiore di m che entra in A' ad uno stadio $s > t$. Per definizione abbiamo che se $\overline{c}^A(k-1,s) > n$ allora $\overline{e}(k,s) \rhd_s n$. Dato che $\overline{c}^A(k-1,s) \geq \overline{c}^A(m,s) = t > x$ si ha $\overline{e}(k,s) \rhd_s x$. Inoltre per ogni $z > k$ e $l > s$ si ha $\overline{e}(z,l) \unrhd_l \overline{e}(k,s) \rhd_l x$. Quindi agli stadi maggiori di s vengono inseriti elementi alla sinistra di x solo se entrano in A' elementi minori di k. Dato che ad ogni stadio viene inserito un numero finito di elementi, x ha finiti predecessori.

Dato che ad ogni stadio vengono eseguite operazioni A-computabili e che $L = \mathbb{N}$, l'intero ordine risulterà A-computabile. Infatti, dati due elementi x,y, per confrontarli basterà aspettare uno stadio s tale che $x,y \in L_s$.

(2) *Ad ogni stadio s, l'ordine* $(L_s,\unlhd_s)$ *soddisfa le condizioni* $(1_{x,s})$ *e* $(2_{x,s})$ *per ogni* $x \in \mathbb{N}$. Le condizioni $(2_{x,s})$ sono tutte banalmente soddisfatte in quanto vengono assunte come definizione di $\overline{e}(x,s)$.

Proviamo per induzione su s che ad ogni stadio sono soddisfatte le condizioni del tipo $(1_{x,s})$. Allo stadio iniziale abbiamo che $\forall x \in \mathbb{N}\ \overline{c}^A(x,0) = 0$, perciò le condizioni $(1_{x,0})$ sono tutte soddisfatte perchè $\forall n(n \geq \overline{c}^A(x,0))$. Supponiamo di essere allo stadio $s+1$ e supponiamo che per ogni $x \in \mathbb{N}$ le condizioni $(1_{x,s})$ siano soddisfatte. Se allo stadio $s+1$ nessun elemento entra in A' allora tutte le funzioni rimangono costanti e nessun elemento viene inserito in L_{s+1}.

Le condizioni rimangono quindi soddistatte. Supponiamo invece che x_0 entri allo stadio $s+1$ in A'. Cerchiamo prima di tutto di comprendere il comportamento della funzione $\overline{c}^A(\cdot, s+1)$. Osservando la definizione non è difficile convincersi che

$$\overline{c}^A(y, s+1) = \begin{cases} \overline{c}^A(y, s) & \text{se } y < x_0 \\ s+1 & \text{se } y \geq x_0 \end{cases} \tag{3.11}$$

Per ogni $y < x_0$ abbiamo anche $\overline{e}(y, s+1) = \overline{e}(y, s)$ (in quanto valgono le condizioni $(2_{y,s+1})$). In tal caso le condioni $(1_{y,s+1})$ sono equivalenti alle $(1_{y,s})$ che, per ipotesi, sono soddisfatte. Per ogni $z > x_0$ si ha che $\overline{c}^A(z, s+1) = \overline{c}^A(z-1, s+1)$ e valgono le condizioni $(2_{z,s+1})$. Allora anche le condizioni $(1_{z,s+1})$ sono soddisfatte. Fissiamo infatti $n < \overline{c}^A(z, s+1) = \overline{c}^A(z-1, s+1)$. Allora, per $(2_{z-1,s+1})$, si ha $n \lhd_{s+1} \overline{e}(z, s+1)$, perciò la condizione $(1_{z,s+1})$ è soddisfatta.

L'unica condizione non automaticamente soddisfatta è quella riguardante $\overline{e}(x_0, s+1)$. La costruzione prevede però l'inserimento del giusto numero di elementi sopra $\overline{e}(x_0, s)$ per fare in modo che anche la condizione $(1_{x_0,s+1})$ venga verificata. Siano infatti $n_0, n_1 < \overline{c}^A(x_0, s+1)$ due elementi di L_{s+1} tali che $\overline{e}(x_0, s+1) \lhd_{s+1} n_0 \lhd_{s+1} n_1$. Allora $d_{L_{s+1}}(n_0, n_1) \leq \max\{d_{L_s}(z_0, z_1) : z_0, z_1 \in D\} = l$ in quanto gli elementi inseriti sopra $\overline{e}(x_0, s+1)$ in questo stadio sono maggiori di $\overline{c}^A(x_0, s+1)$ (vedi *Stadio s+1* per la definizione di D). Inoltre questi elementi fanno in modo che $d_{L_{s+1}}(\overline{e}(x_0, s+1), n_0) \geq l+1 > d_{L_{s+1}}(n_0, n_1)$, rendendo vera la condizione $(1_{x_0,s+1})$.

Per quanto riguarda gli elementi che vengono inseriti sotto $\overline{e}(x_0, s+1)$, proviamo che questi non rovinano nessuna condizione del tipo $(1_{y,s+1})$ per gli $y < x_0$. Sia t l'ultimo stadio minore di $s+1$ in cui è entrato in A' un elemento $< x_0$ (se un tale stadio non esiste risulta tutto banale perchè in tal caso $\overline{c}^A(y, s+1) = 0$ per ogni $y < x_0$). Allora in tale stadio sono stati inseriti in L_t tutti gli elementi $n \in \mathbb{N} \setminus L_{t-1}$ tali che $n \leq t$. Inoltre per ogni $y < x_0$ si ha $\overline{c}^A(y, s+1) \leq t$. Allora allo stadio $s+1$ verranno inseriti sotto $\overline{e}(x_0, s+1)$ solo elementi n tali che $n > t \geq \overline{c}^A(y, s+1)$ per ogni $y < x_0$. Quindi questi non rovineranno nessuna condizione $(1_{y,s+1})$ con $y < x_0$.

(3) *La funzione $\overline{e}(x, s)$ ammette limite. L'ordine $(L, \unlhd)$ e la funzione $e(\cdot) := lim_s e(\cdot, s)$ soddisfano le condizioni (1_x) e (2_x) per ogni $x \in \mathbb{N}$.* Abbiamo già osservato che per ogni x la funzione $\overline{c}^A(x, s)$ ammette limite. Per le condizioni $(2_{x,s})$ si osserva facilmente che anche $\overline{e}(x, s)$ ammette limite. Fissato $x \in \mathbb{N}$ esisterà un t tale che per ogni $s > t$ si ha $\overline{c}^A(x, s) = \overline{c}^A(x, t) = c^A(x)$ e $\overline{e}(x, s) = \overline{e}(x, s) =: e(x)$ e $L|e(x) = L_s|e(x) = L_t|e(x)$. Allora per ogni $s > t$ le condizioni $(1_{x,s})$ e $(2_{x,s})$ in L_s equivalgono alle condizioni (1_x) e (2_x) in L. Per il punto precedente queste sono soddisfatte. $\square$

Corollario 3.35. *Per ogni $A \subseteq \mathbb{N}$, esiste un ordine lineare infinito A-computabile tale che ogni sua autoimmersione non banale i ha la proprietà $A' \leq_T i \oplus A$.*

Dimostrazione. Abbiamo $A' \leq_T c^A \oplus A$, infatti $x \in A' \iff \varphi^A_{x,c^A(x)}(x) \downarrow$. Sia L l'ordine lineare costruito nella dimostrazione precedente e sia i una sua autoimmersione non banale. Allora $A' \leq_T c^A \oplus A \leq (i \oplus A) \oplus A \leq_T i \oplus A$. □

Il Teorema di Downey-Lempp analizza l'importante caso in cui $A = \emptyset$.

Corollario 3.36 (Teorema di Downey-Lempp)**.** *Esiste un ordine lineare infinito computabile, tale che ogni sua autoimmersione non banale computa $\emptyset'$.*

Dimostrazione. Deriva immediatamente dalla banale osservazione che per ogni insieme $C \in \mathbb{N}$ si ha $C \oplus \emptyset \leq_T C$. □

Osservazione 3.37. I risultati appena visti migliorano il precedente Teorema di Hay-Rosenstein (relativizzato nei casi in cui $A \neq \emptyset$). Infatti, fissato A, le autoimmersioni non banali dell'ordine L non sono A-computabili. Se lo fossero avremmo $A' \leq_T i \oplus A \leq_T A \oplus A \leq_T A$, assurdo per il Teorema del jump.

Osservazione 3.38. Un grado di Turing capace di calcolare autoimmersioni non banali di ogni ordine lineare computabile infinito deve per forza essere in grado di computare $\mathbf{0}'$. Il Teorema di Downey-Lempp garantisce quindi un primo limite inferiore.

Capitolo 4

Teorema di Downey-Jockusch-Miller

La seconda sezione di questo capitolo risolve il problema di trovare un grado computazionale capace di calcolare autoimmersioni non banali di ogni ordine lineare computabile, garantendo un limite superiore alla complessità necessaria a tale scopo. La terza sezione presenta l'importante Teorema di Downey-Jockusch-Miller ([DJM06]), che migliora il Teorema di Downey-Lempp. Entrambi i risultati hanno bisogno di alcune nuove nozioni teoriche che vengono introdotte nelle prime due sezioni (si veda [So87, Chapter 4]).

4.1 La gerarchia aritmetica

Definizione 4.1.

- Un insieme è Σ_0 (o Π_0) se è computabile.
- Un insieme è Σ_{n+1} se è la proiezione di un insieme Π_n.
- Un insieme è Π_{n+1} se il suo complemento è Σ_{n+1}.
- Un insieme è Δ_n se è Σ_n e Π_n.

Diremo che un insieme è aritmetico se è Σ_n per qualche n. Con Σ_n indicheremo spesso anche la classe degli insiemi che sono Σ_n (sarà quindi giustificata l'espressione $A \in \Sigma_n$). Analogamente faremo per Π_n e Δ_n.

Osservazione 4.2. Se $A \subseteq \mathbb{N}$ possiamo relativizzare in maniera naturale le definizioni appena viste. Diremo che un insieme è Σ_0^A (o Π_0^A) se è A-computabile, e definiremo Σ_n^A, Π_n^A, Δ_n^A con $n > 0$ allo stesso modo.

Osserviamo che se A è computabile le definizioni di Σ_n^A, Π_n^A e Δ_n^A coincidono con quelle non relativizzate. In tali casi eviteremo quindi di usare gli apici.

Per motivi di semplicità notazionale, i risultati esposti in seguito faranno riferimento al caso in cui A sia un insieme computabile. Risulterà evidente come questi possano essere relativizzati.

Osservazione 4.3. Per induzione si può dimostrare che un insieme $B \subseteq \mathbb{N}$ è Σ_n se e solo se esiste un insieme computabile $R \subseteq \mathbb{N}^{n+1}$ tale che $B = \{x : \exists x_1 \forall x_2 \exists x_3 \dots Q x_n \ (x, x_1, x_2, x_3, \dots, x_n) \in R\}$ dove $Q = \forall$ se n è pari o $Q = \exists$ se n è dispari.

Analogamente B è Π_n se e solo se esiste $R \subseteq \mathbb{N}^{n+1}$ computabile tale che $B = \{x : \forall x_1 \exists x_2 \forall x_3 \dots Q x_n \ (x, x_1, x_2, x_3, \dots, x_n) \in R\}$ con $Q = \exists$ se n pari o $Q = \forall$ se n dispari.

Dimostriamo ora una serie di proprietà riguardanti gli insiemi aritmetici.

Lemma 4.4. *1. A è Σ_n se e solo se $\overline{A}$ è Π_n.*

2. Se $A \in \Sigma_n \cup \Pi_n$ allora $A \in \Sigma_m \cap \Pi_m (= \Delta_m)$ per ogni $m > n$.

3. Se $A, B \in \Sigma_n$ (Π_n) allora $A \cup B, A \cap B \in \Sigma_n$ (Π_n).

4. La classe Σ_n con $n > 0$ è chiusa per proiezioni.

5. Se $A \in \Sigma_n$ (Π_n) e $B \leq_m A$ allora $B \in \Sigma_n$ (Π_n).

6. Le classi Σ_n e Π_n sono chiuse per quantificazioni limitate.

Dimostrazione. (1) Evidente dalla definizione.

(2) Deriva immediatamente dall'osservazione che si possono aggiungere quantificazioni su variabili non presenti.

(3) È sufficiente porre in forma prenessa la disgiunzione (o la congiunzione) delle formule che determinano l'appartenenza ad A e B e unire (a coppie) i quantificatori dello stesso tipo mediante le funzioni π_i.

(4) L'avevamo già notato nel caso degli insiemi Σ_1 (Corollario 2.25). La dimostrazione nel caso generale è la medesima.

(5) Supponiamo $A = \{x : \exists x_1 \forall x_2 \dots Q x_n \ (x, x_1, \dots, x_n) \in R\}$ con R computabile. Sia f testimone della riduzione $B \leq_m A$. Allora $B = \{x : \exists x_1 \forall x_2 \dots Q x_n \ (f(x), x_1, \dots, x_n) \in R\}$ è Σ_n.

(6) Per il punto (4) abbiamo che la classe Σ_n è chiusa per quantificazioni esistenziali e quindi, a maggior ragione, per quelle limitate. Per complementazione si ottiene che la classe Π_n è chiusa per quantificazioni universali limitate.

È sufficiente quindi dimostrare la chiusura dei Σ_n per quantificazioni universali limitate. Usiamo un ragionamento induttivo su n. Per $n = 0$ la classe Σ_0 coincide con quella degli insiemi computabili. Dato allora $R \subseteq \mathbb{N}^3$ computabile verifichiamo che gli insiemi $A = \{(x, y) : \exists z < y \ (x, y, z) \in R\}$ e

$B = \{(x,y) : \forall z < y\ (x,y,z) \in R\}$ sono ancora computabili. Ma questo è immediato perchè, dato (x,y), dobbiamo controllare, sia per A che per B, l'appartenenza a R di un numero finito di elementi. Dato che R è computabile, l'operazione risulta computabile. Supponiamo ora $n > 0$ e sia $A \in \Sigma_n$ con $A = \{(x,y,z) : \exists x_1 \forall x_2 \dots Q x_n\ (x,y,z,x_1,\dots,x_n) \in R\}$ con R computabile. Vogliamo dimostrare che anche $B = \{(x,y) : \forall z < y\ (x,y,z) \in A\}$ è Σ_n. Per definizione esiste $S \in \Pi_{n-1}$ tale che $(x,y,z) \in A \iff \exists u (x,y,z,u) \in S$. Pertanto si ha $(x,y) \in B \iff \forall z < y\ \exists u\ (x,y,z,u) \in S$. Se $(x,y) \in B$ esiste una stringa di valori σ, di lunghezza esattamente y, tale che $\forall z < y\ (x,y,z,\sigma(z)) \in S$, dove con $\sigma(z)$ si intende il z-esimo elemento della successione. Fissata una codifica per le stringhe possiamo quindi scrivere che $(x,y) \in B \iff \exists \sigma\ \forall z < y\ (x,y,z,\sigma(z)) \in S$. Dato che S è Π_{n-1}, anche $\{(x,y,t) : \forall z < y\ (x,y,z,t) \in S\}$ lo è. B è quindi proiezione di un insieme Π_{n-1}, ovvero $B \in \Sigma_n$. $\square$

Tra i sottoinsiemi di $\mathbb{N}$ abbiamo quindi due catene di classi di insiemi aritmetici. Quella formata dagli insiemi Σ_n: $\Sigma_0(= \Pi_0 = \Delta_0) = \Delta_1 \subseteq \Sigma_1 \subseteq \Delta_2 \subseteq \Sigma_2 \subseteq \Delta_3 \subseteq \dots$; e quella formata dagli insiemi Π_n: $\Pi_0(= \Sigma_0 = \Delta_0) = \Delta_1 \subseteq \Pi_1 \subseteq \Delta_2 \subseteq \Pi_2 \subseteq \Delta_3 \subseteq \dots$.

L'insieme di queste due catene viene chiamata gerarchia aritmetica.

Esempio 4.5. L'insieme $Fin = \{x : W_x \text{ è finito}\}$ è Σ_2. Infatti si ha $x \in Fin \iff W_x$ è finito $\iff \exists s \forall t (t > s \to W_{x,t} = W_{x,s})$.

$Cof = \{x : \overline{W_x} \text{ è finito}\}$ è Σ_3. Infatti $x \in Cof \iff \overline{W_x}$ è finito $\iff \exists y \forall z\ (y < z \to z \in W_x) \iff \exists y \forall z \exists s (y < z \to z \in W_{x,s})$.

L'insieme $\{\langle x,y \rangle : W_x \subseteq W_y\}$ è Π_2. Infatti $W_x \subseteq W_y \iff \forall z\ (z \in W_x \to z \in W_y) \iff \forall z\ (z \notin W_x \vee z \in W_y)$. La condizione $z \in W_y$ è Σ_1, quindi $z \notin W_x$ è Π_1. La loro disgiunzione è sicuramente Π_2, quindi l'intera condizione rimane Π_2.

Gli esempi appena visti sottolineano come la classificazione di un insieme all'interno della gerarchia aritmetica possa dipendere da come questo viene descritto. Pertanto siamo interessati a determinare la più piccola classe (Σ_n o Π_n) a cui il nostro insieme appartiene. Per questa ragione diamo la seguente definizione:

Definizione 4.6.

- A è Σ_n-completo se $A \in \Sigma_n$ e $\forall B \in \Sigma_n\ B \leq_1 A$.
- A è Π_n-completo se $A \in \Pi_n$ e $\forall B \in \Pi_n\ B \leq_1 A$.

Osservazione 4.7. Osserviamo che la nuova definizione, nel caso $n = 1$, equivale alla 1-completezza. Quindi, per quanto abbiamo visto nel primo capitolo, possiamo già concludere che K è Σ_1-completo e $\overline{K}$ è Π_1-completo.

Il seguente teorema determina delle importanti connessioni tra le nozioni viste nei precedenti capitoli e la gerarchia aritmetica.

Teorema 4.8 (Teorema di Post). *1.* $B \in \Sigma_{n+1} \iff \exists A \in \Pi_n$ *tale che* B *è* A*-c.e.*

2. $B \in \Sigma_{n+1} \iff \exists C \in \Sigma_n$ *tale che* B *è* C*-c.e.*

3. $\emptyset^{(n)}$ *è* Σ_n*-completo (se* $n > 0$*).*

4. $B \in \Sigma_{n+1} \iff B$ *è* $\emptyset^{(n)}$*-c.e.*

5. $B \in \Delta_{n+1} \iff B \leq_T \emptyset^{(n)}$.

Dimostrazione. (1) ($\rightarrow$) Sia $B \in \Sigma_{n+1}$. Per definizione esiste $R \in \Pi_n$ tale che $x \in B \iff \exists y\ (x, y) \in R$. Allora B è Σ_1^R, quindi B è R-c.e. per la proposizione 3.26. ($\leftarrow$) Sia B insieme A-c.e. con $A \in \Pi_n$. Quindi esiste e tale che $x \in B \iff x \in W_e^A \iff \exists s \exists \sigma\ (x \in W_{e,s}^{\sigma} \wedge \sigma \subset A)$. La condizione $x \in W_{e,s}^{\sigma}$ è computabile. Dobbiamo analizzare quindi il predicato $\sigma \subset A$. Abbiamo che $\sigma \subset A \iff \forall y < \mathrm{lh}(\sigma)\ (\sigma(y) = \chi_A(y)) \iff \forall y < \mathrm{lh}(\sigma)\ ((\sigma(y) = 0 \wedge y \notin A) \wedge (\sigma(y) = 1 \vee y \in A))$. La condizione tra parentesi è del tipo $\Sigma_n \vee \Pi_n$ e la quantificazione è limitata, pertanto l'intera condizione è di tipo Σ_{n+1}.

(2) Deriva immediatamente da (1).

(3) Proviamo per induzione su n che $\emptyset^{(n)}$ è Σ_n. Si ha che $\emptyset^{(0)} = \emptyset$ è computabile. Se $\emptyset^{(n)}$ è Σ_n allora $\emptyset^{(n+1)} = (\emptyset^{(n)})'$ è $\emptyset^{(n)}$-c.e. quindi, per (1), è Σ_{n+1}. Sempre induttivamente dimostriamo la completezza. Il caso $n = 1$ deriva direttamente dall'Osservazione 4.7. Sia ora $B \in \Sigma_{n+1}$ allora B è A-c.e. con $A \in \Sigma_n$. Per ipotesi induttiva $A \leq_1 \emptyset^{(n)}$, quindi per il Teorema del jump abbiamo che B è $\emptyset^{(n)}$ c.e. Ancora per il Teorema del jump otteniamo $B \leq_1 \emptyset^{(n+1)}$.

(4) $B \in \Sigma_{n+1} \iff B \leq_1 \emptyset^{(n+1)} (per (3)) \iff B$ è $\emptyset^{(n)}$-c.e. sempre per il Teorema del jump.

(5) $B \in \Delta_{n+1} \iff B \in \Sigma_{n+1} \wedge \overline{B} \in \Sigma_{n+1} \iff B$ e $\overline{B}$ sono $\emptyset^{(n)}$-c.e. (per (3)) $\iff B \leq_T \emptyset^{(n)}$. □

Corollario 4.9 (Teorema della gerarchia). $\forall n > 0$ *si ha* $\Delta_n \subsetneq \Sigma_n$ *e* $\Delta_n \subsetneq \Pi_n$.

Dimostrazione. Se fosse $\emptyset^{(n)} \in \Delta_n$, per il Teorema di Post avremmo che $\emptyset^{(n)} \leq_T \emptyset^{(n-1)}$, ma ciò è assurdo per il Teorema del jump. □

Osservazione 4.10. Osserviamo che per ogni $n > 0$ vale anche $\Sigma_n \subsetneq \Delta_{n+1}$. Se infatti fosse $\Sigma_n = \Delta_{n+1}$, dato che $\Pi_n \subseteq \Delta_{n+1}$, otterremmo $\Pi_n \subseteq \Sigma_n$. Da questo si avrebbe $\Pi_n = \Delta_n$, che sappiamo essere falso per il Teorema della gerarchia.

Come conseguenza si ha che tutte le inclusioni tra le varie classi all'interno della gerarchia aritmetica sono strette.

Corollario 4.11. *Un insieme Σ_n-completo non è Π_n.*

Dimostrazione. Se A è un insieme Σ_n-completo e fosse $A \in \Pi_n$ allora per ogni $B \in \Sigma_n$ si avrebbe $B \leq_1 A$, perciò $B \in \Pi_n$. Di conseguenza si otterrebbe $\Sigma_n \subseteq \Pi_n$ ovvero $\Delta_n = \Sigma_n$, assurdo per il corollario precedente. □

Osservazione 4.12. Per un insieme A che è $\Sigma_n(\Pi_n)$-completo la minima classe a cui appartiene è proprio $\Sigma_n(\Pi_n)$. Infatti se fosse $A \in \Sigma_{n-1}$ allora avremmo che $A \in \Pi_n$, assurdo per il corollario precedente.

Osservazione 4.13. È facile verificare che se si ha $A \in \Sigma_n$, B è Σ_n-completo e $B \leq_1 A$ allora anche A è Σ_n completo.

Esempio 4.14. Gli insiemi Fin (definito nell'Esempio 4.5) e $Tot := \{x : W_x = \mathbb{N}\}$ sono rispettivamente Σ_2 e Π_2 completi. Sia infatti $A \in \Pi_2$ e sia $R \in \mathbb{N}^3$ computabile tale che $x \in A \iff \forall y\ \exists z\ (x, y, z) \in R$. Definiamo allora la seguente funzione computabile:

$$\varphi_{f(x)}(u) := \begin{cases} 0 & \text{se } \forall y \leq u\ \exists z\ (x, y, z) \in R, \\ \uparrow & \text{altrimenti.} \end{cases} \tag{4.1}$$

Osserviamo che, se $x \in A$, per la caratterizzazione di A, si ha $\mathrm{dom}(\varphi_{f(x)}) = \mathbb{N}$, perciò $f(x) \in Tot$. Se invece $x \in \overline{A}$ esiste y_0 tale che per ogni z si ha $(x, y, z) \notin R$. Quindi $\forall u \geq y_0\ \varphi_{f(x)}(u) \uparrow$, cioè $x \in Fin$.

Dato che Fin e Tot sono insiemi disgiunti, f testimonia le seguenti riduzioni: $A \leq_1 Tot$ e $\overline{A} \leq_1 Fin$. Perciò, per la generalità di A e per il fatto che $A \in \Pi_2 \iff \overline{A} \in \Sigma_2$, abbiamo dimostrato che Tot è Π_2-completo e Fin è Σ_2-completo.

Osserviamo inoltre che, se chiamiamo $Inf := \{x : W_x \text{ è infinito}\}$, la medesima funzione garantisce anche che $A \leq_1 Inf$, ovvero che Inf è Π_2-completo.

4.2 Gradi PA

Introduciamo una nuova relazione tra i gradi di Turing.

Definizione 4.15. Dati $\mathbf{a}$,$\mathbf{b}$ due gradi di Turing, diremo che $\mathbf{b}$ è PA su $\mathbf{a}$ se ogni funzione $\mathbf{a}$-computabile parziale a valori in $\{0, 1\}$ ha un'estensione totale $\mathbf{b}$-computabile. In tal caso scriveremo $\mathbf{b} \gg \mathbf{a}$.

Dati A, B, C insiemi, con A e B disgiunti, diremo che C separa A e B se $A \subseteq C$ e $B \cap C = \emptyset$.

Fissato $\mathbf{a}$ un grado di Turing, diremo che due insiemi disgiunti $A, B \subseteq \mathbb{N}$ sono $\mathbf{a}$-computabilmente inseparabili se qualsiasi insieme che li separa non è $\mathbf{a}$-computabile.

Il seguente lemma sarà utile per inquadrare meglio la nozione di grado PA all'interno di $(\mathbf{D}, \leq)$.

Lemma 4.16. *Dato* $\mathbf{a}$ *grado di Turing, esistono insiemi* $\mathbf{a}$*-c.e. che sono computabilmente inseparabili.*

Dimostrazione. Fissato $A \in \mathbf{a}$, consideriamo $B_0 = \{x : \varphi_x^A(x) = 0\}$ e $B_1 = \{x : \varphi_x^A(x) = 1\}$. È immediato verificare che B_0 e B_1 sono $\mathbf{a}$-c.e. Supponiamo per assurdo che esista un insieme $\mathbf{a}$-computabile C tale che $B_0 \subseteq C$ e $B_1 \cap C = \emptyset$. Allora esiste e per cui $\chi_C = \varphi_e^A$. Se $e \in C$ allora $\varphi_e^A(e) = 1$, quindi $e \in B_1$, ma ciò non è possibile. Pertanto si ha necessariamente $e \notin C$, cioè $\varphi_e^A(e) = 0$, ma questo implica $e \in B_0 \subseteq C$, assurdo. $\square$

Proposizione 4.17. *Siano* $\mathbf{a}, \mathbf{b}$ *due gradi di Turing. Allora valgono:*

1. *se* $\mathbf{b} \gg \mathbf{a}$ *allora* $\mathbf{b} > \mathbf{a}$,

2. *se* $\mathbf{b} \geq \mathbf{a}'$ *allora* $\mathbf{b} \gg \mathbf{a}$.

Dimostrazione. (1) Fissato $A \in \mathbf{a}$, χ_A è una particolare funzione parziale, $\mathbf{a}$-computabile a valori in $\{0, 1\}$. L'unica sua estensione è χ_A stessa che è per questo $\mathbf{b}$-computabile. Quindi $\mathbf{b} \geq \mathbf{a}$.

Supponiamo ora per assurdo che $\mathbf{b} = \mathbf{a}$. Sia $A \in \mathbf{a}$, per la dimostrazione del lemma precedente gli insiemi $B_0 = \{x : \varphi_x^A(x) = 0\}$ e $B_1 = \{x : \varphi_x^A(x) = 1\}$ sono $\mathbf{a}$-computabilmente inseparabili. Consideriamo allora la seguente funzione $\mathbf{a}$-computabile,

$$\psi(x) := \begin{cases} \varphi_x^A(x) & \text{se } \varphi_x^A(x) \downarrow \text{ e } \varphi_x^A(x) \in \{0,1\}, \\ \uparrow & \text{altrimenti.} \end{cases} \tag{4.2}$$

Per ipotesi esiste f estensione $\mathbf{b}$-computabile e totale di ψ. Senza perdita di generalità possiamo assumere che tale estensione assuma solo valori in $\{0, 1\}$, ovvero che f sia la funzione caratteristica di un insieme C. È evidente che C separa gli insiemi B_0 e B_1. Osserviamo inoltre che C è $\mathbf{b}$-computabile ovvero, dato che abbiamo assunto $\mathbf{a} = \mathbf{b}$, C è $\mathbf{a}$-computabile. Ma questo è assurdo perché B_0 e B_1 sono $\mathbf{a}$-computabilmente inseparabili.

(2) Supponiamo $\mathbf{b} \geq \mathbf{a}'$. Siano $A \in \mathbf{a}$, $B \in \mathbf{b}$ e fissiamo $\psi = \varphi_e^A$ una qualsiasi funzione $\mathbf{a}$-computabile. Definiamo allora la seguente estensione totale di ψ:

$$f(x) := \begin{cases} \psi(x) & \text{se } \exists s \ \varphi_{e,s}^A(x)\downarrow, \\ 0 & \text{altrimenti.} \end{cases} \tag{4.3}$$

Dato che $\exists s \ \varphi_{e,s}^A(x)\downarrow$ è una condizione Σ_1^A, questa è computabile da A'. Perciò si ha $f \leq_T A' \leq_T B$, ovvero $\mathbf{b}$ computa un'estensione totale di ψ. $\square$

Il seguente lemma è cruciale per la dimostrazione dell'ultimo teorema di questo capitolo.

Lemma 4.18. *Siano* $\mathbf{a} \in \mathbf{D}$*, un grado di Turing, e* $A \in \mathbf{a}$*. Definiamo i seguenti due insiemi* $\mathbf{a}$*-c.e. disgiunti:* $A_0 := \{\langle e, x\rangle : \varphi_e^A(x) = 0\}$ *e* $A_1 := \{\langle e, x\rangle : \varphi_e^A(x) = 1\}$*. Allora ogni insieme che separa* A_0 *e* A_1 *ha grado PA su* $\mathbf{a}$*.*

Dimostrazione. Fissiamo $A \in \mathbf{a}$ e sia φ_i^A una qualsiasi funzione A-computabile a valori in $\{0, 1\}$. Sia C un insieme che separa A_0 e A_1. Si ha quindi $A_0 \subseteq C$ e $A_1 \cap C = \emptyset$. Consideriamo allora la seguente funzione:

$$f(x) := 1 - \chi_C(\langle i, x\rangle) = \begin{cases} 0 & \text{se } \langle i, x\rangle \in C, \\ 1 & \text{altrimenti.} \end{cases} \tag{4.4}$$

È chiaro che f è un'estensione totale di φ_i^A e $f \leq_T C$. $\square$

4.3 Un limite superiore

Definizione 4.19. Sia $(L, \trianglelefteq)$ un ordine lineare. Chiameremo relazione di adiacenza di L l'insieme $A_L = \{(x, y) \in L^2 : \ y \text{ è successore immediato di } x\}$.

Se $\equiv$ è la relazione di equivalenza su L definita in 1.4, chiameremo relazione di blocco di L l'insieme $B_L = \{(x, y) \in L^2 : x \equiv y\}$.

Lemma 4.20. *Per ogni ordine lineare computabile* $(L, \trianglelefteq)$ *si ha* $B_L \in \Sigma_2$ *e* $A_L \in \Pi_1$*.*

Dimostrazione. Analizziamo i due insiemi A_L e B_L. Abbiamo che $(x, y) \in A_L \iff x \triangleleft y \ \wedge (\forall z \in L \ (z \trianglelefteq x \vee y \trianglelefteq z)\ )$ e, dato che l'ordine $\trianglelefteq$ è computabile, l'insieme A_L risulta di classe Π_1. Per la relazione di blocco si ha $(x, y) \in B_L \iff \exists n \in \mathbb{N}\ \forall t \in L\ (t > n \to t \notin ([x, y] \cup [y, x]))$. La condizione $t \notin [x, y] \cup [y, x]$ equivale a $(t \triangleleft x \vee y \triangleleft t) \wedge (x \triangleleft t \vee t \triangleleft y)$ che è computabile. Quindi $B_L \in \Sigma_2$. $\square$

Osservazione 4.21. Per il Teorema di Post si ha che per ogni $A \in \Pi_1$, $A \leq_T \emptyset'$ e per ogni $B \in \Sigma_2$, $B \leq_T \emptyset''$. Dato quindi un ordine lineare computabile L, per il lemma precedente si ha che $\emptyset''$ computa A_L e B_L, ovvero $A_L \oplus B_L \leq_T \emptyset''$.

Il prossimo teorema, la cui dimostrazione è sostanzialmente la stessa data per il Teorema di Dushnik-Miller, garantisce che i due insiemi appena introdotti permettono sempre di calcolare almeno un'autoimmersione non banale.

Teorema 4.22. *Se $(L, \trianglelefteq)$ è un ordine lineare computabile infinito, allora L ha un'autoimmersione non banale che è Turing-riducibile a $A_L \oplus B_L$.*

Dimostrazione. Fissato un ordine lineare L, possiamo distinguere due casi.
Caso 1. L ha un blocco infinito.

Sia C un tale blocco e fissiamo $x_0 \in C$. Allora C deve essere chiuso sotto l'operazione di successore o sotto quella di predecessore. Supponiamo di essere nel primo caso (il secondo è totalmente analogo a questo). Definiamo la seguente funzione $f : L \to L$:

$$f(x) := \begin{cases} \mu y \ (x, y) \in A_L & \text{se } (x, x_0) \in B_L \\ x & \text{altrimenti} \end{cases} \tag{4.5}$$

È chiaro che f risulta un'autoimmersione non banale dell'ordine L. Notiamo che la sua definizione richiede solamente gli insiemi A_L e B_L, perciò $f \leq_T A_L \oplus B_L$.
Caso 2. Ogni blocco di L è finito.

Abbiamo già visto che in tal caso la condensazione $L/\equiv$ è densa. L'idea in questo caso è quella di costruire l'autoimmersione non banale f di L per stadi, utilizzando l'insieme B_L come oracolo. Costruiremo delle approssimazioni finite f_s di f e, ad ogni stadio, aggiungeremo un elemento nuovo al dominio. Ad ogni stadio utilizzeremo un blocco diverso dove scegliere l'immagine del nuovo elemento (evitando gli eventuali blocchi di minimo e massimo). Dovremo inoltre assicurarci che f rispetti l'ordine $\trianglelefteq$ e che sia diversa dall'identità.

Per fare ciò fissiamo x_m e x_M due elementi rispettivamente dei blocchi minimo e massimo di L (se esistono). Sia poi $x_0 \in L$ un elemento qualsiasi e $y_0 \in L$ tale che $x_0 \neq y_0$, $(y_0, x_m) \notin B_L$ e $(y_0, x_M) \notin B_L$.

- *Stadio 0* Poniamo $f_0 := \{(x_0, y_0)\}$.
- *Stadio s+1* Supponiamo di aver definito $f_s = \{(x_0, y_0), \ldots, (x_s, y_s)\}$. Sia $x_{s+1} := \mu x \in L \ (x \notin \mathrm{dom}(f_s))$. Sia inoltre $y_{s+1} := \mu y \in L \ [(\forall i \leq s \ (y, y_i) \notin B_L) \wedge (\forall i \leq s \ (x_i \lhd x_{s+1} \leftrightarrow y_i \lhd y)\ ) \wedge \ (y, x_m) \notin B_L \wedge \ (y, x_M) \notin B_L]$. Poniamo $f_{s+1} := f_s \cup \{(x_{s+1}, y_{s+1})\}$.

Osserviamo prima di tutto che la costruzione è ben definita. Supponiamo infatti di essere allo stadio $s+1$-esimo. Abbiamo che L è un insieme infinito mentre $\mathrm{dom}(f_s)$ è finito. Perciò esisterà un $x_{s+1} \in L$ (minimo) in cui f_s non è definita. Inoltre, dato che i blocchi sono ovunque densi e si evitano gli eventuali blocchi minimo e massimo (e $\mathrm{ran}(f_s)$ è finito), esisterà sempre un y_{s+1} con le proprietà richieste.

Per quanto riguarda la verifica che $f := \bigcup_s f_s$ sia un'autoimmersione non banale di L, notiamo che ad ogni stadio s la funzione f_s rispetta l'ordine $\trianglelefteq$ ristretto al suo dominio. Quindi anche la f rispetta l'ordine sul suo dominio, che è facile osservare essere L. Inoltre, per la condizione $x_0 \neq y_0$, si ottiene immediatamente che f non è l'identità.

Le condizioni poste all'interno della costruzione richiedono unicamente la conoscenza dell'ordine $(L, \trianglelefteq)$ e dell'insieme B_L. Utilizzando B_L come oracolo, per conoscere $f(x)$ è sufficiente attendere uno stadio s in cui $x \in \mathrm{dom}(f_s)$. Pertanto si ha che $f \leq_T B_L$. □

Utilizzando il risultato appena visto otteniamo un semplice corollario che determina un limite (superiore) alla complessità di almeno un'autoimmersione di qualsiasi ordine lineare computabile.

Corollario 4.23. *Per ogni ordine lineare computabile infinito, esiste una sua autoimmersione non banale che è computabile da $\emptyset''$.*

Dimostrazione. Fissato un ordine lineare computabile infinito, per il teorema precedente esiste un'autoimmersione non banale f tale che $f \leq_T A_L \oplus B_L$. Per l'Osservazione 4.21 otteniamo $f \leq_T \emptyset''$. □

Osservazione 4.24. Se per un qualche ordine lineare computabile L la relazione di blocco B_L è $\emptyset'$-computabile, allora esiste un'autoimmersione non banale di L tale che $f \leq_T \emptyset'$.

È facile vedere che tutto si può relativizzare ad un insieme A qualsiasi, ottenendo il seguente.

Corollario 4.25. *Per ogni ordine lineare infinito A-computabile, esiste un'autoimmersione non banale che è computabile da A''.*

Osservazione 4.26. Osserviamo che, nella dimostrazione del Teorema 4.15, non è stata utilizzata una procedura uniforme. Ovvero è stato costruito il programma solamente dopo aver fissato l'ordine lineare e, in particolare, in dipendenza del fatto che L abbia o meno un blocco infinito. Infatti si è dimostrato che $\forall e(\varphi_e$ è funzione caratteristica di un ordine lineare infinito $L \rightarrow \exists i\ \varphi_i^{\emptyset''}$ è un'autoimmersione non banale di $L)$

Se cerchiamo un metodo uniforme, cioè una funzione che riceva in input gli indici degli insiemi L e $\trianglelefteq$ e calcoli una relativa autoimmersione non banale, non è detto che l'insieme $\emptyset''$ possa bastare. Infatti se si segue, per esempio, la strategia della precedente dimostrazione, si ottiene il risultato seguente.

Teorema 4.27. *Esiste una funzione parziale $\psi : \mathbb{N}^2 \to \mathbb{N}$ tale che, per ogni $e \in \mathbb{N}$, se φ_e è la funzione caratteristica di un ordine lineare computabile e infinito $(L, \trianglelefteq)$ allora si ha:*

- *$\psi(e, \cdot) : L \to \mathbb{N}$ è un'autoimmersione non banale di L,*
- *$\psi \leq_T \emptyset^{(4)}$.*

Dimostrazione. Dobbiamo costruire un programma che sia in grado ripercorrere autonomamente la dimostrazione del teorema precedente. Per poter fare questo esso deve riuscire a determinare la struttura degli ordini lineari che riceve in input.

Prima di tutto dobbiamo valutare se esista o meno un blocco infinito. Questo è equivalente a porre la condizione

$$\exists x \in L \; \forall n \; \exists t \in L \; (t > n \;\wedge\; (x, t) \in B_L). \tag{4.6}$$

Abbiamo già visto che la condizione tra parentesi è Σ_2, perciò l'intera condizione è Σ_4.

Nel caso esista un blocco infinito possiamo scegliere il blocco infinito contenente il numero più piccolo (in $\mathbb{N}$). Questa è una condizione Π_3 e si ottiene dalla precedente togliendo il quantificatore esistenziale. A questo punto dobbiamo verificare se il blocco scelto sia chiuso sotto successore o predecessore. Se x_0 è un rappresentante del blocco, la condizione di chiusura sotto successore è

$$\forall z \exists y \; (x_0 \equiv z \to (z, y) \in A_L) \;). \tag{4.7}$$

La formula tra parentesi è di classe Π_2, quindi l'intera condizione è di classe Π_4. A questo punto la definizione dell'autoimmersione richiede unicamente la conoscenza di B_L e A_L (Σ_2 e Π_1).

Nel caso in cui tutti i blocchi siano finiti, le uniche verifiche aggiuntive da compiere sono quelle riguardanti l'esistenza o meno dei blocchi di minimo e di massimo. Per verificare, per esempio, l'esistenza di un blocco minimo è sufficiente, in questo caso, verificare l'esistenza di un elemento minimo. Basterà porre la condizione:

$$\exists x \in L \; \forall y \in L \; (x \triangleleft y). \tag{4.8}$$

Questa è di classe Σ_2. La condizione per il blocco massimo è del tutto simile, ed anche questa è Σ_2. Scelti quindi dei rappresentanti per gli eventuali blocchi minimo e massimo (usando condizioni Π_1), il resto della costruzione si effettua utilizzando B_L.

In definitiva la costruzione totale necessita di un oracolo di classe Σ_4 e di uno di classe Π_4, ovvero di uno di classe Δ_5. Per il Teorema di Post il programma è quindi computabile da $\emptyset^{(4)}$ ($\emptyset^{(3)}$ non è sufficiente). Tale programma determina perciò una funzione $\psi : \mathbb{N}^3 \to \mathbb{N}$ con le proprietà volute. $\square$

Osservazione 4.28. In altre parole, abbiamo appena dimostrato che: esiste un indice i per cui, qualsiasi sia e, se φ_e è funzione caratteristica di un ordine lineare infinito L, allora $\varphi^{\emptyset^{(4)}}_{\varphi_i(e)}$ è una sua autoimmersione non banale.

4.4 Teorema di Downey-Jockusch-Miller

Grazie al Teorema 4.22 abbiamo dimostrato che $\emptyset''$ è sempre in grado di computare almeno un'autoimmersione non banale di qualsiasi ordine lineare computabile. Ci si può domandare se questo possa valere anche per $\emptyset'$, dato che i risultati visti fino ad adesso non lo escludono. Il teorema di questa sezione, oltre a migliorare il Teorema di Downey-Lempp, risponde negativamente alla domanda.

Lemma 4.29. *Sia $(L, \trianglelefteq)$ un ordine lineare, $x_0 \in L$ e f una qualsiasi autoimmersione di L. Definiamo inoltre $S_\triangleleft(x_0) := \{y : y \triangleleft x_0 \ \vee \ y \triangleleft f(y)\}$ e $S_\triangleright(x_0) := \{y : y \triangleright x_0 \ \vee \ y \triangleright f(y)\}$. Allora valgono i seguenti due fatti:*

1. *se $x_0 \triangleleft f(x_0)$, l'insieme $S_\triangleleft(x_0)$ è chiuso sotto successore,*
2. *se $x_0 \triangleright f(x_0)$, l'insieme $S_\triangleright(x_0)$ è chiuso sotto predecessore.*

Dimostrazione. (1) Osserviamo che basta verificare la proprietà per gli elementi che ammettono successore. Sia quindi $y \in S_\triangleleft(x_0)$ uno di questi e sia $z \in L$ suo successore. Possono verificarsi due casi. Se $y \triangleleft f(y)$, allora si ha $y \triangleleft z \trianglelefteq f(y) \triangleleft f(z)$, perché f è autoimmersione di L. Quindi $z \in S_\triangleleft(x_0)$. Se $y \triangleleft x_0$, allora $z \trianglelefteq x_0$, ovvero $z \triangleleft x_0 \vee z = x_0$. Dato che $x_0 \in S_\triangleleft(x_0)$, in entrambi i casi $z \in S_\triangleleft(x_0)$.

(2) Il ragionamento è completamente simmetrico a quello fatto per (1). $\square$

Per la dimostrazione del prossimo teorema abbiamo bisogno di dare delle definizioni e fissare alcune notazioni. D'ora in avanti indicheremo con A_0 e A_1 gli insiemi definiti nel Lemma 4.18, nel caso in cui $\mathbf{a} = \mathbf{0}'$ e $A = \emptyset'$.

Prima di tutto dobbiamo definire approssimazioni computabili degli insiemi A_0 e A_1. Nel seguito consideriamo sempre $i \in \{0, 1\}$. Osserviamo che, per il Teorema di Post, A_0 e A_1 sono di classe Σ_2. Nell'Esempio 4.14 abbiamo visto che Fin è un insieme Σ_2-completo, pertanto esisterà una funzione computabile h tale che

$$x \in A_i \iff W_{h(x,i)} \text{ è finito.} \tag{4.9}$$

Definiamo quindi i seguenti insiemi computabili per $s \in \mathbb{N}$:

$$A_{i,s} := \{x : W_{h(x,i),s} = W_{h(x,i),s+1}\}. \tag{4.10}$$

Tali insiemi determinano delle approssimazioni computabili di A_0 e A_1. Infatti è facile convincersi che

$$x \in A_i \iff \forall^\infty s \; x \in A_{i,s} \tag{4.11}$$

(dove $\forall^\infty s$ sta per $\exists n \in \mathbb{N} \; \forall s > n$).

Nella prossima dimostrazione avremo anche bisogno di funzioni computabili g_i tali che $W_{g_i(x)} = \{s : x \notin A_{i,s}\}$. Per definirle basta applicare il Teorema s-m-n alla funzione

$$\psi(x,s) := \begin{cases} 0 & \text{se } x \notin A_{i,s}, \\ \uparrow & \text{altrimenti.} \end{cases} \tag{4.12}$$

È importante osservare che se $x \notin A_i$ allora $W_{g_i(x)}$ è infinito. Inversamente se $x \in A_i$ allora $W_{g_i(x)}$ è finito.

Teorema 4.30 (Teorema di Downey-Jockusch-Miller)**.** *Esiste un ordine lineare computabile infinito, tale che ogni sua autoimmersione non banale ha grado PA su* $\mathbf{0}'$.

Dimostrazione. Costruiremo un ordine lineare $(L, \trianglelefteq)$ a stadi computabili. Gli elementi 0 e 1 risulteranno rispettivamente il minimo e il massimo di L. Ad ogni stadio s definiremo delle approssimazioni finite $(L_s, \trianglelefteq_s)$ di L, in modo tale da avere per ogni s, $L_s \subseteq L_{s+1}$, $\trianglelefteq_s \subseteq \trianglelefteq_{s+1}$ e $|L_{s+1} \setminus L_s| \leq 1$. Dovremo inoltre fare in modo che ogni autoimmersione di L abbia grado $\gg \mathbf{0}'$.

L'idea è quella di costruire $(L, \trianglelefteq)$ in modo che ogni sua autoimmersione non banale sia in grado di computare un insieme C che separi gli insiemi A_0 e A_1 definiti sopra.

Vediamo ora, in maniera intuitiva, la strategia che utilizzeremo. Supponiamo di aver definito l'ordine lineare computabile $(L, \trianglelefteq)$ e sia f una sua autoimmersione non banale. Esisterà quindi $x_0 \in L$ tale che $x_0 \neq f(x_0)$. Supponiamo per esempio che $x_0 \triangleleft f(x_0)$ (tutto si può ripetere nel caso $x_0 \triangleright f(x_0)$). Considerando eventualmente $f(x_0)$ al posto di x_0, possiamo supporre $x_0 \neq 0$. Allora, per il Lemma 4.29, l'insieme $S_\triangleleft(x_0)$ è chiuso sotto successore. Per induzione è facile vedere che, se $y \in S_\triangleleft(x_0)$ e $y \triangleleft z$ con $[y,z]$ finito, allora anche $z \in S_\triangleleft(x_0)$. Inoltre è immediato che $0 \in S_\triangleleft(x_0)$ e $1 \notin S_\triangleleft(x_0)$.

Notiamo che l'insieme $S_\triangleleft(x_0)$ è computabile da f, perciò è sufficiente verificare l'esistenza di un insieme $C \leq_T S_\triangleleft(x_0)$ che separi A_0 e A_1. Per fare questo dovremo scegliere un elemento $z_0 \neq 0, 1$, porre $0 \triangleleft z_0 \triangleleft 1$ e, usando le approssimazioni $A_{i,s}$, fare in modo che l'intervallo $[0, z_0]$ rimanga finito se $0 \in A_0$ e che $[z_0, 1]$ rimanga finito nel caso in cui $0 \in A_1$ (ricordiamo che $A_0 \cap A_1 = \emptyset$).

Supponiamo per esempio che sia $0 \in A_0$. Allora $[0, z_0]$ rimarrà finito e, dato che $0 \in S_\triangleleft(x_0)$, avremo che $z_0 \in S_\triangleleft(x_0)$. Analogamente, nel caso in cui $0 \in A_1$, avremo che $z_0 \notin S_\triangleleft(x_0)$. Pertanto, nella costruzione dell'insieme C, porremo $0 \in C$ se e solo se $z_0 \in S$. Osserviamo ora che uno tra $[0, z_0]$ e $[z_0, 1]$, chiamiamolo $[a, b]$, avrà la proprietà che $a \in S_\triangleleft(x_0) \iff b \notin S_\triangleleft(x_0)$. Ripeteremo quindi lo stesso procedimento utilizzando l'intervallo $[a, b]$ al posto di $[0, 1]$. Sceglieremo quindi un elemento $z_1 \notin \{0, 1, z_0\}$ e porremo $a \triangleleft z_1 \triangleleft b$, assicurandoci che $[a, z_1]$ sia finito se $z_1 \in A_0$ e che $[z_1, b]$ sia finito se $z_1 \in A_1$. Porremo poi $1 \in C$ se e solo se $z_1 \in S_\triangleleft(x_0)$. Iterando questo procedimento otterremo quindi un insieme C che separa A_0 da A_1.

Osserviamo però che, durante la costruzione, non sarà possibile effettuare le scelte tra gli intervalli di L nel modo che abbiamo descritto sopra. Infatti la costruzione dovrà risultare computabile e non semplicemente $S_\triangleleft(x_0)$-computabile. Quindi, per esempio, non saremo in grado di determinare, durante la costruzione, l'intervallo $[a, b]$ con la proprietà $a \in S_\triangleleft(x_0) \iff b \notin S_\triangleleft(x_0)$. Pertanto ammetteremo che la costruzione inserisca alcuni elementi anche all'interno degli intervalli che dovranno rimanere finiti. Per esempio, ci sarà un'unico elemento che svolgerà il ruolo, descritto sopra, di z_0, ma ce ne saranno al più due del tipo z_1 e, in generale, fino a 2^n del tipo z_n. Lasceremo quindi che sia $S_\triangleleft(x_0)$ a scegliere quelli da inserire in C.

All'interno della costruzione definiremo, usando g_0 e g_1, una funzione parziale binaria θ, la quale garantirà che alcuni intervalli rimangano finiti. Infatti faremo in modo che, quando $W_{\theta(a,b)}$ è finito, anche l'intervallo $[a, b]$ risulti finito. Definiremo inoltre una funzione unaria r. Quando porremo $r(x) = n$ intenderemo che x svolge il ruolo di uno dei possibili z_n definiti sopra.

Vediamo i dettagli della costruzione.

- *Stadio 0.* Poniamo $L_0 := \{0, 1\}$, $0 \triangleleft_0 1$ e $r(0) = r(1) = -1$. Scegliamo inoltre $\theta(0, 1)$ in modo tale che $W_{\theta(0,1)} = \mathbb{N}$.

- *Stadio s+1.* Supponiamo di aver definito $(L_s, \trianglelefteq_s)$. Diremo che l'intervallo $[x, y]$ richiede attenzione se

  (a) $x, y \in L_s$,

  (b) y è successore immediato di x in L_s,

  (c) per ogni $x', y' \in L_s$ tali che $x' \trianglelefteq_s x$, $y \trianglelefteq_s y'$ con $\theta(x', y') \downarrow$, se $\langle x', y' \rangle \leq \langle x, y \rangle$ allora $|W_{\theta(x',y'),s}| \geq \langle x, y \rangle$.

(la terza condizione garantisce che, se un intervallo $[x', y']$ deve rimanere finito, da un certo stadio in poi non verranno più inseriti elementi al suo interno).

Se nessun intervallo di L_s richiede attenzione allo stadio $s+1$, poniamo $L_{s+1} = L_s$, $\trianglelefteq_{s+1} = \trianglelefteq_s$ e lasciamo invariate le funzioni θ e r.

Altrimenti, fissiamo il minimo $\langle x, y\rangle$ tra gli intervalli $[x, y]$ che richiedono attenzione. Consideriamo $z := \mu t\ t \notin L_s$. Definiamo allora $L_{s+1} := L_s \cup \{z\}$ ed estendiamo $\trianglelefteq_s$ ponendo $x \triangleleft_{s+1} z \triangleleft_{s+1} y$. Definiamo quindi $r(z) := \max\{r(x), r(y)\} + 1$, $\theta(x, z) := g_0(r(z))$ e $\theta(z, y) := g_1(r(z))$. Diremo che l'intervallo $[x, y]$ ha ricevuto attenzione allo stadio $s+1$.

Dimostriamo ora alcuni fatti per assicurarci che la costruzione garantisca la tesi del teorema.

(1) L'ordine $(L, \trianglelefteq)$ *è computabile.* Dimostriamo prima di tutto che $L = \mathbb{N}$. Per farlo verifichiamo per induzione che, ad ogni stadio s, esistono due elementi $x \triangleleft_s y$, consecutivi in L_s, tali che per ogni $x', y' \in L_s$, con $x' \trianglelefteq_s x$, $y \trianglelefteq_s y'$ e $\theta(x', y') \downarrow$, si ha che $W_{\theta(x', y')}$ è infinito.

Il caso $s = 0$ è immediato. Supponiamo ora che questo sia vero allo stadio s per un intervallo $[x, y]$. Se non vengono inseriti elementi in L_s, la condizione rimane vera. Se viene inserito un elemento z fuori dall'intervallo $[x, y]$, la condizione rimane ancora soddisfatta sempre per gli stessi x, y. Infatti se per esempio $z \triangleleft_{s+1} x$ allora, per l'unico y' tale che $\theta(z, y') \downarrow$, deve essere $y' \triangleleft_{s+1} y$. Pertanto z non crea nessun problema. Supponiamo ora che z venga inserito all'interno di $[x, y]$. Dato che gli insiemi A_0 e A_1 sono disgiunti, si ha $r(z) \notin A_0$ oppure $r(z) \notin A_1$. Nel primo caso abbiamo che $W_{\theta(x,z)} = W_{g_0(r(z))}$ è infinito mentre nel secondo caso $W_{\theta(z,y)} = W_{g_1(r(z))}$ è infinito. Pertanto la condizione continua a valere allo stadio $s+1$, grazie all'intervallo $[x, z]$ se $r(z) \notin A_0$ o all'intervallo $[z, y]$ se $r(z) \notin A_1$.

Supponiamo quindi per assurdo che $\bigcup_s L_s \neq \mathbb{N}$. Dato che gli elementi vengono inseriti in L seguendo l'ordine di $\mathbb{N}$, questo significa che $\bigcup_s L_s$ è finito. Fissiamo allora uno stadio t tale che $\forall s \geq t\ L_s = L_t$. Scegliamo un intervallo $[x, y]$ di L_t che soddisfa la condizione descritta sopra. Allora, ad uno stadio $s \geq t$ sufficientemente grande, tale intervallo richiederà attenzione, contro l'ipotesi che $L_s = L_t$.

Dato che $\bigcup_s L_s = \mathbb{N}$ e le istruzioni all'interno della costruzione sono tutte computabili, anche l'ordine finale L risulta computabile.

(2) Se $\theta(x', y') \downarrow$ *e* $W_{\theta(x', y')}$ *è finito, allora l'intervallo* $[x', y']$ *è finito in* L. Sia $n = |W_{\theta(x', y')}|$ e supponiamo che $[x', y']$ abbia ricevuto attenzione allo stadio s. Allora, ad ogni stadio $t > s$, se un intervallo $[x, y]$, con $x' \trianglelefteq_t x$ e $y \trianglelefteq_t y'$, riceve attenzione, si ha necessariamente $\langle x, y\rangle < \langle x', y'\rangle$ oppure $\langle x, y\rangle \leq n$ (per

la condizione (c) della costruzione). Dato che esiste un numero finito di coppie di questo tipo, ed ognuna riceve attenzione al più una volta, l'intervallo $[x', y']$ è finito.

(3) Ogni autoimmersione non banale di L computa un insieme C che separa A_0 e A_1. Sia f un'autoimmersione non banale di L e $x_0 \in L$ tale che $f(x_0) \neq x_0$. Supponiamo che $x_0 \lhd f(x_0)$ e $x_0 \neq 0$. Usando $S_\lhd(x_0)$, che è computabile da f, come oracolo, definiamo ricorsivamente una sequenza di intervalli $[a_n, b_n]$ di L tali che, per ogni $n \in \mathbb{N}$:

(i) $a_n \in S_\lhd(x_0)$ e $b_n \notin S_\lhd(x_0)$,

(ii) $\theta(a_n, b_n) \downarrow$,

(iii) per ogni $a', b' \in L$ con $a' \trianglelefteq a_n$ e $b_n \trianglelefteq b'$, se $\theta(a', b') \downarrow$ allora l'insieme $W_{\theta(a',b')}$ è infinito.

Iniziamo ponendo $[a_0, b_0] := [0, 1]$. Supponiamo ora di aver definito $[a_n, b_n]$ soddisfacente le proprietà richieste sopra. Osserviamo che l'intervallo $[a_n, b_n]$ deve avere almeno un elemento al suo interno. Infatti, per la proprietà (i) e per il lemma 4.29, b_n non può essere successore immediato di a_n. Sia allora z_n il primo elemento inserito in $[a_n, b_n]$ durante la costruzione. Se $z_n \notin S_\lhd(x_0)$ poniamo $[a_{n+1}, b_{n+1}] := [a_n, z_n]$ altrimenti, se $z_n \in S_\lhd(x_0)$ poniamo $[a_{n+1}, b_{n+1}] := [z_n, b_n]$.

Verifichiamo induttivamente che valgono le condizioni richieste. Per $n = 0$ tutte e tre le condizioni sono banalmente soddisfatte. Supponiamo ora che valgano per $[a_n, b_n]$. Allora (i) e (ii) valgono immediatamente per come abbiamo scelto $[a_{n+1}, b_{n+1}]$. Per la condizione (iii) è sufficiente verificare che $W_{\theta(a_{n+1},b_{n+1})}$ è un insieme infinito. Possono verificarsi due casi. Se $z_n \notin S_\lhd(x_0)$ allora, dato che $a_{n+1} = a_n \in S_\lhd(x_0)$ e $b_{n+1} = z_n$, l'intervallo $[a_{n+1}, b_n + 1]$ è infinito. Per il punto (2), dimostrato sopra, questo implica che l'insieme $W_{\theta(a_{n+1},b_{n+1})}$ è infinito. Se $z_n \in S_\lhd(x_0)$ il ragionamento è il medesimo.

Osserviamo ora che per ogni n si ha $r(a_n), r(b_n) < n$ e $r(z_n) = n$. Infatti $r(a_0) = r(b_0) = -1$ e $r(z_0) = \max\{r(a_0), r(b_0)\} + 1 = 0$. Inoltre, dato che si ha $a_{n+1} = z_n$ e $b_{n+1} = b_n$, oppure $a_{n+1} = a_n$ e $b_{n+1} = b_n$, se supponiamo la condizione vera per n, otteniamo immediatamente la tesi.

Definiamo allora il seguente insieme

$$C := \{n : z_n \in S_\lhd(x_0)\}. \tag{4.13}$$

Chiaramente si ha $C \leq_T S_\lhd(x_0)$. Verifichiamo ora che $A_0 \subseteq C$ e $C \cap A_1 = \emptyset$. Sia $n \in A_0$. Allora $W_{\theta(a_n,z_n)} = W_{g_0(r(z_n))} = W_{g_0(n)} = \{s : n \notin A_{0,s}\}$ è finito. Pertanto, sempre per il punto (2), anche l'intervallo $[a_n, z_n]$ è finito e, dato che $a_n \in S_\lhd(x_0)$, anche $z_n \in S_\lhd(x_0)$. Quindi $z_n \in C$. Simmetricamente se $n \in A_1$, allora è $W_{\theta(z_n,b_n)}$ ad essere finito e quindi è tale anche l'intervallo $[z_n, b_n]$. Dato che $b_n \notin S_{\triangleleft}(x_0)$, anche $z_n \notin S_{\triangleleft}(x_0)$. Quindi $z_n \notin C$.

Osserviamo che abbiamo costruito C supponendo l'esistenza di $x_0 \in L$ con $x_0 \triangleleft f(x_0)$. Se un tale elemento non esiste, scegliamo un $x_1 \in L$ tale che $f(x_1) \triangleleft x_1$ e $x_1 \neq 1$. Definiamo allora, usando $S_{\triangleright}(x_1)$, una successione di intervalli $[a_n, b_n]$ in maniera duale a quanto fatto sopra, in modo tale che, per ogni n, $a_n \notin S_{\triangleright}(x_1)$ e $b_n \in S_{\triangleright}(x_1)$ (le condizioni (i) e (ii) rimangono le stesse). Definendo poi gli z_n allo stesso modo, poniamo $C = \{n : z_n \notin S_{\triangleright}(x_1)\}$. Ricordando infine che $S_{\triangleright}(x_1)$ è chiuso per predecessore si dimostra facilmente che $A_0 \subseteq C$ e $C \cap A_1 = \emptyset$. $\square$

Osservazione 4.31. Si può dimostrare che, per ogni insieme A, esistono gradi PA su $\deg(A)$ strettamente minori di $\deg A'$. In particolare esiste un insieme di grado PA su $\mathbf{0}'$ strettamente minore di $\mathbf{0}''$.

È chiaro perciò che rimane ancora aperto il problema di stabilire il minimo grado di Turing $\mathbf{c}$ capace di calcolare almeno un'autoimmersione non banale di ogni ordine lineare computabile ed infinito. La conclusione a cui siamo arrivati è che tale grado deve soddisfare: $\mathbf{c} \gg \mathbf{0}'$ e $\mathbf{c} \leq \mathbf{0}''$.

Il problema sarebbe ovviamente risolto se si costruisse un ordine lineare per cui ogni autoimmersione non banale computi $\emptyset''$. Un'altra soluzione sarebbe quella di trovare un grado PA su $\mathbf{0}'$ strettamente minore di $\mathbf{0}''$ che calcoli un'autoimmersione non banale per ogni ordine lineare computabile. Il problema resta attualmente insoluto.

Bibliografia

[DJM06] R. G. Downey, C. Jockusch, J.S. Miller, *On self-embeddings of computable linear orderings,* Annals of Pure and Applied Logic **138** (2006), 52-61.

[DL99] R.G. Downey, S. Lempp, *On the proof theoretical strength of the Dushnik-Miller theorem for countable linear orders,* in: M.M. Arslanov, S. Lempp, *Recursion Theory and Complexity*, de Gruyter (1999), 55-58.

[DM40] B. Dushnik, E.W. Miller, *Concerning similarity transformations of linearly ordered sets,* Bull. Amer. Math. Soc. **46** (1940), 322-326.

[Ro82] J. G. Rosenstein, *Linear Orderings,* Academic Press (1982).

[So87] R.I. Soare, *Recursively Enumerable Sets and Degrees,* Springer-Verlag (1987).