ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI MATEMATICA

Corso di laurea triennale in

MATEMATICA

Tesi di Laurea in

INFORMATICA

GLI ALGORITMI DI MARKOV E IL PROBLEMA DELLA PAROLA

PRESENTATA DA
Anita Miale

RELATORE
Chiar.mo Prof.
Simone Martini

Sessione IV
Anno Accademico 2024 - 2025

a me stessa

ἐλπίζε ἀνέλπιστον

"Spera l'insperabile"

Eraclito

Introduzione

L'informatica affonda le sue radici su alcune domande primarie: *Che cos'è un algoritmo? Quali sono i suoi limiti? Che cosa può, e che cosa non può, essere calcolato?* [5]

Nella storia dell'uomo gli esempi di algoritmi sono molteplici; uno dei più celebri è l'algoritmo di Euclide per il calcolo del massimo comune divisore tra due numeri interi, reso noto dal matematico greco intorno al 300 a.C. Il concetto di *algoritmo* è un concetto pre-matematico, al quale, per lungo tempo, è stata assegnata una definizione intuitiva; un algoritmo è inteso come una sequenza di istruzioni che possono essere svolte in maniera puramente meccanica con l'obiettivo di portare a termine un determinato compito [10]. In generale, si richiedono tre caratteristiche: la precisione delle istruzioni, cioè non deve mai esserci ambiguità sul passo successivo della computazione; la generalità, cioè la capacità di lavorare su una ampia classe di dati iniziali ed infine, la capacità di produrre il risultato desiderato in un tempo finito [6].

Quando un algoritmo è utilizzato per calcolare i valori di una funzione numerica, tale funzione è detta *effettivamente calcolabile* [1]. I matematici possono limitarsi ad una definizione intuitiva del concetto di calcolabilità nel momento in cui tale nozione compare solo in affermazioni positive, cioè in quei casi in cui un algoritmo è già stato scoperto. Al contrario, se si vuole dimostrare che una certa funzione non sia effettivamente calcolabile è necessario dare una definizione precisa e rigorosa di calcolabilità effettiva [7, Cap. 5, § 5.1].

Intorno agli anni Trenta del secolo scorso, sono state sviluppate innumerevoli teorie nel tentativo di presentare una formalizzazione coerente dei concetti intuitivi di algoritmo e calcolabilità effettiva. Il primo obiettivo della presente tesi è di ripercorrere i risultati di due diversi matematici: A.A. Markov (1903-1979) e Alan M. Turing (1912-1954). Questo rappresenterà il contenuto dei Capitoli 2 e 3.

Nel **Capitolo 2** si introduce una particolare classe di algoritmi che prendono il nome di *algoritmi di Markov* o *algoritmi normali*. Questi operano sulle parole di un alfabeto e fondano la propria costruzione sull'operazione di sostituzione di una parola a un'al-

tra; i passi della computazione sono determinati dalle *produzioni* che costituiscono lo schema di algoritmo. Il secondo paragrafo presenta diversi esempi significativi, quali: gli algoritmi per la concatenazione, la cancellazione del primo carattere di una parola, l'inversione di parole e il calcolo del valore assoluto della differenza tra due numeri naturali. Nel paragrafo successivo si dimostra che la *composizione di algoritmi normali* è ancora un algoritmo normale. Infine, si definisce una particolare classe di funzioni numeriche: le *funzioni parzialmente calcolabili nel senso di Markov* (o parzialmente Markov-calcolabili), nel caso in cui si lavori con funzioni numeriche parziali, oppure le *funzioni calcolabili nel senso di Markov* (o Markov-calcolabili), nel caso di funzioni totali. Come esempi vengono presentate la funzione di successore e la funzione nulla [7, Cap. 5, § 5.1]. Queste nozioni di Markov-calcolabilità rappresentano una delle possibili definizioni rigorose di calcolabilità effettiva, che nel capitolo successivo verrà confrontata con un ulteriore formalismo, cioè le macchine di Turing.

Il **Capitolo 3** presenta le *macchine di Turing*, un modello di calcolo costituito da un nastro potenzialmente infinito diviso in celle, una testina di lettura ed un insieme finito di stati interni. La computazione avviene in momenti discreti di tempo; in base allo stato corrente e al simbolo letto, la macchina può scrivere un nuovo simbolo, spostarsi a destra o a sinistra, oppure fermarsi [9]. Tra gli esempi di macchine di Turing si possono citare: la macchina che scrive indefinitamente su nastro vuoto una sequenza di simboli "0" e "1" e quella che aggiunge "1" alla sinistra di una parola, senza fermarsi mai. Nel paragrafo successivo, il concetto intuitivo di calcolabilità effettiva viene identificato con la nozione rigorosa di *calcolabilità nel senso di Turing* (o Turing-calcolabilità). Tra gli esempi di funzioni Turing-calcolabili si trovano: la funzione di successore e di addizione. La parte centrale del capitolo è dedicata alla dimostrazione dell'equivalenza tra i due concetti di calcolabilità; si prova quindi che la classe delle funzioni calcolabili nel senso di Turing coincide con la classe delle funzioni calcolabili nel senso di Markov. Dimostrare che ogni funzione Turing-calcolabile è parzialmente Markov-calcolabile richiede la costruzione esplicita di un algoritmo normale, che imiti l'azione dell'algoritmo di Turing; similmente, per provare che ogni funzione parzialmente Markov-calcolabile è Turing-calcolabile, è necessario costruire una macchina di Turing che imiti l'azione dell'algoritmo di Markov [7, Cap. 5, § 5.2]. Il capitolo si conclude con la *tesi di Church-Turing*, di cui vengono fornite due diverse formulazioni; la prima dichiara che la classe definita in maniera intuitiva delle funzioni effettivamente calcolabili coincide con la classe delle funzioni calcolabili nel senso di Turing [1], [8]; la seconda, che la nozione intuitiva di algoritmo coincide con la nozione precisa di algoritmo di Turing [10]. A causa della natura intuitiva dei concetti di algoritmo e calcolabilità effettiva, non è possibile fornire una dimostrazione rigorosa

della validità della tesi, ma vi sono due principali argomentazioni a suo sostegno: la robustezza della definizione di macchina di Turing [2], [5] e l'equivalenza tra i diversi formalismi proposti per la calcolabilità [2], [8]. Infine, viene presentato il *principio di normalizzazione* di Markov che afferma che ogni algoritmo è *pienamente equivalente* a un algoritmo normale. Grazie ai risultati dimostrati in precedenza, tale principio risulta equivalente alla tesi di Church-Turing.

La seconda parte della tesi si occupa di indagare i limiti del concetto di algoritmo, presentando un risultato di indecidibilità, cioè l'*indecidibilità del problema della parola*. Il problema della parola è caratterizzato da un gruppo G con insieme di generatori $X = \{x_1, \ldots, x_n\}$; data una parola ω nell'insieme X si vuole determinare se $\omega = 1$ in G, dove il simbolo 1 indica l'elemento neutro del gruppo. Tale problema è un esempio di problema di decisione: è caratterizzato dalla presenza di una lista $\mathscr{L}$ di domande e richiede l'esistenza di un algoritmo che assegni la risposta corretta (della forma sì/no) a tutte le domande della lista. Una soluzione positiva consiste nel presentare un algoritmo che svolga quanto richiesto; al contrario, una soluzione negativa consiste nel dimostrare che non esiste alcun algoritmo per tale scopo; in questo caso il problema si dice *indecidibile* [2]. La tesi di Church-Turing permette di dimostrare tale risultato di indecidibilità sfruttando non una procedura algoritmica qualsiasi, ma le macchine di Turing.

Il **Capitolo 1** raccoglie alcune nozioni di teoria dei gruppi, che forniscono gli strumenti per formalizzare il problema della parola e dimostrare gli enunciati ad esso collegati. Vengono presentate le definizioni di semigruppo e gruppo, con particolare attenzione alla nozione di *gruppo finitamente generato*. Inoltre, a partire dall'insieme delle parole ridotte su X, si definisce il *gruppo libero su* X. Si introduce la nozione di *presentazione* di un gruppo e di *prodotto libero* di una famiglia di gruppi. Infine, si presentano le *estensioni HNN*, dove "HNN" sta per Higman-Neumann-Neumann, in onore dei matematici che hanno introdotto tale costruzione e il *lemma di Britton*, che fornisce una condizione necessaria affinché una parola rappresenti l'elemento neutro in un'estensione HNN [4], [9].

Il **Capitolo 4** introduce una formalizzazione del problema della parola, che sfrutta i concetti di *insieme ricorsivamente enumerabile* (r.e.), ovvero un insieme per cui esiste una macchina di Turing che lo *enumera*, e di *insieme ricorsivo*. Si dimostra che esiste un insieme r.e., ma non ricorsivo; la macchina di Turing che enumera tale insieme si denota con T^*. Il paragrafo successivo affronta il problema della parola per i semigruppi e presenta il *teorema di Markov-Post* che stabilisce l'esistenza di un semigruppo finitamente presentato con problema della parola indecidibile. La dimostrazione si basa sulla costruzione del semigruppo $\Gamma(T)$ associato ad una macchina di Turing T; se questa è

T^*, allora il semigruppo $\Gamma(T^*)$ è quello cercato. Il risultato principale del capitolo è il *teorema di Novikov-Boone-Britton* che assicura l'esistenza di un gruppo $\mathscr{B}$ finitamente presentato con problema della parola indecidibile. Analogamente alla dimostrazione del teorema di Markov-Post, si definisce il gruppo $\mathscr{B} = \mathscr{B}(T)$ associato ad una macchina di Turing T; la tesi segue dalla scelta di T come T^* e dal *lemma di Boone*. La dimostrazione di quest'ultimo, soprattutto della condizione necessaria, non è immediata, ma richiederà diversi lemmi ausiliari e un ampio uso della teoria delle estensioni HNN e del lemma di Britton [9].

Indice

Capitolo 1

Nozioni preliminari

Richiamiamo di seguito alcune nozioni di teoria dei gruppi, utili principalmente per la comprensione dell'ultimo capitolo. Si è preferito raccogliere tali risultati in una sezione apposita, così da non interrompere l'esposizione successiva. Nell'eventualità che il lettore voglia approfondire i concetti riportati di seguito oppure sia interessato alle dimostrazioni di teoremi e proposizioni solamente enunciati, lo si invita a consultare [4, Cap. 1] oppure [9, Cap. 11].

1.1 Semigruppi e gruppi finitamente generati

Nel presente paragrafo riportiamo le definizioni di semigruppo, gruppo e sottogruppo, con particolare attenzione alle nozioni di sottogruppo generato da un insieme e di gruppo finitamente generato.

Definizione 1.1. *Sia G un insieme. Un'**operazione binaria** su G è una funzione $* : G \times G \to G$ che ad ogni coppia ordinata di elementi di G associa un elemento di G.*

Definizione 1.2. *Un'operazione binaria $(a, b) \mapsto a * b$ su un insieme G è **associativa** se*

$$(a * b) * c = a * (b * c),$$

per ogni $a, b, c \in G$.

Definizione 1.3. *Un **semigruppo** $(G, *)$ è un insieme non vuoto G dotato di un'operazione binaria associativa.*

Definizione 1.4. *Un **gruppo** è un semigruppo G che soddisfa le seguenti condizioni:*

- *esiste $e_G \in G$ tale che $e_G * a = a = a * e_G$ per ogni $a \in G$ (esistenza dell'elemento neutro);*

- *per ogni $a \in G$ esiste $b \in G$ tale che $a * b = e_G = b * a$ (esistenza dell'elemento inverso).*

Al fine di alleggerire la notazione, d'ora in avanti indichiamo $a * b$ con ab.

Definizione 1.5. *Sia G un gruppo e sia a un elemento di G; l'**ordine** di a è il più piccolo intero $n > 0$ tale che $a^n = e_G$. Se tale n non esiste, si dice che a ha **ordine infinito**.*

Non richiamiamo la definizione di sottogruppo, ma presentiamo il seguente criterio:

Proposizione 1.6. *Un sottoinsieme non vuoto S di un gruppo G è un **sottogruppo** di G se e solo se $st^{-1} \in S$, per ogni $s, t \in S$.*

Utilizziamo la notazione $S \leq G$ per indicare che S è un sottogruppo del gruppo G.

Corollario 1.7. *Sia G un gruppo e $\{H_i : i \in I\}$ una famiglia non vuota di sottogruppi di G, allora $\bigcap_{i \in I} H_i$ è un sottogruppo di G.*

Definizione 1.8. *Sia G un gruppo e X un sottoinsieme di G. Sia $\{H_i : i \in I\}$ la collezione di tutti i sottogruppi di G che contengono X. Allora $\bigcap_{i \in I} H_i$ è detto il **sottogruppo** di G **generato dall'insieme** X e si denota con $\langle X \rangle$. Gli elementi di X sono i **generatori** del sottogruppo $\langle X \rangle$.*

Proposizione 1.9. *Sia G un gruppo e X un sottoinsieme di G; allora gli elementi di $\langle X \rangle$ sono della forma ${a_1}^{n_1} {a_2}^{n_2} \dots {a_r}^{n_r}$, con $a_i \in X, n_i \in \mathbb{Z}$.*

Osserviamo che esiste sempre un sottoinsieme X di G che genera G; infatti è sufficiente considerare G stesso come insieme di generatori.

Definizione 1.10. *Sia G un gruppo e $X = \{a_1, \dots, a_n\}$ un sottoinsieme di G. G si dice **finitamente generato** se $G = \langle a_1, \dots, a_n \rangle$. In tal caso $a_1, \dots, a_n$ sono detti un **sistema di generatori per** G.*

Definizione 1.11. *Un gruppo G si dice **ciclico** se è generato da un solo elemento, cioè se esiste $a \in G$ tale che $G = \langle a \rangle = \{a^n : n \in \mathbb{Z}\}$.*

Definiamo, infine, una particolare tipologia di sottogruppo:

Definizione 1.12. *Un sottogruppo K di un gruppo G è un **sottogruppo normale** se $aKa^{-1} = K$ per ogni $a \in G$. In particolare, il **sottogruppo normale generato da un insieme** $S \subset G$ è l'intersezione di tutti i sottogruppi normali di G che contengono S.*

1.2 Gruppi liberi, prodotti liberi, generatori e relazioni

Iniziamo con l'introdurre i concetti di parola e parola ridotta, che ci serviranno per definire la nozione di gruppo libero su un insieme X. Successivamente, presentiamo un metodo per descrivere un gruppo mediante i suoi generatori e relazioni, grazie al quale definiamo la nozione di gruppo finitamente presentato. Infine, concludiamo con la definizione di prodotto libero di una famiglia di gruppi.

Definizione 1.13. *Sia X un **alfabeto**, cioè un insieme non vuoto di simboli. Una **parola** non vuota in X è un elemento ω della forma*

$$\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n},$$

*dove $x_i \in X, \epsilon_i = \pm 1$, con $i = 1, \dots, n$ e $n \geq 1$. La sequenza vuota di simboli è detta **parola vuota** e si denota con 1.*

Data la parola $\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$; definiamo la *lunghezza* di ω come il numero naturale n; la lunghezza della parola vuota è 0.

Definizione 1.14. *Una parola ω in X si dice **positiva** se è vuota oppure ha solo esponenti positivi, cioè $\epsilon_i = +1$, per ogni $i = 1, \dots, n$.*

Definizione 1.15. *Sia $\omega = x_1^{\epsilon_1} \dots x_n^{\epsilon_n}$ una parola in X, allora la sua **inversa** è la parola $\omega^{-1} = x_n^{-\epsilon_n} \dots x_1^{-\epsilon_1}$.*

Definizione 1.16. *Una parola ω in X si dice **ridotta** se è vuota oppure della forma $\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$, dove $x_i \in X, \epsilon_i = \pm 1$, per ogni $i = 1, \dots, n$, e x e x^{-1} non sono adiacenti.*

Definizione 1.17. *Una **sottoparola** di $\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$ è o la parola vuota oppure una parola della forma $v = x_i^{\epsilon_i} \dots x_j^{\epsilon_j}$, dove $1 \leq i \leq j \leq n$.*

Una parola v è una sottoparola di ω se esistono due parole ω' e ω'' (anche vuote) tali che $\omega = \omega' v \omega''$. Una parola non vuota ω è ridotta se non contiene sottoparole della forma $x^{\epsilon} x^{-\epsilon}$ oppure 1.

Date due parole $\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$ e $\zeta = y_1^{\delta_1} y_2^{\delta_2} \dots y_m^{\delta_m}$, definiamo il risultato della loro *giustapposizione* come la parola

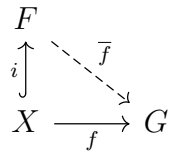
$$\omega\zeta = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n} y_1^{\delta_1} y_2^{\delta_2} \dots y_m^{\delta_m}.$$

Se ω e ζ sono ridotte, la parola risultante $\omega\zeta$ non è detto che sia ridotta.

Denotiamo con F l'insieme delle parole ridotte su X e definiamo su F un'operazione binaria $* : F \times F \to F$, che ad ogni coppia (ω, ζ) in F associa la parola $\omega\zeta$, dopo che è stata "ridotta" cancellando gli elementi x e x^{-1} adiacenti.

Teorema 1.18. *Sia X un insieme non vuoto; l'insieme F delle parole ridotte su X è un gruppo rispetto l'operazione binaria $*$ e con elemento neutro la parola vuota 1. F è detto* ***gruppo libero sull'insieme*** *X.*

Teorema 1.19. *Sia F il gruppo libero sull'insieme X e sia $i : X \hookrightarrow F$ la mappa di inclusione. Dato un generico gruppo G e una mappa $f : X \to G$ tra insiemi, allora esiste un unico omomorfismo di gruppi $\overline{f} : F \to G$ tale che $\overline{f} \circ i = f$.*



Corollario 1.20. *Ogni gruppo G è immagine di un gruppo libero attraverso un omomorfismo di gruppi.*

Dimostrazione. Sia X un insieme di generatori di G e sia F un gruppo libero su X. Consideriamo la mappa di inclusione $\hat{i} : X \to G$; per il teorema precedente, questa induce un omomorfismo $\overline{f} : F \to G$ tale che $x \mapsto x \in G$. Inoltre, poiché $G = \langle X \rangle$, $\overline{f}$ è un epimorfismo, cioè un omomorfismo suriettivo. □

Ricordiamo il seguente risultato:

Teorema 1.21 (**Primo teorema di isomorfismo**). *Siano G e H due gruppi e sia $f : G \to H$ un omomorfismo di gruppi; allora f induce un isomorfismo $G/Kerf \simeq Imf$.*

Sia G un gruppo, X l'insieme dei generatori di G, cioè $G = \langle X \rangle$ e sia F il gruppo libero su X; consideriamo l'epimorfismo di gruppi $\overline{f} : F \to G$ del Corollario 1.20. Dal primo teorema di isomorfismo segue che $G \simeq F/N$, dove $N = Ker\overline{f}$.

Sia $\omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n} \in F$ un generatore di N; allora per l'epimorfismo $\overline{f} : F \to G$ si ha $\omega \mapsto \omega = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n} = e_G \in G$. L'equazione $x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n} = e_G$ in G è detta *relazione* sui generatori x_i.

Viceversa, dati un insieme X ed un insieme Y di parole ridotte su X; esiste un gruppo G tale che G è generato da X e le relazioni della forma $\omega = e_G$, dove $\omega \in Y$, sono ben definite. Si può dimostrare che G è il gruppo quoziente F/N, dove F è il gruppo libero su X e N è il sottogruppo normale di F generato da Y. Per la dimostrazione si veda [4, Cap. I, 9]. Formalizziamo i risultati appena presentati attraverso la seguente definizione:

Definizione 1.22. *Sia X un insieme e Y un insieme di parole ridotte su X. Un gruppo G si dice* ***definito dai generatori*** *$x \in X$* ***e dalle relazioni*** *$\omega = e_G$, dove $\omega \in Y$ se $G \simeq F/N$, dove F è il gruppo libero su X e N è il sottogruppo normale di F generato da Y. Chiamiamo $(X \mid Y)$ una* ***presentazione*** *di G.*

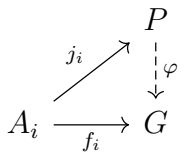
Abbiamo, quindi, introdotto un metodo per descrivere totalmente un gruppo G mediante un insieme X di generatori ed un insieme R di relazioni sui generatori.

Definizione 1.23. *Un gruppo G è* ***finitamente presentato*** *se ha una presentazione con un numero finito di generatori e relazioni.*

Osservazione 1.24. Il gruppo libero F sull'insieme X è detto "libero" perché è il gruppo definito dai generatori $x \in X$ e da nessuna relazione.

Definiamo infine il prodotto libero di una famiglia di gruppi;

Definizione 1.25. *Sia $\{A_i : i \in I\}$ una collezione di gruppi. Un* ***prodotto libero*** *degli A_i è un gruppo P e una famiglia di omomorfismi $j_i : A_i \to P$ tale che, per ogni gruppo G e per ogni famiglia di omomorfismi $f_i : A_i \to G$, esiste un unico omomorfismo di gruppi $\varphi : P \to G$ tale che $\varphi j_i = f_i$, per ogni i.*



Si può dimostrare che il prodotto libero è unico a meno di isomorfismo; introduciamo, quindi, la notazione $P = \underset{i \in I}{*} A_i$ per indicare il prodotto libero della famiglia $\{A_i : i \in I\}$.

Teorema 1.26 (**Forma normale nel prodotto libero**)**.** *Sia $g \in \underset{i \in I}{*} A_i$ e $g \neq 1$, allora g ha un'unica fattorizzazione $g = a_1 \dots a_n$, dove ogni lettera è diversa dall'elemento neutro e lettere adiacenti appartengono a distinti A_i.*

Introduciamo la presentazione del prodotto libero di una famiglia di gruppi, coerentemente alla Definizione 1.22:

Teorema 1.27. *Sia $\{A_i : i \in I\}$ una collezione di gruppi e sia $(X_i \mid \Delta_i)$ una presentazione di A_i, dove gli insiemi $\{X_i : i \in I\}$ sono a due a due disgiunti. Allora una presentazione di $\underset{i \in I}{*} A_i$ è $(\bigcup X_i \mid \bigcup \Delta_i)$.*

1.3 Estensioni HNN e lemma di Britton

Il presente paragrafo contiene solo alcuni dei risultati della teoria sulle estensioni HNN; per una trattazione più estesa, il lettore faccia riferimento a [9, Cap. 11].

Sia G un gruppo con presentazione $(X \mid \Delta)$, denotiamo con $G^\Omega = (G : Y \mid \Delta')$ il gruppo avente presentazione $(X \cup Y \mid \Delta \cup \Delta')$, dove X e Y sono insiemi disgiunti. Osserviamo che se $Y = \varnothing$, allora stiamo solo aggiungendo le relazioni Δ' a G.

Definizione 1.28. *Si considerino un gruppo G, due sottogruppi isomorfi A e B e un isomorfismo $\varphi : A \to B$. Il gruppo avente presentazione $(G; p \mid p^{-1}ap = \varphi(a),\ \text{per ogni } a \in A)$ è detto un'**estensione HNN** di G e si denota con $G\,\Omega_\varphi\, A$. Il gruppo G è detto **base** e il generatore p è detto **lettera stabile** di $G\,\Omega_\varphi\, A$.*

Generalizziamo la definizione di estensione HNN al caso in cui le lettere stabili siano più di una.

Definizione 1.29. *Sia E un gruppo avente una presentazione $(X \mid \Delta)$ e sia $\{p_i : i \in I\}$ un insieme non vuoto disgiunto da X. Consideriamo un insieme J di indici e supponiamo che per ogni $i \in I$ vi sia una famiglia $\{a_{ij}, b_{ij} : j \in J\}$ di coppie di parole in X. Diciamo che il gruppo E^Ω ha una presentazione con **base** E e **lettere stabili** $\{p_i : i \in I\}$ se $E^\Omega = (E; p_i, i \in I \mid p_i^{-1}a_{ij}p_i = b_{ij}\ \text{per ogni } i, j)$.*

Per ogni i, definiamo due sottogruppi di E: $A_i = \langle a_{ij} : j \in J\rangle$ e $B_i = \langle b_{ij} : j \in J\rangle$.

Definizione 1.30. *Un gruppo E^Ω avente una presentazione con base E e lettere stabili $\{p_i : i \in I\}$ è un'**estensione HNN** se, per ogni i, esiste un isomorfismo $\varphi_i : A_i \to B_i$ tale che $\varphi_i(a_{ij}) = b_{ij}$ per ogni $j \in J$.*

Osservazione 1.31. Sia E un gruppo e F un gruppo libero con base $\{p_i : i \in I\}$, allora il prodotto libero $E * F$ è un'estensione HNN di E con lettere stabili $\{p_i : i \in I\}$. In questo caso si ha $a_{ij} = b_{ij} = 1$ per ogni i, j.

Osservazione 1.32. Tornerà utile il seguente risultato, che però non verrà formalizzato: se E^Ω è un'estensione HNN con base $E = (X \mid \Delta)$ e lettere stabili $\{p_1, \ldots, p_t\}$, allora E è un sottogruppo di E^Ω. Per una formulazione rigorosa si faccia riferimento al Teorema 11.78 [9, Cap. 11].

Sia E^Ω un'estensione HNN con base $E = (X \mid \Delta)$ e lettere stabili $\{p_i : i \in I\}$; consideriamo una parola della forma $p_i^e g p_i^{-e}$, dove g è una parola in X tale che $p_i^e g p_i^{-e}$ appartiene ad $A_i \leq A$ se $e = -1$ oppure a $B_i \leq B$ se $e = +1$. Per indicare una parola della forma precedente adottiamo il termine inglese *pinch*.

Teorema 1.33 (**Lemma di Britton**). *Sia E^Ω un'estensione HNN con base $E = (X \mid \Delta)$ e lettere stabili $\{p_i : i \in I\}$. Se ω è una parola che contiene almeno una lettera stabile ed è tale che $\omega = 1$ in E^Ω, allora ω contiene un pinch come sottoparola.*

Definizione 1.34. *Sia E^Ω un'estensione HNN con base $E = (X \mid \Delta)$ e lettere stabili $\{p_1, \dots, p_t\}$. Una parola ω in $X \cup \{p_1, \dots, p_t\}$ si dice p_i-**ridotta** per un qualche i se non contiene sottoparole della forma $p_i^e v p_i^{-e}$.*

Osservazione 1.35. In particolare, le parole p_i-ridotte non contengono sottoparole della forma $p_i^e p_i^{-e}$.

Corollario 1.36. *Sia E^Ω un'estensione HNN con base $E = (X \mid \Delta)$ e lettere stabili $\{p_1, \dots, p_t\}$. Supponiamo che $\alpha \equiv \gamma_0 p_i^{e_1} \gamma_1 \dots p_i^{e_m} \gamma_m$ e $\beta \equiv \delta_0 p_i^{f_1} \delta_1 \dots p_i^{f_n} \delta_n$ siano parole p_i-ridotte per un qualche i, con $e_j, f_k = \pm 1$; inoltre, nessuna delle parole (anche vuote) γ_j o δ_k contiene p_i. Se $\alpha = \beta$ in E^Ω, allora $m = n, (e_1, \dots, e_m) = (f_1, \dots, f_n)$ e la parola $p_i^{e_m} \gamma_m \delta_n^{-1} p_i^{-f_n}$ è un pinch.*

Infine, il seguente teorema caratterizza la forma normale in un'estensione HNN nel caso ci sia una sola parola stabile; tale risultato si può generalizzare anche al caso di più parole stabili.

Teorema 1.37 (**Forma normale in un'estensione HNN**). *Sia E^Ω un'estensione HNN con base $E = (X \mid \Delta)$ e lettera stabile p. Allora ogni parola ω in $\{X, p\}$ è uguale in E^Ω a una parola p-ridotta della forma $\omega_0 p^{e_1} \omega_1 \dots p^{e_n} \omega_n$, dove la lunghezza n e la sequenza $(e_1, \dots, e_n)$ degli esponenti sono univocamente determinate da ω.*

Capitolo 2

Algoritmi di Markov

Il concetto di algoritmo è stato a lungo definito solo in maniera intuitiva; in particolare, A.A. Markov si riferisce ad esso come ad una sequenza di istruzioni esatte che definiscono un processo computazionale e che dai vari dati iniziali conducono al risultato desiderato [6]. Due esempi classici sono l'addizione di due numeri interi o l'algoritmo di Euclide per il calcolo del massimo comune divisore. Tale nozione è strettamente legata alla definizione, anch'essa intuitiva, di funzione effettivamente calcolabile, cioè una funzione $f(x_1, \ldots, x_n)$ per cui esiste un procedimento meccanico, dunque un algoritmo, che riesca a calcolarne il valore a partire da n argomenti iniziali [7, Cap. 5, § 5.1]. Nel presente capitolo illustriamo i risultati forniti da Markov nel tentativo di formalizzare i concetti precedenti. La seguente esposizione fa riferimento principalmente a [7, Cap. 5, § 5.1]; l'utilizzo di fonti diverse sarà esplicitato nel corso della trattazione.

2.1 Costruzione formale degli algoritmi di Markov

Fissiamo le seguenti notazioni: indichiamo i simboli di un alfabeto A con $S_0, S_1, S_2, \ldots$ e la parola vuota con il simbolo Λ. [1] Per i nostri scopi è sufficiente considerare alfabeti costituiti da un numero finito di simboli. Date P_1, P_2, P_3 parole in A; valgono le seguenti due proprietà: $P_1\Lambda = \Lambda P_1 = P_1$ e $(P_1P_2)P_3 = P_1(P_2P_3)$.

Con *algoritmo in un alfabeto* A si intende una lista di istruzioni esatte che agiscono su parole in A. Se P è una parola in A, si dice che l'algoritmo $\mathfrak{A}$ è *applicabile* a P se questo trasforma P in una parola in A, che denotiamo con $\mathfrak{A}(P)$. Un alfabeto A è un'*estensione* di un alfabeto B se e solo se $B \subseteq A$, cioè ogni parola di B è anche una parola di A; in

[1] Per le definizioni di "alfabeto", "parola", "parola vuota" e "giustapposizione di due parole" si veda il Paragrafo 1.2.

particolare, con *algoritmo sopra un alfabeto* A si intende un algoritmo in un'estensione B di A.

Gli *algoritmi di Markov* o *algoritmi normali* fondano la propria costruzione su un'operazione elementare, cioè la sostituzione di una parola a un'altra. Vediamone la costruzione rigorosa.

Definizione 2.1. *Siano P e Q due parole (anche vuote) in un alfabeto A e assumiamo che " $\to$ " e il punto " $\cdot$ " non siano simboli di A; allora $P \to Q$ e $P \to \cdot Q$ sono dette rispettivamente* ***produzione semplice*** *e* ***produzione terminale*** *nell'alfabeto A.*

Definizione 2.2. *Uno* ***schema di algoritmo*** *è costituito da una lista finita di produzioni nell'alfabeto A, cioè:*

$$\begin{aligned} P_1 &\to (\cdot)Q_1 \\ P_2 &\to (\cdot)Q_2 \\ &\vdots \\ P_r &\to (\cdot)Q_r, \end{aligned}$$

dove la scrittura $P \to (\cdot)Q$ sta ad indicare che o $P \to Q$ oppure $P \to \cdot Q$.

Tale schema determina l'algoritmo normale $\mathfrak{A}$ in A.

Diciamo che una parola T *occorre* in Q se esistono parole U, V (anche vuote) tali che $Q = UTV$. Ora, data una parola P in A:

1. scriviamo $\mathfrak{A} : P \sqsupset$ se nessuna delle parole $P_1, \ldots, P_r$ occorre in P;
2. altrimenti, se m è il più piccolo intero con $1 \leq m \leq r$ tale che P_m occorre in P, sostituiamo l'occorrenza più a sinistra di P_m in P con Q_m (coerentemente a quanto dettato dalla produzione $P_m \to (\cdot)Q_m$ nello schema di algoritmo); chiamiamo R la parola ottenuta in seguito a tale operazione. In tal caso scriviamo:
   (a) $\mathfrak{A} : P \vdash R$ se la produzione è semplice, cioè della forma $P_m \to Q_m$; diciamo, dunque, che $\mathfrak{A}$ *trasforma semplicemente* P in R;
   (b) $\mathfrak{A} : P \vdash \cdot R$ se la produzione è terminale, cioè della forma $P_m \to \cdot Q_m$; diciamo, dunque, che $\mathfrak{A}$ *trasforma in modo terminale* P in R.

Con la scrittura $\mathfrak{A} : P \models R$ indichiamo che esiste una sequenza $R_0, R_1, \ldots, R_k$ tale che:

$$\begin{aligned} &P = R_0; \\ &R = R_k; \\ &\mathfrak{A} : R_j \vdash R_{j+1} \quad \text{se } 0 \leq j \leq k-2; \end{aligned}$$

ed infine

$$\mathfrak{A} : R_{k-1} \vdash R_k,$$

oppure

$$\mathfrak{A} : R_{k-1} \vdash \cdot R_k.$$

In quest'ultimo caso scriviamo $\mathfrak{A} : P \models \cdot R$. Poniamo $\mathfrak{A}(P) = R$ se e solo se o $\mathfrak{A} : P \models \cdot R$ oppure $\mathfrak{A} : P \models R$ e $\mathfrak{A} : R \sqsupset$.

Descriviamo in maniera esplicita il comportamento dell'algoritmo $\mathfrak{A}$ appena definito: data una parola P in un dato linguaggio A, si cerca la prima produzione $P_m \to (\cdot)Q_m$ nello schema tale che P_m occorra in P e si sostituisce Q_m all'occorrenza più a sinistra di P_m in P; sia R_1 la parola ottenuta in seguito a questa trasformazione. Se la produzione è terminale, cioè della forma $P_m \to \cdot Q_m$, l'algoritmo termina e il valore prodotto è R_1. Altrimenti, si applica il procedimento appena visto ad R_1 e alle parole risultanti successive. Se si ottiene una parola R_i tale che $\mathfrak{A} : R_i \sqsupset$, cioè nessun P_m occorre in R_i, allora il processo ha termine e il valore ottenuto è R_i. Il procedimento può anche non terminare; in tal caso si dice che $\mathfrak{A}$ *non è applicabile* alla parola P.

2.2 Esempi di algoritmi

Mostriamo ora alcuni esempi di algoritmi normali:

1. (a) Data una parola Q in un alfabeto A; consideriamo l'algoritmo normale $\mathfrak{A}$ definito dalla sola produzione terminale:

$$\Lambda \to \cdot Q. \tag{2.1}$$

   La sostituzione espressa dallo schema di algoritmo è sempre applicabile, infatti ogni parola P in A può essere scritta come $P = \Lambda P$; è inoltre chiaro che il risultato della sostituzione sia la parola QP. Coerentemente alle notazioni introdotte in precedenza possiamo scrivere:

$$\mathfrak{A} : P \vdash \cdot QP,$$
$$\mathfrak{A}(P) = QP.$$

   Tale algoritmo prende il nome di *algoritmo normale per la concatenazione della parola Q a sinistra* .

   (b) Cosa succede se omettiamo il punto nella produzione (2.1) ?

   È immediato osservare che una parola P in A diventa la parola QP, che a sua volta viene trasformata in QQP e così all'infinito. Tale procedimento non

ha fine; questo è un esempio di algoritmo non applicabile a nessuna parola nell'alfabeto A. Si veda [6, Cap. II, § 4.2, 4.3].

2. Sia A un alfabeto che non contiene il simbolo α. Consideriamo lo schema d'algoritmo scritto in forma abbreviata:

$$\alpha\xi \to \cdot\,\Lambda \qquad (\xi \in A) \tag{2.2}$$

$$\alpha \to \cdot\,\Lambda \tag{2.3}$$

$$\Lambda \to \alpha. \tag{2.4}$$

Questo definisce un algoritmo normale $\mathfrak{A}$ in $A \cup \{\alpha\}$ tale che

$$\mathfrak{A}(\xi P) = P \qquad (\xi \in A),$$
$$\mathfrak{A}(\Lambda) = \Lambda,$$

dove P è una parola in A. L'algoritmo $\mathfrak{A}$ cancella la prima lettera di ogni parola non vuota in A e lascia invariata la parola vuota, mostriamolo esplicitamente: se P è una parola non vuota in A e ξ un simbolo in A, per la produzione (2.4) si ha $\xi P = \Lambda\xi P \vdash \alpha\xi P$. Dalla produzione (2.2) segue che $\mathfrak{A} : \xi P \models \cdot\, P$. Consideriamo, infine, la parola vuota Λ; per le produzioni (2.4) e (2.3) si ha $\Lambda \vdash \alpha \vdash \cdot\,\Lambda$, dunque $\mathfrak{A} : \Lambda \models \cdot\,\Lambda$. Si veda [6, Cap. II, § 4.10].

3. Sia A l'alfabeto $\{a_0, a_1, \ldots, a_n\}$ e sia $B = A \cup \{\alpha, \beta\}$ un'estensione di A. Data una parola $P = a_{j_0}a_{j_1}\ldots a_{j_k}$; definiamo l'inversa $\widetilde{P} = a_{j_k}\ldots a_{j_1}a_{j_0}$. Consideriamo il seguente schema d'algoritmo in B:

$$\alpha\alpha \to \beta \tag{2.5}$$

$$\beta\xi \to \xi\beta \qquad (\xi \in A) \tag{2.6}$$

$$\beta\alpha \to \beta \tag{2.7}$$

$$\beta \to \cdot\Lambda \tag{2.8}$$

$$\alpha\eta\xi \to \xi\alpha\eta \qquad (\xi, \eta \in A) \tag{2.9}$$

$$\Lambda \to \alpha. \tag{2.10}$$

Questo definisce un algoritmo normale $\mathfrak{A}$ sopra A che inverte ogni parola di A, cioè $\mathfrak{A}(P) = \widetilde{P}$. Sia infatti $P = a_{j_0}a_{j_1}\ldots a_{j_k}$ una qualsiasi parola in A; allora per la produzione (2.10) $\mathfrak{A} : P \vdash \alpha P$. Per la produzione (2.9) si ha: $\alpha P \vdash a_{j_1}\alpha a_{j_0}\ldots a_{j_k} \vdash a_{j_1}a_{j_2}\alpha a_{j_0}a_{j_3}\ldots a_{j_k} \cdots \vdash a_{j_1}a_{j_2}\ldots a_{j_k}\alpha a_{j_0}$. Perciò $\mathfrak{A} : P \models a_{j_1}a_{j_2}\ldots a_{j_k}\alpha a_{j_0}$. Per la produzione (2.10) $\mathfrak{A} : P \models \alpha a_{j_1}a_{j_2}\ldots a_{j_k}\alpha a_{j_0}$. Applicando la produzione (2.9) ripetute volte si ottiene $\mathfrak{A} : P \models a_{j_2}a_{j_3}\ldots a_{j_k}\alpha a_{j_1}\alpha a_{j_0}$. Ripetendo questo procedimento risulta $\mathfrak{A} : P \models \alpha a_{j_k}\alpha a_{j_{k-1}}\alpha\ldots\alpha a_{j_1}\alpha a_{j_0}$; da cui $\alpha a_{j_k}\alpha a_{j_{k-1}}\alpha\ldots\alpha a_{j_1}\alpha a_{j_0} \overset{(2.10)}{\vdash}$

$\alpha\alpha a_{j_k}\alpha a_{j_{k-1}}\alpha\ldots\alpha a_{j_1}\alpha a_{j_0} \overset{(2.5)}{\vdash} \beta a_{j_k}\alpha a_{j_{k-1}}\alpha\ldots\alpha a_{j_1}\alpha a_{j_0}$. Applicando le produzioni (2.6), (2.7) ed infine (2.8) si ottiene $\mathfrak{A} : P \models\cdot \widetilde{P}$. Si veda [7, Cap. 5, § 5.1].

4. Consideriamo l'alfabeto $A = \{|\}$ e identifichiamo i numeri naturali come parole in A, ad esempio: $0 = \Lambda$, $1 = |$, $2 = ||$ e così via. È possibile costruire degli algoritmi normali sopra A che, data in input una coppia di numeri naturali, restituiscono, ad esempio, il massimo comune divisore oppure il prodotto dei due numeri. Per una trattazione più accurata dei seguenti algoritmi si veda [6, Cap. II, § 4.18].

   Presentiamo di seguito l'algoritmo normale $\mathfrak{A}$ nell'alfabeto $C = \{|, *\}$ che, dati due numeri naturali, restituisce il valore assoluto della loro differenza. Siano N ed M due numeri naturali identificati come parole in C; consideriamo il seguente schema d'algoritmo:

$$| * | \to * \tag{2.11}$$

$$* \to \Lambda. \tag{2.12}$$

   Mostriamo che $\mathfrak{A}(N * M) = |N - M|$: è evidente che se $N \geq M$ per la produzione (2.11) si ha $\mathfrak{A} : N{*}M \vdash (N{-}1){*}(M{-}1) \vdash (N{-}2){*}(M{-}2) \vdash \cdots \vdash (N{-}M){*}$. Infine, applicando (2.12) il simbolo "$*$" viene cancellato; dunque $\mathfrak{A} : N * M \models (N - M)$ e $\mathfrak{A} : (N{-}M)\sqsupset$. Similmente, se $N < M$ si ha $\mathfrak{A} : N{*}M \models (M{-}N)$ e $\mathfrak{A} : (M{-}N)\sqsupset$; dunque, in entrambi i casi vale $\mathfrak{A} : N * M \models |N - M|$ e $\mathfrak{A} : |N - M|\sqsupset$, che è quanto volevamo dimostrare. Si veda [6, Cap. II, § 4.17].

2.3 Composizione di algoritmi normali

Esistono diversi metodi di combinazione di algoritmi che, a partire da due o più algoritmi dati, permettono di costruirne di nuovi; uno di questi metodi è la composizione. In particolare, si può dimostrare che se gli algoritmi iniziali sono normali, la loro composizione sarà ancora normale. Nel presente paragrafo proveremo quanto appena enunciato.

Come prima cosa definiamo alcuni concetti preliminari:

Definizione 2.3. *Siano $\mathfrak{A}, \mathfrak{B}$ due algoritmi sopra un alfabeto A e P una parola in A; scriviamo $\mathfrak{A}(P) \approx \mathfrak{B}(P)$ se e solo se vale una delle due condizioni:*

- *$\mathfrak{A}$ e $\mathfrak{B}$ sono entrambi applicabili a P e $\mathfrak{A}(P) = \mathfrak{B}(P)$;*
- *né $\mathfrak{A}$ né $\mathfrak{B}$ è applicabile a P.*

Definizione 2.4. *Siano* $\mathfrak{A}$ *e* $\mathfrak{B}$ *due algoritmi sopra un alfabeto* A*, diciamo che* $\mathfrak{A}$ *e* $\mathfrak{B}$ *sono* ***pienamente equivalenti*** *rispetto ad* A *se e solo se* $\mathfrak{A}(P) \approx \mathfrak{B}(P)$ *per ogni parola* P *in* A*.*

Definiamo, inoltre, i concetti di algoritmo chiuso e chiusura di un algoritmo:

Definizione 2.5. *Un algoritmo normale* $\mathfrak{A}$ *in un alfabeto* A *si dice* ***chiuso*** *se il suo schema contiene una produzione della forma* $\Lambda \to \cdot Q$*, dove* Q *è una qualsiasi parola in* A*.*

Osservazione 2.6. La computazione di un algoritmo chiuso termina sempre per applicazione di una produzione terminale, cioè $\mathfrak{A}(P) = Q$ se e solo se $\mathfrak{A} : P \models \cdot Q$. Non è possibile che si verifichi il caso in cui $\mathfrak{A} : P \models Q$ e $\mathfrak{A} : Q \sqsupset$, infatti una produzione del tipo $\Lambda \to \cdot Q$ è applicabile ad ogni parola in A

Definizione 2.7. *Dato l'algoritmo normale* $\mathfrak{A}$*; definiamo la* ***chiusura*** *di* $\mathfrak{A}$ *come l'algoritmo dotato dello schema di* $\mathfrak{A}$ *con l'aggiunta della nuova produzione* $\Lambda \to \cdot \Lambda$*. Denotiamo con* $\mathfrak{A}\cdot$ *l'algoritmo appena definito.*

Osservazione 2.8. Segue direttamente dalla definizione precedente che la chiusura di $\mathfrak{A}$ è chiusa.

Proposizione 2.9 (**Composizione di algoritmi normali**)**.** *Siano* $\mathfrak{A}$ *e* $\mathfrak{B}$ *due algoritmi normali in un alfabeto* A*; allora è possibile costruire un algoritmo normale* $\mathfrak{C}$ *sopra* A *tale che* $\mathfrak{C}(P) \approx \mathfrak{B}(\mathfrak{A}(P))$ *per qualsiasi parola* P *in* A*.* $\mathfrak{C}$ *è chiamato la* ***composizione*** *di* $\mathfrak{A}$ *e* $\mathfrak{B}$ *e si denota con* $\mathfrak{B} \circ \mathfrak{A}$*.*

Dimostrazione. Costruiamo uno schema d'algoritmo per $\mathfrak{C}$: definiamo un nuovo alfabeto $\overline{A}$ costituito da tutti i correlati dei simboli in A, cioè dai simboli della forma $\overline{b}$, dove b è un simbolo in A. Siano α e β due simboli non in $A \cup \overline{A}$. Sia $\mathscr{S}_{\mathfrak{A}}$ lo schema d'algoritmo di $\mathfrak{A}\cdot$ in cui il punto di ogni produzione terminale è sostituito da α. Sia $\mathscr{S}_{\mathfrak{B}}$ lo schema di $\mathfrak{B}\cdot$ con le seguenti modifiche:

1. ogni simbolo è sostituito dal suo correlato;
2. ogni punto terminale è rimpiazzato da β;
3. le produzioni $\Lambda \to Q$ assumono la forma $\alpha \to \alpha Q$;
4. le produzioni $\Lambda \to \cdot Q$ sono sostituite da $\alpha \to \alpha\beta Q$.

Consideriamo la seguente lista di produzioni:

$$\begin{aligned}
a\alpha &\to \alpha a \qquad (a \in A)\\
\alpha a &\to \alpha \overline{a} \qquad (a \in A)\\
\overline{a}b &\to \overline{a}\overline{b} \qquad (a, b \in A)\\
\overline{a}\beta &\to \beta \overline{a} \qquad (a \in A)\\
\beta \overline{a} &\to \beta a \qquad (a \in A)\\
a\overline{b} &\to ab \qquad (a, b \in A)\\
\alpha\beta &\to \cdot\, \Lambda\\
&\mathscr{S}_{\mathfrak{B}}\\
&\mathscr{S}_{\mathfrak{A}}.
\end{aligned}$$

Questo rappresenta lo schema dell'algoritmo normale $\mathfrak{C}$. □

Osservazione 2.10. È possibile estendere l'operazione di composizione a più di due algoritmi, infatti: dato $n \geq 2$, si dimostra per induzione su n che la composizione degli algoritmi normali $\mathfrak{A}_1, \dots \mathfrak{A}_n$ è un algoritmo normale sopra l'unione degli alfabeti di partenza. In generale, con $\mathfrak{A}_n \circ \cdots \circ \mathfrak{A}_1$ si intende $\mathfrak{A}_n \circ (\cdots \circ (\mathfrak{A}_3 \circ (\mathfrak{A}_2 \circ \mathfrak{A}_1)) \dots)$.

Per una dimostrazione più dettagliata dei risultati precedenti si veda [6, Cap. III, § 3].

2.4 Funzioni calcolabili nel senso di Markov

Nel presente paragrafo presentiamo una particolare classe di funzioni numeriche con lo scopo di formalizzare la nozione intuitiva di funzione effettivamente calcolabile.

Partiamo definendo il concetto di numerale: indichiamo il simbolo S_1 con 1 e consideriamo l'alfabeto $M = \{1, *\}$. Per ogni numero naturale n definiamo il *numerale* $\overline{n}$ induttivamente come segue:

$$\begin{cases} \overline{0} = 1 \\ \overline{n+1} = \overline{n}1. \end{cases}$$

Ad esempio: $\overline{1} = 11$, $\overline{2} = 111$ e similmente per ogni numero naturale.

La parola $\overline{n}_1 * \overline{n}_2 * \cdots * \overline{n}_k$ in M sta ad indicare la k-upla $(n_1, n_2, \dots, n_k)$; denotiamo tale parola con $\overline{(n_1, n_2, \dots, n_k)}$. Per esempio $\overline{(3, 1, 2)}$ è $1111 * 11 * 111$.

Definizione 2.11. *Una funzione $f : \mathbb{N}^k \to \mathbb{N}$, $k \geq 1$ si dice **parziale** se il suo dominio è un sottoinsieme di $\mathbb{N}^k$.*

Dunque, sia $\varphi : \mathbb{N}^k \to \mathbb{N}$, $k \geq 1$ una funzione numerica parziale effettivamente calcolabile; denotiamo con $\mathfrak{B}_\varphi$ il corrispondente algoritmo in M, ossia $\mathfrak{B}_\varphi(\overline{(n_1, \ldots, n_k)}) = \overline{\varphi(n_1, \ldots, n_k)}$, sotto l'ipotesi che entrambi i membri dell'equazione siano definiti. Si richiede, inoltre, che $\mathfrak{B}_\varphi$ sia applicabile solo a k-uple della forma $\overline{(n_1, \ldots, n_k)}$.

Definizione 2.12. *Se φ soddisfa le ipotesi precedenti, si dice che la funzione φ è **parzialmente calcolabile nel senso di Markov** (o parzialmente Markov-calcolabile) se e solo se esiste un algoritmo normale $\mathfrak{A}$ sopra M che è pienamente equivalente a $\mathfrak{B}_\varphi$ rispetto a M.*

Definizione 2.13. *Una funzione $f : \mathbb{N}^k \to \mathbb{N}$, $k \geq 1$ si dice **totale** se è definita per tutte le k-uple di numeri naturali.*

Definizione 2.14. *Se la funzione precedente φ è totale e parzialmente Markov-calcolabile, si dice che φ è **calcolabile nel senso di Markov** (o Markov-calcolabile).*

Dunque, abbiamo identificato il concetto intuitivo di calcolabilità effettiva con la nozione rigorosa di calcolabilità nel senso di Markov. Possiamo pensare alle funzioni parzialmente Markov-calcolabili come a quelle funzioni per cui esiste un algoritmo in grado di calcolarne il valore, quando viene data in input una k-upla di numeri naturali appartenente al dominio della funzione; al contrario, quando la funzione non è definita sulla k-upla data, la computazione procede all'infinito, nel vano tentativo di produrre un output. In generale, dato un elemento in input, non è possibile sapere a priori se l'esecuzione avrà termine oppure no [2, Cap. 1].

Presentiamo i seguenti esempi di funzioni Markov-calcolabili:

1. Consideriamo la funzione di successore $\varphi(\overline{n}) = \overline{n+1}$, definita per ogni n numero naturale. Sia α un simbolo che non appartiene all'alfabeto $M = \{1, *\}$. Lo schema

$$
\begin{aligned}
* &\to * \\
\alpha 1 &\to \cdot 11 \\
\Lambda &\to \alpha
\end{aligned}
$$

   definisce un algoritmo normale $\mathfrak{A}$ sopra M applicabile solo ai numerali in M e tale che $\mathfrak{A}(\overline{n}) = \overline{n+1}$, per ogni n in $\mathbb{N}$, infatti: $\overline{n} = 1^{n+1} = \Lambda 1^{n+1} \to \alpha 1^{n+1} \to \cdot 11^{n+1} = = \overline{n+1}$.

2. Consideriamo la funzione $\varphi(\overline{n}) = \overline{0}$, definita per ogni n numero naturale. Sia α un simbolo che non appartiene all'alfabeto $M = \{1, *\}$. Lo schema

$$
\begin{aligned}
* &\to * \\
\alpha 11 &\to \alpha 1 \\
\alpha 1 &\to \cdot 1 \\
\Lambda &\to \alpha
\end{aligned}
$$

definisce un algoritmo normale $\mathfrak{A}$ sopra M applicabile solo ai numerali in M e tale che $\mathfrak{A}(\overline{n}) = \overline{0}$, per ogni n in $\mathbb{N}$, infatti: $\overline{n} = 1^{n+1} = \Lambda 1^{n+1} \to \alpha 1^{n+1} = \alpha 11^{n} \to \alpha 1^{n} = \alpha 11^{n-1} \to \alpha 1^{n-1} \to \cdots \to \alpha 1 \to \cdot 1$.

Possiamo, quindi, proseguire con il terzo capitolo, che presenterà un ulteriore formalismo per la calcolabilità: le macchine di Turing.

Capitolo 3

Un altro formalismo per la calcolabilità: le macchine di Turing

Il tentativo di A. Turing di fornire una definizione rigorosa di calcolabilità effettiva si basa sulla costruzione di un modello di calcolo astratto. Per prima cosa egli osserva come un essere umano esegue una computazione, concentrandosi sulle proprietà essenziali del processo di calcolo e tralasciando tutto ciò che non è rilevante; a partire da queste osservazioni definisce un modello di calcolo, che è un'astrazione di un esecutore reale e come tale non risente di limitazioni concrete, come il poter usufruire solo di una quantità limitata di memoria [3], [11]. Tale modello prende il nome di *macchina di Turing*.

3.1 Macchine di Turing

Forniamo una prima descrizione informale: una macchina di Turing è costituita da un nastro potenzialmente infinito, diviso in quadrati, che chiameremo *celle*; in ogni momento della computazione viene utilizzata solo una porzione finita di nastro, ma all'esigenza si possono aggiungere ulteriori quadrati alle due estremità. Fissiamo una testina di lettura, un insieme finito di simboli $\{S_0, S_1, \ldots, S_M\}$ e un insieme finito di stati interni $\{q_0, q_1, \ldots, q_N\}$. Supponiamo che il simbolo S_0 rappresenti uno spazio bianco. In ogni dato momento:

- quasi tutte le celle sono vuote, eccetto un numero finito, in cui può essere contenuto al più un simbolo;
- la macchina si trova in uno degli stati interni;
- la testina di lettura legge (cioè è posizionata sopra) una data cella.

Tutte queste informazioni sono racchiuse in una configurazione (o descrizione istantanea), che rappresenta una vera e propria "istantanea" di un passo di computazione. La configurazione iniziale vede la testina di lettura posta sopra una data cella e con stato interno iniziale q_0. La macchina non funziona con continuità, ma solo in momenti discreti di tempo. Supponiamo di avere una data configurazione in un qualche istante di tempo t; la testina di lettura legge il contenuto della cella e, sulla base della lettura, la macchina compie una delle seguenti azioni:

1. scrive un nuovo simbolo nella cella corrente;
2. si sposta di una cella verso destra;
3. si sposta di una cella verso sinistra;
4. si ferma.

Nei primi tre casi la macchina entra in un nuovo stato interno. L'esecuzione può avere termine oppure può procedere all'infinito; se ha termine, il nastro risultante è detto l'output della macchina applicata al nastro dato.

Vediamo di seguito le definizioni rigorose dei concetti appena presentati [9, Cap. 12].

Definizione 3.1. *Siano* $\{S_0, S_1, S_2, \dots\}$ *e* $\{q_0, q_1, q_2, \dots\}$ *due insiemi infiniti di simboli; una* ***quadrupla*** *è una 4-tupla avente una delle seguenti forme:*

$$q_j S_i S_k q_r,$$
$$q_j S_i R q_r,$$
$$q_j S_i L q_r.$$

Definizione 3.2. *Una* ***macchina di Turing*** T *è un insieme finito di quadruple tale che nessuna coppia abbia gli stessi primi due simboli. L'****alfabeto*** *di* T *è l'insieme* $S = \{S_0, S_1, \dots, S_M\}$ *di tutti i simboli* S_l *che compaiono nelle quadruple; gli* ***stati interni*** *sono i simboli* q_s*.*

I tre tipi di quadruple corrispondono ai primi tre tipi di mosse presentate sopra; ad esempio, la quadrupla $q_j S_i R q_r$ sta ad indicare la seguente istruzione: "quando lo stato corrente è q_j e la testina di lettura legge il simbolo S_i, quest'ultima si sposta sulla cella a destra e la macchina entra nello stato q_r". L'insieme di tutte le istruzioni è detto *struttura iniziale* della macchina e determina il tipo di azione che verrà svolta.

Definizione 3.3. *Una* ***descrizione istantanea*** α *del nastro è una parola positiva della forma* $\alpha = \sigma q_j \tau$*, con* σ *e* τ *parole in* S *e* τ *non vuota.* [1]

Per esempio, la descrizione istantanea $\alpha = S_2S_0q_1S_5S_2S_2$ sta ad indicare che sul nastro vi sono i simboli $S_2S_0S_5S_2S_2$, tutte le restanti celle sono vuote, lo stato corrente è q_1 e la testina di lettura legge il simbolo S_5.

Definizione 3.4. *Sia* T *una macchina di Turing; una coppia ordinata* (α, β) *di descrizioni istantanee è detta un* ***passo di calcolo****, denotato con* $\alpha \xrightarrow{T} \beta$*, se esistono due parole (anche vuote)* σ *e* σ' *tale che valga una delle seguenti condizioni:*

1. $\alpha = \sigma q_j S_i \sigma'$ *e* $\beta = \sigma q_r S_k \sigma'$*, se* $q_j S_i S_k q_r \in T$*;*
2. $\alpha = \sigma q_j S_i S_k \sigma'$ *e* $\beta = \sigma S_i q_r S_k \sigma'$*, se* $q_j S_i R q_r \in T$*;*
3. $\alpha = \sigma q_j S_i$ *e* $\beta = \sigma S_i q_r S_0$*, se* $q_j S_i R q_r \in T$*;*
4. $\alpha = \sigma S_k q_j S_i \sigma'$ *e* $\beta = \sigma q_r S_k S_i \sigma'$*, se* $q_j S_i L q_r \in T$*;*
5. $\alpha = q_j S_i \sigma'$ *e* $\beta = q_r S_0 S_i \sigma'$*, se* $q_j S_i L q_r \in T$*.*

Dunque, α descrive il nastro in un dato momento t: fornisce lo stato q_j di T e il simbolo S_i letto dalla testina di lettura; invece β descrive il nastro al tempo $t+1$, cioè dopo che è stata svolta l'istruzione richiesta: quindi fornisce lo stato successivo di T e il nuovo simbolo letto dalla macchina.

Osservazione 3.5. L'azione successiva di una macchina di Turing è sempre ben definita, infatti nella definizione stessa di macchina è richiesto che nessuna coppia di quadruple abbia gli stessi primi due simboli; dunque, se $\alpha \xrightarrow{T} \beta$ e $\alpha \xrightarrow{T} \gamma$, allora $\beta = \gamma$.

Osservazione 3.6. Osserviamo con maggiore attenzione la condizione (3): questa rappresenta il caso in cui la testina di lettura è posta sull'ultimo simbolo (a destra) del nastro e l'istruzione richieda di spostarsi a destra; ciò è possibile poiché la macchina è in grado, all'esigenza, di aggiungere ulteriori celle vuote, cioè contenenti il simbolo S_0, alle estremità del nastro. Lo stesso vale per la condizione (5), che invece riguarda l'estremità sinistra.

Definizione 3.7. *Una descrizione istantanea* α *è detta* ***terminale*** *se non esiste alcuna descrizione istantanea* β *tale che valga* $\alpha \xrightarrow{T} \beta$*;*

Osserviamo che questo succede quando $q_j S_i$ occorre in α, ma $q_j S_i$ non sono i primi due simboli di una quadrupla di T.

[1] Per la definizione rigorosa di "parola positiva" si veda la Definizione 1.14.

Definizione 3.8. *Data una parola positiva ω nell'alfabeto di T; diciamo che T* ***calcola*** *ω se esiste una sequenza finita di descrizioni istantanee $\alpha_0 = q_0\omega, \alpha_1, \ldots, \alpha_t$ tale che $\alpha_i \xrightarrow{T} \alpha_{i+1}$ per ogni $0 \leq i \leq t-1$ e α_t è terminale.* [2]

Dunque, supponiamo che una parola ω sia stampata sul nastro, che lo stato corrente sia lo stato iniziale q_0 e che la testina di lettura sia posizionata sopra la cella contenente il primo simbolo di ω; consideriamo, inoltre, la sequenza $q_0\omega \xrightarrow{T} \alpha_1 \xrightarrow{T} \alpha_2 \xrightarrow{T} \cdots$. Se T calcola ω tale sequenza è finita; altrimenti è infinita e l'esecuzione di T non ha termine.

3.2 Esempi

Mostriamo ora alcuni esempi di macchine di Turing:

1. Consideriamo la seguente macchina di Turing T definita sull'alfabeto $\{S_0, 0, 1\}$:

$$q_0 S_0 0 q_0$$
$$q_0 0 R q_1$$
$$q_1 S_0 1 q_1$$
$$q_1 1 R q_0.$$

   È immediato osservare che T scrive indefinitamente su nastro vuoto una sequenza di "0" ed "1". [3]

2. Sia T la macchina di Turing definita dalle seguenti quadruple:

$$q_0 S_0 R q_0$$
$$q_0 S_4 R q_0$$
$$q_0 S_5 R q_0$$
$$q_0 S_3 S_5 q_1$$
$$q_1 S_5 L q_2$$
$$q_2 S_0 S_5 q_2$$
$$q_2 S_4 S_5 q_2$$
$$q_2 S_3 S_5 q_2.$$

[2] Si è scelto di mantenere l'espressione "T calcola ω", introdotta in [9, Cap. 12], per garantire coerenza terminologica con la fonte e con la trattazione successiva.

[3] Questo è il primo esempio di macchina di Turing fornito da Turing stesso nel suo articolo "On Computable Numbers, with an Application to the Entscheidungsproblem" (1936) [11].

L'alfabeto di T è l'insieme $\{S_0, S_3, S_4, S_5\}$.

Supponiamo che la descrizione istantanea iniziale sia $S_4q_0S_0S_4S_3$, allora si ha: $S_4q_0S_0S_4S_3 \xrightarrow{T} S_4S_0q_0S_4S_3 \xrightarrow{T} S_4S_0S_4q_0S_3 \xrightarrow{T} S_4S_0S_4q_1S_5 \xrightarrow{T} S_4S_0q_2S_4S_5 \xrightarrow{T}$ $\xrightarrow{T} S_4S_0q_2S_5S_5$. Dunque, la macchina T si sposta a destra fino ad individuare il simbolo S_3; una volta trovato, lo sostituisce con S_5 ed un altro S_5 è contenuto nella cella immediatamente a sinistra; a questo punto la computazione termina. Supponiamo ora che la descrizione istantanea iniziale sia $S_4q_0S_0S_5S_4$, allora si ha: $S_4q_0S_0S_5S_4 \xrightarrow{T} S_4S_0q_0S_5S_4 \xrightarrow{T} S_4S_0S_5q_0S_4 \xrightarrow{T} S_4S_0S_5S_4q_0S_0 \xrightarrow{T} S_4S_0S_5S_4S_0q_0S_0 \xrightarrow{T}$ $\xrightarrow{T} \cdots$. Dunque, se la macchina non trova alcuna occorrenza di S_3, la testina di lettura continua a scorrere sul nastro e l'esecuzione non ha termine. Si veda [2, Cap. 1].

3. Sia T la macchina di Turing definita dalle quadruple:

$$q_0 1 S q_1$$
$$q_1 S_0 1 q_0.$$

L'alfabeto di T è $\{S_0, 1\}$, dove 1 sta ad indicare il simbolo S_1. Osserviamo che sulle parole che iniziano con 1 la macchina aggiunge 1 alla sinistra e prosegue senza fermarsi mai. Si veda [7, Cap. 5, § 5.2].

3.3 Algoritmi di Turing e funzioni calcolabili nel senso di Turing

Per il presente paragrafo ed il successivo si invita il lettore a fare riferimento a [7, Cap. 5, § 5.2].

Consideriamo una qualsiasi macchina di Turing T con alfabeto A; è sempre possibile associarle un algoritmo $\mathfrak{B}_T$ in A, che prende il nome di *algoritmo di Turing*. Vediamone la costruzione: si considera una parola P nell'alfabeto A e si imprime ogni simbolo di P in una cella di un nastro vuoto, procedendo da sinistra verso destra. Il nastro viene immesso nella macchina T, con la testina di lettura posizionata sul quadrato contenente il primo simbolo di P. La computazione ha inizio quando la macchina si trova nello stato interno q_0. Se l'esecuzione ha termine, cioè se si raggiunge una descrizione istantanea terminale, la parola di A stampata sul nastro è il valore dell'algoritmo $\mathfrak{B}_T$.

Sia T una macchina di Turing con alfabeto A e consideriamo l'algoritmo $\mathfrak{B}_{T,C}$ nell'alfabeto C contenente A. Siano P e Q due parole in C; scriviamo $\mathfrak{B}_{T,C}(P) = Q$ se e

solo se esiste una computazione di T che inizia con la descrizione istantanea q_0P e finisce con una descrizione istantanea $R_1q_jR_2$, dove $Q = R_1R_2$.

Definizione 3.9. *Un algoritmo $\mathfrak{A}$ in un alfabeto D si dice **calcolabile nel senso di Turing** se esiste una macchina di Turing T con alfabeto A e un alfabeto C contenente $A \cup D$ tale che $\mathfrak{B}_{T,C}$ e $\mathfrak{A}$ sono pienamente equivalenti rispetto a D, cioè $\mathfrak{B}_{T,C}(P) \approx \mathfrak{A}(P)$ per ogni parola P in D.*

Osservazione 3.10. Ricordiamo che dalla Definizione 2.3 segue che, data P parola in A, $\mathfrak{B}_{T,C}(P) \approx \mathfrak{A}(P)$ se e solo se o $\mathfrak{B}_{T,C}$ e $\mathfrak{A}$ sono entrambi applicabili a P e $\mathfrak{B}_{T,C}(P) = \mathfrak{A}(P)$ oppure né $\mathfrak{B}_{T,C}$ né $\mathfrak{A}$ è applicabile a P.

Introduciamo di seguito un'ulteriore formalizzazione del concetto di calcolabilità effettiva, utilizzando la nozione di macchina di Turing.

Fissiamo le seguenti convenzioni: 1 sta per S_1, * sta per S_2; ricordiamo, inoltre, che abbiamo definito $\overline{m}$ come 1^{m+1} e denotato la parola $\overline{n}_1 * \overline{n}_2 * \cdots * \overline{n}_k$ nell'alfabeto $\{1, *\}$ con $\overline{(n_1, n_2, \ldots, n_k)}$.

Definizione 3.11. *Sia T una macchina di Turing con alfabeto A contenente $\{1, *\}$ e sia $\varphi : \mathbb{N}^k \to \mathbb{N}, k \geq 1$ una funzione numerica parziale;* [4] *diciamo che T **calcola** φ se e solo se, per numeri naturali $n_1, \ldots, n_k$ qualunque e per ogni parola Q, $\mathfrak{B}_{T,A}(\overline{(n_1, n_2 \ldots, n_k)}) = Q$ se e solo se $Q = R_1\overline{\varphi(n_1, \ldots, n_k)}R_2$, dove R_1 e R_2 sono parole formate solo da S_0.*

Osservazione 3.12. La forma $R_1\overline{\varphi(n_1, \ldots, n_k)}R_2$ è lecita, poiché il simbolo S_0 rappresenta uno spazio vuoto.

Definizione 3.13. *Sia $\varphi : \mathbb{N}^k \to \mathbb{N}, k \geq 1$ una funzione numerica parziale; φ si dice **calcolabile nel senso di Turing** (o Turing-calcolabile) se e solo se esiste una macchina di Turing T che calcola φ.*

Mostriamo alcuni esempi di funzioni Turing-calcolabili:

1. Consideriamo la funzione di successore $\varphi(\overline{n}) = \overline{n+1}$, definita per ogni n numero naturale. Sia T la macchina di Turing nell'alfabeto $\{S_0, 1\}$ definita dalle seguenti due istruzioni:

$$q_0 1 S q_1$$
$$q_1 S_0 1 q_2.$$

   È evidente che T calcola φ, infatti: $q_0\overline{n} \xrightarrow{T} q_1 S_0 \overline{n} \xrightarrow{T} q_2 1\overline{n} = q_2\overline{n+1}$.

[4] Ricordiamo la Definizione 2.11: una funzione $f : \mathbb{N}^k \to \mathbb{N}$, $k \geq 1$ si dice **parziale** se il suo dominio è un sottoinsieme di $\mathbb{N}^k$.

2. Definiamo la funzione:
$$x \dot{-} y = \begin{cases} x - y & \text{se } x \geq y, \\ 0 & \text{se } x < y. \end{cases}$$
e consideriamo la macchina di Turing T nell'alfabeto $\{S_0, 1, *\}$ definita dalle seguenti istruzioni:

$$q_0 1 S_0 q_0$$
$$q_0 S_0 D q_1$$
$$q_1 1 D q_1$$
$$q_1 * 1 q_2$$
$$q_2 1 D q_2$$
$$q_2 S_0 S q_3$$
$$q_3 1 S_0 q_3.$$

T calcola la funzione di addizione $\varphi(\overline{m} * \overline{n}) = \overline{m+n}$, per ogni coppia di numeri naturali m, n, infatti: $q_0\overline{m} * \overline{n} = q_0 1^{m+1} * 1^{n+1} \xrightarrow{T} q_0 S_0 1^m * 1^{n+1} \xrightarrow{T} S_0 q_1 1^m * 1^{n+1} \xrightarrow{T} S_0 1 q_1 1^{m\dot{-}1} * 1^{n+1} \xrightarrow{T} \cdots \xrightarrow{T} S_0 1^m q_1 * 1^{n+1} \xrightarrow{T} S_0 1^m q_2 1 1^{n+1} \xrightarrow{T} S_0 1^{m+1} q_2 1^{n+1} \xrightarrow{T} S_0 1^{m+1} 1 q_2 1^n \xrightarrow{T} \cdots \xrightarrow{T} S_0 1^{m+1} 1^{n+1} q_2 S_0 \xrightarrow{T} S_0 1^{m+1} 1^n q_3 1 S_0 \xrightarrow{T} S_0 1^{m+1} 1^n q_3 S_0 S_0 = S_0 1^{m+n+1} q_3 S_0 S_0 = S_0 \overline{m+n} q_3 S_0 S_0$.

3.4 Equivalenza tra Markov-calcolabilità e Turing-calcolabilità

Nel seguente paragrafo dimostriamo che l'approccio alla calcolabilità per mezzo delle macchine di Turing è equivalente a quello per mezzo degli algoritmi di Markov. Proviamo quindi che la classe delle funzioni calcolabili nel senso di Turing coincide con la classe delle funzioni calcolabili nel senso di Markov [7, Cap. 5, § 5.2]. Partiamo dal dimostrare che ogni funzione Turing-calcolabile è parzialmente Markov-calcolabile:

Proposizione 3.14. *Sia T una macchina di Turing con alfabeto A. Sia C un'estensione di A, cioè $C \supseteq A$. Allora esiste un algoritmo normale $\mathfrak{A}$ sopra C che è pienamente equivalente all'algoritmo di Turing $\mathfrak{B}_{T,C}$ rispetto a C.*

Dimostrazione. Sia $D = C \cup \{q_{k_0}, q_{k_1}, \ldots, q_{k_N}\}$ dove $q_{k_0}, q_{k_1}, \ldots, q_{k_N}$ sono gli stati interni di T e poniamo $q_{k_0} = q_0$. Costruiamo lo schema di algoritmo di $\mathfrak{A}$ associando ad ogni tipologia di quadruple di T una corrispondente produzione:

1. alle quadruple della forma $q_jS_iS_kq_r$ di T abbiniamo le produzioni $q_jS_i \to S_kq_r$;

2. a $q_jS_iSq_r$ associamo $S_lq_jS_i \to q_rS_lS_i$, per ogni simbolo S_l di C. In particolare, le produzioni della forma $q_jS_i \to q_rS_0S_i$ rappresentano il caso in cui nella macchina di Turing la testina di lettura è posta sul primo simbolo del nastro e l'istruzione richiede di spostarsi a sinistra; ciò è possibile grazie all'aggiunta di una cella vuota, cioè contenente il simbolo S_0;

3. a $q_jS_iDq_r$ abbiniamo le produzioni $q_jS_iS_l \to S_iq_rS_l$, per ogni simbolo S_l di C. In particolare, le produzioni $q_jS_i \to S_iq_rS_0$ rappresentano il caso in cui la testina di lettura è posta sull'ultimo simbolo del nastro e l'istruzione richiede di spostarsi a destra; anche in questo caso ciò è possibile grazie all'aggiunta di una cella vuota.

Infine, scriviamo le produzioni:

4. $q_{k_i} \to \cdot\, \Lambda$ per ogni stato interno q_{k_i} di T;

5. $\Lambda \to q_0$; ricordiamo infatti che la macchina inizia a funzionare nello stato iniziale q_0.

Questo schema definisce un algoritmo $\mathfrak{A}$ sopra C tale che $\mathfrak{B}_{T,C}(P) \approx \mathfrak{A}(P)$, per ogni parola P in C. □

Corollario 3.15. *Ogni funzione φ calcolabile nel senso di Turing è parzialmente calcolabile nel senso di Markov.*

Dimostrazione. Sia $\varphi : \mathbb{N}^k \to \mathbb{N}$ una funzione calcolabile nel senso di Turing per mezzo di una macchina di Turing T con alfabeto $A \supseteq \{1, *\}$. Allora per la Proposizione 3.14, esiste un algoritmo normale $\mathfrak{A}$ sopra A pienamente equivalente a $\mathfrak{B}_{T,A}$ rispetto ad A: cioè per ogni $n_1, \ldots, n_k$ si ha $\mathfrak{A}(\overline{n}_1 * \cdots * \overline{n}_k) \approx \mathfrak{B}_{T,A}(\overline{n}_1 * \cdots * \overline{n}_k) \approx R_1\overline{\varphi(n_1, \ldots, n_k)}R_2$, dove R_1, R_2 sono parole costituite dal solo simbolo S_0. Costruiamo l'algoritmo normale sopra $M = \{1, *\}$ richiesto dalla definizione di Markov-calcolabilità: sia $\mathfrak{C}_1$ un algoritmo normale sopra $\{1, *, S_0\}$ tale che $\mathfrak{C}_1$ cancelli tutti gli S_0 che occorrono prima del primo 1 o *. Lo schema d'algoritmo di $\mathfrak{C}_1$ è il seguente:

$$\begin{aligned} \alpha S_0 &\to \alpha \\ \alpha 1 &\to \cdot\, 1 \\ \alpha * &\to \cdot\, * \\ \alpha &\to \cdot\, \Lambda \\ \Lambda &\to \alpha. \end{aligned}$$

Sia $\mathfrak{C}_2$ un algoritmo normale sopra $\{1, *, S_0\}$ tale che cancelli tutti gli S_0 che occorrono dopo l'ultimo 1 o *; $\mathfrak{C}_2$ ha il seguente schema di algoritmo:

$$\begin{aligned} \alpha * &\to *\alpha \\ \alpha 1 &\to 1\alpha \\ \alpha S_0 &\to \alpha \\ \alpha &\to \cdot\, \Lambda \\ \Lambda &\to \alpha. \end{aligned}$$

Sia $\mathfrak{C}$ l'algoritmo normale $\mathfrak{C}_2 \circ \mathfrak{C}_1 \circ \mathfrak{A}$ (è stato dimostrato nel Paragrafo 2.3 che la composizione di algoritmi normali è ancora un algoritmo normale). Poiché per ogni $n_1, \ldots, n_k$ vale $\mathfrak{A}(\overline{n}_1 * \cdots * \overline{n}_k) \approx R_1\overline{\varphi(n_1, \ldots, n_k)}R_2$, con R_1, R_2 sequenze di S_0, si ha

$$\begin{aligned} \mathfrak{C}_1(R_1\overline{\varphi(n_1, \ldots, n_k)}R_2) &= \overline{\varphi(n_1, \ldots, n_k)}R_2; \\ \mathfrak{C}_2(\overline{\varphi(n_1, \ldots, n_k)}R_2) &= \overline{\varphi(n_1, \ldots, n_k)}. \end{aligned}$$

Abbiamo, quindi, dimostrato che φ è parzialmente Markov-calcolabile mediante $\mathfrak{C}$. □

Di seguito dimostriamo che ogni funzione parzialmente Markov-calcolabile è Turing-calcolabile:

Proposizione 3.16. *Sia $\mathfrak{A}$ un algoritmo normale in un alfabeto A che non contenga S_0 o δ. Allora esiste una macchina di Turing T tale che l'algoritmo di Turing $\mathfrak{B} = \mathfrak{B}_{T,(A\cup\{S_0,\delta\})}$ nell'alfabeto $A \cup \{S_0, \delta\}$ ha la seguente proprietà: per ogni parola W in A, $\mathfrak{B}$ è applicabile a W se e solo se lo è $\mathfrak{A}$; inoltre $\mathfrak{B}(W)$ ha la forma $(S_0)^m\mathfrak{A}(W)(S_0)^n$, dove m e n sono interi non negativi.*

Osservazione 3.17. $\mathfrak{A}(W)$ e $\mathfrak{B}(W)$ differiscono per la presenza di parole costituite dal solo simbolo S_0; ciò deriva dal fatto che in una macchina di Turing S_0 sta ad indicare uno spazio vuoto, mentre nella teoria degli algoritmi normali viene considerato come un simbolo qualsiasi.

Dimostrazione. Costruiamo una macchina di Turing T che imiti l'azione dell'algoritmo normale $\mathfrak{A}$. Assumiamo, mediante un cambiamento di indici, che $A = \{S_1, S_2, \ldots, S_k\}$. Sia $P \to (\cdot)Q$ una produzione arbitraria; costruiamo le quadruple di una macchina di Turing in modo tale che la loro azione equivalga alla sostituzione dell'occorrenza più a sinistra (se esiste) di P in una parola W con Q. Se $P \neq \Lambda$, supponiamo che P sia $b_0, \ldots, b_r$.

Consideriamo le seguenti quadruple:

$$
\begin{array}{llll}
q_0 & S_i & D & q_0 \qquad (S_i \in A,\ S_i \neq b_0) \\
q_0 & b_0 & \delta & q_0 \\
q_0 & \delta & D & q_2 \\
q_2 & b_1 & D & q_3 \\
q_2 & S_i & S_i & q_{r+2} \qquad (S_i \in A \cup \{S_0\},\ S_i \neq b_1) \\
q_3 & b_2 & D & q_4 \\
q_3 & S_i & S_i & q_{r+2} \qquad (S_i \in A \cup \{S_0\},\ S_i \neq b_2) \\
\vdots & \vdots & \vdots & \vdots \\
q_r & b_{r-1} & D & q_{r+1} \\
q_r & S_i & S_i & q_{r+2} \qquad (S_i \in A \cup \{S_0\},\ S_i \neq b_{r-1}) \\
q_{r+1} & b_r & D & q_{r+4} \\
q_{r+1} & S_i & S_i & q_{r+2} \qquad (S_i \in A \cup \{S_0\},\ S_i \neq b_r) \\
q_{r+2} & S_i & S & q_{r+2} \qquad (S_i \in A \cup \{S_0\}) \\
q_{r+2} & \delta & b_0 & q_{r+3} \\
q_{r+3} & b_0 & D & q_0 \\
q_0 & S_0 & S & q_{r+5} \\
q_{r+5} & S_i & S & q_{r+5} \qquad (S_i \in A) \\
q_{r+5} & \delta & b_0 & q_{r+5} \\
q_{r+5} & S_0 & D & q_Y .
\end{array}
$$

Osserviamo che Y è un intero maggiore di tutti gli indici, il cui valore preciso verrà specificato in seguito.

Se q_0W è immesso nel nastro della macchina, si distinguono due casi possibili:

1. se W non ha occorrenze di P, allora la computazione ha termine con la descrizione istantanea q_YW;

2. se P occorre in W e $W = W_1PW_2$, con il simbolo P che rappresenta l'occorrenza più a sinistra di P in W, allora la descrizione istantanea terminale è $W_1Pq_{r+4}W_2$.

Aggiungiamo ora le quadruple che permettono la sostituzione di P con Q, dove $Q = c_0 \dots c_s$. Distinguiamo tre diversi casi:

1. caso $s = r$, cioè P e Q hanno la stessa lunghezza:

$$\begin{array}{llll} q_{r+4} & S_i & S & q_{r+7} \qquad (S_i \in A \cup \{S_0\}) \\ q_{r+7} & b_r & c_r & q_{r+8} \\ q_{r+8} & c_r & S & q_{r+9} \\ q_{r+9} & b_{r-1} & c_{r-1} & q_{r+10} \\ q_{r+10} & c_{r-1} & S & q_{r+11} \\ \vdots & \vdots & \vdots & \vdots \\ q_{3r+7} & b_0 & c_0 & q_{3r+8} \\ q_{3r+8} & S_i & S & q_{3r+8} \qquad (S_i \in A) \\ q_{3r+8} & S_0 & D & q_u; \end{array}$$

   dove

$$u = \begin{cases} 0 & \text{se } P \to (\cdot)Q \text{ è semplice,} \\ 1 & \text{se } P \to (\cdot)Q \text{ è terminale.} \end{cases}$$

   Applicando queste quadruple a $W_1 P q_{r+4} W_2$, otteniamo $q_u W_1 Q W_2$;

2. caso $s < r$, cioè Q è più breve di P.

$$\begin{array}{llll} q_{r+4} & S_i & S & q_{r+7} \qquad (S_i \in A \cup \{S_0\}) \\ q_{r+7} & b_r & c_s & q_{r+8} \\ q_{r+8} & c_s & S & q_{r+8} \\ \vdots & \vdots & \vdots & \vdots \\ q_{r+7+2s} & b_{r-s} & c_0 & q_{r+7+2s+1} \\ q_{r+7+2s+1} & c_0 & S & q_{r+7+2s+2} \\ q_{r+7+2s+2} & b_{r-s-1} & S_0 & q_{r+7+2s+2} \\ q_{r+7+2s+2} & S_0 & S & q_{r+7+2s+3} \\ q_{r+7+2s+3} & b_{r-s-2} & S_0 & q_{r+7+2s+3} \\ q_{r+7+2s+3} & S_0 & S & q_{r+7+2s+4} \\ \vdots & \vdots & \vdots & \vdots \\ q_{2r+s+8} & b_0 & S_0 & q_{2r+s+8}. \end{array}$$

   L'azione di queste quadruple su $W_1 P q_{r+4} W_2$ produce $W_1 q_{2r+s+8} (S_0)^{r-s} Q W_2$.

   Definiamo ora delle quadruple che spostino W_1 di $r - s$ quadrati a destra per ottenere $W_1 Q W_2$ (preceduto da un certo numero di S_0). Sia M un intero maggiore di tutti gli indici dei q_i e degli S_i precedenti, ad esempio $M = 3r + 9$ e sia $j = 1, 2, \ldots, k$.

Definiamo le seguenti quadruple:

$$
\begin{array}{llll}
q_{2r+s+8} & S_0 & S & q_M \\
q_M & S_j & \delta & q_{M+j} \qquad (S_j \in A) \\
q_{M+j} & \delta & D & q_{M+j} \\
q_{M+j} & S_0 & D & q_{M+j} \\
q_{M+j} & S_l & S & q_{2M+j} \qquad (S_l \in A) \\
q_{2M+j} & S_0 & S_j & q_{2M+j} \\
q_{2M+j} & S_j & S & q_{3M+j} \\
q_{3M+j} & S_0 & S & q_{3M+j} \\
q_{3M+j} & \delta & S_0 & q_{4M+j} \\
q_{4M+j} & S_0 & S & q_{5M+j} \\
q_{5M+j} & S_0 & D & q_{6M+j} \\
q_{6M+j} & S_0 & D & q_{6M+j} \\
q_{6M+j} & S_l & S_l & q_u \qquad (S_l \in A) \\
q_{5M+j} & S_l & S_l & q_M \qquad (S_l \in A);
\end{array}
$$

dove

$$
u = \begin{cases} 0 & \text{se } P \to (\cdot)Q \text{ è semplice,} \\ 1 & \text{se } P \to (\cdot)Q \text{ è terminale.} \end{cases}
$$

L'azione di queste quadruple trasforma $W_1 q_{2r+s+8}(S_0)^{r-s}QW_2$ in $(S_0)^p q_u W_1 Q W_2$, con p intero positivo;

3. caso $s > r$, cioè Q è più lungo di P. Questo caso è analogo al precedente, a meno di qualche modifica; viene quindi lasciato al lettore.

   Similmente, anche il caso in cui P o Q sia vuoto, viene lasciato al lettore.

Sia ora un algoritmo normale $\mathfrak{A}$ in un alfabeto $A = \{S_1, \ldots, S_k\}$ che non contiene S_0 o δ; consideriamo lo schema d'algoritmo di $\mathfrak{A} : P_1 \to (\cdot)Q_1, \ldots, P_h \to (\cdot)Q_h$. Definiamo una macchina di Turing T nel seguente modo: consideriamo la produzione $P_1 \to (\cdot)Q_1$ ed elenchiamo le quadruple associate, come abbiamo fatto in precedenza per la generica produzione $P \to (\cdot)Q$. Assumiamo, inoltre, che Y sia un numero 100 volte maggiore della somma di k e del numero di occorrenze dei simboli nello schema. Dunque, dato $q_0 W$, si distinguono due casi:

1. se W non contiene alcuna occorrenza di P_1, l'azione delle quadruple produce $q_Y W$;

2. se $W = W_1 P_1 W_2$, con il simbolo P_1 che rappresenta l'occorrenza più a sinistra di P_1 in W, allora otteniamo come risultato $(S_0)^v q_u W_1 Q W_2$, dove v è un intero non negativo e $u = 0$ se $P_1 \to (\cdot)Q_1$ è semplice; $u = 1$ se $P_1 \to (\cdot)Q_1$ è terminale.

Consideriamo ora $P_2 \to (\cdot)Q_2$ e costruiamo le quadruple per questa produzione; nel fare ciò aumentiamo di Y gli indici di tutti i q_i, eccetto q_u, in modo tale che non vi sia sovrapposizione con le quadruple associate a $P_1 \to (\cdot)Q_1$. Dunque, se nella parola W non si ha nessuna occorrenza di P_1, si cercano le occorrenze di P_2 in W : se W ne contiene almeno una, si sostituisce l'occorrenza più a sinistra di P_2 con Q_2 e se $P_2 \to (\cdot)Q_2$ è semplice, si ritorna allo stato iniziale q_0 e all'azione del primo gruppo di quadruple; se $P_2 \to (\cdot)Q_2$ è terminale, la computazione ha termine con stato finale q_1. Il seguente procedimento viene ripetuto per $P_3 \to (\cdot)Q_3$; in questo caso gli indici dei q_i vengono incrementati di $2Y$. Si procede nello stesso modo con tutte le produzioni dello schema d'algoritmo.

La macchina di Turing T così definita imita l'azione dell'algoritmo normale $\mathfrak{A}$ in modo tale che, per ogni parola W in A, $\mathfrak{B} = \mathfrak{B}_{T,(A\cup\{S_0,\delta\})}$ è applicabile a W se e solo se lo è $\mathfrak{A}$, e $\mathfrak{B}(W)$ ha la forma $(S_0)^m \mathfrak{A}(W)(S_0)^n$, dove m e n sono interi non negativi. □

Corollario 3.18. *Ogni funzione parzialmente Markov-calcolabile è Turing-calcolabile.*

Dimostrazione. La tesi segue direttamente dalla Proposizione 3.16 e dalla definizione di Turing-calcolabilità □

3.5 Tesi di Church-Turing e principio di normalizzazione

Nel corso di questi ultimi due capitoli le nozioni di algoritmo e calcolabilità effettiva sono state prima introdotte in maniera generale, successivamente sono state formalizzate seguendo due diversi approcci: quello di Markov e quello di Turing. Sorge spontaneo domandarsi in quale misura questi due tentativi esauriscono il problema di cercare una formalizzazione adeguata. La risposta a tale domanda è fornita dalla ***Tesi di Church-Turing*** [1], [8].

La classe definita in maniera intuitiva delle funzioni effettivamente calcolabili coincide con la classe delle funzioni calcolabili nel senso di Turing.

Poiché il concetto di funzione effettivamente calcolabile è un concetto intuitivo, pre-matematico, non è possibile fornire una dimostrazione rigorosa della validità della tesi; tuttavia, vi sono due principali argomentazioni a suo sostegno:

1. la robustezza della definizione di macchina di Turing: infatti, nel tempo sono state proposte molte varianti, quali l'aggiunta di nastri e testine addizionali oppure la definizione di macchine non deterministiche; tuttavia, è stato dimostrato che queste variazioni sono equivalenti alle macchine di Turing classiche; cioè, nonostante le caratteristiche addizionali, il potere espressivo della macchina rimane invariato. Tale considerazione suggerisce una prima idea dell'esistenza di un limite superiore a ciò che è possibile calcolare [2], [5];

2. gli approcci di Turing e di Markov costituiscono solo due delle innumerevoli teorie che si sono sviluppate intorno al concetto di calcolabilità; ricordiamo il lavoro di: Kleen sulle funzioni ricorsive, Church sulla λ-calcolabilità, Herbrand-Gödel, Post sui sistemi normali. Nel paragrafo precedente si è dimostrato che la classe delle funzioni Turing-calcolabili coincide con la classe delle funzioni Markov-calcolabili, ma è possibile dimostrare che tutti gli approcci elencati sopra sono equivalenti, nel senso che la classe delle funzioni calcolabili è sempre la stessa [2], [8].

Nonostante tali considerazioni è possibile che in un futuro la tesi di Church-Turing possa venire confutata; questo richiederebbe la costruzione di un procedimento di calcolo effettivo (nel senso intuitivo) non esprimibile in nessuno dei linguaggi di programmazione finora conosciuti [5]. Per ora non c'è nulla che preannunci un simile scenario.

Consideriamo una seconda domanda, attribuibile a Markov stesso (si veda [6, Cap. II, § 5]): dato un qualsiasi algoritmo in un alfabeto A; è possibile sostituirlo con un algoritmo normale, pienamente equivalente al primo rispetto ad A? In risposta a tale domanda, Markov formula il seguente ***principio di normalizzazione***.

Ogni algoritmo in A è pienamente equivalente rispetto ad A a qualche algoritmo normale sopra A.

Dato un algoritmo in un alfabeto A; diciamo che è *normalizzabile* se è pienamente equivalente rispetto ad A ad un qualche algoritmo normale sopra A. Il principio di normalizzazione può quindi essere riformulato come segue:

Ogni algoritmo in A è normalizzabile.

Questo non costituisce un teorema matematico, dunque non può essere dimostrato in maniera rigorosa, ma risulta equivalente alla tesi di Church-Turing, che può essere riformulata nel seguente modo [10]:

La nozione intuitiva di algoritmo coincide con la nozione precisa di algoritmo di Turing.

L'equivalenza tra questa e il principio di Markov segue direttamente dalla Proposizione 3.14 e dalla Proposizione 3.16.

Disporre di una nozione precisa di algoritmo permette di sviluppare dei risultati sui limiti degli algoritmi stessi [5]; in particolare, dalla tesi di Church-Turing segue che se la soluzione di un dato problema non può essere calcolata da una macchina di Turing, allora non può essere calcolata da alcuna procedura algoritmica. Seguendo questa intuizione, nel capitolo successivo sfrutteremo le macchine di Turing per dimostrare l'indecidibilità del problema della parola.

Capitolo 4

L'indecidibilità del problema della parola

Nel presente capitolo affrontiamo un particolare problema di decisione, noto come "problema della parola". Esso verrà introdotto dapprima in modo informale; successivamente ne sarà fornita una definizione rigorosa basata sulle macchine di Turing. La trattazione fa ampio uso di alcune nozioni di teoria dei gruppi, raccolte nel Capitolo 1, che il lettore è invitato a consultare in caso di necessità. Per approfondire i concetti qui esposti si rimanda, inoltre, a [9, Cap. 12].

4.1 Processi decisionali ed insiemi ricorsivi

I *problemi di decisione* sono una particolare classe di problemi caratterizzati da due componenti: dei dati iniziali e una sequenza di domande che richiedono una risposta della forma sì/no. Data una lista $\mathscr{L}$ di domande; un problema di decisione si dice *indecidibile* se non esiste un *processo decisionale* (o algoritmo) per $\mathscr{L}$, cioè se non esiste alcun procedimento meccanico che, applicato ad ogni domanda di $\mathscr{L}$, riesca a fornire la risposta corretta "sì" o "no", dopo un numero finito di passi.

Il *problema della parola* è un esempio di problema di decisione, infatti: consideriamo un gruppo G finitamente generato con presentazione

$$G = (x_1, \ldots, x_n \mid r_j = 1, j \geq 1),$$

dove con 1 indichiamo l'elemento neutro del gruppo. I dati del problema sono gli elementi di G, cioè le parole ω nell'insieme $X = \{x_1, \ldots, x_n\}$ dei generatori; l'insieme $\mathscr{L}$ è costituito dalle domande della forma: "se ω è una parola in X, $\omega = 1$ in G?"

La dimostrazione rigorosa dell'indecidibilità del problema della parola richiede di formalizzare la nozione intuitiva di processo decisionale; a tale scopo sfrutteremo le macchine di Turing e le definizioni di insieme ricorsivamente enumerabile e ricorsivo.

Definizione 4.1. *Sia Ω l'insieme di tutte le parole positive nei simboli $S = \{S_1, \ldots, S_M\}$ e sia T una macchina di Turing con alfabeto A, contenente S. Definiamo l'insieme*

$$e(T) = \{\omega \in \Omega : T \text{ calcola } \omega\},$$

e diciamo che T ***enumera*** *$e(T)$. Un sottoinsieme E di Ω si dice* ***ricorsivamente enumerabile*** *(**r.e.**) se esiste una macchina di Turing T che enumera E.*

Definizione 4.2. *Sia Ω l'insieme di tutte la parole positive in $\{S_1, \ldots, S_M\}$. Un sottoinsieme E di Ω si dice* ***ricorsivo*** *se sia E sia il suo complementare $\Omega \setminus E$ sono sottoinsiemi r.e.*

Sia T una macchina di Turing e Ω l'insieme delle parole positive nell'alfabeto di T; vogliamo determinare se una parola ω in Ω appartiene a $E = e(T)$ oppure no. È sufficiente fornire in input ω a T; ciò genererà una sequenza di descrizioni istantanee $q_1\omega \to \alpha_2 \to \alpha_3 \to \cdots$. Se ω è un elemento di E, allora T calcola ω, cioè la sequenza di descrizioni istantanee è finita; altrimenti la computazione di T prosegue all'infinito. A priori non è possibile sapere se l'esecuzione avrà termine oppure no, eccetto nel caso in cui E sia un sottoinsieme ricorsivo, infatti: date T e T' macchine di Turing che enumerano rispettivamente $E = e(T)$ e $\Omega \setminus E = e(T')$, è sufficiente fornire in input ω sia a T sia a T'. In una porzione finita di tempo una delle due esecuzioni avrà termine, determinando il sottoinsieme di appartenenza di ω.

Sfruttando le definizioni precedenti possiamo formalizzare la nozione intuitiva di processo decisionale come segue: assumiamo che la lista $\mathscr{L}$ di domande sia formata da parole positive in una dato alfabeto, che E sia l'insieme delle parole che hanno risposta "sì" e il suo complementare sia costituito da tutte le parole per cui la risposta è "no". Possiamo dunque riformulare la *tesi di Church-Turing* nel modo seguente: *gli insiemi ricorsivi sono i sottoinsiemi che ammettono un processo decisionale.*

Avvertiamo il lettore che, al fine di alleggerire la notazione, da qui in avanti la freccia " $\xrightarrow{T}$ ", utilizzata per indicare un passo di calcolo di una macchina di Turing T, verrà rappresentata semplicemente con " $\to$ ". Tale notazione non genera ambiguità con la rappresentazione di una produzione di uno schema d'algoritmo normale, perché nel presente capitolo non tratteremo esplicitamente gli algoritmi di Markov.

Osserviamo che le Definizioni 4.1 e 4.2 possono essere applicate a sottoinsiemi di numeri naturali, identificando $n \in \mathbb{N}$ con la parola positiva ${S_1}^{n+1}$.

Definizione 4.3. *Un sottoinsieme E di $\mathbb{N}$ si dice ricorsivamente enumerabile se esiste una macchina di Turing T in un alfabeto contenente il simbolo S_1, tale che $E = \{n \in \mathbb{N} : T \text{ calcola } {S_1}^{n+1}\}$.*

Dimostriamo ora un risultato che tornerà utile in seguito:

Teorema 4.4. *Esiste un sottoinsieme r.e. dell'insieme dei numeri naturali $\mathbb{N}$ che non è ricorsivo.*

Dimostrazione. Esiste una quantità numerabile di macchine di Turing, infatti: una macchina di Turing T è un insieme finito di quadruple costituite a partire da un insieme numerabile di simboli $\{R, L, S_0, S_1, \dots, q_0, q_1, \dots\}$. Assegnamo un numero naturale ad ognuno di questi:

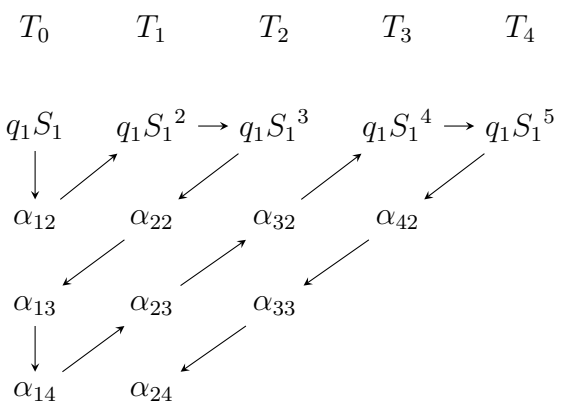
$$\begin{aligned} &R \mapsto 0; \quad L \mapsto 1; \\ &q_0 \mapsto 2; \quad q_1 \mapsto 4; \quad q_2 \mapsto 6; \quad \dots; \\ &S_0 \mapsto 3; \quad S_1 \mapsto 5; \quad S_2 \mapsto 7; \quad \dots. \end{aligned}$$

Supponiamo che T sia costituita da m quadruple; indichiamo con $w(T)$ la parola di lunghezza $4m$, ottenuta dalla giustapposizione di queste. È chiaro che se $T \neq T'$, allora $w(T) \neq w(T')$. Ad ogni macchina di Turing T assegnamo il corrispondente numero di Gödel:

$$G(T) = \prod_{i=1}^{4m} {p_i}^{e_i},$$

dove p_i è l'i-esimo numero primo ed e_i è il numero naturale corrispondente all'i-esimo simbolo, coerentemente alla numerazione definita sopra. Dal Teorema Fondamentale dell'Aritmetica segue che macchine di Turing diverse hanno diversi numeri di Gödel. Dunque, tutte le macchine di Turing possono essere enumerate, cioè $T_0, T_1, \dots, T_n, \dots$ e diciamo che T precede T' se $G(T) < G(T')$.

Definiamo ora l'insieme $E = \{n \in \mathbb{N} : T_n \text{ calcola } {S_1}^{n+1}\}$, cioè $n \in E$ se e solo se l'n-esima macchina di Turing T_n calcola n. Dimostriamo che E è un insieme r.e. Consideriamo la seguente figura:



L'n-esima colonna rappresenta la sequenza di descrizioni istantanee della n-esima macchina di Turing T_n quando $q_1 S_1{}^{n+1}$ viene dato in input sul nastro. È possibile costruire una macchina di Turing T^* con stato finale q_N che esegua le seguenti istruzioni: segue l'ordine delle frecce nella figura e dichiara che n appartiene ad E se individua una descrizione istantanea terminale α_{ni} nell'n-esima colonna.

Dimostriamo, infine, che E non è un insieme ricorsivo, cioè che il suo complementare $E' = \{n \in \mathbb{N} : n \notin E\} = \{n \in \mathbb{N} : T_n \text{ non calcola } S_1{}^{n+1}\}$ non è un sottoinsieme r.e. di $\mathbb{N}$. Utilizziamo una variante dell'argomento diagonale di Cantor: supponiamo per assurdo che esista una macchina di Turing T' che enumeri E'; poiché abbiamo elencato tutte le macchine di Turing, esisterà un $m \in \mathbb{N}$ tale che $T' = T_m$. Se $m \in E' = e(T') = e(T_m)$, allora T_m calcola $S_1{}^{m+1}$, cioè $m \in E$, che è assurdo. Se invece $m \notin E'$, allora $m \in E$, cioè per definizione di E, T_m calcola $S_1{}^{m+1}$; dunque, $m \in e(T_m) = e(T') = E'$, che è assurdo. Ciò prova che E' non è r.e. □

Torneranno utili anche i seguenti risultati, di cui però non riportiamo la dimostrazione:

Osservazione 4.5. Sia Ω l'insieme di tutte le parole positive nei simboli $S = \{S_1, \ldots, S_M\}$. Se E_1, E_2 sono sottoinsiemi r.e. di Ω, allora $E_1 \cup E_2$ e $E_1 \cap E_2$ sono sottoinsiemi r.e. di Ω. Analogamente, se E_1, E_2 sono sottoinsiemi ricorsivi di Ω, allora $E_1 \cup E_2$ e $E_1 \cap E_2$ sono sottoinsiemi ricorsivi di Ω.

4.2 Il problema della parola per i semigruppi

Nel tentativo di affrontare il problema della parola per i gruppi, risulta utile ridursi al caso dei semigruppi. L'obiettivo del presente paragrafo è dimostrare il teorema di Markov-Post, che assicura l'esistenza di un semigruppo finitamente presentato con problema della parola indecidibile.

Definizione 4.6. *Sia Γ un semigruppo con generatori $X = \{x_1, \ldots, x_n\}$ e sia Ω l'insieme delle parole positive su X. Si dice che il semigruppo Γ ha* ***problema della parola indecidibile*** *se esiste una parola $\omega_0 \in \Omega$ tale che l'insieme $\{\omega \in \Omega : \omega = \omega_0 \text{ in } \Gamma\}$ non è ricorsivo.*

Date ω e ω' due parole in un alfabeto X; scriviamo $\omega \equiv \omega'$ se le due parole hanno lo stesso *spelling*.

Definizione 4.7. *Sia Γ un semigruppo con presentazione $\Gamma = (X \mid \alpha_j = \beta_j, j \in J)$ e siano ω ed ω' due parole in X. Si indica con $\omega_i \to \omega_{i+1}$ un'****operazione elementare****,*

cioè o $\omega_i \equiv \sigma\alpha_j\tau$ *e* $\omega_{i+1} \equiv \sigma\beta_j\tau$ *oppure* $\omega_i \equiv \sigma\beta_j\tau$ *e* $\omega_{i+1} \equiv \sigma\alpha_j\tau$*, per* $j \in J$ *e* σ, τ *parole positive in* X*.*

Definizione 4.8. *Sia* Γ *un semigruppo con presentazione* $\Gamma = (X \mid \alpha_j = \beta_j, j \in J)$ *e siano* ω *ed* ω' *due parole in* X*; si pone* $\omega = \omega'$ *in* Γ *se e solo se esiste una sequenza finita* $\omega \equiv \omega_1 \to \omega_2 \to \cdots \to \omega_t \equiv \omega'$*, tale che* $\omega_i \to \omega_{i+1}$ *è un'operazione elementare.*

Definiamo ora il semigruppo associato ad una macchina di Turing:

Definizione 4.9. *Sia* T *una macchina di Turing con stato finale* q_N*; si definisce* $\Gamma(T)$ *il* ***semigruppo associato*** *a* T *con presentazione*

$$\Gamma(T) = (q, h, S_0, S_1, \ldots, S_M, q_0, q_1, \ldots, q_N \mid R(T)),$$

dove $S = \{S_0, S_1, \ldots, S_M\}$ *è l'alfabeto di* T*,* $\{q_0, q_1, \ldots, q_N\}$ *è l'insieme dei suoi stati interni e* q, h *sono dei nuovi simboli.*

$R(T)$ *è costituito dalle seguenti relazioni:*

$$q_iS_j = q_lS_k \quad se \quad q_iS_jS_kq_l \in T, \tag{4.1}$$

per ogni $\beta = 0, 1, \ldots, M$ *:*

$$q_iS_jS_\beta = S_jq_lS_\beta \quad se \quad q_iS_jRq_l \in T, \tag{4.2}$$

$$q_iS_jh = S_jq_lS_0h \quad se \quad q_iS_jRq_l \in T, \tag{4.3}$$

$$S_\beta q_iS_j = q_lS_\beta S_j \quad se \quad q_iS_jLq_l \in T, \tag{4.4}$$

$$hq_iS_j = hq_lS_0S_j \quad se \quad q_iS_jLq_l \in T, \tag{4.5}$$

$$q_NS_\beta = q_N, \tag{4.6}$$

$$S_\beta q_Nh = q_Nh, \tag{4.7}$$

$$hq_Nh = q. \tag{4.8}$$

Osservazione 4.10. È immediato osservare che i primi cinque tipi di relazioni derivano dalle diverse tipologie di passo di calcolo introdotte nella Definizione 3.4.

Nelle relazioni (4.3) e (4.5) il simbolo h denota l'estremità finale del nastro della macchina; possiamo quindi introdurre la seguente categoria di parole:

Definizione 4.11. *Una parola è detta* h***-speciale*** *se è della forma* $h\alpha h$*, dove* α *è una descrizione istantanea.*

Osservazione 4.12. Sia T una macchina di Turing con alfabeto $S = \{S_0, S_1, \ldots, S_M\}$ e stato finale q_N. Consideriamo la parola $h\alpha h$, dove α è una descrizione istantanea terminale della forma $\sigma q_N\tau$, dove σ e τ sono parole in S e τ è non vuota. Segue direttamente dalle relazioni (4.6), (4.7), (4.8) che $h\alpha h = q$ in $\Gamma(T)$ quando α è terminale.

Lemma 4.13. *Sia T una macchina di Turing con stato finale q_N e sia $\Gamma(T)$ il suo semigruppo associato. Allora valgono le seguenti:*

1. *Siano ω ed ω' parole in $\{S_0, S_1, \ldots, S_M, q_0, q_1, \ldots, q_N\}$ con $\omega \not\equiv q$ e $\omega' \not\equiv q$. Se $\omega \to \omega'$ è un'operazione elementare, allora ω è h-speciale se e solo se ω' è h-speciale.*

2. *Se $\omega = h\alpha h$ è h-speciale, $\omega' \not\equiv q$ e $\omega \to \omega'$ è un'operazione elementare che usa una delle relazioni da (4.1) a (4.5), allora $\omega' \equiv h\beta h$, dove $\alpha \to \beta$ o $\beta \to \alpha$ è un passo di calcolo di T.*

Dimostrazione. 1. È sufficiente osservare che $\omega \not\equiv q, \omega' \not\equiv q$ e che la relazione (4.8) è l'unica in grado di creare o cancellare il simbolo h.

2. Dal punto precedente segue che ω' è h-speciale, dunque supponiamo $\omega' \equiv h\beta h$, dove β è una descrizione istantanea. Se $\omega \to \omega'$ è un'operazione elementare, dalla Definizione 4.7 segue che $\omega_i \equiv \sigma\alpha\tau$ e $\omega_{i+1} \equiv \sigma\beta\tau$ oppure $\omega_i \equiv \sigma\beta\tau$ e $\omega_{i+1} \equiv \sigma\alpha\tau$, dove σ e τ sono parole positive nell'insieme dei generatori di Γ e $\alpha = \beta$ è una delle relazioni dei primi cinque tipi. Ad ognuna di queste è associata una quadrupla di T e una quadrupla corrisponde a sua volta ad un passo di calcolo, dunque vale $\alpha \to \beta$ oppure $\beta \to \alpha$. □

Lemma 4.14. *Sia T una macchina di Turing con stato finale q_N, Ω l'insieme di tutte le parole positive nell'alfabeto di T e sia $E = e(T)$. Se $\omega \in \Omega$, allora*

$$\omega \in E \text{ se e solo se } hq_0\omega h = q \text{ in } \Gamma(T).$$

Dimostrazione. Dimostriamo la condizione necessaria. Se $\omega \in E$, esiste una sequenza di descrizioni istantanee $\alpha_0 = q_0\omega, \alpha_1, \ldots, \alpha_t$ tale che $\alpha_i \to \alpha_{i+1}$ è un passo di calcolo e α_t contiene lo stato finale q_N. Dalle operazioni elementari associate ai primi cinque tipi di relazioni in $\Gamma(T)$, si ottiene che $hq_0\omega h = h\alpha_t h$ in $\Gamma(T)$; applicando le relazioni (4.6), (4.7), (4.8) segue direttamente che $h\alpha_t h = q$ in $\Gamma(T)$.

Dimostriamo ora la condizione sufficiente. Se $hq_0\omega h = q$ in $\Gamma(T)$, allora esistono $\omega_0, \ldots, \omega_t$ parole in $\{h, S_0, S_1, \ldots, S_M, q_0, q_1, \ldots, q_N\}$ tale che

$$hq_0\omega h \equiv \omega_0 \to \omega_1 \to \cdots \to \omega_t \equiv hq_N h \to q,$$

dove $\omega_i \to \omega_{i+1}$ è un'operazione elementare. Poiché ω_0 è h-speciale, dal Lemma 4.13 (1) segue che, per ogni i, $\omega_i \equiv h\alpha_i h$ per una qualche descrizione istantanea α_i. Dal Lemma 4.13 (2) si ha che $\alpha_i \to \alpha_{i+1}$ oppure $\alpha_{i+1} \to \alpha_i$. Proviamo per induzione su $t \geq 1$ che $\alpha_i \to \alpha_{i+1}$, per ogni $i \leq t-1$. Per il caso base $t = 1$ è sufficiente osservare che se α_t è terminale, allora l'unico caso possibile è $\alpha_{t-1} \to \alpha_t$. Supponiamo ora che $t > 1$ e che

alcune frecce siano orientate verso sinistra; a partire da $\alpha_{t-1} \to \alpha_t$ scorriamo la sequenza di descrizioni istantanee finché non troviamo un indice i tale che $\alpha_{i-1} \leftarrow \alpha_i \to \alpha_{i+1}$. Ciò non è possibile perché non c'è mai ambiguità sulla mossa successiva di una macchina di Turing, dunque deve essere $\alpha_{i-1} \equiv \alpha_{i+1}$ e $\omega_{i-1} \equiv h\alpha_{i-1}h \equiv h\alpha_{i+1}h \equiv \omega_{i+1}$. Eliminiamo le parole ω_i e ω_{i+1}, così da poter applicare il passo induttivo alla sequenza di lunghezza t. Abbiamo quindi dimostrato che $q_0\omega \to \alpha_1 \to \cdots \to \alpha_t$ è una sequenza di descrizioni istantanee, con α_t che contiene lo stato finale q_N; dunque T calcola ω e $\omega \in E$. □

Teorema 4.15 (**Markov-Post**)**.**

1. *Esiste un semigruppo finitamente presentato*

$$\gamma = (q, h, S_0, S_1, \ldots, S_M, q_0, q_1, \ldots, q_N \mid R),$$

con problema della parola indecidibile.

2. *Non esiste alcun processo decisionale che determini, per una arbitraria parola h-speciale $h\alpha h$, se $h\alpha h = q$ in γ.*

Dimostrazione. 1. Sia T una macchina di Turing con stato finale q_N ed alfabeto $A = \{S_0, S_1, \ldots, S_M\}$, sia Ω l'insieme della parole positive in A e sia $E = e(T) \subset \Omega$. Denotiamo con $\overline{\Omega}$ l'insieme della parole positive in $A \cup \{q, h, q_0, q_1, \ldots, q_N\}$, dove i simboli $q_0, q_1, \ldots, q_N$ appaiono nella quadruple di T. Definiamo l'insieme

$$\overline{E} = \{\overline{\omega} \in \overline{\Omega} : \overline{\omega} = q \text{ in } \Gamma(T)\},$$

e la funzione $\varphi : \Omega \to \overline{\Omega}$ tale che $\omega \mapsto hq_0\omega h$. Identifichiamo Ω con la sua immagine $\Omega_0 \subset \overline{\Omega}$; dunque il sottoinsieme E di Ω si identifica con il sottoinsieme $E_0 = \{hq_0\omega h : \omega \in E\}$ di Ω_0. Sfruttando il Lemma 4.13 (2) e le relazioni da (4.1) a (4.5) è immediato osservare che E_0 è un sottoinsieme ricorsivo di Ω_0 se e solo se E è un sottoinsieme ricorsivo di Ω. Dal Lemma 4.14 segue che

$$E_0 = \{hq_0\omega h : \omega \in E\} = \{hq_0\omega h = q \text{ in } \Gamma(T)\} = \overline{E} \cap \Omega_0.$$

Assumiamo che T sia la macchina di Turing T^* del Teorema 4.4, dunque $E = e(T^*)$ è r.e., ma non ricorsivo. Supponiamo per assurdo che $\overline{E}$ sia ricorsivo; dall'Osservazione 4.5 segue che E_0 è ricorsivo, dunque E è ricorsivo, che è una contraddizione. Sia $\gamma = \Gamma(T^*)$ il semigruppo finitamente presentato associato a T^*; abbiamo dimostrato che esiste q in $\overline{\Omega}$ tale che l'insieme $\overline{E} = \{\overline{\omega} \in \overline{\Omega} : \overline{\omega} = q \text{ in } \Gamma(T)\}$ non è ricorsivo; quindi per definizione γ ha problema della parola indecidibile.

2. Definiamo l'insieme $\overline{S} = \{h\alpha h : h\alpha h = q \text{ in } \Gamma(T^*)\}$. Supponiamo per assurdo che $\overline{S}$ sia ricorsivo; allora per l'Osservazione 4.5 si ha che $\overline{S} \cap \Omega_0$ è un sottoinsieme ricorsivo di Ω_0. Ciò è assurdo, perché $\overline{S} \cap \Omega_0 = E_0$, che sappiamo non essere ricorsivo dal punto precedente. □

Con il seguente corollario riscriviamo generatori e relazioni del semigruppo $\gamma = \Gamma(T^*)$.

Corollario 4.16.

1. *Esiste un semigruppo finitamente presentato*

$$\Gamma = (q, q_0, q_1, \ldots, q_N, S_0, S_1, \ldots, S_M \mid F_i q_{i_1} G_i = H_i q_{i_2} K_i, i \in I),$$

con problema della parola indecidibile; dove F_i, G_i, H_i, K_i sono parole positive (anche vuote) in $S = \{S_0, S_1, \ldots, S_M\}$ e $q_{i_1}, q_{i_2} \in \{q, q_0, \ldots, q_N\}$.

2. *Non esiste alcun processo decisionale che determini se $Xq_{i_j}Y = q$ in Γ, per X ed Y parole positive in $S = \{S_0, S_1, \ldots, S_M\}$ e q_{i_j} arbitrario.*

Dimostrazione. 1. È sufficiente considerare il generatore h del semigruppo $\gamma = \Gamma(T^*)$ ed aggiornare gli indici dei simboli $S_0, \ldots, S_M$ in modo tale che $h = S_M$; in questo modo le relazioni $R(T^*)$ assumono la forma richiesta.

2. Sia $\{q, q_0, \ldots, q_N, S_0, \ldots, S_M\}$ l'insieme dei generatori di Γ, con gli indici assegnati come nel punto precedente e sia Ω_2 l'insieme delle parole positive in tale insieme. Definiamo i seguenti due insiemi:

$$\overline{\Lambda} = \{Xq_{i_j}Y : X, Y \text{ sono parole positive in } S_0, \ldots, S_M \text{ e } Xq_{i_j}Y = q \text{ in } \Gamma\};$$
$$\overline{S}_2 = \{S_M \alpha S_M : \alpha \equiv \sigma q_{i_j} \tau, \text{ dove } \sigma \text{ e } \tau \text{ sono parole positive in } S_0, \ldots, S_{M-1} \text{ e } S_M \alpha S_M = q \text{ in } \Gamma\}.$$

L'insieme $\overline{S}_2$ si ottiene dal sottoinsieme $\overline{S}$ del teorema precedente ponendo $h = S_M$; inoltre, vale $\overline{S}_2 = \overline{\Lambda} \cap \Omega_2$. Poiché $\overline{S}_2$ non è ricorsivo, per l'Osservazione 4.5 possiamo concludere che anche $\overline{\Lambda}$ non è ricorsivo; da cui la tesi. □

4.3 Il problema della parola per i gruppi

Nel presente paragrafo affrontiamo il problema della parola per i gruppi: analogamente al caso dei semigruppi, consideriamo una macchina di Turing T e definiamo il gruppo $\mathscr{B}(T)$ associato. Dimostreremo che $\mathscr{B}$ ha problema della parola non decidibile se T è scelta come T^*, che è la macchina di Turing del teorema di Markov-Post.

4.3.1 Teorema di Novikov-Boone-Britton

Osservazione 4.17. Dato un semigruppo Γ con generatori $X = \{x_1, \dots, x_n\}$; nel paragrafo precedente si è considerato l'insieme Ω delle parole positive in X. Nel caso dei gruppi non è necessario considerare solo parole positive, infatti: se F è un gruppo libero con generatori $X = \{x_1, \dots, x_n\}$, si può considerare l'insieme Ω di tutte le parole in X come l'insieme delle parole positive nell'alfabeto $X = \{x_1, x_1^{-1}, \dots, x_n, x_n^{-1}\}$.

Definizione 4.18. *Sia G un gruppo con presentazione $(x_1, \dots, x_n \mid \Delta)$ e sia Ω l'insieme delle parole in $\{x_1, \dots, x_n\}$. Si dice che il gruppo G ha* ***problema della parola decidibile*** *se l'insieme $\{\omega \in \Omega : \omega = 1 \text{ in } G\}$ è ricorsivo.*

Consideriamo una macchina di Turing T ed il semigruppo $\Gamma = \Gamma(T)$ associato, scritto secondo la notazione del Corollario 4.16:

$$\Gamma = (q, q_0, q_1, \dots, q_N, S_0, S_1, \dots, S_M \mid F_i q_{i_1} G_i = H_i q_{i_2} K_i, i \in I),$$

dove F_i, G_i, H_i, K_i sono parole positive (anche vuote) in $S = \{S_0, S_1, \dots, S_M\}$ e $q_{i_1}, q_{i_2} \in \{q, q_0, \dots, q_N\}$. Presentiamo la definizione equivalente per il caso dei gruppi:

Definizione 4.19. *Sia T una macchina di Turing; si definisce $\mathscr{B} = \mathscr{B}(T)$ il* ***gruppo associato*** *a T avente generatori:*

$$q, q_0, \dots, q_N, S_0, \dots, S_M, r_i, i \in I, x, t, k;$$

e relazioni:

per ogni $i \in I$ e per ogni $\beta = 0, \dots, M$,

$$\left.\begin{array}{ll} xS_\beta = S_\beta x^2, & \Delta_1 \\ r_i S_\beta = S_\beta x r_i x, & \\ r_i^{-1} F_i^{\#} q_{i_1} G_i r_i = H_i^{\#} q_{i_2} K_i, & \Delta_2 \\ tr_i = r_i t, & \\ tx = xt, & \Delta_3 \end{array}\right]$$

$$kr_i = r_i k,$$

$$kx = xk,$$

$$k(q^{-1}tq) = (q^{-1}tq)k.$$

Consideriamo l'insieme $S = \{S_0, S_1, \ldots, S_M\}$ e una parola $X \equiv S_{\beta_1}^{e_1} \ldots S_{\beta_m}^{e_m}$ (non necessariamente positiva) in S; definiamo $X^{\#} \equiv S_{\beta_1}^{-e_1} \ldots S_{\beta_m}^{-e_m}$. Valgono le seguenti due proprietà: $(X^{\#})^{\#} \equiv X$ e $(XY)^{\#} \equiv X^{\#}Y^{\#}$.

Definizione 4.20. *Siano X e Y due parole in S; si definisce $(Xq_jY)^* \equiv X^{\#}q_jY$, dove $q_j \in \{q, q_0, \ldots, q_N\}$.*

Definizione 4.21. *Una parola Σ è **speciale** se $\Sigma \equiv X^{\#}q_jY$, dove X e Y sono parole positive in S e $q_j \in \{q, q_0, \ldots, q_N\}$.*

Osservazione 4.22. Se $\Sigma \equiv X^{\#}q_jY$, dove X e Y sono parole positive in S, allora $\Sigma^* \equiv (X^{\#}q_jY)^* \equiv Xq_jY$ è una parola positiva; in particolare, Σ^* costituisce un elemento del semigruppo Γ.

Le definizioni introdotte ci permettono di enunciare i due risultati fondamentali del presente capitolo:

Lemma 4.23 (**Boone**). *Sia T una macchina di Turing con stato finale q_N e sia $\Gamma = \Gamma(T)$ il suo sottogruppo associato (presentato come nel Corollario 4.16). Se Σ è una parola speciale, allora*

$$k(\Sigma^{-1}t\Sigma) = (\Sigma^{-1}t\Sigma)k \text{ in } \mathscr{B} = \mathscr{B}(T) \text{ se e solo se } \Sigma^* = q \text{ in } \Gamma(T).$$

Teorema 4.24 (**Novikov-Boone-Britton**). *Esiste un gruppo $\mathscr{B}$ finitamente presentato con problema della parola indecidibile.*

Dimostrazione. Consideriamo la macchina di Turing T^* del Teorema di Markov-Post. Procediamo per assurdo: supponiamo che esista un processo decisionale che, per una qualsiasi parola speciale Σ, determini se $k\Sigma^{-1}t\Sigma k^{-1}\Sigma^{-1}t^{-1}\Sigma = 1$ in $\mathscr{B}(T^*)$, allora dal lemma di Boone segue che lo stesso processo decisionale determina se $\Sigma^* = q$ in $\Gamma(T^*)$. Dall'Osservazione 4.22 sappiamo che $\Sigma^* \equiv Xq_jY$, dove X e Y sono parole positive in S; possiamo, quindi, applicare il Corollario 4.16 (2), che afferma che non esiste un processo decisionale per $\Gamma(T^*)$ come richiesto. □

Corollario 4.25. *Sia T una macchina di Turing con stato finale q_N, Ω l'insieme di tutte le parole positive nell'alfabeto di T e sia $E = e(T) \subset \Omega$. Se $\omega \in \Omega$, allora $\omega \in E$ se e solo se $k(h^{-1}q_0\omega h) = (h^{-1}q_0\omega h)k$ in $\mathscr{B}(T)$.*

Dimostrazione. Sia $\omega \in \Omega$; il Lemma 4.14 afferma che $\omega \in E$ se e solo se $hq_0\omega h = q$ in $\Gamma(T)$. In $\mathscr{B}(T)$ vale $(hq_0\omega h)^* = h^{-1}q_0\omega h$, che è una parola speciale. Applicando il Lemma di Boone a quest'ultima otteniamo che $(h^{-1}q_0\omega h)^* \equiv hq_0\omega h = q$ in $\Gamma(T)$ se e solo se $k(h^{-1}q_0\omega h) = (h^{-1}q_0\omega h)k$ in $\mathscr{B}(T)$. □

4.3.2 Lemma di Boone: dimostrazione della condizione sufficiente

Per completare la trattazione del problema della parola per i gruppi, rimane da dimostrare il lemma di Boone. In questo sottoparagrafo trattiamo la condizione sufficiente; nel successivo presenteremo la dimostrazione della condizione necessaria.

Riportiamo di seguito il lemma di Boone:

Lemma 4.26 (**Boone**). *Sia T una macchina di Turing con stato finale q_N e sia $\Gamma = \Gamma(T)$ il suo sottogruppo associato (presentato come nel Corollario 4.16). Se Σ è una parola speciale, allora*

$$k(\Sigma^{-1}t\Sigma) = (\Sigma^{-1}t\Sigma)k \text{ in } \mathscr{B} = \mathscr{B}(T) \text{ se e solo se } \Sigma^* = q \text{ in } \Gamma(T).$$

La dimostrazione farà uso del seguente lemma:

Lemma 4.27. *Consideriamo l'insieme $S = \{S_0, S_1, \ldots, S_M\}$.*

1. *Se V è una parola positiva in S, allora $r_iV = VR$ in $\mathscr{B}$ e $r_i^{-1}V = VR'$ in $\mathscr{B}$, dove R e R' sono parole in $\{x, r_i : i \in I\}$ ed R è una parola positiva.*

2. *Se U è una parola positiva in S, allora $U^{\#}r_i = LU^{\#}$ in $\mathscr{B}$ e $U^{\#}r_i^{-1} = L'U^{\#}$ in $\mathscr{B}$, dove L e L' sono parole in $\{x, r_i : i \in I\}$.*

Dimostrazione. Si veda il Lemma 12.10 [9, Cap. 12]. □

Dimostrazione della condizione sufficiente del lemma di Boone. Sia Σ una parola speciale tale che $\Sigma^* \equiv Xq_jY = q$ in Γ; allora esiste una sequenza finita

$$\Sigma^* \equiv \omega_1 \to \omega_2 \to \cdots \to \omega_n \equiv q \text{ in } \Gamma,$$

tale che, per ogni v, $\omega_v \to \omega_{v+1}$ è un'operazione elementare. Una tra ω_v e ω_{v+1} assume la forma $UF_iq_{i_1}G_iV$ e l'altra $UH_iq_{i_2}K_iV$, dove U e V sono parole positive in $S = \{S_0, S_1, \ldots, S_M\}$. Supponiamo che $\omega_v^* \equiv U^{\#}({H_i}^{\#}q_{i_2}K_i)V$; per le relazioni in $\mathscr{B}$ (si veda la Definizione 4.19) e per il lemma precedente, vale la seguente uguaglianza in $\mathscr{B}$

$$\omega_v^* \equiv U^{\#}({H_i}^{\#}q_{i_2}K_i)V = U^{\#}({r_i}^{-1}{F_i}^{\#}q_{i_1}G_ir_i)V = L'U^{\#}({F_i}^{\#}q_{i_1}G_i)VR' \equiv L'\omega_{v+1}^*R',$$

dove L' e R' sono parole in $\{x, r_i : i \in I\}$.

Similmente, se $\omega_v^* \equiv U^{\#}({F_i}^{\#}q_{i_1}G_i)V$, in $\mathscr{B}$ vale

$$\omega_v^* \equiv U^{\#}({F_i}^{\#}q_{i_1}G_i)V = U^{\#}(r_i{H_i}^{\#}q_{i_2}K_i{r_i}^{-1})V = L''U^{\#}({H_i}^{\#}q_{i_2}K_i)VR'' \equiv L''\omega_{v+1}^*R'',$$

dove L'' e R'' sono parole in $\{x, r_i : i \in I\}$.

Dunque, per ogni v vale $\omega_v^* = L_v \omega_{v+1}^* R_v$ in $\mathscr{B}$, dove L_v e R_v sono parole in $\{x, r_i : i \in I\}$. Definiamo le parole $L \equiv L_1 \dots L_{n-1}$ e $R \equiv R_{n-1} \dots R_1$ in $\{x, r_i : i \in I\}$; allora vale $\omega_1^* = L\omega_n^* R$ in $\mathscr{B}$.

Ma $\omega_1^* \equiv (\Sigma^*)^* \equiv \Sigma$ e $\omega_n^* \equiv q^* \equiv q$, dunque $\Sigma = LqR$ in $\mathscr{B}$. Dalla Definizione 4.19 segue che k e t commutano con x e con tutti gli r_i; quindi commutano anche con L e R. Dunque, si ha

$$\begin{aligned} k\Sigma^{-1}t\Sigma k^{-1}\Sigma^{-1}t^{-1}\Sigma &= kR^{-1}q^{-1}L^{-1}tLqRk^{-1}R^{-1}q^{-1}L^{-1}t^{-1}LqR \\ &= kR^{-1}q^{-1}tqk^{-1}q^{-1}t^{-1}qR \\ &= R^{-1}(kq^{-1}tqk^{-1}q^{-1}t^{-1}q)R \\ &= 1; \end{aligned}$$

l'ultima uguaglianza segue dalla relazione $k(q^{-1}tq) = (q^{-1}tq)k$ della Definizione 4.19. La condizione sufficiente del lemma di Boone è quindi dimostrata. □

4.3.3 Lemma di Boone: dimostrazione della condizione necessaria

La dimostrazione della condizione necessaria del lemma di Boone risulta più complessa rispetto a quella della condizione sufficiente. Per affrontarla, introdurremo nuovi lemmi e definizioni e faremo ampio uso della teoria delle estensioni HNN, presentata nel Paragrafo 1.3.

Ricordiamo che nella Definizione 4.19 le relazioni del gruppo $\mathscr{B}$ sono state raccolte nei sottoinsiemi $\Delta_1 \subset \Delta_2 \subset \Delta_3$; utilizziamo quest'ultimi per definire i gruppi $\mathscr{B}_0, \mathscr{B}_1, \mathscr{B}_2, \mathscr{B}_3$:

$$\mathscr{B}_0 = (x \mid \varnothing),$$

cioè $\mathscr{B}_0 = \langle x \rangle$, che è il gruppo ciclico infinito con generatore x;

$$\begin{aligned} \mathscr{B}_1 &= (\mathscr{B}_0; S_0, \dots, S_M \mid \Delta_1), \\ \mathscr{B}_2 &= (\mathscr{B}_1 * Q; r_i, i \in I \mid \Delta_2), \end{aligned}$$

dove $Q = \langle q, q_0, \dots, q_N \rangle$, cioè il gruppo libero con base $\{q, q_0, \dots, q_N\}$; ed infine

$$\mathscr{B}_3 = (\mathscr{B}_2; t \mid \Delta_3).$$

Lemma 4.28. *Ogni gruppo nella catena $\mathscr{B}_0 \leq \mathscr{B}_1 \leq \mathscr{B}_1 * Q \leq \mathscr{B}_2 \leq \mathscr{B}_3 \leq \mathscr{B}$ è un'estensione HNN del suo predecessore e $\mathscr{B}_1 * Q$ è un'estensione HNN di $\mathscr{B}_0$. In particolare:*

1. *$\mathscr{B}_1$ è un'estensione HNN con base $\mathscr{B}_0$ e lettere stabili $\{S_0, \ldots, S_M\}$;*
2. *$\mathscr{B}_1 * Q$ è un'estensione HNN con base $\mathscr{B}_0$ e lettere stabili $\{S_0, \ldots, S_M\} \cup \{q, q_0, \ldots, q_N\}$;*
3. *$\mathscr{B}_1 * Q$ è un'estensione HNN con base $\mathscr{B}_1$ e lettere stabili $\{q, q_0, \ldots, q_N\}$;*
4. *$\mathscr{B}_2$ è un'estensione HNN con base $\mathscr{B}_1 * Q$ e lettere stabili $\{r_i : i \in I\}$;*
5. *$\mathscr{B}_3$ è un'estensione HNN con base $\mathscr{B}_2$ e lettera stabile t;*
6. *$\mathscr{B}$ è un'estensione HNN con base $\mathscr{B}_3$ e lettera stabile k.*

Dimostrazione. Si veda il Lemma 12.11 [9, Cap. 12]. □

Corollario 4.29.

1. *Il sottogruppo $\langle S_1 x, \ldots, S_M x \rangle \leq \mathscr{B}_1$ è un gruppo libero con base $\{S_1 x, \ldots, S_M x\}$.*
2. *Esiste un automorfismo ψ di $\mathscr{B}_1$ tale che $\psi(x) = x^{-1}$ e $\psi(S_\beta) = S_\beta$, per ogni β.*

Dimostrazione. Si veda il Corollario 12.12 [9, Cap. 12]. □

Lemma 4.30. *Si consideri una parola speciale Σ che soddisfi le ipotesi del lemma di Boone:*

$$w(\Sigma) \equiv k\Sigma^{-1}t\Sigma k^{-1}\Sigma^{-1}t^{-1}\Sigma = 1 \text{ in } \mathscr{B}.$$

Allora esistono L_1 e L_2 parole ridotte in $\{x, r_i : i \in I\}$ tali che

$$L_1 \Sigma L_2 = q \text{ in } \mathscr{B}_2.$$

Osservazione 4.31. Dalle relazioni nella Definizione 4.19 si ha che k commuta con $\{x, q^{-1}tq, r_i : i \in I\}$ e t commuta con $\{x, r_i : i \in I\}$.

Inoltre, dal Lemma 4.28 sappiamo che $\mathscr{B}$ è un'estensione HNN con base $\mathscr{B}_3$ e lettera stabile k, cioè

$$\mathscr{B} = (\mathscr{B}_3; k \mid k^{-1}r_i k = r_i, i \in I, k^{-1}xk = x, k^{-1}(q^{-1}tq)k = q^{-1}tq);$$

$\mathscr{B}_3$ è un'estensione HNN con base $\mathscr{B}_2$ e lettera stabile t, cioè

$$\mathscr{B}_3 = (\mathscr{B}_2; t \mid t^{-1}r_i t = r_i, i \in I, t^{-1}xt = x).$$

Dimostrazione. Sappiamo che $\mathscr{B}$ è un'estensione HNN con base $\mathscr{B}_3$ e lettera stabile k. Consideriamo la parola $k\Sigma^{-1}t\Sigma k^{-1}\Sigma^{-1}t^{-1}\Sigma$; per il Lemma 1.33 di Britton, questa contiene un *pinch* della forma $k\Sigma^{-1}t\Sigma k^{-1}$; inoltre, esiste una parola C in $\{x, q^{-1}tq, r_i :$

$i \in I\}$ tale che $\Sigma^{-1}t\Sigma = C$ in $\mathscr{B}_3$. Perciò esistono parole ω che soddisfano $\Sigma^{-1}t\Sigma C^{-1} = 1$ in $\mathscr{B}_3$, cioè

$$\omega \equiv \Sigma^{-1}t\Sigma R_0(q^{-1}t^{e_1}q)R_1(q^{-1}t^{e_2}q)R_2 \dots (q^{-1}t^{e_n}q)R_n = 1 \text{ in } \mathscr{B}_3,$$

dove R_j sono parole ridotte (anche vuote) in $\{x, r_i : i \in I\}$ e $e_j = \pm 1$. Supponiamo che ω sia scelta in modo tale che n sia minimale.

Poiché $\mathscr{B}_3$ è un'estensione HNN con base $\mathscr{B}_2$ e lettera stabile t; dal Lemma 1.33 di Britton segue che ω contiene un *pinch* della forma $t^e D t^{-e}$ ed esiste una parola R in $\{x, r_i : i \in I\}$ tale che $D = R$ in $\mathscr{B}_2$.

Supponiamo che la sottoparola $t^e D t^{-e}$ contenga la prima occorrenza della lettera t in ω, cioè $t^e D t^{-e} \equiv t\Sigma R_0 q^{-1} t^{e_1}$. Allora $e = +1, e_1 = -1$ e $t\Sigma R_0 q^{-1} t^{e_1} = tRt^{-1}$; dunque vale

$$\Sigma R_0 q^{-1} = R \text{ in } \mathscr{B}_2;$$

equivalentemente,

$$R^{-1}\Sigma R_0 = q \text{ in } \mathscr{B}_2,$$

dove R^{-1} e R_0 sono parole ridotte in $\{x, r_i : i \in I\}$, come richiesto dal lemma.

Supponiamo ora che t^e sia t^{e_j}, per $j \geq 1$; allora $t^e D t^{-e} \equiv t^{e_j} q R_j q^{-1} t^{e_{j+1}}$ ed esiste una parola R in $\{x, r_i : i \in I\}$ tale che $qR_jq^{-1} = R$ in $\mathscr{B}_2$. Dall'Osservazione 1.32 si ha che $\mathscr{B}_2 \leq \mathscr{B}_3$, dunque vale la seguente uguaglianza in $\mathscr{B}_3$:

$$t^{e_j} q R_j q^{-1} t^{e_{j+1}} = t^e q R_j q^{-1} t^{-e} = t^e R t^{-e}. \tag{4.9}$$

Poiché la lettera stabile t commuta con x e r_i, per ogni $i \in I$, si ha

$$qR_jq^{-1} = R \text{ in } \mathscr{B}_3. \tag{4.10}$$

Dunque, in $\mathscr{B}_3$ vale

$$\begin{aligned}
(q^{-1}t^{e_j}q)R_j(q^{-1}t^{e_{j+1}}q) &= q^{-1}t^e R t^{-e} q && (4.9)\\
&= q^{-1}Rq && (t \text{ commuta con } x, r_i)\\
&= q^{-1}(qR_jq^{-1})q && (4.10)\\
&= R_j.
\end{aligned}$$

Ma questa nuova fattorizzazione di ω in $\mathscr{B}_3$ contraddice l'ipotesi di minimalità di n. □

Ricordiamo che una parola speciale Σ è della forma $\Sigma \equiv X^{\#} q_j Y$, dove X e Y sono parole positive in $\{S_0, \dots, S_M\}$ e $q_j \in \{q, q_0, \dots, q_N\}$. Con il lemma precedente si è provato che $L_1 X^{\#} q_j Y L_2 = q$ in $\mathscr{B}_2$, dove L_1 e L_2 sono parole ridotte in $\{x, r_i : i \in I\}$; tale uguaglianza può essere riscritta come $L_1 X^{\#} q_j = q L_2^{-1} Y^{-1}$ in $\mathscr{B}_2$.

Lemma 4.32. *Le parole $L_1X^{\#}q_j$ e $qL_2^{-1}Y^{-1}$ sono r_i-ridotte per ogni $i \in I$.*

Osservazione 4.33. Dal Lemma 4.28 sappiamo che $\mathscr{B}_2$ è un'estensione HNN con base $\mathscr{B}_1 * Q$ e lettere stabili $\{r_i : i \in I\}$, cioè

$$\mathscr{B}_2 = (\mathscr{B}_1 * Q; r_i, i \in I \mid r_i^{-1}(F_i^{\#}q_{i_1}G_i)r_i = H_i^{\#}q_{i_2}K_i, r_i^{-1}(S_\beta x)r_i = S_\beta x^{-1}),$$

dove $Q = \langle q, q_0, \ldots, q_N \rangle$.

Dimostrazione. Dimostriamo che la parola $L_1X^{\#}q_j$ è r_i-ridotta; la dimostrazione per $qL_2^{-1}Y^{-1}$ è analoga.

Supponiamo per assurdo che $L_1X^{\#}q_j$ contenga una sottoparola della forma $r_k^e C r_k^{-e}$; questa è una sottoparola di L_1, essendo $X^{\#}$ una parola in $\{S_0, \ldots, S_M\}$. Ricordiamo che L_1 è una parola ridotta in $\{x, r_i : i \in I\}$; quindi esiste $m \neq 0$ tale che $C \equiv x^m$. Dall'Osservazione 4.33 e dalla definizione di *pinch* si ha che esiste una parola V in $\mathscr{B}_1 * Q$, dove $Q = \langle q, q_1, \ldots, q_N \rangle$, tale che

$$x^m = V \text{ in } \mathscr{B}_1 * Q.$$

La parola V assume la forma

$$V \equiv \omega_0 (F_i^{\#}q_{i_1}G_i)^{e_1} \omega_1 \ldots (F_i^{\#}q_{i_1}G_i)^{e_n} \omega_n,$$

dove $e_j = \pm 1$ e ω_j è una parola in $\{S_1 x, \ldots, S_M x\}$, per ogni j.

Poiché $x^m = V$ in $\mathscr{B}_1 * Q$ e $x^m \in \mathscr{B}_1$, possiamo concludere che V non contiene nessuno dei simboli $\{q, q_1, \ldots, q_N\}$; quindi V non contiene $F_i^{\#}q_{i_1}G_i$, da cui

$$x^m = \omega_0 \equiv (S_{\beta_1}x)^{f_1} \ldots (S_{\beta_p}x)^{f_p} \text{ in } \mathscr{B}_1,$$

dove $f_v = \pm 1$ e vale

$$x^{-m}\omega_0 = 1 \text{ in } \mathscr{B}_1.$$

Ricordiamo che per il Lemma 4.28, $\mathscr{B}_1$ è un'estensione HNN con base $\mathscr{B}_0 = \langle x \rangle$ e lettere stabili $\{S_0, \ldots, S_M\}$ e grazie al Lemma 1.33 di Britton possiamo affermare che la parola $x^{-m}\omega_0$ contiene una sottoparola della forma $S_\beta^f x^\epsilon S_\beta^{-f}$, dove $\epsilon = \pm 1$. È sufficiente osservare lo *spelling* di ω_0 per concludere che non contiene nessuna sottoparola della forma richiesta; quindi $\omega_0 = 1$, da cui $x^m = 1$. Ma questo genera un assurdo, poiché il fatto che x abbia ordine infinito contraddice la richiesta $m \neq 0$. Perciò L_1 è r_i-ridotta, da cui segue che $L_1X^{\#}q_j$ è r_i-ridotta. □

Lemma 4.34. *Siano L_1 e L_2 parole in $\{x, r_i : i \in I\}, r_i$-ridotte per qualunque $i \in I$. Se X e Y sono parole ridotte in $\{S_0, \ldots, S_M\}$ e se*

$$L_1 X^{\#} q_j Y L_2 = q \text{ in } \mathscr{B}_2,$$

allora X e Y sono positive e vale

$$X q_j Y \equiv (X^{\#} q_j Y)^* = q \text{ in } \Gamma.$$

Osservazione 4.35. Il lemma appena enunciato conclude la dimostrazione del Lemma 4.23 di Boone e di conseguenza, del Teorema 4.24 di Novikov-Boone-Britton. Consideriamo, infatti, una parola speciale Σ della forma $\Sigma \equiv X^{\#} q_j Y$, dove X e Y sono parole positive in $S = \{S_0, \ldots, S_M\}$ e $q_j \in \{q, q_0, \ldots, q_N\}$. Supponiamo che Σ soddisfi le ipotesi del lemma di Boone, cioè

$$w(\Sigma) \equiv k\Sigma^{-1} t \Sigma k^{-1} \Sigma^{-1} t^{-1} \Sigma = 1 \text{ in } \mathscr{B}.$$

Dal Lemma 4.30 si ha che

$$L_1 X^{\#} q_j Y L_2 = q \text{ in } \mathscr{B}_2,$$

dove L_1 e L_2 sono parole ridotte e r_i-ridotte (si veda dimostrazione del Lemma 4.32) in $\{x, r_i : i \in I\}$. Poiché X e Y sono parole positive in S, sono anche parole ridotte; quindi le ipotesi del lemma precedente sono soddisfatte. Questo implica che

$$X q_j Y \equiv (X^{\#} q_j Y)^* \equiv \Sigma^* = q \text{ in } \Gamma,$$

che è la tesi del lemma di Boone.

Dimostrazione. Dai lemmi precedenti segue che $L_1 X^{\#} q_j = q L_2^{-1} Y^{-1}$ in $\mathscr{B}_2$ e che entrambe le parole sono r_i-ridotte. Dal Corollario 1.36 si ha che il numero delle lettere r_i è lo stesso in L_1 e L_2; indichiamo tale numero con $\rho \geq 0$. La dimostrazione procede per induzione su ρ.

Sia $\rho = 0$. L'uguaglianza $L_1 X^{\#} q_j Y L_2 = q$ diventa $x^m X^{\#} q_j Y x^n = q$ in $\mathscr{B}_2$. Poiché essa non coinvolge nessun simbolo r_i, può essere vista come un'equazione in $\mathscr{B}_1 * Q \leq \mathscr{B}_2$, dove $Q = \langle q, q_0, \ldots, q_N \rangle$. Dal Teorema 1.26 (forma normale nel prodotto libero) segue che $q_j = q$ e $x^m X^{\#} = 1 = Y x^n$ in $\mathscr{B}_1$. Poiché $\mathscr{B}_1$ è un'estensione HNN con base $\mathscr{B}_0 = \langle x \rangle$ e lettere stabili $\{S_0, \ldots, S_M\}$, dal Lemma 1.33 di Britton si ha che $m = 0 = n$ e che X e Y sono entrambe la parola vuota. Quindi X e Y sono positive e $X q_j Y \equiv q_j \equiv q$ in Γ.

Sia $\rho > 0$. Ricordiamo che dal Lemma 4.32 le parole $L_1 X^{\#} q_j$ e $q L_2^{-1} Y^{-1}$ sono r_i-ridotte per ogni $i \in I$ e che $\mathscr{B}_2$ è un'estensione HNN con base $\mathscr{B}_1 * Q$ e lettere stabili $\{r_i : i \in I\}$. Dunque, esistono L_3 e L_4 sottoparole, rispettivamente, di L_1 e L_2 tali che

$$L_1 X^{\#} q_j Y L_2 \equiv L_3 (r_i^e x^m X^{\#} q_j Y x^n r_i^{-e}) L_4 = q \text{ in } \mathscr{B}_2, \tag{4.11}$$

dove la parola tra parentesi costituisce un *pinch*. Si hanno due casi possibili:

1. se $e = -1$, allora $x^m X^{\#} q_j Y x^n \in A_i = \langle F_i^{\#} q_{i_1} G_i, S_\beta x, \text{ per ogni } \beta \rangle$ e quindi $q_j = q_{i_1}$;
2. se $e = +1$, allora $x^m X^{\#} q_j Y x^n \in B_i = \langle H_i^{\#} q_{i_2} K_i, S_\beta x^{-1}, \text{ per ogni } \beta \rangle$ e quindi $q_j = q_{i_2}$.

Consideriamo solo il caso $e = -1$; il secondo caso è analogo.

Per quanto appena detto si ha che esiste una parola ω tale che

$$\omega \equiv x^m X^{\#} q_j Y x^n u_0 (F_i^{\#} q_j G_i)^{\alpha_1} u_1 \dots (F_i^{\#} q_j G_i)^{\alpha_t} u_t = 1 \text{ in } \mathscr{B}_1 * Q,$$

dove $\alpha_j = \pm 1$ e u_j sono parole (anche vuote) in $\{S_\beta x, \text{ per ogni } \beta\}$. Possiamo assumere che le u_j siano ridotte in $\{S_\beta x, \text{ per ogni } \beta\}$ e per il Corollario 4.29 (1) sappiamo che tale insieme genera un sottogruppo libero di $\mathscr{B}_1$. Supponiamo che ω sia stata scelta con t minimale. Poiché $\omega = 1$ in $\mathscr{B}_1 * Q$, dal Teorema 1.26 (forma normale nel prodotto libero) segue che ogni sottoparola di ω compresa tra due simboli q_j successivi è uguale a 1 in $\mathscr{B}_1$. Inoltre, $\mathscr{B}_1 * Q$ è un'estensione HNN con base $\mathscr{B}_1$ e lettere stabili $\{q, q_0, \dots, q_N\}$ e per l'Osservazione 1.31 i sottogruppi A e B della definizione di estensione HNN sono banali. Quindi, per il Lemma 1.33 di Britton, ω contiene una sottoparola della forma $q_j^{\epsilon} C q_j^{-\epsilon}$, dove $C = 1$ in $\mathscr{B}_1$.

Supponiamo che $q_j^{\epsilon} C q_j^{-\epsilon}$ contenga la prima occorrenza di q_j; allora $-\epsilon = \alpha_1 = -1$ e

$$C \equiv Y x^n u_0 {G_i}^{-1} = 1 \text{ in } \mathscr{B}_1. \tag{4.12}$$

Proviamo che il *pinch* $q_j^{\epsilon} C q_j^{-\epsilon}$ non può apparire in nessun'altra posizione in ω, infatti: consideriamo un indice v e la sottoparola $(F_i^{\#} q_j G_i)^{\alpha_v} u_v (F_i^{\#} q_j G_i)^{\alpha_{v+1}}$. Supponiamo che questa contenga un *pinch*; quindi distinguiamo i due casi possibili:

1. se $\alpha_v = +1$, allora $\alpha_{v+1} = -1$; il *pinch* è $q_j G_i u_v G_i^{-1} q_j^{-1}$ e $C \equiv G_i u_v G_i^{-1} = 1$ in $\mathscr{B}_1$;
2. se $\alpha_v = -1$, allora $\alpha_{v+1} = +1$; il *pinch* è $q_j^{-1} {F_i^{\#}}^{-1} u_v F_i^{\#} q_j$ e $C \equiv {F_i^{\#}}^{-1} u_v F_i^{\#} = 1$ in $\mathscr{B}_1$.

In entrambi i casi si ha $u_v = 1$ in $\mathscr{B}_1$; ma u_v è una parola ridotta in $\{S_\beta x, \text{ per ogni } \beta\}$, quindi $u_v \equiv 1$; ciò contraddice la minimalità di t. Quindi l'unico caso possibile si ha per $t = 1, \alpha_1 = -1$ e

$$\omega \equiv x^m X^{\#} q_j Y x^n u_0 G_i^{-1} q_j^{-1} {F_i^{\#}}^{-1} u_1 = 1 \text{ in } \mathscr{B}_1 * Q.$$

Abbiamo già osservato che ogni sottoparola di ω compresa tra due simboli q_j successivi è uguale a 1 in $\mathscr{B}_1$; quindi si ha ${F_i^{\#}}^{-1} u_1 x^m X^{\#} = 1$ in $\mathscr{B}_1$. Riscriviamo questa uguaglianza

e la (4.12), come

$$\begin{cases} x^n u_0 {G_i}^{-1} Y = 1 & \text{in } \mathscr{B}_1, \\ X^{\#} {F_i^{\#}}^{-1} u_1 x^m = 1 & \text{in } \mathscr{B}_1. \end{cases} \tag{4.13}$$

Consideriamo G_i, che è una parola positiva in $\{S_0, \ldots, S_M\}$ e cancelliamo tutte le sottoparole della forma $S_\beta S_\beta^{-1}$ oppure $S_\beta^{-1} S_\beta$; si può dimostrare che la prima lettera che rimane in $G_i^{-1} Y$ è positiva, cioè esiste una sottoparola Y_1 di Y che inizia con una lettera positiva e tale che $Y \equiv G_i Y_1$. Similmente, data la parola $X^{\#} {F_i^{\#}}^{-1}$; procediamo con la stessa operazione di cancellazione; in questo caso esiste una sottoparola X_1 di X tale che $X_1^{\#}$ termina con una lettera negativa e $X \equiv X_1 F_i$. Non riportiamo la dimostrazione di questi due risultati, ma invitiamo il lettore a consultare la dimostrazione del Lemma 12.15 [9, Cap. 12].

Quindi, in $\mathscr{B}_1$ vale $1 = x^n u_0 {G_i}^{-1} Y = x^n u_0 {G_i}^{-1} G_i Y_1$, da cui $u_0^{-1} = Y_1 x^n$ in $\mathscr{B}_1$. Definiamo inoltre $v_0^{-1} = r_i^{-1} u_0^{-1} r_i$. Poiché u_0 è una parola in $\{S_\beta x, \text{ per ogni } \beta\}$ e dalla Definizione 4.19 vale $r_i^{-1} S_\beta x r_i = S_\beta x^{-1}$ per ogni β, possiamo considerare u_0^{-1} e v_0^{-1} come elementi di $\mathscr{B}_1 = \langle x, S_0, \ldots, S_M \rangle$. Per il Corollario 4.29 (2) esiste un automorfismo ψ di $\mathscr{B}_1$ tale che $\psi(x) = x^{-1}$ e $\psi(S_\beta) = S_\beta$, per ogni β. Quindi, $v_0^{-1} = \psi(u_0^{-1}) = \psi(Y_1 x^n) = Y_1 x^{-n}$; da cui

$$v_0^{-1} = Y_1 x^{-n} \text{ in } \mathscr{B}_1. \tag{4.14}$$

Similmente, possiamo definire $v_1^{-1} = r_i^{-1} u_1^{-1} r_i$, da cui

$$v_1^{-1} = x^{-m} X_1^{\#} \text{ in } \mathscr{B}_1. \tag{4.15}$$

Consideriamo nuovamente l'uguaglianza (4.11), nel caso $e = -1$:

$$L_1 X^{\#} q_j Y L_2 \equiv L_3 (r_i^e x^m X^{\#} q_j Y x^n r_i^{-e}) L_4 = q \text{ in } \mathscr{B}_2. \tag{4.16}$$

In $\mathscr{B}_2$ valgono le seguenti:

$$\begin{aligned} q = L_1 X^{\#} q_j Y L_2 &\equiv L_3 r_i^{-1} (x^m X^{\#}) q_j (Y x^n) r_i L_4 \\ &= L_3 r_i^{-1} (u_1^{-1} F_i^{\#}) q_j (G_i u_0^{-1}) r_i L_4 && (4.13) \\ &= L_3 v_1^{-1} r_i^{-1} (F_i^{\#} q_j G_i) r_i v_0^{-1} L_4 \\ &= L_3 v_1^{-1} (r_i^{-1} F_i^{\#} q_j G_i r_i) v_0^{-1} L_4 \\ &= (L_3 v_1^{-1}) H_i^{\#} q_{i_2} K_i (v_0^{-1} L_4) \\ &= (L_3 x^{-m} X_1^{\#}) H_i^{\#} q_{i_2} K_i (Y_1 x^{-n} L_4) && (4.14), (4.15). \end{aligned}$$

Perciò,

$$L_3 x^{-m} (X_1^{\#} H_i^{\#} q_{i_2} K_i Y_1) x^{-n} L_4 = q \text{ in } \mathscr{B}_2.$$

Osserviamo che $L_3 x^{-m}$ e $x^{-n} L_4$ sono parole in $\{x, r_i : i \in I\}$ aventi $\rho - 1$ occorrenze delle lettere r_i. Inoltre, $X_1^{\#} H_i^{\#}$ e $K_i Y_1$ sono parole ridotte, cioè non contengono sottoparole della forma $S_\beta S_\beta^{-1}$ oppure $S_\beta^{-1} S_\beta$. Infatti, K_i è una parola positiva in $\{S_0, \ldots, S_M\}$, quindi è ridotta; Y_1 è una sottoparola di Y, che è ridotta per ipotesi. Quindi la parola $K_i Y_1$ risultante dalla giustapposizione è ridotta, perché, come abbiamo già osservato, la prima lettera di Y_1 è positiva. Similmente, si può dimostrare che $X_1^{\#} H_i^{\#}$ è ridotta. L'ipotesi induttiva è quindi soddisfatta; per induzione, segue che $X_1 H_i$ e $K_i Y_1$ sono positive e quindi anche le loro sottoparole X_1 e Y_1 e vale

$$\left(X_1^{\#} H_i^{\#} q_{i_2} K_i Y_1\right)^{*} = q \text{ in } \Gamma.$$

Poiché $\left(X_1^{\#} H_i^{\#}\right)^{\#} = X_1 H_i$, si ha

$$X_1 H_i q_{i_2} K_i Y_1 = q \text{ in } \Gamma. \tag{4.17}$$

Ricordiamo che $e = -1, q_j = q_{i_1}$ e che $X \equiv X_1 F_i$ e $Y \equiv G_i Y_1$ sono parole positive; si ha

$$\begin{aligned} X q_j Y &\equiv X_1 F_i q_j G_i Y_1 \\ &\equiv X_1 F_i q_{i_1} G_i Y_1 \\ &= X_1 H_i q_{i_2} K_i Y_1 \qquad \text{(Corollario 4.16(1))} \\ &= q \text{ in } \Gamma \qquad\qquad (4.17), \end{aligned}$$

da cui segue la tesi

$$X q_j Y = q \text{ in } \Gamma. \qquad \square$$

La dimostrazione del Lemma 4.23 di Boone è così conclusa e con essa la nostra trattazione del problema della parola per i gruppi.

Bibliografia

[1] N. Cutland: "Computability: An Introduction to Recursive Function Theory". Cambridge University Press: Cambridge, 1980.

[2] M. Davis: "Computability and Unsolvability". McGraw-Hill Book Company: United States of America, 1958.

[3] F. Frisoni: "Algoritmi e Ricorsività: Macchine di Turing e Algoritmi di Markov", Tesi di Laurea, Anno Accademico 2013/2014. Disponibile all'url

`https://amslaurea.unibo.it/id/eprint/7247/1/frisoni_federica_tesi.pdf`

[4] T.W. Hungerford: "Algebra". Springer-Verlag: New York, 1974.

[5] H.R. Lewis, C.H. Papadimitriou: "Elements of the Theory of Computation". Second edition. Prentice-Hall: New Jersey, 1998.

[6] A.A. Markov: "Theory of Algorithms". Third impression. IPST Press: Jerusalem, 1968.

[7] E. Mendelson: "Introduzione alla Logica Matematica". Quinta ristampa. Bollati Boringhieri: Torino, 1987.

[8] A. Pettorossi: "Elements of Computability, Decidability, and Complexity", Dispensa. Disponibile all'url

`https://art.torvergata.it/retrieve/e291c0d4-b851-cddb-e053-3a05fe0aa144/ComputabilityDecidabilityComplexity_FourthEd_2014.pdf`

[9] J.J. Rotman: "An Introduction to the Theory of Groups". Fourth edition. Springer New York: NY, 1995.

[10] M. Sipser: "Introduction to the Theory of Computation". Second edition. Thomson Course Technology: United States of America, 2006.

[11] A.M. Turing: "On Computable Numbers, with an Application to the Entscheidungsproblem", pp. 230-265. Princeton University, New Jersey. Disponibile all'url

`https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf`