Dispense per il corso di

Teoria della Computabilità

- Modulo di Complessità -

A cura del Dr. Salvatore Cristofaro
cristofaro@dmi.unict.it

Libri di testo di riferimento:

(1) *Computability, Complexity and Languages*, Martin D. Davis, Ron Sigal, Elaine J. Weyuker

(2) *Introduction to the Theory of Computation*, Michael Sipser

(3) *Computers and Intractability*, Michael R. Garey, David S. Johnson

1 Notazioni asintotiche

Definizione 1. Siano $f, g : \mathbb{N} \to \mathbb{R}$ funzioni.

(1) $f = \mathsf{O}(g)$ se esistono $c \in]0, +\infty[$ e $\nu \in \mathbb{N}$ tali che $0 \leq f(n) \leq c \cdot g(n)$, per ogni $n \geq \nu$.

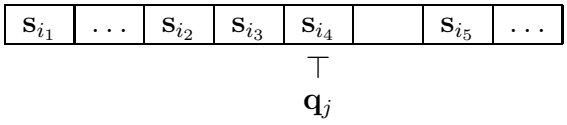
(2) $f = \Omega(g)$ se esistono $c \in]0, +\infty[$ e $\nu \in \mathbb{N}$ tali che $0 \leq c \cdot g(n) \leq f(n)$, per ogni $n \geq \nu$.

(3) $f = \Theta(g)$ se esistono $c_1, c_2 \in]0, +\infty[$ e $\nu \in \mathbb{N}$ tali che $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$, per ogni $n \geq \nu$.

2 Macchine di Turing

Una *Macchina di Turing* è un dispositivo computazionale che effettua operazioni su di un nastro lineare suddiviso in caselle (*squares*). Ogni casella può contenere al suo interno uno qualsiasi dei simboli di un preassegnato insieme finito $\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n\}$ di possibili *simboli di nastro*, oppure una casella può essere vuota. In ogni istante di tempo il nastro ha lunghezza finita, cioè è composto da un numero finito di caselle, però può essere espanso "incollando" nuove caselle vuote alle sue estremità (così il nastro è potenzialmente infinito verso sinistra e verso destra). La macchina interagisce con il nastro tramite una testina meccanica mobile che può leggere e modificare il contenuto delle singole caselle (una casella alla volta). La testina meccanica è collegata ad un *sistema di controllo (a stati finiti)* che determina il comportamento della macchina, cioè le operazioni che vengono eseguite sul nastro. Il sistema di controllo può assumere (o può trovarsi in) un numero finito di possibili *stati interni* $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$, prefissati, dove $\mathbf{q}_0$ è lo *stato iniziale*.

In ogni dato istante di tempo, la testina meccanica si trova posizionata su una particolare casella del nastro (la *casella scandita*) e il sistema di controllo si trova in un particolare stato interno (lo *stato attivo*).



La testina meccanica comunica al sistema di controllo il contenuto della casella scandita e il sistema di controllo, a seconda dello stato attivo in cui si trova, fa eseguire alla testina una ben determinata operazione e successivamente assume un nuovo stato interno (possibilmente lo stesso di prima), oppure il sistema di controllo determina che la macchina si deve fermare. L'esecuzione dell'operazione della testina meccanica e la successiva transizione di stato del sistema di controllo è un atto unico ed indivisibile e costituisce un'*azione (atomica)* della macchina di Turing.

Le operazioni che può eseguire la testina meccanica sono di quattro tipi:

(1) (*cancellazione*) cancellare il contenuto della casella scandita rendendola vuota;

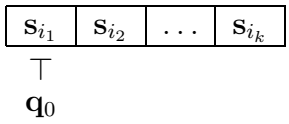
(2) (*stampa*) stampare all'interno della casella scandita un simbolo di nastro;

(3) (*spostamento a sinistra*) spostarsi sulla casella che si trova immediatamente a sinistra della casella scandita. Se prima dell'esecuzione dell'operazione, la casella scandita si trova all'estremità sinistra del nastro, una nuova casella vuota viene aggiunta a questa estremità.

(4) (*spostamento a destra*) spostarsi sulla casella che si trova immediatamente a destra della casella scandita. Se prima dell'esecuzione dell'operazione, la casella scandita si trova all'estremità destra del nastro, una nuova casella vuota viene aggiunta a questa estremità.

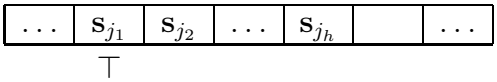
Il sistema di controllo determina la particolare azione che deve compiere la macchina di Turing, oppure che la macchina si deve fermare, consultando un insieme finito di *istruzioni* memorizzate al suo interno (un programma). Se indichiamo il "contenuto di una casella vuota" con il simbolo $\square$ (*blank*) e con $\leftarrow$ e $\rightarrow$ le operazioni che consistono nel muovere la testina meccanica di una casella a sinistra e a destra, rispettivamente, ogni istruzione può essere descritta da una quadrupla (q, t, o, p) dove $q, p \in \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m\}$, $t \in \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n\} \cup \{\square\}$ e $o \in \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n\} \cup \{\square\} \cup \{\leftarrow, \rightarrow\}$. L'interpretazione è la seguente: quando (e se) il sistema di controllo si trova nello stato attivo q e il contenuto della casella scandita è t, la testina meccanica esegue l'operazione rappresentata da o e successivamente il sistema di controllo assume lo stato interno p. $o = \square$ significa che la testina meccanica deve cancellare il contenuto della casella scandita (operazione 1); $o = \mathbf{s}_j$ significa che la testina meccanica deve stampare sulla casella scandita il simbolo $\mathbf{s}_j$ (operazione 2); $o = \leftarrow$ significa che la testina meccanica si deve spostare a sinistra della casella scandita (operazione 3); $o = \rightarrow$ significa che la testina meccanica si deve spostare a destra della casella scandita (operazione 4). Se, in corrispondenza di un certo stato attivo q e di un certo contenuto t della casella scandita non c'è nessuna istruzione nel sistema di controllo che inizia con la coppia (q, t), la macchina si ferma.

Inizialmente il nastro è composto da un numero finito di caselle ciascuna delle quali contiene un simbolo di nastro; il sistema di controllo si trova nello stato iniziale $\mathbf{q}_0$ e la testina meccanica è posizionata sulla casella più a sinistra; questa è la *configurazione iniziale* della macchina:



La macchina di Turing compirà una successione di azioni sotto la guida del sistema di controllo, passando attraverso varie configurazioni, fino a quando eventualmente si fermerà.

La stringa $\mathbf{s}_{i_1}\mathbf{s}_{i_2}\cdots\mathbf{s}_{i_k}$ ottenuta concatenando i simboli contenuti nelle caselle del nastro, da sinistra verso destra, nella configurazione iniziale è l'*input* fornito alla macchina. Quando (e se) la macchina si ferma, raggiungendo una *configurazione terminale*, l'*output* prodotto sarà costituito dalla stringa $\mathbf{s}_{j_1}\mathbf{s}_{j_2}\cdots\mathbf{s}_{j_h}$ ottenuta concatenando i simboli contenuti nelle caselle del nastro a partire dalla casella scandita fino alla prima casella verso destra seguita immediatamente da una casella vuota, ovvero fino all'ultima casella del nastro se alla destra della casella scandita non ci sono caselle vuote. (Se la casella scandita è vuota, l'output è la *stringa vuota*).



3 Formalizzazione delle Macchine di Turing

Siano fissati:

(1) Un insieme $\mathbf{Q}$ di SIMBOLI DI STATO:

$$\mathbf{q}_0 \quad \mathbf{q}_1 \quad \mathbf{q}_2 \quad \mathbf{q}_3 \quad \ldots$$

(2) Un insieme $\mathbf{S}$ di SIMBOLI DI NASTRO:

$$\mathbf{s}_0 \quad \mathbf{s}_1 \quad \mathbf{s}_2 \quad \mathbf{s}_3 \quad \dots$$

(3) Il SIMBOLO DI BLANK:

$$\square$$

(4) I SIMBOLI DI SPOSTAMENTO A SINISTRA E A DESTRA:

$$\lhd \quad \rhd$$

Definizioni preliminari

(1) Per ogni sottoinsieme S di $\mathbf{Q} \cup \mathbf{S} \cup \{\square\}$ indichiamo con S^* l'insieme delle STRINGHE su S, compresa la STRINGA VUOTA ε.

(2) La LUNGHEZZA DI UNA STRINGA X è indicata con $|X|$.

(3) Una n-STRINGA, dove $n \in \mathbb{N}$, è una stringa di lunghezza n.

(4) Un ALFABETO è un sottoinsieme finito di $\mathbf{S}$.

(5) La CARDINALITÀ DI UN ALFABETO Σ è indicata con $|\Sigma|$.

(6) Un LINGUAGGIO è un sottoinsieme di Σ^*, per qualche alfabeto Σ.

(7) L'ALFABETO DI UN LINGUAGGIO $\mathscr{L}$ è il "più piccolo" alfabeto $\Sigma_{\mathscr{L}}$ tale che $\mathscr{L} \subseteq \Sigma_{\mathscr{L}}^*$.

(8) La FUNZIONE CARATTERISTICA DI UN LINGUAGGIO $\mathscr{L}$ è l'insieme $C_{\mathscr{L}}$ delle coppie ordinate (X, ε) dove $X \in \mathscr{L}$.

(9) Il COMPLEMENTARE DI UN LINGUAGGIO $\mathscr{L}$ è il linguaggio $\complement(\mathscr{L}) = \Sigma_{\mathscr{L}}^* \setminus \mathscr{L}$.

(10) Una RELAZIONE (DI STRINGHE) è un sottoinsieme del prodotto cartesiano $\Sigma^* \times \Sigma^*$, per qualche alfabeto Σ.

(11) L'ALFABETO DI UNA RELAZIONE ρ è il più piccolo alfabeto Σ_ρ tale che $\rho \subseteq \Sigma_\rho^* \times \Sigma_\rho^*$.

Definizione 2. Una ISTRUZIONE (O QUADRUPLA) è una 4-stringa $qtop$ dove $q, p \in \mathbf{Q}$, $t \in \mathbf{S} \cup \{\square\}$ e $o \in \mathbf{S} \cup \{\square\} \cup \{\lhd, \rhd\}$.

Ogni 1-stringa o, dove $o \in \mathbf{S} \cup \{\square\} \cup \{\lhd, \rhd\}$, è chiamata un'OPERAZIONE ed ogni 2-stringa op, dove $o \in \mathbf{S} \cup \{\square\} \cup \{\lhd, \rhd\}$ e $p \in \mathbf{Q}$, è chiamata un'AZIONE.

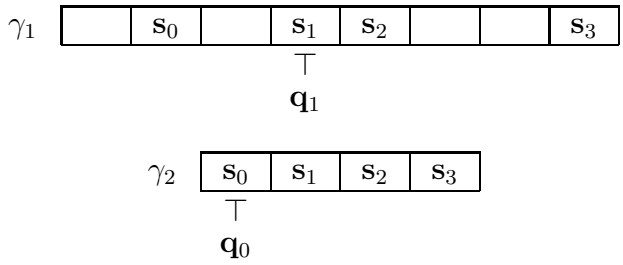
Una CONFIGURAZIONE (O DESCRIZIONE ISTANTANEA) è una stringa γ della forma $XqtY$ dove $t \in \mathbf{S} \cup \{\square\}$, $q \in \mathbf{Q}$ e $X, Y \in (\mathbf{S} \cup \{\square\})^*$.

Data una stringa $Z \in \mathbf{S}^*$, la CONFIGURAZIONE INIZIALE (O DI INPUT) corrispondente a Z è la configurazione $\mathtt{Start}(Z)$ definita come segue:

$$\mathtt{Start}(Z) = \begin{cases} \mathbf{q}_0\square, & \text{se } Z = \varepsilon \\ \mathbf{q}_0 Z, & \text{altrimenti}\,. \end{cases}$$

Una CONFIGURAZIONE γ È INIZIALE se $\gamma = \mathtt{Start}(Z)$, per qualche stringa $Z \in \mathbf{S}^*$. ■

Esempio 1. Le configurazioni $\gamma_1 = \square\mathbf{s}_0\square\mathbf{q}_1\mathbf{s}_1\mathbf{s}_2\square\square\mathbf{s}_3$ e $\gamma_2 = \mathbf{q}_0\mathbf{s}_0\mathbf{s}_1\mathbf{s}_2\mathbf{s}_3$ ammettono, rispettivamente, le seguenti rappresentazioni grafiche:



γ_1 non è iniziale; γ_2 è la configurazione iniziale corrispondente alla stringa $\mathbf{s}_0\mathbf{s}_1\mathbf{s}_2\mathbf{s}_3$. ■

Definizione 3. Sia $\gamma = XqtY$ una configurazione.

(1) Il CONTENUTO ESTRATTO DA γ è la più lunga stringa $\langle\gamma\rangle$ non contenente il simbolo $\square$ che è prefisso di tY. In maniera più precisa, $\langle\gamma\rangle$ è definita come segue:

(1.a) se $t = \square$, allora $\langle\gamma\rangle$ è la stringa vuota ε;

(1.b) se $t \neq \square$ e Y non contiene $\square$, allora $\langle\gamma\rangle = tY$;

(1.c) se $t \neq \square$ e $Y = y_1 \cdots y_k \square y_{k+1} \cdots y_n$, dove $y_1, \ldots, y_k \neq \square$, allora $\langle\gamma\rangle = ty_1y_2 \cdots y_k$.

(2) Per ogni azione $A = op$, il RISULTATO DELL'APPLICAZIONE DI A A γ è la configurazione $A(\gamma)$ definita come segue:

(2.a) se $o \in \mathbf{S} \cup \{\square\}$ allora $A(\gamma) = XpoY$;

(2.b) se $o = \leftarrow$ e $X = Zs$ dove $Z \in (\mathbf{S} \cup \{\square\})^*$ e $s \in \mathbf{S} \cup \{\square\}$, allora $A(\gamma) = ZpstY$;

(2.c) se $o = \leftarrow$ e $X = \varepsilon$ allora $A(\gamma) = p\square tY$;

(2.d) se $o = \rightarrow$ e $Y = sZ$ dove $Z \in (\mathbf{S} \cup \{\square\})^*$ e $s \in \mathbf{S} \cup \{\square\}$ allora $A(\gamma) = XtpsZ$;

(2.e) se $o = \rightarrow$ e $Y = \varepsilon$ allora $A(\gamma) = Xtp\square$. ■

Esempio 2.

$$\langle\square\mathbf{s}_0\mathbf{s}_2\mathbf{q}_0\mathbf{s}_1\mathbf{s}_2\mathbf{s}_3\square\mathbf{s}_4\rangle = \mathbf{s}_1\mathbf{s}_2\mathbf{s}_3$$
$$\langle\square\mathbf{s}_0\mathbf{s}_2\mathbf{q}_5\square\mathbf{s}_4\mathbf{s}_2\mathbf{s}_1\rangle = \varepsilon$$
$$\langle\square\mathbf{s}_0\mathbf{s}_2\mathbf{q}_1\mathbf{s}_4\mathbf{s}_2\square\mathbf{s}_1\square\mathbf{s}_4\mathbf{s}_5\mathbf{s}_6\rangle = \mathbf{s}_4\mathbf{s}_2$$

■

Esempio 3. Sia $\gamma = \mathbf{s}_0\square\mathbf{q}_3\mathbf{s}_2$ e siano $A_1 = {\leftarrow}\mathbf{q}_0$, $A_2 = {\rightarrow}\mathbf{q}_1$ e $A_3 = \mathbf{s}_1\mathbf{q}_3$. Allora

$$A_1(\gamma) = \mathbf{s}_0\mathbf{q}_0\square\mathbf{s}_2\,, \quad A_2(\gamma) = \mathbf{s}_0\square\mathbf{s}_2\mathbf{q}_1\square\,, \quad A_3(\gamma) = \mathbf{s}_0\square\mathbf{q}_3\mathbf{s}_1$$

■

Definizione 4. Una MACCHINA DI TURING (TM) è un insieme finito di istruzioni.

Una MACCHINA DI TURING DETERMINISTICA (DTM) è una macchina di Turing $\mathfrak{M}$ tale che per nessuna coppia di simboli (q, t) esistono due distinte istruzioni di $\mathfrak{M}$ che cominciano con qt.

Una MACCHINA DI TURING NONDETERMINISTICA (NDTM) è una macchina di Turing $\mathfrak{M}$ tale che per qualche coppia di simboli (q, t) esistono (almeno) due distinte istruzioni di $\mathfrak{M}$ che cominciano con qt. ■

Esempio 4. La TM $\{\mathbf{q}_0\Box\mathbf{s}_0\mathbf{q}_1, \mathbf{q}_0\Box{\leftarrow}\mathbf{q}_2, \mathbf{q}_1\mathbf{s}_0{\rightarrow}\mathbf{q}_0\}$ è nondeterministica dato che le istruzioni $\mathbf{q}_0\Box\mathbf{s}_0\mathbf{q}_1$ e $\mathbf{q}_0\Box{\leftarrow}\mathbf{q}_2$ cominciano con la stessa coppia stato-simbolo $(\mathbf{q}_0, \Box)$.
La TM $\{\mathbf{q}_0\Box\mathbf{s}_0\mathbf{q}_1, \mathbf{q}_0\mathbf{s}_1{\leftarrow}\mathbf{q}_2, \mathbf{q}_1\mathbf{s}_0{\rightarrow}\mathbf{q}_0\}$ è invece deterministica. ■

Definizione 5. Sia $\mathfrak{M}$ una TM.

L'ALFABETO DI $\mathfrak{M}$ è l'insieme $\mathbf{S}(\mathfrak{M})$ dei simboli di nastro che occorrono nelle istruzioni di $\mathfrak{M}$. L'insieme dei simboli di stato che occorrono nelle istruzioni di $\mathfrak{M}$ (gli stati di $\mathfrak{M}$) è indicato con $\mathbf{Q}(\mathfrak{M})$. Indichiamo con $\mathbf{F}(\mathfrak{M})$ l'insieme delle coppie $(q,t) \in (\mathbf{Q}(\mathfrak{M})\cup\{\mathbf{q}_0\})\times(\mathbf{S}(\mathfrak{M})\cup\{\Box\})$ tali che nessuna istruzione di $\mathfrak{M}$ inizia con qt.

Una CONFIGURAZIONE $XqtY$ È TERMINALE (RISP. NON-TERMINALE) RISPETTO AD $\mathfrak{M}$, se nessuna (risp. qualche) istruzione di $\mathfrak{M}$ inizia con qt.

Una CONFIGURAZIONE γ'' CONSEGUE (IMMEDIATAMENTE) DA UNA CONFIGURAZIONE $\gamma' = XqtY$ MEDIANTE L'ESECUZIONE DI UNA ISTRUZIONE DI $\mathfrak{M}$, e in tal caso si scrive $\gamma' \vdash_{\mathfrak{M}} \gamma''$, se esiste un'azione $A = op$ tale che $\gamma'' = A(\gamma')$ ed $\mathfrak{M}$ contiene l'istruzione $qtop$.

Un SEGMENTO DI COMPUTAZIONE DI $\mathfrak{M}$ è una sequenza finita $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di configurazioni, dove $n \geq 0$, tali che $\gamma_i \vdash_{\mathfrak{M}} \gamma_{i+1}$, per ogni $i = 0, 1, \ldots, n-1$. La LUNGHEZZA di Γ è $|\Gamma| = n$.

Per configurazioni γ' e γ'' ed un $n \in \mathbb{N}$ scriviamo $\gamma' \vdash^n_{\mathfrak{M}} \gamma''$ se esiste un segmento di computazione $(\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$ (di lunghezza n) dove $\gamma_0 = \gamma'$ e $\gamma_n = \gamma''$. Scriviamo anche $\gamma' \vdash^*_{\mathfrak{M}} \gamma''$ se esiste un $n \in \mathbb{N}$ tale che $\gamma' \vdash^n_{\mathfrak{M}} \gamma''$.

Una COMPUTAZIONE di $\mathfrak{M}$ è un segmento di computazione $(\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$ dove γ_0 è una configurazione iniziale e γ_n è una configurazione terminale rispetto a $\mathfrak{M}$.

Sia $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ una computazione di $\mathfrak{M}$.

(1) L'INPUT DI Γ è la stringa $\mathtt{Input}(\Gamma) = \langle\gamma_0\rangle$;

(2) L'OUTPUT DI Γ è la stringa $\mathtt{Output}(\Gamma) = \langle\gamma_n\rangle$.

Una COMPUTAZIONE PROPRIA DI $\mathfrak{M}$ è una computazione Γ di $\mathfrak{M}$ tale che $\mathtt{Input}(\Gamma) \in \mathbf{S}(\mathfrak{M})^*$.

Una CONFIGURAZIONE DI $\mathfrak{M}$ (O $\mathfrak{M}$-CONFIGURAZIONE) è una configurazione $XqtY$ dove X e Y sono stringhe su $\mathbf{S}(\mathfrak{M}) \cup \{\Box\}$, $t \in \mathbf{S}(\mathfrak{M}) \cup \{\Box\}$ e $q \in \mathbf{Q}(\mathfrak{M}) \cup \{\mathbf{q}_0\}$. ■

Si osservi che se $\mathfrak{M}$ è una DTM allora, per ogni stringa $X \in \mathbf{S}^*$ esiste al più una sola computazione di $\mathfrak{M}$ con input X. Se invece $\mathfrak{M}$ è una NDTM, possono esistere anche due o più computazioni distinte di $\mathfrak{M}$ con lo stesso input X. Si osservi inoltre che ogni configurazione contenuta in una computazione propria di una TM $\mathfrak{M}$ è una $\mathfrak{M}$-configurazione.

Esempio 5. Sia $\mathfrak{M} = \{\mathbf{q}_0\mathbf{s}_0{\rightarrow}\mathbf{q}_0, \mathbf{q}_0\mathbf{s}_1\mathbf{s}_0\mathbf{q}_0, \mathbf{q}_0\Box{\leftarrow}\mathbf{q}_1, \mathbf{q}_1\mathbf{s}_0{\leftarrow}\mathbf{q}_1, \mathbf{q}_1\Box{\rightarrow}\mathbf{q}_2\}$.
Allora:

(1) $\mathbf{S}(\mathfrak{M}) = \{\mathbf{s}_0, \mathbf{s}_1\}$;

(2) $\mathbf{Q}(\mathfrak{M}) = \{\mathbf{q}_o, \mathbf{q}_1, \mathbf{q}_2\}$;

(3) $\mathbf{F}(\mathfrak{M}) = \{(\mathbf{q}_1, \mathbf{s}_1), (\mathbf{q}_2, \Box), (\mathbf{q}_2, \mathbf{s}_0), (\mathbf{q}_2, \mathbf{s}_1)\}$;

(4) La seguente sequenza di configurazioni è una computazione di $\mathfrak{M}$ con input $\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0$ e

output $\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0$ di lunghezza 15:

$$
\begin{array}{l}
\mathbf{q}_0\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{q}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{q}_0\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\mathbf{s}_0\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\mathbf{s}_1\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\mathbf{s}_0\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\mathbf{s}_0 \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_0\square \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_1\mathbf{s}_0\square \\
\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{q}_1\mathbf{s}_0\mathbf{s}_0\square \\
\mathbf{s}_0\mathbf{s}_0\mathbf{q}_1\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\square \\
\mathbf{s}_0\mathbf{q}_1\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\square \\
\mathbf{q}_1\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\square \\
\mathbf{q}_1\square\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\square \\
\square\mathbf{q}_2\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\square
\end{array}
$$

■

Teorema 1. *Sia $\mathfrak{M}$ una DTM. Esiste una NDTM $\mathfrak{N}$ tale che l'insieme delle computazioni di $\mathfrak{N}$ è uguale all'insieme delle computazioni di $\mathfrak{M}$.*

Dimostrazione. Sia q un simbolo di stato tale che $q \notin (\mathbf{Q}(\mathfrak{M}) \cup \{\mathbf{q}_0\})$. Poniamo

$$\mathfrak{N} = \mathfrak{M} \cup \{q\square\square q, q\square\square\mathbf{q}_0\}\,.$$

□

Definizione 6. Siano $\mathfrak{M}$ una TM e $X, Y \in \mathbf{S}^*$ stringhe.

(1) $\mathfrak{M}$ CON INPUT X PRODUCE IN OUTPUT Y, e si scrive $\mathfrak{M}(X)\downarrow Y$, se esiste una computazione Γ di $\mathfrak{M}$ tale che $\mathtt{Input}(\Gamma) = X$ e $\mathtt{Output}(\Gamma) = Y$;

(2) $\mathfrak{M}$ CON INPUT X PRODUCE IN OUTPUT Y IN n-PASSI, O IN TEMPO n, dove $n \in \mathbb{N}$, e si scrive $\mathfrak{M}(X)\downarrow^{(n)} Y$, se esiste una computazione Γ di $\mathfrak{M}$ di lunghezza n tale che $\mathtt{Input}(\Gamma) = X$ e $\mathtt{Output}(\Gamma) = Y$. Scriviamo $\mathfrak{M}(X)\downarrow^{(\leq n)} Y$ se $\mathfrak{M}(X)\downarrow^{(m)} Y$, per qualche $m \leq n$.

(3) $\mathfrak{M}$ ACCETTA (O RICONOSCE) X, e si scrive $\mathfrak{M}(X)\downarrow$, se $\mathfrak{M}(X)\downarrow Z$, per qualche stringa $Z \in \mathbf{S}^*$. Scriviamo $\mathfrak{M}(X)\uparrow$ se $\mathfrak{M}(X)\downarrow$ non vale.

(4) $\mathfrak{M}$ ACCETTA (O RICONOSCE) X IN n-PASSI, O IN TEMPO n, dove $n \in \mathbb{N}$, e si scrive $\mathfrak{M}(X)\downarrow^{(n)}$, se $\mathfrak{M}(X)\downarrow^{(n)} Z$, per qualche stringa $Z \in \mathbf{S}^*$. Scriviamo $\mathfrak{M}(X)\downarrow^{(\leq n)}$ se $\mathfrak{M}(X)\downarrow^{(m)}$, per qualche $m \leq n$. ■

Esempio 6. Riferendoci all'esempio precedente:

(1) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0$;

(2) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow^{(15)}\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0$;

(3) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow^{(\leq n)}\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0\mathbf{s}_0$, per ogni $n \geq 15$;

(4) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow$;

(5) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow^{(15)}$;

(6) $\mathfrak{M}(\mathbf{s}_1\mathbf{s}_0\mathbf{s}_1\mathbf{s}_1\mathbf{s}_0)\downarrow^{(\leq n)}$, per ogni $n \geq 15$. ■

Definizione 7. Sia $\mathfrak{M}$ una TM.

(1) La RELAZIONE COMPUTATA DA $\mathfrak{M}$ è l'insieme $\varrho(\mathfrak{M})$ delle coppie (X, Y) di stringhe su $\mathbf{S}(\mathfrak{M})$ tali che $\mathfrak{M}(X)\downarrow Y$. (Si osservi che se $\mathfrak{M}$ è una DTM allora $\varrho(\mathfrak{M})$ è una funzione: LA FUNZIONE COMPUTATA DA $\mathfrak{M}$.)

(2) Il LINGUAGGIO ACCETTATO (O RICONOSCIUTO) da $\mathfrak{M}$ è l'insieme $\mathscr{L}(\mathfrak{M})$ delle stringhe su $\mathbf{S}(\mathfrak{M})$ che sono accettate da $\mathfrak{M}$.

(3) Una RELAZIONE ρ DI STRINGHE È COMPUTATA DA $\mathfrak{M}$ se e solo se $\rho = \varrho(\mathfrak{M})$.

(4) Un LINGUAGGIO $\mathscr{S}$ È ACCETTATO DA $\mathfrak{M}$ se e solo se $\mathscr{S} = \mathscr{L}(\mathfrak{M})$. ■

Definizione 8. Siano $\mathscr{S}$ un linguaggio ed f una funzione.

(1) $\mathscr{S}$ è (TURING) SEMIDECIDIBILE se esiste una TM $\mathfrak{M}$ tale che $\mathscr{S}$ è accettato da $\mathfrak{M}$.

(2) f è (TURING) COMPUTABILE se esiste una DTM $\mathfrak{M}$ tale che f è computata da $\mathfrak{M}$.

(3) $\mathscr{S}$ È DECISO DA UNA DTM $\mathfrak{D}$ se:

(3.a) $\mathscr{L}(\mathfrak{D}) = \Sigma_{\mathscr{S}}^*$ e

(3.b) $\mathscr{S} = \{X \in \mathbf{S}(\mathfrak{D})^* : \mathfrak{D}(X)\downarrow\varepsilon\}$.

(4) $\mathscr{S}$ è (TURING) DECIDIBILE se esiste una DTM $\mathfrak{D}$ tale che $\mathscr{S}$ è deciso da $\mathfrak{D}$. ■

Definizione 9. Sia $\mathfrak{M}$ una TM.

(1) Per ogni stringa $X \in \mathscr{L}(\mathfrak{M})$, la COMPLESSITÀ TEMPORALE DI $\mathfrak{M}$ SULL'INPUT X è la quantità $\widehat{T}_{\mathfrak{M}}(X)$ definita come segue:

$$\widehat{T}_{\mathfrak{M}}(X) = \min\{n \in \mathbb{N} : \mathfrak{M}(X)\downarrow^{(n)}\}\,.$$

(Cioè $\widehat{T}_{\mathfrak{M}}(X)$ è la lunghezza della più corta computazione di $\mathfrak{M}$ con input X.)

(2) Il RUNNING TIME DI $\mathfrak{M}$ è la funzione $T_{\mathfrak{M}} : \mathbb{N} \to \mathbb{N} \cup \{-1\}$ definita come segue:

$$T_{\mathfrak{M}}(n) = \begin{cases} \max\{\widehat{T}_{\mathfrak{M}}(X) : X \in \mathscr{L}(\mathfrak{M}) \text{ AND } |X| = n\}, & \text{se } \{X \in \mathscr{L}(\mathfrak{M}) : |X| = n\} \neq \emptyset \\ -1, & \text{altrimenti}\,. \end{cases}$$

(3) Diciamo che $\mathfrak{M}$ È REGOLARE (RISPETTO ALLA COMPLESSITÀ TEMPORALE) se:

per ogni $Y, Z \in \mathscr{L}(\mathfrak{M})$, se $|Y| = |Z|$ allora $\widehat{T}_{\mathfrak{M}}(Y) = \widehat{T}_{\mathfrak{M}}(Z)$.

(Cioè, $\mathfrak{M}$ è regolare se per ogni stringa $X \in \mathscr{L}(\mathfrak{M})$, la complessità temporale di $\mathfrak{M}$ con input X dipende solo dalla lunghezza $|X|$ di X.) ■

Nel seguito useremo la seguente notazione: per ogni sottoinsieme non vuoto S di $\mathbf{S} \cup \{\square\}$, ogni stato q e ogni azione A, indicheremo con qSA l'insieme di istruzioni $\{qtA : t \in S\}$.

Esempio 7. La seguente TM $\mathfrak{M}$ accetta il linguaggio $\mathscr{L}$ costituito da tutte le stringhe palindrome sull'alfabeto $\{a, b\}$ ($a = \mathbf{s}_i$, $b = \mathbf{s}_j$, $i \neq j$):

$$
\begin{array}{l}
\mathbf{q}_0 a \square \mathbf{q}_1 \\
\mathbf{q}_0 b \square \mathbf{q}_2 \\
\mathbf{q}_1 \square \mathbin{\vec{\ }} \mathbf{q}_3 \\
\mathbf{q}_2 \square \mathbin{\vec{\ }} \mathbf{q}_4 \\
\mathbf{q}_3 \{a, b\} \mathbin{\vec{\ }} \mathbf{q}_3 \\
\mathbf{q}_4 \{a, b\} \mathbin{\vec{\ }} \mathbf{q}_4 \\
\mathbf{q}_3 \square \mathbin{\overleftarrow{\ }} \mathbf{q}_5 \\
\mathbf{q}_4 \square \mathbin{\overleftarrow{\ }} \mathbf{q}_6 \\
\mathbf{q}_5 bb \mathbf{q}_5 \\
\mathbf{q}_6 aa \mathbf{q}_6 \\
\mathbf{q}_5 a \square \mathbf{q}_5 \\
\mathbf{q}_6 b \square \mathbf{q}_6 \\
\mathbf{q}_5 \square \mathbin{\overleftarrow{\ }} \mathbf{q}_7 \\
\mathbf{q}_6 \square \mathbin{\overleftarrow{\ }} \mathbf{q}_7 \\
\mathbf{q}_7 \{a, b\} \mathbin{\overleftarrow{\ }} \mathbf{q}_7 \\
\mathbf{q}_7 \square \mathbin{\vec{\ }} \mathbf{q}_0
\end{array}
$$

Osserviamo che $\mathfrak{M}$ è regolare e che il suo running time $T_{\mathfrak{M}}$ soddisfa la seguente equazione di ricorrenza:

$$
T_{\mathfrak{M}}(n) = \begin{cases} 0, & \text{se } n = 0 \\ 5, & \text{se } n = 1 \\ T_{\mathfrak{M}}(n-2) + 2n + 3, & \text{se } n \geq 2\,, \end{cases}
$$

per ogni $n \in \mathbb{N}$.
Espandendo l'equazione si ottiene che

$$
T_{\mathfrak{M}}(n) = T_{\mathfrak{M}}(n-2k) + 3k + 2\left[\sum_{i=0}^{k-1}(n-2i)\right] = T_{\mathfrak{M}}(n-2k) + 2nk - 2k^2 + 5k\,,
$$

per ogni $k \leq n/2$, e quindi:

$$
T_{\mathfrak{M}}(n) = \begin{cases} \dfrac{1}{2}n^2 + \dfrac{5}{2}n, & \text{se } n \text{ è pari} \\ \dfrac{1}{2}n^2 + \dfrac{5}{2}n + 2, & \text{altrimenti}\,. \end{cases}
$$

Pertanto $T_{\mathfrak{M}}(n) = \Theta(n^2)$; così il running time di $\mathfrak{M}$ è quadratico. ■

Esempio 8. La seguente DTM computa la funzione $f : \{a, b\}^* \to \{a, b\}^*$ ($a = \mathbf{s}_i$, $b = \mathbf{s}_j$) tale che, per ogni stringa $X \in \{a, b\}^*$, $f(X)$ è la stringa ottenuta da X eliminando in essa

ogni eventuale occorrenza del simbolo a:

$$\begin{array}{l}
\mathbf{q}_0\{a,b\}\mathrm{R}\mathbf{q}_0\\
\mathbf{q}_0 * *\mathbf{q}_0\\
\mathbf{q}_0\Box * \mathbf{q}_1\\
\mathbf{q}_1\{a,b,*\}\mathrm{L}\mathbf{q}_1\\
\mathbf{q}_1\Box\mathrm{R}\mathbf{q}_2\\
\mathbf{q}_2 a\Box\mathbf{q}_3\\
\mathbf{q}_3\Box\mathrm{R}\mathbf{q}_2\\
\mathbf{q}_2 b\Box\mathbf{q}_4\\
\mathbf{q}_4\Box\mathrm{R}\mathbf{q}_5\\
\mathbf{q}_5\{a,b,*\}\mathrm{R}\mathbf{q}_5\\
\mathbf{q}_5\Box b\mathbf{q}_6\\
\mathbf{q}_6\{a,b,*\}\mathrm{L}\mathbf{q}_6\\
\mathbf{q}_6\Box\mathrm{R}\mathbf{q}_2\\
\mathbf{q}_2 * \Box\mathbf{q}_2\\
\mathbf{q}_2\Box\mathrm{R}\mathbf{q}_7
\end{array}$$

(Si osservi che la macchina utilizza un simbolo ausiliario $* = \mathbf{s}_k$ diverso da a e da b). ■

Esempio 9. La seguente DTM computa la funzione $f : \{a,b\}^* \to \{a,b\}^*$ ($a = \mathbf{s}_i$, $b = \mathbf{s}_j$) tale che, per ogni stringa $X \in \{a,b\}^*$, $f(X)$ è il "reversal" di X, cioè X scritta al rovescio:

$$\begin{array}{l}
\mathbf{q}_0\{a,b\}\mathrm{R}\mathbf{q}_0\\
\mathbf{q}_0 * *\mathbf{q}_0\\
\mathbf{q}_0\Box * \mathbf{q}_1\\
\mathbf{q}_1 * \mathrm{L}\mathbf{q}_1\\
\mathbf{q}_1 a * \mathbf{q}_2\\
\mathbf{q}_1 b * \mathbf{q}_3\\
\mathbf{q}_2\{a,b,*\}\mathrm{R}\mathbf{q}_2\\
\mathbf{q}_3\{a,b,*\}\mathrm{R}\mathbf{q}_3\\
\mathbf{q}_2\Box a\mathbf{q}_4\\
\mathbf{q}_3\Box b\mathbf{q}_4\\
\mathbf{q}_4\{a,b\}\mathrm{L}\mathbf{q}_4\\
\mathbf{q}_4 * \mathrm{L}\mathbf{q}_1\\
\mathbf{q}_1\Box\mathrm{R}\mathbf{q}_5\\
\mathbf{q}_5 * \mathrm{R}\mathbf{q}_5
\end{array}$$

(Si osservi che la macchina utilizza un simbolo ausiliario $* = \mathbf{s}_k$ diverso da a e da b). ■

4 Funzioni numeriche (Turing) computabili

Di seguito indichiamo i simboli $\mathbf{s}_0$ ed $\mathbf{s}_1$ con $\mathbf{0}$ e $\mathbf{1}$, rispettivamente.

Dato un $n \in \mathbb{N}$ denotiamo con $\ulcorner n \urcorner$ la stringa di lunghezza n formata concatenando n copie del simbolo $\mathbf{1}$. (Es. $\ulcorner 0 \urcorner = \varepsilon$, $\ulcorner 1 \urcorner = \mathbf{1}$, $\ulcorner 2 \urcorner = \mathbf{11}$, ..., $\ulcorner k \urcorner = \underbrace{\mathbf{11}\ldots\mathbf{1}}_{k \text{ volte}}$, ...)

Sia f una funzione parziale su $\mathbb{N}$ di arietà k, dove $k \geq 1$, e sia D il dominio di f. Indichiamo con $\ulcorner f \urcorner$ la funzione tale che:

(1) il dominio di $\ulcorner f \urcorner$ è l'insieme di stringhe

$$\{\ulcorner n_1 \urcorner \mathbf{0} \ulcorner n_2 \urcorner \mathbf{0} \cdots \mathbf{0} \ulcorner n_k \urcorner : (n_1, n_2, \ldots, n_k) \in D\}\,;$$

(2) per ogni $(n_1, n_2, \ldots, n_k) \in D$,

$$\ulcorner f \urcorner(\ulcorner n_1 \urcorner \mathbf{0} \ulcorner n_2 \urcorner \mathbf{0} \cdots \mathbf{0} \ulcorner n_k \urcorner) = \ulcorner f(n_1, n_2, \ldots, n_k) \urcorner\,.$$

Diciamo che f è (Turing) computabile se lo è la funzione $\ulcorner f \urcorner$.

Esempio 10. Siano $f : \mathbb{N} \to \mathbb{N}$ e $g : \mathbb{N}^2 \to \mathbb{N}$ definite da

$$f(n) = \begin{cases} \dfrac{n}{2}, & \text{se } n \text{ è pari} \\ \uparrow, & \text{altrimenti} \end{cases} \quad \text{e} \quad g(n,m) = \begin{cases} n \times m, & \text{se } n + m = 3 \\ \uparrow, & \text{altrimenti}\,. \end{cases}$$

Il dominio di $\ulcorner f \urcorner$ è composto dalle stringhe ε, **11**, **1111**, **111111**, ..., e si ha:

$$\begin{aligned} &\ulcorner f \urcorner(\varepsilon) = \varepsilon \\ &\ulcorner f \urcorner(\mathbf{11}) = \mathbf{1} \\ &\ulcorner f \urcorner(\mathbf{1111}) = \mathbf{11} \\ &\ulcorner f \urcorner(\mathbf{111111}) = \mathbf{111} \\ &\vdots \end{aligned}$$

Il dominio di $\ulcorner g \urcorner$ è composto dalle stringhe **0111**, **1110**, **1011** e **1101** e si ha:

$$\begin{aligned} &\ulcorner g \urcorner(\mathbf{0111}) = \varepsilon \\ &\ulcorner g \urcorner(\mathbf{1110}) = \varepsilon \\ &\ulcorner g \urcorner(\mathbf{1011}) = \mathbf{11} \\ &\ulcorner g \urcorner(\mathbf{1101}) = \mathbf{11}\,. \end{aligned}$$

Le funzioni f e g sono computabili. Infatti, ad esempio, $\ulcorner f \urcorner$ è computata dalla seguente DTM:

$$\begin{aligned} &\mathbf{q}_0\mathbf{00}\mathbf{q}_0 \\ &\mathbf{q}_0\mathbf{1}\mathbf{\vec{r}}\mathbf{q}_1 \\ &\mathbf{q}_0\square\mathbf{\vec{l}}\mathbf{q}_5 \\ &\mathbf{q}_1\mathbf{00}\mathbf{q}_1 \\ &\mathbf{q}_1\mathbf{10}\mathbf{q}_7 \\ &\mathbf{q}_7\mathbf{0}\mathbf{\vec{r}}\mathbf{q}_2 \\ &\mathbf{q}_1\square\square\mathbf{q}_1 \\ &\mathbf{q}_2\mathbf{00}\mathbf{q}_2 \\ &\mathbf{q}_2\mathbf{1}\mathbf{\vec{r}}\mathbf{q}_2 \\ &\mathbf{q}_2\square\mathbf{\vec{l}}\mathbf{q}_3 \\ &\mathbf{q}_3\mathbf{0}\square\mathbf{q}_0 \\ &\mathbf{q}_3\mathbf{1}\square\mathbf{q}_7 \\ &\mathbf{q}_7\square\mathbf{\vec{l}}\mathbf{q}_4 \\ &\mathbf{q}_4\mathbf{01}\mathbf{q}_0 \\ &\mathbf{q}_4\mathbf{1}\mathbf{\vec{l}}\mathbf{q}_4 \\ &\mathbf{q}_5\mathbf{1}\mathbf{\vec{l}}\mathbf{q}_5 \\ &\mathbf{q}_5\square\mathbf{\vec{r}}\mathbf{q}_6 \end{aligned}$$

■

5 Macchine di Turing Multi-nastro

Una macchina di Turing multi-nastro possiede due o più nastri, ciascuno dotato della propria testina meccanica. Le testine possono operare sui rispettivi nastri in maniera indipendente l'una dall'altra però sono comandate da un unico sistema di controllo che determina il comportamento della macchina. Un'istruzione di una macchina di Turing multi-nastro può quindi essere rappresentata come una quadrupla $(q, (t_1 \dots t_k), (o_1 \dots o_k), p)$ dove k è il numero dei nastri della macchina, q e p sono stati e, per $i = 1, \dots, k$, t_i rappresenta il contenuto della casella scandita dalla testina meccanica nell'i-esimo nastro, mentre o_i rappresenta l'operazione che la testina meccanica dell'i-esimo nastro deve eseguire. Quindi, un'azione di una macchina di Turing con k nastri consiste nell'esecuzione in parallelo di k-operazioni delle testine meccaniche, un'operazione su ogni singolo nastro, e nella successiva transizione di stato del sistema di controllo. Data una stringa input $\mathbf{s}_{i_1} \cdots \mathbf{s}_{i_n}$, inizialmente il sistema di controllo assume lo stato iniziale $\mathbf{q}_0$; il primo nastro è composto da n caselle, contenenti, da sinistra verso destra, i simboli $\mathbf{s}_{i_1}$, ..., $\mathbf{s}_{i_n}$ e la testina meccanica è posizionata sulla casella più a sinistra; i rimanenti nastri sono composti ciascuno da una singola casella vuota. Quando (e se) la macchina si ferma, l'output viene estratto dal primo nastro con la stessa modalità descritta in precedenza per le macchine di Turing con un solo nastro.

Le definizioni date nella Sezione 3 si generalizzano al caso delle macchine di Turing multi-nastro. Riportiamo di seguito quelle principali.

Definizione 10. Sia $k \in \mathbb{N}$, $k > 1$.

Una k-ISTRUZIONE è una k-upla ordinata $(q_1 t_1 o_1 p_1, \dots, q_k t_k o_k p_k)$ di istruzioni tali che $q_1 = \cdots = q_k$ e $p_1 = \cdots = p_k$.

Una MACCHINA DI TURING CON k-NASTRI (k-TM) è un insieme finito di k-istruzioni.

Una k-CONFIGURAZIONE è una k-upla ordinata $\boldsymbol{\gamma} = (X_1 q_1 t_1 Y_1, \dots, X_k q_k t_k Y_k)$ di configurazioni tali che $q_1 = \cdots = q_k$. Il contenuto estratto da $\boldsymbol{\gamma}$ è $\langle \boldsymbol{\gamma} \rangle = \langle X_1 q_1 t_1 Y_1 \rangle$, cioè il contenuto estratto dalla prima configurazione.

La k-CONFIGURAZIONE INIZIALE corrispondente ad una stringa $Z \in \mathbf{S}^*$ è la k-configurazione $(\mathtt{Start}(Z), \mathbf{q}_0\Box, \dots, \mathbf{q}_0\Box)$.

Sia $\mathfrak{M}$ una k-TM.

Per k-configurazioni $\boldsymbol{\gamma}' = (\gamma'_1, \dots, \gamma'_k)$ e $\boldsymbol{\gamma}'' = (\gamma''_1, \dots, \gamma''_k)$, scriviamo $\boldsymbol{\gamma}' \vdash_{\mathfrak{M}} \boldsymbol{\gamma}''$ se esiste una k-istruzione $(I_1, \dots, I_k) \in \mathfrak{M}$ tale che $\gamma'_j \vdash_{\{I_j\}} \gamma''_j$, per ogni $j = 1, \dots, k$.

Una k-configurazione $\boldsymbol{\gamma} = (X_1 q t_1 Y_1, \dots, X_k q t_k Y_k)$ è terminale rispetto a $\mathfrak{M}$ se non esiste alcuna k-istruzione $(I_1, \dots, I_k) \in \mathfrak{M}$ tale che I_j comincia con $q t_j$, per ogni $j = 1, \dots, k$.

Un segmento di computazione di $\mathfrak{M}$ è una sequenza finita $\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_n)$ di k-configurazioni, dove $n \geq 0$, tali che $\boldsymbol{\gamma}_i \vdash_{\mathfrak{M}} \boldsymbol{\gamma}_{i+1}$, per ogni $i = 0, 1, \dots, n-1$. ■

Nel seguito useremo le seguenti notazioni:

(1) ogni k-istruzione $(q_1 t_1 o_1 p_1, \dots, q_k t_k o_k p_k)$ verrà rappresentata come

$$q(t_1, \dots, t_k)(o_1, \dots, o_k)p\,,$$

dove $q_1 = \cdots = q_k = q$ e $p_1 = \cdots = p_k = p$;

(2) ogni k-configurazione $(X_1 q_1 t_1 Y_1, \dots, X_k q_k t_k Y_k)$ verrà rappresentata come

$$(q, X_1 \perp t_1 Y_1, \dots, X_k \perp t_k Y_k)\,,$$

dove $q_1 = \cdots = q_k = q$.

Esempio 11. La seguente 2-TM $\mathfrak{A}$ accetta il linguaggio $\mathscr{L}$ costituito da tutte le stringhe palindrome sull'alfabeto $\{a, b\}$ (si veda l'esempio 7):

1. $\mathbf{q}_0(a, \square)(a, a)\mathbf{q}_1$
2. $\mathbf{q}_0(b, \square)(b, b)\mathbf{q}_1$
3. $\mathbf{q}_1(a, a)(\vec{r}, \vec{r})\mathbf{q}_0$
4. $\mathbf{q}_1(b, b)(\vec{r}, \vec{r})\mathbf{q}_0$
5. $\mathbf{q}_0(\square, \square)(\square, \overleftarrow{\ell})\mathbf{q}_2$
6. $\mathbf{q}_2(\square, a)(\square, \overleftarrow{\ell})\mathbf{q}_2$
7. $\mathbf{q}_2(\square, b)(\square, \overleftarrow{\ell})\mathbf{q}_2$
8. $\mathbf{q}_2(\square, \square)(\overleftarrow{\ell}, \vec{r})\mathbf{q}_3$
9. $\mathbf{q}_3(a, a)(\overleftarrow{\ell}, \vec{r})\mathbf{q}_3$
10. $\mathbf{q}_3(b, b)(\overleftarrow{\ell}, \vec{r})\mathbf{q}_3$
11. $\mathbf{q}_3(a, b)(a, b)\mathbf{q}_3$
12. $\mathbf{q}_3(b, a)(b, a)\mathbf{q}_3$

La macchina $\mathfrak{A}$ copia inizialmente la stringa input $X = s_1 s_2 \cdots s_n$ dal primo nastro sul secondo nastro usando le istruzioni 1, 2, 3 e 4. Il numero delle azioni che esegue $\mathfrak{A}$ per realizzare questa prima fase è $2n$:

$$(\mathbf{q}_0, \bot s_1 s_2 \cdots s_n, \bot\square) \vdash_{\mathfrak{A}}^{2n} (\mathbf{q}_0, s_1 s_2 \cdots s_n \bot\square, s_1 s_2 \cdots s_n \bot\square)\,.$$

Successivamente $\mathfrak{A}$ posiziona la testina del primo nastro in corrispondenza dell'ultimo simbolo di X e quella del secondo nastro in corrispondenza del primo simbolo di X, usando le istruzioni 5, 6, 7 e 8. Per effettuare ciò $\mathfrak{A}$ esegue $n + 2$ azioni:

$$(\mathbf{q}_0, s_1 s_2 \cdots s_n \bot\square, s_1 s_2 \cdots s_n \bot\square) \vdash_{\mathfrak{A}}^{n+2} (\mathbf{q}_3, s_1 s_2 \cdots \bot s_n\square, \square\bot s_1 s_2 \cdots s_n\square)\,.$$

Infine $\mathfrak{A}$ confronta i simboli dei due nastri attraversandoli in ordine inverso: il primo nastro viene attraversato da destra verso sinistra mentre il secondo nastro da sinistra verso destra (istruzioni 9, 10, 11 e 12). Se viene incontrata una coppia di simboli diversi la macchina entra in un loop infinito e l'input non viene accettato. In caso contrario la macchina si ferma, accettando l'input, non appena incontra i blank sui due nastri. Quest'ultima fase, nel caso che la macchina si fermi, richiede n azioni. Pertanto, se X è palindroma si ha che:

$$\widehat{T}_{\mathfrak{A}}(X) = 4n + 2\,.$$

Si osservi poi che $\mathfrak{A}$ è regolare e quindi

$$T_{\mathfrak{A}}(n) = 4n + 2$$

per ogni $n \in \mathbb{N}$. Dunque $T_{\mathfrak{A}}(n) = \Theta(n)$ e il running time di $\mathfrak{A}$ è lineare (nella lunghezza dell'input). ■

5.1 Relazioni tra Macchine di Turing e Macchine di Turing multi-nastro

Vediamo adesso come ogni macchina di Turing multi-nastro $\mathfrak{M}$ possa essere "simulata" da una macchina di Turing $\mathfrak{T}$ con un singolo nastro. Siano

(a) k il numero dei nastri di $\mathfrak{M}$;

(b) $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}\}$ l'alfabeto di $\mathfrak{M}$;

(c) m il minimo indice tale che $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}\} \subseteq \{\mathbf{s}_0, \ldots, \mathbf{s}_m\}$.

L'alfabeto di $\mathfrak{T}$ è l'insieme $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}, \mathbf{s}_{i_0}^{\perp}, \ldots, \mathbf{s}_{i_h}^{\perp}, \square^{\perp}, /, *\}$ dove

(a) $\mathbf{s}_{i_j}^{\perp} = \mathbf{s}_{m+j+1}$, per $j = 0, 1, \ldots, h$;

(b) $\square^{\perp} = \mathbf{s}_{m+h+2}$;

(c) $/ = \mathbf{s}_{m+h+3}$;

(d) $* = \mathbf{s}_{m+h+4}$.

Gli stati di $\mathfrak{T}$ sono suddivisi in due gruppi: *stati semplici* e *stati composti*. Gli stati semplici sono $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{2k+6}$; gli stati composti corrispondono a coppie e triple ordinate del tipo (q, U) e (p, V, W) dove: q e p sono stati di $\mathfrak{M}$; U è una stringa su $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}, \square\}$ di lunghezza al più k che rappresenta i contenuti delle caselle scandite da $\mathfrak{M}$ sui k nastri; V è una stringa su $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}, \square, \text{↰}, \text{↱}\}$ di lunghezza al più k che rappresenta le operazioni che $\mathfrak{M}$ esegue sui nastri; W è una stringa su $\{\mathbf{s}_{i_0}, \ldots, \mathbf{s}_{i_h}, \square, /, \mathbf{s}_{i_0}^{\perp}, \ldots, \mathbf{s}_{i_h}^{\perp}, \square^{\perp}, \text{↰}, \text{↱}, \triangleleft, \triangleright, \curvearrowleft, \curvearrowright, \hookleftarrow, \text{↬}\}$ di lunghezza al più 4 usata per determinare le operazioni che deve eseguire $\mathfrak{T}$ per riprodurre ogni singola operazione di $\mathfrak{M}$.[1]

La simulazione di $\mathfrak{M}$ consiste essenzialmente nel rappresentare ogni k-configurazione $\boldsymbol{\gamma} = (q, X_1 \perp t_1 Y_1, \ldots, X_k \perp t_k Y_k)$ di $\mathfrak{M}$ con la configurazione

$$\mathsf{Simul}(\boldsymbol{\gamma}) = *(q, \varepsilon)/X_1 t_1^{\perp} Y_1/ \cdots /X_k t_k^{\perp} Y_k/* \,.$$

Si osservi che i simboli $*$ e $/$ vengono usati, rispettivamente, come marcatore di margine (sinistro e destro) e come separatore; i simboli $t_1^{\perp}, \ldots, t_k^{\perp}$ (gli *head-marker*) rappresentano invece i contenuti delle caselle scandite da $\mathfrak{M}$ sui k nastri.

Supponiamo che $\mathfrak{M}$ contenga l'istruzione $\boldsymbol{I} = q(t_1 \cdots t_k)(o_1 \cdots o_k)p$. Allora, partendo dalla configurazione $*(q, \varepsilon)/X_1 t_1^{\perp} Y_1/ \cdots /X_k t_k^{\perp} Y_k/*$, $\mathfrak{T}$ simula l'effetto dell'esecuzione di $\boldsymbol{I}$ in due fasi successive. Nella prima fase, $\mathfrak{T}$ percorre il nastro da sinistra verso destra alla ricerca degli head-marker $t_1^{\perp}, \ldots, t_k^{\perp}$. Quando incontra il primo head-marker, cioè $t_1^{\perp}$, si sposta di una casella a destra, "memorizza" il simbolo t_1 nel suo stato interno effettuando la transizione di stato da (q, ε) a (q, t_1) e successivamente prosegue alla ricerca del secondo head-marker mantenendosi nello stato (q, t_1). Quando incontra il secondo head-marker, cioè $t_2^{\perp}$, si sposta di una casella a destra, memorizza il simbolo t_2 nel suo stato interno effettuando la transizione di stato da (q, t_1) a $(q, t_1 t_2)$ e successivamente prosegue alla ricerca del terzo head-marker mantenendosi nello stato $(q, t_1 t_2)$. $\mathfrak{T}$ ripete questo processo fino a quando raggiunge la fine del nastro, cioè incontra il simbolo $*$. A questo punto $\mathfrak{T}$ ha memorizzato l'intera sequenza di simboli $t_1, \ldots, t_k$, cioè si trova nello stato $(q, t_1 \cdots t_k)$, ed ha quindi sufficienti informazioni per determinare che l'azione compiuta da $\mathfrak{M}$ sarà $(o_1, \ldots, o_k)p$. Successivamente $\mathfrak{T}$ si sposta a sinistra, effettua la transizione di stato da $(q, t_1 \cdots t_k)$ a $(p, o_1 \cdots o_k, \varepsilon)$, memorizzando la stringa di operazioni $o_1 \cdots o_k$, ed inizia la seconda fase di simulazione. Nella seconda fase, $\mathfrak{T}$ ripercorre il nastro all'indietro, da destra verso sinistra, e simula le operazioni che $\mathfrak{M}$ esegue sui k nastri, cioè le operazioni $o_1, \ldots, o_k$. Le operazioni vengono simulate in successione, una alla volta, a partire da o_k fino ad o_1 (quindi nell'ordine $o_k, \ldots, o_1$). La simulazione dell'operazione o_i viene realizzata da uno specifico modulo $\mathfrak{A}_i = \mathfrak{A}_i(\boldsymbol{I})$ di istruzioni tramite le quali $\mathfrak{T}$ effettua

[1] Il significato dei simboli $\triangleleft, \triangleright, \curvearrowleft, \curvearrowright, \hookleftarrow, \text{↬}$ sarà chiarito più avanti.

la particolare sequenza di operazioni che richiede la simulazione stessa. Questa sequenza di operazioni viene determinata usando una speciale *stringa (variabile) di controllo* $[o_i]$ che $\mathfrak{T}$ memorizza in una componente del suo stato interno. La stringa di controllo funziona in maniera simile ad uno stack con il top all'estremità sinistra: quando $\mathfrak{T}$ deve eseguire un gruppo di operazioni le inserisce al top dello stack una dopo l'altra, in un ordine specifico. $\mathfrak{T}$ esegue sempre l'operazione che si trova al top; quando la termina, la "consuma" dallo stack facendo un pop e passa all'operazione successiva. La simulazione di o_i ha fine quando lo stack diventa vuoto, cioè la stringa di controllo diventa uguale alla stringa vuota ε. (Maggiori dettagli verranno forniti successivamente).

In maniera più precisa, la seconda fase di simulazione viene dunque effettuata come segue. Procedendo da destra verso sinistra, quando $\mathfrak{T}$ incontra il primo head-marker, cioè $t_k^{\perp}$, effettua la transizione di stato da $(p, o_1 \cdots o_{k-1} o_k, \varepsilon)$ a $(p, o_1 \cdots o_{k-1}, [o_k])$ memorizzando la stringa di controllo $[o_k]$, e successivamente parte la simulazione di o_k mediante il modulo $\mathfrak{A}_k$. Quando $\mathfrak{T}$ ha terminato la simulazione di o_k, ha consumato tutta la stringa di controllo (cioè si ha che $[o_k] = \varepsilon$) e assume quindi lo stato $(p, o_1 \cdots o_{k-1}, \varepsilon)$. A questo punto riprende la scansione del nastro verso sinistra alla ricerca del prossimo head-marker $t_{k-1}^{\perp}$; appena lo incontra memorizza la stringa di controllo $[o_{k-1}]$ assumendo lo stato $(p, o_1 \cdots o_{k-2}, [o_{k-1}])$ e simula l'operazione o_{k-1} mediante il modulo $\mathfrak{A}_{k-1}$; quindi procede verso sinistra alla ricerca del prossimo head-marker $t_{k-2}^{\perp}$, e così via. Quando $\mathfrak{T}$ raggiunge l'estremità sinistra del nastro, cioè incontra il marcatore $*$, ha finito di simulare tutte le operazioni di $\mathfrak{M}$ e si trova quindi nello stato $(p, \varepsilon, \varepsilon)$; si sposta a destra, assume lo stato (p, ε) e riprende il processo di simulazione a partire dalla nuova configurazione.

La prima fase di simulazione viene realizzata dal seguente insieme $\mathfrak{I}_1(\boldsymbol{I})$ di istruzioni, dove $S = \mathbf{S}(\mathfrak{M}) \cup \{\square, /\}$:

$$
\begin{array}{l}
(q, \varepsilon) S \mathrel{\vdash\!\!\!\rightarrow} (q, \varepsilon) \\
(q, \varepsilon) t_1^{\perp} \mathrel{\vdash\!\!\!\rightarrow} (q, t_1) \\
(q, t_1) S \mathrel{\vdash\!\!\!\rightarrow} (q, t_1) \\
(q, t_1) t_2^{\perp} \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2) \\
(q, t_1 t_2) S \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2) \\
(q, t_1 t_2) t_3^{\perp} \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2 t_3) \\
\vdots \\
(q, t_1 t_2 \cdots t_{k-1}) S \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2 \cdots t_{k-1}) \\
(q, t_1 t_2 \cdots t_{k-1}) t_k^{\perp} \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2 \cdots t_{k-1} t_k) \\
(q, t_1 t_2 \cdots t_{k-1} t_k) S \mathrel{\vdash\!\!\!\rightarrow} (q, t_1 t_2 \cdots t_{k-1} t_k) \\
(q, t_1 t_2 \cdots t_{k-1} t_k) * \mathrel{\vdash\!\!\!\leftarrow} (p, o_1 o_2 \cdots o_{k-1} o_k, \varepsilon)\,.
\end{array}
$$

La seconda fase di simulazione viene realizzata dal seguente insieme $\mathfrak{I}_2(\boldsymbol{I})$ di istruzioni, dove $S = \mathbf{S}(\mathfrak{M}) \cup \{\square, /\}$:

$$
\begin{array}{l}
(p, o_1 \cdots o_{k-1} o_k, \varepsilon) S^{\curvearrowleft} (p, o_1 \cdots o_{k-1} o_k, \varepsilon) \\
(p, o_1 \cdots o_{k-1} o_k, \varepsilon) t_k^{\perp} t_k^{\perp} (p, o_1 \cdots o_{k-1}, [o_k]) \\
\mathfrak{A}_k \\
(p, o_1 \cdots o_{k-1}, \varepsilon) S^{\curvearrowleft} (p, o_1 \cdots o_{k-1}, \varepsilon) \\
(p, o_1 \cdots o_{k-1}, \varepsilon) t_{k-1}^{\perp} t_{k-1}^{\perp} (p, o_1 \cdots o_{k-2}, [o_{k-1}]) \\
\mathfrak{A}_{k-1} \\
\vdots \\
(p, o_1, \varepsilon) S^{\curvearrowleft} (p, o_1, \varepsilon) \\
(p, o_1, \varepsilon) t_1^{\perp} t_1^{\perp} (p, \varepsilon, [o_1]) \\
\mathfrak{A}_1 \\
(p, \varepsilon, \varepsilon) S^{\curvearrowleft} (p, \varepsilon, \varepsilon) \\
(p, \varepsilon, \varepsilon) *^{\curvearrowright} (p, \varepsilon) \, .
\end{array}
$$

Per ogni operazione o_i, il contenuto iniziale della stringa di controllo $[o_i]$ e il modulo di istruzioni $\mathfrak{A}_i$ sono definiti come segue.

Se $o_i = t \in \mathbf{S}(\mathfrak{M}) \cup \{\square\}$ allora per simulare l'effetto di o_i, la macchina $\mathfrak{T}$ deve semplicemente stampare il simbolo $t^{\perp}$ e successivamente si deve muovere a sinistra. Per cui:

(1) $[o_i] = t^{\perp} \curvearrowleft$ e

(2) $\mathfrak{A}_i = \{(p, o_1 \cdots o_{i-1}, t^{\perp} \curvearrowleft) t_i^{\perp} t^{\perp} (p, o_1 \cdots o_{i-1}, \curvearrowleft), (p, o_1 \cdots o_{i-1}, \curvearrowleft) t^{\perp} \curvearrowleft (p, o_1 \cdots o_{i-1}, \varepsilon)\}$.

Se $o_i = \curvearrowright$ allora nel simulare l'effetto di o_i, la macchina $\mathfrak{T}$ potrebbe trovarsi di fronte ad un problema: se la casella scandita sull'i-esimo nastro di $\mathfrak{M}$ si trova all'estremità destra del nastro stesso, $\mathfrak{T}$ deve "shiftare" verso destra tutti i simboli del nastro che seguono $t_i^{\perp}$ (oppure shiftare verso sinistra tutti i simboli del nastro che si trovano alla sinistra di $t_i^{\perp}$, compreso $t_i^{\perp}$) in modo da lasciare spazio al blank. Per cui, inizialmente $\mathfrak{T}$ si sposta alla destra di $t_i^{\perp}$ e controlla il simbolo t che incontra. Se questo simbolo è diverso dal separatore /, non è necessario effettuare alcuno shift e $\mathfrak{T}$ semplicemente sostituisce t con il corrispondente head-marker $t^{\perp}$ e poi si sposta a sinistra; in caso contrario $\mathfrak{T}$ deve effettuare lo shift a destra dei simboli. Per fare ciò $\mathfrak{T}$ stampa inizialmente il simbolo $*$ per ricordarsi da dove deve iniziare lo shift; successivamente si sposta alla fine del nastro eseguendo un "loop di spostamento a destra" fino a quando incontra il marcatore $*$; quindi si sposta di una casella a destra e stampa $*$. A questo punto, ripercorre il nastro all'indietro shiftando di una casella verso destra ogni simbolo che incontra fino a $*$. Usiamo il simbolo $\triangleright$ per indicare l'operazione di controllo del simbolo alla destra di $t_i^{\perp}$ e usiamo i simboli $\looparrowright$ e $\curvearrowright$ per indicare le operazioni di loop di spostamento a destra e l'operazione di shifting verso destra dei simboli, rispettivamente.

Allora $[o_i] = t_i \upharpoonright \rhd$ e $\mathfrak{A}_i$ consiste nelle seguenti istruzioni:

$$
\begin{array}{l}
(p, o_1 \cdots o_{i-1}, t_i \upharpoonright \rhd) t_i^{\perp} t_i (p, o_1 \cdots o_{i-1}, \upharpoonright \rhd) \\
(p, o_1 \cdots o_{i-1}, \upharpoonright \rhd) t_i \upharpoonright (p, o_1 \cdots o_{i-1}, \rhd) \\
\{ \\
\quad (p, o_1 \cdots o_{i-1}, \rhd) t t^{\perp} (p, o_1 \cdots o_{i-1}, \leftarrow) \\
\quad (p, o_1 \cdots o_{i-1}, \leftarrow) t^{\perp} \leftarrow (p, o_1 \cdots o_{i-1}, \varepsilon) \ : \ t \in \mathbf{S}(\mathfrak{M}) \cup \{\Box\} \\
\} \\
(p, o_1 \cdots o_{i-1}, \rhd) / * (p, o_1 \cdots o_{i-1}, \upharpoonright \leftarrow\!\!\!\!\circ) \\
(p, o_1 \cdots o_{i-1}, \upharpoonright \leftarrow\!\!\!\!\circ) * \upharpoonright (p, o_1 \cdots o_{i-1}, \leftarrow\!\!\!\!\circ) \\
\{ \\
\quad (p, o_1 \cdots o_{i-1}, \leftarrow\!\!\!\!\circ) s \upharpoonright (p, o_1 \cdots o_{i-1}, \leftarrow\!\!\!\!\circ) \ : \ s \in \mathbf{S}(\mathfrak{T}) \setminus \{*\} \\
\} \\
(p, o_1 \cdots o_{i-1}, \leftarrow\!\!\!\!\circ) * \upharpoonright (p, o_1 \cdots o_{i-1}, * \leftarrow \leftarrow \curvearrowright) \\
(p, o_1 \cdots o_{i-1}, * \leftarrow \leftarrow \curvearrowright) \Box * (p, o_1 \cdots o_{i-1}, \leftarrow \leftarrow \curvearrowright) \\
(p, o_1 \cdots o_{i-1}, \leftarrow \leftarrow \curvearrowright) * \leftarrow (p, o_1 \cdots o_{i-1}, \leftarrow \curvearrowright) \\
(p, o_1 \cdots o_{i-1}, \leftarrow \curvearrowright) * \leftarrow (p, o_1 \cdots o_{i-1}, \curvearrowright) \\
\{ \\
\quad (p, o_1 \cdots o_{i-1}, \curvearrowright) s \upharpoonright (p, o_1 \cdots o_{i-1}, s \leftarrow \leftarrow \curvearrowright) \\
\quad (p, o_1 \cdots o_{i-1}, s \leftarrow \leftarrow \curvearrowright) u s (p, o_1 \cdots o_{i-1}, \leftarrow \leftarrow \curvearrowright) \\
\quad (p, o_1 \cdots o_{i-1}, \leftarrow \leftarrow \curvearrowright) s \leftarrow (p, o_1 \cdots o_{i-1}, \leftarrow \curvearrowright) \\
\quad (p, o_1 \cdots o_{i-1}, \leftarrow \curvearrowright) s \leftarrow (p, o_1 \cdots o_{i-1}, \curvearrowright) \ : s \in \mathbf{S}(\mathfrak{T}) \setminus \{*\}, u \in \mathbf{S}(\mathfrak{T}) \\
\} \\
(p, o_1 \cdots o_{i-1}, \curvearrowright) * \Box^{\perp} (p, o_1 \cdots o_{i-1}, \leftarrow) \\
(p, o_1 \cdots o_{i-1}, \leftarrow) \Box^{\perp} \leftarrow (p, o_1 \cdots o_{i-1}, \varepsilon) \, .
\end{array}
$$

Il caso $o_i = \leftarrow$ è "simmetrico" di quello precedente ... (I simboli usati in questo caso per rappresentare le operazioni principali sono: $\lhd$ per il controllo del simbolo a sinistra di $t_i^{\perp}$; $\curvearrowleft$ per lo shift verso sinistra; $\leftarrow\!\!\!\!\circ$ per il loop di spostamento a sinistra).

Il seguente insieme $\mathfrak{I}'$ di istruzioni permette a $\mathfrak{T}$ di passare da ogni configurazione iniziale $\mathtt{Start}(Z)$, dove $Z \in \mathbf{S}(\mathfrak{M})^*$, alla configurazione $\mathsf{Simul}(\mathtt{Start}(Z))$ che rappresenta la

k-configurazione iniziale di $\mathfrak{M}$ corrispondente a Z:

$$
\begin{array}{l}
\{ \\
\quad \mathbf{q}_0 t t^{\perp} \mathbf{q}_1 \\
\quad \mathbf{q}_0 t^{\perp} t^{\perp} \mathbf{q}_0 : t \in \mathbf{S}(\mathfrak{M}) \cup \{\Box\} \\
\} \\
\mathbf{q}_0 // \mathbf{q}_0 \\
\mathbf{q}_0 ** \mathbf{q}_0 \\
\{ \\
\quad \mathbf{q}_1 t^{\perp} {}^{\leftarrow}\!\urcorner \mathbf{q}_1 : t \in \mathbf{S}(\mathfrak{M}) \cup \{\Box\} \\
\} \\
\mathbf{q}_1 \Box / \mathbf{q}_1 \\
\mathbf{q}_1 / {}^{\leftarrow}\!\urcorner \mathbf{q}_2 \\
\mathbf{q}_2 \Box * \mathbf{q}_2 \\
\mathbf{q}_2 * \lceil^{\rightarrow} \mathbf{q}_2 \\
\mathbf{q}_2 / \lceil^{\rightarrow} \mathbf{q}_2 \\
\{ \\
\quad \mathbf{q}_2 t^{\perp} \lceil^{\rightarrow} \mathbf{q}_3 \\
\quad \mathbf{q}_3 t^{\perp} t^{\perp} \mathbf{q}_0 : t \in \mathbf{S}(\mathfrak{M}) \cup \{\Box\} \\
\} \\
\{ \\
\quad \mathbf{q}_3 t \lceil^{\rightarrow} \mathbf{q}_3 : t \in \mathbf{S}(\mathfrak{M}) \\
\} \\
\mathbf{q}_3 // \mathbf{q}_0 \\
\mathbf{q}_3 ** \mathbf{q}_0 \\
\mathbf{q}_3 \Box / \mathbf{q}_4 \\
\mathbf{q}_4 / \lceil^{\rightarrow} \mathbf{q}_4 \\
\mathbf{q}_4 \Box \Box^{\perp} \mathbf{q}_4 \\
\mathbf{q}_4 \Box^{\perp} \lceil^{\rightarrow} \mathbf{q}_5 \\
\vdots \\
\mathbf{q}_{2k-1} \Box / \mathbf{q}_{2k} \\
\mathbf{q}_{2k} / \lceil^{\rightarrow} \mathbf{q}_{2k} \\
\mathbf{q}_{2k} \Box \Box^{\perp} \mathbf{q}_{2k} \\
\mathbf{q}_{2k} \Box^{\perp} \lceil^{\rightarrow} \mathbf{q}_{2k+1} \\
\mathbf{q}_{2k+1} \Box / \mathbf{q}_{2k+1} \\
\mathbf{q}_{2k+1} / \lceil^{\rightarrow} \mathbf{q}_{2k+2} \\
\mathbf{q}_{2k+2} \Box * \mathbf{q}_{2k+2} \\
\mathbf{q}_{2k+2} * {}^{\leftarrow}\!\urcorner \mathbf{q}_{2k+3} \\
\{ \\
\quad \mathbf{q}_{2k+3} s {}^{\leftarrow}\!\urcorner \mathbf{q}_{2k+3} : s \in \mathbf{S}(\mathfrak{T}) \setminus \{*\} \\
\} \\
\mathbf{q}_{2k+3} * \lceil^{\rightarrow} (\mathbf{q}_0, \varepsilon)\,.
\end{array}
$$

Infatti, l'insieme $\mathfrak{I}'$ ha il seguente effetto:
per ogni stringa non vuota $Z = t_1 t_2 \cdots t_m \in (\mathbf{S}(\mathfrak{T}))^*$, se $Z \in (\mathbf{S}(\mathfrak{M}))^*$ allora

$$\texttt{Start}(Z) \vdash^*_{\mathfrak{I}'} *(\mathbf{q}_0, \varepsilon)/t_1^{\perp} t_2 \cdots t_m/\Box^{\perp}/\Box^{\perp}/\cdots\Box^{\perp}/*\,,$$

dove il numero di occorrenze di / è $k+1$, altrimenti, cioè se $Z \notin (\mathbf{S}(\mathfrak{M}))^*$, allora $\mathfrak{I}'(Z)\uparrow$. Similmente,

$$\texttt{Start}(\varepsilon) \vdash^*_{\mathfrak{I}'} *(\mathbf{q}_0, \varepsilon)/\Box^{\perp}/\Box^{\perp}/\Box^{\perp}/\cdots\Box^{\perp}/*\,.$$

Per completare la costruzione della macchina $\mathfrak{T}$ introduciamo l'insieme di istruzioni $\mathfrak{I}''$ che consente a $\mathfrak{T}$ di passare da una configurazione $\mathsf{Simul}(\boldsymbol{\gamma})$, dove $\boldsymbol{\gamma}$ è una k-configurazione terminale di $\mathfrak{M}$, ad un'opportuna configurazione $\mathsf{Term}(\boldsymbol{\gamma})$ che sia terminale rispetto a $\mathfrak{T}$ e tale che $\langle\boldsymbol{\gamma}\rangle = \langle\mathsf{Term}(\boldsymbol{\gamma})\rangle$, cioè il contenuto estratto da $\boldsymbol{\gamma}$ sia uguale al contenuto estratto da $\mathsf{Term}(\boldsymbol{\gamma})$. Per ogni $(q,(t_1,\ldots,t_k)) \in \mathbf{F}(\mathfrak{M})$, sia $\mathfrak{F}((q,(t_1,\ldots,t_k)))$ il seguente insieme di istruzioni:

$$
\begin{array}{l}
\{ \\
\quad (q,\varepsilon)t\mathrel{\vec{\mathsf{l}}}(q,\varepsilon) \\
\quad (q,\varepsilon)t_1^{\perp}\mathrel{\vec{\mathsf{l}}}(q,t_1) \\
\quad (q,t_1)t\mathrel{\vec{\mathsf{l}}}(q,t_1) \\
\quad (q,t_1)t_2^{\perp}\mathrel{\vec{\mathsf{l}}}(q,t_1t_2) \\
\quad (q,t_1t_2)t\mathrel{\vec{\mathsf{l}}}(q,t_1t_2) \\
\quad (q,t_1t_2)t_3^{\perp}\mathrel{\vec{\mathsf{l}}}(q,t_1t_2t_3) \\
\quad \vdots \\
\quad (q,t_1t_2\cdots t_{k-1})t\mathrel{\vec{\mathsf{l}}}(q,t_1t_2\cdots t_{k-1}) \\
\quad (q,t_1t_2\cdots t_{k-1})t_k^{\perp}\mathrel{\vec{\mathsf{l}}}(q,t_1t_2\cdots t_{k-1}t_k) \\
\quad (q,t_1t_2\cdots t_{k-1}t_k)t\mathrel{\vec{\mathsf{l}}}(q,t_1t_2\cdots t_{k-1}t_k) : t \in \mathbf{S}(\mathfrak{M}) \cup \{\square,/\} \\
\} \\
(q,t_1\cdots t_k)*\mathrel{\overleftarrow{\mathsf{l}}}\mathbf{q}_{2k+4} \\
\{ \\
\quad \mathbf{q}_{2k+4}t\mathrel{\overleftarrow{\mathsf{l}}}\mathbf{q}_{2k+4} : t \in (\mathbf{S}(\mathfrak{T}) \cup \{\square\}) \setminus \{*,/\} \\
\} \\
\mathbf{q}_{2k+4}/\square\mathbf{q}_{2k+4} \\
\mathbf{q}_{2k+4}*\mathrel{\vec{\mathsf{l}}}\mathbf{q}_{2k+5} \\
\{ \\
\quad \mathbf{q}_{2k+5}t\mathrel{\vec{\mathsf{l}}}\mathbf{q}_{2k+5} \\
\quad \mathbf{q}_{2k+5}t^{\perp}t\mathbf{q}_{2k+6} : t \in \mathbf{S}(\mathfrak{M}) \cup \{\square\} \\
\}
\end{array}
$$

Poniamo

$$\mathfrak{I}'' = \bigcup_{(q,(t_1,\ldots,t_k))\in\mathbf{F}(\mathfrak{M})} \mathfrak{F}((q,(t_1,\ldots,t_k)))$$

e

$$\mathfrak{T} = \mathfrak{I}' \cup \left(\bigcup_{\boldsymbol{I}\in\mathfrak{M}} (\mathfrak{I}_1(\boldsymbol{I}) \cup \mathfrak{I}_2(\boldsymbol{I}))\right) \cup \mathfrak{I}''\,.$$

Per ogni k-configurazione $\boldsymbol{\gamma} = (q, X_1\perp t_1Y_1, X_2\perp t_2Y_2, \ldots, X_k\perp t_kY_k)$ di $\mathfrak{M}$, sia

$$\mathsf{Term}(\boldsymbol{\gamma}) = *\square X_1\mathbf{q}_{2k+6}t_1Y_1\square X_2t_2^{\perp}Y_2\square\cdots\square X_kt_k^{\perp}Y_k\square*\,.$$

Si osservi che, se $\boldsymbol{\gamma}$ è una k-configurazione di $\mathfrak{M}$ allora:

(1) $\langle\boldsymbol{\gamma}\rangle = \langle\mathsf{Term}(\boldsymbol{\gamma})\rangle$;

(2) $\mathsf{Term}(\boldsymbol{\gamma})$ é terminale rispetto a $\mathfrak{T}$;

(3) se $\boldsymbol{\gamma}$ é terminale rispetto a $\mathfrak{M}$ allora $\mathsf{Simul}(\boldsymbol{\gamma}) \vdash^*_{\mathfrak{I}''} \mathsf{Term}(\boldsymbol{\gamma})$.

Stimiamo adesso il tempo impiegato da $\mathfrak{T}$ per simulare un'intera computazione propria $\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n)$ di $\mathfrak{M}$. Sia $\mathsf{Simul}(\boldsymbol{\Gamma})$ la computazione che effettua $\mathfrak{T}$ per simulare $\boldsymbol{\Gamma}$.

Cominciamo con l'osservare che $\mathsf{Simul}(\boldsymbol{\Gamma})$ è uguale alla concatenazione dei segmenti di computazione $\Gamma', \Gamma(\boldsymbol{\gamma}_0), \Gamma(\boldsymbol{\gamma}_1), \ldots, \Gamma(\boldsymbol{\gamma}_{n-1}), \Gamma''$ dove:

(a) Γ' è il segmento di computazione che esegue $\mathfrak{T}$ (tramite $\mathfrak{I}'$) per passare dalla configurazione iniziale $\mathtt{Start}(Z)$ alla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_0)$;

(b) $\Gamma(\boldsymbol{\gamma}_i)$ è il segmento di computazione che esegue $\mathfrak{T}$ per passare dalla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_i)$ alla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_{i+1})$, per $i = 0, 1, \ldots, n-1$;

(c) Γ'' è il segmento di computazione che esegue $\mathfrak{T}$ per passare (tramite $\mathfrak{I}''$) dalla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_n)$ alla configurazione $\mathsf{Term}(\boldsymbol{\gamma}_n)$.

Sia $Z = \mathtt{Input}(\boldsymbol{\Gamma})$ e sia $j \in \{0, 1, \ldots, n-1\}$. Poiché $\boldsymbol{\gamma}_j \vdash_{\mathfrak{M}} \boldsymbol{\gamma}_{j+1}$, si ha che $|\mathsf{Simul}(\boldsymbol{\gamma}_{j+1})| \leq |\mathsf{Simul}(\boldsymbol{\gamma}_j)| + k$ e quindi $|\mathsf{Simul}(\boldsymbol{\gamma}_{j+1})| \leq |\mathsf{Simul}(\boldsymbol{\gamma}_0)| + (j+1)k \leq |\mathsf{Simul}(\boldsymbol{\gamma}_0)| + nk$. Dato che $|\mathsf{Simul}(\boldsymbol{\gamma}_0)| = \mathsf{O}(|Z| + k)$, dalla precedente disuguaglianza segue che $|\mathsf{Simul}(\boldsymbol{\gamma}_{j+1})| = \mathsf{O}(|Z| + nk)$; pertanto

$$|\mathsf{Simul}(\boldsymbol{\gamma}_i)| = \mathsf{O}(|Z| + nk)\,, \tag{1}$$

per ogni $i = 0, 1, \ldots, n$. Per passare dalla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_j)$ alla configurazione $\mathsf{Simul}(\boldsymbol{\gamma}_{j+1})$, $\mathfrak{T}$ effettua al più k shift del nastro per simulare gli eventuali spostamenti a destra e/o a sinistra di $\mathfrak{M}$ sui k nastri. Il numero delle azioni che esegue $\mathfrak{T}$ per effettuare ciascuno shift è $\mathsf{O}(|\mathsf{Simul}(\boldsymbol{\gamma}_j)| + k)$;[2] pertanto il numero totale delle azioni che esegue $\mathfrak{T}$ per passare da $\mathsf{Simul}(\boldsymbol{\gamma}_j)$ a $\mathsf{Simul}(\boldsymbol{\gamma}_{j+1})$ (che coincide con la lunghezza di $\Gamma(\boldsymbol{\gamma}_j)$) è $\mathsf{O}(k|\mathsf{Simul}(\boldsymbol{\gamma}_j)| + k^2)$. Dalla (1) segue quindi che $\Gamma(\boldsymbol{\gamma}_j)$ ha lunghezza $\mathsf{O}(k|Z| + nk^2)$, per ogni $j = 0, 1, \ldots, n-1$. Pertanto la lunghezza della concatenazione di $\Gamma(\boldsymbol{\gamma}_0), \Gamma(\boldsymbol{\gamma}_1), \ldots, \Gamma(\boldsymbol{\gamma}_{n-1})$ è $\mathsf{O}(nk|Z| + n^2k^2)$. Si osservi infine che i segmenti Γ' e Γ'' hanno lunghezze proporzionali a $|\mathsf{Simul}(\boldsymbol{\gamma}_0)|$ e $|\mathsf{Simul}(\boldsymbol{\gamma}_n)|$, rispettivamente, e quindi, per la (1), le loro lunghezze sono $\mathsf{O}(|Z| + nk)$. Riassumendo otteniamo che la lunghezza totale di $\mathsf{Simul}(\boldsymbol{\Gamma})$ è $\mathsf{O}(nk|Z| + n^2k^2)$.

Teorema 2. *Sia $\mathfrak{M}$ una k-TM. Esiste una TM $\mathfrak{T}$ con un solo nastro che simula $\mathfrak{M}$ tale che per ogni computazione propria $\boldsymbol{\Gamma}$ di $\mathfrak{M}$ si ha che*

$$|\mathsf{Simul}(\boldsymbol{\Gamma})| = \mathsf{O}(k|\boldsymbol{\Gamma}||\mathtt{Input}(\boldsymbol{\Gamma})| + k^2|\boldsymbol{\Gamma}|^2)\,,$$

dove $\mathsf{Simul}(\boldsymbol{\Gamma})$ è la computazione che effettua $\mathfrak{T}$ per simulare $\boldsymbol{\Gamma}$.

Osservazione 1. Se nel precedente teorema riguardiamo k come una costante e assumiamo (ragionevolmente) che $|\mathtt{Input}(\boldsymbol{\Gamma})| \leq |\boldsymbol{\Gamma}|$ (cioè $\mathfrak{M}$ esamina tutta la stringa input durante la computazione) possiamo concludere che

$$|\mathsf{Simul}(\boldsymbol{\Gamma})| = \mathsf{O}(|\boldsymbol{\Gamma}|^2)\,.$$

■

6 Classi di Complessità Temporale

Definizione 11. Sia $t : \mathbb{N} \to \mathbb{N}$ una funzione e sia $\mathfrak{M}$ una TM.
Un LINGUAGGIO $\mathscr{L}$ È RICONOSCIUTO (O ACCETTATO) DA $\mathfrak{M}$ IN TEMPO (LIMITATO DA) t se:

(1) $\mathscr{L}$ è accettato da $\mathfrak{M}$;

(2) per ogni stringa $X \in \mathscr{L}$ si ha che $\mathfrak{M}(X)\!\downarrow^{(\leq t(|X|))}$.

[2] Si osservi che il termine addizionale k tiene conto del fatto che ogni shift incrementa di 1 la lunghezza del nastro di $\mathfrak{T}$.

Una RELAZIONE DI STRINGHE ρ È COMPUTATA DA $\mathfrak{M}$ IN TEMPO (LIMITATO DA) t se:

(1) ρ è computata da $\mathfrak{M}$;

(2) per ogni coppia di stringhe $(X, Y) \in \rho$ si ha che $\mathfrak{M}(X)\downarrow^{(\leq t(|X|))}Y$. ■

Definizione 12. Sia $t : \mathbb{N} \to \mathbb{N}$ una funzione.

(1) Un LINGUAGGIO $\mathscr{L}$ È RICONOSCIBILE IN TEMPO NONDETERMINISTICO t se esiste una NDTM $\mathfrak{M}$ tale che $\mathscr{L}$ è riconosciuto da $\mathfrak{M}$ in tempo t. La CLASSE DEI LINGUAGGI RICONOSCIBILI IN TEMPO NONDETERMINISTICO t è indicata con $\mathbf{NTIME}(t)$.

(2) Un LINGUAGGIO $\mathscr{L}$ È RICONOSCIBILE IN TEMPO DETERMINISTICO t se esiste una DTM $\mathfrak{D}$ tale che $\mathscr{L}$ è riconosciuto da $\mathfrak{D}$ in tempo t. La CLASSE DEI LINGUAGGI RICONOSCIBILI IN TEMPO DETERMINISTICO t è indicata con $\mathbf{TIME}(t)$.

(3) Una FUNZIONE DI STRINGHE f È COMPUTABILE IN TEMPO t se esiste una DTM $\mathfrak{D}$ tale che f è computata da $\mathfrak{D}$ in tempo t. ■

Osservazione 2. Le precedenti definizioni si estendono al caso di funzioni $t : \mathbb{N} \to \mathbb{R}$ qualsiasi sostituendo $t(|X|)$ con la quantità $\max(0, \lceil t(|X|)\rceil)$, dove $\lceil r\rceil$ denota il "ceiling" di r (cioè il più piccolo numero intero maggiore o uguale a r), per ogni $r \in \mathbb{R}$. ■

Teorema 3. *Siano $\mathfrak{M}$ una TM, $\mathscr{L}$ un linguaggio e $t : \mathbb{N} \to \mathbb{N}$ una funzione.*
Le seguenti condizioni sono equivalenti.

(1) $\mathscr{L}$ è riconosciuto da $\mathfrak{M}$ in tempo t;

(2) $\mathscr{L}$ è accettato da $\mathfrak{M}$ e $T_{\mathfrak{M}}(n) \leq t(n)$, per ogni $n \in \mathbb{N}$.

Dimostrazione. Supponiamo che $\mathscr{L}$ sia riconosciuto da $\mathfrak{M}$ in tempo t e sia $n \in \mathbb{N}$. Se $T_{\mathfrak{M}}(n) = -1$ allora, ovviamente, $T_{\mathfrak{M}}(n) \leq t(n)$. Sia quindi $T_{\mathfrak{M}}(n) \neq -1$. Allora esiste una stringa $X \in \mathscr{L}$ tale che $|X| = n$ e $T_{\mathfrak{M}}(n) = \widehat{T}_{\mathfrak{M}}(X)$. Poiché $\mathscr{L}$ è riconosciuto da $\mathfrak{M}$ in tempo t esiste un $m \leq t(|X|) = t(n)$ tale che $\mathfrak{M}(X)\downarrow^{(m)}$. Dalla definizione di $\widehat{T}_{\mathfrak{M}}(X)$ segue che $\widehat{T}_{\mathfrak{M}}(X) \leq m$ e quindi $T_{\mathfrak{M}}(n) \leq t(n)$. Pertanto (1) implica (2). Il viceversa, cioè che (2) implica (1), segue osservando che se $X \in \mathscr{L}$ allora $\mathfrak{M}(X)\downarrow^{(\widehat{T}_{\mathfrak{M}}(X))}$ e $\widehat{T}_{\mathfrak{M}}(X) \leq T_{\mathfrak{M}}(|X|)$. □

Teorema 4. *Siano $\mathfrak{D}$ una DTM, f una funzione di stringhe e $t : \mathbb{N} \to \mathbb{N}$ una funzione.*
Le seguenti condizioni sono equivalenti.

(1) f è computata da $\mathfrak{D}$ in tempo t;

(2) f è computata da $\mathfrak{D}$ e $T_{\mathfrak{D}}(n) \leq t(n)$, per ogni $n \in \mathbb{N}$;

(3) f è computata da $\mathfrak{D}$ e $|\Gamma| \leq t(|\texttt{Input}(\Gamma)|)$, per ogni computazione propria Γ di $\mathfrak{D}$.

Dimostrazione. Segue osservando che, poichè $\mathfrak{D}$ è deterministica, per ogni stringa $X \in \mathscr{L}$ esiste una (ed una) sola computazione di $\mathfrak{D}$ con input X. □

Teorema 5. *Per ogni funzione $t : \mathbb{N} \to \mathbb{N}$, $\mathbf{TIME}(t) \subseteq \mathbf{NTIME}(t)$.*

Dimostrazione. Segue dal Teorema 1. □

Teorema 6. *Sia $\mathscr{L} \in \mathbf{NTIME}(t)$ dove $t(n) \geq n$ per ogni $n \in \mathbb{N}$. Esistono una costante $c \geq 2$ ed una funzione $u(n) = \mathsf{O}(c^{t(n)})$ tali che $\mathscr{L} \in \mathbf{TIME}(u)$.*

Dimostrazione. Si veda "M. Sipser, *Introduction to the Theory of Computation*", teoremi 7.11 e 3.16. □

Definizione 13. Un LINGUAGGIO $\mathscr{L}$ È RICONOSCIBILE IN TEMPO NONDETERMINISTICO POLINOMIALE se esiste una funzione polinomiale $p : \mathbb{N} \to \mathbb{N}$ tale che $\mathscr{L} \in \mathbf{NTIME}(p)$. La CLASSE DEI LINGUAGGI RICONOSCIBILI IN TEMPO NONDETERMINISTICO POLINOMIALE è indicata con **NP**.

Un LINGUAGGIO $\mathscr{L}$ È RICONOSCIBILE IN TEMPO DETERMINISTICO POLINOMIALE se esiste una funzione polinomiale $p : \mathbb{N} \to \mathbb{N}$ tale che $\mathscr{L} \in \mathbf{TIME}(p)$. La CLASSE DEI LINGUAGGI RICONOSCIBILI IN TEMPO DETERMINISTICO POLINOMIALE è indicata con **P**.

Una funzione di stringhe f è COMPUTABILE IN TEMPO POLINOMIALE se esiste una funzione polinomiale $p : \mathbb{N} \to \mathbb{N}$ tale che f è computabile in tempo p. ■

Teorema 7. $\mathbf{P} \subseteq \mathbf{NP}$. *Se* $\mathscr{L} \in \mathbf{NP}$ *allora* $\mathscr{L}$ *è decidibile.*

Dimostrazione. Che $\mathbf{P} \subseteq \mathbf{NP}$ segue immediatamente dal Teorema 5.

Sia $\mathscr{L} \in \mathbf{NP}$ e siano $\mathfrak{M}$ una TM e $p : \mathbb{N} \to \mathbb{N}$ una funzione polinomiale tali che $\mathfrak{M}$ riconosce $\mathscr{L}$ in tempo p. Per decidere se una stringa $Z \in \Sigma^*_{\mathscr{L}}$ è membro di $\mathscr{L}$ possiamo procedere come segue. Sia $S_0 = \{\mathtt{Start}(Z)\}$. Per ogni $i > 0$ sia S_i l'insieme di tutte le $\mathfrak{M}$-configurazioni γ tali che esiste una configurazione $\delta \in S_{i-1}$ per la quale si ha che $\delta \vdash_{\mathfrak{M}} \gamma$.

Poiché $\mathfrak{M}$ riconosce $\mathscr{L}$ in tempo p, si ha che $Z \in \mathscr{L}$ se e solo se qualcuno degli insiemi $S_0, S_1, \ldots S_{p(|Z|)}$ contiene una configurazione terminale rispetto a $\mathfrak{M}$. Dato che ogni insieme S_i può essere effettivamente costruito a partire da S_{i-1} esaminando semplicemente le istruzioni di $\mathfrak{M}$, per ogni $i > 0$, abbiamo il seguente algoritmo per decidere se $Z \in \mathscr{L}$: partendo da S_0, costruiamo la sequenza $S_0, S_1, \ldots S_{p(|Z|)}$ e successivamente controlliamo se qualcuno degli insiemi S_i contiene una configurazione terminale rispetto a $\mathfrak{M}$. Se è così possiamo concludere che $Z \in \mathscr{L}$; altrimenti si ha che $Z \notin \mathscr{L}$. □

Teorema 8. *Sia* $\mathscr{L}$ *un linguaggio. Allora* $\mathscr{L} \in \mathbf{P}$ *se e solo se la funzione caratteristica di* $\mathscr{L}$ *è computabile in tempo polinomiale.*

Dimostrazione. Basta osservare che se $\mathfrak{D}$ è una DTM tale che $C_{\mathscr{L}}$ è computata da $\mathfrak{D}$ in tempo t, allora $\mathscr{L}$ è accettato da $\mathfrak{D}$ in tempo t. Viceversa, se $\mathfrak{D}$ è una DTM tale che $\mathscr{L}$ è accettato da $\mathfrak{D}$ in tempo t, sia $\mathfrak{A}$ la DTM definita da

$$\mathfrak{A} = \mathfrak{D} \cup \{qt\Box\mathbf{q}_{k+1} : (q, t) \in \mathbf{F}(\mathfrak{D})\},$$

dove k è il minimo indice tale che $\mathbf{Q}(\mathfrak{D}) \subseteq \{\mathbf{q}_0, \ldots, \mathbf{q}_k\}$. Si verifica che $C_{\mathscr{L}}$ è computata da $\mathfrak{A}$ in tempo $t + 1$. □

Definizione 14. Una FUNZIONE POLINOMIALE p È SEMPLICE se esistono costanti $a, b, c \in \mathbb{N} \setminus \{0\}$ tali che $p(n) = a + b \cdot n^c$, per ogni $n \in \mathbb{N}$. ■

Teorema 9. *Sia* $p : \mathbb{N} \to \mathbb{N}$ *una funzione polinomiale. Esiste una funzione polinomiale semplice* q *tale che* $p(n) < q(n)$*, per ogni* $n \in \mathbb{N}$*.*

Dimostrazione. Sia δ il grado di p e sia k il coefficiente del termine di grado δ di p. Se $\delta = 0$ il risultato è banale. Supponiamo quindi che $\delta > 0$, e sia $c = \delta$. Poiché

$$\lim_{n \to +\infty} \frac{p(n)}{n^c} = k,$$

esistono $b \in \mathbb{N} \setminus \{0\}$ e $\nu \in \mathbb{N}$ tali che $p(n) < b \cdot n^c$, per ogni $n \geq \nu$.
Sia $a = \max\{p(0), \ldots, p(\nu)\} + 1$. Allora $a \in \mathbb{N} \setminus \{0\}$ e inoltre $p(n) < a + b \cdot n^c$, per ogni $n \in \mathbb{N}$. □

Teorema 10. *Sia $\mathfrak{M}$ una TM.*
Per ogni computazione Γ di $\mathfrak{M}$, si ha che $|\mathtt{Output}(\Gamma)| \leq |\mathtt{Input}(\Gamma)| + |\Gamma| + 1$.

Dimostrazione. Sia $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ e siano $X = \mathtt{Input}(\Gamma)$ e $Y = \mathtt{Output}(\Gamma)$. Si verifica facilmente che $|\gamma_{i+1}| \leq |\gamma_i| + 1$ per ogni $i = 0, 1, \ldots, n-1$ (poichè $\gamma_i \vdash_{\mathfrak{M}} \gamma_{i+1}$), e pertanto $|\gamma_n| \leq |\gamma_0| + n = |\gamma_0| + |\Gamma|$. Si osservi quindi che $|\gamma_0| \leq |X| + 2$ e $|\gamma_n| \geq |Y| + 1$. □

Teorema 11. *Sia $\mathfrak{M}$ una TM.*
Per ogni $(X, Y) \in \varrho(\mathfrak{M})$ e ogni $n \in \mathbb{N}$, se $\mathfrak{M}(X)\downarrow^{(n)} Y$ allora $|Y| \leq |X| + n + 1$.

Dimostrazione. Segue dal teorema precedente. □

Teorema 12. *Se f è una funzione computabile in tempo t, allora $|f(X)| \leq |X| + t(|X|) + 1$, per ogni stringa X nel dominio di f.*

Dimostrazione. Segue immediatamente dal Teorema 11. □

Teorema 13. *Se f e g sono funzioni computabili in tempo polinomiale allora la funzione composta $f \circ g$ è computabile in tempo polinomiale.*

Dimostrazione. Siano $\mathfrak{M}_f$ e $\mathfrak{M}_g$ DTM e p_f e p_g funzioni polinomiali tali che f è computata da $\mathfrak{M}_f$ in tempo p_f e g è computata da $\mathfrak{M}_g$ in tempo p_g. Senza ledere la generalità possiamo assumere che p_f e p_g siano semplici (cf. Teorema 9).

Per computare la funzione composta $f \circ g$ usiamo una 2-DTM $\mathfrak{D}$. Inizialmente $\mathfrak{D}$ copia l'input X dal primo nastro sul secondo nastro, lasciando il primo nastro vuoto (*fase 1*). Successivamente, $\mathfrak{D}$ usa $\mathfrak{M}_g$ per computare $g(X)$ sul secondo nastro (*fase 2*). Quando $\mathfrak{D}$ ha finito la computazione di $g(X)$ copia il risultato (cioè $g(X)$) sul primo nastro (*fase 3*); quindi $\mathfrak{D}$ usa $\mathfrak{M}_f$ per computare $f(g(X))$ sul primo nastro (*fase 4*).

Sia n il minimo indice tale che $\mathbf{Q}(\mathfrak{M}_g) \subseteq \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n\}$.
La fase 1 viene realizzata dal seguente insieme di 2-istruzioni:

$$\begin{aligned}\mathfrak{I}_1 = \{\, & \mathbf{q}_0(t, \square)(\square, t)\mathbf{q}_0 \\ & \mathbf{q}_0(s, \square)(s, \square)\mathbf{q}_0 \\ & \mathbf{q}_0(\square, t)(\triangleright, \triangleright)\mathbf{q}_0 \\ & \mathbf{q}_0(\square, \square)(\square, \triangleleft)\mathbf{q}_1 \\ & \mathbf{q}_1(\square, t)(\square, \triangleleft)\mathbf{q}_1 \\ & \mathbf{q}_1(\square, \square)(\square, \triangleright)\mathbf{q}_2 : t \in \mathbf{S}(\mathfrak{M}_g), s \in \mathbf{S}(\mathfrak{M}_f) \setminus \mathbf{S}(\mathfrak{M}_g)\,\}\,.\end{aligned}$$

La fase 2 viene realizzata dal seguente insieme di 2-istruzioni:

$$\mathfrak{I}_2 = \{\, \mathbf{q}_{i+2}(\square, t)(\square, o)\mathbf{q}_{j+2} : \mathbf{q}_i to \mathbf{q}_j \in \mathfrak{M}_g \,\}\,.$$

La fase 3 viene realizzata dal seguente insieme di 2-istruzioni:

$$\begin{aligned}\mathfrak{I}_3 = \{\, & \mathbf{q}_{i+2}(\square, u)(\square, u)\mathbf{q}_{n+3} \\ & \mathbf{q}_{n+3}(\square, t)(t, \square)\mathbf{q}_{n+3} \\ & \mathbf{q}_{n+3}(\square, s)(\square, s)\mathbf{q}_{n+3} \\ & \mathbf{q}_{n+3}(t, \square)(\triangleright, \triangleright)\mathbf{q}_{n+3} \\ & \mathbf{q}_{n+3}(\square, \square)(\triangleleft, \square)\mathbf{q}_{n+4} \\ & \mathbf{q}_{n+4}(t, \square)(\triangleleft, \square)\mathbf{q}_{n+4} \\ & \mathbf{q}_{n+4}(\square, \square)(\triangleright, \square)\mathbf{q}_{n+5} : (\mathbf{q}_i, u) \in \mathbf{F}(\mathfrak{M}_g), t \in \mathbf{S}(\mathfrak{M}_f), s \in \mathbf{S}(\mathfrak{M}_g) \setminus \mathbf{S}(\mathfrak{M}_f)\,\}\end{aligned}$$

La fase 4 viene realizzata dal seguente insieme di 2-istruzioni:

$$\mathfrak{I}_4 = \{\, \mathbf{q}_{i+n+5}(\Box, t)(\Box, o)\mathbf{q}_{j+n+5} : \mathbf{q}_i to \mathbf{q}_j \in \mathfrak{M}_f \,\}\,.$$

Pertanto $\mathfrak{D} = \mathfrak{I}_1 \cup \mathfrak{I}_2 \cup \mathfrak{I}_3 \cup \mathfrak{I}_4$.

Sia X una stringa appartenente al dominio di $f \circ g$. Stimiamo la lunghezza della computazione Γ di $\mathfrak{D}$ con input X. Cominciamo osservando che Γ è formata dalla concatenazione dei 4 segmenti di computazione Γ_1, Γ_2, Γ_3 e Γ_4 di $\mathfrak{D}$ corrispondenti alle fasi 1, 2, 3 e 4, rispettivamente. Il segmento Γ_1 ha lunghezza $3|X| + 2$; il segmento Γ_2 ha lunghezza al più $p_g(|X|)$; il segmento Γ_3 ha lunghezza $3|g(X)| + 3$; il segmento Γ_4 ha lunghezza al più $p_f(|g(X)|)$. Quindi $|\Gamma| \leq 3|X| + 2 + p_g(|X|) + 3|g(X)| + 3 + p_f(|g(X)|)$. Per il Teorema 12 si ha che $|g(X)| \leq |X| + p_g(|X|) + 1$ che implica che $p_f(|g(X)|) \leq p_f(|X| + p_g(|X|) + 1)$ (poichè p_f è una funzione crescente in quanto polinomiale semplice) e quindi

$$|\Gamma| \leq 3|X| + 2 + p_g(|X|) + 3(|X| + p_g(|X|) + 1) + 3 + p_f(|X| + p_g(|X|) + 1)\,.$$

Pertanto esiste una funzione polinomiale q tale che, per ogni computazione propria Γ di $\mathfrak{D}$ si ha che

$$|\Gamma| \leq q(|\mathtt{Input}(\Gamma)|)\,.$$

Poiché $f \circ g$ è computata da $\mathfrak{D}$, dai teoremi 2 e 4 segue che $f \circ g$ è computabile in tempo polinomiale. □

7 Una caratterizzazione della classe NP

Definizione 15. Sia ρ una relazione.

(1) Un LINGUAGGIO $\mathscr{L}$ RAPPRESENTA ρ se esiste un "simbolo separatore" $/ \in (\Sigma_{\mathscr{L}} \setminus \Sigma_\rho)$ tale che

$$\mathscr{L} = \{X/Y : (X, Y) \in \rho\}\,.$$

(2) Diciamo che ρ È VERIFICABILE IN TEMPO DETERMINISTICO POLINOMIALE se esiste un linguaggio $\mathscr{L} \in \mathbf{P}$ tale che $\mathscr{L}$ rappresenti ρ. In tal caso, ogni DTM che riconosca $\mathscr{L}$ in tempo p, dove p è una funzione polinomiale, è detta un VERIFICATORE (DETERMINISTICO) POLINOMIALE PER ρ.

(3) Diciamo che ρ È POLINOMIALMENTE BILANCIATA se esiste una funzione polinomiale p tale che per ogni coppia di stringhe $(X, Y) \in \rho$ si ha che $|Y| \leq p(|X|)$. ■

Teorema 14. *Sia $\mathscr{L}$ un linguaggio. Allora, $\mathscr{L} \in \mathbf{NP}$ se e solo se esiste una relazione ρ tale che:*

(1) ρ è verificabile in tempo deterministico polinomiale;

(2) ρ è polinomialmente bilanciata;

(3) una stringa X appartiene a $\mathscr{L}$ se e solo se esiste una stringa Y, chiamata un CERTIFICATO (POLINOMIALMENTE) SUCCINTO PER X, *tale che $(X, Y) \in \rho$, cioè*

$$\mathscr{L} = \{X \in \Sigma_\rho^* : (\exists Y \in \Sigma_\rho^*)((X, Y) \in \rho)\}\,.$$

Dimostrazione. Se $\mathscr{L} \in \mathbf{NP}$ esistono una TM $\mathfrak{M}$ ed una funzione polinomiale p tali che $\mathfrak{M}$ riconosce $\mathscr{L}$ in tempo p. Questo implica che una stringa X appartiene a $\mathscr{L}$ se e solo se esiste una computazione (propria) di $\mathfrak{M}$ con input X la cui lunghezza è al più $p(|X|)$. É possibile rappresentare ogni computazione propria Γ di $\mathfrak{M}$ mediante un stringa $\widetilde{\Gamma}$, su un opportuno alfabeto Σ, in modo tale che la seguente relazione ρ sia polinomialmente bilanciata e verificabile in tempo deterministico polinomiale:

$$\rho = \{(\mathtt{Input}(\Gamma), \widetilde{\Gamma}) : \text{“ } \Gamma \text{ è una computazione propria di } \mathfrak{M} \text{ ” } \textsc{and } |\Gamma| \leq p(\mathtt{Input}(\Gamma))\}\,.$$

Si osservi quindi che una stringa X appartiene a $\mathscr{L}$ se e solo se esiste una stringa Y tale che $(X, Y) \in \rho$.

Viceversa, supponiamo che esista una relazione ρ per cui valgano le condizioni (1), (2) e (3) e sia $\mathfrak{P}$ un verificatore polinomiale per ρ. Allora una TM $\mathfrak{M}$ può riconoscere $\mathscr{L}$ in tempo nondeterministico polinomiale come segue. Partendo da una stringa input X, $\mathfrak{M}$ si sposta alla destra di X, stampa il simbolo separatore / e “genera nondeterministicamente” una stringa Y (si veda l'Osservazione 3 seguente). Successivamente $\mathfrak{M}$ verifica se la coppia (X, Y) appartiene a ρ utilizzando il verificatore $\mathfrak{P}$ con l'input X/Y. □

Osservazione 3. Si osservi che le macchine di Turing possono essere usate anche come *generatori di linguaggi*, oltre che come riconoscitori degli stessi.

Una TM $\mathfrak{M}$ GENERA (NONDETERMINISTICAMENTE) UNA STRINGA X se $\mathfrak{M}(\varepsilon)\downarrow X$.

Ad esempio, sia Σ un alfabeto. La seguente NDTM genera tutte (e sole) le stringhe su Σ:

$$\mathfrak{M} = \{\mathbf{q}_0\square\square\mathbf{q}_1, \mathbf{q}_0\square x\mathbf{q}_1, \mathbf{q}_1 x\overset{\curvearrowright}{}\mathbf{q}_0, \mathbf{q}_1\square\overset{\curvearrowleft}{}\mathbf{q}_2, \mathbf{q}_2 x\overset{\curvearrowleft}{}\mathbf{q}_2, \mathbf{q}_2\square\overset{\curvearrowright}{}\mathbf{q}_3 : x \in \Sigma\}\,.$$

Si osservi che $\mathfrak{M}$ genera ogni stringa $X \in \Sigma^*$ in tempo $\mathsf{O}(|X|)$. ■

8 NP-Completezza

Definizione 16. Siano $\mathscr{L}_1$ e $\mathscr{L}_2$ linguaggi. Una RIDUZIONE DA $\mathscr{L}_1$ A $\mathscr{L}_2$ è una funzione $f : \Sigma_1^* \to \Sigma_2^*$, dove Σ_1 è l'alfabeto di $\mathscr{L}_1$ e Σ_2 è l'alfabeto di $\mathscr{L}_2$, tale che $X \in \mathscr{L}_1 \Leftrightarrow f(X) \in \mathscr{L}_2$ per ogni stringa $X \in \Sigma_1^*$.

Una RIDUZIONE POLINOMIALE DA $\mathscr{L}_1$ A $\mathscr{L}_2$ è una riduzione f da $\mathscr{L}_1$ a $\mathscr{L}_2$ tale che f è computabile in tempo polinomiale.
Scriviamo $\mathscr{L}_1 \leq_p \mathscr{L}_2$ se esiste una riduzione polinomiale da $\mathscr{L}_1$ a $\mathscr{L}_2$. ■

Definizione 17. Un linguaggio $\mathscr{L}$ è **NP**-HARD se $\mathscr{S} \leq_p \mathscr{L}$ per ogni linguaggio $\mathscr{S} \in \mathbf{NP}$. $\mathscr{L}$ è **NP**-COMPLETO se $\mathscr{L}$ è **NP**-hard e $\mathscr{L} \in \mathbf{NP}$. ■

Teorema 15. *Siano $\mathscr{L}_1$ e $\mathscr{L}_2$ linguaggi. Se $\mathscr{L}_1 \leq_p \mathscr{L}_2$ e $\mathscr{L}_2 \in \mathbf{P}$ allora $\mathscr{L}_1 \in \mathbf{P}$.*

Dimostrazione. Sia f una riduzione polinomiale da $\mathscr{L}_1$ a $\mathscr{L}_2$. Per il Teorema 8 la funzione caratteristica $C_{\mathscr{L}_2}$ di $\mathscr{L}_2$ è computabile in tempo polinomiale e quindi, per il Teorema 13, è computabile in tempo polinomiale la funzione composta $C_{\mathscr{L}_2} \circ f$. A questo punto si osservi che $C_{\mathscr{L}_2} \circ f$ è la funzione caratteristica di $\mathscr{L}_1$. □

Teorema 16. *Se esiste un linguaggio* **NP***-completo $\mathscr{L}$ tale che $\mathscr{L} \in \mathbf{P}$ allora* $\mathbf{P} = \mathbf{NP}$.

Dimostrazione. Sia $\mathscr{L} \in \mathbf{P}$ tale che $\mathscr{L}$ è **NP**-completo e sia $\mathscr{S} \in \mathbf{NP}$. Poichè $\mathscr{L}$ è **NP**-completo si ha che $\mathscr{S} \leq_p \mathscr{L}$ e quindi $\mathscr{S} \in \mathbf{P}$ per il Teorema 15. Pertanto $\mathbf{NP} \subseteq \mathbf{P}$. □

Teorema 17 (Transitività della relazione $\leq_p$)**.** *Siano $\mathscr{L}_1$, $\mathscr{L}_2$ e $\mathscr{L}_3$ linguaggi. Se $\mathscr{L}_1 \leq_p \mathscr{L}_2$ e $\mathscr{L}_2 \leq_p \mathscr{L}_3$ allora $\mathscr{L}_1 \leq_p \mathscr{L}_3$.*

Dimostrazione. Sia g una riduzione polinomiale da $\mathscr{L}_1$ a $\mathscr{L}_2$ e sia f una riduzione polinomiale da $\mathscr{L}_2$ a $\mathscr{L}_3$. Poiché g ed f sono computabili in tempo polinomiale, per il Teorema 13, la funzione composta $h = f \circ g$ è computabile in tempo polinomiale. Si osservi quindi che h è una riduzione da $\mathscr{L}_1$ a $\mathscr{L}_3$. □

Teorema 18 (Tecnica di dimostrazione dell'NP-hardness)**.** *Siano $\mathscr{L}_1$ e $\mathscr{L}_2$ linguaggi. Se $\mathscr{L}_1$ è* **NP***-hard e $\mathscr{L}_1 \leq_p \mathscr{L}_2$ allora $\mathscr{L}_2$ è* **NP***-hard.*

Dimostrazione. Segue dal Teorema 17. □

Teorema 19 (Tecnica di dimostrazione dell'NP-completezza)**.** *Siano $\mathscr{L}_1$ e $\mathscr{L}_2$ linguaggi. Se $\mathscr{L}_1$ è* **NP***-completo, $\mathscr{L}_1 \leq_p \mathscr{L}_2$ e $\mathscr{L}_2 \in$* **NP***, allora $\mathscr{L}_2$ è* **NP***-completo.*

Dimostrazione. Segue dal Teorema 18. □

9 Problema della soddisfacibilità (SAT)

Siano fissati:

(1) Un insieme di LETTERE PROPOSIZIONALI (ATOMI)

$$\mathscr{P}_0 \quad \mathscr{P}_1 \quad \mathscr{P}_2 \quad \dots$$

(2) Il SIMBOLO DI NEGAZIONE

$$\neg$$

Definizione 18. Un LETTERALE è o un atomo $\mathscr{P}_i$ oppure la NEGAZIONE DI UN ATOMO $\neg\mathscr{P}_i$.
Una CLAUSOLA è un insieme finito di letterali.
L'insieme vuoto di letterali è chiamato CLAUSOLA VUOTA.
Un'ASSEGNAMENTO (O VALUTAZIONE) è una funzione $v : \{\mathscr{P}_0, \mathscr{P}_1, \mathscr{P}_2, \dots\} \to \{\mathsf{True}, \mathsf{False}\}$ che associa ad ogni atomo $\mathscr{P}_i$ il VALORE DI VERITÀ $v(\mathscr{P}_i)$ ("vero" o "falso"). ■

Definizione 19. Sia v un assegnamento.

(1) Per ogni atomo $\mathscr{P}_i$ poniamo

$$(\mathscr{P}_i)^v = v(\mathscr{P}_i) \quad \text{e} \quad (\neg\mathscr{P}_i)^v = \begin{cases} \mathsf{True}, & \text{se } v(\mathscr{P}_i) = \mathsf{False} \\ \mathsf{False}, & \text{altrimenti}\,. \end{cases}$$

(2) v SODDISFA UNA CLAUSOLA C se esiste un letterale $\ell \in C$ tale che $(\ell)^v = \mathsf{True}$.

(3) v SODDISFA UN INSIEME DI CLAUSOLE S, in simboli $v \models S$, se v soddisfa ogni clausola di S.

(4) v NON SODDISFA UN INSIEME DI CLAUSOLE S, in simboli $v \not\models S$, se non esiste alcuna clausola $C \in S$ tale che v soddisfa C. (La negazione di $v \models S$). ■

Definizione 20. Un INSIEME DI CLAUSOLE È SODDISFACIBILE se esiste un assegnamento che lo soddisfa.

Un INSIEME DI CLAUSOLE È INSODDISFACIBILE se esso non è soddisfacibile, cioè se nessun assegnamento lo soddisfa. ■

Si osservi che nessun assegnamento soddisfa la clausola vuota e che ogni assegnamento soddisfa l'insieme vuoto di clausole.

Definizione 21. Il PROBLEMA SAT consiste nel determinare se un dato insieme finito di clausole è soddisfacibile. ■

Rappresenteremo formalmente il problema SAT con il linguaggio $\mathscr{L}_{\mathsf{SAT}}$ descritto di seguito.

L'alfabeto Σ_{SAT} di $\mathscr{L}_{\mathsf{SAT}}$ è costituito dai seguenti simboli:

$$\mathbf{a} \quad \mathbf{b} \quad \mathbf{1} \quad /$$

Per $i = 0, 1, 2 \ldots$, rappresentiamo i letterali $\mathscr{P}_i$ e $\neg\mathscr{P}_i$ con le stringhe $\alpha_i = \mathbf{a}^{\ulcorner} i^{\urcorner}$ e $\beta_i = \mathbf{b}^{\ulcorner} i^{\urcorner}$, rispettivamente, dove $^{\ulcorner} i^{\urcorner}$ è la stringa formata dalla concatenazione di i copie del simbolo **1**. (Si veda la Sezione 4).

Ogni stringa X della forma $/Y_0Y_1 \cdots Y_m$, dove le stringhe $Y_0, Y_1, \ldots, Y_m$ sono le rappresentazioni dei letterali $\ell_0, \ell_1, \ldots, \ell_m$, rispettivamente, rappresenta la clausola $\{\ell_0, \ell_1, \ldots, \ell_m\}$. (Si osservi che una clausola C può essere rappresentata da più stringhe.) La stringa / rappresenta la clausola vuota.

Ogni stringa W della forma $/X_0/X_1/\cdots/X_p$, dove le stringhe $/X_0, /X_1 \ldots, /X_p$ sono rappresentazioni delle clausole $C_0, C_1, \ldots, C_p$, rispettivamente, rappresenta l'insieme di clausole $\{C_0, C_1, \ldots, C_p\}$. (Quindi ogni insieme di clausole, al pari delle clausole, può essere rappresentato da più stringhe.)

Definizione 22. Definiamo formalmente $\mathscr{L}_{\mathsf{SAT}}$ come il linguaggio costituito da tutte le stringe $W \in \Sigma^*_{\mathsf{SAT}}$ tali che W rappresenta un insieme di clausole soddisfacibile. In simboli

$$\mathscr{L}_{\mathsf{SAT}} = \{W \in \Sigma^*_{\mathsf{SAT}} : W \text{ rappresenta un insieme di clausole soddisfacibile}\}$$

■

Esempio 12. Le stringhe /**ab11a1** e /**a1ab11** rappresentano entrambe la stessa clausola $\{\mathscr{P}_0, \mathscr{P}_1, \neg\mathscr{P}_2\}$. La stringa /**bb1a** rappresenta la clausola $\{\mathscr{P}_0, \neg\mathscr{P}_0, \neg\mathscr{P}_1\}$. La stringa /**1ab11** non rappresenta alcuna clausola. La stringa $W =$ /**ab11a1**/**a111**/**bb1a** rappresenta l'insieme di clausole $\{\{\mathscr{P}_0, \mathscr{P}_1, \neg\mathscr{P}_2\}, \{\mathscr{P}_3\}, \{\mathscr{P}_0, \neg\mathscr{P}_0, \neg\mathscr{P}_1\}\}$. Si osservi che $W \in \mathscr{L}_{\mathsf{SAT}}$ dato che ogni assegnamento v per cui $v(\mathscr{P}_0) = v(\mathscr{P}_3) = \mathsf{True}$ soddisfa l'insieme di clausole che rappresenta W. ■

Teorema 20. $\mathscr{L}_{\mathsf{SAT}} \in \mathbf{NP}$.

Per dimostrare che $\mathscr{L}_{\mathsf{SAT}} \in \mathbf{NP}$ useremo il Teorema 14. Preliminarmente, introduciamo le seguenti definizioni.

Definizione 23. Sia S un insieme di clausole.

(1) Il SUPPORTO DI S è l'insieme $\mathsf{atoms}(S)$ degli atomi che "figurano" nelle clausole di S, cioè

$$\mathsf{atoms}(S) = \{A \in \{\mathscr{P}_0, \mathscr{P}_1, \mathscr{P}_2, \ldots\} : (\exists C \in S)(A \in C \vee \neg A \in C)\}\,.$$

(2) Un insieme H di atomi è un modello di S se:

(2.a) $H \subseteq \mathsf{atoms}(S)$;

(2.b) l'assegnamento v_H definito come segue soddisfa S:

$$v_H(\mathcal{P}_i) = \begin{cases} \mathsf{True}, & \text{se } \mathcal{P}_i \in H \\ \mathsf{False}, & \text{altrimenti}\,, \end{cases}$$

per ogni atomo $\mathcal{P}_i$. ■

Definizione 24. Una stringa $X \in \Sigma^*_{\mathsf{SAT}}$ rappresenta strettamente un insieme H di atomi se X ha la forma $Y_0 Y_1 \cdots Y_m$ dove ogni stringa Y_i rappresenta un atomo (cioè $Y_i \in \{\alpha_0, \alpha_1, \ldots\}$) e $Y_j \neq Y_k$, per ogni $j, k = 0, 1, \ldots, m$ con $j \neq k$.

Sia ρ_{SAT} l'insieme delle coppie ordinate (W, X) di stringe su Σ_{SAT} tali che W rappresenta un insieme S di clausole e X rappresenta strettamente un insieme H di atomi tali che H sia un modello di S. ■

Dalle precedenti definizioni segue che, per ogni stringa $W \in \Sigma^*_{\mathsf{SAT}}$, $W \in \mathscr{L}_{\mathsf{SAT}}$ se e solo se esiste una stringa $X \in \Sigma^*_{\mathsf{SAT}}$ tale che $(W, X) \in \rho_{\mathsf{SAT}}$. Inoltre ρ_{SAT} è polinomialmente bilanciata. Si può ulteriormente dimostrare che ρ_{SAT} è verificabile in tempo deterministico polinomiale. Per il Teorema 14 si ha quindi che $\mathscr{L}_{\mathsf{SAT}} \in \mathbf{NP}$.

Teorema 21. *$\mathscr{L}_{\mathsf{SAT}}$ è* **NP***-hard.*

Dimostrazione. Vedi la Sezione 10. □

Teorema 22 (Teorema di Cook)**.** *$\mathscr{L}_{\mathsf{SAT}}$ è* **NP***-completo.*

Dimostrazione. Segue immediatamente dai teoremi 20 e 21. □

10 Dimostrazione del Teorema di Cook

In questa sezione dimostriamo il Teorema 21.

Preliminari. Nel seguito useremo le seguenti notazioni:

(1) Per ogni insieme finito I indichiamo con $|I|$ la cardinalità di I.

(2) Un'ordinamento di un insieme finito I è una funzione biiettiva $\theta : \{0, 1, \ldots, |I|-1\} \to I$.

(3) Per ogni insieme finito I, ogni ordinamento θ di I e ogni $x \in I$, l'indice di x nell'ordinamento θ è il numero $i \in \{0, 1, \ldots, |I| - 1\}$ tale che $\theta(i) = x$.

(4) Per ogni $a, b, c \in \mathbb{N}$, l'ordinamento lessicografico dell'insieme di terne

$$I = \{0, \ldots, a\} \times \{0, \ldots, b\} \times \{0, \ldots, c\}$$

è l'ordinamento θ di I definito come segue:

(a) $\theta(0) = (0, 0, 0)$;

(b) sia $\theta(n) = (i, j, k) \neq (a, b, c)$; allora

$$\theta(n+1) = \begin{cases} (i, j, k+1), & \text{se } k < c \\ (i, j+1, 0), & \text{se } k = c \text{ e } j < b \\ (i+1, 0, 0), & \text{altrimenti}\,. \end{cases}$$

(5) Per ogni stringa X e ogni indice i tale che $0 \leq i < |X|$, indichiamo con $(X)_i$ l'$(i+1)$-esimo simbolo di X da sinistra verso destra.

(6) Per ogni configurazione $\gamma = XqtY$ poniamo $\widehat{\gamma} = XtY$ (cioè $\widehat{\gamma}$ è la stringa ottenuta da γ eliminando da essa il suo simbolo di stato q).

(7) Per ogni insieme finito $S = \{C_0, C_1, \ldots, C_p\}$ di clausole, poniamo $\mathsf{size}(S) = \sum_{0 \leq i \leq p} |C_i|$ (DIMENSIONE DI S).

(8) Per ogni insieme finito A di atomi, poniamo

$$\nabla A = \{\{\neg a, \neg b\} : a, b \in A \text{ AND } a \neq b\} \cup \{A\}\,.$$

(Si osservi che:

(8.a) $\mathsf{size}(\nabla A) = |A|^2$;

Dimostrazione. Poiché ogni clausola dell'insieme

$$\mathcal{S} = \{\{\neg a, \neg b\} : a, b \in A \text{ AND } a \neq b\}$$

contiene due letterali e $A \notin \mathcal{S}$, si ha che

$$\mathsf{size}(\nabla A) = 2 \cdot |\mathcal{S}| + |A|\,.$$

(Si osservi che A è esso stesso una clausola.)
Il numero delle clausole di $\mathcal{S}$ è uguale al numero dei sottoinsiemi di A contenenti due elementi, e questo numero coincide con il numero delle combinazioni semplici di $|A|$ oggetti presi a 2 a 2; pertanto

$$|\mathcal{S}| = \binom{|A|}{2} = \frac{|A|(|A|-1)}{2}\,.$$

Dalle precedenti relazioni segue che $\mathsf{size}(\nabla A) = |A|(|A|-1) + |A| = |A|^2$. □

(8.b) per ogni assegnamento v, v soddisfa l'insieme di clausole ∇A se e solo se esiste un unico atomo $a \in A$ tale che $v(a) = \mathsf{True}$.)

(9) Per ogni coppia S_1, S_2 di insiemi di clausole poniamo

$$S_1 \wedge S_2 = S_1 \cup S_2 \qquad (\text{CONGIUNZIONE di } S_1 \text{ e } S_2)$$

e

$$S_1 \vee S_2 = \{C \cup K : C \in S_1, K \in S_2\} \qquad (\text{DISGIUNZIONE di } S_1 \text{ e } S_2)\,.$$

(Si osservi che:

(9.a) le operazioni $\wedge$ e $\vee$ sono commutative e associative;

(9.b) $\mathsf{size}(S_1 \wedge S_2) \leq \mathsf{size}(S_1) + \mathsf{size}(S_2)$;

(9.c) $\mathsf{size}(S_1 \vee S_2) \leq |S_2|\mathsf{size}(S_1) + |S_1|\mathsf{size}(S_2)$;

Dimostrazione. Siano $S_1 = \{C_0, C_1, \ldots, C_n\}$ ed $S_2 = \{K_0, K_1, \ldots, K_m\}$. Allora

$$\begin{aligned}
\mathsf{size}(S_1 \vee S_2) &\leq \sum_{i=0}^{n}\sum_{j=0}^{m} |C_i \cup K_j| \\
&\leq \sum_{i=0}^{n}\sum_{j=0}^{m} (|C_i| + |K_j|) \\
&= \sum_{i=0}^{n} \left(\sum_{j=0}^{m} |C_i| + \sum_{j=0}^{m} |K_j| \right) \\
&= \sum_{i=0}^{n} \left(|C_i| \cdot \sum_{j=0}^{m} 1 + \mathsf{size}(S_2) \right) \\
&= \sum_{i=0}^{n} (|C_i| \cdot (m+1) + \mathsf{size}(S_2)) \\
&= \sum_{i=0}^{n} (|C_i| \cdot |S_2| + \mathsf{size}(S_2)) \\
&= |S_2| \cdot \sum_{i=0}^{n} |C_i| + \mathsf{size}(S_2) \cdot \sum_{i=0}^{n} 1 \\
&= |S_2|\mathsf{size}(S_1) + \mathsf{size}(S_2) \cdot (n+1) \\
&= |S_2|\mathsf{size}(S_1) + |S_1|\mathsf{size}(S_2)\,.
\end{aligned}$$

□

(9.d) per ogni assegnamento v, v soddisfa $S_1 \wedge S_2$ se e solo se v soddisfa S_1 e v soddisfa S_2;

(9.e) per ogni assegnamento v, v soddisfa $S_1 \vee S_2$ se e solo se v soddisfa S_1 o v soddisfa S_2;

Dimostrazione. Supponiamo che v soddisfa S_1 e sia C una clausola di $S_1 \vee S_2$. Allora esistono clausole $C' \in S_1$ e $C'' \in S_2$ tali che $C = C' \cup C''$. Poiché v soddisfa S_1 si ha che v soddisfa C' e quindi esiste un letterale $\ell \in C'$ tale che $(\ell)^v = \mathsf{True}$. Ma $\ell \in C' \cup C''$ e quindi v soddisfa $C' \cup C''$, cioè v soddisfa C. Pertanto v soddisfa ogni clausola di $S_1 \vee S_2$. In maniera perfettamente analoga si prova che se v soddisfa S_2 allora v soddisfa $S_1 \vee S_2$.

Viceversa, supponiamo che v soddisfa $S_1 \vee S_2$ ma che, per assurdo, v non soddisfa ne S_1 e neppure S_2. Allora esistono clausole $C \in S_1$ e $K \in S_2$ tali che v non soddisfa C e v non soddisfa K. Ciò implica che v non soddisfa $C \cup K$ e pertanto v non soddisfa $S_1 \vee S_2$ poiché $C \cup K \in S_1 \vee S_2$. Questa è una contraddizione. □

(9.f) se $F_0, F_1, \ldots, F_n$ sono insiemi di *clausole unitarie*, cioè ogni clausola di ogni insieme contiene un solo letterale, allora $\mathsf{size}(\bigvee_{0\leq i\leq n} F_i) \leq (n+1) \cdot \prod_{0\leq i\leq n} |F_i|$.

Dimostrazione. Si procede per induzione su n usando la (9.c) e osservando che se F è un insieme di clausole unitarie allora $\mathsf{size}(F) = |F|$ e inoltre $|\bigvee_{0\leq i\leq m} F_i| \leq \prod_{0\leq i\leq m} |F_i|$, per ogni $m \leq n$.) □

■

Definizione 25. Una TM $\mathfrak{M}$ è in FORMA CANONICA se:

(1) gli stati di $\mathfrak{M}$ sono $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$, per qualche $m > 0$;

(2) i simboli dell'alfabeto di $\mathfrak{M}$ sono $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_r$, per qualche $r \geq 0$;

(3) per ogni $(q, t) \in \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{m-1}\} \times (\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_r\} \cup \{\square\})$ esiste almeno una istruzione di $\mathfrak{M}$ che inizia con qt;

(4) per ogni $t \in \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_r\} \cup \{\square\}$ nessuna istruzione di $\mathfrak{M}$ inizia con $\mathbf{q}_m t$. ■

Osservazione 4. Sia $\mathfrak{M}$ una TM in forma canonica con stati $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$. Si osservi che

> *una configurazione di $\mathfrak{M}$ è terminale rispetto a $\mathfrak{M}$ se e solo se essa contiene lo stato $\mathbf{q}_m$.*

In altri termini, la macchina $\mathfrak{M}$ si ferma esattamente quando raggiunge lo stato $\mathbf{q}_m$.
Nel seguito ci riferiremo a $\mathbf{q}_m$ come allo lo STATO FINALE di $\mathfrak{M}$.

Teorema 23. *Sia $\mathfrak{M}$ una TM. Esiste una TM $\mathfrak{A}$ in forma canonica tale che:*

(1) $\mathbf{S}(\mathfrak{M}) \subseteq \mathbf{S}(\mathfrak{A})$*;*

(2) $\mathfrak{A}(X)\uparrow$*, per ogni* $X \in (\mathbf{S}(\mathfrak{A})^* \setminus \mathbf{S}(\mathfrak{M})^*)$*;*

(3) per ogni $X \in \mathbf{S}(\mathfrak{M})^*$ *ed ogni* $n \in \mathbb{N}$*, se* $\mathfrak{M}(X)\downarrow^{(n)}$ *allora* $\mathfrak{A}(X)\downarrow^{(2|X|+3+n)}$*;*

(4) per ogni $X \in \mathbf{S}(\mathfrak{M})^*$ *ed ogni* $n \in \mathbb{N}$*, se* $\mathfrak{A}(X)\downarrow^{(n)}$ *allora* $n \geq 2|X| + 3$ *e* $\mathfrak{M}(X)\downarrow^{(n-2|X|-3)}$*.*

Dimostrazione. Sia A un insieme di simboli di nastro tale che $A \cap \mathbf{S}(\mathfrak{M}) = \emptyset$ e $A \cup \mathbf{S}(\mathfrak{M}) = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_r\}$, per qualche $r \geq 0$. La macchina $\mathfrak{A}$ avrà come alfabeto l'insieme $A \cup \mathbf{S}(\mathfrak{M})$.

Partendo da una stringa input $X \in \mathbf{S}(\mathfrak{A})^*$, $\mathfrak{A}$ esegue due fasi successive. Nella prima fase percorre il nastro da sinistra verso destra esaminando i simboli di X. Se incontra un simbolo non appartenente a $\mathbf{S}(\mathfrak{M})$, entra in un loop infinito e non accetta X. In caso contrario, dopo aver raggiunto l'estremità destra del nastro, $\mathfrak{A}$ torna indietro e riposiziona la testina in corrispondenza del primo simbolo di X. A questo punto inizia la seconda fase in cui $\mathfrak{A}$ riproduce le operazioni che esegue $\mathfrak{M}$ con input X.

La prima fase viene realizzata dal seguente insieme $\mathfrak{F}_1$ di istruzioni:

$$
\begin{array}{l}
\mathbf{q}_0\mathbf{S}(\mathfrak{M})\mathbb{r}\mathbf{q}_0 \\
\{ \\
\quad \mathbf{q}_0 tt \mathbf{q}_0 : t \in A \\
\} \\
\mathbf{q}_0\square\mathbb{l}\mathbf{q}_1 \\
\mathbf{q}_1\mathbf{S}(\mathfrak{M})\mathbb{l}\mathbf{q}_1 \\
\mathbf{q}_1\square\mathbb{r}\mathbf{q}_2
\end{array}
$$

(Si osservi che, se $X \in \mathbf{S}(\mathfrak{M})^* \setminus \{\varepsilon\}$ allora $\mathbf{q}_0 X \vdash_{\mathfrak{F}_1}^{2|X|+2} \mathbf{q}_2 X$. Inoltre, $\mathbf{q}_0\square \vdash_{\mathfrak{F}_1}^{2} \mathbf{q}_2\square$. Invece, se $X \in ((A \cup \mathbf{S}(\mathfrak{M})^*) \setminus \mathbf{S}(\mathfrak{M})^*)$, allora $\mathfrak{F}_1(X)\uparrow$.)

La seconda fase viene realizzata del seguente insieme di istruzioni:

$$\mathfrak{F}_2 = \{\mathbf{q}_{2+j} to \mathbf{q}_{2+h} : \mathbf{q}_{i_j} to \mathbf{q}_{i_h} \in \mathfrak{M}\},$$

dove $i_0, i_1, \ldots, i_k$ è la sequenza crescente degli indici dei simboli di stato dell'insieme $\mathbf{Q}(\mathfrak{M}) \cup \{\mathbf{q}_0\}$, cioè $\mathbf{Q}(\mathfrak{M}) \cup \{\mathbf{q}_0\} = \{\mathbf{q}_{i_0}, \mathbf{q}_{i_1}, \ldots, \mathbf{q}_{i_k}\}$, con $i_0 < i_1 < \cdots < i_k$. Per completare la costruzione di $\mathfrak{A}$ aggiungiamo il seguente insieme di istruzioni:

$$\mathfrak{F}_3 = \{qtt\mathbf{q}_{k+3} : (q,t) \in \mathbf{F}(\mathfrak{F}_1 \cup \mathfrak{F}_2)\}\,.$$

Poniamo quindi

$$\mathfrak{A} = \mathfrak{F}_1 \cup \mathfrak{F}_2 \cup \mathfrak{F}_3\,.$$

Si osservi che lo stato finale di $\mathfrak{A}$ è $\mathbf{q}_{k+3}$. □

Teorema 24. *Sia $\mathscr{L}$ un linguaggio. $\mathscr{L} \in$ **NP** se e solo se esistono una TM $\mathfrak{M}$ in forma canonica ed una funzione polinomiale semplice p tali che:*

(1) $\mathscr{L}$ è accettato da $\mathfrak{M}$;

(2) per ogni stringa $X \in \mathscr{L}$ si ha che $\mathfrak{M}(X)\downarrow^{(\leq p(|X|))}$.

Dimostrazione. Segue dai teoremi 9 e 23. □

Definizione 26. Sia $\mathfrak{M}$ una TM.

(1) Un n-ARRAY DI $\mathfrak{M}$, dove $n \in \mathbb{N}$, è una sequenza finita $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $n+1$ configurazioni di $\mathfrak{M}$ tali che $|\gamma_0| = |\gamma_1| = \ldots = |\gamma_n| = 2n+2$.

(2) Una n-COMPUTAZIONE DI $\mathfrak{M}$, dove $n \in \mathbb{N}$, è un n-array $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$ soddisfacente le seguenti proprietà:

(2.a) esistono stringhe $U, V \in \{\Box\}^*$ tali che $|U| \geq n$, $|V| \geq n+1-|\langle\gamma_0\rangle|$ e $\gamma_0 = U\mathbf{q}_0\langle\gamma_0\rangle V$;[3]

(2.b) per ogni $i = 0, 1, \ldots, n-1$ almeno una delle condizioni " $\gamma_i \vdash_{\mathfrak{M}} \gamma_{i+1}$ " o " $\gamma_{i+1} = \gamma_i$ " è vera (cioè ogni configurazione nella sequenza è uguale a quella precedente oppure consegue da quest'ultima mediante l'esecuzione di una istruzione di $\mathfrak{M}$);

(2.c) γ_n è una configurazione terminale rispetto a $\mathfrak{M}$.

L'INPUT di Γ è $\mathtt{Input}(\Gamma) = \langle\gamma_0\rangle$; la LUNGHEZZA di Γ è $|\Gamma| = n$. (Si osservi che se Γ è una n-COMPUTAZIONE di $\mathfrak{M}$ allora $|\Gamma| \geq |\mathtt{Input}(\Gamma)| - 1$.) ■

Teorema 25. *Sia $\mathfrak{M}$ una TM e siano $X \in \mathbf{S}(\mathfrak{M})^*$ e $n \in \mathbb{N}$ tali che $n \geq |X| - 1$. Allora,*

$\mathfrak{M}(X)\downarrow^{(\leq n)}$ *se e solo se esiste una n-computazione Γ di $\mathfrak{M}$ tale che* $\mathtt{Input}(\Gamma) = X$.

Teorema 26. *Sia $\mathscr{L}$ un linguaggio. Se $\mathscr{L} \in$ **NP** allora esistono una TM $\mathfrak{M}$ in forma canonica ed una funzione polinomiale semplice p tali che*

$$\mathscr{L} = \{X \in \mathbf{S}(\mathfrak{M})^* : (\exists\Gamma)(\text{“ } \Gamma \textit{ è una } p(|X|)\textit{-computazione di } \mathfrak{M} \textit{ con input } X\text{”})\}\,.$$

Dimostrazione. Segue dai teoremi 24 e 25 osservando che, se p è una funzione polinomiale semplice, allora $p(|X|) \geq |X| - 1$, per ogni stringa X. □

[3] E quindi, poichè $|\gamma_0| = 2n+2$, in quanto Γ è un n-array, si ha che $\gamma_0 = W\mathtt{Start}(X)Z$ dove $X = \langle\gamma_0\rangle$ e W e Z sono le stringhe su $\{\Box\}$ di lunghezze n ed $n+2-|\mathtt{Start}(X)|$, rispettivamente.

Sia adesso $\mathfrak{M}$ una TM in forma canonica e siano $\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m\}$ l'insieme degli stati di $\mathfrak{M}$ e $\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_r\}$ l'alfabeto di $\mathfrak{M}$.

"Costruiamo" la funzione $\delta_{\mathfrak{M}}$ che associa ad ogni coppia (X, n), dove X è una stringa sull'alfabeto di $\mathfrak{M}$ ed n è un numero intero maggiore o uguale a $|X| - 1$, l'insieme di clausole $\delta_{\mathfrak{M}}(X, n)$ tale che $\delta_{\mathfrak{M}}(X, n)$ è soddisfacibile se e solo se $\mathfrak{M}(X)\downarrow^{(\leq n)}$.[4] (Difatti, $\delta_{\mathfrak{M}}(X, n)$ è soddisfacibile se e solo se esiste una n-computazione di $\mathfrak{M}$ con input X (cf. Teorema 25).)

Definizione 27. Per ogni $n \in \mathbb{N}$ siano:

$$\mathbb{A}(n) = \{(i, j, k) : 0 \leq i \leq r + 1, 0 \leq j \leq 2n, 0 \leq k \leq n\}$$

e

$$\mathbb{B}(n) = \{(h, j, k) : 0 \leq h \leq m, 0 \leq j \leq 2n, 0 \leq k \leq n\}.$$

■

Definizione 28. Per ogni $n \in \mathbb{N}$, $(i, j, k) \in \mathbb{A}(n)$ e $(h, j, k) \in \mathbb{B}(n)$ siano:

(1) $\boldsymbol{\alpha}^n_{[i,j,k]}$ il $(p+1)$-esimo atomo della lista $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$, dove p è l'indice della terna (i, j, k) nell'ordinamento lessicografico di $\mathbb{A}(n)$;

(2) $\boldsymbol{\beta}^n_{[h,j,k]}$ il $(p + 1)$-esimo atomo della lista $\mathcal{P}_\ell, \mathcal{P}_{\ell+1}, \mathcal{P}_{\ell+2}, \ldots$ dove p è l'indice della terna (h, j, k) nell'ordinamento lessicografico di $\mathbb{B}(n)$ e $\ell = |\mathbb{A}(n)| = (r + 2)(2n + 1)(n + 1)$;

(3) $\mathcal{A}_n = \{\boldsymbol{\alpha}^n_{[i,j,k]} : (i, j, k) \in \mathbb{A}(n)\}$ e $\mathcal{B}_n = \{\boldsymbol{\beta}^n_{[h,j,k]} : (h, j, k) \in \mathbb{B}(n)\}$.
(Così $\mathcal{A}_n$ è l'insieme dei primi $(r + 2)(2n + 1)(n + 1)$ atomi della lista $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$ e $\mathcal{B}_n$ è l'insieme dei successivi $(m + 1)(2n + 1)(n + 1)$ atomi della stessa lista.) ■

$\delta_{\mathfrak{M}}(X, n)$ avrà come supporto[5] l'insieme di atomi $\mathcal{U}_n = \mathcal{A}_n \cup \mathcal{B}_n$.[6]

Definizione 29. Per ogni $n \in \mathbb{N}$ e ogni n-array $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$ sia v^Γ l'assegnamento definito come segue:

(1) per ogni atomo $a \notin \mathcal{U}_n$: $v^\Gamma(a) = \mathsf{False}$;

(2) per ogni $(i, j, k) \in \mathbb{A}(n)$, con $i < r + 1$:

$$v^\Gamma(\boldsymbol{\alpha}^n_{[i,j,k]}) = \begin{cases} \mathsf{True}, & \text{se } (\widehat{\gamma_k})_j = \mathbf{s}_i \\ \mathsf{False}, & \text{altrimenti} \end{cases} \quad \text{e} \quad v^\Gamma(\boldsymbol{\alpha}^n_{[r+1,j,k]}) = \begin{cases} \mathsf{True}, & \text{se } (\widehat{\gamma_k})_j = \square \\ \mathsf{False}, & \text{altrimenti}; \end{cases}$$

(3) per ogni $(h, j, k) \in \mathbb{B}(n)$:

$$v^\Gamma(\boldsymbol{\beta}^n_{[h,j,k]}) = \begin{cases} \mathsf{True}, & \text{se } (\gamma_k)_j = \mathbf{q}_h \\ \mathsf{False}, & \text{altrimenti}. \end{cases}$$

■

[4] Si veda "Martin D. Davis, Ron Sigal, Elaine J. Weyuker, *Computability, Complexity and Languages*".
[5] cf. Definizione 23.
[6] Si osservi che la cardinalità di $\mathcal{U}_n$ è $(r + m + 3)(2n + 1)(n + 1)$.

$\delta_{\mathfrak{M}}(X,n)$ soddisferà le seguenti condizioni:

(cond.1) per ogni n-array Γ di $\mathfrak{M}$,
" Γ è una n-computazione di $\mathfrak{M}$ con input X " $\Leftrightarrow$ " $v^{\Gamma} \models \delta_{\mathfrak{M}}(X,n)$ ";

(cond.2) se v è un assegnamento che soddisfa $\delta_{\mathfrak{M}}(X,n)$, esiste un unico n-array Γ di $\mathfrak{M}$ tale che $v(a) = v^{\Gamma}(a)$, per ogni atomo $a \in \mathcal{U}_n$.

(E quindi $\delta_{\mathfrak{M}}(X,n)$ è soddisfacibile se e solo se $\mathfrak{M}(X)\downarrow^{(\leq n)}$.)

Si osservi che:

(1) Le condizioni " $v^{\Gamma}(\boldsymbol{\alpha}^n_{[i,j,k]}) = \mathsf{True}$ " e " $v^{\Gamma}(\boldsymbol{\beta}^n_{[h,j,k]}) = \mathsf{True}$ " possono essere espresse, rispettivamente, come segue:

(1.a) " nella $(k+1)$-esima configurazione di Γ, la macchina $\mathfrak{M}$ esamina la $(j+1)$-esima casella del nastro, da destra verso sinistra, e questa casella contiene il simbolo $\mathbf{s}_i$, se $0 \leq i \leq r$, ovvero la casella è vuota, se $i = r+1$ ";

(1.b) " nella $(k+1)$-esima configurazione di Γ, la macchina $\mathfrak{M}$ si trova nello stato interno $\mathbf{q}_h$ ed esamina la $(j+1)$-esima casella del nastro, da destra verso sinistra ".

(2) Per ogni n-array Γ di $\mathfrak{M}$:

(2.a) per ogni $(j,k) \in \{0,\ldots,2n\} \times \{0,\ldots,n\}$ esiste un unico $i \in \{0,\ldots,r+1\}$ tale che $v^{\Gamma}(\boldsymbol{\alpha}^n_{[i,j,k]}) = \mathsf{True}$ (difatti $(\widehat{\gamma_k})_j = \mathbf{s}_i$, se $0 \leq i \leq r$, ovvero $(\widehat{\gamma_k})_j = \square$, nel caso $i = r+1$);

(2.b) per ogni $k \in \{0,\ldots,n\}$ esiste un'unica coppia $(h,j) \in \{0,\ldots,m\} \times \{0,\ldots,2n\}$ tale che $v^{\Gamma}(\boldsymbol{\beta}^n_{[h,j,k]}) = \mathsf{True}$ (difatti $(\gamma_k)_j = \mathbf{q}_h$).

(3) Per ogni Γ_1 e Γ_2 n-array di $\mathfrak{M}$, se $v^{\Gamma_1} = v^{\Gamma_2}$ allora $\Gamma_1 = \Gamma_2$ (come segue dalle proprietà precedenti).

Introduciamo adesso quattro insiemi di clausole la cui unione costituisce $\delta_{\mathfrak{M}}(X,n)$. I primi tre insiemi "forzano" la condizione (cond.1); l'altro la condizione (cond.2).

Insieme di clausole 1. Sia

$$\mathsf{Terminal}_n = \{\{\boldsymbol{\beta}^n_{[m,j,n]} : j \in \{0,\ldots,2n\}\}\}\,.$$

Si osservi che:

(1) Per ogni n-array Γ di $\mathfrak{M}$, $v^{\Gamma} \models \mathsf{Terminal}_n$ se e solo se l'ultima configurazione di Γ è terminale rispetto a $\mathfrak{M}$.

(2) $\mathsf{size}(\mathsf{Terminal}_n) = 2n+1 = \mathsf{O}(n)$.

Insieme di clausole 2. Sia $\mathsf{Initial}^X_n$ l'insieme di clausole definito come segue

(1) se $X = \varepsilon$ allora

$$\mathsf{Initial}^X_n = \{\{\boldsymbol{\alpha}^n_{[r+1,j,0]}\} : 0 \leq j \leq 2n\} \cup \{\{\boldsymbol{\beta}^n_{[0,n,0]}\}\}\,;$$

(2) se $X = \mathbf{s}_{i_0}\mathbf{s}_{i_1}\cdots\mathbf{s}_{i_z}$, dove $0 \leq z \leq n$, allora

$$\begin{aligned}\mathsf{Initial}_n^X = &\{\{\boldsymbol{\alpha}^n_{[r+1,j,0]}\} : 0 \leq j < n\} \cup \\ &\{\{\boldsymbol{\alpha}^n_{[i_j,n+j,0]}\} : 0 \leq j \leq z\} \cup \\ &\{\{\boldsymbol{\alpha}^n_{[r+1,n+z+1+j,0]}\} : 0 \leq j \leq n-z-1\} \cup \\ &\{\{\boldsymbol{\beta}^n_{[0,n,0]}\}\}\,.\end{aligned}$$

Si osservi che, per ogni stringa $X \in \{\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_r\}^*$, con $|X| \leq n+1$, si ha che:

(1) Per ogni n-array Γ di $\mathfrak{M}$, $v^\Gamma \models \mathsf{Initial}_n^X$ se e solo se la prima configurazione di Γ è $W\mathtt{Start}(X)Z$ dove W e Z sono le stringhe su $\{\square\}$ di lunghezze n e $n+2-|\mathtt{Start}(X)|$, rispettivamente.[7]

(2) $\mathsf{size}(\mathsf{Initial}_n^X) = 2n+2 = \mathsf{O}(n)$.

Insieme di clausole 3. Siano

$$\mathfrak{M}_{\mathsf{Print}} = \{I \in \mathfrak{M} : (I)_2 \in \{\mathbf{s}_0, \ldots, \mathbf{s}_r, \square\}\} = \{\mathbf{q}_{i_a} t_{j_a} t_{k_a} \mathbf{q}_{l_a} : 0 \leq a \leq \bar{a}\}\,,$$

$$\mathfrak{M}_{\mathsf{Right}} = \{I \in \mathfrak{M} : (I)_2 = \mathsf{r}\} = \{\mathbf{q}_{i_b} t_{j_b} \mathsf{r} \mathbf{q}_{l_b} : 0 \leq b \leq \bar{b}\}\,,$$

ed

$$\mathfrak{M}_{\mathsf{Left}} = \{I \in \mathfrak{M} : (I)_2 = \mathsf{l}\} = \{\mathbf{q}_{i_c} t_{j_c} \mathsf{l} \mathbf{q}_{l_c} : 0 \leq c \leq \bar{c}\}$$

dove $t_0 = \mathbf{s}_0, t_1 = \mathbf{s}_1, \ldots, t_r = \mathbf{s}_r, t_{r+1} = \square$, gli insiemi delle istruzioni di stampa, movimento a destra e movimento a sinistra di $\mathfrak{M}$, rispettivamente.
Poniamo

$$\mathsf{Cons}_n = \bigwedge_{0\leq k<n} \bigwedge_{0\leq j\leq 2n} (\mathsf{NotHead}(j,k) \vee \mathsf{Ident}(j,k) \vee \mathsf{Print}(j,k) \vee \mathsf{Right}(j,k) \vee \mathsf{Left}(j,k))\,,^{8}$$

dove

(1) $\mathsf{NotHead}(j,k) = \left(\bigvee_{0\leq i\leq r+1} \{\{\boldsymbol{\alpha}^n_{[i,j,k]}\}, \{\boldsymbol{\alpha}^n_{[i,j,k+1]}\}\}\right) \wedge \{\{\neg\boldsymbol{\beta}^n_{[h,j,k]}\} : h \in \{0,\ldots,m\}\}$;

(2) $\mathsf{Ident}(j,k) = \bigvee_{0\leq h\leq m} \bigvee_{0\leq i\leq r+1} \{\{\boldsymbol{\alpha}^n_{[i,j,k]}\}, \{\boldsymbol{\alpha}^n_{[i,j,k+1]}\}, \{\boldsymbol{\beta}^n_{[h,j,k]}\}, \{\boldsymbol{\beta}^n_{[h,j,k+1]}\}\}$;

(3) $\mathsf{Print}(j,k) = \bigvee_{0\leq a\leq \bar{a}} \{\{\boldsymbol{\alpha}^n_{[j_a,j,k]}\}, \{\boldsymbol{\alpha}^n_{[k_a,j,k+1]}\}, \{\boldsymbol{\beta}^n_{[i_a,j,k]}\}, \{\boldsymbol{\beta}^n_{[l_a,j,k+1]}\}\}$;

(4) $\mathsf{Right}(j,k) = \begin{cases} \bigvee_{0\leq b\leq \bar{b}} \{\{\boldsymbol{\alpha}^n_{[j_b,j,k]}\}, \{\boldsymbol{\alpha}^n_{[j_b,j,k+1]}\}, \{\boldsymbol{\beta}^n_{[i_b,j,k]}\}, \{\boldsymbol{\beta}^n_{[l_b,j+1,k+1]}\}\}, & \text{se } j < 2n \\ \{\{\}\}, & \text{altrimenti}\,; \end{cases}$

(5) $\mathsf{Left}(j,k) = \begin{cases} \bigvee_{0\leq c\leq \bar{c}} \{\{\boldsymbol{\alpha}^n_{[j_c,j,k]}\}, \{\boldsymbol{\alpha}^n_{[j_c,j,k+1]}\}, \{\boldsymbol{\beta}^n_{[i_c,j,k]}\}, \{\boldsymbol{\beta}^n_{[l_c,j-1,k+1]}\}\}, & \text{se } j > 0 \\ \{\{\}\}, & \text{altrimenti}\,; \end{cases}$

[7] Si veda la nota a piè pagina N. 3.
[8] Se $n = 0$ allora $\mathsf{Cons}_n = \emptyset$.

Si osservi che:

(1) Per ogni n-array $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$, $v^\Gamma \models \mathsf{Cons}_n$ se e solo se " $\gamma_k \vdash_{\mathfrak{M}} \gamma_{k+1}$ " o " $\gamma_{k+1} = \gamma_k$ ", per ogni $k = 0, 1, \ldots, n-1$.

(2) $\mathsf{size}(\mathsf{NotHead}(j,k)) \leq (r+2)2^{r+2} + (m+1)$.

(3) $\mathsf{size}(\mathsf{Ident}(j,k)) \leq (m+1)(r+2)4^{(m+1)(r+2)}$.

(4) $\mathsf{size}(\mathsf{Print}(j,k)) \leq (\bar{a}+1)4^{\bar{a}+1}$.

(5) $\mathsf{size}(\mathsf{Right}(j,k)) \leq (\bar{b}+1)4^{\bar{b}+1}$.

(6) $\mathsf{size}(\mathsf{Left}(j,k)) \leq (\bar{c}+1)4^{\bar{c}+1}$.

(7) $\mathsf{size}(\mathsf{Cons}_n) = \mathsf{O}(n^2)$ (poiché r, m, $\bar{a}$, $\bar{b}$ e $\bar{c}$ sono costanti).

Nota 1. Forniamo di seguito una dimostrazione della proprietà (1) precedente.

Preliminarmente, si osservi che:

(A) per ogni $k \in \{0, 1, \ldots, n-1\}$:

- (A.1) $\gamma_k \vdash_{\mathfrak{M}_{\mathsf{Print}}} \gamma_{k+1}$ se e solo se esiste $j \in \{0, \ldots, 2n\}$ tale che $(\gamma_k)_j(\widehat{\gamma_k})_j(\widehat{\gamma_{k+1}})_j(\gamma_{k+1})_j \in \mathfrak{M}_{\mathsf{Print}}$ e $(\widehat{\gamma_k})_\ell = (\widehat{\gamma_{k+1}})_\ell$, per ogni $\ell \in \{0, \ldots, 2n\} \setminus \{j\}$;
- (A.2) $\gamma_k \vdash_{\mathfrak{M}_{\mathsf{Right}}} \gamma_{k+1}$ se e solo se esiste $j \in \{0, \ldots, 2n\}$ tale che $(\gamma_k)_j(\widehat{\gamma_k})_j{\upharpoonright}(\gamma_{k+1})_{j+1} \in \mathfrak{M}_{\mathsf{Right}}$ e $(\widehat{\gamma_k})_\ell = (\widehat{\gamma_{k+1}})_\ell$, per ogni $\ell \in \{0, \ldots, 2n\}$;
- (A.3) $\gamma_k \vdash_{\mathfrak{M}_{\mathsf{Left}}} \gamma_{k+1}$ se e solo se esiste $j \in \{0, \ldots, 2n\}$ tale che $(\gamma_k)_j(\widehat{\gamma_k})_j{\upharpoonleft}(\gamma_{k+1})_{j-1} \in \mathfrak{M}_{\mathsf{Left}}$ e $(\widehat{\gamma_k})_\ell = (\widehat{\gamma_{k+1}})_\ell$, per ogni $\ell \in \{0, \ldots, 2n\}$;

(B) per ogni $(j,k) \in \{0, \ldots, 2n\} \times \{0, \ldots, n-1\}$:

- (B.1) $v^\Gamma \models \mathsf{NotHead}(j,k)$ se e solo se $(\gamma_k)_j \notin \{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$ e $(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j$;
- (B.2) $v^\Gamma \models \mathsf{Ident}(j,k)$ se e solo se $(\gamma_k)_j = (\gamma_{k+1})_j \in \{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$ e $(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j$;
- (B.3) $v^\Gamma \models \mathsf{Print}(j,k)$ se e solo se $(\gamma_k)_j(\widehat{\gamma_k})_j(\widehat{\gamma_{k+1}})_j(\gamma_{k+1})_j \in \mathfrak{M}_{\mathsf{Print}}$;
- (B.4) $v^\Gamma \models \mathsf{Right}(j,k)$ se e solo se $(\gamma_k)_j(\widehat{\gamma_k})_j{\upharpoonright}(\gamma_{k+1})_{j+1} \in \mathfrak{M}_{\mathsf{Right}}$ e $(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j$;
- (B.5) $v^\Gamma \models \mathsf{Left}(j,k)$ se e solo se $(\gamma_k)_j(\widehat{\gamma_k})_j{\upharpoonleft}(\gamma_{k+1})_{j-1} \in \mathfrak{M}_{\mathsf{Left}}$ e $(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j$.

Supponiamo che $v^\Gamma \models \mathsf{Cons}_n$ e sia $k \in \{0, \ldots, n-1\}$ tale che $\gamma_{k+1} \neq \gamma_k$. Mostriamo che $\gamma_k \vdash_{\mathfrak{M}} \gamma_{k+1}$.
Poiché $v^\Gamma \models \mathsf{Cons}_n$, si ha che

$$v^\Gamma \models \mathsf{NotHead}(j,k) \vee \mathsf{Ident}(j,k) \vee \mathsf{Print}(j,k) \vee \mathsf{Right}(j,k) \vee \mathsf{Left}(j,k)\,, \qquad \text{(A)}$$

per ogni $j = 0, \ldots, 2n$.

Sia $j^* \in \{0, \ldots, 2n\}$ tale che $(\gamma_k)_{j^*} \in \{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$. Allora, se $j \in \{0, \ldots, 2n\}$ e $j \neq j^*$, si ha che $(\gamma_k)_j \notin \{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$ e quindi

$$v^\Gamma \not\models \mathsf{Ident}(j,k) \vee \mathsf{Print}(j,k) \vee \mathsf{Right}(j,k) \vee \mathsf{Left}(j,k)\,.$$

Da ciò e dalla (A) segue che $v^\Gamma \models \mathsf{NotHead}(j,k)$ e questo implica che $(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j$. Pertanto,

$$(\widehat{\gamma_k})_j = (\widehat{\gamma_{k+1}})_j\,, \tag{B}$$

per ogni $j = 0, \ldots, 2n$, con $j \neq j^*$.
Ora, poiché $(\gamma_k)_{j^*} \in \{\mathbf{q}_0, \ldots, \mathbf{q}_m\}$, si ha che $v^\Gamma \not\models \mathsf{NotHead}(j^*,k)$ e quindi, per la (A),

$$v^\Gamma \models \mathsf{Ident}(j^*,k) \vee \mathsf{Print}(j^*,k) \vee \mathsf{Right}(j^*,k) \vee \mathsf{Left}(j^*,k)\,. \tag{C}$$

D'altra parte, dalla (B) e dal fatto che $\gamma_{k+1} \neq \gamma_k$, segue che $v^\Gamma \not\models \mathsf{Ident}(j^*,k)$ e quindi, per la (C),

$$v^\Gamma \models \mathsf{Print}(j^*,k) \vee \mathsf{Right}(j^*,k) \vee \mathsf{Left}(j^*,k)\,.$$

Quest'ultima relazione e la (B) implicano che $\gamma_k \vdash_{\mathfrak{M}_{\mathsf{Print}} \cup \mathfrak{M}_{\mathsf{Right}} \cup \mathfrak{M}_{\mathsf{Left}}} \gamma_{k+1}$, cioè $\gamma_k \vdash_{\mathfrak{M}} \gamma_{k+1}$. Pertanto, se v^Γ soddisfa Cons_n allora " $\gamma_k \vdash_{\mathfrak{M}} \gamma_{k+1}$ " o " $\gamma_{k+1} = \gamma_k$ ", per ogni $k = 0, \ldots, n-1$.

Il viceversa si dimostra usando argomentazioni simili a quelle precedenti. ■

Insieme di clausole 4. Sia

$$\mathsf{Unique}_n = \mathsf{USS}_n \wedge \mathsf{String}_n$$

dove

$$\mathsf{USS}_n = \bigwedge_{0 \leq k \leq n} \nabla\{\boldsymbol{\beta}^n_{[h,j,k]} : 0 \leq h \leq m, 0 \leq j \leq 2n\}$$

e

$$\mathsf{String}_n = \bigwedge_{0 \leq k \leq n} \bigwedge_{0 \leq j \leq 2n} \nabla\{\boldsymbol{\alpha}^n_{[i,j,k]} : 0 \leq i \leq r+1\}\,.$$

Si osservi che:

(1) Per ogni assegnamento v,

(1.1) $v \models \mathsf{Unique}_n$ se e solo se valgono le seguenti condizioni:

(1.1.1) per ogni $(j,k) \in \{0, \ldots, 2n\} \times \{0, \ldots, n\}$ esiste un unico $i = i(j,k) \in \{0, \ldots, r+1\}$ tale che $v(\boldsymbol{\alpha}^n_{[i(j,k),j,k]}) = \mathsf{True}$;

(1.1.2) per ogni $k \in \{0, \ldots, n\}$ esiste un'unica coppia $(h,j) = (h(k), j(k)) \in \{0, \ldots, m\} \times \{0, \ldots, 2n\}$ tale che $v(\boldsymbol{\beta}^n_{[h(k),j(k),k]}) = \mathsf{True}$;

e quindi

(1.2) se esiste un n-array Γ di $\mathfrak{M}$ tale che $v(a) = v^\Gamma(a)$, per ogni atomo $a \in \mathcal{U}_n$, allora $v \models \mathsf{Unique}_n$

(1.3) se $v \models \mathsf{Unique}_n$ allora l'n-array $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ di $\mathfrak{M}$ tale che

$$(\widehat{\gamma_k})_j = \begin{cases} \mathbf{s}_{i(j,k)}, & \text{se } i(j,k) \neq r+1 \\ \square, & \text{altrimenti} \end{cases} \quad \text{e} \quad (\gamma_k)_{j(k)} = \mathbf{q}_{h(k)}\,,$$

per ogni $(j,k) \in \{0, \ldots, 2n\} \times \{0, \ldots, n\}$, soddisfa $v(a) = v^\Gamma(a)$, per ogni atomo $a \in \mathcal{U}_n$.

(2) $\mathsf{size}(\mathsf{USS}_n) \leq (n+1)(m+1)^2(2n+1)^2 = \mathsf{O}(n^3m^2)$.

(3) $\mathsf{size}(\mathsf{String}_n) \leq (n+1)(2n+1)(r+2)^2 = \mathsf{O}(n^2r^2)$.

(4) $\mathsf{size}(\mathsf{Unique}_n) \leq \mathsf{size}(\mathsf{USS}_n) + \mathsf{size}(\mathsf{String}_n) = \mathsf{O}(n^3)$ (in quanto m ed r sono costanti).

Definizione 30. Poniamo

$$\delta_{\mathfrak{M}}(X, n) = \mathsf{Initial}_n^X \wedge \mathsf{Cons}_n \wedge \mathsf{Terminal}_n \wedge \mathsf{Unique}_n \,.$$

■

Tenendo conto delle osservazioni precedenti, l'insieme di clausole $\delta_{\mathfrak{M}}(X, n)$ appena definito soddisfa le condizioni (cond.1) e (cond.2) e quindi $\delta_{\mathfrak{M}}(X, n)$ è soddisfacibile se e solo se $\mathfrak{M}(X)\downarrow^{(\leq n)}$; inoltre si ha che $\mathsf{size}(\delta_{\mathfrak{M}}(X, n)) = \mathsf{O}(n^3)$. Ulteriormente, poiché

(a) il supporto di $\delta_{\mathfrak{M}}(X, n)$ è $\mathcal{U}_n$;

(b) $\mathcal{U}_n$ è formato dai primi $(r + m + 3)(2n + 1)(n + 1)$ atomi della lista $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$;

(c) ogni atomo $\mathcal{P}_i$ è rappresentato da una stringa su Σ_{SAT} di lunghezza $i + 1$;

l'insieme $\delta_{\mathfrak{M}}(X, n)$ può essere rappresentato da una ben determinata stringa $\Delta_{\mathfrak{M}}(X, n)$ su Σ_{SAT} di lunghezza $\mathsf{O}(n^5)$ e tale stringa può essere computata effettivamente a partire da X e n in tempo proporzionale alla sua stessa lunghezza, e quindi in tempo $\mathsf{O}(n^5)$. Pertanto, se p è una funzione polinomiale semplice, esiste una funzione $\kappa^p_{\mathfrak{M}} : \mathbf{S}(\mathfrak{M})^* \to \Sigma^*_{\mathsf{SAT}}$ (difatti, la funzione $\kappa^p_{\mathfrak{M}}$ definita ponendo $\kappa^p_{\mathfrak{M}}(X) = \Delta_{\mathfrak{M}}(X, p(|X|))$, per ogni $X \in \mathbf{S}(\mathfrak{M})^*$) tale che

(1) $\kappa^p_{\mathfrak{M}}$ è computabile in tempo polinomiale;

(2) per ogni $X \in \mathbf{S}(\mathfrak{M})^*$, la stringa $\kappa^p_{\mathfrak{M}}(X)$ rappresenta l'insieme di clausole $\delta_{\mathfrak{M}}(X, p(|X|))$ e quindi $\mathfrak{M}(X)\downarrow^{(\leq p(|X|))}$ se e solo se $\kappa^p_{\mathfrak{M}}(X) \in \mathscr{L}_{\mathsf{SAT}}$.

Adesso, sia $\mathscr{L} \in \mathbf{NP}$ e siano $\mathfrak{M}$ una TM in forma canonica e p una funzione polinomiale semplice tali che

$$\mathscr{L} = \{X \in \mathbf{S}(\mathfrak{M})^* : (\exists \Gamma)(\text{“ } \Gamma \text{ è una } p(|X|)\text{-computazione di } \mathfrak{M} \text{ con input } X\text{”})\}$$

(cf. Teorema 26). Allora la funzione $\widetilde{\kappa} : \Sigma^*_{\mathscr{L}} \to \Sigma^*_{\mathsf{SAT}}$ definita da

$$\widetilde{\kappa}(X) = \kappa^p_{\mathfrak{M}}(X)\,, \text{ per ogni } X \in \Sigma^*_{\mathscr{L}}\,,$$

cioè la restrizione di $\kappa^p_{\mathfrak{M}}$ a $\Sigma^*_{\mathscr{L}}$,[9] è una riduzione polinomiale da $\mathscr{L}$ in $\mathscr{L}_{\mathsf{SAT}}$ e quindi $\mathscr{L} \leq_p \mathscr{L}_{\mathsf{SAT}}$. Così ogni linguaggio in **NP** è polinomialmente riducibile a $\mathscr{L}_{\mathsf{SAT}}$ e pertanto $\mathscr{L}_{\mathsf{SAT}}$ è **NP**-hard.

11 Catalogo di problemi NP-completi

(Si veda “Michael R. Garey, David S. Johnson, *Computers and Intractability*”.)

[9] Si osservi che $\Sigma^*_{\mathscr{L}} \subseteq \mathbf{S}(\mathfrak{M})^*$.

11.1 Problema 3−SAT

Definizione 31. Una 3-CLAUSOLA è una clausola contenente tre letterali. ■

Definizione 32. Il PROBLEMA 3−SAT consiste nel determinare se un dato insieme finito di 3-clausole è soddisfacibile. ■

Teorema 27. *Il problema* 3−SAT *è* **NP***-completo.*

Dimostrazione. Dobbiamo dimostrare che:

(1) 3−SAT è in **NP**;

(2) 3−SAT è **NP**-hard.

La dimostrazione che 3−SAT è in **NP** segue da argomentazioni simili a quelle usate per dimostrare che SAT è in **NP**.

Per dimostrare che 3−SAT è **NP**-hard usiamo il Teorema 18 e dimostriamo che $\mathsf{SAT} \leq_p \mathsf{3{-}SAT}$, cioè che esiste una funzione f che associa ad ogni insieme S di clausole l'insieme $f(S) = \widetilde{S}$ di 3-clausole tale che S è soddisfacibile se e solo se $\widetilde{S}$ è soddisfacibile; inoltre f deve essere computabile in tempo polinomiale. Procediamo come segue. Sia $S = \{C_1, C_2, \ldots, C_n\}$, dove $C_i = \{\ell_i^1, \ldots, \ell_i^{|C_i|}\}$ per $i = 1, \ldots, n$, e sia $U = \mathsf{atoms}(S)$ il supporto di S. La costruzione di $\widetilde{S}$ coinvolge l'introduzione degli insiemi di atomi $U_1, U_2, \ldots, U_n$ e degli insiemi di 3-clausole $\widetilde{C_1}, \widetilde{C_2}, \ldots, \widetilde{C_n}$ (la coppia $(U_i, \widetilde{C_i})$ corrispondente alla clausola C_i) soddisfacenti alle seguenti condizioni:

(1) gli insiemi $U_1, U_2, \ldots, U_n$ sono a due a due disgiunti;

(2) l'unione di $U_1, U_2, \ldots, U_n$ è disgiunta da U;

(3) per ogni $i = 1, 2, \ldots, n$:

$$|U_i| = \begin{cases} 2, & \text{se } |C_i| = 1 \\ 1, & \text{se } |C_i| = 2 \\ 0, & \text{se } |C_i| = 3 \\ |C_i| - 3, & \text{altrimenti} \end{cases} = ||C_i| - 3|\,;$$

(4) per ogni $i = 1, 2, \ldots, n$:

(4.a) se $|C_i| = 1$, allora $\widetilde{C}_i = \{\{\ell_i^1, u_i^1, u_i^2\}, \{\ell_i^1, u_i^1, \neg u_i^2\}, \{\ell_i^1, \neg u_i^1, u_i^2\}, \{\ell_i^1, \neg u_i^1, \neg u_i^2\}\}$;

(4.b) se $|C_i| = 2$ allora $\widetilde{C}_i = \{\{\ell_i^1, \ell_i^2, u_i^1\}, \{\ell_i^1, \ell_i^2, \neg u_i^1\}\}$;

(4.c) se $|C_i| = 3$ allora $\widetilde{C}_i = \{C_i\}$;

(4.d) se $|C_i| > 3$ allora

$$\widetilde{C}_i = \{\{\ell_i^1, \ell_i^2, u_i^1\}\} \cup \{\{\neg u_i^j, \ell_i^{j+2}, u_i^{j+1}\} : 1 \leq j \leq |C_i| - 4\} \cup \{\{\ell_i^{|C_i|-1}, \ell_i^{|C_i|}, \neg u_i^{|C_i|-3}\}\}\,,$$

dove $U_i = \{u_i^1, u_i^2, \ldots, u_i^{|U_i|}\}$.[10]

[10] Si osservi che se $|C_i| \leq 3$ allora v soddisfa C_i se e solo se v soddisfa l'insieme di clausole $\widetilde{C_i}$, per ogni assegnamento v. Se $|C_i| > 3$ allora ogni assegnamento che soddisfa $\widetilde{C_i}$, soddisfa anche C_i.

Si definisce quindi $\widetilde{S} = \bigcup_{i\in\{1,\ldots,n\}} \widetilde{C}_i$.
Dimostriamo che S è soddisfacibile se e solo se $\widetilde{S}$ è soddisfacibile.

Sia v un assegnamento che soddisfa S. Costruiamo induttivamente la sequenza $v_0, \ldots, v_n$ di assegnamenti dove $v_0 = v$ e v_i è definito come segue, per ogni $i = 1, \ldots, n$. Se $|C_i| \leq 3$ allora $v_i = v_{i-1}$. Se $|C_i| > 3$ poniamo

$$v_i(a) = \begin{cases} \mathsf{False}, & \text{se } (m = 1 \textbf{ OR } m = 2) \textbf{ AND } (a \in \{u_i^1, \ldots, u_i^{|C_i|-3}\}) \\ \mathsf{True}, & \text{se } (m = |C_i| - 1 \textbf{ OR } m = |C_i|) \textbf{ AND } (a \in \{u_i^1, \ldots, u_i^{|C_i|-3}\}) \\ \mathsf{True}, & \text{se } (2 < m < |C_i| - 1) \textbf{ AND } (a \in \{u_i^1, \ldots, u_i^{m-2}\}) \\ \mathsf{False}, & \text{se } (2 < m < |C_i| - 1) \textbf{ AND } (a \in \{u_i^{m-1}, \ldots, u_i^{|C_i|-3}\}) \\ v_{i-1}(a), & \text{altrimenti}, \end{cases}$$

per ogni atomo a, dove $m = \min\{j \in \{1, \ldots, |C_i|\} : v(\ell_i^j) = \mathsf{True}\}$. Procedendo per induzione su i, si verifica che l'assegnamento v_i soddisfa l'insieme di clausole $\bigcup_{1\leq j\leq i} \widetilde{C_j}$, per ogni $i = 1, \ldots, n$. Pertanto v_n soddisfa $\widetilde{S}$. Quindi, se S è soddisfacibile allora anche $\widetilde{S}$ è soddisfacibile.

Viceversa, si verifica che ogni assegnamento che soddisfa $\widetilde{S}$ soddisfa anche S (si veda la nota a piè pagina N. 10). □

11.2 Problema n−SAT ($n > 0$)

Definizione 33. Una n-CLAUSOLA, dove $n > 0$, è una clausola contenente n letterali. ■

Definizione 34. Per ogni $n > 0$, il PROBLEMA n−SAT consiste nel determinare se un dato insieme finito di n-clausole è soddisfacibile. ■

Teorema 28. *Per ogni $n > 2$, il problema* n−SAT *è* **NP***-hard.*

Dimostrazione. Procediamo per induzione su n. Per $n = 3$ abbiamo già dimostrato che 3−SAT è **NP**-hard. Sia $n \geq 3$ e supponiamo, per ipotesi induttiva, che n−SAT è **NP**-hard. Dimostriamo che n−SAT $\leq_p$ (n + 1)−SAT. Sia S un insieme finito di n-clausole. Dobbiamo costruire un insieme $\widetilde{S}$ di $(n+1)$-clausole tale che S è soddisfacibile se e solo se $\widetilde{S}$ è soddisfacibile. Basta usare un nuovo atomo a non appartenente al supporto di S e porre $\widetilde{S} = \{C \cup \{a\}, C \cup \{\neg a\} : C \in S\}$. □

Osservazione 5. È stato dimostrato che il problema 2−SAT è in **P**. Più specificatamente, esiste un algoritmo che risolve il problema 2−SAT in tempo $\mathsf{O}(n + m)$, dove m è il numero delle clausole dell'insieme input S ed n è la cardinalità del supporto di S (il numero di atomi che figurano nelle clausole di S).

11.3 Problemi VertexCover, CLIQUE, IndependentSet

Definizione 35. Sia $G = (V, E)$ un grafo non direzionato.

(1) Un VERTEX COVER per G è un sottoinsieme W di V tale che $\{x, y\} \cap W \neq \emptyset$ per ogni arco $\{x, y\} \in E$.

(2) Una CLIQUE per G è un sottoinsieme W di V tale che $\{x, y\} \in E$ per ogni $x, y \in W$ con $x \neq y$.

(3) Un INDEPENDENT SET per G è un sottoinsieme W di V tale che $\{x, y\} \notin E$ per ogni $x, y \in W$. ■

Definizione 36. Il PROBLEMA VertexCover consiste nel determinare per ogni coppia (G, k), dove $G = (V, E)$ è un grafo e $0 < k \leq |V|$, se esiste un vertex cover W per G tale che $|W| \leq k$. ■

Definizione 37. Il PROBLEMA CLIQUE consiste nel determinare per ogni coppia (G, k), dove $G = (V, E)$ è un grafo e $0 < k \leq |V|$, se esiste una clique W per G tale che $|W| \geq k$. ■

Definizione 38. Il PROBLEMA IndependentSet consiste nel determinare per ogni coppia (G, k), dove $G = (V, E)$ è un grafo e $0 < k \leq |V|$, se esiste un independent set W per G tale che $|W| \geq k$. ■

Teorema 29. *Il problema* VertexCover *è* **NP***-completo.*

Dimostrazione. Dimostriamo che $3{-}\mathsf{SAT} \leq_p \mathsf{VertexCover}$ cosicché VertexCover è **NP**-hard.

Sia $S = \{C_1, C_2, \ldots, C_n\}$ un insieme di 3-clausole, dove $C_i = \{\ell_i^1, \ell_i^2, \ell_i^3\}$ per $i = 1, \ldots, n$, e sia U il supporto di S. Costruiamo il grafo $G_S = (V_S, E_S)$ e l'intero k_S, con $0 < k_S \leq |V_S|$, tali che S è soddisfacibile se e solo se esiste un vertex cover per G_S di cardinalità al più k_S. La costruzione di G_S coinvolge l'introduzione degli insiemi di vertici $V_1, \ldots, V_n$ e dei grafi $G_1 = (V_1 \cup C_1, E_1), \ldots, G_n = (V_n \cup C_n, E_n)$ (il grafo G_i corrispondente alla clausola C_i) soddisfacenti alle seguenti condizioni:

(1) gli insiemi di vertici $V_1, \ldots, V_n$ sono a due a due disgiunti;

(2) l'unione di $V_1, \ldots, V_n$ è disgiunta da $U \cup \{\neg a : a \in U\}$;

(3) per ogni $i = 1, \ldots, n$:

(3.a) $|V_i| = 3$;

(3.b) $E_i = \{\{v_i^1, v_i^2\}, \{v_i^1, v_i^3\}, \{v_i^2, v_i^3\}, \{v_i^1, \ell_i^1\}, \{v_i^2, \ell_i^2\}, \{v_i^3, \ell_i^3\}\}$,

dove $V_i = \{v_i^1, v_i^2, v_i^3\}$.

Si definiscono quindi

$$V_S = U \cup \{\neg a : a \in U\} \cup \left(\bigcup_{i \in \{1,\ldots,n\}} V_i \right)$$

$$E_S = \{\{a, \neg a\} : a \in U\} \cup \left(\bigcup_{i \in \{1,\ldots,n\}} E_i \right)$$

e

$$k_S = |U| + 2|S|\,.$$

Mostriamo che S è soddisfacibile se e solo se esiste un vertex cover per G_S di cardinalità al più k_S.

Sia v un assegnamento che soddisfa S. Poniamo

$$W = \{a : a \in U \text{ AND } v(a) = \mathsf{True}\} \cup \{\neg a : a \in U \text{ AND } v(a) = \mathsf{False}\} \cup \left(\bigcup_{i \in \{1,\ldots,n\}} H_i \right),$$

dove

$$H_i = \begin{cases} \{v_i^2, v_i^3\}, & \text{se } (\ell_i^1)^v = \mathsf{True} \\ \{v_i^1, v_i^3\}, & \text{se } (\ell_i^1)^v = \mathsf{False} \text{ AND } (\ell_i^2)^v = \mathsf{True} \\ \{v_i^1, v_i^2\}, & \text{altrimenti}\,, \end{cases}$$

per ogni $i = 1, \ldots, n$. Allora W è un vertex cover per G_S tale che $|W| = k_S$. Infatti, si osservi innanzitutto che, dalla definizione di W, segue che $|W| = k_S$ e $|W \cap \{a, \neg a\}| = 1$, per ogni atomo $a \in U$; quindi è sufficiente dimostrare che $W \cap e \neq \emptyset$, per ogni arco $e \in \left(\bigcup_{i\in\{1,\ldots,n\}} E_i\right)$. Procediamo come segue. Sia $j \in \{1, \ldots, n\}$. Poiché v soddisfa S, si ha che v soddisfa la clausola C_j e quindi esiste il <u>minimo indice</u> $h \in \{1, 2, 3\}$ tale che $(\ell_j^h)^v = \mathsf{True}$. Dalla definizione di H_j segue che

$$H_j \cap e \neq \emptyset, \text{ per ogni } e \in E_j \setminus \{\{v_j^h, \ell_j^h\}\}$$

e quindi

$$(W \cup \{\ell_j^h\}) \cap e \neq \emptyset, \text{ per ogni } e \in E_j\,.$$

Poiché $(\ell_j^h)^v = \mathsf{True}$ si ha (usando la definizione di W) che $\ell_j^h \in W$ e pertanto

$$W \cap e \neq \emptyset, \text{ per ogni } e \in E_j\,.$$

Data l'arbitrarietà di j concludiamo che, difatti, $W \cap e \neq \emptyset$, per ogni $e \in \left(\bigcup_{i\in\{1,\ldots,n\}} E_i\right)$.

Supponiamo adesso che W sia un vertex cover per G_S tale che $|W| \leq k_S$ e dimostriamo che esiste un assegnamento che soddisfa S. Sia $i \in \{1, \ldots, n\}$. Poichè W contiene almeno un estremo di ogni arco di E_i, esistono almeno due vertici distinti $x, y \in V_i$ tali che $x, y \in W$. Pertanto l'insieme $W \cap \left(\bigcup_{i\in\{1,\ldots,n\}} V_i\right)$ contiene almeno $2n = 2|S|$ vertici distinti. D'altra parte, poichè $W \cap \{a, \neg a\} \neq \emptyset$ vale per ogni $a \in U$, si ha anche che $|W \cap \{a, \neg a : a \in U\}| \geq |U|$. Dato che gli insiemi $\bigcup_{i\in\{1,\ldots,n\}} V_i$ e $\{a, \neg a : a \in U\}$ sono disgiunti otteniamo quindi che $|W| \geq k_S$. Ma allora $|W| = k_S$ e inoltre:

(i) uno solo tra i letterali a e $\neg a$ è un vertice di W, per ogni atomo $a \in U$;

(ii) W contiene esattamente due vertici di V_i, per ogni $i = 1, 2, \ldots, n$.

Sia dunque v l'assegnamento tale che

$$v(a) = \begin{cases} \mathsf{True}, & \text{se } a \in W \\ \mathsf{False}, & \text{altrimenti}\,, \end{cases}$$

per ogni atomo a. Verifichiamo che v soddisfa ogni clausola di S. Sia $i \in \{1, \ldots, n\}$. Poiché $|W \cap V_i| = 2$ (cf. condizione (ii)) e W contiene almeno un estremo di ogni arco di E_i, si ha che $W \cap C_i \neq \emptyset$, cioè W contiene almeno un letterale ℓ_i^h della clausola C_i. Se ℓ_i^h è un atomo, allora si ha che $(\ell_i^h)^v = \mathsf{True}$ (perché $\ell_i^h \in W$) e quindi v soddisfa C_i. Altrimenti, sia $\ell_i^h = \neg a$ dove a è un atomo. Poiché $\neg a \in W$ si ha che $a \notin W$ (cf. condizione (i)) e quindi $v(a) = \mathsf{False}$; questo implica che $(\ell_i^h)^v = (\neg a)^v = \mathsf{True}$ e così v soddisfa C_i. Data l'arbitrarietà di i concludiamo che v soddisfa ogni clausola di S. □

Teorema 30. *Sia $G = (V, E)$ un grafo non direzionato e sia $W \subseteq V$. Le seguenti condizioni sono equivalenti:*

(1) W è un vertex cover per G;

(2) $V \setminus W$ *è un independent set per* G*;*

(3) $V \setminus W$ *è una clique per il* GRAFO COMPLEMENTARE $G^{\complement} = (V, E^{\complement})$ *di* G *dove*

$$E^{\complement} = \{\{x, y\} : x, y \in V, x \neq y, \{x, y\} \notin E\}\,.$$

Teorema 31. *I problemi* CLIQUE *e* IndependentSet *sono* **NP***-completi.*

Dimostrazione. Per dimostrare che i problemi CLIQUE e IndependentSet sono **NP**-hard si dimostri che VertexCover $\leq_p$ IndependentSet e VertexCover $\leq_p$ CLIQUE usando il Teorema 30. (Si osservi che per ogni grafo $G = (V, E)$, ogni $W \subseteq V$ e ogni $0 < k \leq |V|$, si ha che $|W| \leq k$ se e solo se $|V \setminus W| \geq |V| - k$.) □

Osservazione 6. Se nella Definizione 36 del problema VertexCover si sostituisce il vincolo che " $|W| \leq k$ " con " $|W| = k$ " si ottiene un problema equivalente. Infatti, esiste un vertex cover per G di cardinalità minore o uguale a k se e solo se esiste un vertex cover per G avente cardinalità uguale a k. (Se W è un vertex cover per G tale che $|W| < k$, basta aggiungere $k - |W|$ vertici arbitrari a W in modo da ottenere un vertex cover di cardinalità k.) Si osservi inoltre che:

(a) se $k > 1$ allora esiste un vertex cover per G di cardinalità minore di k se e solo se esiste un un vertex cover per G di cardinalità minore o uguale a $k - 1$;

(b) esiste sempre un vertex cover W per G di cardinalità maggiore o uguale a k (basta porre $W = V$);

(c) se $k < |V|$ allora esiste un vertex cover W per G di cardinalità maggiore di k (basta porre $W = V$).

Considerazioni analoghe valgono per i problemi CLIQUE e IndependentSet. ■

Teorema 32. SAT $\leq_p$ CLIQUE*.*

Dimostrazione. Abbiamo dimostrato precedentemente che SAT $\leq_p$ 3−SAT e che 3−SAT $\leq_p$ VertexCover e inoltre abbiamo osservato che VertexCover $\leq_p$ CLIQUE; dalla transitività della relazione " $\leq_p$ " segue quindi che SAT $\leq_p$ CLIQUE. Dimostriamo tuttavia in maniera diretta che SAT $\leq_p$ CLIQUE. Premettiamo la seguente definizione. Due LETTERALI ℓ' ED ℓ'' SONO COMPLEMENTARI, e si scrive $\ell' \bowtie \ell''$, se esiste un atomo a tale che " $\ell' = a$ **AND** $\ell'' = \neg a$ " oppure " $\ell' = \neg a$ **AND** $\ell'' = a$ ". Scriviamo $\ell' \not\bowtie \ell''$ per indicare che ℓ' ed ℓ'' non sono complementari. Si osservi che se $\ell_1, \ell_2, \ldots, \ell_n$ sono letterali "mutuamente non complementari", cioè $\ell_i \not\bowtie \ell_j$ per $0 \leq i < j \leq n$, allora esiste un assegnamento v tale che $(\ell_1)^v = (\ell_2)^v = \cdots = (\ell_n)^v = \mathsf{True}$. (Ad esempio,

$$v(a) = \begin{cases} \mathsf{True}, & \text{se esiste } i \in \{1, 2, \ldots, n\} \text{ tale che } a = \ell_i \\ \mathsf{False}, & \text{altrimenti}\,, \end{cases}$$

per ogni atomo a.)

Dato un insieme $S = \{C_1, C_2, \ldots, C_n\}$ di clausole sia $G_S = (V_S, E_S)$ il grafo non direzionato dove

$$V_S = \bigcup_{1 \leq i \leq n} \{(\ell, i) : \ell \in C_i\}\,,$$

e

$$E_S = \{\{(\ell', i), (\ell'', j)\} : i, j \in \{1, \ldots, n\}, i \neq j, \ell' \in C_i, \ell'' \in C_j, \ell' \not\bowtie \ell''\}.$$

L'insieme S è soddisfacibile se e solo se esiste una clique W per G_S tale che $|W| \geq n$. Infatti sia v un assegnamento che soddisfa S. Per ogni $i = 1, 2, \ldots, n$ esiste un letterale $\ell_i \in C_i$ tale che $(\ell_i)^v = \mathsf{True}$. Sia $W = \{(\ell_i, i) : 1 \leq i \leq n\}$. Ovviamente $W \subseteq V_S$ e inoltre $|W| = n$. Mostriamo che W è una clique per G_S. Siano $(\ell_i, i), (\ell_j, j) \in W$ dove $i \neq j$. Poichè $(\ell_i)^v = (\ell_j)^v = \mathsf{True}$ si ha che $\ell_i \not\bowtie \ell_j$ e quindi $\{(\ell_i, i), (\ell_j, j)\} \in E_S$. Così $\{x, y\} \in E_S$ per ogni $x, y \in W$ con $x \neq y$. Dunque W è una clique per G_S.

Viceversa, supponiamo che esista una clique $W = \{w_1, w_2, \ldots, w_m\}$ per G_S dove $m \geq n$. Per $k = 1, 2, \ldots, m$, sia $w_k = (\ell_{i_k}, i_k)$. Per dimostrare che S è soddisfacibile è sufficiente dimostrare che valgono le seguente proprietà:

(a) $i_j \neq i_h$ per ogni $1 \leq j < h \leq m$ (così $\{i_1, i_2, \ldots, i_n\} = \{1, 2, \ldots, n\}$ perchè $\{i_1, i_2, \ldots, i_n\} \subseteq \{1, 2, \ldots, n\}$);

(b) $\ell_{i_k} \in C_{i_k}$ per ogni $1 \leq k \leq n$;

(c) i letterali $\ell_{i_1}, \ell_{i_2}, \ldots, \ell_{i_n}$ sono mutuamente non complementari.

La (b) è ovvia; (a) e (c) seguono dal fatto che se $1 \leq j < h \leq m$ allora $\{(\ell_{i_j}, i_j), (\ell_{i_h}, i_h)\} \in E_S$ (poichè W è una clique per G_S). □

11.4 Problemi HamiltonianPath e HamiltonianCircuit

Definizione 39. Sia $G = (V, E)$ un grafo non direzionato.

(1) Un CAMMINO in G è una sequenza $(v_0, v_1, \ldots, v_n)$ di vertici di G, dove $n > 0$, tale che $\{v_i, v_{i+1}\} \in E$, per $i = 0, 1, \ldots, n-1$.

(2) La LUNGHEZZA DEL CAMMINO $(v_0, v_1, \ldots, v_n)$ è n.

(3) Un CAMMINO SEMPLICE in G è un cammino $(v_0, v_1, \ldots, v_n)$ in G tale che $v_i \neq v_j$ per ogni $i, j = 0, 1, \ldots, n$ con $i \neq j$.

(4) Un CIRCUITO (O CICLO) SEMPLICE in G è un cammino semplice $(v_0, v_1, \ldots, v_n)$ dove $n > 1$ e $\{v_0, v_n\} \in E$.

(5) Un CAMMINO HAMILTONIANO in G è un cammino semplice $(v_0, v_1, \ldots, v_n)$ dove $V = \{v_0, v_1, \ldots, v_n\}$ (cioè un cammino semplice che include tutti i vertici di G).

(6) Un CIRCUITO HAMILTONIANO in G è un circuito semplice $(v_0, v_1, \ldots, v_n)$ dove $V = \{v_0, v_1, \ldots, v_n\}$ (cioè un circuito semplice che include tutti i vertici di G). ■

Definizione 40. Il PROBLEMA HamiltonianPath consiste nel determinare se un dato grafo G ha un cammino Hamiltoniano. ■

Definizione 41. Il PROBLEMA HamiltonianCircuit consiste nel determinare se un dato grafo G ha un circuito Hamiltoniano. ■

Teorema 33. VertexCover $\leq_p$ HamiltonianCircuit.

Dimostrazione. Sia $G = (V, E)$ una grafo non direzionato e sia $0 < k \leq |V|$. Costruiamo il grafo $\widehat{G} = (\widehat{V}, \widehat{E})$ tale che G ha un vertex cover di cardinalità al più k se e solo se $\widehat{G}$ ha un circuito Hamiltoniano.

Per ogni vertice $v \in V$, sia $\mathsf{Inc}(v)$ l'insieme degli archi di G che incidono su v, cioè

$$\mathsf{Inc}(v) = \{e \in E : v \in e\},$$

e siano $\mathsf{deg}(v) = |\mathsf{Inc}(v)|$ e $e_v^1, e_v^2, \ldots, e_v^{\mathsf{deg}(v)}$ un ordinamento di $\mathsf{Inc}(v)$.

La costruzione di $\widehat{G}$ coinvolge l'introduzione dei seguenti insiemi disgiunti di vertici:

$$A = \{a_1, a_2, \ldots, a_k\}$$

e

$$V' = \bigcup_{e \in E} V'_e$$

dove

$$V'_e = \{(v, e, i), (w, e, i) : 1 \leq i \leq 6\} \quad \text{per ogni arco } e = \{v, w\} \in E$$

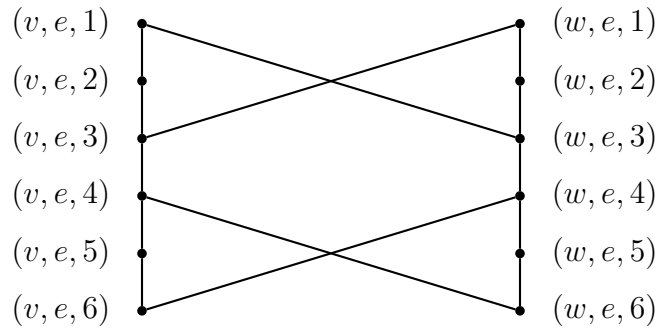
---

e dei seguenti insiemi di archi:

$$E' = \bigcup_{e \in E} E'_e, \qquad E'' = \bigcup_{v \in V} E''_v, \qquad E'''$$

dove

$$\begin{aligned} E'_e = \; & \{\{(v, e, i), (v, e, i+1)\}, \{(w, e, i), (w, e, i+1)\} : 1 \leq i \leq 5\} \cup \\ & \{\{(v, e, 3), (w, e, 1)\}, \{(w, e, 3), (v, e, 1)\}\} \cup \\ & \{\{(v, e, 6), (w, e, 4)\}, \{(w, e, 6), (v, e, 4)\}\} \quad \text{per ogni arco } e = \{v, w\} \in E, \end{aligned}$$



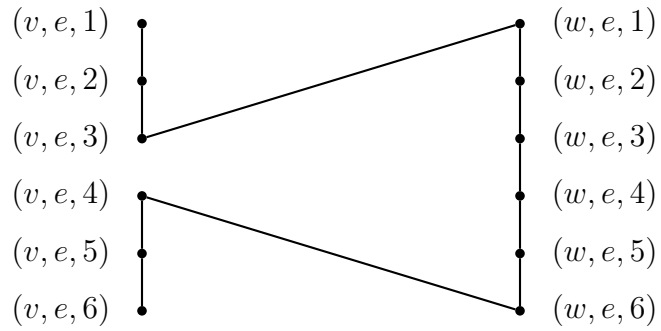
$$E''_v = \{\{(v, e_v^i, 6), (v, e_v^{i+1}, 1)\} : 1 \leq i < \mathsf{deg}(v)\} \quad \text{per ogni vertice } v \in V,$$

$$E''' = \{\{a_i, (v, e_v^1, 1)\}, \{a_i, (v, e_v^{\mathsf{deg}(v)}, 6)\} : 1 \leq i \leq k, v \in V\}.$$

Si osservi che per ogni arco $e = \{v, w\} \in E$, se π è un cammino semplice nel grafo (V'_e, E'_e) dal vertice $(v, e, 1)$ al vertice $(v, e, 6)$ allora o

$$\pi = ((v, e, 1), (v, e, 2), (v, e, 3), (v, e, 4), (v, e, 5), (v, e, 6))$$

oppure π é il seguente cammino:



Poniamo

$$\widehat{V} = A \cup V' \quad \text{e} \quad \widehat{E} = E' \cup E'' \cup E''' .$$

Supponiamo che esista un circuito Hamiltoniano $\pi = (z_0, z_1, \ldots, z_n)$ in $\widehat{G}$.
Siano:

(i) $m = \min(\{i \in \{0, \ldots, n\} : z_i \in A\})$;

(ii) $\pi^* = (x_0, x_1, \ldots, x_n, x_{n+1}) = (z_m, z_{m+1}, \ldots, z_n, z_0, z_1, \ldots, z_m)$;

(iii) $A_i = \{v \in V : \text{“ } \{a_i, (v, e_v^1, 1)\} \text{ è un arco di } \pi^* \text{ ”}\}$, per $i = 1, 2, \ldots, k$.

Allora l'insieme

$$W = \bigcup_{1 \le i \le k} A_i$$

è un vertex cover per G tale che $|W| \le k$.

Infatti, si osservi innanzitutto che per ogni arco $e \in E$ esiste un indice $i \in \{1, 2, \ldots, k\}$ tale che $A_i \cap e \neq \emptyset$. Per cui W è un vertex cover per G. Il fatto che $|W| \le k$ è conseguenza della seguente osservazione.

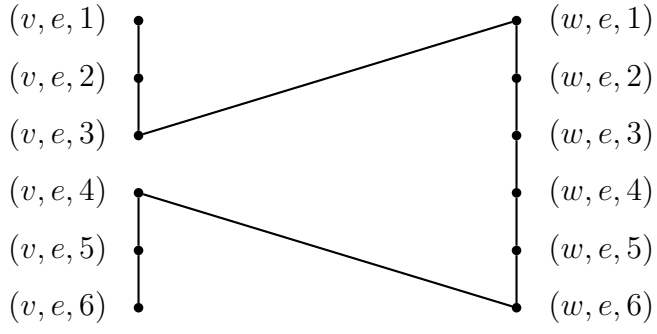
Se $i, j \in \{0, \ldots, n+1\}$ sono indici tali che (a) $i < j$, (b) $x_i, x_j \in A$ e (c) $x_h \notin A$ per ogni $i < h < j$, allora esiste un unico vertice $v \in V$ tale che

$$(x_{i+1} = (v, e_v^1, 1) \textbf{ AND } x_{j-1} = (v, e_v^{\mathsf{deg}(v)}, 6)) \textbf{ OR } (x_{i+1} = (v, e_v^{\mathsf{deg}(v)}, 6) \textbf{ AND } x_{j-1} = (v, e_v^1, 1)) .$$

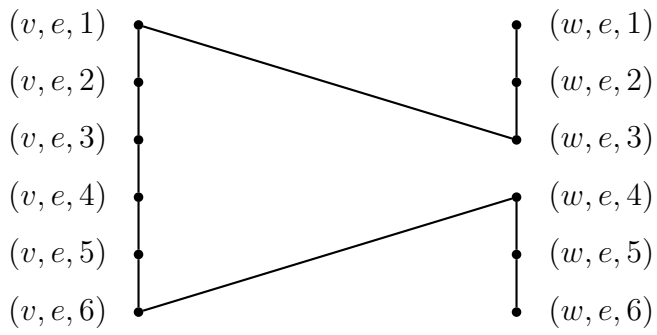
Pertanto W non può contenere più di k vertici perchè, altrimenti, esisterebbero indici $i, j \in \{0, \ldots, n+1\}$ soddisfacenti alle condizioni (a), (b) e (c) precedenti e tali che $x_{i+1} = (v, e_v^1, 1)$ e $x_{j-1} = (w, e_w^1, 1)$ per alcuni vertici $v, w \in W$, contraddicendo l'osservazione appena fatta.

Viceversa, sia W un vertex cover per G tale che $|W| = k$ e siano $v_1, v_2, \ldots, v_k$ i vertici di W. Per ogni arco $e = \{v, w\} \in E$, sia $\mathsf{Edges}(e)$ l'insieme di archi definito come segue:

(1) se $W \cap e = \{v\}$ allora $\mathsf{Edges}(e)$ consiste nei seguenti archi:



(2) se $W \cap e = \{w\}$ allora $\mathsf{Edges}(e)$ consiste nei seguenti archi:



(3) se $W \cap e = \{v, w\}$ allora $\mathsf{Edges}(e)$ consiste nei seguenti archi:

(Si osservi che si verifica esattamente una sola delle precedenti possibilità in quanto W è un vertex cover per G.) Si verifica che un cammino Hamiltoniano in $\widehat{G}$ può essere costruito usando gli archi del seguente insieme

$$\begin{array}{l}\left(\bigcup_{e\in E} \mathsf{Edges}(e)\right) \cup \\ \left(\bigcup_{1\le i\le k} E''_{v_i}\right) \cup \\ \{\{a_i, (v_i, e^1_{v_i}, 1)\} : 1 \le i \le k\} \cup \\ \{\{a_{i+1}, (v_i, e^{\mathsf{deg}(v_i)}_{v_i}, 6)\} : 1 \le i < k\} \cup \\ \{\{a_k, (v_k, e^{\mathsf{deg}(v_k)}_{v_k}, 6)\}\}\,.\end{array}$$

□

Teorema 34. $\mathsf{HamiltonianCircuit} \le_p \mathsf{HamiltonianPath}$.

Dimostrazione. Sia $G = (V, E)$ un grafo. Costruiamo in tempo polinomiale il grafo G' tale che G ha un circuito Hamiltoniano se e solo se G' ha un cammino Hamiltoniano.

Si osservi innanzitutto che se G contiene meno di tre vertici allora G non può avere alcun circuito Hamiltoniano. In questo caso è sufficiente prendere come G' un qualunque grafo che non ha cammini Hamiltoniani (ad esempio un grafo contenente un solo vertice). Supponiamo quindi che G contenga almeno tre vertici.

Sia u un arbitrario vertice di G e siano u^*, v e v^* tre nuovi vertici distinti non appartenenti a V. Poniamo $G' = (V', E')$ dove

$$V' = V \cup \{\mathsf{u}^*, \mathsf{v}, \mathsf{v}^*\}$$

ed

$$E' = E \cup \{\{\mathsf{u}^*, x\} : \{\mathsf{u}, x\} \in E\} \cup \{\{\mathsf{v}, \mathsf{u}\}, \{\mathsf{v}^*, \mathsf{u}^*\}\}\,.$$

Supponiamo che $(v_0, v_1, \ldots, v_n)$ sia un circuito Hamiltoniano in G. Sia i tale che $v_i = \mathsf{u}$. Allora la sequenza $(\mathsf{v}, \mathsf{u}, v_{i+1}, \ldots, v_n, v_0, v_1, \ldots, v_{i-1}, \mathsf{u}^*, \mathsf{v}^*)$ è un cammino Hamiltoniano in G. Viceversa, sia $(v_0, v_1, \ldots, v_n)$ un cammino Hamiltoniano in G'. Poiché

$$\{x \in V' : \{\mathsf{v}, x\} \in E'\} = \{\mathsf{u}\},$$

si ha che $\mathsf{v} \neq v_i$, per ogni $0 < i < n$. Infatti, in caso contrario si avrebbe che $v_{i-1} = \mathsf{u}$, $v_i = \mathsf{v}$ e $v_{i+1} = \mathsf{u}$, per qualche indice i tale che $0 < i < n$ e ciò contraddice il fatto che i vertici $v_0, v_1, \ldots, v_n$ sono tutti distinti tra loro. Analogamente, si ha che $\mathsf{v}^* \neq v_i$, per ogni $0 < i < n$. Pertanto o $v_0 = \mathsf{v}$ e $v_n = \mathsf{v}^*$ oppure $v_0 = \mathsf{v}^*$ e $v_n = \mathsf{v}$. Il primo caso, cioè $v_0 = \mathsf{v}$ e $v_n = \mathsf{v}^*$, implica che $v_1 = \mathsf{u}$, $v_{n-1} = \mathsf{u}^*$ e $\{v_{n-2}, \mathsf{u}\} \in E$, così la sequenza $(v_1, \ldots, v_{n-2})$ è un circuito Hamiltoniano in G. (Si osservi che $n \geq 5$ dato che G contiene almeno tre vertici). Il caso $v_0 = \mathsf{v}^*$ e $v_n = \mathsf{v}$ si tratta in maniera simmetrica. □

Teorema 35. HamiltonianPath $\leq_p$ HamiltonianCircuit.

Dimostrazione. Poiché HamiltonianCircuit è **NP**-completo e HamiltonianPath è in **NP**, si ha chiaramente che HamiltonianPath $\leq_p$ HamiltonianCircuit. Mostriamo tuttavia un riduzione polinomiale diretta da HamiltonianPath a HamiltonianCircuit.

Sia $G = (V, E)$ un grafo non direzionato. Costruiamo il grafo $G' = (V', E')$ tale che G ha un cammino Hamiltoniano se e solo se G' ha un circuito Hamiltoniano. La costruzione coinvolge l'introduzione di un nuovo vertice z non appartenente a V e degli archi $\{\{\mathsf{z}, v\} : v \in V\}$; quindi $V' = V \cup \{\mathsf{z}\}$ ed $E' = E \cup \{\{\mathsf{z}, v\} : v \in V\}$.

Se $(v_0, v_1, \ldots, v_n)$ è un cammino Hamiltoniano in G, si verifica immediatamente che la sequenza di vertici $(v_0, v_1, \ldots, v_n, \mathsf{z})$ è un circuito Hamiltoniano in G'.

Viceversa, sia $(w_0, w_1, \ldots, w_n)$ un circuito Hamiltoniano in G' e sia $i \in \{0, 1, \ldots, n\}$ tale che $w_i = \mathsf{z}$. Se $i = 0$ allora la sequenza di vertici $(w_1, \ldots, w_n)$ è un cammino Hamiltoniano in G. Similmente, se $i = n$ allora la sequenza di vertici $(w_0, w_1, \ldots, w_{n-1})$ è un cammino Hamiltoniano in G. Infine, se $0 < i < n$ allora la sequenza di vertici $(w_{i+1}, \ldots, w_n, w_0, \ldots, w_{i-1})$ è un cammino Hamiltoniano in G. □

Osservazione 7. I problemi sui grafi non direzionati VertexCover, CLIQUE, IndependentSet, HamiltonianCircuit e HamiltonianPath introdotti precedentemente possono essere riformulati anche nel caso di grafi direzionati. Ciascuno dei precedenti problemi risulta polinomialmente riducibile alla corrispondente "versione direzionata". Ad esempio, consideriamo il problema VertexCover. Sia $G = (V, E)$ un grafo non direzionato. Se $\widehat{G} = (V, \widehat{E})$ è il grafo direzionato dove $\widehat{E} = \{(v, w) : \{v, w\} \in E\}$, si verifica che un sottoinsieme W di V è un vertex cover per G se e solo se W è un vertex cover per $\widehat{G}$. ■

11.5 Problema LongestSimplePath

Definizione 42. Il PROBLEMA LongestSimplePath consiste nel determinare per ogni coppia (G, k), dove G è un grafo e $k \in \mathbb{N} \setminus \{0\}$, se G ha un cammino semplice di lunghezza maggiore o uguale a k. ■

Teorema 36. HamiltonianPath $\leq_p$ LongestSimplePath.

Dimostrazione. Sia $G = (V, E)$ un grafo. Poniamo $k = |V| - 1$. Allora esiste un cammino Hamiltoniano in G se e solo se esiste un cammino semplice in G di lunghezza maggiore o uguale a k. □

11.6 Problemi SetCover e HittingSet

Definizione 43. Sia $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ una famiglia di insiemi finiti.
Un RICOPRIMENTO DI $\mathcal{F}$ è una sottofamiglia $\mathcal{R} = \{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$ di $\mathcal{F}$ tale che

$$\bigcup_{1 \le j \le n} S_j = \bigcup_{1 \le h \le m} S_{i_h} .$$

Un HITTING SET di $\mathcal{F}$ è un sottoinsieme I di $\bigcup_{1 \le j \le n} S_j$ tale che

$$I \cap S_j \neq \emptyset ,$$

per ogni $j = 1, 2, \ldots, n$. ■

Definizione 44. Il PROBLEMA SetCover consiste nel determinare per ogni coppia $(\mathcal{F}, k)$, dove $\mathcal{F}$ è una famiglia (finita) di insiemi finiti e $k \in \mathbb{N} \setminus \{0\}$, se esiste un ricoprimento $\mathcal{R}$ di $\mathcal{F}$ contenente al più k insiemi. ■

Definizione 45. Il PROBLEMA HittingSet consiste nel determinare per ogni coppia $(\mathcal{F}, k)$, dove $\mathcal{F}$ è una famiglia (finita) di insiemi finiti e $k \in \mathbb{N} \setminus \{0\}$, se esiste un hitting set I di $\mathcal{F}$ di cardinalità al più k. ■

Teorema 37. VertexCover $\le_p$ SetCover.

Dimostrazione. Sia $G = (V, E)$ un grafo non direzionato dove $V = \{v_1, v_2, \ldots, v_n\}$. Per ogni $i = 1, 2, \ldots, n$, sia

$$S_i = \{e \in E : v_i \in e\} ,$$

e sia $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$. Si osservi che

$$\bigcup_{1 \le i \le n} S_i = E . \tag{2}$$

Per ogni sottoinsieme $\{i_1, i_2, \ldots, i_m\}$ di $\{1, 2, \ldots, n\}$, l'insieme $W = \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ è un vertex cover per G se e solo se $\mathcal{R} = \{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$ è un ricoprimento di $\mathcal{F}$.

Infatti, supponiamo che $W = \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ sia un vertex cover per G. Sia $e \in \bigcup_{1 \le j \le n} S_j$. Allora e è un arco del grafo G e quindi $W \cap e \neq \emptyset$, cioè esiste $h \in \{1, 2, \ldots, m\}$ tale che $v_{i_h} \in e$. Questo implica che $e \in S_{i_h}$ e pertanto $e \in \bigcup_{1 \le h \le m} S_{i_h}$. Data l'arbitrarietà di e concludiamo che $\mathcal{R}$ è un ricoprimento di $\mathcal{F}$.

Viceversa, supponiamo che $\mathcal{R} = \{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$ sia un ricoprimento di $\mathcal{F}$. Sia $e \in E$. Essendo $\mathcal{R}$ un ricoprimento di $\mathcal{F}$, per la (2) esiste $h \in \{1, 2, \ldots, m\}$ tale che $e \in S_{i_h}$. Da ciò segue che $v_{i_h} \in e \cap W$, da cui $W \cap e \neq \emptyset$. Dunque W interseca ogni arco del grafo G e pertanto è un vertex cover per G. □

Teorema 38. VertexCover $\le_p$ HittingSet.

Dimostrazione. Sia $G = (V, E)$ un grafo non direzionato dove $E = \{e_1, e_2, \ldots, e_n\}$.

Si osservi che se W è un vertex cover per G allora l'insieme $I = W \cap \left(\bigcup_{1 \le i \le n} e_i\right)$ è un hitting set per la famiglia $\mathcal{F} = \{e_1, e_2, \ldots, e_n\}$. Inoltre $|I| \le |W|$.

Viceversa, se I è un hitting set per la famiglia $\mathcal{F} = \{e_1, e_2, \ldots, e_n\}$, allora I è un vertex cover per G. □

11.7 Problema SubgraphIsomorphism

Definizione 46. Un SOTTOGRAFO di un grafo $G = (V, E)$ è un grafo $G' = (V', E')$ tale che $V' \subseteq V$ ed $E' \subseteq E$.
Un GRAFO $G_1 = (V_1, E_1)$ È ISOMORFO ad un grafo $G_2 = (V_2, E_2)$, e si scrive $G_1 \approx G_2$, se esiste una funzione biiettiva $f : V_1 \to V_2$, chiamata ISOMORFISMO DA G_1 IN G_2, tale che $\{x, y\} \in E_1 \Leftrightarrow \{f(x), f(y)\} \in E_2$ per ogni $x, y \in V_1$.

Definizione 47. Il PROBLEMA SubgraphIsomorphism consiste nel determinare per ogni coppia di grafi (G_1, G_2) se e esiste un sottografo H di G_1 tale che $H \approx G_2$. ■

Teorema 39. CLIQUE $\leq_p$ SubgraphIsomorphism.

Dimostrazione. Sia $G = (V, E)$ un grafo dove $V = \{v_1, \ldots, v_n\}$ e sia $0 < k \leq |V|$. Indichiamo con $\mathcal{G}_k$ il GRAFO COMPLETO DI DIMENSIONE k CON VERTICI $v_1, \ldots, v_k$, cioè $\mathcal{G}_k = (\{v_1, \ldots, v_k\}, \{\{v_i, v_j\} : i, j \in \{1, \ldots, k\}, i \neq j\})$. Allora esiste una clique W per G tale che $|W| \geq k$ se e solo se esiste un sottografo H di G tale che $H \approx \mathcal{G}_k$. Infatti, sia $W = \{v_{i_1}, \ldots, v_{i_h}\}$ una clique per G dove $h \geq k$. Poniamo

$$H = (\{v_{i_1}, \ldots, v_{i_k}\}, \{\{v_{i_a}, v_{i_b}\} : a, b \in \{1, \ldots, k\}, a \neq b\}) .$$

Ovviamente $H \approx \mathcal{G}_k$. Inoltre H è un sottografo di G dato che W è una clique per G.

Viceversa, sia $H = (V', E')$ un sottografo di G tale che $H \approx \mathcal{G}_k$ e sia f un isomorfismo da H in $\mathcal{G}_k$. Ovviamente $|V'| = k$ (perchè f è un biiezione tra V' e l'insieme dei k vertici di $\mathcal{G}_k$). Verifichiamo che V' è una clique per G. Siano $x, y \in V'$ tali che $x \neq y$. Allora $f(x)$ e $f(y)$ sono vertici distinti di $\mathcal{G}_k$ e quindi $\{f(x), f(y)\}$ è un'arco di $\mathcal{G}_k$. Questo implica che $\{x, y\}$ è un arco di H, cioè $\{x, y\} \in E'$. Poichè $E' \subseteq E$ (in quanto H è un sottografo di G) si ha che $\{x, y\} \in E$. Pertanto V' è una clique per G. □

11.8 Problema TravelingSalesman

Definizione 48. Un TSG è una coppia ordinata $\mathcal{T} = (G, d)$ dove

(1) $G = (V, E)$ è un GRAFO NON DIREZIONATO COMPLETO, cioè $\{v, w\} \in E$ per ogni $v, w \in V$ con $v \neq w$;

(2) $d : V \times V \to \mathbb{N} \setminus \{0\}$ è la "funzione costo" tale che $d(v, w) = d(w, v)$, per ogni $v, w \in V$. ■

Definizione 49. Sia $\mathcal{T} = (G, d)$ un TSG.
Un TOUR in $\mathcal{T}$ è un cammino Hamiltoniano π in G. Il COSTO DEL TOUR $\pi = (v_0, v_1, \ldots, v_n)$ è la quantitá $\widetilde{d}(\pi)$ definita come segue:

$$\widetilde{d}(\pi) = d(v_n, v_0) + \sum_{i=0}^{n-1} d(v_i, v_{i+1}) .$$

■

Definizione 50. Il PROBLEMA TravelingSalesman consiste nel determinare per ogni coppia $(\mathcal{T}, k)$, dove $\mathcal{T}$ è un TSG e $k \in \mathbb{N} \setminus \{0\}$, se esiste un tour π in $\mathcal{T}$ tale che $\widetilde{d}(\pi) \leq k$. ■

Teorema 40. HamiltonianCircuit $\leq_p$ TravelingSalesman.

Dimostrazione. Sia $G = (V, E)$ un grafo non direzionato dove $|V| = k$. Costruiamo il TSG $\mathcal{T}_G$ tale che G ha un circuito Hamiltoniano se e solo se $\mathcal{T}_G$ ha un tour di costo al più k.

Sia $G' = (V, E')$ il grafo non direzionato dove $E' = \{\{v, w\} : v, w \in V \text{ AND } v \neq w\}$ e sia $d : V \times V \to \mathbb{N} \setminus \{0\}$ definita come segue:

$$d(v, w) = \begin{cases} 1, & \text{se } \{v, w\} \in E \\ 2, & \text{altrimenti}\,, \end{cases}$$

per ogni $v, w \in V$. Poniamo $\mathcal{T}_G = (G', d)$.

Sia $\pi = (v_0, v_1, \ldots, v_{k-1})$ un circuito Hamiltoniano in G. Allora π è un tour in $\mathcal{T}_G$. Inoltre, poichè $\{v_0, v_1\}, \ldots, \{v_{k-2}, v_{k-1}\}, \{v_{k-1}, v_0\} \in E$, si ha che $d(v_0, v_1) = \cdots = d(v_{k-2}, v_{k-1}) = d(v_{k-1}, v_0) = 1$ e quindi il costo del tour π è uguale a k.

Viceversa, sia $\pi = (v_0, v_1, \ldots, v_{k-1})$ un tour in $\mathcal{T}_G$ tale che $\widetilde{d}(\pi) \leq k$. Poichè la funzione d assume solo valori dell'insieme $\{1, 2\}$, deve aversi necessariamente che $d(v_0, v_1) = \cdots = d(v_{k-2}, v_{k-1}) = d(v_{k-1}, v_0) = 1$ e quindi $\{v_0, v_1\}, \ldots, \{v_{k-2}, v_{k-1}\}, \{v_{k-1}, v_0\} \in E$. Pertanto π è un circuito Hamiltoniano in G. □

Osservazione 8. Se nella dimostrazione del teorema precedente si sostituisce la definizione della funzione d con la seguente:

$$d(v, w) = \begin{cases} 2, & \text{se } \{v, w\} \in E \\ 1, & \text{altrimenti}\,, \end{cases}$$

per ogni $v, w \in V$, allora ogni circuito Hamiltoniano in G è un tour in $\mathcal{T}_G$ di costo maggiore o uguale a $2k$ e, viceversa, se π è un tour in $\mathcal{T}_G$ di costo maggiore o uguale a $2k$ allora π è un circuito Hamiltoniano in G. ■

11.9 Problema 3−DimensionalMatching

Definizione 51. Il PROBLEMA DEL 3−DimensionalMatching è definito come segue.

Dati tre insiemi finiti e disgiunti X, Y e Z tali che $|X| = |Y| = |Z| = q$ e assegnato un sottoinsieme M di $X \times Y \times Z$, esiste un sottoinsieme N di M tale che $|N| = q$ e per ogni $(x', y', z'), (x'', y'', z'') \in N$, con $(x', y', z') \neq (x'', y'', z'')$, si abbia che $x' \neq x''$, $y' \neq y''$ e $z' \neq z''$? In tal caso l'insieme N costituisce un MATCHING di M. ■

Teorema 41. 3−SAT $\leq_p$ 3−DimensionalMatching.

Dimostrazione. Sia $S = \{C_1, C_2, \ldots, C_m\}$ un insieme di 3-clausole e sia $U = \{u_1, u_2, \ldots, u_n\}$ il supporto di S. Costruiamo gli insiemi finiti e disgiunti X, Y e Z, con $|X| = |Y| = |Z|$, e il sottoinsieme M di $X \times Y \times Z$ tali che S è soddisfacibile se e solo se esiste un matching N di M. La costruzione coinvolge l'introduzione degli insiemi disgiunti $A_1, \ldots, A_n$, $B_1, \ldots, B_n$, $V_1, \ldots, V_n$, $F_1, \ldots, F_n$, S_1, S_2, G_1 e G_2 (la quadrupla (A_i, B_i, V_i, F_i) corrispondente all'atomo u_i), dove:

(1) $A_i = \{a_i[1], \ldots, a_i[m]\}$,

(2) $B_i = \{b_i[1], \ldots, b_i[m]\}$,

(3) $V_i = \{v_i[1], \ldots, v_i[m]\}$,

(4) $F_i = \{f_i[1], \ldots, f_i[m]\}$,

(5) $S_1 = \{s_1[1], \ldots, s_1[m]\}$,

(6) $S_2 = \{s_2[1], \ldots, s_2[m]\}$,

(7) $G_1 = \{g_1[1], \ldots, g_1[m(n-1)]\}$,

(8) $G_2 = \{g_2[1], \ldots, g_2[m(n-1)]\}$,

per $i = 1, 2, \ldots, n$.
Poniamo:

$$X = \bigcup_{1 \le i \le n} (V_i \cup F_i)\,, \quad Y = A \cup S_1 \cup G_1 \quad \text{e} \quad Z = B \cup S_2 \cup G_2\,,$$

dove $A = \bigcup_{1 \le i \le n} A_i$ e $B = \bigcup_{1 \le i \le n} B_i$. Si osservi che $|X| = |Y| = |Z| = 2mn$.
Per $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, m$, siano:

$$T_i^{(\mathsf{v})} = \{(f_i[j], a_i[j], b_i[j]) : 1 \le j \le m\}\,,$$

$$T_i^{(\mathsf{f})} = \{(v_i[j], a_i[j+1], b_i[j]) : 1 \le j < m\} \cup \{v_i[m], a_i[1], b_i[m]\}\,,$$

$$T_i = T_i^{(\mathsf{v})} \cup T_i^{(\mathsf{f})}$$

e

$$K_j = \{(v_i[j], s_1[j], s_2[j]) : u_i \in C_j\} \cup \{(f_i[j], s_1[j], s_2[j]) : \neg u_i \in C_j\}\,,$$

e sia

$$G = \{(v_i[j], g_1[k], g_2[k]), (f_i[j], g_1[k], g_2[k]) : 1 \le i \le n, 1 \le j \le m, 1 \le k \le m(n-1)\}\,.$$

Poniamo

$$M = \left(\bigcup_{1 \le i \le n} T_i\right) \cup \left(\bigcup_{1 \le j \le m} K_j\right) \cup G\,.$$

Ovviamente $M \subseteq X \times Y \times Z$.

Dimostriamo che l'insieme S è soddisfacibile se e solo se esiste un matching N di M. Premettiamo le seguenti definizioni.
Per ogni tripla $t = (x, y, x) \in X \times Y \times Z$ e ogni sottoinsieme I di $X \times Y \times Z$, poniamo

$$\alpha(t) = x \quad \text{e} \quad \alpha(I) = \{\alpha(t) : t \in I\}\,.$$

Un sottoinsieme N di $X \times Y \times Z$ è OK se per ogni $(x', y'.x'), (x'', y'', z'') \in N$, con $(x', y'.x') \neq (x'', y'', z'')$, si ha che $x' \neq x''$, $y' \neq y''$ e $z' \neq z''$. Così, un sottoinsieme N di M costituisce un matching di M se N è OK e inoltre $|N| = 2mn$.

Supponiamo che S sia soddisfacibile, e sia v un assegnamento che soddisfa S.
Per $k = 1, 2, \ldots, 2n$, sia

$$\ell_k = \begin{cases} u_k, & \text{se } k \le n \\ \neg u_{k-n}, & \text{altrimenti}\,. \end{cases}$$

Poiché v soddisfa ogni clausola di S, si ha che per ogni $j = 1, 2, \ldots, m$ esiste un letterale $\ell \in C_j$ tale che $(\ell)^v = \mathsf{True}$; pertanto l'insieme $\{k \in \{1, 2, \ldots, 2n\} : \ell_k \in C_j \wedge (\ell_k)^v = \mathsf{True}\}$ è non vuoto per ogni $j = 1, 2, \ldots, m$. Poniamo quindi

$$m(j) = \min\{k \in \{1, 2, \ldots, 2n\} : \ell_k \in C_j \wedge (\ell_k)^v = \mathsf{True}\}$$

e

$$w_j = \begin{cases} v_{m(j)}[j], & \text{se } m(j) \leq n \\ f_{m(j)-n}[j], & \text{altrimenti}\,, \end{cases}$$

per $j = 1, 2, \ldots, m$. Adesso, siano

$$T_1 = \bigcup_{v(u_i)=\mathsf{True}} T_i^{(\mathsf{v})}\,, \quad T_2 = \bigcup_{v(u_i)=\mathsf{False}} T_i^{(\mathsf{f})}\,, \quad W = \bigcup_{1\leq j\leq m} \{(w_j, s_1[j], s_2[j])\}$$

e

$$G' = \{(x_i, g_1[i], g_2[i]) : 1 \leq i \leq \min(h, m(n-1))\}\,,$$

dove $\{x_1, x_2, \ldots, x_h\} = X \setminus (\alpha(T_1) \cup \alpha(T_2) \cup \alpha(W))$ e sia

$$N = T_1 \cup T_2 \cup W \cup G'\,.$$

Allora N costituisce un matching di M. Infatti, si osservi innanzitutto che, chiaramente, $N \subseteq M$. Inoltre valgono le seguenti proprietà:

(A) $\alpha(T_1) \cap \alpha(T_2) = \emptyset$;

(B) $(\alpha(T_1) \cup \alpha(T_2)) \cap \alpha(W) = \emptyset$;

(C) $(\alpha(T_1) \cup \alpha(T_2) \cup \alpha(W)) \cap \alpha(G') = \emptyset$.

La (C) è conseguenza immediata della definizione di G'. La (A) segue dal fatto che $\alpha(T_1) \subseteq \bigcup_{1\leq i\leq n} F_i$, $\alpha(T_2) \subseteq \bigcup_{1\leq i\leq n} V_i$ e gli insiemi $\bigcup_{1\leq i\leq n} F_i$ e $\bigcup_{1\leq i\leq n} V_i$ sono disgiunti. Per quanto riguarda la (B), supponiamo, per assurdo, che si abbia $(\alpha(T_1) \cup \alpha(T_2)) \cap \alpha(W) \neq \emptyset$. Allora esistono triple $t' \in (T_1 \cup T_2)$ e $t'' \in W$ tali che

$$\alpha(t') = \alpha(t'')\,. \tag{B1}$$

Dalla definizione di W segue che esistono indici $i \in \{1, 2, \ldots, n\}$ e $j \in \{1, 2, \ldots, m\}$ tali che

$$(\alpha(t'') = v_i[j] \wedge v(u_i) = \mathsf{True}) \vee (\alpha(t'') = f_i[j] \wedge v(u_i) = \mathsf{False})\,. \tag{B2}$$

D'altra parte, dalle definizioni di T_1 e T_2 segue che esistono indici $r \in \{1, 2, \ldots, n\}$ e $s \in \{1, 2, \ldots, m\}$ tali che

$$(\alpha(t') = f_r[s] \wedge v(u_r) = \mathsf{True}) \vee (\alpha(t') = v_r[s] \wedge v(u_r) = \mathsf{False})\,. \tag{B3}$$

Le condizioni (B1), (B2) e (B3) comportano una contraddizione e pertanto la (B) è vera.

Dalle definizioni degli insiemi T_1, T_2, W e G', usando le proprietà (A), (B) e (C), segue facilmente che l'insieme N è OK. Inoltre abbiamo già osservato che $N \subseteq M$. Pertanto rimane da dimostrare che $|N| = 2mn$, che facciamo come segue.

Le proprietà (A), (B) e (C) implicano che gli insiemi $\alpha(T_1)$, $\alpha(T_2)$, $\alpha(W)$ e $\alpha(G')$ sono a due a due disgiunti e questo implica, a sua volta, che anche gli insiemi T_1, T_2, W e G' sono a due a due disgiunti. Si osservi inoltre che $|\alpha(T_1)| = |T_1|$, $|\alpha(T_2)| = |T_2|$, $|\alpha(T_1)| + |\alpha(T_2)| = nm$, $|\alpha(W)| = |W| = m$ e $|\alpha(G')| = |G'| = \min\{|X \setminus (\alpha(T_1) \cup \alpha(T_2) \cup \alpha(W))|, m(n-1)\}$. Ma allora si ha che

$$|X \setminus (\alpha(T_1) \cup \alpha(T_2) \cup \alpha(W))| = 2mn - (mn + m) = m(n-1)$$

e quindi $|G'| = m(n-1)$. Poichè gli insiemi T_1, T_2, W e G' sono a due a due disgiunti si ha che

$$|N| = |T_1| + |T_2| + |W| + |G'|$$

e pertanto

$$|N| = nm + m + m(n-1) = 2mn$$

e ciò conclude la dimostrazione che N costituisce un matching di M.

Supponiamo adesso che esista un matching N di M. Allora, l'assegnamento v tale che $v(a) = \mathsf{False}$, per ogni atomo $a \notin U$ e

$$v(u_i) = \begin{cases} \mathsf{True}, & \text{se } N \cap T_i = T_i^{(\mathsf{v})} \\ \mathsf{False}, & \text{altrimenti}\,, \end{cases}$$

per $i = 1, 2, \ldots, n$, soddisfa l'insieme S. Infatti, poichè N è OK, dalle definizioni degli insiemi $T_1, \ldots, T_n$, $K_1, \ldots, K_m$ e G, segue che:

(M1) $|N \cap T_i| \leq m$;

(M2) se $|N \cap T_i| = m$, allora o $N \cap T_i = T_i^{(\mathsf{v})}$ oppure $N \cap T_i = T_i^{(\mathsf{f})}$;

(M3) $|N \cap K_j| \leq 1$;

(M4) $|N \cap G| \leq m(n-1)$;

(M5) se $v_i[j] \in \alpha(N \cap K_j)$ allora $u_i \in C_j$ e $N \cap T_i^{(\mathsf{f})} = \emptyset$;

(M6) se $f_i[j] \in \alpha(N \cap K_j)$ allora $\neg u_i \in C_j$ e $N \cap T_i^{(\mathsf{v})} = \emptyset$;

per ogni $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, m$. Dato che $|N| = 2mn$, dalle (M1), (M3) e (M4) segue che

$$|N \cap T_i| = m\,, \quad |N \cap K_j| = 1\,, \quad |N \cap G| = m(n-1)\,,$$

per ogni $i = 1, 2, \ldots, n$ e $j = 1, 2, \ldots, m$. La prima delle precedenti uguaglianze e (M2) implicano che

$$N \cap T_i = T_i^{(\mathsf{v})} \quad \text{oppure} \quad N \cap T_i = T_i^{(\mathsf{f})}\,, \tag{M7}$$

per ogni $i = 1, 2, \ldots, n$. Adesso, sia $j \in \{1, 2, \ldots, m\}$. Mostriamo che v soddisfa la clausola C_j. Poichè $|N \cap K_j| = 1$, esiste un (unico) indice $i \in \{1, 2, \ldots, n\}$ tale che $v_i[j] \in \alpha(N \cap K_j)$ oppure $f_i[j] \in \alpha(N \cap K_j)$. Se $v_i[j] \in \alpha(N \cap K_j)$ dalle (M5) e (M7) segue che $u_i \in C_j$ e $N \cap T_i = T_i^{(\mathsf{v})}$; pertanto $u_i \in C_j$ e $v(u_i) = \mathsf{True}$ da cui segue che v soddisfa C_j. Se $f_i[j] \in \alpha(N \cap K_j)$ dalla (M6) segue che $\neg u_i \in C_j$ e $N \cap T_i \neq T_i^{(\mathsf{v})}$ e quindi $\neg u_i \in C_j$ e $v(u_i) = \mathsf{False}$; ciò implica che v soddisfa C_j. Data l'arbitrarietà di j concludiamo che v soddisfa ogni clausola di S e pertamto S è soddisfacibile. □

11.10 Problema Partition

Definizione 52. Il PROBLEMA Partition consiste nel determinare per ogni coppia (A, s), dove A è un insieme finito e $s : A \to \mathbb{N} \setminus \{0\}$ è la "funzione dimensione", se esiste un sottoinsieme B di A tale che

$$\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x)\,.$$

■

Teorema 42. 3−DimensionalMatching $\leq_p$ Partition.

Dimostrazione. Siano $X = \{x_1, \ldots, x_q\}$, $Y = \{y_1, \ldots, y_q\}$ e $Z = \{z_1, \ldots, z_q\}$ insiemi finiti e disgiunti e sia $M = \{m_1, \ldots, m_k\} \subseteq X \times Y \times Z$. Introduciamo l'insieme $A = \{a_1, \ldots, a_k, b_1, b_2\}$ contenente $k + 2$ elementi. Per $i = 1, \ldots, k$, sia

$$s(a_i) = 2^{\rho(3q-r)} + 2^{\rho(2q-s)} + 2^{\rho(q-t)}\,,$$

dove $\rho = \lceil \log_2(k+1) \rceil$ e $(x_r, y_s, z_t) = m_i$ e siano

$$s(b_1) = 2\left(\sum_{i=1}^{k} s(a_i)\right) - \omega \quad \text{e} \quad s(b_2) = \left(\sum_{i=1}^{k} s(a_i)\right) + \omega\,,$$

dove

$$\omega = \sum_{j=0}^{3q-1} 2^{j\rho}\,.$$

Si osservi che un sottoinsieme $N = \{m_{i_1}, m_{i_2}, \ldots, m_{i_n}\}$ di M costituisce un matching di M se e solo se

$$\sum_{j=1}^{n} s(a_{i_j}) = \omega\,.$$

Sia B un sottoinsieme di A tale che

$$\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x)\,.$$

Allora,

$$\begin{aligned} 2\sum_{x \in B} s(x) &= \sum_{x \in B} s(x) + \sum_{x \in A \setminus B} s(x) \\ &= \sum_{x \in A} s(x) \\ &= \sum_{i=1}^{k} s(a_i) + s(b_1) + s(b_2) \\ &= 4\sum_{i=1}^{k} s(a_i)\,, \end{aligned}$$

e quindi

$$\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x) = 2\sum_{i=1}^{k} s(a_i)\,.$$

Pertanto esiste $C \in \{B, A \setminus B\}$ tale che $b_1 \in C$ e $b_2 \notin C$. Sia $C \setminus \{b_1\} = \{a_{i_1}, a_{i_2}, \ldots, a_{i_n}\}$. Allora

$$\sum_{j=1}^{n} s(a_{i_j}) = \omega$$

e quindi per l'osservazione precedente si ha che $N = \{m_{i_1}, m_{i_2}, \ldots, m_{i_n}\}$ costituisce un matching di M. Viceversa, se $N = \{m_{i_1}, m_{i_2}, \ldots, m_{i_q}\}$ costituisce un matching di M si ha che

$$\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x)\,,$$

dove $B = \{b_1\} \cup \{a_{i_1}, a_{i_2}, \ldots, a_{i_q}\}$. □

11.11 Problemi SubsetSum e Knapsack

Definizione 53. Il PROBLEMA SubsetSum consiste nel determinare per ogni tripla (A, s, k), dove A è un insieme finito, $s : A \to \mathbb{N} \setminus \{0\}$ è la "funzione dimensione" e $k \in \mathbb{N} \setminus \{0\}$ se esiste un sottoinsieme B di A tale che

$$\sum_{x \in B} s(x) = k \,.$$

■

Teorema 43. Partition $\leq_p$ SubsetSum.

Dimostrazione. Siano A un insieme finito e $s : A \to \mathbb{N} \setminus \{0\}$. Poniamo

$$k = \begin{cases} \dfrac{s(A)}{2}, & \text{se } s(A) \text{ è pari} \\ s(A) + 1, & \text{altrimenti}\,, \end{cases}$$

dove

$$s(A) = \sum_{x \in A} s(x) \,.$$

Si verifica che se B è un sottoinsieme di A allora $\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x)$ se e solo se $\sum_{x \in B} s(x) = k$. □

Definizione 54. Il PROBLEMA Knapsack consiste nel determinare per ogni data quintupla (A, s, v, k, h), dove A è un insieme finito, $s, v : A \to \mathbb{N} \setminus \{0\}$ e $k, h \in \mathbb{N} \setminus \{0\}$ se esiste un sottoinsieme B di A tale che

$$\sum_{x \in B} s(x) \leq k \quad \text{e} \quad \sum_{x \in B} v(x) \geq h \,.$$

■

Teorema 44. Partition $\leq_p$ Knapsack.

Dimostrazione. Siano A un insieme finito e $s : A \to \mathbb{N} \setminus \{0\}$. Poniamo

$$k = h = \begin{cases} \dfrac{s(A)}{2}, & \text{se } s(A) \text{ è pari} \\ s(A) + 1, & \text{altrimenti} \end{cases} \quad \text{e} \quad v = s \,,$$

dove

$$s(A) = \sum_{x \in A} s(x) \,.$$

Si verifica che se B è un sottoinsieme di A allora $\sum_{x \in B} s(x) = \sum_{x \in A \setminus B} s(x)$ se e solo se $\sum_{x \in B} s(x) \leq k$ e $\sum_{x \in B} v(x) \geq h$. □

12 Complessità Spaziale delle Macchine di Turing

Definizione 55. Per ogni configurazione $\gamma = XqtY$, poniamo

$$\mathsf{L}(\gamma) = |X| \qquad \text{e} \qquad \mathsf{R}(\gamma) = |Y| \,.$$

■

Definizione 56. Sia $\mathfrak{M}$ una TM e sia $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_n)$ una computazione di $\mathfrak{M}$.

Lo SPAZIO USATO DA ($\mathfrak{M}$ IN) Γ è il numero $\mathsf{space}(\Gamma)$ delle caselle del nastro che esamina $\mathfrak{M}$ durante l'effettuazione di Γ, dove ogni casella viene contata una sola volta. In maniera più precisa, $\mathsf{space}(\Gamma)$ viene definito come segue. Per $i = 0, \ldots, n-1$ sia $\mathcal{I}_i = q_i t_i o_i p_i$ l'istruzione di $\mathfrak{M}$ tale che $\gamma_i \vdash_{\{\mathcal{I}_i\}} \gamma_{i+1}$ (si osservi che $\mathcal{I}_i$ è unica) e sia

$$s_{i+1} = s_i + \begin{cases} 1, & \text{se } o_i = \mathbin{\vec{\urcorner}} \\ -1, & \text{se } o_i = \mathbin{\urcorner} \\ 0, & \text{altrimenti}\,, \end{cases}$$

dove $s_0 = 0$. Poniamo

$$\mathsf{space}(\Gamma) = \max(\{s_0, s_1, \ldots, s_n\}) - \min(\{s_0, s_1, \ldots, s_n\}) + 1\,.$$

■

Teorema 45. *Riferendoci alle notazioni della definizione (56), si ha che:*

(1) $\max(S_i) - \min(S_i) \leq i$;

(2) $\mathsf{space}(\Gamma) \leq |\Gamma| + 1$;

(3) $\mathsf{R}(\gamma_i) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i$;

(4) $\mathsf{L}(\gamma_i) = s_i - \min(S_i)$;

(5) $|\gamma_i| = \max(S_i) - \min(S_i) + 2 + \max\{0, \mathsf{R}(\gamma_0) - \max(S_i)\}$;

(6) $|\gamma_i| \leq \mathsf{space}(\Gamma) + 1 + \mathsf{R}(\gamma_0)$;

dove $S_i = \{s_0, s_1, \ldots, s_i\}$, *per ogni* $i = 0, 1, \ldots, n$.

Dimostrazione. (1): Per induzione su i.
Per $i = 0$ si ha che $\max(S_0) - \min(S_0) = s_0 - s_0 = 0 \leq 0$. Supponiamo, per ipotesi induttiva, che $\max(S_j) - \min(S_j) \leq j$, dove $0 \leq j < n$. Se $o_j \notin \{\mathbin{\vec{\urcorner}}, \mathbin{\urcorner}\}$ allora $S_{j+1} = S_j$ e quindi $\max(S_{j+1}) - \min(S_{j+1}) = \max(S_j) - \min(S_j) \leq j < j+1$. Se $o_j = \mathbin{\vec{\urcorner}}$ allora $S_{j+1} = S_j \cup \{s_j + 1\}$ che implica che $\min(S_{j+1}) = \min(S_j)$ e $\max(S_{j+1}) \leq \max(S_j) + 1$. Da queste ultime due relazioni segue che $\max(S_{j+1}) - \min(S_{j+1}) \leq \max(S_j) + 1 - \min(S_j)$ e quindi, per l'ipotesi induttiva, si ha che $\max(S_{j+1}) - \min(S_{j+1}) \leq j + 1$. Infine, se $o_j = \mathbin{\urcorner}$ si ha che $S_{j+1} = S_j \cup \{s_j - 1\}$ e usando argomentazioni simili a quelle precedenti si conclude ancora che $\max(S_{j+1}) - \min(S_{j+1}) \leq j + 1$.
(2): Segue da (1).
(3): Per induzione su i.
Per $i = 0$ si ha che $s_0 = 0$ e $\max(S_0 \cup \{\mathsf{R}(\gamma_0)\}) = \mathsf{R}(\gamma_0)$ e quindi $\mathsf{R}(\gamma_0) = \max(S_0 \cup \{\mathsf{R}(\gamma_0)\}) - s_0$. Supponiamo, per ipotesi induttiva, che $\mathsf{R}(\gamma_i) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i$, dove $0 \leq i < n$. Dobbiamo considerare i seguenti casi.
(Caso 1) $o_i \notin \{\mathbin{\urcorner}, \mathbin{\vec{\urcorner}}\}$.
Poiché $o_i \notin \{\mathbin{\urcorner}, \mathbin{\vec{\urcorner}}\}$ si ha che (a) $\mathsf{R}(\gamma_{i+1}) = \mathsf{R}(\gamma_i)$ e (b) $s_{i+1} = s_i$. Dalla (b) segue che $S_{i+1} = S_i$ e quindi $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\})$. Quest'ultima uguaglianza e la (b) implicano che $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1} = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i$ e quindi, per l'ipotesi induttiva, si ha che $\mathsf{R}(\gamma_i) = \max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1}$. Da ciò e dalla (a) segue che $\mathsf{R}(\gamma_{i+1}) = \max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1}$.

(Caso 2) $o_i = \leftarrow$.
In questo caso si ha che (a) $\mathsf{R}(\gamma_{i+1}) = \mathsf{R}(\gamma_i) + 1$ e (b) $s_{i+1} = s_i - 1$. La (b) implica che $S_{i+1} = S_i \cup \{s_i - 1\}$ e quindi $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\})$, in quanto $s_i \in S_i$. Da questa uguaglianza e dalla (b) segue che $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1} = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i + 1$ e quindi, per l'ipotesi induttiva, si ha che $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1} = \mathsf{R}(\gamma_i) + 1$. Da ciò e dalla (a) segue che $\mathsf{R}(\gamma_{i+1}) = \max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1}$.
(Caso 3) $o_i = \nearrow$.
In questo caso si ha che $s_{i+1} = s_i + 1$ e (quindi) $S_{i+1} = S_i \cup \{s_i + 1\}$. Dobbiamo distinguere i seguenti due sottocasi.
(3a) $\mathsf{R}(\gamma_i) = 0$.
Poiché $\mathsf{R}(\gamma_i) = 0$, per l'ipotesi induttiva, si ha che $\max(S_i \cup \{\mathsf{R}(\gamma_0)\}) = s_i$ e quindi $\max(S_i) = s_i$ (in quanto $s_i \in S_i$) ed $s_i \geq \mathsf{R}(\gamma_0)$. Pertanto

$$\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) = \max(S_i \cup \{s_i + 1, \mathsf{R}(\gamma_0)\}) = s_i + 1 = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) + 1$$

e così abbiamo che $\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1} = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i = \mathsf{R}(\gamma_i) = 0$. Si osservi quindi che $\mathsf{R}(\gamma_{i+1}) = 0$ perchè $o_i = \nearrow$ e $\mathsf{R}(\gamma_i) = 0$. Dunque

$$\mathsf{R}(\gamma_{i+1}) = \max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1}\,.$$

(3b) $\mathsf{R}(\gamma_i) > 0$.
Poichè $\mathsf{R}(\gamma_i) > 0$, per l'ipotesi induttiva, si ha che $\max(S_i \cup \{\mathsf{R}(\gamma_0)\}) > s_i$ e quindi

$$\max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) = \max(S_i \cup \{s_i + 1, \mathsf{R}(\gamma_0)\}) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\})\,.$$

D'altra parte, $\mathsf{R}(\gamma_i) > 0$ ed $o_i = \nearrow$ implicano che $\mathsf{R}(\gamma_{i+1}) = \mathsf{R}(\gamma_i) - 1$ e pertanto

$$\mathsf{R}(\gamma_{i+1}) = \max(S_i \cup \{\mathsf{R}(\gamma_0)\}) - s_i - 1 = \max(S_{i+1} \cup \{\mathsf{R}(\gamma_0)\}) - s_{i+1}\,.$$

(4): La dimostrazione è analoga a quella della (3).
(5): Segue da (3) e (4) osservando che $\max(S_i \cup \{\mathsf{R}(\gamma_0)\}) = \max(S_i) + \max\{0, \mathsf{R}(\gamma_0) - \max(S_i)\}$ e che $|\gamma_i| = \mathsf{L}(\gamma_i) + \mathsf{R}(\gamma_i) + 2$.
(6): Si osservi che $|\gamma_0| \leq |\gamma_1| \leq \cdots \leq |\gamma_n|$ (perché $\gamma_0 \vdash_{\mathfrak{M}} \gamma_1 \vdash_{\mathfrak{M}} \cdots \vdash_{\mathfrak{M}} \gamma_n$) e quindi, per la (5), si ha che

$$\begin{array}{rcl} |\gamma_i| & \leq & \max(S_n) - \min(S_n) + 2 + \max\{0, \mathsf{R}(\gamma_0) - \max(S_n)\} \\ & = & \mathsf{space}(\Gamma) + 1 + \max\{0, \mathsf{R}(\gamma_0) - \max(S_n)\}\,, \end{array}$$

per ogni $i = 0, 1, \ldots, n$. Poiché $\max(S_n) \geq 0$ si ha che $\max\{0, \mathsf{R}(\gamma_0) - \max(S_n)\} \leq \mathsf{R}(\gamma_0)$ e quindi, per la precedente relazione,

$$|\gamma_i| \leq \mathsf{space}(\Gamma) + 1 + \mathsf{R}(\gamma_0)\,,$$

per ogni $i = 0, 1, \ldots, n$. □

Definizione 57. Siano $\mathfrak{M}$ una TM, $\mathscr{L}$ un linguaggio e $s : \mathbb{N} \to \mathbb{N}$ una funzione. Diciamo che $\mathfrak{M}$ ACCETTA (O RICONOSCE) $\mathscr{L}$ USANDO SPAZIO (LIMITATO DA) s se

(1) $\mathscr{L}$ è accettato da $\mathfrak{M}$ e

(2) per ogni stringa $X \in \mathscr{L}$ esiste una computazione Γ di $\mathfrak{M}$ tale che $\mathtt{Input}(\Gamma) = X$ e $\mathsf{space}(\Gamma) \leq s(|X|)$. ■

Teorema 46. *Sia $\mathfrak{M}$ una TM e sia $c_{\mathfrak{M}} = m(r+2)$, dove m è il numero degli stati di $\mathfrak{M}$ ed r è il numero dei simboli dell'alfabeto di $\mathfrak{M}$. Per ogni computazione propria Ω di $\mathfrak{M}$ esiste una computazione Γ di $\mathfrak{M}$ tale che:*

(1) $\mathtt{Input}(\Gamma) = \mathtt{Input}(\Omega)$*;*

(2) $|\Gamma| \leq (\mathsf{space}(\Omega) + |\mathtt{Input}(\Omega)| + 1) c_{\mathfrak{M}}^{\mathsf{space}(\Omega)+|\mathtt{Input}(\Omega)|+1}$

Dimostrazione. Data la computazione propria Ω di $\mathfrak{M}$, sia $\Gamma = (\gamma_0, \gamma_1, \ldots, \gamma_k)$ una computazione di $\mathfrak{M}$ di lunghezza minimale tale che $\mathtt{Input}(\Gamma) = X$ e $\mathsf{space}(\Gamma) \leq S$, dove $X = \mathtt{Input}(\Omega)$ e $S = \mathsf{space}(\Omega)$.

Poiché Γ ha lunghezza minimale, si ha che $\gamma_i \neq \gamma_j$ per ogni $0 \leq i < j \leq k$,[11] cioè le configurazioni di Γ sono tutte distinte tra loro. Inoltre, dato che $|\gamma_0| \leq |\gamma_1| \leq \cdots \leq |\gamma_k|$ e $|\gamma_k| \leq \mathsf{space}(\Gamma) + 1 + \max\{0, |X| - 1\}$ (cf. (6) del Teorema 45), si ha che $|\gamma_i| \leq S + 1 + |X|$, per ogni $i = 0, 1, \ldots, k$. Pertanto $k + 1 \leq N$ dove N è il numero delle configurazioni di $\mathfrak{M}$ di lunghezza minore o uguale a $S + 1 + |X|$. Determiniamo N. Per ogni $\ell \geq 1$ il numero delle configurazioni di $\mathfrak{M}$ di lunghezza ℓ è uguale a $(r+1)^{\ell-1} m(\ell - 1)$, e quindi

$$
\begin{aligned}
N &= \sum_{\ell=1}^{S+1+|X|} (r+1)^{\ell-1} m(\ell-1) = \sum_{\ell=0}^{S+|X|} (r+1)^{\ell} m\ell \\
&= m \sum_{\ell=0}^{S+|X|} (r+1)^{\ell} \ell \\
&\leq m(S+|X|+1) \sum_{\ell=0}^{S+|X|} (r+1)^{\ell} \\
&\leq m(S+|X|+1) \sum_{\ell=0}^{S+|X|} (r+2)^{\ell} \\
&= m(S+|X|+1) \frac{(r+2)^{S+1+|X|} - 1}{r+1} \\
&\leq m(S+|X|+1)(r+2)^{S+1+|X|} \\
&\leq (S+|X|+1)[m(r+2)]^{S+|X|+1} = (S+|X|+1) c_{\mathfrak{M}}^{S+|X|+1}.
\end{aligned}
$$

Pertanto $N \leq (S+|X|+1) c_{\mathfrak{M}}^{S+|X|+1}$ e quindi $|\Gamma| = k \leq (S+|X|+1) c_{\mathfrak{M}}^{S+|X|+1}$. □

Definizione 58. La CLASSE DEI LINGUAGGI ACCETTATI IN SPAZIO NONDETERMINISTICO POLINOMIALE è la classe **NPSPACE** dei linguaggi $\mathscr{L}$ per cui esistono una TM $\mathfrak{M}$ ed una funzione polinomiale p tali che $\mathfrak{M}$ accetta $\mathscr{L}$ usando spazio p.

La CLASSE DEI LINGUAGGI ACCETTATI IN SPAZIO DETERMINISTICO POLINOMIALE è la classe **PSPACE** dei linguaggi $\mathscr{L}$ per cui esistono una DTM $\mathfrak{D}$ ed una funzione polinomiale p tali che $\mathfrak{D}$ accetta $\mathscr{L}$ usando spazio p. ■

Teorema 47. *Se $\mathscr{L} \in$ **NPSPACE** allora esistono $c \in \mathbb{N}$ ed una funzione polinomiale p tali che $\mathscr{L} \in$ **NTIME**(t), dove $t(n) = p(n)(c+2)^{p(n)}$, per ogni $n \in \mathbb{N}$.*

[11] Infatti, se si avesse $\gamma_i = \gamma_j$, dove $i < j$, allora la sequenza $(\gamma_0, \ldots, \gamma_i, \gamma_{j+1}, \ldots, \gamma_k)$ sarebbe una computazione di $\mathfrak{M}$ con input X, di lunghezza minore della lunghezza di Γ e che usa spazio al più S, contraddicendo la "minimalità" di Γ.

Dimostrazione. Sia $\mathscr{L} \in$ **NPSPACE** e siano $\mathfrak{M}$ una TM e q una funzione polinomiale tali che $\mathfrak{M}$ riconosce $\mathscr{L}$ usando spazio q. Ponendo $c = m(r+2)-2$ e $p(n) = n+q(n)+1$, dove m è il numero degli stati di $\mathfrak{M}$ ed r è il numero dei simboli dell'alfabeto di $\mathfrak{M}$, dal Teorema 46 segue che $\mathscr{L} \in$ **NTIME**$(p(n)(c+2)^{p(n)})$. □

Teorema 48. PSPACE $\subseteq$ NPSPACE. *Se $\mathscr{L} \in$* **NPSPACE** *allora $\mathscr{L}$ è decidibile.*

Dimostrazione. Se $\mathscr{L} \in$ **NPSPACE**, la decidibilità di $\mathscr{L}$ segue dal Teorema 47 usando argomentazioni simili a quelle del Teorema 7. □

Teorema 49. NP $\subseteq$ NPSPACE.

Dimostrazione. Segue da (2) del Teorema 45. □

Teorema 50 (Teorema di Savitch)**. PSPACE = NPSPACE**.

Definizione 59. Indichiamo con **EXPTIME** la classe dei linguaggi $\mathscr{L}$ per i quali esiste una funzione polinomiale $p(n)$ tale che $\mathscr{L} \in$ **TIME**$(2^{p(n)})$. ■

Teorema 51. PSPACE $\subseteq$ EXPTIME.

Dimostrazione. Segue dal Teorema 46. (Si veda la dimostrazione del Teorema 47). □

Teorema 52. P $\subseteq$ NP $\subseteq$ NPSPACE = PSPACE $\subseteq$ EXPTIME.

Dimostrazione. Segue dai teoremi 7, 49, 50 e 51. □

Teorema 53. P $\neq$ EXPTIME.