

del grafo,  $\varphi_G$  usa le seguenti formule booleane, per  $0 \leq i, j < n$ ,

$$C_{i,j} = a_{i,j} \Rightarrow [(r_i \wedge \neg r_j) \vee (g_i \wedge \neg g_j) \vee (b_i \wedge \neg b_j)] = \neg a_{i,j} \vee (r_i \wedge \neg r_j) \vee (g_i \wedge \neg g_j) \vee (b_i \wedge \neg b_j)$$

(informalmente,  $C_{i,j}$  afferma che se vi è un arco tra i due vertici  $i$  e  $j$ , allora questi due vertici non possono avere lo stesso colore). In conclusione, la formula  $\varphi_G$  è la seguente.

$$\bigwedge_{0 \leq i, j < n} A_{i,j} \wedge \bigwedge_{0 \leq i < n} B_i \wedge \bigwedge_{0 \leq i, j < n} C_{i,j}$$

Chiaramente,  $\varphi_G$  è soddisfacibile se e solo se i vertici di  $G$  possono essere colorati con tre colori. Il teorema di Cook-Levin afferma sostanzialmente che quanto abbiamo appena fatto per il problema della colorazione può in realtà essere fatto per qualunque linguaggio in NP.

**Teorema 7.2**

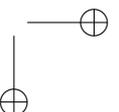
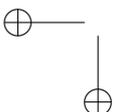
SAT è NP-completo.

**Dimostrazione.** Sia  $L$  un linguaggio in NP e siano  $p$  e  $V$  come nella definizione di NP, ovvero tali che, per ogni stringa  $x$ ,

$$x \in L \Leftrightarrow \exists y [ |y| = p(|x|) \wedge V(x, y) \text{ termina in } q_{si} ]$$

(osserviamo che non è restrittivo assumere che il certificato abbia esattamente lunghezza pari a  $p(|x|)$ ). Sia poi  $q$  il polinomio che limita il tempo di calcolo di  $V$  ovvero, con input  $x$  e  $y$ ,  $V(x, y)$  esegue al più  $q(|x|)$  passi. Per semplicità, assumiamo che l'alfabeto di lavoro di  $V$  sia  $\{\sigma_0 = @, \sigma_1 = 0, \sigma_2 = 1\}$ , che gli stati di  $V$  siano  $q_0, q_1, \dots, q_k$  e che  $q_0$  sia lo stato iniziale, che  $q_1 = q_{si}$  e che  $q_2 = q_{no}$ . Assumiamo inoltre che il nastro di  $V$  sia semi-infinito e che la testina possa solo spostarsi a destra oppure a sinistra. Come già detto in precedenza, l'idea della dimostrazione è quella di simulare, per ogni  $x$ , attraverso le assegnazioni di verità a una formula booleana (non necessariamente in forma normale congiuntiva) la computazione  $V(x, y)$  con  $y$  da determinare. A tale scopo, faremo uso delle seguenti variabili.

- $P_{s,t}^i$ : tale variabile ha il valore `true` se e solo se la cella  $s$  contiene il simbolo  $\sigma_i$  al tempo  $t$  ( $0 \leq i \leq 2, 0 \leq s \leq q(|x|), 0 \leq t \leq q(|x|)$ ).
- $Q_t^i$ : tale variabile ha il valore `true` se e solo se  $V$  è nello stato  $q_i$  al tempo  $t$  ( $0 \leq i \leq k, 0 \leq t \leq q(|x|)$ )
- $S_{s,t}$ : tale variabile ha il valore `true` se e solo se la testina è posizionata sulla cella  $s$  al tempo  $t$  ( $0 \leq s \leq q(|x|), 0 \leq t \leq q(|x|)$ )



La formula booleana globale è costituita dalla congiunzione delle seguenti formule booleane, il cui scopo è quello di verificare specifiche caratteristiche della computazione di  $V$  con input  $x$  e  $y$  (da determinare).

**Posizione della testina** Se soddisfatta, la formula  $A$  afferma che, in ogni istante  $t$ , la testina è posizionata esattamente su una cella. In particolare,

$$A = A_0 \wedge A_1 \wedge \dots \wedge A_{q(|x|)}$$

dove

$$A_t = (S_{0,t} \vee S_{1,t} \vee \dots \vee S_{q(|x|),t}) \wedge \left( \bigwedge_{(i,j)} : 0 \leq i < q(|x|), i < j \leq q(|x|) \right) [S_{i,t} \rightarrow \neg S_{j,t}]$$

**Contenuto di una cella** Se soddisfatta, la formula  $B$  afferma che, in ogni istante  $t$ , ogni cella contiene esattamente un simbolo. In particolare,

$$B = B_{0,0} \wedge \dots \wedge B_{q(|x|),0} \wedge B_{0,1} \wedge \dots \wedge B_{q(|x|),1} \wedge \dots \wedge B_{0,q(|x|)} \wedge \dots \wedge B_{q(|x|),q(|x|)}$$

dove

$$B_{s,t} = (P_{s,t}^0 \vee P_{s,t}^1 \vee P_{s,t}^2) \wedge \left( \bigwedge_{(i,j)} : 0 \leq i < 2, i < j \leq 2 \right) [P_{s,t}^i \rightarrow \neg P_{s,t}^j]$$

**Stato** Se soddisfatta, la formula  $C$  afferma che, in ogni istante  $t$ ,  $V$  si trova in un solo stato. In particolare,

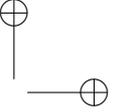
$$C = C_1 \wedge C_2 \wedge \dots \wedge C_{q(|x|)}$$

dove

$$C_t = (Q_t^0 \vee \dots \vee Q_t^k) \wedge \left( \bigwedge_{(i,j)} : 0 \leq i < k, i < j \leq k \right) [Q_t^i \rightarrow \neg Q_t^j]$$

**Input** Se soddisfatta, la formula  $D_x$  afferma che, all'istante 0, le prime  $n = |x|$  celle contengono la stringa  $x = \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_n}$  e che la cella successiva contiene  $\mathcal{O}$ . In particolare,

$$D_x = P_{0,0}^{i_1} \wedge \dots \wedge P_{n-1,0}^{i_n} \wedge P_{n,0}^{\mathcal{O}}$$



**Certificato** Se soddisfatta, la formula  $D^y$  afferma che, all'istante 0, le successive  $m = p(|x|)$  celle contengono una stringa binaria. In particolare,

$$D^y = D_{n+1}^y \wedge D_{n+2}^y \wedge \dots \wedge D_{n+p(|x|)}^y$$

dove

$$D_s^y = (P_{s,0}^1 \vee P_{s,0}^2) \wedge (\neg P_{s,0}^1 \vee \neg P_{s,0}^2)$$

**Blank** Se soddisfatta, la formula  $D^{\mathcal{Q}}$  afferma che, all'istante 0, le successive celle contengono  $\mathcal{Q}$ . In particolare,

$$D^{\mathcal{Q}} = P_{n+m+1,0}^{\mathcal{Q}} \wedge P_{n+m+2,0}^{\mathcal{Q}} \wedge \dots \wedge P_{q(|x|),0}^{\mathcal{Q}}$$

**Stato iniziale** Se soddisfatta, la formula  $E$  afferma che, all'istante 0, lo stato è quello iniziale e che la testina è posizionata sulla prima cella. In particolare,

$$E = Q_0^0 \wedge S_{0,0}$$

**Stato finale** Se soddisfatta, la formula  $F$  afferma che, a un certo istante,  $V$  termina nello stato di accettazione. In particolare,

$$F = Q_0^1 \vee Q_1^1 \vee \dots \vee Q_{q(|x|)}^1$$

**Transizioni** Se soddisfatta,  $G$  afferma che, a ogni istante,  $V$  esegue una transizione legittima. Chiaramente  $G$  dipende dal grafo delle transizioni di  $V$ . Supponiamo, ad esempio, che, leggendo  $\sigma_j$ ,  $V$  va dallo stato  $q_i$  allo stato  $q_{i_1}$ , scrive  $\sigma_{j_1}$  e esegue il movimento a destra. Allora, per ogni istante  $t$  e per ogni cella  $s$ ,  $G$  include la formula

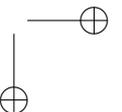
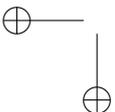
$$(Q_t^i \wedge P_{s,t}^j \wedge S_{s,t}) \rightarrow (Q_{t+1}^{i_1} \wedge P_{s,t+1}^{j_1} \wedge S_{s+1,t+1})$$

e, poiché dobbiamo anche assicurarci che le altre celle non cambino contenuto, la formula

$$P_{s,t}^j \wedge \neg S_{s,t} \rightarrow P_{s,t+1}^j$$

Dalla costruzione segue immediatamente che se esiste una stringa  $y$ , con  $|y| = p(|x|)$ , tale che  $V(x, y)$  termina in  $q_{si}$ , allora a partire da  $y$  e dalla computazione  $V(x, y)$  possiamo derivare un'assegnazione che soddisfa la formula. Viceversa, da un'assegnazione che soddisfa la formula possiamo ricavare dal valore delle variabili che appaiono in  $D^y$  un certificato  $y$  tale che  $V(x, y)$  termina in  $q_{si}$ . La costruzione della formula può chiaramente essere fatta in tempo polinomiale, per cui  $L$  è polinomialmente riducibile a SAT: poiché  $L$  era un generico linguaggio in NP, il teorema di Cook-Levin risulta essere dimostrato.  $\square$

Una volta dimostrata l'esistenza di un primo linguaggio NP-completo, possiamo ora mostrare la NP-completezza di altri linguaggi partendo da SAT. Questo è quanto faremo nel resto di questo capitolo, riesaminando i quattro problemi descritti precedentemente.



**10.68.** Any boolean formula can be converted to an equivalent **cnf**. In general, this **conversion** is an exponential process. However, there exists a **polynomial time conversion** of boolean formulas that preserves satisfiability. This process of **polynomial time cnf conversion** justifies our starting point with a **cnf** for defining SAT. To see this, prove the following:

(a) Let  $X = C_1 \wedge C_2 \wedge \dots \wedge C_m$  and  $Y = D_1 \wedge D_2 \wedge \dots \wedge D_n$  be two cnfs, where the  $C_i, D_j$  are disjunctive clauses. Let  $x$  be a propositional variable occurring neither in  $X$  nor in  $Y$ . Construct the **cnf**:

$$Z = (C_1 \vee x) \wedge (C_2 \vee x) \wedge \dots \wedge (C_m \vee x) \wedge (D_1 \vee \neg x) \wedge (D_2 \vee \neg x) \dots \wedge (D_n \vee \neg x).$$

Suppose  $f$  is a model of  $X$ . Extend  $f$  to the interpretation  $g$  by assigning  $x$  to 0 (0 for falsity and 1 for truth). Then,  $g$  is a model of  $Z$ .

(b) Let  $X, Y, Z$  be as in (a). Each model of  $X \vee Y$  can be extended to a model of  $Z$ . Conversely, each model of  $Z$  is a model of  $X \vee Y$ .

(c) Given a boolean formula, move the  $\neg$  sign to the variables by using the laws of De Morgan and the double negation:

$$\neg(A \vee B) \equiv \neg A \wedge \neg B, \quad \neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg\neg A \equiv A.$$

It takes a time **polynomial** in the length of the given boolean formula.

(d) Define the **conversion** procedure inductively by assuming that the given expression is in either of the forms  $A \wedge B$  or  $A \vee B$ , where  $A, B$  are in **cnf**. Note that for an actual construction, you have to identify the innermost propositions that are not in **cnf**, and then build it up. In the first case, the expression is a **cnf**. In the second case, use the construction in (a) by taking  $A, B$  as  $X, Y$  for obtaining the expression  $Z$ . If  $E$  is a proposition and  $F$  is the **cnf** obtained by this construction, then  $E$  is satisfiable iff  $F$  is satisfiable.

(e) The construction of  $F$  from  $E$  as described here takes an  $O(n^2)$  time, where  $n$  is the length of the expression  $E$ .