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

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

(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 φ_G è la seguente.

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

Chiaramente, φ_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,

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

(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 $\{\sigma_0 = \mathtt{@}, \sigma_1 = \mathtt{0}, \sigma_2 = \mathtt{1}\}$, che gli stati di V siano $q_0, q_1, \ldots, q_k$ e che q_0 sia lo stato iniziale, che $q_1 = q_{\mathtt{si}}$ e che $q_2 = q_{\mathtt{no}}$. 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.

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

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,

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

dove

$$\begin{aligned} A_t &= (S_{0,t} \vee S_{1,t} \vee \ldots \vee S_{q(|x|),t}) \\ &\wedge \left(\bigwedge_{(i,j)} : 0 \leqslant i < q(|x|), i < j \leqslant q(|x|) \right) [S_{i,t} \rightarrow \neg S_{j,t}] \end{aligned}$$

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

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

dove

$$\begin{aligned} B_{s,t} &= (P^0_{s,t} \vee P^1_{s,t} \vee P^2_{s,t}) \\ &\wedge \left(\bigwedge_{(i,j)} : 0 \leqslant i < 2, i < j \leqslant 2 \right) [P^i_{s,t} \rightarrow \neg P^j_{s,t}] \end{aligned}$$

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

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

dove

$$\begin{aligned} C_t &= (Q^0_t \vee \ldots \vee Q^k_t) \\ &\wedge \left(\bigwedge_{(i,j)} : 0 \leqslant i < k, i < j \leqslant k \right) [Q^i_t \rightarrow \neg Q^j_t] \end{aligned}$$

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

$$D_x = P^{i_1}_{0,0} \wedge \ldots \wedge P^{i_n}_{n-1,0} \wedge P^0_{n,0}$$

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

$$D^y = D^y_{n+1} \land D^y_{n+2} \land \ldots \land D^y_{n+p(|x|)}$$

dove

$$D^y_s = (P^1_{s,0} \lor P^2_{s,0}) \land (\neg P^1_{s,0} \lor \neg P^2_{s,0})$$

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

$$D^{@} = P^0_{n+m+1,0} \land P^0_{n+m+2,0} \land \cdots \land P^0_{q(|x|),0}$$

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,

$$E = Q^0_0 \land S_{0,0}$$

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

$$F = Q^1_0 \lor Q^1_1 \lor \ldots \lor Q^1_{q(|x|)}$$

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 σ_j, V va dallo stato q_i allo stato q_{i_1}, scrive σ_{j_1} e esegue il movimento a destra. Allora, per ogni istante t e per ogni cella s, G include la formula

$$(Q^i_t \land P^j_{s,t} \land S_{s,t}) \rightarrow (Q^{i_1}_{t+1} \land P^{j_1}_{s,t+1} \land S_{s+1,t+1})$$

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

$$P^j_{s,t} \land \neg S_{s,t} \rightarrow P^j_{s,t+1}$$

Dalla costruzione segue immediatamente che se esiste una stringa y, con $|y| = p(|x|)$, tale che $V(x,y)$ termina in $q_{\mathtt{si}}$, 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 $q_{\mathtt{si}}$. 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. □

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 $X = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ and $Y = D_1 \wedge D_2 \wedge \cdots \wedge D_n$ be two cnfs, where the C_i, D_j are disjunctive clauses. Let x be a propositional variable occurring neither in X not in Y. Construct the cnf:

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

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:

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

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.

Using DeMorgan only can be exponential: (a or b) and (c or d) and ....
Unknwon how to convert DNF in poly time since DNF can be satisified in poly time