11.5.1 Σχέση μεταξύ P και NP

Προφανώς ισχύει $P \subseteq NP$. Δηλαδή οποιοδήποτε πρόβλημα λύνεται σε πολυωνυμικό χρόνο από έναν ντετερμινιστικό αλγόριθμο μπορεί να λυθεί σε πολυωνυμικό χρόνο επίσης και από έναν μη- ντετερμινιστικό αλγόριθμο, επειδή ο ντετερμινιστικός αλγόριθμος είναι ειδική περίπτωση μη- ντετερμινιστικού αλγορίθμου (σε κάθε βήμα έχουμε μία μόνο επιλογή). Το ερώτημα είναι αν τα προβλήματα που λύνονται σε πολυωνυμικό χρόνο με ένα NDTM μπορούν να λυθούν σε πολυωνυμικό χρόνο και με ένα DTM, δηλαδή αν $P = NP$.

Γνωρίζουμε πάντως, ότι ένας μη-ντετερμινιστικός αλγόριθμος μπορεί να εξομοιωθεί από έναν ντετερμινιστικό με εκθετική όμως απώλεια χρόνου. Δηλαδή ισχύει:

$$NP \subseteq DEXPTIME$$

Οι περισσότεροι ερευνητές πιστεύουν πως το P είναι διαφορετικό από το NP. Παρ' όλα αυτά η απόδειξη αυτής της εικασίας παραμένει ως σήμερα ένα άλυτο πρόβλημα.

11.6 Η έννοια της αναγωγής - Το θεώρημα του Cook

11.6.1 Αναγωγή κατά Karp (Karp reduction)

Λέμε ότι ένα πρόβλημα A ανάγεται πολυωνυμικά κατά $Karp$ σε ένα πρόβλημα B και συμβολίζουμε, $A \leq_m^p B$, τότε και μόνο τότε, όταν υπάρχει μία συνάρτηση f υπολογίσιμη σε πολυωνυμικό χρόνο $f \in P_f$[1] τέτοια ώστε $\forall x(x \in A \iff f(x) \in B)$. Δηλαδή:

$$A \leq_m^p B : \exists f \in P_f, \forall x(x \in A \iff f(x) \in B)$$

Ισχύουν οι εξής ιδιότητες:

1. Ανακλαστική: $A \leq_m^p A$.

2. Μεταβατική: Αν $A \leq_m^p B$ και $B \leq_m^p C$ τότε $A \leq_m^p C$. Αυτό αποδεικνύεται εύκολα ως εξής:

   - $A \leq_m^p B : \exists f \in P_f, \forall x(x \in A \iff f(x) \in B)$
   - $B \leq_m^p C : \exists g \in P_f, \forall x(x \in B \iff g(x) \in C)$

[1] P_f είναι η κλάση των συναρτήσεων που υπολογίζονται σε πολυωνυμικό χρόνο.

Δηλαδή:$\exists f, g \in P_f, \forall x((x \in A \iff f(x) \in B) \iff g(f(x)) \in C)$. Άρα

$$\exists h \in P_f, \forall x(x \in Ax \iff h(x) \in C)$$

όπου h είναι η σύνθεση των f, g και είναι πολυωνυμική αφού η σύνθεση πολυωνύμων είναι πολυώνυμο. Συνεπώς $A \leq_m^p C$.

3. Αν $A \leq_m^p B$ και $B \leq_m^p A$ τότε $A \equiv_m^p B$, και λέμε ότι τα προβλήματα A, B είναι ισοδύναμα ως προς $\leq_m^p$, (π.χ. αν $A,\ B \in P \Rightarrow A \equiv_m^p B$).

4. Αν $A \leq_m^p B$ και $B \in P \Rightarrow A \in P$. Αυτό αποδεικνύεται εύκολα ως εξής: Αφού $A \leq_m^p B$ σημαίνει ότι υπάρχει συνάρτηση f υπολογίσιμη σε πολυωνυμικό χρόνο τέτοια ώστε:

$$\forall x(x \in A \iff f(x) \in B)$$

Κάθε στιγμιότυπο x του προβλήματος A λοιπόν, που μας δίνεται μπορούμε να το μετασχηματίζουμε (μέσω της f) σε ένα στιγμιότυπο $f(x)$, του προβλήματος B. Εκτός αυτού, το μήκος του $f(x)$ είναι πολυωνυμικό ως προς το μήκος του x. Όμως $B \in P$, δηλαδή υπάρχει ντετερμινιστικός αλγόριθμος ο οποίος λύνει σε πολυωνυμικό χρόνο το B. Τρέχουμε αυτόν τον αλγόριθμο και αν απαντά ότι $f(x) \in B$ τότε σημαίνει ότι $x \in A$, ειδάλλως, αν δηλαδή $f(x) \notin B$ τότε $x \notin A$. Άρα έχουμε έναν ντετερμινιστικό αλγόριθμο, ο οποίος σε πολυωνυμικό χρόνο αποφασίζει το A. Συνεπώς $A \in P$.

11.6.2 Hardness - Completeness

Αν έχουμε μία κλάση προβλημάτων C και ισχύει ότι: $\forall B \in C\colon B \leq A$ τότε το πρόβλημα A ονομάζεται C-δύσκολο (C-*hard*) ως προς $\leq$. Αν επιπλέον $A \in C$ τότε το A ονομάζεται C-**πλήρες** (C-*complete*), δηλαδή ανήκει στα πιο δύσκολα προβλήματα της κλάσης C (χαρακτηρίζει την δυσκολία της κλάσης C).

Ένα πρόβλημα L είναι **NP-complete** ως προς $\leq_m^p$ αν:

$$(L \in NP) \wedge (\forall L' \in NP :\ L' \leq_m^p L)$$

Τα **NP-complete** είναι τα πιο δύσκολα προβλήματα της κλάσης NP. Αν ένα **NP-complete** πρόβλημα αποδειχθεί ότι ανήκει στο P, τότε (αφού όλα τα προβλήματα που ανήκουν στο NP ανάγονται σ' αυτό), σύμφωνα με την ιδιότητα (4), όλα τα προβλήματα του NP θα ανήκουν στο P.

Λήμμα 11.6.1. Αν $L_1 \leq_m^p L_2$, το L_1 είναι **NP-complete** και $L_2 \in NP$ τότε το L_2 είναι **NP-complete**.

Proof. Αφού L_1 είναι NP-complete σημαίνει ότι για οποιοδήποτε $L_3 \in NP$ ισχύει $L_3 \leq_m^p L_1$. Όμως από την υπόθεση έχουμε: $L_1 \leq_m^p L_2$ και σύμφωνα με την ιδιότητα (2) της αναγωγής $\leq_m^p$, έχουμε $L_3 \leq_m^p L_2$. Συνεπώς το L_2 είναι NP-complete. □

Το παραπάνω λήμμα υποδεικνύει τον τρόπο με τον οποίο χρησιμοποιούμε την αναγωγή. Προσπαθούμε να ανάγουμε ένα γνωστό **NP-complete** πρόβλημα σε ένα πρόβλημα για το οποίο ξέρουμε ότι ανήκει στο NP, αποδεικνύοντας έτσι ότι και αυτό είναι NP-complete. Για να δείξουμε όμως ότι κάποια προβλήματα είναι NP-complete, χρειαζόμαστε ένα πρόβλημα που να ξέρουμε ότι είναι **NP-complete** προκειμένου να το ανάγουμε σε αυτά. Αυτό το «πρώτο» **NP-complete** πρόβλημα μας το έδωσε ο Cook με το θεώρημά του που θα διατυπώσουμε σε επόμενη παράγραφο.

11.6.3 Αναγωγή κατά Cook (Cook reduction)

Λέμε ότι ένα πρόβλημα A ανάγεται κατά Cook σε ένα πρόβλημα B και συμβολίζουμε, $A \leq_T^P B$, αν το A μπορεί να αποφασιστεί από μία πολυωνυμικού χρόνου ντετερμινιστική μηχανή Turing η οποία χρησιμοποιεί ένα **μαντείο** (*oracle*) για το B. Αυτό σημαίνει ότι η DTM μπορεί να κάνει οσεσδήποτε ερωτήσεις για οποιοδήποτε στιγμιότυπο του B και να πάρει στιγμιαία σωστές απαντήσεις.

Για να το επιτύχει αυτό η μηχανή Turing M έχει μια επιπλέον ταινία ονομαζόμενη *query -tape* στην οποία π.χ. τοποθετεί ένα string που κωδικοποιεί στιγμιότυπο x του προβλήματος B. Το μαντείο τότε αποφαίνεται $x \in B$ ή $x \notin B$ επάνω στην query -tape σε χρόνο που δεν μετράει για την πολυπλοκότητα της M. Η M μπορεί να χρησιμοποιήσει την απάντηση και αργότερα να τοποθετήσει άλλα ερωτήματα -strings στην query -tape.

Συμβολίζουμε με P^A την κλάση των προβλημάτων που μπορούν να λυθούν με ένα DTM σε πολυωνυμικό χρόνο χρησιμοποιώντας μαντείο για το A, ή αλλιώς:

$$P^A = \{L \mid L \leq_T^P A\}$$

Εντελώς ανάλογα με την αναγωγή κατά Karp, ισχύουν και εδώ οι ιδιότητες:

- ανακλαστική,
- μεταβατική,
- $\equiv_T^P$

Επίσης ορίζονται εντελώς ανάλογα οι έννοιες:

- NP-hard ως προς $\leq_T^P$
- NP-complete ως προς $\leq_T^P$

Επιπλέον ισχύει ότι:$A \leq_m^p B \Rightarrow A \leq_T^P B$

Συνεπώς αν ένα πρόβλημα είναι NP-complete ως προς $\leq_m^p$ τότε είναι NP-complete ως προς $\leq_T^P$. Το αντίστροφο όμως δεν ισχύει. Από εδώ και στο εξής (δηλαδή στις αναγωγές των προβλημάτων, αλλά και στο θεώρημα του Cook), θα χρησιμοποιήσουμε την αναγωγή κατά Karp ($\leq_m^p$) διότι είναι πιο εύχρηστη.

11.6.4 Το Θεώρημα του Cook

Πριν διατυπώσουμε το θεώρημα του Cook, κρίνεται σκόπιμη μια επανάληψη βασικών στοιχείων του προτασιακού λογισμού. Οι λογικές μεταβλητές που χρησιμοποιούμε στον προτασιακό λογισμό, παίρνουν τιμές στο σύνολο $\{T, F\}$. Οι τελεστές που συνδέουν λογικές μεταβλητές για το σχηματισμό λογικών εκφράσεων (boolean formulas) είναι: $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$.

Έτσι λοιπόν αν $p_1,\ p_2$ είναι λογικές μεταβλητές τότε αυτές μπορούν να δώσουν τις παρακάτω λογικές εκφράσεις:

- $\neg p$
- $p_1 \wedge p_2$
- $p_1 \vee p_2$
- $p_1 \rightarrow p_2$
- $p_1 \leftrightarrow p_2$

Δύο λογικές εκφράσεις λέγονται ισοδύναμες όταν έχουν την ίδια αληθοτιμή για όλες τις δυνατές απονομές αληθοτιμών στις λογικές μεταβλητές. Αυτό μπορεί να ελεγχθεί π.χ. με πίνακες αληθείας.

Η λογική έκφραση $p_1 \leftrightarrow p_2$ είναι ισοδύναμη με την $(p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$. Συνεπώς δεν χρειαζόμαστε τον δυαδικό τελεστή $\leftrightarrow$. Επίσης η έκφραση $p_1 \rightarrow p_2$ είναι ισοδύναμη με την $\neg p_1 \vee p_2$. Άρα δεν χρειαζόμαστε ούτε τον τελεστή $\rightarrow$. Παρ' όλο που και από τους τρεις εναπομείναντες τελεστές, δεν είναι όλοι αναγκαίοι, θα χρησιμοποιήσουμε και τους τρεις για μεγαλύτερη ευκολία.

Ορισμός 11.6.2. Αν $x_1, x_2, \ldots$ είναι προτασιακές μεταβλητές, ονομάζουμε:

- literals: όρους όπως $x_1, x_3, \neg x_1, \neg x_5$
- clauses: διαζεύξεις (disjunctions) από litetals, π.χ. $(x_1 \vee \neg x_2 \vee \neg x_5)$

- Conjuctive Normal Form (CNF) την ακόλουθη μορφή που μπορεί να έχει η boolean formula:

$$(clause_1 \wedge clause_2 \wedge \ldots \wedge clause_m)$$

Θεώρημα 11.6.3. *Κάθε λογική έκφραση είναι ισοδύναμη με μία σε CNF.*

Ορισμός 11.6.4. Ικανοποιήσιμη (satisfiable) ονομάζεται μία boolean formula όταν υπάρχει απονομή αλήθειας (truth assignment) στις προτασιακές μεταβλητές που την αποτελούν, έτσι ώστε η έκφραση να παίρνει την τιμή True (T).

Το πρόβλημα SAT
Δεδομένα: Μία boolean formula σε CNF.
Ερώτηση: Είναι η boolean formula ικανοποιήσιμη;

Παράδειγμα 11.6.5. Η φόρμουλα $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$ είναι ικανοποιήσιμη. Μία απονομή αλήθειας που την ικανοποιεί είναι η $(x_1, x_2) = (T, T)$. Η φόρμουλα, $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \neg x_1$ δεν είναι ικανοποιήσιμη.

Θεώρημα 11.6.6. *(Cook) Το πρόβλημα SAT είναι NP-complete.*

Proof. Κατ' αρχάς πρέπει να αποδείξουμε ότι $SAT \in NP$. Αυτό όμως είναι εύκολο. Ένας μη-ντετερμινιστικός αλγόριθμος που λύνει το SAT είναι ο εξής:

1. μάντεψε μία απονομή αλήθειας

2. έλεγξε αν η έκφραση αποτιμάται σε True

Ο έλεγχος μπορεί να γίνει σε γραμμικό χρόνο ως προς το μέγεθος της φόρμουλας και συνεπώς $SAT \in NP$.

Το επόμενο που πρέπει να δείξουμε είναι ότι όλα τα προβλήματα που ανήκουν στο NP, ανάγονται (κατά Karp) στο SAT. Όλα αυτά τα προβλήματα όμως, μπορούν να επιλυθούν (εξ ορισμού αφού ανήκουν στο NP) από μία NDTM που τρέχει σε πολυωνυμικό χρόνο. Έτσι λοιπόν, αρκεί να αποδείξουμε το εξής: Ένας οποιοσδήποτε υπολογισμός που κάνει μία NDTM πολυωνυμικού χρόνου M μπορεί να αναπαρασταθεί από μία boolean formula Φ πολυωνυμικού μήκους έτσι ώστε, να υπάρχει υπολογισμός του M που αποδέχεται το x αν και μόνο αν υπάρχει απονομή αλήθειας που ικανοποιεί την boolean formula $\Phi(x)$.

Το NDTM πρόγραμμα M θα δέχεται λοιπόν, σαν είσοδο ένα string x με $|x| = n$. Αν υπάρχει υπολογισμός στο τέλος του οποίου το M αποδέχεται το

x , αυτός ο υπολογισμός θα γίνεται σε πολυωνυμικό χρόνο $T_M(n)$. Δηλαδή θα υπάρχει ένα πολυώνυμο $p(n)$ έτσι ώστε:

$$T_M(n) \leq p(n), \forall n$$

Θεωρούμε ότι η κεφαλή βρίσκεται αρχικά στη θέση 0 και εφόσον ο αριθμός βημάτων είναι το πολύ $p(n)$, μπορούν να γραφτούν οι θέσεις της ταινίας από $-p(n)$ έως $p(n)$. Οι διαφορετικές χρονικές στιγμές είναι $p(n)+1$ γιατί μέσα σ' αυτό το χρόνο γίνονται $p(n)$ βήματα. Δηλαδή αν το πρόγραμμα M αποδέχεται το x, σημαίνει ότι μετά από $p(n)$ βήματα φτάνει σε κατάσταση q_Y. Φυσικά μπορεί να φτάσει και πιο νωρίς. Σ' αυτήν την τελευταία περίπτωση θεωρούμε ότι το configuration παραμένει σταθερό σε όλες τις μεταγενέστερες χρονικές στιγμές. θεωρούμε πως η χρονική στιγμή στην οποία αρχίζουν οι υπολογισμοί για την επαλήθευση είναι η στιγμή 0. Άρα η τελευταία χρονική στιγμή είναι η $p(n)$.

Το σύνολο των καταστάσεων στις οποίες μπορεί να βρεθεί η T.M. είναι,

$$Q = \{q_0, q_1, \ldots, q_r\}, \qquad r = |Q| - 1, \quad q_1 = q_Y, \quad q_2 = q_N$$

Το σύνολο των συμβόλων που μπορεί να γράφει η T.M. είναι,

$$\Gamma = \{s_0, s_1, \ldots, s_v\}, \qquad v = |\Gamma| - 1, \quad s_0 = \text{blank}$$

Οι λογικές μεταβλητές που θα χρησιμοποιήσουμε για να κατασκευάσουμε τη φόρμουλα $\Phi(x)$ είναι:

- $Q[i,k],\ 0 \leq i \leq p(n),\ 0 \leq k \leq r$ κωδικοποιούν την κατάσταση q_k στην οποία βρίσκεται η T.M. τη χρονική στιγμή i . Δηλαδή η μεταβλητή $Q[i,k]$ θα είναι True αν και μόνο αν τη χρονική στιγμή i η T.M. βρίσκεται στην κατάσταση q_k. Ο αριθμός αυτών των μεταβλητών είναι $(r+1)\cdot(p(n)+1)$.

- $H[i,j],\ 0 \leq i \leq p(n),\ -p(n) \leq j \leq p(n)$ κωδικοποιούν τη θέση j στην οποία βρίσκεται η κεφαλή τη χρονική στιγμή i . Δηλαδή η μεταβλητή $H[i,j]$ θα είναι True ανν τη χρονική στιγμή i η κεφαλή βρίσκεται στη θέση (κυψέλη) j. Ο αριθμός αυτών των μεταβλητών είναι $(p(n)+1)\cdot(2p(n)+1)$.

- $S[i,j,l],\ 0 \leq i \leq p(n),\ -p(n) \leq j \leq p(n),\ 0 \leq l \leq v$ κωδικοποιούν το σύμβολο s_l, το οποίο περιέχεται στη θέση j τη χρονική στιγμή i. Δηλαδή η μεταβλητή $S[i,j,l]$ θα είναι True ανν τη χρονική στιγμή i, στη θέση j περιέχεται το σύμβολο s_l. Ο αριθμός αυτών των μεταβλητών είναι $(p(n)+1)\cdot(2p(n)+1)\cdot(v+1)$.

Ο αριθμός όλων των μεταβλητών που χρησιμοποιήσαμε είναι $O(p^2(n))$ (r, v είναι σταθερές της μηχανής), δηλαδή πολυωνυμικό ως προς το μέγεθος του input n.

Τα clauses που θα περιέχει η φόρμουλα $\Phi(x)$ είναι χωρισμένα σε 6 groups:

- G_1: Το group αυτό θα εκφράζει το γεγονός, ότι σε μία δεδομένη χρονική στιγμή η T.M. θα βρίσκεται ακριβώς σε μία κατάσταση. Δηλαδή μία δεδομένη χρονική στιγμή i, θα είναι True ακριβώς μία από τις μεταβλητές $Q[i,k], 0 \leq k \leq r$. Αυτό θα πρέπει να ισχύει για κάθε χρονική στιγμή. Το G_1 λοιπόν, θα λέει ότι μία δεδομένη χρονική στιγμή i, κάποια από τις μεταβλητές $Q[i,k]$ είναι True ενώ η σύζευξη οποιωνδήποτε δύο μεταβλητών απ' αυτές είναι False. Αυτό κωδικοποιείται εύκολα, όπως μπορεί να επαληθεύσει κανείς ως εξής:

$$\wedge_{i=0}^{p(n)}((\vee_{j=0}^{r} Q[i,j]) \wedge (\wedge_{j=0}^{r} \wedge_{k=0}^{j-1} (\neg Q[i,j] \vee \neg Q[i,k]))) \tag{11.1}$$

  Ο πρώτος όρος της παραπάνω έκφρασης εξασφαλίζει ότι σε μία δεδομένη χρονική στιγμή η μηχανή βρίσκεται τουλάχιστον σε μία κατάσταση και ο δεύτερος όρος ότι βρίσκεται το πολύ σε μία κατάσταση. Ο ολικός αριθμός των literals που περιέχονται σ' αυτά τα clauses είναι:

$$(p(n)+1) \cdot [(r+1) + \frac{r(r+1)}{2} \cdot 2] = (p(n)+1) \cdot (r+1)^2 = O(p(n))$$

- G_2: Το group αυτό θα εκφράζει το γεγονός ότι σε μία δεδομένη χρονική στιγμή, η κεφαλή θα βρίσκεται σε μία ακριβώς θέση. Δηλαδή μία δεδομένη χρονική στιγμή i, θα είναι True ακριβώς μία από τις μεταβλητές $H[i,j], -p(n) \leq j \leq p(n)$ και αυτό θα πρέπει να ισχύει για κάθε χρονική στιγμή. Εντελώς ανάλογα λοιπόν, με το group G_1, το G_2 κωδικοποιείται ως εξής:

$$\wedge_{i=0}^{p(n)}((\vee_{j=-p(n)}^{p(n)} H[i,j]) \wedge (\wedge_{j=-p(n)}^{p(n)} \wedge_{k=-p(n)}^{j-1} (\neg H[i,j] \vee \neg H[i,k]))) \tag{11.2}$$

  Ο ολικός αριθμός των literals που περιέχονται σ' αυτά τα clauses είναι,

$$(p(n)+1) \cdot (2p(n)+1)^2) = O(p^3(n))$$

- G_3: Το group αυτό θα εκφράζει το γεγονός ότι σε μία δεδομένη χρονική στιγμή, σε κάθε θέση στην ταινία περιέχεται ακριβώς ένα σύμβολο. Δηλαδή μία δεδομένη χρονική στιγμή i, για μία συγκεκριμένη θέση j, θα είναι True ακριβώς μία από τις μεταβλητές $S[i,j,l], 0 \leq l \leq v$ και αυτό θα πρέπει να ισχύει για κάθε χρονική στιγμή και για κάθε θέση.

Εντελώς ανάλογα λοιπόν, με τα groups G_1 και G_2, το G_3 θα αποτελείται από τα εξής clauses:

$$\wedge_{i=0}^{p(n)} \wedge_{j=-p(n)}^{p(n)} ((\vee_{l=0}^{v} S[i,j,l]) \wedge (\wedge_{l=0}^{v} \wedge_{k=0}^{l-1} (\neg S[i,j,l] \vee \neg S[i,j,k]))) \quad (11.3)$$

Συνολικά ο αριθμός των literals που περιέχονται στα clauses του G_3 είναι:

$$(p(n)+1) \cdot (2p(n)+1) \cdot (v+1)^2 = O(p^2(n))$$

- G_4: Το group αυτό θα δηλώνει ότι τη χρονική στιγμή 0 η T.M. βρίσκεται στο αρχικό configuration. Δηλαδή: Η κατάσταση στην οποία βρίσκεται η T.M. είναι q_0: $Q[0,0]$, η κεφαλή βρίσκεται στη θέση 1: $H[0,1]$, ότι στις θέσεις 1 έως n είναι γραμμένο το input x: $S[0,1,l_1] \wedge S[0,2,l_2] \wedge \ldots \wedge S[0,n,l_n]$, αν υποθέσουμε ότι $x = S_{l_1}, S_{l_2}, \ldots S_{l_n}$ και από τη θέση $n+1$ έως τη θέση $p(n)$ έχουμε κενά (*blanks*):

  $$S[0,n+1,0] \wedge S[0,n+2,0] \wedge \ldots \wedge S[0,p(n),0]$$

  Ακόμα, οι θέσεις $-p(n)$ έως 0 τη χρονική στιγμή 0 περιέχουν τον κενό χαρακτήρα ($S[0,0,0]$, κ.τ.λ.).

  Η σύζευξη όλων αυτών των clauses αποτελεί το group G_4. Ο συνολικός αριθμός των literals είναι $2p(n)+3 = O(p(n))$.

- G_5: Το group αυτό αποτελείται μόνο από ένα clause το οποίο έχει ένα litetal που δηλώνει, ότι τη χρονική στιγμή $p(n)$, η κατάσταση στην οποία βρίσκεται η T.M. είναι η $q_1 = q_Y$. Δηλαδή:

  $$Q[p(n),1]$$

- G_6: Το τελευταίο αυτό group θα δηλώνει πως το configuration της T.M. τη χρονική στιγμή $i+1$ προκύπτει από την εφαρμογή της συνάρτησης μετάβασης (transition function) δ, στο configuration της χρονικής στιγμής i. Κατ' αρχάς το G_6 θα δηλώνει πως το περιεχόμενο της θέσης j, δεν μπορεί να αλλάξει τη χρονική στιγμή $i+1$, αν τη χρονική στιγμή i η κεφαλή δεν βρισκόταν στη θέση j. Δηλαδή:

  $$(\neg H[i,j] \wedge S[i,j,l]) \rightarrow S[i+1,j,l],$$

  ή αλλιώς:

  $$\begin{cases} (H[i,j] \vee \neg S[i,j,l]) \vee S[i+1,j,l] \\ 0 \leq i \leq p(n), -p(n) \leq j \leq p(n), 0 \leq l \leq v \end{cases}$$

Έχουμε δηλαδή $3(2p(n)+1)\cdot p(n)\cdot(v+1)$ literals. Επιπλέον το G_6 θα δηλώνει πως οι αλλαγές που γίνονται στο configuration τη χρονική στιγμή $i+1$ προκύπτουν από την εφαρμογή της συνάρτησης μετάβασης δ, στο configuration της χρονικής στιγμής i. Δηλαδή:

$$(H[i,j]\wedge Q[i,k]\wedge S[i,j,l])\rightarrow \\ \vee_{m=1}^{|\delta(q_k,s_l)|}(H[i+1,j+\Delta_m]\wedge Q[i+1,k'_m]\wedge S[i+1,j,l'_m]) \quad (11.4)$$

Αυτό σημαίνει πως αν τη χρονική στιγμή i η T.M. βρίσκεται στην κατάσταση q_k με την κεφαλή στη θέση j να διαβάζει το σύμβολο s_l, τότε τη χρονική στιγμή $i+1$, το περιεχόμενο της θέσης j, η κατάσταση και η θέση της κεφαλής θα πρέπει να ικανοποιούν την μη ντετερμινιστική συνάρτηση μετάβασης. δηλαδή, αν $q_k\in Q\{q_Y,q_N\}$ τότε οι τιμές των Δ_m,k'_m,l'_m είναι τέτοιες ώστε να ισχύει:

$$(q_{k'_m},s_{l'_m},\Delta_m)\in\delta(q_k,s_l),\ \text{όπου}\Delta_m\in\{-1,0,1\}.$$

Την παραπάνω έκφραση μπορούμε να την φέρουμε σε CNF κάνοντας μερικές πράξεις και τελικά ο ολικός αριθμός των literals για δεδομένα i,j,k,l είναι $(c+3)3^c$, όπου $c=|\delta(q_k,s_l)|$. Αν $q_k\in\{q_Y,q_N\}$ τότε $\Delta=0,k'=k,l'=l$ (δηλαδή αν η T.M. έχει ήδη αποδεχθεί ή απορρίψει το x πριν τη στιγμή $p(n)$, διατηρεί το configuration όπως είναι μέχρι τη στιγμή $p(n)$). Ο ολικός αριθμός των literals που περιέχονται στα clauses του G_6 είναι: $O(p^2(n))$.

Η τελική φόρμουλα λοιπόν, είναι:

$$\Phi(x)=G_1\wedge G_2\wedge G_3\wedge G_4\wedge G_5\wedge G_6$$

Το μήκος της είναι: $O(p^3(n))$ (καθοριστικό είναι το μήκος του G_2). Συνεπώς μπορούμε να την κατασκευάσουμε σε πολυωνυμικό χρόνο ως προς n.

Επιπλέον, αν υπάρχει υπολογισμός του NDTM προγράμματος M που κάνει αποδεχτό το x σε χρόνο $p(n)$, τότε σίγουρα η $\Phi(x)$ έχει απονομή αλήθειας που την ικανοποιεί, λόγω της κατασκευής της.

Αντίστροφα, αν η $\Phi(x)$ έχει απονομή αλήθειας που την ικανοποιεί, τότε λόγω της κατασκευής της, αυτή η απονομή αλήθειας αντιστοιχεί σε έναν υπολογισμό του M που κάνει αποδεχτό το x. **Άρα το SAT είναι NP-complete.**

□