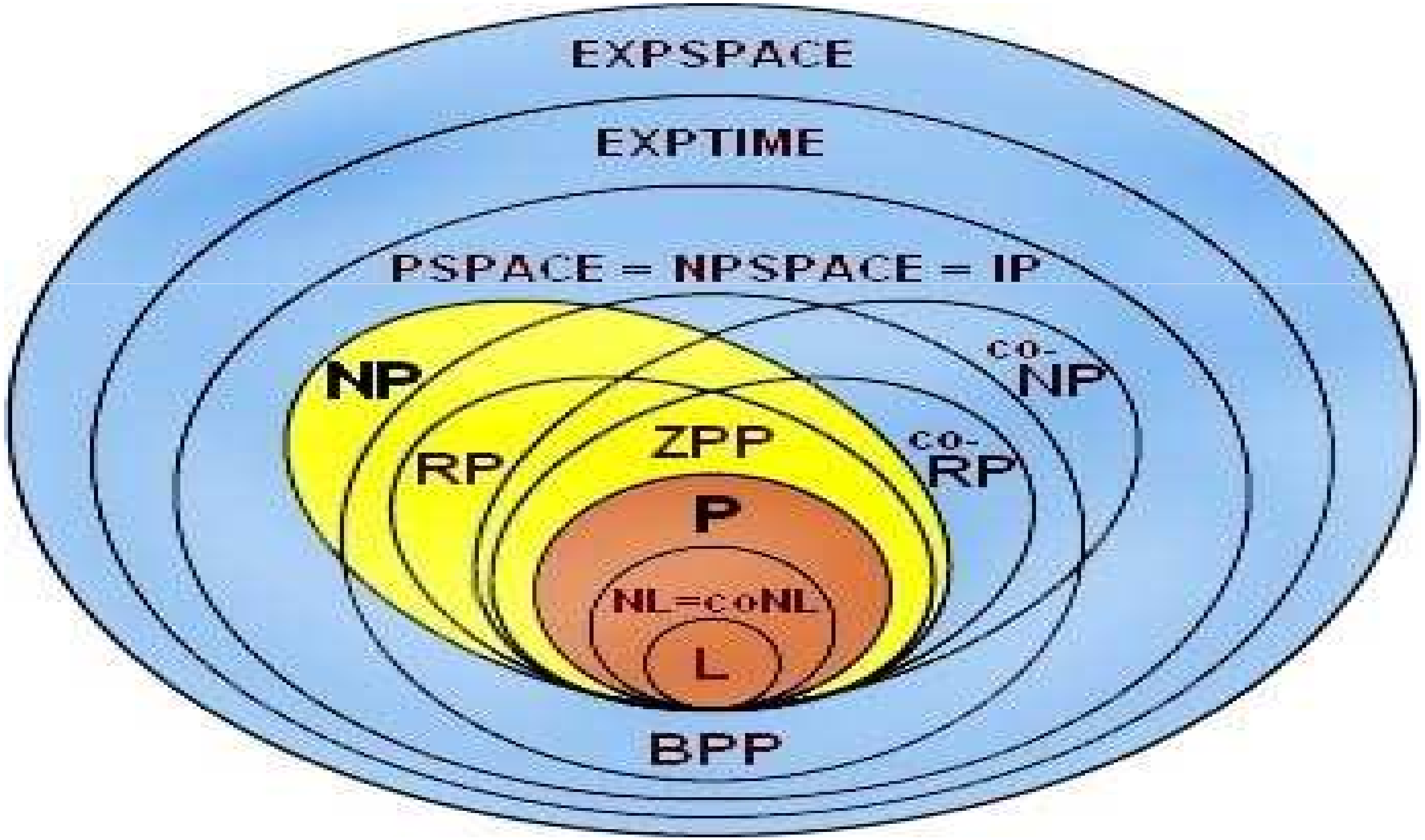
ΣΧΕΣΕΙΣ ΜΕΤΑΞΥ ΚΛΑΣΕΩΝ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ



Κλάσεις Πολυπλοκότητας

- Περιλαμβάνουν <u>αναδρομικές γλώσσες</u>
- Οι γλώσσες ταξινομούνται στις κλάσεις πολυπλοκότητας ανάλογα με τη δυσκολία απόφασης τους (<u>ποσότητα απαιτούμενων υπολογιστικών πόρων</u> απο τους αντίστοιχους αλγορίθμους)
- Προσδιορίζονται πλήρως απο τις εξής <u>παραμέτρους:</u>
  - Υπολογιστικό Μοντέλο → Μηχανή Turing πολλαπλών ταινιών
  - Τρόπος Υπολογισμού → Ντετερμινιστικός ή μη ντετερμινιστικός
  - Είδος υπολογιστικών πόρων προς περιορισμό → χώρος, χρόνος (συνηθέστερα)
  - « Κατάλληλη» συνάρτηση περιορισμού των υπολογιστικών πόρων f (proper complexity function)

<u>Ορισμός κλάσης πολυπλοκότητας</u>

Το σύνολο όλων των γλωσσών που αποφασίζονται απο μια *TM πολλαπλών ταινιών* που λειτουργεί με *καθορισμένο τρόπο*, έτσι ώστε, για οποιαδήποτε είσοδο x η TM να απαιτεί το πολύ $f(|x|)$ μονάδες απο το καθορισμένο *είδος υπολογιστικών πόρων*

Παραδείγματα κλάσεων πολυπλοκότητας: TIME(f), SPACE(f), NTIME(f), NSPACE(f)

Proper Complexity Functions

Μπορούμε να χρησιμοποιήσουμε οποιαδήποτε μη φθίνουσα συνάρτηση $f:\mathbb{N}^+ \to \mathbb{N}^+$ *ως συνάρτηση πολυπλοκότητας στον ορισμό μιας κλάσης?*

<u>Τυπικά *ΝΑΙ*</u>

Μπορούμε να κατασκευάσουμε συνάρτηση $f:\mathbb{N}^+ \to \mathbb{N}^+$ ώστε:

- Να έχει «περιέργη» ασυμπτωτική συμπεριφορά
- Να μη μπορεί να υπολογιστεί ακόμα και στον χώρο ή στον χρόνο που επιτρέπει
- Να αλλάζει τάξη μεγέθους αύξησης με την αύξηση του ορίσματος (πολυωνυμική $\longrightarrow$ εκθετική)

Η χρήση μιας τέτοιας συνάρτησης οδηγεί σε ανατρεπτικά συμπεράσματα σχετικά με τις σχέσεις μεταξύ των κλάσεων πολυπλοκότητας

Π.χ. $\mathrm{TIME}(f(n)) = \mathrm{TIME}(2^{f(n)})$ (Gap Theorem)

Proper Complexity Functions

- Χρησιμοποιούμε στον ορισμό των κλάσεων πολυπλοκότητας ΜΟΝΟ proper complexity functions $\longrightarrow$ «λογική» ασυμπτωτική συμπεριφορά

  <u>Ορισμός Proper Complexity Function:</u>

  Μια συνάρτηση $f: \mathbb{N}^{+} \rightarrow \mathbb{N}^{+}$ είναι proper complexity function εάν για οποιαδήποτε είσοδο x με οποιοδήποτε μήκος n υπάρχει μια μηχανή Turing M τέτοια ώστε σε χρόνο $O(n+f(n))$ να χρησιμοποιεί χώρο $O(f(n))$ και να κατασκευάζει συμβολοακολουθία μήκους $f(n)$

  Προφανώς θα πρέπει να ισχύει : $f(n+1) \geq f(n)$ for all n

Παραδείγματα proper complexity functions:

$$n^2, n\log n, \log n^2, n^3 + 3n, 2^n, \sqrt{n}, n!$$

Precise Turing Machines

Ορισμός Precise TM

Μια μηχανή Turing M (NTM ή DTM) είναι precise εάν υπάρχουν συναρτήσεις f και g, τέτοιες ώστε για κάθε είσοδο x μήκους n με $n \geq 0$ και για κάθε υπολογισμό της μηχανής (στην περίπτωση NTM):

- η μηχανή να σταματά μετά απο f(n) ακριβώς βήματα
- όλες οι ταινίες της (εκτός απο την ταινία εισόδου και εξόδου) να έχουν στον τερματισμό μήκος ακριβώς g(n)

Σημαντικές Κλάσεις Πολυπλοκότητας

Ανάλογα με το <u>είδος της συνάρτηση πολυπλοκότητας f</u> οι γενικές κλάσεις TIME(f),SPACE(f), NTIME(f) και NSPACE(f) συμβολίζονται συντομογραφικά ως εξής:

<u>Πολυωνυμική:</u>

$$P = TIME(n^k) = \bigcup_{j \succ 0} TIME(n^j)$$

$$PSPACE = SPACE(n^k) = \bigcup_{j \succ 0} SPACE(n^j)$$

$$NP = NTIME(n^k) = \bigcup_{j \succ 0} NTIME(n^j)$$

$$NPSPACE = NSPACE(n^k) = \bigcup_{j \succ 0} NSPACE(n^j)$$

<u>Εκθετική:</u>

$$EXP = TIME(2^{n^k}) = \bigcup_{j \succ 0} TIME(2^{n^j})$$

<u>Λογαριθμική:</u> $NL = NSPACE(\log n)$ $L = SPACE(\log n)$

Συμπληρώματα Κλάσεων Πολυπλοκότητας

Έστω L μια <u>αναδρομική γλώσσα</u>:

- Η L θα αποφασίζεται απο μια μηχανή Turing M
- Το συμπλήρωμα της L , $\overline{L}$ θα αποφασίζεται επίσης απο μια μηχανή Turing $\overline{\text{M}}$

⇓

Θα ανήκουν στην ίδια κλάση πολυπλοκότητας οι δύο συμπληρωματικές γλώσσες?

⇕

Είναι οι κλάσεις πολυπλοκότητας κλειστές ως προς το συμπλήρωμα των γλωσσών τους?

Είναι οι Κλάσεις Πόλυπλοκότητας κλειστές ως προς το συμπλήρωμα των γλωσσών τους?

- Οι ντετερμινιστικές κλάσεις πολυπλοκότητας (χώρου ή χρόνου) **ΕΙΝΑΙ** κλειστές ως προς το συμπλήρωμα

  Κάθε DTM που αποφασίζει μια γλώσσα σε περιορισμένο χώρο ή χρόνο μπορεί να μετατραπεί στην μηχανή που αποφασίζει το συμπλήρωμα της γλώσσας στον ίδιο χώρο ή χρόνο αντίστοιχα, αντιστρέφοντας τους ρόλους των “yes” και “no”

- Για τις μη ντετερμινιστικές κλάσεις πολυπλοκότητας χώρου ισχύει το ίδιο (Immerman –Szelepscenyi Theorem)

- Για τις μη ντετερμινιστικές κλάσεις πολυπλοκότητας χρόνου?

  → ανοικτό πρόβλημα (**P = NP** ⇒ **NP = coNP !**)

Θεωρήμα Ιεραρχίας Κλάσεων Πολυπλοκότητας

- Αφορά στην ιεραρχία μεταξύ κλάσεων πολυπλοκότητας του <u>ιδίου είδους</u> (η μόνη παράμετρος που μεταβάλλεται είναι η συνάρτηση πολυπλοκότητας)
- Προφανώς για δύο κλάσεις πολυπλοκότητας του ιδίου ειδους με συναρτήσει ς πολυπλοκότητας f και g όπου $f(n) \geq g(n)\ n \in \mathbb{N}^+$ η κλάση με συνάρτηση πολυπλοκότητας g θα είναι υποσύνολο της κλάσης με τη συνάρτηση f. Π.χ. $\mathrm{TIME}(g) \subseteq \mathrm{TIME}(f)$

Ποιά σχέση πρέπει να υπάρχει μεταξύ των f και g για να είναι η μια κλάση πολυπλοκότητας γνήσιο υποσύνολο της άλλης?

The Hierarchy Theorem

Το Θεώρημα Ιεραρχίας Χρόνου

- Έστω $f(n) \geq n$ μια proper complexity function και η εξής παραλλαγή του προβλήματος τερματισμού με περιορισμό στον χρόνο:

$$H_f = \{M; x : M \text{ accepts input } x \text{ after at most } f(|x|) \text{ steps}\}$$

- **ΛΗΜΜΑ 1** :Κατασκευάζεται μια μηχανή Turing U_f , η οποία αποφασίζει το πρόβλημα H_f σε χρόνο $O(f(n)^3)$ $\longrightarrow$ $H_f \in TIME((f(n))^3)$

- **ΛΗΜΜΑ 2**: Αποδεικνύεται ότι δεν υπάρχει μηχανή που να αποφασίζει το H_f σε χρόνο $O(f(\lfloor \frac{n}{2} \rfloor))$ $\longrightarrow$ $H_f \notin TIME(f(\lfloor \frac{n}{2} \rfloor))$

Άρα υπάρχει <u>γνήσιος εγκλεισμός</u> στις <u>ντετερμινιστικές κλάσεις πολυπλοκότητας χρόνου</u>

$$H_f \in TIME((f(n))^3)$$

- Κατασκευάζουμε μια μηχανή U_f η οποία :

- προσομοιώνει τη λειτουργία της M μηχανής για την είσοδο x με alarm clock χρόνου $f(|x|)$
- Αποδέχεται την είσοδο $M;x$ ανν η μηχανή M μέσα σε χρόνο $f(|x|)$ αποδεχτεί την είσοδο x

Ο <u>συνολικός υπολογιστικός χρόνος</u> που απαιτείται είναι $O(l_M k_M^2 f^2(|x|))$

l_M είναι το μήκος κωδικοποίησης κάθε συμβόλου και κάθε κατάστασης της μηχανής M και

k_M είναι το πλήθος των ταινιών της μηχανής M

- Ισχύει $l_M k_M^2 = O(n)$ $\Rightarrow$ $O(f^3(|x|))$

$$H_f \notin TIME(f(\lfloor \frac{n}{2} \rfloor))$$

- Έστω ότι υπάρχει TM η οποία να αποφασίζει το πρόβλημα H_f σε χρόνο $f(\lfloor \frac{n}{2} \rfloor) \longrightarrow H_f \in TIME(f(\lfloor \frac{n}{2} \rfloor))$
- Κατασκευάζουμε τη μηχανή D_f έτσι ώστε:

  $if\ M_{H_f}(M;M) = \text{"yes" then "no" else "yes"}$
- Η D_f με είσοδο M τρέχει στον ίδιο χρόνο με την M_{H_f} με είσοδο $(M;M)$

  $$f(\lfloor \frac{2n+1}{2} \rfloor) = f(n),\ \text{where } n = |M|$$

$D_f(D_f) = \text{"yes"}$

$\Rightarrow D_f;D_f \notin H_f$

$\Rightarrow D_f$ does not accept D_f within time $f(|D_f|)$

$\Rightarrow D_f(D_f) = \text{"}no\text{"}$ Άτοπο

The Hierarchy Theorem

- **Θεώρημα Ιεραρχίας Χρόνου**: εάν $f(n) \geq n$ είναι μια proper complexity function, τότε ισχύει

$$\mathrm{TIME}(f(n)) \subset \mathrm{TIME}((f(2n+1))^3)$$

Πόρισμα: $\mathrm{P} \subset \mathrm{EXP}$

Απόδειξη: Κάθε πολυωνυμική συνάρτηση με όριο στο άπειρο γίνεται μικρότερη απο κάθε εκθετική

Απο το Λήμμα 1 όμως $\longrightarrow$ $\mathrm{TIME}(2^n) \subset \mathrm{TIME}((2^{2n+1})^3) \subseteq \mathrm{TIME}(2^{n^2}) \subseteq \mathrm{EXP}$

Άρα $\mathrm{P} \subset \mathrm{EXP}$

- **Θεώρημα Ιεραρχίας Χώρου**: εάν $f(n) \geq n$ είναι μια proper complexity function, τότε ισχύει $\mathrm{SPACE}(f(n)) \subset \mathrm{SPACE}(f(n)\log f(n))$

Πόρισμα: $\mathrm{L} \subset \mathrm{PSPACE}$

- **Time and Space Hierarchy Theorems**:

  <u>ιεραρχία κλάσεων πολυπλοκότητας ιδίου είδους</u>

  ⇩

  Αποδεικνύουν ότι με <u>επαρκώς μεγαλύτερη διαθέσιμη ποσότητα χώρου ή χρόνου οι TM είναι ικανές να αναγνωρίσουν μεγαλύτερα σύνολα γλωσσών</u>

- Περισσότερο <u>**ενδιαφέρουσα η ιεραρχία ανομοίων κλάσεων πολυπλοκότητας**</u> . Τα συμπεράσματα σε αυτή την περιοχή μελέτης είναι πολύ λίγα και τα ανοικτά προβλήματα αρκετά → <u>**P=NP?**</u>

$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$ και $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$

Απόδειξη (trivial) :

Κάθε ντετερμινιστική μηχανή Turing είναι ειδική περίπτωση μη ντετερμινιστικής μηχανής Turing (με μια επιλογή)

<u>Πορίσμα</u>: $L \subseteq NL$ $\quad P \subseteq NP$

$\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$

$L \in \mathrm{NTIME}(f(n)) \longrightarrow$ Υπάρχει μια precise NTM η οποία αποφασίζει τη γλώσσα σε <u>χρόνο f(n)</u> ,δηλαδή «μαντεύει» μια ακολουθία f(n) διαδοχικών μη ντετερμινιστικών επιλογών η οποία οδηγεί σε accepting state (εφόσον η είσοδος ανήκει στη γλώσσα)

<u>Κατασκευάζεται μια DTM η οποία θα αποφασίζει την ίδια γλώσσα σε χώρο f(n)</u> $\longrightarrow$ Η DTM θα πρέπει να ελέγξει όλες τις διαφορετικές ακολουθίες μη ντετερμινιστικών επιλογών της NTM προκειμένου να βρεί μια που να οδηγεί σε accepting state.

<u>Ιδέα</u>: Η DTM :

- αποθηκεύει την f(n)-long ακολουθία επιλογών της NTM (f(n) χώρος)
- προσομοιώνει τη λειτουργία της NTM σε χρόνο f(n) (f(n) χώρος)
- διατηρεί την τρέχουσα ακολουθία μη ντετερμινιστικών επιλογών στη μνήμη και δημιουργεί την επόμενη (O(f(n)) χώρος)

Συνολικά απαιτείται O(f(n)) χώρος

Πόρισμα: $\mathrm{NP} \subseteq \mathrm{PSPACE}$

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{\log n + f(n)})$

$L \in \mathrm{NSPACE}(f(n))$ $\longrightarrow$ Υπάρχει μια NTM k ταινιών η οποία αποφασίζει τη γλώσσα σε χώρο f(n) . H NTM «μαντεύει» μια ακολουθία διαδοχικών μη ντετερμινιστικών επιλογών η οποία οδηγεί σε accepting state (εφόσον η είσοδος ανήκει στη γλώσσα)

Κατασκευάζεται μια DTM η οποία θα αποφασίζει την ίδια γλώσσα σε χρόνο $k^{\log n + f(n)}$ $\longrightarrow$ H DTM θα πρέπει να ελέγξει αν κάποια απο όλες τις δυνατές ακολουθίες μη ντετερμινιστικών επιλογών οδηγεί σε accepting state, ώστε να αποδεχτεί την είσοδο

Configurations of TM…

<u>TM's Configuration</u> : Ένα στιγμιότυπο στον υπολογισμό της TM . Αναπαρίσταται με την (2κ+1)-tuple:

$$(q , w_1 , u_1 , w_2 , u_2 , \dots , w_k u_k)$$

<u>Προσδιορίζεται</u> πλήρως απο :

- Την κατάσταση της μηχανής
- Το περιεχόμενο των ταινιών
- Τη θέση των κεφαλών στις αντίστοιχες ταινίες

Πόσες διαφορετικές configurations μπορεί να λάβει μια TM?

Για μηχανή που αποφασίζει μια γλώσσα: $(q,i,w_2,u_2,\dots,w_{k-1}u_{k-1})$ (2κ-2)-tuple

i=θέση της κεφαλής στην είσοδο

Άρα <u>οι διαφορετικές configurations μιας μηχανής που έχει είσοδο μήκους n θα έχουν πλήθος</u>:

$$|K| * (n+1) * |\Sigma|^{(2\kappa-4)f(n)} \longrightarrow nc_1^{f(n)} = c_1^{\log n + f(n)}$$

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{\log n + f(n)})$

Configuration Graph of M on input x ($G(M,x)$):
Είναι ο κατευθυνόμενος γράφος, ο οποίος έχει ως κόμβους όλες τις δυνατές configurations της M και δύο configurations C1 και C2 έχουν μεταξύ τους μια κατευθυνόμενη ακμή ανν $C_1 \xrightarrow{M} C_2$

Προφανώς η μηχανή M αποδέχεται την είσοδο x ανν υπάρχει μονοπάτι στον configuration graph: $C_0 = (s, 0, \triangleright, \varepsilon, \ldots, \varepsilon) \xrightarrow{M^*} C = (\text{"yes"}, i, \ldots)$

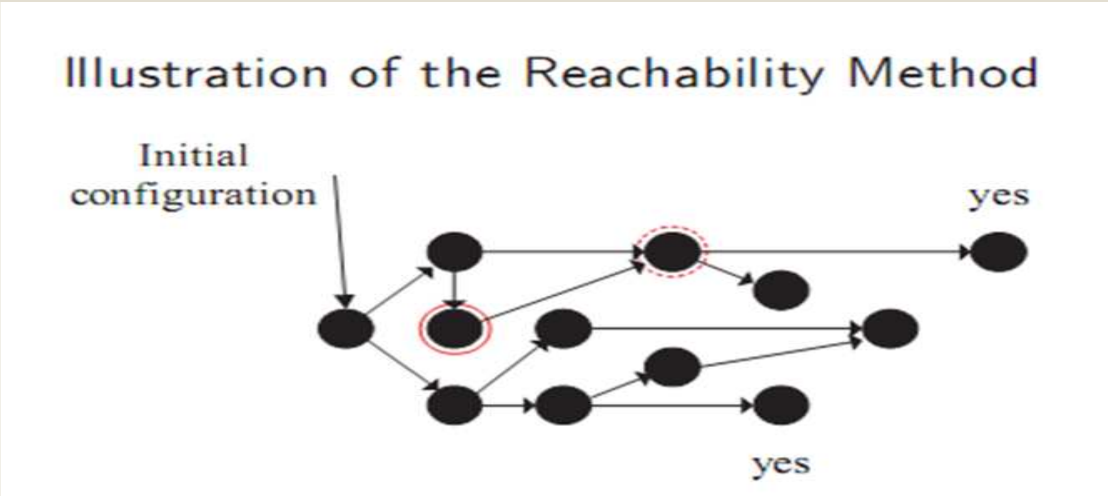
Αλγοριθμική ιδέα: Παράγουμε τον configuration graph της NTM. Το πρόβλημα ευρέσεως απο την DTM μιας ακολουθίας μη ντετερμινιστικών επιλογών της NTM ανάγεται στο πρόβλημα ευρέσεως ενός μονοπατιού στον γράφο που να οδηγεί σε accepting configuration ⟹ **ΜΕΘΟΔΟΣ REACHABILITY**

Η Μέθοδος REACHABILITY

- Προσομοιώνεται η λειτουργία μιας space-bounded NTM με είσοδο χ μέσω της απόφασης απο μια DTM του προβλήματος REACHABILITY στον γράφο των configurations της μηχανής.

Η DTM αποδέχεται ⟶ εύρεση μονοπατιού απο την configuration της εισόδου σε accepting configuration

Η DTM απορρίπτει ⟶ Αδυναμία εύρεσης μονοπατιού που να οδηγεί σε accepting configuration



Το Πρόβλημα REACHABILITY

Το **REACHABILITY** πρόβλημα: **Δεδομένου ενός γράφου** $G=(V,E)$ **και δύο κόμβων** $1, n \in V$ **υπάρχει μονοπάτι απο τον κόμβο 1 στον n?**

<u>Κλασικός Αλγορίθμου Επίλυσης</u>: Διατήρηση στον χώρο της TM ενός συνόλου κόμβων S και χειρισμός του ως ουρά ή στοίβα για την κατάλληλη προσθαφαίρεση κόμβων. Με τον τερματισμό του αλγορίθμου εάν ο κόμβος n έχει βρεθεί έστω και μια φορά στο σύνολο S ο αλγόριθμος αποδέχεται.

<u>Απαιτούμενος υπολογιστικός χρόνος</u>: $O(n^2)$ (Κάθε στοιχείο της adjacency μήτρας ελέγχεται μόνο μια φορά)

Το πρόβλημα REACHABILITY «συλλαμβάνει» πολύ καλά το μη ντετερμινισμό στον χώρο και για τον λόγο αυτό χρησιμοποιείται για την αναγωγή σε αυτό προβλημάτων απόφασης μη ντετερμινιστικών μηχανών με περιορισμό στο χώρο.

$\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{\log n + f(n)})$

- Η DTM αποφασίζει την γλώσσα $L \in \text{NSPACE}(f(n))$ αποφασίζοντας το REACHABILITY πρόβλημα.
- Το REACHABILITY αποφασίζεται σε $O(n^2)$ χρόνο δηλαδή σε $c_2 n^2$ υπολογιστικά βήματα (n =# κόμβων του γράφου)

Άρα για $c_1^{\log n + f(n)}$ κόμβους του configuration graph η προσομοίωση ολοκληρώνεται το πολύ σε $c_2 c_1^{2(\log n + f(n))}$ υπολογιστικά βήματα.

$$\boxed{L \in \text{TIME}(k^{\log n + f(n)})}$$

Η Μεγάλη Αλυσίδα των Εγκλεισμών

Απο τα θεωρήματα που αποδείχτηκαν προκύπτει:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$$

- Η αλυσίδα θα πρέπει να <u>«σπάει» </u>κάπου ανάμεσα στα L και EXP διότι ξέρουμε ότι $L \subset EXP$
- Υπάρχει η υποψία ότι όλοι οι εγκλεισμοί είναι γνήσιοι ⟶ <u>ανοικτό πρόβλημα</u>

Σημαντικά συμπεράσματα απο την εφαρμογή της μεθόδου REACHABILITY

- Ο <u>μη ντετερμινισμός στο χώρο είναι λιγότερο ισχυρός </u>απ' ότι στον χρόνο.

  Ισχύει κάτι παρόμοιο για την περίπτωση του μη ντετερμινισμού στο χρόνο? → περίφημη <u>εικασία P=NP</u>

  (*Ο μη ντετερμινισμός στο χρόνο είναι εκθετικά ισχυρότερος απο τον ντετερμινισμό στο χρόνο)*

- Οι <u>μη ντετερμινιστικές κλάσεις πολυπλοκότητας χώρου είναι κλειστές ως προς το συμπλήρωμα</u>

  Οι μη ντετερμινιστικές κλάσεις πολυπλοκότητας χρόνου είναι πολύ αμφίβολο εάν είναι επίσης κλειστές ως προς το συμπλήρωμα…

Θεώρημα Savitch- $\text{REACHABILITY} \in \text{SPACE}(\log^2 n)$

- **Κλασικός Αλγόριθμος** για το REACHABILITY $\longrightarrow$ $O(n)$ χώρος
- **Αλγόριθμος Savitch** για το REACHABILITY: $\longrightarrow$ $O(\log^2 n)$ χώρος

Ιδέα:

Ορίζουμε το κατηγόρημα **PATH(x,y,i)** να είναι αληθές εάν υπάρχει μονοπάτι στον γράφο απο τον κόμβο x στον y μήκους το πολύ 2^i.

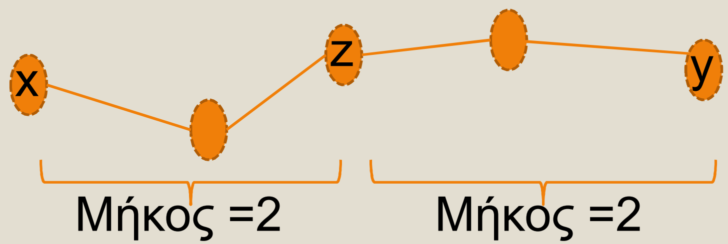
Παρατήρηση: σε ένα γράφο με n κόμβους αν υπάρχει μονοπάτι απο τον χ στον y τότε θα έχει μήκος το πολύ n

Άρα για κάθε δύο κόμβους που συνδέονται στον γράφο το κατηγόρημα **PATH (x,y,logn)** είναι αληθές και αντίστροφα. ($2^{\lceil \log n \rceil} = n$)

Ο αλγόριθμος του Savitch στηρίζεται ακριβώς στο αντίστροφο αυτής της πρότασης $\longrightarrow$ υπολογίζει την αληθοτιμή του κατηγορήματος **PATH (x,y,logn)** και αποφασίζει ανάλογα εάν υπάρχει μονοπάτι απο το χ στο y

Το Savitch Θεώρημα- Ιδέα Αναδρομής

*Ιδέα **Αναδρομής**: Κάθε μονοπάτι μήκους* 2^i *απο το x στο y θα έχει ένα ενδιάμεσο σημείο τέτοιο ώστε τόσο το x όσο και το y να απέχουν το πολύ* απόσταση 2^{i-1} ακμών απο αυτό. Π.χ. για i=2:



*Άρα αν υπάρχει μονοπάτι (x,y,i,) στον γράφο θα πρέπει να υπάρχει **midpoint z** για το οποίο τα κατηγορήματα **PATH(x,z,i-1)** και **PATH(z,y,i-1)** θα είναι αληθή.*

Αναδρομική κλήση του κατηγορήματος PATH: για τον υπολογισμό του PATH(x,y,i) κάνουμε i αναδρομικές κλήσεις του PATH .H i-οστή κλήση του PATH υπολογίζεται κατευθείαν απο τη adjacency matrix διότι έχει το τρίτο όρισμα μηδένικό. Όλες οι αναδρομικές κλήσεις του PATH αποθηκεύονται διαδοχικά σε ένα string της μηχανής → O(logn*logn) χώρος

Θεώρημα Savitch-
Αλγόριθμος για το PATH(x,y,i)


```
if i = 0 then
    if x = y or (x, y) ∈ G then
        return true;
    else
        return false;
    end if
else
    for z = 1, 2, . . . , n do
        if PATH(x, z, i − 1) and PATH(z, y, i − 1) then
            return true;
        end if
    end for
    return false;
end if
```

Θεώρημα Savitch-Αλγόριθμος

Η μηχανή έχει την input string και 2 working strings:

Input string → adjacency matrix

1st working string → (x,y,i) (in binary),(x,y,i-1), … , (x,y,0)

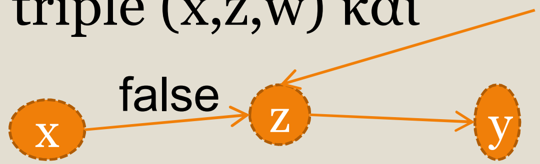
2nd working string → scratch space

- Δημιουργούνται όλα τα πιθανά midpoints z το ένα μετά το άλλο για εξοικονόμηση χώρου
- Κάθε φορά που δημιουργείται ένα καινούργιο z προστίθεται στην 1st working string μια triple (x,z,i-1) και ξεκινά η αναδρομική λύση του προβλήματος της αληθοτιμής αυτής της triple (αναδρομικά προστίθονται και οι υπόλοιπες triples μέχρι και την triple τύπου (x,y,0))
- Η adjacency matrix δίνει απάντηση μόνο για τα κατηγορήματα τύπου PATH(x,y,**0)**

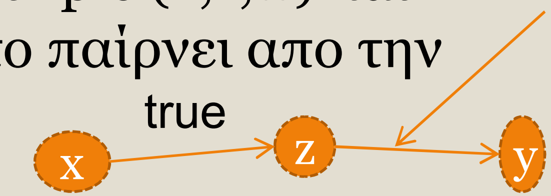
Θεώρημα Savitch-Αλγόριθμος

Αν κατά την αναδρομή βρεθεί:

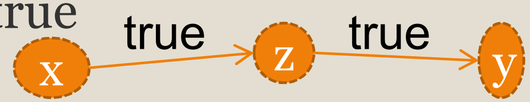
- **PATH(x,z,w)=false** ➡ Διαγράφεται η triple (x,z,w) και προστίθεται μια καινούργια με νέο z



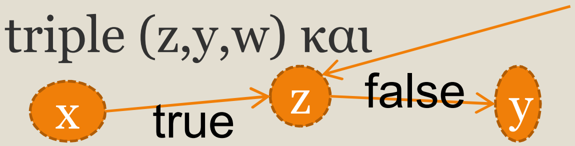
- **PATH(x,z,w)=true** ➡ Διαφάφεται η triple (x,z,w) και προστίθεται η triple (z,y,w) για έλεγχο (το y το παίρνει απο την αμέσως αριστερά triple στο string)



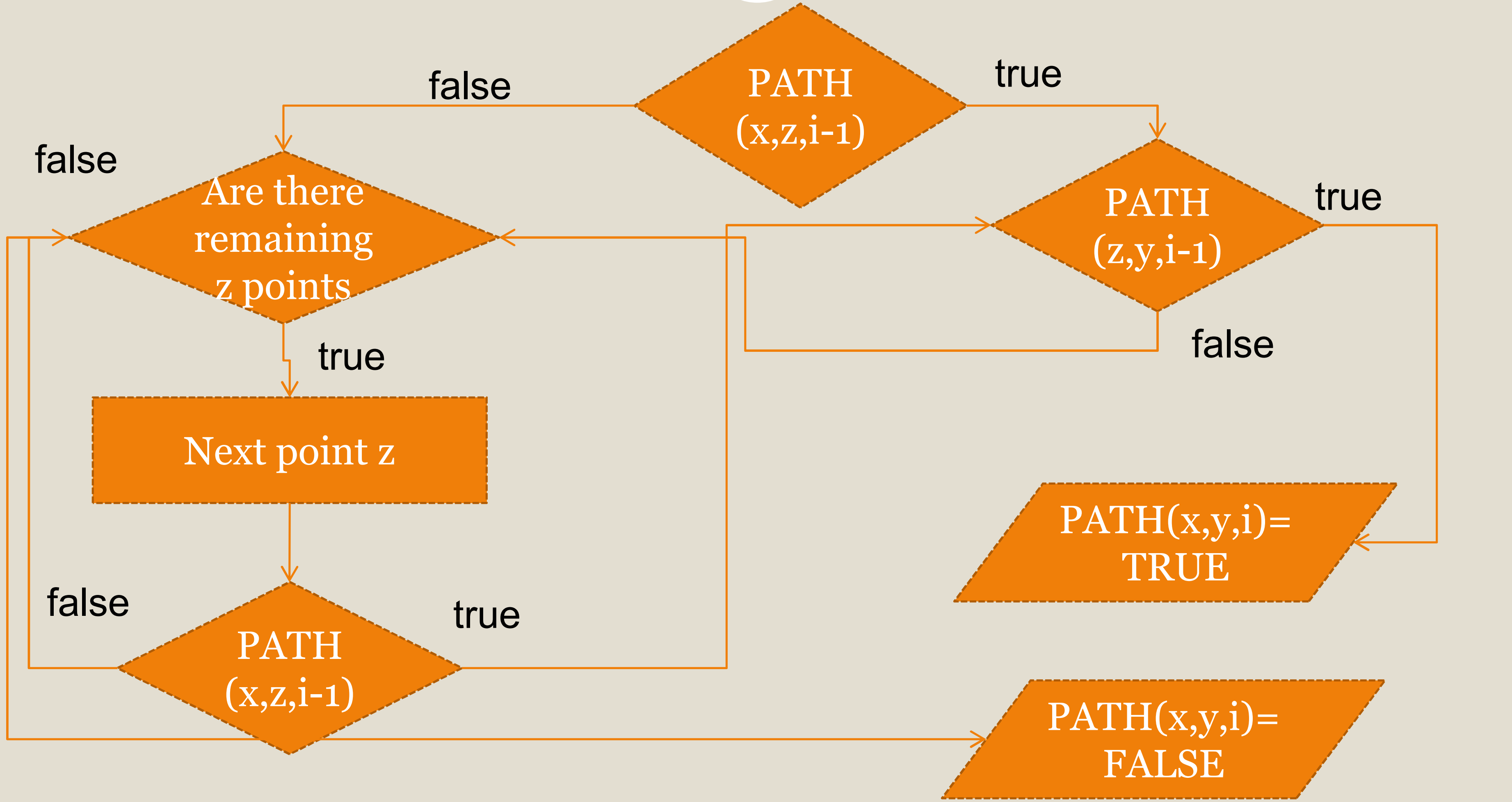
  - **PATH(z,y,w)=true** ➡ Ελέγχεται αν είναι η δεύτερη αναδρομική κλήση για το PATH(x,y,w+1) Διαγράφεται η triple (z,y,w), και αποδίδεται αληθοτιμή στο PATH(x,y,w+1)=true



  - **PATH(z,y,w)=false** ➡ Διαγράφεται η triple (z,y,w) και προστίθεται μια triple (x,z,w) με νέο z



Αλγόριθμος Savitch



Το Savitch Θεώρημα στην προσομοίωση space-bounded NTM

Θ.Δ.Ο. $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$

- Προσομοιώνουμε μια <u>NTM περιορισμένου χώρου</u> για την είσοδο x με μια DTM επίσης περιορισμένου χώρου ως εξής:

  «Τρέχουμε» τον αλγόριθμο του θεωρήματος Savitch στο γράφο των configurations της NTM, δηλαδή επιλύουμε το πρόβλημα REACHABILITY σε χώρο $O(\log^2 n)$.

  Όμως το πλήθος των κόμβων στον configuration graph είναι $c^{f(n)}$

  <u>Άρα επαρκεί χώρος</u> $O(f^2(n))$

  Οι υπόλοιπες εργασίες (διατήρηση στη μνήμη της τρέχουσας διαμόρφωσης και δημιουργία της επόμενης) καταλαμβάνουν χώρο $O(f(n))$

ΠΟΡΙΣΜΑ: PSPACE=NPSPACE

Το Immerman-Szelepscenyi Θεώρημα

« Δεδομένου ενός γράφου G και ενός κόμβου x , ο αριθμός των κόμβων που είναι προσπελάσιμοι απο τον x στον G μπορούν να υπολογιστούν απο μια NTM σε χώρο $\log n$ «

ΠΡΟΕΚΤΑΣΗ ΤΟΥ ΘΕΩΡΗΜΑΤΟΣ: Επιλύεται σε λογαριθμικό χώρο μια σημαντική παραλλαγή του προβλήματος REACHABILITY ⟶ ο υπολογισμός του πλήθους των κόμβων που είναι προσπελάσιμοι απο έναν κόμβο χ

Ο υπολογισμός αυτός συνεπάγεται ταυτόχρονα και τον υπολογισμό του πλήθους των κόμβων που δεν είναι προσπελάσιμοι απο τον χ (συμπληρωματικό προβλημα)

ΠΑΡΑΤΗΡΗΣΗ : Το παραπάνω πρόβλημα καθώς και το συμπληρωματικό του βρίσκονται στην ίδια μη ντετερμινιστική κλάση πολυπλοκότητας χώρου.

Αλγόριθμος Θεωρήματος Immerman-Szelepscenyi

$|S(0)| := 1;$ for k=1,2,...,n-1 do: compute $|S(k)|$ from $|S(k-1)|$

$l := 0;$ for each node $u = 1, 2, \ldots, n$ do: if $u \in S(k)$ then $l := l + 1$

$m := 0;$ reply:=false; for each node υ=1,2,...,n repeat:

if $\upsilon \in S(k-1)$ then $m := m+1;$ if furthermore $G(\upsilon, u)$ then reply:=true

if in the end $m \prec |S(k-1)|$ then "no" (give up), else return reply

$w_0 := x;$ for $p = 1, \ldots, k-1$ do:

guess a node w_p and check that $G(w_{p-1}, w_p)$ (if not, give up)

if $w_{k-1} = \upsilon$ then report $\upsilon \in S(k-1)$, otherwise give up

$\mathrm{NSPACE}(f(n)) = \mathrm{coNSPACE}(f(n))$

Έστω $L \in \mathrm{NSPACE}(f(n))$. Η γλώσσα L θα αποφασίζεται απο μια μη ντετερμινιστική μηχανή M

Κατασκευάζεται μια NTM $\overline{\mathrm{M}}$ που αποφασίζει την $\overline{L}$

Η μηχανή $\overline{\mathrm{M}}$ «τρέχει» με είσοδο x τον αλγόριθμο του θεωρήματος Immerman στον configuration graph της M και αποφαίνεται αν δύο διαμορφώσεις έχουν μονοπάτι μεταξύ τους στη βάση απόφασης της M.

Οι ρόλοι αντιστρέφονται ως εξής: αν σε κάποιο στάδιο ο αλγόριθμος βρεί μια accepting configuration στο S(k) για οποιοδήποτε κ τότε σταματάει και απορρίπτει την είσοδο. Αν δε βρεθεί καμμία accepting configuration στο S(k) για κ=n τότε αποδέχεται.

Ο μη ντετερμινισμός στον χώρο είναι κλειστός ως προς το συμπλήρωμα

Summary