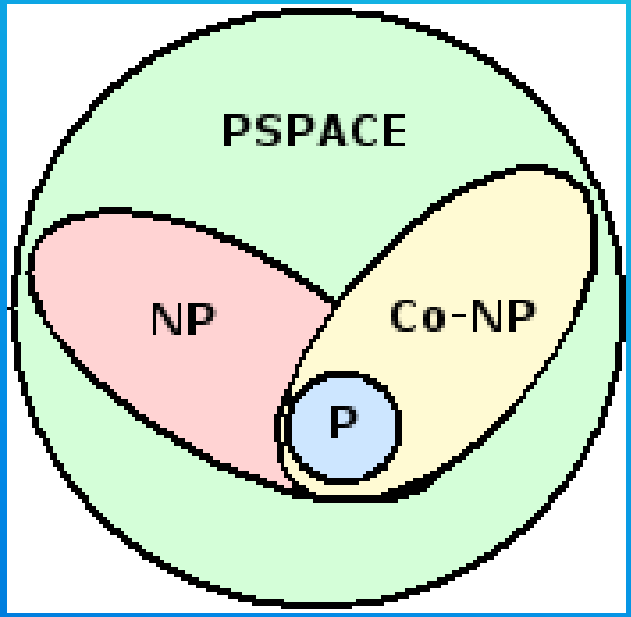
coNP and Function Problems



- **Ένα πρόβλημα απόφασης λέμε ότι επιλύεται σε μη-ντετερμινιστικό πολυωνυμικό χρόνο αν υπάρχει ένας μη-ντετερμινιστικός αλγόριθμος που, εκμεταλλευόμενος μια τυχαία επιλογή, μπορεί σε πολυωνυμικό χρόνο να δώσει καταφατική απάντηση στο πρόβλημα.**

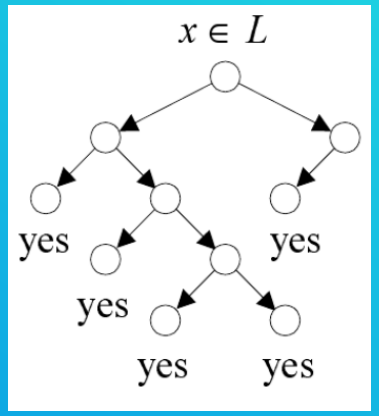
- **Η κλάση αυτών των προβλημάτων συμβολίζεται με NP (Nondeterministic Polynomial time).**

- **Αντίστοιχα το coNP είναι η κλάση των προβλημάτων που ένας αλγόριθμος απαντά καταφατικά στο συμπληρωματικό πρόβλημα.**

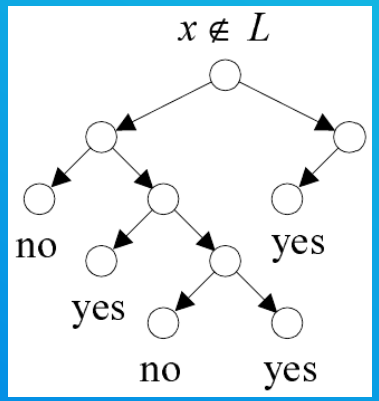
➢ **Δηλαδή η περίπτωση 'no' ενός προβλήματος στο coNP εμπεριέχει μια σύντομη απόδειξη του ότι αποτελεί τέτοια περίπτωση.**
Μόνο οι περιπτώσεις 'no' διαθέτουν τέτοιες αποδείξεις.

➢ **Εάν υποθέσουμε ότι L είναι ένα coNP πρόβλημα.**
Τότε υπάρχει πολυωνυμικού χρόνου μη-ντετερμινιστικός αλγόριθμος M όπου:

- **Αν** $x \in L$ **, τότε M(x) = "yes"**
  για όλα τα υπολογιστικά μονοπάτια.



- **Αν** $x \notin L$ **, τότε M(x) = "no"**
  για κάποια υπολογιστικά μονοπάτια.



- **Καταλήγουμε ότι** $P \subseteq coNP$ **.**

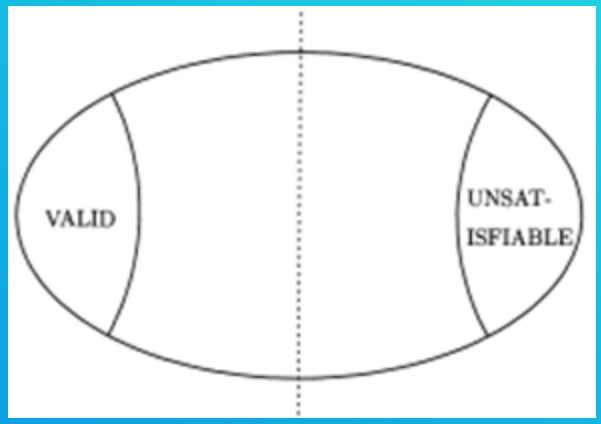
- **Φυσικά δεν ξέρουμε εάν** $P = NP \cap coNP$ **.**

- **Σε αντίθεση με αυτό** $R = RE \cap coRE$ **.**

➢ Κάποια παραδείγματα coNP προβλημάτων:

- **Validity ∈ coNP**
  Η εγκυρότητα (Validity) Boolean εκφράσεων είναι ένα τυπικό πρόβλημα στο coNP.
  Εάν η φ δεν είναι έγκυρη, τότε μπορούμε να βρούμε succinct disqualifications: μια αληθής πρόταση που δεν την ικανοποιεί.

- **SAT complement ∈ coNP.**
  (Validity) ≈ (το ακριβώς αντίθετο του SAT complement)



- **HAMILTONIAN PATH complement ∈ coNP.**
  disqualification = a Hamiltonian path

- **<u>Ένας εναλλακτικός χαρακτηρισμός για το coNP:</u>**

- **Ας υποθέσουμε ότι $L \subseteq \Sigma^*$ είναι μια γλώσσα.**

Τότε L ανήκει στο coNP εαν υπάρχει πολυωνυμικά αποφασίσιμη (polynomially decidable) και polynomially balanced σχέση R ώστε $L = \{x : \underline{\text{για όλα}}$ τα y $(x, y) \in R\}$.

- **$L' = \{x : (x, y) \in \neg R$ <u>για κάποια</u> $y\}$.**

- **Επειδή η $\neg R$ παραμένει polynomially balanced, $L' \in NP$**

- **Άρα $L \in$ coNP εξ' ορισμού.**

- **Πρόταση:**
  Εαν L είναι NP-complete, τότε το συμπλήρωμα του L' = Σ* - L είναι coNP-complete.

- ***Απόδειξη:***
  (➔)Έστω L1' όποιαδήποτε coNP γλώσσα.
  Άρα L1 ∈ NP.
  Έστω R η αναγωγή από την L1 στην L
  Τότε το x ∈ L1 αν και μόνο αν R(x) ∈ L .
  Τότε x ∈ L1' αν και μόνο αν R(x) ∈ L' .
  Οπότε R είναι η αναγωγή από το L1' στο L' .
  (➔) Συμμετρικά αποδεικνύεται.

➢ Κάποια coNP-complete προβλήματα:

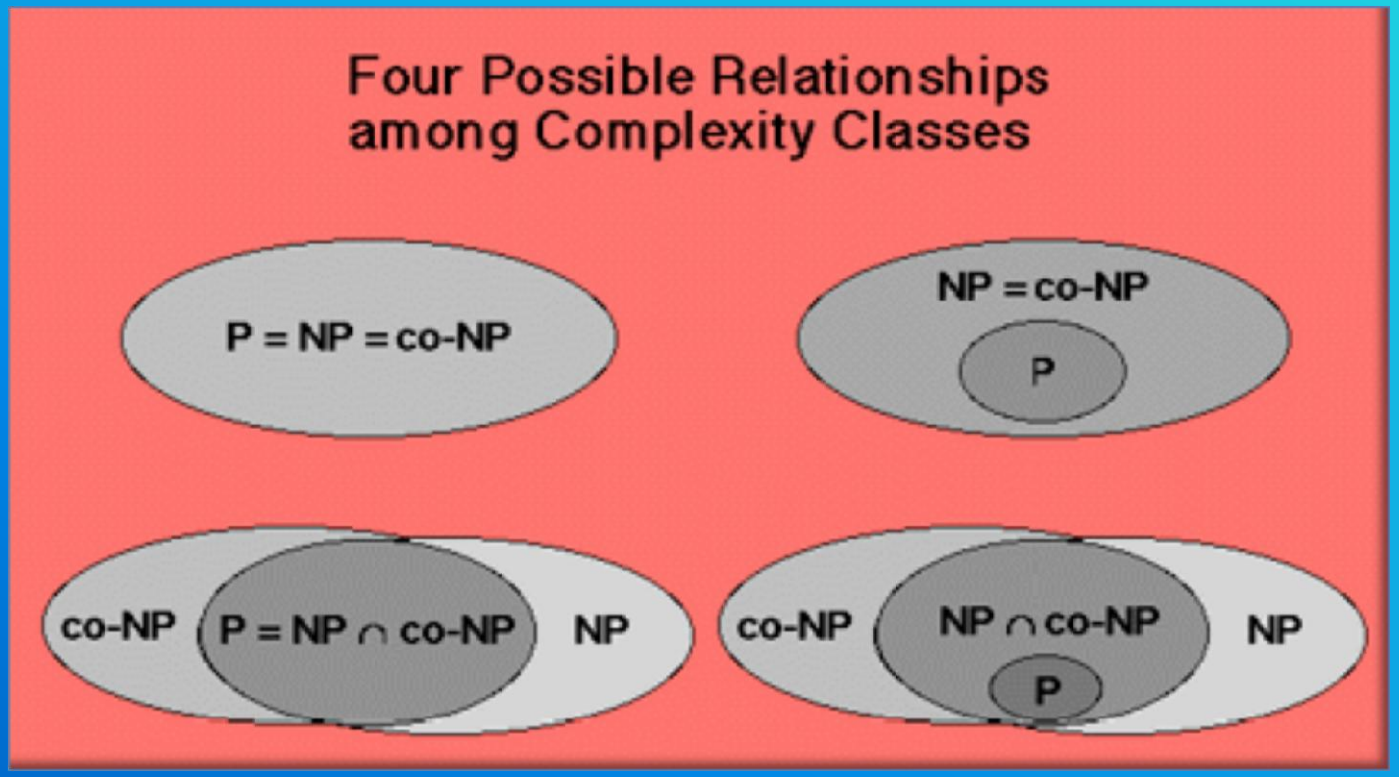
- **SAT complement είναι coNP-complete.**
  SAT complement είναι το συμπλήρωμα του sat.

- **HAMILTONIAN PATH complement είναι coNP-complete.**

- **VALIDITY είναι coNP-complete.**
  Η φ είναι έγκυρη αν και μόνο αν η ¬ φ δεν είναι ικανοποιήσιμη.
  Η αναγωγή από το SAT complement στο VALIDITY είναι πολύ εύκολη.

- **Πιθανές σχέσεις μεταξύ P, NP και coNP:**

  - **P = NP = coNP**
  - **NP = coNP αλλά P ≠ NP**
  - **NP ≠ coNP και P ≠ NP**



- **Πρόταση:**
  Εαν ένα coNP-complete problem είναι στο NP ,
  τότε NP = coNP .

- ***Απόδειξη:***
  Έστω L η coNP-πλήρης γλώσσα και έστω L∈NP.
  Άρα L'∈coNP και άρα L'≤L (άφού η L είναι coNP-πλήρης).
  Από την προηγούμενη πρόταση έχουμε πως η L' είναι NP-πλήρης και άρα L≤L' (αφού L∈NP).
  (➔) (coNP ⊆ NP) Έστω A ∈coNP. Έχουμε A≤L (η L είναι coNP-πλήρης) και L≤L'. Η αναγωγισιμότητα ≤ είναι μεταβατική και άρα A≤L'. Αφού L'∈NP, συμπεραίνουμε πως A ∈NP λόγω κλειστότητας της κλάσης NP ως προς ≤ αναγωγισιμότητα.
  (➔) (NP ⊆ coNP) Έστω B∈NP. Έχουμε B≤L'≤L και με τον ίδιο ακριβώς σκεπτικό συμπεραίνουμε πως B∈coNP .

- **Συμπεράσματα:**

  - **Όμοια εάν ένα NP-hard problem είναι στο coNP, τότε NP = coNP.**

  - **Άρα NP-complete προβλήματα είναι απίθανο να είναι στο coNP και coNP-complete προβλήματα είναι απίθανο να είναι στο NP.**

➢ The Primality Problem:

Ένας ακέραιος (integer) p είναι πρώτος αν p>1 και όλοι οι θετικοί διαιρέτες του είναι ο εαυτός του και η μονάδα.

- **Το πρόβλημα PRIMES , εξετάζει αν ένας ακέραιος N που δίνεται στο δυαδικό σύστημα είναι πρώτος αριθμός.**

- **Το να διαιρέσουμε το N με το 2, 3, ... ,$\sqrt{n}$ δεν είναι αποτελεσματικό.**
  - **το μήκος του N είναι μόνο logN, αλλά $\sqrt{n} = 2^{0.5 \log N}$.**

- **Αλγόριθμος πολυωνυμικού χρόνου για το πρόβλημα primes δεν είχε βρεθεί... μέχρι που το 2002 οι Agrawal, Kayal, και Saxena κατάφεραν να βρουν!**

- **ΔNP:**

ΔNP ≡ NP ∩ coNP είναι η κλάση των προβλημάτων που έχουν succinct certificates και succinct disqualifications.

- **Κάθε “yes” περίπτωση έχει succinct certificate.**
- **Κάθε “no” περίπτωση έχει succinct disqualification.**
- **Σε καμία περίπτωση δεν μπορούμε να έχουμε ταυτόχρονα και “yes” και “no” .**

- **Πρωταρχικές Ρίζες σε Πεπερασμένα Πεδία:** ***(Primitive Roots in Finite Fields)***

ΘΕΩΡΗΜΑ (Lucas and Lehmer (1927)):
Ένας αριθμός p>1 είναι πρώτος αν και μόνο αν υπάρχει αριθμός r, 1<r<p (που ονομάζεται πρωταρχική ρίζα ή γεννήτορας) τέτοιος ώστε:

- $r^{p-1} = 1 \bmod p$, και
- $r^{(p-1)/q} \neq 1 \bmod p$ για όλους τους πρώτους διαιρέτες q του p−1 .

- **<u>Πόρισμα</u> (Pratt's Theorem (1975)):**
  Το PRIMES ανήκει στο NP ∩ coNP.

- ***<u>Απόδειξη:</u>***

Το PRIMES είναι στο coNP γιατί ένα succinct disqualification είναι ένας διαιρέτης του.
Υποθέτουμε ότι p είναι πρώτος.
p's certificate εμπεριέχει το r (από Θεώρημα Lucas and Lehmer).
Χρησιμοποιούμε recursive doubling για να ελέγξουμε αν $r^{p-1} = 1 \bmod p$ σε πολυωνυμικό-χρόνο ως προς το μήκος του input log p .
Χρειαζόμαστε επίσης όλους τους πρώτους διαιρέτες του p-1: $q_1, q_2, \dots, q_k$.
Το να ελέγξουμε αν το $r^{(p-1)/q_i} \neq 1 \bmod p$ είναι επίσης εύκολο.
Ελέγχουμε πολύ εύκολα αν $q_1, q_2, \dots, q_k$ είναι όλοι οι διαιρέτες του p-1 .

(συνέχεια απόδειξης):

Χρειαζόμαστε ακόμη να βρούμε certificates που να δείχνουν ότι τα q_i είναι πρώτοι.

Το πλήρες certificate είναι επαναλαμβανόμενο (recursive) και δενδροειδές (treelike):
$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k))$.

C(p) μπορεί να ελεγθεί σε πολυωνυμικό χρόνο.

Στη συνέχεια αποδεικνύουμε ότι C(p) είναι succinct.

➢ The Succinctness of the Certificate:

• Το μήκος του C(p) είναι το πολύ τετραγωνικό $5 \log^2 p$.

• Αυτός ο ισχυρισμός ισχύει όταν $p = 2$ ή $p = 3$.

• Γενικά, $p - 1$ έχει $k < \log p$ πρώτους διαιρέτες $q_1 = 2, q_2, \ldots, q_k$.

• C(p) απαιτεί: 2 παρενθέσεις και $2k < 2 \log p$ διαχωριστικά (μήκους το πολύ $2 \log p$), r (μήκους το πολύ $\log p$), q1 = 2 και το certificate του 1 (μήκους το πολύ 5 bits), τα qi 's (μήκους το πολύ $2 \log p$), και τα C(qi)s.

• C(p) είναι succinct επειδή

$$|C(p)| \leq \ldots$$
$$\leq 5 \log^2 p$$
Για $p \geq 4$.

➢ Function Problems:

• Προβλήματα Απόφασης (decision problems) είναι yes/no προβλήματα (sat, tsp (d), κ.λ.π).

• Τα function problems απαιτούν μια λύση (για το SAT μια τιμή αληθείας που να το ικανοποιεί, για το tsp μια καλύτερη διαδρομή, κ.λ.π).

• Προβλήματα Βελτιστοποίησης (Optimization Problems) είναι καθαρά function problems.

• Ποιά είναι η σχέση μεταξύ function και decision problems ?

• Ποιό από τα δύο προβλήματα είναι το πιο δύσκολο ?

➢ **Τα Function Problems δεν μπορούν να είναι πιο εύκολα από τα Decision Problems:**

• Αν γνωρίζουμε πως να δημιουργήσουμε μια λύση, μπορούμε να επιλύσουμε το αντίστοιχο decision problem.

• Αν μπορούμε να βρούμε μια τιμή αληθείας που να το ικανοποιεί, τότε το sat είναι στο P.

• Αν μπορούμε να βρούμε την καλύτερη tsp διαδρομή, τότε το tsp (d) είναι στο P.

• ‘Ομως τα decision problems μπορούν να είναι τόσο δύσκολα όσο και τα αντίστοιχα function problems.

➢ FSAT:

- Το FSAT είναι ένα function problem:
  - Έστω $\varphi(x_1, x_2, \ldots, x_n)$ είναι μια Boolean έκφραση.
  - Αν φ είναι ικανοποιήσιμη, τότε επιστρέφει μια τιμή αληθείας.
  - Διαφορετικά, επιστρέφει "no".

- Στη συνέχεια θα δείξουμε ότι αν SAT $\in$ P , τότε το FSAT έχει πολυωνυμικού χρόνου αλγόριθμο.

➢ Αλγόριθμος για το FSAT χρησιμοποιώντας το SAT:

1: t := ε;
2: if φ ∈ SAT then
3: for i = 1, 2, . . . , n do
4: if φ[xi = true] ∈ SAT then
5: t := t ∪ { xi = true };
6: φ := φ[xi = true];
7: else
8: t := t ∪ { xi = false };
9: φ := φ[xi = false];
10: end if
11: end for
12: return t;
13: else
14: return “no”;
15: end if

- **Ανάλυση:**

• Γίνονται ≤n+1 κλήσεις στον αλγόριθμο για το SAT.

• Πιο σύντομες Boolean εκφράσεις από την φ χρησιμοποιούνται σε κάθε κλήση για τον αλγόριθμο του sat.

•Έτσι αν το SAT μπορεί να λυθεί σε πολυωνυμικό χρόνο, να μπορεί και το FSAT.

• Οπότε SAT και FSAT είναι ίσης δυσκολίας (ή ευκολίας).

- **TSP και TSP (d) Revisited:**

• Μας δίνονται n πόλεις 1,2, … , n
και οι ακέραιες αποστάσεις $d_{ij} = d_{ji}$
μεταξύ δύο οποιονδήποτε πόλεων i και j.

• Το TSP ψάχνει να βρει ποιά είναι εκείνη διαδρομή (tour) με την μικρότερη συνολική απόσταση (και όχι απλά πόση είναι η μικρότερη συνολική διαδρομή).

• Η μικρότερη συνολική απόσταση πρέπει να είναι το πολύ $2^{|x|}$, όπου x είναι το input μας.

• Το TSP (d) ψάχνει να βρει εάν υπάρχει διαδρομή με συνολική απόσταση το πολύ B.

• Στη συνέχεια θα δείξουμε ότι εάν TSP (d) $\in$ P τότε το TSP έχει αλγόριθμο πολυωνυμικού χρόνου.

- **Αλγόριθμος για το TSP χρησιμοποιώντας τον TSP (d):**

1. Perform a binary search over interval [0, $2^{|x|}$] by calling tsp (d) to obtain the shortest distance C;
2. For i, j = 1, 2, . . . , n do
3. Call tsp (d) with B = C and $d_{ij} = C + 1$;
4. If “no” then
5. Restore d_{ij} to old value; {Edge [i, j] is critical.}
6. End if
7. End for
8. Return the tour with edges whose $d_{ij} \leq C$;

- **Ανάλυση:**

- Μια ακμή που δεν βρίσκεται σε καμιά βέλτιστη διαδρομή εξαλείφεται, με το $d_{ij} = C+1$.

- Μια ακμή που δεν είναι σε όλες τις υπόλοιπες βέλτιστες διαδρομές θα εξαληφθεί επίσης.

- Έτσι ο αλγόριθμος τελειώνει με n ακμές οι οποίες δεν εξαλείφθηκαν.

- Γίνονται $O(|x|+n^2)$ κλήσεις στον αλγόριθμο του tsp (d).

- Έτσι αν το tsp (d) μπορεί να επιλυθεί σε πολυωνυμικό χρόνο, να μπορεί και το tsp.

- Συμπεραίνουμε λοιπόν ότι το tsp (d) και το tsp είναι ίσης δυσκολίας (ή ευκολίας).

➢ FNP και FP:

• **L ∈ NP εανν υπάρχει πολυωνυμικού χρόνου σχέση $R_L(x , y)$ τέτοια ώστε:**

$$x \in L \iff y \{ |y| \leq p(|x|) \ \& \ R_L(x , y) \}$$

- **Προσέξτε πως η R_L καθορίζει το πρόβλημα-γλώσσα L.**

• **Το αντίστοιχο function πρόβλημα $\Pi_{R(L)} \in$ FNP είναι:**
μας δίνεται ένα x και πρέπει να βρούμε κάποιο y τέτοιο ώστε: να ικανοποιείται η $R_L(x , y)$ ή να επιστρέφει "no" αν δεν υπάρχει κανένα y που να την ικανοποιεί.

- **Είναι όλα τα FNP προβλήματα self-reducible όπως το FSAT ??? [ανοικτό]**

• **FP είναι η υποκλάση (subclass) των FNP όπου έχουμε μόνο προβλήματα για τα οποία υπάρχει πολυωνυμικού χρόνου αλγόριθμος.**

- **FP <?> FNP**

- **Η απόδειξη του Cook δείχνει ότι το FSAT είναι FNP-complete.**

- **Άρα εάν FSAT ∈ FP τότε FNP = FP.**

- **‘Ομως δείξαμε την ιδιότητα αυτοελαχιστοποίησης (self-reducibility) για το SAT, οπότε καταλήγουμε στο εξής Θεώρημα:**

- **ΘΕΩΡΗΜΑ: FP = FNP εαν P=NP .**

- **TFNP**:

- Τι συμβαίνει όταν η σχέση R είναι ολική (total) ;

π.χ. : για κάθε x υπάρχει τουλάχιστον ένα y τέτοιο ώστε R(x, y).

- Ορίζω το TFNP να είναι η υποκλάση (subclass) του FNP όπου η σχέση R να είναι ολική (total).

  - TFNP περιλαμβάνει προβλήματα που πάντα έχουν λύση, π.χ. Παραγοντοποίηση (factoring), θεωρήματα σταθερού σημείου (fix-point theorems), γραφοθεωρητικά προβλήματα (graph-theoretic problems), ...

  - Πώς γνωρίζουμε ότι υπάρχει πάντα λύση;
    Με μια "μη-κατασκευαστική απόδειξη της ύπαρξης" .