CoNP and Function Problems
By definition, coNP is the class of problems whose complement is in NP.

NP is the class of problems that have succinct certificates.

coNP is therefore the class of problems that have succinct disqualifications:

- A “no” instance of a problem in coNP possesses a short proof of its being a “no” instance.
- Only “no” instances have such proofs.
Suppose $L$ is a coNP problem.

There exists a polynomial-time nondeterministic algorithm $M$ such that:

- If $x \in L$, then $M(x) = \text{"yes"}$ for all computation paths.
- If $x \in L$, then $M(x) = \text{"no"}$ for some computation path.
\[ x \in L \]

\[ x \notin L \]
Clearly $P \subseteq \text{coNP}$. 

It is not known if $P = \text{NP} \cap \text{coNP}$. 

- Contrast this with $R = \text{RE} \cap \text{coRE}$.
Some coNP Problems

- **VALIDITY ∈ coNP.**
  - If \( \phi \) is not valid, it can be disqualified very succinctly: a truth assignment that does not satisfy it.

- **SAT complement ∈ coNP.**
  - The disqualification is a truth assignment that satisfies it.

- **HAMILTONIAN PATH complement ∈ coNP.**
  - The disqualification is a Hamiltonian path.
Let $L \subseteq \Sigma^*$ be a language. Then $L \in \text{coNP}$ if and only if there is a polynomially decidable and polynomially balanced relation $R$ such that $L = \{x : \forall y (x, y) \in R\}$.

$L' = \{x : (x, y) \in \neg R \text{ for some } y\}$.

Because $\neg R$ remains polynomially balanced, $L \in \text{NP}$.

Hence $L \in \text{coNP}$ by definition.
L is NP-complete if and only if its complement $L' = \Sigma^* \setminus L$ is coNP-complete.

Proof ($\Rightarrow$; the $\Leftarrow$ part is symmetric)

- Let $L1'$ be any coNP language.
- Hence $L1 \in$ NP.
- Let $R$ be the reduction from $L1$ to $L$.
- So $x \in L1$ if and only if $R(x) \in L$.
- So $x \in L1'$ if and only if $R(x) \in L'$.
- $R$ is a reduction from $L1'$ to $L'$. 
Some coNP-Complete Problems

- SAT complement is coNP-complete.
  - SAT complement is the complement of sat.
- VALIDITY is coNP-complete.
  - $\phi$ is valid if and only if $\neg \phi$ is not satisfiable.
  - The reduction from sat complement to VALIDITY is hence easy.
- HAMILTONIAN PATH complement is coNP-complete.
Possible Relations between P, NP, coNP

1. $P = NP = \text{coNP}$. 
2. $NP = \text{coNP}$ but $P \neq NP$. 
3. $NP \neq \text{coNP}$ and $P \neq NP$.  
   - This is current “consensus.”
coNP Hardness and NP Hardness

- If a coNP-hard problem is in NP, then NP = coNP.
- Let $L \in NP$ be coNP-hard.
- Let PNTM $M$ decide $L$.
- For any $L_1 \in coNP$, there is a reduction $R$ from $L_1$ to $L$.
- $L_1 \in NP$ as it is decided by PNTM $M(R(x))$.
  - Alternatively, NP is closed under complement.
- Hence $coNP \subseteq NP$.
- The other direction $NP \subseteq coNP$ is symmetric.
Similarly, if an NP-hard problem is in coNP, then NP = coNP.

Hence NP-complete problems are unlikely to be in coNP and coNP-complete problems are unlikely to be in NP.
The Primality Problem

- An integer $p$ is prime if $p > 1$ and all positive numbers other than 1 and $p$ itself cannot divide it.
- PRIMES asks if an integer $N$ is a prime number.
- Dividing $N$ by 2, 3, $\ldots$, $\sqrt{N}$ is not efficient.
  - The length of $N$ is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
- A polynomial-time algorithm for primes was not found until 2002 by Agrawal, Kayal, and Saxena!
- We will focus on efficient “probabilistic” algorithms for primes (used in practice).
\( \Delta \text{NP} \equiv \text{NP} \cap \text{coNP} \) is the class of problems that have succinct certificates and succinct disqualifications.

- Each “yes” instance has a succinct certificate.
- Each “no” instance has a succinct disqualification.
- No instances have both.

\( \text{P} \subseteq \Delta \text{NP} \).

We will see that primes \( \in \text{DP} \).

- In fact, primes \( \in \text{P} \) as mentioned earlier.
Theorem (Lucas and Lehmer (1927)) A number $p > 1$ is prime if and only if there is a number $1 < r < p$ (called the primitive root or generator) s.t.

1. $r^{p-1} = 1 \mod p$, and
2. $r^{(p-1)/q} = 1 \mod p$ for all prime divisors $q$ of $p-1$.

Proof excluded.
(Pratt (1975)) PRIMES ∈ NP ∩ coNP.

Primes is in coNP because a succinct disqualification is a divisor.

Suppose p is a prime.

p’s certificate includes the r in L.L. Theorem

Use recursive doubling to check if \( r^{p-1} \equiv 1 \mod p \) in time polynomial in the length of the input, \( \log p \).

We also need all prime divisors of \( p - 1 \): \( q_1, q_2, \ldots, q_k \).

Checking \( r^{(p-1)/q_i} \not\equiv 1 \mod p \) is also easy.
The Proof (concluded)

- Checking \( q_1, q_2, \ldots, q_k \) are all the divisors of \( p - 1 \) is easy.
- We still need certificates for the primality of the \( q_i \)'s.
- The complete certificate is recursive and tree-like: \( C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)) \).
- \( C(p) \) can also be checked in polynomial time.
- We next prove that \( C(p) \) is succinct.
The Succinctness of the Certificate

- The length of $C(p)$ is at most quadratic at $5 \log^2 p$.
- This claim holds when $p = 2$ or $p = 3$.
- In general, $p - 1$ has $k < \log p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
- $C(p)$ requires: 2 parentheses and $2k < 2 \log p$ separators (length at most $2 \log p$ long), $r$ (length at most $\log p$), $q_1 = 2$ and its certificate 1 (length at most 5 bits), the $q_i$’s (length at most $2 \log p$), and the $C(q_i)$s.
C(p) is succinct because
\[ |C(p)| \leq 5 \log p + 5 + 5 \sum_{i=2}^{k} \log^2 q_i \]
\[ \leq 5 \log p + 5 + 5 (\sum_{i=2}^{k} \log 2 q_i)^2 \]
\[ \leq 5 \log p + 5 + 5 \log (p-1)/2 \]
\[ < 5 \log p + 5 + 5 (\log 2 p - 1)^2 \]
\[ = 5 \log^2 p + 10 - 5 \log 2 p \leq 5 \log^2 p \]
for \( p \geq 4 \).
Function Problems

- Decisions problem are yes/no problems (sat, tsp (d), etc.).
- Function problems require a solution (a satisfying truth assignment, a best tsp tour, etc.).
- Optimization problems are clearly function problems.
- What is the relation between function and decision problems?
- Which one is harder?
If we know how to generate a solution, we can solve the corresponding decision problem.

- If you can find a satisfying truth assignment efficiently, then sat is in P.
- If you can find the best tsp tour efficiently, then tsp(d) is in P.

But decision problems can be as hard as the corresponding function problems.
FSAT is this function problem:

- Let $\varphi(x_1, x_2, \ldots, x_n)$ be a boolean expression.
- If $\varphi$ is satisfiable, then return a satisfying truth assignment.
- Otherwise, return “no.”

We next show that if SAT $\in P$, then FSAT has a polynomial-time algorithm.
An Algorithm for FSAT Using 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
Analysis

- There are \( \leq n + 1 \) calls to the algorithm for SAT
- Shorter boolean expressions than \( \varphi \) are used in each call to the algorithm for SAT.
- So if SAT can be solved in polynomial time, so can FSAT.
- Hence SAT and FSAT are equally hard (or easy).
We are given \( n \) cities 1, 2, \ldots, \( n \) and integer distances \( d_{ij} = d_{ji} \) between any two cities \( i \) and \( j \).

The TSP asks for a tour with the shortest total distance (not just the shortest total distance, as earlier).

The shortest total distance must be at most \( 2^{\left| x \right|} \) where \( x \) is the input.

TSP (d) asks if there is a tour with a total distance at most \( B \).

We next show that if TSP (d) \( \in P \), then TSP has a polynomial-time algorithm.
An Algorithm for tsp Using tsp (d)

1: Perform a binary search over interval \([ 0, 2^{\lvert x \rvert} ]\) by calling tsp (d) to obtain the shortest distance \(C\);

2: for \(i, j = 1, 2, \ldots, n\) 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 \(d_{ij} \leq C\);
Analysis

- An edge that is not on any optimal tour will be eliminated, with its dij set to C + 1.
- An edge which is not on all remaining optimal tours will also be eliminated.
- So the algorithm ends with n edges which are not eliminated.
- There are $O(|x| + n^2)$ calls to the algorithm for tsp (d).
- So if tsp (d) can be solved in polynomial time, so can tsp.
- Hence tsp (d) and tsp are equally hard (or easy).
FNP and FP

- $L \in \text{NP}$ iff there exists poly-time computable $R_L(x,y)$ s.t.
  $$X \in L \iff \exists y \{ |y| \leq p(|x|) & R_L(x,y) \}$$
  - Note how $R_L$ defines the problem-language $L$

- The corresponding search problem $\Pi_{R(L)} \in \text{FNP}$ is: given an $x$ find any $y$ s.t. $R_L(x,y)$ and reply “no” if none exist
  - Are all FNP problems self-reducible like FSAT? [open?]

- FP is the subclass of FNP where we only consider problems for which a poly-time algorithm is known
A proof a-la-Cook shows that FSAT is FNP-complete.

Hence, if FSAT $\in$ FP then FNP = FP.

But we showed self-reducibility for SAT, so the theorem follows:

**Theorem:** FP = FNP iff P=NP
What happens if the relation R is total? i.e., for each x there is at least one y s.t. R(x,y)

Define TFNP to be the subclass of FNP where the relation R is total

- TFNP contains problems that always have a solution, e.g. factoring, fix-point theorems, graph-theoretic problems, ...
- How do we know a solution exists? By an "inefficient proof of existence", i.e. non-(efficiently)-constructive proof
- The idea is to categorize the problems in TFNP based on the type of inefficient argument that guarantees their solution
Properties of TFNP

1. \( \text{FP} \subseteq \text{TFNP} \subseteq \text{FNP} \)

2. \( \text{TFNP} = \text{F}(\text{NP} \cap \text{coNP}) \)
   - \( \text{NP} \) = problems with “yes” certificate \( y \) s.t. \( R_1(x,y) \)
   - \( \text{coNP} \) = problems with “no” certificate \( z \) s.t. \( R_2(x,y) \)
   - for TFNP \( \text{F}(\text{NP} \cap \text{coNP}) \) take \( R = R_1 \cup R_2 \)
   - for \( \text{F}(\text{NP} \cap \text{coNP}) \) TFNP take \( R_1 = R \) and \( R_2 = \emptyset \)

3. There is an FNP-complete problem in TFNP iff \( \text{NP} = \text{coNP} \)
   - \( \Rightarrow \): If \( \text{NP} = \text{coNP} \) then trivially \( \text{FNP} = \text{TFNP} \)
   - \( \Leftarrow \): If the FNP-complete problem \( \Pi_R \) is in TFNP then:
     - \( \text{FSAT} \) reduces to \( \Pi_R \) via \( f \) and \( g \), hence any unsatisfiable formula \( \varphi \) has a “no” certificate \( y \), s.t. \( R(f(\varphi),y) \) (\( y \) exists since \( \Pi_R \) is in TFNP) and \( g(y) = \text{“no”} \)

4. TFNP is a semantic complexity class \( \Rightarrow \) no complete problems!
   - note how telling whether a relation is total is undecidable (and not even RE!!)
ANOTHER HC is in TFNP

- **Thm:** any graph with odd degrees has an even number of HC through edge xy

- **Proof Idea:**
  - take a HC
  - remove edge (1,2) & take a HP
  - fix endpoint 1 and start "rotating" from the other end
  - each HP has two "valid" neighbors (d=3) except for those paths with endpoints 1,2