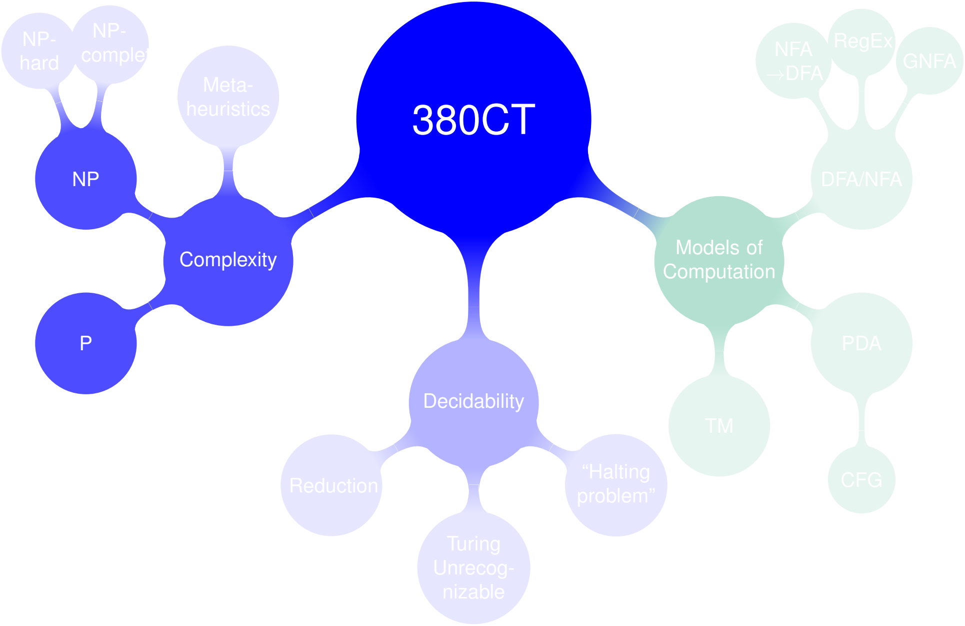
Complexity

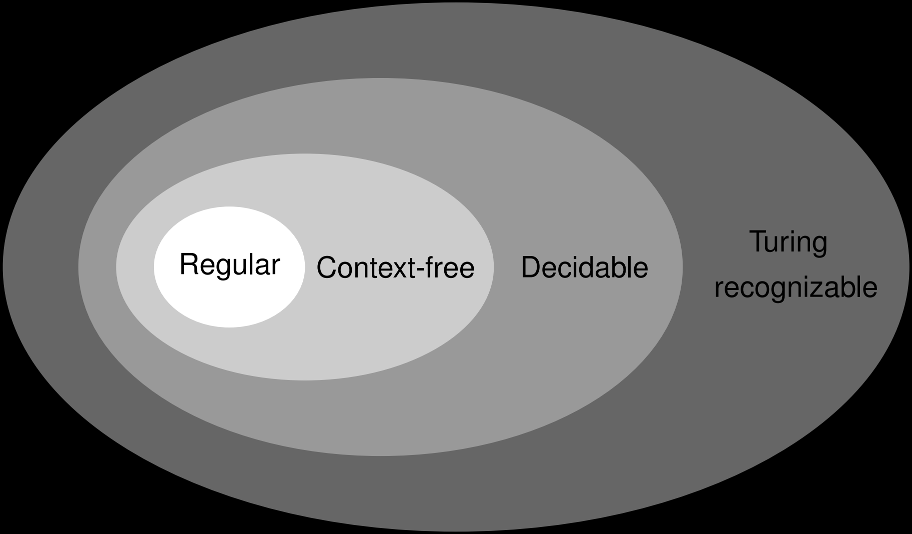
Dr Kamal Bentahar

School of Computing, Electronics and Mathematics
Coventry University

Week 9 – 19/3/2019


380CT
Complexity
NP
P
NP-hard
NP-complet
Meta-heuristics
Decidability
Reduction
Turing Unrecog-nizable
“Halting problem”
Models of Computation
DFA/NFA
NFA →DFA
RegEx
GNFA
PDA
CFG
TM

The “computation universe” discovered so far…



Time Complexity

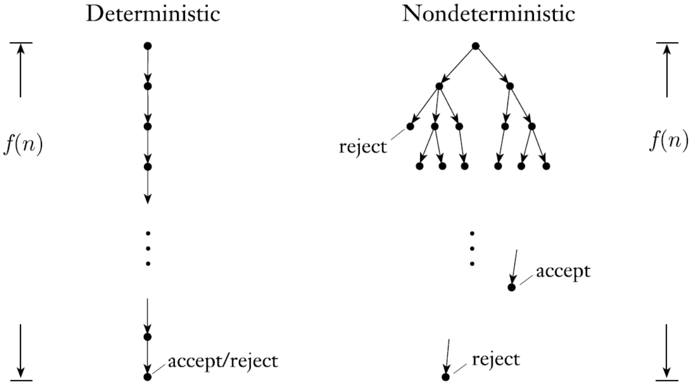
- Being **decidable** means that an **algorithm** exists to decide the problem.
- However, the algorithm may still be *practically* ineffective because of its **time** and/or **space** cost.

Running time / Time complexity

The **running time** or **time complexity** of a TM that always halts is the maximum number of steps $f(n)$ that it makes on any input of length n.

For nondeterministic TMs consider **all the branches** of its computation.

We say that it runs in time $f(n)$; and that it is an $f(n)$-time TM.



Reminder – Big-O notation cheat-sheet

- Constant $O(1)$
- Polynomial $O(n), O(n^2), \ldots, O(n^k), \ldots$ $(k \geq 1)$
- Exponential $O(2^n), O(3^n), \ldots, O(b^n), \ldots$ $(b > 1)$
- Factorial $O(n!)$
- $O(n^n)$

Formal definition of big-O notation

$f(n) = O(g(n))$ means that, when n becomes “sufficiently large,” $f(n)$ becomes bounded by a multiple of $g(n)$.

Big-O notation

Let f and g be functions

$$f, g \colon \mathbb{N} \to \mathbb{R}^+$$

Say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$,

$$f(n) \leq c\, g(n).$$

We say that $g(n)$ is an **(asymptotic) upper bound** for $f(n)$.

Time complexity class

Interesting result:

Complexity relationships among TM variants

Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time multi-tape TM has an equivalent $O(t^2(n))$ time single-tape TM.

This suggests making the following definition:

Time complexity class

Let $t\colon \mathbb{N} \to \mathbb{R}^+$ be a function.
Define the **time complexity class**

$$TIME(t(n))$$

to be the collection of all languages that are decidable by an $O(t(n))$ time TM.

The class **P**

The class **P**

P is the class of languages that are decidable in polynomial time on a deterministic TM.

$$\mathbf{P} = \mathit{TIME}(1) \cup \mathit{TIME}(n) \cup \mathit{TIME}(n^2) \cup \mathit{TIME}(n^3) \cup \cdots.$$

P examples – path between two vertices in a graph

$$PATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph that has a directed path from } s \text{ to } t\}.$$

A polynomial time algorithm for PATH

Input: $\langle G, s, t\rangle$, where G is a directed graph with vertices s and t.
Output: *true* if there is a path between s and t; *false* otherwise.

1: Place a mark on vertex s.
2: **repeat**
3: Scan all the edges of G.
4: If an edge (a, b) is found going from a marked vertex a to an unmarked vertex b, then mark vertex b.
5: **until** no additional vertices are marked
6: **If** t is marked **then** *accept* **else** *reject*

P examples – relatively prime numbers

$$\textit{RELPRIME} = \{\langle x, y\rangle \mid x \text{ and } y \text{ are relatively prime}\}.$$

E : *Euclidean algorithm* for computing the *greatest common divisor*

Input: $\langle x, y\rangle$, where x and y are natural numbers.
Output: $\gcd(x, y)$

```
repeat
    x ← x mod y
    Exchange x and y.
until y = 0
return x.
```

Algorithm that solves RELPRIME, using **E** as a **subroutine**

On input $\langle x, y\rangle$, where x and y are natural numbers:

1. Run E on $\langle x, y\rangle$.
2. If the result is 1, *accept*. Otherwise, *reject*."

The class **NP** – Solution vs Verification

- **Solving**: finding/searching for a solution.
- **Verifying**: confirming that a proposed solution is correct.

Example

Given a candidate for a tour around England, we just need to check if it:

1. contains all the required cities
2. uses no city more than once
3. finishes at its starting point
4. uses only valid routes

The class **NP** – verifiers

Verifiers

A verifier for a language L is an algorithm V, where

$$L = \{w \mid \ V \text{ accepts } \langle w, c\rangle \text{ for some } \textit{certificate} \text{ string } c\}.$$

- A **polynomial time verifier** runs in polynomial time in the length of w.
- A language is **polynomially verifiable** if it has a polynomial time verifier.

The class **NP**

Nondeterministic Polynomial time complexity class

$$NTIME(t(n)) = \{\text{Language decided by an } O(t(n)) \text{ time nondeterministic TM}\}.$$

The class **NP**

NP is the class of languages that have polynomial time verifiers.

Equivalently:

$$\mathbf{NP} = NTIME(1) \cup NTIME(n) \cup NTIME(n^2) \cup NTIME(n^3) \cup \cdots .$$

NP examples – the subset sum problem

$$\mathit{SUBSETSUM} = \{\langle S, t\rangle \mid S = \{x_1, \dots, x_k\}, \text{ and for some } \{y_1, \dots, y_\ell\} \subseteq \{x_1, \dots, x_k\} \colon y_1 + \cdots + y_\ell = t\}.$$

Verifier: "On input $\langle\langle S, t\rangle, c\rangle$:

1. Test whether c is a collection of numbers that sum to t.
2. Test whether S contains all the numbers in c.
3. If both pass, *accept*. Otherwise, *reject*."

Alternatively, polynomial time NDTM: "On input $\langle S, t\rangle$:

1. Non-deterministically select a subset c of the numbers in S.
2. Test whether c is a collection of numbers that sum to t.
3. If the test passes, *accept*. Otherwise, *reject*."

NP examples – cliques in a graph

Cliques

A **clique** in an undirected graph is a subgraph, wherein every two vertices are connected by an edge.
A k**-clique** is a clique that contains k vertices.

$$CLIQUE = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}.$$

Verifier for $CLIQUE$: "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a subgraph with k vertices in G.
2. Test whether G contains all edges connecting vertices in c.
3. If both pass, *accept*. Otherwise, *reject*."

Alternatively, polynomial time NDTM: "On input $\langle G, k\rangle$, where G is a graph:

1. Nondeterministically select a subset c of k vertices of G.
2. Test whether G contains all edges connecting vertices in c.
3. If yes, accept; otherwise, reject."

Summary

P Polynomial-time

Class of languages that are decidable in polynomial time.

$$\mathbf{P} = \bigcup_{k \geq 0} TIME(n^k).$$

NP Nondeterministic Polynomial time

Class of languages that have polynomial time verifiers.

$$\mathbf{NP} = \bigcup_{k \geq 0} NTIME(n^k).$$

P = NP?