Algos: P vs NP, Part 2

Recap:

- P: polynomial time
- NP: non-deterministic polynomial time
- NP-Hard: exponential time decision problems
  NP-Complete: you'll find out

COMP1202: Formal Language Frameworks

- language L over Σ = any set of strings, which are groupings of symbols over alphabet Σ

  empty string: ε
  empty language: $\emptyset$

  Σ^*: language of all words over Σ

- OPs: $\cap$, $\cup$, $L^+ = \Sigma - L$, decision problems

Decision Problems using FL

- Output Q characterised by everything which gives a 1

  $\therefore$ Q is language L over $\Sigma = \{0,1\} \mid L = \{x \in \Sigma^* : Q(x) = 1\}$

  (e.g.) graph PATH defined like so:

  $$PATH = \{\langle G, u, v, k\rangle : G = (V, E) \text{ is undirected graph}, (u, v) \in V, (k \geq 0) \in \mathbb{Z}, \exists \text{ path } uv^0 \in G \mid \text{edges(path)} \leq k\}$$

NP Completeness

- Definition 2: polynomial-time reducible (PTR)
  - Languages: L_1, L_2
  - L_1 is PTR to L_2 ($L_1 \leq_p L_2$) iff
    - $\exists$ (PT computable function $F: \{0,1\}^* \to \{0,1\}^*$) | $\forall (x \in \{0,1\}^*).\ (x \in L_1 \underset{\text{iff}}{\Longleftrightarrow} F(x) \in L_2)$
- language $L \subseteq \{0,1\}^*$ is NP-complete if
  1) L is NP (i.e. L is verifiable in polynomial time)
  2) $\forall L' \in NP.\ L' \leq_p L$ (all other NP languages L' are PTR to L)
- Language L is NP-hard if property 2 met BUT L isn't verifiable in polynomial time

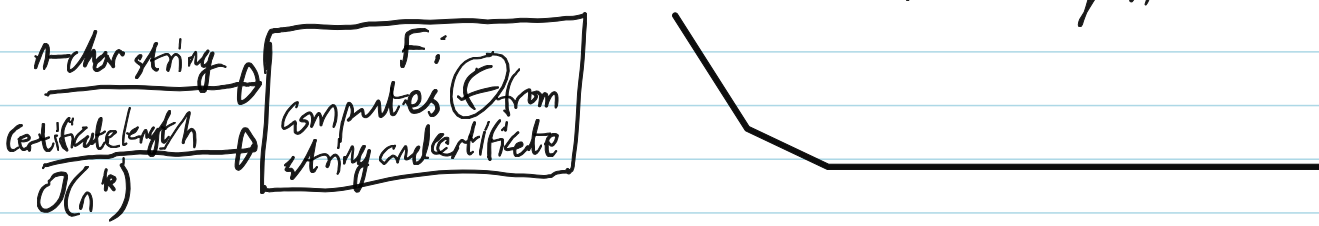
Circuit Satisfiability

- Boolean circuit is SAT if $\exists$ combo | output = 1
- Size: number of Boolean combinational elements + number of wires
  - ☺ k inputs $\longrightarrow$ 2^k assignments to be checked
  - ☹ when size(C) is polynomial in k, checking combo takes $\Omega(2^k)$
- Proof: **circuit SAT is NP**
  - whenever satisfiable circuit is input to algorithm (A), $\exists$ certificate | length = polynomial to size of C
    - ☺ A should output 1

(!) when I set circuit is input, no certificate might trick A to think "circuit is satisfiable"
↳ but it's not ☹

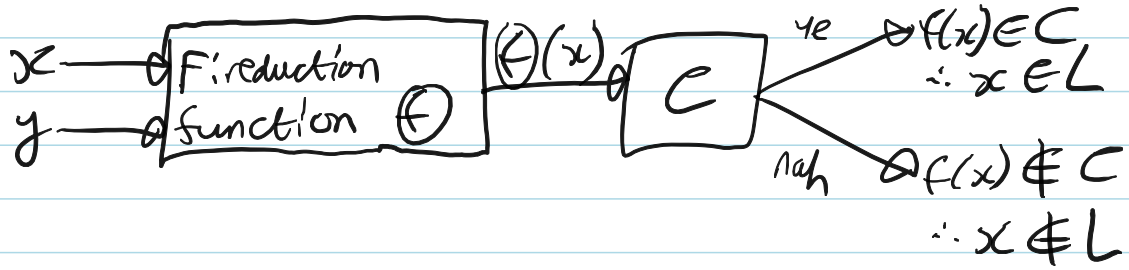
- How to prove CSAT is NP-hard?
  - if L is any NP-language, $\forall L \leq_p \textcircled{f}(x) \mid x$ is a string $\in L$
    - (i-e) $\exists$ PT algorithm $\textcircled{f}$ (reduction function) mapping ALL binary strings $x \in L$ to circuits $C = \textcircled{f}(x)$
    - (e) $\forall x.\ (x \in L \longleftrightarrow C$ is SAT)
      $\exists \textcircled{f}(x) \mid \textcircled{f}(x) = 1$
  - ... how is $\textcircled{f}$ calculated? (Assumptions)
    - $\exists$ another algo F which takes takes 2-input algo A to determine $\textcircled{f}$
      - (*) let $T(n)$ = worst case runtime of A on input strings with length (n)
        let $k \in \mathbb{Z} \mid k \geq 1$ && $T(n) = O(n^k)$
        ↳ certificate length

$A \wedge (B \vee C)$
$= (A \wedge B) \vee (A \wedge C)$
$\therefore A \vee (B \wedge C)$
$= (A \vee B) \wedge (A \vee C)$



- Computer Architecture: combo circuit, program counter, auxiliary state, working storage

- Back to proving CSAT was NP-hard:
  1) does algo F CORRECTLY compute a reduction function Φ?
     - i.e, does $\exists$ certificate $y \mid A(x,y) = 1$?] - is C satisfiable?



  2) does (f) run in polynomial time?
     - let (n) = length(x) $\mid x \in L$
     - number of bits needed to represent a configuration is polynomial in (n)
     - A has CONSTANT size, independent of length(x)

(☺ independent of (n) (since (n) = length(x)))
(☹ length of certificate y is $O(n^k)$)

     - worst case runtime for $A = O(n^k)$ steps
       ☹ amount of memory is also polynomial in (n)

- How to prove something is NP-complete: (e.g language L)
  1) Prove L is NP (ie verifiable in polynomial time)
  2) Choose definite NP-complete language L'
  3) Show algorithm which makes all $x \in \{0,1\}^*$ in L' PTR to an $\Phi(x)$ in L, via a reduction function Φ
  4) Prove Φ satisfies $(x \in L') \longleftrightarrow (\Phi(x) \in L)$, $\forall L'$ (i.e $\forall x \in \{0,1\}^*$)
  5) Prove (f) runs in POLYNOMIAL TIME