CS F364
Design & Analysis of Algorithms

# ALGORITHMS - COMPLEXITY

## Complexity Classes

- NP-Completeness Via Reductions
  - Examples: kSAT

## PROBLEMS: kSAT

- kSAT
  - Satisifiability problem, where input instances are in CNF with exactly k distinct literals in clause.
- 2SAT is kSAT for k=2
- 2SAT can be solved in polynomial time.

# 2SAT IS IN $\mathbb{P}$

- Exercise: Reduce 2SAT to a problem on directed graphs that is efficiently solvable:
  1. Note that L1 | L2 is equivalent to !L1 --> L2.
  2. Given a formula F in 2CNF, construct a directed graph G:
     1. add vertices labeled x and !x for each variable x occurring in the formula.
     2. add an edge from L1 to L2 if L1 --> L2 is a clause in F
  3. Argue that:

     *there is a path in G from x to !x for some variable x*

     **iff**

     *F is not satisfiable*.

# PROBLEMS: 2SAT VS. HORN-SAT

- 2SAT can be solved in polynomial time.
- Exercise:
  1. Can you reduce 2SAT to HORN-SAT?
- HORN-SAT is *satisfiability of Horn formulas*.
  - A Horn formula is a conjunction of Horn clauses and
  - a Horn clause is of the form x --> y (i.e. !x | y)

[Note that HORN-SAT is solvable in polynomial time.

Refer to *Huth & Ryan: Logic in CS* for an algorithm for HORN-SAT.]

## PROBLEM: 3SAT

- 3CNF-SAT (*which is commonly referred to as* 3SAT):
  - Given a Boolean expression in CNF with *exactly 3 distinct literals* in each clause,
  - find whether there is an input assignment such that the expression is satisfied.
- Solving 3SAT:
  - There is no known polynomial time algorithm to solve 3SAT.

# 3SAT IS $\mathbb{NP}$-COMPLETE

- 3SAT is $\mathbb{NP}$-complete
  - 3SAT is in $\mathbb{NP}$
    - Proof: Trivial
  - 3SAT is $\mathbb{NP}$-hard
    - Proof: CNF-SAT $\precsim$ 3SAT
    - Reduction:
      - Map each clause in the CNF expression to one or more 3-literal clauses.
      - (*see next slide for the mapping*)

# CNF-SAT $\precsim$ 3SAT

**Mapping M:** Let the input formula F be the conjunction $AND_i\ C_i$

Then for each i:

1. case **$C_i$ is L** : Replace $C_i$ with

   $(L \mid p \mid q)$ & $(L \mid !p \mid q)$ & $(L \mid p \mid !q)$ & $(L \mid ! \mid !q)$

2. case **$C_i$ is $L_1 \mid L_2$** : Replace $C_i$ with

   $(L_1 \mid L_2 \mid p)$ & $(L_1 \mid L_2 \mid !p)$

3. case **$C_i$ is $L_1|L_2|L_3$** : Replace $C_i$ with $C_i$
4. case **$C_i$ is $L_1|L_2|L_3|L_4$ :** Replace $C_i$ with

   $(p \mid L_3 \mid L_4)$ & $(!p \mid L_1 \mid L_2)$ & $(p \mid !L_1)$ & $(p \mid !L_2)$

   and recursively handle the last two clauses.

1. case **$C_i$ is $L_1 \mid \ldots \mid L_k$** (k>4): Replace $C_i$ with

   $(p \mid L_3 \mid L_4 \mid \ldots \mid L_k)$ & $(!p \mid L_1 \mid L_2)$ & $(p \mid !L_1)$ & $(p \mid !L_2)$

   and recursively handle the first clause and the last two clauses.

p and q are new variables introduced.

- Note:
  - The last three clauses in the replaced expression for the fourth case correspond to $(p \longleftrightarrow (L_1 \mid L_2))$.

# CNF-SAT $\precsim$ 3SAT [2]

- Claim (about the mapping M described in the previous slide):
  - M is a polynomial time algorithm that
    - given an input F outputs F' in 3CNF such that
      - F' is satisfiable iff F is satisfiable and
      - the size of F' is a polynomial function of the size of F.
- Exercises:
  - Implement the algorithm M.
  - Derive the time complexity of M.
  - Estimate the size of the output of M as a function of the size of its input.

# kSAT is $\mathbb{NP}$-complete

- Exercise:
  - Prove that kSAT is $\mathbb{NP}$-complete