innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Satisfiability

Satisfiability (SAT)

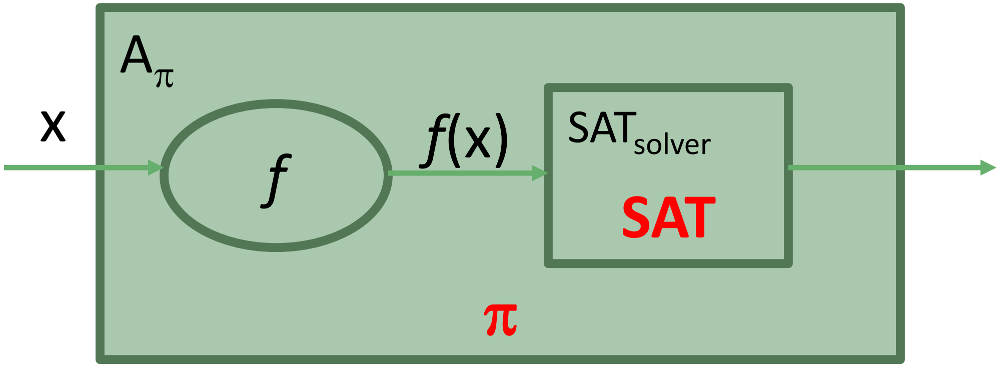
- The satisfiability problem (SAT) is *not known to have a polynomial time algorithm*
  - Satisfiability of formulas in CNF is also known to be equally difficult
    - i.e. there is no known polynomial time algorithm for finding whether a formula in CNF is satisfiable.
- Reconcile this with:
  - there is a polynomial time algorithm for finding validity of formulas in CNF
    - [**Hint**: *What is the time taken to convert a propositional logic formula to CNF*? **End of Hint**.]

Complexity of SAT

- Recall that:
  - SAT is in NP
  - but is not known to be in P.
- Furthermore SAT is known to be among *the most difficult problems in **NP***
  - finding an efficient i.e. polynomial-time algorithm for SAT would result in
    - efficient solutions for a host of several thousands of such difficult problems in **NP**
      - for none of which we have polynomial time algorithms.

SAT is NP-complete

- SAT is at least as difficult as any other problem in **NP**
  - i.e. *any problem in* **NP** *can be reduced to* SAT in polynomial-time
  - i.e. for any problem π in **NP**, there is a polynomial-time mapping function *f* such that:
    - π(x) returns TRUE if and only if SAT(*f*(x)) returns TRUE



CNF-SAT and variations

- The satisfiability problem for CNF formulas is referred to as CNF-SAT:
  - There is no known polynomial time algorithm for CNF-SAT.
- What if the form is further restricted to k-CNF?
  - i.e. a formula is a conjunction of clauses
    - where a clause is a disjunction of (exactly) k literals
- Satisfiability for k-CNF is referred to as k-SAT.
  - k-SAT is as difficult as CNF-SAT for k>=3
    - i.e. 3-SAT is NP-complete.
    - and k-SAT is NP-complete for all k>=3

2-SAT

- 2-SAT is k-SAT for k=2
- There is a polynomial time algorithm for 2-SAT:
  - Look at every clause in a given formula as an implication:
    - $L_1 \vee L_2$ as either $\neg L_1 \dashrightarrow L_2$ or as $\neg L_2 \dashrightarrow L_1$
  - Apply transitivity:
    - L_1 --> L_2 and L_2 --> L_3 would result in L_1 --> L_3 as well.
  - If, by repeated application of transitivity, you end up with $x_i \vee x_i$ as well as $\neg x_i \vee \neg x_i$.
    - Then we have a contradiction i.e.
      - *the formula is not satisfiable.*

Summary

- SAT is NP-complete
  - DNF-SAT is in P
- CNF-SAT is NP-complete
  - k-SAT is NP-complete for k>2
  - 2-SAT is in P
- Horn-SAT is in P

- Implications (for formulas in in propositional logic):
  - Time for conversion of formulas to DNF ?
  - Time for conversion of CNF formulas to DNF ?
  - Which formulas can be expressed in Horn form?
  - Which formulas can be expressed in 2-CNF?

SAT Solvers

- Efficiency issues
  - Restricted versions
  - Use of Heuristics – may lead to faster than exponential algorithms
- SAT competition

http://www.satcompetition.org/