590N: Logical Methods in Information Security Lecture 9

Ninghui Li

February 4, 2008

- Propositional Logic: Semantics and Normal Forms
- The Propositional SAT Problem

Semantic Equivalence

- Definition (Semantic entailment)

  If, for all valuations in which all $\phi_1, \phi_2, \cdots, \phi_n$ evaluates to T, ψ also evaluates to T, we say that

$$\phi_1, \phi_2, \cdots, \phi_n \models \psi$$

  holds and call $\models$ the *semantic entailment* relation.

Semantic Equivalence

- Definition (Semantic entailment)

  If, for all valuations in which all $\phi_1, \phi_2, \cdots, \phi_n$ evaluates to T, ψ also evaluates to T, we say that

  $$\phi_1, \phi_2, \cdots, \phi_n \models \psi$$

  holds and call $\models$ the *semantic entailment* relation.

- Definition (Semantic equivalence)

  Two propositional logical formulas ϕ and ψ are *semantically equivalent* iff. $\phi \models \psi$ and $\psi \models \phi$. In that case we write, $\phi \equiv \psi$.

Validity and Satisfiability

- Definition (Validity)

  We say a formula ϕ is valid iff $\models \phi$ holds.

- Definition (Satisfiability)

  We say a formula ϕ is satisfiable iff there exists a valuation in which it evaluates to T.

- Proposition

  A formula ϕ is satisfiable iff $\neg\phi$ is not valid.

Conjunctive Normal Forms

- **atom**: proposition, e.g., p, q
- **literal**: atom or the negation of an atom, e.g., $p, \neg p$
- **clause**: disjunction of literals

Definition
A formula is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses.

Example
$(\neg q \lor p \lor r) \land (\neg p \lor r \lor \neg r) \land q$

Every formula can be transformed into an *equivalent* formula in CNF.

Transforming formulas into CNF

- IMPL_FREE: remove all implications by replacing $\phi \rightarrow \eta$ with $\neg\phi \vee \eta$.
- NNF: transform formula into *negation normal form* (NNF), i.e., negation occur only in front of atoms by
  - applying De-Morgan law $\neg(\phi \vee \eta) \equiv \neg\phi \wedge \neg\eta$ and $\neg(\phi \wedge \eta) \equiv \neg\phi \vee \neg\eta$
  - removing double negations $\neg\neg\phi \equiv \phi$.
- DISTR: push all occurrences of $\vee$ inside $\wedge$ by applying the distributive law $(\phi_1 \wedge \phi 2) \vee \phi_3 \equiv (\phi_1 \vee \phi 3) \wedge (\phi_1 \vee \phi_3)$.

Satisfiability and Validity of Formulas in CNF

- Satisfiability in CNF is NP-Complete.
  - For 3-SAT, monotone 3-SAT, and monotone 3-2-SAT
  - 2-SAT can be solved in polynomial time.

Satisfiability and Validity of Formulas in CNF

- Satisfiability in CNF is NP-Complete.
  - For 3-SAT, monotone 3-SAT, and monotone 3-2-SAT
  - 2-SAT can be solved in polynomial time.
- Validity in CNF can be solved in linear time.
  - $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_n$ is valid iff each c_i is valid.
  - $L_1 \vee L_2 \vee \cdots \vee L_m$ is valid iff there exist i, j such that L_i is $\neg L_j$.

Disjunctive Normal Forms

Definition

A formula is in Disjunctive Normal Form (DNF) if it is a disjunction of conjunctions.

Example

$(\neg q \wedge p \wedge r) \vee (\neg p \wedge r \wedge \neg r) \vee q$

- Every formula can be transformed into an *equivalent* formula in DNF.
- Checking satisfiability of formulas in DNF can be solved in linear time.
- Validity of formulas in DNF is co-NP complete.

The SAT Problem

- NP-complete, worst-case exponential time, however, large instances can be solved in practice.
- SAT solvers widely used in verification.
- Hardness of purely randomly generated SAT instances depends primarily upon the ratio of # clauses to # variables.
  - Ration too large, easily unsatisfiable (too constrained).
  - Ratio too small, easily satisfiable (many solutions).
  - For random 3SAT, ratio ≈ 4.2 hardest in an early study. Later study shows dependence on SAT solvers.

Algorithms for SAT

- Modern variants of the DPLL (Davis-Putnam-Logemann-Loveland) algorithm.
    - complete, backtracking,
- Stochastic local search algorithms, e.g., WALKSAT.

DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.

DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.
- **The simplification step**: removes all clauses which become true, and all literals that become false.

DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.
- **The simplification step**: removes all clauses which become true, and all literals that become false.
- Optimizations
  - **Unit propagation**: If a clause is a unit clause, the truth value of the literal is determined. In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.
  - **Pure literal elimination**: If all occurrences of a propositional variable are positive (or negative), it is called pure. Not very useful due to cost of detecting.