Propositional Logic and Resolution

CSE 505 – Computing with Logic

Stony Brook University

http://www.cs.stonybrook.edu/~cse505

Propositional logic

- Alphabet A:
  - Propositional symbols (identifiers)
  - Connectives:
    - $\wedge$ (conjunction),
    - $\vee$ (disjunction),
    - $\neg$ (negation),
    - $\leftrightarrow$ (logical equivalence),
    - $\rightarrow$ (implication).
  - *Well-formed formulas* (*wffs*, denoted by *F*) over alphabet A is the smallest set such that:
    - If p is a predicate symbol in A then p $\in$ *F*.
    - If the wffs F, G $\in$ *F* then so are ($\neg$F), (F $\wedge$ G), (F $\vee$ G), (F $\rightarrow$ G) and (F $\leftrightarrow$ G).

Interpretation

- An *interpretation* I is a subset of propositions in an alphabet A.
- Alternatively, you can view I as a mapping from the set of all propositions in A to a 2-values Boolean domain {true, false}.
- This name, "interpretation", is more commonly used for predicate logic; in the propositional case, this is sometimes called a "*substitution*" or "*truth assignment*".

Semantics of Well-Formed Formulae

- A formula's meaning is given w.r.t. an interpretation I:

  I |= p iff p ∈ I

  I |= ¬F iff I \|= F (i.e., I does not entail F)

  I |= F ∧ G iff I |= F and I |= G

  I |= F ∨ G iff I |= F or I |= G (or both)

  I |= F → G iff I |= G whenever I |= F

  I |= F ↔ G iff I |= F → G and I |= G → F

Models

- An interpretation I such that I |= F is called "*a model*" of F.
- "G is a *logical consequence* of F" (denoted by F |= G) iff every model of F is also a model of G.
  - (in other words, G holds in every model of F;
  or G is true in every interpretation that makes F true)

Models

- A formula that has at least one model is said to be "*satisfiable*".
- A formula for which every interpretation is a model is called a "*tautology*".
- A formula is "*inconsistent*" if it has no models.
- Checking whether or not a formula is satisfiable is NP-Complete (There are exponentially many interpretations)
- Many interesting combinatorial problems can be reduced to checking satisfiability: hence, there is a significant interest in efficient algorithms/heuristics/systems for solving the "SAT" problem.

Logical Consequence

- Let P be a set of clauses {C1, C2,…, Cn}, where
  - each clause Ci is of the form (L1 V L2 V …V Lk), and where
  - each Lj is a literal: i.e. a possibly negated proposition
- A model for P makes every one of Ci 's in P true.
- Let G be a literal (called "Goal")
- Consider the question: does P |= G?
- We can use a proof procedure, based on *resolution* to answer this question.

Proof System for Resolution

$$\frac{}{\{C\} \cup P \vdash C} \quad (\in P)$$

$$\frac{P \vdash (A \vee C_1) \qquad P \vdash (\neg A \vee C_2)}{P \vdash (C_1 \vee C_2)} \quad \text{Resolution}$$

- The above notation is of "*inference rules*" where each rule is of the form:

$$\frac{Antecedent(s)}{Conclusion}$$

- P |- C is called as a "sequent" (P⊢C means C can be *proved* if P is assumed)

Proof System for Resolution

- Given a sequent, a *derivation* of a sequent (sometimes called its "proof") is a tree with:
  - that sequent as the root,
  - empty leaves, and
  - each internal node is an instance of an inference rule.
- A proof system based on Resolution is
  - Sound: i.e. if F |- G then F |= G.
  - not Complete: i.e. there are F,G s.t. F |= G but F not |- G.
- E.g., p |= (p V q) but there is no way to derive

  p |- (p V q).

Resolution Proof (in pictures)

$$P = \{(p \vee q), (\neg p \vee r), (\neg q \vee r)\}$$

$$\dfrac{\dfrac{\dfrac{}{(p \vee q)} \qquad \dfrac{}{(\neg p \vee r)}}{(q \vee r)} \qquad \dfrac{}{(\neg q \vee r)}}{r}$$

Resolution Proof (An Alternative View)

- The clauses of P are all in a "pool".
- Resolution rule picks two clauses from the "pool", of the form $A \vee C1$ and $\neg A \vee C2$.
- and adds $C1 \vee C2$ to the "pool".
- The newly added clause can now interact with other clauses and produce yet more clauses.
- Ultimately, the "pool" consists of all clauses C such that $P \mid\!- C$.

Resolution Proof (An Example)

- $P = \{(p \lor q), (\neg p \lor r), (\neg q \lor r)\}$
- Here is a proof for $P \models r$:

Clause Number	Clause	How Derived
1	$p \lor q$	$\in P$
2	$\neg p \lor r$	$\in P$
3	$\neg q \lor r$	$\in P$
4	$q \lor r$	Res. 1 & 2
5	r	Res. 3 & 4

Refutation Proofs

- While resolution alone is incomplete for determining logical consequence, resolution is sufficient to show inconsistency (i.e. show when P has no model).
- This leads to Refutation proofs for showing logical consequence.
- Say we want to determine P $|=$ r? , where r is a proposition.
- This is equivalent to checking if P $\cup$ $\{\neg r\}$ has an empty model.
- This we can check by constructing a resolution proof for P $\cup$ $\{\neg r\}$ $\vdash$ $\Box$, where $\Box$ denotes the unsatisfiable empty clause.

Refutation Proofs (An Example)

- Let P = {(p V q), (¬p V r), (¬q V r), (p V s)}, and
- G = (r V s)

Clause Number	Clause	How Derived
1	$p \vee q$	$\in P \cup \neg G$
2	$\neg p \vee r$	$\in P \cup \neg G$
3	$\neg q \vee r$	$\in P \cup \neg G$
4	$\neg r$	$\in P \cup \neg G$
5	$\neg s$	$\in P \cup \neg G$
6	$q \vee r$	Res. 1 & 2
7	r	Res. 3 & 6
8	$\square$	Res. 4 & 7

Soundness of Resolution

- If F |- G then F |= G:
  - For F |- G, we will have a derivation (aka "proof") of finite length.
  - We can show that F |= G by induction on the length of derivation.

Refutation-Completeness of Resolution

- If F is inconsistent, then F |- □:
  - Note that F is a set of clauses. A clause is called an unit clause if it consists of a single literal.
  - If all clauses in F are unit clauses, then for F to be inconsistent, clearly a literal and its negation will be two of the clauses in F. Then resolving those two will generate the empty clause.
  - A clause with n + 1 literals has "n excess literals". The proof of refutation-completeness is by induction on the number of excess literals in F.

Refutation-Completeness of Resolution

- If F is inconsistent, then F |- □:
  - Assume refutation completeness holds for all clauses with n excess literals; show that it holds for clauses with n + 1 excess literals.
  - From F, pick some clause C with excess literals. Pick some literal, say A from C. Consider C' = C-{A}.
  - Both F1=(F–{C})∪{C'} and F2=(F–{C}) ∪ {A} are inconsistent and have at most n excess literals.
  - By induction hypothesis, both have refutations. If there is a refutation of F1 not using C', then that is a refutation for F as well.
  - If refutation of F1 uses C', then construct a resolution of F by adding A to the first occurrence of C' (and its descendants); that will end with {A}. From here on, follow the refutation of F2. This constructs a refutation of F.