COMP9414: Artificial Intelligence

Propositional Logic: Automated Reasoning

Wayne Wobcke

Room J17-433
wobcke@cse.unsw.edu.au
Based on slides by Maurice Pagnucco

 ©UNSW, 2004

Propositional Logic

- So far we have considered propositional logic as a knowledge representation language
- We can write sentences in this language (syntax) with some logical structure
- We can define the interpretations of these sentences using truth tables (semantics)
- What remains is reasoning; to draw new conclusions from what we know (proof system) and to do so using a computer to automate the process
- References:
  - Ivan Bratko, Prolog Programming for Artificial Intelligence, Addison-Wesley, 2001. (Chapter 15)
  - Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall International, 1995. (Chapter 6)

Overview

- Proof systems (including soundness and completeness)
- Normal Forms
- Resolution
- Refutation Systems
- Correctness of the resolution rule — soundness and completeness
- Conclusion

What is a Logic?

- A logic consists of
  1. A formal system for expressing knowledge about a domain consisting of

     Syntax Set of legal sentences (well formed formulae)
     Semantics Interpretation of legal sentences
  2. A proof system — a set of axioms plus rules of inference for deducing sentences from a knowledge base

Mechanising Proof

Question: Assuming knowledge can be captured using propositional logic, how do we automate reasoning (i.e. perform inference)?

- One answer: a proof of a formula from a set of premises is a sequence of steps in which any step of the proof is:
  1. An axiom or premise
  2. A formula deduced from previous steps of the proof using some rule of inference

  The last step of the proof should deduce the formula we wish to prove
- This is intended to formally capture the notion of proof that is commonly applied in other fields (e.g. mathematics)
- We use the notation $S \vdash P$ to denote that the set of formulae S "prove" the formula P. Alternatively, we say that P follows from (premises) S

Soundness and Completeness

- A logic is sound if it preserves truth (i.e. if a set of premises are all true, any conclusion drawn from those premises **must** also be true)
- Technically, a proof system $\vdash$ is sound if whenever $S \vdash P$ (P follows from S using the proof system), $S \models P$ (P is entailed by S, e.g. using truth tables)
- A logic is complete if it is capable of proving all consequences of any knowledge base
- Technically, a proof system $\vdash$ is complete if whenever $S \models P$ (P is entailed by S, e.g. using truth tables), $S \vdash P$ (P follows from S using the proof system)
- A logic is decidable if there is a mechanical procedure (computer program) which when asked whether $S \vdash P$, can always answer 'yes' or 'no' (correctly)

Resolution

- Another type of proof system based on refutation
- Better suited to computer implementation than systems of axioms and rules (can give correct 'no' answers)
- Generalizes to first-order logic (see next week)
- The basis of Prolog's inference method
- To apply resolution, all formulae in the knowledge base and the query must be in clausal form (c.f. Prolog clauses)

Normal Forms

- A literal is a propositional letter or the negation of a propositional letter
- A clause is a disjunction of literals
- Conjunctive Normal Form (CNF) — a conjunction of clauses, e.g. $(P \vee Q \vee \neg R) \wedge (\neg S \vee \neg R)$
- Disjunctive Normal Form (DNF) — a disjunction of conjunctions of literals, e.g. $(P \wedge Q \wedge \neg R) \vee (\neg S \wedge \neg R)$
- Every propositional logic formula can be converted to CNF and DNF

Conversion to Conjunctive Normal Form

- Eliminate $\leftrightarrow$ rewriting $P \leftrightarrow Q$ as $(P \rightarrow Q) \wedge (Q \rightarrow P)$
- Eliminate $\rightarrow$ rewriting $P \rightarrow Q$ as $\neg P \vee Q$
- Use De Morgan's laws to push $\neg$ inwards:
  - rewrite $\neg(P \wedge Q)$ as $\neg P \vee \neg Q$
  - rewrite $\neg(P \vee Q)$ as $\neg P \wedge \neg Q$
- Eliminate double negations: rewrite $\neg\neg P$ as P
- Use the distributive laws to get CNF:
  - rewrite $(P \wedge Q) \vee R$ as $(P \vee R) \wedge (Q \vee R)$
  - rewrite $(P \vee Q) \wedge R$ as $(P \wedge R) \vee (Q \wedge R)$

Example

- $\neg(P \rightarrow (Q \wedge R))$
- $\neg(\neg P \vee (Q \wedge R))$
- $\neg\neg P \wedge \neg(Q \wedge R)$
- $\neg\neg P \wedge (\neg Q \vee \neg R)$
- $P \wedge (\neg Q \vee \neg R)$
- Two clauses: P, $\neg Q \vee \neg R$

Resolution Rule of Inference

Resolution Rule

$$\frac{A \vee B \qquad \neg B \vee C}{A \vee C}$$

- where B is a propositional letter and A and C are clauses (possibly empty)
- $A \vee C$ is the resolvent of the two clauses

Resolution Rule: Key Idea

- Consider $A \vee B$ and $\neg B \vee C$
  - if B is True, $\neg B$ is False and truth of second formula depends only on C
  - if B is False, truth of first formula depends only on A
- Only one of B, $\neg B$ is True, so if both $A \vee B$ and $\neg B \vee C$ are True, either A or C is True, i.e. $A \vee C$ is True

Applying Resolution

- The resolution rule is sound (resolvent entailed by two 'parent' clauses)
- How can we use the resolution rule? One way:
  - Convert knowledge base into clausal form
  - Repeatedly apply resolution rule to the resulting clauses
  - A query A follows from the knowledge base if and only if each of the clauses in the CNF of A can be derived using resolution
- There is a better way ...

Refutation Systems

- To show that P follows from S (i.e. $S \vdash P$) using refutation, start with S and $\neg P$ in clausal form and derive a contradiction using resolution
- A contradiction is the "empty clause" (a clause with no literals)
- The empty clause $\Box$ is unsatisfiable (always False)
- So if the empty clause $\Box$ is derived using resolution, the original set of clauses is unsatisfiable (never all True together)
- That is, if we can derive $\Box$ from the clausal forms of S and $\neg P$, these clauses can never be all True together
- Hence whenever the clauses of S are all True, at least one clause from $\neg P$ must be False, i.e. $\neg P$ must be False and P must be True
- By definition, $S \models P$ (so P can correctly be concluded from S)

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated conclusion into CNF and extract clauses
- Repeatedly apply resolution until either the empty clause (contradiction) is derived or no more clauses can be derived
- If the empty clause is derived, answer 'yes' (query follows from knowledge base), otherwise answer 'no' (query does not follow from knowledge base)

Resolution: Example 1

$(G \vee H) \rightarrow (\neg J \wedge \neg K),\ G \vdash \neg J$

Clausal form of $(G \vee H) \rightarrow (\neg J \wedge \neg K)$ is $\{\neg G \vee \neg J,\ \neg H \vee \neg J,\ \neg G \vee \neg K,\ \neg H \vee \neg K\}$

1. $\neg G \vee \neg J$ [Premise]
2. $\neg H \vee \neg J$ [Premise]
3. $\neg G \vee \neg K$ [Premise]
4. $\neg H \vee \neg K$ [Premise]
5. G [Premise]
6. J [$\neg$ Conclusion]
7. $\neg G$ [1, 6. Resolution]
8. $\Box$ [5, 7. Resolution]

Resolution: Example 2

$P \rightarrow \neg Q,\ \neg Q \rightarrow R \vdash P \rightarrow R$

Recall $P \rightarrow R \equiv \neg P \vee R$

Clausal form of $\neg(\neg P \vee R)$ is $\{P,\ \neg R\}$

1. $\neg P \vee \neg Q$ [Premise]
2. $Q \vee R$ [Premise]
3. P [$\neg$ Conclusion]
4. $\neg R$ [$\neg$ Conclusion]
5. $\neg Q$ [1, 3. Resolution]
6. R [2, 5. Resolution]
7. $\Box$ [4, 6. Resolution]

Resolution: Example 3

$\vdash ((P \vee Q) \wedge \neg P) \rightarrow Q$

Clausal form of $\neg(((P \vee Q) \wedge \neg P) \rightarrow Q)$ is $\{P \vee Q,\ \neg P,\ \neg Q\}$

1. $P \vee Q$ [$\neg$ Conclusion]
2. $\neg P$ [$\neg$ Conclusion]
3. $\neg Q$ [$\neg$ Conclusion]
4. Q [1, 2. Resolution]
5. $\Box$ [3, 4. Resolution]

Soundness and Completeness Again

- Resolution refutation is sound, i.e. it preserves truth (if a set of premises are all true, any conclusion drawn from those premises **must** also be true)
- Resolution refutation is complete, i.e. it is capable of proving all consequences of any knowledge base (not shown here!)
- Resolution refutation is decidable, i.e. there is an algorithm implementing resolution which when asked whether $S \vdash P$, can always answer 'yes' or 'no' (correctly)

Heuristics in Applying Resolution

- Clause elimination — can disregard certain types of clauses
  - Pure clauses: contain literal L where $\neg L$ doesn't appear elsewhere
  - Tautologies: clauses containing both L and $\neg L$
  - Subsumed clauses: another clause exists containing a subset of the literals
- Ordering strategies
  - Resolve unit clauses (only one literal) first
  - Start with query clauses
  - Aim to shorten clauses

Conclusion

- We have now investigated one knowledge representation and reasoning formalism
- This means we can draw new conclusions from the knowledge we have: we can reason
- Have enough to build a knowledge-based agent
- However, propositional logic is a weak language; there are many things that cannot be expressed
- To express knowledge about objects, their properties and the relationships that exist between objects, we need a more expressive language: first-order logic

 ©UNSW, 2004