COMP4418: Knowledge Representation and Reasoning

Propositional Logic: Automating Reasoning

Maurice Pagnucco
School of Computer Science and Engineering
University of New South Wales
NSW 2052, AUSTRALIA
morri@cse.unsw.edu.au
Propositional Logic

- Thus far we have considered propositional logic as a knowledge representation language
- We can now write sentences in this language (syntax)
- We can also determine the truth or falsity of these sentences (semantics)
- What remains is to reason; to draw new conclusions from what we know (proof theory) and to do so using a computer to automate the process

References:
- Ivan Bratko, Prolog Programming for Artificial Intelligence, Addison-Wesley, 2001. (Chapter 15)
Overview

- Normal Forms
- Resolution
- Refutation Systems
- Correctness of resolution rule — soundness and completeness revisited
- Conclusion
Motivation

If either George or Herbert wins, then both Jack and Kenneth lose

George wins

Therefore, Jack loses

\[(G \lor H) \rightarrow (\neg J \land \neg K)\]

\[G\]

\[\neg J\]
Normal Forms

- A normal form is a “standardised” version of a formula

- Common normal forms:
  - Negation Normal Form — negation symbols occur in front of propositional letters only (e.g., $(P ∨ ¬Q) → (P ∧ (¬R ∨ S))$
  (A literal is a propositional letter or the negation of a propositional letter.)
  - Conjunctive Normal Form (CNF) — a conjunct of disjunctions (e.g., $(P ∨ Q ∨ ¬R) ∧ (¬S ∨ ¬R)$)
  Disjunctions of literals are known as clauses
  - Disjunctive Normal Form (DNF) — a disjunct of conjunctions (e.g., $(P ∧ Q ∧ ¬R) ∨ (¬S ∧ ¬R)$)
Negation Normal Form

- To simplify matters, let us suppose we are only dealing with formulae containing the connectives $\neg, \land, \lor$

- A (sub)formula $\phi \rightarrow \psi$ is equivalent to $\neg\phi \lor \psi$

- A (sub) formula $\phi \leftrightarrow \psi$ is equivalent to $\phi \rightarrow \psi$ and $\psi \rightarrow \phi$

- DeMorgan’s laws:
  - $\neg (\phi \land \psi) \equiv \neg\phi \lor \neg\psi$
  - $\neg (\phi \lor \psi) \equiv \neg\phi \land \neg\psi$

- Double Negation: $\neg\neg P \equiv P$

- To put a formula in negation normal form, repeatedly apply De Morgan’s laws and double negation

- For example, $\neg (P \lor (\neg R \land P)) \equiv \neg P \land \neg (\neg R \land P) \equiv \neg P \land (R \lor \neg P)$
Conjunctive Normal Form

- Note the following distributive identities:
  \[(\phi \land \psi) \lor \chi \equiv (\phi \lor \chi) \land (\psi \lor \chi)\]
  \[(\phi \lor \psi) \land \chi \equiv (\phi \land \chi) \lor (\psi \land \chi)\]

- To put a formula in conjunctive normal form (CNF) firstly put the formula into negation normal form and then repeatedly apply the identities above

- For example, \( R \rightarrow (P \land Q) \equiv (\neg R \lor P) \land (\neg R \lor Q) \)
Resolution Rule

Resolution Rule:

\[ \alpha \lor \beta \quad \text{and} \quad \neg \beta \lor \gamma \]

\[ \alpha \lor \gamma \]

- Where \( \beta \) is a literal (i.e., a propositional letter or its negation)
Resolution Rule

Resolution is essentially equivalent to the transitivity of material implication

In fact, it is a form of the well known cut rule in logic
Applying Resolution

- The resolution rule is sound
- What does that mean?
- How can we use the resolution rule?
  - Convert premises into CNF
  - Repeatedly apply resolution rule to the resultant clauses
  - Each clause produced can be inferred from the original premises
  - If you have a query sentence goal, it follows from the premises if and only if each of the clauses in CNF(goal) is produced by resolution
- There is a better way . . .
Refutation Systems

■ If we would like to prove a sentence $\phi$ is a theorem (i.e., $\vdash \phi$), we start with $\neg \phi$ and produce a contradiction

■ A “proof by contradiction”

■ Similarly, if we wish to prove $\psi_1, \ldots, \psi_n \vdash \phi$, start with $\neg \phi$ and together with $\psi_1, \ldots, \psi_n$ produce a contradiction

■ Resolution can be used to implement a refutation system

■ Repeatedly apply resolution rule until empty clause results
Applying Resolution

- Negate conclusion (resolution is a refutation system)
- Convert premises and negated conclusion into CNF (clausal form)
- Repeatedly apply Resolution Rule, Double Negation
- If empty clause results you have a contradiction and can conclude that the conclusion follows from the premises
Resolution — Example 1

\[(G \lor H) \to (\neg J \land \neg K), \ G \vdash \neg J\]

\[CNF[(G \lor H) \to (\neg J \land \neg K)] \equiv (\neg G \lor \neg J) \land (\neg H \lor \neg J) \land (\neg G \lor \neg K) \land (\neg H \lor \neg K)\]

1. \(\neg G \lor \neg J\) [Premise]
2. \(\neg H \lor \neg J\) [Premise]
3. \(\neg G \lor \neg K\) [Premise]
4. \(\neg H \lor \neg K\) [Premise]
5. \(G\) [Premise]
6. \(\neg \neg J\) [\(\neg\) Conclusion]
7. \(J\) [6. Double Negation]
8. \(\neg G\) [1, 7. Resolution]
9. \(\Box\) [5, 8. Resolution]
Resolution — Example 2

\[ P \rightarrow \neg Q, \quad \neg Q \rightarrow R \vdash P \rightarrow R \]

\[ P \rightarrow R \equiv \neg P \lor R \]

\[ CNF[\neg(\neg P \lor R)] \equiv \{\neg\neg P, \neg R\} \]

1. \(\neg P \lor \neg Q\) [Premise]
2. \(\neg \neg Q \lor R\) [Premise]
3. \(\neg \neg P\) [\(\neg\) Conclusion]
4. \(\neg R\) [\(\neg\) Conclusion]
5. \(P\) [3. Double Negation]
6. \(\neg Q\) [1, 5. Resolution]
7. \(R\) [2, 6. Resolution]
8. \(\Box\) [4, 7. Resolution]
Resolution — Example 3

\[ \vdash ((P \lor Q) \land \neg P) \rightarrow Q \]

\[ CNF[\neg(((P \lor Q) \land \neg P) \rightarrow Q)] \equiv (P \lor Q) \land \neg P \land \neg Q \]

1. \( P \lor Q \) [\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 — Recap

■ An inference procedure (and hence a logic) is sound if and only if it preserves truth

■ In other words $\vdash$ is sound iff whenever $\lambda \vdash \rho$, then $\lambda \models \rho$

■ A logic is complete if and only if it is capable of proving all truths

■ In other words, whenever $\lambda \models \rho$, then $\lambda \vdash \rho$

Decidability

■ A logic is decidable if and only if there is a mechanical procedure that, when asked $\lambda \vdash \rho$, can eventually halt and answer “yes” or halt and answer “no”

■ Propositional logic is decidable
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
  - Unit preference: resolve unit clauses (only one literal) first

- Many others . . .
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 we can’t express in it
- It cannot be used to express knowledge about objects, their properties and the relationships that exist between objects
- For this purpose we need a more expressive language: first-order logic