COMP210: Artificial Intelligence

Lecture 23. Propositional resolution

Trevor Bench-Capon

http://www.csc.liv.ac.uk/~tbc/COMP210/

Overview

- In the last lecture we studied the use of predicate logic for knowledge representation.
  - Computer methods are needed to deal with huge knowledge bases.
  - Enumeration of models is not possible in predicate logic.
  - Natural deduction contains too many rules; hard to implement search.
- In this lecture we describe the resolution proof method.
- This requires formulae to be in conjunctive normal form (CNF).

Validity, Satisfiability, and Entailment

Implications for Knowledge Representation

- *Deduction Theorem*:
  $KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid
- Or,...
  $KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable
  - *reductio ad absurdum*

For propositional, predicate and many other logics

Resolution

- Resolution is a proof method for classical propositional and first-order logic.
- Given a formula φ resolution will decide whether the formula is *unsatisfiable* or not.
- Resolution was suggested by Robinson in the 1960s and claimed it to be *machine oriented* as it had only one rule of inference.

Resolution Method

The method involves:-

- translation to a normal form (CNF);
- At each step, a new clause is derived from two clauses you already have
- Proof steps all use the same rule
  - resolution rule;
- repeat until false is derived (i.e. the formula contains a literal **and** its negation) or no new formulae can be derived.

We first introduce the method for propositional logic and then (next lecture) extend it to predicate logic.

Resolution Rule

- Each A_i is known as a *clause* and we consider the set of clauses $\{A_1, A_2, \ldots, A_k\}$
- The (propositional) resolution rule is as follows.

$$\frac{\begin{array}{c} A \vee p \\ B \vee \neg p \end{array}}{A \vee B}$$

- $A \vee B$ is called the *resolvent*.
- $A \vee p$ and $B \vee \neg p$ are called *parents of the resolvent*.
- p and $\neg p$ are called *complementary literals*.
- Note in the above A or B can be empty.

Resolution applied to Sets of Clauses

Show by resolution that the following set of clauses is unsatisfiable.

$$\{p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q\}$$

1.	$p \vee q$	
2.	$p \vee \neg q$	
3.	$\neg p \vee q$	
4.	$\neg p \vee \neg q$	
5.	p	$[1, 2]$
6.	$\neg p$	$[3, 4]$
7.	**false**	$[5, 6]$

Resolution algorithm

Proof by contradiction, i.e. show that $KB \wedge \alpha$ unsatisfiable

```
function PL-RESOLUTION(KB,α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
       for each Ci, Cj in clauses do
           resolvents ← PL-RESOLVE(Ci,Cj)
           if resolvents contains the empty clause then return true
           new ← new ∪ resolvents
       if new ⊆ clauses then return false
       clauses ← clauses ∪ new
```

Full Circle Example

- Using resolution show

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

- show that

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

  is unsatisfiable
  - translate to CNF.
  - apply the resolution algorithm

1. Transformation to CNF

$$
\begin{aligned}
& ((q \wedge p) \Rightarrow r) \wedge \neg(\neg p \vee \neg q \vee r) \\
\equiv\ & (\neg(q \wedge p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\
\equiv\ & ((\neg q \vee \neg p) \vee r) \wedge \neg(\neg p \vee \neg q \vee r) \\
\equiv\ & (\neg q \vee \neg p \vee r) \wedge (\neg\neg p \wedge \neg\neg q \wedge \neg r) \\
\equiv\ & (\neg q \vee \neg p \vee r) \wedge (p \wedge q \wedge \neg r) \\
\equiv\ & (\neg q \vee \neg p \vee r) \wedge p \wedge q \wedge \neg r
\end{aligned}
$$

2. Resolution

1. $\neg q \vee \neg p \vee r$
2. p
3. q
4. $\neg r$

Finally apply the resolution rule.

5. $\neg q \vee r$ $[1, 2]$
6. r $[5, 3]$
7. **false** $[4, 6]$

Discussion

- As we have derived false then that means the formula was unsatisfiable.
- Note if we couldn't obtain false that means the original formula was satisfiable.
- This means that the original formula was a consequence of the KB.

Comment I

- The resolution rule is derived from a generalisation of the modus ponens inference rule given below.

$$\frac{\begin{array}{l} P \\ P \Rightarrow B \text{ (equivalent to } \neg P \vee B) \end{array}}{B}$$

- This can be generalised to

$$\frac{\begin{array}{l} A \Rightarrow P \text{ (equivalent to } \neg A \vee P) \\ P \Rightarrow B \text{ (equivalent to } \neg P \vee B) \end{array}}{A \Rightarrow B \text{ (equivalent to } \neg A \vee B)}$$

Comment II

- Resolution restricts the P so it is a proposition, i.e.

$$\frac{\begin{array}{c} A \Rightarrow p \\ p \Rightarrow B \end{array}}{A \Rightarrow B}$$

- Given a set of clauses $A_1 \wedge A_2 \ldots A_k$ to which we apply the resolution rule, if we derive **false** we have obtained $A_1 \wedge \ldots \wedge$ **false** which is equivalent to **false**. Thus the set of clauses is unsatisfiable.

Theoretical Issues

- Resolution is *refutation complete*. That is if given an unsatisfiable set of clauses the procedure is guaranteed to produce **false**.
- Resolution is *sound*. That is if we derive **false** from a set of clauses then the set of clauses is unsatisfiable.
- The resolution method *terminates*. That is we apply the resolution rule until we derive false or no new clauses can be derived and will always stop.

Reducing the Search Space I

- Although the basic resolution method is complete it is not very efficient. This is due to the search space that has to be explored.
- A lot of effort has been applied in trying to reduce the search space.
  - The elimination of tautologies (eg clauses such as $p \vee q \vee \neg q$) or
  - subsumption (if a clause set contains the clauses p and $p \vee q$, $p \vee q$ may be discarded) removes useless or redundant rules.

Reducing the Search Space II

- Some forms of resolution restrict which clauses may be resolved together eg *unit resolution* (always resolve using at least one unit clause) or *set of support* (after the first step, use at most one original clause).
- Heuristics may be applied to guide the proof search eg *weighting strategies*.
- Applying strategies such as set of support or heuristics may affect completeness.

Automated Reasoning

- The resolution proof method may be automated, i.e. carried out by a computer program.
- Theorem provers based on resolution have been developed eg Otter, Spass.
- The topic of automated reasoning lies within the area of AI.
- Prolog also uses resolution, but only for a subset (Horn Clauses).
  - At most one positive literal in any clause
  - p :- q, r is equivalent to $p \vee \neg q \vee \neg r$
  - This greatly improves efficiency, making Prolog usable as a programming language.

Resolution in Prolog

(1) p:- q, r. i.e. $p \vee \neg q \vee \neg r$
(2) q:- t. i.e. $q \vee \neg t$
(3) r:- u. i.e. $r \vee \neg u$
(4) t. (5) u.

To show (6) p first add $\neg p$. Use unit clause and set of support.

Resolve (6) and (1) to get (7) $\neg q \vee \neg r$
Resolve (7) and (2) to get (8) $\neg t \vee \neg r$
Resolve (4) and (8) to get (9) $\neg r$
Resolve (9) and (3) to get (10) $\neg u$
Resolve (10) and (5) to get empty clause.

$\neg p$ is unsatifiable and hence P is true.

Summary

- We have described how to apply the proof method *resolution*.
- First formulae need to be in conjunctive normal form.
- There is only one rule of inference.
- In the next lecture we look at applying resolution to first-order logic.