CSC 665: Artificial Intelligence

Lecture: Logic III

1 A word on contradiction

So far we've been studying inference algorithms (model checking and forward inference with modus ponens) for entailment. But in the case that a query α is not entailed by our knowledge base KB, we'd like to know whether α contradicts KB. Do we need an entirely new set of algorithms for contradiction?

As it turns out, we don't. We can reduce contradiction to entailment, thus enabling us to take advantage of the inference procedures we've already developed. In particular, we claim that α contradicts KB if and only if KB $\models \neg\alpha$. As an exercise, you should convince yourself that this is true using the definitions of entailment and contradiction.

2 Complete inference via resolution

We've seen that forward inference with modus ponens is not complete on all of propositional logic. Last time we took the approach of simplifying our representation language, and that led us to the major result that forward inference with modus ponens is in fact complete on Horn clauses alone. Here we take an alternative approach and build up a set of more powerful inference rules that will enable us to do complete inference on all of propositional logic.

2.1 The resolution inference rule

To start, note that we can rewrite modus ponens without using the implication connective:

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

Framing modus ponens in this way lets us see it as an inference rule for "canceling out" the two literals p and $\neg p$, leaving us with q.

The same inference rule works when both of the antecedents are clauses. For example,

$$\frac{\text{Sun} \vee \text{Rain},\ \neg\text{Rain} \vee \text{Traffic}}{\text{Sun} \vee \text{Traffic}}.$$

In this case, Rain and ¬Rain cancel out to leave the other literals in each clause. Generalizing this to clauses with arbitrary numbers of literals gives us the key inference rule for devising a complete inference procedure.

Definition: The resolution inference rule is

$$\frac{f_1 \vee \cdots \vee f_n \vee p,\ \neg p \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m},$$

where all of the f_i's, g_i's, and p are literals.

Although the rule may look complex, note that all we are doing is canceling out p with $\neg p$ to get the conclusion in the bottom half of the rule.

2.2 Conjunctive normal form

Resolution is an inference rule that takes two clauses as antecedents and produces another clause as its conclusion. If resolution is just transforming clauses into other clauses, how do we deal with formulas in our knowledge base that aren't clauses? As we show below, it turns out that we can always write our knowledge base as a conjunction of clauses, after which we can repeatedly apply resolution to appropriate pairs of clauses. Thus there is no loss of generality in using an inference rule that works solely on clauses.

Definition: A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses (i.e., an "and of ors").

Example: The following formulas are in CNF:

- $(A \vee B) \wedge (C \vee D)$
- $(A \vee B) \wedge C$
- $A \vee B$
- $A \wedge B$
- $\neg A \wedge (B \vee \neg C)$

The following formulas are *not* in CNF:

- $A \vee (C \wedge B)$ is a disjunction of conjunctions (an "or of ands"), which gets the definition backwards.
- $A \wedge (B \vee (C \wedge \neg D))$ is a conjunction of disjunctions, but the disjunction $B \vee (C \wedge \neg D)$ is not a clause because $C \wedge \neg D$ is not a literal.

Theorem: Every formula f in propositional logic can be converted to an equivalent formula f' which is in CNF and which satisfies $M(f) = M(f')$.

We won't prove this theorem, but we'll illustrate the conversion process with an example.

Example: Starting with the formula $(A \implies B) \implies C$, we apply a sequence of inference rules to derive an equivalent formula in CNF.

$$
\begin{gathered}
(A \implies B) \implies C \\
\downarrow \\
\neg(\neg A \vee B) \vee C \\
\downarrow \\
(\neg\neg A \wedge \neg B) \vee C \\
\downarrow \\
(A \wedge \neg B) \vee C \\
\downarrow \\
(A \vee C) \wedge (\neg B \vee C)
\end{gathered}
$$

Here is the full list of additional inference rules we need in order to convert any formula to CNF. In fact, if you apply these rules in the order given, you are guaranteed to end up with a formula in CNF.

- **Biconditional elimination**

$$\frac{p \iff q}{(p \implies q) \wedge (q \implies p)}$$

- **Implication elimination**

$$\frac{p \implies q}{\neg p \vee q}$$

- **De Morgan's laws**

$$\frac{\neg(p \wedge q)}{\neg p \vee \neg q} \text{ and } \frac{\neg(p \vee q)}{\neg p \wedge \neg q}$$

- **Double negation elimination**

$$\frac{\neg\neg p}{p}$$

- **Distributive laws**

$$\frac{p \vee (q_1 \wedge q_2)}{(p \vee q_1) \wedge (p \vee q_2)} \text{ and } \frac{p \wedge (q_1 \vee q_2)}{(p \wedge q_1) \vee (p \wedge q_2)}$$

As an exercise, you should label each step in the example above with the inference rule used to derive the new formula in that step.

2.3 Inference by resolution

In section 1 we noted that KB contradicts α if and only if KB $\models \neg\alpha$. Equivalently, KB $\models \alpha$ if and only if KB contradicts $\neg\alpha$. That is, we can also reduce entailment to contradiction, enabling us to turn an algorithm for contradiction into one for entailment. This will be our approach for doing inference by resolution (Algorithm 1).

Algorithm 1 Inference by resolution

```
1: procedure RESOLVE(KB, α)
2:     add ¬α to KB
3:     convert KB to conjunctive normal form
4:     while we can keep adding new formulas to KB do
5:         pick two clauses f, g ∈ KB that match the resolution rule to yield the new formula h
6:         if h is the empty clause (False) then
7:             return True
8:     return False
```

Algorithm 1 attempts to find a contradiction between KB and α. If it does, it returns True to indicate entailment; otherwise it returns False to indicate that α is not entailed by KB. The idea is that, starting with a knowledge base containing a contradiction, a complete inference procedure will always be able to derive False. (This is essentially the same idea behind proofs by contradiction.)

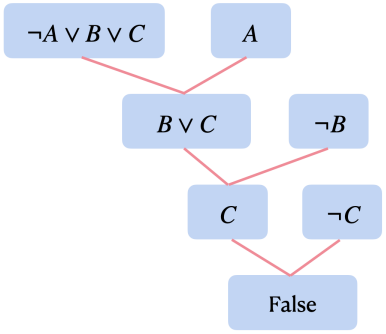
Example: To illustrate inference by resolution, let

$$\begin{aligned} \text{KB} &= \{A \implies (B \vee C),\ A,\ \neg B\} \\ \alpha &= C. \end{aligned}$$

Then we begin by adding $\neg\alpha$ to KB and converting this augmented KB to CNF, giving us the new knowedge base

$$\text{KB}' = \{\neg A \vee B \vee C,\ A,\ \neg B,\ \neg C\}.$$

From here, we repeatedly resolve pairs of clauses, canceling out the literal whose negation status differs each time:



The following theorem shows that this repeated application of resolution does exactly what we want it to do.

Theorem: Inference by resolution is sound and complete.

3 Summary of inference

We've seen a number of algorithms for logical inference on propositional logic. We summarize some of the key tradeoffs between these algorithms in the following table.

Language	Expressivity	Inference procedure	Sound?	Complete?	Time complexity
propositional logic	more	model checking	✓	✓	exponential
propositional logic	more	modus ponens	✓	✗	linear
Horn clauses	less	modus ponens	✓	✓	linear
propositional logic	more	resolution	✓	✓	exponential