Logic: resolution

Review: tradeoffs

Formulas allowed	Inference rule	Complete?
Propositional logic	modus ponens	no
Propositional logic (only Horn clauses)	modus ponens	yes
Propositional logic	**resolution**	yes

- We saw that if our logical language was restricted to Horn clauses, then modus ponens alone was sufficient for completeness. For general propositional logic, modus ponens is insufficient.
- In this lecture, we'll see that a more powerful inference rule, **resolution**, is complete for all of propositional logic.

Horn clauses and disjunction

Written with implication	Written with disjunction
$A \to C$	$\neg A \vee C$
$A \wedge B \to C$	$\neg A \vee \neg B \vee C$

- **Literal**: either p or $\neg p$, where p is a propositional symbol
- **Clause**: disjunction of literals
- **Horn clauses**: at most one positive literal

Modus ponens (rewritten):

$$\frac{A, \quad \neg A \vee C}{C}$$

- Intuition: cancel out A and $\neg A$

- Modus ponens can only deal with Horn clauses, so let's see why Horn clauses are limiting. We can equivalently write implication using negation and disjunction. Then it's clear that Horn clauses are just disjunctions of literals where there is at most one positive literal and zero or more negative literals. The negative literals correspond to the propositional symbols on the left side of the implication, and the positive literal corresponds to the propositional symbol on the right side of the implication.
- If we rewrite modus ponens, we can see a "canceling out" intuition emerging. To make the intuition a bit more explicit, remember that, to respect soundness, we require $\{A, \neg A \vee C\} \models C$; this is equivalent to: if $A \wedge (\neg A \vee C)$ is true, then C is also true. This is clearly the case.
- But modus ponens cannot operate on general clauses.

Resolution [Robinson, 1965]

General clauses have any number of literals:

$$\neg A \vee B \vee \neg C \vee D \vee \neg E \vee F$$

Example: resolution inference rule

$$\frac{\text{Rain} \vee \text{Snow}, \quad \neg\text{Snow} \vee \text{Traffic}}{\text{Rain} \vee \text{Traffic}}$$

Definition: resolution inference rule

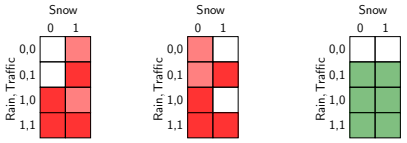
$$\frac{f_1 \vee \cdots \vee f_n \vee p, \quad \neg p \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

- Let's try to generalize modus ponens by allowing it to work on general clauses. This generalized inference rule is called **resolution**, which was invented in 1965 by John Alan Robinson.
- The idea behind resolution is that it takes two general clauses, where one of them has some propositional symbol p and the other clause has its negation $\neg p$, and simply takes the disjunction of the two clauses with p and $\neg p$ removed. Here, $f_1, \ldots, f_n, g_1, \ldots, g_m$ are arbitrary literals.

Soundness of resolution

$$\frac{\text{Rain} \vee \text{Snow}, \quad \neg\text{Snow} \vee \text{Traffic}}{\text{Rain} \vee \text{Traffic}} \text{ (resolution rule)}$$

$\mathcal{M}(\text{Rain} \vee \text{Snow}) \cap \mathcal{M}(\neg\text{Snow} \vee \text{Traffic}) \subseteq ?\ \mathcal{M}(\text{Rain} \vee \text{Traffic})$



Sound!

- Why is resolution logically sound? We can verify the soundness of resolution by checking its semantic interpretation. Indeed, the intersection of the models of f and g is a subset of models of $f \vee g$.

Conjunctive normal form

So far: resolution only works on clauses...but that's enough!

Definition: conjunctive normal form (CNF)

A **CNF formula** is a conjunction of clauses.

Example: $(A \vee B \vee \neg C) \wedge (\neg B \vee D)$

Equivalent: knowledge base where each formula is a clause

Proposition: conversion to CNF

Every formula f in propositional logic can be converted into an equivalent CNF formula f':

$$\mathcal{M}(f) = \mathcal{M}(f')$$

- But so far, we've only considered clauses, which are disjunctions of literals. Surely this can't be all of propositional logic... But it turns out it actually is in the following sense.
- A conjunction of clauses is called a CNF formula, and every formula in propositional logic can be converted into an equivalent CNF. Given a CNF formula, we can toss each of its clauses into the knowledge base.
- But why can every formula be put in CNF?

Conversion to CNF: example

Initial formula:

$$(\text{Summer} \rightarrow \text{Snow}) \rightarrow \text{Bizzare}$$

Remove implication ($\rightarrow$):

$$\neg(\neg\text{Summer} \vee \text{Snow}) \vee \text{Bizzare}$$

Push negation ($\neg$) inwards (de Morgan):

$$(\neg\neg\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Remove double negation:

$$(\text{Summer} \wedge \neg\text{Snow}) \vee \text{Bizzare}$$

Distribute $\vee$ over $\wedge$:

$$(\text{Summer} \vee \text{Bizzare}) \wedge (\neg\text{Snow} \vee \text{Bizzare})$$

- The answer is by construction. There is a six-step procedure that takes any propositional formula and turns it into CNF. Here is an example of how it works (only four of the six steps apply here).

Conversion to CNF: general

Conversion rules:

- Eliminate $\leftrightarrow$: $\frac{f \leftrightarrow g}{(f \rightarrow g) \wedge (g \rightarrow f)}$
- Eliminate $\rightarrow$: $\frac{f \rightarrow g}{\neg f \vee g}$
- Move $\neg$ inwards: $\frac{\neg(f \wedge g)}{\neg f \vee \neg g}$
- Move $\neg$ inwards: $\frac{\neg(f \vee g)}{\neg f \wedge \neg g}$
- Eliminate double negation: $\frac{\neg\neg f}{f}$
- Distribute $\vee$ over $\wedge$: $\frac{f \vee (g \wedge h)}{(f \vee g) \wedge (f \vee h)}$

- Here are the general rules that convert any formula to CNF. First, we try to reduce everything to negation, conjunction, and disjunction.
- Next, we try to push negation inwards so that they sit on the propositional symbols (forming literals). Note that when negation gets pushed inside, it flips conjunction to disjunction, and vice-versa.
- Finally, we distribute so that the conjunctions are on the outside, and the disjunctions are on the inside.
- Note that each of these operations preserves the semantics of the logical form (remember there are many formula that map to the same set of models). This is in contrast with most inference rules, where the conclusion is more general than the conjunction of the premises.
- Also, when we apply a CNF rewrite rule, we replace the old formula with the new one, so there is no blow-up in the number of formulas. This is in contrast to applying general inference rules. An analogy: conversion to CNF does simplification in the context of full inference, just like AC-3 does simplification in the context of backtracking search.

Resolution algorithm

Recall: relationship between entailment and contradiction (basically "proof by contradiction")

$$\text{KB} \models f \quad \Longleftrightarrow \quad \text{KB} \cup \{\neg f\} \text{ is unsatisfiable}$$

Algorithm: resolution-based inference

- Add $\neg f$ into KB.
- Convert all formulas into **CNF**.
- Repeatedly apply **resolution** rule.
- Return entailment iff derive false.

- After we have converted all the formulas to CNF, we can repeatedly apply the resolution rule. But what is the final target?
- Recall that both testing for entailment and contradiction boil down to checking satisfiability. Resolution can be used to do this very thing. If we ever apply a resolution rule (e.g., to premises A and $\neg A$) and we derive false (which represents a contradiction), then the set of formulas in the knowledge base is unsatisfiable.
- If we are unable to derive false, that means the knowledge base is satisfiable because resolution is complete. However, unlike in model checking, we don't actually produce a concrete model that satisfies the KB.

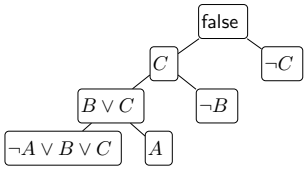
Resolution: example

$\text{KB}' = \{A \to (B \vee C), A, \neg B, \neg C\}$

Convert to CNF:

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

Repeatedly apply **resolution** rule:



Conclusion: KB **entails** f

- Here's an example of taking a knowledge base, converting it into CNF, and applying resolution. In this case, we derive false, which means that the original knowledge base was unsatisfiable.

Time complexity

Definition: modus ponens inference rule

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \to q}{q}$$

- Each rule application adds clause with **one** propositional symbol $\Rightarrow$ linear time

Definition: resolution inference rule

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

- Each rule application adds clause with **many** propositional symbols $\Rightarrow$ exponential time

- There we have it — a sound and complete inference procedure for all of propositional logic (although we didn't prove completeness). But what do we have to pay computationally for this increase?
- If we only have to apply modus ponens, each propositional symbol can only get added once, so with the appropriate algorithm (forward chaining), we can apply all necessary modus ponens rules in linear time.
- But with resolution, we can end up adding clauses with many propositional symbols, and possibly any subset of them! Therefore, this can take exponential time.

Summary

Horn clauses	**any clauses**
modus ponens	resolution
linear time	exponential time
less expressive	more expressive

- To summarize, we can either content ourselves with the limited expressivity of Horn clauses and obtain an efficient inference procedure (via modus ponens).
- If we wanted the expressivity of full propositional logic, then we need to use resolution and thus pay more.