Chapter 5
Propositional Resolution

§5.1 Introduction

Propositional resolution is an extremely powerful rule of inference for Propositional Logic. Using propositional resolution (without axiom schemata or other rules of inference), it is possible to build a theorem prover that is sound and complete for all of Propositional Logic. What's more, the search space using propositional resolution is much smaller than for standard propositional logic.

This chapter is devoted entirely to propositional resolution. We start with a look at clausal form, a variation of the language of propositional logic. We then examine the resolution rule itself. We close with some examples.

§5.2 Clausal Form

Propositional resolution works only on expressions in *clausal form*. Before the rule can be applied, the premises and conclusions must be converted to this form. Fortunately, as we shall see, there is a simple procedure for making this conversion.

A *literal* is either an atomic sentence or a negation of an atomic sentence. For example, if *p* is a logical constant, the following sentences are both literals.

$$p$$
$$\neg p$$

A *clause expression* is either a literal or a disjunction of literals. If *p* and *q* are logical constants, then the following are clause expressions.

$$p$$
$$\neg p$$
$$p \vee q$$

A *clause* is the set of literals in a clause expression. For example, the following sets are the clauses corresponding to the clause expressions above.

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

Note that the empty set {} is also a clause. It is equivalent to an empty disjunction and, therefore, is unsatisfiable. As we shall see, it is a particularly important special case.

As mentioned earlier, there is a simple procedure for converting an arbitrary set of Propositional Logic sentences to an equivalent set of clauses. The conversion rules are summarized below and should be applied in order.

1. Implications (I):

$$\begin{array}{lcl} \varphi_1 \Rightarrow \varphi_2 & \rightarrow & \neg\varphi_1 \vee \varphi_2 \\ \varphi_1 \Leftarrow \varphi_2 & \rightarrow & \varphi_1 \vee \neg\varphi_2 \\ \varphi_1 \Leftrightarrow \varphi_2 & \rightarrow & (\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2) \end{array}$$

2. Negations (N):

$$\begin{array}{lcl} \neg\neg\varphi & \rightarrow & \varphi \\ \neg(\varphi_1 \wedge \varphi_2) & \rightarrow & \neg\varphi_1 \vee \neg\varphi_2 \\ \neg(\varphi_1 \vee \varphi_2) & \rightarrow & \neg\varphi_1 \wedge \neg\varphi_2 \end{array}$$

3. Distribution (D):

$$\begin{array}{lcl} \varphi_1 \vee (\varphi_2 \wedge \varphi_3) & \rightarrow & (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3) \\ (\varphi_1 \wedge \varphi_2) \vee \varphi_3 & \rightarrow & (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3) \\ (\varphi_1 \vee \varphi_2) \vee \varphi_3 & \rightarrow & \varphi_1 \vee (\varphi_2 \vee \varphi_3) \\ (\varphi_1 \wedge \varphi_2) \wedge \varphi_3 & \rightarrow & \varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \end{array}$$

4. Operators (O):

$$\begin{array}{lcl} \varphi_1 \vee \ldots \vee \varphi_n & \rightarrow & \{\varphi_1, \ldots, \varphi_n\} \\ \varphi_1 \wedge \ldots \wedge \varphi_n & \rightarrow & \{\varphi_1\} \ldots \{\varphi_n\} \end{array}$$

As an example, consider the job of converting the sentence $(g \wedge (r \Rightarrow f))$ to clausal form. The conversion process is shown below.

$$\begin{array}{ll} & g \wedge (r \Rightarrow f) \\ \mathrm{I} & g \wedge (\neg r \vee f) \\ \mathrm{N} & g \wedge (\neg r \vee f) \\ \mathrm{D} & g \wedge (\neg r \vee f) \\ \mathrm{O} & \{g\} \\ & \{\neg r, f\} \end{array}$$

As a slightly more complicated case, consider the following conversion. We start with the same sentence except that, in this case, it is negated.

$$\begin{array}{ll}
 & \neg(g \wedge (r \Rightarrow f)) \\
\text{I} & \neg(g \wedge (\neg r \vee f)) \\
\text{N} & \neg g \vee \neg(\neg r \vee f)) \\
 & \neg g \vee (\neg\neg r \wedge \neg f) \\
 & \neg g \vee (r \wedge \neg f) \\
\text{D} & (\neg g \vee r) \wedge (\neg g \vee \neg f) \\
\text{O} & \{\neg g, r\} \\
 & \{\neg g, \neg f\}
\end{array}$$

Note that, even though the sentences in these two examples are similar to start with (disagreeing on just one ¬ operator), the results are quite different.

§5.3 Propositional Resolution

The idea of resolution is simple. Suppose we know that *p* is true or *q* is true, and suppose we also know that *p* is false or *r* is true. One clause contains *p*, and the other contains ¬*p*. If *p* is false, then by the first clause *q* must be true. If *p* is true, then, by the second clause, *r* must be true. Since *p* must be either true or false, then it must be the case that *q* is true or *r* is true. In other words, we can cancel the *p* literals.

$$\begin{array}{l}
\{p, q\} \\
\underline{\{\neg p, r\}} \\
\{q, r\}
\end{array}$$

More generally, given a clause containing a literal χ and another clause containing the literal $\neg\chi$, we can infer the clause consisting of all the literals of both clauses without the complementary pair. This rule of inference is called *propositional resolution*.

$$\begin{array}{l}
\{\varphi_1, \ldots, \chi, \ldots, \varphi_m\} \\
\underline{\{\psi_1, \ldots, \neg\chi, \ldots, \psi_n\}} \\
\{\varphi_1, \ldots, \varphi_m, \psi_1, \ldots, \psi_n\}
\end{array}$$

Since clauses are sets, there cannot be two occurrences of any literal in a clause. Therefore, in drawing a conclusion from two clauses that share a literal, we merge the two occurrences into one, as in the following example.

$$\begin{array}{l}
\{\neg p, q\} \\
\underline{\{p, q\}} \\
\{q\}
\end{array}$$

If either of the clauses is a singleton set, we see that the number of literals in the result is less than the number of literals in the other clause. From the clause $\{\neg p,q\}$ and the singleton clause $\{p\}$, we can derive the singleton clause $\{q\}$. Note the correspondence between this deduction and that of Modus Ponens.

$$\begin{array}{l} \{\neg p, q\} \\ \{p\} \\ \hline \{q\} \end{array}$$

Resolving two singleton clauses leads to the *empty clause*; i.e. the clause consisting of no literals at all, as shown below. The derivation of the empty clause means that the database contains a contradiction.

$$\begin{array}{l} \{p\} \\ \{\neg p\} \\ \hline \{\,\} \end{array}$$

If two clauses resolve, they may have more than one resolvent because there may be more than one way in which to choose the resolvents. Consider the following deductions.

$$\begin{array}{l} \{p, q\} \\ \{\neg p, \neg q\} \\ \hline \{p, \neg p\} \\ \{q, \neg q\} \end{array}$$

Note that in resolving two clauses, only *one* pair of literals may be resolved at a time, even though there are multiple resolvable pairs. For example, the following is *not* a legal application of propositional resolution.

$$\begin{array}{ll} \{p, q\} & \\ \{\neg p, \neg q\} & \text{Wrong!} \\ \cline{1-1} \{\,\} & \end{array}$$

Here are some common cases of resolution in action. The resolution derivations are shown on the right, and the corresponding natural deductions are shown on the left.

$$\begin{array}{lcl} p \Rightarrow q & \qquad & \{\neg p, q\} \\ p & & \{p\} \\ \cline{1-1} \cline{3-3} q & & \{q\} \end{array}$$

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

$p \Rightarrow q$	$\{\neg p, q\}$
$q \Rightarrow r$	$\{\neg q, r\}$
$p \Rightarrow r$	$\{\neg p, r\}$

§5.4 Applications

In order to determine whether a set of clauses is unsatisfiable, we look for a resolution proof of the empty clause from the set of premises.

1.	$\{p, q\}$	Premise
2.	$\{\neg p, q\}$	Premise
3.	$\{p, \neg q\}$	Premise
4.	$\{\neg p, \neg q\}$	Premise
5.	$\{q\}$	1,2
6.	$\{\neg q\}$	3,4
7.	$\{\}$	5,6

Validity checking is a little more difficult. The problem is that propositional resolution is not *generatively* complete, i.e. we cannot directly derive all consequences from a set of premises. For example, we cannot derive $(p \Rightarrow (q \Rightarrow p))$ directly using propositional resolution. There are no premises. Consequently, there are no conclusions.

This apparent problem disappears if we take the clausal form of the premises (if any) together with the negated goal and try to derive the empty clause. The conversion to clausal form is shown below. From here, it is easy to see the result; the $\{p\}$ and $\{\neg$p$\}$ clauses resolve to produce the empty clause in a single step.

	$\neg(p \Rightarrow (q \Rightarrow p))$
I	$\neg(\neg p \vee \neg q \vee p)$
N	$\neg\neg p \wedge \neg\neg q \wedge \neg p$
	$p \wedge q \wedge \neg p$
D	$p \wedge q \wedge \neg p$
O	$\{p\}$
	$\{q\}$
	$\{\neg p\}$

To determine whether a set Δ of sentences logically entails a sentence φ, rewrite Δ∪{¬φ} in clausal form and try to derive the empty clause.

Suppose, for example we began with the premises $(p \Rightarrow q)$ and $(q \Rightarrow r)$ and we want to prove $(p \Rightarrow r)$. To do this, we add the negation of the conclusion, i.e. $\neg(p \Rightarrow r)$ and derive the empty clause.

1.	$\{\neg p, q\}$	Premise
2.	$\{\neg q, r\}$	Premise
3.	$\{p\}$	Negated Goal
4.	$\{\neg r\}$	Negated Goal
5.	$\{q\}$	3,1
6.	$\{r\}$	5,2
7.	$\{\}$	6,4

As another example, consider the case of Mary and Pat and Quincy introduced in an earlier chapter. We know that, if Mary loves Pat, then Mary loves Quincy. We also know that, if it is Monday, then Mary loves Pat or Quincy. Our job is to prove that, if it is Monday, then Mary loves Quincy. The proof goes as follows.

1.	$\{\neg p, q\}$	Premise
2.	$\{\neg m, p, q\}$	Premise
3.	$\{m\}$	Negated Goal
4.	$\{\neg q\}$	Negated Goal
5.	$\{p, q\}$	3,2
6.	$\{q\}$	5,1
7.	$\{\}$	6,4

§5.5 Resolution Provability

A sentence φ is *provable* from a set of sentences Δ by propositional resolution (written Δ |- φ) if and only if there is a derivation of the empty clause from the clausal from of Δ∪{¬φ}.

Resolution Theorem: Propositional Resolution is sound and complete, i.e. Δ |= φ if and only if Δ |- φ.

One nice feature of propositional resolution *vis-a-vis* the more general proof method described in the preceding chapter is that propositional resolution always terminates. We simply search the resolution graph in breadth-first fashion. Since there are only finitely many clauses that can be constructed from a finite set of logical constants, the procedure will eventually run out of new conclusions to draw.

Exercises

1. *Propositional Resolution.* Use propositional resolution to prove the following sentence.

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

2. *Propositional Resolution.* Use propositional resolution to show that the following sets of clauses are unsatisfiable.
   (a) $\{p,q\}$, $\{\neg p,\mathrm{r}\}$, $\{\neg p,\neg r\}$,$\{p, \neg q\}$
   (b) $\{p,q, \neg r,s\}$, $\{\neg p,r,s\}$, $\{\neg q, \neg r\}$, $\{p, \neg s\}$, $\{\neg p, \neg r\}$, $\{r\}$

3. *Formalization and Proof.* Heads, I win. Tails, you lose. Use propositional resolution to prove that I always win.

4. *Formalization and Proof.* There are three suspects for a murder: Adams, Brown, and Clark. Adams says“I didn't do it. The victim was old acquaintance of Brown's. But Clark hated him.” Brown states “I didn't do it. I didn't know the guy. Besides I was out of town all the week.” Clark says“I didn't do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it.” Assume that the two innocent men are telling the truth, but that the guilty man might not be. Write out the facts as sentences in Propositional Logic, and use propositional resolution to solve the crime.