Resolution: Propositional Logic

Definition: (Literal) A literal is either an atomic formula or the negation of an atomic formula.

When a literal is an atomic formula, we will say that it is a *positive* literal. And when a literal is the negation of an atomic formula, then we will say it is a *negative* literal.

Definition: (Clause) A clause is a disjunction of $n \geq 0$ literals.

When a clause is a disjunction of n literals, $L_1, \ldots, L_n$ then we will represent it as $[L_1, \ldots, L_n]$. The empty clause ($n = 0$) is thus represented as [ ].

Definition: (Horn Clause) A clause is said to be a Horn clause if it has at most one positive literal.

Note that in prolog we use Horn clauses only. A *rule* can be thought of as a Horn clause with one positive literal (the head of the rule) and $n \geq 1$ negative literals. A *fact*, on the other hand, is one with one positive literal an no negative literals. A *query* is one with 0 positive literals but has one or more negative literals.

Definition: (Clausal Form) Any set of clauses is said to be a clausal form representation.

Since a set of formulae can be thought of as a conjunction of the formulae it contains, a clausal form representation is an alternate representation for a formula in *conjunctive normal form*.

Definition Let L be a clause. Then

$$\overline{L} = \begin{cases} \neg A & \text{if } L = A\text{, where } A \text{ is an atomic formula} \\ A & \text{if } L = \neg A \end{cases}$$

Definition Let C_1 be a clause containing a literal, L, and C_2 be a clause containing the literal $\overline{L}$. Let $C' = C_1 - \{L\}$ and $C'' = C_2 - \{\overline{L}\}$. Then we say the clause $R = C' \bigcup C''$ is said to be obtained by resolving C_1 and C_2 (on the literal L). We will say that R is the resolvent of C_1 and C_2.

Definition A derivation of the empty clause (or proof) from a clause set F is a sequence of clauses $C_1, C_2, \ldots, C_m$ such that

- C_m is the empty clause, and
- Each clause C_i $(1 \leq i \leq m)$ is either a clause in F or obtained by resolving two clauses C_j, C_k with $1 \leq j, k < i$.

An *atomic formula* is a statement letter. A *literal* is either an atomic formula (positive literal) or the negation of an atomic formula (negative literal). A *clause* contains literals and is interpreted as their disjunction. If a clause contains the literals $l_1, \ldots, l_k$ then we will write it as $[l_1, \ldots, l_k]$ rather than the notation $\{l_1, \ldots, l_k\}$. Thus, $[l_1, \ldots, l_k]$ is interpreted the same way as the formula $(l_1 \vee \ldots \vee l_k)$.

Here is a (*non-deterministic*) algorithm to convert a formula, $\mathcal{B}$, (assumed to not include $\Leftrightarrow$) into a set of clauses. Start with $\{[\mathcal{B}]\}$.

At any point, we will have a set that has the form, $\{C_1, \ldots, C_i, \ldots, C_k\}$, where the $C's$ have the form $[\mathcal{B}_1, \ldots, \mathcal{B}_n]$, and the $\mathcal{B}'s$ are formulae. We are finished with a C if every formula in C is already a literal.

While not done do (i.e., some C has a non-literal)

Let C_i include a non-literal.

WLOG, we can express C_i as $[\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{B}_n]$ where $\mathcal{B}_n$ is a non-literal.

Case 1: $\mathcal{B}_n = \neg\neg\mathcal{D}$, for some formula $\mathcal{D}_2$.

Replace C_i by $[\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{D}]$.

Case 2: (disjunctive case).

Case 2a: $\mathcal{B}_n = (\mathcal{D}_1 \vee \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by $[\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{D}_1, \mathcal{D}_2]$.

Case 2b: $\mathcal{B}_n = (\mathcal{D}_1 \Rightarrow \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by $[\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \neg\mathcal{D}_1, \mathcal{D}_2]$.

Case 2c: $\mathcal{B}_n = \neg(\mathcal{D}_1 \wedge \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by $[\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \neg\mathcal{D}_1, \neg\mathcal{D}_2]$.

Case 3: (conjunctive case).

Case 3a: $\mathcal{B}_n = (\mathcal{D}_1 \wedge \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by C_i^1 and C_i^2, where

$C_i^1 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{D}_1]$, and

$C_i^2 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{D}_2]$

Case 3b: $\mathcal{B}_n = \neg(\mathcal{D}_1 \Rightarrow \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by C_i^1 and C_i^2, where

$C_i^1 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \mathcal{D}_1]$, and

$C_i^2 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \neg\mathcal{D}_2]$

Case 3c: $\mathcal{B}_n = \neg(\mathcal{D}_1 \vee \mathcal{D}_2)$ for some $\mathcal{D}_1, \mathcal{D}_2$.

Replace C_i by C_i^1 and C_i^2, where

$C_i^1 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \neg\mathcal{D}_1]$, and

$C_i^2 = [\mathcal{B}_1, \ldots \mathcal{B}_{n-1}, \neg\mathcal{D}_2]$

Example: Converting $\mathcal{B} = (\neg A \vee B) \Rightarrow (C \vee A)$ into a set of clauses:

$\{[(\neg A \vee B) \Rightarrow (C \vee A)]\}$ (starting with $\{[\mathcal{B}]\}$)

$\{[\neg(\neg A \vee B), (C \vee A)]\}$ (applying rule 2b)

$\{[\neg(\neg A \vee B), C, A]\}$ (applying rule 2a)

$\{[\neg\neg A, C, A], [\neg B, C, A]\}$ (applying rule 3c)

$\{[A, C], [A \neg B, C]\}$ (applying rule 1) Note we write $[A, C]$ rather than $[A, C, A]$ and that $[\neg B, C, A]$ can also be written as $[A, \neg B, C]$.