Definite Clauses and Horn Clauses in Logic

1 Definite Clauses

A **definite clause** is a disjunction of literals with exactly one positive literal. It can be written as:

$$(\neg L_{1,1} \lor \neg \text{Breeze} \lor B_{1,1})$$

This is an example of a definite clause, as there is exactly one positive literal, $B_{1,1}$.

However, the following is **not** a definite clause:

$$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$$

This clause contains more than one positive literal and, therefore, is not considered a definite clause.

2 Horn Clauses

A **Horn clause** is a disjunction of literals with at most one positive literal. All definite clauses are Horn clauses, but not all Horn clauses are definite clauses. For instance:

- Clauses with zero positive literals are also Horn clauses and are often referred to as *goal clauses*.

3 Examples

- The clause:
  $$(\neg L_{1,1} \lor \neg \text{Breeze} \lor B_{1,1})$$
  is both a definite clause and a Horn clause.
- The clause:
  $$(A \lor \neg B)$$
  is a goal clause and a Horn clause, but it is not a definite clause.

4 Properties of Horn Clauses

Horn clauses are closed under resolution. This means that if you resolve two Horn clauses, the result will also be a Horn clause. This property makes them particularly useful in inference algorithms.

5 Importance of Definite Clauses in Knowledge Bases

- Every definite clause can be written as an implication where the premise is a conjunction of positive literals and the conclusion is a single positive literal.
- For example:

$$(\neg L_{1,1} \vee \neg \text{Breeze} \vee B_{1,1}) \implies (L_{1,1} \wedge \text{Breeze}) \Rightarrow B_{1,1}$$

  Here, $L_{1,1} \wedge \text{Breeze}$ is the **premise**, and $B_{1,1}$ is the **conclusion**.

6 Inference with Horn Clauses

Inference with Horn clauses uses two primary algorithms:

- **Forward-Chaining Algorithm:** This algorithm deduces new facts from known facts by applying rules repeatedly.
- **Backward-Chaining Algorithm:** This algorithm works backward from the goal, attempting to prove it by breaking it down into smaller subgoals.

These inference methods form the basis for logic programming and reasoning.

7 Efficiency of Entailment with Horn Clauses

The process of checking entailment with Horn clauses can be done efficiently in time complexity:

$$O(\text{size of the knowledge base})$$