Logic Programming (end of Section 8.3)
A logic program is a set of clauses with the restriction that there is exactly one positive literal in each clause. Such clauses are often called definite clauses.

Example: Let $p(x, y)$ mean $x$ is a parent of $y$ and let $g(x, y)$ mean $x$ is a grandparent of $y$. Here are some ways to represent a definition of the grandparent relation:

First-order logic: $\forall x \forall y \forall z \,(p(x, z) \land p(x, y) \to g(x, y))$
First-order clause: $g(x, y) \lor \neg p(x, z) \lor \neg p(x, y)$
Logic programming: $g(x, y) \leftarrow p(x, z), p(z, y)$
Prolog: $g(X, Y) : \neg p(X, Z), p(Z, Y)$. 
A query or goal is a question that asks whether the program infers something. The something is a sequence of one or more atoms and the question is whether there is a substitution that can be applied to the atoms so that the resulting atoms are inferred by the program.

**Example:** Suppose we have the following little logic program:

- \( p(a, b) \).
- \( p(b, d) \).
- \( g(x, y) \leftarrow p(x, z), p(z, y) \).

Let \( g(a, w) \) be a query. It asks whether \( a \) has a grandchild. If we let \( \theta = \{w/d\} \), then \( g(a, w)\theta = g(a, d) \), which says \( a \) has a grandchild \( d \). This follows from the two program facts \( p(a, b) \) and \( p(b, d) \) and the definition of \( g \). So \( g(a, d) \) is inferred by the program.
Logic Programming Representation of a Query

• To see whether a query can be inferred from a program, a resolution proof is attempted. The premises are the program clauses together with the negation of the query, which can be written as a clause with only negative literals.

• Example: Given the query $g(a, w), p(u, w)$. Its formal meaning is $\exists w \exists u (g(a, w) \land p(u, w))$. So we negate it and convert it to a clause. Here are some representations of the query:

  **First-order logic:**
  
  $\neg \exists w \exists u (g(a, w) \land p(u, w))$
  
  $\equiv \forall w \forall u \neg (g(a, w) \land p(u, w))$

  **First-order clause:**
  
  $\neg g(a, w) \lor \neg (u, w)$

  **Logic programming:**
  
  $\leftarrow g(a, w), p(u, w)$

  **Prolog:**
  
  $|? - g(a, W), p(U, W)$.
SLD Resolution

- **SLD-resolution** is a form of resolution used to execute logic programs. SLD means selective linear resolution of definite clauses. Select the leftmost atom of the goal; Linear (each resolvant depends on the previous resolvant) and Definite clauses are the clauses of a logic program.

- **Example:** Given the following little logic program and a query, where the clauses are also listed in first-order form:

  \[
  \text{Logic Programming Syntax} \quad \text{First-Order Clauses}
  \]
  \[
  p(a, b). \\
  p(b, d). \\
  g(x, y) \leftarrow p(x, z), p(z, y).
  \]
  \[
  p(a, b). \\
  p(b, d). \\
  g(x, y) \lor \neg p(x, z) \lor \neg p(z, y).
  \]
  
  The query:
  \[
  \leftarrow g(a, w), p(u, w). \quad \neg g(a, w) \lor \neg p(u, w).
  \]
Resolution Proof

The resolution proof:

**Logic Programming Syntax**
1. \( p(a, b) \)
2. \( p(b, d) \)
3. \( g(x, y) \leftarrow p(x, z), p(z, y) \)
4. \( g(a, w), p(u, w) \)
5. \( p(a, z), p(z, y), p(u, y) \)
6. \( p(b, y), p(u, y) \)
7. \( p(u, d) \)
8. \[ \]

**First-Order Clauses**
1. \( p(a, b) \)
2. \( p(b, d) \)
3. \( g(x, y) \lor \neg p(x, z) \lor \neg p(z, y) \)
4. \( \neg g(a, w) \lor \neg p(u, w) \)
5. \( \neg p(a, z) \lor \neg p(z, y) \lor \neg p(u, y) \)
6. \( \neg p(b, y) \lor \neg p(u, y) \)
7. \( \neg p(u, d) \)
8. \[ \]

QED