Another Paradigm

**QUESTION:** "What is a computer program?"

**ANSWER:**
"It is an executable representation of some algorithm designed to solve some real world problem."

- Kowalski (CACM, 1979):
  - There are two key elements to a computer program
  - Logic – what we want the program to achieve
  - Control – how we are going to achieve it

\[ \text{ALGORITHM} = \text{LOGIC} + \text{CONTROL} \]

- Difference between imperative and declarative programming
Example: Computing Factorial

Imperative (Ada) implementation

```ada
with CS_IO; use CS_IO;

procedure FACTORIAL is
    N: integer;  
    T: integer := 1;
begin
    put("Input "); get(N);
    for K in 2..N loop
        T := T*K;
    end loop;
    put("Factorial "); put(N); newlines;
    put("is "); put(T); newline;
end FACTORIAL;
```

For imperative language programmer needs to concentrate on both the logic and control.

Declarative (PROLOG) implementation

```prolog
factorial(0,1): - .
factorial(N1,T2): -
    N2 is N1-1,
    factorial(N2,T1),
    T2 is N1*T1.
```

For declarative language, we define the facts (the desired goal or result) but not the controls (how we achieve the desired goal).

Declarative Languages

- Declarative languages make extensive use of relationships between objects.
- There are two principal styles of defining relationships:
  1. Functional Programming
     - Relationships are expressed using functions.
     - The square function expresses the relationship between the input n and the output value n*n.
   ```prolog
   square(N, M):-
   M is N*M.
   ```
   - Clauses can be used to express both facts and rules.

What is logic?

- Encyclopedia Britannica:
  - Logic is the study of propositions and their use in argumentation.

- Encarta Encyclopedia:
  - Logic is the science dealing with the principles of valid reasoning and argument.

- Factasia Logic:
  - Logic is the study of necessary truths and systematic methods for expressing and demonstrating such truths.
Logic Programming

- Logic programming
  - expresses programs in the form of symbolic logic
  - uses a logical inferencing process for reasoning
- Logic programs are declarative rather than procedural
  - Programs do not state exactly how a result is to be computed but rather describe the form of the result
  - It is assumed that the computer can determine how the result is to be obtained
  - One needs to provide the computer with the relevant information and a method of inference for computing desirable results
- Programming languages based on symbolic logic are called logic programming languages
  - Prolog is the most widely used logic programming language

Terminology

- Proposition
  - a logical statement that may be true or false
- Symbolic logic is used for three purposes:
  - express propositions
  - express the relationships between propositions
  - describe how new propositions may be inferred from others
- Two primary forms of symbolic logic
  - Propositional calculus
  - Predicate calculus
    - Predicate calculus is the form of symbolic logic used for logic programming

Propositions

- Objects in logic programming propositions are
  - Constants
    - symbols that represent an object
    - Example: man, jake, like bob, and steak
  - Variables
    - symbols that can represent different objects at different times
- Atomic propositions are the simplest propositions and consist of compound terms
Atomic Propositions

- Compound term has two parts
- Functor: symbol that names the relation
- An ordered list of parameters
- Examples:
  - man (jake)
  - like (bob, steak)
- Compound term with single parameter called a 1-tuple
- Compound term with two params is called a 2-tuple, etc.
- These propositions have no intrinsic semantics
- Father (john, jake) could mean several things
- Propositions are stated in two modes
  - Fact: one in which the proposition is defined to be true
  - Query: one in which the truth of the proposition is to be determined

Compound Propositions

- Compound propositions have two or more atomic propositions connected by logical operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>$\neg$</td>
<td>$\neg a$</td>
<td>not $a$</td>
</tr>
<tr>
<td>Conjunction</td>
<td>$\land$</td>
<td>$a \land b$</td>
<td>$a$ and $b$</td>
</tr>
<tr>
<td>Disjunction</td>
<td>$\lor$</td>
<td>$a \lor b$</td>
<td>$a$ or $b$</td>
</tr>
<tr>
<td>Equivalence</td>
<td>$\iff$</td>
<td>$a \iff b$</td>
<td>$a$ is eqv to $b$</td>
</tr>
<tr>
<td>Implication</td>
<td>$\implies$</td>
<td>$a \implies b$</td>
<td>$a$ implies $b$</td>
</tr>
<tr>
<td>(In Prolog $a \implies b$ is written as $a :- b$)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compound proposition examples:

- $a \land b \land c$
- $a \lor b \lor d$ equivalent to $(a \lor (\neg b)) \lor d$

Precedence of logical connectors:

- $\neg$ highest precedence
- $\land$, $\lor$, $\implies$ next
- $\land$, $\lor$ lowest precedence
Compound Proposition

- Implication: p \implies q
- Meaning:
  - if p then q
  - p implies q
  - p is the premise or antecedent
  - q is the conclusion or consequent
- Can write p \implies q in disjunctive normal form
  - ? p OR q
- Truth table shows equivalence

\[ \begin{array}{c|c|c|c|c}
 p & q & ?p & p \implies q & p OR q \\
\hline
 1 & 1 & 0 & 1 & 1 \\
 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 1 & 1 \\
 0 & 0 & 1 & 1 & 1 \\
\end{array} \]

Propositions with Quantifiers

Variables may appear in propositions - only when introduced by symbols called quantifiers

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>universal</td>
<td>? X, P</td>
<td>For all X, P is true</td>
</tr>
<tr>
<td>existential</td>
<td>? X, P</td>
<td>There exists a value of X such that P is true</td>
</tr>
</tbody>
</table>

Note: the period separates the variable from the proposition
Propositions with Quantifiers

Examples of propositions with quantifiers

\[ \forall X \left( \text{woman}(X) \rightarrow \text{human}(X) \right) \]
For any value of \( X \), if \( X \) is a woman, then \( X \) is human

\[ \exists X \left( \text{mother}(\text{mary}, X) \rightarrow \text{male}(X) \right) \]
There exists a value of \( X \), such that \( \text{mary} \) is the mother of \( X \) and \( X \) is a male

Note: quantifiers have a higher precedence than any of the logical operators

First Order Predicate Calculus

Provides a method of expressing collections of propositions
Collection of propositions can be used to determine whether any interesting or useful facts can be inferred from them

0 is a natural number.
2 is a natural number.
For all \( X \), if \( X \) is a natural number, then so is the successor of \( X \).
-1 is a natural number

Predicate calculus:

\[ \text{natural}(0) \]
\[ \forall X \left( \text{natural}(X) \right) \]
\[ \text{natural}(\text{successor}(X)) \]

First-Order Predicate Calculus

A horse is a mammal
A human is a mammal
Mammals have four legs and no arms, or two legs and two arms
A horse has no arms
A human has no legs

mammal(horse)
mammal(human)
\[ \forall X \left( \text{mammal}(X) \rightarrow \left( \text{legs}(X,4) \lor \text{arms}(X,0) \right) \right) \]
\[ \text{legs}(X,2) \rightarrow \text{arms}(X,2) \]
arms(horse, 0)
legs(human, 0)
Clausal Form

- Redundancy is a problem with predicate calculus
- there are many different ways of stating propositions that have the same meaning
  - Example: \( p \land q \equiv p \lor q \equiv \neg (p \land q) \equiv \neg p \lor \neg q \)
- not a problem for logicians but for computerized system, redundancy is a problem

- Clausal form is one standard form of propositions used for simplification and has the syntax:
  \[
  B_1 \lor B_2 \lor ... \lor B_n \land A_1 \land A_2 \land ... \land A_m
  \]
  Meaning: If all \( A \)'s are true, then at least one B is true

Characteristics

- Existential quantifiers are not required
- Universal quantifiers are implicit in the use of variable in the atomic propositions
- Only the conjunction and disjunction operators are required
- Disjunction appears on the left side of the clausal form and conjunction on the right side
- The left side is called the consequent
- The right side is called the antecedent

Clausal Form Examples

\[
\text{likes (bob, trout)} \land \text{likes (bob, fish)} \land \text{fish (trout)}
\]
Meaning: if bob likes fish and a trout is a fish, then bob likes trout

\[
\text{father(louis, al)} \land \text{father(louis, violet)} \land \text{father(al, bob)} \land \text{mother(violet, bob)} \land \text{grandfather(louis, bob)}
\]
Meaning: if al is bob's father and violet is bob's mother and louis is bob's grandfather, then louis is either al's father or violet's father
Theorems

- One of the most significant breakthroughs in automatic theorem-proving was the discovery of the resolution principle by Robinson in 1965.
- Resolution is an inference rule that allows inferred propositions to be computed from given propositions.
- Resolution was devised to be applied to propositions in clausal form.

Resolution

- The concept of resolution is the following:
  Given two propositions:
  \[ P_1 \land P_2 \]
  \[ Q_1 \land Q_2 \]
  Suppose \( P_1 \) is identical to \( Q_2 \) and we rename them as \( T \).
  Then
  \[ T \land P_2 \]
  \[ Q_1 \land T \]
  Resolution:
  Since \( P_2 \) implies \( T \) and \( T \) implies \( Q_1 \), it is logically obvious that \( P_2 \) implies \( Q_1 \).

Resolution Example

- Consider the two propositions:
  older (Joanne, Jake) \land mother (Joanne, Jake)
  wiser (Joanne, Jake) \land older (Joanne, Jake)

- Mechanics of Resolution:
  1. Terms on the left hand side are ANDed together
  2. Terms on the right hand side are ANDed together
  3. Any term that appears on both sides of the new proposition is removed from both sides

wiser (Joanne, Jake) \land mother (Joanne, Jake)
Resolution Expanded Example

- Given
  father(bob, jake) ? mother(bob, jake) ? parent(bob, jake)
  grandfather(bob, fred) ? father(bob, jake) ?
  father(jake, fred)

- The resolution process cancels the common term on both the left and right sides
  mother(bob, jake) ? grandfather(bob, fred) ?
  parent(bob, jake) ?
  father(jake, fred)

Resolution for Variables

- Presence of variables in propositions requires resolution to find values for those variables that allow the matching process to succeed

- Unification
  - The process of determining useful values for variables in propositions to find values for variables that allow the resolution process to succeed.

- Instantiation
  - The temporary assigning of values to variables to allow unification

Example

- Add facts -- what is known --
  parent(sue, tom).
  parent(bob, tom).
  parent(bob, kate).
  parent(tom, laura).
  parent(tom, mark).
  parent(mark, anne).

- Query:
  ? parent(tom, X).
  X = laura ;
  X = mark ;
Resolution for Variables

- Inconsistency detection
  - An important property of resolution is its ability to detect any inconsistency in a given set of propositions
  - This property allows resolution to be used to prove theorems

- Theorem proving
  - Use negation of the theorem as a new proposition
  - Theorem is negated so that resolution can be used to prove the theorem by finding an inconsistency
  - This is the basis for proof by contradiction

Example

- Facts:
  - A
  - B \rightarrow A

- Given the facts, prove that B is true
- Query: ?????(add new proposition)
- Resolution:
  - A \rightarrow B \rightarrow A
  - B
  - Contradicts with ?????
  - So, conclude that ????? is false
  - Therefore, B is true

Horn Clauses

- Recall that a clausal form has the following form:
  - B1 \rightarrow B2 \rightarrow \ldots \rightarrow Bn \rightarrow A1 \rightarrow A2 \rightarrow \ldots \rightarrow Am

- When propositions are used for resolution, only a restricted kind of clausal form can be used

- Horn clauses
  - special kind of clausal form to simplify resolution
  - two forms:
    - single atomic proposition on the left side, or
    - an empty left side
  - left side of Horn clause is called the head
  - Horn clauses with left sides are called headed Horn clauses
Horn Clauses

- Headed Horn clauses are used to state relationships:
  - likes(bob, trout) \(\Rightarrow\) likes (bob, fish) \(\Rightarrow\) fish(trout)

- Headless Horn clauses are used to state facts:
  - father(bob,jake)

- Most propositions may be stated as Horn clauses

Semantics

- Semantics of logic programming languages are called declarative semantics
- meaning of propositions can be determined from the statements themselves
  - Unlike imperative languages where semantics of a simple assignment statement requires examination of
    - local declarations,
    - knowledge of scoping rules of the language,
    - and possibly, examination of programs in other files to determine the types of variables

Origins of Prolog

- Colmerauer and Roussle (University of Aix-Marseille) and Kowalski (University of Edinburgh) developed the fundamental design of Prolog

- Collaboration between both universities continued until mid-70s when independent efforts kicked off resulting in two syntactically different dialects of Prolog

- With Japan’s announcement of a project called Fifth Generation Computing Systems in 1981, came their choice of Prolog to develop intelligent machines
  - This results in strong sudden interest in logic programming and AI in U.S. and Europe
A Prolog term is a constant, a variable, or a structure.

- A constant is either an atom or an integer.
  - Atoms are the symbolic values of Prolog.
    - Either a string of letters, digits, and underscores that begins with a lowercase letter or a string of any printable ASCII characters delimited by apostrophes.
  - Variables are any string of letters, digits, and underscores that begins with an uppercase letter.
    - Not bound to types by declarations.
    - Binding of a value (and type) to a variable is called an instantiation.
    - Instantiations last only through completion of goal.

- Structures represent the atomic proposition of predicate calculus.
  - Form is functor (parameter list).
    - Functor can be any atom and is used to identify the structure.
    - Parameter list can be any list of atoms, variables, or other structures.

Fact statements are used to construct hypotheses from which new information may be inferred.

- Fact statements are headless Horn clauses assumed to be true.

Examples:
- male(bill).
- female(mary).
- male(jake).
- father(bill, jake).
- mother(mary, jake).

Rule statements are headed Horn clauses for constructing the database.

- The RHS is the antecedent (if), and the LHS is the consequent (then).
- Consequent is a single term because it is a Horn clause.
- Conjunctions may contain multiple terms that are separated by logical ANDs or commas, e.g.
  
  female(shelley), child {shelley}.
### Prolog - Rule Statements

- General form of the Prolog headed Horn clause
  \[
  \text{consequence}_i \leftarrow \text{antecedent}_i
  \]

- Example:
  \[
  \text{ancestor}(\text{mary}, \text{shelley}) \leftarrow \text{mother}(\text{mary}, \text{shelley}).
  \]

- Variables can be used to generalize meanings of statements
  \[
  \begin{align*}
  \text{parent}(X, Y) & :\neg \text{mother}(X, Y). \\
  \text{parent}(X, Y) & :\neg \text{father}(X, Y). \\
  \text{grandparent}(X, Z) & :\neg \text{parent}(X, Y), \text{parent}(Y, Z). \\
  \text{grandparent}(X, Z) & :\neg \text{mother}(X, Y), \text{mother}(M, Y), \\
                            & \text{father}(F, X), \text{father}(F, Y).
  \end{align*}
  \]

- These statements give rules of implication among some variables, or universal objects (universal objects are X, Y, Z, M, and F)

### Prolog - Goal Statements

- Facts and Rules are used to prove or disprove a theorem that is in the form of a proposition (called goal or query)

- Syntactic form of Prolog goal statement is identical to headless Horn clauses:
  \[
  \text{e.g. } \text{man}(\text{fred}).
  \]
  to which the system will respond yes or no

- Conjunctive propositions and propositions with variables are also legal goals. For example,
  \[
  \text{father}(X, \text{mike}).
  \]
- When variables are present, the system identifies the instantiations of the variables that make the goal true

### Prolog - Basic Elements

- Because goal and some nongoal statements have the same form (headless Horn clauses), it is imperative to distinguish between the two

- Interactive Prolog implementations do this by simply having two modes, indicated by different prompts: one for entering goals and one for entering fact and rule statements

- Gnu Prolog uses `?` for goals
Horn Clauses in Prolog

A :- B₁, . . . , Bₙ

Head
Body
End of clause marker

Rule
- The head and the body are nonempty.
- The body is the conditional part.
- The head is the conclusion.

Fact
- The body is empty, and is written as:

A

Inferencing Process of Prolog

Goals (queries) may be compound propositions; each of facts (structures) is called a subgoal
- Father(X, Y), Likes(X, steak).

The inferencing process must find a chain of inference rules/facts in the database that connect the goal to one or more facts in the database
- If Q is the goal, then either
  - Q must be found as fact in the database, or
  - the inferencing process must find a sequence of propositions that give that result

Matching
- the process of proving (or satisfying) a subgoal by a proposition-matching process

Consider the goal or query: man(bob).
If the database includes the fact man(bob), the proof is trivial.
If the database contains the following fact and inference:
father (bob).
man (X) :- father (X)
Prolog would need to find these and infer the truth. This requires unification to instantiate X temporarily to bob.
Inferencing Process of Prolog

- Consider the goal: man(X).
  - Prolog must match the goal against the propositions in the database.
  - The first proposition that it finds that has the form of the goal, with an object as its parameter, will cause X to be instantiated with that object's value and this result displayed.
- If there is no proposition with the form of the goal, the system indicates with a no that the goal can't be satisfied.

Solution Search Approaches

- Depth-first
  - finds a complete sequence of propositions - a proof - for the first subgoal before working on the others.
- Breadth-first
  - works on all subgoals of a given goal in parallel.
- Backtracking
  - when the system finds a subgoal it cannot prove, it reconsiders the previous one to attempt to find an alternative solution and then continue the search.
  - multiple solutions to a subgoal result from different instantiations of its variables.

Inferencing Process – Backtracking

- Suppose we have the following compound goal:
  - male (X), parent (X, shelley)
  - Is there an instantiation of X, such that X is a male and X is a parent of shelley?
- The search
  - Prolog finds the first fact in the database with male as its functor; it then instantiates X to the parameter of the found fact, say john.
  - Then it attempts to prove that parent(john, shelley) is true.
  - If it fails, it backtracks to the first subgoal, male(X) and attempts to satisfy it with another alternative to X.
  - More efficient processing possible.