Topics

- The Basic Elements of Prolog
  - Constant, variable, and structure
  - Fact, rule and goal statements
  - Bottom-up resolution and top-down resolution
  - Depth-first and breath-first search
  - Backtracking
  - Tracing

The Origins of Prolog

- University of Aix-Marseille (Calmerauer & Roussel)
  - Natural language processing
- University of Edinburgh (Kowalski)
  - Automated theorem proving

Terms

- This book uses the Edinburgh syntax of Prolog
- *Term*: a constant, variable, or structure
- *Constant*: an atom or an integer
- *Atom*: symbolic value of Prolog
- Atom consists of either:
  - a string of letters, digits, and underscores beginning with a lowercase letter
  - a string of printable ASCII characters delimited by apostrophes

Terms: Variables and Structures

- *Variable*: any string of letters, digits, and underscores beginning with an uppercase letter
- *Instantiation*: binding of a variable to a value
  - Lasts only as long as it takes to satisfy one complete goal
- *Structure*: represents atomic proposition

  functor (*parameter list*)

Fact Statements

- Used for the hypotheses
- Headless Horn clauses

```
female(shelley).
male(bill).
father(bill, jake).
```

Rule Statements

- Used for the hypotheses
- Headed Horn clause
- Right side: *antecedent* (*if* part)
  - May be single term or conjunction
- Left side: *consequent* (***then*** part)
  - Must be single term
- *Conjunction*: multiple terms separated by logical AND operations (implied)

Example Rules

```
ancestor(mary,shelley):- mother(mary,shelley).
```

- Can use variables (*universal objects*) to generalize meaning:

```
parent(X,Y):- mother(X,Y).
parent(X,Y):- father(X,Y).
grandparent(X,Z):- parent(X,Y), parent(Y,Z).
```

Goal Statements

- For theorem proving, theorem is in form of proposition that we want system to prove or disprove – *goal statement*
- Same format as headless Horn

```
man(fred)
```

- Conjunctive propositions and propositions with variables also legal goals

```
father(X, mike)
```

Inferencing Process of Prolog

- Queries are called goals
- If a goal is a compound proposition, each of the facts is a subgoal
- To prove a goal is true, must find a chain of inference rules and/or facts. For goal Q:

```
P2 :- P1
P3 :- P2
...
Q :- Pn
```

- Process of proving a subgoal called matching, satisfying, or resolution

Approaches

- *Matching* is the process of proving a proposition
- Proving a subgoal is called *satisfying* the subgoal
- *Bottom-up resolution, forward chaining*
  - Begin with facts and rules of database and attempt to find sequence that leads to goal
  - Works well with a large set of possibly correct answers
- *Top-down resolution, backward chaining*
  - Begin with goal and attempt to find sequence that leads to some set of facts in database
  - Works well with a small set of possibly correct answers
- Prolog implementations use backward chaining

Subgoal Strategies

- When goal has more than one subgoal, can use either
  - Depth-first search: find a complete proof for the first subgoal before working on others
  - Breadth-first search: work on all subgoals in parallel
- Prolog uses depth-first search
  - Can be done with fewer computer resources

Backtracking

- With a goal with multiple subgoals, if fail to show truth of one of subgoals, reconsider previous subgoal to find an alternative solution: *backtracking*
- Begin search where previous search left off
- Can take lots of time and space because may find all possible proofs to every subgoal

Simple Arithmetic

- Prolog supports integer variables and integer arithmetic
- `is` operator: takes an arithmetic expression as right operand and variable as left operand

```
A is B / 17 + C
```

- Not the same as an assignment statement!
  - The following is illegal:

```
Sum is Sum + Number.
```

Example

```
speed(ford,100).
speed(chevy,105).
speed(dodge,95).
speed(volvo,80).
time(ford,20).
time(chevy,21).
time(dodge,24).
time(volvo,24).
distance(X,Y) :-   speed(X,Speed),
                   time(X,Time),
                   Y is Speed * Time.
```

A query: `distance(chevy, Chevy_Distance).`

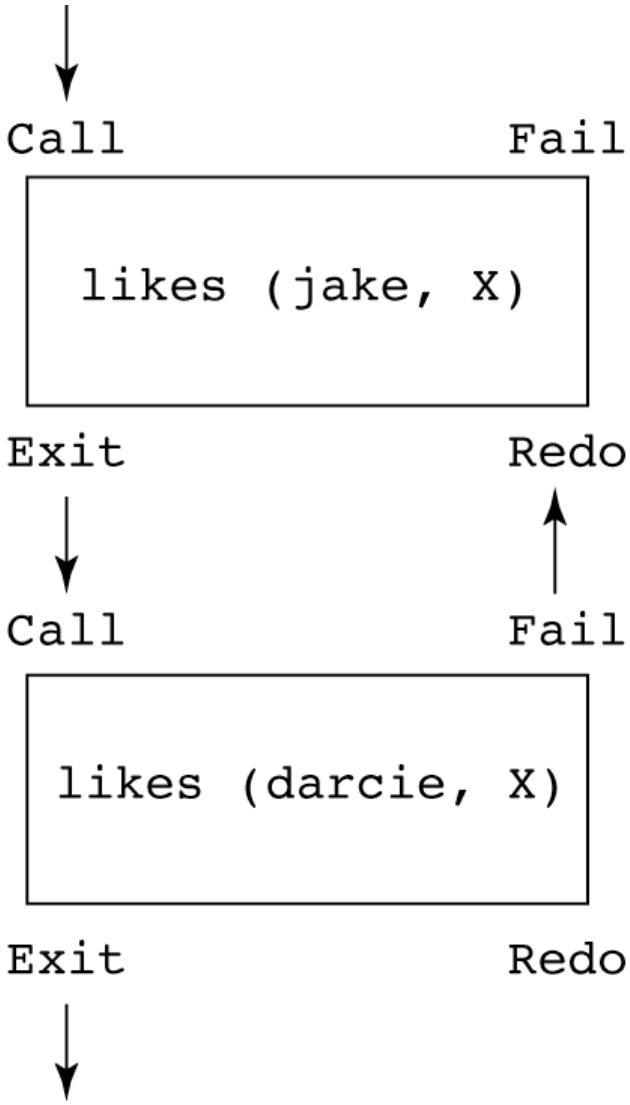
Trace

- Built-in structure that displays instantiations at each step
- *Tracing model* of execution – four events:
  - *Call* (beginning of attempt to satisfy goal)
  - *Exit* (when a goal has been satisfied)
  - *Redo* (when backtrack occurs)
  - *Fail* (when goal fails)

Example

```
likes(jake,chocolate).
likes(jake, apricots).
likes(darcie, licorice).
likes(darcie, apricots).

trace.
likes(jake, X), likes(darcie, X).
(1) 1 Call: likes(jake, _0)?
(1) 1 Exit: likes(jake, chocolate)
(2) 1 Call: likes(darcie, chocolate)?
(2) 1 Fail: likes(darcie, chocolate)
(1) 1 Redo: likes(jake, _0)?
(1) 1 Exit: likes(jake, apricots)
(3) 1 Call: likes(darcie, apricots)?
(3) 1 Exit: likes(darcie, apricots)
X = apricots
```



List Structures

- Other basic data structure (besides atomic propositions we have already seen): list
- *List* is a sequence of any number of elements
- Elements can be atoms, atomic propositions, or other terms (including other lists)

```
[apple, prune, grape, kumquat]
[]          (empty list)
[X | Y]     (head X and tail Y)
```

Append Example

```
append([], List, List).
append([Head | List_1], List_2, [Head | List_3]) :-
            append (List_1, List_2, List_3).
```

More Examples

```
reverse([], []).
reverse([Head | Tail], List) :-
  reverse (Tail, Result),
           append (Result, [Head], List).


member(Element, [Element | _]).
member(Element, [_ | List]) :-
                        member(Element, List).
```

The underscore character means an anonymous variable—it means we do not care what instantiation it might get from unification

GNU Prolog

- `ssh username@cs-parallel.ua.edu`
- **`gprolog`**

```
| ?- X is 3+5.
X = 8
| ?- append([a,b],[c,d],X).
X = [a,b,c,d]
| ?- halt.
```

- Or type in **end_of_file.** to quit
- Or type in <**CTRL-D**> to quit

Steps to use GNU Prolog

- Use a text editor to create a Prolog database, say, `first.pl` to include facts and rules
- Start Prolog by typing in `gprolog`
- Consult a database by typing in `consult('first.pl').` or simply `[first].`
- Run queries/goals

Examples

- first.pl
- cars.pl
- food.pl
- append.pl
- reverse.pl
- member.pl