CS2136: Paradigms of Computation

Class 04:
Prolog:
  Goals
Backtracking
Syntax

Copyright 2001, 2002, 2003 Michael J. Ciaraldi and David Finkel
Satisfying Goals
Variables
(same as last time)

z They can stand for objects.
z A variable is
  y “Instantiated” if it currently stands for an object.
  y “Not instantiated” if it doesn’t.
z Start with an upper-case letter.
Questions with Variables
(same as last time)

z Also known as “Goals”.
z Prolog looks for a match.
z likes(john,X).

y Logically means: “Is there anything John likes?”

y Really means: “Is there an X such that this is true?”
What It Means to Prolog

"Now if an uninstantiated variable appears in an argument, Prolog will allow that argument to match any other argument in the same position in the fact." – C&M p. 9
Example

File ‘likes.pl’:
likes(john,mary).
likes(john,gold).
likes(mary,john).
likes(john,football).
likes(john,mary).

Questions & Answers:
?- likes(john,X).
X = mary [I hit RETURN here]
Yes
?-
How It Works

- Prolog searches top-to-bottom through the database.
- When it finds a match ("satisfies the goal"), X is instantiated to mary.
- Prolog marks the place in the database.
- Prolog prints all objects that variables now stand for.
- You hit RETURN—you’re done!
Continuing Instead

?- likes(john,X).
X = mary
X = gold
X = football
X = mary
No
?-
How This Case Works

- Prolog searches top-to-bottom through the database.
- When it finds a match, X is instantiated to mary.
- Prolog marks the place in the database.
- Prolog prints all objects that variables now stand for.
- You hit semicolon—Prolog attempts to re-satisfy the goal, starting where it left off (not from the beginning).
What’s Happening

?- \texttt{likes(john,X)}. \quad \text{[At this point, X is not instantiated]}
\begin{align*}
X &= \text{mary} ; \quad \text{[X is instantiated to mary]} \\
X &= \text{gold} ; \quad \text{[X is instantiated to gold]} \\
X &= \text{football} ; \quad \text{[etc.]} \\
X &= \text{mary} ; \quad \text{[etc.]} \\
\text{No} \quad \quad \quad \quad \text{[No more matches]} \\
?-&
One More Time: Satisfy vs. Re-Satisfy

- When Prolog tries to **satisfy** a goal, it starts from the top of the database, looking for a match.

- When Prolog tries to **re-satisfy** a goal, it starts looking in the database from where it left off (i.e. just **after** the last match of that goal).
Conjunctions

- Prolog uses a comma to indicate AND.
- To get OR, for now just use separate goals or rules.
  - Could also use semicolon.
Conjunctions: AND

Do John and Mary like each other?
?- likes(john,mary), likes(mary,john).
Yes
?- likes(john,gold), likes(gold,john).
No

Prolog tries to satisfy each goal.
Each of these examples has two goals.
Satisfying Multiple Goals I

What if the multiple goals involve variables?

For the goal to be satisfied, the variable X must be instantiated to the same value in each goal.

In other words, is there one object X that is liked by both Mary and John?
Satisfying Multiple Goals II

- Prolog tries to satisfy the first goal.
- If first goal is satisfied, Prolog marks the place and tries to satisfy the second goal.
- If second goal becomes satisfied, Prolog also marks that place.
  - Each goal has its own place marker.
- What if the second goal is not satisfied?
  - Not satisfied = no match OR user says to try again.
  - Must “backtrack”.


Backtracking
Backtracking

z When the second goal fails, **uninstantiate** X, then try to **re-satisfy** the first goal.
  y Start where you left off on the first goal.

z If the first goal gets re-satisfied, try to **satisfy** the second goal.
  y **Not re-satisfy**.
  y Start matching second goal from the beginning.

z Order is very important.
  y Prolog always tries to satisfy goals left-to-right.
Backtracking Exercise

1. The first goal succeeds, instantiating X to food.
2. Next, attempt to satisfy the second goal.

3. The second goal fails.
4. Next, backtrack: forget the previous X, and attempt to re-satisfy the first goal.

5. The first goal succeeds again, instantiating X to wine.
6. Next attempt to satisfy the second goal:

7. The second goal succeeds.
8. Prolog notifies you of success, and waits for a reply.

Exercise 1.1: Continue the pencil-and-paper simulation of the example given above, assuming that you have just typed a semicolon to initiate backtracking in order to find out if John and Mary both like anything else.
Rules
Rules

z Rules are general statements about objects and their relationships.

z Example:
  y John likes everyone
  y Same as: John likes every object, if the object is a person.

z Another example:
  y John likes any woman who likes wine.
  y Same as: John likes X if X is a woman and X likes wine.
Rule Syntax

z Head :- Body.

z “:-” (colon, hyphen) pronounced “if”.

z Head is the thing that is true if the Body is true.
Rule Example

z Same example:
  y John likes any woman who likes wine.
  y Same as: John likes X if X is a woman and X likes wine.
  y likes(john,X) :- likes(X,wine),female(X).
  y Don’t forget the dot!
Scope

\[ \text{likes}(\text{john}, X) :- \text{likes}(X, \text{wine}), \text{female}(X). \]

When X becomes instantiated, all X’s are instantiated within the scope of X.

In this case, within the same rule.
Another Example

male(albert).
male(Edward).
female(alice).
female(victoria).
parents(Edward, Albert, Victoria).
parents(Alice, Albert, Victoria).

z Note: parents(child, father, mother).

z Rule to test if X is the sister of Y:
sister_of(X,Y) :-
female(X),
parents(X,M,F),
parents(Y,M,F).

z How to write are_sisters()?
Clauses

2 ways to give Prolog information about predicates:
- Facts
- Rules

These are called clauses.
Prolog Syntax: Comments and Integers

* Comments like C or PL/I:
  */ your comment here! */

* Integers
  Some versions of Prolog do not allow negative numbers.
    SWI-Prolog does.
    It even allows non-integers.
Prolog Syntax: Atoms

- Atoms (objects and predicates) must be lower case.
  - Can include digits and underscores (but cannot start with them).
  - Objects can be enclosed in single quotes.
Prolog Syntax: Variables

z Variables must start with upper case or underscore ("_").

z Single underscore is the anonymous variable.

y Do not have to match each other, even in the same clause.
Structures a.k.a. Compound Terms

- Made up of “functor” and “components”.
- Each component acts as an object.
- Examples:
  - owns(john, book(wuthering_heights, bronte)).
  - owns(john, book(wuthering_heights, author(emily, bronte))).
File ‘books.pl’

owns(john, book(wuthering_heights, author(emily, bronte))).
owns(john, book(jane_eyre, author(charlotte, bronte))).
owns(mary, book(wuthering_heights, author(emily, bronte))).
owns(john, book(the_professor, author(charlotte, bronte))).
owns(john, book(the_invisible_man, author(h_g, wells))).
owns(mary, book(invisible_man, author(ralph, ellison))).
Structures and Goals

What do these test for?

owns(john, X).
owns(john, book(X, author(Y, bronte))).
owns(X, Y), owns(Z, Y).
Equality

- Equals: “=”
- Not Equals: “\=”
- Can we eliminate duplicate owners from the last slide?
Next Time

- More Prolog
  - Structures
  - Operators
  - Complicated rules