Today in CS 42

Hw #10: Prolog

Coming up…
Monday 11/14: HW10 due (Prolog intro)
Thursday 11/18: Midterm distributed
Monday 11/21: NO HW DUE
Tuesday 11/22: Midterm due
Thursday 11/24: Thanksgiving (no class)
Tuesday 11/29: HW11 due (Prolog games)
Review: Tail Recursion and Hmmm

(define (tower N)
  (tower-tr N 2))

(define (tower-tr N A)
  (if (= N 0)
      A
      (power A (tower-tr (- N 1) A)))))

Is tower-tr tail recursive?
A. Yes
B. No
C. Not sure
Review: Tail Recursion and Hmmm

```
(define (power-tr b n A)
  (if (= n 0)
      A
      (power-tr b (- n 1) (* A b))))
```

Which is a "faithful" implementation of the Racket code above in Hmm?

A. left side only
B. right side only
C. both
D. neither
Big O comparison

Problem

Find the maximum element in a list

First, verify the < operator with 2,000 known comparisons (1 < 2)
Second, look at each list element, compare it with the current max
    -- Replace the current max if necessary.
As you do this, make each comparison 1,000 times just to “be sure”

You have a minimum-element finder already.
All elements start unmarked.
Find the smallest unmarked element and mark it.
Repeat, keeping track of the last element marked each time.
When they’re all marked, return the last element marked.

Please do not try these at home!
The Big Picture on “Big Oh”

Big Oh (“Big Oh”) compares running times for algorithms. If $f$ is $O(g)$, then $g$ is not $O(f)$.
The Formal Definition of Big-O

f(n) is O(g(n)) if there exists constants c and n₀ s.t.

\[ f(n) \leq c \cdot g(n) \text{ for all } n \geq n₀ \]
Big O in practice

Ignore everything but the dominant term

\[ 420n^2 + 3n^3 - 201 \]

\[ 17 + 300\sqrt{n} + 0.5\log(n) \]

\[ \frac{117}{n} + \frac{2^n}{3^n} + 1700 \]

Big-O is simply an upper bound, but normally we want the **BEST** upper bound…

big-O = hand-waving math!
What is Big-O running time of the following mystery function? (and what does it do?)

(define (mystery N)
  (reverse (mystery-help N)))

(define (mystery-help N)
  (cond
   ((= N 0) '())
   ((= 1 (modulo N 2)) (cons 1 (mystery-help (/ (- N 1) 2))))
   (else (cons 0 (mystery-help (/ N 2)))))))

A. O(log N)  B. O(N)  C. O(N log N)  D. All of these  E. None of these
Midterm #2

• Take home, distributed Thursday Nov 17
• Same ground rules as first midterm.
• Everything up through Prolog (this week) even if we only saw it in class. including topics from before midterm 1
• Short reviews over the next two lectures
More review problems posted online soon
And now, for something completely different...

Penguins are black and white. Some old TV shows are black and white. Therefore, some penguins are old TV shows.

Logic: another thing that penguins aren’t very good at.
What mood am I in today?

- closer to the machine
- more abstraction away from the machine

Taxonomy of Programming Models

Programming

- Declarative Programming
  - Logic Programming
  - Functional Programming
- Imperative Programming
  - Object-Oriented Programming
  - Low-level Programming

What mood am I in today?

Prolog
- simply describe the problem - and I'll solve it!

Python
- data structures

Racket
- algorithms via functions

JFlap
- closer to the machine
We feed Prolog a bunch of facts relevant to our problem

pairs(apple, walnut).  pairs(strawberry, honey).
pairs(apple, honey).   pairs(strawberry, ginger).
pairs(walnut, avocado). pairs(strawberry, tea).
pairs(walnut, banana).  pairs(tea, walnut).
pairs(apple, banana).   pairs(tea, tomato).
pairs(banana, ginger).  pairs(tea, milk).
pairs(banana, cloves).   pairs(X, X).
pairs(banana, strawberry).  pairs(X, coconut).
pairs(banana, coriander).

We describe how to *recognize* a solution

\[
\text{yummy_triple}(X,Y,Z) :- \text{pairs}(X,Y), \ X \neq Y, \\
\text{pairs}(Y,Z), \ Y \neq Z, \\
\text{pairs}(X,Z), \ X \neq Z.
\]

Prolog then finds solution(s) for us.

So buy now!  
http://www.swi-prolog.org/  

Demo...
Prolog: Just the Facts!

%% Here is a comment in Prolog
/* this is also a comment */

chocolate_is_good.  %% A fact, but not in a very useful form

good(spam).  %% a debatable fact, but now prolog thinks so
good(42).  %% no one can deny this...

weird_list([3.14,42,spam,'+','hi']).  %% Well, I think so.

better(tofu, spam).  %% “tofu” is better than spam
better(chocolate, tofu).  %% “chocolate” is better than “tofu”
better(money, chocolate).  %% “money” is better than “chocolate”

%% Does prolog know that money is better than spam?  Does prolog even
%% know what spam is???  NO!
Prolog can be alien…

- To run: `/opt/local/bin/swipl` (macs)
  `swipl` (ssh to knuth.cs.hmc.edu server)
- To exit: `halt`.
- To (re)load a file: `[file]`. or `reconsult('file.pl')`.

- The Prolog environment is for queries only!
Use your favorite editor to change files…

some facts and rules from the file `simpsons.pl`

```prolog
parent(homer,bart).
parent(homer,lisa).
parent(marge,bart).
parent(marge,lisa).

child(A,B) :- parent(B,A).
```

queries within the Prolog environment

```prolog
?: parent(homer,lisa).
yes

?: parent(X,bart).
X = homer ;
X = marge ;
no
```
The idea: Provide data/relationships that describe the world. Then ask questions.

Relationships:

prolog syntax is the stranger

mother(X, Y) :- parent(X, Y), female(X).

child(X, Y) :- parent(Y, X).

ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

you mean L’Étranger…

colon-hyphen means “IF”
commas mean “AND”
variables always start with a capital letter
predicates and constants start with a lowercase letter
multiple rules express “OR” ; also means “OR”
recursion welcome!

these relationships are predicates not functions. They have a value, but they do not return a value!
/*
 * the parent predicate
 */
parent(olf, skug).
parent(helga, skug).
parent(skug, esmerelda).
parent(skugerina, esmerelda).
parent(esmerelda, klotho).
parent(gemini, klotho).
parent(esmerelda, atropos).
parent(gemini, atropos).
parent(esmerelda, lachesis).
parent(gemini, lachesis).
parent(olf, homericus).
parent(helga, homericus).
parent(ug, matilda).
parent(uggette, matilda).
parent(homericus, homer).
parent(matilda, homer).
parent(homericus, gomer).
parent(matilda, gomer).
parent(homer, bart).
parent(marge, bart).
parent(homer, lisa).
parent(marge, lisa).
parent(homer, maggie).
parent(marge, maggie).
parent(john, marge).
parent(jackie, marge).
parent(john, selma).
parent(jackie, selma).
parent(john, patty).
parent(jackie, patty).
parent(john, glu).
parent(jackie, glu).
parent(glu, millhouse).
parent(cher, millhouse).
parent(glu, wentworth).
parent(cher, wentworth).

/*
 * the age predicate
 */
age(helga, 97).
age(olf, 99).
age(uggette, 93).
age(ug, 92).
age(matilda, 65).
age(homericus, 76).
age(skugerina, 101).
age(skug, 78).
age(esmerelda, 55).
age(gemini, 54).
age(klotho, 20).
age(atropos, 19).
age(lachesis, 18).
age(marge, 35).
age(homer, 38).
age(lisa, 8).
age(maggie, 1).
age(bart, 10).
age(gomer, 41).
age(john, 62).
age(jackie, 59).
age(patty, 38).
age(selma, 38).
age(glu, 27).
age(cher, 44).
age(millhouse, 8).
age(wentworth, 8).

/*
 * the female predicate
 */
female(helga).
female(esmerelda).
female(skugerina).
female(uggette).
female(matilda).
female(marge).
female(jackie).
female(selma).
female(patty).
female(cher).
female(lisa).
female(maggie).
female(klotho).
female(atropos).
female(lachesis).

/*
 * the male predicate
 */
male(olf).
male(skug).
male(homericus).
male(ug).
male(homer).
male(gomer).
male(gemini).
male(john).
male(glum).
male(bart).
male(millhouse).
male(wentworth).

/*
 * Here are three rules about families
 */
child(X, Y) :- parent(Y, X).
mother(X, Y) :- female(X), parent(X, Y).
anc(X, Y) :- parent(X, Y).
anc(X, Y) :- parent(Z, Y), anc(X, Z).
A more complete picture ...
The idea: Provide data/relationships that describe the world. Then ask questions.

Questions:

child(lisa,marge).  Is Lisa a child of Marge?

parent(marge,X).  Who has Marge as a parent?

Is Ug a parent of Bart?

Who are the ancestors of Lisa?

Who are uggette's descendants?

Who is who's parent?

Who are John's grandchildren?
Write these three family-relationship predicates in prolog:

**sibs(X,Y)**
true if X and Y are siblings

**aunt(A,N)**
true if A is N's aunt

**rel(X,Y)**
true if X and Y are related (by blood)
Are these definitions correct? A. Yes  B. No

**sibs(X,Y)**  true if X and Y are siblings

\[
sibs( X, Y ) :- \text{parent}(A,X), \text{parent}(A,Y).
\]

**aunt(A,N)**  true if A is N’s aunt

\[
aunt( X, Y ) :- \text{parent}( Q, N ), \text{sibs}( Q, A ), \text{female}( A ).
\]

**rel(X,Y)**  true if X and Y are related (by blood)

\[
\text{rel}( X, Y ) :- \text{anc}( Z, X ), \text{anc}( Z, Y ).
\]
What's wrong here?

sibs(X, Y) :- parent (A,X),
             parent (A,Y),
             X \== Y.

Nothing is wrong...
What's wrong here?

`sibs(X, Y) :- parent (A,X), parent (A,Y), X \not= Y.`

Nothing IS wrong...
**setof**

\[ \text{setof}(R, p(R), S) \]

is true when \( S \) is the set (list) of \( R \)s that satisfy predicate \( p \).

?- setof(A,aunt(A,bart),S).
?- setof(N,aunt(marge,N),S).
Prolog is DFS

Who are Bart’s aunts? aunt(A,bart)


parent(P,N)
- is there a parent?
  parent(homer,bart)
  YES N = bart
  P = homer
- other parents?
  parent(marge,bart)
  YES N = bart
  P = marge

sibling(A,P)
- is there a sibling?
  sibling(gomer,homer)
  YES N = bart
  P = homer
  A = gomer
- other siblings?
  sibling(glum,marge)
  YES N = bart
  P = marge
  A = glum
  backtrack!
- female(A)
  female(gomer) ?
  NO (fails)

female(A)
- female(gomer) ?
  NO (fails)
- female(glum) ?
  NO

other siblings?
- sibling(glum,marge)
  YES N = bart
  P = marge
  A = glum
  backtrack!
- sibling(selma,marge)
  YES N = bart
  P = marge
  A = selma
  backtrack!
- female(selma)
  SUCCESS
  YES!
Prolog: The Dream

We feed Prolog a bunch of facts relevant to our problem

- pairs(apple, walnut).
- pairs(apple, honey).
- pairs(walnut, avocado).
- pairs(walnut, banana).
- pairs(apple, banana).
- pairs(banana, ginger).
- pairs(banana, cloves).
- pairs(banana, strawberry).
- pairs(banana, coriander).
- pairs(strawberry, honey).
- pairs(strawberry, ginger).
- pairs(strawberry, tea).
- pairs(tea, walnut).
- pairs(tea, tomato).
- pairs(tea, milk).
- pairs(X, X).
- pairs(X, coconut).

We describe how to recognize a solution

yummy_triple(X,Y,Z) :- pairs(X,Y), X \= Y,
                          pairs(Y,Z), Y \= Z,
                          pairs(X,Z), X \= Z.

Prolog then finds solution(s) for us.

So buy now!  http://www.swi-prolog.org/
We feed Prolog a bunch of facts relevant to our problem

\[
\text{pairs(apple, walnut).} \\
\text{pairs(apple, honey).} \\
\text{pairs(walnut, avocado).} \\
\text{pairs(walnut, banana).} \\
\text{pairs(X,X).} \\
\text{pairs(X, coconut).} \\
\text{pairs(X,Y) :- pairs(Y,X).} \\
\text{pairs(X,Y) :- pairs(Y,X).}
\]

\% etc.

We describe how to \textit{recognize} a solution

\[
\text{yummy_triple(X,Y,Z) :- pairs(X,Y), X \neq Y,} \\
\text{pairs(Y,Z), Y \neq Z,} \\
\text{pairs(X,Z), X \neq Z.}
\]

Prolog then finds solution(s) for us?

\textit{In practice, you have to think hard about DFS...}

\textit{Infinite loop Demo…}
We feed Prolog a bunch of facts relevant to our problem

\[
\text{pairs}(X, Y) :\text{pairs}(Y, X).
\]

pairs(apple, walnut).
pairs(apple, honey).

pairs(X, X).
pairs(walnut, avocado).

pairs(X, coconut).
pairs(walnut, banana).
%

We describe how to \textit{recognize} a solution

\[
\text{yummy_triple}(X, Y, Z) :\text{pairs}(X, Y), \ X \neq Y, \\
\text{pairs}(Y, Z), \ Y \neq Z, \\
\text{pairs}(X, Z), \ X \neq Z.
\]

Prolog then finds solution(s) for us?

\text{In practice, you often have to think hard about DFS...}

\text{Infinite Loop Demo...}