CSc 372

Comparative Programming Languages

18 : Prolog — Structures

Christian Collberg

collberg+372@gmail.com

Department of Computer Science

University of Arizona

Prolog Structures

- Aka, structured or compound objects
- An object with several components.
- Similar to Pascal's Record-type, C's struct, Haskell's tuples.
- Used to group things together.
- `course(prolog,chris,mon,11)` — functor: `course`; arguments: `prolog,chris,mon,11`
- The arity of a functor is the number of arguments.

Structures – Courses

- Below is a database of courses and when they meet. Write the following predicates:
  - `lectures(Lecturer, Day)` succeeds if `Lecturer` has a class on `Day`.
  - `duration(Course, Length)` computes how many hours `Course` meets.
  - `occupied(Room, Day, Time)` succeeds if `Room` is being used on `Day` at `Time`.

```
% course(class, meetingtime, prof, hall).
course(c231, time(mon,4,5), cc, plt1).
course(c231, time(wed,10,11), cc, plt1).
course(c231, time(thu,4,5), cc, plt1).
course(c363, time(mon,11,12), cc, slt1).
course(c363, time(thu,11,12), cc, slt1).
```


Structures – Courses. . .

```
lectures(Lecturer, Day) :-
   course(Course, time(Day,_,_), Lecturer, _).

duration(Course, Length) :-
   course(Course,
         time(Day,Start,Finish), Lec, Loc),
   Length is Finish - Start.

occupied(Room, Day, Time) :-
   course(Course,
         time(Day,Start,Finish), Lec, Room),
   Start =< Time,
   Time =< Finish.
```

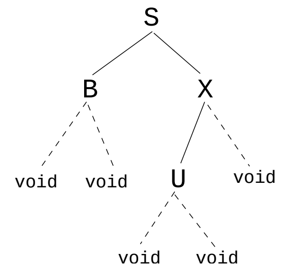
Structures – Courses. . .

```
course(c231, time(mon,4,5), cc, plt1).
course(c231, time(wed,10,11), cc, plt1).
course(c231, time(thu,4,5), cc, plt1).
course(c363, time(mon,11,12), cc, slt1).
course(c363, time(thu,11,12), cc, slt1).

?- occupied(slt1, mon, 11).
yes
?- lectures(cc, mon).
yes
```


Binary Trees

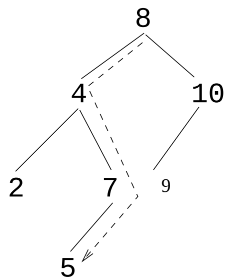
- We can represent trees as nested structures:

```
tree(Element, Left, Right)

tree(s,
  tree(b, void, void),
  tree(x,
    tree(u, void, void),
    void).
```



Binary Search Trees

- Write a predicate member(T,x) that succeeds if x is a member of the binary search tree T:



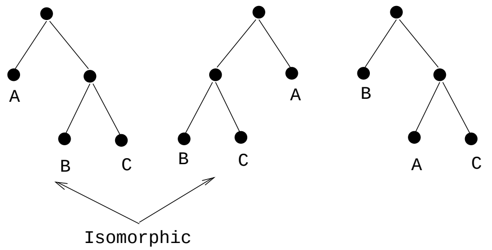

```
atree(
  tree(8,
    tree(4,
      tree(2,void,void),
      tree(7,
        tree(5,void,void),
        void)),
    tree(10,
      tree(9,void,void),
      void))).

?- atree(T),tree_member(T,5).
```


Binary Search Trees. . .

```
tree_member(X, tree(X,_, _)).
tree_member(X, tree(Y,Left,_)) :-
   X < Y,
   tree_member(Y, Left).
tree_member(X, tree(Y,_,Right)) :-
   X > Y,
   tree_member(Y, Right).
```

Binary Trees – Isomorphism

Tree isomorphism:



Two binary trees T_1 and T_2 are isomorphic if T_2 can be obtained by reordering the branches of the subtrees of T_1.

- Write a predicate `tree_iso(T1, T2)` that succeeds if the two trees are isomorphic.

Binary Trees – Isomorphism...

```
tree_iso(void, void).

tree_iso(tree(X, L1, R1), tree(X, L2, R2)) :-
   tree_iso(L1, L2), tree_iso(R1, R2).

tree_iso(tree(X, L1, R1), tree(X, L2, R2)) :-
   tree_iso(L1, R2), tree_iso(R1, L2).
```

1. Check if the roots of the current subtrees are identical;
2. Check if the subtrees are isomorphic;
3. If they are not, backtrack, swap the subtrees, and again check if they are isomorphic.

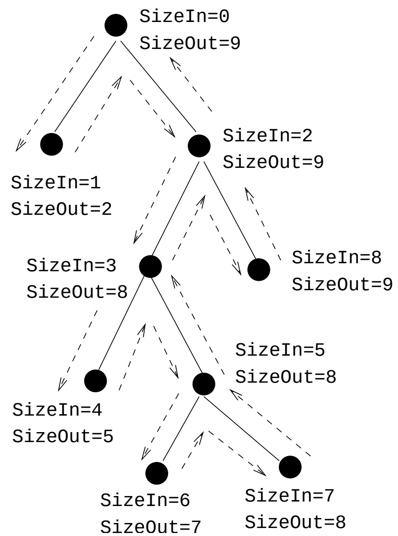
Binary Trees – Counting Nodes

- Write a predicate `size_of_tree(Tree,Size)` which computes the number of nodes in a tree.

```
size_of_tree(Tree, Size) :-
   size_of_tree(Tree, 0, Size).

size_of_tree(void, Size, Size).
size_of_tree(tree(_, L, R), SizeIn, SizeOut) :-
   Size1 is SizeIn + 1,
   size_of_tree(L, Size1, Size2),
   size_of_tree(R, Size2, SizeOut).
```

- We use a so-called accumulator pair to pass around the current size of the tree.

Binary Trees – Counting Nodes...

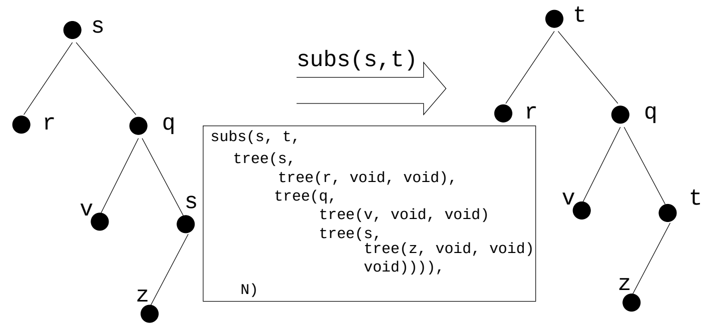


Binary Trees – Tree Substitution

- Write a predicate `subs(T1,T2,Old,New)` which replaces all occurences of `Old` with `New` in tree `T1`:

```
ubs(X, Y, void, void).
ubs(X, Y, tree(X, L1, R1), tree(Y, L2, R2)) :-
  subs(X, Y, L1, L2),
  subs(X, Y, R1, R2).
ubs(X, Y, tree(Z, L1, R1), tree(Z, L2, R2)) :-
  X =\= Y, subs(X, Y, L1, L2),
  subs(X, Y, R1, R2).
```


Binary Trees – Tree Substitution. . .



Symbolic Differentiation

$$\frac{\mathrm{d}c}{\mathrm{d}x} = 0 \quad (1)$$

$$\frac{\mathrm{d}x}{\mathrm{d}x} = 1 \quad (2)$$

$$\frac{\mathrm{d}(U^c)}{\mathrm{d}x} = cU^{c-1}\frac{\mathrm{d}U}{\mathrm{d}x} \quad (3)$$

$$\frac{\mathrm{d}(-U)}{\mathrm{d}x} = -\frac{\mathrm{d}U}{\mathrm{d}x} \quad (4)$$

$$\frac{\mathrm{d}(U+V)}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x} + \frac{\mathrm{d}V}{\mathrm{d}x} \quad (5)$$

$$\frac{\mathrm{d}(U-V)}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x} - \frac{\mathrm{d}V}{\mathrm{d}x} \quad (6)$$

Symbolic Differentiation. . .

$$\frac{\mathrm{d}(cU)}{\mathrm{d}x} = c\frac{\mathrm{d}U}{\mathrm{d}x} \quad (7)$$

$$\frac{\mathrm{d}(UV)}{\mathrm{d}x} = U\frac{\mathrm{d}V}{\mathrm{d}x} + V\frac{\mathrm{d}U}{\mathrm{d}x} \quad (8)$$

$$\frac{\mathrm{d}(\frac{U}{V})}{\mathrm{d}x} = \frac{V\frac{\mathrm{d}U}{\mathrm{d}x} - U\frac{\mathrm{d}V}{\mathrm{d}x}}{V^2} \quad (9)$$

$$\frac{\mathrm{d}(\ln U)}{\mathrm{d}x} = U^{-1}\frac{\mathrm{d}U}{\mathrm{d}x} \quad (10)$$

$$\frac{\mathrm{d}(\sin(U))}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x}\cos(U) \quad (11)$$

$$\frac{\mathrm{d}(\cos(U))}{\mathrm{d}x} = -\frac{\mathrm{d}U}{\mathrm{d}x}\sin(U) \quad (12)$$

Symbolic Differentiation…

$$\frac{\mathrm{d}c}{\mathrm{d}x} = 0 \quad (1)$$

$$\frac{\mathrm{d}x}{\mathrm{d}x} = 1 \quad (2)$$

$$\frac{\mathrm{d}(U^c)}{\mathrm{d}x} = cU^{c-1}\frac{\mathrm{d}U}{\mathrm{d}x} \quad (3)$$

```
deriv(C, X, 0) :- number(C).

deriv(X, X, 1).

deriv(U ^C, X, C * U ^L * DU) :-
      number(C), L is C - 1, deriv(U, X, DU).
```


Symbolic Differentiation…

$$\frac{\mathrm{d}(-U)}{\mathrm{d}x} = -\frac{\mathrm{d}U}{\mathrm{d}x} \quad (4)$$

$$\frac{\mathrm{d}(U+V)}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x} + \frac{\mathrm{d}V}{\mathrm{d}x} \quad (5)$$

```
deriv(-U, X, -DU) :-
   deriv(U, X, DU).

deriv(U+V, X, DU + DV) :-
   deriv(U, X, DU),
   deriv(V, X, DV).
```


Symbolic Differentiation…

$$\frac{\mathrm{d}(U-V)}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x} - \frac{\mathrm{d}V}{\mathrm{d}x} \quad (6)$$

$$\frac{\mathrm{d}(cU)}{\mathrm{d}x} = c\frac{\mathrm{d}V}{\mathrm{d}x} \quad (7)$$

```
deriv(U-V, X, __________) :-
   <left as an exercise>

deriv(C*U, X, __________) :-
   <left as an exercise>
```


Symbolic Differentiation…

$$\frac{\mathrm{d}(UV)}{\mathrm{d}x} = U\frac{\mathrm{d}V}{\mathrm{d}x} + V\frac{\mathrm{d}U}{\mathrm{d}x} \quad (8)$$

$$\frac{\mathrm{d}(\frac{U}{V})}{\mathrm{d}x} = \frac{V\frac{\mathrm{d}U}{\mathrm{d}x} - U\frac{\mathrm{d}V}{\mathrm{d}x}}{V^2} \quad (9)$$

```
deriv(U*V, X, __________) :-
   <left as an exercise>

deriv(U/V, X, __________) :-
   <left as an exercise>
```

Symbolic Differentiation...

$$\frac{\mathrm{d}(\ln U)}{\mathrm{d}x} = U^{-1}\frac{\mathrm{d}U}{\mathrm{d}x} \quad (10)$$

$$\frac{\mathrm{d}(\sin(U))}{\mathrm{d}x} = \frac{\mathrm{d}U}{\mathrm{d}x}\cos(U) \quad (11)$$

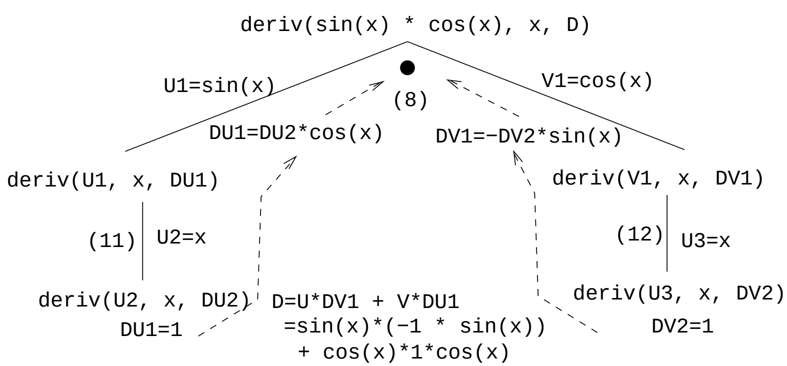
$$\frac{\mathrm{d}(\cos(U))}{\mathrm{d}x} = -\frac{\mathrm{d}U}{\mathrm{d}x}\sin(U) \quad (12)$$

```
eriv(log(U), X, ________) :- <left as an exercise>

eriv(sin(U), X, ________) :- <left as an exercise>

eriv(cos(U), X, ________) :- <left as an exercise>
```


Symbolic Differentiation...

```
?- deriv(x, x, D).
   D = 1

?- deriv(sin(x), x, D).
   D = 1*cos(x)

?- deriv(sin(x) + cos(x), x, D).
   D = 1*cos(x)+ (-1*sin(x))

?- deriv(sin(x) * cos(x), x, D).
   D = sin(x)* (-1*sin(x)) +cos(x)* (1*cos(x))

?- deriv(1 / x, x, D).
   D = (x*0-1*1)/ (x*x)
```


Symbolic Differentiation...



Symbolic Differentiation...

```
?- deriv(1/sin(x), x, D).
   D = (sin(x)*0-1* (1*cos(x)))+(sin(x)*sin(x))

?- deriv(x ^3, x, D).
   D = 1*3*x^2

?- deriv(x^3 + x^2 + 1, x, D).
   D = 1*3*x^2+1*2*x^1+0

?- deriv(3 * x ^3, x, D).
   D = 3* (1*3*x^2)+x^3*0

?- deriv(4* x ^3 + 4 * x^2 + x - 1, x, D).
   D = 4* (1*3*x^2)+x^3*0+(4* (1*2*x^1)+x^2*0)+1-0
```

Readings and References

- Read Clocksin-Mellish, Sections 2.1.3, 3.1.

Prolog So Far...

- Prolog terms:
  - atoms (`a, 1, 3.14`)
  - structures

    `guitar(ovation, 1111, 1975)`
- Infix expressions are abbreviations of "normal" Prolog terms:

infix	prefix
`a + b`	`+(a, b)`
`a + b∗ c`	`+(a, ∗(b, c))`