Prolog Structures

- Aka, **structured** or **compound** objects
- An object with several components.
- Similar to Pascal’s **Record**-type.
- Used to group things together.

\[
\text{functor course } (\text{prolog, chris, mon, 11})
\]

- The **arity** of a functor is the number of arguments.
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).

[3]
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.
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
We can represent trees as nested structures:

$$\text{tree(}\text{Element, Left, Right})$$

$$\text{tree(s,}
  \text{tree(b, void, void),}
  \text{tree(x,}
  \text{tree(u, void, void),}
  \text{void}).$$
Write a predicate \texttt{member}(T,x) that succeeds if \( x \) is a member of the binary search tree \( T \):

\[
\text{atree(}
\begin{array}{c}
\text{tree(8,}
\begin{array}{c}
\text{tree(4,}
\begin{array}{c}
\text{tree(2,void,void),}
\begin{array}{c}
\text{tree(7,}
\begin{array}{c}
\text{tree(5,void,void),}
\begin{array}{c}
\text{void})},
\text{void})},
\text{tree(10,}
\begin{array}{c}
\text{tree(9,void,void),}
\text{void}))).
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\]

?- \text{atree(T),tree_member(T,5)}.
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).
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...

```prolog
\[\text{tree}_\text{iso}(\text{void}, \text{void}).\]

\[\text{tree}_\text{iso}(\text{tree}(X, L1, R1), \text{tree}(X, L2, R2)) :\]
\[\quad \text{tree}_\text{iso}(L1, L2), \text{tree}_\text{iso}(R1, R2).\]

\[\text{tree}_\text{iso}(\text{tree}(X, L1, R1), \text{tree}(X, L2, R2)) :\]
\[\quad \text{tree}_\text{iso}(L1, R2), \text{tree}_\text{iso}(R1, L2).\]

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.
```
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...
Write a predicate \texttt{subs(T1, T2, Old, New)} which replaces all occurrences of \texttt{Old} with \texttt{New} in tree \texttt{T1}:

\begin{verbatim}
subs(X, Y, void, void).
subs(X, Y, tree(X, L1, R1), tree(Y, L2, R2)) :-
    subs(X, Y, L1, L2),
    subs(X, Y, R1, R2).
subs(X, Y, tree(Z, L1, R1), tree(Z, L2, R2)) :-
    X =\neq Y, subs(X, Y, L1, L2),
    subs(X, Y, R1, R2).
\end{verbatim}
Binary Trees – Tree Substitution...

```
subs(s, t, 
  tree(s, 
    tree(r, void, void),
    tree(q, 
      tree(v, void, void)
      tree(s, 
        tree(z, void, void) 
        void))))
```

```
subs(s, t)
```
Symbolic Differentiation

\( \frac{dc}{dx} = 0 \)  

\( \frac{dx}{dx} = 1 \)

\( \frac{d(U^c)}{dx} = cU^{c-1} \frac{dU}{dx} \)

\( \frac{d(-U)}{dx} = -\frac{dU}{dx} \)

\( \frac{d(U + V)}{dx} = \frac{dU}{dx} + \frac{dV}{dx} \)

\( \frac{d(U - V)}{dx} = \frac{dU}{dx} - \frac{dV}{dx} \)

372—Fall 2004—11 [15]
Symbolic Differentiation...

(7) \[
\frac{d(cU)}{dx} = c \frac{dU}{dx}
\]

(8) \[
\frac{d(UV)}{dx} = U \frac{dV}{dx} + V \frac{dU}{dx}
\]

(9) \[
\frac{d\left(\frac{U}{V}\right)}{dx} = \frac{V \frac{dU}{dx} - U \frac{dV}{dx}}{V^2}
\]

(10) \[
\frac{d(\ln U)}{dx} = U^{-1} \frac{dU}{dx}
\]

(11) \[
\frac{d(\sin(U))}{dx} = \frac{dU}{dx} \cos(U)
\]

(12) \[
\frac{d(\cos(U))}{dx} = -\frac{dU}{dx} \sin(U)
\]
Symbolic Differentiation... 

(1) \[
\frac{dc}{dx} = 0
\]

(2) \[
\frac{dx}{dx} = 1
\]

(3) \[
\frac{d(U^c)}{dx} = cU^{c-1} \frac{dU}{dx}
\]

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...

\[
\frac{d(-U)}{dx} = -\frac{dU}{dx}
\]

\[
\frac{d(U + V)}{dx} = \frac{dU}{dx} + \frac{dV}{dx}
\]

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

driv(U+V, X, DU + DV) :-
    deriv(U, X, DU),
    deriv(V, X, DV).
Symbolic Differentiation...

\[
\frac{d(U - V)}{dx} = \frac{dU}{dx} - \frac{dV}{dx}
\]

\[
\frac{d(cU)}{dx} = c\frac{dV}{dx}
\]

derv(U-V, X, \_\_\_\_) :-
    <left as an exercise>

derv(C*U, X, \_\_\_\_) :-
    <left as an exercise>
Symbolic Differentiation...

\[
\frac{d(UV)}{dx} = U \frac{dV}{dx} + V \frac{dU}{dx}
\]

\[
\frac{d(U/V)}{dx} = \frac{V \frac{dU}{dx} - U \frac{dV}{dx}}{V^2}
\]

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

deriv(U/V, X, ________) :-
<left as an exercise>
Symbolic Differentiation...

\[
\begin{align*}
\frac{d(\ln U)}{dx} &= U^{-1} \frac{dU}{dx} \\
\frac{d(\sin(U))}{dx} &= \frac{dU}{dx} \cos(U) \\
\frac{d(\cos(U))}{dx} &= -\frac{dU}{dx} \sin(U)
\end{align*}
\]

\text{deriv}(\log(U), X, \text{______}) :- \text{<left as an exercise>}
\text{deriv}(\sin(U), X, \text{______}) :- \text{<left as an exercise>}
\text{deriv}(\cos(U), X, \text{______}) :- \text{<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...

\[ D = U \cdot DV_1 + V \cdot DU_1 \]

\[ DU_1 = DU_2 \cdot \cos(x) \]
\[ DV_1 = -DV_2 \cdot \sin(x) \]

\[ DU_2 = 1 \]
\[ DV_2 = 1 \]

\[ U_1 = \sin(x) \]
\[ U_2 = x \]
\[ U_3 = x \]

\[ V_1 = \cos(x) \]

\[ \text{deriv}(U_1, x, DU_1) \]
\[ \text{deriv}(U_2, x, DU_2) \]
\[ \text{deriv}(U_3, x, DV_2) \]

\[ \text{deriv}(V_1, x, DV_1) \]

\[ D = U \cdot DV_1 + V \cdot DU_1 \]
\[ = \sin(x) \ast (-1 \ast \sin(x)) \]
\[ + \cos(x) \ast 1 \ast \cos(x) \]
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
Read *Clocksin-Mellish, Chapter ???*. 
Prolog So Far...

- Prolog terms:
  - atoms \((a, 1, 3.14)\)
  - structures
    \(\text{guitar(ovation, 1111, 1975)}\)

- Infix expressions are abbreviations of “normal” Prolog terms:

<table>
<thead>
<tr>
<th>Infix</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a + b)</td>
<td>(+ (a, b))</td>
</tr>
<tr>
<td>(a + b\times c)</td>
<td>(+ (a, \times(b, c)))</td>
</tr>
</tbody>
</table>