Prolog lecture 3

Go to:

http://etc.ch/t7Hj

Or scan the barcode

Today's discussion

Videos:

Arithmetic

Backtracking

Q: next time you do questions like "does this terminate" can you do polls so you can get see when ppl have thought about it before you talk about it? thanks

A: here goes...

Which of these are true statements?

2 is 1+1

2 is +(1,1)

1+1 is 1+1

A is 1+1, A = 2

1+1 is A, A = 2

http://etc.ch/t7Hj

Q: What are the advantages of using "is" vs the clpfd's equality constraint "#="? the second seems more declarative

A: #= is more complicated. It uses "is" when it can and does other stuff too. When I learn a language I like to start with the simple versions and then build up.

```
X is 1 + 2.          ---  X = 3

X #= 1 + 2.          ---  X = 3

1 is X + 2.          --- error: args not instantiated

2 #= X + 2.          --- X = 0  (cool!)

4 #= X * X.          --- X in 2 \/ -2 (cool!)

(X+2)*(X-4) #= 0.  --- _4036+4#=X, X+2#=_4056,_4056*_4036#=0. !!!
```

#= is a relation over arithmetic equality -> but integer arithmetic is undecidable.

What does LCO stand for?

Last Call Optimisation

Logical Call Order

Let's Collect Otters

London Chamber Orchestra

http://etc.ch/t7Hj

DirectPoll

Could you apply LCO to this?

```
last([L],L).

last([_|T],L) :- last(T,L).
```

http://etc.ch/t7Hj

Q: When does LCO get applied? Is it during compilation, or during execution when the call is made? (And does that make it partly determined by the arguments?)

Q: When does LCO get applied? Is it during compilation, or during execution when the call is made? (And does that make it partly determined by the arguments?)

A: Hmm.

When does LCO get applied?

Interpreted Prolog

Easy - it's applied during execution. The interpreter basically avoids allocating a new stack frame when the predicate is determinate at the point that the last clause needs to be checked

Compiled Prolog

Depends how you compiled it. But you can tell statically that LCO is applicable

Does that make it partly determined by the arguments?

It's not determined by the type of the arguments: there's only one type! (everything is a term)

It's not determined by the value of the arguments:

> think about how the search happens
>
> Prolog would need to try the unification to know if it needs to come back

How would you represent a binary search tree?

I've finished

I need a hint

A binary search tree stores one value per node. The values in the left subtree are all less than the node value and the values in the right subtree are all more than it.

http://etc.ch/t7Hj

DirectPoll

How would you represent a binary search tree?

One solution:

Represent nodes as a compound term n(L,V,R) where L is the left sub-tree, V is the value and R is the right sub-tree.

Represent leaves as the atom lf.

Write a predicate that inserts items into your tree

I've finished

I need a hint

http://etc.ch/t7Hj

Implement '='

I've finished

I need a hint

http://etc.ch/t7Hj

DirectPoll