Prolog Past Paper

Basics

- y2011p3q8 (a), y2012p3q8 (a,b)
    - unification, variable bindings, arithmetic

Generate (Iterative deepening) and Test

Lists

- y2021p7q10
    - Cipher text
    - Base case, Recursive case, Declarative explanation.
- y2020p7q10
    - generate and test
    - Symmetric relation
- y2019p7q10
    - map, data structure
    - LCO
- y2008p4q8
    - perm/2
- y1996p5q7
    - ordered integer BST
- y1997p12q8
    - next-highest member

Cut, !

It commits any parts of the rule that have done so far, removing all the choice points which come before the cut in the rule body and that caused you to try that instance of the rule in the first place.

- y2016p3q7
  - cut, choice-points
  - Last Call Optimisation
- y2014p3q8(f)
  - fix for LCO (green cut)
- y2009p3q7(b), y2010p3q7 (a), y2011p3q8 (b), y2012p3q8 (b)
  - rule matching, variable bindings
  - query and backtracking results
- y2001p5q7
  - cut, max, bug finding (red cut)

Difference Lists

They maintain a variable at the tail of each list, achieving more efficient *append in O(1) rather than O(n)*.

- y2014p3q8
  - bfs/2
  - (c) append(T-[L,R|A], [L,R|A]-A, T-A), (d) empty difference lists
- y2011p3q8 (c), y2012p3q8 (c-e)
  - binary tree
- y1996p6q7

```
y1996p6q7
  [l0, l1, l2]   ~rotate~  [l1, l2, l0]
= [l0|l1, l2]

% rotate outputs a list L, which put the first element of the input list at the end.
rotate ([H|T0], L0) : -  append(T0,[H],L0).
% To transform into difference list version, now
% T is a list T0 and an extra T1 at the tail
   T = [ ... , T1] = T0 + T1
% L is a list L0 and an extra L1 at the tail
   L = [ ... , L1] = L0 + L1

rotate2([H|T]-T1, L-L1) :- append(T-T1, [H|A]-A, L-L1).
                  A-B,      B-C, A-C
> L=T, T1=[H|A], A=L1.

rotate2([H|T]-[H|A], T-A).

Why?
T = [ ..., T1] = T0 + T1 = T0 + [H|A], thus
T-A = T0 + [H|A] - A.
```

- y1997p6q7
    - binary tree

```
y1997p6q7

enum(0, [0|A]-A, B-B).
enum(1, A-A, [1|B]-B).
% A is the original list A0 + A1
enum(n(L,R), A-A1, B-B1) :- enum(L, AL-AL1, BL-BL1), enum(R, AR-AR1 , BR-BR1).

enum(n(L,R), A-A1, B-B1) :- enum(L, AL-AR, BL-BR), enum(R, AR-A1 , BR-B1).
```