List Manipulation

Mark Neal

Concatenation

- Combining two lists to give a third bigger list
- [a,b,c] and [d,e,f] results in [a,b,c,d,e,f]
- Trivial to express, how to implement in Prolog?

```
concatenate([], List, List).

concatenate([Item|A], B, [Item|C]):-
  concatenate(A, B, C).
```

- Can you figure it out?

Concatenation

- How would you do it in Java?
- How would you do it recursively?
- A Prolog approach to life:
  - Identify a base case
  - Find a recursive way to make it simpler and simpler until you reach the base case
  - Each recursion adds a little something as it returns
  - Out pops the answer!

Built-in List Manipulation(1)

- append(List1,List2,List3). – concatenates list
- length(List). – length of a list
- member(A,List). – true if first argument appears as an element in the second argument
- last(List,A). - true if second argument matches last element in the first
- reverse(List1, List2). - true if List1 is the same as List2 but in reverse order
- select(Item, List1, List2). - true if List2 is the same as List1 but with Item removed from it

Built-in List Manipulation (2)

- These things do what they say...
- ...but what they say is not quite what you're used to

```
length(X, 5).
```

```
select(X,[dish, washer, tablet,
fish],[dish, washer, tablet]).
```

```
member(fish, List).
```

- What are these going to do?
- Are they doing what you expect?

How might these be useful?

- Maybe you want a chatbot to talk about some set of subjects
- Using `member` you can determine which subject you are addressing by looking for keywords
- Using `select` you can strip out words that you don't need when constructing a response
- Concatenating lists with `append` is indispensable for list manipulation