University of Malta
BSc(Hons.) IT Year II

Problem Solving Techniques in Artificial Intelligence

Lecture 9.1

Finding All Solutions

- Findall
- Bagof
- Setof

Department of Computer Science and AI
1998/9

Generating Successors

- In the represention of the eight puzzle already discussed, we defined the successor of a given state in terms of the relation

  swap(PB,PT,State,SState)

  where SState is the state that results from swapping the blank PB with some tile PT in State.

- swap will typically yield several solutions, each a successor to State. However, it will yield them one at a time.

- This method of generating successors is incompatible with our agenda-based search method, which requires a *set* of successors

- We can get round this problem by employing an *all solutions* predicate.

All Solutions Predicates
see Bratko p. 182ff

When a query has more than one solution, we can generate the solutions one-by-one using backtracking.

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- age(X,5).

X=anne ;
X=tom
```

Each time a new solution is generated, the previous one disappears. Sometimes we would like to have all the generated objects together.

The predicates **findall**, **bagof** and **setof** do just this.

Findall

The goal **findall(X,G,L)** where X is a term, G is a (possibly complex) goal involving X will generate the list L of all objects X such that G is satisfied

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- findall(C,age(C,A),L).
L = [peter,anne,pat,tom]
```

N.B. If there is no object X that satisfies P then findall(X,P,L) succeeds with L=[ ]

?- findall(C,foo(C,A),L).
L = [ ]

Bagof

The goal **bagof(X,G,L)** is similar to **findall(X,G,L)**, so that, for instance, we can obtain the list of all children aged 5:

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- bagof(C,age(C,5),L).

L = [anne,tom]
```

Note that unlike findall, the bagof goal *fails* if G cannot be satisfied .

?- bagof(X,age(X,10),L).
No

Bagof

Another difference is revealed if we leave age unspecified:

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- bagof(C,age(C,Age),L).

Age = 7 L = [peter];
Age = 5 L = [anne,tom];
Age = 8 L = [pat]
```

With bagof, we get a separate result for each particular value of Age.

findall(X,P,L) versus bagof(X,P,L)

The main difference concerns the way they handle unbound variable occurring in P that do not occur in X. An example of such a variable would be Y in **`bagof(X,age(X,Y),L)`**.

- findall treats such variables as uninstantiated on every pass. Thus

  findall(X,age(X,Age),Z)

  means *find everyone who has age Age. You need not have the same Age for each person*.

- bagof gives a separate set of solutions for each possible value of such variables.

- findall always succeeds (possibly with an empty result) whilst bagof can fail.

The ^ Operator

We can indicate that bagof should treat particular variables in the same way as findall using the the ^ operator. For instance, we may prefer to have all the children in one set regardless of their age.

```
?- bagof(C,Age^age(C,Age),L).

L = [peter,anne,pat,tom]
```

The goal `Age^age(C,Age)` means roughly there are one or more values of Age (without caring which or how many) for which the goal age(C,Age) succeeds.

Duplicate Results

We might also decide to concentrate on the set of ages

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- bagof(Age,age(C,Age),L).

C = peter Age = [7];
C = anne Age = [5];
C = pat Age = [8];
C = tom Age = [5]

?- bagof(Age,C^age(C,Age),L).

L = [7,5,8,5]
```

N.B. In this case there are duplicate results.

Setof

Setof is just like bagof execpt that duplicates are removed, and results are sorted *in alphanumeric order*.

```
age(peter,7).
age(anne,5).
age(pat,8).
age(tom,5).

?- bagof(Age,C^age(C,Age),L).
L = [7,5,8,5]

?- setof(Age,C^age(C,Age),L).
L = [5,7,8]

?- setof(C,Age^age(C,Age),L).
L = [anne,pat,peter,tom]
```

Checkpoint Define youngest(X), which yields the youngest member of the family, in terms of setof.

Collecting Objects of Arbitrary Shape

There is no restriction on the kinds of object that can be collected. We could, for example, construct the list of children ordered by their age by collecting pairs of the form Age/Child

```
?- setof(Age/C,age(C,Age),L).
```

```
L = [5/anne,5/tom,7/peter,8/pat]
```

Same Set

```
same_set(S1,S2) :-
 setof(X,member(X,S1),L1),
 setof(X,member(X,S2),L2),
 L1 = L2.

?- same_set([b,c,a],[a,b,c]).
yes
?- same_set([a,b,c],[a,b,d]).
no
```