# Principles of Programming Languages – Tutorial 10(B)

*Alexander Kress*

Acknowledgement: questions in this review are based on material by Diane Horton

# Today

- **Review: Prolog**

# Review: Prolog

Write a Prolog predicate `atoms(L, Ans)` that succeeds iff `L` is a list, and `Ans` is a list containing all the atoms at the top level of `L`, in order. For example,

```
atoms( [f, [a, b], f, g, h, [], [c, [d]], i], [f, f, g, h, [], i] ).
```

should succeed.

# Review: Prolog

Write a Prolog predicate `atoms(L, Ans)` that succeeds iff `L` is a list, and `Ans` is a list containing all the atoms at the top level of `L`, in order. For example,

```
atoms( [f, [a, b], f, g, h, [], [c, [d]], i], [f, f, g, h, [], i] ).
```

should succeed.

```
atoms([],[]).

atoms([H|R], [H|New]) :- atom(H), !, atoms(R,New).

atoms([_|R], New) :- atoms(R,New).
```

# Review: Prolog

The predicate `numOccurrences(Key, List, N)` should succeed iff `Key` occurs at the top level in `List` exactly N times. It doesn't work.

```
numOccurrences( _, [], 0 ).
numOccurrences( Key, [Key|Rest], N ) :- numOccurrences( Key, Rest, R ),
                                        N is R+1.
numOccurrences( Key, [Head|Rest], N ) :- numOccurrences( Key, Rest, N ).
```

(a) [3 marks]
Give an example query that will lead to the wrong answer on the first attempt to satisfy it (*i.e.*, before using ";"), and state the answer that would be given.

(b) [2 marks]
Fix the predicate. Make your changes directly on the code above.

# Review: Prolog

The predicate `numOccurrences(Key, List, N)` should succeed iff `Key` occurs at the top level in `List` exactly N times. It doesn't work.

```
numOccurrences( _, [], 0 ).
numOccurrences( Key, [Key|Rest], N ) :- numOccurrences( Key, Rest, R ),
                                        N is R+1.
numOccurrences( Key, [Head|Rest], N ) :- numOccurrences( Key, Rest, N ).
```

(a) [3 marks]
Give an example query that will lead to the wrong answer on the first attempt to satisfy it (*i.e.*, before using ";"), and state the answer that would be given.

Query: `numOccurrences(x,[x,y,z],0).`

Answer: `0`

(b) [2 marks]
Fix the predicate. Make your changes directly on the code above.

```
% A cut is needed here to prevent the last rule from being applied
% (which would not count this occurrence of Key.
numOccurrences( Key, [Key|Rest], N ) :- !, numOccurrences( Key, Rest, R ),
                                        N is R+1.
```

# Review: Prolog

Consider the following Prolog facts and rules:

```
team(diane, nlu).
team(mel, nlu).
team(suz, nlu).
team(tom, num).
team(des, num).
team(toni, theory).
team(rich, ai).
teammates(P, Q) :- team(P, X), team(Q, X).
```

*(a)* Express the teammates rule in logic.

*(b)* What will be Prolog's first answer to the query `teammates(des, X).`?

*(c)* What will be Prolog's first answer to the query `teammates(X, tom).`?

*(d)* What will be Prolog's first answer to the query `team(Tom, X).`?

# Review: Prolog

Consider the following Prolog facts and rules:

```
team(diane, nlu).
team(mel, nlu).
team(suz, nlu).
team(tom, num).
team(des, num).
team(toni, theory).
team(rich, ai).
teammates(P, Q) :- team(P, X), team(Q, X).
```

*(a)* Express the teammates rule in logic.

```
For all p, q [There exists x [team(p,x) & team(q,x)] => teammates(p,q)]
```

or

```
For all p, q, x [team(p,x) & team(q x) => teammates(p,q)]
```

*(b)* What will be Prolog's first answer to the query `teammates(des, X).`?

```
X = tom
```

*(c)* What will be Prolog's first answer to the query `teammates(X, tom).`?

```
X = tom
```

*(d)* What will be Prolog's first answer to the query `team(Tom, X).`?

```
Tom = diane
X = nlu
```

# Review: Prolog

Consider the following Prolog code:

```
frobely(annie, milka).
frobely(joey, katie).
frobely(annie, mary).
frobely(katie, eva).
frobely(eva, annie).
grobby(joey).
```

What is the *first* result of the query `frobely(A, B), frobely(B, C), not grobby(A).`

What is the *first* result of the query `not grobby(A), frobely(A, B), frobely(B, C).`

# Review: Prolog

Consider the following Prolog code:

```
frobely(annie, milka).
frobely(joey, katie).
frobely(annie, mary).
frobely(katie, eva).
frobely(eva, annie).
grobby(joey).
```

What is the *first* result of the query `frobely(A, B), frobely(B, C), not grobby(A).`

```
A = katie
B = eva
C = annie;
```

What is the *first* result of the query `not grobby(A), frobely(A, B), frobely(B, C).`

```
no
```

# Review: Prolog

Assuming that `A` is a list of integers, the predicate `blah(A,B)` is supposed to sum up all the positive values in `A`. It doesn't work.

```
blah([], 0).

blah([X|Y], XAns) :- X > 0, blah(Y, YAns), XAns = YAns + X.

blah([_|Y], Ans) :- blah(Y, Ans).
```

(a) What is the first result of the query `blah([5,1,-3,8], Sum).`

(b) Fix the predicate by making changes directly on the code above.

# Review: Prolog

Assuming that A is a list of integers, the predicate blah(A,B) is supposed to sum up all the positive values in A. It doesn't work.

```
blah([], 0).

blah([X|Y], XAns) :- X > 0, blah(Y, YAns), XAns = YAns + X.

blah([_|Y], Ans) :- blah(Y, Ans).
```

(a) What is the first result of the query blah([5,1,-3,8], Sum).

```
Sum = 0 + 8 + 1 + 5;
```

(b) Fix the predicate by making changes directly on the code above.

We need to use "is" rather than "=" to get evaluation of the expression rather than unification. And we also need a cut so that the third clause does not succeed in cases where the first element of the list is greater than 0:

```
blah([], 0).

blah([X|Y], XAns) :- X > 0, !, blah(Y, YAns), XAns is YAns + X.

blah([_|Y], Ans) :- blah(Y, Ans).
```

# Review: Prolog

Consider the following Prolog program:

```
a(X) :- b(X).
a(X) :- f(X).
b(X) :- g(X), v(X).
b(X) :- X = 44, v(X).
g(11).
g(3).
v(X).
f(5).
```

(a) Give all possible answers to the query `a(X)`, in the order that Prolog would produce them.

(b) Now consider the same Prolog program but with the first rule for `b` replaced by:

```
b(X) :- g(X), !, v(X).
```

Give all possible answer to the query `a(X)`, in the order that Prolog would produce them.

# Review: Prolog

Consider the following Prolog program:

```
a(X) :- b(X).
a(X) :- f(X).
b(X) :- g(X), v(X).
b(X) :- X = 44, v(X).
g(11).
g(3).
v(X).
f(5).
```

(a) Give all possible answers to the query `a(X)`, in the order that Prolog would produce them.

```
X = 11;
X = 3;
X = 44;
X = 5;
```

(b) Now consider the same Prolog program but with the first rule for `b` replaced by:

```
b(X) :- g(X), !, v(X).
```

Give all possible answer to the query `a(X)`, in the order that Prolog would produce them.

```
X = 11;
X = 5;
```