

Re: Prolog Programming for AI, problem 7.1

Source: <http://coding.derkeiler.com/Archive/Prolog/comp.lang.prolog/2006-08/msg00293.html>

- *From:* "growe" <orders@xxxxxxxxxxxxxx>
 - *Date:* 31 Aug 2006 09:29:14 -0700
-

Lash Rambo wrote:

I'm soldiering through "Prolog Programming for Artificial Intelligence" by Bratko, but have run into a nasty exercise. It's 7.1, and reads:

"
Write a procedure 'simplify' to symbolically simplify summation expressions with numbers and symbols (lower-case letters). Let the procedure rearrange the expressions so that all the symbols precede numbers. These are examples of its use:

```
?- simplify(1 + 1 + a, E).  
E = a + 2
```

```
?- simplify(1 + a + 4 + 2 + b + c, E).  
E = a + b + c + 7
```

```
?- simplify(3 + x + x, E).  
E = 2*x + 3  
"
```

I hit the same exercise today, and eventually came up with the following. It doesn't use anything that Bratko hasn't covered in his book up to that point. The only outside functions it uses are the count procedure from his book (which appears in the same section as the exercise so I suspect you're supposed to use it) and the delete procedure that comes with SWI Prolog (but Bratko does a delete earlier in the book on page 69).

The only thing I couldn't get rid of were the parentheses in the answer (maybe this is an artifact of SWI?). The output it gives for the examples in Bratko's question are:

```
134 ?- simplify(1+1+a,E).  
  
E = a+2
```

Re: Prolog Programming for AI, problem 7.1

Yes

135 ?- simplify(1+a+4+2+b+c,E).

E = a+ (b+ (c+7))

Yes

136 ?- simplify(3+x+x,E).

E = 2*x+3

=====

% Ex 7.1. simplify additions

% X is compound expression, A is simplified expression

% The idea is to calculate the numeric sum, and build

% a list of the atoms in the compound expression, then

% convert the list of atoms into the correct format

simplify(X, A) :-

simplify(X, Sum, Atoms),

atomSums(A, Sum, Atoms).

% If X is an atom, there are no numbers so Sum is 0

% Atom list contains the atom

simplify(X, Sum, [X]) :-

atom(X),

Sum is 0, !.

% If X is a number, then Sum is X, and the atom list

% is empty

simplify(X, Sum, []) :-

integer(X),

Sum is X, !.

% Recursive case

simplify(X, Sum, Atoms) :-

X = Y + Z, % Split compound expression

simplify(Z, SumZ, AtomsZ), % Simplify each part

simplify(Y, SumY, AtomsY),

Sum is SumY + SumZ, % Add the subsums together

append(AtomsY, AtomsZ, Atoms),!. % Join the sub-lists of atoms

% Convert the list of atoms to the simplified format

atomSums(A, A, []). % If no atoms, expression is just

the sum

% Recursive case

atomSums(A, Sum, [T|Atoms]) :- % Get first atom T in list

count(T, [T|Atoms], CountT), % Count number of times it occurs

in list

delete(Atoms, T, NewAtoms), % Delete it from list so you don't

count it again

atomSums(B, Sum, NewAtoms), % Convert remainder of list

buildExp(A, B, T, CountT), !. % Build the expression

Re: Prolog Programming for AI, problem 7.1

```
% Only print out the count for an atom if it's > 1
buildExp(A, B, T, CountT) :-
CountT > 1,
A = CountT * T + B, !
;
A = T + B, !.

% count procedure from Bratko's book, page 150.
count(_, [], 0).
count(A, [B|L], N) :-
atom(A), A = B, !,
count(A, L, N1),
N is N1 + 1
;
count(A, L, N).
```