CORDIC Prolog Implementation (1A)

-
-

Young Won Lim
04/15/2014

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Background

M. Huntbach, http://www.eecs.qmul.ac.uk/~mmh/AINotes/AINotes3.pdf

Standard Order of Terms

Comparison and unification of arbitrary terms. Terms are ordered in the so-called ``standard order''. This order is defined as follows:

Variables < Numbers < Strings < Atoms < Compound Terms

Variables are sorted by address. Attaching attributes (see section 7.1) does not affect the ordering.

Numbers are compared by value. Mixed integer/float are compared as floats. If the comparison is equal, the float is considered the smaller value. If the Prolog flag iso is defined, all floating point numbers precede all integers.

Strings are compared alphabetically.

Atoms are compared alphabetically.

Compound terms are first checked on their arity, then on their functor name (alphabetically) and finally recursively on their arguments, leftmost argument first.

Standard Order of Terms

@Term1 == @Term2
    True if Term1 is equivalent to Term2. A variable is only identical to a sharing variable.
@Term1 \== @Term2
    Equivalent to \+Term1 == Term2.
@Term1 @< @Term2
    True if Term1 is before Term2 in the standard order of terms.
@Term1 @=< @Term2
    True if both terms are equal (==/2) or Term1 is before Term2 in the standard order of terms.
@Term1 @> @Term2
    True if Term1 is after Term2 in the standard order of terms.
@Term1 @>= @Term2
    True if both terms are equal (==/2) or Term1 is after Term2 in the standard order of terms.
compare(?Order, @Term1, @Term2)
    Determine or test the Order between two terms in the standard order of terms. Order is one of <, > or =, with the obvious meaning.

Insert tree

```
insert(N,empty, tree(empty,N,empty)).

insert(N, tree(Left,M,Right), tree(NewLeft,M,Right)) :-

       N@=<M, !, insert(N,Left,NewLeft).

insert(N,tree(Left,M,Right), tree(Left,M,NewRight)) :-

insert(N,Right,NewRight).
```

tree(L,N,R)
the left branch is L
the right branch is R
the store data N

Delete tree

```
delete(N, tree(Left,M,Right), tree(NewLeft,M,Right)) :-
        N@=<M, !, delete(N,Left,NewLeft).
delete(N,tree(Left,M,Right), tree(Left,M,NewRight)) :-
delete(N,Right,NewRight).
```

delete_root

```
delete_root(Left,empty,Left) :- !.

delete_root(empty,Right,Right) :- !.

delete_root(Left,Right,tree(NewLeft,N,Right)) :-
```

Remove_rightmost

```
remove_rightmost(Left,NewLeft,N).

remove_rightmost(tree(Left,N,empty),Left,N) :- !.

remove_rightmost(tree(Left,N,Right),tree(Left,N,NewRight),M) :-

remove_rightmost(Right,NewRight,M).
```

Build, flatten, append

```
build([],empty).

build([H|T],Tree) :- build(T,Tree1), insert(H,Tree1,Tree).



flatten(empty,[]).

flatten(tree(Left,N,Right),L) :-

flatten(Left,FlatLeft),

flatten(Right,FlatRight),



append(FlatLeft,[N|FlatRight],L).

append([],L,L).

append([H|T],L,[H|A]) :- append(T,L,A).
```

Binary Tree in Prolog

```
insert(N,empty,tree(empty,N,empty)).
insert(N,tree(Left,M,Right),tree(NewLeft,M,Right)) :-
N@=<M, !, insert(N,Left,NewLeft).
insert(N,tree(Left,M,Right),tree(Left,M,NewRight)) :-
insert(N,Right,NewRight).
delete(N,tree(Left,M,Right),T) :-
N==M, !, delete_root(Left,Right,T).
delete(N,tree(Left,M,Right),tree(NewLeft,M,Right)) :-
N@=<M, !, delete(N,Left,NewLeft).
delete(N,tree(Left,M,Right),tree(Left,M,NewRight)) :-
delete(N,Right,NewRight).
delete_root(Left,empty,Left) :- !.
delete_root(empty,Right,Right) :- !.
delete_root(Left,Right,tree(NewLeft,N,Right)) :-
remove_rightmost(Left,NewLeft,N).
remove_rightmost(tree(Left,N,empty),Left,N) :- !.
remove_rightmost(tree(Left,N,Right),tree(Left,N,NewRight),M) :-
remove_rightmost(Right,NewRight,M).
build([],empty).
build([H|T],Tree) :- build(T,Tree1), insert(H,Tree1,Tree).
flatten(empty,[]).
flatten(tree(Left,N,Right),L) :-
flatten(Left,FlatLeft),
flatten(Right,FlatRight),
append(FlatLeft,[N|FlatRight],L).
append([],L,L).
append([H|T],L,[H|A]) :- append(T,L,A).
```

References

[1] M. Huntbach, http://www.eecs.qmul.ac.uk/~mmh/AINotes/AINotes3.pdf