lib/search/binary-search-tree.ath

```
# Binary search trees, a subset of binary trees defined by a
# predicate, BST

load "ordered-list"
load "binary-tree"

#--------------------------------------------------------------------------

extend-module SWO {
 open BinTree

 declare BST: (S) [(BinTree S)] -> Boolean

 module BST {

  declare in: (S) [S (BinTree S)] -> Boolean

  module in {

   define empty := (forall x . ~ x in null)
   define nonempty :=
    (forall x L y R .  x in (node L y R) <==> x E y | x in L | x in R)

   (add-axioms theory [empty nonempty])   # SWO.Theory

   define root := (forall x L y R . x E y ==> x in (node L y R))
   define left := (forall x L y R . x in L ==> x in (node L y R))
   define right := (forall x L y R . x in R ==> x in (node L y R))

   define proofs :=
    method (theorem adapt)
      let {[get prove chain chain-> chain<-] := (proof-tools adapt theory);
           [E in] := (adapt [E in])}
      match theorem {
        (val-of root) =>
        pick-any x L y R
          (!chain
           [(x E y) ==> (x E y | x in L | x in R)    [alternate]
                    ==> (x in (node L y R))          [nonempty]])
      | (val-of left) =>
        pick-any x L y R
          (!chain
           [(x in L) ==> (x in L | x in R)            [alternate]
                     ==> (x E y | x in L | x in R)    [alternate]
                     ==> (x in (node L y R))          [nonempty]])
      | (val-of right) =>
        pick-any x L y R
          assume (x in R)
            (!chain->
             [(x in R) ==> (x in L | x in R)          [alternate]
                       ==> (x E y | x in L | x in R)  [alternate]
                       ==> (x in (node L y R))        [nonempty]])
      }

   (add-theorems theory |{[root left right] := proofs}|)
  } # in

  define empty := (BST null)
  define nonempty :=
   (forall L y R .
     BST (node L y R) <==>
     BST L & (forall x . x in L ==> x <E y) &
     BST R & (forall z . z in R ==> y <E z))

  (add-axioms theory [empty nonempty])

 } # BST
```

```
} # SWO
```