FoVeOOS competition 2011
Second Challenge

Krakatoa/Why3 team

October 4, 2011

1 Solution in Why3, first part

This first part is only a partial specification, proving that the code returns a value greater or equal the maximum of the tree

The Why3ML version of the challenge code, is as follows. (verbatim copy, sorry for the length file).

```
(*
COST Verification Competition
Please send solutions to vladimir@cost-ic0701.org

Challenge 2: Maximum in a tree


Given: A non-empty binary tree, where every node carries an integer.

Implement and verify a program that computes the maximum of the values
in the tree.

Please base your program on the following data structure signature:

public class Tree {

    int value;
    Tree left;
    Tree right;

}

You may represent empty trees as null references or as you consider
appropriate.
*)
```

```
theory BinTree

  use import int.Int
  use import int.MinMax

  type tree = Null | Tree int tree tree

  (* tests whether an integer v occurs in t *)
  predicate mem (v:int) (t:tree) =
    match t with
    | Null -> false
    | Tree x l r -> x=v /\ mem v l /\ mem v r
    end

  (* tests whether an integer is greater or equal to all values of a tree *)
  predicate ge_tree (v:int) (t:tree) =
    match t with
    | Null -> true
    | Tree x l r -> v >= x /\ ge_tree v l /\ ge_tree v r
    end

  lemma ge_trans : forall t:tree, x y:int.
     x >= y /\ ge_tree y t -> ge_tree x t

end

module Tree

  use import int.Int
  use int.MinMax
  use import BinTree

  let rec max_aux (t:tree) (acc:int) =
    { true }
    match t with
    | Null -> acc
    | Tree v l r ->
        max_aux l (max_aux r (MinMax.max v acc))
    end
    { ge_tree result t /\ result >= acc }

  let max (t: tree) =
    { t <> Null }
    match t with
    | Null -> absurd
    | Tree v l r ->
```

```
        max_aux l (max_aux r v)
    end
    { ge_tree result t }

end

(*
Local Variables:
compile-command: "why3ide challenge2_why3.mlw"
End:
*)
```

predicate `ge_tree` defines what means for a given integer to be greater or equal all values in a tree.

The proof proceeds automatically by several theorem provers, except the lemma which is proved in Coq

Proof of the lemma:

Proof obligations	Coq 8.2pl1
ge_trans	0.44

Coq script (excerpt from file `challenge2_why3/challenge2_why3_BinTree_ge_trans_1.v`)

```
induction t.
intros; simpl; auto.
intros x y.
simpl.
intros (H,(I,(J,K))).
split; auto with zarith.
split.
apply IHt1 with (y:=y); auto with zarith.
apply IHt2 with (y:=y); auto with zarith.
```

Proof of the code:

Proof obligations		Alt-Ergo 0.93	CVC3 2.2	Z3 2.19
parameter max_aux	parameter max_aux	0.02	0.01	0.03
	parameter max_aux	0.03		
parameter max		0.02		0.06

2 Solution in Why3, second part

This times we also specify that the result belongs to the tree

```
(*
COST Verification Competition
Please send solutions to vladimir@cost-ic0701.org

Challenge 2: Maximum in a tree


Given: A non-empty binary tree, where every node carries an integer.

Implement and verify a program that computes the maximum of the values
in the tree.

Please base your program on the following data structure signature:

public class Tree {

    int value;
    Tree left;
    Tree right;

}

You may represent empty trees as null references or as you consider
appropriate.
*)

theory BinTree

  use import int.Int
  use import int.MinMax

  type tree = Null | Tree int tree tree

  (* tests whether an integer v occurs in t *)
  predicate mem (v:int) (t:tree) =
    match t with
    | Null -> false
    | Tree x l r -> x=v /\ mem v l /\ mem v r
    end

  (* tests whether an integer is greater or equal to all values of a tree *)
  predicate ge_tree (v:int) (t:tree) =
```

```
    match t with
    | Null -> true
    | Tree x l r -> v >= x /\ ge_tree v l /\ ge_tree v r
    end

  lemma ge_trans : forall t:tree, x y:int.
     x >= y /\ ge_tree y t -> ge_tree x t

end

module Tree

  use import int.Int
  use int.MinMax
  use import BinTree

  let rec max_aux (t:tree) (acc:int) =
    { true }
    match t with
    | Null -> acc
    | Tree v l r ->
        max_aux l (max_aux r (MinMax.max v acc))
    end
    { ge_tree result t /\ result >= acc /\
      (result = acc /\ mem result t)
      }

  let max (t: tree) =
    { t <> Null }
    match t with
    | Null -> absurd
    | Tree v l r ->
        max_aux l (max_aux r v)
    end
    { ge_tree result t /\ mem result t }

end

(*
Local Variables:
compile-command: "why3ide challenge2part2_why3.mlw"
End:
*)
```

function max is proved automatically, but not function `max_aux`. A few lemmas are needed here, not enough time.

3 About a Solution using Krakatoa

Proving the same example in Java may rise the difficulty that the subtrees may be shared. However in that particular case, that is finding the maximum, it should be correct even if shared.

Another major issue would be about termination: one should specify that the tree is finite.