

Isabelle/HOL Exercises

Trees, Inductive Data Types

Complete Binary Trees

Let's work with skeletons of binary trees where neither the leaves ("tip") nor the nodes contain any information:

```
datatype tree = Tp | Nd tree tree
```

Define a function *tips* that counts the tips of a tree, and a function *height* that computes the height of a tree.

Complete binary trees of a given height are generated as follows:

```
consts cbt :: "nat  $\Rightarrow$  tree"  
primrec  
  "cbt 0          = Tp"  
  "cbt (Suc n) = Nd (cbt n) (cbt n)"
```

We will now focus on these complete binary trees.

Instead of generating complete binary trees, we can also *test* if a binary tree is complete. Define a function *iscbt* *f* (where *f* is a function on trees) that checks for completeness: *Tp* is complete, and *Nd l r* is complete iff *l* and *r* are complete and *f l = f r*.

We now have 3 functions on trees, namely *tips*, *height* and *size*. The latter is defined automatically – look it up in the tutorial. Thus we also have 3 kinds of completeness: complete wrt. *tips*, complete wrt. *height* and complete wrt. *size*. Show that

- the 3 notions are the same (e.g. *iscbt tips t = iscbt size t*), and
- the 3 notions describe exactly the trees generated by *cbt*: the result of *cbt* is complete (in the sense of *iscbt*, wrt. any function on trees), and if a tree is complete in the sense of *iscbt*, it is the result of *cbt* (applied to a suitable number – which one?).

Hints:

- Work out and prove suitable relationships between *tips*, *height* und *size*.
- If you need lemmas dealing only with the basic arithmetic operations (+, *, ^ etc), you may “prove” them with the command *sorry*, if neither *arith* nor you can find a proof. Not *apply sorry*, just *sorry*.

- You do not need to show that every notion is equal to every other notion. It suffices to show that $A = C$ und $B = C - A = B$ is a trivial consequence. However, the difficulty of the proof will depend on which of the equivalences you prove.
- There is \wedge and \longrightarrow .

Find a function f such that *iscbt f* is different from *iscbt size*.