

Praktikum Spezifikation und Verifikation

1 Bäume

`theory Aufgabe2 = Main:`

Wir arbeiten im Folgenden mit einer sehr einfachen Form der Binärbäume, deren Blättern (“tip”) und Knoten (“node”) keine Information enthalten:

`datatype tree = Tp | Nd tree tree`

Definieren Sie eine Funktion `tips`, die die Anzahl der Blätter in einem Baum zählt, und eine Funktion `height`, die die Höhe eines Baumes berechnet.

Vollständigen Binärbäume (“complete binary trees”) einer gewissen Höhe werden wie folgt erzeugt:

`consts cbt :: "nat \Rightarrow tree"`

`primrec`

`"cbt 0 = Tp"`

`"cbt(Suc n) = Nd (cbt n) (cbt n)"`

Um diese speziellen Bäume wird es im Folgenden gehen.

Die Funktion `cbt` erzeugt vollständige Binärbäume. Man kann aber auch testen, ob ein gegebener Baum vollständig ist. Definieren sie eine allgemeine Testfunktion `iscbt` (bzgl. einer Funktion auf Bäumen), die folgende verbale Beschreibung implementiert: `Tp` ist vollständig (bzgl. jeder Funktion auf Bäumen), und `Nd l r` ist vollständig (bzgl. einer Funktion `f`) gdw `l` und `r` vollständig sind (bzgl. `f`) und `f l = f r` gilt.

Wir haben nun 3 Funktionen auf Bäumen, nämlich `tips`, `height` und `size`. Letztere wird automatisch definiert — lesen sie im Tutorial nach, wie sie sich verhält. Damit haben wir auch 3 Versionen des Begriffs *Vollständig*: vollständig bzgl. `tips`, vollständig bzgl. `height` und vollständig bzgl. `size`. Zeigen Sie nun,

- dass alle 3 Begriffe das gleiche bedeuten (z.B. `iscbt tips t = iscbt size t`), und

- *dass alle 3 Begriffe genau die von `cbt` erzeugten Bäume beschreiben:* Das Resultat von `cbt` ist vollständig im Sinne von `iscbt` (bzgl. jeder Funktion auf Bäumen), und wenn ein Baum im Sinne von `iscbt` vollständig ist, so ist er das Ergebnis von `cbt` (angewandt auf eine passende, von Ihnen zu findende Zahl).

Geben Sie eine Funktion `f` an, so dass `iscbt f` verschieden von `iscbt size` etc ist.

Hinweise:

- Überlegen Sie Sich (und beweisen Sie!) geeignete Beziehungen zwischen den Funktionen `tips`, `height` und `size`.
- Eventuell benötigen Sie Lemmas, in denen es nur um die arithmetischen Grundfunktionen (`+`, `*`, `^` etc) geht. Diese können Sie einfach mit `sorry` „beweisen“, falls weder die Methode `arith` noch Sie selbst einen Beweis finden. Nicht `apply sorry` sondern nur `sorry`.
- Sie brauchen nicht explizit zu beweisen, dass jeder Begriff gleich jedem anderen ist. So genügt es z.B. zu zeigen, dass $A = C$ und $B = C \implies A = B$ ist eine triviale Konsequenz. Die Schwierigkeit der Beweise wird aber davon abhängen, welche der Äquivalenzen sie beweisen.
- Es gibt \wedge und \implies .

end