Braun Trees in Agda

Camilo Andrés Rodríguez Garzón
University EAFIT, Medellín
Colombia

May 25, 2017
Logic and Computation Research Group
What are Braun trees?

Among the many types of balanced binary trees, the Braun trees (Braun & Rem, 1983) are perhaps the most limited. A Braun tree is a binary tree which is as balanced as it can possibly be, every node satisfies the following condition:

• The left subtree has either the same number of nodes as the right subtree or one more.
A binary tree is a Braun tree if:

• It is empty or
• Its left and right subtrees are Braun trees

Braun trees are balanced, their maximum depth is $O \left( \log_2 n \right)$, where $n$ is the number of elements in the tree.
With dependent types (Stump, 2015) we can define the type of height-balanced trees of a certain size, the type \texttt{BraunTree} is indexed by a natural number which represents the size of the tree.

\begin{verbatim}
BraunTree 0
BraunTree 1
  :
BraunTree n
\end{verbatim}
Property balanced of Braun trees

The trick for maintain the property of balanced Braun trees occur during insertion data.
Data type Braun trees

The index \( n \) is the size of the tree (number of elements of type \( A \))

\[
\text{postulate}
\begin{align*}
A & : \text{Set} \\
_<A_ & : A \to A \to \mathbb{B}
\end{align*}
\]

data \text{BraunTree} : (n : \mathbb{N}) \to \text{Set} \text{ where} \\
\text{empty} & : \text{BraunTree} \ 0 \\
\text{node} & : \forall \{m \ n}\n\text{.} \\
& \text{to } A \to \text{BraunTree} \ m \to \text{BraunTree} \ n \\
& \text{m }\equiv\text{ n }\lor \text{m }\equiv\text{ suc n} \\
& \text{BraunTree (suc (m + n))}
\]
Data type Braun trees

postulate
  a : A

data₁ : BraunTree 0
data₁ = empty

data₂ : BraunTree 1
data₂ = node a
  empty
  empty
  (inj₁ refl)

data₃ : BraunTree 2
data₃ = node a
  empty
  empty
  (inj₁ refl)
  empty
  (inj₂ refl)
Method of Braun trees

- Insert

```latex
\{ - \text{we will keep smaller \(<_A\) elements closer to the root of the Braun tree as we insert -} \}
btInsert : \forall \{n\} \to A \to BraunTree \ n \to BraunTree \ (suc \ n)
bInsert x empty = node x empty empty (inj_1 refl)
bInsert x (node\{m\}\{n\} y tree_l tree_r p)
    rewrite +comm m n
    with p | if x <_A y then (x, y) else (y, x)
    ... | inj_1 m≡n | (v_1, v_2) = node v_1 (bInsert v_2 tree_r) tree_l (inj_2 (cong suc (sym m≡n)))
    ... | inj_2 m≡suc\ n | (v_1, v_2) = node v_1 (bInsert v_2 tree_r) tree_l (inj_1 (sym m≡suc\ n))
```
• Insert

\[
\begin{align*}
\text{insert}_1 &: \text{BraunTree 2} \\
\text{insert}_1 &= \text{btInsert a} \\
&\quad (\text{btInsert a empty}) \\
\text{insert}_2 &: \text{BraunTree 1} \\
\text{insert}_2 &= \text{btInsert a} \\
&\quad \text{empty} \\
\text{insert}_3 &: \text{BraunTree 3} \\
\text{insert}_3 &= \text{btInsert a} \\
&\quad \text{data}_3 \\
\text{insert}_4 &: \text{BraunTree 2} \\
\text{insert}_4 &= \text{btInsert a} \\
&\quad \text{data}_2
\end{align*}
\]

THANKS!