# Binary Search Trees

## Binary trees

A **labelled binary tree** is either **empty**: $\bullet$ or a triple $\dfrac{n}{l \mid r}$ where $n \in \mathbb{N}$ and $l$, $r$ are binary trees.

We **code** binary trees by **constructors**

$$\bullet \equiv E = 0,0 \qquad\qquad \frac{n}{l \mid r} \equiv Nd(n,l,r) = 1,n,l,r.$$

The following **format** holds of (codes of) binary trees:

$$Bt(\bullet)$$

$$Bt\left(\frac{n}{l \mid r}\right) \leftarrow N(n) \wedge Bt(l) \wedge Bt(r)$$

## Basic operations on binary trees

$|t|_b$ yields the number of **nodes** in a binary tree:

$$|\bullet|_b = 0$$
$$\left|\frac{n}{l \mid r}\right|_b \leftarrow |l|_b + |r|_b + 1$$

For $t$ a binary tree $x \,\varepsilon\, t$ holds iff $x$ is a label in $t$:

$$Bt(t) \rightarrow x \,\varepsilon\, t \leftrightarrow \exists n \exists l \exists r (t = \frac{n}{l \mid r} \wedge (x = n \vee x \,\varepsilon\, l \vee x \,\varepsilon\, r))$$

Note that any clausal definition of the predicate will have to search the whole tree.

## Traversals of binary trees

A **traversal** of a binary tree $t$ is a function which forms a list out of the nodes of $t$.

**Preorder**, **Inorder**, and **Postorder** are functions which traverse first left and then right subtrees. Labels are written out in that order **before**, **in the middle**, **after** the traversals.

For instance

$$Inorder\,(\bullet) = 0$$
$$Inorder\left(\frac{n}{l \mid r}\right) = Inorder(l) \oplus (n, Inorder(r))$$

## Subtree predicate

For binary trees $s$, $t$ we say $s$ **is a subtree of** $t$ and write $s \sqsubseteq_b t$, when

$$s \sqsubseteq_b \bullet \leftrightarrow s = \bullet$$

$$s \sqsubseteq_b \frac{n}{l \mid r} \leftrightarrow s = \frac{n}{l \mid r} \vee s \sqsubseteq_b l \vee s \sqsubseteq_b r$$

## Binary Search Trees

We define the predicate $Bst(t)$ to hold of binary search trees as follows:

$$Bst(t) \leftrightarrow Bt(t) \wedge \forall n \forall l \forall r (\frac{n}{l \mid r} \sqsubseteq_b t \rightarrow \\ \forall m (m \, \varepsilon_b \, l \rightarrow m < n) \wedge \\ \forall m (m \, \varepsilon_b \, r \rightarrow m > n))$$

We could use also the equivalent definition:

$$Bst(t) \leftrightarrow Bt(t) \wedge SetInorder(t)$$

Binary search trees can be used to implement **finite sets** in a more optimal way than **lists**.