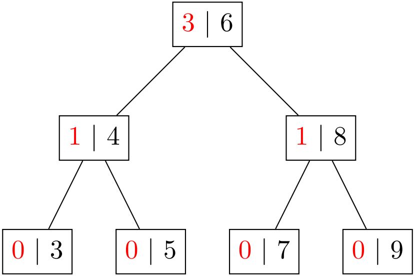
Rank Annotated Trees

Todor Peev, Vincent Trélat

École Nationale Supérieure des Mines de Nancy
Département Informatique

May 23, 2022

Definition



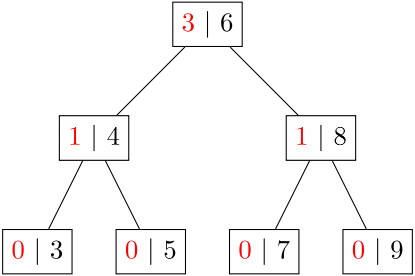
(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

$\langle\langle\langle\langle\rangle, 0, 3, \langle\rangle\rangle, 1, 4, \langle\rangle\rangle, 3, 6 :: \mathtt{nat}, \langle\langle\langle\rangle, 0, 7, \langle\rangle\rangle, 1, 8, \langle\langle\rangle, 0, 9, \langle\rangle\rangle\rangle\rangle$

Definition



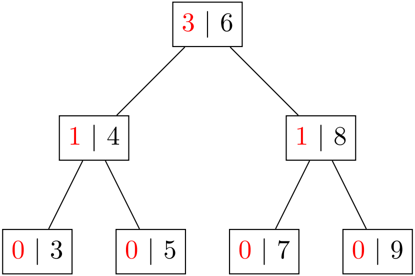
(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

$\langle\langle\langle\langle\rangle, 0, 3, \langle\rangle\rangle, 1, 4, \langle\rangle\rangle, 3, 6 :: \mathtt{nat}, \langle\langle\langle\rangle, 0, 7, \langle\rangle\rangle, 1, 8, \langle\langle\rangle, 0, 9, \langle\rangle\rangle\rangle\rangle$

Definition



(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

$\langle\langle\langle\langle\rangle, 0, 3, \langle\rangle\rangle, 1, 4, \langle\rangle\rangle, 3, 6 :: \texttt{nat}, \langle\langle\langle\rangle, 0, 7, \langle\rangle\rangle, 1, 8, \langle\langle\rangle, 0, 9, \langle\rangle\rangle\rangle\rangle$

First functions

```
fun num_nodes ::  "'a rtree ⇒ nat" where
  "num_nodes ⟨⟩ = 0" |
  "num_nodes ⟨ l, _, _, r⟩ = 1 + num_nodes l + num_nodes r"
```

```
fun rbst ::  "('a::linorder) rtree ⇒ bool" where
  "rbst ⟨⟩ = True" |
  "rbst ⟨ l, n, x, x⟩ = ((∀a ∈ set_rtree l.  a < x) ∧
    (∀a ∈ set_rtree r.  x < a) ∧
    rbst l ∧
    rbst r ∧
    n = num_nodes l)"
```

First functions

```
fun num_nodes ::  "'a rtree ⇒ nat" where
   "num_nodes ⟨⟩ = 0" |
   "num_nodes ⟨ l, _, _, r⟩ = 1 + num_nodes l + num_nodes r"
```

```
fun rbst ::  "('a::linorder) rtree ⇒ bool" where
   "rbst ⟨⟩ = True" |
   "rbst ⟨ l, n, x, x⟩ = ((∀a ∈ set_rtree l.  a < x) ∧
      (∀a ∈ set_rtree r.  x < a) ∧
      rbst l ∧
      rbst r ∧
      n = num_nodes l)"
```

Some useful lemmas

```
lemma set_rtree_rbst:
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨ l, n, x, r ⟩ ⇒ a < x ⇒
  a ∈ set_rtree l"
```

```
lemma rins_set: "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar: "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨ l, n, x, r ⟩ ⇒ a < x ⇒
  a ∈ set_rtree l"
```

```
lemma rins_set:  "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar:  "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨ l, n, x, r ⟩ ⇒ a < x ⇒
  a ∈ set_rtree l"
```

```
lemma rins_set:  "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar:  "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨ l, n, x, r ⟩ ⇒ a < x ⇒
  a ∈ set_rtree l"
```

```
lemma rins_set:  "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar:  "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

Inorder traversal and getting rank

- Tree traversal : `inorder` function (in-order DFS)
- Getting rank : `rank` function w.r.t. the structure of the rank annotated tree

```
fun rank::  "'a::linorder ⇒ 'a rtree ⇒ nat" where
  "rank a ⟨⟩ = 0" |
  "rank a ⟨ l, n, x, r⟩ =
    (if a = x then n
    else if a > x then 1 + n + rank a r
    else rank a l)"
```

- Selection : `select` function

- Tree traversal : `inorder` function (in-order DFS)
- Getting rank : `rank` function w.r.t. the structure of the rank annotated tree

```
fun rank::  "'a::linorder ⇒ 'a rtree ⇒ nat" where
   "rank a ⟨⟩ = 0" |
   "rank a ⟨ l, n, x, r⟩ =
      (if a = x then n
      else if a > x then 1 + n + rank a r
      else rank a l)"
```

- Selection : `select` function

- Tree traversal : `inorder` function (in-order DFS)
- Getting rank : `rank` function w.r.t. the structure of the rank annotated tree

```
fun rank::  "'a::linorder ⇒ 'a rtree ⇒ nat" where
   "rank a ⟨⟩ = 0" |
   "rank a ⟨ l, n, x, r⟩ =
      (if a = x then n
      else if a > x then 1 + n + rank a r
      else rank a l)"
```

- Selection : `select` function

(Selection)

```
fun sel::  "nat ⇒ 'a::linorder rtree ⇒ 'a" where
  "sel _ ⟨⟩ = undefined" |
  "sel i ⟨l, n, x, r⟩ =
    (if i = n then x
    else if i < n then sel i l
    else sel (i - n - 1) r)"
```

sel 4 t

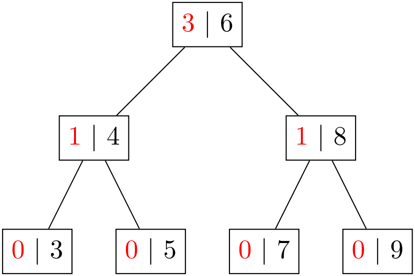
$$\texttt{inorder } t = [\underbrace{\underbrace{3,\ 4,\ 5}_{l},\ 6,}_{\texttt{offset: } n\ +\ 1}\ \underbrace{7,\ 8,\ 9}_{r}]$$

(Selection)

```
fun sel::  "nat ⇒ 'a::linorder rtree ⇒ 'a" where
   "sel _ ⟨⟩ = undefined" |
   "sel i ⟨l, n, x, r⟩ =
      (if i = n then x
      else if i < n then sel i l
      else sel (i - n - 1) r)"
```

`sel 4 t`

$$\texttt{inorder t} = \underbrace{[\underbrace{3,\ 4,\ 5,}_{\texttt{l}}\ 6,}_{\texttt{offset: n + 1}}\ \underbrace{7,\ 8,\ 9]}_{\texttt{r}}$$



```
lemma select_correct:
  "rbst t ⟹ i < length (inorder t) ⟹ sel i t = inorder t!i"
```

Idea of the proof:

- By induction on `t` with `i` arbitrary
- Cases are split w.r.t. the body of the function `sel`
- In the third case, we show:
  - `sel i ⟨l, n, x, r⟩ = sel (i - n - 1) r`
  - `sel (i - n - 1) r = inorder r!(i - n - 1)`

```
lemma rank_sel_id:
  "rbst t ⟹ i < length (inorder t) ⟹ rank (sel i t) t = i"
```

Idea of the proof:

- By induction on `t` with `i` arbitrary
- Trivialized with lemma `select_correct`