Script generated by TTT

Title: FDS (16.06.2017)

Date: Fri Jun 16 08:31:01 CEST 2017

Duration: 97:31 min

Pages: 83

Why is this implementation correct?

Because	*empty*	*insert*	*delete*	*isin*
simulate	$\{\}$	$\cup\,\{.\}$	$-\,\{.\}$	$\in$

$$set\_tree\ empty = \{\}$$
$$set\_tree\ (insert\ x\ t) = set\_tree\ t \cup \{x\}$$
$$set\_tree\ (delete\ x\ t) = set\_tree\ t - \{x\}$$
$$isin\ t\ x = (x \in set\_tree\ t)$$

Under the assumption $bst\ t$

Also: bst must be invariant

$$bst\ empty$$
$$bst\ t \Longrightarrow bst\ (insert\ x\ t)$$
$$bst\ t \Longrightarrow bst\ (delete\ x\ t)$$

8 Unbalanced BST
Correctness
Correctness Proof Method Based on Sorted Lists

$sorted :: 'a\ list \Rightarrow bool$
$sorted\ [] = True$
$sorted\ [x] = True$
$sorted\ (x\ \#\ y\ \#\ zs) = (x < y \wedge sorted\ (y\ \#\ zs))$

Structural invariant

The proof method works not just for unbalanced trees.

Structural invariant

The proof method works not just for unbalanced trees.
We assume that there is some structural invariant on the search tree:

$inv : 's \Rightarrow bool$

e.g. some balance criterion.

Correctness of $insert$

$inv\ t \wedge sorted\ (inorder\ t) \Longrightarrow$
$inorder\ (insert\ x\ t) = ins\_list\ x\ (inorder\ t)$

Correctness of *insert*

$inv\ t \wedge sorted\ (inorder\ t) \Longrightarrow$
$inorder\ (insert\ x\ t) = ins\_list\ x\ (inorder\ t)$

where

$ins\_list :: 'a \Rightarrow 'a\ list \Rightarrow 'a\ list$

inserts an element into a sorted list.

Correctness of *insert*

$inv\ t \wedge sorted\ (inorder\ t) \Longrightarrow$
$inorder\ (insert\ x\ t) = ins\_list\ x\ (inorder\ t)$

where

$ins\_list :: 'a \Rightarrow 'a\ list \Rightarrow 'a\ list$

inserts an element into a sorted list.

Also covers preservation of bst

Correctness of *delete*

$inv\ t \wedge sorted\ (inorder\ t) \Longrightarrow$
$inorder\ (delete\ x\ t) = del\_list\ x\ (inorder\ t)$

Correctness via sorted lists

Correctness proofs of all search trees
covered in this course
can be automated.

Correctness via sorted lists

Correctness proofs of all search trees
covered in this course
can be automated.

Except for the structural invariants.

Correctness via sorted lists

Correctness proofs of all search trees
covered in this course
can be automated.

Except for the structural invariants.

Therefore we concentrate on the latter from now on.

8 Unbalanced BST

9 2-3 Trees

10 Red-Black Trees

`Thys/Data_Structures/Tree23_Set.`

2-3 Trees

datatype $'a\ tree23 = \langle\rangle$
$\quad \mid Node2\ ('a\ tree23)\ 'a\ ('a\ tree23)$
$\quad \mid Node3\ ('a\ tree23)\ 'a\ ('a\ tree23)\ 'a\ ('a\ tree23)$

Abbreviations:

$$\begin{array}{rcl} \langle l,\ a,\ r\rangle & \equiv & Node2\ l\ a\ r \\ \langle l,\ a,\ m,\ b,\ r\rangle & \equiv & Node3\ l\ a\ m\ b\ r \end{array}$$

isin

$isin\ \langle l,\ a,\ m,\ b,\ r\rangle\ x =$
(case $cmp\ x\ a$ of
$\quad LT \Rightarrow isin\ l\ x$
$\mid EQ \Rightarrow True$
$\mid GT \Rightarrow$ case $cmp\ x\ b$ of
$\qquad\qquad LT \Rightarrow isin\ m\ x$
$\qquad\quad \mid EQ \Rightarrow True$
$\qquad\quad \mid GT \Rightarrow isin\ r\ x)$

isin

$isin\ \langle l,\ a,\ m,\ b,\ r\rangle\ x =$
(case $cmp\ x\ a$ of
$\quad LT \Rightarrow isin\ l\ x$
$\mid EQ \Rightarrow True$
$\mid GT \Rightarrow$ case $cmp\ x\ b$ of
$\qquad\qquad LT \Rightarrow isin\ m\ x$
$\qquad\quad \mid EQ \Rightarrow True$
$\qquad\quad \mid GT \Rightarrow isin\ r\ x)$

Assumes the usual ordering invariant

Invariant *bal*

All leaves are at the same level:

Invariant bal

All leaves are at the same level:

$bal\ \langle\rangle = True$

Invariant bal

All leaves are at the same level:

$bal\ \langle\rangle = True$

$bal\ \langle l,\ \_,\ r\rangle = (bal\ l \wedge bal\ r \wedge h(l) = h(r))$

Invariant bal

All leaves are at the same level:

$bal\ \langle\rangle = True$

$bal\ \langle l,\ \_,\ r\rangle = (bal\ l \wedge bal\ r \wedge h(l) = h(r))$

$bal\ \langle l,\ \_,\ m,\ \_,\ r\rangle =$
$(bal\ l \wedge bal\ m \wedge bal\ r \wedge h(l) = h(m) \wedge h(m) = h(r))$

Insertion

The idea:

$$\begin{array}{rcl} Leaf & \leadsto & Node2 \\ Node2 & \leadsto & Node3 \\ Node3 & \leadsto & \text{overflow, pass 1 element back up} \end{array}$$

Insertion

Two possible return values:

- tree accommodates new element without increasing height: $T_i\ t$

Insertion

Two possible return values:

- tree accommodates new element without increasing height: $T_i\ t$
- tree overflows: $Up_i\ l\ x\ r$

Insertion

Two possible return values:

- tree accommodates new element without increasing height: $T_i\ t$
- tree overflows: $Up_i\ l\ x\ r$

datatype $'a\ up_i = T_i\ ('a\ tree23)$
$\mid\ Up_i\ ('a\ tree23)\ 'a\ ('a\ tree23)$

$tree_i :: 'a\ up_i \Rightarrow 'a\ tree23$

Insertion

Two possible return values:

- tree accommodates new element without increasing height: $T_i\ t$
- tree overflows: $Up_i\ l\ x\ r$

datatype $'a\ up_i = T_i\ ('a\ tree23)$
$\mid\ Up_i\ ('a\ tree23)\ 'a\ ('a\ tree23)$

$tree_i :: 'a\ up_i \Rightarrow 'a\ tree23$

$tree_i\ (T_i\ t) = t$
$tree_i\ (Up_i\ l\ a\ r) = \langle l,\ a,\ r\rangle$

Insertion

$insert :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ tree23$

$insert\ x\ t = tree_i\ (ins\ x\ t)$

Insertion

$insert :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ tree23$

$insert\ x\ t = tree_i\ (ins\ x\ t)$

$ins :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ up_i$

68

Insertion

$ins\ x\ \langle\rangle = Up_i\ \langle\rangle\ x\ \langle\rangle$

$ins\ x\ \langle l,\ a,\ r\rangle =$

Insertion

$ins\ x\ \langle\rangle = Up_i\ \langle\rangle\ x\ \langle\rangle$

69

Insertion

$ins\ x\ \langle l,\ a,\ m,\ b,\ r\rangle =$

Insertion

$ins\ x\ \langle l,\ a,\ m,\ b,\ r\rangle =$
case $cmp\ x\ a$ of
  $LT \Rightarrow$ case $ins\ x\ l$ of
      $T_i\ l' \Rightarrow T_i\ \langle l',\ a,\ m,\ b,\ r\rangle$
    | $Up_i\ l_1\ c\ l_2 \Rightarrow Up_i\ \langle l_1,\ c,\ l_2\rangle\ a\ \langle m,\ b,\ r\rangle$
| $EQ \Rightarrow T_i\ \langle l,\ a,\ m,\ b,\ r\rangle$
| $GT \Rightarrow$
    case $cmp\ x\ b$ of
      $LT \Rightarrow$
        case $ins\ x\ m$ of
          $T_i\ m' \Rightarrow T_i\ \langle l,\ a,\ m',\ b,\ r\rangle$
        | $Up_i\ m_1\ c\ m_2 \Rightarrow Up_i\ \langle l,\ a,\ m_1\rangle\ c\ \langle m_2,\ b,\ r\rangle$
    | $EQ \Rightarrow T_i\ \langle l,\ a,\ m,\ b,\ r\rangle$
    | $GT \Rightarrow$

Insertion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t))$

Insertion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t))$

Proof by induction on t

Insertion

$$
\begin{array}{l}
ins\ x\ \langle\rangle = Up_i\ \langle\rangle\ x\ \langle\rangle \\
ins\ x\ \langle l,\ a,\ r\rangle = \\
\textsf{case}\ cmp\ x\ a\ \textsf{of} \\
\quad LT \Rightarrow \textsf{case}\ ins\ x\ l\ \textsf{of} \\
\qquad\qquad T_i\ l' \Rightarrow T_i\ \langle l',\ a,\ r\rangle \\
\qquad\quad \mid Up_i\ l_1\ b\ l_2 \Rightarrow T_i\ \langle l_1,\ b,\ l_2,\ a,\ r\rangle \\
\mid EQ \Rightarrow T_i\ \langle l,\ x,\ r\rangle \\
\mid GT \Rightarrow \textsf{case}\ ins\ x\ r\ \textsf{of} \\
\qquad\qquad T_i\ r' \Rightarrow T_i\ \langle l,\ a,\ r'\rangle \\
\qquad\quad \mid Up_i\ r_1\ b\ r_2 \Rightarrow T_i\ \langle l,\ a,\ r_1,\ b,\ r_2\rangle
\end{array}
$$

Insertion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t))$

Proof by induction on t

Insertion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t))$
where $h :: 'a\ up_i \Rightarrow nat$
$h(T_i\ t) = h(t)$
$h(Up_i\ l\ a\ r) = h(l)$
Proof by induction on t

Insertion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t)) \wedge h(ins\ a\ t) = h(t)$
where $h :: 'a\ up_i \Rightarrow nat$
$h(T_i\ t) = h(t)$
$h(Up_i\ l\ a\ r) = h(l)$
Proof by induction on t

Insertion preserves bal

Lemma

$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t)) \wedge h(ins\ a\ t) = h(t)$

where $h :: 'a\ up_i \Rightarrow nat$

$h(\ T_i\ t) = h(t)$

$h(\ Up_i\ l\ a\ r) = h(l)$

Proof by induction on t

Corollary

$bal\ t \Longrightarrow bal\ (insert\ a\ t)$

Insertion preserves bal

Lemma

$bal\ t \Longrightarrow bal\ (tree_i\ (ins\ a\ t))$

where $h :: 'a\ up_i \Rightarrow nat$

$h(\ T_i\ t) = h(t)$

$h(\ Up_i\ l\ a\ r) = h(l)$

Proof by induction on t

Deletion

Two possible return values:

- height unchanged: $T_d\ t$
- height decreased by 1: $Up_d\ t$

Deletion

$delete :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ tree23$

$delete\ x\ t = tree_d\ (del\ x\ t)$

$del\ x\ \langle l,\ a,\ r\rangle =$
(case $cmp\ x\ a$ of
$LT \Rightarrow node21\ (del\ x\ l)\ a\ r$
| $EQ \Rightarrow$ let $(a',\ t) = del\_min\ r$ in $node22\ l\ a'\ t$
| $GT \Rightarrow node22\ l\ a\ (del\ x\ r)$)

$del\ x\ \langle l,\ a,\ r\rangle =$
(case $cmp\ x\ a$ of
$LT \Rightarrow node21\ (del\ x\ l)\ a\ r$
| $EQ \Rightarrow$ let $(a',\ t) = del\_min\ r$ in $node22\ l\ a'\ t$
| $GT \Rightarrow node22\ l\ a\ (del\ x\ r)$)

$node21\ (T_d\ t_1)\ a\ t_2 = T_d\ \langle t_1,\ a,\ t_2\rangle$
$node21\ (Up_d\ t_1)\ a\ \langle t_2,\ b,\ t_3\rangle = Up_d\ \langle t_1,\ a,\ t_2,\ b,\ t_3\rangle$
$node21\ (Up_d\ t_1)\ a\ \langle t_2,\ b,\ t_3,\ c,\ t_4\rangle =$
$T_d\ \langle\langle t_1,\ a,\ t_2\rangle,\ b,\ \langle t_3,\ c,\ t_4\rangle\rangle$

76

Deletion preserves bal

Lemma
$bal\ t \Longrightarrow bal\ (tree_d\ (del\ x\ t))$

Corollary
$bal\ t \Longrightarrow bal\ (delete\ x\ t)$

Beyond 2-3 trees

datatype $'a\ tree234 =$
$Leaf \mid Node2\ \ldots \mid Node3\ \ldots \mid Node4\ \ldots$

Beyond 2-3 trees

datatype $'a\ tree234 =$
$Leaf \mid Node2\ ... \mid Node3\ ... \mid Node4\ ...$

Like 2-3 tress, but with many more cases

The general case:

B-trees and (a,b) trees

Relationship to 2-3-4 trees

Idea: encode 2-3-4 trees as binary trees;
use color to express grouping

$$\langle\rangle \approx \langle\rangle$$

Relationship to 2-3-4 trees

Idea: encode 2-3-4 trees as binary trees;

Relationship to 2-3-4 trees

Idea: encode 2-3-4 trees as binary trees;
use color to express grouping

$$\begin{aligned}
\langle\rangle &\approx \langle\rangle \\
\langle t_1, a, t_2\rangle &\approx \langle t_1, a, t_2\rangle \\
\langle t_1, a, t_2, b, t_3\rangle &\approx \langle\langle t_1, a, t_2\rangle, b, t_3\rangle\ \langle t_1, a, \langle t_2, b, t_3\rangle\rangle
\end{aligned}$$

Relationship to 2-3-4 trees

Idea: encode 2-3-4 trees as binary trees;
use color to express grouping

$$
\begin{array}{rcl}
\langle\rangle & \approx & \langle\rangle \\
\langle t_1, a, t_2\rangle & \approx & \langle t_1, a, t_2\rangle \\
\langle t_1, a, t_2, b, t_3\rangle & \approx & \langle\langle t_1, a, t_2\rangle, b, t_3\rangle\ \langle t_1, a, \langle t_2, b, t_3\rangle\rangle \\
\langle t_1, a, t_2, b, t_3, c, t_4\rangle & \approx & \langle\langle t_1, a, t_2\rangle, b, \langle t_3, c, t_4\rangle\rangle
\end{array}
$$

Invariants

Invariants

- The root is Black.
- Every $\langle\rangle$ is considered Black.

Invariants

- The root is Black.
- Every $\langle\rangle$ is considered Black.
- If a node is Red,

Invariants

- The root is Black.
- Every $\langle\rangle$ is considered Black.
- If a node is Red, its children are Black.
- All paths from a node to a leaf have the same number of

Red-black trees

datatype $color = Red \mid Black$

Red-black trees

datatype $color = Red \mid Black$

datatype
$'a\ rbt = Leaf \mid Node\ color\ ('a\ tree)\ 'a\ ('a\ tree)$

Red-black trees

datatype $color = Red \mid Black$

datatype
$'a\ rbt = Leaf \mid Node\ color\ ('a\ tree)\ 'a\ ('a\ tree)$

Abbreviations:

$$\begin{aligned} \langle\rangle &\equiv Leaf \\ \langle c,\ l,\ a,\ r\rangle &\equiv Node\ c\ l\ a\ r \\ R\ l\ a\ r &\equiv Node\ Red\ l\ a\ r \end{aligned}$$

Red-black trees

datatype $color = Red \mid Black$

datatype
$'a\ rbt = Leaf \mid Node\ color\ ('a\ tree)\ 'a\ ('a\ tree)$

Abbreviations:

$$\begin{array}{rcl} \langle\rangle & \equiv & Leaf \\ \langle c,\ l,\ a,\ r\rangle & \equiv & Node\ c\ l\ a\ r \\ R\ l\ a\ r & \equiv & Node\ Red\ l\ a\ r \\ B\ l\ a\ r & \equiv & Node\ Black\ l\ a\ r \end{array}$$

Color

$color :: 'a\ rbt \Rightarrow color$
$color\ \langle\rangle = Black$
$color\ \langle c,\ \_,\ \_,\ \_\rangle = c$

$paint :: color \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$
$paint\ c\ \langle\rangle = \langle\rangle$
$paint\ c\ \langle\_,\ l,\ a,\ r\rangle = \langle c,\ l,\ a,\ r\rangle$

Invariants

$rbt :: 'a\ rbt \Rightarrow bool$
$rbt\ t = (invc\ t \wedge invh\ t \wedge color\ t = Black)$

Invariants

$rbt :: 'a\ rbt \Rightarrow bool$
$rbt\ t = (invc\ t \wedge invh\ t \wedge color\ t = Black)$

$invc :: 'a\ rbt \Rightarrow bool$
$invc\ \langle\rangle = True$
$invc\ \langle c,\ l,\ \_,\ r\rangle =$
$(invc\ l \wedge invc\ r \wedge$
$(c = Red \longrightarrow color\ l = Black \wedge color\ r = Black))$

Invariants

$invh :: 'a\ rbt \Rightarrow bool$

Invariants

$invh :: 'a\ rbt \Rightarrow bool$

$invh\ \langle\rangle = True$

$invh\ \langle \_,\ l,\ \_,\ r\rangle = (invh\ l \wedge invh\ r \wedge bh(l) = bh(r))$

Invariants

$invh :: 'a\ rbt \Rightarrow bool$

$invh\ \langle\rangle = True$

$invh\ \langle \_,\ l,\ \_,\ r\rangle = (invh\ l \wedge invh\ r \wedge bh(l) = bh(r))$

$bheight :: 'a\ rbt \Rightarrow nat$

$bh(\langle\rangle) = 0$

$bh(\langle c,\ l,\ \_,\ \_\rangle) =$

(if $c = Black$ then $bh(l) + 1$ else $bh(l)$)

Logarithmic height

Lemma

$rbt\ t \Longrightarrow h(t) \leq 2 * \log_2 |t|_1$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$
$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$
$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

$ins :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$
$ins\ x\ \langle\rangle = R\ \langle\rangle\ x\ \langle\rangle$
$ins\ x\ (B\ l\ a\ r) = (\mathsf{case}\ cmp\ x\ a\ \mathsf{of}$
$\quad LT \Rightarrow baliL\ (ins\ x\ l)\ a\ r$
$\quad \mid EQ \Rightarrow B\ l\ a\ r$
$\quad \mid GT \Rightarrow baliR\ l\ a\ (ins\ x\ r))$
$ins\ x\ (R\ l\ a\ r) = (\mathsf{case}\ cmp\ x\ a\ \mathsf{of}$
$\quad LT \Rightarrow R\ (ins\ x\ l)\ a\ r$
$\quad \mid EQ \Rightarrow R\ l\ a\ r$
$\quad \mid GT \Rightarrow R\ l\ a\ (ins\ x\ r))$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

$baliL\ (R\ (R\ t_1\ a_1\ t_2)\ a_2\ t_3)\ a_3\ t_4$
$= R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

$baliL\ (R\ (R\ t_1\ a_1\ t_2)\ a_2\ t_3)\ a_3\ t_4$
$= R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)$

$baliL\ (R\ t_1\ a_1\ (R\ t_2\ a_2\ t_3))\ a_3\ t_4$
$= R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)$

- Principle: replace Red-Red by Red-Black
- Final equation:
$baliL\ l\ a\ r = B\ l\ a\ r$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

$$baliL\ (R\ (R\ t_1\ a_1\ t_2)\ a_2\ t_3)\ a_3\ t_4 = R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)$$

$$baliL\ (R\ t_1\ a_1\ (R\ t_2\ a_2\ t_3))\ a_3\ t_4 = R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)$$

- Principle: replace Red-Red by Red-Black

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$

89

Preservation of invariant

Theorem

$rbt\ t \Longrightarrow rbt\ (insert\ x\ t)$

Deletion

$delete\ x\ t = paint\ Black\ (del\ x\ t)$

Deletion

$$delete\ x\ t = paint\ Black\ (del\ x\ t)$$

$$\begin{aligned}
&del\ _\ \langle\rangle = \langle\rangle \\
&del\ x\ \langle _,\ l,\ a,\ r\rangle = (\textsf{case}\ cmp\ x\ a\ \textsf{of} \\
&\qquad\qquad LT \Rightarrow delL\ x\ l\ a\ r \\
&\qquad\quad \mid EQ \Rightarrow combine\ l\ r \\
&\qquad\quad \mid GT \Rightarrow delR\ x\ l\ a\ r) \\
&delL\ x\ (B\ t_1\ a\ t_2)\ b\ t_3 = baldL\ (del\ x\ (B\ t_1\ a\ t_2))\ b\ t_3 \\
&delL\ x\ l\ a\ r = R\ (del\ x\ l)\ a\ r \\
&delR\ x\ t_1\ a\ (B\ t_2\ b\ t_3) = baldR\ t_1\ a\ (del\ x\ (B\ t_2\ b\ t_3)) \\
&delR\ x\ l\ a\ r = R\ l\ a\ (del\ x\ r)
\end{aligned}$$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

$$\begin{aligned}
&baliL\ (R\ (R\ t_1\ a_1\ t_2)\ a_2\ t_3)\ a_3\ t_4 \\
&\quad = R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4) \\
&baliL\ (R\ t_1\ a_1\ (R\ t_2\ a_2\ t_3))\ a_3\ t_4 \\
&\quad = R\ (B\ t_1\ a_1\ t_2)\ a_2\ (B\ t_3\ a_3\ t_4)
\end{aligned}$$

- Principle: replace Red-Red by Red-Black
- Final equation: $baliL\ l\ a\ r = B\ l\ a\ r$
- Symmetric: $baliR$

Adjusting colors

$baliL,\ baliR :: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments $l\ a\ r$ into tree, ideally $\langle l,\ a,\ r\rangle$
- Treat invariant violation Red-Red in l/r

Deletion

$$delete\ x\ t = paint\ Black\ (del\ x\ t)$$

$$\begin{aligned}
&del\ _\ \langle\rangle = \langle\rangle \\
&del\ x\ \langle _,\ l,\ a,\ r\rangle = (\textsf{case}\ cmp\ x\ a\ \textsf{of} \\
&\qquad\qquad LT \Rightarrow delL\ x\ l\ a\ r \\
&\qquad\quad \mid EQ \Rightarrow combine\ l\ r \\
&\qquad\quad \mid GT \Rightarrow delR\ x\ l\ a\ r) \\
&delL\ x\ (B\ t_1\ a\ t_2)\ b\ t_3 = baldL\ (del\ x\ (B\ t_1\ a\ t_2))\ b\ t_3 \\
&delL\ x\ l\ a\ r = R\ (del\ x\ l)\ a\ r \\
&delR\ x\ t_1\ a\ (B\ t_2\ b\ t_3) = baldR\ t_1\ a\ (del\ x\ (B\ t_2\ b\ t_3)) \\
&delR\ x\ l\ a\ r = R\ l\ a\ (del\ x\ r)
\end{aligned}$$