Script **generated by TTT**

Title: FDS (14.06.2019)

Date: Fri Jun 14 08:36:12 CEST 2019

Duration: 83:01 min

Pages: 87

2-3 Trees

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

isin

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

Structural invariant bal

All leaves are at the same level:

Structural invariant bal

All leaves are at the same level:

$bal\ \langle\rangle\ =\ True$

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

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

Structural invariant bal

All leaves are at the same level:

$bal\ \langle\rangle\ =\ True$

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

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

Lemma

$bal\ t \Longrightarrow 2^{h(t)} \leq |t| + 1$

Insertion

The idea:

$Leaf \rightsquigarrow Node2$
$Node2 \rightsquigarrow Node3$
$Node3 \rightsquigarrow$ overflow, pass 1 element back up

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

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

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

$insert :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ tree23$
$insert\ x\ t = tree_i\ (ins\ x\ t)$

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

Insertion

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

Insertion

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

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

89

Insertion

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

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

case $cmp\ x\ a$ of
  $LT \Rightarrow$ case $ins\ x\ l$ of
    $T_i\ l' \Rightarrow T_i\ \langle l',\ a,\ r\rangle$
    $\mid Up_i\ l_1\ b\ l_2 \Rightarrow T_i\ \langle l_1,\ b,\ l_2,\ a,\ r\rangle$
$\mid EQ \Rightarrow T_i\ \langle l,\ x,\ r\rangle$
$\mid GT \Rightarrow$ case $ins\ x\ r$ of
    $T_i\ r' \Rightarrow T_i\ \langle l,\ a,\ r'\rangle$
    $\mid Up_i\ r_1\ b\ r_2 \Rightarrow T_i\ \langle l,\ a,\ r_1,\ b,\ r_2\rangle$

89

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

Insertion preserves *bal*

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

Insertion

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

Insertion preserves *bal*

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

Proof by induction on t.

Insertion

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

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

case $cmp\ x\ a$ of

$\quad LT \Rightarrow$ case $ins\ x\ l$ of

$\qquad\qquad T_i\ l' \Rightarrow T_i\ \langle l',\ a,\ r\rangle$

$\qquad\quad\ \mid Up_i\ l_1\ b\ l_2 \Rightarrow T_i\ \langle l_1,\ b,\ l_2,\ a,\ r\rangle$

$\mid EQ \Rightarrow T_i\ \langle l,\ x,\ r\rangle$

$\mid GT \Rightarrow$ case $ins\ x\ r$ of

$\qquad\qquad T_i\ r' \Rightarrow T_i\ \langle l,\ a,\ r'\rangle$

$\qquad\quad\ \mid Up_i\ r_1\ b\ r_2 \Rightarrow T_i\ \langle l,\ a,\ r_1,\ b,\ r_2\rangle$

Insertion preserves bal

Lemma

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

Insertion

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

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

case $cmp\ x\ a$ of

$\quad LT \Rightarrow$ case $ins\ x\ l$ of

$\qquad\qquad T_i\ l' \Rightarrow T_i\ \langle l',\ a,\ r\rangle$

$\qquad\quad\ \mid Up_i\ l_1\ b\ l_2 \Rightarrow T_i\ \langle l_1,\ b,\ l_2,\ a,\ r\rangle$

$\mid EQ \Rightarrow T_i\ \langle l,\ x,\ r\rangle$

$\mid GT \Rightarrow$ case $ins\ x\ r$ of

$\qquad\qquad T_i\ r' \Rightarrow T_i\ \langle l,\ a,\ r'\rangle$

$\qquad\quad\ \mid Up_i\ r_1\ b\ r_2 \Rightarrow T_i\ \langle l,\ a,\ r_1,\ b,\ r_2\rangle$

89

Insertion preserves bal

Lemma

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

91

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. Base and step automatic.

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

Deletion

The idea:

Deletion

Two possible return values:

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

datatype $'a\ up_d = T_d\ ('a\ tree23) \mid Up_d\ ('a\ tree23)$

$tree_d\ (T_d\ t) = t$
$tree_d\ (Up_d\ t) = t$

Deletion

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

Deletion

$delete :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ tree23$
$delete\ x\ t = tree_d\ (del\ x\ t)$

Deletion

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

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

$del :: 'a \Rightarrow 'a\ tree23 \Rightarrow 'a\ up_d$

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

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

Deletion preserves bal

After 13 simple lemmas:

Lemma

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

Corollary

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

Beyond 2-3 trees

Beyond 2-3 trees

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

Like 2-3 trees, 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;

Relationship to 2-3-4 trees

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

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;
use color to express grouping

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

Relationship to 2-3-4 trees

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

$$\begin{aligned} \langle\rangle &\approx \langle\rangle \\ \langle t_1,a,t_2\rangle &\approx \langle t_1,a,t_2\rangle \\ \langle t_1,a,t_2,b,t_3\rangle &\approx \langle\langle t_1,a,t_2\rangle,b,t_3\rangle \text{ or } \langle t_1,a,\langle t_2,b,t_3\rangle\rangle \\ \langle t_1,a,t_2,b,t_3,c,t_4\rangle &\approx \langle\langle t_1,a,t_2\rangle,b,\langle t_3,c,t_4\rangle\rangle \end{aligned}$$

Red means "I am part of a bigger node"

Structural invariants

- The root is

Structural invariants

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

Structural invariants

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

Structural 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 $color = Red \mid Black$

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:

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

Red-black trees

datatype $color = Red \mid Black$

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

Abbreviations:

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

Red-black trees

datatype $color = Red \mid Black$

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

Abbreviations:

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

Structural invariants

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

Structural invariants

Red-black trees

datatype $color = Red \mid Black$

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

Structural invariants

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

Structural invariants

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

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

Structural invariants

$invh :: 'a\ rbt \Rightarrow bool$
$invh\ \langle\rangle = True$
$invh\ \langle l,\ \_,\ \_,\ r\rangle = (invh\ l \wedge invh\ r \wedge bh(l) = bh(r))$

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

Structural invariants

$invh :: 'a\ rbt \Rightarrow bool$
$invh\ \langle\rangle = True$
$invh\ \langle l,\ \_,\ \_,\ r\rangle = (invh\ l \wedge invh\ r \wedge bh(l) = bh(r))$

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

$bh(\langle\rangle) = 0$
$bh(\langle l,\ \_,\ c,\ \_\rangle) =$
$(\text{if } c = Black \text{ then } bh(l) + 1 \text{ else } bh(l))$

Logarithmic height

Lemma

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

Structural invariants

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

$invh\ \langle\rangle = True$

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

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

Structural invariants

$rbt :: {'a}\ rbt \Rightarrow bool$

$rbt\ t = (invc\ t \wedge invh\ t \wedge color\ t = Black)$

$invc :: {'a}\ rbt \Rightarrow bool$

Insertion

$insert :: {'a} \Rightarrow {'a}\ rbt \Rightarrow {'a}\ rbt$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

$ins :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$ins\ x\ \langle\rangle = R\ \langle\rangle\ x\ \langle\rangle$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

$ins :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$ins\ x\ \langle\rangle = R\ \langle\rangle\ x\ \langle\rangle$

$ins\ x\ (B\ l\ a\ r) = (\textsf{case}\ cmp\ x\ a\ \textsf{of}$
$\quad LT \Rightarrow baliL\ (ins\ x\ l)\ a\ r$
$\quad |\ EQ \Rightarrow B\ l\ a\ r$
$\quad |\ GT \Rightarrow baliR\ l\ a\ (ins\ x\ r))$

Insertion

$insert :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$insert\ x\ t = paint\ Black\ (ins\ x\ t)$

$ins :: 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

$ins\ x\ \langle\rangle = R\ \langle\rangle\ x\ \langle\rangle$

$ins\ x\ (B\ l\ a\ r) = (\textsf{case}\ cmp\ x\ a\ \textsf{of}$
$\quad LT \Rightarrow baliL\ (ins\ x\ l)\ a\ r$
$\quad |\ EQ \Rightarrow B\ l\ a\ r$
$\quad |\ GT \Rightarrow baliR\ l\ a\ (ins\ x\ r))$

$ins\ x\ (R\ l\ a\ r) = (\textsf{case}\ cmp\ x\ a\ \textsf{of}$
$\quad LT \Rightarrow R\ (ins\ x\ l)\ a\ r$
$\quad |\ EQ \Rightarrow R\ l\ a\ r$
$\quad |\ GT \Rightarrow R\ l\ a\ (ins\ x\ r))$

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

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

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

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

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 $\langle l,\ a,\ r\rangle$
- 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:

  $baliL\ l\ a\ r = B\ l\ a\ r$

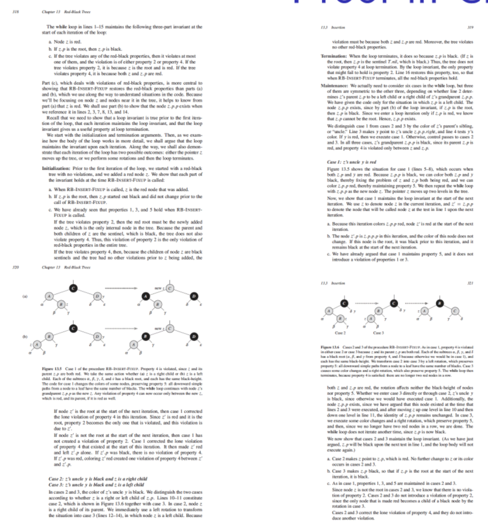
Preservation of invariant

After 14 simple lemmas:

Theorem

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

Proof in CLRS



Deletion

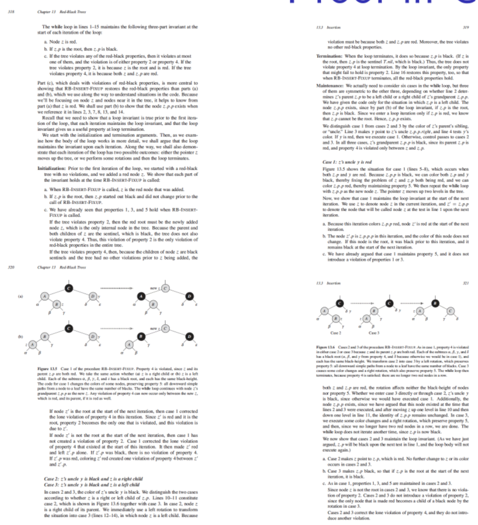
$delete\ x\ t = paint\ Black\ (del\ x\ t)$

Deletion

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

$del\ \_\ \langle\rangle = \langle\rangle$
$del\ x\ \langle l,\ a,\ \_,\ r\rangle =$
$(\text{case}\ cmp\ x\ a\ \text{of}$
$\quad LT \Rightarrow$
$\qquad \text{if}\ l \neq \langle\rangle \wedge color\ l = Black$
$\qquad \text{then}\ baldL\ (del\ x\ l)\ a\ r\ \text{else}\ R\ (del\ x\ l)\ a\ r$
$\ \mid EQ \Rightarrow combine\ l\ r$
$\ \mid GT \Rightarrow$
$\qquad \text{if}\ r \neq \langle\rangle \wedge color\ r = Black$
$\qquad \text{then}\ baldR\ l\ a\ (del\ x\ r)\ \text{else}\ R\ l\ a\ (del\ x\ r))$

Proof in CLRS



Deletion

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

$del\ \_\ \langle\rangle = \langle\rangle$
$del\ x\ \langle l,\ a,\ \_,\ r\rangle =$
$(\text{case}\ cmp\ x\ a\ \text{of}$
$\quad LT \Rightarrow$
$\qquad \text{if}\ l \neq \langle\rangle \wedge color\ l = Black$
$\qquad \text{then}\ baldL\ (del\ x\ l)\ a\ r\ \text{else}\ R\ (del\ x\ l)\ a\ r$
$\ \mid EQ \Rightarrow combine\ l\ r$
$\ \mid GT \Rightarrow$
$\qquad \text{if}\ r \neq \langle\rangle \wedge color\ r = Black$
$\qquad \text{then}\ baldR\ l\ a\ (del\ x\ r)\ \text{else}\ R\ l\ a\ (del\ x\ r))$

112

Deletion

Tricky functions: *baldL, baldR, combine*

12 short but tricky to find invariant lemmas with short proofs. The worst:

$$[\![invh\ t;\ invc\ t]\!] \Longrightarrow invh\ (del\ x\ t)\ \wedge$$
$$(color\ t = Red\ \wedge$$
$$bh(del\ x\ t) = bh(t)\ \wedge\ invc\ (del\ x\ t)\ \vee$$
$$color\ t = Black\ \wedge$$
$$bh(del\ x\ t) = bh(t) - 1\ \wedge\ invc2\ (del\ x\ t))$$

Deletion

Tricky functions: *baldL, baldR, combine*

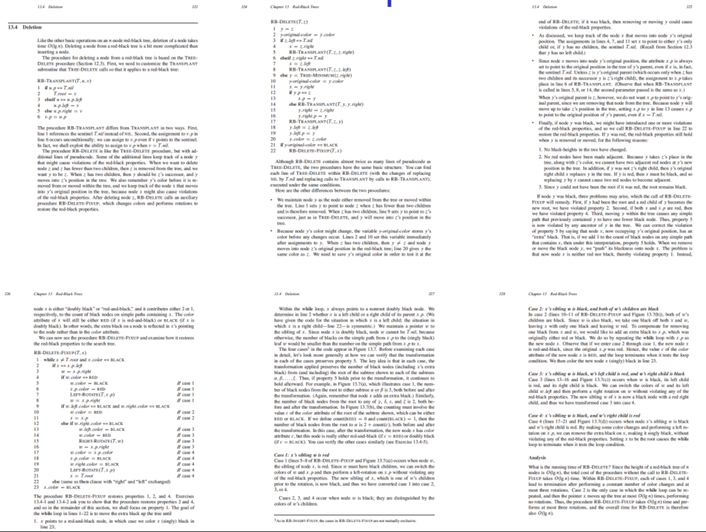
12 short but tricky to find invariant lemmas with short proofs. The worst:

$$[\![invh\ t;\ invc\ t]\!] \Longrightarrow invh\ (del\ x\ t)\ \wedge$$
$$(color\ t = Red\ \wedge$$
$$bh(del\ x\ t) = bh(t)\ \wedge\ invc\ (del\ x\ t)\ \vee$$
$$color\ t = Black\ \wedge$$
$$bh(del\ x\ t) = bh(t) - 1\ \wedge\ invc2\ (del\ x\ t))$$

Theorem

$rbt\ t \Longrightarrow rbt\ (delete\ k\ t)$

Code and proof in CLRS



Source of code

Insertion:

Okasaki's *Purely Functional Data Structures*

Deletion:

Stefan Kahrs. *Red Black Trees with Types.*
J. Functional Programming. 1996.