8.2 Binary Search Trees

8.2.1 Introduction. In this section we will study *binary search trees* which are useful for representing finite sets of natural numbers. The time required to search for an element of a binary search tree t takes the number of steps proportional to the depth of the tree t. If the tree is reasonably balanced then the time is order $\lg |t|$. The same holds for the basic operations over binary search trees such as insertion and deletion.

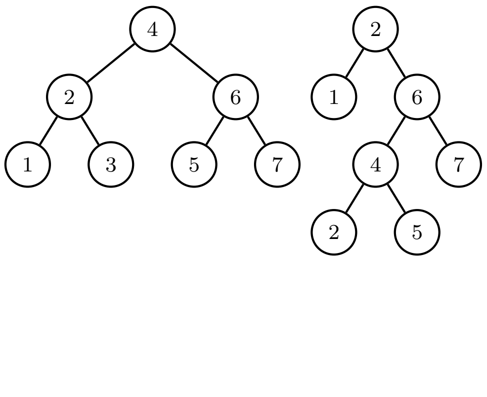

Fig. 8.3 Examples of three different binary search trees representing the same finite set of natural numbers $\{1, 2, 3, 4, 5, 6, 7\}$

A binary search tree t is a binary tree satisfying the following *search condition*:

for every non-empty subtree of t, its root label is strictly greater than the labels of its left son and strictly less than the labels of its right son.

Figure 8.3 shows three binary search trees representing the following three finite sets of natural numbers: $\{1, 2, 3\}$, $\{1, 2, 3, 4\}$ and $\{1, 2, \ldots, 15\}$.

8.2.2 Auxiliary specification predicates. The predicate $x \prec t$ holds if x is a *strict lower bound* of the labels of the binary tree t, i.e.

$$x \prec t \leftrightarrow \forall y(y \in t \rightarrow x < y).$$

The predicate $x \succ t$ holds if x is a *strict upper bound* of the labels of the binary tree t, i.e.

$$x \succ t \leftrightarrow \forall y(y \in t \rightarrow x > y).$$

Later, in Par. 8.2.8, we will find useful the binary predicate $t_1 \prec_b t_2$ holding if the labels of the binary tree t_1 are strictly less than the labels of the binary tree t_2. The predicate has the following explicit definition:

$$t_1 \prec_{\mathrm{b}} t_2 \leftrightarrow \forall x_1 \forall x_2 (x_1 \in t_1 \wedge x_2 \in t_2 \rightarrow x_1 < x_2).$$

8.2.3 Binary search trees. The predicate $Bst(t)$ holding of binary search trees is defined explicitly as a primitive recursive predicate by

$$Bst(t) \leftrightarrow Bt(t) \wedge \forall x \forall l \forall r (\langle l \mid x \mid r\rangle \trianglelefteq t \rightarrow x \succ l \wedge x \prec r).$$

From the results of Par. 8.1.10 we obtain that the predicate satisfies

$$\vdash_{\mathrm{PA}} Bst\,\langle\rangle$$
$$\vdash_{\mathrm{PA}} Bst\,\langle l \mid x \mid r\rangle \leftrightarrow x \succ l \wedge x \prec r \wedge Bst(l) \wedge Bst(r).$$

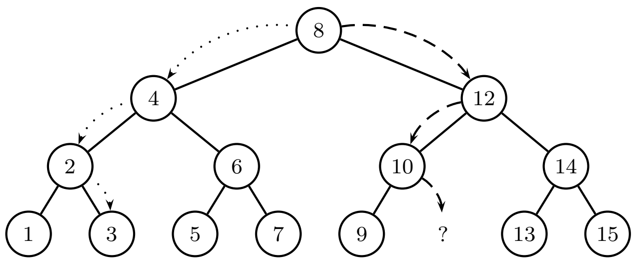

Fig. 8.4 Testing for membership of numbers 3 (*dotted arrows*) and 11 (*dashed arrows*) in the binary search tree representing the finite set $\{1, \ldots, 10\} \cup \{12, \ldots, 15\}$

8.2.4 Membership in binary search trees. Testing for membership in binary search trees is simple (see Fig. 8.4). To determine whether a binary search tree t contains a node labelled with x it suffices to compare x with the root label of t. If x is smaller than the root label then it can only appear in the left subtree; if x is greater then it appears in the right subtree. Otherwise they are equal and we are done. Note that the time to evaluate $x \in t$ in binary search trees is order $d(t)$.

The above reasoning can be expressed by the following property of its characteristic function:

```
$Bst(t) \to x \in_* t \leftrightarrow$ case
                  $t = \langle\rangle \Rightarrow 0$
                  $t = \langle l \mid y \mid r\rangle \Rightarrow$
                    case
                      $x < y \Rightarrow x \in_* l$
                      $x = y \Rightarrow 1$
                      $x > y \Rightarrow x \in_* r.$
                    end
                end
```

We can take the property as an alternative (conditional) program for computing the tree membership predicate. Its conditions of regularity

$$\vdash_{\mathrm{PA}} Bst(t) \land t = \langle l \mid x \mid r\rangle \land x < y \to l < t \land Bst(l)$$
$$\vdash_{\mathrm{PA}} Bst(t) \land t = \langle l \mid x \mid r\rangle \land x > y \to r < t \land Bst(r)$$

are trivially satisfied.

8.2.5 Extreme values of binary search tree. Now we consider the problem of computing extreme values of binary search trees. Because the labels of a binary search tree t are sorted in increasing order the leftmost node contains the smallest label of the tree and the rightmost node contains the largest (see Fig. 8.5).

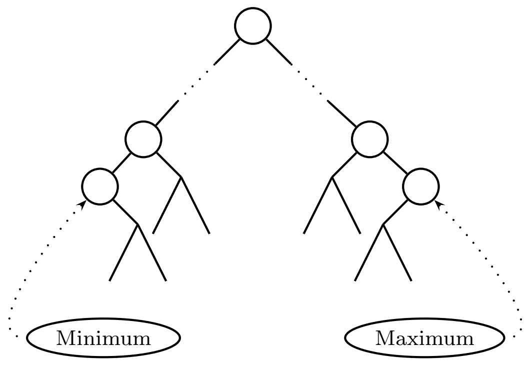

Fig. 8.5 Extreme values of binary search trees

The problem is illustrated for the function $Max(t)$ computing the largest elements in binary search trees. The function satisfies

$$\vdash_{\mathrm{PA}} Bst(t) \land t \neq \langle\rangle \to Max(t) \in t \tag{1}$$
$$\vdash_{\mathrm{PA}} Bst(t) \land t \neq \langle\rangle \land x \in t \to x \leq Max(t) \tag{2}$$

and it is defined by structural recursion on binary trees as a p.r. function:

$Max\,\langle l \mid x \mid \langle\rangle\rangle = x$
$Max\,\langle l \mid x \mid r\rangle = Max(r) \leftarrow r \neq \langle\rangle.$

We intend to apply the operation $Max(t)$ only in cases when t is a non-empty tree. For that we can take the following property as an alternative (conditional) program for computing the function:

$$\vdash_{\mathrm{PA}} Bt(t) \wedge t \neq \langle\rangle \rightarrow Max(t) = \begin{array}[t]{l}\textbf{let } t = \langle l \mid x \mid r\rangle \textbf{ in}\\ \textbf{case}\\ \quad r = \langle\rangle \Rightarrow x\\ \quad r \neq \langle\rangle \Rightarrow Max(r)\\ \textbf{end}.\end{array}$$

Its condition of regularity

$$\vdash_{\mathrm{PA}} Bt(t) \wedge t \neq \langle\rangle \wedge t = \langle l \mid x \mid r\rangle \wedge r \neq \langle\rangle \rightarrow r < t \wedge Bt(r) \wedge r \neq \langle\rangle$$

is trivially satisfied.

Verification. (1): This follows from

$$\vdash_{\mathrm{PA}} Bt(t) \wedge t \neq \langle\rangle \rightarrow Max(t) \in t$$

which can be proved by a straightforward structural induction on the binary tree t.

(2): By structural induction on the binary tree t. In the base case there is nothing to prove. In the induction step, when $t = \langle l \mid y \mid r\rangle$, assume $x \in \langle l \mid y \mid r\rangle$ and consider two cases. If $r = \langle\rangle$ then either $x = y$ or $x \in l$ and then $x < y$ by definition. In either case we have $x \leq y$ and the claim follows from definition since $Max\,\langle l \mid y \mid \langle\rangle\rangle = y$. If $r \neq \langle\rangle$ we continue by considering three subcases:

$$x = y \overset{(1)}{\Rightarrow} y < Max(r) = Max\,\langle l \mid y \mid r\rangle$$
$$x \in l \Rightarrow x < y \overset{(1)}{\Rightarrow} x < y < Max(r) = Max\,\langle l \mid y \mid r\rangle$$
$$x \in r \overset{\mathrm{IH}}{\Rightarrow} x \leq Max(r) = Max\,\langle l \mid y \mid r\rangle. \qquad \square$$

8.2.6 Insertion in binary search trees. The function $t \cup \{x\}$ takes a binary search tree t and inserts x into it (see Fig. 8.6). The function satisfies

$$\vdash_{\mathrm{PA}} Bst(t) \rightarrow Bst(t \cup \{x\}) \tag{1}$$
$$\vdash_{\mathrm{PA}} Bst(t) \rightarrow y \in t \cup \{x\} \leftrightarrow y \in t \vee y = x \tag{2}$$

and it is defined by structural recursion as a p.r. function:

$\langle\rangle \cup \{x\} = \langle\langle\rangle \mid x \mid \langle\rangle\rangle$
$\langle l \mid y \mid r\rangle \cup \{x\} = \langle l \mid y \mid r \cup \{x\}\rangle \leftarrow y < x$
$\langle l \mid y \mid r\rangle \cup \{x\} = \langle l \mid y \mid r\rangle \leftarrow y = x$
$\langle l \mid y \mid r\rangle \cup \{x\} = \langle l \cup \{x\} \mid y \mid r\rangle \leftarrow y > x.$

Note that the time to evaluate $t \cup \{x\}$ is order $d(t)$.

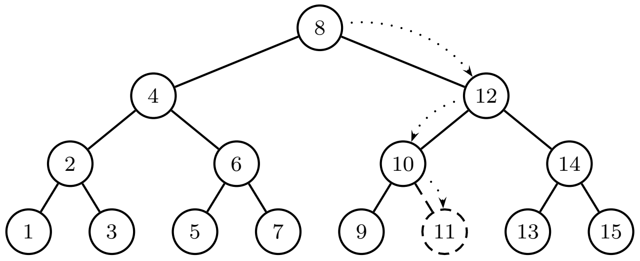

Fig. 8.6 Insertion of the number 11 into the binary search tree representing the finite set of natural numbers $\{1,\dots,10\} \cup \{12,\dots,15\}$

Verification. (2): This follows from

$$\vdash_{\mathrm{PA}} Bt(t) \rightarrow y \in t \cup \{x\} \leftrightarrow y \in t \vee y = x, \tag{$\dagger_1$}$$

which is proved by structural induction on the binary tree t. The base case is straightforward. In the induction step when $t = \langle l \mid {}^{z} \mid r\rangle$ we consider three cases. If $z < x$ then we have

$$y \in \langle l \mid {}^{z} \mid r\rangle \cup \{x\} \Leftrightarrow y \in \langle l \mid {}^{z} \mid r \cup \{x\}\rangle \Leftrightarrow y = z \vee y \in l \vee y \in r \cup \{x\} \overset{\mathrm{IH}}{\Leftrightarrow}$$
$$\Leftrightarrow y = z \vee y \in l \vee y \in r \vee y = x \Leftrightarrow y \in \langle l \mid {}^{z} \mid r\rangle \vee y = x.$$

The case when $z = x$ is obvious and the case when $z > x$ is proved similarly. As a simple consequence of ($\dagger_1$) we get

$$\vdash_{\mathrm{PA}} Bt(t) \rightarrow y \prec t \cup \{x\} \leftrightarrow y \prec t \wedge y < x \tag{$\dagger_2$}$$
$$\vdash_{\mathrm{PA}} Bt(t) \rightarrow y \succ t \cup \{x\} \leftrightarrow y \succ t \wedge y > x. \tag{$\dagger_3$}$$

(1): By structural induction on the binary tree t. The base case is straightforward. In the induction step when $t = \langle l \mid {}^{y} \mid r\rangle$ we consider three cases. If $y < x$ then we have

$$Bst(\langle l \mid {}^{y} \mid r\rangle \cup \{x\}) \Leftrightarrow Bst\,\langle l \mid {}^{y} \mid r \cup \{x\}\rangle \Leftrightarrow$$
$$\Leftrightarrow y \succ l \wedge y \prec r \cup \{x\} \wedge Bst(l) \wedge Bst(r \cup \{x\}).$$

The last follows from IH and ($\dagger_2$). The case when $y = x$ is trivial and the case when $y > x$ is proved similarly. □

8.2.7 Deletion of extreme values in binary search trees. In this paragraph we will consider the problem of deletion of extremal values from binary

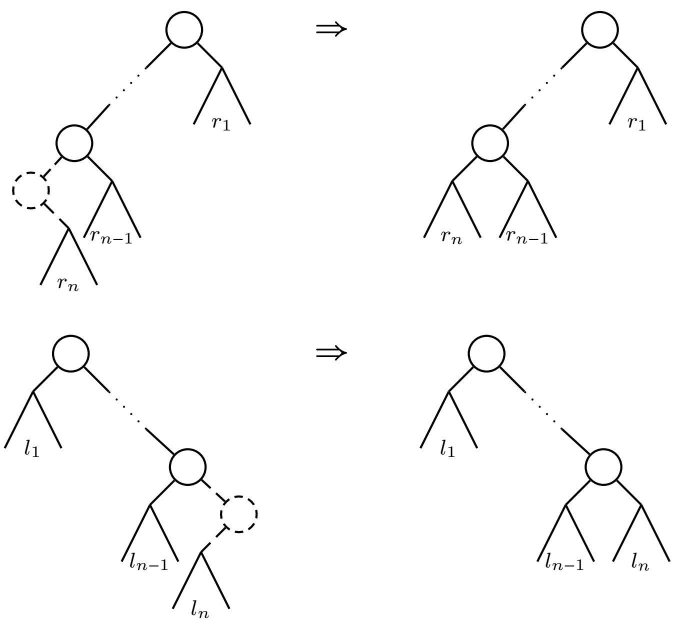

Fig. 8.7 Deletion of extreme values in binary search trees

search trees (see Fig. 8.7). We show here only the implementation and verification of the function deleting the gretest label in a binary search tree. We leave to the reader the implementation and verification of the function deleting the smallest label.

The function $Delmax(t)$ deleting the largest label in the binary search tree t satisfies

$$\vdash_{\mathrm{PA}} \; Bst(t) \land t \neq \langle\rangle \rightarrow Bst\, Delmax(t) \tag{1}$$

$$\vdash_{\mathrm{PA}} \; Bst(t) \land t \neq \langle\rangle \rightarrow x \in Delmax(t) \leftrightarrow x \in t \land x \neq Max(t) \tag{2}$$

and it is defined by structural recursion as a p.r. function (see Fig. 8.7):

$$Delmax\,\langle l \mid x \mid \langle\rangle\rangle = l$$
$$Delmax\,\langle l \mid x \mid r\rangle = \langle l \mid x \mid Delmax(r)\rangle \leftarrow r \neq \langle\rangle.$$

We intend to apply the operation $Delmax(t)$ only in cases when t is a non-empty tree. For that we can take the following property as an alternative (conditional) program for computing the function:

```
⊢PA  $Bt(t) \land t \neq \langle\rangle \rightarrow Delmax(t) =$ let $t = \langle l \mid x \mid r\rangle$ in
                                      case
                                        $r = \langle\rangle \Rightarrow l$
                                        $r \neq \langle\rangle \Rightarrow \langle l \mid x \mid Delmax(r)\rangle$
                                      end.
```

Its condition of regularity

$$\vdash_{\mathrm{PA}} \; Bt(t) \land t \neq \langle\rangle \land t = \langle l \mid x \mid r\rangle \land r \neq \langle\rangle \to r < t \land Bt(r) \land r \neq \langle\rangle$$

is trivially satisfied.

Note also that as a simple consequence of 8.2.5(2) and (2) we have

$$\vdash_{\mathrm{PA}} \; Bst(t) \land t \neq \langle\rangle \to Max(t) \succ Delmax(t). \tag{3}$$

Verification. (2) By structural induction on the binary tree t. In the base case there is nothing to prove. In the induction step when $t = \langle l \mid y \mid r\rangle$ we consider two cases. If $r = \langle\rangle$ then we get the following by noting that $y \notin l$:

$$\begin{aligned} x \in Delmax\,\langle l \mid y \mid \langle\rangle\rangle &\Leftrightarrow x \in l \Leftrightarrow (x = y \lor x \in l) \land x \neq y \Leftrightarrow \\ &\Leftrightarrow x \in \langle l \mid y \mid \langle\rangle\rangle \land x \neq Max\,\langle l \mid y \mid \langle\rangle\rangle. \end{aligned}$$

If $r \neq \langle\rangle$ then we have

$$\begin{aligned} &x \in Delmax\,\langle l \mid y \mid r\rangle \Leftrightarrow x \in \langle l \mid y \mid Delmax(r)\rangle \Leftrightarrow \\ &x = y \lor x \in l \lor x \in Delmax(r) \overset{\mathrm{IH}}{\Leftrightarrow} x = y \lor x \in l \lor x \in r \land x \neq Max(r) \overset{(*)}{\Leftrightarrow} \\ &(x = y \lor x \in l \lor x \in r) \land x \neq Max(r) \Leftrightarrow x \in \langle l \mid y \mid r\rangle \land x \neq Max\,\langle l \mid y \mid r\rangle. \end{aligned}$$

The step marked by $(*)$ follows from $Max(r) \neq y$ and $Max(r) \notin l$ by 8.2.5(1).

As a simple consequence of (2) we get

$$\vdash_{\mathrm{PA}} \; Bst(t) \land t \neq \langle\rangle \land x \prec t \to x \prec Delmax(t)\ . \tag{$\dagger_1$}$$

(1): By structural induction on the binary tree t. In the base case there is nothing to prove. In the induction step when $t = \langle l \mid x \mid r\rangle$ we consider two cases. If $r = \langle\rangle$ then the claim follows directly from the definition. If $r \neq \langle\rangle$ then we have

$$\begin{aligned} &Bst\; Delmax\,\langle l \mid x \mid r\rangle \Leftrightarrow Bst\,\langle l \mid x \mid Delmax(r)\rangle \Leftrightarrow \\ &\qquad \Leftrightarrow x \succ l \land x \prec Delmax(r) \land Bst(l) \land Bst\; Delmax(r). \end{aligned}$$

The last follows from IH and $(\dagger_1)$. □

8.2.8 Deletion in binary search trees. Deletion of labels from binary search trees is much harder than insertion. We wish to define the function $t \smallsetminus \{x\}$ deleting a number x from a binary search tree t with the following specification:

$$\vdash_{\mathrm{PA}} \; Bst(t) \to Bst(t \smallsetminus \{x\}) \tag{1}$$

$$\vdash_{\mathrm{PA}} \; Bst(t) \to y \in t \smallsetminus \{x\} \leftrightarrow y \in t \land y \neq x. \tag{2}$$

The key problem in finding the implementation of the deletion function is: how to define $t \setminus \{x\}$ when x is the root label of t, i.e. when $t = \langle l \mid {}^{x} \mid r \rangle$ for some l and r. Figure 8.8 gives two answers to the problem.

In the first solution the right son r is appended as a new subtree at the bottom right to the left son l. Note that it may happen that the depth of the resulting tree is greater than the depth of t.

In the second solution we take the largest label of the left son l as a new root label with the left son obtained from l by deleting its maximal element and with r as its right son. The result is a tree which depth does not exceed the depth of the original tree.

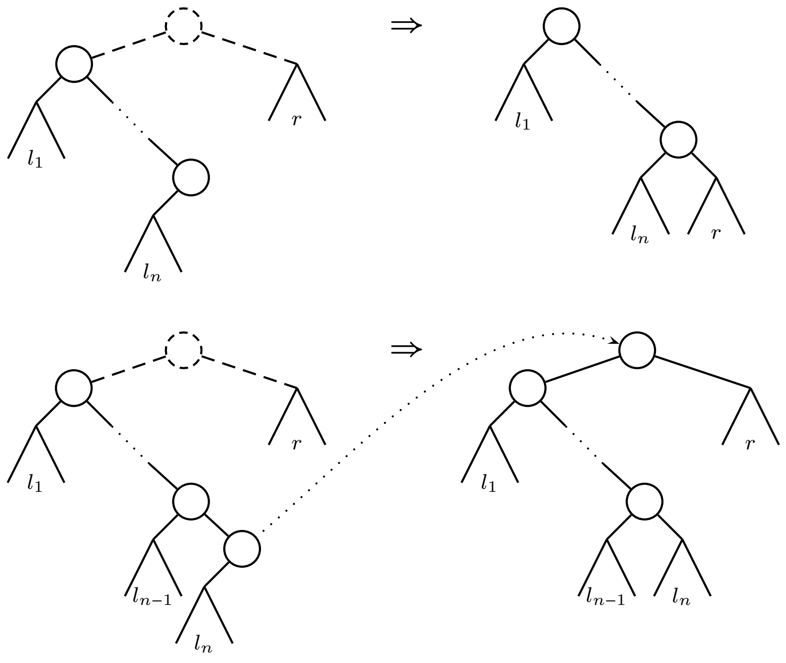

Fig. 8.8 Deletion operation in binary search trees

We therefore take the second method as a basis of our implementation of the deletion function. We define $t \setminus \{x\}$ by structural recursion on t as a primitive recursive function:

$$t_1 \sqcup t_2 = t_2 \leftarrow t_1 = \langle\rangle$$
$$t_1 \sqcup t_2 = \langle \mathit{Delmax}(t_1) \mid {}^{\mathit{Max}(t_1)} \mid t_2 \rangle \leftarrow t_1 \neq \langle\rangle.$$

$$\langle\rangle \setminus \{x\} = \langle\rangle$$
$$\langle l \mid {}^{y} \mid r \rangle \setminus \{x\} = \langle l \mid {}^{y} \mid r \setminus \{x\} \rangle \leftarrow y < x$$
$$\langle l \mid {}^{y} \mid r \rangle \setminus \{x\} = l \sqcup r \leftarrow y = x$$
$$\langle l \mid {}^{y} \mid r \rangle \setminus \{x\} = \langle l \setminus \{x\} \mid {}^{y} \mid r \rangle \leftarrow y > x,$$

where the auxiliary primitive recursive function $t_1 \sqcup t_2$ joining two trees as shown in Fig. 8.8 satisfies

$$\vdash_{\mathrm{PA}} \; Bst(t_1) \wedge Bst(t_2) \wedge t_1 \prec_{\mathrm{b}} t_2 \rightarrow Bst(t_1 \sqcup t_2) \tag{3}$$
$$\vdash_{\mathrm{PA}} \; Bst(t_1) \wedge Bst(t_2) \rightarrow x \in t_1 \sqcup t_2 \leftrightarrow x \in t_1 \vee x \in t_2. \tag{4}$$

Verification. Property (4) is proved by considering two cases. The case when $t_1 = \langle\rangle$ is trivial and in the case when $t_1 \neq \langle\rangle$ we have

$$x \in t_1 \sqcup t_2 \Leftrightarrow x \in \langle Delmax(t_1) \mid {}^{Max(t_1)} \mid t_2 \rangle \Leftrightarrow$$
$$x = Max(t_1) \vee x \in Delmax(t_1) \vee x \in t_2 \overset{8.2.7(2)}{\Leftrightarrow}$$
$$x = Max(t_1) \vee x \in t_1 \wedge x \neq Max(t_1) \vee x \in t_2 \overset{(*)}{\Leftrightarrow} x \in t_1 \vee x \in t_2.$$

The step marked by $(*)$ follows from $Max(t_1) \in t_1$ which holds by 8.2.5(1). In the proof of the property (3) we consider two cases. The case when $t_1 = \langle\rangle$ is trivial and in the case when $t_1 \neq \langle\rangle$ we have

$$Bst(t_1 \sqcup t_2) \Leftrightarrow Bst\, \langle Delmax(t_1) \mid {}^{Max(t_1)} \mid t_2 \rangle \Leftrightarrow$$
$$Max(t_1) \succ Delmax(t_1) \wedge Max(t_1) \prec t_2 \wedge Bst\, Delmax(t_1) \wedge Bst(t_2).$$

The last follows from 8.2.7(3), 8.2.5(1), and 8.2.7(1).

Property (2) is proved by structural induction on t. The base case is straightforward. In the induction step when $t = \langle l \mid {}^{z} \mid r \rangle$ we consider three cases. If $z < x$ then we have

$$y \in \langle l \mid {}^{z} \mid r \rangle \smallsetminus \{x\} \Leftrightarrow y \in \langle l \mid {}^{z} \mid r \smallsetminus \{x\} \rangle \Leftrightarrow y = z \vee y \in l \vee y \in r \smallsetminus \{x\} \overset{\mathrm{IH}}{\Leftrightarrow}$$
$$y = z \vee y \in l \vee y \in r \wedge y \neq x \overset{(*)}{\Leftrightarrow} (y = z \vee y \in l \vee y \in r) \wedge y \neq x \Leftrightarrow$$
$$y \in \langle l \mid {}^{z} \mid r \rangle \wedge y \neq x.$$

The step marked by $(*)$ follows by a simple case analysis on $y = x$ and $y \neq x$ by noting that we have $x \notin l$. The case when $z = x$ follows from (4) by similar arguments. The case when $z > x$ is left to the reader.

As a simple consequence of (2) we get

$$\vdash_{\mathrm{PA}} \; Bst(t) \wedge y < x \rightarrow y \prec t \smallsetminus \{x\} \leftrightarrow y \prec t \tag{5}$$
$$\vdash_{\mathrm{PA}} \; Bst(t) \wedge y > x \rightarrow y \succ t \smallsetminus \{x\} \leftrightarrow y \succ t. \tag{6}$$

Property (1) is proved by structural induction on t. The base case is straightforward. In the induction step when $t = \langle l \mid {}^{y} \mid r \rangle$ we consider three cases. If $y < x$ then we have

$$Bst(\langle l \mid {}^{y} \mid r \rangle \smallsetminus \{x\}) \Leftrightarrow Bst\, \langle l \mid {}^{y} \mid r \smallsetminus \{x\} \rangle \Leftrightarrow$$
$$y \succ l \wedge y \prec r \smallsetminus \{x\} \wedge Bst(l) \wedge Bst(r \smallsetminus \{x\}).$$

The last follows from IH and (5). If $y = x$ then the claim follows from (3). The case when $y > x$ is proved similarly. $\square$