

# Verifying Balanced Trees

Zohar Manna<sup>1</sup>, Henny B. Sipma<sup>1</sup>, Ting Zhang<sup>2</sup>

<sup>1</sup> Stanford University {zm, sipma}@cs.stanford.edu

<sup>2</sup> Microsoft Research Asia tingz@microsoft.com

**Abstract.** Balanced search trees provide guaranteed worst-case time performance and hence they form a very important class of data structures. However, the self-balancing ability comes at a price; balanced trees are more complex than their unbalanced counterparts both in terms of data structure themselves and related manipulation operations. In this paper we present a framework to model balanced trees in decidable first-order theories of term algebras with Presburger arithmetic. In this framework, a theory of term algebras (i.e., a theory of finite trees) is extended with Presburger arithmetic and with certain connecting functions that map terms (trees) to integers. Our framework is flexible in the sense that we can obtain a variety of decidable theories by tuning the connecting functions. By adding *maximal path* and *minimal path* functions, we obtain a theory of red-black trees in which the transition relation of tree self-balancing (rotation) operations is expressible. We then show how to reduce the verification problem of the red-black tree algorithm to constraint satisfiability problems in the extended theory.

## 1 Introduction.

Balanced search trees provide guaranteed worst-case time performance and hence they form a very important class of data structures. Also they are the basis of efficient implementations of many advanced data structures such as associative arrays and associative sets. However, the self-balancing ability comes at a cost; balanced trees are more complex than their unbalanced counterparts both in terms of data structure themselves and related manipulation operations. Moreover, as balanced trees are not regular trees, their properties cannot be directly characterized by standard tree automata techniques [4].

In this paper we present a framework to model balanced trees in decidable first-order theories of term algebras with Presburger arithmetic [23]. In this framework, a theory of term algebras (i.e., a theory of finite trees) is extended with Presburger arithmetic and certain connecting functions that map terms (trees) to integers. Given connecting functions and a fixed signature of a term algebra, the corresponding extended theory is two sorted with *integer sort* and

---

<sup>1</sup> The first and the second author were supported in part by NSF grants CCR-01-21403, CCR-02-20134, CCR-02-09237, CNS-0411363, and CCF-0430102, by ARO grant DAAD19-01-1-0723, and by NAVY/ONR contract N00014-03-1-0939.

*term sort*. The language is the set-theoretic union of the language of term algebras and the language of Presburger arithmetic augmented with connecting functions from  $\mathbb{T}$  to  $\mathbb{N}$ . Formulae are formed from term literals and *integer literals* using logical connectives and quantifications.

Our framework is flexible in the sense that we can obtain a variety of decidable theories by varying the connecting functions. By adding *maximal path* and *minimal path* functions, we obtain a theory of red-black trees in which the transition relation of tree self-balancing (color exchange and rotation) operations are expressible. We then show how to reduce the verification problem of the red-black tree algorithm to a constraint satisfiability problem in the extended theory.

*Related Work and Comparison.* There has been a considerably amount of work in *shape analysis*, a kind of *pointer analysis* aimed at statically inferring properties on heap allocated data structures. Shape analysis tools can partially detect pointer linkage properties such as sharing, aliasing, cyclicity and reachability. The property of being a balanced tree, however, is a much higher-level property that can not be inferred from pointed-to relations on heaps. Rugina presents a method, called quantitative shape analysis, to verify rebalancing operations on AVL trees [19]. Based on abstract interpretation, the method performs forward propagation in an abstract heap where each location (node) is associated with quantitative attributes and relations to characterize the balancing property.

Tree automata techniques are widely used in solving constraints on tree languages [4]. However, balanced trees are not regular trees (by the pumping lemma), and hence their corresponding tree languages cannot be directly characterized by standard tree automata [4]. Habermehl et al. present extended tree automata with size constraints on transition relations (TASC) [10]. TASCs are able to represent pre- and post-conditions of a program involving tree rotation operations. Hence, given the right invariants, the verification of a program reduces to checking the validity of Hoare triples that state that after execution of the program from the starting state satisfying the pre-condition, the resulting reachable states are included in the states represented by the post-condition. However, TASCs that encode transition relations of tree operations are fairly complicated. The lack of intuitive connections between low level program statements and the corresponding automata representations makes this formalism unattractive for use in practical verification tools.

Baldan et al. treat red-black trees as hypergraphs and tree update operations as rewritings on hypergraphs [2]. They use approximate unfolding to compute the reachable states of a graph rewriting system to prove the property that no red node has a red parent. The balancing property itself, however, is not expressible in graph rewriting grammar and an additional type system is introduced to prove it. Calcagno et al. present a context logic to model trees with *local updates*, which are destructive operations at pointed locations [3]. They present a deductive proof system based on Hoare triples and prove soundness and completeness of the proof system. The balancing property, however, is not

expressible this formal system and the verification of Hoare triples is not fully automatic.

Like [10] we reduce the verification problem to checking the validity of Hoare triples. In our approach, however, the pre- and post-conditions and transition relations are all represented directly by the first-order formulas in the extended theory of term algebras. Different from [10,2,3], we do not have an update function in the theory and hence we can not express local updates at an arbitrary pointed location. On the other hand, local updates only affect subtrees around the focus point, and hence our theory can still express and prove verification conditions of step-by-step tree operations. An informal but easy induction will give us global safety properties.

The contributions of this paper are the following: (1) we develop a first-order theory of red-black trees, that is, a theory of term algebras augmented with Presburger arithmetic; (2) we show how to use this theory to represent the transition relations of the tree operations directly from the program statements, and how to use them to construct Hoare triples; (3) we provide a decision procedure for automatically checking validity of the resulting verification conditions. To the best of our knowledge, this is the first decidable logic theory for red-black trees. Moreover, it can be easily generalized to model other balanced tree structures, such as AVL trees and B-trees.

*Paper Organization* Section 2 presents the notation and terminology for term algebras. Section 3 introduces the theory of red-black trees and states its decidability result. Section 4 shows how to use the theory to analyze the red-black tree insertion algorithm. Section 5 concludes with a discussion of future work. Because of space limitations all decidability proofs have been omitted. They are available for reference on the third author's web site.

## 2 Preliminaries

We assume the first-order syntactic notions of variables, parameters and quantifiers, and semantic notions of structures, satisfiability and validity as in [9]. We use  $\llbracket x \rrbracket$  to denote the value given by an assignment and  $\bar{x}$  to denote a sequence of variables.

**Definition 1 (Term Algebras).** *A term algebra  $TA : \langle \mathbb{T}; C, \mathcal{A}, \mathcal{S}, \mathcal{T} \rangle$  consists of*

1.  $\mathbb{T}$ : *The term domain, which exclusively consists of terms recursively built up from constants by applying non-nullary constructors. Objects in  $\mathbb{T}$  are called TA-terms. The type of a term  $t$ , denoted by  $\text{type}(t)$ , is the outermost constructor symbol of  $t$ . We say that  $t$  is  $\alpha$ -typed (or is an  $\alpha$ -term) if  $\text{type}(t) = \alpha$ .*
2.  $C$ : *A set of constructors:  $\alpha, \beta, \gamma, \dots$ . The arity of  $\alpha$  is denoted by  $\text{ar}(\alpha)$ .*
3.  $\mathcal{A}$ : *A set of constants:  $a, b, c, \dots$ . We require  $\mathcal{A} \neq \emptyset$  and  $\mathcal{A} \subseteq C$ . For  $a \in \mathcal{A}$ ,  $\text{ar}(a) = 0$  and  $\text{type}(a) = a$ .*
4.  $\mathcal{S}$ : *A set of selectors. For a constructor  $\alpha$  with arity  $k > 0$ , there are  $k$  selectors  $s_1^\alpha, \dots, s_k^\alpha$  in  $\mathcal{S}$ . We call  $s_i^\alpha$  ( $1 \leq i \leq k$ ) the  $i^{\text{th}}$   $\alpha$ -selector. For a term  $x$ ,  $s_i^\alpha(x)$  returns the  $i^{\text{th}}$  immediate subterm of  $x$  if  $x$  is an  $\alpha$ -term and  $x$  itself otherwise.*

5.  $\mathcal{T}$ : A set of testers. For each constructor  $\alpha$  there is a corresponding tester  $\text{Is}_\alpha$ . For a term  $x$ ,  $\text{Is}_\alpha(x)$  is true if and only if  $x$  is an  $\alpha$ -term. For a constant  $a$ ,  $\text{Is}_a(x)$  is just  $x = a$ . In addition there is a special tester  $\text{Is}_A$  such that  $\text{Is}_A(x)$  is true if and only if  $x$  is a constant.

We use  $\mathcal{L}_T$  to denote the language of term algebras.

Term domain  $\mathbb{T}$  consists of only ground terms built from constructors. Selectors only exist in the *formal* language. In fact selectors can be defined by constructors in the existential fragment of the language; for a quantifier-free formula  $\Phi(\bar{x})$  containing selectors, we can obtain an equivalent and selector-free formula  $\exists \bar{y} \Phi'(\bar{x}, \bar{y})$  where  $\Phi'(\bar{x}, \bar{y})$  is quantifier-free and  $\bar{y}$  is fresh.

A term  $t$  is called a *constructor term* if  $t$  is a variable or the outermost function symbol of  $t$  is a constructor. Constants are constructor terms. A term  $t$  is called a *selector term* if either  $t$  is a variable or the outermost function symbol of  $t$  is a selector. Variables are both constructor terms and selector terms. We assume that no constructors appear immediately inside selectors as simplification can always be done. A term is called *proper* if it is not a constant or a variable.

The first-order theory of term algebras was shown to be decidable by Mal'cev using quantifier elimination [15]. Decision procedures for the quantifier-free theory were discovered by Nelson, Oppen et al. [16, 17, 8]. Oppen gave a linear algorithm for acyclic structures [17] and (with Nelson) a quadratic algorithm for cyclic structures [16]. If the values of the selector functions on constants are specified, then the problem is NP-complete [17].

Presburger arithmetic is the first-order theory of addition in the arithmetic of integers. The corresponding structure is denoted by  $\text{PA} = \langle \mathbb{Z}; 0, +, < \rangle$ . We use  $\mathcal{L}_{\mathbb{Z}}$  to denote the formal language of PA. The first-order theory of Presburger arithmetic (PA) was first shown to be decidable in 1929 by the quantifier elimination method [9]. More efficient algorithms were later discovered by [6] and further improved in [18].

There has been a great interest in generalizing Mal'cev's result on term algebras. Maher showed the decidability of the theory of infinite and rational trees [14]. Comon and Delor presented an elimination procedure for term algebras with membership predicate in the regular tree language [5]. Backofen presented an elimination procedure for structures of feature trees with arity constraints [1]. Rybina and Voronkov showed the decidability of term algebras with queues [20]. Kuncak and Rinard showed the decidability of term powers, which are term algebras augmented with coordinate-wise defined predicates [13]. A combination of Presburger arithmetic and term algebras was used by Korovin and Voronkov to show that the quantifier-free theory of term algebras with Knuth-Bendix order is NP-complete [11, 12]. In [21, 23] we presented decision procedures both for the first-order theory and the corresponding quantifier-free fragments of term algebras with integer functions. In [22] we extended the decidability result to the first-order theory of term algebras with Knuth-Bendix order.

### 3 The Theory of Red-Black Trees

In this section we present a theory of a term algebra with two integer functions to express the properties of red-black trees.

**Definition 2 (Red-black Trees [7]).** A red-black tree is a binary tree with the following coloring properties:

1. Every node is either red or black.
2. Every leaf node is black.
3. The root is black.
4. Every red node has two black children.
5. All paths from the root to leaf nodes contain the same number of black nodes.

Properties (1)-(3) can be modeled in a theory of term algebras as follows:

**Definition 3 (Structure of Colored Trees).** The structure of red-black colored trees is

$$\text{RB} = \langle \mathbb{T}_{\text{rb}}; \{\text{red}, \text{black}, \text{nil}\}, \{\text{nil}\}, \\ \{\text{car}^{\text{red}}, \text{cdr}^{\text{red}}, \text{car}^{\text{black}}, \text{cdr}^{\text{black}}\}, \{\text{Is}_{\text{red}}, \text{Is}_{\text{black}}, \text{Is}_{\text{nil}}\} \rangle ,$$

where  $\mathbb{T}_{\text{rb}}$  denotes the domain,  $\text{nil}$  denotes a leaf,  $\text{red}$  and  $\text{black}$  are binary constructors,  $\text{car}^{\#}$  and  $\text{cdr}^{\#}$ , respectively, are the left and the right  $\#$ -selectors ( $\# \in \{\text{red}, \text{black}\}$ ). The corresponding language is denoted by  $\mathcal{L}_{\text{RB}}$ .

For notation simplicity we use  $\text{car}$  to denote either  $\text{car}^{\text{red}}$  or  $\text{car}^{\text{black}}$ , which should be clear from the context. Similar for the use of  $\text{cdr}$ . If a term  $t$  appears in a selector, we assume either  $\text{Is}_{\text{red}}(t)$  or  $\text{Is}_{\text{black}}(t)$  holds. For example,  $\text{car}(x) = y$  should be understood as an abbreviation of

$$(\text{Is}_{\text{red}}(x) \wedge y = \text{car}^{\text{red}}(x)) \vee (\text{Is}_{\text{black}}(x) \wedge y = \text{car}^{\text{black}}(x)) .$$

In the following we use *terms* and *trees*, respectively, to refer to syntactic objects and semantic objects. We call terms (trees) of red-type (resp. of black-type) *red-terms (-trees)* (resp. *black-terms (-trees)*).

We extend RB with PA to express balancing properties (4)-(5):

**Definition 4 (Structure of Red-black Trees).** The structure of red-black trees is

$$\text{RB}_{\mathbb{Z}} = \langle \text{RB}; \text{PA}; |\cdot|_{\text{max}}, |\cdot|_{\text{min}} : \mathbb{T}_{\text{rb}} \rightarrow \mathbb{N} \rangle ,$$

where,  $|\cdot|_{\text{max}}$  and  $|\cdot|_{\text{min}}$  are two integer functions defined recursively as

$$|x|_{\star} = \begin{cases} 1 & x = \text{nil} , \\ 0 & \text{Vio}(x) , \\ \star(|x_1|_{\star}, |x_2|_{\star}) + 1 & \text{GB}(x, x_1, x_2) , \\ \star(|x_1|_{\star}, |x_2|_{\star}) & \text{GR}(x, x_1, x_2) . \end{cases}$$

where  $\star \in \{\max, \min\}$  and  $\text{GB}(x, x_1, x_2)$ ,  $\text{GR}(x, x_1, x_2)$  and  $\text{Vio}(x)$  are

$$\begin{aligned} \text{Vio}(x) &\stackrel{\text{def}}{=} x \neq \text{nil} \wedge \forall x_1 \forall x_2 \left( \neg \text{GB}(x, x_1, x_2) \vee \neg \text{GR}(x, x_1, x_2) \right), \\ \text{GB}(x, x_1, x_2) &\stackrel{\text{def}}{=} x = \text{black}(x_1, x_2) \wedge |x|_{\max} \neq 0 \wedge |x|_{\min} \neq 0, \\ \text{GR}(x, x_1, x_2) &\stackrel{\text{def}}{=} x = \text{red}(x_1, x_2) \wedge |x|_{\max} \neq 0 \wedge |x|_{\min} \neq 0 \\ &\quad \wedge \neg \text{Is}_{\text{red}}(x_1) \wedge \neg \text{Is}_{\text{red}}(x_2). \end{aligned}$$

We denote the corresponding language by  $\mathcal{L}_{\text{RB}}^{\mathbb{Z}}$ .

$\text{Vio}(x)$  states that  $x$  violates property (4) of red-black trees.  $\text{GB}(x, x_1, x_2)$  states  $x$  is a black tree with two good subtrees  $x_1$  and  $x_2$ . Similarly for  $\text{GR}(x, x_1, x_2)$ .  $|x|_{\max}$  (resp.  $|x|_{\min}$ ) gives the maximal (resp. minimal) number of black nodes that  $x$  can have on a maximal path. A maximal path of  $x$  that contains the largest (resp. smallest) number of black nodes is called a *maximal black path* (resp. *minimal black path*) of  $x$ . We call  $|x|_{\max}$  the *maximal black length* of  $x$ ,  $|x|_{\min}$  the *minimal black length*, and the pair  $(|x|_{\max}, |x|_{\min})$  the *measure* of  $x$ , denoted by  $\|x\|$ .

In this theory, properties (1) and (2) of Definition 2, which state that every node is either black or red, and that a nil node is black, are trivially satisfied by the choice of signature and the integer functions. Therefore  $x$  is a red-black tree if  $x$  satisfies the following three conditions.

- (§1)  $|x|_{\max} = |x|_{\min}$  *any maximal path of  $x$  contains the same number of black nodes,*
- (§2)  $|x|_{\max} > 0$  *any red node of  $x$  must have two black children,*
- (§3)  $\text{Is}_{\text{black}}(x)$  *the root of  $x$  is black.*

We denote by  $\varphi_{\text{RB}}^-(x)$  the conjunction of (§1) and (§2), and by  $\varphi_{\text{RB}}(x)$  the conjunction of (§1)-(§3). We note that  $\varphi_{\text{RB}}(x)$  defines a subdomain of  $\mathbb{T}_{\text{rb}}$  and the theory of this subdomain can be obtained by relativizing quantifiers to  $\varphi_{\text{RB}}(x)$ . Formally,  $\forall x(\varphi_{\text{RB}}(x) \rightarrow \Phi(x))$  (resp.  $\exists x(\varphi_{\text{RB}}(x) \wedge \Phi(x))$ ) expresses that  $\Phi(x)$  is a universal (resp. existential) property of red-black trees. We have

**Theorem 1 (Decidability of  $\text{RB}_{\mathbb{Z}}$ ).**

1. *The first-order theory of  $\text{RB}_{\mathbb{Z}}$  is decidable and admits quantifier elimination.*
2. *The decision problem for the quantifier-free fragment is NP-complete.*

## 4 Analysis of Red-black Trees

### 4.1 Algorithm and Example

In this section we consider the insertion-fixup operation of red-black trees represented by Algorithm 2, a slightly modified version of the algorithm given in [7]. We illustrate the algorithm on the same example as in [7], inserting 4 at the bottom of the tree and showing how the algorithm restores the red-black tree property.

**Algorithm 2** (RB-INSERTION-FIXUP)Input:  $root$ ,  $T$ ,  $x$ .

```

1: while ( $x \neq root$  and  $T[x-1].color = \text{RED}$ ) do
2:   if  $T[x-1].dir = \text{right}$  then
3:     if ( $T[x-1].tree$  IS  $\text{RED}$ ) {Case 1} then
4:        $T[x-1].tree := \text{BLACK}(\text{car}(T[x-1].tree), \text{cdr}(T[x-1].tree))$ 
5:        $T[x-1].color := \text{BLACK}$ 
6:        $T[x-2].color := \text{RED}$ 
7:        $x := x-2$ 
8:     else
9:       if ( $T[x].dir = \text{left}$ ) {Case 2} then
10:         $\text{swap}(T[x].tree, T[x+1].tree)$ 
11:         $T[x].dir := \text{right}$ 
12:         $T[x+1].dir := \text{left}$ 
13:      end if
14:       $T[x-1].color := \text{BLACK}$  {Case 3}
15:       $T[x].tree := \text{RED}(T[x].tree, T[x-1].tree)$ 
16:       $T[x-1].tree := T[x-2].tree$ 
17:       $T[x-1].dir := T[x-2].dir$ 
18:      if ( $x-2 \neq root$ ) then
19:         $x-3+1 := x-1$ 
20:      else
21:         $root := x-1$ 
22:      end if
23:    end if
24:  else if ( $T[x-1].dir = \text{left}$ ) then
25:    similar code as the then clause with left and right swapped
26:  end if
27: end while
28:  $T[root].color := \text{BLACK}$ 

```

Recall that our language does not have an update function to express the relation between the original tree and updated tree if the update happens at an unbounded depth inside the tree. We know that the restoring updates will begin at the newly inserted node and traverse upwards to the root and that all local updates will happen on this path. We represent the tree as a sequence of subtrees indexed by nodes on the path from the root to the newly inserted node. We treat the path as a doubly linked list (denoted by  $T$  in the algorithm) in which each element contains three fields,  $.color$ ,  $.dir$  and  $.tree$ . Field  $.color$  denotes the type of the node. Field  $.dir$  indicates whether the subtree at this node is the left child or the right child. Field  $.tree$  denotes the sibling subtree of this node. We have  $root.dir = \perp$  and  $root.tree = \perp$ . For simplicity, we omit the value field as it has no role in restoring the red-black tree property. We treat  $root$  and  $x$  as iterators and we use array notation  $T[x]$  to denote the element pointed to by  $x$ . We use  $x+1$  and  $x-1$  to denote previous iterator and next iterator of  $x$ , respectively. For example, the statement  $x+3-1 = x-1$  at line 19 means  $x.pre.pre.pre.next := x.pre$ .

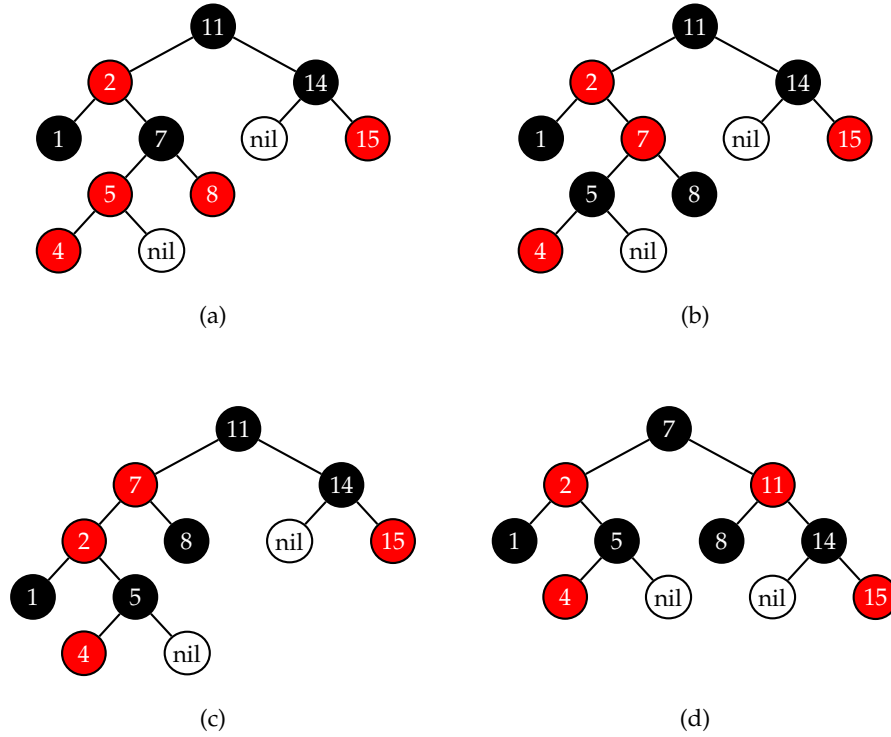


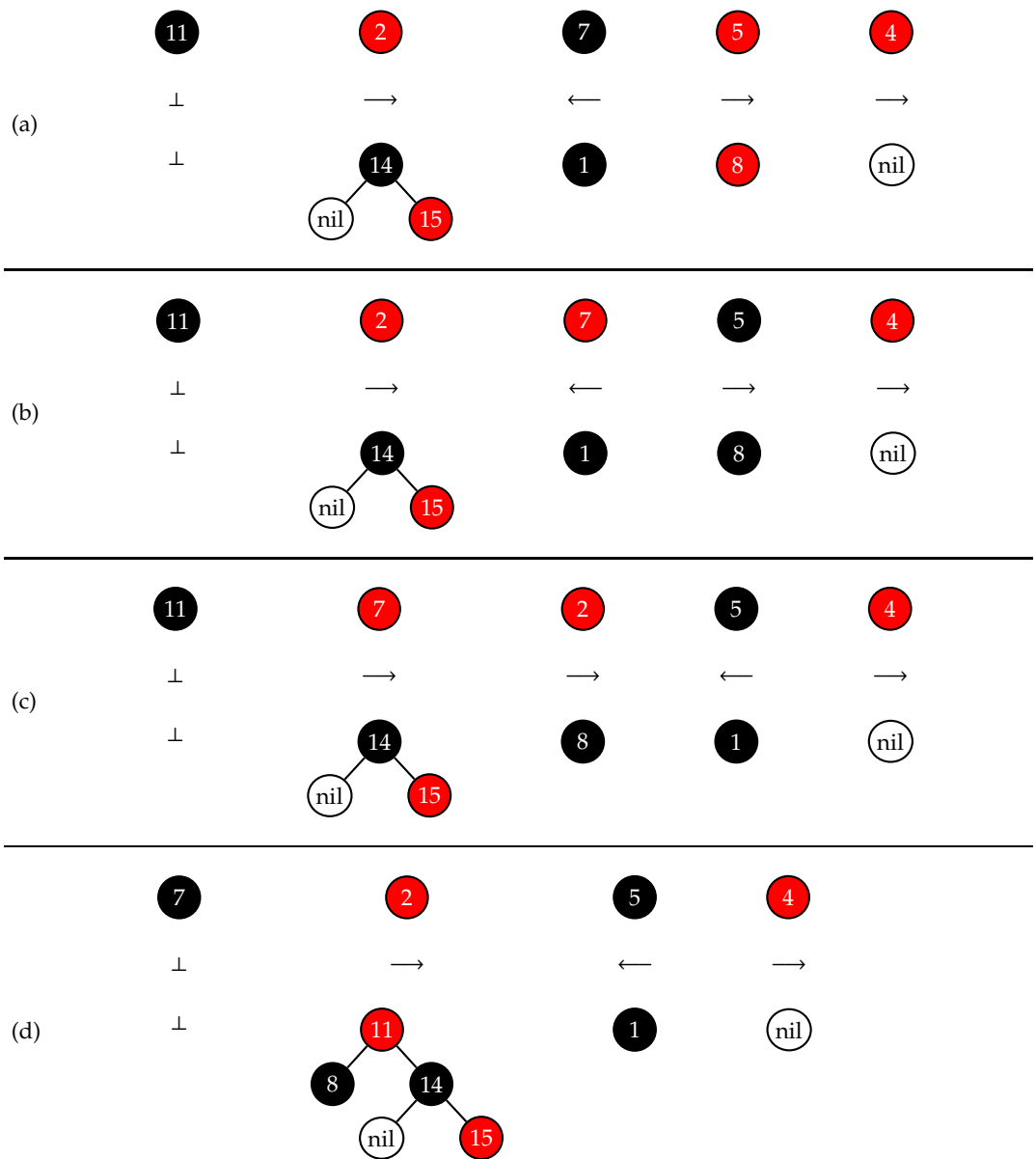
Fig. 1. A run of RB-INSERTION-FIXUP.

Figure 1 shows the results of the operations performed to restore the balanced-tree property after inserting 4. Figure 2 gives a more detailed picture of the data structures of the nodes on the path from the root to  $x$ . Figure 1 (b) shows the tree obtained by recoloring. The new violation now corresponds to Case 2 in Algorithm 2. Figure 1 (c) shows the tree obtained from a left rotation. There is still a violation which corresponds to Case 3 in Algorithm 2. Figure 1 (d) shows a new red-black tree after a right rotation. Figures 2 (b)-(d) show the corresponding changes of the data structure during the run<sup>3</sup>.

## 4.2 Verification Conditions

We now show how to use  $\mathcal{L}_{RB}^Z$  to express the verification conditions for statements restoring that red-black tree property in Algorithm 2. Recall that in the algorithm  $x$  is an iterator and  $T[x]$  is a node pointed by  $x$  in a linked list and it contains three fields,  $.dir$ ,  $.color$  and  $.tree$ . At the semantic level, however, we view  $x$  as an integer index and  $T[x]$  as a subtree indexed by  $x$ . If  $x \neq root$ , then  $T[x].tree$  denotes the sibling tree of  $T[x]$ , and  $T[x - 1]$  represents the immediate

<sup>3</sup> To save space, in all figures we do not draw a nil node if its sibling is not nil, and for this reason, we do not draw nil nodes black.



**Fig. 2.** Paths from the root of the tree to  $x$ . In each of (a)-(d), the first row shows the sequence of nodes from the root to  $x$ ; the second row shows whether the node above it is a left ( $\leftarrow$ ) or right ( $\rightarrow$ ) sibling; the third row shows the sibling tree of the node in the top row.

super-tree containing  $T[x]$ . For example, if  $T[x].dir$  is right and  $T[x-1].color$  is red, then  $T[x-1] = \text{red}(T[x], T[x].tree)$ . We have three field operators,  $.dir$ ,  $.color$  and  $.tree$ . Among them  $.dir$  can only take three values,  $left$ ,  $right$  and  $\perp$ , so expressions involving  $.dir$  can be removed by disjunctive splitting. Similar for  $.color$ , but it can be directly expressed in  $\mathcal{L}_{RB}$  as below.

$$\begin{aligned} T[x].color = \text{red} &\stackrel{\text{def}}{=} \text{Is}_{\text{red}}(T[x]) , \\ T[x].color = \text{black} &\stackrel{\text{def}}{=} x = \text{nil} \vee \text{Is}_{\text{black}}(T[x]) . \end{aligned}$$

With the help of  $.dir$ ,  $.tree$  can be expressed in  $\mathcal{L}_{RB}$  as follows.

$$\begin{aligned} T[x].tree = y \neq \perp &\stackrel{\text{def}}{=} x \neq \text{root} \wedge \left( (y = \text{car}(T[x-1]) \wedge T[x].dir = \text{right}) \right. \\ &\quad \left. \vee (y = \text{cdr}(T[x-1]) \wedge T[x].dir = \text{left}) \right) , \\ T[x].tree = \perp &\stackrel{\text{def}}{=} x = \text{root} . \end{aligned}$$

Therefore from now on we treat field access expressions as abbreviations in  $\mathcal{L}_{RB}$ . Note that we use array and record notations for clarity. At the formula level terms of index access or field access are simply variables. For example,  $T[x]$ ,  $T[x].tree$ ,  $T[x].color$  and  $T[x].dir$  can be represented by variables  $f_x$ ,  $g_x$ ,  $h_x$  and  $k_x$  indexed by  $x$ , respectively. Similarly for terms indexed by  $x-i$  and  $x+i$ .

Let  $\bar{v}$  denote the variables in the current state and  $\bar{v}'$  denote the corresponding variables in the next state. The transition relation of a statement  $q$  is denoted by  $\rho_q(\bar{v}, \bar{v}')$ . The post-condition  $\text{post}(q, \varphi)$  of  $\varphi(\bar{v})$  after executing a statement  $q$  is

$$(\exists \bar{v}^0) \left( \rho_q(\bar{v}^0, \bar{v}) \wedge \varphi(\bar{v}^0) \right) .$$

The transition relation of two sequential statements can be computed as follows. Let  $\rho_q(\bar{v}, \bar{v}^1)$  and  $\rho_r(\bar{v}^1, \bar{v}')$  be the transition relations for statements  $q$  and  $r$  respectively. Then the transition relation of the composite statement  $\langle q; r \rangle$  is

$$(\exists \bar{v}^1) \left( \rho_q(\bar{v}, \bar{v}^1) \wedge \rho_r(\bar{v}^1, \bar{v}') \right) .$$

The validity checking of a Hoare triples  $\{\varphi\}q\{\psi\}$  is equivalent to proving that  $\text{post}(q, \varphi) \rightarrow \psi$ .

The lack of update functions makes it impossible to express the tree operational semantics precisely in a finite formula. For example, when  $T[x]$  is changed, not only should  $T'[x]$  appear in  $\rho_q(\bar{v}, \bar{v}')$ , but also all ancestors of  $T[x]$ . In fact  $\rho_q(\bar{v}, \bar{v}')$  has an unbounded number of conjuncts of the form  $\text{car}(T'[x-i]) = T'[x-i+1]$  or  $\text{cdr}(T'[x-i]) = T'[x-i+1]$ . We can still, however, prove *safety properties* about tree operations with the help of an informal induction. As an example, we show that  $\varphi_{RB}^-(T[x])$ , introduced in Section 3, is an invariant with respect to each code fragment (corresponding to Case 1, 2 or 3 in Algorithm 2). This can be obtained by establishing the Hoare triple  $\{\varphi\}Q\{\psi\}$  where  $\varphi$  is the pre-condition

$$\begin{aligned} x \neq \text{root} \wedge x-1 \neq \text{root} \rightarrow \\ \left( \varphi_{RB}^-(T[x]) \wedge \varphi_{RB}^-(T[x].tree) \wedge \neg \varphi_{RB}^-(T[x-1]) \wedge \varphi_{RB}^-(T[x-1].tree) \right) \end{aligned}$$

$\psi$  is the post-condition  $\varphi_{\text{RB}}^-(T[x])$ , and  $Q$  is a code fragment corresponding to Case 1, 2 or 3. Here we need another invariant  $\forall x(x \neq \text{root} \rightarrow \varphi_{\text{RB}}^-(T[x].\text{tree}))$ . This invariant can not be formally proved in our theory because of the universal quantification on indexes. But it is easy to verify that the parametric Hoare triples

$$\{x \neq \text{root} \rightarrow \varphi_{\text{RB}}^-(T[x \pm i].\text{tree})\} \quad q \quad \{x \neq \text{root} \rightarrow \varphi_{\text{RB}}^-(T[x \pm i].\text{tree})\}$$

can be established for each statement  $q$  not modifying index  $x$  (see below for the transition relations of those statements). In the following we list *local* transition relations of all statements involving tree update and use guard conditions to simplify those transition relations.

Case 1 is implemented by statements 4-7. The guard conditions are  $x \neq \text{root} \wedge \text{Is}_{\text{red}}(T[x-1])$  (line 1),  $T[x-1].\text{dir} = \text{right}$  (line 2) and  $\text{Is}_{\text{red}}(T[x-1].\text{tree})$  (line 3). Under these conditions the transition relations for statements 4-7 are, respectively,

$$\begin{aligned} T'[x-1].\text{tree} &= \text{cdr}(T'[x-2]) \\ &= \text{black}(\text{car}(T[x-1].\text{tree}), \text{cdr}(T[x-1].\text{tree})) , \end{aligned} \quad (\text{S-4})$$

$$\text{car}(T'[x-2]) = T'[x-1] = \text{black}(\text{car}(T[x-1]), \text{cdr}(T[x-1])) , \quad (\text{S-5})$$

$$T'[x-2] = \text{red}(\text{car}(T[x-2]), \text{cdr}(T[x-2])) , \quad (\text{S-6})$$

$$x' = x - 2 . \quad (\text{S-7})$$

The composite transition relation for statements 4-7 is

$$\begin{aligned} &T'[x-1].\text{tree} = \text{black}(\text{car}(T[x-1].\text{tree}), \text{cdr}(T[x-1].\text{tree})) \\ \wedge \quad &T'[x-1] = \text{black}(\text{car}(T[x-1]), \text{cdr}(T[x-1])) \\ \wedge \quad &T'[x-2] = \text{red}(T'[x-1], T'[x-1].\text{tree}) \\ \wedge \quad &x' = x - 2 . \end{aligned}$$

Recall that  $T[x]$ ,  $T[x].\text{tree}$ ,  $T[x].\text{color}$  and  $T[x].\text{dir}$  are just more informative aliases of indexed variables  $f_x$ ,  $g_x$ ,  $h_x$  and  $k_x$ , respectively. Similarly for terms indexed by  $x-1$  and  $x-2$ . The next state variable for  $T[x]$  should be  $T'[x']$ , but by default we write  $T'[x]$  when  $x' = x$ . When we do transition relation composition, statement 7 requires us to hard code the integer indexing properties in the formula by adding equalities like  $T'[x-1] = T'[x'+1]$ ,  $T'[x-2].\text{tree} = T'[x'].\text{tree}$  and so on. To save space, however, we omit them in the above example.

Figure 3 illustrates Case 1 by a run on tree (a) in Figure 1 (copied as (b-0)). Trees (b-1), (b-2) and (b-3) are the outcomes of statements 4, 5 and 6, respectively.

The code fragment for Case 2 consists of statements 10-12. We take into account the conditions  $T[x-1].\text{color} = \text{red}$  (line 1),  $T[x-1].\text{dir} = \text{right}$  (line 2) and  $T[x].\text{dir} = \text{left}$  (line 9). Under these condition the transition relations for statements 10-12 are, respectively,

$$\text{cdr}(T'[x-1]) = T'[x] \wedge (T'[x+1].\text{tree} = \text{cdr}(T'[x]) = T[x].\text{tree})$$

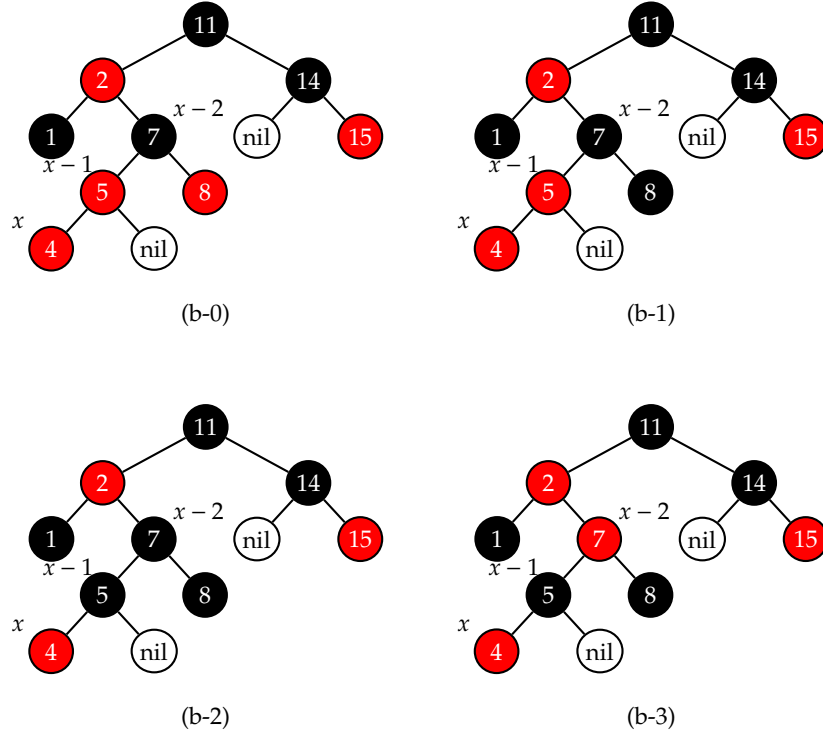


Fig. 3. A detailed run of RB-INSERTION-FIXUP step (b).

$$\wedge (T'[x].tree = \text{car}(T'[x-1]) = T[x+1].tree) , \quad (\text{S-10})$$

$$T'[x].dir = \text{right} \wedge T'[x-1] = \text{red}(\text{cdr}(T[x-1]), \text{car}(T[x-1])) , \quad (\text{S-11})$$

$$T'[x+1].dir = \text{left} \wedge \text{car}(T'[x-1]) = T'[x] \\ \wedge T'[x] = \text{red}(\text{cdr}(T[x]), \text{car}(T[x])) . \quad (\text{S-12})$$

The composite transition relation for statements 10-12 is

$$T'[x+1].tree = T[x].tree \wedge T'[x].tree = T[x+1].tree \\ \wedge T'[x].dir = \text{right} \wedge T'[x-1] = \text{red}(T'[x], T[x+1].tree) \\ \wedge T'[x+1].dir = \text{left} \wedge T'[x] = \text{red}(T[x].tree, \text{car}(T[x])) .$$

Figure 4 illustrates Case 2 by a run on tree (b) in Figure 1 (copied as (c-0)). Trees (c-1), (c-2) and (c-3) are the outcomes of statements 10, 11 and 12, respectively. Recall that we ignored value labels at internal nodes as they are irrelevant to the red-black tree properties. But the binary search tree property may be violated without adjusting the value labels. So for the sake of illustration we switched positions of 2 and 7 in Figure 4 (c-1) although statement 10 does not have this effect.

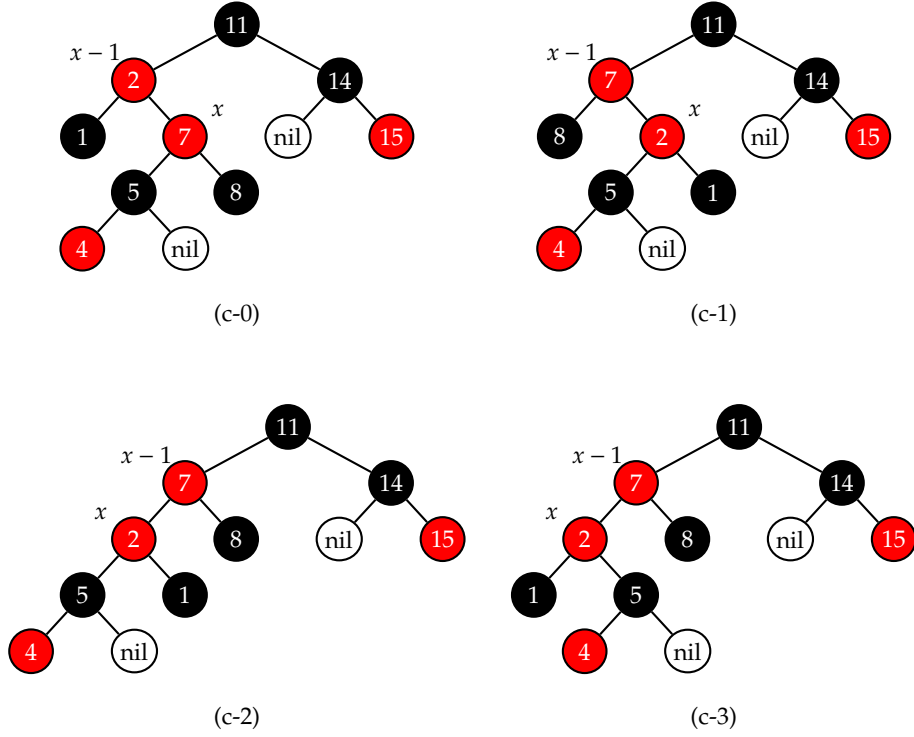


Fig. 4. A detailed run of RB-INSERTION-FIXUP step (c).

Case 3 consists of statements 14-21. We take into account the conditions  $T[x-1].color = red$  (line 1),  $T[x-1].dir = right$  (line 2) and  $T[x].dir = right$  (line 11). Under these conditions the transition relations for statements 14-21 are, respectively,

$$\text{car}(T'[x-2]) = T'[x-1] = \text{black}(\text{car}(T[x-1]), \text{cdr}(T[x-1])) , \quad (\text{S-14})$$

$$\text{cdr}(T'[x-1]) = T'[x].tree = \text{red}(T[x].tree, T[x-1].tree) , \quad (\text{S-15})$$

$$T'[x-1].tree = T[x-2].tree \\ \wedge (x-2 \neq \text{root} \rightarrow \text{cdr}(T'[x-2]) = T[x-2].tree) , \quad (\text{S-16})$$

$$T'[x-1].dir = T[x-2].dir \\ \wedge (x-2 \neq \text{root} \wedge T[x-1].dir \neq T[x-2].dir \rightarrow \\ T'[x-2] = \text{black}(\text{cdr}(T[x-2]), \text{car}(T[x-2]))) , \quad (\text{S-17})$$

$$x'-2 = x-3 \wedge ((\text{cdr}(T'[x-3]) = T[x-1] \wedge T[x-2].dir = \text{left}) \\ \vee (\text{car}(T'[x-3]) = T[x-1] \wedge T[x-2].dir = \text{right})) , \quad (\text{S-19})$$

$$x'-1 = \text{root} . \quad (\text{S-21})$$

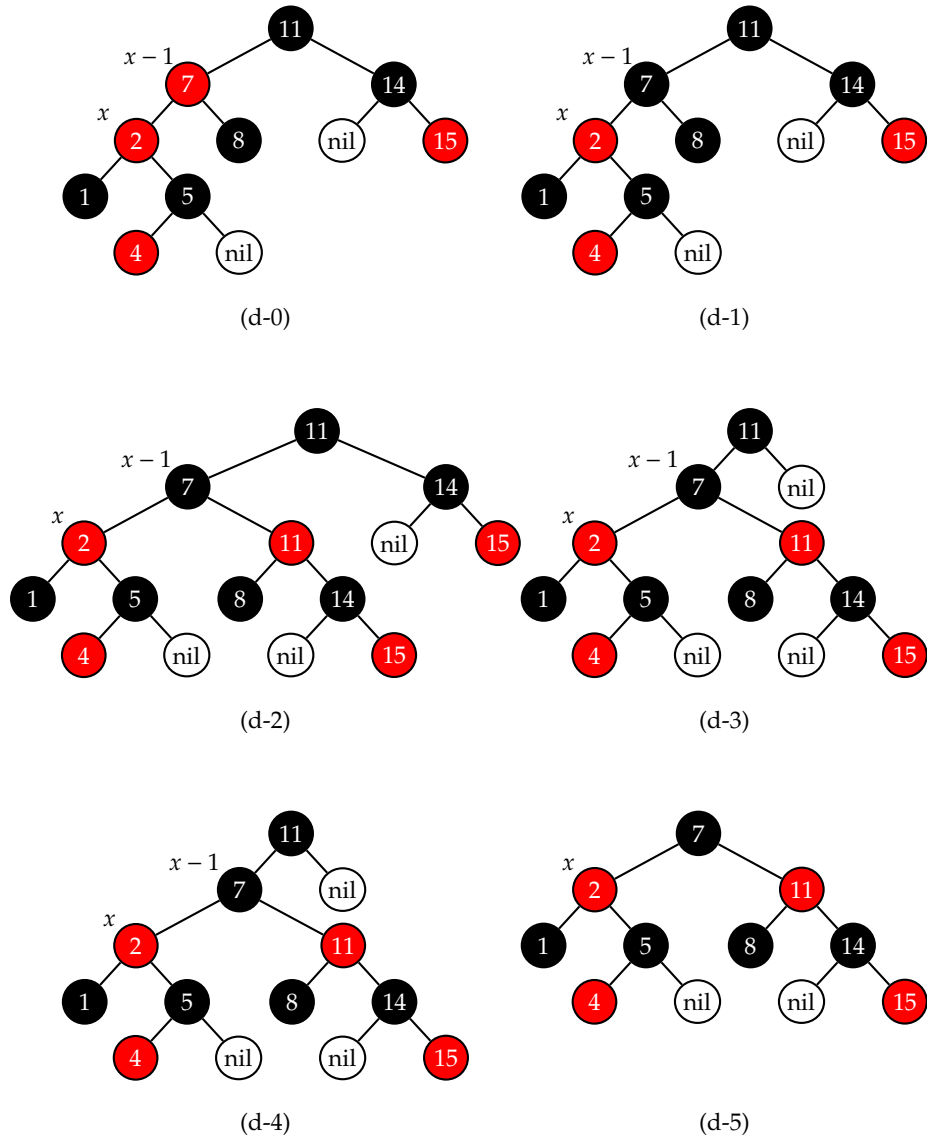


Fig. 5. A detailed run of RB-INSERTION-FIXUP step (d) with  $x - 2 = \text{root}$ .

Assuming  $x - 2 = \text{root}$ , the composite transition relation for statements 14-21 is

$$\begin{aligned}
 & \text{car}(T'[x - 2]) = T'[x - 1] = \text{black}(\text{car}(T[x - 1]), T'[x].\text{tree}) \\
 \wedge & \quad T'[x].\text{tree} = \text{red}(T[x].\text{tree}, T[x - 1].\text{tree}) \\
 \wedge & \quad T'[x - 1].\text{tree} = T[x - 2].\text{tree} \wedge T'[x - 1].\text{dir} = T[x - 2].\text{dir} \\
 \wedge & \quad x' - 1 = \text{root} .
 \end{aligned}$$

Assuming  $x - 2 \neq \text{root}$ , the composite transition relation for statements 14-21 is

$$\begin{aligned}
& \text{car}(T'[x - 2]) = T'[x - 1] = \text{black}(\text{car}(T[x - 1]), T'[x].\text{tree}) \\
\wedge & T'[x].\text{tree} = \text{red}(T[x].\text{tree}, T[x - 1].\text{tree}) \\
\wedge & T'[x - 1].\text{tree} = T[x - 2].\text{tree} \wedge \text{cdr}(T'[x - 2]) = T[x - 2].\text{tree} \\
\wedge & T'[x - 1].\text{dir} = T[x - 2].\text{dir} \wedge (T[x - 1].\text{dir} \neq T[x - 2].\text{dir} \rightarrow \\
& \quad T'[x - 2] = \text{black}(\text{cdr}(T[x - 2]), \text{car}(T[x - 2]))) \\
\wedge & x' - 2 = x - 3 \wedge ((\text{cdr}(T'[x - 3]) = T[x - 1] \wedge T[x - 2].\text{dir} = \text{left}) \\
& \quad \vee (\text{car}(T'[x - 3]) = T[x - 1] \wedge T[x - 2].\text{dir} = \text{right})) .
\end{aligned}$$

Figure 5 illustrates Case 3 by a run on tree (c) in Figure 1 (copied as (d-0)). Trees (d-1)-(d-5) are the outcomes of statements 14-17 and 21, respectively, under the assumption that  $x - 2 = \text{root}$ . Here (d-3) and (d-4) are the same because  $T[x - 1].\text{dir} = T[x - 2].\text{dir}$  and hence statement 17 has no effect.

## 5 Conclusion

We presented a decidable theory of red-black trees, which is an extension of the theory of term algebras with two size functions. We showed how the red-black tree insertion algorithm can be analyzed using this theory. We plan to extend this theory to express local updates at an arbitrary pointed location in a tree. We note that adding a standard update function easily makes the first-order theory undecidable. We will investigate ways to enhance the expressiveness of the theory while maintaining the decidability.

## References

1. Rolf Backofen. A complete axiomatization of a theory with feature and arity constraints. *Journal of Logical Programming*, 24(1&2):37–71, 1995.
2. Paolo Baldan, Andrea Corradini, Javier Esparza, Tobias Heindel, Barbara König, and Vitali Kozioura. Verifying red-black trees. In *Proceedings of the 1st International Workshop on the Verification of Concurrent Systems with Dynamic Allocated Heaps (COSMICAH 2005)*, 2005.
3. Cristiano Calcagno, Philippa Gardner, and Uri Zarfaty. Context logic and tree update. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 271–282. ACM Press, 2005.
4. Hubert Comon, Max Dauchet, Remi Gilleron, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. Electronic edition at <http://13ux02.univ-lille3.fr/tata/tata.pdf>, 2002.
5. Hubert Comon and Catherine Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.
6. D. C. Cooper. Theorem proving in arithmetic without multiplication. In *Machine Intelligence*, volume 7, pages 91–99. American Elsevier, 1972.
7. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 2001.

8. J. Downey, R. Sethi, and R. E. Tarjan. Variations of the common subexpression problem. *Journal of the ACM*, 27:758–771, 1980.
9. H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2001.
10. Peter Habermehl, Radu Iosif, and Tomas Vojnar. Automata-based verification of programs with tree updates. In *Proceedings of 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*, pages 350–364. Springer-Verlag, 2006.
11. Konstantin Korovin and Andrei Voronkov. A decision procedure for the existential theory of term algebras with the Knuth-Bendix ordering. In *Proceedings of 15th IEEE Symposium on Logic in Computer Science*, pages 291 – 302. IEEE Computer Society Press, 2000.
12. Konstantin Korovin and Andrei Voronkov. Knuth-Bendix constraint solving is NP-complete. In *Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 979–992. Springer-Verlag, 2001.
13. Viktor Kuncak and Martin Rinard. The structural subtyping of non-recursive types is decidable. In *Proceedings of 18th IEEE Symposium on Logic in Computer Science*, pages 96–107. IEEE Computer Society Press, 2003.
14. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite tree. In *Proceedings of the 3rd IEEE Symposium on Logic in Computer Science*, pages 348–357. IEEE Computer Society Press, 1988.
15. A. I. Mal'cev. Axiomatizable classes of locally free algebras of various types. In *The Meta-mathematics of Algebraic Systems, Collected Papers*, chapter 23, pages 262–281. North Holland, 1971.
16. Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, April 1980.
17. Derek C. Oppen. Reasoning about recursively defined data structures. *Journal of the ACM*, 27(3):403–411, July 1980.
18. C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 320–325. ACM Press, 1978.
19. Radu Rugina. Quantitative shape analysis. In *Proceedings of the 11th International Static Analysis Symposium (SAS'04)*, volume 3148 of *Lecture Notes in Computer Science*, pages 228–245. Springer-Verlag, 2004.
20. Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. *ACM Transactions on Computational Logic*, 2(2):155–181, 2001.
21. Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In *the 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *Lecture Notes in Computer Science*, pages 152–167. Springer-Verlag, 2004.
22. Ting Zhang, Henny B. Sipma, and Zohar Manna. The decidability of the first-order theory of term algebras with Knuth-Bendix order. In Robert Nieuwenhuis, editor, *the 20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Computer Science*, pages 131–148. Springer-Verlag, 2005.
23. Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for term algebras with integer constraints. *Information and Computation*, 204:1526–1574, October 2006.