IST AUSTRIA

Institute of Science and Technology

On an Efficient Decision Procedure for Imperative Tree Data Structures

Thomas Wies , Marco Muñiz , and Viktor Kuncak

IST Austria (Institute of Science and Technology Austria)

Am Campus 1

A-3400 Klosterneuburg

Technical Report No. IST-2011-0005

http://pub.ist.ac.at/Pubs/TechRpts/2011/IST-2011-0005.pdf

April 26, 2011

On an Efficient Decision Procedure for Imperative Tree Data Structures

IST-2011-0005
EPFL-REPORT-165193

Thomas Wies[1], Marco Muñiz[2], and Viktor Kuncak[3]

[1] Institute of Science and Technology (IST), Austria
wies@ist.ac.at
[2] University of Freiburg, Germany
muniz@informatik.uni-freiburg.de
[3] EPFL, Switzerland
viktor.kuncak@epfl.ch

Abstract. We present a new decidable logic called TREX for expressing constraints about imperative tree data structures. In particular, TREX supports a transitive closure operator that can express reachability constraints, which often appear in data structure invariants. We show that our logic is closed under weakest precondition computation, which enables its use for automated software verification. We further show that satisfiability of formulas in TREX is decidable in NP. The low complexity makes it an attractive alternative to more expensive logics such as monadic second-order logic (MSOL) over trees, which have been traditionally used for reasoning about tree data structures.

1 Introduction

This paper introduces a new decision procedure for reasoning about imperative manipulations of tree data structures. Our logic of trees with reachability expressions (TREX) supports reasoning about reachability in trees and a form of quantification, which enables its use for expressing invariants of tree data structures, including the tree property itself. Despite the expressive power of the logic, we exhibit a non-deterministic polynomial-time decision procedure for its satisfiability problem, showing that TREX is NP-complete. Our development is directly motivated by our experience with verifying tree data structures in the Jahob verification system [15, 19, 21] in which we used the MONA decision procedure [11] for MSOL over trees. Although MONA contributed great expressive power to our specification language and, in our experience, works well for programs that manipulate lists, there were many tree-manipulating programs whose verification failed due to MONA running out of resources.

It was therefore a natural goal to identify a logic that suits our needs, but can be decided much more efficiently. There are other expressive logics supporting reachability but with lower complexity than MSOL [4, 7, 10, 20]. However, we did not find them suitable as a MONA alternative, for several reasons. First, we faced difficulties in the expressive power: some of the logics can only reason about sets but not individual objects, others have tree model property and thus cannot detect violations of the tree invariants. Moreover, the complexity of these logics is still at least EXPTIME, and their decision procedures are given in terms of automata-theoretic techniques or tableaux procedures, which can be difficult to combine efficiently with existing SMT solvers. Similarly, the logic of reachable patterns [20] is decidable through a highly non-trivial construction, but the complexity is at least NEXPTIME, as is the complexity of the Bernays-Schönfinkel Class with Datalog [5]. The logic [2] can express nested list structures of bounded nesting along with constraints on data fields and numerical constraints on paths, but cannot express constraints on arbitrary trees. On the other hand, TREX does not support reasoning on data fields; although such an extension is in principle possible. Other approaches generate induction scheme instances to prove transitive closure properties in general graphs [14]. While this strategy can succeed for certain examples, it does not have completeness or complexity guarantees, and suffers from the difficulties of first-order provers in handling transitive relations. Tree automata with size constraints can express properties such as the red-black tree invariant [8]. However, this work does not state the complexity of the reasoning task and the presented automata constructions appear to require running time beyond NP. Regular tree model checking with abstraction has yielded excellent results so far [3] and continues to improve, but has so far not resulted in a logic whose complexity is in NP, which we believe to be an important milestone.

The primary inspiration for our solution came from the efficient SMT-based techniques for reasoning about list structures [13], as well as the idea of viewing single-parent heaps as duals of lists [1]. However, there are several differences from this immediate inspiration. For integration with other decision procedures, as well as for modular reasoning with preconditions and postconditions, it was essential to obtain a logic and not only a finite-model property for the analysis of systems as in [1]. Furthermore, the need to support imperative updates on trees led to technical challenges that are very different than those of [13]. To address these challenges, we introduced a reachability predicate that is parameterized by a carefully chosen class of formulas to control the reachability relation. We show that the resulting logic of trees is closed under weakest preconditions with respect to imperative heap updates, which makes it suitable for expressing verification conditions in imperative programs. We devised a four-step decision procedure that contains formula transformations and ultimately reduces to a Ψ-local theory extension [9, 16]. Consequently, our logic can be encoded using a quantifier instantiation recipe within an SMT solver. We have encoded the axiomatization of TREX in Jahob and used Z3 [6] with a default instantiation strategy to verify tree and list manipulating programs. We have obtained verification times of around 1s, reducing the running times by two orders of magnitude compared to MONA.

2 Motivating Example

We next show how to use our decision procedure to verify functional correctness of a Java method that manipulates a binary tree data structure.

```
class Node {Node l, r, p;}
class Tree {
    private static Node root;
    invariant "ptree p [l,r]"; invariant "p root = null";
    private static specvar content :: objset;
    vardefs "content=={x. root ≠ null ∧ (x,root) ∈ {(x,y). p x = y}*}";
    public void insertLeftOf(Node pos, Node e)
      requires "pos ∈ content ∧ pos ≠ null ∧ l pos = null ∧
                e ∉ content ∧ e ≠ null ∧ p e = null ∧ l e = null ∧ r e = null"
      modifies content,l,p
      ensures "content = old content ∪ {e}"
    {
        e.p = pos; pos.l = e;
    } }
```

Fig. 1. Fragment of insertion into a tree

Example: insertion into a binary search tree. Fig. 1 shows a fragment of Java code for insertion into a binary search tree, factored out into a separate insertLeftOf method. In addition to Java statements, the example in Fig. 1 contains preconditions and postconditions, written in the notation of the Jahob verification system [12, 15, 18, 19, 21].

The search tree has fields (l, r) that form a binary tree, and field p, which for each node in the tree points to its parent (or null, if the node is the root of the tree). This property is expressed by the first class invariant using the special predicate ptree, which takes the parent field and a list of successor fields of the tree structure as arguments. The second invariant expresses that the field root points to the root node of the tree. The **vardefs** notation introduces the set content denoting the useful content of the tree. Note that if we are given a program that manipulates a tree data structure without explicit parent field then we can always introduce one as a specification variable that is solely used for the purpose of verification. This is possible because the parent field in a tree is uniquely determined by the successor fields.

The insertLeftOf method is meant to be invoked when the insertion procedure has traversed the tree and found a node pos that has no left child, as illustrated in Figure 2. The node e then becomes the new left child of pos. Our system checks that after each execution of the method insertLeftOf the specified class invariants still hold and that its postcondition is satisfied. The postcondition states that the node e has been properly inserted into the tree.

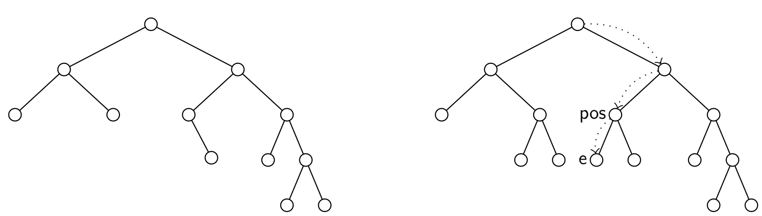

Fig. 2. State of insertion method before and after call to insertLeftOf

$$
\begin{aligned}
& \forall x.\, \langle \mathsf{p} \rangle^*(x, \mathsf{null}) \\
\wedge\ & \forall x.\, \mathsf{p}(\mathsf{l}(x)) = x \vee \mathsf{l}(x) = \mathsf{null} \\
\wedge\ & \forall x.\, \mathsf{p}(\mathsf{r}(x)) = x \vee \mathsf{r}(x) = \mathsf{null} \\
\wedge\ & \forall x.\, \mathsf{l}(\mathsf{p}(x)) = x \vee \mathsf{r}(\mathsf{p}(x)) = x \vee \mathsf{p}(x) = \mathsf{null} \\
\wedge\ & \forall xy.\, \mathsf{l}(x) = y \wedge \mathsf{r}(x) = y \ \rightarrow\ y = \mathsf{null} \\
\wedge\ & \forall xy.\, \mathsf{p}(x) = \mathsf{null} \wedge \mathsf{l}(y) = x \rightarrow x = \mathsf{null} \\
\wedge\ & \forall xy.\, \mathsf{p}(x) = \mathsf{null} \wedge \mathsf{r}(y) = x \rightarrow x = \mathsf{null}
\end{aligned}
$$

Fig. 3. Defining formula of ptree p [l, r]

$$
\begin{aligned}
& \mathsf{p}(\mathsf{root}) = \mathsf{null} \wedge \mathsf{root} \neq \mathsf{null} \wedge \langle \mathsf{p} \rangle^*(\mathsf{pos}, \mathsf{root}) \wedge \neg \langle \mathsf{p} \rangle^*(\mathsf{e}, \mathsf{root}) \wedge \\
& \mathsf{e} \neq \mathsf{null} \wedge \mathsf{p}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{l}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{r}(\mathsf{e}) = \mathsf{null} \\
& \rightarrow\ (\forall z. \langle \mathsf{upd}(\mathsf{p}, \mathsf{e}, \mathsf{pos}) \rangle^*(z, \mathsf{null}))
\end{aligned}
$$

Fig. 4. Verification condition expressing that, after execution of method insertLeftOf, the heap graph projected to field p is still acyclic

The full verification condition of method insertLeftOf can be expressed in our logic. Figure 4 shows one of the subgoals of this verification condition. It expresses that after execution of method insertLeftOf the heap graph projected to field p is still acyclic. This is a proof subgoal for checking that the ptree invariant is preserved by method insertLeftOf. Note that our logic supports field update expressions upd(p, e, pos) so that we can express the verification condition directly in the logic. Note further that the precondition stating that the ptree invariant holds at entry to the method is not explicitly part of the verification condition, i.e., it does not appear on the left hand side of the implication. This is because in the semantics of TREX we only consider models that satisfy the ptree invariant. Nevertheless, TREX can still be used to prove preservation of the ptree invariant because this invariant is expressible in the logic, as shown in Figure 3, and the logic is closed under computation of weakest preconditions for heap manipulating statements.

Our logic also supports reasoning about forward reachability $\langle \mathsf{l}, \mathsf{r} \rangle^*$ in the trees (i.e., transitive closure of the successor fields rather than the parent field) and quantification over sets of reachable objects. The latter is used, e.g., to prove the postcondition of method insertLeftOf stating that the node e was properly inserted and that no elements have been removed from the tree.

While we only consider a logic of *binary* trees in this paper; the generalization to trees of arbitrary finite arity is straightforward. In particular, an acyclic doubly-linked list is a special case of a tree with parent pointers, so reasoning about such structures is also supported by our decision procedure.

3 Decision Procedure Through an Example

We consider the negation of the verification condition shown in Figure 4, which is unsatisfiable in tree structures. Our decision procedure is described in Section 6 and proceeds in four steps.

$$\begin{array}{l}\mathsf{p}(\mathsf{root}) = \mathsf{null} \wedge \mathsf{root} \neq \mathsf{null} \wedge \langle \mathsf{p} \rangle^*(\mathsf{pos}, \mathsf{root}) \wedge \neg \langle \mathsf{p} \rangle^*(\mathsf{e}, \mathsf{root}) \wedge \\ \mathsf{e} \neq \mathsf{null} \wedge \mathsf{p}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{l}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{r}(\mathsf{e}) = \mathsf{null} \wedge \\ \neg(\forall z. \langle \mathsf{p} \rangle^*_{(x \neq \mathsf{e})}(z, \mathsf{null}) \vee \langle \mathsf{p} \rangle^*(z, \mathsf{e}) \wedge \langle \mathsf{p} \rangle^*_{(x \neq \mathsf{e})}(\mathsf{pos}, \mathsf{null}))\end{array}$$

Fig. 5. Negated verification condition from Fig. 4 after function update elimination

$$\begin{array}{l}\mathsf{p}(\mathsf{root}) = \mathsf{null} \wedge \mathsf{root} \neq \mathsf{null} \wedge P(\mathsf{pos}, \mathsf{root}) \wedge \neg P(\mathsf{e}, \mathsf{root}) \wedge \\ \mathsf{e} \neq \mathsf{null} \wedge \mathsf{p}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{l}(\mathsf{e}) = \mathsf{null} \wedge \mathsf{r}(\mathsf{e}) = \mathsf{null} \wedge \\ \neg(\forall z. P(z, \mathsf{null}) \wedge P(\mathsf{null}, bp_{(x \neq \mathsf{e})}(z)) \vee P(z, \mathsf{e}) \wedge P(\mathsf{pos}, \mathsf{null}) \wedge P(\mathsf{null}, bp_{(x \neq \mathsf{e})}(\mathsf{pos}))) \wedge \\ (\forall z.\, P(z, \mathsf{null})) \wedge (\forall z.\, P(z, z)) \wedge (\forall wz.\, P(w, z) \wedge P(z, w) \ \rightarrow \ z = w) \wedge \\ (\forall vwz.\, P(v, w) \wedge P(v, z) \ \rightarrow \ P(w, z) \vee P(z, w)) \wedge \\ (\forall wz.\, P(w, z) \ \rightarrow \ w = z \vee P(\mathsf{p}(w), z)) \wedge \\ (\forall z.\, P(z, bp_{(z \neq \mathsf{e})}(z))) \ \wedge (\forall z.\, bp_{(x \neq \mathsf{e})}(z) \neq e \rightarrow bp_{(x \neq \mathsf{e})}(z) = \mathsf{null}) \wedge \\ (\forall wz.\, P(w, z) \wedge P(z, bp_{(x \neq \mathsf{e})}(w)) \rightarrow z \neq \mathsf{e} \vee z = bp_{(x \neq \mathsf{e})}(w)) \wedge \ldots\end{array}$$

Fig. 6. Negated verification condition from Figure 4 after the reduction step to first-order logic. Only the axioms that are necessary for proving unsatisfiability of the formula are shown.

The first step (described in Section 6.1) is to eliminate all function update expressions in the formula. The result of this step is shown in Figure 5. Our logic supports so called *constrained reachability expressions* of the form $\langle \mathsf{p} \rangle^*_Q$ where Q is a binary predicate over dedicated variables x, y. The semantics of this predicate is that $\langle \mathsf{p} \rangle^*_Q(u, v)$ holds iff there exists a p-path connecting u and v and between every consecutive nodes w_1, w_2 on this path, $Q(w_1, w_2)$ holds. Using these constrained reachability expressions we can reduce reachability expressions over updated fields to reachability expressions over the non-updated fields, as shown in the example. This elimination even works for updates of successor functions below forward reachability expressions of the form $\langle \mathsf{l}, \mathsf{r} \rangle^*$.

The second step (described in Section 6.2) eliminates all forward reachability constraints over fields l, r from the formula and expresses them in terms of the relation $\langle \mathsf{p} \rangle^*$. Since there are no such constraints in our formula, we immediately proceed to Step 3.

The third step (described in Section 6.3) reduces the formula to a formula in first-order logic, whose finite models are exactly the models of the formula from the previous step, which is still expressed in TREX. For the purpose of the reduction, all occurrences of the reachability relation $\langle \mathsf{p} \rangle^*$ are replaced by a binary predicate symbol P, which is then axiomatized using universally quantified first-order axioms so that $\langle \mathsf{p} \rangle^*$ and P coincide in all finite models. All remaining reachability constraints are of the form $\langle \mathsf{p} \rangle^*_Q$. We can express these constraints in terms of P by introducing a unary function bp_Q (called *break point function*) that maps each node u to the first p-reachable node v of u for which $Q(v, \mathsf{p}(v))$ does not hold, i.e., $bp_Q(u)$ marks the end of the segment of nodes w that satisfy $\langle \mathsf{p} \rangle^*_Q(u, w)$. The function bp_Q can be axiomatized in terms of P and Q. Figure 6 shows the resulting formula (including only the necessary axioms for proving unsatisfiability of the formula).

The fourth step (described in Section 6.4) computes prenex normal form and skolemizes remaining top-level existential quantifiers. Then we add additional axioms that ensure Ψ-locality of the universally quantified axioms in the formula obtained from Step 3. The key property of the resulting formula is that its universal quantifiers can be instantiated finitely many times with terms syntactically derived from the terms within the formula. The result is an equisatisfiable quantifier-free formula, which can be handled by the SMT solver's congruence closure and the SAT solver.

4 Preliminaries

In the following, we define the syntax and semantics of formulas. We further recall the notions of partial structures and Ψ-local theories as defined in [9].

Sorted logic. We present our problem in sorted logic with equality. A *signature* Σ is a tuple (S, Ω), where S is a countable set of sorts and Ω is a countable set of function symbols f with associated arity $n \geq 0$ and associated sort $s_1 \times \cdots \times s_n \to s_0$ with $s_i \in S$ for all $i \leq n$. Function symbols of arity 0 are called *constant symbols*. In this paper we will only consider signatures with sorts $S = \{\mathsf{bool}, \mathsf{node}\}$ and the dedicated equality symbol $= \in \Omega$ of sort $\mathsf{node} \times \mathsf{node} \to \mathsf{bool}$. Note that we generally treat predicate symbols of sort $s_1, \ldots, s_n$ as function symbols of sort $s_1 \times \ldots \times s_n \to \mathsf{bool}$. Terms are built as usual from the function symbols in Ω and (sorted) variables taken from a countably infinite set X that is disjoint from Ω. A term t is said to be *ground*, if no variable appears in t. We denote by $\mathsf{Terms}(\Sigma)$ the set of all ground Σ-terms.

A Σ-*atom* A is a Σ-term of sort bool. We use infix notation for atoms built using the equality predicate $=$. A Σ-*formula* F is defined using structural recursion as either one of A, $\neg F_1$, $F_1 \wedge F_2$, or $\forall x : s.F_1$, where A is a Σ-atom, F_1 and F_2 are Σ-formulas, and $x \in X$ is a variable of sort $s \in S$. In the formulas appearing in this paper we will only ever quantify over variables of sort node, so we typically drop the sort annotation. We use syntactic sugar for Boolean constants ($\top$, $\bot$), disjunctions ($F_1 \vee F_2$), implications ($F_1 \to F_2$), and existential quantification ($\exists x.F_1$). For a finite index set $\mathcal{I}$ and Σ-formulas F_i, for all $i \in \mathcal{I}$, we write $\bigwedge_{i \in \mathcal{I}} F_i$ for the conjunction of the F_i (respectively, $\top$ if $\mathcal{I}$ is empty) and similarly $\bigvee_{i \in \mathcal{I}} F_i$ for their disjunction. We further write $F[x_1 := t_1, \ldots, x_n := t_n]$ for the simultaneous substitutions of the free variables x_i appearing in F by the terms t_i. We define literals and clauses as usual. A clause C is called *flat* if no term that occurs in C below a predicate symbol or the symbol $=$ contains nested function symbols. A clause C is called *linear* if (i) whenever a variable occurs in two non-variable terms in C that do not start with a predicate or the equality symbol, the two terms are identical, and if (ii) no such term contains two occurrences of the same variable.

Total and partial structures. Given a signature $\Sigma = (S, \Omega)$, a *partial* Σ-*structure* α is a function that maps each sort $s \in S$ to a non-empty set $\alpha(s)$ and each function symbol $f \in \Omega$ of sort $s_1 \times \cdots \times s_n \to s_0$ to a partial function $\alpha(f) : \alpha(s_1) \times \cdots \times \alpha(s_n) \rightharpoonup \alpha(s_0)$. If α is understood, we write just t instead of $\alpha(t)$ whenever this is not ambiguous. We assume that all partial structures interpret the sort bool by the two-element set of Booleans $\{0, 1\}$. We therefore call $\alpha(\mathsf{node})$ the *universe* of α and often identify $\alpha(\mathsf{node})$ and α. We further assume that all structures α interpret the symbol

$=$ by the equality relation on $\alpha(\mathsf{node})$. A partial structure α is called *total structure* or simply *structure* if it interprets all function symbols by total functions. For a Σ-structure α where Σ extends a signature Σ_0 with additional sorts and function symbols, we write $\alpha|_{\Sigma_0}$ for the Σ_0-structure obtained by restricting α to Σ_0.

Given a total structure α and a *variable assignment* $\beta : X \to \alpha(S)$, the evaluation $[\![t]\!]_{\alpha,\beta}$ of a term t in α, β is defined as usual. For a ground term t we typically write just $[\![t]\!]_{\alpha}$. A quantified variable of sort s ranges over all elements of $\alpha(s)$. From the interpretation of terms the notions of satisfiability, validity, and entailment of atoms, formulas, clauses, and sets of clauses in total structures are then derived as usual. In particular, we use the standard interpretations for propositional connectives of classical logic. We write $\alpha, \beta \models F$ if α satisfies F under β where F is a formula, a clause, or a set of clauses. Similarly, we write $\alpha \models F$ if F is valid in α. In this case we also call α a *model* of F. The interpretation $[\![t]\!]_{\alpha,\beta}$ of a term t in a partial structure α is as for total structures, except that if $t = f(t_1, \dots, t_n)$ for $f \in \Omega$ then $[\![t]\!]_{\alpha,\beta}$ is undefined if either $[\![t_i]\!]_{\alpha,\beta}$ is undefined for some i, or $([\![t_1]\!]_{\alpha,\beta}, \dots, [\![t_n]\!]_{\alpha,\beta})$ is not in the domain of $\alpha(f)$. We say that a partial structure α *weakly satisfies* a literal L under β, written $\alpha, \beta \models_w L$, if (i) L is an atom A and either $[\![A]\!]_{\alpha,\beta} = 1$ or $[\![A]\!]_{\alpha,\beta}$ is undefined, or (ii) L is a negated atom $\neg A$ and either $[\![A]\!]_{\alpha,\beta} = 0$ or $[\![A]\!]_{\alpha,\beta}$ is undefined. The notion of weak satisfiability is extended to clauses and sets of clauses as for total structures. A clause C (respectively, a set of clauses) is *weakly valid* in a partial structure α if α weakly satisfies α for all variable assignments β. In this case we call α a *weak partial model* of C.

Ψ-local theories. Note that the following definition is a particular special case of the more general notion of Ψ-local theory extensions. For the general definitions of local theory extensions, respectively, Ψ-local theory extensions, we direct the reader to [9, 16].

Let $\Sigma = (S, \Omega)$ be a signature. A *theory* $\mathcal{T}$ for a signature Σ is simply a set of Σ-formulas. We consider theories $\mathcal{T}(\mathcal{K})$ defined as a set of Σ-formulas that are consequences of a given set of clauses $\mathcal{K}$. We call $\mathcal{K}$ the *axioms* of the theory $\mathcal{T}(\mathcal{K})$ and we often identify $\mathcal{K}$ and $\mathcal{T}(\mathcal{K})$. In the following, when we refer to a set of ground clauses G, we assume they are over the signature $\Sigma^c = (S, \Omega \cup \Omega_c)$ where Ω_c is a set of new constant symbols. For a set of clauses $\mathcal{K}$, we denote by $\mathsf{st}(\mathcal{K})$ the set of all ground subterms that appear in $\mathcal{K}$. Let Ψ be a function associating with a set of (universally quantified) clauses $\mathcal{K}$ and a set of ground terms T a set $\Psi(\mathcal{K}, T)$ of ground terms such that (i) all ground subterms in $\mathcal{K}$ and T are in $\Psi(\mathcal{K}, T)$; (ii) for all sets of ground terms T, T' if $T \subseteq T'$ then $\Psi(\mathcal{K}, T) \subseteq \Psi(\mathcal{K}, T')$; (iii) Ψ is a closure operation, i.e., for all sets of ground terms T, $\Psi(\mathcal{K}, \Psi(\mathcal{K}, T)) \subseteq \Psi(\mathcal{K}, T)$. (iv) Ψ is compatible with any map h between constants, i.e., for any map $h \in \Omega_c \to \Omega_c$, $\Psi(\mathcal{K}, \overline{h}(T)) = \overline{h}(\Psi(\mathcal{K}, T))$ where $\overline{h}$ is the unique extension of h to terms. Let $\mathcal{K}[\Psi(\mathcal{K}, G)]$ be the set of instances of $\mathcal{K}$ in which all terms are in $\Psi(\mathcal{K}, \mathsf{st}(G))$, which here will be denoted by $\Psi(\mathcal{K}, G)$. We say that $\mathcal{K}$ is Ψ-local if it satisfies condition (Loc^{Ψ}):

(Loc^{Ψ}) For every finite set of ground clauses G, $\mathcal{K} \cup G \models \bot$ iff $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ has no weak partial model in which all terms in $\Psi(\mathcal{K}, G)$ are defined.

5 TREX: Logic of Trees with Reachability Expressions

We now formally define the formulas of our logic of trees with reachability expressions (TREX), whose satisfiability we study. For simplifying the exposition in the remainder of this paper, we restrict ourselves to binary trees. The decidability and complexity result carries over to trees of arbitrary finite arity in a straightforward manner.

5.1 Syntax of TREX Formulas

Figure 7 defines the TREX formulas. A TREX formula is an arbitrary propositional combination of atomic formulas. An atomic formula is either an equality between terms, a reachability expression, or a restricted quantified formula. A term t is either a constant $c \in \mathit{\Gamma}$ or a function term f applied to a term t. The set of constants $\mathit{\Gamma}$ is an arbitrary countably infinite set of symbols disjoint from all other symbols used in the syntax of formulas. However, we assume that $\mathit{\Gamma}$ contains the special constant symbol null. A function term is either one of the function symbols l, r (standing for the two successor functions of a tree), and p (standing for the parent function of a tree), or an update $\mathsf{upd}(f, t_1, t_2)$ of a function term f. In the latter case we call t_1 the *index* of the update and t_2 the *target*. A forward reachability expression relates two terms by a relation $\langle f_\mathsf{l}, f_\mathsf{r}\rangle^*_Q$ where f_l and f_r are the possibly updated successor functions and Q is a predicate built from boolean combinations of equalities between constants and the dedicated variables x and y. The syntactic restrictions on Q ensure that if one computes the disjunctive normal form of Q then the resulting formula will contain a disjunct, which is a conjunction of disequalities between constants and variables. A backward reachability expression is similar but refers to the possibly updated parent function instead of the successor functions. We call the relations $\langle f_\mathsf{l}, f_\mathsf{r}\rangle^*_Q$ *descendant relations* and the relations $\langle f_\mathsf{p}\rangle^*_Q$ *ancestor relations*. Finally, the formulas below restricted quantified formulas are almost like TREX formulas, except that the quantified variable may only appear at particular positions below function symbols and only as arguments of constrained ancestor relations.

For ease of notation we use the same syntactic shorthands in TREX formulas as for first-order formulas. For a predicate Q over the variables x and y and terms t_1, t_2, we typically write $Q(t_1, t_2)$ for the formula $Q[x := t_1, y := t_2]$. Finally, we simply write p^* as a shorthand for the *unconstrained* ancestor relation $\langle \mathsf{p}\rangle^*_\top$.

5.2 Semantics of TREX Formulas

TREX formulas are interpreted over finite forests of finite binary trees. We formally define these forests as first-order structures $\alpha_\mathcal{F}$ over the signature $\mathit{\Sigma}_\mathcal{F}$ of constant symbols $\mathit{\Gamma}$ and the unary function symbols l, r and p. To this end define the set of *tree nodes* $\mathcal{N}$ as the set of strings consisting of the empty string ϵ and all strings over alphabet $\mathbb{N} \cup \{\mathrm{L}, \mathrm{R}\}$ that satisfy the regular expression $\mathbb{N} \cdot (\mathrm{L} \mid \mathrm{R})^*$, i.e., we enumerate the trees comprising a forest by attaching a natural number to the nodes in each tree. A *forest* $\alpha_\mathcal{F}$ is then a structure whose universe is a finite prefixed-closed subset of tree nodes. The interpretation of the special constant symbol $\mathsf{null} \in \mathit{\Gamma}$ and the function symbols l, r, and p are determined by the universe of $\alpha_\mathcal{F}$ as in Figure 8. The remaining constant

$$
\begin{aligned}
F &::= A \mid F \wedge F \mid \neg F \\
A &::= t = t \mid \langle f_\mathsf{l}, f_\mathsf{r}\rangle_Q^*(t,t) \mid \langle f_\mathsf{p}\rangle_Q^*(t,t) \mid F_\forall \\
t &::= c \mid f(t) \\
f &::= f_\mathsf{l} \mid f_\mathsf{r} \mid f_\mathsf{p} \\
f_\mathsf{l} &::= \mathsf{upd}(f_\mathsf{l}, t, t) \mid \mathsf{l} \\
f_\mathsf{r} &::= \mathsf{upd}(f_\mathsf{r}, t, t) \mid \mathsf{r} \\
f_\mathsf{p} &::= \mathsf{upd}(f_\mathsf{p}, t, t) \mid \mathsf{p} \\
Q &::= v = c \to R \mid Q \wedge Q \\
R &::= t_R = t_R \mid R \wedge R \mid \neg R \\
t_R &::= v \mid c \\
F_\forall &::= \forall z.\, G_\mathsf{in} \\
G_\mathsf{in} &::= f(z) = t \to G_\mathsf{in} \mid F_\mathsf{in} \\
F_\mathsf{in} &::= A_\mathsf{in} \mid F_\mathsf{in} \wedge F_\mathsf{in} \mid \neg F_\mathsf{in} \\
A_\mathsf{in} &::= t_\mathsf{in} = t_\mathsf{in} \mid \langle f_\mathsf{p}\rangle_Q^*(t_\mathsf{in}, t_\mathsf{in}) \\
t_\mathsf{in} &::= z \mid t
\end{aligned}
$$

terminals:
$c \in \Gamma$ - constant symbol
$\mathsf{l}, \mathsf{r}, \mathsf{p}$ - function symbols
$v \in \{x, y\}$ - dedicated variable
$z \in X$ - variable

Fig. 7. Logic of trees with reachability TREX

symbols in Γ may be interpreted by any tree node in $\alpha_\mathcal{F}$. Let $\mathcal{F}$ be the set of all forests and let $\mathcal{M}_\mathcal{F}$ be the set of all first-order structures over signature $\Sigma_\mathcal{F}$ that are isomorphic to some structure in $\mathcal{F}$. We extend the term *forest* to all the structures in $\mathcal{M}_\mathcal{F}$.

$$
\begin{aligned}
\alpha_\mathcal{F}(\mathsf{null}) &= \epsilon \\
\alpha_\mathcal{F}(\mathsf{l})(n) &= \begin{cases} n\mathrm{L} & \text{if } n\mathrm{L} \in \alpha_\mathcal{F} \\ \epsilon & \text{otherwise} \end{cases} \\
\alpha_\mathcal{F}(\mathsf{r})(n) &= \begin{cases} n\mathrm{R} & \text{if } n\mathrm{R} \in \alpha_\mathcal{F} \\ \epsilon & \text{otherwise} \end{cases} \\
\alpha_\mathcal{F}(\mathsf{p})(n) &= \begin{cases} n' & \text{if } n = n's \text{ for some } s \in \mathbb{N} \cup \{\mathrm{L}, \mathrm{R}\} \text{ and } n' \in \alpha_\mathcal{F} \\ \epsilon & \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 8. Semantics of functions and constants in the forest model.

For defining the semantics of TREX formulas, let $\alpha_\mathcal{F} \in \mathcal{M}_\mathcal{F}$. We only explain the interpretation of terms, function terms, and reachability expressions in detail, the remaining constructs are interpreted as expected. The notions of satisfiability, entailment, etc. for TREX formulas are defined as in Section 4.

The interpretation of terms and function terms in $\alpha_\mathcal{F}$ under a variable assignment β recursively extend the interpretation of $\Sigma_\mathcal{F}$-terms as follows:

$$
\begin{aligned}
[\![f]\!]_{\alpha_\mathcal{F},\beta} &\stackrel{\text{def}}{=} \alpha_\mathcal{F}(f), \text{ for } f \in \{\mathsf{l}, \mathsf{r}, \mathsf{p}\} \\
[\![\mathsf{upd}(f, t_1, t_2)]\!]_{\alpha_\mathcal{F},\beta} &\stackrel{\text{def}}{=} [\![f]\!]_{\alpha_\mathcal{F},\beta}[[\![t_1]\!]_{\alpha_\mathcal{F},\beta} \mapsto [\![t_2]\!]_{\alpha_\mathcal{F},\beta}] \\
[\![f(t)]\!]_{\alpha_\mathcal{F},\beta} &\stackrel{\text{def}}{=} [\![f]\!]_{\alpha_\mathcal{F},\beta}([\![t]\!]_{\alpha_\mathcal{F},\beta})
\end{aligned}
$$

In order to define the semantics of reachability expressions compactly, we write $\langle Fn \rangle_Q^*(t_1, t_2)$ for either a forward reachability expression $\langle f_\mathsf{l}, f_\mathsf{r} \rangle_Q^*(t_1, t_2)$ or a backward reachability expression $\langle f_\mathsf{p} \rangle_Q^*(t_1, t_2)$. In the first case, the meta variable Fn denotes the set of function terms $\{f_\mathsf{l}, f_\mathsf{r}\}$ and in the second case the set $\{f_\mathsf{p}\}$. We will later also use the notation $\langle f, Fn \rangle_Q^*(t_1, t_2)$ meaning $\langle f_\mathsf{p} \rangle_Q^*(t_1, t_2)$ if $f = f_\mathsf{p}$ and $Fn = \emptyset$, respectively, $\langle f_\mathsf{l}, f_\mathsf{r} \rangle_Q^*(t_1, t_2)$ if $Fn = \{f_\mathsf{r}\}$ and $f = f_\mathsf{l}$ or $Fn = \{f_\mathsf{l}\}$ and $f = f_\mathsf{r}$. A reachability expression $\langle Fn \rangle_Q^*(t_1, t_2)$ expresses that the node defined by t_2 can be obtained from the node defined by t_1, by successively applying the functions defined by the function terms in Fn, where at each step Q holds between the current node and its image. Formally, we define the binary predicate $R_{Q,Fn}$ by the formula $\left(\bigvee_{f \in Fn} f(x) = y\right) \wedge Q$ and interpret the reachability relation $\langle Fn \rangle_Q^*$ as the reflexive transitive closure of $R_{Q,Fn}$:

$$[\![\langle Fn \rangle_Q^*]\!]_{\alpha_\mathcal{F},\beta} \stackrel{\text{def}}{=} \left\{\, (u, v) \in \alpha_\mathcal{F} \times \alpha_\mathcal{F} \mid [\![R_{Q,Fn}]\!]_{\alpha_\mathcal{F},\beta[x \mapsto u, y \mapsto v]} \,\right\}^*$$

The interpretation of $\langle Fn \rangle_Q^*(t_1, t_2)$ is then defined as expected.

Definition 1 (Satisfiability Problem for TREX). *The satisfiability problem for TREX asks whether, given a TREX formula F, there exists a forest $\alpha_\mathcal{F}$ that satisfies F.*

6 Decision Procedure for TREX

The logic TREX is a proper subset of MSOL over finite trees. Thus, decidability of the satisfiability problem for TREX follows from the result described in [17]. In fact TREX formulas can be expressed in terms of MSOL formulas with at most two quantifier alternations, which gives a 2-EXPTIME upper-bound for the complexity. In the following, we show that the satisfiability problem for TREX is actually in NP.

For the remainder of this section we fix a TREX formula F_0. Our decision procedure proceeds in four steps. The first two steps eliminate function updates and forward reachability expressions from F_0, resulting in equisatisfiable TREX formulas F_1 and then F_2. In the third step the formula F_2 is reduced to a first-order formula F_3 that has the same finite models as the original formula F. We then use results on local theories [9, 16] to prove a small model property for the obtained formulas. This allows us to use an existing decision procedure to check satisfiability of F_3 in the final step of our algorithm and obtain NP completeness. Proofs of Lemmas stated in this section can be found in Appendix A.

6.1 Elimination of Function Updates

We first describe the elimination of function updates from the input formula F_0. The algorithm that achieves this is as follows:

1. Flatten the index and target terms of function updates in F_0 by exhaustively applying the following rewrite rule:
   $C[\mathsf{upd}(f, i, t)] \;\rightsquigarrow\; C[\mathsf{upd}(f, c_i, c_t)] \wedge c_i = i \wedge c_t = t$
   where i, t are non-constant terms and $c_i, c_t \in \Gamma$ are fresh constant symbols

2. Eliminate function updates in reachability expressions by exhaustively applying the following rewrite rule:
   $C[\langle \mathsf{upd}(f, c_i, c_t), Fn\rangle^*_Q(t_1, t_2)] \;\rightsquigarrow\; C[H] \wedge \bigwedge_{f' \in Fn} c_{f'} = f'(c_i)$
   where the $c_{f'}$ are fresh constant symbols and
   $H \stackrel{\text{def}}{=} \langle f, Fn\rangle^*_R(t_1, t_2) \;\vee\; \langle f, Fn\rangle^*_Q(t_1, c_i) \;\wedge\; \langle f, Fn\rangle^*_R(c_t, t_2) \;\wedge\; Q(c_i, c_t)$
   $R \stackrel{\text{def}}{=} Q \wedge (x = c_i \rightarrow \bigvee_{f' \in Fn} y = c_{f'})$
3. Eliminate all remaining function updates by exhaustively applying the following rewrite rule:
   $t_1 = C[\mathsf{upd}(f, c_i, c_t)(t_2)] \;\rightsquigarrow\; t_2 = c_i \wedge t_1 = C[c_t] \vee t_2 \neq c_i \wedge t_1 = C[f(t_2)]$

Note that the exhaustive application of the rule in each of the steps 1. to 3. is guaranteed to terminate. Thus, let F_1 be one of the possible normal forms obtained after exhaustive application of these rules to F_0.

Lemma 2. *F_1 is a TREX formula and is equisatisfiable with F_0.*

We briefly sketch the key arguments in the proof of Lemma 2. First, one can easily check that each application of the rewrite rules in steps 1. and 3. produces equisatisfiable formulas. We discuss the rule in Step 2. in more detail. Given a structure α, let G_α be the graph spanned by the functions that are defined by the interpretations of function terms f, Fn in α, respectively, G'_α the graph spanned by $\mathsf{upd}(f, c_i, c_t), Fn$. There are two cases to consider for $\langle \mathsf{upd}(f, c_i, c_t), Fn\rangle^*_Q(t_1, t_2)$ to be true in α. The first case is that there already exists a Q-path in G_α between nodes t_1 and t_2, and if this path uses the edge $(c_i, f(c_i))$ (which might no longer exist in the updated function graph G'_α) then it is by means of one of the other functions defined by Fn, which are not affected by the update. The first disjunct in the formula H captures precisely this case.

The second case for $\langle \mathsf{upd}(f, c_i, c_t), Fn\rangle^*_Q(t_1, t_2)$ to be true is that the update of f creates a new Q-path between t_1 and t_2 in G'_α that is not present in G_α (respectively, such a path exists but the update does not change the function f). Consider the shortest of these paths between t_1 and t_2. Then we can split this path into three segments: (i) the segment from t_1 to c_i, (ii) the new edge (c_i, c_t), and (iii) the segment from c_t to t_2. Because we consider the shortest path, the segment (i) is not affected by the update so it must already we present in G_α. The segment (iii) can only exist in G'_α if it is already present in G_α and does not use the updated edge $(c_i, f(c_i))$. This is precisely captured by the second disjunct in formula H.

It is worth mentioning that Lemma 2 does not rely on the fact that we only consider forests for the interpretation of TREX formulas.

6.2 Elimination of Descendant Relations

We next describe the second step of our decision procedure, which eliminates all descendant relations from the formula F_1. The elimination is performed using the following rewrite rule:

$$\langle \mathsf{l}, \mathsf{r}\rangle^*_Q(s, t) \;\rightsquigarrow\; s = t \;\vee\; s \neq \mathsf{null} \wedge (\exists z.\, (\mathsf{l}(z) = t \vee \mathsf{r}(z) = t) \wedge \langle \mathsf{p}\rangle^*_{Q^{-1}}(z, s) \wedge Q(z, t))$$

where $Q^{-1} \stackrel{\text{def}}{=} Q[x := y, y := x]$. Let F_2 be one of the normal forms obtained by exhaustively applying this rewrite rule to F_1.

Lemma 3. *F_2 is a TREX formula and is equisatisfiable with F_1.*

6.3 Reduction to First-Order Logic

In the third step of our decision procedure we reduce the formula F_2 obtained after the second step to a formula F_3 in first-order logic.

The idea of the reduction is to provide a first-order axiomatization of the unconstrained ancestor relation p^* whose finite models are precisely the forests $\mathcal{M}_\mathcal{F}$ defined in Section 5.2. For this purpose we introduce a fresh binary predicate symbol P representing p^*. The axioms defining P are given in Figure 9. We can then axiomatize each constrained ancestor relation $\langle \mathsf{p} \rangle_Q^*$ in terms of p^*. To achieve this we exploit that, in forests, the relations $\langle \mathsf{p} \rangle_Q^*$ can be characterized as follows:

$$\forall xy.\, \langle \mathsf{p} \rangle_Q^*(x, y) \leftrightarrow \mathsf{p}^*(x, y) \wedge \mathsf{p}^*(y, bp_Q(x)) \tag{1}$$

where bp_Q is the function that maps a node x to the first ancestor z of x such that $Q(z, \mathsf{p}(z))$ does not hold (or null if such a node does not exist). We call bp_Q the *break point* function for $\langle \mathsf{p} \rangle_Q^*$. The intuition behind the above definition is that for $\langle \mathsf{p} \rangle_Q^*(x, y)$ to be true, the break point for the path of ancestor nodes of x must come after y has been reached (respectively, y itself is the break point of x). Note that this definition exploits the fact that forests are acyclic graphs. The axioms defining the functions bp_Q are given in Figure 10.

Formally, the reduction of F_2 to a first-order logic formula F_3 is defined as follows:

1. Let P be a fresh binary predicate symbol and let $F_{3,1}$ be the formula obtained by conjoining F_2 with the axioms shown in Figure 9.
2. Let $\mathcal{Q}$ be the set of predicates Q appearing in reachability expressions $\langle \mathsf{p} \rangle_Q^*(t_1, t_2)$ in F_2. For each $Q \in \mathcal{Q}$, let bp_Q be a fresh unary function symbol. For each $Q \in \mathcal{Q}$, replace all occurrences of the form $\langle \mathsf{p} \rangle_Q^*(t_1, t_2)$ in F_2 by $P(t_1, t_2) \wedge P(t_2, bp_Q(t_1))$. Let the result be $F_{3,2}$.
3. Finally, for each $Q \in \mathcal{Q}$, conjoin $F_{3,2}$ with the axioms shown in Figure 10. Let F_3 be the resulting formula.

Let Σ_P be the extension of the signature $\Sigma_\mathcal{F}$ with the symbols P, and bp_Q, for all $Q \in \mathcal{Q}$.

Lemma 4. *For every finite Σ_P-model α of the axioms in Figure 9, $\alpha(P) = \alpha(\mathsf{p})^*$ and $\alpha|_{\Sigma_\mathcal{F}} \in \mathcal{M}_\mathcal{F}$.*

Lemma 5. *The TREX formula F_2 has a model in $\mathcal{M}_\mathcal{F}$ iff the Σ_P-formula F_3 has a finite Σ_P-model.*

6.4 Ψ-Locality

Now let F_4 be the formula obtained by transforming F_3 into prenex normal from and skolemizing all existential quantifiers. Note that our syntactic restrictions on TREX formulas ensure that there are no alternating quantifiers appearing in the formulas F_0, F_1, F_2, and hence F_3. So skolemization only introduces additional Skolem constants, but no additional function symbols.

$$
\begin{array}{rl}
\textsf{l-Child:} & \mathsf{p}(\mathsf{l}(x)) = x \lor \mathsf{l}(x) = \mathsf{null} \\
\textsf{r-Child:} & \mathsf{p}(\mathsf{r}(x)) = x \lor \mathsf{r}(x) = \mathsf{null} \\
\textsf{Parent:} & \mathsf{l}(\mathsf{p}(x)) = x \lor \mathsf{r}(\mathsf{p}(x)) = x \lor \mathsf{p}(x) = \mathsf{null} \\
\textsf{lr-Distinct:} & \mathsf{l}(x) = y \land \mathsf{r}(x) = y \;\rightarrow\; y = \mathsf{null} \\
\textsf{l-Root:} & \mathsf{p}(x) = \mathsf{null} \land \mathsf{l}(z) = x \rightarrow x = \mathsf{null} \\
\textsf{r-Root:} & \mathsf{p}(x) = \mathsf{null} \land \mathsf{r}(z) = x \rightarrow x = \mathsf{null} \\
\textsf{p-Loop:} & \mathsf{p}(x) = x \rightarrow x = \mathsf{null} \\
\textsf{NullTerm:} & P(x, \mathsf{null}) \\
\textsf{Refl:} & P(x, x) \\
\textsf{Trans:} & P(x, y) \land P(y, z) \;\rightarrow\; P(x, z) \\
\textsf{AntiSym:} & P(x, y) \land P(y, x) \;\rightarrow\; x = y \\
\textsf{Total:} & P(x, y) \land P(x, z) \;\rightarrow\; P(z, y) \lor P(y, z) \\
\textsf{p-Step:} & P(x, \mathsf{p}(x)) \\
\textsf{p-Unfold:} & P(x, y) \;\rightarrow\; x = y \;\lor\; P(\mathsf{p}(x), y)
\end{array}
$$

Fig. 9. First-order axioms for the unconstrained ancestor relation p^* (represented by the binary predicate symbol P) and the functions l, r, and p in a forest

$$
\begin{array}{rl}
bp_Q\textsf{-Def1:} & P(x, bp_Q(x)) \\
bp_Q\textsf{-Def2:} & Q(bp_Q(x), \mathsf{p}(bp_Q(x))) \rightarrow bp_Q(x) = \mathsf{null} \\
bp_Q\textsf{-Def3:} & P(x, y) \land P(y, bp_Q(x)) \rightarrow Q(y, p(y)) \lor y = bp_Q(x)
\end{array}
$$

Fig. 10. First-order axioms defining the break point function bp_Q used to express a constrained ancestor relation $\langle \mathsf{p} \rangle_Q^*$ in a forest

Let C be the set of clauses obtained by transforming F_4 into clausal normal form. Then partition C into sets of ground clauses G and non-ground clauses $\mathcal{K}_P$ in which all terms have been linearized and flattened. The idea is now to define a closure operator Ψ such that condition (Loc^{Ψ}) from Section 4 holds for the particular pair $\mathcal{K}_P, G$. We can then use the decision procedure described in [9, Section 3.1] to check satisfiability of $\mathcal{K}_P, G$ using finite instantiation of the quantified variables in $\mathcal{K}_P$. To ensure that we can extend finite weak partial models of $\mathcal{K}_P[\Psi(\mathcal{K}_P, G)] \cup G$ to finite total models of $\mathcal{K}_P \cup G$, we have to make sure that $\Psi(\mathcal{K}_P, G)$ contains sufficiently many ground terms.

Specifically, we have to make sure that we can always construct a finite total model that satisfies the axioms in Figure 9 defining the ancestor relation, the parent function, and the successor functions, respectively, the axioms in Figure 10 defining the break point functions. Also note that the break point functions may appear below universal quantifiers in F_4 that come from restricted quantified subformulas in the original formula F_0, respectively, universal quantifiers that have been introduced in the rewrite steps that eliminate all occurrences of descendant relations (as described in Section 6.2). To guarantee that weak partial models of $\mathcal{K}_P[\Psi(\mathcal{K}_P, G)] \cup G$ can be extended to total models of $\mathcal{K}_P \cup G$, we will therefore define Ψ such that in every finite weak partial model of $\mathcal{K}_P[\Psi(\mathcal{K}_P, G)] \cup G$, both P and the break point functions are already totally defined. However, for this we have to bound the possible values of the break point functions. In fact, each predicate $Q \in \mathcal{Q}$ bounds the possible values that bp_Q can take. Let $\Gamma(Q)$ be the set of constants appearing in Q and let α be a finite total model of $\mathcal{K}_P$, then for all $u \in \alpha$, $bp_Q(u)$ is one of null, c, $\mathsf{l}(c)$, or $\mathsf{r}(c)$ for some $c \in \Gamma(Q)$. Thus,

$$bp_Q\text{-Def4: } P(x,y) \wedge P(y, bp_Q(x)) \rightarrow bp_Q(x) = bp_Q(y)$$
$$bp_Q\text{-Def5: } \bigvee_{t \in BP(Q)} bp_Q(x) = t$$

Fig. 11. Additional first-order axioms for bounding the break point functions

for each predicate $Q \in \mathcal{Q}$ define the set of its potential break points $BP(Q)$ as follows. For sets of ground terms T and a k-ary function symbol f, let $f(T)$ be the set of all (properly sorted) ground terms $f(t_1, \ldots, t_k)$ for some $t_1, \ldots, t_k \in T$. Then define

$$BP(Q) \stackrel{\text{def}}{=} \Gamma(Q) \cup \mathsf{l}(\Gamma(Q)) \cup \mathsf{r}(\Gamma(Q)) \cup \{\mathsf{null}\}$$

Let further $BP(\mathcal{Q})$ be the union of all sets $BP(Q)$ for $Q \in \mathcal{Q}$. This leads us to our first approximation Ψ_{bp} of Ψ. To this end let $f^i(T)$ be the set $f(T)$ restricted to the terms in which the function symbol f appears at most i times, and let $bp^-(T)$ be the set of ground terms obtained by removing from each ground term in T all appearances of the function symbols $\{\, bp_Q \mid Q \in \mathcal{Q} \,\}$. Then define

$$\begin{aligned}
\Psi_0(T) &\stackrel{\text{def}}{=} T \cup \{\, \mathsf{p}(t) \mid t \in T, t = \mathsf{l}(t') \vee t = \mathsf{r}(t') \,\} \cup BP(\mathcal{Q}) \cup \mathsf{p}(BP(\mathcal{Q})) \\
\Psi_4(T) &\stackrel{\text{def}}{=} T \cup \textstyle\bigcup_{Q \in \mathcal{Q}} bp_Q(bp^-(T)) \\
\Psi_5(T) &\stackrel{\text{def}}{=} T \cup P(T) \\
\Psi_{bp}(\mathcal{K}, T) &\stackrel{\text{def}}{=} \Psi_5 \circ \Psi_4 \circ \Psi_0(\mathsf{st}(\mathcal{K}) \cup T)
\end{aligned}$$

Let further $\mathcal{K}_{bp}$ be the set of universally quantified clauses obtained from $\mathcal{K}_P$ by adding the linearized and flattened clauses corresponding to the axioms shown in Figure 11. These additional axioms ensure that the interpretation of the break point functions in weak partial models of $\mathcal{K}_P$ are consistent with those in total models of $\mathcal{K}_P$.

However, the above definition is not yet sufficient to ensure Ψ-locality. Assume that a clause of the form $z = c \vee z = d$ appears in $\mathcal{K}_{bp}$ that results from a restricted quantified formula $\forall z. z = c \vee z = d$ in F_0. Then this clause imposes an upper bound of 2 on the cardinality of the models of F_4. We thus have to make sure that for any weak partial model of $\mathcal{K}_{bp}[\Psi_{bp}(\mathcal{K}_{bp}, G)] \cup G$, we can find a total model of the same cardinality. Unfortunately, for $\mathcal{K}_{bp}$ and Ψ_{bp} this is not always possible. For instance, assume G is the formula

$$\begin{aligned}
&d \neq \mathsf{null} \wedge P(a,d) \wedge P(b,d) \wedge P(c,d) \wedge \\
&\neg P(a,b) \wedge \neg P(a,c) \wedge \neg P(b,a) \wedge \neg P(b,c) \wedge \neg P(c,a) \wedge \neg P(c,b)
\end{aligned}$$

Then $\mathcal{K}_{bp}[\Psi_{bp}(\mathcal{K}_{bp}, G)] \cup G$ has a weak partial Σ_P-model of cardinality 5, but all total Σ_P-models of $\mathcal{K}_{bp} \cup G$ have cardinality at least 6. We can ensure that total models of matching cardinality exist by enforcing that every weak partial model already determines the *first common ancestor* of every pair of nodes. We can axiomatize the first common ancestor of two nodes by introducing a fresh binary function symbol fca and then adding the axioms shown in Figure 12. Let Σ_{fca} be the signature Σ_P extended with the binary function symbol fca and let $\mathcal{K}_{fca}$ be the set of universally quantified clauses obtained by adding to $\mathcal{K}_{bp}$ the linearized and flattened clauses corresponding to the axioms in Figure 12.

fca-Def1: $P(x, \mathit{fca}(x, y))$
fca-Def2: $P(y, \mathit{fca}(x, y))$
fca-Def3: $P(x, z) \land P(y, z) \rightarrow P(\mathit{fca}(x, y), z)$
fca-Def4: $\mathit{fca}(x, y) = w \land \mathit{fca}(x, z) = w \land \mathit{fca}(y, z) = w \rightarrow x = y \lor x = z \lor y = z \lor w = \mathsf{null}$

Fig. 12. Axioms defining the first common ancestor of two nodes in a forest

Lemma 6. *Let α be a finite Σ_{fca}-model of $\mathcal{K}_{\mathit{fca}}$. Then α also satisfies the following formulas*

$$\forall xy.\, \mathit{fca}(x, y) = \mathit{fca}(y, x)$$
$$\forall xyzw.\, \mathit{fca}(\mathit{fca}(x, y), z) = w \rightarrow w = \mathit{fca}(x, y) \lor w = \mathit{fca}(y, z) \lor w = \mathit{fca}(x, z)$$

Lemma 6 bounds the values that the function fca can take in models of $\mathcal{K}_{\mathit{fca}}$. This leads us to our second attempt at defining Ψ, which is as follows:

$$\Psi_3(T) \stackrel{\text{def}}{=} T \cup \mathit{fca}^1(T) \cup \mathit{fca}^2(T \cup \mathit{fca}^1(T))$$
$$\Psi_{\mathit{fca}}(\mathcal{K}, T) \stackrel{\text{def}}{=} \Psi_5 \circ \Psi_4 \circ \Psi_3 \circ \Psi_0(\mathsf{st}(\mathcal{K}) \cup T)$$

Note that axiom fca-Def4 ensures that in weak partial models of $\mathcal{K}_{\mathit{fca}}[\Psi_{\mathit{fca}}(\mathcal{K}_{\mathit{fca}}, G)] \cup G$ all proper nodes u of trees have at most two immediate P-predecessors even if l and r are not defined on u. For instance, assume G is

$$a \neq b \land P(b, a) \land \neg P(b, \mathsf{l}(a)) \land \neg P(b, \mathsf{r}(a))$$

Then G has no Σ_P-model that is also a model of $\mathcal{K}_P$ and $\mathcal{K}_{\mathit{fca}}[\Psi_{\mathit{fca}}(\mathcal{K}_{\mathit{fca}}, G)]$ has no weak partial model in which all terms in $\Psi_{\mathit{fca}}(\mathcal{K}_{\mathit{fca}}, G)$ are defined. On the other hand, $\mathcal{K}_{bp}[\Psi_{bp}(\mathcal{K}_{bp}, G)]$ has such weak partial models.

Unfortunately, the operator Ψ_{fca} is still not good enough to ensure Ψ-locality. Assume that a universally quantified clause of the form

$$f(z) = t \rightarrow H \qquad (2)$$

appears in $\mathcal{K}_{\mathit{fca}}$ that resulted from a restricted quantified formula in F_0 of the form $\forall z.\, f(z) = t \rightarrow H$ and where f is either one of p, l, or r. Assume that $f = \mathsf{p}$. To ensure that (2) remains valid whenever we complete p to a total function in some weak partial model α of (2), we have to ensure that we never have to define $\mathsf{p}(u) = t$, for any $u \in \alpha$ for which p is undefined. Consider first the case that in said model t is not null, then we can guarantee that we never have to define $\mathsf{p}(u) = t$ by making sure that α is already defined on the ground terms $\mathsf{p}(\mathsf{l}(t))$ and $\mathsf{p}(\mathsf{r}(t))$. This suggests that we should add the following additional ground terms to the set of ground terms generated by $\Psi_0(T)$:

$$\begin{aligned} \Psi_1(T) \stackrel{\text{def}}{=} T \;&\cup\; \{\, \mathsf{l}(t), \mathsf{p}(\mathsf{l}(t)), \mathsf{r}(t), \mathsf{p}(\mathsf{r}(t)) \mid (\mathsf{p}, t) \in \mathit{Grd} \,\} \\ &\cup\; \{\, \mathsf{p}(t), \mathsf{l}(\mathsf{p}(t)), \mathsf{p}(\mathsf{l}(\mathsf{p}(t))) \mid (\mathsf{l}, t) \in \mathit{Grd} \,\} \\ &\cup\; \{\, \mathsf{p}(t), \mathsf{r}(\mathsf{p}(t)), \mathsf{p}(\mathsf{r}(\mathsf{p}(t))) \mid (\mathsf{r}, t) \in \mathit{Grd} \,\} \end{aligned}$$

where Grd is the set of all pairs (f, t) of function symbols and ground terms appearing in guards of clauses of the form (2) in $\mathcal{K}_{\mathit{fca}}$.

Root1: $P(x, y) \to P(y, root(x)) \vee y = \mathsf{null}$
Root2: $root(x) = \mathsf{null} \leftrightarrow x = \mathsf{null}$
l-Leaf1: $P(lleaf(x), x) \vee lleaf(x) = \mathsf{null}$
r-Leaf1: $P(rleaf(x), x) \vee rleaf(x) = \mathsf{null}$
l-Leaf2: $P(lleaf(x), \mathsf{l}(x))$
r-Leaf2: $P(rleaf(x), \mathsf{r}(x))$
l-Leaf3: $lleaf(lleaf(x)) = \mathsf{null}$
r-Leaf3: $rleaf(rleaf(x)) = \mathsf{null}$
l-Leaf4: $lleaf(rleaf(x)) = \mathsf{null}$
r-Leaf4: $rleaf(rleaf(x)) = \mathsf{null}$
Leaves1: $fca(lleaf(x), rleaf(x)) = x \vee lleaf(x) = \mathsf{null} \vee rleaf(x) = \mathsf{null}$
Leaves2: $(lleaf(x) = \mathsf{null} \vee rleaf(x) = \mathsf{null}) \wedge fca(y, z) = x \to x = y \vee x = z \vee x = \mathsf{null}$
Leaves3: $lleaf(x) = \mathsf{null} \wedge rleaf(x) = \mathsf{null} \wedge P(y, x) \to y = x \vee x = \mathsf{null}$

Fig. 13. Axioms for the auxiliary function symbols $root$, $lleaf$, and $rleaf$

If for some guard (f, t) the weak partial model α satisfies $t = \mathsf{null}$ then the situation is not quite so simple. In this case we have to make sure that α already explicitly determines which nodes $u \in \alpha$ satisfy $f(u) = \mathsf{null}$, even if f is not defined on u. However, there is no finite set of ground terms T over the signature Σ_{fca} such that instantiation of $\mathcal{K}_{fca}$ with the terms in T will enforce this property. To enable the construction of such a finite set of terms, we introduce auxiliary functions $root$, $lleaf$, and $rleaf$ that determine the root, a left child, and a right child of every node in a forest. More precisely, the semantics of these functions is as follows: for each $u \in \alpha$, $root(u)$ determines the root of the tree in α to which u belongs (i.e., in all total models α of $\mathcal{K}_{fca}$ and $u \in \alpha$, $\mathsf{p}(u) = \mathsf{null}$ if and only if $root(u) = u$). Similarly, $lleaf(u)$ is some leaf of the tree to which u belongs such that $lleaf(u)$ is a left descendant of u, or null if u has no left descendant (i.e., in all total models α of $\mathcal{K}_{fca}$ and $u \in \alpha$, $\mathsf{l}(u) = \mathsf{null}$ holds if and only if $lleaf(u) = \mathsf{null}$). The semantics of $rleaf$ is analogous to $lleaf$. Let Σ be the signature Σ_{fca} extended with fresh unary function symbols $root$, $lleaf$, and $rleaf$. The axioms capturing the above semantics of the auxiliary functions are given in Figure 13. We can then replace every clause of the form (2) in $\mathcal{K}_{fca}$ by the following two clauses:

$$f(z) = t \to t = \mathsf{null} \vee H$$
$$t = \mathsf{null} \wedge N_f(z) \to H$$

where $N_f(z)$ is $root(z) = z$ if f is p, $lleaf(z) = \mathsf{null}$ if f is l, and $rleaf(z) = \mathsf{null}$ if f is r. Let $\mathcal{K}$ be the resulting set of clauses extended with the linearized and flattened clauses obtained from the axioms in Figure 13.

Lemma 7. *The formula F_3 has a finite Σ_P-model if and only if $\mathcal{K} \cup G$ has a finite Σ-model.*

Lemma 8. *Let α be a finite Σ-model of $\mathcal{K}$. Then α also satisfies the following formulas*

$$\forall xyw.\, root(fca(x, y)) = w \to w = root(x) \vee w = root(y)$$
$$\forall x.\, root(root(x)) = root(x)$$
$$\forall x.\, root(lleaf(x)) = root(x)$$
$$\forall x.\, root(rleaf(x)) = root(x)$$

Lemma 8 now suggests the following final definition for the closure operator Ψ:

$$\begin{aligned}
\mathit{Roots}(T) &\stackrel{\text{def}}{=} \mathit{root}^1(T) \cup \mathit{root}(\mathit{root}^1(T)) \\
\mathit{Leaves}(T) &\stackrel{\text{def}}{=} \mathit{lleaf}^1(T \cup \mathit{root}^1(T)) \cup \mathit{rleaf}^1(T \cup \mathit{root}^1(T)) \\
\Psi_2(T) &\stackrel{\text{def}}{=} T \cup \mathit{Roots}(T) \cup \mathit{Leaves}(T) \cup \mathit{lleaf}(\mathit{Leaves}(T)) \cup \mathit{rleaf}(\mathit{Leaves}(T)) \\
\Psi(\mathcal{K}, T) &\stackrel{\text{def}}{=} \Psi_5 \circ \Psi_4 \circ \Psi_3 \circ \Psi_2 \circ \Psi_1 \circ \Psi_0(\mathsf{st}(\mathcal{K}) \cup T)
\end{aligned}$$

One can easily check that Ψ satisfies the conditions (i) to (iv) on the closure operator of a Ψ-local theory, as defined in Section 4.

Lemma 9. *If there exists a weak partial model of $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ in which all terms in $\Psi(\mathcal{K}, G)$ are defined, then there exists a finite total model of $\mathcal{K} \cup G$.*

Lemma 9 implies that we can decide satisfiability of $\mathcal{K} \cup G$ using the decision procedure described in [9, Section 3.1]. Together with the previous Lemmas we conclude that the combination of the steps described in this section result in a decision procedure for the satisfiability problem of TREX.

Complexity. Note that the number of terms in $\Psi(\mathcal{K}, G)$ is polynomial in the size of $\mathcal{K} \cup G$. From the parametric complexity considerations for Ψ-local theories in [9, 16] follows that satisfiability of $\mathcal{K} \cup G$ can be checked in NP. Further note that all steps of the reduction, except for the elimination of function updates, increase the size of the formula at most by a polynomial factor. The case splits in the rewrite steps 2. and 3. of the function update elimination may cause that the size of the formula increases exponentially in the nesting depth of function updates in the original formula F_0. However, this exponential blowup can be easily avoided using standard techniques that are used, e.g., for efficient clausal normal form computation.

Theorem 10. *The satisfiability problem for TREX is NP-complete.*

Implementation and experiments. We started implementation of our decision procedure in the Jahob system. Our current prototype implements the first three steps of our decision procedure and already integrates with the verification condition generator of Jahob. Instead of manually instantiating the generated axioms, as described in the fourth step of our decision procedure, we currently give the generated axioms directly to the SMT solver and use triggers to encode some of the instantiation restrictions imposed by Ψ. While this implementation is not yet complete, we already successfully used it to verify implementations of operations on doubly-linked lists and a full insertion method on binary search trees (including the loop traversing the tree). The speedup obtained compared to using the MONA decision procedure is significant. For instance, using our implementation the verification of all 16 subgoals for the insert method takes about 1s in total. Checking the same subgoals using MONA takes 135s. We find these initial results encouraging and consistent with other success stories of using SMT solvers to encode NP decision procedures.

7 Conclusion

This paper introduced the logic TREX for reasoning about imperative tree data structures. The logic supports a transitive closure operator and a form of universal quantification. It is closed under propositional operations and weakest preconditions for heap manipulating statements. By analyzing the structure of partial and finite models, we have exhibited a particular Ψ-local axiomatization of TREX. This result then implies that the satisfiability problem for TREX is in NP. It also yields algorithms for generating model representations for satisfiable formulas, respectively, proofs of unsatisfiability.

References

1. I. Balaban, A. Pnueli, and L. D. Zuck. Shape analysis of single-parent heaps. In *VMCAI*, Lect. Notes in Comp. Sci. Springer, 2007.
2. A. Bouajjani, C. Dragoi, C. Enea, and M. Sighireanu. A logic-based framework for reasoning about composite data structures. In *CONCUR*, 2009.
3. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In *SAS*, 2006.
4. D. Calvanese, G. di Giacomo, D. Nardi, and M. Lenzerini. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier, 2001.
5. W. Charatonik and P. Witkowski. On the complexity of the bernays-schönfinkel class with datalog. In *LPAR (Yogyakarta)*, 2010.
6. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.
7. P. Genevès, N. Layaïda, and A. Schmitt. Efficient static analysis of XML paths and types. In *ACM PLDI*, 2007.
8. P. Habermehl, R. Iosif, and T. Vojnar. Automata-based verification of programs with tree updates. *Acta Inf.*, 47:1–31, January 2010.
9. C. Ihlemann, S. Jacobs, and V. Sofronie-Stokkermans. On local reasoning in verification. In *TACAS*, pages 265–281, 2008.
10. N. Immerman, A. M. Rabinovich, T. W. Reps, S. Sagiv, and G. Yorsh. The boundary between decidability and undecidability for transitive-closure logics. In *Computer Science Logic (CSL)*, pages 160–174, 2004.
11. N. Klarlund and A. Møller. *MONA Version 1.4 User Manual*. BRICS Notes Series NS-01-1, Department of Computer Science, University of Aarhus, January 2001.
12. V. Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.
13. S. Lahiri and S. Qadeer. Back to the future: revisiting precise program verification using SMT solvers. In *POPL*, 2008.
14. T. Lev-Ami, N. Immerman, T. Reps, M. Sagiv, S. Srivastava, and G. Yorsh. Simulating reachability using first-order logic with applications to verification of linked data structures. In *CADE-20*, 2005.
15. A. Podelski and T. Wies. Counterexample-guided focus. In *ACM Symposium on the Principles of Programming Languages (POPL 2010)*, pages 249–260. ACM, 2010.
16. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In *CADE*, pages 219–234, 2005.
17. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, August 1968.
18. T. Wies. *Symbolic Shape Analysis*. PhD thesis, University of Freiburg, 2009.
19. T. Wies, V. Kuncak, P. Lam, A. Podelski, and M. Rinard. Field constraint analysis. In *Proc. Int. Conf. Verification, Model Checking, and Abstract Interpratation*, 2006.
20. G. Yorsh, A. M. Rabinovich, M. Sagiv, A. Meyer, and A. Bouajjani. A logic of reachable patterns in linked data-structures. *J. Log. Algebr. Program.*, 2007.
21. K. Zee, V. Kuncak, and M. Rinard. Full functional verification of linked data structures. In *PLDI*, 2008.

A Additional Proofs

Lemma 4. *For every finite Σ_P-model α of the axioms in Figure 9,* $\alpha(P) = \alpha(\mathsf{p})^*$ *and* $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$.

Proof. Let α be a finite Σ_P-model of the axioms in Figure 9. We first prove $\alpha(\mathsf{p})^* = \alpha(P)$. For proving the left-to-right inclusion, let $u, v \in \alpha$ such that $\mathsf{p}^*(u, v)$. Then there exist $u_1, \ldots, u_n$ with $n \geq 1$ such that $u = u_1$, $v = u_n$, and for all $1 \leq i < n$, $\mathsf{p}(u_i) = u_{i+1}$. If $n = 1$ then $u = v$ and by axiom Refl we immediately have $P(u, v)$. If on the other hand $u \neq v$ then by p-Step we have for all $1 \leq i < n$, $P(u_i, \mathsf{p}(u_i))$ and thus $P(u_i, u_{i+1})$. Using axiom Trans we then conclude by induction on i that for all $1 < i \leq n$, $P(u_1, u_i)$. Hence, $P(u, v)$.

For proving the other inclusion, let $u \in \alpha$ and let $S_u = \{\, v \in \alpha \mid P(u, v)\,\}$. We show that for all $v \in S_u$, $\mathsf{p}^*(u, v)$. Since α is finite, so is S_u. Thus, using axioms Total and AntiSym we can construct an enumeration $u_1, \ldots, u_n$ of the elements of S_u such that for all $1 \leq i < j \leq n$, $P(u_i, u_j)$ but not $P(u_j, u_i)$. In particular $u_1 = u$. We prove by induction on i that for all $1 \leq i \leq n$, $\mathsf{p}^*(u_1, u_i)$. By reflexivity of p^* we immediately have $\mathsf{p}^*(u_1, u_1)$. Now assume that $\mathsf{p}^*(u_1, u_i)$. Since $u_i \in S_u$, we know by p-Step and Trans that $\mathsf{p}(u_i) \in S_u$. Hence, $\mathsf{p}(u_i) = u_j$ for some $j \geq i$. By p-Unfold we know that for all $j \geq i$, $u_i = u_j$ or $P(\mathsf{p}(u_i), u_j)$. Hence, by construction of the enumeration it follows that either $u_i = \mathsf{p}(u_i)$ or $u_{i+1} = \mathsf{p}(u_i)$. Assume $u_i = \mathsf{p}(u_i)$. Then from axiom p-Loop follows that $u_i = \alpha(\mathsf{null})$. However, we do not have $P(u_{i+1}, u_i)$, which contradicts the fact that α satisfies axiom NullTerm. Hence, we must have $u_{i+1} = \mathsf{p}(u_i)$. Together with the induction hypothesis we then conclude $\mathsf{p}^*(u_1, u_{i+1})$. We, thus, proved $P = \mathsf{p}^*$.

It remains to show that $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$. To this end, let $u_0, \ldots, u_n$ be an enumeration of all elements $u \in \alpha$ such that $\mathsf{p}(u) = \mathsf{null}$ and $u \neq \mathsf{null}$. Let further $V \stackrel{\text{def}}{=} \alpha \setminus \{u_0, \ldots, u_n, \mathsf{null}\}$. Note that p^+ is well-founded on V and moreover for all $v \in V$, $\mathsf{p}^+(v, u_i)$ for some u_i, $0 \leq i \leq n$. We can thus recursively define a function h from α to tree nodes as follows:

1. $h(\mathsf{null}) = \epsilon$,
2. $h(u_i) = i$, for all $0 \leq i \leq n$,
3. $h(v) = h(\mathsf{p}(v))L$, for all $v \in V$ with $\mathsf{l}(\mathsf{p}(v)) = v$, and
4. $h(v) = h(\mathsf{p}(v))R$, for all $v \in V$ with $\mathsf{r}(\mathsf{p}(v)) = v$

Axiom lr-Distinct and the definitions of V and the u_i ensure that the 4 cases are exclusive, thus h is well-defined. Moreover, axiom Parent ensures that h is totally defined on α. Furthermore, h is injective by construction. Let N be the range of h, then N is a prefix-closed set of tree nodes, again by construction of h. Let $\alpha_{\mathcal{F}} \in \mathcal{F}$ be the forest with $\alpha_{\mathcal{F}}(\mathsf{node}) = N$ and $\alpha_{\mathcal{F}}(c) = h(\alpha(c))$ for all $c \in \Gamma$. From the axioms l-Child, r-Child, l-Root, r-Root and the construction of h and $\alpha_{\mathcal{F}}$ now follows that h is a structure isomorphism between $\alpha|_{\Sigma_{\mathcal{F}}}$ and $\alpha_{\mathcal{F}}$. Thus $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$. □

Lemma 5. *The TREX formula F_2 has a model in $\mathcal{M}_{\mathcal{F}}$ iff the Σ_P-formula F_3 has a finite Σ_P-model.*

Proof. We first prove that all finite Σ_P-models of the axioms in Figure 9 and Figure 10 satisfy property (1). Let α be such a model and let $u, v \in \alpha$. Assume $\langle \mathsf{p} \rangle_Q^*(u, v)$. First note that by axiom bp_Q-Def4 (and $\alpha(P) = \alpha(\mathsf{p})^*$) we have $\mathsf{p}^*(u, bp_Q(u))$. Furthermore, by definition either (i) $u = v$ or (ii) there exist distinct $u_0, \ldots, u_n$ with $n \geq 1$ such that $u_0 = u$, $u_n = v$, and for all i, $0 \leq i < n$, $\mathsf{p}(u_i) = u_{i+1}$ and $Q(u_i, u_{i+1})$. In case (i) we then immediately have $p^*(u, v)$ and $\mathsf{p}^*(v, bp_Q(u))$. In case (ii) we also immediately have $\mathsf{p}^*(u, v)$. Moreover, $\mathsf{p}^*(u, bp_Q(u))$ and functionality of p imply that either $bp_Q(u) = u_i$ for some i, $1 \leq i < n$, or $\mathsf{p}^*(v, bp_Q(u))$. In the first case we have by axiom bp_Q-Def2, $bp_Q(u) = \mathsf{null}$ and thus $u_i = u_{i+1} = \mathsf{null}$, which contradicts that u_i and u_{i+1} are distinct. Hence, $\mathsf{p}^*(v, bp_Q(u))$.

For proving the other direction assume $\mathsf{p}^*(u, v)$ and $\mathsf{p}^*(v, bp_Q(u))$. If $u = v$ then $\langle \mathsf{p} \rangle_Q^*(u, v)$ by definition. Otherwise there exist distinct $u_0, ..., u_n$ with $n \geq 1$ such that $u_0 = u$, $u_n = v$, and for all i, $0 \leq i < n$, $\mathsf{p}(u_i) = u_{i+1}$. Then for all $0 \leq i < n$ we have $\mathsf{p}^*(u, u_i)$ and $\mathsf{p}^*(u_i, bp_Q(u))$ by transitivity of p^*. Thus by axiom bp_Q-Def3 we have for all $0 \leq i < n$, $Q(u_i, u_{i+1})$ or $u_i = bp_Q(u)$. If for some i we have $u_i = bp_Q(u)$ then by AntiSym, $u_i = u_j$ for all j, $i \leq j \leq n$, which gives a contradiction. Hence, for all i, $0 \leq i < n$, $Q(u_i, u_{i+1})$ and therefore $\langle \mathsf{p} \rangle_Q^*(u, v)$.

Now let $\alpha_{\mathcal{F}} \in \mathcal{M}_{\mathcal{F}}$ be a model of F_2. Then extend $\alpha_{\mathcal{F}}$ to a Σ_P-structure by defining $\alpha_{\mathcal{F}}(P) = \mathsf{p}^*$ and for all $Q \in \mathcal{Q}$ and $u \in \alpha_{\mathcal{F}}$ let $\alpha_{\mathcal{F}}(bp_Q)(u) = \mathsf{p}^i(u)$ where $i = \min \left\{ j \in \mathbb{N} \mid \neg Q(\mathsf{p}^j(u), \mathsf{p}^{j+1}(u)) \lor \mathsf{p}^j(u) = \mathsf{null} \right\}$. One can easily verify that $\alpha_{\mathcal{F}}$ is a model of the axioms in figures 9 and 10. Hence, $\alpha_{\mathcal{F}}$ satisfies property (1) and is a finite model of F_3.

For proving the other direction of the lemma, assume that α is a finite Σ_P-model of F_3. Then from Lemma 4 we know that $\alpha(P) = \alpha(\mathsf{p})^*$ and $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$. Hence, α satisfies property (1) and therefore $\alpha|_{\Sigma_{\mathcal{F}}} \models F_2$. $\square$

Lemma 7. *The formula F_3 has a finite Σ_P-model if and only if $\mathcal{K} \cup G$ has a finite Σ-model.*

Proof. We first show that for all $f \in \{\mathsf{p}, \mathsf{l}, \mathsf{r}\}$, all models α of the axioms in Figures 9, 12, and 13, and all $u \in \alpha$:

$$N_f(u) \text{ iff } f(u) = \mathsf{null} \tag{3}$$

Let α be such a model and $u \in \alpha$. Note that we have $\alpha(P) = \alpha(\mathsf{p})^*$ and $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$ according to Lemma 4. First, consider the case where $f = \mathsf{p}$. For proving the left-to-right direction, assume $root(u) = u$ but $\mathsf{p}(u) \neq \mathsf{null}$. Then $\mathsf{p}(root(u)) \neq root(u)$ and $\mathsf{p}(root(u)) \neq \mathsf{null}$. Then $P(root(u), \mathsf{p}(root(u))$ implies $P(u, \mathsf{p}(root(u))$ but we do not have $P(\mathsf{p}(root(u), root(u)))$, which contradicts axiom Root1. For proving the other direction assume $\mathsf{p}(u) = \mathsf{null}$. We distinguish two cases: if $root(u) = \mathsf{null}$ then $u = \mathsf{null} = root(u)$ according to axiom Root2. If however $root(u) \neq \mathsf{null}$ then $u \neq \mathsf{null}$, again by axiom Root2. Hence $P(u, u)$ implies $P(u, root(u))$ by axiom Root1 and thus $u = root(u)$ because $\mathsf{p}(u) = \mathsf{null}$.

Next consider the case where $f = \mathsf{l}$. Assume $lleaf(u) = \mathsf{null}$. Then from axiom l-Leaf2 immediately follows $\mathsf{l}(u) = \mathsf{null}$. For proving the other direction assume $\mathsf{l}(u) = \mathsf{null}$ but $lleaf(u) \neq \mathsf{null}$. From l-Leaf1 now follows $P(lleaf(u), u)$. Since axiom l-Leaf3 implies $lleaf(u) \neq u$, we must have $P(lleaf(u), \mathsf{r}(u))$ and $\mathsf{r}(u) \neq \mathsf{null}$. Then axiom r-Leaf2 implies $P(rleaf(u), \mathsf{r}(u))$ and $rleaf(u) \neq \mathsf{null}$. From axiom Leaves1 we then conclude $fca(lleaf(u), rleaf(u)) = u$ and therefore $u = \mathsf{r}(u)$. But this implies $u = \mathsf{r}(u) = \mathsf{null}$, which gives a contradiction. The case for $f = \mathsf{r}$ is analogous.

Now, for proving the lemma first note that skolemization, linearization, flattening, and computation of clausal normal form are all satisfiability-preserving transformations. In particular, $K_P \cup G$ is satisfiability if and only if F_3 is satisfiable. Moreover, every finite Σ_P-model of $K_P \cup G$ is also a model of F_3. Thus, let α be a finite Σ_P-model of $K_P \cup G$. Let $u \in \alpha$ and $Q \in \mathcal{Q}$. From axiom bp_Q-Def3 follows that $\neg Q(bp_Q(u), p(bp_Q(u)))$ or $bp_Q(u) = \mathsf{null}$. In the first case, the syntactic restrictions on Q imply that we must have $bp_Q(u) \in BP(Q)$. In the second case $bp_Q(u) \in BP(Q)$ follows immediately from the definition of BP. Hence, α satisfies axiom bp_Q-Def5. Axiom bp_Q-Def4 is a consequence of axioms Refl, Trans, AntiSym, NullTerm, and the axioms in Figure 10. Thus, α also satisfies bp_Q-Def4.

The axioms Refl, Trans, AntiSym, and NullTerm ensure that P is a partial order on α for which all upper bounds exist. Define $\alpha(fca)$ as the function associating with every pair $u, v \in \alpha$ the least upper bound of u and v with respect to P. Then α satisfies axioms fca-Def1, fca-Def2, and fca-Def3. Moreover, using $\alpha(P) = \alpha(\mathsf{p})^*$ and axiom Parent one case easily show that α also satisfies fca-Def4.

Now for each $u \in \alpha$, define $\alpha(root)(u) = u$ if $\mathsf{p}(u) = \mathsf{null}$ and otherwise define $\alpha(root)(u) = v$ where v is the unique element in α such that $P(u, v)$, $v \neq \mathsf{null}$ and $\mathsf{p}(v) = \mathsf{null}$. Similarly, define $lleaf(u) = v$ where v is chosen freely from all $v' \in \alpha$ such that $\mathsf{l}(v') = \mathsf{null}$, $\mathsf{r}(v') = \mathsf{null}$, and $P(v', \mathsf{l}(u))$. The fact that at least one such v' exists easily follows from $\alpha|_{\Sigma_{\mathcal{F}}} \in \mathcal{M}_{\mathcal{F}}$ and $\alpha(P) = \alpha(\mathsf{p})^*$. Define $rleaf(u)$ analogously. Then one can easily verify that α satisfies all axioms in Figure 13. Hence, α satisfies property (3) and is a model of $\mathcal{K} \cup G$.

For proving the other direction let α be a finite Σ-model of $\mathcal{K} \cup G$. Then from property (3) immediately follows that $\alpha|_{\Sigma_P}$ satisfies $\mathcal{K}_P \cup G$. $\square$

Lemma 9. *If there exists a weak partial model of $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ in which all terms in $\Psi(\mathcal{K}, G)$ are defined, then there exists a finite total model of $\mathcal{K} \cup G$.*

Proof Sketch. Let α be a weak partial model of $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ in which all terms in $\Psi_{\mathcal{K}}(G)$ are defined. We can obtain a finite partial substructure α_0 from α by restricting the universe of α to the elements that are used to interpret the ground terms in $\Psi_{\mathcal{K}}(G)$. Then α_0 still weakly satisfies $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$. Furthermore, from the definition of $\Psi(\mathcal{K}, G)$, Lemma 6, and the fact that α_0 weakly satisfies $\mathcal{K}[\Psi(\mathcal{K}, G)]$ follows:

(a) P is totally defined in α_0,
(b) fca is totally defined in α_0,
(c) for each $Q \in \mathcal{Q}$, bp_Q is totally defined in α_0,
(d) for each $Q \in \mathcal{Q}$ and $u \in \alpha_0$, p is defined on $bp_Q(u)$.
(e) for each $u \in \alpha_0$ and $f \in \{\mathsf{l}, \mathsf{r}\}$, if f is defined on u then p is defined on $f(u)$.

From these properties follows that α_0 is already a model of all clauses in $\mathcal{K}$ that result from the axioms: NullTerm, Refl, Trans, AntiSym, Total, fca-Def1, fca-Def2, fca-Def3, fca-Def4, as well as the axioms bp_Q-Def1, bp_Q-Def2, bp_Q-Def4, and bp_Q-Def5 for all

$Q \in \mathcal{Q}$. Note further that α_0 is a model of all clauses in $\mathcal{K}$ that result from restricted quantified formulas of the form $\forall z.\, F_{\mathsf{in}}$ occuring in F_0 because such clauses only contain constant symbols, the function symbols bp_Q, and the predicate symbol P.

We now complete α_0 to a total model of $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ step by step starting with the interpretation of the function symbol $root$. Let α_1 be the structure α_0 where the interpretation of $root$ is completed to a total function as follows: we know that even if $root$ is undefined for some $u \in \alpha_0$ then the root $r \in \alpha_0$ of the tree to which u belongs is already determined, i.e., $r = root(v)$ for some other $v \in \alpha_0$ that belongs to the same tree as u. This can be seen as follows: from the definition of Ψ we conclude that if $root$ is undefined for some $u \in \alpha_0$ then u is either one of (i) $lleaf(v)$ for some $v \in \alpha_0$ and $r = root(v)$, (ii) $rleaf(v)$ for some $v \in \alpha_0$ and $r = root(v)$, or (iii) $fca(v, w)$ for some $v, w \in \alpha_0$ such that either $root(v) = r$ or $root(w) = r$, or cases (i),(ii) apply to both v and w. We can thus define $\alpha_1(root)(u) = r$ where r is determined by the case above that applies. Note that in all cases $root$ is already defined on r in α_0, i.e., $r \neq u$. We thus conclude that α_1 still weakly satisfies $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ and, by construction, is now also a model of the clauses in $\mathcal{K}$ that result from the axioms Root1 and Root2, as well as the clauses that result from restricted quantified formulas of the form $\forall z.\, f_{\mathsf{p}}(z) = t \rightarrow G_{\mathsf{in}}$ in F_0.

Using similar reasoning we can show that whenever $lleaf$ or $rleaf$ are undefined for some $u \in \alpha_1$ then there exists a corresponding descendant $v \in \alpha_1$ of u such that $lleaf$, respectively $rleaf$ is defined on v with image $w \in \alpha_1$, and w is different from null. We can thus extend α_1 to a weak partial model α_2 of $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ such that α_2 also interprets the function symbols $lleaf$ and $rleaf$ as total functions and is now a model of all clauses that result from restricted quantified formulas in F_0, as well as all clauses that result from the axioms given in Figure 13, except for the axioms l-Leaf2 and r-Leaf2.

Now define the function $parent \in \alpha_2 \to \alpha_2$ as follows: for all $u \in \alpha_2$, if $u = \mathsf{null}$ then define $parent(u) = \mathsf{null}$ and otherwise define $parent(u) = v$ such that

$$v \neq u, P(u, v), \text{ and for all } w \in \alpha_2, \text{ if } w \neq u \text{ and } P(u, w) \text{ then } P(v, w) \tag{4}$$

We argue that $parent$ is well-defined. First, from the fact that α_2 is finite and satisfies axioms NullTerm, Total, and Trans we conclude that there exists at least one $v \in \alpha_2$ satisfying condition (4). Furthermore, from axioms AntiSym follows that this v is unique. We further argue that for all $u \in \alpha_2$ if p is defined on u then $parent(u) = \mathsf{p}(u)$. If $u = \mathsf{null}$ then $\mathsf{p}(\mathsf{null}) = \mathsf{null}$ follows from axioms p-Step, NullTerm, and AntiSym. If on the other hand $u \neq \mathsf{null}$ then axiom p-Loop implies $\mathsf{p}(u) \neq u$ and axiom p-Step implies $P(u, \mathsf{p}(u))$. Furthermore, axiom p-Unfold implies that for all $w \in \alpha_2$ with $w \neq u$, $P(\mathsf{p}(u), w)$ holds. Hence, condition (4) is satisfied for $v = \mathsf{p}(u)$.

Now define $\alpha_3 \stackrel{\text{def}}{=} \alpha_2[\mathsf{p} \mapsto parent]$ then α_3 still weakly satisfies $\mathcal{K}[\Psi(\mathcal{K}, G)] \cup G$ and, by construction, is a model of the axioms p-Loop, p-Step, and p-Unfold. From property (d) and the fact that α_2 is a model of axiom bp_Q-Def4, for all $Q \in \mathcal{Q}$, further follows that α_3 is also a model of the clauses resulting from the axiom bp_Q-Def3, for all $Q \in \mathcal{Q}$.

In order to complete the interpretations of l and r to total functions, and hence α_3 to a total structure, we first define a function $Children$ that maps every node $u \in \alpha_3$ with

$u \neq \mathsf{null}$, to the set of its proper children as follows:

$$Children(u) \stackrel{\text{def}}{=} \{\, v \in \alpha_3 \mid \mathsf{p}(v) = u \,\}$$

Let $v, w \in Children(u)$ such that $v \neq w$. Then from axioms fca-Def1, fca-Def2, fca-Def3, and condition (4) follows that $u = fca(v, w)$. Axiom fca-Def4 thus implies that $Children(u)$ contains at most two elements. We can then use the function $Children$ to define functions $left, right \in \alpha_3 \to \alpha_3$ as follows: first define $left(\mathsf{null}) = right(\mathsf{null}) = \mathsf{null}$. For $u \in \alpha_3$ with $u \neq \mathsf{null}$ we distinguish three cases:

Case 1: both $lleaf(u)$ and $rleaf(u)$ are different from null. Then from axiom Leaves1, l-Leaf1, r-Leaf1, the axioms for fca, and condition (4) follows that $Children(u)$ must contain two distinct elements $v_{\mathsf{l}}, v_{\mathsf{r}}$ such that $P(lleaf(u), v_{\mathsf{l}})$ but not $P(rleaf(u), v_{\mathsf{l}})$ and, vice versa, $P(rleaf(u), v_{\mathsf{r}})$ but not $P(lleaf(u), v_{\mathsf{r}})$. Then define $left(u) = v_{\mathsf{l}}$ and $right(u) = v_{\mathsf{r}}$.

Case 2: exactly one of $lleaf(u)$ and $rleaf(u)$ is null. Assume $rleaf(u) = \mathsf{null}$ then from axiom Leaves2, the axioms for fca, and condition (4) follows that $Children(u)$ contains exactly one element v, moreover $P(lleaf(u), v)$ must hold because of axiom l-Leaf1. Thus define $left(u) = v$ and $right(u) = \mathsf{null}$. The case for $lleaf = \mathsf{null}$ is analogous.

Case 3: both $lleaf(u)$ and $rleaf(u)$ are null in α_3. Then from axiom Leaves3 and condition (4) follows that $Children(u) = \emptyset$. We thus define $left(u) = right(u) = \mathsf{null}$.

We show that $left$ and $right$ extend the interpretations of l and r in α_3 to total functions. Assume $f(u)$ is defined for some $u \in \alpha_3$ where $f \in \{\mathsf{l}, \mathsf{r}\}$. If $u = \mathsf{null}$ then $f(u) = \mathsf{null}$ follows from property (e) and the fact that α_3 weakly satisfies axioms l-Child and l-Root, respectively, r-Child and r-Root. If on the other hand $u \neq \mathsf{null}$ then from property (e) and the fact that α_3 weakly satisfies axioms l-Leaf2 and r-Leaf2 follows $P(lleaf(u), f(u))$ if $f = \mathsf{l}$, respectively, $P(rleaf(u), f(u))$ if $f = \mathsf{r}$. If $f(u) = \mathsf{null}$ then together with axioms NullTerm and AntiSym this immediately implies $lleaf(u) = \mathsf{null}$, respectively, $rleaf(u) = \mathsf{null}$ and hence $left(u) = \mathsf{null}$, respectively, $right(u) = \mathsf{null}$. If $f(u) \neq \mathsf{null}$ then from the fact that α_3 weakly satisfies the axioms l-Child and r-Child follows that $f(u) \in Children(u)$. Together with $P(lleaf(u), f(u))$, respectively, $P(rleaf(u), f(u))$ this again implies $left(u) = f(u)$, respectively, $right(u) = f(u)$.

Now, define $\alpha_4 \stackrel{\text{def}}{=} \alpha_3[\mathsf{l} \mapsto left, \mathsf{r} \mapsto right]$. Then by construction α_4 is also a model of all remaining clauses in $\mathcal{K}$ that result from axioms in which the function symbols l and r occur and, thus, α_4 is a model of $\mathcal{K} \cup G$. Moreover, α_4 is finite. $\square$