Decidability and Complexity of Tree Share Formulas

Xuan Bach Le¹, Aquinas Hobor², and Anthony W. Lin³

¹ School of Computing, National University of Singapore
lxbach@comp.nus.edu.sg
² Yale-NUS College and School of Computing, National University of Singapore
hobor@comp.nus.edu.sg
³ Yale-NUS College
anthony.w.lin@yale-nus.edu.sg

Abstract

Fractional share models are used to reason about how multiple actors share ownership of resources. We examine the decidability and complexity of reasoning over the “tree share” model of Dockins et al. using first-order logic, or fragments thereof. We pinpoint a connection between the basic operations on trees union \( \sqcup \), intersection \( \sqcap \), and complement \( \Box \) and countable atomless Boolean algebras, allowing us to obtain decidability with the precise complexity of both first-order and existential theories over the tree share model with the aforementioned operations. We establish a connection between the multiplication operation \( \bowtie \) on trees and the theory of word equations, allowing us to derive the decidability of its existential theory and the undecidability of its full first-order theory. We prove that the full first-order theory over the model with both the Boolean operations \( (\sqcup, \sqcap, \Box) \) and the restricted multiplication operation \( \bowtie \) with constants on the right hand side) is decidable via an embedding to tree-automatic structures.

1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases Fractional Share Models, Resource Accounting, Countable Atomless Boolean Algebras, Word Equations, Tree Automatic Structures

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

The state of the art: Fractional shares enable reasoning about shared ownership of resources between multiple parties, e.g. ownership of memory cells by different threads in a concurrent program [7]. Threads are then allowed to take actions depending on the amount of ownership they have, e.g. with full ownership allowing both reading and writing, partial ownership allowing only reading, and empty ownership allowing nothing. Although rational numbers are the most obvious model for fractional shares, they are unfortunately not a good model for realistic program verification because they do not satisfy the so-called “disjointness” axiom [3], i.e. \( \forall x, y. x + x = y \Rightarrow x = y = 0 \). Dockins et al. proposed a better model for fractional shares based on binary trees with Boolean leaves [10]. A tree share \( \tau \in \mathbb{T} \) is inductively defined as follows: \( \tau \triangleq \circ | \bullet | \tau \bowtie \tau \), where \( \circ \) denotes an “empty” leaf while \( \bullet \) a “full” leaf. The tree \( \circ \) is thus the empty share, and \( \bullet \) the full share. There are two “half” shares: \( \circ \bullet \) and \( \bullet \circ \), and four “quarter” shares, beginning with \( \circ \bullet \circ \circ \). It is a feature that the two half shares in \( \mathbb{T} \) are distinct, as compared to the two half shares in \( \mathbb{Q} \), 0.5 and 0.5,
which are of course equal. The ability to represent distinct partial shares of “equal measure” is closely related to why the disjointness axiom holds. The basic operations for combining trees are union $\sqcup$, intersection $\sqcap$, and complement $\Box$; these will be defined formally in §2 but to a first approximation they are all defined leafwise, e.g. $\circ \sqcup \bullet = \bullet \circ \circ$.

A number of program logics incorporate tree shares to model fractional ownership [11, 12, 29, 3], but it has been unclear how to reason about them automatically, which has posed a significant barrier to their use in verification tools. One reason for this barrier is the lack of foundational results regarding decidability and complexity of theories over tree shares. The only published result of this kind proves the decidability of entailment between systems of equations over tree shares, a less-expressive format than general first-order formulae [19].

In addition to union, intersection, and complement, Dockins et al. defined a “multiplication” operator on tree shares, written $\tau_1 \triangleright \tau_2$ [10]. The basic idea is that you take each $\bullet$ leaf in $\tau_1$ and replace it with a full copy of $\tau_2$, e.g. $\circ \triangleright \bullet = \bullet \circ \circ \circ \circ$.

Dockins et al. showed that $\triangleright$ could be used to split any nonempty tree $\tau$ into two nonempty trees that joined together to equal the original since $\forall \tau. \tau = (\tau \triangleright \circ \circ) \sqcup (\tau \triangleright \circ \circ \circ \circ \circ)$. More generally, the $\triangleright$ operator can be used as a kind of “scoping” or “gluing” operator to combine different uses of tree shares together. Although $\triangleright$ has been used in metatheory [3], it has never been used in an automated tool because its decidability properties were unclear.

Contributions: In this paper, we provide the first systematic study of decidability and complexity of theories over the tree share model.

First (§3), we show that the tree share model $M \triangleq (\sqcap, \sqcup, \Box, \circ, \bullet)$ is a Countable Atomless Boolean Algebra (CABA), which are known to be unique up to isomorphism [28]. The first-order theory over CABAs is known to be decidable and, in fact, complete for the class STA$(\ast, 2^m, n)$ of problems solvable by an alternating Turing machine with $n$ alternations in exponential time [17], the same complexity class as the first-order theory over $(\mathbb{R}, +, 0, 1)$ [4]. In addition, the full existential theory over CABAs is known to be NP-complete [24]. Our connection shows that these decidability and complexity results transfer to $M$.

We then (§4) proceed to decision problems over the tree shares with the multiplication operator $\triangleright$. Our main result here is that the tree share model $S \triangleq (T, \triangleright)$ that only allows $\triangleright$ (i.e. but not $\sqcup, \sqcap$, and $\Box$) is—in a technical sense—“equivalent” to the logical structure of words with the concatenation operator. Makanin [20] showed that reasoning about a single equation over this structure (a.k.a. word equations) is decidable. More complex problems are known to be reducible to this basic case in polynomial-time, e.g. the existential theory over the structure [8]. Accordingly, we deduce that the existential theory over $S$ is decidable in polynomial space but NP-hard, whereas the first-order theory over $S$ is undecidable.

Finally (§5), we consider restrictions on $\triangleright$ that admit a decidable theory. We define the family of one-argument functions indexed by tree constants that applies bowtie on the right-hand side $\triangleright_{\Box}$, i.e. $\triangleright_{\Box}(\tau') \triangleq \tau' \triangleright \tau$. We prove that the combined theory of $T' \triangleq (T, \sqcap, \sqcup, \Box, \circ, \triangleright_{\Box})$ has an embedding into tree-automatic structures. Since the first-order theory of tree-automatic structures is decidable [6], we obtain the decidability of the first-order theory of this extension of the tree share model with $\triangleright_{\Box}$. This suggests the potential application of powerful heuristics for automata (e.g. antichain and simulation [1]) for providing a practical decision procedure for the tree share model.
2 Formal preliminaries: the Tree Share model $T$ of Dockins et al. [10]

Here we summarize additional details of tree shares and their associated theory from [10].

**Canonical forms.** In the first paragraph of §1 we presented the first quarter share as $\circ$ instead of e.g. $\circ$. This is deliberate: the second choice is not a valid share because the tree is not in **canonical form**. A tree is in canonical form when it is in its most compact representation under the inductively-defined equivalence relation $\cong$:

$$
\begin{array}{ccc}
\circ & \cong & \circ \\
\cdot & \cong & \cdot
\end{array}
\quad
\begin{array}{ccc}
\circ & \cong & \circ \\
\circ & \cong & \circ
\end{array}
\quad
\begin{array}{ccc}
\circ & \cong & \circ \\
\circ & \cong & \circ
\end{array}
$$

As we will see, operations on tree shares sometimes need to fold/unfold trees to/from canonical form, a practice we will indicate using the symbol $\cong$. Canonicality is needed to guarantee some of the algebraic properties of tree shares: managing it requires a little care in the proofs but does not pose any fundamental difficulties to the overall theory.

**Boolean algebra operations.** The connectives $\sqcup$ and $\sqcap$ first unfold both trees to the same shape; then calculate leafwise using the rules $\circ \sqcup \tau = \tau \sqcup \circ = \tau$, $\cdot \sqcup \tau = \tau \sqcup \cdot = \tau$, $\circ \sqcap \tau = \tau \sqcap \circ = 0$, and $\cdot \sqcap \tau = \tau \sqcap \cdot = \tau$; and finally refold back into canonical form, e.g:

$$
\begin{array}{ccc}
\circ & \circ & \circ \\
\cdot & \cdot & \cdot
\end{array}
\quad
\begin{array}{ccc}
\cdot & \cdot & \cdot \\
\circ & \circ & \circ
\end{array}
\quad
\begin{array}{ccc}
\circ & \circ & \circ \\
\circ & \circ & \circ
\end{array}
$$

Complementation is simpler, since flipping leaves between $\circ$ and $\cdot$ does not affect whether a tree is in canonical form, e.g. $\circ \cdot \cong \cdot \circ$. Using these definitions we get all of the usual properties for Boolean algebras, e.g. $\overline{\tau_1 \sqcup \tau_2} = \overline{\tau_1} \sqcap \overline{\tau_2}$. Moreover, we can define a partial ordering between trees using intersection in the usual way, i.e. $\tau_1 \sqsubseteq \tau_2 \triangleq \tau_1 \sqcap \tau_2 = \tau_1$. We can enjoy a strict partial order as well: $\tau_1 \sqsubset \tau_2 \triangleq \tau_1 \sqsubseteq \tau_2 \wedge \tau_1 \neq \tau_2$.

**Properties of tree multiplication $\bowtie$.** Since it is nonstandard, the “tree multiplication” operator $\bowtie$ deserves some additional attention. The good news first: $\bowtie$ is associative, has an identity $\cdot$, and is injective for non-$\circ$ elements, i.e. $S^+ \triangleq (T \setminus \{\circ\}, \bowtie)$ forms a cancellative monoid. Somewhat unsurprisingly, multiplication by the “additive identity” $\circ$ reduces to $\circ$.

Unfortunately, $\bowtie$ is not commutative:

$$
\begin{array}{ccc}
\circ & \bowtie & \circ \\
\cdot & \bowtie & \cdot
\end{array}
\quad
\begin{array}{ccc}
\circ & \bowtie & \circ \\
\circ & \bowtie & \circ
\end{array}
\quad
\begin{array}{ccc}
\circ & \bowtie & \circ \\
\circ & \bowtie & \circ
\end{array}
$$

although we do enjoy a distributive property over $\sqcup$ and $\sqcap$ on the right hand side. Accordingly:

▸ **Lemma 1** (Properties of $\bowtie$).

\begin{align}
\text{Associativity:} & \quad \tau_1 \bowtie (\tau_2 \bowtie \tau_3) = (\tau_1 \bowtie \tau_2) \bowtie \tau_3 \\
\text{Identity element:} & \quad \tau \bowtie \cdot = \cdot \bowtie \tau = \tau \\
\text{Zero element:} & \quad \tau \bowtie \circ = \circ \bowtie \tau = \circ \\
\text{Left cancellation:} & \quad \tau \neq \circ \Rightarrow \tau \bowtie \tau_1 = \tau \bowtie \tau_2 \Rightarrow \tau_1 = \tau_2 \\
\text{Right cancellation:} & \quad \tau \neq \circ \Rightarrow \tau_1 \bowtie \tau = \tau_2 \bowtie \tau \Rightarrow \tau_1 = \tau_2 \\
\text{Right distributivity over $\sqcap$:} & \quad \tau_1 \bowtie (\tau_2 \sqcap \tau_3) = (\tau_1 \bowtie \tau_2) \sqcap (\tau_1 \bowtie \tau_3) \\
\text{Right distributivity over $\sqcup$:} & \quad \tau_1 \bowtie (\tau_2 \sqcup \tau_3) = (\tau_1 \bowtie \tau_2) \sqcup (\tau_1 \bowtie \tau_3)
\end{align}
23:4 Decidability and Complexity of Tree Share Formulas

Typical use of $T$ in program verification. A standard way to use fractional shares in program verification is by modifying the standard maps-to predicate of separation logic to take a share as an additional argument. The predicate $x \mapsto y$ then means that the heap has a cell at address $x$, which is owned with nonempty fraction $\pi \neq \circ$ and whose value is $y$. We use $\pi$ here because we often use share variables rather than concrete trees $\tau$.

To combine divided fractional ownership stakes back together it is traditional to use the “join” relation, written $\tau_1 \oplus \tau_2 = \tau_3$. The join relation is defined in turn using the primitive Boolean algebra operators: $\tau_1 \oplus \tau_2 = \tau_3 \overset{\Delta}{=} \tau_1 \sqcup \tau_2 = \tau_3 \land \tau_1 \sqcap \tau_2 = \circ$. In other words, the join relation is a kind of disjoint union; it is partial because $\circ \oplus \circ$ is undefined. Critically for verification $\oplus$ does satisfy the disjointness axiom: $\forall x, y. x \oplus x = y \Rightarrow x = y = \circ$. Using $\oplus$ we can state the following relationship between the spatial conjunction $\land$ and the underlying Boolean operators as $x \mapsto y \ast x \mapsto z \Rightarrow y = z \land x \mapsto \pi_1 \oplus \pi_2 = y$.

It is common that we want to “split” a share $\pi$ into sub-shares $\pi_1, \pi_2$ so that the permission can be transferred. This can be done within $M \overset{\Delta}{=} (\sqcap, \sqcup, \circ, \bullet)$ using the following rule:

$$\begin{array}{c}
\pi \neq \circ \\
x \mapsto v \quad \vdash \exists \pi_1, \pi_2. \ (x \mapsto v \ast x \mapsto v) \land \pi_1 \oplus \pi_2 = \pi \land \pi_1 \neq \circ \land \pi_2 \neq \circ \\
\end{array} \quad \text{SplitJoin}$$

This rule has some drawbacks. The most obvious is the lengthy size of the entailment’s consequent, even though we only split $\pi$ into two pieces. Second, existential quantifiers are expensive in program verification since they tend to increase the size of the proof obligations and here we introduce two of them. Third, we have no control over what the shares $\pi_1$ and $\pi_2$ are—that is, $\pi_1$ and $\pi_2$ are not uniquely determined. Moreover, they are indistinguishable, which makes it difficult to assign different permitted actions for them.

On the other hand, each of these issues can be solved nicely using $\bowtie$ due to its right distributivity over $(\sqcap, \sqcup)$, and thus over $\oplus$, yielding the following rules:

$$\begin{array}{c}
\tau_1 \oplus \cdots \oplus \tau_n = \tau \quad \pi \neq \circ \\
\vdash \bigwedge_{i=1}^{n} \tau_i \neq \circ \\
x \mapsto v \quad \vdash x \mapsto v \ast \cdots \ast x \mapsto v \\
\end{array} \quad \text{SplitJoin} \bowtie (8)$$

3 Tree Shares are a model for Countable Atomless Boolean Algebras

In this section, we pinpoint the fact that $M = (\sqcap, \sqcup, \circ, \bullet)$ is a model for Countable Atomless Boolean Algebra (CABA). Let $B = (\sqcap, \sqcup, \circ, \bullet)$ be a Boolean Algebra (BA), we define a partial order $\subseteq$ on $B$ (for $M$, resp.): $a_1 \sqsubseteq a_2 \overset{\Delta}{=} a_1 \sqcap a_2 = \circ$ and $a_1 \subset a_2 \overset{\Delta}{=} a_1 \sqsubseteq a_2 \land a_2 \not\sqsubseteq a_1$. $B$ is atomless if $\forall a. \ 0 \subset a \Rightarrow \exists a'. \ 0 \subset a' \subset a$. $B$ is countable if its domain is countable. Dockins et al. [10] proved that $M$ is a model for BA where $0 \overset{\Delta}{=} \circ, \ 1 \overset{\Delta}{=} \bullet, \ \sqcup \overset{\Delta}{=} \sqcup, \ \sqcap \overset{\Delta}{=} \sqcap$. The atomless property can be derived from the Infinite Splitability property of tree shares [19]; let $a \sqcup \circ$ and $a_1 \sqcup a_2 \neq \circ$ such that $a_1 \sqcup a_2 = a$. This implies $a_1 \sqcup a_2 = a \land a_1 \sqcap a_2 = \circ$ and thus $a_1 \sqcup a$. The proof that $T$ is countable is achieved by enumerating $T$ in the ascending order of tree height $|\tau|$ using the following total strict order $\prec$:

$$\begin{array}{c}
\tau_1 \prec \tau_2 \\
|\tau_1| < |\tau_2| \\
\end{array}$$

It is known that there is a unique model for CABA up to isomorphism [28], so we can reason about the complexity of $M$ in terms of CABA. Let $\text{STA}(f(n), g(n), h(n))$ be the
class of sets accepted by alternating Turing machines which use at most \( f(n) \) space, \( g(n) \) time and \( h(n) \) alternations between universal and existential states on a given input of length \( n \). Any field in the description can be replaced with symbol \( * \) to indicate no bound is required. Kozen [17] proved that the elementary theory of infinite BAs is \( \leq \log \)-complete for the Berman complexity class \( \bigcup_{c < \omega} \text{STA}(*, 2^{cn}, n) \), which lies between the class of deterministic exponential space and non-deterministic exponential space.

We now investigate the complexity of an important sub-theory of \( \mathcal{M} \), namely the existential theory. Basically, this sub-theory includes all valid sentences whose prenex normal form contains only existential quantifiers. Its counterpart is the universal theory in which all the quantifiers are universal. A result by Marriott et al. [24] showed that the existential theory and universal theory for infinite BAs are in NP-complete and co-NP-complete respectively.

### 4 Decidability of general multiplication \( \bowtie \) over Tree Shares

In this section, we will prove the following results about \( S = (\mathbb{T}, \bowtie) \):

- **Theorem 2** (Complexity of \( S \)).
  1. The existential theory of \( S \) is decidable in PSPACE.
  2. The existential theory of \( S \) is NP-hard.
  3. The general first-order theory over \( S \) is undecidable.

The proof of Theorem 2 largely rests on the identical conclusions for the key subtheory \( S^+ \equiv (\mathbb{T}^+, \bowtie) \), where \( \mathbb{T}^+ \equiv \mathbb{T} \setminus \{0\} \) has removed the “zero element” \( 0 \) from \( \mathbb{T} \):

- **Lemma 3** (Complexity of \( S^+ \)).
  1. The existential theory of \( S^+ \) is decidable in PSPACE.
  2. The existential theory of \( S^+ \) is NP-hard.
  3. The general first-order theory over \( S^+ \) is undecidable.

We will prove Lemma 3 shortly, but first let us use it to polish off Theorem 2:

**Proof of Theorem 2.** We take each part in turn as follows:

1. Represent the set of variables \( V = \{x_1, \ldots, x_n\} \) in a given formula \( F \) of \( S \) as a \( n \)-length bitvector. We can enumerate through all possibilities \( P_1, \ldots, P_{2^n} \) for this vector using linear space and binary addition. For each possibility \( P_j \), variable \( x_i \)'s bit is 0 to indicate that \( x_i \) must be \( 0 \) and 1 when \( x_i \) must be non-\( 0 \). For each \( x_k \) that is marked as \( 0 \), we substitute \( 0 \) for \( x_k \) in \( F \) to reach \( F_j \) and simplify using the rules

\[
\begin{align*}
\pi_1 \bowtie 0 = \pi_2 & \quad \pi_1 = \pi_2 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_2 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_1 & = 0 \lor \pi_2 = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_2 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_1 & = 0 \land \pi_2 = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_2 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_1 & = 0 \lor \pi_2 = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_2 \\
\pi_2 & = 0 \\
\pi_1 \bowtie \pi_2 & = \pi_1 \\
\pi_1 & = 0 \land \pi_2 = 0 
\end{align*}
\]

We can then just check to make sure that the resulting “fresh” (in)equalities are consistent with the current value of the bitvector \( P_j \). If not, we have reached a contradiction and can proceed to the next bitvector \( P_{j+1} \). If so, then after removing the trivial equalities (e.g. \( 0 = 0 \)) from \( F_j \) we are left with an equivalent formula \( F^+_j \) which is in \( S^+ \), so by Lemma 3.1 we can check if \( F_j \) is satisfiable in PSPACE. If so, we know that \( F_j \) is satisfiable, and thus that \( F \) is satisfiable. If not, we proceed to the next bitvector \( P_{j+1} \); if all \( F_j \) are unsatisfiable then \( F \) is unsatisfiable.
2. By Lemma 3.2 it is sufficient to reduce a formula \( F^+ \) in \( \mathcal{S}^+ \) to \( \mathcal{S} \). Let \( V \) be the set of variables in \( F^+ \) and define \( F \triangleq F^+ \land \left( \bigwedge_{x \in V} x \neq \circ \right) \); note that we construct \( F \) in linear time from \( |F^+| \). \( F \) is satisfiable in \( \mathcal{S} \) if and only if \( F^+ \) is satisfiable in \( \mathcal{S}^+ \), so we are done.
3. Any extension of an undecidable theory is also undecidable; by Lemma 3.3 we are done.

\[ \triangle \]

### 4.1 Word equations

To prove Lemma 3 we will show that \( \mathcal{S}^+ \) is isomorphic to the theory of word equations. Let us recall this theory. Let \( A = \{a_1, a_2, \ldots\} \) be a finite set of letters and \( \cdot \) be a concatenation operator that combines letters into words. Let \( A^* \) be the Kleene closure of \( A \) using \( \cdot \). We define a model for the alphabet \( A \), written \( \mathcal{W}_A \) as the pair \( (A^*, \cdot) \). Now let \( V = \{v_1, v_2, \ldots\} \) be a finite set of variables, and \( w \in \mathcal{W} \triangleq (A \cup V)^* \) a finitely generated word that includes both letters and variables. We extend a word context \( \rho : V \rightarrow A^* \) to the domain \( A \cup V \) by mapping constants to themselves, and further to the domain \( \mathcal{W} \) by replacing each letter within a word with its value in \( \rho \). A word equation \( E_{\mathcal{W}} \) is a pair of words \( (w_1, w_2) \in \mathcal{W} \times \mathcal{W} \). We say that \( \rho \) is a solution of \( E_{\mathcal{W}} \) if \( \rho(w_1) = \rho(w_2) \).

The satisfiability of word equation asks whether a word equation \( E_{\mathcal{W}} \) has a solution \( \rho \), denoted \( \text{SAT}_{\mathcal{W}}(E_{\mathcal{W}}) \). Makanin proposed a complete treatment to this problem in a series of papers \([20, 21, 22]\) but his method was highly intractable (quadruple-exponential non-deterministic time \([16]\)). Substantial research since has improved this bound, e.g. \([2, 13]\). The best known complexity bound for this problem is \( \text{PSpace} \) and \( \text{NP-hard} \) \([25, 26]\) and it is hypothesized to be \( \text{NP-complete} \). Importantly for our present result, the existential theory over word equations is known to be reducible to \( \text{SAT}_{\mathcal{W}} \) in polynomial time \([8]\). Finally, the first order theory over \( \mathcal{W} \) is known to be undecidable \([23, 18]\).

**Infinite alphabets.** To define our isomorphism from \( \mathbb{T}^+ \) to \( A^* \) it will be convenient if the alphabet \( A \) can be countably infinite. Accordingly, we must reduce word equations over an infinite alphabet to the standard finite case. Let \( \sigma : \mathcal{W} \rightarrow \mathcal{P}(A) \) be the function that extracts the set of letters from a word \( w \), e.g. \( \sigma(v_1a_1a_3v_2) = \{a_1, a_3\} \) and extend \( \sigma \) to \( \mathcal{W} \times \mathcal{W} \) by \( \sigma(w_1, w_2) = \sigma(w_1) \cup \sigma(w_2) \). Let \( \phi : \mathcal{W} \times \mathcal{P}(A) \rightarrow \mathcal{W} \) be the projection function that takes a word \( w \) and a set of letters \( B \subseteq A \) and removes all letters in \( w \) that are not in \( B \), e.g. \( \phi(v_1a_1a_3v_2, \{a_1, a_2\}) = v_1a_1v_2 \). It is not hard to prove that \( \phi \) with fixed \( B \) is homomorphism over \( \mathcal{W}_A \). Now we are ready to state and prove the extension to infinite alphabets:

\[ \triangleright \text{Lemma 4 (Infinite alphabet word equations).} \quad \text{Let } A \text{ be infinite and } E_{\mathcal{W}} = (w_1, w_2) \text{ a word equation over } A. \ E_{\mathcal{W}} \text{ is satisfiable in } \mathcal{W}_A \text{ iff } E_{\mathcal{W}} \text{ is satisfiable in } \mathcal{W}_{\phi(E_{\mathcal{W}})}. \]

**Proof.** \( \Leftarrow \) is trivial. Let \( \rho : V \rightarrow A^* \) be a solution of \( E_{\mathcal{W}} \) over \( A \) and \( \rho' = \lambda v. \phi(\rho(v), \sigma(E_{\mathcal{W}})) \). Notice \( \rho' \) preserves all the letters in \( E_{\mathcal{W}} \) and \( \rho(w_1) = \rho'(w_1) \) implies \( \rho'(w_1) = \rho'(w_2) \). Thus \( \rho' \) is a new solution of \( E_{\mathcal{W}} \) that only contains letters from \( \sigma(E_{\mathcal{W}}) \).

\[ \triangleright \]

### 4.2 Finding an infinite alphabet inside \( \mathbb{T}^+ \)

Since \( \times \) is a kind of multiplication operation, and the fundamental building blocks of \( (\mathbb{N}, \times) \) are prime numbers, it is natural to wonder whether there is an analogue on trees. There is:

\[ \triangleright \text{Definition 5 (Prime trees).} \quad \tau \in \mathbb{T}^+ \setminus \{\bullet\} \text{ is prime if } \forall \tau_1, \tau_2. \; \tau = \tau_1 \bowtie \tau_2 \Rightarrow (\tau_1 = \bullet \vee \tau_2 = \bullet). \]

Furthermore, let \( \text{Prime}(\tau) \) indicate \( \tau \) is prime and \( \mathbb{T}_p \) be the set of all prime trees.
Examples of tree primes are \( \text{\includegraphics[width=0.3\textwidth]{example_tree}} \). On the other hand, the tree \( \text{\includegraphics[width=0.3\textwidth]{example_not_tree}} \) is not prime since it can be factored as \( \text{\includegraphics[width=0.3\textwidth]{example_not_tree}} \). Prime trees have many nice properties:

**Lemma 6** (Properties of prime trees).

1. There are countably infinitely many prime trees.
2. Let \( \tau_1, \tau_1', \tau_2, \tau_2' \in \mathbb{T}_p \), \( \tau_1 \Implies \tau_2 = \tau_1' \Implies \tau_2' \) iff \( \tau_1 = \tau_1' \) and \( \tau_2 = \tau_2' \).
3. Given two prime tree sequences \( S_1 = \tau_1^1, \ldots, \tau_2^k \) and \( S_2 = \tau_2^1, \ldots, \tau_2^k \), \( S_1 = S_2 \) iff their \( \Implies \) products are equal: \( \bigwedge_{i=1}^{k} \tau_i^1 = \bigwedge_{i=1}^{k} \tau_i^2 \Leftrightarrow (k_1 = k_2 \land \bigwedge_{i=1}^{k} \tau_i^1 = \tau_i^2) \).

To prove Lemma 6 we must define the notation \( |\tau| \) to be the height of \( \tau \) (with \( |o| = |\bullet| = 0 \) and counting up from there). Given this notation it is simple to define the set of all trees up to height \( n \), written \( \mathbb{T}^n \). We will also need the following technical lemma which allows us to split an application of bowtie \( \tau_2 \Implies \tau_3 \) to children of \( \tau_2 

**Lemma 7** (Split for \( \Implies \)). Let \( \tau_1, \tau_1', \tau_2, \tau_2' \in \mathbb{T}^+ \) and \( \tau_1 = \tau_2 \Implies \tau_3 \land \tau_1 = \tau_1' \Implies \tau_3 \) then either (1) \( \tau_2 = \bullet \land \tau_1 = \tau_3 \) or (2) \( \exists \tau_1'', \tau_1'. \tau_2 = \tau_1'' \Implies \tau_3 \land \tau_1' = \tau_1' \Implies \tau_3 \).

**Proof.** The case \( \tau_2 = \bullet \) is trivial. Otherwise, there exists \( \tau_1'', \tau_1' \in \mathbb{T}^+ \) such that \( \tau_2 = \tau_1'' \Implies \tau_1' \). By definition of \( \Implies \), \( \tau_1 = \tau_2 \Implies \tau_3 \) is computed by replacing each leaf \( \bullet \) in \( \tau_2 \) with \( \tau_3 \), which is equivalent to replace each leaf \( \bullet \) in \( \tau_1'' \Implies \tau_1' \) with \( \tau_3 \). Thus, \( \tau_1'' = \tau_1' \Implies \tau_2' \Implies \tau_3 \).

**Proof of Lemma 6.**

1. We construct an infinite sequence \( S \) of prime trees: let \( p_1 \triangleq \text{\includegraphics[width=0.2\textwidth]{example_tree}}, p_j \triangleq \text{\includegraphics[width=0.2\textwidth]{example_tree}}, i.e. \( S \triangleq \left\{ \begin{array}{c} \text{\includegraphics[width=0.2\textwidth]{example_tree}}, \text{\includegraphics[width=0.2\textwidth]{example_tree}}, \text{\includegraphics[width=0.2\textwidth]{example_tree}}, \ldots \end{array} \right\} \)

   It is immediate that \( p_1 \) is prime. To prove that \( p_i \) is prime for \( i > 1 \), we proceed as follows. Suppose \( p_i = \tau_1 \Implies \tau_2 \) and neither \( \tau_1 \) nor \( \tau_2 \) is \( \bullet \). The right subtree of each \( p_i \) is just \( \bullet \) and by the definition of \( \Implies \) must contain a copy of \( \tau_2 \), i.e. \( \tau_2 = \bullet \), so we have a contradiction and \( p_i \) is prime.

2. We prove by induction on the height of \( \tau_1, \tau_1' \). The base case \( \mathbb{T}^0 \) is easy to verify. Assume it holds for \( \mathbb{T}^k \) and \( \tau_1, \tau_1' \in \mathbb{T}^{k+1} \). Let \( \tau_1 = \tau_1'' \Implies \tau_2 = \tau_1' \Implies \tau_2' \). Then by Lemma 7, we derive \( \tau_1'' \Implies \tau_2 = \tau_1' \Implies \tau_2' \). By our induction hypothesis, \( \tau_1 = \tau_1' \). Consequently, \( \tau_1 = \tau_1' \).

3. This is a simple generalization of property 2.

Of course the real fun with prime numbers is the the unique factorization theorem. Since \( \Implies \) is not commutative we get a stronger version of the traditional theorem:

**Lemma 8** (Unique representation of \( S^+ \)). For each \( \tau \in \mathbb{T}^+ \setminus \{ \bullet \} \), there exists a unique sequence \( \tau_1, \ldots, \tau_n \in \mathbb{T}_p \) such that \( \tau = \Implies_{i=1}^{n} \tau_i \). Each \( \tau_i \) is called a prime factor of \( \tau \).
We are ready to make the central connection needed for Lemma 3:

**Corollary 9 (Basis of \( S^+ \)).** \( T_p \cup \{\bullet\} \) is a basis of \( S^+ \), i.e. the closure of \( T_p \) over \( \bowtie \) together with \( \bullet \) is \( \mathbb{T}^+ \). Furthermore, it is the smallest basis: if \( B \) is a basis of \( S^+ \) then \( T_p \cup \{\bullet\} \subseteq B \).

Accordingly, we will use \( T_p \) as our “infinite alphabet” in our isomorphism.

### 4.3 Connecting Tree Shares to Word Equations

We are ready to make the central connection needed for Lemma 3:

**Lemma 10.** \((\mathbb{T}^+, \bowtie)\) is isomorphic to \((\mathbb{T}_p^+, \star)\)

**Proof.** Let \( f : \mathbb{T}^+ \to \mathbb{T}_p^+ \) be defined as follows. First, map the identity element \( \bullet \) to the empty word \( \epsilon \) and then for each prime tree \( \tau_p \in \mathbb{T}^+ \), map \( \tau_p \) to itself. Finally, for each composite \( \tau \in \mathbb{T}^+ \), map \( \tau \) to exactly the concatenation of its (unique) prime factors.

We now wish to prove that for any \( \tau_1 \) and \( \tau_2 \), \( f(\tau_1 \bowtie \tau_2) = f(\tau_1) \cdot f(\tau_2) \). Let us consider the easy cases first. If \( \tau_1 = \bullet \) then \( f(\tau_1 \bowtie \tau_2) = f(\tau_2) = \epsilon \cdot f(\tau_2) = f(\tau_1) \cdot f(\tau_2) \). The situation is symmetric when \( \tau_2 = \bullet \). Now let us consider the case when neither \( \tau_1 \) nor \( \tau_2 \) is \( \bullet \). Let \( p_1, \ldots, p_i \) be the unique prime factors of \( \tau_1 \) and \( p'_1, \ldots, p'_j \) be the unique prime factors of \( \tau_2 \). By Lemma 8, \( p_1, \ldots, p_i, p'_1, \ldots, p'_j \) are exactly the unique prime factors of \( \tau_1 \bowtie \tau_2 \), so:

\[
f(\tau_1 \bowtie \tau_2) = f(p_1 \bowtie \cdots \bowtie p_i \bowtie p'_1 \bowtie \cdots \bowtie p'_j) = p_1 \cdots p_i \cdot p'_1 \cdots p'_j = (p_1 \cdots p_i) \cdot (p'_1 \cdots p'_j) = f(\tau_1) \cdot f(\tau_2)
\]

To prove \( f \) is surjective, let \( w \in \mathbb{T}_p^+ \) be the concatenation of primes \( p_1 \cdots p_i \); then by the definition \( f(p_1 \bowtie \cdots \bowtie p_i) = w \). To prove \( f \) is injective, suppose \( f(\tau_1) = f(\tau_2) \). Let \( p_1, \ldots, p_i \) be the prime factors of \( \tau_1 \) and \( p'_1, \ldots, p'_j \) be the prime factors of \( \tau_2 \). Accordingly we know that \( p_1 \cdots p_i = p'_1 \cdots p'_j \), and since equality over words can only occur if the words are the same length and have the same letters, we know \( i = j \) and \( p_k = p'_k \) for all \( k \).

**Corollary 11 (Equations over Positive Tree Shares are Word Equations).** Equations \( E_{T^+} \) over \((\mathbb{T}^+, \bowtie)\) contain both tree constants \( \tau \in \mathbb{T}^+ \) and variables \( v \in V \); we can map these to word equations \( E_W \) over \((\mathbb{T}_p^+, \star)\) by mapping variables to themselves, constants to the concatenation of their prime factors, and multiplication \( \bowtie \) to concatenation \( \cdot \). The resulting system is equivalent, i.e. if \( \rho : V \to \mathbb{T}^+ \) satisfies \( E_{T^+} \), then \( f \circ \rho \) satisfies \( E_W \), where \( \circ \) in this case means functional composition and \( f \) is the isomorphism constructed in Lemma 10.

We are now ready to start tackling Lemma 3. We start with the simplest:

**Proof of Lemma 3.3.** As previously mentioned, the first order theory over word equations is known to be undecidable [23, 18]. By Corollary 10 we know that this theory is isomorphic to the first order theory over tree shares with \( \bowtie \), which accordingly must be undecidable.

To show Lemma 1 we need to know that tree factorization can be done within PSPACE. In fact we can do much better:

**Lemma 12 (Factorization).** Factoring an arbitrary positive tree share \( \tau \) is in \( P \).
Proof. Let $S(\tau)$ be the set of all sub-trees of $\tau$ and $S_n(\tau) \subset S(\tau)$ be the set of all sub-trees of $\tau$ with height exactly $n$. $S(\tau)$ can be computed recursively: $S(\varepsilon) = \{\varepsilon\}$, $S(\bullet) = \{\bullet\}$. $S(\tau_1 \tau_2) = S(\tau_1) \cup S(\tau_2) \cup \{\tau_1 \tau_2\}$. If $\tau = \tau_1 \bowtie \tau_2 \{\{\tau_1, \tau_2\} \subset T^+ \{\bullet\}\}$, then there exists $n \in \mathbb{N}$ such that $S_n(\tau) = \{\tau_2\}$, that is, $S_{[\tau]}$ is exactly the singleton set $\{\tau_2\}$.

Thus we can find all the prime factors of $\tau$ as follows: let $i \in \mathbb{N}$ be the smallest number such that $S_i(\tau)$ is the singleton set $\{\tau_1\}$ for some $\tau_1 \in T$ (note that $i$ must be larger than 0 since $S_0(\tau) = \{\varepsilon, \bullet\}$). If $i = |\tau|$ then $\tau$ itself is a prime, otherwise, we replace all sub-trees $\tau_1$ of $\tau$ with $\bullet$ and call the new tree $\tau'$. If all the “old” $\bullet$ leaves of $\tau$ are replaced and $\tau'$ is in canonical form then $\tau = \tau' \bowtie \tau_1$, $\tau_1$ is a prime factor of $\tau$, and we can repeat the process with $\tau'$ to find the next prime factor. Otherwise, we consider the next singleton set $S_j(\tau)$.

To verify the complexity, assume $\tau$ has $n$ leaves then $|S(\tau, k)| \leq n!$ as there is at least $k + 1$ leaves in a tree of height $k$. Thus the number of sub-trees from height $1$ to $n$ is at most $\Sigma_{i=1}^n \frac{n!}{i!} \leq n^2$. We need to reset the process at most $n - 1$ times whenever a prime factor is found, so the time complexity of factorization is polynomial to the size of the input tree. ◀

Tree factorization is fundamentally simpler than integer factorization since the representation of a tree already contains the descriptions of all of its tree factors. In contrast, the connection between the representation of a number and the representation of its prime factors is vague: e.g. among the 24 factors of 74,611,647 are 333 (which does not appear at all in the representation of the original) and 8,290,183 (which only shares a single 1 with the original).

Proof of Lemma 3.1. We take the tree shares and factor them using Lemma 12 and then construct the isomorphic system of word equations using the calculated prime factors as the alphabet using Corollary 11. As mentioned, the best known complexity bound for the existential word equation problem is PSPACE [25, 26].

For Lemma 3.2 we need one final fact:

Lemma 13 (Existence of small primes). For any $n$ (represented in unary) we can find a length-$n$ sequence of tree primes $S$ in polynomial time of $n$.

Proof. Consider the sequence $S$ from Lemma 6: the description of $p_i$ is only a constant size larger than the description of $p_{i-1}$ so the description of $S$ is quadratic in $n$.

Proof of Lemma 3.2. Suppose we have an arbitrary problem $Q$ in NP. We can reduce $Q$ to word equations in polynomial time [25, 26]. We then use Lemma 13 to construct a set of primes the size of the number of alphabet letters that appear in the equations and map each letter in the word alphabet to a distinct prime, creating a set of word equations over $T^+_p$. Since the representation of the constants does not affect the computational properties of the theory, we can conclude that $T^+_p$ is NP-hard.

5 A reduction to Tree Automatic Structures

Theorem 2 shows that the first order theory (FO) over $S$ is undecidable, so of course any extension of $S$—e.g. with $(\cap, \cup, \Box)$—also has an undecidable FO. However, if we restrict the form of equations to be $p_1 \bowtie \tau = p_2$ where $\tau \in T$, then the FO of $S$ is decidable because the relation is tree-automatic. This type of restriction is inspired by Jain et al.’s concept of semi-automatic structures [14], in which relations are restricted so that all input arguments are fixed constants except for one argument which is a variable. As a result, certain relations become automatic, e.g. multiplication in unary language.
Let $\tau_\times \colon T \to T$ and $\tau_\triangleright \colon T \to T$ be the left and right restricted form of $\tau$ with respect to $\tau \in T$, i.e., $\tau_\times(\tau_1) \triangleq \tau \times \tau_1$ and $\tau_\triangleright(\tau_1) \triangleq \tau_1 \times \tau$. We will show $\mathcal{T} \triangleq (\{\tau, \triangleright, \times, \times, \times\})$, where $\triangleright$ denotes the family of all right-restricted forms of $\tau$ indexed by tree constants, is tree-automatic by constructing bottom-up tree automata that recognize the domain $T$ and the relations in $\mathcal{T}$:

**Theorem 14 (Decidability of $(T, \cap, \cup, \sqcup, \triangleright)$).** $\mathcal{T}$ is tree-automatic. As a result, the first-order theory of $\mathcal{T}$ is decidable.

As indicated by equation (8) (from §2), one use for $\triangleright$ is to split/join ownership of maps-to predicates in separation logic. Here the splitting/joining occurs on the right-hand side of the $\triangleright$. Moreover, many functions need to divide their ownership only a finite number of times before e.g. calling other functions or indeed themselves recursively. This is because the program text of functions is finite. Accordingly, we believe that $\mathcal{T}$ is worthy of attention.

### 5.1 Tree automatic structures

Before proving Theorem 14, we recall the definition of tree automatic structures [27, 15].

**Tree automaton.** A bottom up tree automaton is a 4-tuple $\mathcal{A} = (Q, F, Q_f, \Delta)$ where $Q$ is the set of states, $F$ is the ranked alphabet, $Q_f \subseteq Q$ is the set of accepting states and $\Delta$ is the set of transitions $f(q_1(v_1), ..., q_n(v_n)) \rightarrow q_{n+1}(f(v_1, ..., v_n))$ where $f \in F$ is an n-ary letter, $v_i \in V$ is a variable and $q_j \in Q$. Let $T$ be a tree constructed from $F$ then $\mathcal{A}$ runs on $T$ by applying $\Delta$ at each leaf of $T$ spontaneously and proceeding upward. $\mathcal{A}$ accepts $T$ if the state associated with the root of $T$ is in $Q_f$. We will put a temporary superscript number ($n$) above each letter in the definition of $F$ to indicate its arity, i.e. $f^{(n)}$ means $f$ is n-ary. For instance, the ranked alphabet for the tree domain $\mathcal{T}$ is $F = \{\circ^{(0)}, \bullet^{(0)}, \text{Node}^{(2)}\}$.

**Tree automatic structures.** Let $\mathcal{R} \subseteq D^k$ be a k-ary tree relation on some tree domain $\mathcal{D}$, its convolution set is constructed by overlapping all $k$ trees of each element in $\mathcal{R}$ to form a single tree. As trees can have different shapes and thus their convolution contains “holes”, we fill the holes with a special nullary character $\circ^{(0)}$ that is not in $F$, e.g. $(\text{Node}(a_1, a_2), b, \text{Node}(c_1, c_2)) \mapsto [\text{Node}, b, \text{Node}][(\circ^{(0)}, \circ^{(0)}, \circ^{(0)})]$. As a result, if $F_\mathcal{D}$ is the ranked alphabet of $\mathcal{D}$ then $(F_\mathcal{D} \cup \{\circ\})^k$ is the ranked alphabet for $D^k$ and the arity of each convolution letter is the maximal arity among its letter components. $\mathcal{R}$ is tree-automatic if its convolution set is accepted by a tree automaton. Generally, a structure $(\mathcal{D}, \mathcal{R}_1, \ldots, \mathcal{R}_n)$ is tree-automatic if its domain $\mathcal{D}$ and each of its relations $\mathcal{R}_i$ are tree-automatic. We restate a well-known decidability result for tree automatic structures:

**Lemma 15 ([5, 6]).** The first order theory of tree automatic structures is decidable.

### 5.2 Tree automata construction for $\mathcal{T}$

**Construction of $\mathcal{A}^\mathcal{T}$.** For the domain $T$, it suffices to check the canonical form. Let $\mathcal{A}^\mathcal{T} = (Q^\mathcal{T}, F^\mathcal{T}, Q_f^\mathcal{T}, \Delta^\mathcal{T})$ such that $Q^\mathcal{T} = \{q, q_\circ, q_\bullet\}$, $F^\mathcal{T} = F$, $Q_f^\mathcal{T} = \{q\}$ and the transition relation $\Delta^\mathcal{T}$ contains the following:

- $\circ \mapsto q(\circ)$ and $\bullet \mapsto q(\bullet)$
- $\text{Node}(q_1(v_1), q_2(v_2)) \mapsto q(\text{Node}(v_1, v_2))$ where $(q_1, q_2) \in \{(q_\circ, q_\circ), (q_\bullet, q_\bullet), (q, q)\}$

**Construction of $\mathcal{A}^\mathcal{□}$.** From now on, we assume that the input trees are in domain $T$, which will make other constructions more pleasant. The automaton $\mathcal{A}^\mathcal{□}$ for complement $\square$ is easy: we need to verify the opposite values leaf-wise between two trees. To be precise, let $(Q^\mathcal{□}, F^\mathcal{□}, Q_f^\mathcal{□}, \Delta^\mathcal{□})$ such that $Q^\mathcal{□} = Q_f^\mathcal{□} = \{q\}$, $F^\mathcal{□} = (F \cup \{\circ\})^2$ and $\Delta^\mathcal{□}$ is defined as:

\[ \Delta^\mathcal{□}(q)(q_1(v_1), q_2(v_2)) = q(\text{Node}(\neg v_1, \neg v_2)) \]
WLOG, we assume that both similar shape (\(S\) that recognizes \(\tau\)) and \(\phi\) are automatic. As we are about to prove, the other relation \(\phi\) is not tree-automatic in this representation. We leave an open question whether there exists a representation in which both \(\phi\) and \(\phi\) are automatic, i.e., whether \(\phi\) is semi-automatic.
Decidability and Complexity of Tree Share Formulas

Proposition 16 (τ∞). In the current representation of tree shares, there exists infinitely many τ such that τ∞ is not tree-automatic.

First, we recall the Pumping Lemma for tree automata:

Definition 17 (Term, context and substitution [9]). Let \( \mathcal{A} = (Q, F, Q_f, \Delta) \) be a tree automaton and \( V \) the set of variables. We define \( T(F, V) \) the set of all tree terms derived from \( F \cup V \) and \( T(F, \emptyset) \) is the set of ground terms. A term \( t \) is linear if each variable appears at most once in \( t \). A context \( C \) is a linear term of \( T(F, V) \) and \( \mathcal{C}(F) \) denotes the set of all contexts with single variable. A context is trivial if it is reduced to a variable. Let \( C[t] \) be the substitution of \( C \in \mathcal{C}(F) \) by replacing the variable in \( C \) with the term \( t \). We define \( C^0[t] = v \) where \( v \) is the variable in \( C \), \( C^1[t] = C[t] \) and \( C^{n+1}[t] = \mathcal{C}[C^n[t]] \).\[1\]

Lemma 18 (Pumping Lemma for Tree Automata [9]). Let \( \mathcal{L} \) be the set of all ground terms recognizable by a tree automaton. There is a constant \( k \in \mathbb{N}^+ \) satisfying the following condition: for all ground term \( t \in \mathcal{L} \) and \( |t| > k \), there exists a context \( C \in \mathcal{C}(F) \), a non-trivial context \( C' \in \mathcal{C}(F) \) and a ground term \( t' \) such that \( t = \mathcal{C}[C'[t']] \) and \( \forall n \in \mathbb{N} \). \( \mathcal{C}[C^n[t']] \in \mathcal{L} \).

Proof of Proposition 16. Let \( \tau \tau^\infty \) where \( \tau = \begin{array}{ccc} & \circ & \\ \circ & - & \circ \\ \circ & \circ & \circ \end{array} \). Notice that for an input tree \( \tau \in \mathbb{T}^+ \), the result tree is \( \begin{array}{ccc} & \circ & \\ \circ & - & \circ \\ \circ & \circ & \circ \end{array} \) in which the left and right sub-tree are identical. If \( \tau \tau^\infty \) is automatic then it satisfies the Pumping Lemma. However, the Pumping Lemma only allows us to pump either the left or the right sub-tree and thus they will be different after pumping, which is a contradiction. Now consider the following sequence: \( \tau_1 = \begin{array}{ccc} & \circ & \\ \circ & - & \circ \\ \circ & \circ & \circ \end{array} \), \( \tau_{n+1} = \begin{array}{ccc} & \circ & \\ \circ & - & \circ \\ \circ & \circ & \circ \end{array} \) then each of the \( \tau \tau^\infty \) is not automatic.\[1\]

6 Future Work

We identify several directions for future research. One obvious question is the decidability of the existential theory of the tree share model with \( \infty, \sqcup, \sqcap, \sqsubseteq, \) and \( \square \), where \( \infty \) is unrestricted. Our connection to word equations does not seem to provide an answer to decidability of this problem. Another question is the computational complexity of adding more general constants (i.e., other than \( \circ \) and \( \bullet \)) to the first order theory of \( \mathcal{M} \). Although such constants are in some way “definable” (e.g., in first-order logic over the vocabulary with \( \circ \) and \( \bullet \)), the definition in general requires a formula of superpolynomial size. For this reason, connections to CABA do not immediately yield the same computational complexity. A final direction for future work is to investigate the integration of \( \infty \square \) into a practical program logic.

7 Conclusion

We have demonstrated the decidability and complexity of a first-order theory over the Boolean logic of tree shares by pinpointing the connection to countable atomless Boolean algebras. We have provided the first serious look at the complexity of the tree multiplication \( \infty \) operator and by way of an isomorphism to word equations prove that the existential theory is in \( \text{PSPACE} \) while the general first-order theory is undecidable. We have identified a restricted version of \( \infty \) that takes constants on the right-hand side \( \infty_{\mathbb{N}} \) and have proven that the system \( (\mathcal{T}, \sqcup, \sqcap, \square, \infty_{\mathbb{N}}) \) has a decidable first-order theory via an embedding to tree-automatic structures.
References


