Decidability and Complexity of Tree Share Formulas

Xuan Bach Le[1], Aquinas Hobor[2], and Anthony W. Lin[3]

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 $\bar{\square}$ 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 ($\sqcup$, $\sqcap$, $\bar{\square}$) and the restricted multiplication operation ($\bowtie$ with constants on the right hand side) is decidable via an embedding to tree-automatic structures.

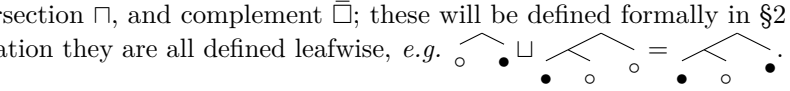
1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, F.4.3 Formal Languages

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

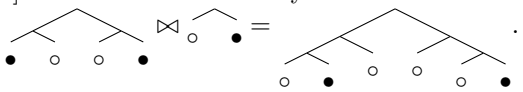
Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.

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: $\tau \triangleq \circ \mid \bullet \mid \overset{\tau}{\frown}\!\!\underset{\tau}{}$, 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: $\overset{\circ}{\frown}\!\!\underset{\bullet}{}$ and $\overset{\bullet}{\frown}\!\!\underset{\circ}{}$, and four "quarter" shares, beginning with $\overset{\bullet}{\diagdown}\!\!\underset{\circ}{}\!\!\diagup\!\!\overset{}{\circ}$. 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 $\bar{\square}$; these will be defined formally in §2 but to a first approximation they are all defined leafwise, *e.g.*  .

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 $\tau_1 \bowtie \tau_2$ [10]. The basic idea is that you take each $\bullet$ leaf in τ_1 and replace it with a full copy of τ_2, *e.g.*  .

Dockins *et al.* showed that $\bowtie$ could be used to split any nonempty tree τ into two nonempty trees that joined together to equal the original since $\forall \tau.\ \tau = (\tau \bowtie \bullet\!\circ) \sqcup (\tau \bowtie \circ\!\bullet)$. More generally, the $\bowtie$ operator can be used as a kind of "scoping" or "gluing" operator to combine different uses of tree shares together. Although $\bowtie$ 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.

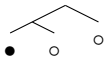
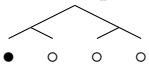
First (§3), we show that the tree share model $\mathcal{M} \triangleq (\sqcap, \sqcup, \bar{\square}, \circ, \bullet)$ is a Countable Atomless Boolean Algebra (CABA), which are known to be unique up to isomorphism [28]. The first-order theory over CABAs is known to be decidable and, in fact, complete for the class $\mathrm{STA}(*, 2^{cn}, n)$ 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 $\mathcal{M}$.

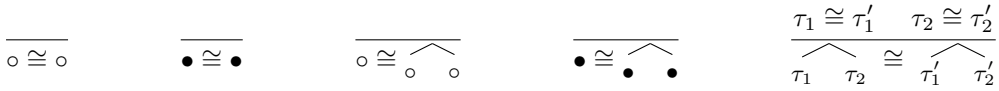
We then (§4) proceed to decision problems over the tree shares with the multiplication operator $\bowtie$. Our main result here is that the tree share model $\mathcal{S} \triangleq (\mathbb{T}, \bowtie)$ that only allows $\bowtie$ (*i.e.* but not $\sqcup$, $\sqcap$, and $\bar{\square}$) is—in a technical 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 $\mathcal{S}$ is decidable in polynomial space but NP-hard, whereas the first-order theory over $\mathcal{S}$ is undecidable.

Finally (§5), we consider restrictions on $\bowtie$ 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 $\bowtie_\square$, *i.e.* $\bowtie_\tau (\tau') \triangleq \tau' \bowtie \tau$. We prove that the combined theory of $\mathcal{T} \triangleq (\mathbb{T}, \sqcap, \sqcup, \bar{\square}, \bowtie_\square)$ 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 $\bowtie_\square$. 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 $\mathbb{T}$ of Dockins *et al.* [10]

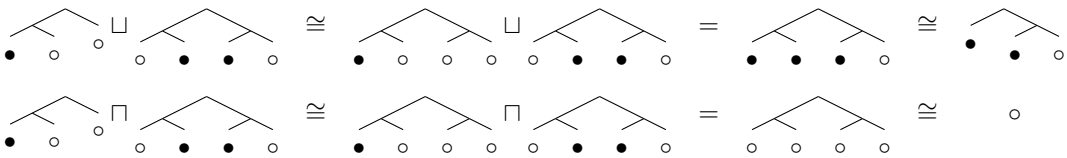
Here we summarize additional details of tree shares and their associated theory from [10].

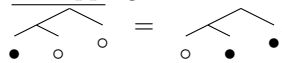
Canonical forms. In the first paragraph of §1 we presented the first quarter share as  instead of *e.g.*  . This is deliberate: the second choice is not a valid share because the tree is not in *canonical form*. A tree is in canonical form when it is in its most compact representation under the inductively-defined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

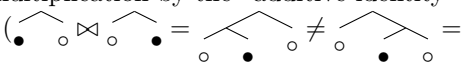


As we will see, operations on tree shares sometimes need to fold/unfold trees to/from canonical form, a practice we will indicate using the symbol $\cong$. Canonicality is needed to guarantee some of the algebraic properties of tree shares; managing it requires a little care in the proofs but does not 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 $\circ \sqcup \tau = \tau \sqcup \circ = \tau$, $\bullet \sqcup \tau = \tau \sqcup \bullet = \bullet$, $\circ \sqcap \tau = \tau \sqcap \circ = \circ$, and $\bullet \sqcap \tau = \tau \sqcap \bullet = \tau$; and finally refold back into canonical form, *e.g.*:



Complementation is simpler, since flipping leaves between $\circ$ and $\bullet$ does not affect whether a tree is in canonical form, *e.g.*:  . Using these definitions we get all of the usual properties for Boolean algebras, *e.g.* $\overline{\tau_1 \sqcap \tau_2} = \overline{\tau_1} \sqcup \overline{\tau_2}$. 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 $\bullet$, and is injective for non-$\circ$ elements, *i.e.* $\mathcal{S}^+ \triangleq (\mathbb{T} \setminus \{\circ\}, \bowtie)$ forms a cancellative monoid. Somewhat unsurprisingly, multiplication by the "additive identity" $\circ$ reduces to $\circ$. Unfortunately, $\bowtie$ is not commutative (), although we do enjoy a distributive property over $\sqcup$ and $\sqcap$ on the right hand side. Accordingly:

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

$$\text{Associativity}: \quad \tau_1 \bowtie (\tau_2 \bowtie \tau_3) = (\tau_1 \bowtie \tau_2) \bowtie \tau_3 \tag{1}$$

$$\text{Identity element}: \quad \tau \bowtie \bullet = \bullet \bowtie \tau = \tau \tag{2}$$

$$\text{Zero element}: \quad \tau \bowtie \circ = \circ \bowtie \tau = \circ \tag{3}$$

$$\text{Left cancellation}: \quad \tau \neq \circ \;\Rightarrow\; \tau \bowtie \tau_1 = \tau \bowtie \tau_2 \;\Rightarrow\; \tau_1 = \tau_2 \tag{4}$$

$$\text{Right cancellation}: \quad \tau \neq \circ \;\Rightarrow\; \tau_1 \bowtie \tau = \tau_2 \bowtie \tau \;\Rightarrow\; \tau_1 = \tau_2 \tag{5}$$

$$\tag{6}$$

Typical use of $\mathbb{T}$ 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 $x \overset{\pi}{\mapsto} y$ 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 π here because we often use share variables rather than concrete trees τ.

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: $\tau_1 \oplus \tau_2 = \tau_3 \triangleq \tau_1 \sqcup \tau_2 = \tau_3 \wedge \tau_1 \sqcap \tau_2 = \circ$. In other words, the join relation is a kind of disjoint union; it is partial because *e.g.* $\bullet \oplus \bullet$ 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 $*$ and the underlying Boolean operators as $x \overset{\pi_1}{\mapsto} y * x \overset{\pi_2}{\mapsto} z \dashv\vdash y = z \wedge x \overset{\pi_1 \oplus \pi_2}{\longmapsto} y$ (using $\dashv\vdash$ for bientailment).

It is common that we want to "split" a share π into sub-shares π_1, π_2 so that the permission can be transferred. This can be done within $\mathcal{M} \triangleq (\sqcap, \sqcup, \overline{\square}, \circ, \bullet)$ using the following rule:

$$\frac{\pi \neq \circ}{x \overset{\pi}{\mapsto} v \quad \dashv\vdash \quad \exists \pi_1, \pi_2.\ (x \overset{\pi_1}{\mapsto} v * x \overset{\pi_2}{\mapsto} v) \wedge \pi_1 \oplus \pi_2 = \pi \wedge \pi_1 \neq \circ \wedge \pi_2 \neq \circ} \ \textsc{SplitJoin}$$

This rule has some drawbacks. The most obvious is the lengthy size of the entailment's consequent, even though we only split π 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 π_1 and π_2 are—that is, π_1 and π_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:

$$\frac{\tau_1 \oplus \ldots \oplus \tau_n = \tau \qquad \pi \neq \circ \qquad \bigwedge_{i=1}^{n} \tau_i \neq \circ}{x \overset{\pi \bowtie \tau}{\longmapsto} v \quad \dashv\vdash \quad x \overset{\pi \bowtie \tau_1}{\longmapsto} v * \ldots * x \overset{\pi \bowtie \tau_n}{\longmapsto} v} \ \textsc{SplitJoin} \bowtie \tag{7}$$

3 Tree Shares are a model for Countable Atomless Boolean Algebras

In this section, we pinpoint the fact that $\mathcal{M} = (\sqcap, \sqcup, \overline{\square}, \circ, \bullet)$ is a model for Countable Atomless Boolean Algebra (CABA). Let $\mathcal{B} = (\cap, \cup, \overline{\square}, \mathbf{0}, \mathbf{1})$ be a Boolean Algebra (BA), we define a partial order $\subseteq$ on $\mathcal{B}$ ($\sqsubseteq$ for $\mathcal{M}$, resp.): $a_1 \subseteq a_2 \triangleq a_1 \cap \overline{a_2} = \mathbf{0}$ and $a_1 \subset a_2 \triangleq a_1 \subseteq a_2 \wedge a_2 \not\subseteq a_1$. $\mathcal{B}$ is *atomless* if $\forall a.\ \mathbf{0} \subset a \Rightarrow \exists a'.\ \mathbf{0} \subset a' \subset a$. $\mathcal{B}$ is *countable* if its domain is countable. Dockins *et al.* [10] proved that $\mathcal{M}$ is a model for BA where $\mathbf{0} \triangleq \circ$, $\mathbf{1} \triangleq \bullet$, $\sqcup \triangleq \cup$, $\sqcap \triangleq \cap$. The atomless property can be derived from the Infinite Splitability property of tree shares [19]: let $a \sqsupset \circ$ and $a_1, a_2 \neq \circ$ such that $a_1 \oplus a_2 = a$. This implies $a_1 \sqcup a_2 = a \wedge a_1 \sqcap a_2 = \circ$. By Stone's representation theorem each BA is isomorphic to a BA of powerset, thus $a_1 \sqcup a_2 = a$ implies $a_1 \subseteq a$ and $a_2 \subseteq a$. Suppose that $a_1 = a$ then $a_2 \sqcap a = \mathbf{0}$ which is a contradiction because $\mathbf{0} \sqsubset a_2 \sqsubset a$. As a result, $a_1 \neq a$ and thus $a_1 \sqsubset a$. The proof that $\mathbb{T}$ is countable is achieved by enumerating $\mathbb{T}$ in the ascending order of tree height $|\tau|$ using the following total strict order $\prec$:

$$\frac{}{\circ \prec \bullet} \qquad \frac{|\tau_1| < |\tau_2|}{\tau_1 \prec \tau_2} \qquad \frac{|\overset{\tau_1}{\frown}\ \overset{\tau_1'}{\frown}| = |\overset{\tau_2}{\frown}\ \overset{\tau_2'}{\frown}| \quad \tau_1 \prec \tau_2}{\overset{\tau_1}{\frown}\ \overset{\tau_1'}{\frown} \prec \overset{\tau_2}{\frown}\ \overset{\tau_2'}{\frown}} \qquad \frac{|\overset{\tau}{\frown}\ \overset{\tau_1}{\frown}| = |\overset{\tau}{\frown}\ \overset{\tau_2}{\frown}| \quad \tau_1 \prec \tau_2}{\overset{\tau}{\frown}\ \overset{\tau_1}{\frown} \prec \overset{\tau}{\frown}\ \overset{\tau_2}{\frown}}$$

It is known that there is a unique model for CABA up to isomorphism [28], so we can reason about the complexity of $\mathcal{M}$ in terms of CABA. Let $\mathrm{STA}(f(n), g(n), h(n))$ 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 $\bigcup_{c<\omega} \mathrm{STA}(*, 2^{cn}, n)$, 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 $\mathcal{S} = (\mathbb{T}, \bowtie)$:

▶ **Theorem 2** (Complexity of $\mathcal{S}$)**.**

1. *The existential theory of $\mathcal{S}$ is decidable in PSPACE.*
2. *The existential theory of $\mathcal{S}$ is NP-hard.*
3. *The general first-order theory over $\mathcal{S}$ is undecidable.*

The proof of Theorem 2 largely rests on the identical conclusions for the key subtheory $\mathcal{S}^+ \triangleq (\mathbb{T}^+, \bowtie)$, where $\mathbb{T}^+ \triangleq \mathbb{T} \setminus \{\circ\}$ are the "positive trees" obtained by removing the "zero element" $\circ$ from $\mathbb{T}$:

▶ **Lemma 3** (Complexity of $\mathcal{S}^+$)**.**

1. *The existential theory of $\mathcal{S}^+$ is decidable in PSPACE.*
2. *The existential theory of $\mathcal{S}^+$ is NP-hard.*
3. *The general first-order theory over $\mathcal{S}^+$ 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 $\mathcal{S}$ 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 $\circ$ and 1 when x_i must be non-$\circ$. For each x_k that is marked as $\circ$, we substitute $\circ$ for x_k in F to reach F_j and simplify using the rules

$$\frac{\pi_1 \bowtie \circ = \pi_2}{\pi_2 = \circ} \qquad \frac{\circ \bowtie \pi_1 = \pi_2}{\pi_2 = \circ} \qquad \frac{\pi_1 \bowtie \pi_2 = \circ}{\pi_1 = \circ \vee \pi_2 = \circ}$$

$$\frac{\pi_1 \bowtie \circ \neq \pi_2}{\pi_2 \neq \circ} \qquad \frac{\circ \bowtie \pi_1 \neq \pi_2}{\pi_2 \neq \circ} \qquad \frac{\pi_1 \bowtie \pi_2 \neq \circ}{\pi_1 \neq \circ \wedge \pi_2 \neq \circ}$$

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.* $\circ = \circ$) from F_j we are left with an equivalent formula F_j^+ which is in $\mathcal{S}^+$, 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 $F \triangleq F^+ \wedge \left(\bigwedge_{x \in V} x \neq \circ \right)$; 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.

◀

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 $\bullet$ be a concatenation operator that combines letters into words. Let A^* be the Kleene closure of A using $\bullet$. We define a model for the alphabet A, written $\mathcal{W}_A$ as the pair $(A^*, \bullet)$. Now let $V = \{v_1, v_2, \ldots\}$ be a finite set of variables, and $w \in \mathbb{W} \triangleq (A \cup V)^*$ a finitely generated word that includes both letters and variables. We extend a *word context* $\rho : V \to A^*$ to the domain $A \cup V$ by mapping constants to themselves, and further to the domain $\mathbb{W}$ by replacing each letter within a word with its value in ρ. A *word equation* $E_{\mathbb{W}}$ is a pair of words $(w_1, w_2) \in \mathbb{W} \times \mathbb{W}$. We say that ρ is a solution of $E_{\mathbb{W}}$ if $\rho(w_1) = \rho(w_2)$.

The satisfiability of word equation asks whether a word equation $E_{\mathbb{W}}$ has a solution ρ, denoted $\mathbf{SAT}_{\mathbb{W}}(E_{\mathbb{W}})$. 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 PSPACE and NP-hard [25, 26] and it is hypothesized to be NP-complete. Importantly for our present result, the existential theory over word equations is known to be reducible to $\mathbf{SAT}_{\mathbb{W}}$ in polynomial time [8]. Finally, the first order theory over $\mathbb{W}$ 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 $\sigma : \mathbb{W} \to \mathcal{P}(A)$ be the function that extracts the set of letters from a word w, *e.g.* $\sigma(v_1 a_1 a_3 v_2) = \{a_1, a_3\}$ and extend σ to $\mathbb{W} \times \mathbb{W}$ by $\sigma(w_1, w_2) = \sigma(w_1) \cup \sigma(w_2)$. Let $\phi : \mathbb{W} \times \mathcal{P}(A) \to \mathbb{W}$ 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.* $\phi(v_1 a_1 a_3 v_2, \{a_1, a_2\}) = v_1 a_1 v_2$. It is not hard to prove that ϕ with fixed B is an homomorphism over $\mathcal{W}_A$. Now we are ready to state and prove the extension to infinite alphabets:

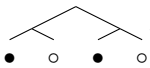
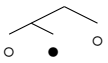
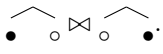
▶ **Lemma 4** (Infinite alphabet word equations). *Let A be infinite and $E_{\mathbb{W}} = (w_1, w_2)$ a word equation over A. $E_{\mathbb{W}}$ is satisfiable in $\mathcal{W}_A$ iff $E_{\mathbb{W}}$ is satisfiable in $\mathcal{W}_{\sigma(E_{\mathbb{W}})}$.*

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

4.2 Finding an infinite alphabet inside $\mathbb{T}^+$

Since $\bowtie$ 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:

▶ **Definition 5** (Prime trees). $\tau \in \mathbb{T}^+ \backslash \{\bullet\}$ is *prime* if $\forall \tau_1, \tau_2.\ \tau = \tau_1 \bowtie \tau_2 \Rightarrow (\tau_1 = \bullet \vee \tau_2 = \bullet)$. Furthermore, let $\mathsf{Prime}(\tau)$ indicate τ is prime and $\mathbb{T}_p$ be the set of all prime trees.

Examples of tree primes are  and  . On the other hand, the tree  is not prime since it can be factored as  . Prime trees have many nice properties:

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

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

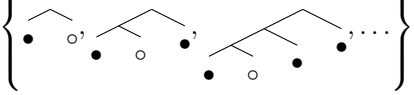
To prove Lemma 6 we must define the notation $|\tau|$ to be the height of τ (with $|\circ| = |\bullet| = 0$ and counting up from there). Given this notation it is simple to define the set of all trees up to height n, written $\mathbb{T}^n$. We will also need the following technical lemma which allows us to split an application of bowtie $\tau_2 \bowtie \tau_3$ to children of τ_2:

▶ **Lemma 7** (Split for $\bowtie$). *Let* $\tau_1, \tau_2, \tau_3, \tau_1^l, \tau_1^r \in \mathbb{T}^+$ *and* $\tau_1 = \tau_2 \bowtie \tau_3 \wedge \tau_1 =$  *then either (1)* $\tau_2 = \bullet \wedge \tau_1 = \tau_3$ *or (2)* $\exists \tau_2^l, \tau_2^r.\ \tau_2 =$  $\wedge\ \tau_1^l = \tau_2^l \bowtie \tau_3 \wedge \tau_1^r = \tau_2^r \bowtie \tau_3$.

Proof. The case $\tau_2 = \bullet$ is trivial. Otherwise, there exists $\tau_2^l, \tau_2^r \in \mathbb{T}$ such that $\tau_2 =$  . By definition of $\bowtie$, $\tau_1 = \tau_2 \bowtie \tau_3$ is computed by replacing each leaf $\bullet$ in τ_2 with τ_3, which is equivalent to replace each leaf $\bullet$ in τ_2^l and τ_2^r with τ_3. Thus, $\tau_1^l = \tau_2^l \bowtie \tau_3$ and $\tau_1^r = \tau_2^r \bowtie \tau_3$. ◀

Proof of Lemma 6.

1. We construct an infinite sequence S of prime trees: let $p_1 \triangleq$  , $p_j \triangleq$  , *i.e.*

$$S \triangleq \left\{ \text{  } \right\}$$

   It is immediate that p_1 is prime. To prove that p_i is prime for $i > 1$, we proceed as follows. Suppose $p_i = \tau_1 \bowtie \tau_2$ and neither τ_1 nor τ_2 is $\bullet$. The right subtree of each p_i is just $\bullet$ and by the definition of $\bowtie$ must contain a copy of τ_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 τ_1, τ_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 $\tau_1 =$  , $\tau_1' =$  then by Lemma 7, we derive $\tau_1^l \bowtie \tau_2 = \tau_1^{l'} \bowtie \tau_2', \tau_1^r \bowtie \tau_2 = \tau_1^{r'} \bowtie \tau_2'$. By our induction hypothesis, $\tau_1^l = \tau_1^{l'}, \tau_1^r = \tau_1^{r'}, \tau_2 = \tau_2'$. 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 $\bowtie$ is not commutative we get a stronger version of the traditional theorem:

▶ **Lemma 8** (Unique representation of $\mathcal{S}^+$). *For each* $\tau \in \mathbb{T}^+ \backslash \{\bullet\}$, *there exists a unique sequence* $\tau_1, ..., \tau_n \in \mathbb{T}_p$ *such that* $\tau = \bowtie_{i=1}^n \tau_i$. *Each* τ_i *is called a prime factor of* τ.

Proof. We prove by induction on the height of τ. The base case $\mathbb{T}^1$ is trivial. Assume it holds for $\mathbb{T}^k$ and let $\tau \in \mathbb{T}^{k+1}$. If τ is prime then we are done. Otherwise, let $\tau^1, \tau^2 \in \mathbb{T}^k \backslash \{\bullet\}$ and $\tau = \tau^1 \bowtie \tau^2$. By our induction hypothesis, there are 2 sequences $\tau^1_1, ..., \tau^1_{k_1} \in \mathbb{T}_p$ and $\tau^2_1, ..., \tau^2_{k_2} \in \mathbb{T}_p$ such that $\tau^1 = \bowtie_{i=1}^{k_1} \tau^1_i$ and $\tau^2 = \bowtie_{i=1}^{k_2} \tau^2_i$ and thus $\tau = (\bowtie_{i=1}^{k_1} \tau^1_i) \bowtie (\bowtie_{i=1}^{k_2} \tau^2_i)$. The uniqueness is a consequence of property 3 from Lemma 6. ◀

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

Accordingly, we will use $\mathbb{T}_p$ 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 $(\mathbb{T}_p^*, \cdot)$*

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

We now wish to prove that for any τ_1 and τ_2, $f(\tau_1 \bowtie \tau_2) = f(\tau_1) \cdot f(\tau_1)$. 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 τ_1 nor τ_2 is $\bullet$. Let $p_1, \ldots, p_i$ be the unique prime factors of τ_1 and $p'_1, \ldots, p'_j$ be the unique prime factors of τ_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:

$$f(\tau_1 \bowtie \tau_2) \;=\; f(p_1 \bowtie \cdots \bowtie p_i \bowtie p'_1 \bowtie \cdots \bowtie p'_j) \;=\; p_1 \bullet \cdots \bullet p_i \bullet p'_1 \bullet \cdots \bullet p'_j \;=$$
$$(p_1 \bullet \cdots \bullet p_i) \bullet (p'_1 \bullet \cdots \bullet p'_j) \;=\; f(\tau_1) \bullet f(\tau_2)$$

To prove f is surjective, let $w \in \mathbb{T}_p^*$ be the concatenation of primes $p_1 \bullet \cdots \bullet p_i$; 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 τ_1 and $p'_1, \ldots p'_j$ be the prime factors of τ_2. Accordingly we know that $p_1 \bullet \cdots \bullet p_i = p'_1 \bullet \cdots \bullet p'_j$, and since equality over words can only occur if the words have 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 $E_{\mathbb{T}^+}$ over $(\mathbb{T}^+, \bowtie)$ contain both tree constants $\tau \in \mathbb{T}^+$ and variables $v \in V$; we can map these to word equations $E_\mathbb{W}$ over $(\mathbb{T}_p^*, \cdot)$ 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 $E_{\mathbb{T}^+}$ then $f \circ \rho$ satisfies $E_\mathbb{W}$, 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 Lemma 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 3.1 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 τ is in P.*

Proof. Let $\mathbb{S}(\tau)$ be the set of all subtrees of τ and $\mathbb{S}_n(\tau) \subset \mathbb{S}(\tau)$ be the set of all subtrees of τ with height exactly n. $\mathbb{S}(\tau)$ can be computed recursively: $\mathbb{S}(\circ) = \{\circ\}$, $\mathbb{S}(\bullet) = \{\bullet\}$, $\mathbb{S}(\overset{\tau_1}{\frown}\overset{\tau_2}{}) = \mathbb{S}(\tau_1) \cup \mathbb{S}(\tau_2) \cup \{\overset{\tau_1}{\frown}\overset{\tau_2}{}\}$. If $\tau = \tau_1 \bowtie \tau_2$ ($\{\tau_1, \tau_2\} \subset \mathbb{T}^+ \backslash \{\bullet\}$), then there exists $n \in \mathbb{N}$ such that $\mathbb{S}_n(\tau) = \{\tau_2\}$, that is, $\mathbb{S}_{|\tau_2|}(\tau)$ is exactly the singleton set $\{\tau_2\}$. Additionally, $S(\tau) = \bigcup_{i=0}^{|\tau|} S_i(\tau)$.

Thus we can find all the prime factors of τ (which is inspired from the well-known sieve of Eratosthenes) as follows: first we compute $\mathbb{S}(\tau)$ and partition it into $\mathbb{S}_0(\tau), \ldots, \mathbb{S}_{|\tau|}(\tau)$. Let $i \in \mathbb{N}$ be the smallest number such that $\mathbb{S}_i(\tau)$ is the singleton set $\{\tau_1\}$ for some $\tau_1 \in \mathbb{T}$ (note that i must be larger than 0 since $\mathbb{S}_0(\tau) = \{\circ, \bullet\}$). If $i = |\tau|$ then τ itself is a prime, otherwise, we replace all subtrees τ_1 of τ with $\bullet$ and call the new tree τ'. If all the "old" $\bullet$ leaves of τ are replaced and τ' is in canonical form then $\tau = \tau' \bowtie \tau_1$, τ_1 is a prime factor of τ, and we can repeat the process with τ' to find the next prime factor. Otherwise, we consider the next singleton set $\mathbb{S}_j(\tau)$.

If τ has n leaves than its description requires $O(n)$ bits and the time to compute $\mathbb{S}(\tau)$ is $O(n)$. Note that $|\mathbb{S}_k(\tau)| \leq \frac{n}{k+1}$ because there are at least $k+1$ leaves in a tree of height k. Therefore, the number of subtrees from height 1 to n is at most $\Sigma_{i=1}^{n} \frac{n}{i+1} \leq n^2$. Computing the height of a subtree τ' of τ requires $O(n)$, thus the time to partition $\mathbb{S}(\tau)$ is $O(n^3)$. The number of times we need to restart the process is $O(n^2)$. Consequently, the time for tree factorization is $O(n^5)$, polynomial in the description of τ (more efficient solutions exist). ◀

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 $\mathbb{T}_p^*$. Since the representation of the constants does not affect the computational properties of the theory, we can conclude that $\mathbb{T}^+$ is NP-hard. ◀

5 A reduction to Tree Automatic Structures

Theorem 2 shows that the first order theory (FO) over $\mathcal{S}$ is undecidable, so of course any extension of $\mathcal{S}$—*e.g.* with $(\sqcap, \sqcup, \bar{\sqsubset})$—also has an undecidable FO. However, if we restrict the

form of $\bowtie$-equations to be $\pi_1 \bowtie \tau = \pi_2$ where $\tau \in \mathbb{T}$, then the FO of $\mathcal{S}$ 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 ${}_\tau\bowtie : \mathbb{T} \to \mathbb{T}$ and $\bowtie_\tau : \mathbb{T} \to \mathbb{T}$ be the *left* and *right restricted form* of $\bowtie$ with respect to $\tau \in \mathbb{T}$, *i.e.* ${}_\tau\bowtie(\tau_1) \triangleq \tau \bowtie \tau_1$ and $\bowtie_\tau(\tau_1) \triangleq \tau_1 \bowtie \tau$. We will show $\mathcal{T} \triangleq (\mathbb{T}, \sqcap, \sqcup, \bar{\square}, \bowtie_\square)$, where $\bowtie_\square$ denotes the family of all right-restricted forms of $\bowtie$ indexed by tree constants, is tree-automatic by constructing bottom-up tree automata that recognize the domain $\mathbb{T}$ and the relations in $\mathcal{T}$:

▶ **Theorem 14** (Decidability of $(\mathbb{T}, \sqcap, \sqcup, \bar{\square}, \bowtie_\square)$)**.** *$\mathcal{T}$ is tree-automatic. As a result, the first-order theory of $\mathcal{T}$ is decidable.*

As indicated by equation (7) (from §2), one use for $\bowtie$ is to split/join ownership of maps-to predicates in separation logic. Here the splitting/joining occurs on the right-hand side of the $\bowtie$. 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*, $Q_f \subset Q$ is the *set of accepting states* and Δ is the *set of transitions* $f(q_1(v_1), ..., q_n(v_n)) \to q_{n+1}(f(v_1, ..., v_n))$ where $f \in F$ is a 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 Δ 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 $\mathbb{T}$ is $\mathcal{F} = \{\circ^{(0)}, \bullet^{(0)}, \mathsf{Node}^{(2)}\}$.

Tree automatic structures. Let $\mathcal{R} \subset \mathcal{D}^k$ 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 $\diamond^{(0)}$ that is not in F, *e.g.* $(\mathsf{Node}(a_1, a_2), b, \mathsf{Node}(c_1, c_2)) \mapsto [\mathsf{Node}, b, \mathsf{Node}]([a_1, \diamond, c_1], [a_2, \diamond, c_2])$. As a result, if $F_\mathcal{D}$ is the ranked alphabet of $\mathcal{D}$ then $(F_\mathcal{D} \cup \{\diamond\})^k$ is the ranked alphabet for $\mathcal{D}^k$ 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 a tree automatic structures is decidable.*

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

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

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

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

- $[\bullet, \circ] \mapsto q([\bullet, \circ])$ and $[\circ, \bullet] \mapsto q([\circ, \bullet])$
- $[\mathsf{Node}, \mathsf{Node}](q(v_1), q(v_2)) \mapsto q([\mathsf{Node}, \mathsf{Node}](v_1, v_2))\}$

Construction of $\mathcal{A}^{\sqcup}$ and $\mathcal{A}^{\sqcap}$. $\mathcal{A}^{\sqcup}$ and $\mathcal{A}^{\sqcap}$ are more sophisticated as trees can have different shapes and thus are inapplicable for leaf-wise comparison. For instance, the convolution of the relation $\overset{\circ}{\underset{\bullet}{\frown}} \sqcup \bullet = \bullet$ is $[\mathsf{Node}, \bullet, \bullet]([\circ, \diamond, \diamond], [\bullet, \diamond, \diamond])$. When we proceed upward, the leaf values of the second and third tree are initially unknown (denoted by $\diamond$) but later recognized as $\bullet$. The trick to build the automaton is by *guessing*: when moving bottom-up, the states of the automaton record the set of all possible values for the unknown leaves and later check whether the observed values belong to the guessing set. When proceeding upward, if two guessing sets meet at a certain step, they are *unified* into a single guessing set: first we compute their intersection and then unify it with the observed values at the current node. In details, we define $F^{\sqcup} = (\mathcal{F} \cup \{\diamond\})^3$, $Q^{\sqcup} = \{q_S \mid S \subseteq \{(\tau_1, \tau_2, \tau_3) \mid \tau_i \in \{\circ, \bullet, \star\}\}\}$ and $Q^{\sqcup}_f = \{q_{\{(\star,\star,\star)\}}\}$ where $\star$ indicates the value was already seen. Let $D = \{\circ, \bullet, \diamond, \mathsf{Node}, \star\}$ and $\phi : D \times D \to \{\star, \diamond\}$ be the *unit unification*:

- $\phi(\mathsf{Node}, \star) = \phi(\circ, \circ) = \phi(\bullet, \bullet) = \star, \phi(\diamond, \diamond) = \diamond$
- $\phi(\tau_1, \tau_2)$ is undefined otherwise

We extend ϕ to $\phi' : D^3 \times \mathbb{P}(D^3) \to \mathbb{P}(D^3)$ to handle the convolution form of $\sqcup$: $\phi'((\tau_1, \tau_2, \tau_3), S) = \{(\tau'_1, \tau'_2, \tau'_3) \mid \exists k_1, k_2, k_3.((k_1, k_2, k_3) \in S \bigwedge_{i=1}^{3} \phi(\tau_i, k_i) = \tau'_i)\}$. Finally, the transition relation is defined to be $\Delta^{\sqcup} = S_g \cup S_u$ where S_g contains all *guessing rules* at leaf level for $\sqcup$ and S_u is the transition part for unification that contains relations of the form $[\tau_1, \tau_2, \tau_3](q_{S_1}(v_1), q_{S_2}(v_2)) \mapsto q_S([\tau_1, \tau_2, \tau_3](v_1, v_2))$ such that $S = \phi'((\tau_1, \tau_2, \tau_3), S_1 \cap S_2)$. Similarly, the automaton for $\sqcap$ can be constructed by modifying S_g. For demonstration, consider $\overset{\circ}{\underset{\bullet}{\frown}} \sqcup \bullet = \bullet$ whose convolution $[\mathsf{Node}, \bullet, \bullet]([\circ, \diamond, \diamond], [\bullet, \diamond, \diamond])$ has the following run:

- $[\circ, \diamond, \diamond] \mapsto q_{S_1}([\circ, \diamond, \diamond])$ where $S_1 = \{(\star, \bullet, \bullet), (\star, \circ, \circ)\}$
- $[\bullet, \diamond, \diamond] \mapsto q_{S_2}([\bullet, \diamond, \diamond])$ where $S_2 = \{(\star, \bullet, \bullet), (\star, \circ, \bullet)\}$
- As $S = S_1 \cap S_2 = \{(\star, \bullet, \bullet)\}$ and $\phi'((\mathsf{Node}, \bullet, \bullet), S) = \{(\star, \star, \star)\}$, we have:
  $[\mathsf{Node}, \bullet, \bullet](q_{S_1}([\circ, \diamond, \diamond]), q_{S_2}([\bullet, \diamond, \diamond])) \mapsto q_{\{(\star,\star,\star)\}}([\mathsf{Node}, \bullet, \bullet]([\circ, \diamond, \diamond], [\bullet, \diamond, \diamond]))$

Construction of $\mathcal{A}^{\bowtie_\tau}$. Next, we give the description of bottom-up tree automaton $\mathcal{A}^{\bowtie_\tau}$ that recognizes $\bowtie_\tau$. Let $\mathbb{S} : \mathbb{T} \to \mathcal{P}(\mathbb{T})$ be the function that extracts all subtrees: $\mathbb{S}(\bullet) = \{\bullet\}$, $\mathbb{S}(\circ) = \{\circ\}$, $\mathbb{S}(\overset{\tau_1}{\underset{\tau_2}{\frown}}) = \{\overset{\tau_1}{\underset{\tau_2}{\frown}}\} \cup \mathbb{S}(\tau_1) \cup \mathbb{S}(\tau_2)$. The ranked alphabet for $\mathcal{A}^{\bowtie_\tau}$ is $F^{\bowtie_\tau} = (\mathcal{F} \cup \{\diamond\})^2$. The state space for $A^{\bowtie_\tau}$ is $Q^{\bowtie_\tau} = \{q_{\tau'} \mid \tau' \in \mathbb{S}(\tau)\} \cup \{q_f\}$ and $Q^{\bowtie_\tau}_f = \{q_f\}$. W.l.o.g., we assume that $\tau \notin \{\circ, \bullet\}$ as those cases can be trivially reduced to equalities, which is a special case of $\sqcup$: $\tau_1 = \tau_2 \Leftrightarrow \tau_1 \sqcup \circ = \tau_2$. The transition relation $\Delta^{\bowtie_\tau}$ is defined as follows:

a_1. $[\circ, \circ] \mapsto q_f([\circ, \circ])$

a_2. $[\diamond, \tau'] \mapsto q_{\tau'}([\diamond, \tau'])$ if $\tau' \in \mathbb{S}(\tau)$

a_3. $[\diamond, \mathsf{Node}](q_{\tau_1}(v_1), q_{\tau_2}(v_2)) \mapsto q_{\tau_3}([\diamond, \mathsf{Node}](v_1, v_2))$ if $\tau_3 = \overset{\tau_1}{\underset{\tau_2}{\frown}} \in \mathbb{S}(\tau)$ and $\tau_3 \neq \tau$

b_1. $[\bullet, \mathsf{Node}](q_{\tau_1}(v_1), q_{\tau_2}(v_2)) \mapsto q_f([\bullet, \mathsf{Node}](v_1, v_2))$ if $\tau = \overset{\tau_1}{\underset{\tau_2}{\frown}}$

b_2. $[\mathsf{Node}, \mathsf{Node}](q_f(v_1), q_f(v_2)) \mapsto q_f([\mathsf{Node}, \mathsf{Node}](v_1, v_2))$

Briefly speaking, we traverse upward and check whether τ is a subtree of the second tree ($a_1 - a_3$). When $\bullet$ is seen from the first tree (b_1), we check whether the two trees have similar shape (b_2).

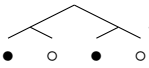
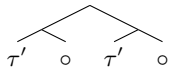
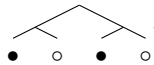
▶ Remark. As we are about to prove, the other relation ${}_\tau\bowtie$ is not tree-automatic in this representation. We leave an open question whether there exists a representation in which both $\bowtie_\tau$ and ${}_\tau\bowtie$ are automatic *i.e.* whether $\bowtie$ is semi-automatic.

▶ **Proposition 16** (${}_\tau\bowtie$). *In the current representation of tree shares, there exists infinitely many τ such that ${}_\tau\bowtie$ is not tree-automatic.*

First, we recall the Pumping Lemma for tree automata:

▶ **Definition 17** (Term, context and substitution [9]). Let $\mathcal{A} = (\mathcal{Q}, \mathcal{F}, \mathcal{Q}_f, \Delta)$ be a tree automaton and V the set of variables. We define $T(\mathcal{F}, V)$ the set of all tree terms derived from $\mathcal{F} \cup V$ and $T(\mathcal{F}, \emptyset)$ 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 $T(\mathcal{F}, V)$ and $\mathcal{C}(\mathcal{F})$ 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 $C \in \mathcal{C}(\mathcal{F})$ 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 $C^{n+1}[t] = C[C^n[t]]$.

▶ **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 $C \in \mathcal{C}(\mathcal{F})$, a nontrivial context $C' \in \mathcal{C}(\mathcal{F})$ and a ground term t' such that $t = C[C'[t']]$ and $\forall n \in \mathbb{N}$. $C[C'^n[t']] \in \mathcal{L}$.*

Proof of Proposition 16. Let ${}_\tau\bowtie$ where $\tau =$  . For an input tree $\tau' \in \mathbb{T}^+$, the result tree is  in which the left and right subtree are identical. If ${}_\tau\bowtie$ is automatic then it satisfies the Pumping Lemma. However, the Pumping Lemma only allows us to pump either the left or the right subtree and thus they will be different after pumping, which is a contradiction. Now consider the following sequence: $\tau_1 =$  , $\tau_{n+1} =$  then each of the ${}_{\tau_i}\bowtie$ is not automatic. ◀

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 $\bowtie$, $\sqcup$, $\sqcap$, and $\overline{\square}$, where $\bowtie$ 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 $\bowtie_\square$ 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 $\bowtie$ operator and by way of an isomorphism to word equations prove that the existential theory is in PSPACE while the general first-order theory is undecidable. We have identified a restricted version of $\bowtie$ that takes constants on the right-hand side $\bowtie_\square$ and have proven that the system $(\mathbb{T}, \sqcup, \sqcap, \overline{\square}, \bowtie_\square)$ has a decidable first-order theory via an embedding to tree-automatic structures.

References

1. Parosh Aziz Abdulla, Yu-Fang Chen, Lukás Holík, Richard Mayr, and Tomás Vojnar. When simulation meets antichains. In *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, pages 158–174, 2010.
2. H. Abdulrab and J.P Pechuchet. Solving word equations. In *Journal of Symbolic Computation*, pages 499–521, 1989.
3. Andrew W. Appel, Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy, and Xavier Leroy. *Program Logics for Certified Compilers*. Cambridge University Press, New York, NY, USA, 2014.
4. Leonard Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–77, May 1980.
5. A. Blumensath. *Automatic Structures*. PhD thesis, RWTH Aachen, 1999.
6. A. Blumensath and E. Grade. Finite presentations of infinite structures: automata and interpretations. In *Theory of Computer Systems*, pages 641–674, 2004.
7. John Boyland. Checking interference with fractional permissions. In *Static Analysis, 10th International Symposium, SAS 2003, San Diego, CA, USA, June 11-13, 2003, Proceedings*, pages 55–72, 2003.
8. J Richard Büchi and Steven Senger. Definability in the existential theory of concatenation and undecidable extensions of this theory. In *The Collected Works of J. Richard Büchi*, pages 671–683. Springer, 1990.
9. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.
10. Robert Dockins, Aquinas Hobor, and Andrew W. Appel. A fresh look at separation algebras and share accounting. In *Programming Languages and Systems, 7th Asian Symposium, APLAS 2009, Seoul, Korea, December 14-16, 2009. Proceedings*, pages 161–177, 2009.
11. Aquinas Hobor, Andrew W. Appel, and Francesco Zappa Nardelli. Oracle semantics for concurrent separation logic. In *17th European Symposium of Programming (ESOP 2008)*, pages 353–367, April 2008.
12. Aquinas Hobor and Cristian Gherghina. Barriers in concurrent separation logic. In *Programming Languages and Systems - 20th European Symposium on Programming, ESOP 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 276–296, 2011.
13. Joxan Jaffar. Minimal and complete word unification. *J. ACM*, 37(1):47–85, 1990.
14. Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng, and Siyuan Zou. Semi-automatic structures. In *Computer Science - Theory and Applications - 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, pages 204–217, 2014.

15 Bakhadyr Khoussainov and Mia Minnes. Three lectures on automatic structures. In *Logic Colloquium 2007 (F. Delon, U. Kohlenbach, P. Maddy, and F. Stephan, editors), Lecture Notes in Logic, vol. 35, Association for Symbolic Logic*, pages 132–176, 2007.

16 Antoni Koscielski and Leszek Pacholski. Complexity of Makanin's algorithm. *J. ACM*, 43(4):670–684, 1996.

17 Dexter Kozen. Complexity of boolean algebras. In *Theoretical Computer Science 10*, pages 221–247, 1980.

18 D. Kuske. Theories of orders on the set of words. In *Theoretical Informatics and Applications 40*, pages 53–74, 2006.

19 Xuan Bach Le, Cristian Gherghina, and Aquinas Hobor. Decision procedures over sophisticated fractional permissions. In *Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*, pages 368–385, 2012.

20 G. S Makanin. The problem of solvability of equations in a free semigroup. In *Mat. Sbornik*, pages 147–236, 1977.

21 G. S Makanin. Equations in a free group. In *Izvestiya AN SSSR*, pages 1199–1273, 1982-1983.

22 G.S Makanin. Decidability of the universal and positive theories of a free group. In *Izvestiya AN SSSR*, pages 735–749, 1984-1985.

23 S. Marchenkov. Unsolvability of positive $\forall\exists$-theory of free semi-group. In *Sibirsky mathmatichesky jurnal*, pages 196–198, 1982.

24 Kim Marriott and Martin Odersky. Negative boolean constraints. In *Theoretical Computer Science 160*, pages 365–380, 1996.

25 W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Journal of the Association for Computing Machinery*, pages 483–496, 2004.

26 W. Plandowski. An efficient algorithm for solving word equations. In *Proc 38th Annual Symposium on Theory of Computing*, pages 467–476, 2006.

27 M.O. Rabin. Decidability of second-order theories and automata on infinite trees. In *Trans AMS*, pages 341–360, 1960.

28 Stijn Vermeeren. Embeddings into the countable atomless Boolean algebra. `http://stijnvermeeren.be/download/mathematics/embeddings.pdf`, 2010.

29 Jules Villard. *Heaps and Hops*. Ph.D. thesis, Laboratoire Spécification et Vérification, École Normale Supérieure de Cachan, France, February 2011.