Complexity analysis of tree share operations

Xuan Bach Le† Aquinas Hobor‡† Anthony W. Lin∗

†National University of Singapore ‡Yale-NUS College ∗University of Oxford

Abstract. We investigate the complexity of the tree share model of Dockins et al., which is used to reason about shared ownership of resources in e.g. concurrent programs. We prove that the complexity of the full Boolean algebra of tree shares (that is, with all tree-share constants) is the same as the complexity of the whole class of countably atomless Boolean algebras (that is, where the only constants allowed in formulae are 0 and 1); the previous bound was nonelementary (that is, not bounded by any fixed tower of exponentials). We also prove a 2EXSPACE upper bound on the complexity of the first-order theory over the “relativization” multiplication operator on trees if one operand is a constant; the previous bound was nonelementary if the constant was on the right and not known to be decidable if the constant was on the left. Finally, we prove that the first-order theory over the structure that combines the Boolean algebra operators and the right-handed multiplication has a nonelementary lower bound even though the key subtheories are elementary, showing that the existing nonelementary upper bound cannot be improved.

1 Introduction

Fractional shares are used to reason about shared ownership of resources. For example, a memory cell may be “fully owned” by a thread, permitting e.g. both reading and writing; or “partially owned”, permitting only reading. The initial model of fractional shares was rational numbers in the interval [0, 1], but over the years a number of defects were discovered and several improved models were developed, culminating in the “tree share” model of Dockins et al. [8].

A tree share τ ∈ T is a binary tree with Boolean leaves. Full ownership is represented by • and no ownership by ◦. To represent fractional ownership one can use e.g. ◦• to represent a half-owned resource. Importantly and usefully, ◦• is a distinct tree share representing the “other half”. In §2.4 we will develop the structure of tree shares precisely and define the operators of Boolean algebra (⊔, ⊓, ¬) over them; the details are a little subtle but the basic idea is to do the operations leafwise and so e.g. ◦• ⊔ ◦• = •, ◦• ⊓ ◦• = ◦, and ◦• = ◦•. We use the symbol M to indicate the structure (T, ⊔, ⊓, ¬).

We studied the decidability and computational complexity of automated reasoning over the tree share model and previously established some basic results over the existential and first-order theories [18]. We showed that the structure (T, ⊔, ⊓, ¬, •, ◦) was a Countable Atomless Boolean Algebra (CABA) and thus
complete for the class $\text{STA}(\ast, 2^n, n)$ of problems that are solvable by alternating exponential-time Turing machines with unrestricted space and $n$ alternations, i.e., the same complexity as the first-order theory over the reals $(\mathbb{R}, +, 0, 1)$ with addition but no multiplication [3]. Similarly, we also showed that the existential theory of over this structure is NP-complete.

Note that the formulas over this structure can only use $\bullet$ and $\circ$ for constants, whereas the structure $\mathcal{M}$ permits arbitrary constants; the purpose of this paper is to explore the algorithmic consequences of introducing explicitly given constants in tree share formulas. Generally speaking, the ability to directly talk about constants in a logical formula is natural, e.g., in Presburger Arithmetic, it is natural to write a formula like $\forall x, y \exists z(x + y + 7 \geq z)$ wherein integer constants (in this case 7) are allowed in the formula. A recent benchmark using tree shares for program verification [19] made approximately 16k calls into a solver for a fragment of the first-order theory over $\mathcal{M}$; approximately 21.1% (71k/335k) of the constants used in practice were neither $\bullet$ nor $\circ$.

Dockins et al. also defined a kind of “multiplication” operator on trees denoted by $\tau_1 \triangleright \triangleright \tau_2$, in which each $\bullet$ inside the tree $\tau_1$ is replaced with a copy of $\tau_2$, e.g., $\bullet \circ \circ \bullet \triangleright \triangleright \circ \bullet = \circ \circ \circ \circ \circ \circ \circ$. The $\triangleright \triangleright$ operator enjoys the unit $\bullet$, is associative, injective over non-$\circ$ arguments, and distributes over $\sqcup$ and $\sqcap$ on the left [8]. When the multiplication operator $\triangleright \triangleright$ is added, things become harder. The structure $(T, \triangleright \triangleright)$—that is, without the Boolean operators—is isomorphic to word equations [18]. Accordingly, its first-order theory is undecidable while its existential theory is decidable (starting from Makanin’s intricate argument [20] in 1977, which is still continuously being simplified e.g. see [14]).

To recover decidability, we examined the structure $\mathcal{K} = (T, \sqcup, \sqcap, \bar{\Box}, \triangleright \triangleright)$, in which we only allow multiplication by a constant on the right hand side, i.e. $\triangleright \triangleright_\tau (x) = x \triangleright \triangleright \tau$. The first-order theory over this structure is decidable by an encoding into tree automata [18], giving it a nonelementary upper bound (that is, this upper bound cannot be bounded by any fixed tower of exponentials). Because $\mathcal{M}$ is a subtheory of $\mathcal{K}$, this yields an nonelementary upper bound for the first-order Boolean theory with constants and an elementary lower bound with only $\bullet$/$\circ$ constants (via the CABA argument). Tree automata also yields a nonelementary upper bound for the first-order subtheory $(T, \triangleright \triangleright_\tau)$.

Contributions. We study the complexity of first-order theories over tree shares when formulas include constants and/or when certain operands are constant.

§3 On the Boolean algebra side, we show that the first order theory over $\mathcal{M}$ (with constants) is in the same complexity class as for standard CABAs (with only 0 and 1 for constants).

§4 On the multiplicative side, we show that the theory $\mathcal{R} = (T, \ast, \circ, \triangleright \triangleright_\tau)$, which allows multiplication by constants on both the left and the right, is decidable in $2\text{EXSPACE}$, i.e., double exponential space.

§5 When we combine these theories together, we show that we cannot do better than the existing upper bound. In particular, we show that $\mathcal{K}$, despite being the combination of elementary theories, has a nonelementary lower bound.
Due to their good metatheoretic properties, a number of program logics incorporate tree shares to model fractional ownership [11, 10, 24, 2]. Using tree shares in verification tools has been historically challenging due to a lack of foundational results regarding decidability and complexity of theories over tree shares, although several tools have incorporated the Boolean algebra structure with varying levels of success and completeness [12, 17, 24, 19]. With our improved understanding of the combined structure $K$ we are actively exploring how to incorporate their multiplicative structure into such verification tools as well, e.g., by using powerful heuristics for automata such as antichain and simulation [1].

2 Preliminaries and notations

Here we document the preliminaries for our result. Some of these are standard (§2.1–§2.3) while others are specific to the domain of tree shares (§2.4).

2.1 Language and structure

A *signature* is a triple $σ = ⟨F, P, \text{arity}⟩$ in which $F = \{f_1, \ldots, f_n\}$ and $P = \{P_1, \ldots, P_m\}$ are two disjoint sets of function symbols and predicate symbols respectively whereas \text{arity} : $F \cup P \mapsto \mathbb{N}$ is the arity function that specifies the number of arguments for functions and predicates. Let $V = \{v_1, v_2, \ldots\}$ be the set of variables, a $σ$-term is either a variable or of the form $f_k(t_1, \ldots, t_k)$ in which $\{t_i\}_{i=1}^k$ are $σ$-terms and $f_k$ is a $k$-ary function. An *atomic $σ$-formula* is either the equality between two $σ$-terms $t_1 = t_2$ or a predicate of $σ$-terms $P_k(t_1, \ldots, t_n)$ in which $P_k$ is a $k$-ary predicate. A *first-order $σ$-formula* is an element of the closure of atomic $σ$-formulas under logical connectives $\land, \lor, \rightarrow, \neg$ and quantifiers $\forall, \exists$. A variable instance $v$ is *bounded* if it is within the scope of some quantifier $\forall v$ or $\exists v$, otherwise $v$ is *free*. A $σ$-formula $Φ$ is a *sentence* if it does not contain any free variables. A $σ$-theory is simply a set of *first-order $σ$-sentences*. A $σ$-theory $T$ is *complete* if for each $σ$-sentence $Φ$, either $Φ$ or $¬Φ$ is in $T$. We call $T$ *decidable* if membership testing of $σ$-sentences in $T$ is decidable. Let $Σ_1$ be the set of existential formulas $\exists v_1 \ldots \exists v_n. Φ$ and $Π_1$ the set of universal formulas $\forall v_1 \ldots \forall v_n. Φ$ in which $Φ$ is quantifier-free; then let $Σ_{i+1}$ be the set of formulas $∃v_1 \ldots ∃v_n. Φ$ for $Φ ∈ Π_i$ and $Π_{i+1}$ the set of formulas $∀v_1 \ldots ∀v_n. Φ$ for $Φ ∈ Σ_i$.

A $σ$-structure is an interpretation of the symbols in $σ$. Formally, a $σ$-structure is the triple $A = ⟨U, F^A, P^A⟩$ in which $U$ is the *universe of discourse*. For each $k$-ary function symbol $f$ in $F$, we have a corresponding $k$-ary function $f^A : U^k \mapsto U$ in $F^A$. Similarly, each $k$-ary predicate symbol $P$ in $P$ corresponds to a $k$-ary predicate $P^A \subseteq U^k$ in $P^A$. The structure $A$ satisfies a $σ$-formula $Φ$, denoted by $A \models Φ$, if $Φ$ is true under the interpretation of $A$. The *first-order theory of $A$, denoted by Th($A$), is the set of $σ$-sentences that are satisfied by $A$, i.e., Th($A$) = $\{Φ \mid A \models Φ\}$. Let $A = \{Ψ_1, Ψ_2, \ldots\}$ be a set of $σ$-sentences called *axioms*, the *first-order theory of $A$ is the set of sentences Th($A$) = $\{Φ \mid \text{for each } σ\text{-structure } A, \text{if } A \models Φ \text{ then } A \models Φ\}$. Any $σ$-structure that satisfies a theory $T$ is called $T$-model. Two $σ$-structures $A_1$ and $A_2$ are *elementary equivalent* if they satisfy the same set of first-order $σ$-sentences, i.e., $Th(A_1) = Th(A_2)$. Recall a well-known fact about atomless Boolean Algebras:
Proposition 1 (Folklore). Let $BA$ be the finite set of axioms for atomless Boolean Algebra then $\text{Th}(BA)$, the first-order theory of atomless Boolean Algebra, is complete and $\omega$-categorical, i.e., any two models are elementary equivalent and the theory has exactly one countably infinite model up to isomorphism.

If the signature $\sigma$ is clear from the context or not important, we will usually omit the prefix $\sigma$ in the corresponding names. For convenience, we will usually abuse a function (predicate) with its symbol, i.e., $f$ represents both the function symbol in $\sigma$ and the function $f^A$ in $A$. As a result, we will usually mention structures without introducing their signatures as such signatures can be derived from the structures themselves. For the purpose of this paper, the universe of the structure is a part of the signature, i.e., it is also the set of constant symbols in the signature. In addition, we will introduce a structure as $A = (U, X_1, \ldots, X_n)$ in which $A$ is the name of the structure, $U$ is its universe and $X_i$ is either a function or predicate whose arity is implicitly known. Also, we reuse some notations in different domains as long as there is no confusion.

2.2 Complexity

If $T$ be a decidable theory, the complexity of $T$ is measured by the (time or space) complexity of the halted (deterministic or nondeterministic) Turing machine that decides $T$, e.g., NP or PSPACE. Let the exponent function $\exp : \mathbb{N}^2 \rightarrow \mathbb{N}$ be defined as $\exp(n, 0) = n$ and $\exp(n, k + 1) = 2^{\exp(n, k)}$, then the complexity class $k\text{EXP}$ contains problems which can be decided by a halted deterministic Turing of time complexity $\exp(cn, k)$ for input of length $n$ and some constant $c$. Similarly, we use $k\text{NEXP}$ and $k\text{EXSPACE}$ for exponential time complexity of nondeterministic halted Turing machines and space complexity of deterministic halted Turing machines respectively. A problem is elementary if it is in $k\text{EXP}$ for some $k$, otherwise it is called non-elementary. Furthermore, let $\text{STA}(p(n), t(n), a(n))$ denote the class of problems decided by an alternating Turing machine that uses at most $p(n)$ space, $t(n)$ time and $a(n)$ alternations for input of length $n$. If one of the three bound is not specified, we can use *. Recall some classical complexity results for Boolean Algebras that we will need in subsequent sections:

Proposition 2 ([21]). Let $B$ be an infinite Boolean Algebra then the complexity of $\Sigma_1 \cap \text{Th}(B)$, the existential theory of $B$, is NP-complete.

Proposition 3 ([15]). Let $BA$ be the finite set of axioms for atomless Boolean Algebra then the complexity of $\text{Th}(BA)$, the first-order theory of atomless Boolean Algebras, is $\text{STA}(*, 2^{O(n)}, n)$-complete.

2.3 Automatic structures with bounded degree

Fix a signature $\sigma = (F, P, \text{ary})$ and a $\sigma$-structure $A = (U, F^A, P^A)$. For simplicity, we treat a $k$-ary function as $(k + 1)$-ary predicate and thus represent the $\sigma$-structure as $A = (U, P^A)$. The Gaifman-graph $G(A) = (V, E)$ of $A$ is a symmetric graph whose vertex set $V$ is the universe $U$ and there is an edge between
a and b if there exists a k-ary tuple \( t = (r_1, \ldots, r_k) \) and a k-ary predicate \( P^A \) s.t. \( t \in P^A \) and both a, b are in \( t \). The structure \( A \) has bounded degree of \( k \), if its Gaifman-graph \( G(A) \) also has bounded degree of \( k \), i.e., each vertex \( a \in V \) is adjacent to at most \( k \) other vertices. We call \( A \) has bounded degree if \( A \) has bounded degree of \( k \) for some \( k \).

The structure \( A \) is automatic if its universe \( U \) and predicates can be computed using finite automata. One classical result is the first order theory of automatic structures is decidable but the decision procedure is non-elementary (e.g. [4, 5]). However, a precise elementary bound can be obtained for automatic structures with bounded degrees:

**Proposition 4 ([16]).** The complexity of the class of first-order theory of automatic structures with bounded degree is \( 2\text{EXSPACE}\)-complete.

### 2.4 Boolean binary tree structure

Here we summarize additional details of tree shares and their associated properties from Dockins et al. [8].

**Canonical forms.** A tree share is either \( \cdot \), \( \circ \) or Node\((\tau_1, \tau_2)\) in which \( \tau_1, \tau_2 \) are tree shares and Node is a binary function. To make the representation more appealing, we will refer Node\((\tau_1, \tau_2)\) as \( \tau_1 \tau_2 \). Additionally, we require tree shares are in canonical form, i.e., it is in its most compact representation under the inductively-defined equivalence relation \( \cong \):

\[
\begin{align*}
\circ & \cong \circ \\
\cdot & \cong \cdot \\
\circ \cong \circ \circ \\
\cdot & \cong \cdot \cdot \\
\end{align*}
\]

For example, \( \cdot \circ \circ \cdot \) is not canonical whereas \( \circ \cdot \circ \) is canonical. 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.

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

\[
\begin{align*}
\circ \circ \sqcup \circ \circ \circ & \cong \circ \circ \circ \circ \circ \\
\cdot \circ \circ \sqcup \cdot \circ \circ \circ & \cong \cdot \circ \circ \circ \circ \circ \\
\circ \circ \sqcap \circ \circ \circ & \cong \circ \circ \circ \circ \circ \\
\cdot \circ \circ \sqcap \cdot \circ \circ \circ & \cong \cdot \circ \circ \circ \circ \circ \\
\end{align*}
\]

Complementation is simpler, since flipping leaves between \( \circ \) and \( \cdot \) does not affect whether a tree is in canonical form, e.g.: \( \circ \circ \circ = \circ \circ \circ \). Using these definitions
we get all of the usual properties for Boolean algebras, e.g. $\tau_1 \sqcap \tau_2 = \tau_1 \sqcup \tau_2$. Moreover, we can define a partial ordering between trees using intersection in the usual way, i.e. $\tau_1 \sqsubseteq \tau_2 \overset{\text{def}}{=} \tau_1 \cap \tau_2 = \tau_1$. We can enjoy a strict partial order as well: $\tau_1 \sqsubset \tau_2 \overset{\text{def}}{=} \tau_1 \subseteq \tau_2 \land \tau_1 \neq \tau_2$. We recall a useful result from [18]:

**Proposition 5 ([18])**. *The structure $\mathcal{M} = (T, \sqcup, \sqcap, \bar{\Box}, \circ)$ is a countable atomless Boolean Algebra.*

Properties of tree multiplication $\triangleright$. Since it is nonstandard, the “tree multiplication” operator $\triangleright$ deserves some additional attention. The good news first: $\triangleright$ is associative, has an identity $\bullet$, and is injective for non-$\circ$ elements, i.e. $\mathcal{M}^+ \overset{\text{def}}{=} (T \setminus \{\circ\}, \triangleright)$ forms a cancellative monoid. Somewhat unsurprisingly, multiplication by the “additive identity” $\circ$ reduces to $\circ$. Unfortunately, $\triangleright$ is not commutative ($\circ \triangleright \bullet \overset{\text{def}}{=} \circ \triangleright \circ \neq \circ \triangleright \bullet$), although we do enjoy a distributive property over $\sqcup$ and $\sqcap$ on the right hand side. Accordingly:

**Proposition 6 (Properties of $\triangleright$).**

- **Associativity**: $\tau_1 \triangleright (\tau_2 \triangleright \tau_3) = (\tau_1 \triangleright \tau_2) \triangleright \tau_3$ (1)
- **Identity element**: $\tau \triangleright \bullet = \circ \triangleright \tau = \tau$ (2)
- **Zero element**: $\tau \triangleright \circ = \circ \triangleright \tau = \circ$ (3)
- **Left cancellation**: $\tau \neq \circ \Rightarrow \tau \triangleright \tau_1 = \tau \triangleright \tau_2 \Rightarrow \tau_1 = \tau_2$ (4)
- **Right cancellation**: $\tau \neq \circ \Rightarrow \tau_1 \triangleright \tau = \tau_2 \triangleright \tau \Rightarrow \tau_1 = \tau_2$ (5)
- **Right distributivity over $\sqcap$**: $\tau_1 \triangleright (\tau_2 \sqcap \tau_3) = (\tau_1 \triangleright \tau_2) \sqcap (\tau_1 \triangleright \tau_3)$ (6)
- **Right distributivity over $\sqcup$**: $\tau_1 \triangleright (\tau_2 \sqcup \tau_3) = (\tau_1 \triangleright \tau_2) \sqcup (\tau_1 \triangleright \tau_3)$ (7)

We adopt some notations hereafter: the tree domain is denoted as $T$ while the set of tree lists is denoted as $\mathcal{L}(T)$. We use the symbol nil for empty list, $[e_1, \ldots, e_n]$ for list of $n$ elements $e_1, \ldots, e_n$ and $l_1 \# l_2$ for the concatenation of two lists $l_1, l_2$. We let $|\tau|$ be the height of the tree $\tau$, starting from 0 and $|\Phi|$ be the height of the formula $\Phi$, which is the largest height of tree constants in $\Phi$ (0 otherwise).

## 3 Complexity of $\mathcal{M} = (T, \sqcup, \sqcap, \bar{\Box})$

Although $\mathcal{M}$ is countable atomless Boolean Algebra which is known to be decidable, formulae of $\mathcal{M}$ contain tree constants which do not belong to the Boolean Algebra language. Fortunately, by analyzing the semantics of $\mathcal{M}$, we can transform a tree formula $\Phi$ into an equivalent tree formula $\Phi'$ in polynomial time such that $\Phi'$ contains only $\bullet$ and $\circ$ as constants. As a result, we can interpret $\bullet$ as 1 and $\circ$ as 0 in the language of Boolean Algebra and thus $\Phi'$ can be decided in the complexity class of atomless Boolean Algebra. More precisely, we will show the following complexity results of $\mathcal{M}$:
Theorem 1. The complexity of $\text{Th}(\mathcal{M})$ is $\text{STA}(\ast, 2^O(n), n)$-complete.

While the transformation is straightforward, it complexity requires certain measurement definitions to be formally analyzed. To begin with, we introduce tree shape, which is basically the skeleton of the tree without leaves:

**Definition 1 (Tree shape).** The shape of a tree $\tau$, denoted by $\langle \tau \rangle$, is obtained by replacing its leaves with $\ast$:

$$
\langle e \rangle \overset{\text{def}}{=} \ast, \quad e \in \{\bullet, \circ\} \\
\langle \underbrace{\tau_1 \tau_2} \rangle \overset{\text{def}}{=} \langle \tau_1 \rangle \langle \tau_2 \rangle
$$

The set of tree shapes is denoted by $\mathbb{S}$. Let $s_1$ and $s_2$ be two tree shapes. Their combined shape, denoted by $s_1 \sqcup s_2$, is a tree shape such that:

$$
\ast \sqcup s_2 \overset{\text{def}}{=} s_2 \\
\ast \sqcup s_1 \overset{\text{def}}{=} s_1 \\
\langle \tau_1 \rangle \sqcup \langle \tau_2 \rangle \overset{\text{def}}{=} \langle \tau_1 \rangle \sqcup \langle \tau_2 \rangle
$$

Note that shapes are not folded into any canonical form (if they were then the only one would be $\ast$).

Furthermore, a shape $s_1$ is included in $s_2$, denoted by $s_1 \sqsubseteq s_2$, if $s_2 = s_1 \sqcup s_2$.

The shape of formula $\Phi$, denoted by $\langle \Phi \rangle$, is the combined shape of all tree shapes in $\Phi$, i.e. if $T = \{\tau_1, \ldots, \tau_n\}$ is the set of tree constants in $\Phi$ then:

$$
\langle \Phi \rangle \overset{\text{def}}{=} \bigcup_{i=1}^{n} \langle \tau_i \rangle \\
\langle \Phi \rangle \overset{\text{def}}{=} \ast \quad \text{if} \ T = \emptyset
$$

Using tree shape, we are able to define the size of trees and formulas, which is essential for measuring the size of the input:

**Definition 2 (Tree size).** The size of a tree shape $s$, denoted by $\|s\|$, is the number of its leaves:

$$
\|\ast\| \overset{\text{def}}{=} 1 \\
\|s_1 \sqcup s_2\| \overset{\text{def}}{=} \|s_1\| + \|s_2\|
$$

Meanwhile, the size of a tree $\tau$, denoted by $\|\tau\|$, is the number of its leaves and internal nodes:

$$
\|\bullet\| = \|\circ\| \overset{\text{def}}{=} 1 \\
\|\tau_1 \tau_2\| \overset{\text{def}}{=} \|\tau_1\| + \|\tau_2\| + 1
$$

The size of a formula $\Phi$ in $\mathcal{M}$, denoted by $\|\Phi\|$, is defined recursively:

$$
\|v\| \overset{\text{def}}{=} 1, \text{ }v \text{ is a variable} \\
\|\pi_1 = \pi_2\| \overset{\text{def}}{=} \|\pi_1\| + \|\pi_2\| \\
\|\pi_1 \ast \pi_2 = \pi_3\| \overset{\text{def}}{=} \|\pi_1\| + \|\pi_2\| + \|\pi_3\| \quad \text{for} \ \ast \in \{\sqcup, \sqsubseteq\} \\
\|\Phi_1 \ast \Phi_2\| \overset{\text{def}}{=} \|\Phi_1\| + \|\Phi_2\| + 1 \quad \text{for} \ \ast \in \{\land, \lor, \to\} \\
\|\neg \Phi\| \overset{\text{def}}{=} \|\Phi\| + 1 \\
\|Qv.\Phi\| \overset{\text{def}}{=} \|\Phi\| + 1 \quad \text{for} \ Q \in \{\forall, \exists\}
$$

Note that the size of a tree $\|\tau\|$ is not the same as the height of a tree $|\tau|$. 

Lemma 1. Let $s_1, \ldots, s_n$ be tree shapes then the size of their combined shape is at most the sum of their sizes:

$$\| \bigcup_{i=1}^{n} s_i \| \leq \sum_{i=1}^{n} \| s_i \|$$

Proof. When $n = 1$ the result is trivial. The general case follows from induction and the base case when $n = 2$, i.e., $\| s_1 \sqcup s_2 \| < \| s_1 \| + \| s_2 \|$ which can be done by induction on $s_1$. □

Lemma 2. Let $\Phi$ be a formula then the size of its shape is at most its size, i.e.,

$$\| \langle \Phi \rangle \| \leq \| \Phi \|$$

Proof. If $\Phi$ does not contain any constant then $\| \langle \Phi \rangle \| = \| * \| = 1$ and thus the inequality trivially holds given the fact that $\| \Phi \| \geq 2$. Otherwise, let $T = \{ \tau_1, \ldots, \tau_n \}$ be the set of tree constants in $\Phi$ then by Lemma 1, we have:

$$\| \langle \Phi \rangle \| = \| \bigcup_{i=1}^{n} \langle \tau_i \rangle \| \leq \sum_{i=1}^{n} \| \langle \tau_i \rangle \| \leq \| \Phi \|$$

□

Next, we define $\text{Shape\_decompose} : T \times S \mapsto \mathcal{L}(T)$ (Alg. 1) that takes a tree $\tau$ (or a variable $v$) with a shape $s$ and then decomposes $\tau$ into subtrees (or makes many copies of $v$ with different names) according to $s$. For example, let $\tau = \bullet \circ \bullet$ and $s = \star \star \star \star$ then $\text{Shape\_decompose}(\tau, s) = [\bullet, \bullet, \circ, \circ]$ and $\text{Shape\_decompose}(v, s) = [v_00, v_01, v_1]$. We extend the domain $T$ to the $n$-dimension domain $T^n$ and let $\mathcal{M}^n = (T^n, \sqcup_n, \sqcap_n, \Box_n)$ in which $\sqcup_n, \sqcap_n, \Box_n$ are defined by applying $\sqcup, \sqcap, \Box$ component-wise respectively. It is not hard to verify $\mathcal{M}^n$ is also a countable atomless Boolean Algebra and thus by Prop. 1, it follows that $\mathcal{M}$ and $\mathcal{M}^n$ are isomorphic. Additionally, we can construct an effective isomorphism between them:

Lemma 3. For a fix shape $s$ s.t. $\| s \| = n$, $\text{Shape\_decompose}(\_\_ s) : T \mapsto T^n$ is an isomorphism from $T$ to $T^n$.

Proof. By induction on the structure of $s$. □

Since the result list of $l = \text{Shape\_decompose}(\tau, s)$ contains subtrees of $\tau$, their heights are strictly smaller than $|\tau|$ if $|\tau| > 0$ and $s \neq *$. Moreover, if we choose $s$ sufficiently large then $l$ will contain only subtrees of height zero:

Lemma 4. Let $\tau$ be a tree and $s$ a shape s.t. $\langle \tau \rangle \subseteq s$ then all trees in $\text{Shape\_decompose}(\tau, s)$ have height zero.

Proof. By induction on the structure of $s$. □
Algorithm 1 Flatten a formula into an equivalent formula of height zero

1: function Shape_decompose(π, s)
Require: π is either a variable or a tree constant and s is a shape
Ensure: Return a list of subtrees of π by decomposing π according to shape s
2: if s = ∗ then return [π]
3: else let s = s₁ s₂ in
4: if π = v is a variable then
5: return Shape_decompose(v₀, s₁) # Shape_decompose(v₁, s₂)
6: else if π ∈ {→, ⊓} then
7: return Shape_decompose(π₁, s₁) # Shape_decompose(π₂, s₂)
8: else let π = π₁ π₂ in
9: return Shape_decompose(π₁, s₁) # Shape_decompose(π₂, s₂)
10: end if
11: end if
12: end function

14: function Atomic_decompose(Φ, s)
Require: Φ is an atomic formula and s is a shape
Ensure: Return an equivalent formula of height zero
15: if |Φ| = 0 then return Φ
16: else if Φ is π₁ = π₂ then
17: let Shape_decompose(πᵢ, ⟨Φ⟩) := [πᵢ¹, . . . , πᵢⁿ] for i ∈ {1, 2}
18: return \bigwedge_{i=1}^{n} πᵢ¹ = πᵢ₂
19: else if Φ is π₁ * π₂ = π₃ for * ∈ {⊔, ⊓} then
20: let Shape_decompose(πᵢ, ⟨Φ⟩) := [πᵢ¹, . . . , πᵢⁿ] for i ∈ {1, 2, 3}
21: return \bigwedge_{i=1}^{n} πᵢ¹ * πᵢ₂ = π₃
22: end if
23: end function

25: function Flatten(Φ)
Require: Φ is a sentence
Ensure: Return an equivalent sentence Φ’ s.t. → and • are the only constants.
26: if |Φ| = 0 then return Φ
27: else
28: for each atomic sub-formula Ψ in Φ do
29: replace Ψ with Atomic_decompose(Ψ, ⟨Φ⟩)
30: end for
31: for each quantifier Qv in Φ do
32: let Shape_decompose(v) := [v₁, . . . , vₙ]
33: replace Qv with Qv₁ . . . Qvₙ
34: end for
35: let Φ’ be the new formula after the replacement
36: return Φ’
37: end if
38: end function
We now explain how to decompose a tree formula $\Phi$ of size $n = \|\Phi\|$ using \textsc{Shape\_decompose}. First, we compute $s = \langle \Phi \rangle$, the shape of $\Phi$, which can be done in $O(n)$ by collectively combining all the tree shapes in $\Phi$. Next, for each atomic sub-formula $\Psi$ in $\Phi$, we replace it with the conjunction $\bigwedge_{i=1}^{\|s\|} \Psi_i$ by applying \textsc{Shape\_decompose} to each term in $\Psi$ and then combining them component-wise. This process is captured inside the function \textsc{Flatten} that utilizes a sub-routine \textsc{Atomic\_decompose} (Alg. 1). The correctness of \textsc{Flatten} follows from Lemmas 2, 3 and 4:

**Lemma 5.** Let $\Phi$ be a tree formula then $\Phi' = \textsc{Flatten}(\Phi)$ is a formula of height zero that is equivalent to $\Phi$. Furthermore, the time complexity of \textsc{Flatten} is $O(n^2)$ for $n = \|\Phi\|$ is the size of $\Phi$.

**Proof.** The only nontrivial claim is the time complexity of \textsc{Flatten} which can be measured in term of the length of the transformed formula. Let $s$ be the shape of $\Phi$, i.e., $s = \langle \Phi \rangle$, and $n = \|s\|$ then for each variable $v$, \textsc{Shape\_decompose}($v$, $s$) is a list of $n$ variables. For atomic formula $\Psi$ s.t. \textsc{Atomic\_decompose}($\Psi, s$) $= \bigwedge_{i=1}^{\|s\|} \Psi_i$, we have $\|\Psi_i\| = O(\|\Psi\|)$ and thus $\|\bigwedge_{i=1}^{\|s\|} \Psi_i\| = O(n\|\Psi\|)$. As a result, $\|\Phi\|$ and $\|\Phi'\|$ differ by a factor of $O(n)$, i.e., $\|\Phi'\| = O(n\|\Phi\|)$. By Lemma 2, we have $n = \|\langle \Phi \rangle\| \leq \|\Phi\|$ and hence $\|\Phi'\| = O(\|\Phi\|^2)$.

**Example 1.** Let $\Phi \defeq \forall a \exists b. a \lor b = \textcircled{\bullet} \textcircled{\circ} \lor \neg (a = \textcircled{\bullet} \textcircled{\circ})$ then $\langle \Phi \rangle = \textcircled{\bullet} \lor \textcircled{\circ} = \textcircled{\bullet} \textcircled{\circ}$. There are two atomic sub-formulas, namely $\Psi_1 \defeq a \lor b = \textcircled{\bullet} \textcircled{\circ}$ and $\Psi_2 \defeq a = \textcircled{\bullet} \textcircled{\circ}$. Thus, we have:

$$\begin{align*}
\Psi_1' \defeq &\text{\textsc{Atomic\_decompose}}(\Psi_1, \langle \Phi \rangle) \\
= &a_{00} \lor b_{00} = \textcircled{\bullet} \land a_{01} \lor b_{01} = \textcircled{\bullet} \land a_{10} \lor b_{10} = \textcircled{\bullet} \land a_{11} \lor b_{11} = \textcircled{\circ} \\
\Psi_2' \defeq &\text{\textsc{Atomic\_decompose}}(\Psi_2, \langle \Phi \rangle) \\
= &a_{00} = \textcircled{\bullet} \land a_{01} = \textcircled{\circ} \land a_{10} = \textcircled{\bullet} \land a_{11} = \textcircled{\circ}
\end{align*}$$

As a result, the transformed formula of height zero is

$$\Phi' = \forall a_{00} \forall a_{01} \forall a_{10} \forall a_{11} \exists b_{00} \exists b_{01} \exists b_{10} \exists b_{11}. (\Psi_1' \lor \neg (\psi_2'))$$

**Corollary 1.** The complexity of $\Sigma_1 \cap \text{Th}(M)$ is \textsc{NP}\text{-}complete.

**Proof.** Notice that \textsc{Flatten} does not increase the number of quantifier alternations in the formula and thus by Prop. 2, $\Sigma_1 \cap \text{Th}(M)$ is \textsc{NP}\text{-}complete.

We are now ready to justify the correctness of Theorem 1:

**Proof of Theorem 1.** By Prop. 3, it suffices to show that the complexity of $\text{Th}(M)$ is exactly the same as the complexity of the class atomless Boolean algebra. By Prop. 1, any atomless Boolean Algebra model is elementary equivalent to $M$. Using Lemma 5, we can transform, in polynomial time, a tree formula of $M$ into an equivalent formula that uses only $\textcircled{\bullet}, \textcircled{\circ}$ as constants and thus is a Boolean formula. As a result, the Turing machine that decides the theory of atomless Boolean algebra can be used to decide $\text{Th}(M)$ and vice versa.
4 Complexity of $\mathcal{R} = (T, \tau \times, \times_{\tau})$

Multiplication operator $\times$ is more complex than the Boolean operators. In particular, the first-order theory over $(T, \times)$ is undecidable, although its existential theory is decidable in $\text{PSPACE}$ [18]. Accordingly, we are interested in a restriction of that theory that will recover decidability for first-order reasoning. Inspired by the notion of “semiautomatic structures” [13], we restrict $\times$ to take only constants as one operand, obtaining the two families of unary operators indexed by constants $\tau$: $\tau \times_{\tau}(x) \overset{\text{def}}{=} \tau \times x$ and $x \times_{\tau}(x) \overset{\text{def}}{=} x \times \tau$.

In this section, we will show that the first-order theory of $\mathcal{R}$ is elementary:

**Theorem 2.** The complexity of $\text{Th}(\mathcal{R})$ can be decided in $2\text{EXSPACE}$.

We prove Theorem 2 by solving a similar problem in which $\cdot, \circ$ are excluded from the domain $T$. That is, let $T^+ = T \setminus \{\cdot, \circ\}$ and $\mathcal{R}^+ = (T^+, \tau \times, \times_{\tau})$, we want:

**Lemma 6.** The complexity of $\text{Th}(\mathcal{R}^+)$ can be decided in $2\text{EXSPACE}$.

The proof of Theorem 2 is derived from Lemma 6 by ‘guessing’ the values of variables. In particular, we partition the tree domain $T$ into three sets $S_0 = \{\circ\}, S_1 = \{\cdot\}$ and $S_2 = T^+$ and use a ternary vector of length $n$ to ‘guess’ the partition domain of $n$ variables in the input formula, e.g., $i \mapsto S_i$ for $i = 0, 1, 2$. If a variable $v$ is ‘guessed’ to be in $S_0$ or $S_1$, we substitute $v$ with either $\circ$ or $\cdot$ respectively. Next, each term $\times_{\tau}(a)$ or $\tau \times_{\tau}(a)$ that contains $\cdot$ or $\circ$ is simplified using the following identities:

\[
\tau \times \cdot = \cdot \times \tau = \tau \quad \tau \times \circ = \circ \times \tau = \circ
\]

After this step, all the atomic sub-formulas that contain $\circ$ or $\cdot$ are trivial equalities that can be replaced by either $\top$ or $\bot$. As a result, the new equivalent formula is free of $\cdot$ and $\circ$ and all variables are belong to $T^+$ which can be solved by the Turing machine that decides $\text{Th}(\mathcal{R}^+)$. The whole guessing process can be done in $\text{PSPACE}$ and thus will not increase the total complexity of $\text{Th}(\mathcal{R})$.

The rest of this section is devoted for the proof of Lemma 6. To prove the complexity $\text{Th}(\mathcal{R}^+)$, we construct a polynomial-time reduction from $\text{Th}(\mathcal{L})$ to the structure $\mathcal{T}$ of binary trees with prefix and suffix successors. As shown in Lemma 8, the complexity of $\text{Th}(\mathcal{T})$ is in $2\text{EXSPACE}$ which becomes the upper bound for $\text{Th}(\mathcal{R})$. To begin with, we recall some key results from [18] that establishes an isomorphism between trees and strings in word equation:

**Definition 3 ([18]).** A tree $\tau$ in $T \setminus \{\circ, \cdot\}$ is prime if for all $\tau_1, \tau_2 \in T$, $\tau = \tau_1 \times \tau_2$ implies either $\tau_1$ or $\tau_2$ is $\cdot$.

**Proposition 7 ([18]).** Each tree in $T \setminus \{\circ, \cdot\}$ is uniquely represented as a sequence of prime trees $\{\tau_i\}_{i=1}^n$ s.t. $\tau = \tau_1 \times \cdots \times \tau_n$. As a result, each tree in $T \setminus \{\circ, \cdot\}$ can be treated as a string in a word equation in which the alphabet is $\mathbb{P}$, the countably infinite set of prime trees, and $\times$ is the string concatenation.
We show how to encode trees using binary strings. Since \( \mathbb{F} \) is countably infinite, we can find a bijective index function \( I : \mathbb{P} \mapsto \mathbb{N} \) that maps each prime tree to a natural. The mapping \( \hat{I} \) from \( \mathbb{T}^+ \) to \( \{0, 1\}^* \) is constructed from \( I \) by:

**Definition 4.** Let \( \hat{I} : \mathbb{T}^+ \mapsto \{0, 1\}^* \) be a mapping s.t.:

1. For each prime tree \( \tau \), \( \hat{I}(\tau) = 1^I(\tau) \) (with \( 1^0 = \epsilon \), the empty string).
2. For any tree \( \tau \in \mathbb{T}^+ \) s.t. \( \tau = \tau_1 \uplus \ldots \uplus \tau_n \) for \( \tau_i \in \mathbb{P} \), we represent \( \tau \) with \( \hat{I}(\tau_1)0 \ldots 0\hat{I}(\tau_n) \), i.e., \( 0 \) is the delimiter between two prime trees.

**Lemma 7.** The mapping \( \hat{I} \) is bijective and for two trees \( \tau_1, \tau_2 \in \mathbb{T}^+ \), we have \( \hat{I}(\tau_1 \uplus \tau_2) = \hat{I}(\tau_1)\hat{I}(\tau_2) \).

**Proof.** Follow directly from the unique representation property of prime trees.

We now develop an automatic structure that will be isomorphic to the theory \( \mathbb{R}^+ \). Let \( \mathcal{T} = (\{0, 1\}^*, \mathcal{S}_0, \mathcal{S}_1, \mathcal{P}_0, \mathcal{P}_1) \) be the structure of binary strings with two prefix and suffix successors, i.e., the universe is the set of binary strings and for each string \( s \in \{0, 1\}^* \), \( \mathcal{S}_0(s) = s0, \mathcal{S}_1(s) = s1, \mathcal{P}_0(s) = 0s \) and \( \mathcal{P}_1(s) = 1s \).

**Lemma 8.** \( \mathcal{T} \) is an automatic structure with bounded degree of 9. As a result, the complexity of \( \text{Th}(\mathcal{T}) \) is in \( \text{2EXSPACE} \).

**Proof.** The domain \( \{0, 1\}^* \) is a regular language while \( \mathcal{S}_i(s) \) (\( \mathcal{P}_i(s) \)) simply means “append the letter \( i \) at the end (beginning) of the string \( s \)” and thus can be computed by finite automaton. As a result, \( \mathcal{T} \) is an automatic structure. For a binary string \( s \), there are at most 9 strings adjacent to its in the graph \( G(\mathcal{T}) \), namely one from equality and two from each injective successor. Consequently, \( \mathcal{T} \) has bounded degree of 9. The complexity of \( \text{Th}(\mathcal{T}) \) is from Prop. 4.

Now let us extend \( \mathcal{T} \) to handle fixed constant prefixes and suffixes, as follows:

**Definition 5.** Let \( s_1, \ldots, s_n \in \{0, 1\} \) and \( s = s_1 \ldots s_n \) a binary string, the \( s \)-prefix function \( \mathcal{P}_s \) and \( s \)-suffix function \( \mathcal{S}_s \) are definable in \( \mathcal{T} \) with linear size:

1. \( \mathcal{P}_s(s') \) def \( \exists v_1 \ldots \exists v_n. v_1 = s_n \land \left( \bigwedge_{i=1}^{n-1} v_{i+1} = \mathcal{P}_{s_{n-i}}(v_i) \right) \land s' = v_n \).
2. \( \mathcal{S}_s(s') \) def \( \exists v_1 \ldots \exists v_n. v_1 = s_1 \land \left( \bigwedge_{i=1}^{n-1} v_{i+1} = \mathcal{S}_{s_{i+1}}(v_i) \right) \land s' = v_n \).

Now let \( \hat{\mathcal{T}} \) be the extended structure of \( \mathcal{T} \) in which \( s \)-prefix functions \( \mathcal{P}_s \) and \( s \)-suffix functions \( \mathcal{S}_s \) are included. Then, since the definitions of \( \mathcal{P}_s \) and \( \mathcal{S}_s \) are linear in the size of \( s \), the complexity of \( \text{Th}(\hat{\mathcal{T}}) \) is the same as \( \text{Th}(\mathcal{T}) \), which is in \( \text{2EXSPACE} \).

We are now ready to construct the isomorphism from \( \mathbb{R}^+ \) to \( \hat{\mathcal{T}} \) whose correctness is direct from Lemma 7:

**Lemma 9.** Let \( M \) be a mapping from \( \mathbb{R}^+ \) to \( \hat{\mathcal{T}} \) s.t.:

1. For each tree \( \tau \in \mathbb{T}^+ \), we let \( M(\tau) = \hat{I}(\tau) \).
2. For each function \( \triangleright_{\tau} \), we let \( M(\triangleright_{\tau}) = \mathcal{S}_{\hat{I}(\tau)} \).


3. For each function $\tau \triangleleft \triangledown$, we let $M(\tau \triangleleft \triangledown) = P_{I(\tau)}$.

Then $M$ is an isomorphism.

There is one technical issue left: the mapped binary string should not have exponential length with respect to the size of the input tree. This can be done by constructing the index function $I$ after observing the input formula. In detail, for a tree formula $\Phi$ of $R$, we first factorize all its tree constants into prime trees, which is in $\text{PTIME}$ [18]. Suppose the formula contains $n$ prime trees $\{\tau_i\}_{i=0}^{n-1}$ then we use the most efficient indexing, e.g., we let $I(\tau_i) = i$ and thus the size $\|\tau_i\|$ and the length of $\hat{I}(\tau)$ differ by a factor of $O(n)$. Since a tree $\tau_1 \triangledown \ldots \triangledown \tau_n$ only needs $O(\sum_{i=1}^{n} \hat{I}(\tau_i))$ to represent, its size $\|\tau\|$ and the length of $\hat{I}(\tau)$ also differs by a factor of $O(n)$. Hence, the result follows.

Example 2. Let $\Phi \overset{\text{def}}{=} \forall a \exists b \exists c. a = \triangleleft \triangledown \circ \circ (b) \wedge b = \circ \circ \circ \circ \circ (c)$. First we factorize all tree constants in $\Phi$:

\[
\overset{\circ}{\circ} \circ = \overset{\circ}{\circ} \triangledown \circ \circ, \quad \overset{\circ}{\circ} \circ = \overset{\circ}{\circ} \triangledown \circ \circ
\]

Let $I(\circ \circ) = 0$ and $I(\overset{\circ}{\circ}) = 1$ then $\hat{I}(\circ \circ) = 1^0 = \epsilon$ and $\hat{I}(\overset{\circ}{\circ}) = 1^1 = 1$ and the equivalent formula in $\hat{T}$ is:

$\forall a \exists b \exists c. a = S_{01}(b) \wedge b = P_{10}(c)$.

5 Complexity of $\mathcal{K} = (\mathbb{T}, \sqcup, \sqcap, \Box, \triangleleft \triangledown, \Box \triangledown)$

In [18], we show that the first-order theory $\text{Th}(\mathcal{K})$ is decidable by providing a tree automatic representation of $\mathcal{K}$. Since the complexity of tree automatic structures are non-elementary in general, no useful upper bound for $\mathcal{K}$ was derived at that point. In this section, we will show that the complexity of $\text{Th}(\mathcal{K})$ is non-elementary by reducing the binary tree structure with prefix relation [7] into $\mathcal{K}$ which is well-known to be non-elementary:

**Theorem 3.** The complexity of $\text{Th}(\mathcal{K})$ is non-elementary.

First, we recall the definition and complexity result of binary tree structure with prefix relation:

**Definition 6.** Let $\mathcal{B} = ([0,1]^*, S_0, S_1, \preceq)$ be the binary tree structure in which $\{0,1\}^*$ is the set of binary strings, $S_i$ is the successor function s.t. $S_i(s) = si$ and $\preceq$ is the binary prefix relation, i.e., $x \preceq y$ iff there exists $z$ such that $xz = y$.

**Proposition 8 ([7, 23]).** The first-order theory of $\mathcal{B}$ is non-elementary.

We proceed to construct the reduction. The main idea is to map the set of strings $\{0,1\}^*$ into the set of unary trees $U(T) \subset T$ that consists of trees with exactly one black leaf, e.g., $\overset{\circ}{\circ}$ and $\overset{\circ}{\circ} \circ$. In detail:
Definition 7. Let $E$ be a mapping from $B$ to $K$ s.t.:

1. $E(\epsilon) \triangleq \bullet$, $E(0) \triangleq \circ \circ$, $E(1) \triangleq \circ \bullet$.
2. $E(s_1 \ldots s_n) \triangleq E(s_1) \Join \ldots \Join E(s_n)$.
3. $E(\sigma_0(s)) \triangleq E(s) \Join E(0) = \circ \circ \Join E(s)$.
4. $E(\sigma_1(s)) \triangleq E(s) \Join E(1) = \circ \bullet \Join E(s)$.
5. $E(x \preceq y) \triangleq E(x) \subseteq E(y)$.

Lemma 10. $E$ is a bijection from $\{0,1\}^*$ to $U(T)$.

Proof. By induction on the length of the binary string. Intuitively, the string $s$ represents the path to the black leaf in the tree $E(s)$ in which 0 and 1 mean ‘left’ and ‘right’ respectively. For example, $E(110) = \circ \bullet \Join \circ \bullet \Join \circ = \circ \bullet \bullet \circ$, which represents the path ‘right, right, left’.

One essential criterion of the reduction is to express the type of $U(T)$ using the signature from $K$. We show that $U(T)$ is expressible using $\Join$ and $\circ$ (recall from §2.4 that $\tau_1 \sqcup \tau_2 \triangleq \tau_1 \cap \tau_2 = \tau_1 \land \tau_1 \neq \tau_2$). Our observation is for a tree $\tau$ in $U(T)$ and any tree $\tau'$ s.t. $\tau' \Join \circ \bullet \cap \tau$, we also have $\tau' \Join \circ \bullet \sqcup \tau$ and vice versa. For example, let $\tau \triangleq \circ \bullet \circ$ and $\tau' = \circ \bullet \circ$ then both $\tau' \Join \circ \bullet \sqcup \tau$ and $\tau' \Join \circ \bullet \sqcup \tau$. This property is not true for trees with at least two black leaves, e.g., let $\tau = \circ \circ \bullet$ and $\tau' = \circ \circ \bullet$ then $\tau' \Join \circ \bullet \sqcup \tau$ but $\tau' \Join \circ \bullet \not\subseteq \tau$. Formally:

Lemma 11. The type $U(T)$ is expressible in $K$ using a $\forall$-formula:

$$\tau \in U(T) \iff \tau \neq \circ \land (\forall \tau'. \Join \circ \bullet \sqcup (\tau') \subseteq \tau \leftrightarrow \Join \circ \bullet (\tau') \subseteq \tau).$$

Proof. For $\Rightarrow$, we prove by induction on the height of $\tau$. The base case $|\tau| = 0$ is trivial. We proceed to prove the case $|\tau| = n + 1$. The proof $\tau \neq \circ$ is easy. Let $\tau'$ s.t. $\Join \circ \bullet (\tau') = \tau' \Join \circ \bullet \subset \tau$ and we want to prove $\tau' \Join \circ \bullet (\tau') = \tau' \Join \circ \bullet \subset \tau$ (the other direction is similar). The case $\tau' = \circ$ is trivial. Otherwise, we have $\tau' \Join \circ \bullet \subset \tau_1$ and thus it must be the case that $\tau' = \tau_1$. Consequently, we have $\tau'_1 \Join \circ \bullet \subset \tau_1$. Using the induction hypothesis for $\tau_1$, we have $\tau'_1 \Join \circ \bullet \subset \tau_1$ and hence $\tau' \Join \circ \bullet \subset \tau_1 \Join \circ \bullet \subset \tau_1 \Join \circ \bullet = \tau$.

For $\Leftarrow$, assume $\tau \notin U(T)$ and since $\tau \neq \circ$, we imply $\tau$ contains at least two black leaves in its representation. Let $\tau_1 \in U(T)$ be the subtree that contains exactly one of the black leaves of $\tau$, i.e. there exists $\tau'_1 \in U(T)$ s.t. either $\tau_1 = \tau'_1 \Join \circ \bullet$ or $\tau_1 = \tau'_1 \Join \circ \bullet$ and exactly one of $\tau'_1 \Join \circ \bullet \subset \tau$ and $\tau'_1 \Join \circ \bullet \subset \tau$ holds. As $\tau$ has at least two black leaves, $\tau_1$ is strictly smaller than $\tau$, i.e., $\tau_1 \subset \tau$. Using the premise, we conclude that both $\tau'_1 \Join \circ \bullet \subset \tau$ and $\tau'_1 \Join \circ \bullet \subset \tau$ which is a contradiction.

\square
The last ingredient is the justification for the prefix relation:

**Lemma 12.** Let $x, y \in \{0, 1\}^*$ then $x \preceq y$ iff $\mathcal{E}(x) \sqsubseteq \mathcal{E}(y)$.

**Proof.** For any two unary trees $\tau_1, \tau_2 \in \mathcal{U}(T)$, we have $\tau_1 \sqsubseteq \tau_2$ iff there exists a unary tree $\tau_3 \in \mathcal{U}(T)$ s.t. $\tau_2 \sloppymerge \tau_3 = \tau_1$. Thus $x \preceq y$ iff $\exists z. \mathcal{E}(x) \sloppymerge \mathcal{E}(z) = \mathcal{E}(y)$ iff $\exists z. \mathcal{E}(x) \sqsubseteq \mathcal{E}(y) \iff \mathcal{E}(y) \sqsubseteq \mathcal{E}(y)$.

**Proof of Theorem 3.** The technique is similar to [9] in which we interpret formulas of $\mathcal{B}$ using the signature of $\mathcal{K}$. The interpretation of constants and symbols is already in Definition 7. Next we replace sub-formula $\exists x. \Phi$ with $\exists x. x \in \mathcal{U}(T) \land \Phi$ and $\forall x. \Phi$ with $\forall x. x \in \mathcal{U}(T) \rightarrow \Phi$. Thus by Lemma 10, 11 and 12, the tree formula in $\mathcal{K}$ is equivalent to the tree formula in $\mathcal{B}$. Hence the first-order complexity of $\mathcal{K}$ is bounded below by the first-order complexity of $\mathcal{B}$. By Prop. 8, we conclude that the first-order complexity of $\mathcal{K}$ is non-elementary. 

6 Future work and conclusion

We have developed a more precise understanding of the complexity of the tree share model. As Boolean algebras, their first-order theory is $\text{STA}(\ast, 2^{O(n)}, n)$-complete, even when arbitrary constants are allowed in the formulae. Although the first-order theory over tree multiplication is undecidable [18], we have found that by restricting multiplication to be by a constant (on both the left and right sides) we obtain a subtheory $\mathcal{R}$ whose first-order theory is decidable in $2\text{EXSPACE}$. Accordingly, we have two theories whose first-order theory is elementary decidable. Unfortunately, their combined theory is at best non-elementary, even if we only allow multiplication by a constant on the right side $\sloppymerge$.

We have several directions for future work. We do not have a nontrivial lower bound on the complexity of the existential theory with the Boolean operators and right-sided multiplication $\sloppymerge$ (structure $\mathcal{K}$) since our encoding of $\mathcal{U}(T)$ in Lemma 11 uses universals. We do not know if the Boolean operators ($\lor, \land, \square$) in combination with the left-sided multiplication $\sloppymerge$ is decidable (existential or first order, with or without the right-sided multiplication $\sloppymerge$). Determining if the existential theory with the Boolean operators and *unrestricted* multiplication $\sloppymerge$ is decidable also seems challenging. We would also like to know if the monadic second-order theory over these structures is decidable.

Despite these remaining questions, our understanding of the structure has improved meaningfully, allowing us to contemplate using it inside practical verification tools. We have already incorporated tree shares and their Boolean structure into the HIP/SLEEK verification toolchain [22, 19] and are actively exploring how to incorporate their multiplicative structure as well.

References


19. Xuan Bach Le, Thanh Toan Nguyen, and Aquinas Hobor. A certified decision procedure for tree shares, 2017. under submission.


