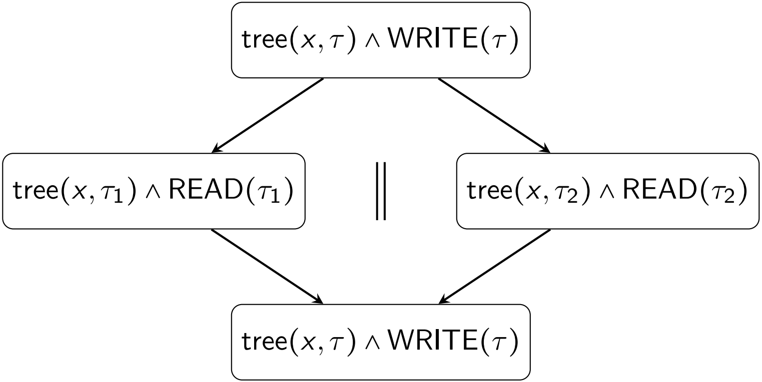
Decidability and Complexity of Tree Share Formulas

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

[1] National University of Singapore [2] University of Oxford

December 14, 2016


$\mathsf{tree}(x, \tau) \land \mathsf{WRITE}(\tau)$
$\mathsf{tree}(x, \tau_1) \land \mathsf{READ}(\tau_1)$
$\|$
$\mathsf{tree}(x, \tau_2) \land \mathsf{READ}(\tau_2)$
$\mathsf{tree}(x, \tau) \land \mathsf{WRITE}(\tau)$

Shares

Shares are embedded into separation logic to reason about resource accounting:

$$\text{addr} \overset{\tau_1 \oplus \tau_2}{\mapsto} \text{val} \quad \Leftrightarrow \quad \text{addr} \overset{\tau_1}{\mapsto} \text{val} \star \text{addr} \overset{\tau_2}{\mapsto} \text{val}$$

Shares

Shares are embedded into separation logic to reason about resource accounting:

$$\mathsf{addr} \overset{\tau_1 \oplus \tau_2}{\mapsto} \mathsf{val} \quad \Leftrightarrow \quad \mathsf{addr} \overset{\tau_1}{\mapsto} \mathsf{val} \star \mathsf{addr} \overset{\tau_2}{\mapsto} \mathsf{val}$$

- Allow resources to be split and shared in large scale:

$$\mathsf{tree}(\ell, \tau) \quad \overset{\mathtt{def}}{=} \quad (\ell = \mathsf{null} \ \wedge \mathsf{emp}) \ \vee \exists \ell_l, \ell_r.\ (\ell \overset{\tau}{\mapsto} (\ell_l, \ell_r) \star \mathsf{tree}(\ell_l, \tau) \star \mathsf{tree}(\ell_r, \tau))$$

$$\mathsf{tree}(\ell, \tau_1 \oplus \tau_2) \quad \Leftrightarrow \quad \mathsf{tree}(\ell, \tau_1) \star \mathsf{tree}(\ell, \tau_2)$$

Shares

Shares are embedded into separation logic to reason about resource accounting:

$$\text{addr} \overset{\tau_1 \oplus \tau_2}{\mapsto} \text{val} \quad \Leftrightarrow \quad \text{addr} \overset{\tau_1}{\mapsto} \text{val} \star \text{addr} \overset{\tau_2}{\mapsto} \text{val}$$

- Allow resources to be split and shared in large scale:

$$\text{tree}(\ell, \tau) \quad \overset{\text{def}}{=} \quad (\ell = \text{null} \ \wedge \text{emp}) \ \vee \exists \ell_l, \ell_r. \ (\ell \overset{\tau}{\mapsto} (\ell_l, \ell_r) \star \text{tree}(\ell_l, \tau) \star \text{tree}(\ell_r, \tau))$$

$$\text{tree}(\ell, \tau_1 \oplus \tau_2) \quad \Leftrightarrow \quad \text{tree}(\ell, \tau_1) \star \text{tree}(\ell, \tau_2)$$

- Share policies to reason about permissions for single writer and multiple readers:

$$\frac{\text{WRITE}(\tau)}{\text{READ}(\tau)} \ \textsc{Write-Read} \qquad \frac{\text{READ}(\tau)}{\exists \tau_1, \tau_2. \ \tau_1 \oplus \tau_2 = \tau \ \wedge \ \text{READ}(\tau_1) \wedge \text{READ}(\tau_2)} \ \textsc{Split-Read}$$

Shares

Shares enable resource reasoning in concurrent programming

Shares

Shares enable resource reasoning in concurrent programming

- Rational numbers [Boyland (2003)]: disjointness problem makes tree split equivalence false:
  $\neg\big(\mathrm{tree}(\ell, \tau_1 \oplus \tau_2) \Leftarrow \mathrm{tree}(\ell, \tau_1) \star \mathrm{tree}(\ell, \tau_2)\big)$

Shares

Shares enable resource reasoning in concurrent programming

- Rational numbers [Boyland (2003)]: disjointness problem makes tree split equivalence false:
  $\neg\big(\mathrm{tree}(\ell, \tau_1 \oplus \tau_2) \Leftarrow \mathrm{tree}(\ell, \tau_1) \star \mathrm{tree}(\ell, \tau_2)\big)$
- Subsets of natural numbers [Parkinson (2005)]
  - Finite sets: recursion depth is finite
  - Infinite sets: intersections may not be in the model

Tree Share Definition

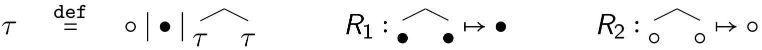
- A tree share $\tau \in \mathbb{T}$ is a boolean binary tree equipped with the reduction rules R_1 and R_2 (their inverses are E_1, E_2 resp.):

$$\tau \stackrel{\text{def}}{=} \circ \mid \bullet \mid \widehat{\tau \quad \tau} \qquad R_1 : \widehat{\bullet \quad \bullet} \mapsto \bullet \qquad R_2 : \widehat{\circ \quad \circ} \mapsto \circ$$

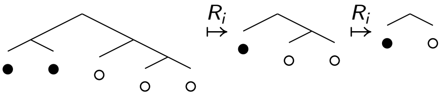
- The tree domain $\mathbb{T}$ contains canonical trees which are irreducible with respect to the reduction rules.

Tree Share Definition

- A tree share $\tau \in \mathbb{T}$ is a boolean binary tree equipped with the reduction rules R_1 and R_2 (their inverses are E_1, E_2 resp.):

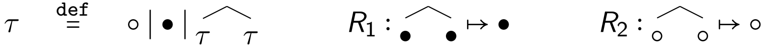


- The tree domain $\mathbb{T}$ contains canonical trees which are irreducible with respect to the reduction rules.

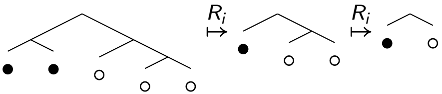


Tree Share Definition

- A tree share $\tau \in \mathbb{T}$ is a boolean binary tree equipped with the reduction rules R_1 and R_2 (their inverses are E_1, E_2 resp.):



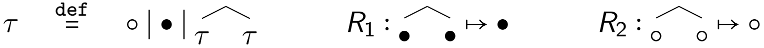
- The tree domain $\mathbb{T}$ contains canonical trees which are irreducible with respect to the reduction rules.



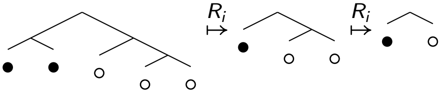
  - ∘ is the empty tree, and • the full tree.

Tree Share Definition

- A tree share $\tau \in \mathbb{T}$ is a boolean binary tree equipped with the reduction rules R_1 and R_2 (their inverses are E_1, E_2 resp.):



- The tree domain $\mathbb{T}$ contains canonical trees which are irreducible with respect to the reduction rules.



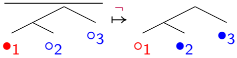
  - $\circ$ is the empty tree, and $\bullet$ the full tree.
  - READ$(\tau) \stackrel{\text{def}}{=} \tau \neq \circ$ WRITE$(\tau) \stackrel{\text{def}}{=} \tau = \bullet$

Tree Share Operators

- The complement $\bar{\square}$:

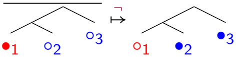
Tree Share Operators

- The complement $\bar{\square}$:



Tree Share Operators

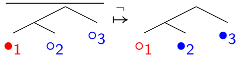
- The complement $\bar{\square}$:



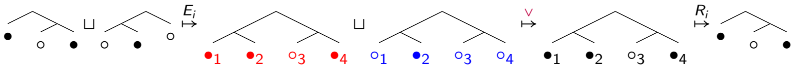
- The Boolean function union $\sqcup$ and intersection $\sqcap$ operator:

Tree Share Operators

- The complement $\overline{\square}$:

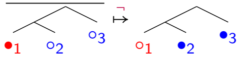


- The Boolean function union $\sqcup$ and intersection $\sqcap$ operator:

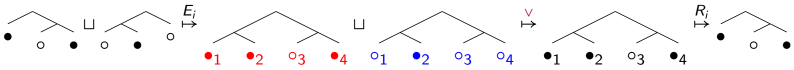


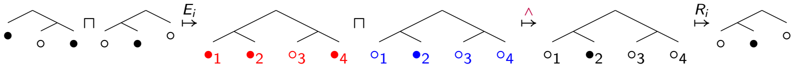
Tree Share Operators

- The complement $\overline{\square}$:



- The Boolean function union $\sqcup$ and intersection $\sqcap$ operator:





Properties of $\sqcup$, $\sqcap$ and $\bar{\Box}$

$\mathcal{M} = (\sqcup, \sqcap, \bar{\Box}, \bullet, \circ)$ forms a Boolean Algebra [Dockins et al. (2009)]:

$B1a.\ (\tau_1 \sqcap \tau_2) \sqcap \tau_3 = \tau_1 \sqcap (\tau_2 \sqcap \tau_3)$	$B1b.\ (\tau_1 \sqcup \tau_2) \sqcup \tau_3 = \tau_1 \sqcup (\tau_2 \sqcup \tau_3)$	(associativity)
$B2a.\ \tau_1 \sqcap \tau_2 = \tau_2 \sqcap \tau_1$	$B2b.\ \tau_1 \sqcup \tau_2 = \tau_2 \sqcup \tau_1$	(commutativity)
$B3a.\ \tau_1 \sqcap (\tau_2 \sqcup \tau_3) = (\tau_1 \sqcap \tau_2) \sqcup (\tau_1 \sqcap \tau_3)$	$B3b.\ \tau_1 \sqcup (\tau_2 \sqcap \tau_3) = (\tau_1 \sqcup \tau_2) \sqcap (\tau_1 \sqcup \tau_3)$	(distributivity)
$B4a.\ \tau_1 \sqcap (\tau_1 \sqcup \tau_2) = \tau_1$	$B4b.\ \tau_1 \sqcup (\tau_1 \sqcap \tau_2) = \tau_1$	(absorption)
$B5a.\ \tau \sqcap \bullet = \tau$	$B5b.\ \tau \sqcup \circ = \tau$	(identity)
$B6a.\ \tau \sqcap \bar{\tau} = \circ$	$B6b.\ \tau \sqcup \bar{\tau} = \bullet$	(complement)

Tree Share Operators(cont.)

The partial join function $\oplus$:

Tree Share Operators(cont.)

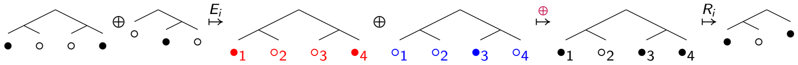
The partial join function $\oplus$:

$$\tau_1 \oplus \tau_2 = \tau_3 \quad \stackrel{\texttt{def}}{=} \quad \tau_1 \sqcup \tau_2 = \tau_3 \wedge \tau_1 \sqcap \tau_2 = \circ$$

Tree Share Operators(cont.)

The partial join function $\oplus$:

$$\tau_1 \oplus \tau_2 = \tau_3 \quad \stackrel{\text{def}}{=} \quad \tau_1 \sqcup \tau_2 = \tau_3 \wedge \tau_1 \sqcap \tau_2 = \circ$$



Properties of $\oplus$

$\mathcal{O} = (\mathbb{T}, \oplus)$ for fractional permission in Separation Logic [Dockins et al. (2009)]:

$J1.\ \tau_1 \oplus \tau_2 = \tau_3 \Rightarrow \tau_1 \oplus \tau_2 = \tau_3' \Rightarrow \tau_3 = \tau_3'$ (functionality)

$J2.\ \tau_1 \oplus \tau_2 = \tau_2 \oplus \tau_1$ (commutativity)

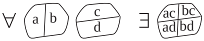
$J3.\ \tau_1 \oplus (\tau_2 \oplus \tau_3) = (\tau_1 \oplus \tau_2) \oplus \tau_3$ (associativity)

$J4.\ \tau_1 \oplus \tau_2 = \tau_3 \Rightarrow \tau_1' \oplus \tau_2 = \tau_3 \Rightarrow \tau_1 = \tau_1'$ (cancellation)

$J5.\ \exists u.\ \forall \tau.\ \tau \oplus u = \tau$ (unit)

$J6.\ \tau_1 \oplus \tau_1 = \tau_2 \Rightarrow \tau_1 = \tau_2$ (disjointness)

$J7.\ a \oplus b = z \wedge c \oplus d = z \Rightarrow \exists ac, ad, bc, bd.$



$ac \oplus ad = a \wedge bc \oplus bd = b \wedge ac \oplus bc = c \wedge ad \oplus bd = d$ (cross split)

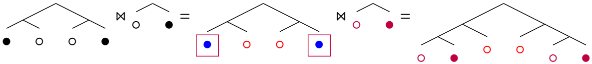
$J8.\ \tau \neq \circ \Rightarrow \exists \tau_1, \tau_2.\ \tau_1 \neq \circ\ \wedge\ \tau_2 \neq \circ\ \wedge\ \tau_1 \oplus \tau_2 = \tau$ (infinite split)

Tree Share Operators(cont.)

The injection bowtie function $\bowtie$ replaces $\bullet$ with tree:

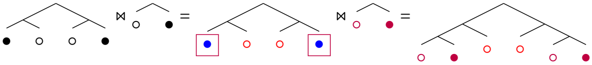
Tree Share Operators(cont.)

The injection bowtie function $\bowtie$ replaces $\bullet$ with tree:



Tree Share Operators(cont.)

The injection bowtie function $\bowtie$ replaces $\bullet$ with tree:



Allow resources to be split uniformly:

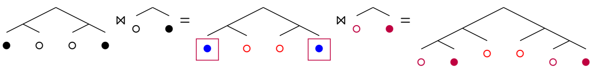
$$\tau_1 \cdot \mathsf{tree}(\ell, \tau_2) \quad \stackrel{\mathtt{def}}{=} \quad \mathsf{tree}(\ell, \tau_2 \bowtie \tau_1)$$

$$(\tau_1 \oplus \tau_2) \cdot \mathsf{tree}(\ell, \tau) \quad \Leftrightarrow \quad \tau_1 \cdot \mathsf{tree}(\ell, \tau) \star \tau_2 \cdot \mathsf{tree}(\ell, \tau)$$

$$\tau_1 \cdot \mathsf{tree}(\ell, \tau_2 \bowtie \tau_3) \quad \Leftrightarrow \quad (\tau_3 \bowtie \tau_1) \cdot \mathsf{tree}(\ell, \tau_2)$$

Tree Share Operators(cont.)

The injection bowtie function $\bowtie$ replaces $\bullet$ with tree:



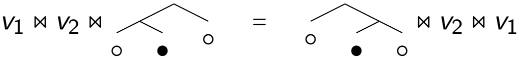
Allow resources to be split uniformly:

$$\tau_1 \cdot \mathrm{tree}(\ell, \tau_2) \overset{\mathrm{def}}{=} \mathrm{tree}(\ell, \tau_2 \bowtie \tau_1)$$

$$(\tau_1 \oplus \tau_2) \cdot \mathrm{tree}(\ell, \tau) \Leftrightarrow \tau_1 \cdot \mathrm{tree}(\ell, \tau) \star \tau_2 \cdot \mathrm{tree}(\ell, \tau)$$

$$\tau_1 \cdot \mathrm{tree}(\ell, \tau_2 \bowtie \tau_3) \Leftrightarrow (\tau_3 \bowtie \tau_1) \cdot \mathrm{tree}(\ell, \tau_2)$$

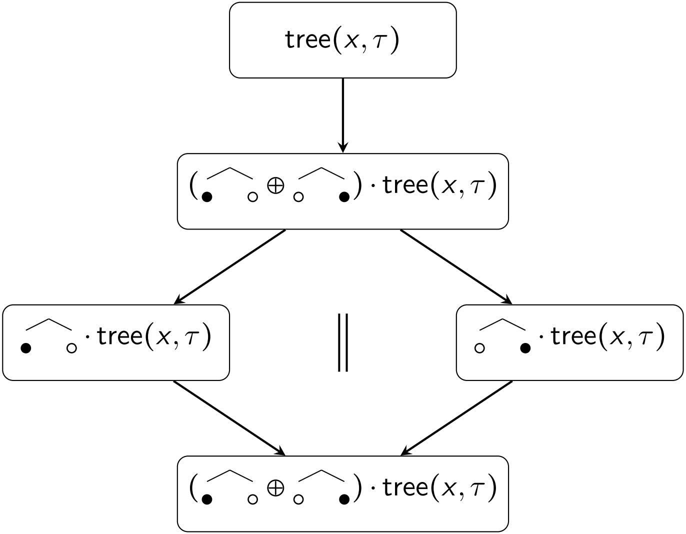
$\bowtie$ can be hard to think about. Is this equation satisfiable?



Properties of $\bowtie$

$\mathcal{S} = (\bowtie, \bullet)$ forms an Monoid with additional properties [Dockins et al. (2009)]:

$M1.\ (\tau_1 \bowtie \tau_2) \bowtie \tau_3 = \tau_1 \bowtie (\tau_2 \bowtie \tau_3)$	(associativity)
$M2.\ \tau \bowtie \bullet = \bullet \bowtie \tau = \tau$	(identity)
$M3.\ \tau \bowtie \circ = \circ \bowtie \tau = \circ$	(collapse point)
$M4.\ \tau_1 \bowtie (\tau_2 \diamond \tau_3) = (\tau_1 \diamond \tau_2) \bowtie (\tau_1 \diamond \tau_3),\ \diamond \in \{\sqcap, \sqcup, \oplus\}$	(distributivity)
$M5.\ \tau \bowtie \tau_1 = \tau \bowtie \tau_2 \Rightarrow \tau \neq \circ \Rightarrow \tau_1 = \tau_2$	(left cancellation)
$M6.\ \tau_1 \bowtie \tau = \tau_2 \bowtie \tau \Rightarrow \tau \neq \circ \Rightarrow \tau_1 = \tau_2$	(right cancellation)


tree(x, τ)
$\oplus$) · tree(x, τ)
· tree(x, τ)
$\|$
· tree(x, τ)
$\oplus$) · tree(x, τ)

Outline

1. Introduction
2. Decidability and Complexity results
   - Model for Countable Atomless Boolean Algebra
   - From $\bowtie$ to string concatenation
   - Tree Automatic Structures
3. Conclusion

Tree Shares as Countable Atomless Boolean Algebra

- $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra because the domain $\mathbb{T}$ is countable.

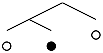
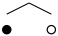
Tree Shares as Countable Atomless Boolean Algebra

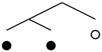
- $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra because the domain $\mathbb{T}$ is countable.
- Atomless properties of $\mathcal{M}$:
  - Let $\tau_1 \neq \tau_2$, we denote $\tau_1 \sqsubset \tau_2$ iff $\tau_1 \sqcup \tau_2 = \tau_2$.

Tree Shares as Countable Atomless Boolean Algebra

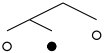
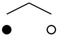
- $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra because the domain $\mathbb{T}$ is countable.
- Atomless properties of $\mathcal{M}$:
  - Let $\tau_1 \neq \tau_2$, we denote $\tau_1 \sqsubset \tau_2$ iff $\tau_1 \sqcup \tau_2 = \tau_2$.
  - $\mathcal{M}$ is atomless if for $\tau_1 \sqsubset \tau_3$, there exists τ_2 such that $\tau_1 \sqsubset \tau_2 \sqsubset \tau_3$.

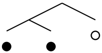
Tree Shares as Countable Atomless Boolean Algebra

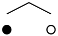
- $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra because the domain $\mathbb{T}$ is countable.
- Atomless properties of $\mathcal{M}$:
  - Let $\tau_1 \neq \tau_2$, we denote $\tau_1 \sqsubset \tau_2$ iff $\tau_1 \sqcup \tau_2 = \tau_2$.
  - $\mathcal{M}$ is atomless if for $\tau_1 \sqsubset \tau_3$, there exists τ_2 such that $\tau_1 \sqsubset \tau_2 \sqsubset \tau_3$.
  - Let $\tau_1 =$  and $\tau_3 =$  then $\tau_1 \sqsubset \tau_3$. We extend τ_3 to the shape of τ_1:

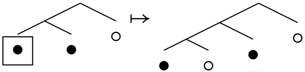
$\tau_3 \overset{E_i}{\mapsto}$ 

Tree Shares as Countable Atomless Boolean Algebra

- $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra because the domain $\mathbb{T}$ is countable.
- Atomless properties of $\mathcal{M}$:
  - Let $\tau_1 \neq \tau_2$, we denote $\tau_1 \sqsubset \tau_2$ iff $\tau_1 \sqcup \tau_2 = \tau_2$.
  - $\mathcal{M}$ is atomless if for $\tau_1 \sqsubset \tau_3$, there exists τ_2 such that $\tau_1 \sqsubset \tau_2 \sqsubset \tau_3$.
  - Let $\tau_1 =$  and $\tau_3 =$  then $\tau_1 \sqsubset \tau_3$. We extend τ_3 to the shape of τ_1:

    $\tau_3 \overset{E_i}{\mapsto}$ 

    then replace one of the $\bullet$ leaves of τ_3 that is not in τ_1 with :

Decidability of $\mathcal{M}$

The first-order theory of $\mathcal{M}$ is decidable. The lower bound for its complexity is $\bigcup_{c<\omega} \mathrm{STA}(*, 2^{cn}, n)$ [Kozen (1980)].

Decidability of $\bowtie$

Decidability of $\mathcal{S} = (\mathbb{T}, \bowtie)$

Let $\mathcal{S} = (\mathbb{T}, \bowtie)$ then:

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

Decidability of $\bowtie$

Decidability of $\mathcal{S} = (\mathbb{T}, \bowtie)$

Let $\mathcal{S} = (\mathbb{T}, \bowtie)$ then:

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

Decidability of $\mathcal{S}^+ = (\mathbb{T} \backslash \{\circ\}, \bowtie)$

Let $\mathcal{S}^+ = (\mathbb{T} \backslash \{\circ\}, \bowtie)$ then:

- The existential theory of $\mathcal{S}^+$ is decidable in PSPACE.
- The existential theory of $\mathcal{S}^+$ is NP-hard.
- The general first-order theory over $\mathcal{S}^+$ is undecidable.

Isomorphism between $\bowtie$ and $\cdot$

To prove these results on $\mathcal{S}^+ = (\mathbb{T}\backslash\{\circ\}, \bowtie)$, we will construct an isomorphism between $\mathcal{S}^+$ equations and word equations.

Isomorphism between $\bowtie$ and $\cdot$

To prove these results on $\mathcal{S}^+ = (\mathbb{T} \backslash \{\circ\}, \bowtie)$, we will construct an isomorphism between $\mathcal{S}^+$ equations and word equations.

In particular, we will transform $\bowtie$ into string concatenation. The trick is that we must find an "alphabet" for $\mathcal{S}^+$ equations.

Review of Word Equations

- Let $\mathcal{A} = \{a, b, \ldots\}$ be the finite set of alphabet and $\mathcal{V} = \{v_1, v_2, \ldots\}$ the set of variables.

Review of Word Equations

- Let $\mathcal{A} = \{a, b, \ldots\}$ be the finite set of alphabet and $\mathcal{V} = \{v_1, v_2, \ldots\}$ the set of variables.
- A word w is a string in $(\mathcal{A} \cup \mathcal{V})^*$. A word equation E is a pair of words $w_1 = w_2$.

Review of Word Equations

- Let $\mathcal{A} = \{a, b, \ldots\}$ be the finite set of alphabet and $\mathcal{V} = \{v_1, v_2, \ldots\}$ the set of variables.
- A word w is a string in $(\mathcal{A} \cup \mathcal{V})^*$. A word equation E is a pair of words $w_1 = w_2$.
- E has a solution if there exists a homomorphism $f : \mathcal{A} \cup \mathcal{V} \mapsto \mathcal{A}^*$ that maps each letter in $\mathcal{A}$ to itself.

Review of Word Equations

- Let $\mathcal{A} = \{a, b, \ldots\}$ be the finite set of alphabet and $\mathcal{V} = \{v_1, v_2, \ldots\}$ the set of variables.
- A word w is a string in $(\mathcal{A} \cup \mathcal{V})^*$. A word equation E is a pair of words $w_1 = w_2$.
- E has a solution if there exists a homomorphism $f : \mathcal{A} \cup \mathcal{V} \mapsto \mathcal{A}^*$ that maps each letter in $\mathcal{A}$ to itself.
- For example, the equation $v_1 v_2 ab = bav_2 v_1$ has a solution:

$$f(v_1) = b, f(v_2) = a$$

Review of Word Equations

- Let $\mathcal{A} = \{a, b, \ldots\}$ be the finite set of alphabet and $\mathcal{V} = \{v_1, v_2, \ldots\}$ the set of variables.
- A word w is a string in $(\mathcal{A} \cup \mathcal{V})^*$. A word equation E is a pair of words $w_1 = w_2$.
- E has a solution if there exists a homomorphism $f : \mathcal{A} \cup \mathcal{V} \mapsto \mathcal{A}^*$ that maps each letter in $\mathcal{A}$ to itself.
- For example, the equation $v_1 v_2 ab = bav_2 v_1$ has a solution:

$$f(v_1) = b, f(v_2) = a$$

- The satisfiability problem of word equation: checking whether a word equation E has a solution.

Word Equation Results

Decidability and Complexity of Word Equation

- The satisfiability problem of word equation is decidable. The lower bound is NP-complete while the upper bound is PSPACE [Plandowski (1999)].

Word Equation Results

Decidability and Complexity of Word Equation

- The satisfiability problem of word equation is decidable. The lower bound is NP-complete while the upper bound is PSPACE [Plandowski (1999)].
- The satisfiability of a system of word equations with regular constraints $v_i \in \mathtt{REG}_i$ can be reduced to the satisfiability of a single word equation [Schulz (1990)].

Word Equation Results

Decidability and Complexity of Word Equation

- The satisfiability problem of word equation is decidable. The lower bound is NP-complete while the upper bound is PSPACE [Plandowski (1999)].
- The satisfiability of a system of word equations with regular constraints $v_i \in \mathtt{REG}_i$ can be reduced to the satisfiability of a single word equation [Schulz (1990)].
- The existential theory of string concatenation is decidable with lower bound NP-complete and upper bound PSPACE. The first-order theory of string concatenation is undecidable (forklore).

Tree factorization

Prime trees

A tree $\tau \in \mathbb{T} \backslash \{\bullet, \circ\}$ is prime iff $\tau = \tau_1 \bowtie \tau_2$ then either $\tau_1 = \bullet$ or $\tau_2 = \bullet$.

Tree factorization

Prime trees

A tree $\tau \in \mathbb{T} \backslash \{\bullet, \circ\}$ is prime iff $\tau = \tau_1 \bowtie \tau_2$ then either $\tau_1 = \bullet$ or $\tau_2 = \bullet$.

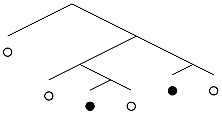
A tree share τ can be *factorized* into *prime trees* using $\bowtie$:

Tree factorization

Prime trees

A tree $\tau \in \mathbb{T} \backslash \{\bullet, \circ\}$ is prime iff $\tau = \tau_1 \bowtie \tau_2$ then either $\tau_1 = \bullet$ or $\tau_2 = \bullet$.

A tree share τ can be *factorized* into *prime trees* using $\bowtie$:

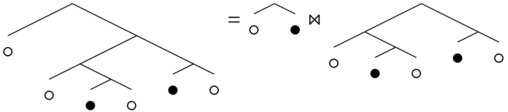


Tree factorization

Prime trees

A tree $\tau \in \mathbb{T} \backslash \{\bullet, \circ\}$ is prime iff $\tau = \tau_1 \bowtie \tau_2$ then either $\tau_1 = \bullet$ or $\tau_2 = \bullet$.

A tree share τ can be *factorized* into *prime trees* using $\bowtie$:

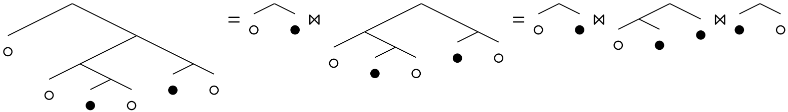


Tree factorization

Prime trees

A tree $\tau \in \mathbb{T} \backslash \{\bullet, \circ\}$ is prime iff $\tau = \tau_1 \bowtie \tau_2$ then either $\tau_1 = \bullet$ or $\tau_2 = \bullet$.

A tree share τ can be *factorized* into *prime trees* using $\bowtie$:



Tree factorization(cont.)

Unique factorization

Let $\tau \in \mathbb{T}\backslash\{\circ, \bullet\}$ then there exists a unique sequence of prime trees $\tau_1, \ldots, \tau_n$ such that:

$$\tau = \tau_1 \bowtie \ldots \bowtie \tau_n$$

Furthermore, the factorization problem is in PTIME.

Proof sketch: By induction on the structure of the tree.

Infinite alphabet

- Let $\mathbb{T}_p \subset \mathbb{T}$ be the set of prime trees then $\mathbb{T}_p$ is countably infinite.

Infinite alphabet

- Let $\mathbb{T}_p \subset \mathbb{T}$ be the set of prime trees then $\mathbb{T}_p$ is countably infinite.
- $\mathbb{T}_p$ is our *alphabet* for the word equation but we need to reduce it to finite alphabet.

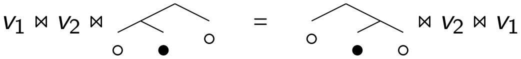
Infinite alphabet(cont.)

From infinity to finite

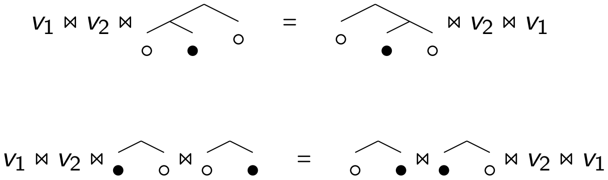
Let Σ be the set of word equations and inequations over infinite alphabet $\mathcal{A}$ then Σ has a solution iff it has a solution over some finite alphabet $\mathcal{B} \subset \mathcal{A}$ such that:

1. $\mathcal{A}(\Sigma) \subset \mathcal{B}$
2. $|\mathcal{B}| = |\mathcal{A}(\Sigma)| + n$ where n is the number of inequations in Σ. The choice of the extra letters in $\mathcal{B}$ is not important.

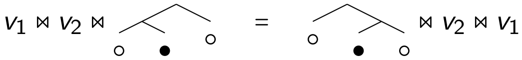
Example

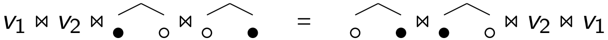


Example


$v_1 \bowtie v_2 \bowtie$ = $\bowtie v_2 \bowtie v_1$
$v_1 \bowtie v_2 \bowtie$ $\bowtie$ = $\bowtie$ $\bowtie v_2 \bowtie v_1$

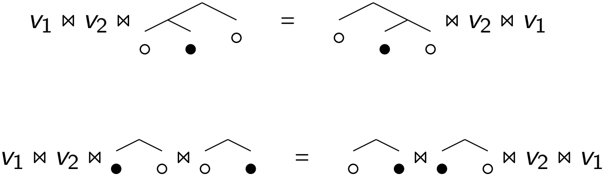
Example





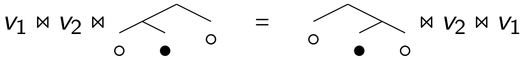
$$v_1 v_2 a b \quad = \quad b a v_2 v_1$$

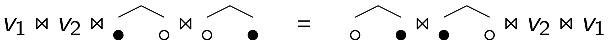
Example



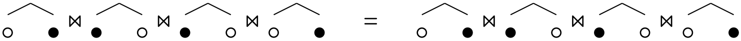
$v_1 v_2 ab \quad = \quad bav_2 v_1$ solution: $v_1 = b,\ v_2 = a$

Example





$v_1 v_2 ab \quad = \quad bav_2 v_1$ solution: $v_1 = b,\ v_2 = a$



Find a decidable fragment for $\bowtie$

Since the first-order theory of $\mathcal{S} = (\mathbb{T}, \bowtie)$ is undecidable, we want to find a decidable fragment of $\bowtie$ together with $(\sqcup, \sqcap, \bar{\square})$.

Connection to Tree Automatic Structures

- Let $\bowtie_\tau$ be the unary function over trees such that

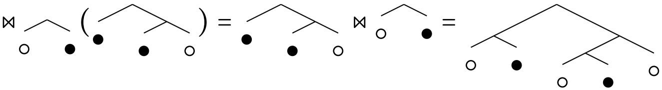
$$\bowtie_\tau(\tau') = \tau' \bowtie \tau$$

Connection to Tree Automatic Structures

- Let $\bowtie_\tau$ be the unary function over trees such that

$$\bowtie_\tau(\tau') = \tau' \bowtie \tau$$

- Example:

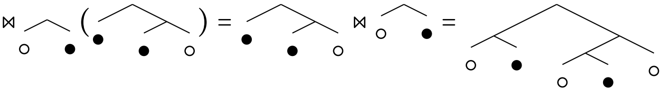


Connection to Tree Automatic Structures

- Let $\bowtie_{\tau}$ be the unary function over trees such that

$$\bowtie_{\tau}(\tau') = \tau' \bowtie \tau$$

- Example:



Tree automatic structure

Let $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_{\tau})$ then $\mathcal{T}$ is tree automatic, *i.e.*, its domain and relations are recognized by tree automata. Consequently, the first-order theory of $\mathcal{T}$ is decidable [Blumensath (1999); Blumensath and Gradel (2004)].

Outline

Contributions

- We show that $\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ forms a Countably Atomless Boolean Algebra.
- We reduce $\bowtie$ to string concatenation.
- We show $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_{\mathcal{T}})$ is tree-automatic.

Future Work

- Complexity of $(\mathbb{T}, \sqcap, \sqcup)$ ($\exists$-theory and first-order theory).
- Decidability of $(\mathbb{T}, \sqcap, \sqcup, \bowtie)$ ($\exists$-theory).
- Complexity of $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_{\tau})$ ($\exists$-theory and first-order theory).
- Extension of word equation to tree equation.

Thank you! ☺

Proof sketch:

- Let f be a solution of Σ. For each inequation $w_1 \neq w_2$ to hold, it suffices to have a single position where they differs.

Proof sketch:

- Let f be a solution of Σ. For each inequation $w_1 \neq w_2$ to hold, it suffices to have a single position where they differs.
- Therefore, there is at most one letter $a_i \notin \mathcal{A}(\Sigma)$ in each inequation that we need to preserve.

Proof sketch:

- Let f be a solution of Σ. For each inequation $w_1 \neq w_2$ to hold, it suffices to have a single position where they differs.
- Therefore, there is at most one letter $a_i \notin \mathcal{A}(\Sigma)$ in each inequation that we need to preserve.
- For other letters $b_i \notin \mathcal{A}(\Sigma)$, we simply replace them with a letter in $\mathcal{A}(\Sigma)$.

Proof sketch:

- Let f be a solution of Σ. For each inequation $w_1 \neq w_2$ to hold, it suffices to have a single position where they differs.
- Therefore, there is at most one letter $a_i \notin \mathcal{A}(\Sigma)$ in each inequation that we need to preserve.
- For other letters $b_i \notin \mathcal{A}(\Sigma)$, we simply replace them with a letter in $\mathcal{A}(\Sigma)$.
- As a result, the new solution satisfies the alphabet constraint.

A. Blumensath. *Automatic Structures*. PhD thesis, RWTH Aachen, 1999.

A. Blumensath and E. Gradel. Finite presentations of infinite structures: automata and interpretations. In *Theory of Computer Systems*, pages 641–674, 2004.

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. doi: 10.1007/3-540-44898-5_4. URL http://dx.doi.org/10.1007/3-540-44898-5_4.

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. doi: 10.1007/978-3-642-10672-9_13. URL http://dx.doi.org/10.1007/978-3-642-10672-9_13.

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

Matthew Parkinson. *Local Reasoning for Java*. PhD thesis, University of Cambridge, 2005.

Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 495–500, 1999. doi: 10.1109/SFFCS.1999.814622. URL http://dx.doi.org/10.1109/SFFCS.1999.814622.

Klaus U. Schulz. Makanin's algorithm for word equations - two improvements and a generalization. In *Word Equations and Related Topics, First International Workshop, IWWERT '90, Tübingen, Germany, October 1-3, 1990, Proceedings*, pages 85–150, 1990. doi: 10.1007/3-540-55124-7_4. URL http://dx.doi.org/10.1007/3-540-55124-7_4.