PhD Seminar

Computability and Complexity of Boolean Binary Tree Operations

Le Xuan Bach

Supervisor: Prof. Aquinas Hobor Joint with: Prof. Anthony Lin

School of Computing, NUS

April 26, 2017

`My life amounts to no more than one drop in a limitless ocean. Yet what is any ocean, but a multitude of drops? - David Mitchell, Cloud Atlas (2012)`

Fractional permissions in concurrency

```
struct tree{int d; struct tree* l; struct tree* r;}

void processTree(struct tree* x) {
  if (x == 0) { return; }

  print(x -> d);           ||  print(x -> d);
  processTree(x -> l);     ||  processTree(x -> l);
  processTree(x -> r);     ||  processTree(x -> r);
}
```

Fractional permissions in concurrency

```
struct tree{int d; struct tree* l; struct tree* r;}

void processTree(struct tree* x) {
  if (x == 0) { return; }

  print(x -> d);                  ||   print(x -> d);
  processTree(x -> l);            ||   processTree(x -> l);
  processTree(x -> r);            ||   processTree(x -> r);
}
```

$$\mathsf{tree}(\ell, \pi) \stackrel{\text{def}}{=} (\ell = \mathsf{null} \ \wedge \ \mathsf{emp}) \vee$$
$$\exists d, \ell_l, \ell_r.\ (\ell \stackrel{\pi}{\mapsto} (d, \ell_l, \ell_r) \ \star \ \mathsf{tree}(\ell_l, \pi) \ \star \ \mathsf{tree}(\ell_r, \pi))$$

Fractional permissions in concurrency

```
struct tree{int d; struct tree* l; struct tree* r;}

void processTree(struct tree* x) {
  if (x == 0) { return; }

  print(x -> d);                ||  print(x -> d);
  processTree(x -> l);          ||  processTree(x -> l);
  processTree(x -> r);          ||  processTree(x -> r);
}
```

$$\mathsf{tree}(\ell, \pi) \stackrel{\mathrm{def}}{=} (\ell = \mathsf{null} \ \wedge \ \mathsf{emp}) \vee$$
$$\exists d, \ell_l, \ell_r. \ (\ell \stackrel{\pi}{\mapsto} (d, \ell_l, \ell_r) \ \star \ \mathsf{tree}(\ell_l, \pi) \ \star \ \mathsf{tree}(\ell_r, \pi))$$

Spec: $\forall x, \pi. \ (\ \{\mathsf{tree}(x, \pi)\} \ \mathtt{processTree(x)} \ \{\mathsf{tree}(x, \pi)\})$

An abstract example

$$
\begin{tikzcd}
& \mathrm{tree}(x, \pi) \arrow[dl] \arrow[dr] & \\
\mathrm{tree}(x, \pi_1) \arrow[dr] & \| & \mathrm{tree}(x, \pi_2) \arrow[dl] \\
& \mathrm{tree}(x, \pi) &
\end{tikzcd}
$$

A concrete example: Rationals in [0,1]

$$\begin{tikzcd} & \mathrm{tree}(x, 1) \arrow[dl] \arrow[dr] & \\ \mathrm{tree}(x, \frac{1}{2}) \arrow[dr] & \| & \mathrm{tree}(x, \frac{1}{2}) \arrow[dl] \\ & \mathrm{tree}(x, 1) & \end{tikzcd}$$

A concrete example: Rationals in [0,1]

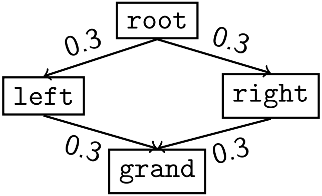
$$\begin{tikzcd} & \mathrm{tree}(x, 1) \arrow[dl] \arrow[dr] & \\ \mathrm{tree}(x, \frac{1}{2}) \arrow[dr] & \| & \mathrm{tree}(x, \frac{1}{2}) \arrow[dl] \\ & \mathrm{tree}(x, 1) & \end{tikzcd}$$

What could go wrong?

The shortcoming of rational model

$$\mathsf{tree}(\ell, \pi) \stackrel{\text{def}}{=} (\ell = \mathsf{null} \ \wedge \ \mathsf{emp}) \vee$$
$$\exists d, \ell_l, \ell_r. \ (\ell \stackrel{\pi}{\mapsto} (d, \ell_l, \ell_r) \ * \ \mathsf{tree}(\ell_l, \pi) \ * \ \mathsf{tree}(\ell_r, \pi))$$

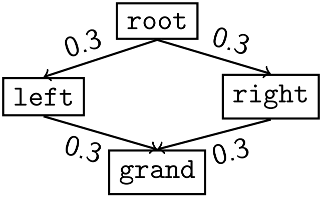
$$\begin{aligned}
&\mathtt{root} \stackrel{0.3}{\mapsto} (1, \mathtt{left}, \mathtt{right}) \, * \\
&\mathtt{left} \stackrel{0.3}{\mapsto} (1, \mathsf{null}, \mathtt{grand}) \ \ * \\
&\mathtt{right} \stackrel{0.3}{\mapsto} (1, \mathtt{grand}, \mathsf{null}) \, * \\
&\mathtt{grand} \stackrel{0.6}{\mapsto} (1, \mathsf{null}, \mathsf{null})
\end{aligned}$$



The shortcoming of rational model

$$\begin{aligned}\mathsf{tree}(\ell, \pi) \quad \overset{\mathtt{def}}{=} \quad & (\ell = \mathsf{null} \ \wedge \ \mathsf{emp}) \ \vee \\ & \exists d, \ell_l, \ell_r. \ (\ell \overset{\pi}{\mapsto} (d, \ell_l, \ell_r) \ \star \ \mathsf{tree}(\ell_l, \pi) \ \star \ \mathsf{tree}(\ell_r, \pi))\end{aligned}$$

$$\begin{aligned}&\mathtt{root} \overset{0.3}{\mapsto} (1, \mathtt{left}, \mathtt{right}) \star \\ &\mathtt{left} \overset{0.3}{\mapsto} (1, \mathsf{null}, \mathtt{grand}) \ \star \\ &\mathtt{right} \overset{0.3}{\mapsto} (1, \mathtt{grand}, \mathsf{null}) \star \\ &\mathtt{grand} \overset{0.6}{\mapsto} (1, \mathsf{null}, \mathsf{null})\end{aligned}$$



This is a dag, not a tree!

A quick diagnosis

- The disjointness property of SL:
  - $x \mapsto 1 * y \mapsto 1$: x and y are disjoint
  - $x \overset{\frac{1}{2}}{\mapsto} 1 * y \overset{\frac{1}{2}}{\mapsto} 1$: Hard to say …

A quick diagnosis

- The disjointness property of SL:
  - $x \mapsto 1 * y \mapsto 1$: x and y are disjoint
  - $x \overset{\frac{1}{2}}{\mapsto} 1 * y \overset{\frac{1}{2}}{\mapsto} 1$: Hard to say ...
- Facts about normal heap may be false in fractional heap, *e.g.*, heap's size.

A quick diagnosis

- The disjointness property of SL:
  - $x \mapsto 1 * y \mapsto 1$: x and y are disjoint
  - $x \overset{\frac{1}{2}}{\mapsto} 1 * y \overset{\frac{1}{2}}{\mapsto} 1$: Hard to say ...
- Facts about normal heap may be false in fractional heap, *e.g.*, heap's size.
- Desired property (Parkinson (2005)):

$$x \overset{\pi}{\mapsto} v \ * \ x \overset{\pi}{\mapsto} v \quad \vdash \quad \perp$$

A brief history of permission models

Rationals in $[0, 1]$

- John Boyland. Checking interference with fractional permissions. In *SAS*, 2003.
- Duy-Khanh Le *et al*. Threads as resource for concurrency verification. In *PEPM*, 2015.

A brief history of permission models

Rationals in $[0, 1]$

- John Boyland. Checking interference with fractional permissions. In *SAS*, 2003.
- Duy-Khanh Le *et al.* Threads as resource for concurrency verification. In *PEPM*, 2015.

Subsets of $\mathbb{N}$

- Matthew Parkinson. Local Reasoning for Java. PhD thesis, 2005.

A brief history of permission models

Rationals in $[0, 1]$

- John Boyland. Checking interference with fractional permissions. In *SAS*, 2003.
- Duy-Khanh Le *et al*. Threads as resource for concurrency verification. In *PEPM*, 2015.

Subsets of $\mathbb{N}$

- Matthew Parkinson. Local Reasoning for Java. PhD thesis, 2005.

Naturals, hybrid

- Richard Bornat *et al*. Permission accounting in separation logic. In *POPL*, 2005.

A brief history of permission models

Rationals in $[0, 1]$

- John Boyland. Checking interference with fractional permissions. In *SAS*, 2003.
- Duy-Khanh Le *et al.* Threads as resource for concurrency verification. In *PEPM*, 2015.

Subsets of $\mathbb{N}$

- Matthew Parkinson. Local Reasoning for Java. PhD thesis, 2005.

Naturals, hybrid

- Richard Bornat *et al.* Permission accounting in separation logic. In *POPL*, 2005.

Others

- Marieke Muisman *et al.* A symbolic approach to permission accounting for concurrent reasoning. In *ISPDC*, 2015.

A cure for disjointness

- Dockins *et al.* A fresh look at separation algebras and share accounting. In *APLAS*, 2009.

A cure for disjointness

- Dockins *et al.* A fresh look at separation algebras and share accounting. In *APLAS*, 2009.
- Disjointness property (permission layer):

$$\forall a \forall b.\ a \oplus a = b \ \rightarrow \ a = b$$

both a, b can be derived to be identity element.

A cure for disjointness

- Dockins *et al.* A fresh look at separation algebras and share accounting. In *APLAS*, 2009.
- Disjointness property (permission layer):

$$\forall a \forall b.\ a \oplus a = b \ \rightarrow \ a = b$$

both a, b can be derived to be identity element.

- Disjointness property (predicate layer):

$$x \overset{\pi}{\mapsto} v * x \overset{\pi}{\mapsto} v \vdash \bot$$

.

A checklist for my PhD

Three main pillars: application, theory and system.

A checklist for my PhD

Three main pillars: application, theory and system.

1. Application: Permission reasoning in Separation Logic.

A checklist for my PhD

Three main pillars: application, theory and system.

1. Application: Permission reasoning in Separation Logic.
2. Theory: Complexity and decidability of tree shares.

A checklist for my PhD

Three main pillars: application, theory and system.

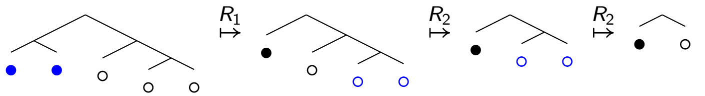
1. Application: Permission reasoning in Separation Logic.
2. Theory: Complexity and decidability of tree shares.
3. System: Certified tool in Coq, implementation and benchmarking in HIP/SLEEK.

Tree Share Definition

- Definition:

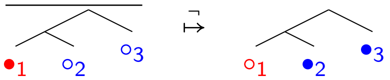
$$\tau \stackrel{\texttt{def}}{=} \circ \mid \bullet \mid \widehat{\tau \quad \tau} \qquad R_1 : \widehat{\bullet \quad \bullet} \mapsto \bullet \qquad R_2 : \widehat{\circ \quad \circ} \mapsto \circ$$

- Example:



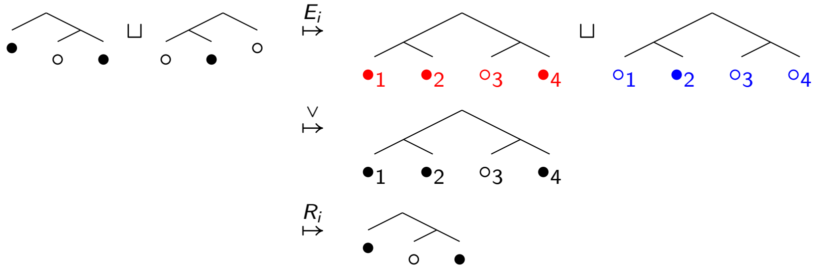
Tree Share Operators

The complement $\bar{\square}$:



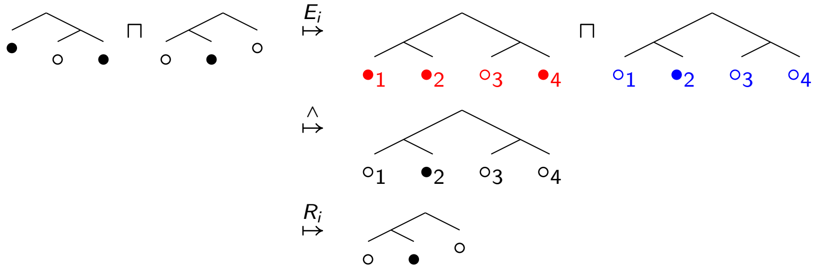
Tree Share Operators

The union $\sqcup$:



Tree Share Operators

The intersection $\sqcap$:



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

$\mathcal{M} = (\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is 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.)

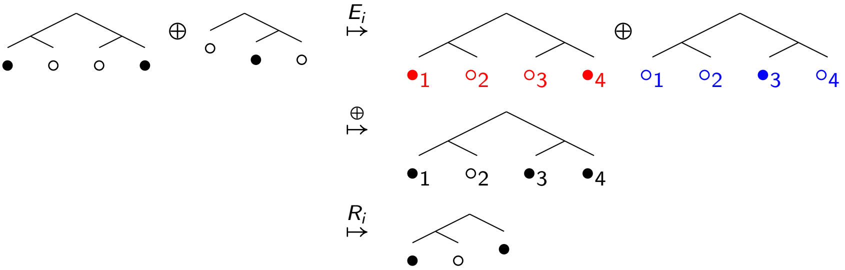
The join relation $\oplus$:

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

Tree Share Operators(cont.)

The join relation $\oplus$:

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



Properties of $\oplus$

$\mathcal{O} = (\mathbb{T}, \oplus)$ is a model for fractional permission in SL (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)

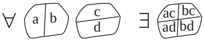
$J2.\ \tau_1 \oplus \tau_2 = \tau_2 \oplus \tau_1$ (commutativity)

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



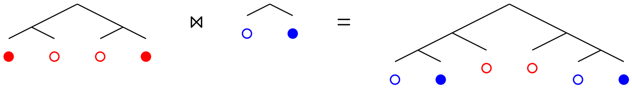
$J7.\ a \oplus b = z \wedge c \oplus d = z \Rightarrow \exists ac, ad, bc, bd.$

$ac \oplus ad = a \wedge bc \oplus bd = b \wedge ac \oplus bc = c \wedge ad \oplus bd = d$ (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 injective bowtie function $\bowtie$ replaces $\bullet$ with tree:



Properties of $\bowtie$

$\mathcal{S} = (\bowtie, \bullet)$ is a cancellative monoid (Dockins et al. (2009)):

*M*1. $(\tau_1 \bowtie \tau_2) \bowtie \tau_3 = \tau_1 \bowtie (\tau_2 \bowtie \tau_3)$	(associativity)
*M*2. $\tau \bowtie \bullet = \bullet \bowtie \tau = \tau$	(identity)
*M*3. $\tau \bowtie \tau_1 = \tau \bowtie \tau_2 \Rightarrow \tau \neq \circ \Rightarrow \tau_1 = \tau_2$	(left cancellation)
*M*4. $\tau_1 \bowtie \tau = \tau_2 \bowtie \tau \Rightarrow \tau \neq \circ \Rightarrow \tau_1 = \tau_2$	(right cancellation)
*M*5. $\tau \bowtie \circ = \circ \bowtie \tau = \circ$	(collapse point)
*M*6. $\tau_1 \bowtie (\tau_2 \diamond \tau_3) = (\tau_1 \diamond \tau_2) \bowtie (\tau_1 \diamond \tau_3),\ \diamond \in \{\sqcap, \sqcup, \oplus\}$	(distributivity)

Outline

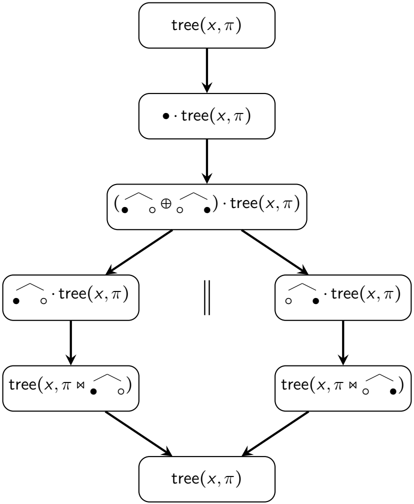
Scaling permissions for scaling predicates

$\mathcal{T} = (\mathbb{T}, \oplus, \bowtie)$ builds the *scaling permissions* for scaling predicates:

$$
\begin{array}{rclr}
\bullet \cdot P & \dashv\vdash & P & (\textsc{DotFull}) \\
(\pi_1 \oplus \pi_2) \cdot P & \dashv\vdash & (\pi_1 \cdot P) \ \ast \ (\pi_2 \cdot P) & (\textsc{DotPlus}) \\
\pi \cdot (P_1 \ast P_2) & \dashv\vdash & (\pi \cdot P_1) \ \ast \ (\pi \cdot P_2) & (\textsc{DotStar}) \\
\pi_1 \cdot (\pi_2 \cdot P) & \dashv\vdash & (\pi_2 \bowtie \pi_1) \cdot P & (\textsc{DotDot}) \\
\pi \cdot \mathsf{emp} & \dashv\vdash & \mathsf{emp} & (\textsc{DotEmp}) \\
\pi \cdot (x \overset{\pi'}{\mapsto} v) & \dashv\vdash & x \overset{\pi' \bowtie \pi}{\mapsto} v & (\textsc{DotMap})
\end{array}
$$

. . . with some side conditions.

Concrete example with tree shares



Ongoing work

We show that:

1. The scaling permissions work well with inductive predicates.
2. Reasonable side conditions can be checked syntactically.
3. A general induction framework for reasoning SL formulas with fractional permissions.
4. Our set of rules for scaling permissions is sound (Coq check).

Outline

Tree Shares as Countable Atomless Boolean Algebra

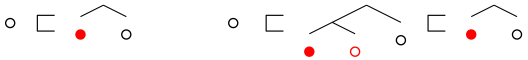
- $\mathcal{M} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ is Countable Boolean Algebra.
- Strict partial order:

$$\tau_1 \sqsubset \tau_2 \quad \stackrel{\text{def}}{=} \quad \tau_1 \sqcup \tau_2 = \tau_2 \wedge \tau_1 \neq \tau_2$$

- Atomless:

$$\circ \sqsubset \tau \ \rightarrow \ \exists \tau_1 .\ \circ \sqsubset \tau_1 \sqsubset \tau$$

- Example:



Decidability of $\mathcal{M} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bullet, \circ)$

The first-order theory of $\mathcal{M}$ is decidable. Furthermore, it is:

1. $\bigcup_{c<\omega} \mathsf{STA}(*, 2^{cn}, n)$-complete if only $\{\bullet, \circ\}$ are allowed (Le et al. (2016)).
2. Still $\bigcup_{c<\omega} \mathsf{STA}(*, 2^{cn}, n)$-complete with arbitrary constants (under submission).

Decidability of $\bowtie$

Decidability of $\mathcal{S} = (\mathbb{T}, \bowtie)$ (Le et al. (2016))

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

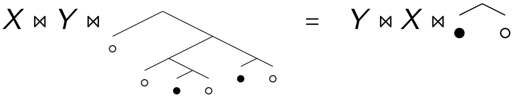
- The $\exists$-theory of $\mathcal{S}$ is decidable in PSPACE.
- The $\exists$-theory of $\mathcal{S}$ is NP-hard.
- The first-order theory of $\mathcal{S}$ is undecidable.

Main idea: Reduction to word equation.

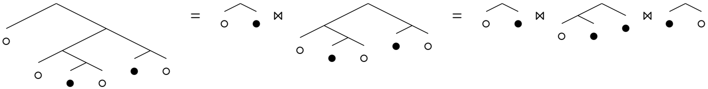
Isomorphism between $\bowtie$ and $\cdot$

Proof skech:

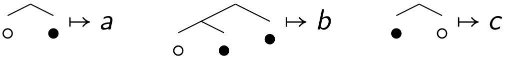
1. Tree equation:



2. Factorize tree into prime trees:



3. Prime trees are the alphabet:



4. Word equation: $XYabc = YXc$

Key observations

Unique tree 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 PTIME.

Decidability and Complexity of Word Equation (Marchenkov (1982); Plandowski (2006))

The satisfiability problem of word equation is decidable with:

- Lower bound: NP-hard
- Upper bound: PSPACE

Also, the first-order theory of word equation is undecidable.

Under submission

Restricted bowtie with constants on the left/right:

$$\bowtie_{\tau}(x) \overset{\text{def}}{=} x \bowtie \tau \qquad {}_{\tau}\!\bowtie(x) \overset{\text{def}}{=} \tau \bowtie x$$

Decidability of restricted $\bowtie$

The first-order theory of $(\mathbb{T}, \bowtie_{\tau}, {}_{\tau}\!\bowtie)$ is:

1. Lower bound: PSPACE-hard
2. Upper bound: 2EXSPACE

Proof sketch: Reduction to automatic structure with bounded degree.

Connection to Tree Automatic Structures

Restrict $\bowtie$ with constants on the right: $\bowtie_{\mathcal{T}}$.

Tree automatic structure (Le et al. (2016))

Let $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_{\mathcal{T}})$ then:

- $\mathcal{T}$ is tree automatic
- The first-order theory of $\mathcal{T}$ is decidable [Blumensath (1999); Blumensath and Gradel (2004)].

Important for scaling permissions.

Under submission

- The FO of $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_\tau)$ is non-elementary.
  - Key idea: Reduce to FO of two successors + prefix relation.

Under submission

- The FO of $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_\tau)$ is non-elementary.
  - Key idea: Reduce to FO of two successors + prefix relation.
- No shame using tree automatic solvers (*e.g.* MONA)!

Outline

Integrate tree shares into HIP/SLEEK

- Extract share constraint from SL entailment:

$$x \overset{\bullet}{\mapsto} 1 \vdash x \overset{\widehat{\bullet \quad \circ}}{\mapsto} 1 \quad \leadsto \quad v_1 = \bullet \vdash \exists v_2.\ v_1 = \widehat{\bullet \quad \circ} \oplus v_2$$

Integrate tree shares into HIP/SLEEK

- Extract share constraint from SL entailment:

$$x \overset{\bullet}{\mapsto} 1 \ \vdash \ x \overset{\widehat{\bullet \quad \circ}}{\mapsto} 1 \quad \leadsto \quad v_1 = \bullet \ \vdash \ \exists v_2.\ v_1 = \widehat{\bullet \quad \circ} \oplus v_2$$

- Let $\Sigma = \{v_1 \oplus v_2 = v_3\}$. Two procedures (Le et al. (2012)):
  1. SAT(Σ)
  2. $\Sigma_1 \vdash \Sigma_2$

Integrate tree shares into HIP/SLEEK

- Extract share constraint from SL entailment:

$$x \overset{\bullet}{\mapsto} 1 \;\vdash\; x \overset{\widehat{\bullet\quad\circ}}{\mapsto} 1 \quad\rightsquigarrow\quad v_1 = \bullet \;\vdash\; \exists v_2.\ v_1 = \widehat{\bullet\quad\circ} \oplus v_2$$

- Let $\Sigma = \{v_1 \oplus v_2 = v_3\}$. Two procedures (Le et al. (2012)):
  1. SAT(Σ)
  2. $\Sigma_1 \vdash \Sigma_2$
- Sound and complete. Main idea: Small model property.

Implementation in Coq (under submission)

- Can handle negative constraints: $\neg(a \oplus b = c)$.
- Implementation + certified proof: 35k+ LOC. Can run entirely in Coq.
- Some bugs in the old tool and HIP/SLEEK ...

Implementation in Coq (under submission)

- Can handle negative constraints: $\neg(a \oplus b = c)$.
- Implementation + certified proof: 35k+ LOC. Can run entirely in Coq.
- Some bugs in the old tool and HIP/SLEEK ...
- Several optimizations:
  1. Partition into independent sub-systems.
  2. Heuristic rules + bound algorithm (Hobor and Gherghina (2012)).
  3. Faster: 4.6%.

Implementation in Coq (under submission)

- Can handle negative constraints: $\neg(a \oplus b = c)$.
- Implementation + certified proof: 35k+ LOC. Can run entirely in Coq.
- Some bugs in the old tool and HIP/SLEEK ...
- Several optimizations:
  1. Partition into independent sub-systems.
  2. Heuristic rules + bound algorithm (Hobor and Gherghina (2012)).
  3. Faster: 4.6%.
- Extracted to Ocaml or Haskell.

Outline

Contributions

- We show how model tree shares for scaling permission in SL.
- We investigate the decidability and complexity of tree shares.
- We certify tree share decision procedures in Coq and integrate them into HIP/SLEEK.

Future Work

- Generalize the tree share model to "generic binary trees".
- Close the complexity gap in certain tree share sub-structures.
- Develop decision procedures for $\mathcal{T} = (\mathbb{T}, \sqcup, \sqcap, \bar{\square}, \bowtie_{\mathcal{T}})$.
- Integrate the scaling permissions in HIP/SLEEK for benchmarking.

Thank you! ☺

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.

Robert Dockins, Aquinas Hobor, and Andrew W. Appel. A fresh look at separation algebras and share accounting. In *APLAS*, pages 161–177, 2009.

Aquinas Hobor and Cristian Gherghina. Barriers in concurrent separation logic: Now with tool support! Accepted for the journal of Selected Papers of the 20th European Symposium on Programming (ESOP 2011), 2012.

Xuan Bach Le, Cristian Gherghina, and Aquinas Hobor. Decision procedures over sophisticated fractional permissions. In *APLAS*, pages 368–385, 2012.

Xuan Bach Le, Aquinas Hobor, and Anthony W. Lin. Decidability and complexity of tree shares formulas. In *FSTTCS*, 2016.

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

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

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