Decidability and Complexity of Tree Share Formulas

Xuan Bach Le[†] Aquinas Hobor[‡,†] Anthony W. Lin[‡]

[†] School of Computing and [‡] Yale-NUS College, National University of Singapore

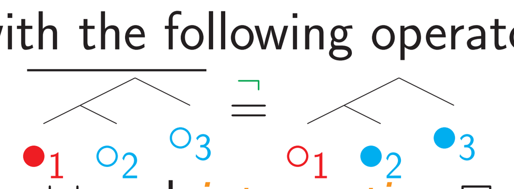
Introduction

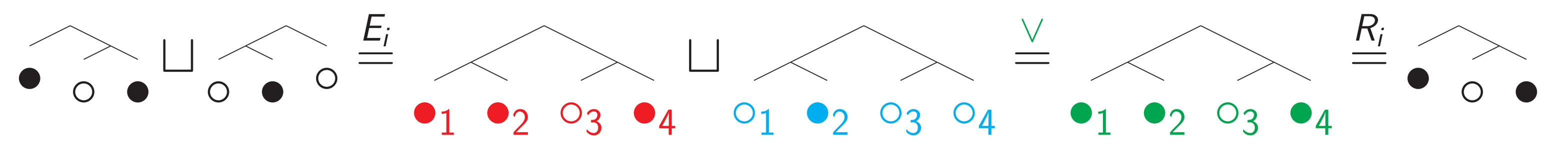
- A tree share $\tau \in \mathbb{T}$ is inductively defined as a boolean binary tree equipped with the reduction rules R_1 and R_2 (their inverses are E_1, E_2 resp.):

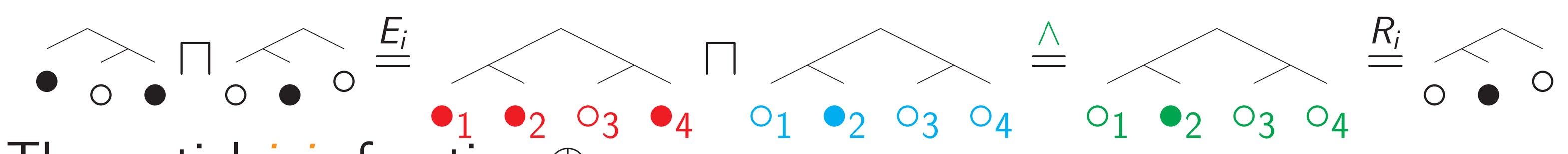
$$\tau \stackrel{\text{def}}{=} \circ \mid \bullet \mid \widehat{\tau\ \tau} \qquad R_1 : \widehat{\bullet\ \bullet} \mapsto \bullet \qquad R_2 : \widehat{\circ\ \circ} \mapsto \circ \qquad (1)$$

- The tree domain $\mathbb{T}$ contains *canonical trees* which are irreducible with respect to the reduction rules. Here $\circ$ denotes an "empty" leaf while $\bullet$ a "full" leaf. The tree $\circ$ is thus the empty tree, and $\bullet$ the full tree. There are two "half" shares: $\widehat{\circ\ \bullet}$ and $\widehat{\bullet\ \circ}$, and four "quarter" shares, beginning with $\widehat{\widehat{\bullet\ \circ}\ \circ}$.

- The domain $\mathbb{T}$ is equipped with the following operators:

1. The *complement* $\bar{\square}$:  (2)
2. The Boolean function *union* $\sqcup$ and *intersection* $\sqcap$ operator:

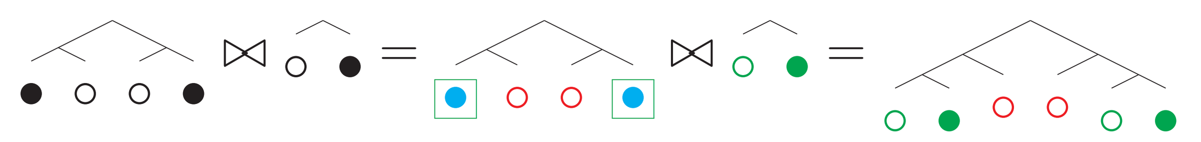
 (3)

 (4)

3. The partial *join* function $\oplus$:

$$\tau_1 \oplus \tau_2 = \tau_3 \stackrel{\text{def}}{=} \tau_1 \sqcup \tau_2 = \tau_3 \wedge \tau_1 \sqcap \tau_2 = \circ \qquad (5)$$

4. The *injection bowtie* function $\bowtie$ generalized from string concatenation:

 (6)

Applications of tree shares

Tree shares are embedded into separation logic to reason about resource accouting: $\text{addr} \xmapsto{\tau_1 \oplus \tau_2} \text{val} \stackrel{\text{equiv}}{=} \text{addr} \xmapsto{\tau_1} \text{val} \star \text{addr} \xmapsto{\tau_2} \text{val}$

1. Share policies to reason about permissions:

$$\text{WRITE}(\tau) \stackrel{\text{def}}{=} \tau = \bullet \qquad \text{READ}(\tau) \stackrel{\text{def}}{=} \tau \neq \circ$$

$$\frac{}{\text{WRITE}(\bullet)}\ \textsc{FullWrite} \qquad \frac{\text{WRITE}(\tau)}{\tau = \bullet}\ \textsc{WriteFull}$$

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

Figure: Simple policy inference rules for single writer and multiple readers

2. Allow resources to be split and shared in large scale:

$$\text{tree}(\ell, \tau) \stackrel{\text{def}}{=} (\ell = \text{null} \wedge \text{emp}) \vee \exists \ell_l, \ell_r.\ (\ell \xmapsto{\tau} (\ell_l, \ell_r) \star \text{tree}(\ell_l, \tau) \star \text{tree}(\ell_r, \tau)) \qquad (7)$$

$$\text{tree}(\ell, \tau_1 \oplus \tau_2) \stackrel{\text{equiv}}{=} \text{tree}(\ell, \tau_1) \star \text{tree}(\ell, \tau_2) \qquad (8)$$

3. Allow resources to be split uniformly:

$$\tau_1 \cdot \text{tree}(\ell, \tau_2) \stackrel{\text{def}}{=} \text{tree}(\ell, \tau_2 \bowtie \tau_1) \qquad (9)$$

$$(\tau_1 \oplus \tau_2) \cdot \text{tree}(\ell, \tau) \stackrel{\text{equiv}}{=} \tau_1 \cdot \text{tree}(\ell, \tau) \star \tau_2 \cdot \text{tree}(\ell, \tau) \qquad (10)$$

$$\tau_1 \cdot \text{tree}(\ell, \tau_2 \bowtie \tau_3) \stackrel{\text{equiv}}{=} (\tau_3 \bowtie \tau_1) \cdot \text{tree}(\ell, \tau_2) \qquad (11)$$

4. Allow resources to be locally transformed back to non-share version:

$$\text{tree}(\ell, \tau) \stackrel{\text{equiv}}{=} \tau \cdot \text{tree}(\ell, \bullet) \stackrel{\text{equiv}}{=} \tau \cdot \text{tree}(\ell) \qquad (12)$$

Properties of tree shares

- $(\sqcup, \sqcap, \bar{\square}, \bullet, \circ)$ forms a Boolean Algebra:

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)

- $(\bowtie, \bullet)$ forms an Algebraic Monoid with additional properties:

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)

- Properties of $\oplus$:

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)

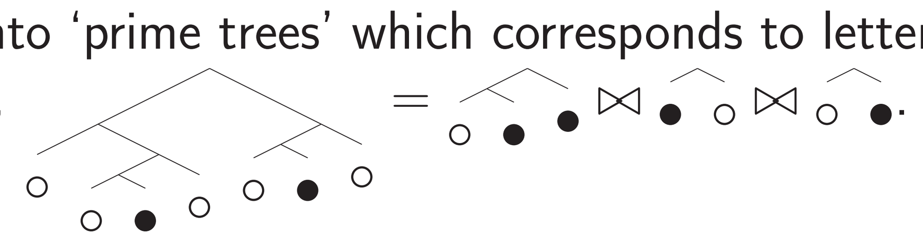
Decidability and Complexity of Tree Structures

Theorem 1. (Decidability of $\bowtie$)

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

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

Proof sketch. Reduction to word equation problem. We show each tree τ can be uniquely factorized into 'prime trees' which corresponds to letters in word alphabet. For example,



Theorem 2. (Tree automatic)

Let $\mathcal{M} = (\mathbb{T}, \sqcap, \sqcup, \bar{\square}, \bowtie_\tau)$ where $\bowtie_\tau (\tau') = \tau' \bowtie \tau$ then $\mathcal{M}$ is tree-automatic, *i.e.* the domain and operators of $\mathcal{M}$ are recognized by tree automata. As a result, the first-order theory of $\mathcal{M}$ is decidable.

Proof sketch. By explicitly constructing the automata.

Theorem 3. (Finite search)

Let Σ be system of equation constraints $\pi_1 \oplus \pi_2 = \pi_3$ (π_i is either tree or variable) and $S(\Sigma)$ be the solution space of Σ. We define the height of Σ, denoted by $|\Sigma|$, to be the height of the tallest tree in Σ or zero otherwise. In order to check $S(\Sigma) = \emptyset$ (satisfiability) or $S(\Sigma_1) \subseteq S(\Sigma_2)$ (entailment), it is sufficient to consider trees of heights at most the height of the system.

Proof sketch. Let $\mathbb{T}_n$ be set of trees of heights at most n, we construct an isomorphism $\mathbb{T}_{n+1} \mapsto \mathbb{T}_n \times \mathbb{T}_n$ that preserves the join relation. This shows that 'big solutions' are basically combinations of 'smaller solutions'. As a result, we can reduce the search space to small solutions only.

Share solver

- We develop a solver to handle the satisfiability and entailment problem in Theorem 3. Our tool is actually *more powerful*: it can handle negative constraints $\neg(\pi_1 \oplus \pi_2 = \pi_3)$ and existential variables. The tool is implemented and certified in Coq, a theorem prover. Its main purpose is to verify the share constraints generated from separation logic entailment tools.
- Instances that the tool can verify:
  - $\exists \Phi$ (satisfiability) and $\forall(\exists \Phi_1 \rightarrow \exists \Phi_2)$ (entailment).
  - All properties $J1 - J8$ of join relation $\oplus$ with an exception that $J5$ is changed to a weaker form $\forall \tau. \exists u.\ \tau \oplus u = \tau$.

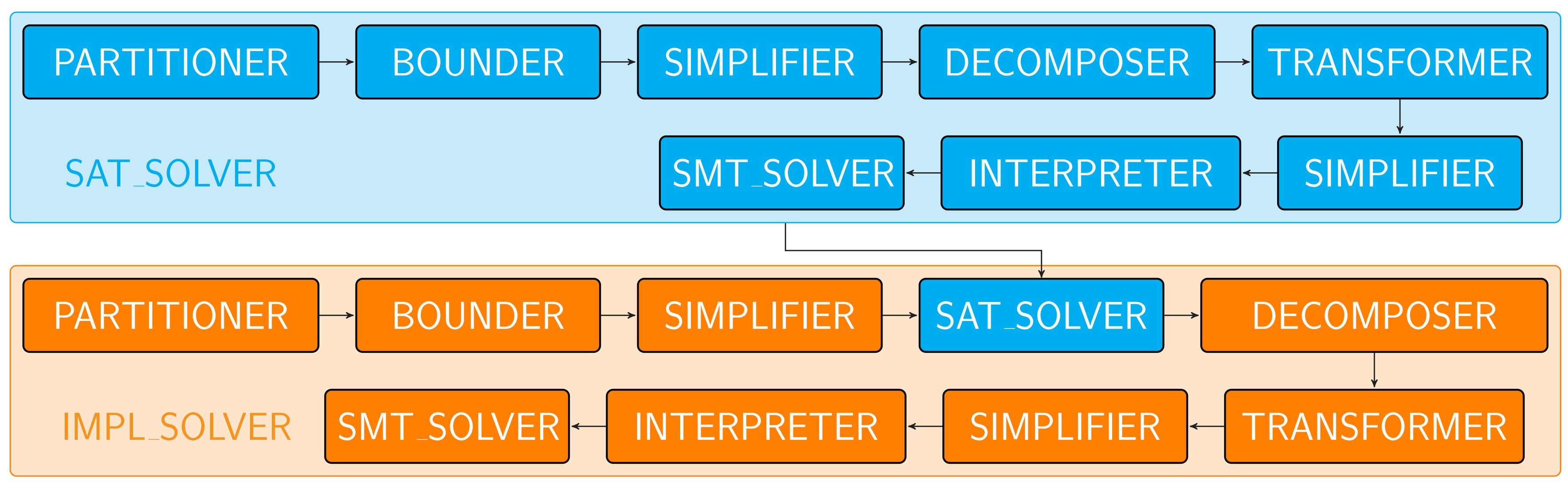


Figure: SAT solver and IMPL solver

Components of share solver

- PARTITIONER: partition the system into independent subsystems.
- BOUNDER: use order theory to prune space.
- SIMPLIFIER: apply effective generic heuristics for reduction the overall difficulty via computation.
- DECOMPOSER: decompose share system into subsystems of height zero.
- TRANSFORMER: for share system of height zero, the component converts constants and variables from share type to boolean type.
- INTERPRETER: transform boolean system into equivalent boolean formula.
- SMT SOLVER: check the validity of the boolean formula.
- Link to the tool: `www.comp.nus.edu.sg/~lxbach/share_prover/`

References

1. Xuan Bach Le, Aquinas Hobor and Anthony W. Lin. Decidability and Complexity of Tree Share Formulas. In *FSTTCS*, 2016.
2. Xuan Bach Le and Aquinas Hobor. A Certified Decision Procedure for Sophisticated Fractional Permissions. In submission, 2016.
3. Xuan Bach Le, C. Gherghina, and Aquinas Hobor. Decision procedures over sophisticated fractional permissions. In *APLAS*, 2012.

Contact information: lxbach@comp.nus.edu.sg hobor@comp.nus.edu.sg anthony.w.lin@yale-nus.edu.sg