ICFEM 2017, Xi'an China

A Certified Decision Procedure for Tree Shares

(to reason about resource sharing in concurrent programs)

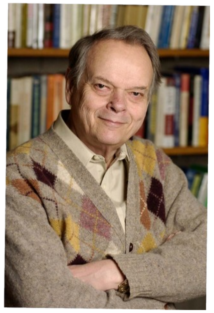
Xuan-Bach Le Thanh-Toan Nguyen Wei-Ngan Chin Aquinas Hobor

School of Computing, National University of Singapore

November 13, 2017

Concurent Separation Logic

John R. Reynolds



Steve Brooke

Peter O’Hearn

1. Reason about correctness of concurrent programs.

2. Precusor: Separation Logic (SL).

3. Simple, Compositional Reasoning.

4. Used in many automatic verification tools:
   - HIP/SLEEK (Nguyen et al. (2007))
   - Infer (Calcagno et al. (2015))
   - Viper (Müller et al. (2016))
   - VeriFast (Jacobs et al. (2010))
   - Staring (Windsor et al. (2017))
   - Caper (Young et al. (2017))

Some basics

- Maps-to predicate

$$\mathsf{address} \mapsto \mathsf{value}$$

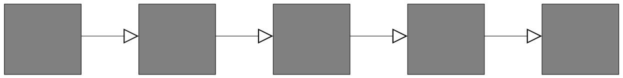
- Disjoint conjunction

$$\boxed{x \mapsto 1} \star \boxed{y \mapsto 1}$$

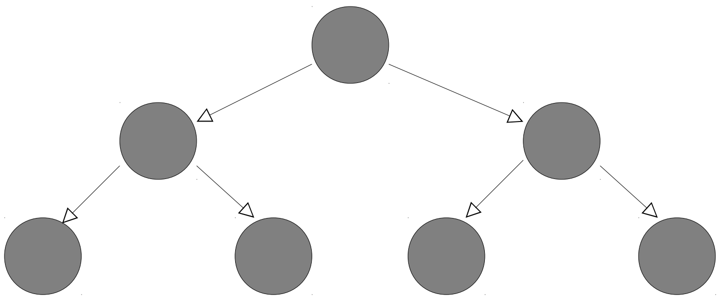
Some basics

Shape inductive predicates

$$\mathsf{list}(x) \stackrel{\texttt{def}}{=} (x = \mathsf{null}) \lor \exists d, x_1.\ x \mapsto (d, x_1) \star \mathsf{list}(x_1)$$

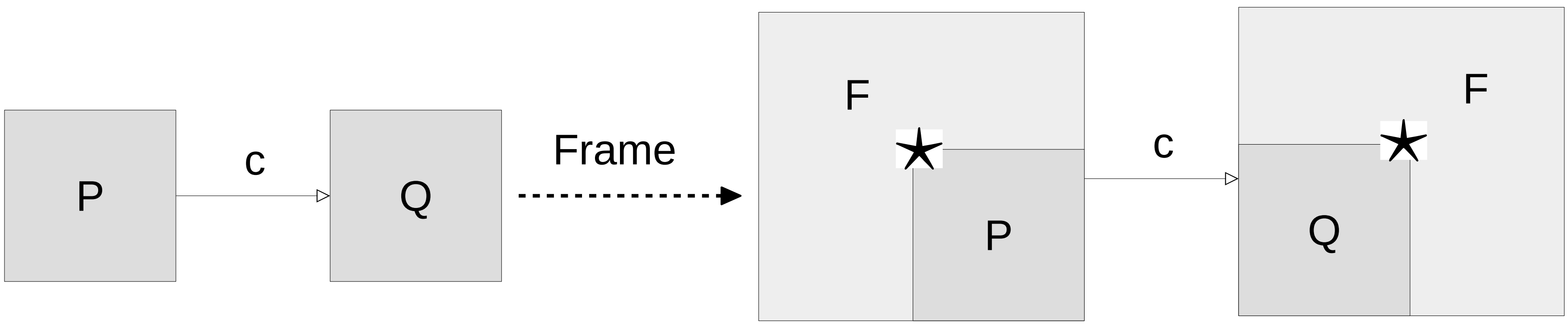


$$\mathsf{tree}(x) \stackrel{\texttt{def}}{=} (x = \mathsf{null}) \lor \exists d, x_1, x_2.\ x \mapsto (d, x_1, x_2) \star \mathsf{tree}(x_1) \star \mathsf{tree}(x_2)$$



Some basics

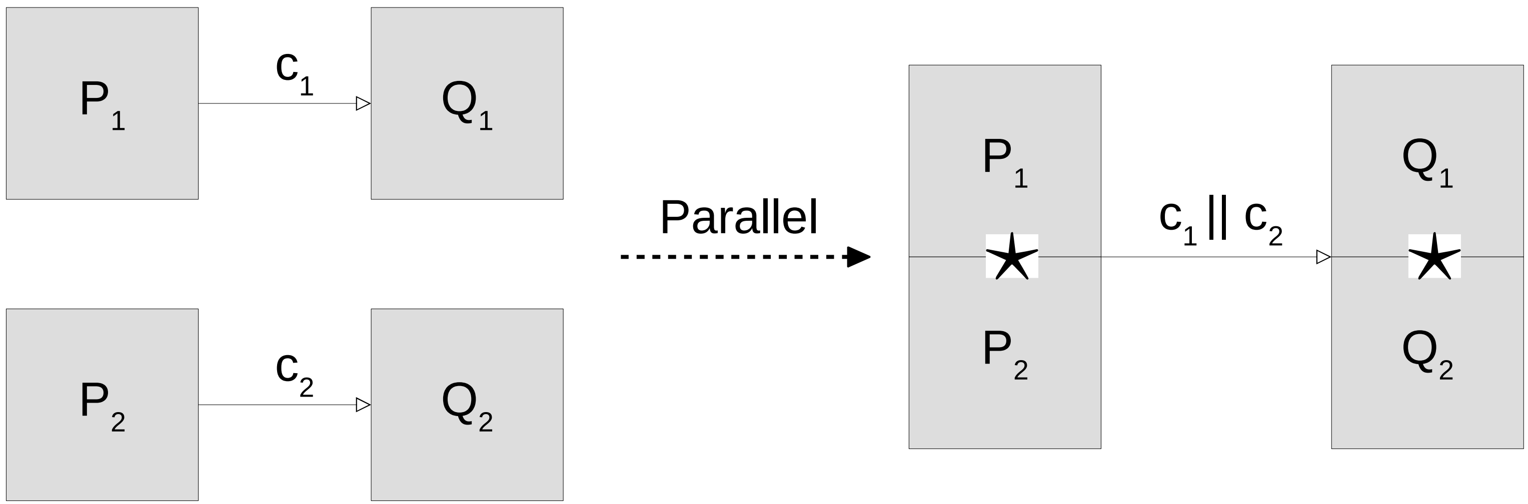
Frame Rule



$$\frac{\{P\}\ c\ \{Q\} \qquad \mathsf{mod}(c) \cap \mathsf{fv}(F) = \emptyset}{\{F \star P\}\ c\ \{F \star Q\}}\ \mathsf{Frame}$$

Some basics

Parallel Rule



$$\frac{\{P_1\}\ \mathsf{c}_1\ \{Q_1\} \qquad \mathsf{fv}(\mathsf{c}_1, P_1, Q_1) \cap \mathsf{modified}(\mathsf{c}_2) = \emptyset \\ \{P_2\}\ \mathsf{c}_2\ \{Q_2\} \qquad \mathsf{fv}(\mathsf{c}_2, P_2, Q_2) \cap \mathsf{modified}(\mathsf{c}_1) = \emptyset}{\{P_1 \star P_2\}\ \mathsf{c}_1\ ||\ \mathsf{c}_2\ \{Q_1 \star Q_2\}}\ \mathsf{Parallel}$$

Permissions in CSL

- Fractional maps-to

$$\text{address} \xmapsto{\text{permission}} \text{value}$$

- Rational permission model $\langle (0,1], + \rangle$:
  - $\pi \in (0,1], \quad 1 : \mathrm{WRITE}, \quad (0,1) : \mathrm{READ}$
  - Join/split permissions:

$$x \xmapsto{\pi_1 + \pi_2} v \dashv\vdash x \xmapsto{\pi_1} v \star x \xmapsto{\pi_2} v$$

  - Example:

$$x \xmapsto{1} v \dashv\vdash x \xmapsto{0.5} v \star x \xmapsto{0.5} v$$

Shortcomings of rational permissions

Lack of disjointness:

- In traditional SL:

$$\boxed{x \mapsto v} \star \boxed{x \mapsto v} \vdash \bot$$

- With rational permissions:

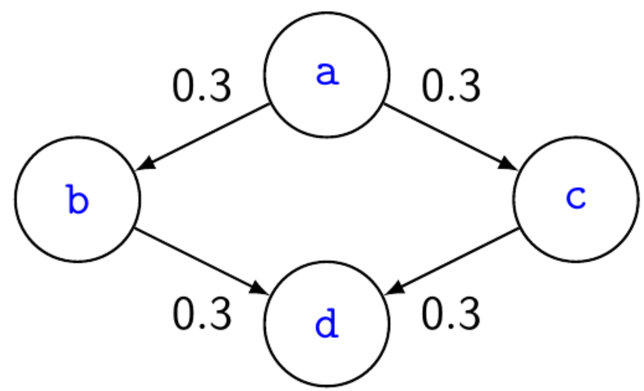
$$\boxed{x \xmapsto{0.5} v} \star \boxed{x \xmapsto{0.5} v} \vdash \boxed{x \xmapsto{1} v}$$

Shortcomings of rational permissions

Deformation of shape predicates

$$\begin{aligned}\texttt{tree}(x,\pi) &\stackrel{\text{def}}{=} (x = \texttt{null}) \lor \\ &\exists d, x_1, x_2.\ x \stackrel{\pi}{\mapsto} (d, x_1, x_2) \star \texttt{tree}(x_1,\pi) \star \texttt{tree}(x_2,\pi)\end{aligned}$$

$$\begin{aligned}&\texttt{a} \stackrel{0.3}{\mapsto} (1, \texttt{b}, \texttt{c}) \quad * \\ &\texttt{b} \stackrel{0.3}{\mapsto} (1, \texttt{null}, \texttt{d}) \quad * \\ &\texttt{c} \stackrel{0.3}{\mapsto} (1, \texttt{d}, \texttt{null}) \quad * \\ &\texttt{d} \stackrel{0.6}{\mapsto} (1, \texttt{null}, \texttt{null})\end{aligned}$$



$$\texttt{tree}(a, 0.3)$$

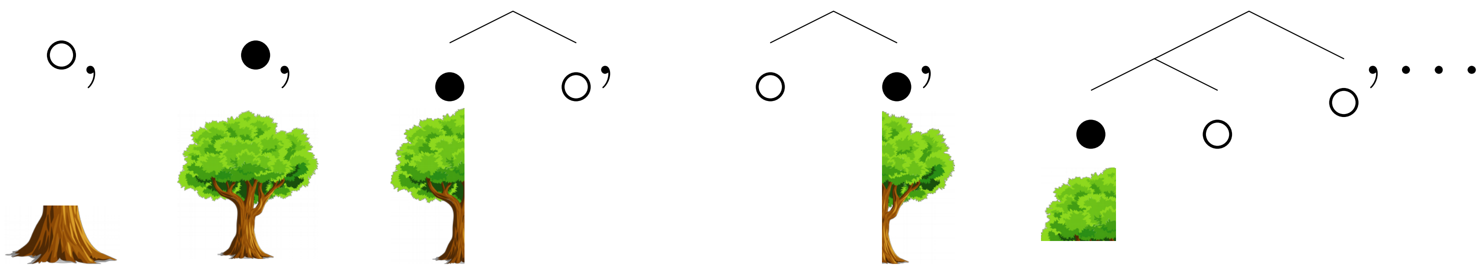
Shortcomings of rational permissions

Poor support of complete decision procedures

- Not finitely axiomatized in first-order logic.
- The addition group $\langle \mathbb{Q}, + \rangle$ is not finitely generated.
- First-order theory is undecidable (Robinson, 1949).

Tree share permissions

- By Dockins et al. (2009)
- Boolean binary trees



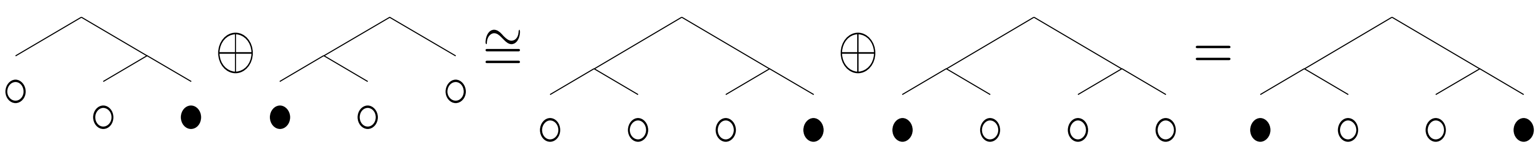
- Canonical form

Tree addition $\bigoplus$

- Base cases: $\circ \equiv 0 \qquad \bullet \equiv 1$

$$\circ \bigoplus \circ = \circ \qquad \bullet \bigoplus \circ = \circ \bigoplus \bullet = \bullet \qquad \bullet \bigoplus \bullet \ \text{undefined}$$

- General case: leaf-wise



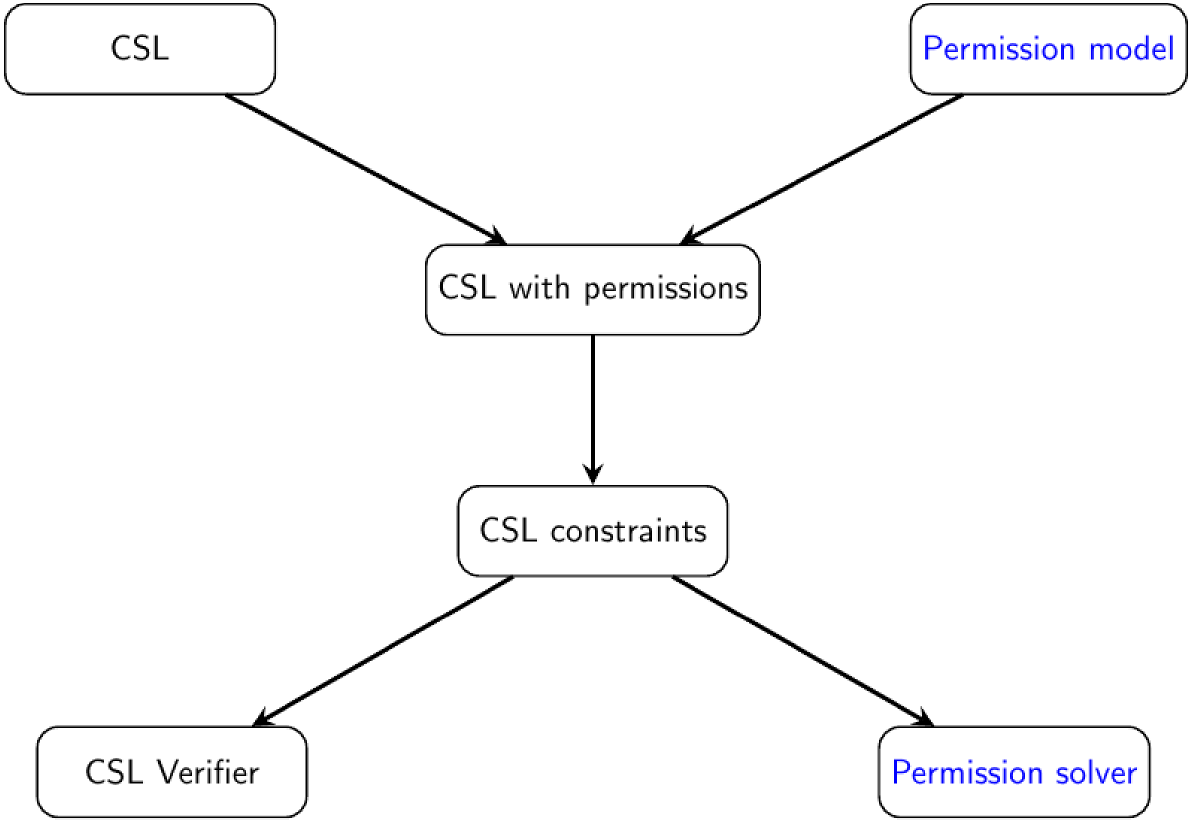
Tree permission model

$$\langle \mathbb{T}, \bigoplus \rangle$$

tree shares

tree addition

Why permission solver?



Previous work

- Complete procedures for $\langle \mathbb{T}, \oplus \rangle$
  - SAT: $\exists \bar{X}.\ \Phi$
  - IMP: $\forall \bar{X}_1.\ (\Phi_1 \Rightarrow \exists \bar{X}_2 . \Phi_2)$

    where $$\Phi = \bigwedge a \oplus b = c$$
- NP-hard. Reduce to Boolean formulae.
- Correctness proof: small model technique.
- Benchmarked in HIP/SLEEK.

Shortcomings

- Not certified (code bug).

- Only handle restricted form of negation

$$x \neq \circ$$

- Soundness proof for restricted negation contained a bug (proof bug).

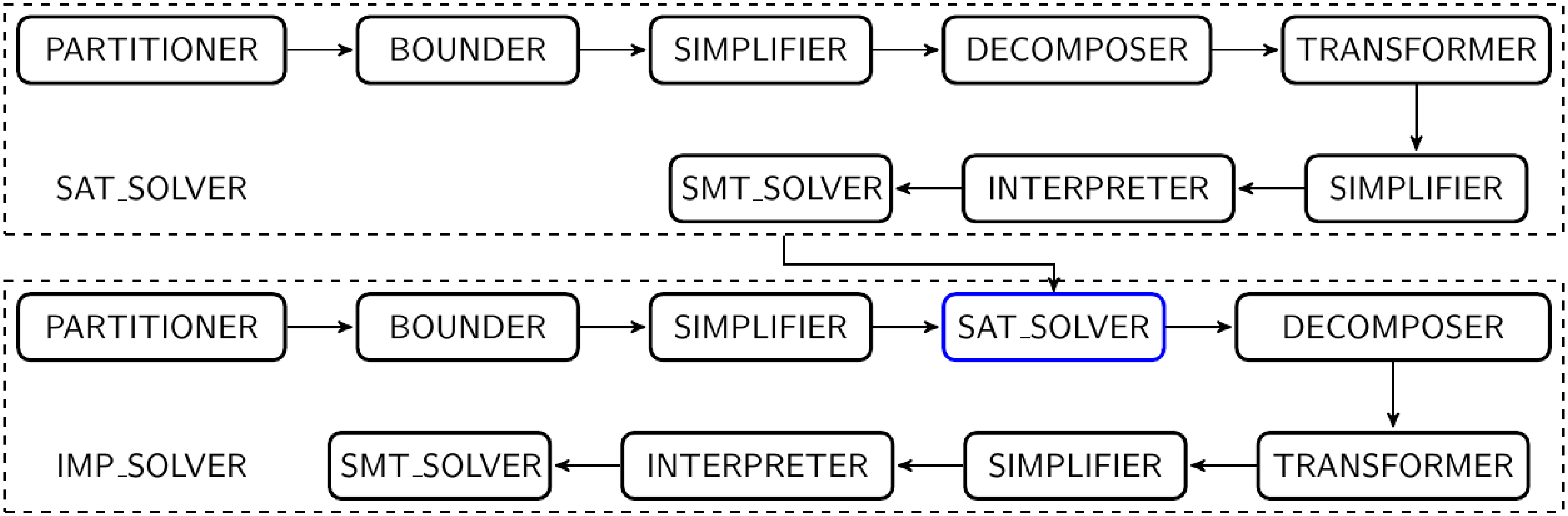
Contributions

We fix the previous issues:

- Complete procedures for SAT and IMP with general negative constraints:
  - SAT: $\exists \bar{X}.\ \Phi$
  - IMP: $\forall \bar{X}_1.\ (\Phi_1 \Rightarrow \exists \bar{X}_2 . \Phi_2)$

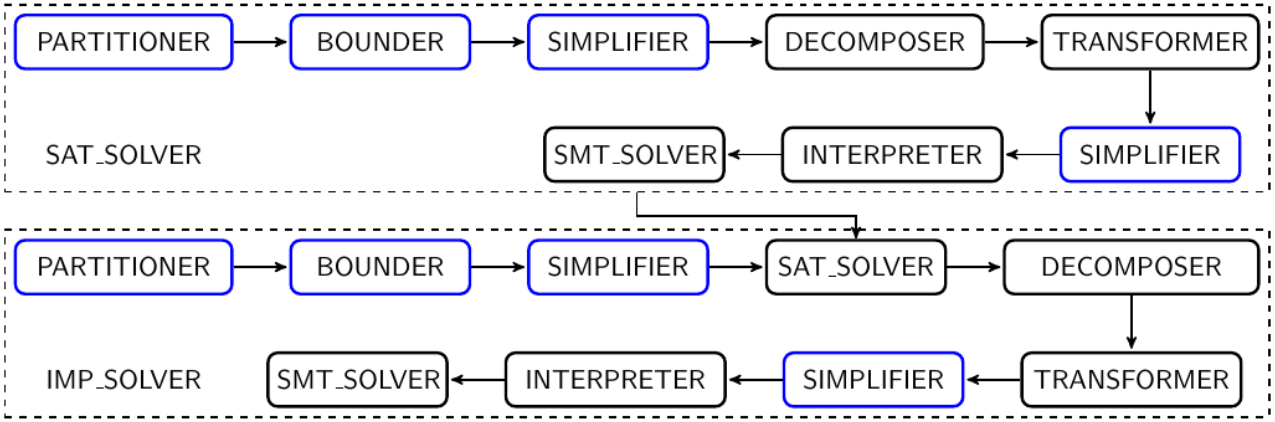
  where $\Phi = \bigwedge a \oplus b = c \wedge \bigwedge a' \oplus b' \neq c'$
- Certified in Coq.
- New correctness proofs.
- Benchmarked in HIP/SLEEK.

Overview of procedures

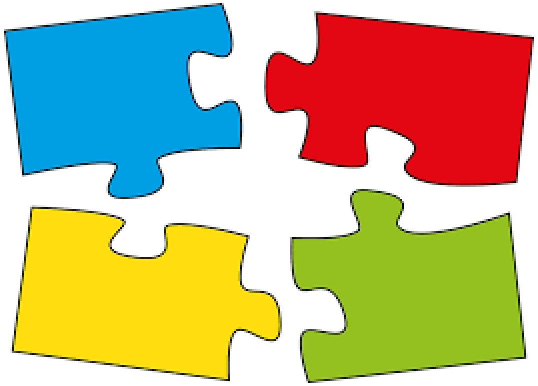


IMP_SOLVER needs to call SAT_SOLVER.

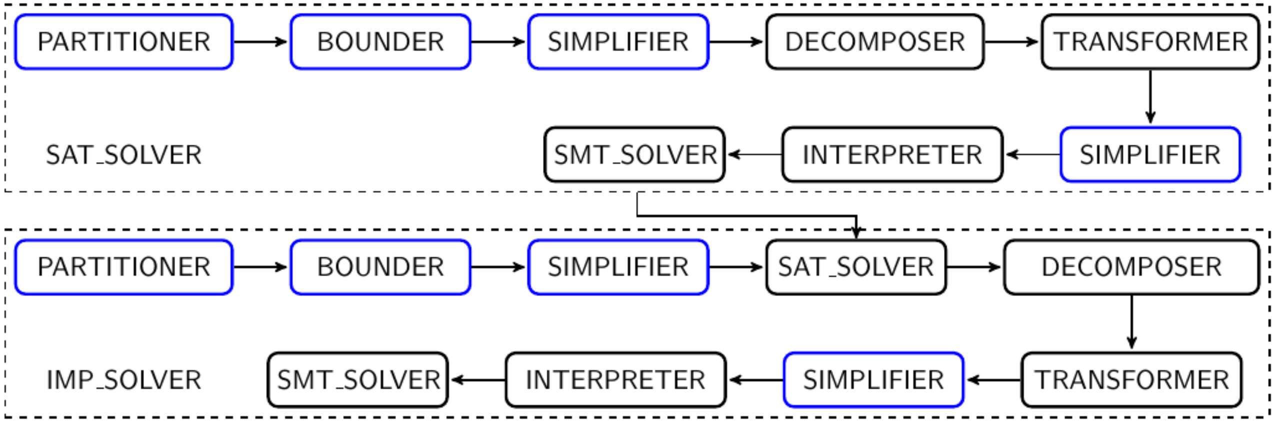
Optimization components



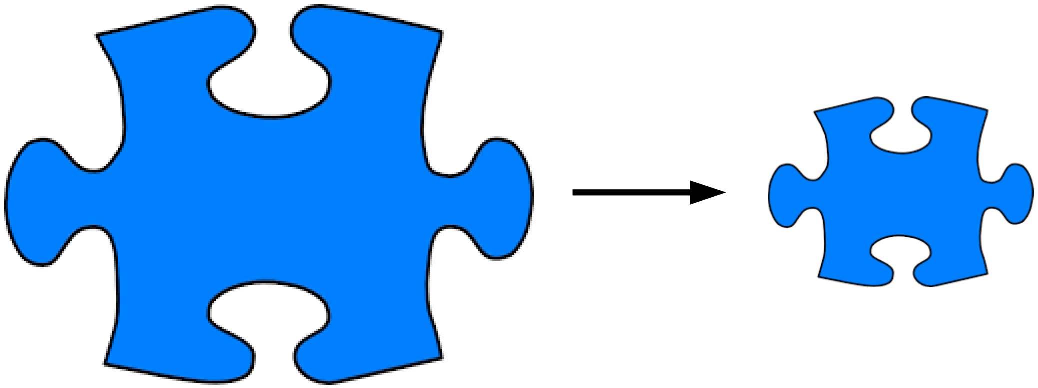
PARTITIONER: split problem into independent problems.



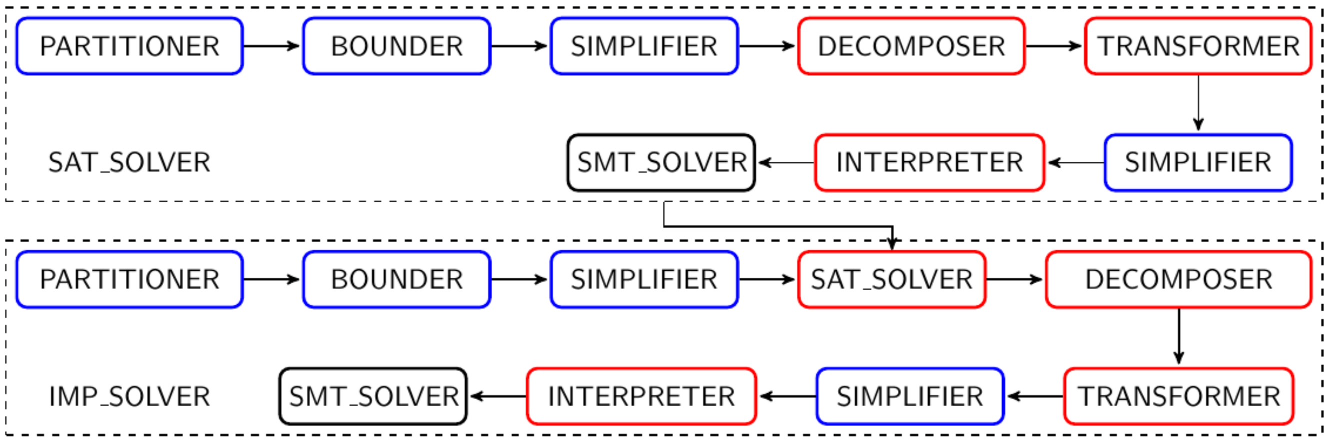
Optimization components



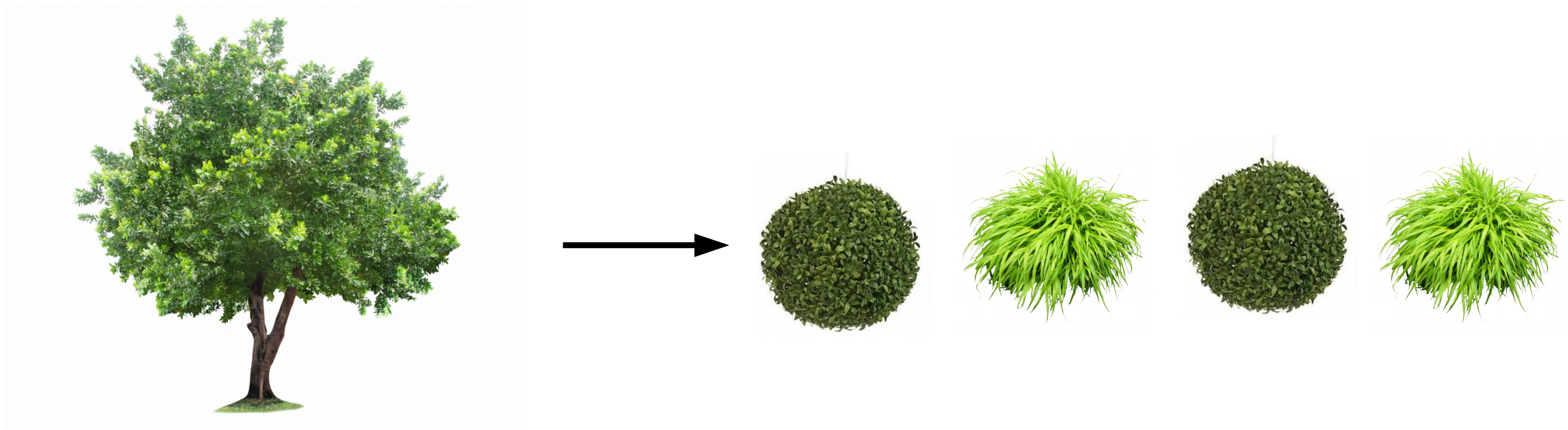
BOUNDER + SIMPLIFIER: reduce the problem's size.



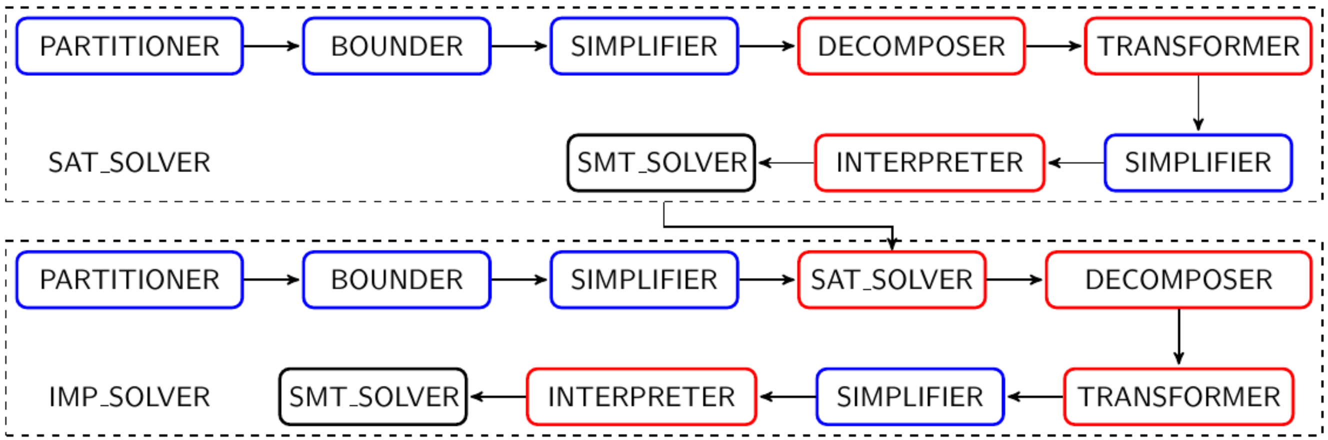
Correctness components



DECOMPOSER: reduce the formula into equivalent formula of height zero



Correctness components

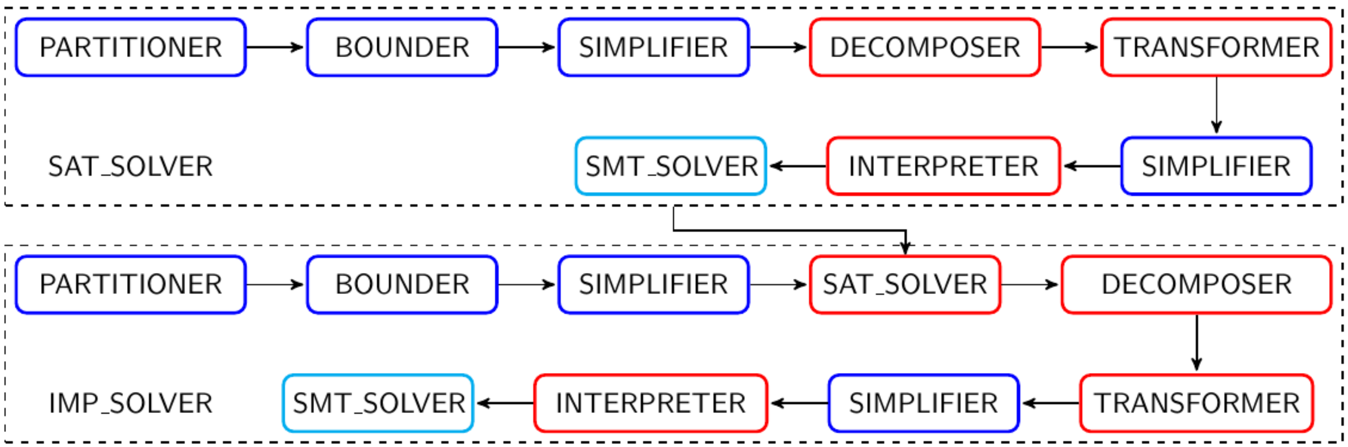


TRANSFORMER + INTERPRETER: transform tree formula of height zero into equivalent Boolean formula.

 → True

 → False

SMT solver component



- SMT_SOLVER: Boolean formulae $\exists\overline{X}.\Phi$ and $\forall\overline{X_1}\exists\overline{X_2}.\Phi$.

Correctness proof for SAT

- Reduce $\mathrm{SAT}(\Phi)$ into $\bigwedge \mathrm{SAT}(\Phi_i)$ where each Φ_i contains a single negative constraint.
- Example:
  - Let $\Phi = e_1 \wedge e_2 \wedge e_3 \wedge d_1 \wedge d_2$ and

    $\Phi_1 = e_1 \wedge e_2 \wedge e_3 \wedge d_1 \qquad \Phi_2 = e_1 \wedge e_2 \wedge e_3 \wedge d_2$

    then

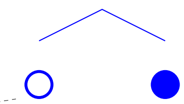
$$\mathrm{SAT}(\Phi) = \mathrm{SAT}(\Phi_1) \wedge \mathrm{SAT}(\Phi_2)$$

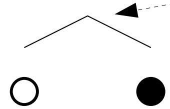
Correctness proof for SAT

- Each Φ_i satisfies the small-model property:
  - Small-model property: P has a solution iff it has a small solution.
  - Theorem: Each Φ_i is satisfiable iff it has a tree solution whose height is at most $\left|\Phi_i\right|$.
- Reduce into equivalent Boolean formula.

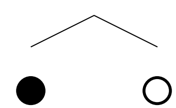
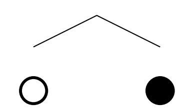
Correctness proof for SAT

Example:

- $\Phi \quad = \quad a \oplus b = \bullet \quad \wedge \quad b \oplus c =$  $\quad \wedge \quad b \neq \circ$

- $|\Phi| \quad = \quad |$  $| \quad = \quad 1$

- $\mathrm{SAT}(\Phi)$ iff Φ has a solution of height at most 1.

- 4 possible candidates: $\circ, \quad \bullet,$ , 

Correctness proof for SAT

Reduce into equivalent Boolean formula:

$$\Phi \quad = \quad a \oplus b = \bullet \quad \wedge \quad b \oplus c = \widehat{\circ \qquad \bullet} \quad \wedge \quad b \neq \circ$$

$$a_1 \oplus b_1 = \bullet \ \wedge \ a_2 \oplus b_2 = \bullet \qquad\qquad b_1 \neq \circ \ \vee \ b_2 \neq \circ$$

$$b_1 \oplus c_1 = \circ \ \wedge \ b_2 \oplus c_2 = \bullet$$

$$a_1, a_2, b_1, b_2, c_1, c_2 \in \{\circ, \bullet\}$$

Correctness for IMP

- The idea is similar:
  - Reduce to smaller problems that satisfy small-model property.

- More complicated:
  - Negative constraints are in both antecedent and consequent.

Bug-free guarantee

- Certified in Coq.
- Optimization components e.g. partitioner are generic => reusable.
- With built-in Boolean solver.
- Around 34k LOC.

Experiment and Result

- Benchmark taken from 3 papers
  - “Barriers in Concurrent Separation Logic”

    (Aquinas Hobor and Cristian Gherghina, 2011).
  - “Decision procedures over sophisticated fractional permissions” (Le et al., 2012).
  - “Automated verification of countdownlatch”

    (Wei-Ngan Chin et al., 2017).
- Test against our old solver (Le et al. 2012).
- 23 program tests + 111 standalone tests.
- Using HIP/SLEEK.

Experiment and Result

File	LOC	# calls	# wrong	Old solver	New solver
MISD_ex1_th1.ss	36	294	48	2.21	2.37
MISD_ex1_th2.ss	36	495	67	4.36	4.48
MISD_ex1_th3.ss	36	726	94	6.95	6.58
MISD_ex1_th4.ss	36	1,003	123	9.09	8.36
MISD_ex1_th5.ss	36	1,320	134	15.74	12.38
MISD_ex2_th1.ss	47	837	107	16.77	18.97
MISD_ex2_th2.ss	52	1,044	157	29.34	26.02
MISD_ex2_th3.ss	87	1,841	260	69.09	64.21
MISD_ex2_th4.ss	105	3,023	374	194.17	194.64
PIPE_ex1_th2.ss	35	283	7	2.49	2.78
PIPE_ex1_th3.ss	44	467	12	4.92	4.65
PIPE_ex1_th4.ss	56	678	15	7.00	7.53
PIPE_ex1_th5.ss	66	931	18	9.67	9.37
SIMD_ex1_v2_th1.ss	74	1,167	281	18.46	17.64
SIMD_ex1_v2_th2.ss	95	2,029	392	63.83	53.50
cdl-ex1a-fm.ss	49	7	0	0.10	0.08
cdl-ex2-fm.ss	50	9	0	0.12	0.09
cdl-ex3-fm.ss	51	10	0	0.11	0.12
cdl-ex4-race.ss	50	5	0	0.09	0.09
cdl-ex4a-race.ss	50	9	0	0.10	0.08
cdl-ex5-deadlock.ss	42	5	0	0.10	0.10
cdl-ex5a-deadlock.ss	42	9	0	0.08	0.08
ex-fork-join.ss	25	47	22	0.19	0.16
total		**10,252**	534	455.01	434.30

Table 1. Evaluation of our procedures using HIP/SLEEK

Experiment and Result

Old solver has bugs:

- 534 / 10,252 : **5.2%**.
- HIP/SLEEK: code rot, poor error signaling/handling.
- Permission solver: correctness bug for handling negative constraints.

Experiment and Result

New solver:

- Faster (434 seconds vs. 455 seconds): **4.6%**.
- Bug-free.

Conclusion

Two decision procedures to handle SAT and IMP for tree share permissions:

- Certified (bug-free).
- Optimized (faster than old solver).
- Handle general negative constraints.

Future work

New (certified) procedures to handle:

- First-order theory of $\langle \mathbb{T}, \oplus \rangle$.
- Formulae from the combined structure of tree share with addition and multiplication.

Thank you for listening!

Correctness proof for IMP

Checking $\Phi_1 \vdash \Phi_2$

- Let l_i be the list of disequations of Φ_i
- Let $[\Phi_i]$ be Φ_i with all equations and without disequations
- Let $[\Phi_i^k]$ be Φ_i with all equations and with a single disequation $d_k \in l_i$

Correctness proof for IMP

Assume $\mathrm{SAT}(\Sigma_1) \quad \wedge \quad [\Phi_1] \vdash [\Phi_2]$.Three cases:

- $l_2 = \mathrm{nil}$: is equivalent to $[\Phi_1] \vdash [\Phi_2]$
- $l_1 = \mathrm{nil} \wedge l2 \neq \mathrm{nil}$: is equivalent to

$$\bigwedge_{d_k \in l_2} [\Phi_1] \vdash [\Phi_2^k]$$

- $l_1 \neq \mathrm{nil} \wedge l_2 \neq \mathrm{nil}$:
  - If $[\Phi_1] \vdash \Phi_2$ (case 2) then Yes.
  - Else equivalent to

$$\bigwedge_{d_k \in l_2} (\bigvee_{d'_h \in l_1} [\Phi_1^h] \vdash [\Phi_2^k])$$

Correctness proof for IMP

Small model property:

- Theorem: Each $[\Phi_1] \vdash [\Phi_2], [\Phi_1] \vdash [\Phi_2^j], [\Phi_1^i] \vdash [\Phi_2^j]$

  holds iff it holds for all solution of height at most the height of the constraint.