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
Concurrent Separation Logic

1. Reason about correctness of concurrent programs.


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

address $\mapsto$ value

• Disjoint conjunction

\[
x \mapsto 1 \ast y \mapsto 1
\]
Some basics

Shape inductive predicates

\[ \text{list}(x) \overset{\text{def}}{=} (x = \text{null}) \lor \]
\[ \exists d, x_1. \ x \mapsto (d, x_1) \star \text{list}(x_1) \]

\[ \text{tree}(x) \overset{\text{def}}{=} (x = \text{null}) \lor \]
\[ \exists d, x_1, x_2. \ x \mapsto (d, x_1, x_2) \star \text{tree}(x_1) \star \text{tree}(x_2) \]
Some basics

Frame Rule

\[
\{ P \} \ c \ \{ Q \} \quad \text{mod}(c) \cap \text{fv}(F) = \emptyset
\]

\[
\{ F \ast P \} \ c \ \{ F \ast Q \}
\]
Some basics

Parallel Rule

\[
\begin{align*}
\{P_1\} & \ c_1 \ \{Q_1\} & \ f_v(c_1, P_1, Q_1) \cap \text{modified}(c_2) = \emptyset \\
\{P_2\} & \ c_2 \ \{Q_2\} & \ f_v(c_2, P_2, Q_2) \cap \text{modified}(c_1) = \emptyset \\
& \ \{P_1 \ast P_2\} & \ c_1 \ || \ c_2 \ \{Q_1 \ast Q_2\}
\end{align*}
\]

Parallel
Permissions in CSL

• Fractional maps-to

\[ \text{address} \xrightarrow{\text{permission}} \text{value} \]

• Rational permission model \( \langle (0, 1], + \rangle \):
  - \( \pi \in (0, 1] \), 1 : WRITE, (0, 1) : READ
  - Join/split permissions:

\[
\begin{align*}
  x \xrightarrow{\pi_1 + \pi_2} v & \quad \parallel \quad x \xrightarrow{\pi_1} v \quad \star x \xrightarrow{\pi_2} v \\
  x \xrightarrow{1} v & \quad \parallel \quad x \xrightarrow{0.5} v \quad \star x \xrightarrow{0.5} v
\end{align*}
\]

• Example:
Shortcomings of rational permissions

Lack of disjointness:

- In traditional SL:
  \[ x \mapsto v \star x \mapsto v \vdash \bot \]

- With rational permissions:
  \[ x \overset{0.5}{\mapsto} v \star x \overset{0.5}{\mapsto} v \vdash x \overset{1}{\mapsto} v \]
Shortcomings of rational permissions

Deformation of shape predicates

\[ \text{tree}(x, \pi) \overset{\text{def}}{=} (x = \text{null}) \lor \exists d, x_1, x_2. x \xmapsto{\pi} (d, x_1, x_2) \ast \text{tree}(x_1, \pi) \ast \text{tree}(x_2, \pi) \]

\[
\begin{array}{c|c|c}
\text{a} & (1, b, c) & 0.3 \\
\text{b} & (1, \text{null}, d) & 0.3 \\
\text{c} & (1, d, \text{null}) & 0.3 \\
\text{d} & (1, \text{null}, \text{null}) & 0.6 \\
\end{array}
\]

\[ \text{tree}(a, 0.3) \]
Shortcomings of rational permissions

Poor support of complete decision procedures

- Not finitely axiomatized in first-order logic.
- The addition group \(<\mathbb{Q},+>\) 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 $\oplus$

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

  $\circ \oplus \circ = \circ$ $\bullet \oplus \circ = \circ \oplus \bullet = \bullet$ $\bullet \oplus \bullet$ undefined

- General case: leaf-wise
Tree permission model

\[(T, \oplus)\]

- tree shares
- tree addition
Why permission solver?
Previous work

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

  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: \( \exists \overline{X}. \Phi \)
  
  • IMP: \( \forall \overline{X}_1. (\Phi_1 \Rightarrow \exists \overline{X}_2. \Phi_2) \)
    
    where \( \Phi = \bigwedge a \oplus b = c \land \bigwedge a' \oplus b' \neq c' \)

- 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 $\text{SAT}(\Phi)$ into $\bigwedge \text{SAT}(\Phi_i)$ where each $\Phi_i$ contains a single negative constraint.

• Example:
  - Let $\Phi = e_1 \land e_2 \land e_3 \land d_1 \land d_2$ and
    $\Phi_1 = e_1 \land e_2 \land e_3 \land d_1$ \quad $\Phi_2 = e_1 \land e_2 \land e_3 \land d_2$
  then
    $\text{SAT}(\Phi) = \text{SAT}(\Phi_1) \land \text{SAT}(\Phi_2)$
Correctness proof for SAT

• Each $\Phi_i$ satisfies the small-model property:
  
  – Small-model property: $P$ has a solution iff it has a small solution.

  – Theorem: Each $\Phi_i$ is satisfiable iff it has a tree solution whose height is at most $|\Phi_i|$.

• Reduce into equivalent Boolean formula.
Correctness proof for SAT

Example:

- $\Phi = a \uplus b = \bullet \land b \uplus c = \circ \; \triangleleft \; \bullet \land b \neq \circ$

- $|\Phi| = |\circ \; \triangleleft \; \bullet| = 1$

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

- 4 possible candidates: $\circ$, $\bullet$, $\bullet \; \circ$, $\circ \; \bullet$
Correctness proof for SAT

Reduce into equivalent Boolean formula:

\[ \Phi = a \oplus b = \bullet \land b \oplus c = \circ \land b \neq \circ \]

\[ a_1 \oplus b_1 = \bullet \land a_2 \oplus b_2 = \bullet \]

\[ b_1 \oplus c_1 = \circ \land b_2 \oplus c_2 = \bullet \]

\[ a_1, a_2, b_1, b_2, c_1, c_2 \in \{\circ, \bullet\} \]
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
  – “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

<table>
<thead>
<tr>
<th>File</th>
<th>LOC</th>
<th># calls</th>
<th># wrong</th>
<th>Old solver</th>
<th>New solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISD.ex1.th1.ss</td>
<td>36</td>
<td>294</td>
<td>48</td>
<td>2.21</td>
<td>2.37</td>
</tr>
<tr>
<td>MISD.ex1.th2.ss</td>
<td>36</td>
<td>495</td>
<td>67</td>
<td>4.36</td>
<td>4.48</td>
</tr>
<tr>
<td>MISD.ex1.th3.ss</td>
<td>36</td>
<td>726</td>
<td>94</td>
<td>6.95</td>
<td>6.58</td>
</tr>
<tr>
<td>MISD.ex1.th4.ss</td>
<td>36</td>
<td>1,003</td>
<td>123</td>
<td>9.09</td>
<td>8.36</td>
</tr>
<tr>
<td>MISD.ex1.th5.ss</td>
<td>36</td>
<td>1,320</td>
<td>134</td>
<td>15.74</td>
<td>12.38</td>
</tr>
<tr>
<td>MISD.ex2.th1.ss</td>
<td>47</td>
<td>837</td>
<td>107</td>
<td>16.77</td>
<td>18.97</td>
</tr>
<tr>
<td>MISD.ex2.th2.ss</td>
<td>52</td>
<td>1,044</td>
<td>157</td>
<td>29.34</td>
<td>26.02</td>
</tr>
<tr>
<td>MISD.ex2.th3.ss</td>
<td>87</td>
<td>1,841</td>
<td>260</td>
<td>69.09</td>
<td>64.21</td>
</tr>
<tr>
<td>MISD.ex2.th4.ss</td>
<td>105</td>
<td>3,023</td>
<td>374</td>
<td>194.17</td>
<td>194.64</td>
</tr>
<tr>
<td>PIPE.ex1.th2.ss</td>
<td>35</td>
<td>283</td>
<td>7</td>
<td>2.49</td>
<td>2.78</td>
</tr>
<tr>
<td>PIPE.ex1.th3.ss</td>
<td>44</td>
<td>467</td>
<td>12</td>
<td>4.92</td>
<td>4.65</td>
</tr>
<tr>
<td>PIPE.ex1.th4.ss</td>
<td>56</td>
<td>678</td>
<td>15</td>
<td>7.00</td>
<td>7.53</td>
</tr>
<tr>
<td>PIPE.ex1.th5.ss</td>
<td>66</td>
<td>931</td>
<td>18</td>
<td>9.67</td>
<td>9.37</td>
</tr>
<tr>
<td>SIMD.ex1.v2.th1.ss</td>
<td>74</td>
<td>1,167</td>
<td>281</td>
<td>18.46</td>
<td>17.64</td>
</tr>
<tr>
<td>SIMD.ex1.v2.th2.ss</td>
<td>95</td>
<td>2,029</td>
<td>392</td>
<td>63.83</td>
<td>53.50</td>
</tr>
<tr>
<td>cdl-ex1a-fm.ss</td>
<td>49</td>
<td>7</td>
<td>0</td>
<td>0.10</td>
<td>0.08</td>
</tr>
<tr>
<td>cdl-ex2-fm.ss</td>
<td>50</td>
<td>9</td>
<td>0</td>
<td>0.12</td>
<td>0.09</td>
</tr>
<tr>
<td>cdl-ex3-fm.ss</td>
<td>51</td>
<td>10</td>
<td>0</td>
<td>0.11</td>
<td>0.12</td>
</tr>
<tr>
<td>cdl-ex4-race.ss</td>
<td>50</td>
<td>5</td>
<td>0</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>cdl-ex4a-race.ss</td>
<td>50</td>
<td>9</td>
<td>0</td>
<td>0.10</td>
<td>0.08</td>
</tr>
<tr>
<td>cdl-ex5-deadlock.ss</td>
<td>42</td>
<td>5</td>
<td>0</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>cdl-ex5a-deadlock.ss</td>
<td>42</td>
<td>9</td>
<td>0</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>ex-fork-join.ss</td>
<td>25</td>
<td>47</td>
<td>22</td>
<td>0.19</td>
<td>0.16</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td>10,252</td>
<td></td>
<td>534</td>
<td>455.01</td>
<td>434.30</td>
</tr>
</tbody>
</table>

### 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 $\langle T, \oplus \rangle$.
- 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 \( \Phi_i \)

- Let \([\Phi_i]\) be \( \Phi_i \) with all equations and without disequations

- Let \([\Phi_i^k]\) be \( \Phi_i \) with all equations and with a single disequation \( d_k \in l_i \)
Correctness proof for IMP

Assume $SAT(\Sigma_1) \land [\Phi_1] \models [\Phi_2]$. Three cases:

- $l_2 = \text{nil}$: is equivalent to $[\Phi_1] \models [\Phi_2]$

- $l_1 = \text{nil} \land l_2 \neq \text{nil}$: is equivalent to

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

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

$$\bigwedge_{d_k \in l_2, d'_h \in l_1} \left( \bigvee [\Phi_1^h] \models [\Phi_2^k] \right)$$
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.