Salsa: Beyond Model Checking*

Ramesh Bharadwaj
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375 USA
ramesh@itd.nrl.navy.mil

Salsa web site: www.reactive-systems.com/salsa

* Work Supported by the Office of Naval Research.
# Salsa: Beyond Model Checking

The original document contains color images.
What does Salsa do?

SAL Specification $S$

Potential Invariant $I$

Is $I$ an invariant of $S$?

- Yes!
- No/Counterexample
Why does Salsa work?

Goals:

- As easy to use as a model checker
- Scales like a theorem prover

Practical usage of Salsa:

- NRL: Cryptographic Device (CD) [27]
- Reactive-Systems Inc/Ford: Simulink/Stateflow specifications
- SUNY Stony Brook: CAN bus protocol
Given the following:

- A system description
- A set of environmental assumptions
- A set of required properties (one-state or two-state)

Verification is the process of:

- Extracting models from the system description. *Sufficient to establish the properties of interest.*
- Applying a verification tool to the model to verify/refute properties.

Very likely that a property is not provable (or wrong).

A tool should provide *diagnostic information.*
How can a tool help?

1. **Diagnostic information**
   Counterexamples.

2. **Comprehensible diagnostics**
   In the “language” of the original description.

3. **Compact diagnostics**

4. **No misdiagnoses**
   Very hard to achieve in practice.
Conventional Wisdom

Model Checking

- Automatic, easy to use, counterexamples.

Theorem Proving

- Too general, too expensive, hard to use,..

But the reality is...
Attributes of Model Checking:

1. Completeness
2. Termination
3. Diagnostic information
4. State explosion problem

Attributes of Theorem Proving:

1. Incompleteness
   Auxiliary lemmas.
2. Not guaranteed to terminate
   Decision procedures.
3. Diagnostic information?
   Make it comprehensible to layfolk.
4. Infinite State
Combines Model Checking and Theorem Proving

**Strengths:**

- Reliance on *decision procedures*
- Combination of decision procedures
- Guaranteed termination
- Counterexamples
- Push-button automation

**Weaknesses:**

- Counterexample not a trace
- Incomplete – counterexamples must be validated
Process for using Salsa

Is $I$ an invariant of $S$?

Yes

No/Counterexample

Salsa

Potentially Invariant $I$

New $I = I \land L$

Produce auxiliary Lemma $L$
(Manually or with automatic generator)

Is Counterexample Reachable?

Yes

No
Salsa vs. Model Checking

- Can handle specs too large for model checkers
  (single pre-image vs fixed point computation)

- More automatic!
  (no manual abstractions)

- Counterexamples
Related Work

- CCC of the SCR Toolset (tautology checking vs UC) [22,23,24]
- TAME/PVS [3,30]
- InVeSt [5,6]
- Graf’s tools [21,32]
- STeP [11]
- SPIN and SMV on software specs [2,9,16,22]

Notes:  
- First four designed for ease-of use.
- First three provide counterexamples.
- STeP requires user interaction.
- Model checkers require the application of abstraction [2,7,8,9,10,27]; they may not always scale (i.e., neither verify/refute) [27].
## Empirical Results

<table>
<thead>
<tr>
<th>Specification</th>
<th>Number of VCs</th>
<th>Number of Vars</th>
<th>Number of Constrs</th>
<th>Time (in S) to Check Disj</th>
<th>Number of Failed VCs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BDD</td>
<td>SCR Tool</td>
<td>Salsa Tool</td>
<td>SCR Tool</td>
<td>Salsa Tool</td>
</tr>
<tr>
<td>Specification containing mostly booleans and enumerated types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>safety-injection</td>
<td>13</td>
<td>16</td>
<td>3</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>bomb-release-1</td>
<td>12</td>
<td>34</td>
<td>9</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>a7-modes</td>
<td>6171</td>
<td>158</td>
<td>3</td>
<td>145.9</td>
<td>68.9</td>
</tr>
<tr>
<td>Specification containing mostly numerical variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>autopilot</td>
<td>29</td>
<td>50</td>
<td>27</td>
<td>1.5</td>
<td>1.0</td>
</tr>
<tr>
<td>home-heating</td>
<td>98</td>
<td>112</td>
<td>55</td>
<td>$\infty_t$</td>
<td>4.8</td>
</tr>
<tr>
<td>cruise-control</td>
<td>123</td>
<td>114</td>
<td>75</td>
<td>21.0</td>
<td>3.6</td>
</tr>
<tr>
<td>navy-1</td>
<td>252</td>
<td>115</td>
<td>78</td>
<td>322.8</td>
<td>59.7</td>
</tr>
<tr>
<td>navy</td>
<td>397</td>
<td>147</td>
<td>102</td>
<td>390.1</td>
<td>198.2</td>
</tr>
<tr>
<td>bomb-release-2</td>
<td>339</td>
<td>319</td>
<td>230</td>
<td>$\infty_t$</td>
<td>246.0</td>
</tr>
<tr>
<td>wcp</td>
<td>58</td>
<td>611</td>
<td>104</td>
<td>-</td>
<td>77.4</td>
</tr>
</tbody>
</table>
## Empirical Results (con’t)

<table>
<thead>
<tr>
<th>Spec.</th>
<th>Props. or #</th>
<th>Time (in seconds)</th>
<th>Props. True?</th>
<th>Aux. Lemmas?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Salsa</td>
<td>SPIN</td>
<td>SMV</td>
</tr>
<tr>
<td>sis</td>
<td>4</td>
<td>0.8</td>
<td>36.0</td>
<td>155.0</td>
</tr>
<tr>
<td>brel1</td>
<td>2</td>
<td>1.3</td>
<td>$\infty_t$</td>
<td>$\infty_t$</td>
</tr>
<tr>
<td>autop</td>
<td>2</td>
<td>1.5</td>
<td>$\infty_t$</td>
<td>$\infty_t$</td>
</tr>
<tr>
<td>navy</td>
<td>7</td>
<td>396.0</td>
<td>$\infty_t$</td>
<td>-</td>
</tr>
<tr>
<td>wcp</td>
<td># 303</td>
<td>295.4</td>
<td>$\infty_t$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td># 304</td>
<td>923.3*</td>
<td>$\infty_t$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td># 305</td>
<td>2.4</td>
<td>$\infty_t$</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusions

- Consistency checking goals achieved
  - Faster
  - Handles integers
  - Able to handle bigger specs
- Bonus: also handles user properties.
  - Handles specs too big for model checkers
  - Seems to be “in the ball park” with PVS
- Weaknesses (w.r.t model checkers):
  - Incompleteness
  - Two-state counterexamples rather than trace from start state.
An innovation has three stages of acceptance: First, it is dismissed as rubbish, then it’s merely nonviable, and finally it’s obvious and trivial – “What we’ve done all along.”

John Vlissides.