**Learning to Solve SMT Formulas**

Mislav Balunović, Pavol Bielik, Martin Vechev

**fastsmt.ethz.ch**

---

**SMT Theories**
- Integers
- Reals
- Arrays
- Bitvectors
- Strings

**SMT Solvers and Formulas**
Find an assignment to all free variables in $\phi$ such that $\phi$ evaluates to true

**Applications**
- SW/SW Verification
- Neural Network Verification
- Symbolic Execution
- Program Synthesis

**Formula Measures**
- Formula Solving Time
- Percentage of Formulas Solved

**How are Formulas Solved?**
Formulas are solved by applying sequence of transformations until a base form (true/false) is reached

**Learning**
- Search over sequences of tactics

**Given a formula $\phi$ find sequential strategies that are fast at solving $\phi$**

Please the learning problem as a tree search over sequences of tactics.

Different tactics often apply different sequences of tactics and so be redundant.

**Examples of Strategies**
- Simplify with arith:true, num:true; zero_bound; lin2pb; pb2bv; bit_blast; sat
- Simplify with local_ctx:true; sat; bit_blast; sat
- Learn a policy to select next tactic

**Training algorithm based on Dagger**

**Formula Representation**
BGW/Skip/Trim/AST

**Loss** = weighted average of cross-entropy loss + mean-square-error metrics

**Evaluation: Learning**
Find fastest strategy for each formula

<table>
<thead>
<tr>
<th>Number of Samples</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td>1.33</td>
</tr>
<tr>
<td>40</td>
<td>1.67</td>
</tr>
<tr>
<td>60</td>
<td>2.00</td>
</tr>
<tr>
<td>80</td>
<td>2.33</td>
</tr>
<tr>
<td>100</td>
<td>2.67</td>
</tr>
</tbody>
</table>

- Formula parameters denote that all of the formulas have speedup at least 2x.

---

**Evaluation: Synthesis**

**Speedup over State-of-the-art 32 Solver**

<table>
<thead>
<tr>
<th>Formula Solving Speedup Percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
</tr>
<tr>
<td>25%</td>
</tr>
<tr>
<td>50%</td>
</tr>
<tr>
<td>75%</td>
</tr>
<tr>
<td>100%</td>
</tr>
</tbody>
</table>

**Synthesized Strategy vs Handcrafted Z3 Strategy**

**Training**
- 10 seconds
- 10 minutes

**Testing**
- 10 seconds
- 10 minutes

---

**Our Work**
- Replace the handcrafted strategies by learning to solve the SMT formulas

**Learning**
- Find fastest strategy
- Synthesize an interpretable program with branches that selects a strategy that performs best on a dataset of formulas

**Policy**
- Enables integration with state-of-the-art SMT solvers

**Strategy with Branches**
- Single strategy with synthesized branch for each state with multiple outgoing edges

---

**Syntax of Strategy Language used by Z3 solver**

**Examples**
- A strategy is defined for all formulas

---

**Conclusion**
- Our approach enables integration with state-of-the-art SMT solvers
- Synthesizes a strategy that achieves 2.6x speedup on average
- Enables automated learning of effective strategy language

---

**Acknowledgements**
- We thank fastsmt.ethz.ch for providing the datasets and the tooling.
- We gratefully acknowledge the support of the ETH Zurich FastSMT Laboratory.

---

**References**
1. S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. AISTATS’11
2. Many thanks to the anonymous reviewers for their constructive feedback.