domain-specific SMT solving for neural network verification (or anything else)

Lindsey Kuper
UC Santa Cruz
sometimes it’s worth trading generality for productivity + performance

[Olukotun et al., 2012]
sometimes it’s worth trading generality for productivity + performance
sometimes it’s worth trading generality for productivity + performance

[Olukotun et al., 2012]
sometimes it’s worth trading generality for productivity + performance

[Olukotun et al., 2012]
Policy Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian*, Jessica Lopez, Jeffrey S. Brush, Michael P. Owen and Mykel J. Kochenderfer

*Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305
1Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, 20723
2Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, 02420

Abstract—One approach to designing the decision making logic for an aircraft collision avoidance system is to frame the problem as Markov decision process and optimize the system using dynamic programming. The resulting strategy can be represented as a numeric table. This methodology has been used in the development of the ACAS X family of collision avoidance systems for manned and unmanned aircraft. However, due to the high dimensionality of the state space, discretizing the state variables can lead to very large tables. To improve storage efficiency, we propose two approaches for compressing the lookup table. The first approach exploits redundancy in the table. The table is decomposed into a set of lower-dimensional tables, some of which can be represented by single tables in areas where the lower-dimensional tables are identical or nearly identical with respect to a similarity metric. The second approach uses a deep neural network to learn a complex non-linear function approximation of the table. With the use of an asymmetric loss function and a

is extremely large, requiring hundreds of gigabytes of floating point storage. A simple technique to reduce the size of the score table is to downsample the table after dynamic programming. To minimize the deterioration in decision quality, states are removed in areas where the variation between values in the table are smooth. This allows the table to be downsampled with only minor impact on overall decision performance. The downsampling reduces the size of the table by a factor of 180 from that produced by dynamic programming. For the rest of this paper, we refer to the downsampled ACAS Xa horizontal table as our baseline, original table.

Even after downsampling, the current table requires over 2GB of floating point storage. Discretized score tables like this have been compressed with Gaussian processes [6] and
Policy Compression for Aircraft Collision Avoidance Systems

to a similarity metric. The second approach uses a deep neural network to learn a complex non-linear function approximation of the table. With the use of an asymmetric loss function and a preserving the relative preferences of the possible advisories for each state. As a result, the table can be approximately represented by only the parameters of the network, which reduces the required storage space by a factor of 1000. Simulation studies show that system performance is very similar using either concrete representation. Although there are significant certification concerns with neural network representations, which may be addressed in the future, these results indicate a promising way.
input: sensor data once per second

output: one of five resolution advisories (COC, weak left, weak right, strong left, strong right)

Source: www.ll.mit.edu/publications/technotes/TechNote_ACASX.pdf
input: sensor data once per second
output: one of five resolution advisories (COC, weak left, weak right, strong left, strong right)

\[ \rho \]: distance from ownship to intruder
\[ \theta \]: angle to intruder
\[ \psi \]: heading angle of intruder
\[ v_{\text{own}} \]: speed of ownship
\[ v_{\text{int}} \]: speed of intruder
\[ \tau \]: time until loss of vertical separation
\[ a_{\text{prev}} \]: previous advisory

~120M 7-dimensional states

Source: www.ll.mit.edu/publications/technotes/TechNote_ACASX.pdf
\~120M 7-dimensional states

\( \rho \): distance from ownship to intruder
\( \theta \): angle to intruder
\( \psi \): heading angle of intruder
\( v_{\text{own}} \): speed of ownship
\( v_{\text{int}} \): speed of intruder
\( \tau \): time until loss of vertical separation
\( a_{\text{prev}} \): previous advisory

needs 100s of GBs of storage—too big to fit in memory on verified hardware

Source: www.ll.mit.edu/publications/technotes/TechNote_ACASX.pdf
9 fully-connected layers with ReLU activations ($f(x) = \max(0, x)$)
9 fully-connected layers with ReLU activations ($f(x) = \max(0, x)$)

COC: 0.230892
weak right: 0.703941

[Julian et al., 2016]
verification strategy

**SAT**: determine if a Boolean formula (containing only Boolean variables, parens, $\land$, $\lor$, $\neg$) is satisfiable

**SMT**: determine satisfiability of a formula with respect to some theory (e.g., theory of linear real arithmetic)

---

SAT formula:

$$\ldots (n == 5 + m) \lor (p \land \neg q) \ldots$$

Description of network

Property to be shown

“if intruder is near and approaching from left, network will advise ‘strong right’”

SMT Solver

- satisfiable
- unsatisfiable
the virtues of laziness

eager approach: convert whole SMT formula to SAT formula immediately, then solve with SAT solver

lazy approach: use *theory solvers*, each specific to a particular theory

Source: fm.csl.sri.com/SSFT16/slides.pdf
the virtues of laziness

eager approach: convert whole SMT formula to SAT formula immediately, then solve with SAT solver

lazy approach: use theory solvers, each specific to a particular theory

key idea: to exploit domain knowledge and unlock efficiency, use a theory solver specifically for handling neural networks
lazily handling ReLU activations

[Katz et al., 2017]

```
... (n == 5 + m) \lor (p \land \neg q) ...
```

SMT Solver

satisfiable  unsatisfiable
lazily handling ReLU activations

[Katz et al., 2017]

“if intruder is near and approaching from left, network will advise ‘strong right’”

SMT formula

\[ ... (n == 5 + m) \lor (p \land \neg q) ... \]

SMT solver

LP solver \quad \text{Solver core} \quad \text{SAT solver}

satisfiable \quad \text{unsatisfiable}
lazily handling ReLU activations

[Katz et al., 2017]

Description of network

Property to be shown

“if intruder is near and approaching from left, network will advise ‘strong right’”

SMT formula

... (n == 5 + m) ∨ (p ∧ ¬q) ...

SMT solver

LP solver ↔ Solver core ↔ SAT solver

satisfiable unsatisfiable

ReLU constraints like $y = \max(0, x)$ can only be encoded as disjunctions:

$$(x \geq 0 \land y = x) \lor (x < 0 \land y = 0)$$

lazily handling ReLU activations

[Katz et al., 2017]

Description of network

Property to be shown

“if intruder is near and approaching from left, network will advise ‘strong right’”

SMT formula

\[ ... (n == 5 + m) \lor (p \land \neg q) ... \]

SMT solver

LP + ReLU solver

Solver core

SAT solver

satisfiable

unsatisfiable

ReLU constraints like \( y = \max(0, x) \) can only be encoded as disjunctions:

\( (x \geq 0 \land y = x) \lor (x < 0 \land y = 0) \)
lazily handling ReLU activations

[Katz et al., 2017]

<table>
<thead>
<tr>
<th>Property description</th>
<th>Does it hold?</th>
<th>Solver time</th>
<th>Max. ReLU split depth (out of 300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“if intruder is directly ahead and is moving towards ownship, network will not advise COC”</td>
<td>✓</td>
<td>7.8h</td>
<td>22</td>
</tr>
<tr>
<td>“if intruder is near and approaching from left, network advises ‘strong right’”</td>
<td>✓</td>
<td>5.4h</td>
<td>46</td>
</tr>
<tr>
<td>“if intruder is sufficiently far away, network advises COC”</td>
<td>✓</td>
<td>50h</td>
<td>50</td>
</tr>
<tr>
<td>“for large vertical separation and previous ‘weak left’ advisory, network will either advise COC or continue advising ‘weak left’”</td>
<td>✗</td>
<td>11h</td>
<td>69</td>
</tr>
</tbody>
</table>

ReLUs can be handled as disjunctions

ReLU constraints like \( y = \max(0, x) \) can only be encoded as disjunctions:

\[(x \geq 0 \land y = x) \lor (x < 0 \land y = 0)\]
the distributed consistency model zoo

[Paolo Viotti and Marko Vukolić, Consistency in Non-Transactional Distributed Storage Systems]
the distributed consistency model zoo

(smaller regions admit fewer executions)
name that consistency bug/feature!

initial balance: $25

① deposit $25

② withdraw $30

balance: $20

initial balance: $25

① deposit $25

② withdraw $30

balance: $20
name that consistency bug/feature!

Diagram:

1. **deposit $25**
2. **withdraw $30**

**FIFO consistency violation**

Initial balance: $25

Balance: $20

Initial balance: $25

Balance: $20
name that consistency bug/feature!

Initial balance: $25

1. Deposit $100
2. Withdraw $50

Balance: $75
name that consistency bug/feature!

- Initial balance: $25
  - ① Deposit $100
  - ② Withdraw $50
  - Balance: $75
name that consistency bug/feature!

Initial balance: $25

1. Deposit $100
2. Withdraw $50

Balance: $75

Initial balance: $25

1. Deposit $100
2. Withdraw $50

Balance: $75

FIFO OK
causal consistency violation
name that consistency bug/feature!

initial balance: $25

① deposit $100

② withdraw $30

balance: $95

initial balance: $25

① deposit $100

② withdraw $30

balance: $95
name that consistency bug/feature!

initial balance: $25

① deposit $100

② withdraw $30

balance: $95

FIFO OK, causal OK

initial balance: $25

① deposit $100

② withdraw $30

balance: $95
name that consistency bug/feature!

initial balance: $25

① deposit $100

② withdraw $30

balance: $95

FIFO OK, causal OK
linearizability violation

initial balance: $25

① deposit $100

② withdraw $30

balance: $95
consistency contracts
[Sivaramakrishnan et al., 2015]

\[\psi \in \text{Contract} ::= \forall (x : \tau).\psi \mid \forall x.\psi \mid \pi\]
\[\tau \in \text{EffType} ::= \text{Op} \mid \tau \lor \tau\]
\[\pi \in \text{Prop} ::= \text{true} \mid R(x, y) \mid \pi \lor \pi\]
\[\mid \pi \land \pi \mid \pi \Rightarrow \pi\]
\[R \in \text{Relation} ::= \text{vis} \mid \text{so} \mid \text{sameobj} \mid =\]
\[\mid R \cup R \mid R \cap R \mid R^+\]

\[x, y, \hat{\eta} \in \text{EffVar} \quad \text{Op} \in \text{OperName}\]
consistency contracts
[Sivaramakrishnan et al., 2015]

\[
\psi \in \text{Contract} ::= \forall (x : \tau).\psi \mid \forall x.\psi \mid \pi
\]
\[
\tau \in \text{EffType} ::= \text{Op} \mid \tau \lor \tau
\]
\[
\pi \in \text{Prop} ::= \text{true} \mid R(x,y) \mid \pi \lor \pi
\]
\[
| \pi \land \pi \mid \pi \Rightarrow \pi
\]
\[
R \in \text{Relation} ::= \text{vis} \mid \text{so} \mid \text{sameobj} \mid =
\]
\[
| R \cup R \mid R \cap R \mid R^+
\]

\[
x, y, \hat{\eta} \in \text{EffVar} \quad \text{Op} \in \text{OperName}
\]
a contract is a first-order logic formula
universal quantification over EffVars allowed
consistency contracts
[Sivaramakrishnan et al., 2015]

A contract is a first-order logic formula with universal quantification over EffVars allowed.

\[
\psi \in \text{Contract} ::= \forall (x : \tau).\psi \mid \forall x.\psi \mid \pi
\]

\[
\tau \in \text{EffType} ::= \text{Op} \mid \tau \lor \tau
\]

\[
\pi \in \text{Prop} ::= \text{true} \mid R(x, y) \mid \pi \lor \pi
\]

\[
R \in \text{Relation} ::= \text{vis} \mid \text{so} \mid \text{sameobj} \mid \pi
\]

\[
| \pi \land \pi \mid \pi \Rightarrow \pi
\]

\[
| R \cup R \mid R \cap R \mid R^+
\]

\[
x, y, \hat{\eta} \in \text{EffVar} \quad \text{Op} \in \text{OperName}
\]
consistency contracts
[Sivaramakrishnan et al., 2015]

A contract is a first-order logic formula with universal quantification over EffVars allowed. See: Sebastian Burckhardt's book.

\[
\psi \in \text{Contract} ::= \forall (x : \tau). \psi \mid \forall x. \psi \mid \pi
\]

\[
\tau \in \text{EffType} ::= \text{Op} \mid \tau \lor \tau
\]

\[
\pi \in \text{Prop} ::= \text{true} \mid R(x, y) \mid \pi \lor \pi
\]

\[
\mid \pi \land \pi \mid \pi \Rightarrow \pi
\]

\[
R \in \text{Relation} ::= \text{vis} \mid \text{so} \mid \text{sameobj} \mid =
\]

\[
\mid R \cup R \mid R \cap R \mid R^+
\]

\[
x, y, \hat{\eta} \in \text{EffVar} \quad \text{Op} \in \text{OperName}
\]

Example contracts for bank account operations
(\(\hat{\eta}\) is the current operation/effect)

**For withdraw:**
\[
\forall (a : \text{withdraw}).
\forall (x : \text{Obj}).
\exists (\eta).\; \text{sameobj}(a, \eta) \Rightarrow a = \eta \lor \text{vis}(a, \eta) \lor \text{vis}(\eta, a)
\]

**For getBalance:**
\[
\forall (a : \text{deposit}), (b : \text{withdraw}), (c : \text{deposit} \lor \text{withdraw}).
\forall (x : \text{Obj}).
\forall (\eta).\; (\text{vis}(a, b) \land \text{vis}(b, \eta) \Rightarrow \text{vis}(a, \eta))
\land ((\text{so} \cap \text{sameobj})(c, \eta) \Rightarrow \text{vis}(c, \eta))
\]
consistency contracts
[Sivaramakrishnan et al., 2015]

\[
\psi \in \text{Contract} ::= \forall (x : \tau).\psi \mid \forall x.\psi \mid \pi
\]
\[
\tau \in \text{EffType} ::= \text{Op} \mid \tau \lor \tau
\]
\[
\pi \in \text{Prop} ::= \text{true} \mid R(x, y) \mid \pi \lor \pi
\]
\[
\mid \pi \land \pi \mid \pi \Rightarrow \pi
\]
\[
R \in \text{Relation} ::= \text{vis} \mid \text{so} \mid \text{sameobj} \mid = \mid R \cup R \mid R \cap R \mid R^+
\]

x, y, \hat{\eta} \in \text{EffVar} \quad \text{Op} \in \text{OperName}

example contracts for bank account operations
(\(\hat{\eta}\) is the current operation/effect)

for withdraw:
\[
\forall (a : \text{withdraw}).
\]
\[
\text{sameobj}(a, \hat{\eta}) \Rightarrow a = \hat{\eta} \lor \text{vis}(a, \hat{\eta}) \lor \text{vis}(\hat{\eta}, a)
\]

for getBalance:
\[
\forall (a : \text{deposit}), (b : \text{withdraw}), (c : \text{deposit} \lor \text{withdraw}).
\]
\[
(\text{vis}(a, b) \land \text{vis}(b, \hat{\eta}) \Rightarrow \text{vis}(a, \hat{\eta}))
\]
\[
\land ((\text{so} \cap \text{sameobj})(c, \hat{\eta}) \Rightarrow \text{vis}(c, \hat{\eta}))
\]

(what’s the contract for deposit? why?)
specialized theory solvers are an old idea

domain-specific solvers = high-level theory solvers

one domain of interest: consistency-aware solvers

existing contract languages a possible starting point

for now, PL folk can bravely dig into solver internals

in the long run: democratize solver development!
“Delite for domain-specific solvers”