Workshop on Software Model Checking

Trail-Directed Model Checking

Stefan Edelkamp, Alberto Lluch-Lafuente and Stefan Leue

Institut für Informatik,
Albert-Ludwigs-Universität,
Georges-Köhler-Allee, D-79110 Freiburg
eMail: {edelkamp, lafuente, leue}@informatik.uni-freiburg.de
1 Introduction

HSF-SPIN is a Promela model checker based on heuristic search strategies.

It utilizes heuristic estimates in order to direct the search for finding software bugs in concurrent systems.

As a consequence, HSF-SPIN is able to find shorter trails than blind depth-first search.

We propose an extension to the paradigm of directed model checking to shorten already established unacceptable long error trails.

For selected benchmark and industrial communication protocols we give experimental evidence that this approach of trail-directed model-checking effectively shortcuts existing witness paths.
2 Algorithm A*

Optimal heuristic graph search algorithm if heuristic $V \rightarrow R$ is \textit{admissible}, i.e. a lower bound.

\begin{verbatim}
Open ← \{(s, h(s))\}; Closed← {};
while (Open ≠ ∅)
    \text{u} ← \text{Deletemin}(Open); Insert(Closed,u)
    if (failure(\text{u})) exit \textbf{Goal Found}
    for all \text{v} in Γ(\text{u})
        \text{f}'(\text{v}) ← \text{f}(\text{u}) + 1 + h(\text{v}) - h(\text{u})
        if (Search(Open, \text{v}))
            if (\text{f}'(\text{v}) < \text{f}(\text{v}))
                \text{DecreaseKey}(Open, (\text{v}, \text{f}'(\text{v})))
        else if (Search(Closed, \text{v}))
            if (\text{f}'(\text{v}) < \text{f}(\text{v}))
                \text{Delete}(Closed, \text{v}); Insert(Open, (\text{v}, \text{f}'(\text{v})))
        else Insert(Open, (\text{v}, \text{f}'(\text{v})))
\end{verbatim}

Difference to Single-Source Shortest-Path Algorithm of Dijkstra highlighted.

Trail-Directed Model Checking
Algorithm A*
3 A First Example

Dining Philosophers checked for Deadlock with HSF-Spin and DFS:

HSF-SPIN 1.0
A Simple Promela Verifier based on Heuristic Search
This tool is based on Spin 3.4.5 (by G.J. Holzmann)
and on HsfLight 2.0 (by S. Edelkamp)
Verifying models/deadlock.philosophers.prm...
Checking for deadlocks with Depth-First Search...
invalid endstate (at depth 1362)
Printing Statistics...
State-vector 120 bytes, depth reached 1362, errors: 1
1341 states, stored
431 states, matched
1772 transitions (transitions performed)
25 atomic steps
1341 states, expanded
Range of heuristic was: [0..0]
Writing Trail
Wrote models/deadlock.philosophers.prm.trail
Length of trail is 1362

The A* algorithm and a simple heuristic estimate for
deadlock detection finds a deadlock at optimal depth
34, expanding and storing less states (17 and 67, respectively), and performing less transitions (73).
One natural option to improve the trail is to impose a shallower depth on the depth-first search engine.

Bounds might increase the search efforts by magnitudes, since a fixed traversal ordering in bounded depth-first exploration in large search depths might miss the lasting error states for a fairly long time.

Therefore, even if the first error is found fast, improvements are possibly difficult to obtain.

Moreover, to find shorter trails by manual adjusting bounds is time consuming, e.g., trying to improve an optimal witness will fail and result in a full state exploration.
5 Anomaly in Depth-Bounded Search

It can be observed when experimenting with explicit state model checkers that allow the search depth to be limited to a maximum, such as it can be done in SPIN.

Visited states are kept in a hash-table to avoid an exponential increase in the number of expanded nodes due to the tree expansion of the underlying graph.

Implicit pruning result in the fact, that duplicate errors in smaller depths will not necessarily be detected anymore, since they might be blocked by nodes that are already stored.
Let $S$ be the current state of the search space and $S'$ be the error state.

Heuristic is estimate for the number of transitions necessary to get from $S$ to $S'$

**Hamming Distance Heuristic**

$$H_{HD}(S, S') = \sum_{i=1}^{k} |s_i - s'_i|$$

**FSM Distance Heuristic**

$$H_{FSM}(S, S') = \sum_{i=1}^{k} D_i(pc_i, pc'_i)$$
7 Errors

Search trees of ordinary search, full-state trail-directed search, and partial-state trail-directed search.

Safety error trail is shortened by trail-directed search.

Liveness error is trail shortened in two phases.
8 Experiments

As the first examples we choose the industrial GIOP protocol with a seeded bug and a model of a concurrent program that solves the stable marriage problem.

GIOP(N=2,M=1)

<table>
<thead>
<tr>
<th></th>
<th>DFS</th>
<th>TDA*, $H_{HD}$</th>
<th>TDA*, $H_{FSM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored States</td>
<td>326</td>
<td>988</td>
<td>30,629</td>
</tr>
<tr>
<td>Transitions</td>
<td>364</td>
<td>1,535</td>
<td>98,884</td>
</tr>
<tr>
<td>Expanded States</td>
<td>326</td>
<td>432</td>
<td>24,485</td>
</tr>
<tr>
<td>Witness Trail</td>
<td>134</td>
<td>67</td>
<td>65</td>
</tr>
</tbody>
</table>

Marriers($n = 4$)

<table>
<thead>
<tr>
<th></th>
<th>DFS</th>
<th>TDA*, $H_{HD}$</th>
<th>TDA*, $H_{FSM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored States</td>
<td>407,009</td>
<td>26,545</td>
<td>225,404</td>
</tr>
<tr>
<td>Transitions</td>
<td>1,513,651</td>
<td>56,977</td>
<td>467,704</td>
</tr>
<tr>
<td>Exp. States</td>
<td>407,009</td>
<td>16,639</td>
<td>192,902</td>
</tr>
<tr>
<td>Witness Trail</td>
<td>121</td>
<td>99</td>
<td>66</td>
</tr>
</tbody>
</table>
9 Experiments (cont.)

In the second set of examples we examine another safety property class, namely state invariants.

The two protocols we consider are a Promela model of an Elevator system and the POTS telephony protocol model.

Elevator \((N = 3)\)

<table>
<thead>
<tr>
<th></th>
<th>DFS</th>
<th>$TDA^*,H_{HD}$</th>
<th>$TDA^*,H_{FSM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored States</td>
<td>292</td>
<td>38,363</td>
<td>38,538</td>
</tr>
<tr>
<td>Transitions</td>
<td>348</td>
<td>146,827</td>
<td>147,277</td>
</tr>
<tr>
<td>Expanded States</td>
<td>292</td>
<td>38,423</td>
<td>38,259</td>
</tr>
<tr>
<td>Witness Trail</td>
<td>510</td>
<td>203</td>
<td>203</td>
</tr>
</tbody>
</table>

POTS

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored States</td>
<td>506,751</td>
<td>2,668</td>
<td>2,019</td>
</tr>
<tr>
<td>Transitions</td>
<td>$1,468 \times 10^6$</td>
<td>6,519</td>
<td>4,889</td>
</tr>
<tr>
<td>Expanded States</td>
<td>506,751</td>
<td>2,326</td>
<td>997</td>
</tr>
<tr>
<td>Witness Trail</td>
<td>756</td>
<td>67</td>
<td>67</td>
</tr>
</tbody>
</table>
A bad sequence corresponds to a violation of a liveness property. In this scenario we experiment with a model of a Fundamental-Mode Circuit.

FMC ($N = 3$)

<table>
<thead>
<tr>
<th></th>
<th>DFS</th>
<th>TDA*,$H_{HD}$</th>
<th>TDA*,$H_{FSM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored States</td>
<td>270</td>
<td>438</td>
<td>419</td>
</tr>
<tr>
<td>Transitions</td>
<td>364</td>
<td>664</td>
<td>624</td>
</tr>
<tr>
<td>Expanded States</td>
<td>279</td>
<td>437</td>
<td>412</td>
</tr>
<tr>
<td>Witness Trail</td>
<td>259</td>
<td>73</td>
<td>73</td>
</tr>
</tbody>
</table>
11 Conclusions

HSF-SPIN tries to find errors faster than traditional tools by employing heuristic search strategies for non-exhaustive, guided state space exploration.

While HSF-SPIN can be used for full verification through exhaustive state space search, this is not its primary objective and we note that other model checkers, like Spin or SMV, are more time and space efficient for this purpose.

We propose error trail improvement, an apparent need in practical software development. We expect that a possibly long witness for an error is already given.

We expect that a possibly long witness for an error is already given.

This trail might be found by simulation, test, random walk, or depth-first model checking. This path is read as an additional input, reproduced in the model and then significantly improved by directed search.