Completion of Partial Automata

Abhishek Udupa

With Rajeev Alur, Milo Martin, Mukund Raghothaman, Christos Stergiou, Stavros Tripakis
(Penn, Berkeley, Aalto University)
The Model Checking Problem

- Given a model of a system and a specification
  - Exists an execution in which error state reachable?
- Two approaches:
  - Explicit State Search
    - Explore the reachable state space by an explicit construction
  - Symbolic Search
    - Symbolic representation of state space reachable by all executions
    - Often implemented using BDDs
The Automata Completion Problem

• Given a spec and set of incomplete automata
  – Constructed from Scenarios

• What happens on a message B or C?

• Complete the automata so that the model satisfies the specification
The Automata Completion Problem

• Completion
  – Only involves *adding* transitions
  – *Fixed* transitions correspond to behaviors in scenarios
    • Deletion of edges not permitted

• Other constraints apart from specification
  – Determinism
  – Deadlock Freedom

• Search through *all* possible completions
Two Solution Techniques

• Explicit Search
  – Construct automaton for each possible completion
  – Model check resulting automaton
  – Heuristic for order in which completions are explored

• Symbolic Search
  – Encode all possible completions as (parameterized) automata
  – BDD based model checking to find a correct completion

• The approaches seem to complement each other
Explicit Search

$q_0$ \rightarrow $q_1$ \rightarrow $q_2$

O! \ A? \ B? \ C?
Explicit Search

\[ q_0 \xrightarrow{O!} q_1 \xrightarrow{A?} q_2 \]

\[ \{(q_1, B?, q_0)\} \xrightarrow{} \{(q_1, B?, q_2)\} \]
Explicit Search

$q_0 \rightarrow q_1 \rightarrow q_2$

O! → B? → C?

$\{(q_1, B?, q_0)\}$

{(q₁, B?, q₀), (q₁, C?, q₀)}

{(q₁, B?, q₀), (q₁, C?, q₂)}

{(q₁, B?, q₂)}
Explicit Search

- Maintain a tree of possible completions
  - Each node labelled with set of additional edges
- Explore the tree
  - Multiple paths can lead to same labels
  - Nodes with same label not re-explored
The Symbolic Approach

• Introduce variables representing completions

\[ t_{\{q_1,B\}} = q_0 \]
\[ t_{\{q_1,B\}} = q_2 \]

• Transition relation parameterized by these variables
  – Parameters can take any value in the initial state
  – Retain the same values at all subsequent states
The Symbolic Approach

• Given any specification \( \varphi \)
• Model check the parameterized system with the CTL property \( \text{EF} \neg \varphi \)
  – If property is true
    • All valuations of parameters lead to erroneous state. No completion possible
  – If property can be falsified
    • Obtain a witness for the falsification
    • Valuations of the parameters in the initial state of witness gives a correct completion
    • Not possible to reach a \( \neg \varphi \) state with this completion
## Experimental Results: ABP

<table>
<thead>
<tr>
<th>Scenario</th>
<th># of states in incomplete automaton</th>
<th># of transitions to be completed</th>
<th>Computational Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sender</td>
<td>Receiver</td>
<td></td>
</tr>
<tr>
<td>No Scenarios</td>
<td>6</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>Scenario 1</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>10</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>All Scenarios</td>
<td>12</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- More Scenarios ➔ More States
- Heuristic guided Explicit Search better with more scenarios
  - Larger number of “similar” states to guide heuristic
- Symbolic Algorithm performs better with fewer scenarios
  - Larger number of states ➔ More choices
  - BDDs begin to blow up
Future and Ongoing Work

• Scalability
  – CEGIS-like approach for the BDD-based symbolic approach
    • Guess a completion
    • If unsuccessful, generalize and rule out a class of completions
    • Use generalization to constrain future guesses
  – Synthesize for simpler environments, verify on full env.
    • Environment makes bounded number of requests
    • State space is acyclic → Constant space model checking
    • Unroll until completion found

• Extension to EFSMs
  – Combining with earlier work (TRANSIT)
  – Use TRANSIT to synthesize EFSM state updates
Summary

• Scenarios seem to reduce the complexity of distributed protocol synthesis
• Successfully synthesized ABP with textbook scenarios
• Two approaches to solving the completion problem
• Few scenarios ⇒ Fewer states ⇒ Symbolic approach performs well
• More scenarios ⇒ Larger number of “similar” states ⇒ Heuristic-guided Explicit search performs well
• Neither approach worked without scenarios
Thank you.
Questions?
Explicit Search

• When model checking the completion fails
• Safety/Liveness violation?
  – All completions in the subtree are violating as well!
  – The entire subtree can be pruned
• Deadlock?
  – Might be fixed by adding additional edges
  – Continue search normally
• Terminate when either
  – Good completion found
  – All completions explored (no completion exists)
The Symbolic Approach
Deadlocks, Determinism, Non-blockingness

• Deadlock is a safety property
  – Can be handled in a similar manner

• Determinism constraints
  – Handled by constraining the initial values of parameters

• Non-blockingness
  – Strong non-blockingness is a safety property
  – Weak non-blockingness can be expressed as a CTL property
The Symbolic Approach
Liveness

• We consider CTL or LTL properties in the symbolic approach for liveness, rather than monitors

• Given a CTL property $\psi$, we model check for $\text{EF } \neg \psi$

• If $\text{EF } \neg \psi$ holds
  – No completion exists

• If not
  – Witness for falsification gives a valuation for parameters which represents a correct completion

• LTL model checking can be reduced to CTL model checking under fairness constraints