Problem

- Missing or inaccurate system documentation makes it challenging to understand the intended system behaviors
- Complex logs of unintended system behaviors makes it difficult to debug them

Goal

- Take as input a set of observed traces (usually represented as sequences of events) of a given system
- Automatically produce an inferred model (often represented as an FSA) that must accept all of the observed traces
  - May also accept some unobserved traces

Model inference tool: Architecture

[Bisim algorithm]

https://people.cs.umass.edu/~brun/pubs/Beschastnikh11fse.pdf
Shopping cart: API (written in PHP)

- invalid-coupon
- valid-coupon
- reduce-price
- check-out
- get-credit-card

BisimH algorithm

1. Extract a trace graph from a given log using the regular expressions
2. Mine invariants from the trace graph
3. Create initial inferred model by partitioning the trace graph
4. While (current inferred model violates invariants)
   1. Generate counterexample path through the current inferred model illustrating a violation of a given invariant
   2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

BisimH algorithm

1. Extract: Log and regular expressions

   • Log, e.g.,

   • Regular expression(s), e.g.,
     Line parsing: \(<ip>.*/\(<TYPE>.+\).php\)

   • Special events for INITIAL and TERMINAL. Each event specified as a triple <trace ID, timestamp, event type>, e.g., <74.15.155.103, 06/Jan/2011:07:24:13, check-out>
1. Extract: Trace graph

- Each event specified as a triple <trace ID, timestamp, event type>
  - Also special events for INITIAL and TERMINAL

- Each trace is a linear graph where:
  - Each event corresponds to a vertex
  - The total ordering among the events (specified by their timestamps) corresponds to the edges among the nodes

- A trace graph is the union of the set of traces

---

Shopping cart: Inputs

<table>
<thead>
<tr>
<th>Log:</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.15.155.103 [06/Jan/2011:07:25:33] &quot;GET HTTP/1.1 invalid-coupon.php&quot;</td>
</tr>
<tr>
<td>33.15.155.103 [06/Jan/2011:07:27:22] &quot;GET HTTP/1.1 invalid-coupon.php&quot;</td>
</tr>
<tr>
<td>37.15.155.103 [06/Jan/2011:07:28:43] &quot;GET HTTP/1.1 invalid-coupon.php&quot;</td>
</tr>
<tr>
<td>41.15.155.103 [06/Jan/2011:07:28:14] &quot;GET HTTP/1.1 reduce-price.php&quot;</td>
</tr>
<tr>
<td>45.15.155.103 [06/Jan/2011:07:29:02] &quot;GET HTTP/1.1 get-credit-card.php&quot;</td>
</tr>
<tr>
<td>49.15.155.103 [06/Jan/2011:07:30:30] &quot;GET HTTP/1.1 reduce-price.php&quot;</td>
</tr>
<tr>
<td>53.15.155.103 [06/Jan/2011:07:30:50] &quot;GET HTTP/1.1 check-out.php&quot;</td>
</tr>
<tr>
<td>57.15.155.103 [06/Jan/2011:07:31:17] &quot;GET HTTP/1.1 check-out.php&quot;</td>
</tr>
<tr>
<td>61.15.155.103 [06/Jan/2011:07:31:20] &quot;GET HTTP/1.1 get-credit-card.php&quot;</td>
</tr>
<tr>
<td>65.15.155.103 [06/Jan/2011:07:31:44] &quot;GET HTTP/1.1 check-out.php&quot;</td>
</tr>
</tbody>
</table>

| Regular | Line parsing: "(?<ip>\./\.[?\<TYPE\>\.+].php"
| Expressions | Trace mapping: "\k<ip>" |

---

Shopping cart: Trace graph

BisimH algorithm

1. Extract a trace graph from a given log using the regular expressions
2. Mine invariants from the trace graph
3. Create initial inferred model by partitioning trace graph
4. While (current inferred model violates invariants)
   1. Generate counterexample path through the current inferred model illustrating a violation of a given invariant
   2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model
2. Mine invariants

- Capture temporal relationships between **event types** in a given trace graph
  - Always Followed by \( a \rightarrow b \)
    e.g., INITIAL \( \rightarrow \) check-out
  - Never Followed by \( a \nrightarrow b \)
    e.g., valid-coupon \( \nrightarrow \) invalid-coupon
  - Always Precedes \( a \rightarrow b \)
    e.g., check-out \( \rightarrow \) get-credit-card

- Must be satisfied by all of the potential traces through that graph

---

### Shopping cart: Invariants

Use the Dwyer et al. property patterns

---

### BisimH algorithm

1. Extract a **trace graph** from a given log using the regular expressions
2. **Mine invariants** from the **trace graph**
3. Create initial inferred model by partitioning trace graph
4. **While (current inferred model violates invariants)**
   1. Generate **counterexample path** through the current inferred model illustrating a violation of a given invariant
   2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model
BisimH algorithm
1. Extract a trace graph from a given log using the regular expressions
2. Mine invariants from the trace graph
3. Create initial inferred model by partitioning trace graph
4. While (current inferred model violates invariants)
   1. Generate counterexample path through the current inferred model illustrating a violation of a given invariant
   2. Refine current inferred model based on that counterexample path to produce next inferred model satisfying that invariant
5. Coarsen current inferred model to produce final inferred model

Use counterexample guided abstraction refinement (CEGAR)
4. CEGAR: Generate counterexample path

- Employ a model checker to determine if all potential paths through the current inferred model satisfy the invariants
  - If so, report satisfied
  - If not, report violated and a counterexample path illustrating a violation of one of the invariants

4. CEGAR: Refine

1. Refine the current inferred model based on the violated invariant
   - Heuristically select a vertex relevant to that invariant

2. Create the refined inferred model that satisfies that invariant
   - Split the selected vertex into multiple vertices
5. Coarsen

- Coarsen the refined inferred model
  - Search for vertices that didn’t need to be split by the refine heuristic and merge them

- The coarsened inferred model satisfies all of the invariants. (It is the final inferred model.)

Does the inferred model illustrate any unintended behavior (i.e. a bug)?
Shopping cart: Final model

Does the inferred model illustrate any unintended behavior (i.e. a bug)? **YES**

Challenges for model inference

- How does the selection of the event sequence set affect the final model? affect performance?
- How does the parameterization of the model inference algorithm affect the final model?
- How do you compare/contrast (or diff) two inferred models?
- How to support other high-level language features such as concurrency, real-time constraints, …?

Learning from system traces

- Usage scenarios, e.g.,  
  https://dl.acm.org/doi/pdf/10.1145/1656250.1656252  
- Logs, e.g.,  
  Synoptic, Invarimint  
- Counterexample paths, e.g.,  
  libalf: The automata learning framework  
  (http://libalf.informatik.rwth-aachen.de)  
- …

MVC architecture pattern
MVC architecture pattern

- **Client** uses **View**
- **View** updates **Model**
- **Model** manipulates **Controller**
- **Controller** updates **Client**

Observer design pattern

The **Observable** maintains its state and a list of **Observers**. If the Observable undergoes a state change, it must update each Observer about that state change.

Examples:
- Model is Observable and View is Observer
- View is Observable and Controller is Observer

Presentation Abstraction Control (PAC) architecture pattern


Presentation Abstraction Control (PAC) architecture pattern

Mediator design pattern

The **Mediator** is responsible for all of the communications among a set of **Colleagues**. (The Colleagues never communicate directly with each other.)

Example:
- Control is the Mediator and Colleagues are Presentation and Abstraction.

https://en.wikipedia.org/wiki/Mediator_pattern

Current assignments

- Guest lecture forum questions by each final project group
- Final project mid-point presentations will be held this Thursday 4/2 and next Tuesday 4/7
- Homework 2 due next Thursday 4/9