Leveraging Existing Instrumentation to Automatically Infer Invariant-Constrained Models
Synoptic:
Mining from

Ivan Beschastnikh
Yuriy Brun
Sigurd Schneider
Michael Sloan
Michael D. Ernst

Saarland University
University of Washington

* http://en.wikipedia.org/wiki/File:DFAexample.svg
Motivating question

I am a developer.

Why does my system behave in a certain manner?
One answer is to...

Study the relevant artifacts

Specification

Code

Log
This Client State Diagram illustrates all the legal state changes that can occur in the client.

- **ReadOnly (RW)**: The client is in a read-only state.
- **Invalid**: The client is in an invalid state.
- **Write (W)**: The client is in a write state.
- **Get**: The client is in a get state.
- **Write Append**: The client is in a write append state.

**Logs**

- **Trivial instrumentation**
- **Easy to Create**
- **Concise and exact**
- **A low-level view**

**Models**

- **Manually defined**
- **Hard to Create**
- **Easy to Use**

The diagram shows the actions of the client upon receiving a packet and the corresponding state changes.

- Send RDC back: Set state to RO
- Send IC: Set state to IN
- Send WC back: Set state to RW
- Send RC back: Set state to RO

The logs show a sequence of events involving different clients and servers, indicating actions such as committing, acknowledging, preparing, and committing transactions.
Tool
- Specification mining
- Process discovery
- Model inference

Prior work:
- Cook et al. TSE 1998
- Dallmeier et al. WODA 2006
- Lo et al. ASE 2010
How Synoptic works

Input

Does not require access to code

Does not require access to code

Mines and preserves temporal log invariants

Invariants

Uses refinement instead of coarsening

Output

...
Use cases for Synoptic models

- Log summarization
- Mental model validation
- Test case generation
- Comparison to formal specifications
- Evaluate test suite against actual usage
- Verifying a code fix
Motivation

Synoptic’s design

Evaluation results

Future work

Conclusion
Synoptic inputs

1. Log file
2. Regular expressions to parse the log file
Two phase commit protocol log

1. Log file

Two phase commit protocol log

I. Manager proposes a transaction (TX), each replica replies with an abort or a commit.

II. Manager sends

i) TX commit if all replicas commit

ii) TX abort otherwise

Manager maintains a totally ordered log of events for all transactions in the system
Regular expressions

1. Log file (e.g., two phase commit log)

2. Regular expressions to parse the log file
   - Split log into executions
   - From each log line extract:
     - Total event relation (e.g., time)
     - Event type
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

- X: initial event
- Y: intermediate event
- Z: terminal event
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type
1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type
Synoptic overview

1/5. Parse log into a trace graph

2/5. Construct the initial model

3/5. Mine temporal invariants

A compact model with one node per event type

- abort → tx-abort
- abort ↔ tx-commit
- abort ← tx-abort
- ...
4/5. **Refine** the initial model until all invariants satisfied.

Choose an invariant invalid in the model

- abort → tx-abort

5/5. **Coarsen** model without unsatisfying any invariants.

True for log, false for initial model.
4/5. **Refine** the initial model until all invariants satisfied

5/5. **Coarsen** model without unsatisfying any invariants

Choose an invariant invalid in the model

- abort \(\rightarrow\) tx-abort

Unsatisfied invariants exist

Refine a model node

- propose

- abort

- commit

- tx-abort

- bx-commit

- propose

- abort

- commit

- tx-abort

- bx-commit
4/5. Refine the initial model until all invariants satisfied

5/5. Coarsen model without unsatisfying any invariants

Choose an invariant invalid in the model
abort → tx-abort

Unsatisfied invariants exist

All invariants satisfied

Refine a model node

propose

abort

commit

propose

abort

commit

tx-abort

tx-commit

bx-commit
4/5. **Refine** the initial model until all invariants satisfied

Choose an invariant invalid in the model
- abort $\rightarrow$ tx-abort

Unsatisfied invariants exist

Refine a model node

All invariants satisfied

Merge nodes that exhibit the same behaviors

5/5. **Coarsen** model without unsatisfying any invariants
Assumptions

Logs
- Multiple executions
- Events in an execution are totally ordered

System
- Process generating the log can be modeled as an FSM
- Temporal system properties are evident in the orderings among events
Talk outline

• Motivation
• Synoptic’s design
  • Overview
  • Assumptions
  • Temporal invariants
  • Model space exploration
• Evaluation results
• Future work
• Conclusion
## Temporal log invariants

- Synoptic mines three kinds of temporal log invariants that are true for all the logged executions.

<table>
<thead>
<tr>
<th>Invariant</th>
<th>Example</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \rightarrow y$</td>
<td>$x \ Y \ x \ Y \ y \ y$</td>
<td>liveness</td>
</tr>
<tr>
<td>always followed by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \leftarrow y$</td>
<td>$x \ y \ x \ y \ y \ z$</td>
<td>safety</td>
</tr>
<tr>
<td>always precedes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \leftrightarrow y$</td>
<td>$x \ x \ y \ y \ y \ z$</td>
<td>safety</td>
</tr>
<tr>
<td>never followed by</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dwyer et al. ICSE 1999
Yang et al. PASTE 2004
# Temporal log invariants

- Synoptic mines three kinds of **temporal log invariants** that are true for all the logged executions.

<table>
<thead>
<tr>
<th>Invariant</th>
<th>Example</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x → y</td>
<td>X _ Y</td>
<td>_ X _ Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x ← y</td>
<td>X _ Y</td>
<td>_ X _ Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x → y</td>
<td>X _ x</td>
<td>_ y _ y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dwyer et al. ICSE 1999  
Yang et al. PASTE 2004
Temporal log invariants

- Synoptic mines three kinds of temporal log invariants that are true for all the logged executions

<table>
<thead>
<tr>
<th>Invariant</th>
<th>Example</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \rightarrow y )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>always followed by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x \leftarrow y )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>always precedes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x \leftrightarrow y )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>never followed by</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dwyer et al. ICSE 1999  
Yang et al. PASTE 2004
Two phase commit invariants

- **Mined invariants:**
  - abort → tx-abort
  - abort → tx-commit
  - abort ← tx-abort
  - commit ← tx-commit
  - \{ abort, commit, tx-abort, tx-commit \} → propose
  - propose ← \{ abort, commit, tx-abort, tx-commit \}
Satisfying invariants with refinement

• The initial graph violates 3 mined invariants:
  - abort $\xrightarrow{\text{tx-abort}}$
  - abort $\xrightarrow{\text{tx-commit}}$
  - abort $\xleftarrow{\text{tx-abort}}$
  - commit $\xleftarrow{\text{tx-commit}}$
  - $\{\text{abort, commit, tx-abort, tx-commit}\} \xrightarrow{\text{propose}}$
  - propose $\xleftarrow{\{\text{abort, commit, tx-abort, tx-commit}\}}$

• For each violation find a corresponding path
• Use refinement to eliminate such path counter-examples

One node per event type

Diagram:

- **propose**
- **abort** $\xrightarrow{\text{commit}}$
- **commit** $\xrightarrow{\text{tx-commit}}$
- **tx-abort** $\xrightarrow{\text{abort}}$
- **tx-commit** $\xrightarrow{\text{propose}}$

• The initial graph violates 3 mined invariants:
Counter-example based refinement

Example invariant: abort → tx-abort

1. Initial model

2. Invariant counter-example

3. Identify partition to refine

4. Eliminate counter-example

5. Refined model
Model space

Larger models: fewer behaviors

Smaller models: more behaviors

Trace graph

Initial model

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph

Larger models:
Fewer behaviors

Smaller models:
More behaviors

Initial model

Trace graph
Larger models: fewer behaviors

Smaller models: more behaviors

Initial model

Trace graph
Model space

Larger models: fewer behaviors

Smaller models: more behaviors

Some invariants falsified

Trace graph

Initial model

Some invariants falsified

Larger models: fewer behaviors

Smaller models: more behaviors
Model space

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Initial model

Trace graph

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Initial model

Trace graph
Model space

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Initial model

Synoptic’s goal: Find the smallest model satisfying all invariants

Trace graph
Model space

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Coarsening

Refinement

Trace graph

Initial model

Synoptic’s goal:
Find the smallest model satisfying all invariants

Larger models: fewer behaviors

All invariants satisfied

Smaller models: more behaviors

Some invariants falsified

Coarsening

Refinement

Trace graph

Initial model

Synoptic’s goal:
Find the smallest model satisfying all invariants
All invariants satisfied

Some invariants satisfied

Synoptic finds a local optimum, not a global one

Select a coarsening that maintains all the invariants

Select a refinement that satisfies an unsatisfied invariant

Initial model

Refine

Coarsen

Refine

Refine

Refine

Refine

Synoptic finds a local optimum, not a global one

Select a coarsening that maintains all the invariants

Select a refinement that satisfies an unsatisfied invariant

Initial model
All invariants satisfied

Some invariants satisfied

Synoptic finds a local optimum, not a global one

Select a coarsening that maintains all the invariants

Select a refinement that satisfies an unsatisfied invariant

Coarsening from initial traces is inefficient

- k-Tail \textit{Biermann et al. 1972} and GK-Tail \textit{Lorenzoli et al. ICSE 2008}

- Refinement is well known \textit{Paige et al. 1987}
  - Using mined invariant constraints is a new idea

Initial model

• Coarsening from initial traces is inefficient
  - k-Tail \textit{Biermann et al. 1972} and GK-Tail \textit{Lorenzoli et al. ICSE 2008}
  
• Refinement is well known \textit{Paige et al. 1987}

- Using mined invariant constraints is a new idea

Initial model

Select a coarsening that maintains all the invariants

Select a refinement that satisfies an unsatisfied invariant
Talk outline

- Motivation
- Synoptic’s design
- Evaluation results
  - Formal evaluation
  - Reverse traceroute
  - Distributed systems course
- Future work
- Conclusion
Formal evaluation

• Termination
  • Always finds a model

• Correctness
  • Satisfies all the mined invariants, and no others

• Conciseness
  • Local minimum

• Efficiency
  • Never violate a mined invariant
  • Make progress towards satisfying some mined invariant

See the paper for more
Reverse traceroute case-study

• Reverse traceroute  
  • Finds the likely reverse traceroute from an arbitrary destination to a source host  
  • In live deployment for over a year  
  • Deployed internally by a large Internet company

• Ran Synoptic on reverse traceroute server logs  
  • Developer added 16 lines of logging code  
  • Synoptic processed the 900,000 events in 12 minutes
Reverse traceroute case-study

2 bugs found:
- **Shaded** nodes should not be terminal
- **Red** edges should not exist

Synoptic was useful for:
- Finding bugs in code
- Increasing developer confidence
- Building system understanding
Synoptic limitations

- Quality and usefulness of the model depends on granularity of logging statements and user input
- Only considers three kinds of temporal invariants
- For some applications, the small model size may be irrelevant
- Online log analysis is not supported
- The input log must be totally ordered
Future work

• Handle logs generated by distributed systems
• Generalize invariants
  • Relax *always* and *never* constraints \cite{Lou et al. KDD 2010}
  • Explore composition of temporal invariants \cite{Gabel et al. FSE 2008}
  • Incorporating structural/data invariants \cite{Ernst et al. TSE 2001}
• Tool improvements
  • Explicating model paths in the GUI
  • Eclipse plug-in for ease of use
• Automatically infer log format \cite{Vaarandi et al. IFIP ICICS 2004} \cite{Zhu et al. OSR 2010}
Contributions

Synoptic: a tool that extracts models from logs

• Specification mining algorithm
  • Uses refinement for efficiency
  • Mines and satisfies temporal log invariants for accuracy
• Proved important properties of refinement
• Showed that Synoptic helps developers to study large and complex logs, and can help find bugs

Try it!

http://synoptic.googlecode.com