Pictures at QRS 2015

Day 2 (August 4)
HRML: a hybrid relational modelling language

He Jifeng
Hybrid Systems

- Systems are composed by continuous physical components and discrete control components.
- The system state evolves over time according to interactions between discrete and continuous dynamics.
  - For discrete dynamics, it changes state instantaneously.
  - During continuous transitions, its state is a continuous function of continuous time and varies according to a differential equation.
- Modelers mix discrete-time reactive systems with continuous-time ones.
Parallel by merge

Let $P$ and $Q$ be hybrid relations with $x' \in \text{out}aP \cap \text{out}aQ$. We define their parallel composition equipped with the merge mechanism $M$, denoted by $P \parallel_M Q$, as follows:

$$P \parallel_M Q = \exists m, n : \text{Val} \cdot (P[m/x'] \land Q[n/x'] \land (x' = (m \circ n)))$$

$$\text{in'a} = \text{in'a}P \cup \text{in'a}Q$$

$$\text{out'a} = \text{out'a}P \cup \text{out'a}Q$$

$$\text{con'a} = \text{con'a}P \cup \text{con'a}Q$$
Closure of healthy hybrid relations

Theorem

1. $H(P) \cap H(Q) = H(P \cap Q)$
2. $H(P) \triangleleft_{b} H(Q) = H(P \triangleleft_{b} Q)$
3. $H(P); H(Q) = H(P; H(Q))$
4. If $P$ and $Q$ lie in the complete lattice $L$, then $(P \parallel_{MQ})$

where the merge mechanism
HRML: a hybrid relational modelling language

\[ AP ::= \text{skip} \mid \text{chaos} \mid \text{stop} \mid x := e \mid s \mid \text{delay}(\delta) \]

\[ EQ ::= R(v, \delta) \mid EQ \text{init} v_0 \mid EQ \parallel EQ \]

\[ P ::= AP \mid P \cap P \mid P 
\triangleleft \triangle < h(x) \triangleright P \parallel P \parallel P \parallel P \parallel P \mid
\]

\[ EQ \text{ until} g \mid \text{when}(G) \mid \mu X 
\bullet P(X) \]

\[ \text{timer} c \bullet P \mid \text{signal} s \bullet P \]

\[ g ::= \text{skip} \mid s \mid \text{test} \mid g \cdot g \mid g + g \]

\[ \text{test} ::= \text{true} \mid v \geq c \mid v \leq c \mid \text{test} \land \text{test} \mid \text{test} \lor \text{test} \]

\[ G ::= g \& P \mid G \mid G \]

\[ G ::= g \& P \mid G \mid G \]

\[ G ::= g \& P \mid G \mid G \]
Adjust the sampling rate

The following theorem is used to reduce the sampling rate of the controller by estimating the change speed of physical state.

Theorem Let $I < m < n$.

If $R \supseteq \{0 < i \leq r\}$ and $\delta < (n - m)/r$, then

$$(R \text{ init } \text{ until } (c \geq \delta))$$

$$= \text{ signal } s, u \left( \begin{array}{c}
\text{ delay}(d); \\
\text{ when } ((c \geq r) \lor (c \leq (d + r)))
\end{array} \right)$$
Overview

1. Challenges to Software Quality Assurance
2. Our Solution
3. Specification Animation
4. Specification-Based Program Testing and Inspection
5. Open Problems
6. Conclusions
7. Future Work
1. Challenges to Software Quality Assurance

- The scale and complexity of software development projects
- The scale of documentation
- The complexity of documentation
- The complexity of situations (e.g., requirements changing, people moving, client complaining, manager worrying, and developer fighting)
1. Challenges to Software Quality Assurance

- The scale and complexity of software development projects.
- The scale of documentation.
- The accuracy of documentation.
- The velocity of situations, e.g., requirements changing, people moving, client complaining, manager worrying, and developer fighting.
Deficiencies of techniques available for use

- Formal proof of correctness: ideal but tedious, ineffective (for faulty programs), requiring skills (loop invariants), error-prone, and time consuming.

- Model checking: needs appropriate abstraction of a real system to a FSM model and faces the state explosion problem (two state space explosions for software: initial state space and program state space).

- Testing: can tell the existence of bugs, but cannot tell their absence in general. Nevertheless, it is a common practice in industry.

- Review and inspection: easy to carry out, but heavily depend on human judgment, ability, and experience.
2. Our Solution

- Specification
- Animation
- Specification-Based Testing
- Software defects
3. Specification Animation

Specification animation is a technique for dynamic and visualized demonstration of the system behaviors defined in the specification.

Three expected effects: improving understanding of requirements or designs, strengthening communication, and verifying/validating specifications.
I. Specification Animation

Specification animation is a technique for dynamic and visualized demonstration of the system behaviors defined in the specification.

Expected effects: improving understanding of requirements or designs, strengthening communication, and verifying/validating specifications.
Potential Component Leaks in Android Apps
An Investigation into a new Feature Set for Malware Detection

Lijing Li
SnT, University of Luxembourg

Joint work with:
- Kevin Allix, Dapuyan Li, Tegwendi Bwamande, Jacques Klein, SnT
- Alexandre Bartel, TU Darmstadt
Android

Android browser flaw found to leak sensitive user data.

Leaks

Dozens of popular Android apps leak sensitive user data.

A group of researchers from the University of Luxembourg has discovered vulnerabilities in several popular Android apps, including Instagram, Vine, OKCupid, and more. The bugs could expose the sensitive information of over 500 million users who have installed the affected applications on their devices.

SnT, University of Luxembourg
An Effective Approach to Continuous User Authentication for Touch Screen Smart Devices

Arun Balaji Buduru and Stephen S. Yau
Information Assurance Center
School of Computing, Informatics, and Decision Systems Eng.
Arizona State University
Current State of the Art

- Multi-modal continuous authentication techniques use factors, such as fingerprints, iris and face recognition to continuously authenticate the legitimate user.
  - Muncaster and Turk presented an approach to performing continuous, score-level multi-modal authentication based on a weighted sum of scores from each modality.
  - A continuous multi-modal biometrics system using a hidden Markov model (HMM) was developed by Sim, et al.
  - Shi, et al, used multimodal inputs, such as voice, location, multi-touch and motion, to perform continuous user authentication.

- These techniques are inherently infeasible due to the requirements of additional hardware in low-cost touch-screen smart devices, and frequent conscious user interactions.
Naïve Bayes is an accurate algorithm, and not user
identified as a trusted
A CEC contains Simulink blocks and conditionals, and
A Simulink CEC is a 4-tuple $E = \{D, G, B, V\}$, where:
- $D$ is a finite set of decisions determined by the Predicate Blocks of each conditional subsystem;
- $G$ is a finite set of decisions and conditions formed
- $B$ is the set of decisions and conditions that are formed by functional blocks;
- $V$ is the set of values associated with the conditions and decisions.
SVM Score distribution

Training Set Normal Model - SVM Score

Training Set Malicious Model - SVM Score
Detection Logic

```python
algorithm {
  anomaly = normal_model_score
  malicious = malicious_model_score
  if anomaly < n, && malicious < m,
    predict benign
  else if anomaly < n, && malicious > m,
    predict malicious
  else
    predict ambiguous
}
```
1. Introduction

Reliability of the real time system is depend on the distribution of uncertainty (attribute bias).

Uncertainty analysis is used to investigate the reliability of RTS with uncertainty variables for decision-making program.
XML Bomb/Billion Laughs (BIL)

- DoS attack on XML parser using well-formed XML
- Exponential entity expansion

**Impact**
- Higher memory & CPU Consumption
- DoS on parsing system

```xml
<?xml version="1.0"?>
<!DOCTYPE xmlBomb [ 
<!ENTITY a "Random Text"> 
<!ENTITY b "&a; &a; &a; &a; ">
<!ENTITY c "&b; &b; &b; &b; ">
<!ENTITY d "&c; &c; &c; &c; ">
]>
<xmlBomb>&d;</xmlBomb>
```
Background

Dynamic Atomicity Violation Detectors

- Examples
  - CTrigger [ASPLOS 2009, TSE 2012]
  - AtomFuzzer [FSE 2008]
  - AssetFuzzer [ICSE 2010]
  - Maple [OOPSLA 2012]

- Two-phase strategy is often used
  - Prediction phase
  - Confirmation phase
My Research

- Building reliable and secure software applications

- Compiler & runtime techniques for error resilience
  - Partitioning data for differential resilience [ASPLOS’11]
  - Error detection in different programs [DSN’12][DSN’13]
  - Fault Injection techniques and tools [DSN’14][ISPASS’14]

- This tutorial
  - Reliability of modern web applications (Part 1)
  - Tools for building robust web applications (Part 2)
## Running Example

<table>
<thead>
<tr>
<th>Cell</th>
<th>G4</th>
<th>H4</th>
<th>J2</th>
<th>J3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classification</th>
<th>d</th>
<th>true</th>
<th>false</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
</table>

Blog reference: [Fast classification in the light of faulty user input](http://example.com).
### Running Example

<table>
<thead>
<tr>
<th>Cell</th>
<th>G4</th>
<th>H4</th>
<th>J2</th>
<th>J3</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>E7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000</td>
</tr>
<tr>
<td>E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>G2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.707</td>
</tr>
<tr>
<td>G3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.816</td>
</tr>
<tr>
<td>G4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>H2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.500</td>
</tr>
<tr>
<td>H3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000</td>
</tr>
<tr>
<td>H4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.707</td>
</tr>
<tr>
<td>J2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>J3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.707</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classification d</th>
<th>true</th>
<th>false</th>
<th>false</th>
<th>false</th>
</tr>
</thead>
</table>

*References:*
Highly-impactful vs. Other bugs

- Highly-impactful bugs
- Other bugs

- Reporter experience: Highly-impactful > Other
- Comment length: Highly-impactful > Other
- Closed %: Highly-impactful < Other
Problem

- Detection of information flow bugs during software design and coding time
Boolean Differentiation for Formalizing Myers’ Cause-Effect Graph Testing Technique

Tolga Ayav, Fevzi Belli
Izmir Institute of Technology
Dept. of Computer Engineering

5th IEEE International Workshop on Model-Based Verification & Validation
Vancouver, August 3-5, 2015
Cause-Effect Graphs

- Cause-Effect Graphs (CEG) assists deriving tests from a given specification given in natural language.
- CEG is constructed by an experienced test engineer.
- Test cases are derived from the graph.
- My experience, the generation method from CEG is intuition-based.
In multilevel hybrid cloud, most of the access control policies often deal with user’s single access behavior only:

- Multiple access between malicious users may make data flow violating security policy
- Malicious behavior must be prevented in the high cloud systems
Prof. Dr. Franz Wotawa / TU Graz
General Chair QRS 2016, Vienna, Austria