CS 267: Automated Verification

**Lecture 18, Part 2: Data Model Analysis for Web Applications**

**Instructor:** Tevfik Bultan
Web Application Dependability

Bank of America

We're sorry, our site is running slowly.
As a result, you may experience delays or have difficulty accessing parts of bankofamerica.com, including Online Banking.

We're actively working to resolve these issues. You may wish to defer your transaction until a non-peak time.

If it's convenient, we invite you to conduct your banking business at a banking center or ATM.

Otherwise, you may continue to Online Banking from the site.

Thank you for your patience.

Bank of America

Online Banking is temporarily unavailable for CA customers.
We apologize for the inconvenience.
Please try again later.
Sign in to other services  Sign-in help/options
Web Application Dependability

500 Internal Server Error

Sorry, something went wrong.

A team of highly trained monkeys has been dispatched to deal with this situation. Please report this incident to customer service.

Also, please include the following information in your error report:

318e0cDLx3dDQwvTC92CTQwiwNg09GfaHjqQHUt5rfuAr3EMILAzBlIdhQ22hIQ
Viu2DdbWH5ih3XgcxtaJ3is12amWVSAzU3cSD52LWkTsasuK8-WK8cKcCtbA
hQku-XtfQbo0Z5h56csgMKGb3yBlky7F1whjhhGzUJPuMzJPET_y06egnSc2qpo
o_1Xmohrne-fO0RKSSvFbLd6DbmqQ16E5Oru0P915rX1axbw1YbpvEx3D
DCd2s0vbo2QNP-w06z_Cb9vM711B4sRvVqXUcvz1H7ePQn3SNcElVnK7
N7jeBKQiqeKrm4LzcrRF92p9ZKDPJQARQmUP2G7VpxsrxDgQn_ANOS5R0vlg90
76eiP0WkdJDUU1jLe1tjw6CBjG6t2-nwA-f2XH54WkIA_5NaJh8VC0s3Bb
HscgrJesXwjg2sLXg8szWzAXJ_jQXjKhnHhxR8VEx17uEQWv7rUsH1RIT_Frx5
eq0AlyKOKdJyuplacXTAgR4OYCNSML5dp-icU3kAFaNQJ5raoTpW_HKYXxT
y4kb9H2K6jL5Aou_ouPDOkHjHZV3dtkL0x23f9x1u01e63t3ifM3P0wgsQu
kF4_ANDE9m2UK3z6-koyPuyla9xWNCu3T01JFQ12aJWR_zFCuWH8rINvFaxd
wFkG4uRcsSPmMU75MLfl1Jh-UFyjxc+fjIrSh1ios5cvb3WFPDqOHzL3FXv1
TTXfRae3cCFDm7T726JaX1DENfJgWnisIImtnndniJPByWT_kzkSI0uy65410
E_z71RwWCpBTc0ibnaQOsRt_iW1BS7zjHZ8SO3KpJ-itrzFccqdDLVnWjFnIo
Price Tag for Healthcare.gov Repairs Jumps to $121 Million; 'Back End' Still a Mess

Obama administration soliciting 2015 contractors based on race and gender.

7:28 AM, APR 30, 2014 • BY WHITNEY BLAKE

After shelling out $677 million to build the federal health care website, the government will spend an additional $121 million in 2014 to repair it—$30 million more than previously estimated—the Washington Times reported last night. This comes just as the Obama administration is starting the hunt for next year's diverse group of contractors.

President Obama: “I want to go in and fix myself, but I don't write code"
Web Application Dependability

- **TRACKS**: A todo list application

## Context

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>HOME WORK</th>
</tr>
</thead>
</table>

## Recurring Todo

- Feed the Dog

---

### NoMethodError in Recurring.todos#edit

Showing app/views/recurring_todos/_edit_form.html.erb where line #38 raised:

You have a nil object when you didn’t expect it!
The error occurred while evaluating nil.name

Extracted source (around line #38):

```ruby
35:     </script>
36:     
37:     <label for="edit_recurring_todo_context_name">Context</label>
38:     <input id="edit_recurring_todo_context_name" name="context_name" autocomplete="off" tabindex="4" size="30" type="text">
39:     <div class="page_name_auto_complete" id="edit_context_list" style="display:none"></div>
40:     <script type="text/javascript">
41:         var contextAutoCompleter;
```
Web Application Architecture

- Model View Controller (MVC) pattern:
  Ruby on Rails, Zend for PHP, CakePHP, Struts for Java, Django for Python, ...
- Object Relational Mapping (ORM)
  ActiveRecord, Hibernate, ...
An Example Rails Data Model

Static Data Model

class User < ActiveRecord::Base
  has_many :todos
  has_many :projects
end

class Project < ActiveRecord::Base
  belongs_to :user
  has_many :todos
  has_many :notes
end

class Todo < ActiveRecord::Base
  belongs_to :user
  belongs_to :project
end

class Note < ActiveRecord::Base
  belongs_to :project
end

Data Model Updates: Actions

class ProjectsController < ApplicationController
  def destroy
    @project = Project.find(params[:project_id])
    @project.notes.each do |note|
      note.delete
    end
    @project.delete
    respond_to(...) end
end
Static Data Model

- **ActiveRecord class declarations**
  - sets of objects

- **ActiveRecord association declarations**
  - has_one, has_many, belongs_to, has_and_belongs_to_many

- Association declarations can be used to declare the three basic types of relations between classes
  - one-to-one
  - one-to-many
  - many-to-many
Extensions to Static Data Model

• :through Option
  • To express relations which are composition of other relations

• :conditions Option
  • To relate a subset of objects to another class

• :polymorphic Option
  • To express polymorphic relationships

• :dependent Option
  • On delete, this option expresses whether to delete the associated objects or not
The :through Option

class User < ActiveRecord::Base
  has_one :profile
  has_many :photos, :through => :profile
end

class Profile < ActiveRecord::Base
  belongs_to :user
  has_many :photos
end

class Photo < ActiveRecord::Base
  belongs_to :profile
end
The `:dependent` Option

- **`:delete`**: directly delete the associated objects without looking at its dependencies
- **`:destroy`**: first checks whether the associated objects themselves have associations with the `:dependent` option set

```ruby
class User < ActiveRecord::Base
  has_one :profile, :dependent => :destroy
end

class Profile < ActiveRecord::Base
  belongs_to :user
  has_many :photos, :dependent => :destroy
end
```
Data Model Verification

• Formalize the static data model as
  • A set of classes
  • A set of relations between those classes
  • A set of constraints on the relations that are imposed by the association declarations

• Given a formal data model we can automatically check if a given property holds for the data model
  • Automated verification determines: Do the constraints of the data model imply the property?
Data Model Verification

BOUNDED VERIFICATION

- Alloy Analyzer
  - formula
  - instance or unsat
- Alloy Encoder
  - bound
- Results Interpreter
  - Property Verified

UNBOUNDED VERIFICATION

- SMT-LIB Encoder
  - formula
- SMT Solver
  - instance or unsat or unknown
- Results Interpreter
  - Property Failed + Counterexample
  - Unknown
How Automated is Automated Verification?

• All except one step: Property specification
• Example: It is possible to have a User who does not have any Photos.
  • In Alloy:
    
    ```plaintext
    pred prop { all s: PreState | some u: User | all p: Photo |
    (p not in (s.photo_user).u) }
    ```
  
  • In SMT-LIB:

    ```plaintext
    (assert (exists ((a PolymorphicClass)) (forall ((p Photo))
    (and (isUser a) (not (= p (a user_photo p)))))))
    ```

• Can we make it easier?
Property Templates

• Property templates for property specification
  • Language-neutral
  • Do not require familiarity with SMT-LIB and Alloy

• Example property template:
  • noOrphans[classA, classB]
    • To check that deleting an object from classA does not cause related objects in classB to be orphaned

• Easily rerun tool and switch the verification technique, without having to rewrite the property

• We developed seven property templates for the most common data model properties
Can We Do More?
Automatic Property Inference

- Automatically infer properties based on data model schema
  - Data model schema: A directed, annotated graph that represents the relations
- Look for patterns in the data model schema and infer a property if a pattern that corresponds to a property appears
- For example, orphan prevention
Can We Do Even More?

Automated Data Model Repair

- **noOrphans**\((X, Y)\) property failing means deleting an object from class \(X\) creates an orphan chain that starts with associated object in class \(Y\)

- **Repair:** Set `:dependent option to :destroy` on association declaration in class \(X\) and on remaining relations in the chain that starts with class \(Y\)

```ruby
Set :dependent => :destroy on all relations in chain
```
Summary

Active Records → Model Extraction → Formal Data Model → Property Inference → Formal Data Model + Properties → Verification → Verification Results

for failing properties → Data Model Repair
## Experiment Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Property Type</th>
<th># Inferred</th>
<th># Timeout</th>
<th># Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>LovdByLess</td>
<td>deletePropagates</td>
<td>13</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>noOrphans</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>transitive</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Substruct</td>
<td>deletePropagates</td>
<td>27</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>noOrphans</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>transitive</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Tracks</td>
<td>deletePropagates</td>
<td>15</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>noOrphans</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>transitive</td>
<td>12</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>FatFreeCRM</td>
<td>deletePropagates</td>
<td>32</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>noOrphans</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>transitive</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>OSR</td>
<td>deletePropagates</td>
<td>19</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>noOrphans</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>transitive</td>
<td>7</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td><strong>145</strong></td>
<td><strong>3</strong></td>
<td><strong>96</strong></td>
</tr>
<tr>
<td>Property Type</td>
<td># Data Model &amp; Application Errors</td>
<td># Data Model Errors</td>
<td># Failures Due to Rails Limitations</td>
<td># False Positives</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------</td>
<td>---------------------</td>
<td>-----------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>deletePropagates</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>noOrphans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>transitive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>deletePropagates</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>noOrphans</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>transitive</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>deletePropagates</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>noOrphans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>transitive</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>deletePropagates</td>
<td>0</td>
<td>18</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>noOrphans</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>transitive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>deletePropagates</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>noOrphans</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>transitive</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3</td>
<td>60</td>
<td>9</td>
<td>28</td>
</tr>
</tbody>
</table>
What About Data Model Actions?

Static Data Model

class User < ActiveRecord::Base
  has_many :todos
  has_many :projects
end
class Project < ActiveRecord::Base
  belongs_to :user
  has_many :todos
  has_many :notes
end
class Todo < ActiveRecord::Base
  belongs_to :user
  belongs_to :project
end
class Note < ActiveRecord::Base
  belongs_to :project
end

Data Model Updates: Actions

class ProjectsController < ApplicationController
def destroy
  @project = Project.find(params[:project_id])
  @project.notes.each do |note|
    note.delete
  end
  @project.delete
  respond_to(...)
end
end
Verification of Data Model Actions

Rails code (+ invariants)

Extractor

Instrumentation → Execution

Abstract Data Store

Verifier

Translation to First Order Logic

First Order Logic Theorem Prover

Falsified (action, invariant) pairs

Verified (action, invariant) pairs
# Abstract Data Stores

## Rails

```ruby
class User
  has_many :todos
  has_many :projects
end

class Project
  belongs_to :user
  has_many :todos
  has_many :notes
end

class Todo
  belongs_to :user
  belongs_to :project
end

class Note
  belongs_to :project
end
```

## Abstract Data Store

```ruby
class User {
  0+ Todo todos inverseof user
  0+ Project projects inverseof user
}

class Project {
  0..1 User user
  0+ Todo todos inverseof project
  0+ Note notes inverseof project
}

class Todo {
  0..1 User user
  0..1 Project project
}

class Note {
  0..1 Project project
}
```
Our library allows developers to specify invariants in native Ruby.
Extraction

Extraction is hard for actions
• Dynamic type system
• Metaprogramming
• Eval
• Ghost Methods such as: User.find_by_name('Rob')

Observations
• The schema is static
• Action declarations are static
• ORM classes and methods do not change their semantic during execution
  • even if the implementation code is generated dynamically
Extraction via Instrumented Execution

- Boot-up the Rails runtime in a simulated environment
  - Without opening sockets or connecting to the database

- Prepare action methods for extraction
  - ORM operations will record their invocation instead of communicating with the database
  - Method calls propagate instrumentation just before execution
  - Extraction is path insensitive, executing both branches subsequently

- Trigger an HTTP request that triggers an action
Verification via Translation to FOL

- A predicate is generated for each class and association
  
  \( \text{User}(o) \) means that \( o \) is an instance of User
  
  \( \text{Project}_\text{user}(t) \) means that \( t \) represents an association between a Project object and User object
  
- Type system constraints become axioms
  \[ \forall u: \text{User}(u) \rightarrow \neg(\text{Project}(u) \lor \text{Todo}(u) \ldots) \]
  
- Cardinality of associations is expressed through axioms
  eg. 0..1:
  \[ \forall t_1, t_2: \]
  \[ (\text{Project}_\text{user}(t_1) \land \text{Project}_\text{user}(t_2) \land \]
  \[ \text{Project}_\text{user}_{\text{lhs}}(t_1) = \text{Project}_\text{user}_{\text{lhs}}(t_2)) \]
  \[ \rightarrow \text{Project}_\text{user}_{\text{rhs}}(t_1) = \text{Project}_\text{user}_{\text{rhs}}(t_2) \]
Translation of Statements to FOL

• An action is a sequential composition of statements.

• Statements
  • A state is represented with a predicate denoting all entities that exist in a state
  • A statement is a migration between states

  e.g., a create Note statement:
  \[
  \neg \text{pre}_\text{state}(\text{newly}_\text{created}()) \\
  \neg \exists t: \text{post}_\text{state}(t) \land \text{Note}_\text{project}_\text{lhs}(t) = \text{newly}_\text{created}() \\
  \forall o: (\text{post}_\text{state}(o) \leftrightarrow (\text{pre}_\text{state}(o) \lor o = \text{newly}_\text{created}()))
  \]
Translation of Loops to FOL

- We only support ForEach loops (for now)
  - They correspond to universal quantification

- Statements can execute multiple times in loops
  - Contexts to differentiate iterations

- Ordering of iterations
  - Iteration interdependence
Inductive Verification

• \( \text{Inv}(s) \) is a formula denoting that all invariants hold in state \( s \)

• \( \text{Action}(s, s') \) is a formula denoting that the action may transition from state \( s \) to state \( s' \)

Check if: \( \forall s, s': \text{Inv}(s) \land \text{Action}(s, s') \rightarrow \text{Inv}(s') \)
Experiments

Experimented on 3 open source Rails applications
• FatFreeCRM, Tracks, Kandan
• 272 actions, 23 invariants

Identified 4 bugs
• Reported to original developers
• All immediately confirmed and, since, fixed
• Missed by previous verification efforts on these applications
# Experiments

<table>
<thead>
<tr>
<th></th>
<th>FatFreeCRM</th>
<th>Tracks</th>
<th>Kandán</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>30358</td>
<td>18023</td>
<td>2173</td>
</tr>
<tr>
<td># ADS Nodes</td>
<td>85447</td>
<td>37755</td>
<td>907</td>
</tr>
<tr>
<td># Nodes after optimization</td>
<td>1611</td>
<td>1483</td>
<td>280</td>
</tr>
<tr>
<td># Classes</td>
<td>30</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td># Actions</td>
<td>167</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td># Invariants</td>
<td>8</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td># Empty actions</td>
<td>107</td>
<td>52</td>
<td>31</td>
</tr>
<tr>
<td>Avg. # of predicates</td>
<td>286</td>
<td>205</td>
<td>69</td>
</tr>
<tr>
<td>Theorem prover peak memory</td>
<td>243 Mb</td>
<td>203 Mb</td>
<td>126 Mb</td>
</tr>
<tr>
<td>Avg. time per action/invariant</td>
<td>3.1 sec</td>
<td>40.5 sec</td>
<td>10.5 sec</td>
</tr>
<tr>
<td># Action/invariant pairs</td>
<td>480</td>
<td>180</td>
<td>20</td>
</tr>
<tr>
<td>Verified</td>
<td>468</td>
<td>133</td>
<td>17</td>
</tr>
<tr>
<td>Falsified</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Inconclusive</td>
<td>8</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>False positives</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Detected Exceptions</td>
<td>0</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>
Publications


• Ivan Bocic, and Tevfik Bultan. **Inductive Verification of Data Model Invariants for Web Applications**. In *Proc. International Conference on Software Engineering (ICSE), 2014*

• Jaideep Nijjar, Ivan Bocic, and Tevfik Bultan. **Data Model Property Inference, Verification and Repair for Web Applications**. (Submitted to *ACM Translations on Software Engineering and Methodology*).