Automatically Generating Predicates and Solutions for Configuration Troubleshooting

* Ya-Yunn Su  Jason Flinn
NEC Laboratories  University of Michigan
America  Michigan
Troubleshooting misconfigurations is hard!

• Users may have to
  – Edit configuration files
  – Resolve library dependencies
  – Change environment variables

• Automated troubleshooting tools can help
  – Chronus: finds when a misconfiguration entered
  – AutoBash: automatically resolves misconfigurations
  – Both assume test cases or solutions exist
Current method: manual predicate creation

- Predicates
  - Test if an application works or not
  - Returns true/false if the test passes/fails
- E.g. test if an Apache Web server is working
  
  wget http://localhost

- Manually writing predicates requires
  - Experts and time
  - Domain knowledge
- Can we automatically generate predicates?
Limitations in existing approaches

- Automatic test case generation requires
  - Program source code or specifications
- Automatic solution generation requires
  - Golden state as a reference

- Users already troubleshoot misconfigurations
  - They try potential solutions
  - They test if a solution works

Valuable source to generate predicates/solutions for others to use
Generating predicates from user traces

- Users troubleshoot using our modified shell
- Our modified shell generates:
  - Which command is a predicate
  - If a predicate succeeds/fails
  - Which commands are solutions
Goals

• Minimize false positives
  – A false positive is worse than a false negative
  – Aggregate across multiple user traces

• Be as unobtrusive as possible
  – Users do not need to provide extra input

• Generate complete predicates
  – Predicates may contain multiple steps
Minimizing false positives

- Observation: troubleshooting pattern
  - Users test the system state multiple times
  - Users rely on output to know test outcome
- Generate predicates following this pattern

```
Command C

Time

System was not working

System was working
```

Command C₀ = False

Command C₁ = True
Our approach

• Predicates
  – Repeated commands
  – Differ in more than two out of three output features

• Output features for a command:
  – exit code: the return value of a process
  – screen output contains error message
  – output set: kernel objects a command modifies
Tracking output sets

- Output set: kernel objects a command causally affects

Command: `echo hi > foo`  
Output set = `{file foo}`

- Process `echo` forked
- File `foo` created
- File `foo` exits
- Process `echo` exits

Output set:
- Process `echo`
- File metadata `foo`
- File content `foo`
- Directory entry `foo`
Example

% cvs -d /home/cvsroot import test_project
cvs [import aborted]: /home/cvsroot/CVSROOT:
   No such file or directory

% cvs -d /home/cvsroot init

% cvs -d /home/cvsroot import test_project
N  test_project/testfile
No conflicts created by this import

Problem: CVS repository not initialized
Example

```
% cvs -d /home/cvsroot import test_project
  cvs [import aborted]: /home/cvsroot/CVSROOT:
      No such file or directory

% cvs -d /home/cvsroot init

% cvs -d /home/cvsroot import test_project
  N  test_project/testfile
  No conflicts created by this import
```

• Find repeated commands
Example

% cvs –d /home/cvsroot import test_project

cvs [import aborted]: /home/cvsroot/CVSROOT:
    No such file or directory

% cvs –d /home/cvsroot init

% cvs –d /home/cvsroot import test_project

N  test_project/testfile

No conflicts created by this import

• Compare output features of repeated commands
Example

% cvs -d /home/cvsroot import test_project    exit code = 1
cvs [import aborted]: /home/cvsroot/CVSROOT:
    No such file or directory

% cvs -d /home/cvsroot init

% cvs -d /home/cvsroot import test_project    exit code = 0
N  test_project/testfile
No conflicts created by this import

Output feature: exit codes differ
Example

```
% cvs -d /home/cvsroot import test_project
  cvs [import aborted]: /home/cvsroot/CVSROOT:
    No such file or directory

% cvs -d /home/cvsroot init

% cvs -d /home/cvsroot import test_project
  N  test_project/testfile
  No conflicts created by this import
```

- Output feature: screen outputs differ
  - First execution prints error message
  - Second execution does not contain error msg
Example

% cvs –d /home/cvsroot import test_project

cvs [import aborted]: /home/cvsroot/CVSROOT:
   No such file or directory

=> Output set = {}

% cvs –d /home/cvsroot init

% cvs –d /home/cvsroot import test_project

N  test_project/testfile

No conflicts created by this import

=> Output set = {file:/home/cvsroot/
                 test_project, ...}

• Output feature: output sets differ
  • First execution: output set is empty
  • Second execution: output set contains created files
Example

% cvs –d /home/cvsroot import test_project  => predicate fails
  cvs [import aborted]: /home/cvsroot/CVSROOT:
    No such file or directory

% cvs –d /home/cvsroot init

% cvs –d /home/cvsroot import test_project  => predicate succeeds
  N  test_project/testfile
  No conflicts created by this import

• Repeated commands differ in three output features
• First execution considered to be a failed predicate
Generating complete predicates

• Some predicates depend on preconditions to be executed first to work correctly

Problem: user2 is not in CVS group

Initial state: CVS repository is empty
Causal relationships between commands

% echo hi > foo
% cat foo

• “cat foo” causally depends on “echo”
  cat is in echo’s output set
Applying causality to find preconditions

- Candidate preconditions: find
  - All commands a predicate depends on
  - All commands whose output set a predicate is in

We also find solution!
Heuristic to differentiate them

- **Solutions**: occurred after all failed predicates
- **Preconditions**:
  - occurred before any failed predicate

```
cvs import as user1
cvs co as user2 fails
Add user2 to CVS group
cvs co as user2 succeeds

File: test_project
File: /etc/group
```
Ranking solutions

• Users solve the same problem differently

• Goal: better solutions are ranked higher
  – Heuristic: solutions applied by more users are better
  – Aggregate solutions among traces and rank them

• Ex. Apache not having search permission
  – chmod 777 /home/USERID
  – chmod 755 USERID/
  – chmod 755 /home/USERID

Different commands can be used to do the same thing.
• State delta: the difference in system state caused by the execution of a command
  – Track output set for that command
  – Compute diff for each entity in the output set

• Solution ranking results:

  Group 1 (size = 2)
  1. chmod 755 /home/USERID
  2. chmod 755 USERID/

  Group 2 (size = 1)
  1. chmod 777 /home/USERID
Evaluation

• Questions:
  – How well can we generate predicates?
  – How well does the solution ranking heuristic work?

• Methodology
  – Conducted a user study of user troubleshooting
  – Generate predicates/solutions from traces
  – Manually verify predicate correctness
User study procedure

• 12 participants:
  – graduate students
  – system administrators

• Each given four configuration problems
  – Two CVS and two Apache configuration problems
  – Each problem runs in a virtual machine

• Collected traces of users troubleshooting
  – All commands a user typed
  – Collect exit code, screen output, and output set
### Predicate result summary

<table>
<thead>
<tr>
<th></th>
<th>CVS problem 1</th>
<th>CVS problem 2</th>
<th>Apache problem 1</th>
<th>Apache problem 2</th>
</tr>
</thead>
<tbody>
<tr>
<td># of correct predicates</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td># of wrong predicates</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total # of traces</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

- All correct predicates are complete
- Very few wrong predicates (false positives)
- Both false positives come from traces of user not solving the problem
- Why were no predicates generated for some traces?
Apache problem: predicate results

- Problem: Apache process not having search permission on /home/USERID
- Solution: give /home/USERID search permission

<table>
<thead>
<tr>
<th>Predicates Generated</th>
<th>Number of traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>No predicate generated (User did not use repeated commands)</td>
<td>3</td>
</tr>
<tr>
<td>No predicate generated (User did not fix the problem)</td>
<td>2</td>
</tr>
<tr>
<td>Incorrect predicate (User did not fix the problem)</td>
<td>1</td>
</tr>
</tbody>
</table>

- To minimize FP, we compare current directory and user id
- User executed commands in different directories
Apache problem: predicate results

- Problem: Apache process not having search permission on /home/USERID
- Solution: give /home/USERID search permission

<table>
<thead>
<tr>
<th>Predicates Generated</th>
<th>Number of traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>No predicate generated (User did not use repeated commands)</td>
<td>3</td>
</tr>
<tr>
<td>No predicate generated (User did not fix the problem)</td>
<td>2</td>
</tr>
<tr>
<td>Incorrect predicate (User did not fix the problem)</td>
<td>1</td>
</tr>
</tbody>
</table>

- User did not fix the problem => output features did not differ
Apache problem: predicate results

- Problem: Apache process not having search permission on /home/USERID
- Solution: give /home/USERID search permission

### Predicates Generated

<table>
<thead>
<tr>
<th>Predicates Generated</th>
<th>Number of traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>No predicate generated (User did not use repeated commands)</td>
<td>3</td>
</tr>
<tr>
<td>No predicate generated (User did not fix the problem)</td>
<td>2</td>
</tr>
<tr>
<td>Incorrect predicate (User did not fix the problem)</td>
<td>1 (circled)</td>
</tr>
</tbody>
</table>

- Predicate: open configuration file in an editor
- Could be eliminated if we asked user whether problem was fixed
Apache problem: solution ranking results

<table>
<thead>
<tr>
<th>Solution</th>
<th>Number of Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmod 755 /home/USERID</td>
<td>2</td>
</tr>
<tr>
<td>chmod –R 777 USERID/</td>
<td>1</td>
</tr>
<tr>
<td>chmod o+rx /home/USERID</td>
<td>1</td>
</tr>
<tr>
<td>chmod 777 /home/USERID</td>
<td>1</td>
</tr>
<tr>
<td>vim /etc/httpd/conf/httpd.conf</td>
<td>1</td>
</tr>
</tbody>
</table>

Why is editing configuration file a solution?

- Predicate: `apachectl stop`
- User-introduced errors in conf file caused `apachectl stop` fail
Future work

• Extend this work to handle GUI applications

• Challenges:
  – identifying individual tasks, finding repeated tasks
  – exit code does not map to each task

• Advantages: more semantic information
Conclusion

- Automatically generate predicates and solutions from user troubleshooting traces

- Our approach
  - Minimizes false positives
  - Is unobtrusive to users
  - Generates complete predicates

Thank you!