Automatic Trigger Generation for Rule-based Smart Homes

ACM SIGPLAN PLAS, Vienna, Austria
10-24-2016

Chandrakana Nandi, Michael D. Ernst
UW Seattle, USA
Common architecture
How to control your home?
How to control your home?

Automation rules:

when I come home then turn lights on
How to control your home?

Automation rules are easy and useful

Ur+ CHI 2014, 2016
Ur+ HUPS 2014
Dey+ Pervasive 2006
How to control your home?

Writing **correct** automation rules is hard

Huang+ Ubicomp 2015
How to control your home?

Writing **correct** automation rules is hard

Huang+ Ubicomp 2015
Effects of wrong rules

- Likely unexpected behavior
- Security vulnerabilities
Overview

● Background on automation rules
● Problem statement
● Solution
● Algorithm and tool development
● Experiments
Overview

- Background on automation rules
- Problem statement
- Solution
- Algorithm and tool development
- Experiments
Rule Example

```plaintext
rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```
Rule Example

```plaintext
rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```
rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
Rule Example

```plaintext
rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```

```
16
rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```

rule "Away rule"
when
    Item State_Away changed
or Item State_Sleeping changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
  Item State_Away changed
  or Item State_Sleeping changed
then
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end
Overview

- Background on automation rules
- Problem statement
- Solution
- Algorithm and tool development
- Experiments
Possible mistakes in rules
Wrong trigger block

```java
rule "Away rule"
when
    Item State_Roomheater changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```
Wrong trigger block

rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
Wrong trigger block

```java
rule "Away rule"
when
    Item trigger_1 changed
    Item trigger_2 changed
    Item trigger_n changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```
Conflicts

rule “rule 1”
when
    Item owner_entering_home changed
then
    if (owner_entering_home == true) {
        sendCommand (hall_light, “ON”)
    }
end

rule “rule 2”
when
    Item past_midnight changed
then
    if (past_midnight == true) {
        sendCommand (hall_light, “OFF”)
    }
end

(owner_entering_home == true && past_midnight == true)
- Wrong trigger blocks
- Conflicts
● Wrong trigger blocks
● Conflicts
Why is it bad?
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end

!(State_Away = ON && State_Sleeping = ON)
```
rule "Away rule"
when
  Item State_Away changed
then
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end
```
rule "Away rule"
when
  Item State_Away changed
then
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end
rule "Away rule"
when
  Item State_Away changed
then
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
   Item State_Away changed
then
   if (State_Away.state == ON) {
      if (State_Sleeping.state != OFF) {
         postUpdate (State_Sleeping, OFF)
      }
   }
end
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
Both states can be set to true!
rule "Visitor notification system rule"
when
    Item State_Sleeping changed
then
    if (State_Sleeping.state == ON) {
        postUpdate (Notification_System , OFF)
    } else {
        postUpdate (Notification_System , ON)
    }
end
Example Attack

**Visitor notification system rule**

```java
rule "Visitor notification system rule"
when
  Item State_Sleeping changed
then
  if (State_Sleeping.state == ON) {
    postUpdate (Notification_System, OFF)
  } else {
    postUpdate (Notification_System, ON)
  }
end
```

Wrongly deactivates notification system

**Away rule**

```java
rule "Away rule"
when
  Item State_Away changed
then
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  } else {
    postUpdate (Notification_System, OFF)
  }
end
```
Overview

- Background on automation rules
- Problem statement
- **Solution**
- Algorithm and tool development
- Experiments
Solution

```java
rule "Away rule"
when
    Item State_Away changed
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
```
Solution

rule "Away rule"
when
    Item State_Away changed
    or Item State_Sleeping changed  // Fix
then
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
Overview

- Background on automation rules
- Problem statement
- Solution
- Algorithm and tool development
- Experiments
TrigGen: automatically infer triggers from actions using static analysis

Idea: live items must be triggers
Idea: live items must be triggers

Items that are read from before being written to, at the beginning of the action block
rule "Away rule"
when
    Item State_Away changed
then
    State_Notify = ON
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end
rule "Away rule"
when
    Item State_Away changed
then
    State_Notify = ON
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end

- Identify all items in the action block AST
  - potential triggers
rule "Away rule"
when
    Item State_Away changed
then
    State_Notify = ON
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
            postUpdate (State_Sleeping, OFF)
        }
    }
end

- Identify all items in the action block AST
  - potential triggers
rule "Away rule"
when
  Item State_Away changed
then
  State_Notify = ON
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end

- Identify all items in the action block AST
  - *potential triggers*

- eliminate those that are not live
  - *redundant triggers*
    - State_Notify
rule "Away rule"
when
  Item State_Away changed
then
  State_Notify = ON
  if (State_Away.state == ON) {
    if (State_Sleeping.state != OFF) {
      postUpdate (State_Sleeping, OFF)
    }
  }
end

- Identify all items in the action block AST
  - *potential triggers*
- eliminate those that are not *live*
  - *redundant triggers*
    - State_Notify
- State_Away, State_Sleeping: *live*
Implementation

rule

```
rule "Rule 1"
  when
  trigger
  then
  action
end
```

item repository

identify all potential triggers

eliminate redundant triggers by live variable analysis

TrigGen

Laundry_Machine

State_Sleeping

wrong/extra

missing

Comparer

end-user written

State_Away

Laundry_Machine

State_Away

State_Sleeping

State_Sleeping
Overview

- Background on automation rules
- Problem statement
- Solution
- Algorithm and tool development
- Experiments
Experiments

- 96 real end-user written rules for openHAB
- Action block size: 1 - 220 LOC
- Featuring categories such as...
Experiments

- Ground truth
  - Set of necessary and sufficient triggers, i.e. all non-redundant triggers
  - Verified by
    - contacting the end user
    - manual inspection of rules
## Trigger generation

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrigGen suggested a set of necessary and sufficient triggers</td>
<td>91 (95%)</td>
</tr>
<tr>
<td>False positives</td>
<td>0</td>
</tr>
<tr>
<td>False negatives</td>
<td>5 (5%)</td>
</tr>
<tr>
<td>Missing triggers detected</td>
<td>77 (80%)</td>
</tr>
</tbody>
</table>
Number of missing triggers
Number of missing triggers

55%
## Conflicts

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total conflicts detected</td>
<td>18</td>
</tr>
<tr>
<td>True positives</td>
<td>11 (61%)</td>
</tr>
<tr>
<td>False negatives</td>
<td>0</td>
</tr>
</tbody>
</table>
More in the paper
Conflict resolution
Group enumeration
Proving non-live triggers as redundant
Remarks

- TrigGen is applicable to any domain that has trigger based rules
- We aimed at home automation involving
  - end users
  - different deployments: every home is different!
Conclusions

TrigGen automatically generates a set of necessary and sufficient triggers so that rules don’t have:

- likely unexpected behavior
- certain security vulnerabilities

TrigGen found 80% real rules used for experimentation to have insufficient triggers