Automatic Trigger Generation for Rule-based Smart Homes

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

Chandrakana Nandi, Michael D. Ernst
UW Seattle, USA

SmartThings™

Microsoft®

HomeOS

wink

freedomotic

openHAB
empowering the smart home

IFTTT
ifthisthenthat

eclipse smarthome

hue
PHILIPS

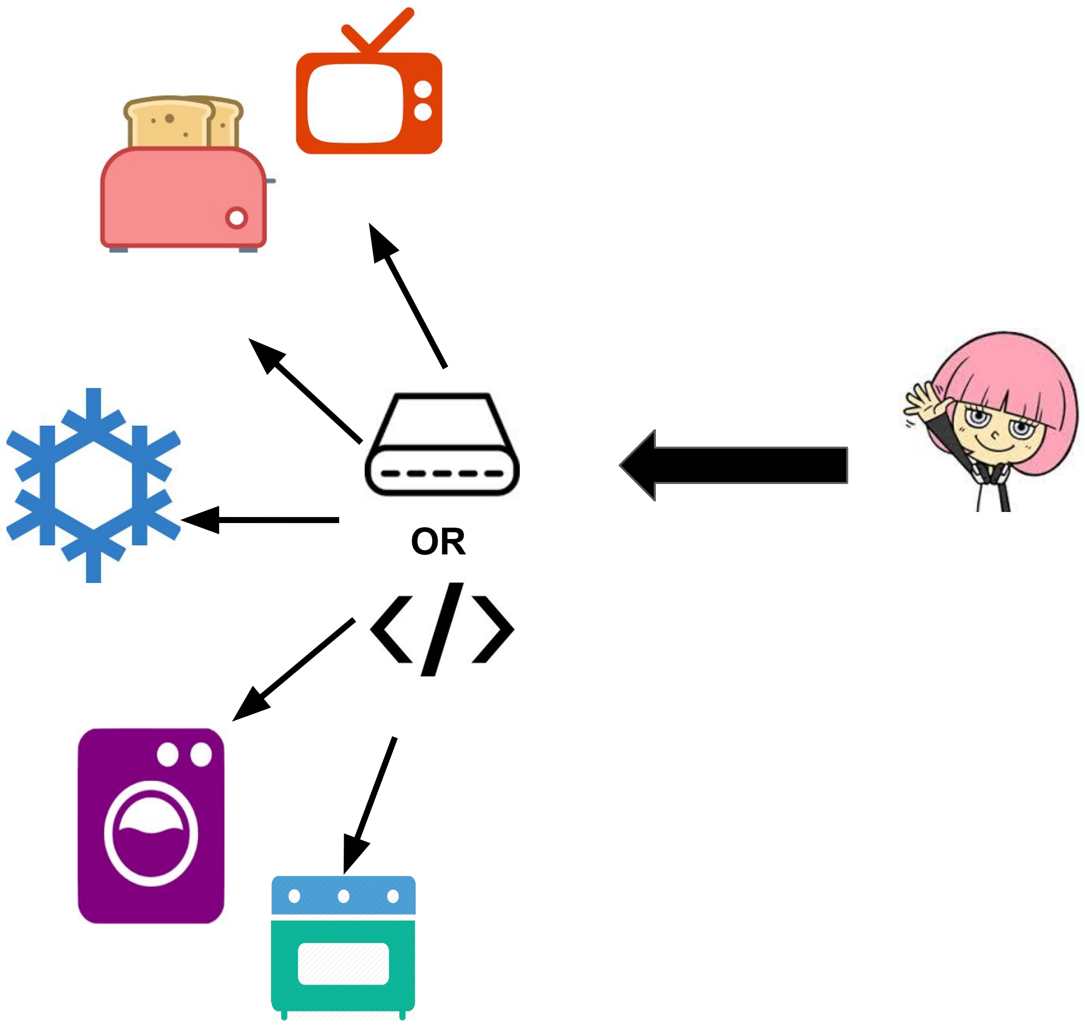
ago control

Domoticz
control at your fingertips.

belkin™

Google Brillo OS
Operating System for Internet of Things

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?

mental model

≠

actual rule

```
rule "start laundry"
when
   Item laundry_machine changed
then
   if (laundry_machine == FULL) {
      sendCommand(laundry_machine, "ON")
   }
end
```

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

```
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

```
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

```
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
```

trigger block

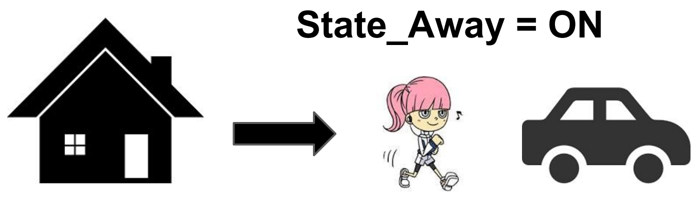
Rule Example

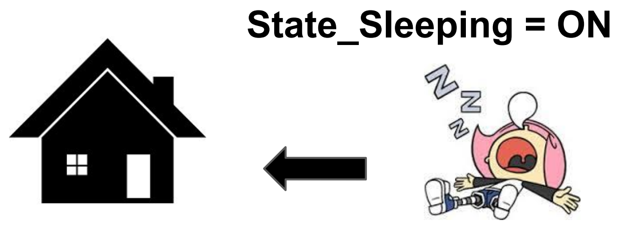
```
rule "Away rule"
when                          trigger item
    Item State_Away changed                       trigger block
    or Item State_Sleeping changed
then                                 trigger item
    if (State_Away.state == ON) {
        if (State_Sleeping.state != OFF) {
          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

Rule Example

```
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
```

action block





```
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

```
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

```
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)

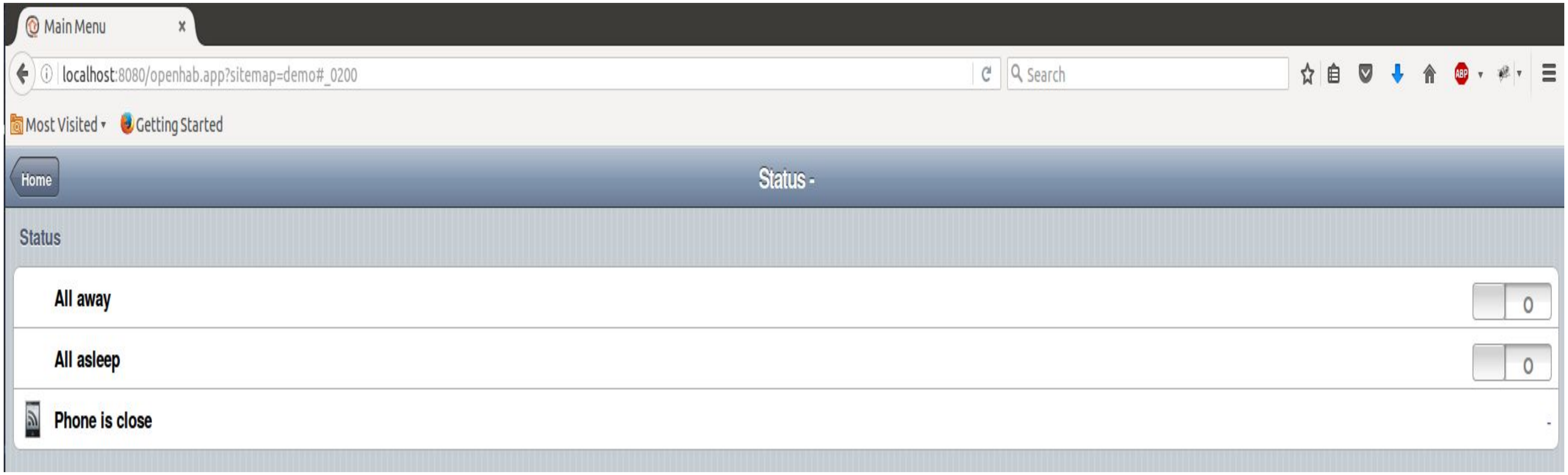
- **Wrong trigger blocks**
- **Conflicts**

- **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
```

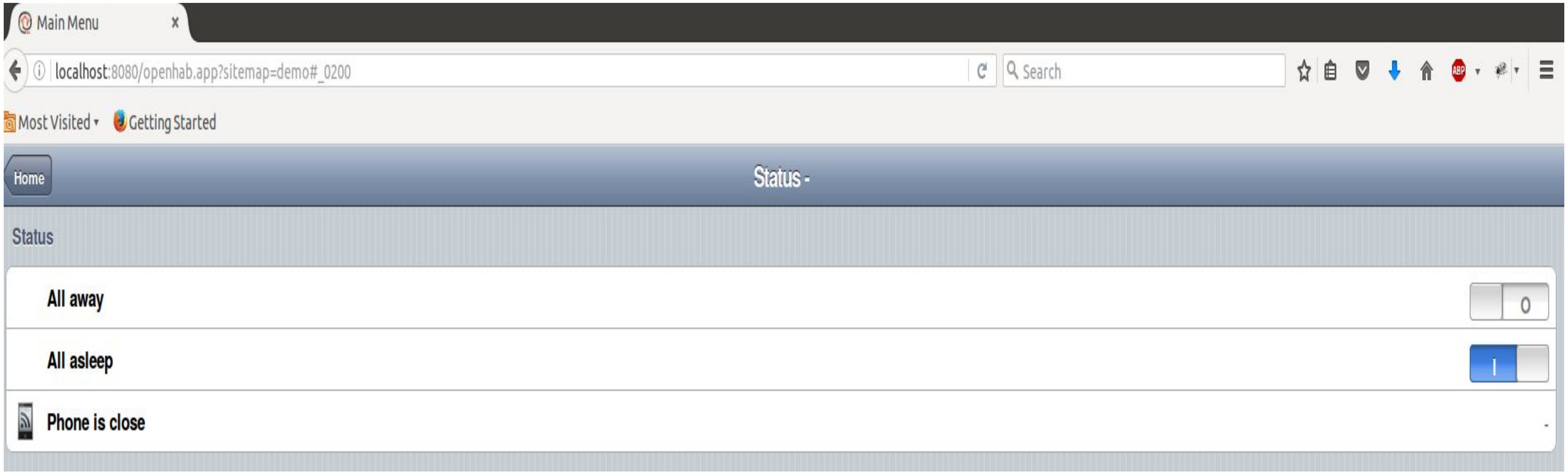
!(State_Away = ON && State_Sleeping = ON)



Status

All away	0
All asleep	0
Phone is close	-

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

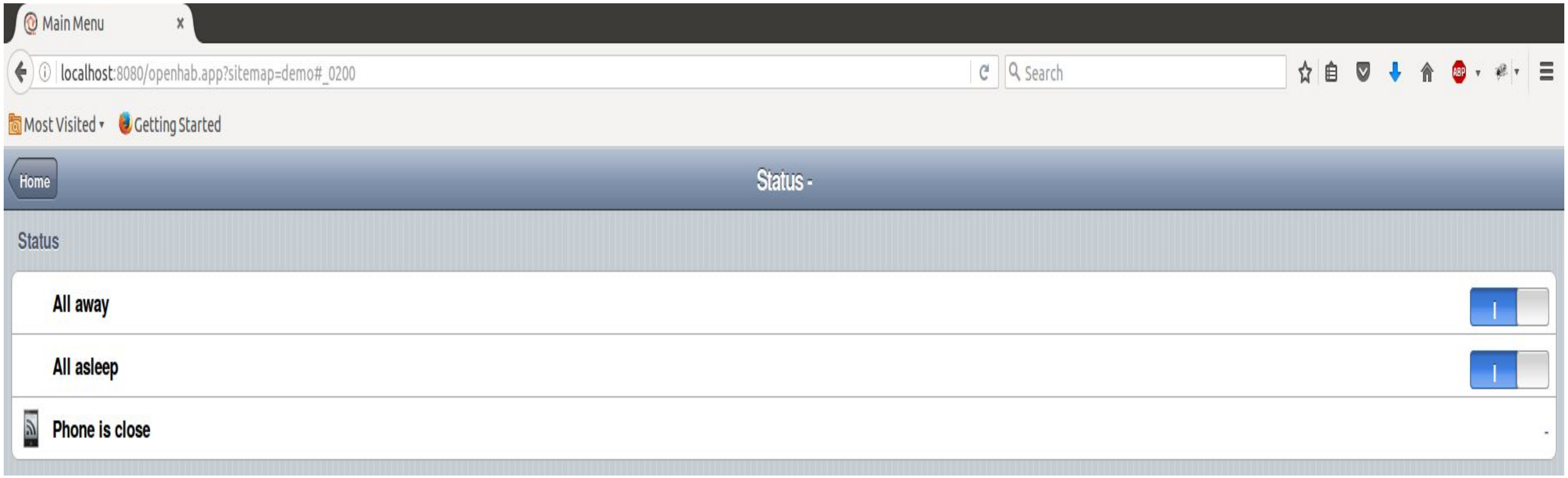


All away — 0

All asleep — I

Phone is close -

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


Main Menu

localhost:8080/openhab.app?sitemap=demo#_0200

Status -

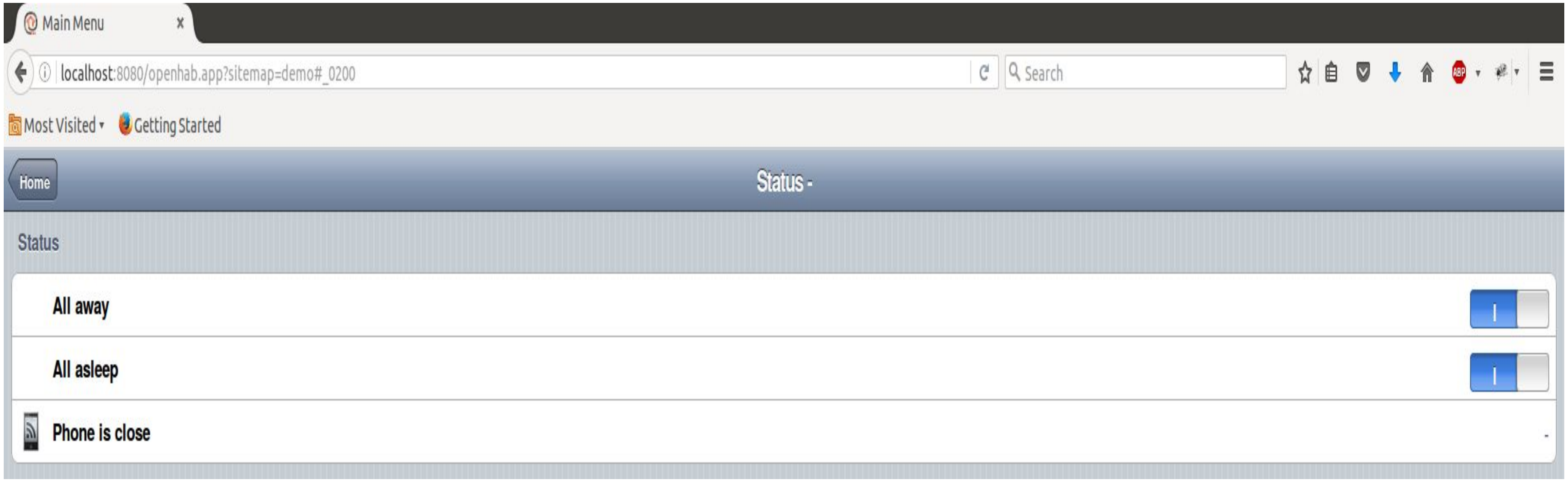
Status

All away

All asleep

Phone is close -

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



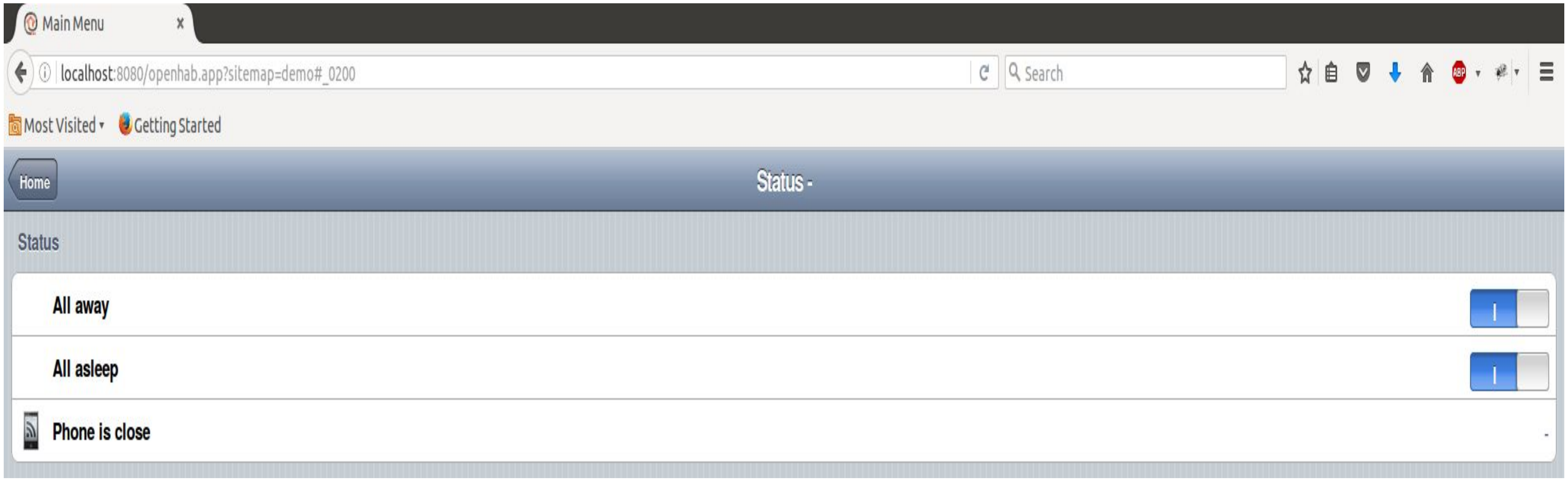
Status

All away

All asleep

Phone is close -

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

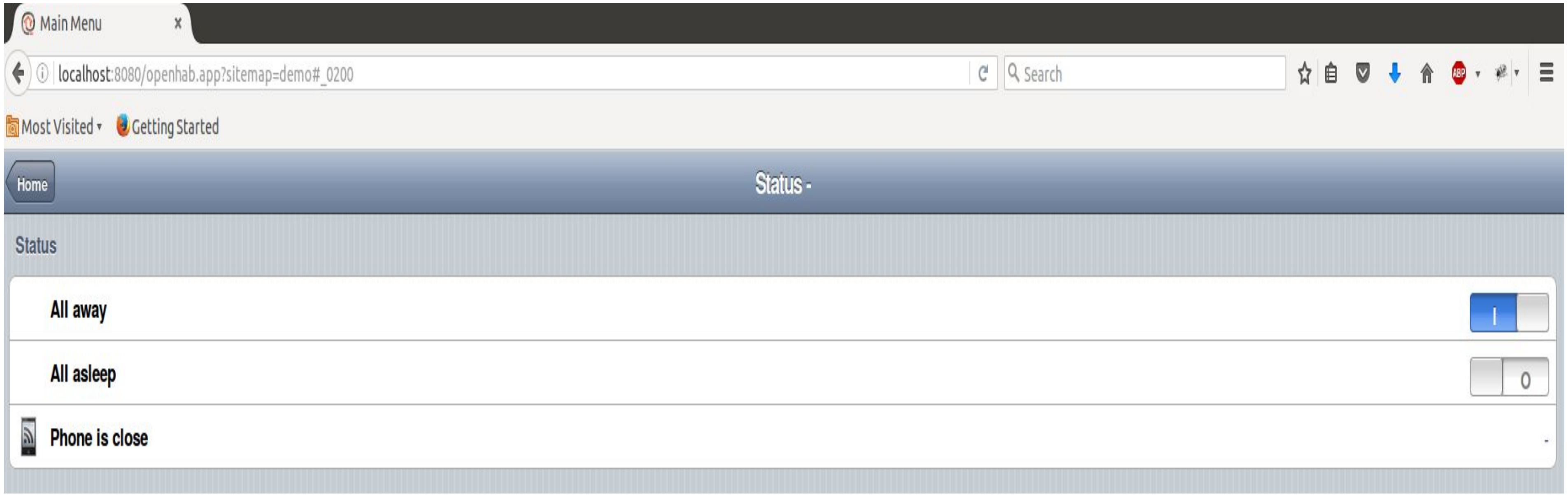


All away

All asleep

Phone is close

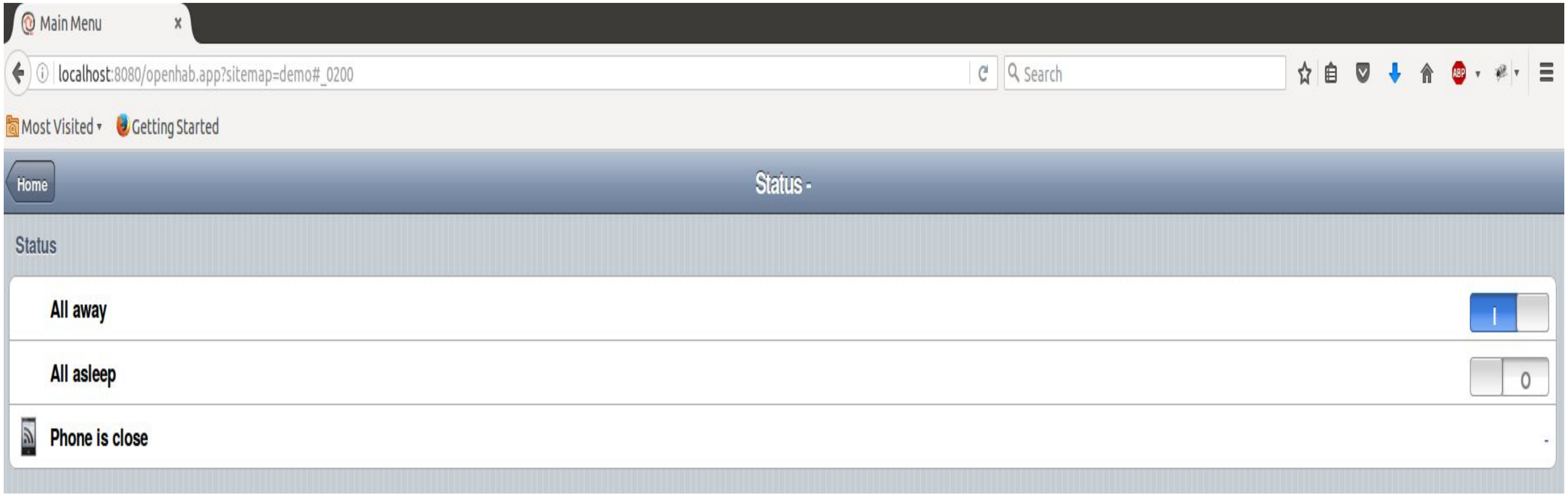
```
rule "Away rule"
when
     Item State_Away changed
then
     if (State_Away.state == ON) {
            if (State_Sleeping.state != OFF) {
                postUpdate (State_Sleeping, OFF)
            }
     }
end
```



Status

All away	I
All asleep	0
Phone is close	-

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


Main Menu

localhost:8080/openhab.app?sitemap=demo#_0200

Search

Most Visited Getting Started

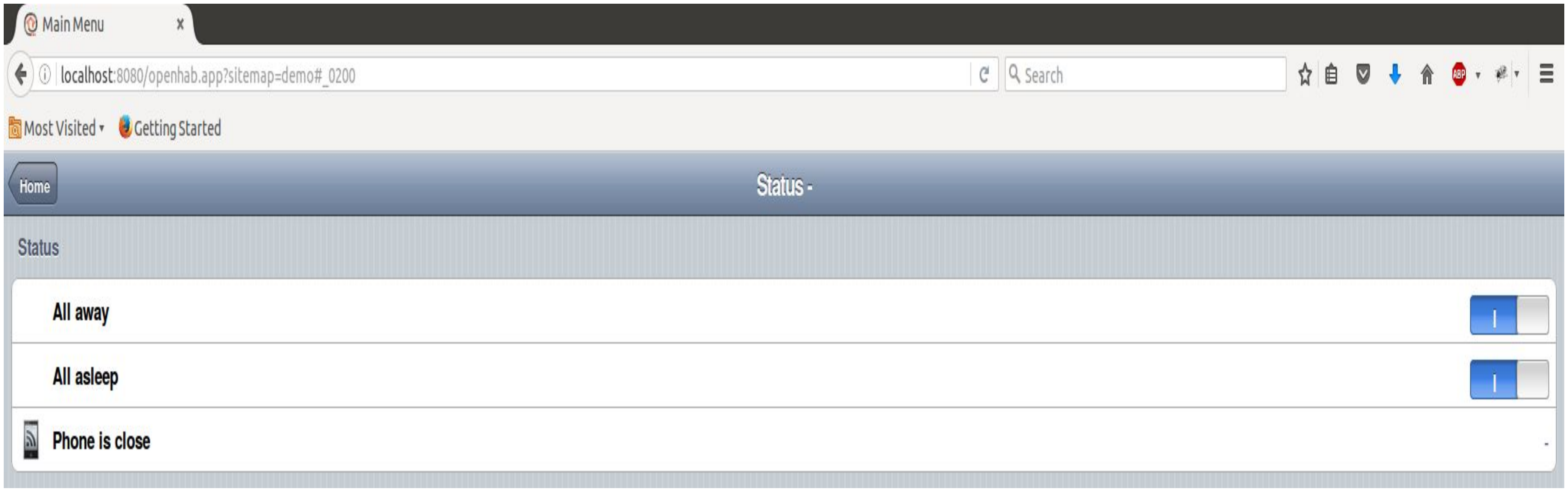
Home

Status -

Status

All away	I
All asleep	0
Phone is close	-

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



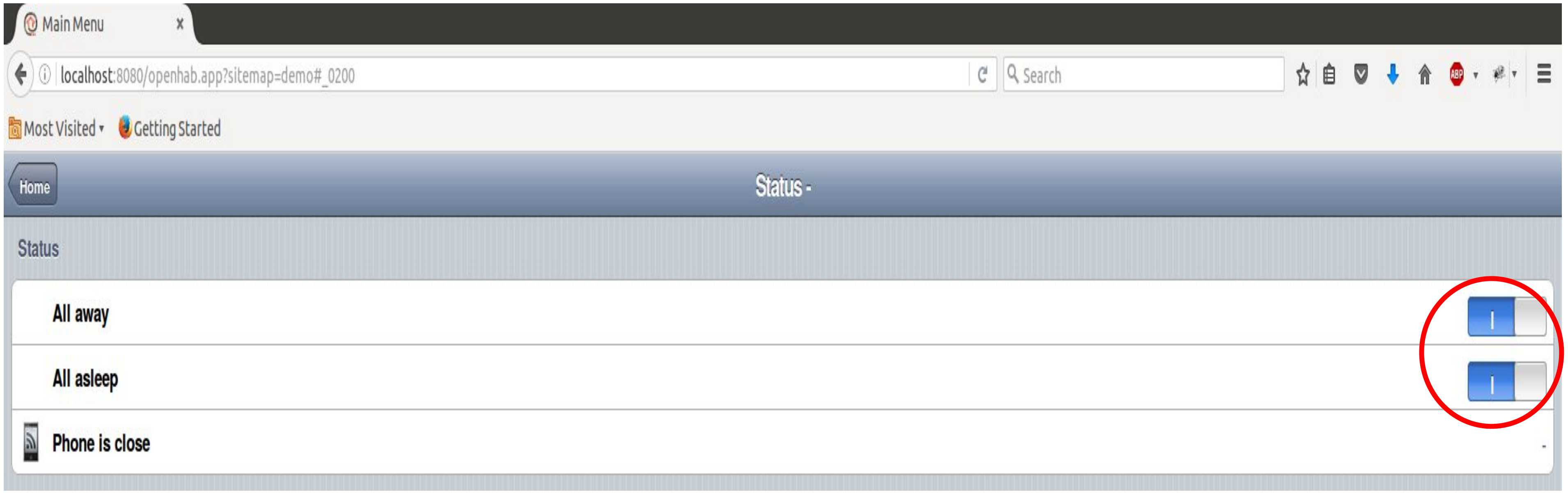
Status

All away

All asleep

Phone is close -

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



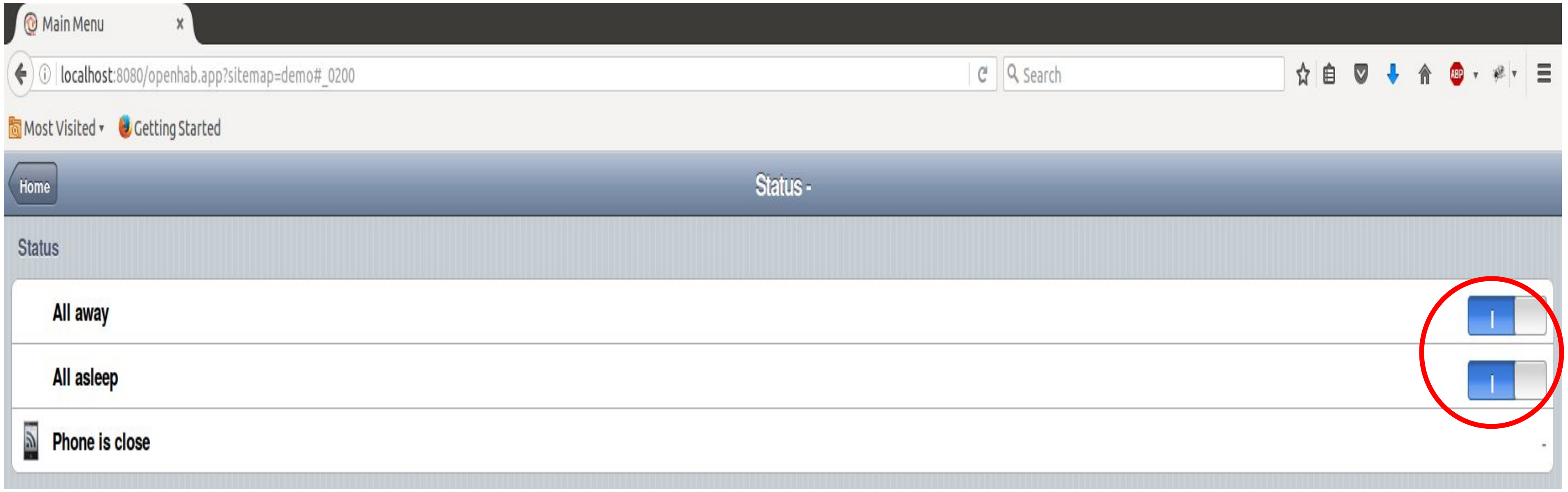
Status

All away

All asleep

Phone is close

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



Status

All away

All asleep

Phone is close -

Both states can be set to true!

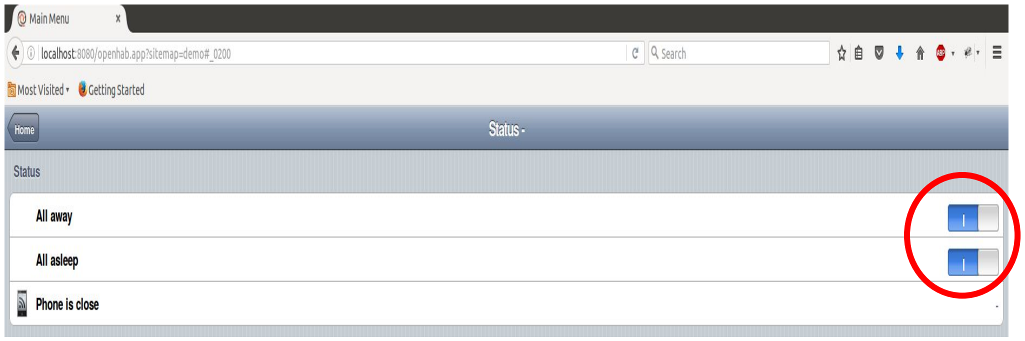
```
rule "Visitor notification system rule"
when
   Item State_Sleeping changed
then
   if (State_Sleeping.state == ON) {
      postUpdate (Notification_System , OFF)
   } else {
      postUpdate (Notification_System , ON)
   }
end
```

Example Attack

```
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
```

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

Wrongly deactivates notification system



Status -

Status

All away

All asleep

Phone is close

Overview

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

Solution

```
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

$\downarrow$

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***

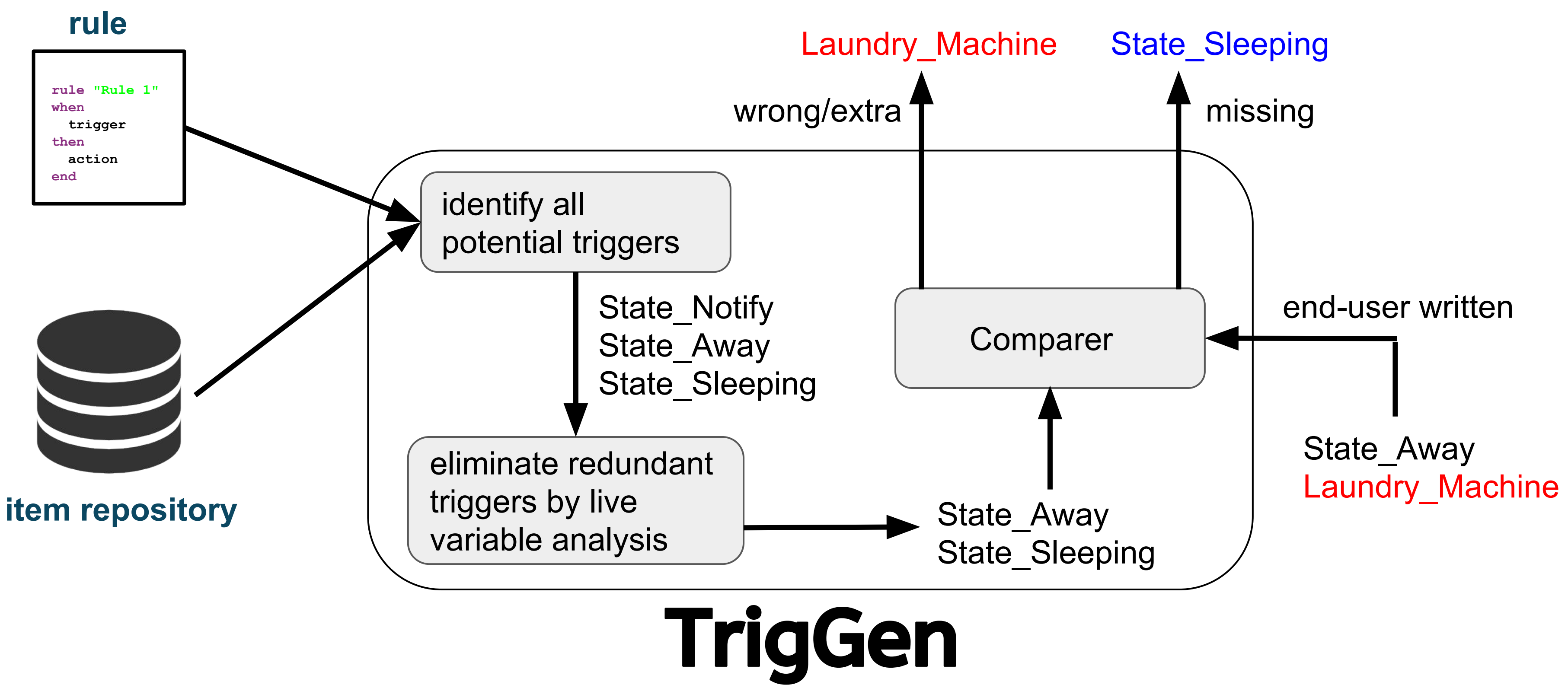
```
rule "Away rule"
when
     Item State_Away changed
then
     State_Notify = ON
     if (State_Away.state == ON) {
           if (State_Sleeping.state != OFF) {
                postUpdate (State_Sleeping, OFF)
           }
    }
end
```

- 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 1"
when
  trigger
then
  action
end
```

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

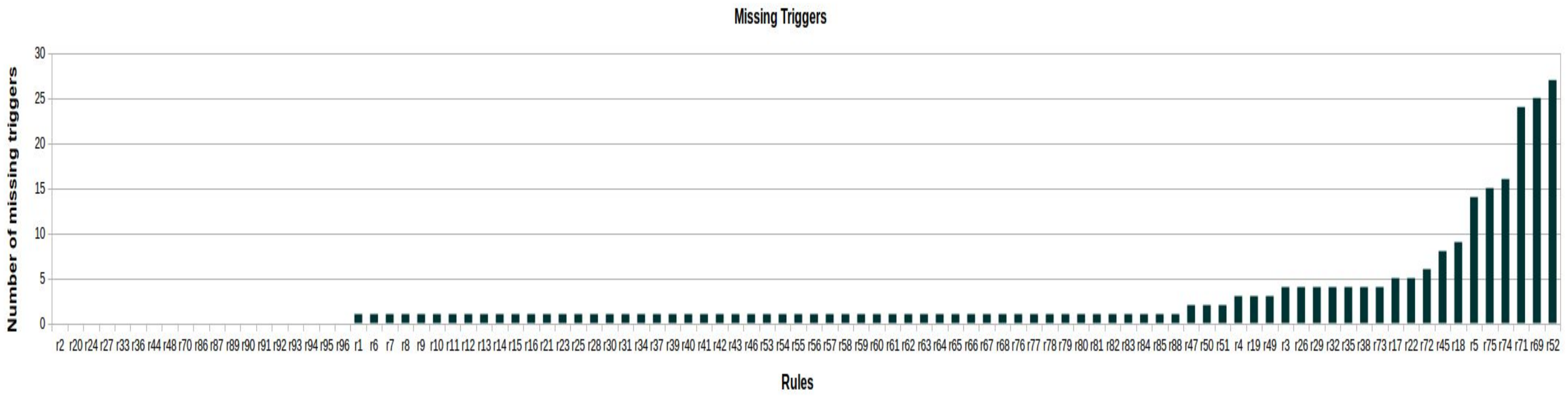
Experiments

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

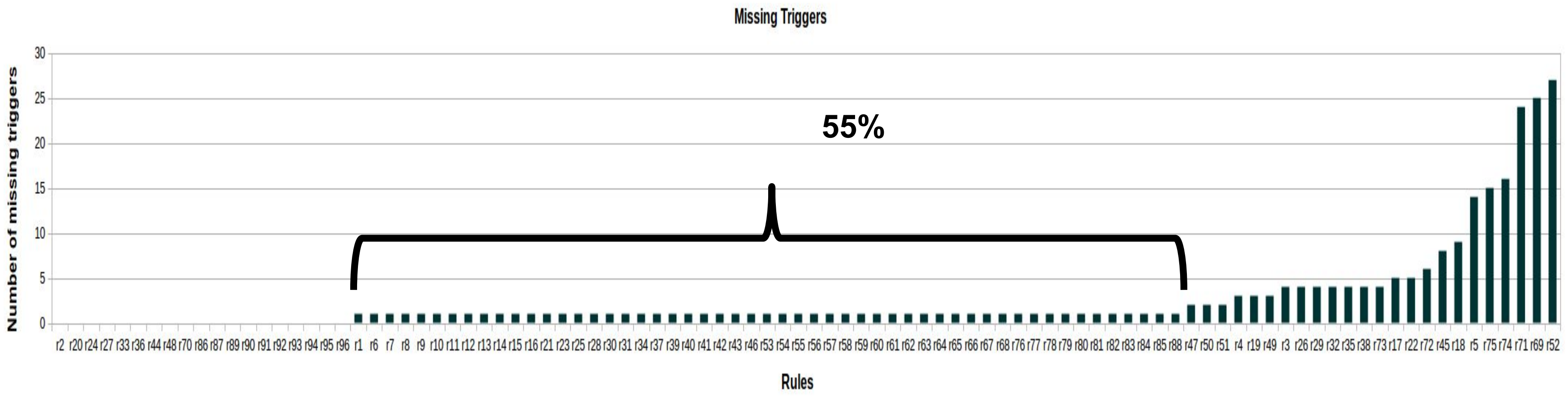
Trigger generation

TrigGen suggested a set of necessary and sufficient triggers	91 (95%)
False positives	0
False negatives	5 (5%)
Missing triggers detected	77 (80%)

Number of missing triggers



Number of missing triggers



Conflicts

Total conflicts detected	18
True positives	11 (61%)
False negatives	0

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