Distributed State Monitoring User Guide

COMP SCI 4ZP6 – GROUP 1

ANDREW AZORES, EVAN HOLTROP, JAZZ KERSELL, DARREN KITAMURA

Table of Contents

0 Revision History

Developer	Date	Changes	Revision
Darren Kitamura	February 27, 2015	Initial	0

1 Legal and Copyright Information

1.1 Legal

By usage of this software on you agree that McMaster's Computing and Software department and the developers are not in anyway liable for any damage on hardware or persons.

1.2 Copyright

2 Introduction

Autonomous state monitoring is a practical Android based implementation of a theoretical algorithm. Using various Android devices on a network one can interact with system and ensure the conditions for the global state set are either satisfied or violated. The use of the Android platform allows for expansion into other things such as interfacing with drones or other peripherals.

3 Definitions

LTL – Linear Temporal Logic
NFC – Near Field Communication

4 Installation

4.1 Android

There are only a few steps to make sure that this application runs correctly from the app side.

1) Transfer the pre-compiled APK to the device and install it on the system.
2) Transfer the pre-generated JSON automata file and a Conjunct Mapping file. For more instruction on this please refer to the PC based instructions.
3) Ensure all devices are on the same network.
4) Launch the application.

4.2 PC (Linux)

The preferred Linux distribution for this setup is Arch-Linux which can be downloaded for free from the Arch website.

1) Install SPOT which can be found in the package manager as "spot", and if you wish to be able to view the automata graphically install GraphViz (optional)
2) Install Ruby
3) Use the bash script included in the Git repository under /ltl-testing and Ruby script under /tools/convert_automaton for generating the automata as below:

```
./formula_to_automaton.sh "Your LTL Formula here" | ruby automaton_to_json.rb > automaton.json
```

4) Create an initial_state.json file to set the initial state of each device. Example format below:

```
    {
"^o":"InitialState",
"valuations":[
  {
    "^o":"valuation",
    "variables":[
      {
        "variable":"x1",
        "value":"0.0"
      }
    ]
  },
  {
    "^o":"valuation",
    "variables":[
      {
        "variable":"x2",
```

```
            "value":"0.0"
          }
        ]
      },
      {
      "^o":"valuation",
        "variables":[
          {
            "variable":"x3",
            "value":"0.0"
          }
        ]
      },
      {
        "^o":"valuation",
        "variables":[
          {
            "variable":"x4",
            "value":"0.0"
          }
        ]
      }
  ]
}
```


5) Create a file called numPeers that contains only the number of devices that will be on the network, for example for 4 devices the file would **only** contain the number 4 in it.
6) Create a conjunct mapping file called conjunct_mapping.my the example

contents are as follows:

```
#conjunct_name,owner_process,expression
A,1,x1 == 0.0
!A,1,x1 != 0.0
B,2,x2 == 0.0
!B,2,x2 != 0.0
C,3,x3 == 1.0
!C,3,x3 != 1.0
D,4,x4 == 1.0
!D,4,x4 != 1.0
```

7) Load all of the files on the device into the monitorInit found on the Android devices sdcard folder
8) Launch the Android application

5 Using the Android Application

Upon transferring files from the PC to the Android device, launch the application. When opening at the top you will be shown a list of other devices on the network that have detected each other and below a list of different options on how to trigger the device variables. At the top of each view the device will display the current variable and if the formula is satisfied or not.

5.1 Vision

This view will open up the device's camera and start tracking objects that are a circle. When the device finds a circle every other device on the network will be made aware of a state change.

5.2 NFC

This view will use the device's NFC reader and chips. Using a predetermined list of NFC uuid's the device can respond positively or negatively depending on if the tag was expected.

5.3 Cube

This view generates a 3d model that rotates as the device is turned up. When the device reaches a 90 degree vertical the model will turn green and the variable on the device will update.

6 FAQ

1) The app is crashing or complaining of missing files.
   Make sure the automaton.json, initial_state.json, numPeers, and conjunct_mapping.my are all loaded into the app's folder.

2) My device doesn't see any others that are running the app.
   Ensure all the devices are on the same network. Some networks partition devices off in different subnets and if that is the case try setting one device to a hotspot and have all of them join that.