A Runtime Verification Framework for Access Control

Minh Van Nguyen
nguyenminh2@gmail.com

01 February 2008

1 Why runtime verification?

Runtime verification [2] is a formal verification technique that has recently become a tool of choice to complement other techniques such as model checking and unit testing. In a nutshell, we deploy a monitor M to dynamically monitor some system S while the latter is in operation (see Figure 1). Based upon a recent snapshot of the input and output of S, M decides whether or not S behaves according to its specifications. An alarm is raised if M happens to detect some behaviour that deviates from specifications.

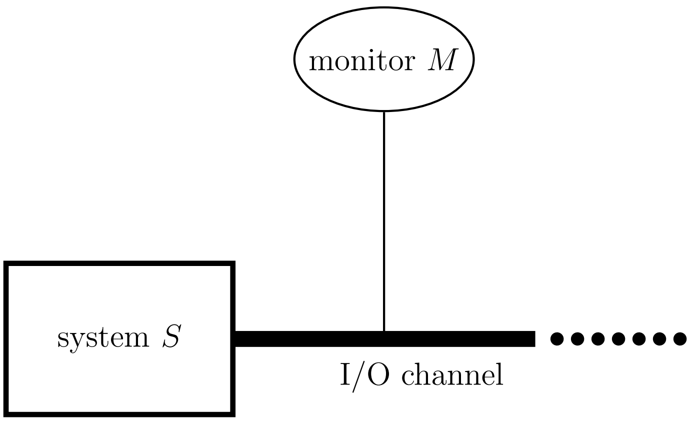

Figure 1: System monitoring during operation.

Other formal verification techniques suffer one common limitation; namely, that they attempt to verify the correctness of a system based upon its formal specifications or a simplified version of the specifications. On the other hand, runtime verification attempts to remedy this defect by passively monitoring a system of interest S as it is running within its operating environment. For various systems, there are many advantages in using a passive monitor. Since S is being monitored for conformance to specifications during its operation, this increases our confidence in the implementation of S, in addition to the confidence resulting from verifying the formal specifications. Verifying the correctness of the formal specifications of S shows us that in theory S behaves according to our conception of how it is to behave. However, in practice, certain information is only available during runtime. In some cases, the behaviours of S depend on its operating environment, so that S must not be considered as an entity separate from its environment. Furthermore, a crucial advantage of runtime verification is that the technique can be used to monitor critical systems, such as to ensure that a computer system grants access only to those with access privilege.

2 Automatic generation of monitors

The utility `ltl2mon` is a program for system supervision based on runtime verification. A description of a 3-valued semantics monitor of real-time properties is contained in [1]. The current project aims to realize this description by producing a working implementation in the form of the `ltl2mon` monitor. Listing 1 shows the usage information for `ltl2mon`. The utility translates a linear temporal logic (LTL) formula to a runtime verification monitor. This implementation uses the Java wrapper LTL2BA4J [5] by Eric Bodden to translate an LTL formula to the dot format representation of the corresponding Büchi automaton. Bodden's Java library is a wrapper around the tool LTL2BA [4] by Denis Oddoux and Paul Gastin. The dot format representation is then parsed as a directed graph using the JGraphT [3] library, followed by an emptiness check on the directed graph representation.

Figure 2 shows two screenshots of a sample execution of `ltl2mon` with input LTL formula $a\mathrm{U}b$; i.e. a until b. The left screenshot shows a trace that satisfies this formula; the right screenshot is an instance of a trace that violates the formula. At present, there is little support for a configuration file for `ltl2mon`. It is recommended that your configuration file, if one exists, be named "ltl2monrc", although this is not a strict requirement. Each entry in your configuration file must follow the format:

```
# document your entry
<entryLabel>=''<entryValue>''
```

where the hash symbol # is used for one-line comments.

```
Usage: java ltl2mon [-chis] [[-f formula] | [-F file]] [-o file]

    c          : configuration file for ltl2mon
    f formula  : an LTL formula enclosed within quotation marks
    F file     : reads an LTL formula from the specified file
    h          : prints this help message
    i          : output whether or not the initial state is empty
    o file     : output dot format of Buchi automaton to the specified
                 file
    s          : output the dot format of Buchi automaton to the terminal
                 screen. You can't use the s switch with the h or i switch

    mandatory argument: exactly one of f or F must be present
```

Listing 1: Usage information for `ltl2mon`.

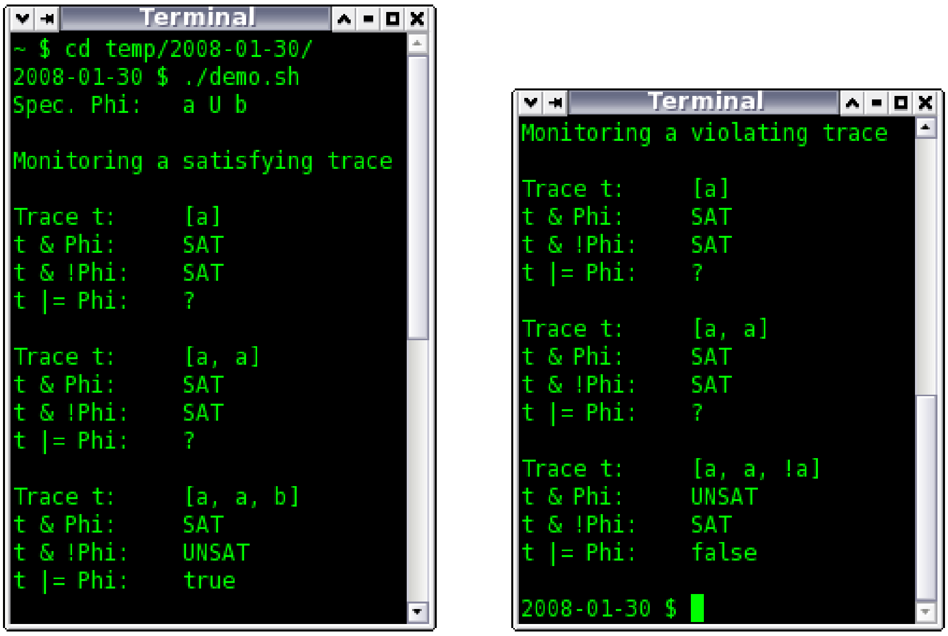

Figure 2: Sample run of `ltl2mon` corresponding to LTL formula aUb.

References

[1] Bauer, Andreas, Martin Leucker and Christian Schallhart. "Monitoring of Real-Time Properties" in S Arun-Kumar and N Garg (eds.). *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science* (LNCS 4337). Springer-Verlag, 2006, pp.260–272.

[2] Colin, S and L Mariani. "Run-Time Verification", chapter 18 in M Broy, B Jonsson, J-P Katoen, M Leucker and A Pretschner (eds.). *Model-based Testing of Reactive Systems*. LNCS 3472, Springer, 2005.

[3] JGraphT - Java graph library
`http://jgrapht.sourceforge.net/`
Viewed 29 January 2008

[4] LTL2BA: fast translation from LTL formulae to Büchi automata
`http://www.lsv.ens-cachan.fr/ gastin/ltl2ba/`
Viewed 29 January 2008

[5] LTL2BA4J - Java bridge to ltl2ba
`http://www.sable.mcgill.ca/ ebodde/rv//ltl2ba4j/`
Viewed 29 January 2008