Runtime Verification of Real-Time Systems

Limitations of classic (static) approaches:
• Number of reachable states too large for testing
• Potential blow-up when automatically exploring the system’s state-space (e.g., model-checking)
• Limited automation in machine assisted proof construction tools (e.g., SMT solvers, proof-assistants)
• Difficulties in capturing data expected to be available only at runtime (need for abstraction leads to lack of precision)

Limitations of existing Runtime Verification solutions:
• Vast majority of tools developed for non-real-time applications;
• In most cases, it is difficult to capture extra-functional properties:
  • either no support at all; or
  • via complex specifications that are not accessible for the non-expert or the typical industrial practitioner
• Lack of a specification language that is user friendly, and that allows to capture distinct classes of timing properties

The REVERT Framework

1) A new specification language:
• Intuitive, easy to use domain specification language
• Capture changes in the system via guarded state-machine transitions between nodes (monitor states)
• Functional behavior as extended regular expressions
• Support for associating events with job specifications
• Three classes of timing constraints relevant for real-time systems: time, duration and jitter
  • Timing constraint on sequences of events,
  • Execution time of a job,
  • Jitter on time and duration
• Local variables and local code (e.g., for monitor initialization, calling counter-measure actions, etc)

2) A new monitor generation process:
1. REVERT specifications are parsed into intermediate data-structure;
2. Generation of the corresponding automata (via combination of intermediate types of finite automata)
3. Translation of the generated timed state-machine into XML format

Generated Monitor XML
• Generate the corresponding code from the XML in order to make the monitor execution online, together with the target monitored system;
• Use the monitor to verify traces offline and therefore detect unexpected/unknown behaviors
• Overall, improve the reliability and trust on the target system

Concluding Remarks
• New specification language for runtime verification of RTSs
• Novel method to generate timed finite state machines that avoids state blowup in run-time
• Implemented the framework as a tool-chain