Working Group 3
Challenging Computational Domains

Chair: Keiko Nakata
Vice-chair: Cesar Sanchez

April 8, 2015

Outline

Section 1

MoU and WG3

Main Objective of the Action (from the MoU)

Memorandum of Understanding

The main objective of the Action is to consolidate a network of runtime verification experts and practitioners in application domains, so that they jointly find new principles for reliable system engineering using monitoring as a building block.

Main Objective of the Action (from the MoU)

Memorandum of Understanding

The main objective of the Action is to consolidate a network of runtime verification experts and practitioners in application domains, so that they jointly find new principles for reliable system engineering using monitoring as a building block.

A. Abstract and Keywords

The main goal is to overcome the fragmentation of RV research by: (1) the design of common input formats for tool cooperation and comparison; (2) the evaluation of different tools, building a growing sets benchmarks and running tool competitions; and (3) by designing a road-map and grand challenges extracted from application domains.

Main Objective of the Action (from the MoU) (2)

C.2 Objectives

To achieve the overall aim, the main objectives are:

- the development of a common infrastructure that enables the development of a collection of runtime verification problems and benchmarks for the comparison of algorithms and tools, and to increase their collaboration
- the development and sharing of current challenges in runtime verification and monitoring
- the development of an interaction between the runtime verification community of experts at large with practitioners from application domains that could benefit from this technology, and influence its developments
- education of young researchers and potential users of monitoring technologies
- coordination of European research on monitoring, runtime verification and its applications

MoU (3)

C.4 Potential impact of the Action

This Action will coordinate the European efforts in the field of runtime verification and applications.

[...]

Concrete outcomes of the Action will include (1) a taxonomy of problems and techniques, and a taxonomy of existing tools, (2) a common family of input languages for describing problems and solutions, (3) a collection of benchmarks that allows to compare the different tools, (4) a set of challenges for the applicability of runtime verification to important areas of application.

MoU (4). D.1 Scientific Focus

Challenging computation domains

Runtime verification has been studied in the context of several domains including embedded systems, hardware, distributed systems, and unreliable/approximate domains. To date such studies are fragmented and scattered at best. Through the Action, the existing work will be reviewed and a roadmap for future work will be drawn up.

Section 2

WG3. Aim and Objectives

WG3 Aim

To study novel and challenging computational domains for runtime verification and monitoring that result from the study of other application areas than programming languages.

WG3 Objectives

The objectives of this Working Group will be to identify the challenges for monitoring in the following application domains:

- Distributed systems, where the timing of observations may vary widely in a non- synchronized manner.
- Embedded systems, where the resources of the monitor are constrained.
- Hardware, where the timing must be precise and the monitor must operate non disruptively.
- Unreliable domains and approximated domains, where either the system is not reliable, or aggregation or sampling is necessary due to large amounts of data.

These areas involve expertise from more than one domain and have a much higher chance of success if attacked cooperatively.

(From MoU: **D.2 Scientific work plan methods and means**)

WG3 Output

The concrete outputs of WG3:

- First, a series of documents will be worked out giving a roadmap for the application of runtime verification techniques to the areas listed above, identifying connections with established work in the respective sub-areas of computer science, and challenges and opportunities.
- Second, a concrete case study will performed, in which a runtime verification solution for multicore systems will be developed using dedicated monitoring hardware based on FPGAs to show the feasibility and general applicability of runtime verification techniques.

(From MoU: **D.2 Scientific work plan methods and means**)

WG3 Milestones

- M1 (end of Year 1): First Annual Report of the Action. Proposals on taxonomy, infrastructure, challenges. [...]
- M2 (end of Year 2): Second Annual Report of the Action. First document with challenges. [...]
- M3 (end of Year 3): Third Annual Report of the Action. First Summer school. [...]
- M4 (end of Year 4): Final report of the Action. Published surveys. Second Summer School. [...] Report on case studies using monitoring hardware and application domains, including medical devices.

(From MoU: **E.1 Coordination and organisation**)

Section 3

Organization

Lines/ Domains

- Distributed systems: Goal: how to monitor global properties from (local) observations.
  - Theory: study what is monitorable. How much communication? Extra communication?
  - Theroy: develop a general framework (logics, algorithsm).
  - Practice: identify practical case studies.
  - Practice: itentify useful fragments of the framework.
- Embedded systems, where resources are constrained.
  - Theory: study how to incorporate resources in monitors.
  - How to instrument an embedded system.
  - When to sampling a continuous signal (hybrid system).
- Hardware: two themes ("monitoring hardware" and "hardware for monitoring"). Issues:
  - Precise timing.
  - Non-intrusive monitoring.
  - How much hardware is needed?
- Unreliable domains and approximated domains.
  - E.g: monitoring dynamic large systems by sampling.
  - E.g: what if the observation is imperfect (value, timing)?
  - How to characterize imprecision?

Line: Distributed Systems

- Distributed systems: Goal: how to monitor global properties from (local) observations.
  - Theory: study what is monitorable. How much communication? Extra communication?
  - Theroy: develop a general framework (logics, algorithsm).
  - Practice: identify practical case studies.
  - Practice: itentify useful fragments of the framework.

Line: Embedded Systems

(CPSs)

- Embedded systems, where resources are constrained.
  - Theory: study how to incorporate resources in monitors.
  - How to instrument an embedded system.
  - When to sampling a continuous signal (hybrid system).

Lines: Hardware

- Hardware: two themes ("monitoring hardware" and "hardware for monitoring"). Issues:
  - Precise timing.
  - Non-intrusive monitoring.
  - How much hardware is needed?

Lines: Unreliable domain

- Unreliable domains and approximated monitoring.
  - E.g: monitoring dynamic or large systems by sampling.
  - E.g: what if the observation is imperfect (value, timing)?
  - How to characterize imprecision?

Lines

Perhaps a common theme:

Static analysis/information about the system is always exploitable for monitoring (at least for efficiency)

but

but it (static information) maybe crucial for the monitorability of challenging domains

One Organizational Option

Create task-forces for each line to cooperatively create a quick survey on the concrete state-of-the-art, including:

- publications (from action members and others)
- tools, intended applications
- current limitations
- related areas of CS, related research groups
- (speculate) on unknow results

These surveys can be the basis for:

- discussions,
- shaping challenging problems
- identifying potential applications,
- itentifying collaborations

Embedded Debugging

Topics for discussion

When an embedded software is tested in the deployed platform, the engineers cannot use an interactive debugger.
An alternative is to produce logs to later analyze them.
However,

- the modifications in the program must be minimal (to prevent changes in the timings)
- the inspection of traces is tedious
- the logging maybe unfeasible because large storages are typically not available.

Can we offer the debugging engineer a language to express interesting error trace descriptions (an RV language) which must then be used to modify the code?
Instrumented code often suffers a slowdown of orders of magnitude. The slowdown may be significantly alleviated through static analysis.

Concurrency

- Coordination (wait/notify) is least explored area.
- Which properties can you observe locally?
- Prove statically that some global property follows from a bunch of local properties (assumptions), and you can monitor those local properties at runtime.
- How do you learn a context-specific behavior of a component in a disributed system?
- Determinism can be more abstract and expressive than absence of deadlocks.
- Can we use rely-guarantee techniques at runtime to make monitoring more composition?

Challenges for distributed systems

- Modularity, partial static analysis, updates on components.
- Contracts: with run-time checks we can have more permissive contracts between components.
- Analyze contracts incrementally.
- Need to constrain design of systems to provide guarantees.
- Current systems produce much data/logs but still forencics cannot find root cause of some failures.
- Use static analysis to enhance effectiveness of monitors.
- Need specific assumptions on system design/implementation for verification.
- Each component has a monitor but monitor may not be reliable. How can we identify misbehaving processes?