# Working Group 3
# **Challenging Computational Domains**

Chair: Keiko Nakata
Vice-chair: Cesar Sanchez

April 6, 2015

# Outline

Section 1

# Aim and Objectives

# 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

# Objectives

1. Enrich the RV taxonomy identified in Working Group 1 to classify tools and problems.
2. Design common representation formats for inputs to monitoring tools organised in terms of the taxonomy.
3. Implement interfaces, using these common representation formats, to enable different RV tools to be attached to a variety of programs and systems
4. Create and maintain a collection of common benchmarks using the common representation formats
5. Coordinate a regular tool competition

# 1. Enrich the taxonomy

- The idea is to have a taxonomy of RV techniques
- This is a focus of WG1 but obviously effects this WG a lot

- From the perspective of this WG it is important that it allows us to separate techniques that do not easily interoperate

- The ‘enrichment’ should involve input from the developers of tools and benchmarks so that the classification reflects reality

# 2. Design common representation formats

- Will not be a one-size-fits-all approach, many aspects vary widely. For example, but not limited to, the following:
  - Definition of events and their data contents
  - Notion of time in traces
  - Dependence between a trace and the system that generates the trace; both for online instrumentation and trace-recording
  - Input formats for offline tools i.e. file format of traces

- Output will be a small family of suitable languages, following the taxonomy
  - Important that taxonomy reflects the varying aspects
  - Usability/readability concerns
  - Use an existing language or introduce a new one?
  - Explore translations between existing and selected languages
  - Consider theoretic expressiveness?

# 2. Design common representation formats (continue)

- Further considerations.... how to deal with
  - Programming language specific concepts
  - Embedded specification languages (internal DSL)
  - Coupling with instrumentation techniques
  - Anything Else?

- Need to have enough formats to separate classes but not too many that there is one tool per class!

# 3. Implement interfaces

- Idea is to separate monitoring systems and the systems they monitor via common interfaces

- Currently RV tools are generally either
  1. offline and system agnostic... but not useful for online
  2. outline requiring events to be submitted manually
  3. inline (online) and instrumentation-technique dependent
- Observations
  - If case 1 are incremental then they can be used outline
  - Inline tools can typically be separated into instrumentation+outline
- Therefore if we
  1. Create an interface for outline monitoring
  2. Define one (or more) programming language specific instrumentation technique(s) to use this interface
  3. Extend monitoring techniques to instantiate this interface
- Then RV tools can be programming language agnostic

# 3. Implement interfaces (continued)

- There are some issues to consider when implementing interfaces
  - Interfaces should be modulo the taxonomy
  - Some tools depend on language-dependent constructs internally. Should these must be captured by the interface?
    - Identity i.e. semantic (equals) vs reference (==) in Java
    - Garbage collection in Java
  - Data parameters as runtime objects i.e. passing a collection object to a monitor and then calling the `size` method from inside the monitor.
  - Anything Else?

# 4. Collect benchmarks

- Idea: to collect benchmarks that can be used by the community to compare techniques and encourage/focus research
- Bonus: can be used by the competition

- The taxonomy and common representation formats would be used
- Need to also record meta-information i.e. description, references, comments

# 4. Collect benchmarks (continued)

- The aim of this WG mentions **challenges**
- What does it mean for a benchmark to be challenging?
  - Require efficient monitors? i.e. long traces, lots of data
  - Require responsive monitors? i.e. impose maximum per-event overhead
  - Require expressive languages? i.e. complex concepts
  - Require usable tools?
  - Anything Else?

# 5. Coordinate competition

- Already in progress.
- But still finding its feet.
- Should discuss what best supports the community
  - Promote further research
  - Encourage collaboration and cooperation
  - Consider emerging subfields? (e.g. distributed monitoring)
  - Consider the usability aspect?
  - Should we evaluate tools or monitoring techniques; if the latter we should separate instrumentation and trace parsing methods
  - Anything Else?

Section 2

# Current Status

# CRV14

- First competition
- Very successful
- Running again this year (15)

- There is an upcoming journal paper giving lots of details

# CRV14 Benchmarks

- Domains
  - Banking
  - Concurrency
  - Aerospace
  - Programming (Java API)
  - Databases
  - Networking
  - And others...
- Languages
  - Temporal logics: LTL, MFOTL, LTL + FO theories, LOLA
  - Automata based: DATEs (enriched FSAs), QEA, prm4j
  - Rule-based (LogFire)
  - JavaMOP can use LTL, ERA, FSM, CFG, SRS
- Input formats
  - Traces - organisers defined CSV, XML and JSON formats
  - Program instrumentation - AspectJ, Java reflection, manual

# CRV14 Tools

- Offline monitoring
  - RiTHM2
  - Monpoly
  - STePr
  - MarQ
- Online monitoring of Java programs
  - Larva
  - JUnitRV-MMT
  - JavaMOP
  - MarQ
- Online monitoring of C programs
  - RiTHM
  - E-ACSL
  - RTC

- Others entered but did not make it to the evaluation stage (6 dropped out)

# CRV14 Benchmarks, other observations

- Lack of common representations made exporting properties difficult.
  - It was often the case that properties were not specified equivalently in different languages
  - i.e. usually a small set of corner cases are valid in one and invalid in another
  - No easy way to check equivalence... does it matter?

- Almost all benchmarks dealt with *data* in some way
  - Trace slicing, first-order temporal logic
  - Treating runtime objects as data (i.e. calling methods on them)

- Lack of clear notion of instrumentation and input formats for online monitoring meant that it was difficult to tell if the same events were being monitored

# RV tools in general

- There does not exist a recent survey of RV tools (correct if wrong), or even a list (beyond those entering the competition)
- The 2004 "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools" by Delgado et al. is the only tool-focussed survey to date and is over 10 years out of date
- Anything Else?

# Benchmarking in RV Research

- Very little focus on developing benchmarks
- The DaCapo benchmarks + some Java API properties are commonly used
  - These are very restricted to a single domain and quite low-level
  - The combination of properties+programs are all quite similar in exhibited behaviour
- The recent SMock platform is a step towards better benchmarking systems. But more can be done to reflect different domains.
- Anything Else?

# Interoperability in RV Research

- By interoperability here we mean a focus on making RV tools work together, either directly or via standardisation of inputs
- It does not seem that this has been a key focus of any research efforts (correct if wrong)
- There are some unifying factors
  - Finite-trace LTL definitions are often reused
  - Multiple systems now use the notion of trace-slicing
  - Some instrumentation techniques (AspectJ) unify systems
  - Anything Else?
- Anything Else?

Section 3

# Planning

# Support taxonomy development

- Part of Working Group 1
- Objectives 2 and 3 depend on this

- Talk to WG1 and see how we can support this
- Ensure that development takes pragmatic issues into account

# Create benchmark/tool repository

- Can start without taxonomy... but will need to be able to extend to support this

- Idea: Central website that stores benchmarks and tools
- What is a benchmark in this respect?
  - Informal and formal descriptions
  - Traces and/or programs
  - ? specifications in multiple RV tools

- Collecting/submitting benchmarks should be independent of competition i.e. for the benefit of the whole community
- Encourage input from WG3 and WG4

- Consider adapting benchmarks from other program verification communities?

# Support competition

- Best way to support competition is to enter it
- Might still be possible to enter this year... but it is already well under way. Talk to the chairs. (This is at time of initial distribution; by the Malta meeting it is likely it will be too late)
- Consider setting up permanent website (potentially same as benchmark repository)?
- Discuss future iterations

# CRV15 benchmarks

- This year's competition is trying to create a repository of all artefacts. See here

  `https://forge.imag.fr/plugins/mediawiki/wiki/crv15/index.php/Main_Page`

- This year there will also be prizes for 'best' benchmarks, which will hopefully involve the community

# Timeline and What's next

- Official Milestone(s) for WG2:
  - End of 2nd year: First version of the electronic infrastructure and benchmarks
  - End of 3rd year: Second version of the infrastructure and benchmark, infrastructure for competitions
  - End of 4th year: Final electronic infrastructure and competition.
- One of the meeting's goal: define a more detailed timeline