Bertinoro Seminar on Distributed Runtime Verification
Final program

(May 17, 2016)

May 16–20, 2016, Bertinoro, Italy

The slides of some the presentations are available here: `http://www.labri.fr/perso/travers/DRV2016/talks/index.html`

Tuesday

07:30-08:30 Breakfast

08:45-09:00 Forewords

09:00-10:00 Tutorial 1: Fault-tolerant Distributed Computability and Topology (`slides`)

Maurice Herlihy, Brown University

Abstract:

10:00-10:30 Coffee break

10:30-12:30 Tutorial 1 (ctd.): Fault-tolerant Distributed Computability and Topology (`slides`)

Maurice Herlihy, Brown University

Abstract:

12:30-13:30 Lunch

14:00-15:00 Tutorial 3: Crash Course on Linear-Time Temporal Logic

Rotem Oshman, Tel Aviv University

Abstract: In this short tutorial we will review the basics of linear-time logic and the automata-theoretic approach to model-checking.

Linear-time temporal logic (LTL) is a language for specifying correctness properties of programs: the program we want to verify is modeled as an automaton (finite or infinite-state), and the executions of the program are all the words generated by the automaton; in LTL model-checking, our goal is to verify that all executions belong to the set of "correct" executions, specified as a formula in LTL.

One approach to LTL model-checking, which is particularly suitable for runtime verification, is to convert the LTL formula we wish to check into an automaton that accepts exactly those executions that do *not* satisfy the LTL formula, and then check whether the program automaton generates any executions that are accepted by it. In the tutorial we will define the syntax and semantics of LTL, show how an LTL formula is converted to an automaton, and how to check language inclusion to verify that the program has no bad executions.

15:00-15:30 Coffee break

15:30-16:30 Tutorial 4: Overview of Distributed Wait-free RV

Sergio Rajsbaum, UNAM, Mexico

Abstract:

16:30-17:30 Discussion 1
20:00-22:00 Dinner

Wednesday

07:30-08:45 Breakfast
09:00-10:00 Tutorial 5: Formal Methods for the Verification of Distributed Algorithms (`slides`)
Paul Gastin, CNRS, France
Abstract:
10:00-10:30 Coffee break
10:30-11:30 Tutorial 6: Algorithms and lower bounds on the number of opinions in wait-free RV (`slides`)
Corentin Travers, LaBRI Bordeaux, France
Abstract:
11:30-12:30 Tutorial 7: Logic Approach to Distributed RV (`slides`)
Borzoo Bonakdarpour, McMaster University, Canada
Abstract:
12:30-13:30 Lunch
14:00-15:00 Tutorial 8: Synthesis and Control: a Distributed Perspective (`slides`)
Anca Muscholl, Technische Universität München, Germany & Université de Bordeaux, France
Abstract:
15:00-15:30 Coffee break
15:30-16:00 Trustworthy Self-Assembly: A Use-Case For Distributed Runtime Verification
John Rushby, SRI International, USA
Abstract: Systems necessarily interact with their neighbors through the effect each has on the environment of the others (so-called stigmergic interactions), particularly those involving the "plant." Unmanaged interactions can be deleterious and it is better if systems are open to interactions, so these can be coordinated. Furthermore, systems should be adaptive so they can adjust their behavior to their circumstances. We then have systems of systems, which can exhibit desired and positive behavior, but it is also possible that normal or adjusted (or faulty) behavior by one component system adversely affects others, leading to emergent misbehavior. Notice that not all interactions can be planned ahead of time (because of stigmergic effects) so systems need to self integrate at runtime. But we want the resulting integrations to guarantee properties such as safety, so self-integration needs to construct or update an assurance case. One way accomplish all this is for the self-integrating systems to exchange models of their architecture, behavior, and local assurance case: this is (safety) models at runtime (M@RT and SM@RT). The models are used to guide adaptation: for example, one component system may synthesize a runtime monitor or wrapper for its interaction with another to eliminate some class of behaviors that violate its assumptions. Some system-level assurance claims can be ensured by the composition of local claims, but others may be truly emergent (i.e., ontologically distinct from all component properties) and these are candidates for distributed runtime verification. In summary, trustworthy self-integration requires autonomous adaptation, synthesis, and verification at integration time, and thismeans that embedded automated deduction and distributed runtime verification will be the engines of integration.
16:00-16:30 Runtime Enforcement of Privacy Policies for Social Networks (`slides`)
Gerardo Schneider, University of Gothenburg, Sweden
Abstract:

16:30-17:30 Discussion 2
20:00-22:00 Dinner

Thursday

07:30-08:45 Breakfast
09:00-10:00 Tutorial 9: Runtime Verification of Multiagent Systems (`slides`)
Viviana Mascardi, University Genova, Italy
Abstract: Multiagent systems are distributed by definition, so their verification (usually performed "a priori", but sometimes also at runtime) is a problem which falls in the scope of the workshop. I may introduce multiagent systems first, then briefly review the literature on their (runtime) verification, and briefly present a recent proposal of ours, all at a high level of detail.
10:00-10:30 Coffee break
10:30-11:30 Tutorial 10: Wait-free RV and determinism (`slides`)
Pierre Fraigniaud, CNRS, France
Abstract:
11:30-12:00 Decentralized Monitoring of Artifact-Centric Systems
Sylvain Hallé, University of Quebec
Abstract:
12:00-12:30 On the Decentralised Monitoring of Systems with a Global Clock (`slides`)
Ylies Falcone, CNRS, France
Abstract: Users wanting to monitor distributed systems often prefer to abstract away the architecture of the system by directly specifying correctness properties on the global system behaviour. To support this abstraction, a compilation of the properties would not only involve the typical choice of monitoring algorithm, but also the organisation of submonitors across the component network. Approaches, in the context of LTL properties over systems with a global clock, include the so-called orchestration, migration, and choreography approaches. In the orchestration approach, a central monitor receives the events from all subsystems. In the migration approach, LTL formulae transfer themselves across subsystems to gather local information. In choreography, monitors are organised as a tree across the system, and each child feeds intermediate results to its parent.
In this talk, we will review, formalise, and empirically compare the approaches using several monitoring metrics tailored to decentralised monitoring. We will also discuss open perspectives and remaining challenges of decentralised monitoring.
12:30-13:30 Lunch
14:00-14:30 Verification of population protocols (`slides`)
Javier Esparza, Technical University of Munich
Abstract: Population protocols (Angluin et al., 2004) are a formal model of sensor networks consisting of identical mobile devices. When two devices come into the range of each other, they interact and change their states. Computations are infinite sequences of pairwise interactions where the interacting processes are picked by a fair scheduler.
A population protocol is well-specified if for every initial configuration C of devices, and every computation starting at C, all devices eventually agree on a consensus value depending only on C. If a protocol is well-specified, then it is said to compute the predicate that assigns to each initial configuration its consensus value.

While the predicates computable by well-specified protocols have been extensively studied, the two basic verification problems remain open: is a given protocol well-specified? Does a protocol compute a given predicate? We study these problems, and also discuss the more general question of checking if a population protocol with a random scheduler satisfies a PLTL property.

14:30- 15:00 On parametrized Verification (`slides`)
Arnaud Sangnier, University Paris Diderot
Abstract:

15:00-15:30 Coffee break

15:30-16:00 Importance of Static Knowledge (`slides`)
Cesar Sanchez, IMDEA Software Institute, Spain
Abstract: Importance of assumptions about the system or abstractions/models on the ability to monitor or impose at runtime.

16:00-16:30 Runtime Monitoring of Real-time Tasks in Distributed Architectures to Increase Parallelism
Matthieu Roy, LAAS-CNRS, France
Abstract: quite a general talk, not too technical yet with theory and practical flavors.

16:30-17:00 Towards Combined Static and Runtime Verification of Distributed Software (`slides`)
Wolfgang Ahrendt, Chalmers University of Technology, Sweeden
Abstract:

17:00-18:30 Discussion 3

20:00-22:00 Dinner

Friday

07:30-08:45 Breakfast

9:00-9:30 Runtime Reflection
Martin Leucker, University of Luebeck, Germany
Abstract: Using Runtime Verfication Diagnosis from First Principles and Controller Synthesis to Build Self-organizing Systems - a sketch of an idea.

9:30-10:00 Liveness and universal reachability for parametrised asynchronous shared memory systems (`slides`)
Igor Walukiewicz, Technische Universität München, Germany & CNRS, France
Abstract: We consider the model of parametrized asynchronous shared-memory pushdown systems as introduced by Hague. In a series of recent papers it has been shown that reachability in this model is PSPACE-complete [Hague'11] [Esparza, Ganty, Majumdar'13] and that liveness is decidable in NEXPTIME [Ganty, Durand-Gasselin, Esparza, Majumdar'15]. We show that the liveness problem is PSPACE-complete. We also consider universal reachability problem. We show that it is decidable, and coNEXPTIME-complete. These results imply that verification of all regular properties of traces, satisfying some stuttering condition, is also decidable in coNEXPTIME.

10:00-10:20 Coffee break

10:20-10:50 A Constructive Approach for Proving Correctness of Concurrent Data Structures
Gregory Chockler, Royal Holloway, University of London
Abstract: We present a comprehensive methodology for proving correctness of concurrent data structures. Correctness is based on new abstractions connecting the implementation's concurrent behaviour with its sequential invariants. We demonstrate effectiveness of our

methodology by using it to show safety properties of several nontrivial implementations of concurrent data types. Our ongoing work explores integration with formal frameworks to enable automated correctness verification, testing, and synthesis of concurrent implementations from sequential code.

Joint work with Kfir Lev-Ari and Idit Keidar, Technion

10:50-11:20 Consistency Criteria for Distributed Data Structures (`slides`)

Constantin Enea, IRIF, University Paris Diderot

Abstract:

11:20-11:40 Coffee break

11:40-12:10 Sensitivity of Debugging in Partial Synchronous Systems

Sandeep Kulkarni, Michigan State University

Abstract: This talk focuses on the issue of debugging while considering the gap between theory and practice of distributed system. The former typically assumes an asynchronous system whereas the latter uses ad-hoc approaches based on partial clock synchronization/timeouts. We focus on the effectiveness of debugging algorithms when the knowledge about underlying synchrony requirements in the protocol is uncertain. Using analytical model and simulations, we identify the amount of uncertainty can be reasonably handled by different algorithms.

12:10-12:40 Modal Logic & Distributed Automata (`slides`)

Fabian Reiter, IRIF, University Paris Diderot, France

Abstract: This talk focuses on the observation that basic modal logic is closely related to a very weak model of distributed computing, where the nodes are finite-state machines with constant running time. Could this be exploited for Hoare-style verification of distributed algorithms?

12:40-13:40 Lunch

13:40-14:30 Wrap-up discussion

14:30 Workshop ends