10451 Abstracts Collection
Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems
— Dagstuhl Seminar —

Klaus Havelund[1], Martin Leucker[2], Martin Sachenbacher[3], Oleg Sokolsky[4] and Brian C. Williams[5]

[1] Jet Propulsion Laboratory/Caltech, US
Klaus.Havelund@jpl.nasa.gov
[2] Universität Lübeck, DE
leucker@isp.uni-luebeck.de
[3] TU München, DE
sachenba@in.tum.de
[4] University of Pennsylvania, US
sokolsky@cis.upenn.edu
[5] Massachusetts Institute of Technology, US
williams@mit.edu

Abstract. From November 7 to 12, 2010, the Dagstuhl Seminar 10451 "Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems" was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, 35 participants presented their current research and discussed ongoing work and open problems. This document puts together abstracts of the presentations given during the seminar, and provides links to extended abstracts or full papers, if available.

Keywords. Runtime Verification, Model-based Diagnosis, Planning, Control, Autonomous Systems

10451 Executive Summary – Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems

From November 7 to 12, 2010, the Dagstuhl Seminar 10451 "Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems" was held in Schloss Dagstuhl - Leibniz Center for Informatics. During the seminar, 35 participants presented their current research and discussed ongoing work and open problems. This executive summary provides an overview of the goals and topics of the seminar.

Keywords: Runtime Verification, Model-based Diagnosis, Planning, Control, Autonomous Systems

Dagstuhl Seminar Proceedings 10451
Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems
http://drops.dagstuhl.de/opus/volltexte/2011/2948

Joint work of: Havelund, Klaus; Leucker, Martin, Sachenbacher, Martin; Sokolsky, Oleg; Williams, Brian C.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2011/2947

The trace server - bridging the gap between model checking and run-time verification

Cyrille Artho (AIST - Tokyo, JP)

Traces generated by a software model checker can become very large, too large to be held in main memory on the machine where the system under test is analyzed. A way out of this problem is to store trace information off-line on a separate server. This allows the model checker to fully utilize the memory on its own machine.

However, the resulting change in the architecture has much bigger ramifications. By moving the trace analysis into its own module, it is now decoupled from the state space exploration of the model checker. As a result, the model checker can be seen as only driving the state space search, with run-time verification algorithms performing the property checks on the trace server.

Keywords: Software model checking, execution trace, run-time verification

See also:
http://babelfish.arc.nasa.gov/trac/jpf/wiki/summer-projects/2010-trace-server

Monitorability of omega-regular languages

Andreas Bauer (NICTA - Canberra, AU)

A presentation on the concepts, definitions, and open problems of monitoring infinite-word languages, such as given by LTL formulae or Buechi automata.

Keywords: Monitorability, temporal logic, formal languages

Planning to Gather Information

Richard W. Dearden (University of Birmingham, GB)

Many information gathering tasks, such as sensor placement, dialogue, or field science, can be represented as partially observable Markov decision processes (POMDPs). This representation is very useful as it explicitly includes models of noisy observations and allows reasoning over information states. However, these problems are frequently too large to slove with standard POMDP techniques. In this talk we briefly examine how the representation can be used and describe a variety of approaches to solving these problems approximately.

Keywords: Planning, sensor placement, information gathering, uncertainty

Innovation in Mission Control: Challenges, Experience, Questions

Alessandro Donati (ESA / ESOC - Darmstadt, DE)

Mature Innovative Technologies in Space Mission Operations are functionally and economically instrumental for flying future Missions. The first part of the presentation is about the challenges and approach followed to introduce innovation in an operational environment, followed by an update of recent applications supporting planning, scheduling, diagnostic and data management. Third and central part is focusing on a list of open questions covering aspects like space domain constraints, quality, massive historical data and complexity. The questions address all the audience of the workshop.

Keywords: Space mission operation, diagnostic monitoring, planning, scheduling, autonomy, rover

Coordination Logic

Bernd Finkbeiner (Universität des Saarlandes, DE)

Coordination Logic (CL) is a new temporal logic that reasons about the interplay between behavior and informedness in distributed systems. CL provides a logical representation for the distributed realizability problem and extends the game-based temporal logics, including the alternating-time temporal logics, strategy logic, and game logic, with quantification over strategies under incomplete information. We show that the structure in CL that results from the nesting of the quantifiers is sufficient to guarantee the decidability of the logic and at the same time general enough to subsume and extend all previously known decidable cases of the distributed realizability problem.

Joint work of: Bernd Finkbeiner, Sven Schewe

Diagnosis of Discrete Event Systems by SAT

Alban Grastien (NICTA - Canberra, AU)

This talk is about the use of SAT techniques to solve problems of diagnosis of discrete event systems. The talk show how the diagnosis problem can be translated in a sequence of diagnosis questions and how these questions can be translated to a SAT problem. It also show some experimental results and some open problems.

Keywords: Diagnosis, DES, SAT

Should my Monitoring DSL be External or Internal?

Klaus Havelund (Jet Propulsion Laboratory/Caltech - Pasadena, US)

The Runtime Verification literature has produced a stream of monitoring logics. These logics support the expression of properties over execution traces, and include state machines, regular expressions, temporal logics and grammars, to mention the most common. These logics/languages can be considered as *Domain Specific Languages (DSLs)*, to use a popular terminology. They are usually designed as *external* DSLs, meaning that they stand alone without much integration with an underlying implementation language. At best they sit on top of such a programming language, allowing one to make calls to it. We have made several experiments with such externals monitoring DSLs. They usually require a serious effort to implement, making it hard to extend them with new features. They are usually also limited in expressiveness. We shall illustrate how one can define *internal* monitoring DSLs in the Scala programming language, with much less effort and with much more powerful DSLs as result.

Keywords: Runtime verification, state machines, temporal logic, external DSL, internal DSL, Scala

Joint work of: Klaus Havelund, Howard Barringer

Model-Programmed Diagnosis and Control

Michael W. Hofbaur (UMIT - Hall in Tirol, AT)

Model-based approaches for systems analysis, control design and control verification are nowadays standard in control engineering. A control engineer builds, or better crafts, a model of the artifact that is then used for simulation, control law synthesis and verification. Control synthesis is often done in a human-guided computer-aided control design process that builds upon the rich experience of the control engineer. This human craftsmanship within the design process often provides the sophisticated performance of the resulting control law. However, it also detaches the resulting controller that is then given in its parameterization and executional form from the original model and control-design specification. Contrarily, model-based methods in AI often use a problem-specific solver to directly operate on the model that describes the artifact and its operating environment and the operational goal. Thus, a controller utilizes the model directly to infer the appropriate control actions.

In mobile robotics one would find both approaches. AI based control schemes typically operate at the supervisory control layer, e.g. for task- and path-planning. Classic control schemes, on the other hand, are used within the lower-level control

layers that command the individual mechatronic components of the robot. The stringent timing constraints of the lower-level control tasks in robotic artifacts is often used to justify this separation between the 'intelligent' part of the controller and the straight-forward computational part that executes a pre-defined control sequence (e.g. a dynamic filter) for the low-level control task.

A flexible, robust and fault-tolerant operation of a robot that considers operational and fault modes within the robot's mechatronic components, however, stresses such an approach for the low-level controllers as it is difficult to precompute the appropriate control laws for all possible operational and fault scenarios within the robot. It is thus desirable to include reasoning and AI-type model-based approaches within the lower-level control layers of a robot as well.

In our talk, we will present such a control scheme for a re-configurable mobile robot. The controller incorporates efficient on-line reasoning methods that deduce the appropriate state/mode-estimators and control laws from a model, thus we use the term *model programmed* to emphasize (a) the difference from traditional model-based design schemes and (b) the fact that we program the controller by specifying a model of the artifact and utilize on-line reasoning schemes to perform control law deduction and controller-parameterization during run-time of the robot. This provides an adaptive control scheme that enables a robot to autonomously adapt to changing operational goals and overcome faults or even failures.

Keywords: Model-based reasoning, model-based control, diagnosis, mobile robotics

Verification of Hybrid Systems

Andreas Hofmann (MIT - Cambridge, US)

Hybrid system models incorporate both discrete and continuous state variables and constraints, and can be used to represent a large class of systems that perform tasks in the real world. Autonomous ground vehicles, for example, have mechanical, electrical, and control components that are best represented using hybrid models. Our approach to verification of such systems begins with expression of test cases as Qualitative State Plans (QSPs). A key feature of a QSP is that it represents the spatial and temporal plan goals and constraints in a flexible manner. This allows a control policy to take advantage of flexibility to maximize other characteristics, such as robustness or energy efficiency. For verification, it means that there is a potentially rich set of trajectories that satisfy the requirements. We use Flow Tubes to represent these trajectory sets. A Flow Tube is a complete representation of a trajectory set that supports analysis of a number of important properties, including computation of success probabilities. We compute Flow Tubes based on the intersection of constraints represented by the hybrid plant model, and by the QSP. This intersection is critical for evaluating design alternatives.

Keywords: Hybrid Systems, Verification, Flow Tubes

Joint work of: Andreas Hofmann, Brian C. Williams

Robotics, Temporal Logic and Hybrid Systems

Hadas Kress-Gazit (Cornell University, US)

These slides are a short and incomplete description of how formal methods such as model checking, synthesis and abstractions are used to transform high-level robot tasks into correct-by-construction controllers.

There are a few slides at the end that define hybrid systems and that list the questions and methods the hybrid systems community has been looking at.

Runtime Verification

Martin Leucker (Universität Lübeck, DE)

Runtime Verification is a lightweight verification technique complementing model checking and testing. In runtime verification, a finite prefix of a potentially infinite execution is checked incrementally against a given correctness property. To this end, typically a monitor is synthesized from a high-level linear-time logical specification. In this presentation, an overview on Runtime Verification is given. Moreover, a uniform approach for synthesizing monitors checking correctness properties specified in a linear-time logic is provided. The merits of the presented framework are shown by providing monitor synthesis approaches for a variety of different logics such as LTL, the linear-time mu-calculus, LTL with modulo constraints, S1S, and RLTL. Finally, we briefly sketch extensions of runtime verification such as monitor-oriented programming, and monitor-based runtime reflection and discuss their similarities and differences. We conclude by presenting a preliminary taxonomy on runtime verification approaches

Keywords: Monitor, LTL, partial run

What's Planning?

Derek Long (The University of Strathclyde - Glasgow, GB)

This talk is an overview of AI Planning and the current state of the art in terms of expressivity of planners, the performance and capabilities of some of the current systems and the approaches on which they are based.

Keywords: Planning

Universal Planning for Hybrid Domains

Daniele Magazzeni (The University of Strathclyde - Glasgow, GB)

Many realistic domains where planning is required are represented by hybrid systems, i.e., systems containing both discrete and continuous variables. The PDDL+ language allows one to model these domains, however the current tools can generally handle only planning problems on hybrid systems with linear dynamics. Moreover, when a system presents a stochastic behaviour, it can be modelled as a Markov Decision Process, and an effective policy is required. Universal planning can be used to find such a policy. In this talk, we present two techniques to perform universal planning on hybrid nonlinear systems based on model checking and classification, respectively. Both techniques make use of a discretise and validate approach, which is also discussed in the talk, to handle the continuous behaviour, which represents the key issue when dealing with hybrid nonlinear systems.

Keywords: Universal planning, hybrid nonlinear systems, policy learning

A Framework for Runtime Enforcement Systems

Somayeh Malakuti (University of Twente, NL)

Although several RE systems have been proposed in the literature, they are dedicated to specific programming languages, specific formalism and software with specific process model. This reduces the applicability of these systems to complex software and increases the effort to utilize these systems. For example, if a runtime verification system only supports centralized software, it cannot directly be applied to distributed software to verify system-wide properties of the software; and a developer must develop workarounds to overcome this shortcoming. There are several other features that increase the applicability of RE systems in practice. Examples are supporting libraries of verification, diagnosis and recovery specifications, composing reusable specifications with each other to form more complex specifications, arbitrary composition verification specifications with diagnosis and recovery specifications, etc. However, these features are neglected by the existing RE systems. We propose a reference model to build a framework for runtime enforcement systems. A prototype implementation of this framework is provided in the Compose* aspect-oriented language. In this talk I discuss this framework, Compose* and the extensions made to Compose* to facilitate the implementation of the framework.

Keywords: Runtime Enforcement, Aspect-Oriented Framework, Separation and Composition of Concerns

Model-based Goal Formation: Closing a Gap Between the Results of Acting and Deciding What to do Next

Robert A. Morris (NASA - Moffett Field, US)

Since the 1970s, Artificial Intelligence systems have sought to automate aspects of the process of planning, analysis and discovery for the enterprise of science. One of the earliest of these programs, DENDRAL, was motivated in part by a design for a robot assistant to analyze Viking mission Mars soil samples to find evidence for life.

More recently, the idea has emerged of a closed-loop architecture for an automated system that will continuously plan and execute measurements, evaluate their results, and decide the next set of measurements. Such systems could be used not only in the service of science (improving the predictive models of the measured system) but also for societal service (for example, for disaster management). To be effective, such a system must be efficient (have a rapid throughput), produce high quality results, and deal with potentially large amounts of data.

Closing the gap between data analysis and goal (hypothesis) formation, and from there to planning new actions, requires simultaneous reasoning with rich domain models, as well as models of the process itself, for example, modeling the robotic (or human-driven) device that will be executing the actions.

Forty years after DENDRAL, the exploration of space continues to be a fertile area for technology innovation in Artificial Intelligence for the automation of the continuous process of transforming collected data into plans for new data collection. This presentation will survey the component technologies, methodologies and architectures currently being developed for closed loop measurement systems for diverse applications including conducting Earth science, developing noise-minimal trajectories for rotorcraft, and safeguarding autonomous explorers.

Keywords: Planning, data analysis

Control and monitoring based on model checking of knowledge properties

Doron A. Peled (Bar-Ilan University - Ramat-Gan, IL)

An approach for controlling distributed systems based on model checking of knowledge properties is presented.

Keywords: Control, monitoring, concurrency, priorities, model-checking

Automatic Generation of Control and Diagnostics Code for Distributed Embedded Systems

Gregory Provan (Univ. College Cork, IE)

Embedded code for control and diagnostics are typically generated independently. This can lead to inconsistent code, which then results in sub-optimal performance and many false-alarms. To avoid this problem, we describe a methodology to generate all types of embedded code from a single Integrated Reference Model M. This approach enables us to guarantee consistency of embedded tools using a correct-by-design approach, where we verify M and the code-generation process (to ensure verified embedded code). In addition, it can avoid inconsistency introduced by future system modifications, since we can update all embedded code given updates to M through re-generation of the embedded code.

We describe our correct-by-design embedded code-generation methodology. We use a single centralized component library as the basis for model synthesis. We represent each component using a hybrid systems language that captures both nominal and failure modes. We employ model transformations, given a core meta-model for M, to generate target models for diagnostics and control. To illustrate our approach, we demonstrate how to construct embeddable diagnostics and control code for sustainable building management applications.

We show how a library of components for lighting and Heating, Ventilation and Air Conditioning (HVAC) can be employed to create models for systems that simultaneously control, monitor and diagnose faults for integrated lighting and HVAC systems. We are currently applying our approach to real-world buldings, which we retrofit with wireless sensor/actuator networks running the embedded code.

Sampling

Kanna Rajan (MBARI - Moss Landing, US)

We show the sampling problem in the ocean science domain and how they could relate to Diagnosis and Run-time Verification. Sampling can be both continuous and discrete. For the continuous case, the objective function is to record data of multiple parameters (e.g temperature, salinity, chlorophyll fluorescence, oxygen, nitrate etc) for either in-situ analysis or post-facto to characterize a dynamic ocean field for reconstruction on shore for subsequent robotic adaptation. In the discrete case, specific number of water samples have to be returned to shore for analysis. Such lab analysis returns information on the field which can further drive robotic exploration. Sampling continues to be a hard problem with complex constraints given spatio-temporal variability in the coastal ocean, sensor noise and point location of samples (as against a synoptic view). Poor to no communication to shore especially when the robots are underwater contributes to significant difficulty in realizing real-time monitoring and control.

Keywords: Autonomy, robotic exploration, sampling

Stream Runtime Verification Revisited

Cesar Sanchez (IMDEA Software - Madrid, ES)

We revisit the notion of Stream Runtime Verification (SRV), a specification formalism for runtime verification, first introduced in the runtime verification language Lola. Most formalisms for runtime verification consist of adaptations of existing specification languages for reactive systems, like LTL, mu-calculus - or are defined in terms of rewriting or rule based operational semantics. In contrast, SRV es based on the definition of streams - like in datalog or synchronous language - except that SRV drops the causality assumption, so values of streams can depend on present and future values, and not only on past values.

We present here an extension of SRV to multiple domains beyond purely Boolean specifications, like numerical values for computation of statistics or sets for session based verification. The second extension we present is the generalization of clock variables to deal with nested time which allows the reasoning about calls, returns and scopes. We finally present some positive and negative decidability results.

Keywords: Runtime verification, streams, reactive systems, monitoring

Aspect-Oriented Instrumentation with GCC

Justin Seyster (SUNY - Stony Brook, US)

We present the InterAspect instrumentation framework for GCC, a widely used compiler infrastructure. The addition of plug-in support in the latest release of GCC makes it an attractive platform for runtime instrumentation, as GCC plug-ins can directly add instrumentation by transforming the compiler's intermediate representation. Such transformations, however, require expert knowledge of GCC internals.

InterAspect addresses this situation by allowing instrumentation plug-ins to be developed using the familiar vocabulary of Aspect-Oriented Programming pointcuts, join points, and advice functions. InterAspect also supports powerful customized instrumentation, where specific information about each join point in a pointcut, as well as results of static analysis, can be used to customize the inserted instrumentation. We introduce the InterAspect API and present several examples that illustrate how it can be applied to useful runtime verification problems.

Joint work of: Justin Seyster, Ketan Dixit, Xiaowan Huang, Radu Grosu, Klaus Havelund, Scott A. Smolka, Scott D. Stoller, Erez Zadok

See also:
http://www.fsl.cs.sunysb.edu/interaspect

Dynamic Plan Execution Applied to Fluid Coordination of Human-Robot Teams

Julie Shah (MIT - Cambridge, US)

Collaboration between humans and robots is indispensible to our work in many high-intensity domains, ranging from surgery to space exploration. To harness the relative strengths of humans and robots, we must develop robots that seamlessly integrate with human group dynamics. Robots should preserve the essential qualities of a good human partner by robustly anticipating and adapting to other team members and avoid constraining their human partners' flexibility to act. The robot partner must be capable of reasoning quickly online, and adapting to the humans' actions in a temporally fluid way.

In this talk, I present a capability named Chaski that enables a robot to work with a human teammate under a flexible plan containing choices in the task assignment and timing of activities. Chaski generalizes the state-of-the-art in dynamic plan execution to provide a powerful framework for explicitly modeling and efficiently reasoning on temporal information for human-robot interaction. This capability is efficiently realized by an incremental algorithm that reasons on perturbations over possible futures. Chaski enables a human and robot to work together under different models of teamwork: as Equal Partners and as Leader & Assistant. I develop models that distinguish these two styles of teamwork based on the predictability of the partner.

Finally, I present recent work applying Chaski to perform multi-manipulator coordination using two Barrett Whole Arm Manipulators, and describe ongoing work to demonstrate temporally fluid human-robot teaming using the Mobile-Dexterous-Social (MDS) robot.

Toward Patient Safety in Closed-Loop Medical Device Systems

Oleg Sokolsky (University of Pennsylvania, US)

Closed-loop medical device systems hold the promise of improving patient safety by delivering continuous reliable monitoring that human caregivers cannot always provide. However, parameters of patient dynamics vary widely, making design of automatic controllers based on monitoring of vital sign streams difficult.

Full Paper:
http://repository.upenn.edu/cis_papers/427

Integrating RV into Software Engineering/MDE

Volker Stolz (United Nations University - Macau, CN)

We give a short overview of our UML-based tool for model-driven development of component-based systems. As Runtime Verification becomes more mature, it should become part of the software development activity. In this talk we raise the issue of modeling monitors, documenting them, and check their consistency with regard to the overall model. They can then be used in code generation, test case generation, or actively contribute behaviour to a specification.

Keywords: Runtime verification, model-driven development, UML, formal methods

Online Verification and Control of Hybrid Autonomous Systems in Dynamic Environments

Olaf Stursberg (Universität Kassel, DE)

Autonomous systems (AS) like mobile robots, UAVs, UGVs, or UUVs typically operate in environments which are not completely known during the design time of the control algorithms implemented on the systems. Thus, AS need to adapt to the momentary situation during operation in the sense that a dynamic model of the environment is updated based on measurements and that a strategy for goal-attaining future behavior is computed using (probabilistic) model-based predictions of the environment behavior.

To represent behavior, modular hybrid automata are used which enable to encode continuous behavior (like the motion of a vehicle) as well as different modes of operation (e.g. to distinguish between forward and backward motion, or different types of ground contact). While the dynamics of the vehicle to be controlled usually can be modeled a-priori, the environment model is (partially) identified during operation from sensor data.

Within this context, the talk sketches two techniques for controller design: The first can be termed 'model predictive verification' and aims at analyzing if a control trajectory (realizing, e.g., a path of a motion of a UGV) can be safely executed in the sense that collision with dynamic objects in the environment of the AS is avoided with a sufficiently high probability. The technique computes stochastic reachable sets for all relevant objects over a prediction horizon and determines by set intersection the probability of collision. If too high, the control trajectory has to be rejected and recomputed.

The second technique considers the problem of computing the control trajectory as the solution of an optimization problem that takes the constraints into account which arise from the environment of the AS. This method, which resembles the principle of 'model predictive control', leads to mixed-integer programming problems for which specific measures need to be considered to enable the solution in real-time.

The approaches are illustrated for the example of the motion of a UGV in an uncertain environment.

Keywords: Autonomous systems, control, reachability analysis, optimization

Pre- and Post-Deployment Runtime Verification to Guard Against Concurrency Errors

Serdar Tasiran (Koc University - Istanbul, TR)

We present Goldilocks, a Java runtime that monitors program executions and throws a DataRaceException when a data race is about to occur. This prevents racy accesses from taking place, and allows race conditions to be handled before they cause errors that may be difficult to diagnose later. The DataRaceException is a valuable debugging tool, and, if supported with reasonable computational overhead, can be an important safety feature for deployed programs. Experiments by us and others on race-aware Java runtimes indicate that the DataRaceException may be a viable mechanism to enforce the safety of executions of multithreaded Java programs.

An important benefit of DataRaceException is that executions in our runtime are guaranteed to be race free and thus sequentially consistent as per the Java Memory Model. This strong guarantee provides an easy-to-use, clean semantics to programmers, and helps to rule out many concurrency-related possibilities as the cause of errors. To support the DataRaceException, our runtime incorporates the novel Goldilocks algorithm for precise dynamic race detection. The Goldilocks algorithm is general, intuitive, and can handle different synchronization patterns uniformly.

Full Paper:
http://cacm.acm.org/magazines/2010/11/100635-goldilocks-a-race-aware-java-runtime/fulltext

Embedded Control Software Synthesis

Ufuk Topcu (CalTech - Pasadena, US)

This talk focuses on our recent work on the synthesis of control protocols for distributed embedded systems. Applications in autonomy, vehicle management, and multi-target tracking are discussed. Receding horizon temporal logic planning and compositional synthesis have been introduced to improve the scalability of synthesis.

Keywords: Embedded systems, control protocol synthesis, temporal logic planning

Diagnosability analysis

Louise Trave-Massuyes (LAAS - Toulouse, FR)

Diagnosability analysis consists of assessing at design time the level at which a system is expected to be diagnosable at runtime, i.e. which will be the precision of the outputs of a diagnoser applied to the system during operation. This analysis can be used by the designer to improve the design of the system, in particular to tune the number and placement of monitors. The Model-Based Diagnosis community has developed specific approaches for state-based systems and for event-based systems in distinct and parallel tracks.

This paper establishes a model independent framework for defining diagnosability, based on the notion of fault signature. The most classical diagnosability approaches of each track are gathered into two categories named state-based systems (SBS) and event-based systems (EBS), and the framework is instantiated into these two contexts. It is proved that the existing definitions of diagnosability can be stated as a property of the system's fault signatures, and a unified model independent definition of diagnosability is established. These results bridge existing definitions of diagnosability, and open perspectives for hybrid model based diagnosis.

Keywords: Model based diagnosis, diagnosability, event based systems, state based systems

Static and Dynamic Models for Discrete Event Dynamic Systems

Gerard Verfaillie (ONERA - Toulouse, FR)

In this talk, we will discuss how discrete event dynamic systems and associated planning, diagnosis, and verification problems can be modeled using a combination of static and dynamic models. To support such a discussion we will present the Parameterized Hybrid Constraint Automata framework we are currently working on.

Keywords: Discrete Event Dynamic Systems, Planning, Scheduling

Joint work of: Gérard Verfaillie, Cédric Pralet

Debugging (and diagnosis) at runtime to improve robustness and dependability

Franz Wotawa (TU Graz, AT)

Increasing robustness is especially important in case of autonomous mobile systems in order to achieve given mission goals even in case of hardware or software faults, or other unexpected situation occurring at runtime.

Solving the underlying challenge only at design time seems to be not very much realistic because handling all possible interactions of autonomous systems with the real world can hardly be achieved. Hence, a smarter way for providing robustness and adaptability has to be developed. In our research we focus on the use of model-based reasoning to achieve increased robustness. In particular we present a system that is able to monitor an autonomous robot where the control system comprises interacting components. In case of anomalies the system is also able to determine potential root causes and to repair the software via restarting the necessary components. Because of restarting only necessary components the overall availability of the whole system can be increased.

A second branch of model-based reasoning research we are discussing is debugging, i.e., fault localization based on test cases. The idea behind is to represent programs as constraint systems in order to compute fault candidates. A fault candidate is a set of statements that when assumed to work incorrectly explains the detected misbehavior. Using this approach we are able to correctly identify faults for smaller programs. A combination with fault based techniques, i.e., program mutations, and runtime verification seems to be very promising. The latter allows for introducing new knowledge to debugging, which supports distinguishing competing diagnosis candidates. The proposed debugging approach can be used not only a design time but also at runtime allowing to localize and correct faults in programs.

Keywords: Debugging, model-based diagnosis, self-adaptive systems