Amir Pnueli Memorial Symposium, NYU, May 7-9, 2010

May 08

08:15 AM	Welcome	
08:30 AM	David Harel	Can we Verify an Elephant?
09:00 AM	Krzysztof Apt	Juggling using Temporal Logic
09:30 AM	Krishna Palem	The Arrow of Time Through the Lens of Computing
10:00 AM	Break	
10:30 AM	Egon Börger	Ambient Abstract State Machines with Applications
11:00 AM	Manfred Broy	Realizability of System Interface Specifications
11:30 AM	Ofer Strichman	Proving Equivalence between Similar Programs: a Progress Report
12:00 PM	Giora Slutzki	Inverting Proof Systems for Secrecy under OWA
12:30 PM	Lunch	
02:00 PM	Robert Kurshan	Verification-Guided Hierarchical Design
02:30 PM	Werner Damm	Towards Component Based Design of Hybrid Systems
03:00 PM	Ken McMillan	Invisible Invariants: Underapproximating to Overapproximate
03:30 PM	Break	
04:00 PM	Allen Emerson	Time for Time
04:30 PM	Leslie Lamport	Temporal Logic: The Lesser of Three Evils
05:00 PM	Stephan Merz	A Mechanized Proof System for TLA+ Specifications

May 09

08:30 AM	Moshe Vardi	From Löwenheim to Pnueli, from Pnueli to PSL and SVA
09:00 AM	Tom Henzinger	Quantitative Modeling and Verification
09:30 AM	Patrick Cousot	A Scalable Segmented Decision Tree Abstract Domain
10:00 AM	Break	
10:30 AM	Oded Maler	Properties and Verification in the Continuous Domain
11:00 AM	Roni Rosner	The Challenge of Evolutionary Verification
11:30 AM	Javier Esparza	Newtonian Program Analysis: Solving Sharir and Pnueli's Equations.
12:00 PM	Nir Piterman	p-Automata: New Foundations for Discrete-Time Probabilistic Verification
12:30 PM	Lunch	
02:00 PM	Catuscia Palamidessi	Information-Theoretic Approaches to Information Flow and Model Checking Techniques to Measure it
02:30 PM	Rajeev Alur	Architecture-aware Analysis of Concurrent Software
03:00 PM	Willem-Paul De Roever	What is in a Step: New Perspectives on a Classical Question
03:30 PM	Break	
04:00 PM	Muli Safra	Collaboration with Amir: what are the chances?
04:30 PM	Lenore Zuck	Amir: The Axis of Acsys

Amir Pnueli Memorial Symposium, NYU, May 7-9, 2010

Abstracts

From Löwenheim to Pnueli, from Pnueli to PSL and SVA

Moshe Y. Vardi

One of the surprising developments in the area of program verification is how ideas introduced by logicians in the first part of the 20th century ended up yielding at the start of the 21st century industry-standard property-specification languages called PSL and SVA. Amir Pnueli played a key role in this development. This talk attempts to trace the tangled threads of this story.

Juggling using Temporal Logic

Krzysztof Apt

We explain the use of temporal logic in a study of infinite simulations concerned with contingencies such as time, space, shape, size, abstracted into a finite set of qualitative relations. We implemented this approach in the constraint programming system Eclipse by drawing on the ideas from bounded model checking. We illustrate it by discussing a simulation of juggling. This is a joint work with Sebastian Brand.

The Arrow of Time Through the Lens of Computing

Krishna Palem

The concepts of temporal logic were introduced by Amir Pnueli into the realm of computer science in general and programs in particular, to great effect. Given a program specification, a crucial element of reasoning through temporal logic is our ability to assert that one program event occurs before or after the other, with an order intuitively rooted in our notion of "time". In the realm of temporal logic, such assertions are abstracted as pure mathematical facts. An alternative is to consider the physical realization by executing the specified program through, for example, a microprocessor-based system. In such a case, a mechanism is used to ensure that the desired temporal relationships from the program specification are obeyed, and quite often, such a mechanism takes the form of a clock. In physical instantiations however, such mechanisms have an associated energy cost and are guided by the laws of physics in general and thermodynamics in particular, with which the arrow of time and the associated irreversibility are intimately intertwined. Viewed through this lens, a key question arises whether the need for ensuring that the temporal norms needed for program correctness accrue an inevitable energy cost. In this talk, we sketch some of the intricacies underlying this question, and we will hint at the subtle interactions between models of computing, time as it is represented in them, and the associated thermodynamic cost attributes. Amir in his early work relied as much on the philosophy of reasoning about time as on the technical intricacies of mathematical logic. In recognition of the richness of his intellectual endeavor, in this exposition, we adopt a philosophical style mimicking that of the ancient Greek philosopher Zeno.

Amir Pnueli Memorial Symposium, NYU, May 7-9, 2010

Ambient Abstract State Machines with Applications

Egon Börger

We define a simple and flexible abstract ambient concept which turned out to support current programming practice, in fact can be instantiated to apparently all environment paradigms in use in frameworks for distributed computing with heterogeneous components. For the sake of generality and to also support rigorous high-level system design practice we give the definition in terms of Abstract State Machines (ASMs). Their most general notion of state allows one to fully exploit the power of parameterization for defining an arguably most general abstract notion $\AMB exp \IN M $ of machines working in a given environment. We show the definition to uniformly capture the common static and dynamic disciplines for isolating states or run behavior as well as for sharing memory, numerous well-known patterns of object-oriented programming (e.g. for subclassing, encapsulation, delegation, views), but also Cardelli's and Gordon's ambient calculus for mobile agents. Joint work with Antonio Cisternino and Vincenzo Gervasi.

Verification-Guided Hierarchical Design

Robert Kurshan

Amir Pnueli was a forceful early proponent of deductive reasoning as a basis for program verification. In the early years, this appeared to be in conflict with automated verification (primarily model checking). Yet over time, these two directions have been found to be complementary, not antagonistic. Today deductive elements such as compositional reasoning are essential to main-stream model checking and algorithmic drivers of deductive reasoning: preeminently, automatic proof extraction in the context of SAT solving are prevalent.

I will describe how abstraction-based deductive reasoning is being combined with model checking to provide a means for exploiting design hierarchy to overcome performance and capacity limitations in the software verification tools of the Electronic Design Automation industry.

Towards Component Based Design of Hybrid Systems

Werner Damm

We propose a library based incremental design methodology for constructing hybrid controllers from a component library of models of hybrid controllers, such that global safety and stability properties are preserved. To this end, we propose hybrid interface specifications of components characterizing plant regions for which safety and stability properties are guaranteed, as well as exception mechanisms allowing safe and stability-preserving transfer of control whenever the plant evolves towards the boundary of controllable dynamics. We then propose a composition operator for constructing hybrid automata from a library of such pre-characterized components supported by compositional and automatable proofs of hybrid interface specifications.

Invisible Invariants: Underapproximating to Overapproximate

Ken McMillan

In 2001, Pnueli and his co-workers introduced a method they called "invisible invariants". This technique produces proofs of parameterized or infinite-state systems, essentially by generalizing the proof of a finite instance. The method is both surprising and subtle: surprising in that it successfully produces overapproximations by underapproximating, and subtle because it produces a parameterized proof while performing only standard operations of finite-state model checking.

I'll describe the method in terms of abstract interpretation, and relate it to other infinite-state methods, such as indexed predicate abstraction and shape analysis.

Amir Pnueli Memorial Symposium, NYU, May 7-9, 2010

p-Automata: New Foundations for Discrete-Time Probabilistic Verification

Nir Piterman

We develop a new approach to probabilistic verification by adapting notions and techniques from alternating tree automata to the realm of Markov chains. The resulting p-automata determine languages of Markov chains which are proved to be closed under Boolean operations, to subsume bisimulation equivalence classes of Markov chains, and to subsume the set of models of any PCTL formula. Our acceptance game for an input Markov chain to a p-automata is shown to be well-defined and to be in EXPTIME in general; but its complexity is that of PCTL model checking for automata that represent PCTL formulas. We also derive a notion of simulation between p-automata that approximates language containment in EXPTIME. These foundations therefore enable abstraction-based probabilistic model checking for probabilistic specifications that subsume Markov chains, and LTL and CTL* like logics.

Information-Theoretic approaches to Information Flow and Model Checking techniques to measure it.

Catuscia Palamidessi

In recent years, there has been a growing interest in considering the quantitative aspects of Information Flow, partly because often the a priori knowledge of the secret information can be represented by a probability distribution, and partly because the mechanisms to protect the information may use randomization to obfuscate the relation between the secrets and the observables. Among the quantitative approaches to Information Flow, the most prominent is the one based on Information Theory, which interprets a system producing information leakage as a (noisy) channel between the secrets (input) and the observables (output), and the leakage itself as the difference between the Shannon entropies of the input before and after the output (Mutual Information). This approach however suffers from two shortcomings: (1) The Shannon entropy is not the most suitable measure in case of the typical attacks in security (in particular, the one-try attacks), and (2) the analogy with the (simple) channel collapses in case of an interactive system. In this talk we discuss these issues and propose some solutions. Finally, we discuss some methods, based on model checking, to compute the Information Flow associated to a system.

Architecture-aware Analysis of Concurrent Software

Rajeev Alur

Our ability to effectively harness the computational power of multi-processor and multi-core architectures is predicated upon advances in programming languages and tools for developing concurrent software. Recent years have seen intensive research in methods for verifying concurrent software resulting in powerful tools for finding concurrency-related bugs. Almost all of such tools are based on the assumption that the instructions of the same thread are executed according to the program order. This model is called the interleaving model in the verification community, and the sequential consistency model in the computer architecture literature. While this is a commonly accepted language-level memory model, modern multi-processors relax sequential consistency in different ways for performance reasons resulting in weaker models. The goal of our research is to develop tools for analyzing system-level concurrent software along with such details of the underlying architecture. A first step in our research has resulted in a tool called CheckFence. CheckFence analyzes C code for concurrent data types with respect to an axiomatic specification of a memory model. Using a satisfiability solver, for a given client test program, CheckFence searches for discrepancy between operation-level sequential consistency semantics for the data type and concurrent executions feasible with respect to the specified model. We have analyzed a number of benchmarks successfully using CheckFence. Our analysis has revealed a number of potential bugs, and the memory ordering fences needed to fix the bugs. We conclude by discussing research opportunities and challenges for analysis tools that can bridge the gap

between the programmers' desire for simplicity of concurrency abstractions and architects' ability to expose hardware parallelism.

What is in a Step: New Perspectives on a Classical Question

Willem-Paul De Roever

In their seminal 1991 paper "What is in a Step: On the Semantics of Statecharts", Pnueli and Shalev showed how, in the presence of global consistency and while observing causality, the synchronous language Statecharts can be given coinciding operational and declarative macro-step semantics. Over the past decade, this semantics has been supplemented with denotational, game-theoretic and axiomatic characterisations, thus revealing itself as a rather canonical interpretation of the synchrony hypothesis. In this paper, we survey these characterisations and use them to emphasise the close but not widely known relations of Statecharts to the synchronous language Esterel and to the field of logic programming. Additionally, we highlight some early reminiscences on Amir Pnueli's first attempts to characterise the semantics of Statecharts.