Lynne and William Frankel Center for Computer Science

Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel 84105. Tel: (972-8) 6428032 Fax: (972-8) 6428021
www.cs.bgu.ac.il/~frankel

Seminar Series Supported by Jeffrey and Holly Ullman

Verification Day 2008

9 April, 2008

10:30 Coffee and tagging

10:50 Methods for Liveness
Amir Pnueli, New York University and Weizmann Institute (Emeritus)

Abstract: It is a known fact that finitary state abstraction methods, such as predicate abstraction, are inadequate for verifying general liveness properties or even termination of sequential programs. In this talk we will present an abstraction approach called "ranking abstraction" which is sound and complete for verifying all temporally specified properties, including all liveness properties.

We will start by presenting a general simple framework for state abstraction emphasizing that, in order to get soundness, it is necessary to apply an over-approximating abstraction to the system and an under-approximating abstraction to the (temporal) property. We show that the finitary version of this abstraction is complete for verifying all safety properties. We then consider abstraction approaches to the verification of deadlock freedom, presenting some sufficient conditions guaranteeing that deadlock freedom is inherited from the concrete to the abstract.

Finally, we introduce the method of ranking abstraction and illustrate its application to the verification of termination and more general liveness properties. In this presentation we emphasize the similarity between predicate abstraction and its extension into ranking abstraction. In particular, we illustrate how abstraction refinement can be applied to ranking abstraction. Time permitting, we will present a brief comparison between ranking abstraction and the methods of transition abstraction developed by Podelski, Rybalchenko, and Cook which underlie the "Terminator" system.

11:30 Shape Analysis of Concurrent Programs
Mooly Sagiv, TAU

Abstract: Shape analysis concerns the problem of automatically inferring shape invariants for programs that perform destructive updating on dynamically allocated storage. I will describe recent works in shape analysis of concurrent programs in order to prove that concurrent data structure implementations are correct.

12:10 Synthesis from Component Libraries
Yoad Lustig, Student at Hebrew University of Jerusalem

Abstract: Synthesis is the automatic construction of a system from its specification. In the classical synthesis algorithms it is always assumed the system is "constructed from scratch" rather than "composed" from reusable components. This, of course, rarely happens in real life. In real life, almost every non-trivial commercial system, either in hardware or in software system, relies heavily on using libraries of reusable components. Furthermore, other contexts, such as web service orchestration, can be modeled as synthesis of a system from a library of components.

We define and study the problem of LTL synthesis from libraries of reusable components. We define two notions of composition: functional composition, for which we prove the problem is undecidable, and structural composition, for which we prove the problem is 2EXPTIME-complete. As a side benefit we derive an explicit characterization of the information needed by the synthesizer on the underlying components. This characterization can be used as a specification formalism between component providers and integrators.

Joint work with Prof. Moshe Y. Vardi

12:35 Lunch

13:20 Compositional Verification and 3-Valued Abstractions Join Forces
Orna Grumberg, Technion

Abstract: Model checking is an automated technique for the verification of hardware and software systems. It gets a description of a system by means of a state machine (graph). It also gets a specification given as a formula in some temporal logic. It returns "yes" if the system satisfies the specification and "no", otherwise.

Model checking has been successfully applied to verifying complex systems. However, its application to very large systems is limited due to its high space requirements.

Two of the most promising approaches to fighting the high space requirements are abstraction and compositional verification. In this work we join their forces to obtain a novel fully automatic compositional technique that can determine the truth value of the full Mu-calculus logic with respect to a given system.

In the talk we will give the needed background and describe our results.

Joint work with Sharon Shoham

14:00 Tarskian Structures and Languages in the Study of Concurrency
Uri Abraham, Ben-Gurion University

Abstract: Whereas mathematicians argue and formulate their investigations in terms of Tarskian structures and languages, in Verification and Formal Specification research computer scientists rely on different approaches (such as temporal logic) which are amenable to machine checking. In this lecture I will argue that Tarskian structures have a place in the research of concurrency, mainly in that they can promote understanding at a formal level of complex phenomena that arise in that study.

14:40 Self-* Programming: Run-Time RSR-Box Control Synthesis
Olga Brukman, Student at Ben-Gurion University

Abstract: Real life situations may require an automatic fast update of the control of a plant, whether the plant is an airplane that needs to overcome an emergency situation due to drastic environment change, or a process that needs to continue executing an application in spite of a change in an operating system behavior. Settings for run-time control synthesis are defined, assuming the environment maybe totally dynamic, but is reentrant and history oblivious for long enough periods. A reentrant environment allows several copies of a plant to interact with the environment independently; a history oblivious environment ensures a repetition of an interaction (in the probabilistic case with the same probability) starting with a plant in a certain state and replaying its output to the environment.

Total dynamic changes of the environment do not allow a definition of weakly realizable specifications, as weakly realizable specifications depend on the environment behavior. Online experiments of the environment assist in the implementation of unrealizable specifications; automatically checking whether the unrealizable specifications define weakly realizable specifications, given the behavior restriction of the current environment. A successful search for a control implicitly identifies the weakly realizable specifications, and explicitly the implementation that respect the specifications.

Joint work with Prof. Shlomi Dolev

15:05 Coffee

15:20 Correctness and Interference-Freedom for Libraries of Reusable Aspects
Shmuel Katz, Technion

Abstract: Aspects have become popular constructs for modularizing treatment of concerns that otherwise cross-cut object-oriented systems. In this talk, the specification of aspects is defined, and model checking is used to show correctness and interference-freedom among collections of reusable aspects.

16:00 Discriminative Model Checking
Doron Peled, Bar-Ilan University

Abstract: Model checking typically compares a system description with a formal specification, and returns either a counterexample or an affirmation of compatibility between the two descriptions. Counterexamples provide evidence to the existence of an error, but it can still be very difficult to understand what the cause for this error is.

We propose a model checking methodology which uses two levels of specification. Under this methodology, we group executions as good} and bad with respect to satisfying a base LTL specification. We use an analysis specification, in CTL* style, quantifying over the good and bad executions. This specification allows checking not only whether the base specification holds or fails to hold in a system, but also how it does so. Some examples are checking whether one needs infinite number of scheduling decisions to reach a bad execution, or verifying the safety part of a property that does not itself hold for a system. We propose a model checking algorithm in the style of the standard CTL* decision procedure. This framework can be used for comparing between good and bad executions in a system and outside it, providing assistance in locating the design or programming errors.

16:40 On the Succinctness of Nondeterminism
Binyamin Aminoff, Student at Hebrew University of Jerusalem

Abstract: Much is known about the differences in expressiveness and succinctness between nondeterministic and deterministic automata on infinite words. Much less is known about the relative succinctness of the different classes of nondeterministic automata. For example, while the best translation from a nondeterministic Büchi automaton to a nondeterministic co-Büchi automaton is exponential, and involves determinization, no super-linear lower bound is known. This annoying situation, of not being able to use the power of nondeterminism, nor to show that it is powerless, is shared by more problems, with direct applications in formal verification. In this talk I will present a family of problems of this class.

Joint work with Prof. Orna Kupferman