Invited talk: Martin Odersky

Martin Odersky is a professor at EPFL in Lausanne, Switzerland. He is best known as the creator and principal designer of the Scala programming language. Prior to that, he made several contributions to the development of Java. He created the Pizza and GJ languages, designed the original version of generic scs for Java, and wrote the javac reference compiler. He is interested in programming languages and methods, in particular how object-oriented and functional programming can be made to work seamlessly together.

Martin Odersky received his doctorate from ETH Zürich, in 1989. He held researcher positions at the IBM T.J. Watson Research Center and Yale University and faculty positions at the University of Karlsruhe from 1993 and at the University of South Australia from 1997. In 1999 he joined EPFL as full professor. Martin is a fellow of the ACM.

The title of Martin’s talk
Future-proofing collections: from mutable to persistent to parallel

Martin’s Message: In my talk I show the bright side and the dark side of programming with persistent collections. The bright side: It’s really easy and safe to do. The dark side: To make it easy, one has to employ some pretty uncommon and hairy implementation techniques. This is a showcase of the potential and also the subtleties of combining functional and object-oriented programming.

Invited talk: Andreas Podelski

Andreas Podelski studied Mathematics, Logic and Computer Science at Westfälische Wilhelms-Universität Münster. He received his PhD at LITP, University of Paris 7, where he had learned a beautiful part of Theoretical Computer Science, the mathematics of words and free monoids (“an automaton is a quotient of the free monoid”). After his PhD, he worked at the DIGITAL Paris Research Lab (DEC PRL), where he participated in the development of the concurrent constraint programming language LIFE and learned that constraints are first-class data in programs. From 1994-2006 he worked in the group of Harald Ganzinger at the Max Planck Institute for Informatics in Saarbrücken (MPI-INF); Since 2006 he is full professor at the Albert-Ludwigs-University, Freiburg. Andreas’ research concerns different verification methods, in particular abstract interpretation and model checking, and different aspects, e.g. stability in hybrid systems and topology properties in dynamic communication systems.

The title of Andreas’ talk:
An Automata-Theoretic Approach to Program Analysis

The message Andreas wants to convey in his invited talk:
“Go back to the time before the beginning!”

A 1994 survey paper by Leslie Lamport is quite intriguing, not only because it starts with the sentence, “In the beginning, there was Floyd” (referring to Floyd’s famous 1967 paper). The paper explains, “Floyd and Hoare changed the way we think about programs. They taught us to view a program not as a generator of events, but as a state transformer. The concept of state became paramount. States are the province of everyday mathematics—they are described in terms of numbers, sequences, sets, functions, and so on.” Let us go back to the time before the beginning. I.e., let us go back to the event view (not to replace the state view but to complement it). The motivation is to make use of the mathematics of automata over finite and infinite words in program analysis, i.e., in abstraction-based methods to extract runtime properties from programs. Let us now say that an event is nothing but a letter of an alphabet, and a trace is a word over this alphabet (we use transition, action, statement etc. as synonyms for event). We observe that we have immediately freed ourselves from two restrictions (wrt. the classical notion of a trace as an execution): 1. a trace is not necessarily a program trace (i.e., the order of events in a trace does not necessarily obey the order prescribed by the program structure), and 2. a trace is not necessarily feasible (take, e.g., the trace “x:=1; assume(x==1)”). Let us think of a proof of program correctness as putting into relation the three sets of, respectively, program traces, feasible traces, and correct traces. This opens the perspective of decomposing a proof into proofs for subsets, or the perspective of reusing a proof of a superset for the proof of a subset, etc. The program analysis manipulates automata in order to define, represent, and reason about sets of traces. We can generalize traces from words to other mathematical objects (e.g., infinite words, nested words, trees, Petri nets) and use corresponding notions of automata to reason about infinite, recursive, branching resp. concurrent executions. Vardi and Wolper have brought automata to model checking. They exploited the fact that both the behaviour of a finite-state model and the temporal-logic correctness property can be defined by automata. It is now time to integrate automata into program analysis. It is now time to go back to the time before the beginning.
The ETAPS Awards

There are traditionally three ETAPS awards. Each of them is sponsored by one of the European ETAPS partner associations. EATCS, the European Association for Theoretical Computer Science honors the best paper in theoretical Computer Science. The best paper on software technology is honored by the European Association of Software Science and Technology EASST. A third award, dedicated to programming language advances is presented by EAPLS, the European Association for Programming Languages and Systems.

This year, a fourth award will be given for the first time ever. It goes to the best student paper and is sponsored by Springer-Verlag, the publisher of the ETAPS main conference proceedings. The 2011 proceedings are published as Volume 6601 - 6605 in Lecture Notes in Computer Science, LNCS, as usual in the LNCS Advanced Research in Computing and Software Science subline, ARCOSS.

The following papers are nominated for the ETAPS 2011 awards:
- SSA-based Register Allocation with PBQP Buchwald, Zwinkau, Bersh (Mon 12:00 at CC)
- Boosting Lazy Abstraction for SystemC with Partial Order Reduction Cimatti, Narasamya, Roveri (Thu 10:30 at TACAS)
- Canonized rewriting and Ground AC-Completion Modulo Shostak Theories Conchon, Contejean, Iguernelala (Mon 14:00 at TACAS)
- The reduced product of abstract domains and the combination of decision procedures, Cousot, Mauborgne (Wed 17:45 at FOSSACS)

An Interface Theory for Service-Oriented Design Fiaideiro, Lopes (Wed 10:30 at FASE)
Efficient Interpolant in Satisfiability Modulo Linear Integer Algebra Griggio, Le, Sebastiani (Tue 10:30 at TACAS)
Church Synthesis Problem for Noisy Input Velner, Rabinovich (Tue 17:15 at FOSSACS)
Probabilistic Modal mu-Calculus with Independent Product Matteo, Mio (Tue 17:45 at FOSSACS)

Change of rooms

Today, TACAS is held in E1.3/002, not in E1.3/001. The new room is right to the old room (!). It can hold 50% more participants, it is in service since yesterday after lunch.

Lunch options on Tuesday

**Menu A**
- Roman roast with vegetables, seasonal salad, soup, dessert

**Menu B**
- Spring roll with sweet and sour sauce, vegetables, seasonal salad, dessert

Free Flow
- Pork steak with pepper sauce
- Noodles with vegetables
- Steamed fish fillet with sauce

Two important guidelines from AOAA, our Administration Office for Administrative Administration:
1) Sessions are synchronized. Session chairs will make sure time is strictly kept. Feel free to leave the room to switch conference after a talk; that’s the idea. In any case, each lecture hall has a rear entrance.
2) In the break before your session, please try out the connection of your laptop to our projector. Our helpers are ready to assist you. They wear black T-Shirts with a white owl.

What’s in this building?

The Max Planck Institute for Informatics, MPI-INF, is devoted to cutting-edge research in informatics with a focus on algorithms and their applications in a broad sense. Our research ranges from foundations (algorithms and complexity, programming logics) to applications in a variety of fields.

And in this?

The Center for Bioinformatics, CBI, is a scientific institution of Saarland University. It has just last week celebrated its 10th anniversary.

The Intel Visual Computing Institute, IVCI, was founded as a collaborative, basic research and development enterprise of Intel Corporation, Saarland University Computer Science, and the surrounding computer science institutes. Visual Computing, i.e. the acquisition, processing, storing, transport and rendering of visual data, is a central theme in driving the understanding of and interaction with natural and synthetic worlds that modern computers are able to present to their users.