Seminar-Themen: Methoden und Werkzeuge zur Verifikation von Software

January 27, 2000

1. **Intuitionistische und konstruktive Logik**
   Quellen: Kapitel aus Isabelle-Introduction,
   Buch von Raymond Turner: constructive functional programming, Kapitel 2, 11.1, 11.2, 13.

   L. Paulson: ML for the working programmer: Kapitel 10.

2. **Isabelle:**
   `http://www4.informatik.tu-muenchen.de/~isabelle/`

   "Introduction to Isabelle" (intro.pdf aus dem Isabelle Dokumentationspaket 24. August 1999)

   (a) **Grundlagen für Isabelle mit Ausnahme der intuitionistischen Logik:**
   Teil I komplett

   (b) **Anwendung von Isabelle:**
   Teil II komplett Teil III, Abschnitte 10 bis einschl. 12

   (c) **Fallstudie Isabelle:**
   Verifikation eines lexikalischen Analyse-programms.

3. **HOL:**
   `http://www.cl.cam.ac.uk/Research/HVG/HOL/HOL.html`

   The HOL System is an environment for interactive theorem proving in a higher-order logic. Its most outstanding feature is its high degree of programmability through the meta-language ML. The system has a wide variety of uses from formalizing pure mathematics to verification of industrial hardware. Academic and industrial sites world-wide are using HOL. The name 'HOL' is pronounced either as a word to rhyme with 'doll' or letter by letter. The system is available without charge.

4. **C formalised in HOL**

   http://www.cl.cam.ac.uk/ mn200/PhD/

   (Michael Norrish's PhD Thesis):

   Es wird eine formale Semantik für die Programmiersprache C angegeben, die das Typsystem und das operationale Verhalten von Programmen abdeckt. Die Semantik läßt die C Bibliothek außen vor, deckt aber sonst bis auf Sprungbefehle den Sprachumfang von C ab. Es wird eine Transitionsrelation für die Ausdrücke, Befehle und Deklarationen von C angegeben.

5. **KIV: Karlsruhe Interactive Verifier**
   `http://www.informatik.uni-ulm.de/pm/kiv/`

   Papiere: The KIV-approach to software verification,

   Structured specifications and interactive proofs with KIV. (auf www-Seite)
   Wolfgang Reif: Formale Methoden für sicherheitskritische Software - Der KIV-Ansatz

   The Formal Methods Group Ulm performs research, development, and education in software quality, computer safety, and security. The emphasis is put on the systematic construction of high assurance systems for technical, safety-critical, and commercial applications. A major competence are formal methods for rigorous specification, mathematical analysis, and verification of systems. For more than a decade the group has experience with a variety of formal methods in different application areas. The group has developed industrial strength support tools, and has a strong background in mechanized reasoning. The group distributes the tools VSE and KIV, and offers various services to industry and academia.

6. **Coq:** `http://pauillac.inria.fr/coq/pres-eng.html`

   Coq is a research project common to INRIA, LIP (ENS Lyon, CNRS) and LRI.

   Our work deals with effectively machine-checked formal mathematics. In practice, this includes the study of mathematical formalisms well-suited for implementations, the implementations themselves and the use of these for various applications. We are particulary interested in software correctness proofs.

   More precisely, we currently focus on:

   The development of the Coq proof assistant. The study and development of the Calculus of Inductive Constructions which is the formalism used in Coq; it comes from proof theory and allows a mix of abstract axiomatizations, constructive specifications and functional programming. A high order specification language, Gallina, and a modular programming language (FML) in order to provide development environment for certified programs with corresponding automatic verification tools. Zero-defect software in mission - critical fields (on-board software, communication protocols, etc).

7. **Larch:** `http://www.sds.lcs.mit.edu/spd/larch/`

   Spezifikations-Sprache und Werkzeuge, der Ansatz geht zurück in die siebziger und ist daher evtl. schon von historischem Interesse. Ein Beweiser (LP) existiert.

   Quellen: "An Overview of Larch" + Artikel dazu.

8. **CafeOBJ:** `http://www.theta.ro/cafeobj/`

   An Overview of CAFE Specification Environment (abstract)

   Kokichi Futatsugi and Ataru Nakagawa

CAFE is the name of a network based environment now under development for supporting systematic creation, checking, verification, and maintenance of formal specifications. CAFE has an algebraic specification language called CafeOBJ as its main specification language, and adopts algebraic specification paradigm as its foundation. CafeOBJ is a successor of the OBJ language, and has important new features for concurrency and behavioral specifications. Concurrency and behavior are specified based on rewriting logic and behavioral (hidden-sorted) algebra respectively. These new features make it possible to provide powerful language constructs for formal object-oriented specifications.

CAFE is designed to be a network based environment. For sharing specification documents systematically over the network, a new document formatting language called Forsdonnet (FORmal Specification Document ON NETwork) is designed by extending HTML. Forsdonnet includes constructs for formal and informal specifications, commands for executing (prototyping) and checking/verifying CafeOBJ specifications, etc. Forsdonnet is designed to be based on already established standard network infrastructure components like HTML and Netscape.

"Logical Semantics for CafeOBJ (abstract)"

9. **ACL2:** `http://www.cs.utexas.edu/users/moore/acl2/`

   ACL2 is an automated reasoning system developed (for the first 9 years) at Computational Logic, Inc. and (from January, 1997) at the University of Texas at Austin. It is the successor to the Nqthm (or Boyer-Moore) logic and proof system and its Pc-Nqthm interactive enhancement. The acronym ACL2 actually stands for "A Computational Logic for Applicative Common Lisp". This title suggests several distinct but related aspects of ACL2.

   We assume that readers of the ACL2 documentation have at least a very slight familiarity with some Lisp-like language. We will address the issue of prerequisites further, in "ABOUT THIS TUTORIAL" below.

10. **Letos:** `http://www.ecs.soton.ac.uk/~phh/latos.html`

    LETOS – A Lightweight Execution Tool for Operational Semantics

    Pieter H. Hartel, Univ. of Southampton, May 1999

    A lightweight tool is proposed to aid in the development of operational semantics. To use letos, an operational semantics must be expressed in its meta-language, which itself is a superset of Miranda. The letos compiler is smaller than comparable tools, yet letos is powerful enough to support publication quality rendering using LaTeX, fast enough to provide competitive execution and tracing using Miranda or Haskell, and versatile enough to support derivation tree browsing using Netscape. Letos has been applied to a Java Secure Processor, which is a version of the Java Virtual Machine intended to run on smart cards. Letos has also been used with subsets of various programming languages. Letos is unique in that it helps to check that a specification is operationally conservative.