ETAPS 2003 Report

Patryk Zadarnowski

Waraw
27 May 2003

University of New South Wales
Sydney
OVERVIEW

ETAPS — Joint European Conferences on Theory and Practice of Software:

→ CC — International Conference on Compiler Construction
→ ESOP — European Symposium on Programming
→ FASE — Fundamental Approaches to Software Engineering
→ FOSSACS — Foundations of Software Science and Computation Structures
→ TACAS — Tools and Algorithms for the Construction and Analysis of Systems
→ plus 22 satellite workshops...
Generic Theories and Theories of Genericity
by Samson Abramsky (Oxford)

- Generic Programming as games
- Arrange to play two games with a chess master, one as white, one as black, watch his moves in one game and replay it in another
- You’re guaranteed to tie.
Combined Code Motion and Register Allocation using the VSDG
Neil Johnson, Alan Mycroft (Cambridge)

- VSDG normalizes program by removing all unnecessary dependencies
- Partitions the graph into slices, each no wider than the number of available registers
- Use code motion if necessary, duplicate expressions, spill as last resort
- Partitioning performed by adding bogus state (control) dependencies
Register Allocation by Optimal Graph Coloring
Christian Andersson (Lund Institute of Technology)

- General graph coloring NP-hard
- But register allocation is not general
- Special case of 1-perfect graphs easy to color
- Analyzed 28,000 real-life dependence graphs: all are 1-perfect
- So, optimal graph coloring possible, but WHY?!
CC LANGUAGE CONSTRUCTS SESSION

A Pattern Matching Compiler for Multiple Target Languages
Pierre-Etienne Moreau, Christophe Ringeissen (LORIA-INRIA)

→ A pattern-matching preprocessor for C
→ How boring can you get?
A New One-Pass Transformation into Monadic Normal Form
Olivier Danvy (Aarhus)

- Source: call-by-value λ-calculus
- Target: Monadic Normal Form (just like ANF, but monadic)
- Normally, first translated into ANF, then into monadic normal form
- Olivier does it in one pass
- Gets quite tricky
- Since MNF directly corresponds to CPS, and CPS corresponds to SSA, this may be used as a one-pass transformation from λ-calculus to SSA.
The Verifying Compiler: still a Grand Challenge for Computing Research

Tony Hoare (Microsoft Research)

- Set the tone for the rest of the conference
- A project for next 20 years
- Goal: to promote large-scale international collaboration
- Feasible because of the vast amount of OS code usable as testbed
- Incremental (initial stages quite simple)
- Aiding specification part of the goal
- Verifying of the compiler itself not a goal
- “Finally, it must be recognized that a verifying compiler will be only part of an integrated and rational tool-set for reliable software construction and evolution, based on sound scientific principles.”
ETAPS INVITED LECTURE

Computer Security from a Programming Language and Static Analysis Perspective
Xavier Leroy (INRIA)

→ Making JavaCard secure through static analysis
→ Doesn’t believe in verifying compilers
→ Can prevent hardware attacks with software
ESOP TECHNIQUES & METHODS SESSION

A Tail-Recursive Semantics for Stack Inspection
John Clements, Matthias Felleisen

» Doing tail-calls securely in a JVM-like environment
» Allows tail-calls to untrusted procedures
» Requires only one bit of context per permission
» No need to recompile library code (no annotations == full permissions)
What Are We Trying To Prove?
Peter Lee

- Presents experiences with real-life certifying Java compiler
- In the end, could compile staroffice (?) without annotations
- Safety policy simple enough to correspond to Java type system
- Conclusion: feasible, a long way to go
- Remarkable quote: “I wrote it in ML, I don’t consider Java a civilized language”
Two interesting papers:

*Learning Assumptions for Compositional Verification*
Jamieson Cobleigh, Dimitra Giannakopoulou, Corina Pasareanu (NASA)
- Learns assumptions about component interfaces by observing component interactions
- Used to certify Mars robots, etc.

*Automated Module Composition*
Stavros Tripakis (VERIMAG)
- Automatically constructs systems from modules given requirements and module interface specifications
Correctness of Data Representations Involving Heap Data Structures
Uday Reddy, Hongseok Yang

- Defines semantics for a Pascal-like language with pointers and heap variables.
- Lots of category theory
- Interesting, but heavy-going.
Building Certified Libraries for PCC: Dynamic Storage Allocation
Dachuan Yu, Nadeem Hamid, Zhong Shao

- Problem: certifying malloc and free
- Solution: PCC
- Problem: complexity of proofs
- Solution: three guys on Yale salaries with no life and lots of spare time on their hands
- Works on a certified assembly-like language CAP
Register Allocation by Proof Transformation
Atsushi Ohori

- Utilizes judgements-as-types correspondence
- Programs == proofs
- Use types to express register annotations
- Then, program transformations are just proof transformations
- Very readable paper.