Topic of the seminar

• Basic information about ETAPS
• Selected papers from ETAPS 2010
Basic information about ETAPS
What is ETAPS

- Five main conferences
  - CC: 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 Construction and Analysis of Systems

- Many workshops (~ 15)
  - BYTECODE, FESCA, …
What is ETAPS

• Main conferences
  ▪ Proceedings: LNCS
    • New sub-line: ARCoSS
      ▪ Advanced Research in Computing and Software Science
      ▪ More prestigious than plain LNCS
  ▪ Acceptance ratio: 20-25 %
  ▪ Timing: Monday – Friday
    • Parallel tracks of different conferences (3-4)

• Workshops
  ▪ Proceedings: mostly ENTCS
  ▪ Timing: weekends before and after
Topics

• Logic and automata
• Program verification
• Static code analysis
• Software testing
• Model transformations
• Lifecycle and evolution
• Performance analysis
• Decision procedures
• Probabilistic systems
• … and many other
General impression

• Focus
  - Practical aspects of software science
    - Algorithms and their application using tools
  - Really significant theoretical or experimental results published elsewhere
    - CAV, PLDI, OOPSLA, … (A+)

• Community
  - Large groups of people from top institutions 50 %
    - MSR, Oxford, Cambridge, ETHZ, UIUC, …
    - They know well each other ➔ little bit hard to get inside
  - People from other institutions (1-2 each) 50 %
Why it is good to attend

• “Probably the best conference on Software Science in the world” (SC chair)
  ▪ Probably the best in Europe?

• Opportunity to meet many people
  ▪ Famous researchers and professors
    • “You don’t talk with Thomas Ball every day”
  ▪ Young PhD students and assistant professors

• All participants are very open to questions and discussion (> 300)
  ▪ New contacts ➔ projects, internships, post-docs
Why it is good to submit

• Good publication
  ▪ Main conferences have level A

• Not so hard to get on main conferences
  ▪ Our experience: FASE’09 (OS), TACAS’10 (PP+TK)
  ▪ Compared to CAV, PLDI, POPL, OOPSLA, … (A+)
    ▪ These require much more work (implementation, experiments, proofs, and paper writing)
  ▪ Not only for people from UK/USA (CH, FR, DE)
    ▪ Papers from Poland, VUT Brno (best paper!!), Italy, …
What to do if your paper is accepted

• Go there alone and avoid Czech people
  ▪ “You don’t have to go to ETAPS to talk with your colleague from DSRG/Brno (UKA, …).”
It is not only about work and research...
Selected papers from ETAPS 2010
Keynote presentations

• M. Harman: Why the Virtual Nature of Software Makes it Ideal for Search Based Optimization
• D. Naumann: Dynamic Boundaries: Information Hiding by Second Order Framing with First Order Assertions
• M. Vardi: Logics in Computer Science
  ▪ History of CS from the perspective of logic
• P. Wadler: The Audacity of Hope: Thoughts on Reclaiming the Database Dream [skipped]
• J.-F. Raskin: Antichain Algorithms for Finite Automata
• C. Stirling: An Introduction to Decidability of Higher-Order Matching
• J. Esparza: Newtonian Program Analysis
  ▪ Numerical methods in computation of fix-points over CFGs
Research papers

- Modular verification of concurrent programs
- Search for concurrency errors
- HW/SW co-verification
- Performance analysis
Modular verification of concurrent programs

  - Modular verification of deadlock freedom
    - Concurrent programs involving locks and message passing
    - Translation into Boogie IVL and then to logical formula
      - Checking validity of formula with SMT solver (Z3)
  - Key ideas
    - Global wait order → common over locks and channels in program
    - Contracts describe permissions (locks) and credits (messaging)
  - Implemented in Chalice
    - Programming language and verifier for concurrent programs
    - Key features: contracts (pre/post/inv), permissions, locks

- Reminder: seminar about fractional permissions and fine-grained locking by OS a year ago
class Account {
    var balance: int;

    invariant acc(this.balance);

    method Main() {
        var a := new Account;
        a.balance := 10;
        var b := new Account;
        call b.SetBalance(20);
        share a above waitlevel;
        share b above a;
        call Transfer(a, b, 5);
    }

    ...
}

method SetBalance(a: int) {
    requires acc(balance);
    ensures acc(balance)
        && balance == a;
    { balance := a; }
}

method Transfer(from: Account, to: Account, amount: int) {
    requires waitlevel << from.mu
        && from.mu << to.mu;
    { acquire from; acquire to;
        fork tok := to.SetBalance(
            to.balance + amount);
        call from.SetBalance(
            from.balance - amount;
        join tok;
        release to; release from;
    }
method Produce(ch: Ch)
    requires credit(ch, -1);
    {
        var i := 0;
        while (i < 10)
            invariant 0 <= i && i <= 10 && credit(ch, -1);
            { send ch(i); i := i + 1; }
        send ch(-1);
    }

method Consume(ch: Ch)
    requires credit(ch, 1) &&
        waitlevel << ch.mu;
    {
        var x: int;
        receive x := ch;
        while (0 <= x)
            invariant waitlevel << ch.mu;
            invariant 0 <= x
                ==> credit(ch, 1);
            { receive x := ch; }
    }
Search for concurrency errors


  - **Motivation (goals)**
    - Detection of multiple distinct bugs ➞ pipelining of testing process
    - Compositional testing ➞ focus on specific components (layers)

  - **Overall approach: systematic concurrency testing of programs using the Chess tool**
    - Input: set of tests representing different concurrency scenarios
    - Basic algorithm: repeated execution of the given program
      - Different thread schedule explored in each run
    - Challenge: selecting useful thread schedules with respect to goals

  - **Contribution of this paper: preemption sealing**
    - Idea: scheduler disables preemption in particular scopes (methods)
Preemption sealing: example

(a) public class Acct {
    volatile int bal;

    public Acct(int n) {
        bal = n;
    }

    public void Withdraw(int n) {
        int tmp = Read();
        lock (this) {
            bal = tmp - n;
        }
        public int Read() {
            return bal;
        }

        public void Deposit(int n) {
            lock (this) {
                var tmp = bal;
                bal = 0;
            }
            bal = tmp + n;
        }
    }
}

void TestAcct() {
    var acc = new Acct(10);

    var t1 = new Thread(o =>
        { (o as Acct).Withdraw(2);
        L9:  });

    var t2 = new Thread(o =>
        { var b = (o as Acct).Read();
        LA:   assert(b>=8);
        LB:   });

    var t3 = new Thread(o =>
        { (o as Acct).Deposit(1);
        LC:   });

    t1.Start(acc); t2.Start(acc);
    t3.Start(acc);

    LD: assert(account.Read() == 9);
    LE:
Chess: practical view

• Addressing the goals
  ▪ Identifying multiple errors: disable preemptions in the scope related to the root cause of already discovered bug
  ▪ Enabling compositional testing: user provides a list of methods that were already tested

• Evaluation
  ▪ Concurrency libraries for .NET
    • Concurrency and Coordination Runtime, Parallel LINQ
  ▪ Results: much improved efficiency ➔ many thread interleavings eliminated

• Integrated into MS Visual Studio

• Conclusion: relatively simple idea, but implemented and applied in practice (good evaluation)
HW/SW co-verification

- J. Li, F. Xie, T. Ball, V. Levin, and C. McGarvey. An Automata-Theoretic Approach to Hardware/Software Co-verification, FASE
  - Goal: checking HW/SW interface properties
    - These are not covered by SDV/SLAM
  - Key idea: hardware and software behavior verified together
    - Integration of HW and SW representations in one formal model
  - Behavior co-specification
    - Software: C program of device driver
      - Modeled by push-down system (PDS)
    - Hardware: VHDL or SystemC
      - Modeled by Buchi automation (BA)
  - Case study
    - Windows driver and hardware model for PIO-24 digital I/O card
Verification algorithm

- HW/SW interface: shared states (memory) and interface events (interrupts)
  - Modeled by Buchi Pushdown System (BPDS), which synchronizes PDS and BA

![Diagram showing Co-specification, Abstraction, Formal Model BPDS, Model-checking]

- Model checking reachability properties of BPDS
  - Predicate abstraction of HW and SW models is performed before BPDS is created
  - tool CoVer: BPDS2PDS + Moped (model checker for PDS)

![Diagram showing BPDS, BPDS2PDS, PDS, Moped Model-checker]
Performance analysis

• V. Cortellessa, A. Martens, R. Reussner, and C. Trubiani. A Process to Effectively Identify “Guilty” Performance Antipatterns, FASE

  ▪ Motivation
    • Problem: interpreting results of performance analysis
    • Large gap between representation of results and feedback expected by software designers
      ▪ mean response time X design suggestions
  ▪ Overall goal: identification of performance problems
    • Automatic detection and solving of performance anti-patterns
    • Modification of the software system model to remove some performance issues
Interpretation: whole process
• Contribution: method for determining “guilty” anti-patterns w.r.t. requirements
  ▪ Assumptions
    • Violated requirements (too high response time, …)
    • Complete list of anti-patterns (blob, one-lane bridge, …)

• Ranking based on
  ▪ Involved entities: processor, component, …
  ▪ Metrics: utilization, response time, throughput

• Case study: business reporting system
  ▪ web server, database, reporting engine, GUI, …

• Opinion: very few technical details provided
Workshops

• Bytecode
  ▪ Very nice invited talks
    • F. Logozzo (MSR): Code Contracts + Clousot
    • M. Parkinson (Cambridge): Design of jStar
    • F. Spoto (Uni Verona): Static analysis of Java code
  ▪ Key message: it takes years (5+) to develop a reasonable code analysis tool

• FESCA
  ▪ Talks about service composition and adaptation
• (many others)
Next time
Next time

• ETAPS 2011
  ▪ Location: Saarbrucken
  ▪ Date: 26.3. – 3.4. 2011
  ▪ Submission
    • Main conferences: beginning of October 2010
    • Workshops: deadlines in December or January

• ETAPS 2012
  ▪ Location: Estonia (Tallinn ?)