Relation to Current Research @IAIK

and Some Administrative Issues

Georg Hofferek
IAIK – Graz University of Technology
georg.hofferek@iaik.tugraz.at
Motivation

- Realize that …
  - … we learned useful things
  - … people use this in practice

- Motivate to …
  - … deepen your knowledge
  - … work with us
    - Bachelor thesis, etc.
Outline

- Course Summary
- Short talks by
  - Georg Hofferek
    - on behalf of Bettina Könighofer
  - Franz Röck
  - Ayrat Kahlimov
  - Robert Könighofer

- Some Administrative Things
Syntax

- Atomic propositions
  - Model (simple) declarative sentences

- Connectives
  - $\land$
  - $\lor$
  - $\neg$
  - $\rightarrow$
Semantics

- Meaning of Formulas
  - True or False

- Truth Tables
  - Define connectives

- Models
  - Valuations of atoms
Natural Deduction

- Syntax-based rules
  - Deduce new statements from premises
  - Independent of meaning

- Soundness
  - Only correct statements can be deduced

- Completeness
  - All correct statements can be deduced
Combinational Equivalence Checking

- Reduction to Satisfiability
  - Table of problem reductions

- Normal Forms
  - Conjunctive Normal Form
  - Disjunctive Normal Form

- Equisatisfiability
  - Tseitin-Encoding
SAT Solving

- DPLL Algorithm
  - Boolean Constraint Propagation
  - Pure Literals

- Conflict-Driven Clause Learning

- Resolution Proofs of Unsatisfiability
Symbolic Computations

- Symbolically Encoded Sets
  - Characteristic Functions
  - Set Operations $\Rightarrow$ Boolean Operations

- Efficient
  - Representation
  - Manipulation

- Interpolation
  - Cheap operation
  - Useful in many scenarios
Binary Decision Diagrams

- Representation for Boolean Functions
  - Graph-based
  - Efficient
  - Canonical

- Well-suited for symbolic computations
Predicate Logic

- More Expressive

- But at a price
  - (possibly) infinite models
  - only semi-decidable
Natural Deduction for Predicate Logic

- Rules for Quantifiers
- Basis for “real” proofs
Theories and SMT

- Common use cases
- Decidable fragments
- Satisfiability Modulo Theories
  - Eager Encoding
  - Lazy Encoding
  - DPLL(T)
Decidability

- Limits of computation
  - Some problems not decidable

- Limits of formal system
  - Gödel’s incompleteness theorem
Summary

How?
- Syntax

What?
- Semantics
- Modeling concrete problem
- Symbolic encodings

Data Structures
- Conjunctive Normal Forms (DIMACS)
- Binary Decision Diagrams

Solve
- Proof by Natural Deduction
- SAT/SMT Solver
Synthesis of Atomic Sections

Bettina Könighofer
(represented today by Georg Hofferek)

IAIK – Graz University of Technology
bettina.koenighofer@iaik.tugraz.at
Easy to specify – Hard to implement

“Play the Austrian national anthem on a violin.”

1. Pick up violin and bow
2. Press ring finger on d-string
3. Stroke bow downwards over d-string for one beat
4. Stroke bow upwards over d-string for one beat
5. Stroke bow downwards over a-string for one beat
6. Press middle finger on d-string
7. Stroke bow upwards on d-string for two beats
8. Press ring finger on d-string
9. Stroke bow downwards on d-string for half a beat
10. Stroke bow upwards on a-string for half a beat
11. …
Hard to specify – Easy to implement

The internet is broken!

Specify:
- How to fix it
- Using natural language
- On the phone

I’ll just come over and fix it for you.
Concurrent Programs

Example: RSA Decryption

1. \( m_1 = c^d \mod p \)
2. \( m_2 = c^d \mod q \)
3. \( m_1 = \text{crt}(m_1, m_2) \)

Parallel Program

Thread1()
1. \( m_1 = c^d \mod p \)
2. \( \text{finished}_1 = \text{true} \)
3. IF !\text{merged AND } \text{finished}_2
4. \( \text{merged} = \text{true} \)
5. \( m_1 = \text{crt}(m_1, m_2) \)

Thread2()
1. \( m_2 = c^d \mod q \)
2. \( \text{finished}_2 = \text{true} \)
3. IF !\text{merged AND } \text{finished}_1
4. \( \text{merged} = \text{true} \)
5. \( m_1 = \text{crt}(m_1, m_2) \)

Sequential Program

Compute \( m_1, m_2 \) in parallel

Is this correct?

\[ m_1^{\text{seq}} ? = m_1^{\text{parallel}} \]
Insert Atomic Sections

Thread 1()
1. \( m_1 = c^d \mod p \)
2. \( \text{finished}_1 = \text{true} \)
3. IF \( \neg \text{merged} \) AND \( \text{finished}_2 \)
4. \( \text{merged} = \text{true} \)
5. \( m_1 = \text{crt}(m_1, m_2) \)

Thread 2()
1. \( m_2 = c^d \mod q \)
2. \( \text{finished}_2 = \text{true} \)
3. IF \( \neg \text{merged} \) AND \( \text{finished}_1 \)
4. \( \text{merged} = \text{true} \)
5. \( m_1 = \text{crt}(m_1, m_2) \)

⚠️ Hard to implement
⚠️ Hard to test
😊 Easy to specify:

Concurrent program should behave as if executed sequentially

Let’s insert atomic sections automatically!
The Need for Abstraction

- Specify in propositional logic:

  \[ a \text{[64bit]} \rightarrow \text{Multiplier} \rightarrow b \text{[64bit]} \rightarrow \text{p [128bit]} \]

- Scales exponentially!
Abstraction with Uninterpreted Functions

- Detailed semantics often irrelevant
  - E.g. Program equivalence:

```c
int program_1(int x, int y) {
    int tmp1 = f(x);
    int tmp2 = f(y);
    return plus(tmp1, tmp2);
}
```

```c
int program_2(int x, int y) {
    int tmp = plus(f(y), f(x));
    return tmp;
}
```

- But: Spurious counterexamples
  - `plus(f(x), f(y)) == plus(f(y), f(x))`?

Theories in Predicate Logic
In many cases, correctness is independent of semantic of functions. Use uninterpreted functions.
Verifying (Abstract) Concurrent Programs

- Preprocessing
  - Loop Unrolling
  - Single Static Assignment Form

- Do Abstraction
  - E.g. Uninterpreted Functions

- Encode in SMT Formula ...
  - Deal with scheduling

... such that:
- UNSAT: Program correct
- SAT: Model represents invalid run
Synthesis Approach

Concurrent Program
Abstraction
SMT Encoding
Verification
Synchronized Program

Atomic Sections Candidates
Counterexample Analysis
Counterexample
Valid

Uninterpreted Functions
Syntax & Semantics
SMT Solving
Models
Results

- **AtoSS**
  - “Atomic Section Synthesizer”
  - [http://www.iaik.tugraz.at/content/research/design_verification/atoss/](http://www.iaik.tugraz.at/content/research/design_verification/atoss/)

- **Fixed real Linux kernel bug**
  - **CVE-2014-0196**
    - [https://bugzilla.redhat.com/show_bug.cgi?id=1094232](https://bugzilla.redhat.com/show_bug.cgi?id=1094232)

Many questions, optimizations, and different approaches left! ➔ You can help!
Automated Test Case Generation

Franz Roeck
IAIK – Graz University of Technology
franz.roeck@iaik.tugraz.at
Software Development and Certification

Program Specification

Test Suite

Formal Model

Implementation

PDF
Cover the Specification Logically

- Apply coverage criteria on a guard
  - Decision Coverage, Condition Coverage, MCDC, ...

SAT assignment to achieve e.g. decision coverage

decision
+
Not (decision)
Cover a Graph

- Generate test cases covering all Nodes, all Edges, all paths up to length n, ...

- Basic approach with model checker:
  Trap properties
  Result: arbitrary number of test cases
Cover Formal Properties

- Requirement: *Whenever we notice that we are under attack, shut down.*

- Simple test case: Attack $\rightarrow$ Shutdown

But we can‘t test if this works in every situation at any time.
Enhance Existing Test Suite

- Code coverage 60%
- Find an input to increase the code coverage

```c
x = foo(input);
If (x == 42)
    printf("not yet covered\n");
```
Parameterized Synthesis

Based on SMT-Solving

Swen Jacobs, Ayrat Khalimov
ayrat.khalimov@gmail.com
Why? Scalability!

Synthesis Time of AMBA Arbiter
Why? *Distributed algs!*
Parameterized Synthesis Problem

Given \textit{spec}, synthesize \( P \):

\[ \forall n \ P^n \models spec \]
Idea? Cutoffs!

\[ \models f_{\text{spec}} \]

\[ \models spec \]
Topics?

- **Decidability**
  - If a system can simulate a partial run of a *Turing machine*
    - don’t bother – remember the *halting problem*?

- **Linear Temporal Logic**

- ‘Logic’ reasoning
Part 2

Bounded Synthesis
how to synthesize?

\[
\models \Longrightarrow \text{spec}
\]

**Bounded synthesis!**
Bounded Synthesis? *SMT*!

- Convert specification and architecture into (quantified) constraints
  - $\forall i \forall t. P_1(t) \rightarrow (P_1(\text{next}(t, i)) \land \text{val}(\text{next}(t, i)) \geq \text{val}(t))$
- Solve
  - $\forall i \forall t. \phi(i, t)$ for fixed set of states $t$
  - If yes: found implementation
  - If no: need bigger set of states

**SMT Formula**

**First-order Predicate**

**Uninterpreted Function**

**Quantifier Instantiation, SMT Solving**
Interested? Join!

1. Parameterized synthesis
   - deadlocks in resource allocation systems

2. Comparative synthesis
   - given a system -- synthesize a better one

3. Robotics synthesis problems
   - generator of benchmarks
   - optimized synthesizer

ayrat.khalimov@gmail.com
Automatic Debugging of Software Programs

Based on SMT-Solving

Robert Könighofer
IAIK – Graz University of Technology
robert.koenighofer@iaik.tugraz.at
Automatic Debugging: Challenge

**Given:**
- Incorrect program
- Specification
  - Assertions
  - Reference implementation
  - Test cases with expected outputs

**Find:**
- Possible error locations
- Possible error corrections

```c
int max(int x, int y) {
    int res = x;
    if(y > x)
        res = y;  // “x” in line 4 is incorrect
    assert(res>=x && res>=y);
    return res;
}
```

Replace it by “y”
Automatic Debugging: Solution

For each “part” of the program:

- Replace part with placeholder \( R \)
  - \( R \) means: “replace me with something”

- Lift to the domain of logic
  - \( \text{correct}(x, y, R) = y \leq x \lor R \geq y \)

- Symbolic Execution

- Solve
  - \( \forall x, y: \exists R: \text{correct}(x, y, R) \)
  - If yes: possible error location
  - Compute \( R = f(x, y) \)
Automatic Debugging: Tool

- **FoREnSiC**
  - Error localization and correction
  - For C programs
  - Various different methods

Many questions, optimizations, and different approaches left! ➔ You can help!
Hardware Synthesis

Robert Könighofer
IAIK – Graz University of Technology
robert.koenighofer@iaik.tugraz.at
Hardware Synthesis

What? shall be done

Symbolic Encoding

Strategy Computation

Relation Determinization

Implementation

How? can it be achieved

Solvers for Predicate Logic

Synthesis Tool

QBF Solver

Tseitin’s Encoding

BDDs

SAT-Solver

BDDs

SAT-Solver

Craig Interpolation

Georg Hofferek

Logic and Computability
Hardware Synthesis: Tool

- **Demiurge**
  - Implements many different techniques
  - Participates in an international synthesis competition
    - Goals: Be fast
      - Produce small implementations
    - Won a Kurt Gödel medal in 2014

Many approaches to try and compare! ➔ You can help!
Administrative Issues

Georg Hofferek
IAIK – Graz University of Technology
georg.hofferek@iaik.tugraz.at
Question Hour(s)

- **Extra Practice Class**
  - Thursday, January 22
  - 05pm, Lecture Hall i2

- **Question Hour**
  - Monday, January 26
  - 9am – 12 noon
  - Lecture Hall i2

---

Mail me questions!
Exam

- Exam
  - Tuesday, January 27
  - 9am – 12 noon
    - Working Time: 2 hours (=120 minutes)
    - Lecture Halls G, D, and ?

- Registration
  - Already open
  - Please register asap!
  - Unregister if you will not come!
Results

- Thursday, January 28 (tentatively)
- Follow @GeorgHofferek for
  - Progress updates on grading
  - Funny incidents

Questions 1 & 2 done. Already seen lots of correct and lots of "creative" answers...
#iaikLuB

Why do people first write a wrong answer, followed by a correct answer, but marking the correct part as "not part of the answer"? #iaikLuB

OMG... How can you write "SAT" in the DPLL table and then tick "UNSAT" in the following multiple-choice question??
*facepalm* #iaikLuB
Check Your Results

- Check of Examination Records ("Einsichtnahme")
  - Friday, January 30 (tent.)
  - Possibly Thursday afternoon already

Check your records!
- Even if you passed
- You’ll learn something
Other Exam Dates

- Written exam in early March
  - Date not fixed yet
  - Last written exam offered by me

- Oral exams
  - Contact me to make an appointment
  - Possible up until I leave TU Graz
    - After that: possibly no exams until October
Course Evaluation

- Starting Today
  - For Practicals & Lecture, separately
  - Please fill both!

- If you were not satisfied…
  - … feedback fosters improvement

- If you were satisfied…
  - … saying so helps to keep the quality up

- Comments

- Course-specific questions
Further Reading

- Douglas Hofstadter: *Gödel, Escher, Bach: An Eternal Golden Braid*
  
  - [http://en.wikipedia.org/wiki/G%C3%B6del,_Escher,_Bach](http://en.wikipedia.org/wiki/G%C3%B6del,_Escher,_Bach)
For the Future

- **Student Project**
  - Bachelor Thesis
  - Master Project
  - Master Thesis
  - (Paid) Summer Internship

- **Course 705.070 & 705.071:**
  - “AK Design and Verification”
  - Summer Term
  - Topic (tent.): Automated Synthesis of Systems
HOW LUCKY I am to have something that makes SAYING GOODBYE so hard.
Thank you!

Image source: http://xkcd.com/1033/