Winter School in Software Engineering 2017

Monday December 11, 2017	**Day 1**
08:00 - 08:30	**Registration**
08:30 - 10:00	Programming by Examples: Applications, Algorithms and Ambiguity Resolution - Session I **Sumit Gulwani, Microsoft Research**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Data Flow Analysis: A Pointer Centric View - Session I **Uday Khedker, IIT Bombay**
12:00 - 12:15	*Short break*
12:15 - 13:00	[Academic Research Talk] - Refinement-based Verification of FreeRTOS in VCC **Sumesh Divakaran, GEC Idukki**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	[Academic Research Talk] - Refinement-based Verification of FreeRTOS in VCC **Sumesh Divakaran, GEC Idukki**
14:45 - 15:00	*Short break*
15:00 - 16:30	Programming by Examples: Applications, Algorithms and Ambiguity Resolution - Session II **Sumit Gulwani, Microsoft Research**
16:30 - 17:00	*Tea/Coffee break*
17:00 - 18:30	Data Flow Analysis: A Pointer Centric View - Session II **Uday Khedker, IIT Bombay**

Tuesday December 12, 2017	**Day 2**
08:30 - 10:00	Data Flow Analysis: A Pointer Centric View - Session III **Uday Khedker, IIT Bombay**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Programming by Examples: Applications, Algorithms and Ambiguity Resolution - Session III **Sumit Gulwani, Microsoft Research**
12:00 - 12:15	*Short break*

12:15 - 13:00	[Industry Talk] - Applying Program Analysis for Model based Design **Prahlad Sampath, Mathworks India**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	[Industry Talk] - Applying Program Analysis for Model based Design **Prahlad Sampath, Mathworks India**
14:45 - 15:00	*Short break*
15:00 - 16:30	Data Flow Analysis: A Pointer Centric View - Session IV **Uday Khedker, IIT Bombay**
16:30 - 17:00	*Tea/Coffee break*
17:00 - 18:30	Programming by Examples: Applications, Algorithms and Ambiguity Resolution - Session IV **Sumit Gulwani, Microsoft Research**

Wednesday December 13, 2017 **Day 3**

08:30 - 10:00	Bounded Model Checking and its Applications - Session I **R Venkatesh, TRDDC Pune**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Software Testing and Debugging: State of the Art and Open Issues - Session I **Alessandro Orso, Georgia Institute of Technology**
12:00 - 12:15	*Short break*
12:15 - 13:00	**Student Tool demonstrations**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	**Student Tool Demonstrations**
14:45 - 15:00	*Short break*
15:00 - 16:30	Bounded Model Checking and its Applications - Session II **R Venkatesh, TRDDC Pune**
16:30 - 17:00	*Tea/Coffee break*
17:00 - 18:30	Software Testing and Debugging: State of the Art and Open Issues - Session II **Alessandro Orso, Georgia Institute of Technology**

Banquet Dinner (19:30 PM *onwards*)
Venue: TRDDC, Foyer

Thursday December 14, 2017	**Day 4**
08:30 - 10:00	Software Testing and Debugging: State of the Art and Open Issues - Session III **Alessandro Orso, Georgia Institute of Technology**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Bounded Model Checking and its Applications - Session III **R Venkatesh, TRDDC Pune**
12:00 - 12:15	*Short break*
12:15 - 13:00	Software Testing and Debugging: State of the Art and Open Issues - Session IV **Alessandro Orso, Georgia Institute of Technology**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	Testing and Debugging with some aspects of Security - Session IV **Alessandro Orso, Georgia Institute of Technology**
14:45 - 15:00	*Short break*
15:00 - 16:30	[Industry Talk] - A Graphical Data Flow Programming Approach to High-performance Computing **Somashekhara Bhaskaracharya, National Instruments India**
16:30 - 17:00	*Tea/Coffee break*
17:00 - 18:30	Bounded Model Checking and its Applications - Session IV **R Venkatesh, TRDDC Pune**

Friday December 15, 2017	**Day 5**
08:30 - 10:00	Introduction to Program Verification using F* - Session I **Aseem Rastogi, Microsoft Research India**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Abstract Interpretation and Program Verification - Session I **Supratik Chakraborty, IIT Bombay**
12:00 - 12:15	*Short break*

12:15 - 13:00	[Industry Talk] - Automated Tracing, Debugging and Repairing Data-Centric Programs **Diptikalyan Saha, IBM Research India**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	[Industry Talk] - Automated Tracing, Debugging and Repairing Data-Centric Programs **Diptikalyan Saha, IBM Research India**
14:45 - 15:00	*Short break*
15:00 - 16:30	Introduction to Program Verification using F* - Session II **Aseem Rastogi, Microsoft Research India**
16:30 - 17:00	*Tea/Coffee break*
17:00 - 18:30	Abstract Interpretation and Program Verification - Session II **Supratik Chakraborty, IIT Bombay**

Saturday December 16, 2017 **Day 6**

08:30 - 10:00	Abstract Interpretation and Program Verification - Session III **Supratik Chakraborty, IIT Bombay**
10:00 - 10:30	*Tea/Coffee break*
10:30 - 12:00	Introduction to Program Verification using F* - Session III **Aseem Rastogi, Microsoft Research India**
12:00 - 12:15	*Short break*
12:15 - 13:00	Abstract Interpretation and Program Verification - Session IV **Supratik Chakraborty, IIT Bombay**
13:00 - 14:00	*Lunch break*
14:00 - 14:45	Abstract Interpretation and Program Verification - Session IV **Supratik Chakraborty, IIT Bombay**
14:45 - 15:00	*Short break*
15:00 - 16:30	Introduction to Program Verification using F* - Session IV **Aseem Rastogi, Microsoft Research India**

Speaker and Talk Details

Speaker : Sumit Gulwani, Microsoft Research

Lecture Topic

Programming by Examples: Applications, Algorithms and Ambiguity Resolution

Lecture Abstract

Programming by Examples (PBE) involves synthesizing intended programs in an underlying domain-specic language (DSL) from example-based specications. PBE is set to revolutionize the programming experience for both developers and end users. It can provide a 10-100x productivity increase for developers in some task domains, and can enable computer users, 99% of whom are non-programmers, to create small scripts to automate repetitive tasks. Two killer applications for this technology include data wrangling (an activity where data scientists today spend 80% time) and code refactoring (an activity where developers spend up to 40% time in a typical application migration scenario).

We will discuss some principles behind designing useful DSLs for program synthesis. A key technical challenge in PBE is to search for programs in the underlying DSL that are consistent with the examples provided by the user. We will discuss a divide-and-conquer based search paradigm that inductively reduces the problem of synthesizing a program with a certain top-level operator to simpler synthesis problems over its sub-programs by leveraging the operator's inverse semantics. Another challenge in PBE is to resolve the ambiguity in the example-based specification. We will discuss two complementary approaches: (a) ranking techniques that can pick an intended program from among those that satisfy the specification, and (b) active-learning based user interaction models. The various concepts will be illustrated using Flash Fill, FlashExtract, and FlashRelate—PBE technologies for data manipulation domains. The Microsoft PROSE SDK allows easy construction of such technologies. We may do a hands-on exercise that will involve building a synthesizer for a small part of the Flash Fill DSL using the PROSE framework.

Time permitting, I will discuss extensions along various dimensions: (i) integrating data-driven ML techniques with logical reasoning to facilitate construction of intelligent systems like program synthesizers, (ii) programming by natural language, and (iii) applications in computer-aided education including feedback generation and problem generation.

Lecture 1: Applications and DSLs for Synthesis
Lecture 2: Algorithms and Ambiguity Resolution
Lecture 3: Hands-on session
Lecture 4: Extensions

Speaker : Uday Khedker, IIT Bombay

Lecture Topic

Data Flow Analysis: A Pointer Centric View

Lecture Abstract

Pointer analysis provides information to disambiguate indirect reads and writes of data through pointers and indirect control flow through function pointers or virtual functions. Its precision influences the precision and scalability of client program analyses significantly. Computationally intensive analyses such as model checking are noted as being ineffective on programs containing pointers, partly because of imprecision of points-to analyses. Thus pointer analysis is a key enabler of precision and efficiency in any analysis.

This tutorial introduces the audience to the world of data flow analysis through the lense of pointer analysis. Although it primarily focuses on the C/C++ language model, the concepts are generic and are applicable to Java too.

We plan to cover the following topics

- Introduction to program analysis through liveness analysis, program model, soundness and precision of program analysis
- Defining live variables analysis, generalization to bit vector frameworks, lattice theoretic modelling, monotonicity, distributivity.
- Importance of pointer analysis, Issues in pointer analysis, points-to analysis: an engineering landscape.
- Modelling flow insensitivity. Flow insensitive points-to analysis. Andersens and Steensgaards methods.
- Flow sensitive points-to analysis, liveness based points-to analysis,
- Modelling context-sensitivity. Issues in interprocedural points-to analysis. Top down and bottom up methods. Value contexts, Procedure Summaries. The influence of pointer analysis on interprocedural analyses and vice-versa (function Pointers, receiver objects of method calls (and its relation to object sensitivity).

Speaker : R Venkatesh, TRDDC Pune

Lecture Topic

Bounded Model Checking and its Applications

Lecture Abstract

The lectures will introduce to the participants a few techniques that apply bounded model checking to prove properties of C programs. Bounded model checkers (BMC) like CBMC analyse only k (the given bound)length paths of a program and report a property as unsatisfiable if it is violated by a run of length k. A BMC is therefore very effective in finding violations if one exists within the given bound but cannot be used to prove the property. In these lectures we will describe a few techniques that transform a program with loops to an abstract program in which loops are either eliminated or replaced with loops that have small known bounds so that a BMC can be used to prove a property of the program. The transformed program is an abstraction w.r.t the property to be proved, which means that if the property holds in the transformed program then it holds in the original too.

The first lecture will introduce bounded model checking of C programs using CBMC as the model checker. The next three lectures will introduce techniques for abstraction of programs with loops and arrays. The second lecture will present ideas to transform loops that do not manipulate arrays to loops with small known bounds that can be analysed by CBMC. The next two lectures will present two different techniques that transform array manipulating loops: one that uses induction and the other that exploits a small model property.

Lecture 1: Introduction to bounded model checking with CBMC
Lecture 2: Applying CBMC to prove properties of programs with simple loops
Lecture 3: Proving properties of programs with arrays using induction
Lecture 4: Proving properties of programs with arrays by using a small model property of the program

Speaker : Alessandro Orso, Georgia Institute of Technology

Lecture Topic

Software Testing and Debugging: State of the Art and Open Issues

Lecture Abstract

Despite decades of work by researchers and practitioners on numerous software verification techniques, testing remains one of the most widely used approaches for assessing and improving software quality. Because of its relevance and cost, software testing (and dynamic verification in general) has been and still is extensively studied, and a countless number of testing techniques have been proposed in the research literature. An activity closely related to testing is software debugging, which consists of locating, understanding, and removing the faults revealed by testing. Also in the case of debugging, researchers have investigated for decades ways of making this task less time consuming and expensive in terms of human effort.

In this module, we will focus on software testing and debugging and (1) provide an overview of the main techniques proposed in the literature, (2) discuss open issues and potential research directions, and (3) present some recent techniques that try to address some of these open issues. Specifically, we will first discuss some fundamental aspects of static and dynamic verification techniques, their differences and their interplay. Second, we will discuss the state of the art and the main open issues in the area of software testing, with particular emphasis on recent approaches based on symbolic analysis. We will then present a historical perspective on software debugging, covering the main research breakthroughs in the area and the most promising current research directions. Finally, we will have a hands-on session in which we will experiment with some of the techniques presented in the previous classes.

Lesson 1: Fundamentals of (Static and Dynamic) Software Verification
Lesson 2: State of the Art and Open Issues in Software Testing
Lesson 3: Software Debugging: Past, Present, and Future
Lesson 4: Hands-on Session

Speaker : Aseem Rastogi, Microsoft Research India

Lecture Topic

Introduction to Program Verification using F*

Lecture Abstract

Software today is ubiquitous and critical. And yet, writing correct and secure software is notoriously hard.

The goal of Program Verification is to help programmers build "correct-by-construction" software. Specifically, it enables programmers to write concise, logical specifications for their programs, and construct rigorous, mathematical proofs that the programs meet their specifications. Over the last decade or so, program verification techniques have been successfully applied to build a fully verified C compiler CompCert), fully verified OS kernels (Verve, CertiKos), with ongoing efforts to verify and deploy the security critical TLS internet transport protocol (the Everest project).

In this tutorial, I will give an introduction to program verification using the F* language (www.fstar-lang.org). The course will touch upon several topics including functional programming, type systems, and logical reasoning. I will not assume any background on these topics, though some mathematical maturity (such as familiarity with proofs by induction) will be good to have. The tutorial will include both a theory component (for the underlying techniques) as well as hands-on experience with verifying programs in F*.

Speaker : Supratik Chakraborty, IIT Bombay

Lecture Topic

Abstract Interpretation and Program Verification

Lecture Abstract

Abstract interpretation provides a mathematically rigorous unifying framework for program analysis and verification. In this series of lectures, we will go over a light-weight introduction to the mathematical foundations of abstract interpretation, and focus on its use in verification of sequential programs. The discussion will include a few common and useful abstract domains used in practice, including relational and non-relational ones. We will also discuss domains for reasoning about both numerical and non-numerical properties of programs.

The paradigm of counterexample-based abstraction refinement (CEGAR) has found spectacular success in proving properties of several complex programs. We will discuss the basic principles of CEGAR, illustrate it with some examples, and discuss its connections with constraint solving techniques.

Time-permitting, we will also cover the basic ideas involved in program analysis/verification with multiple abstract domains – the most successful abstract interpretation based static analysis tools employ a combination of abstract domains to achieve a good precision-performance tradeoff.

Short/Industry Talk Speakers and Details

Speaker : Sumesh Divakaran, GEC Idukki

Talk Title

Refinement-based Verification of FreeRTOS in VCC

Talk Abstract

This talk is about an approach that has been developed by the speaker to verify the functional correctness of a popular open-source real-time operating system called FreeRTOS. FreeRTOS is a real-time kernel meant for use in embedded applications that run on micro-controllers with small to mid-sized memory. FreeRTOS has a large community of users. There are more than 100,000 downloads from SourceForge every year, putting it in the top 100 most-downloaded SourceForge codes.

The scheduling-related functionality of FreeRTOS was verified by applying a theory of refinement. An abstract mathematical model of the scheduling-related functionality was developed in the Z modeling language and this abstract model was then refined to the existing implementation of FreeRTOS, using a sequence of refinement steps. The Program Verification tool VCC was used to verify the refinement conditions in each refinement step. A number of subtle bugs, which cause deviations from the intended behavior were identified and corrected during the verification process. The modified code of FreeRTOS is proved to be functionally correct with respect to the abstract mathematical model.

Speaker : Prahlad Sampath, Mathworks India

Talk Title

Applying Program Analysis for Model based Design

Talk Abstract

You would be learning about many techniques for program analysis as part of this Winter School. As part of my talk, I would like to paint a picture of a few commercial applications of these technologies specifically from the perspective of model-based design using the MATLAB/Simulink tool suite.

I will start with an overview of model-based design and show how it provides an environment where many program analysis technologies can be applied to great effect. I will drive the presentation with demos and examples.

Speaker : Somashekhara Bhaskaracharya, National Instruments India

Talk Title

A Graphical Data Flow Programming Approach to High-performance Computing

Talk Abstract

LabVIEW is a proprietary graphical dataflow programming language, owned by National Instruments and used by scientists and engineers around the world. It is commonly used for implementing control and measurement systems, embedded applications. While the graphical programming interface of LabVIEW makes it intuitive to express parallelism in computational tasks, the dataflow semantics further make it ideally suited for exploiting such parallelism.

This talk will provide an introduction to the LabVIEW language and compiler, the existing support in it for high performance computing as well as the challenges in leveraging the latest compilation techniques for achieving sophisticated program transformations to exploit parallelism and locality.

Speaker : Diptikalyan Saha, IBM Research India

Talk Title

Automated Tracing, Debugging and Repairing Data-Centric Programs

Talk Abstract

Database-centric programs form the backbone of many enterprise systems. Fixing defects in such programs take much human effort due to the interplay between imperative code and database-centric logic. Data-centric programs primarily interact with databases to get collections of content, process each entry in the collection(s), and output another collection or write it back to the database. One or more entries in the output may be faulty. The goal of our work is to automatically localize and repair the bugs causing the incorrect output.

Our approach has three phases. We first gather the execution trace of a faulty program for performing dynamic fault localization analysis. Industry constraints make the trace collection particularly challenging. Trace collection causes considerable overhead to the program execution. Various techniques have addressed this problem by minimizing the number of probes/witnesses used to collect traces. We present a novel distributed trace collection framework wherein, a program is executed multiple times with the same input for different sets of witnesses. The partial traces such obtained are then merged to create the whole program trace. Such divide-and-conquer strategy enables parallel collection of partial traces, thereby reducing the total time of collection.

In the second phase, we use a novel, precise slicing algorithm to break the trace into multiple slices, such that each slice maps to an entry in the output collection. We then compute the semantic difference between the slices that correspond to correct entries and those that correspond to incorrect ones. The “diff” helps to identify potentially faulty statements.

In the third phase, we use a novel data-driven approach for automated repairing of bugs in the selection condition of database statements (e.g., WHERE clause of SELECT statements) a common form of bugs in such data-centric programs. Our key observation is that in real-world data, there is information latent in the distribution of data that can be useful to repair selection conditions efficiently. Given a faulty database program and input data, only a part of which induces the defect, our novelty is in determining the correct behavior for the defect-inducing data by taking advantage of the information revealed by the rest of the data. We accomplish this by employing semi-supervised learning to predict the correct behavior for defect-inducing data and by patching up any inaccuracies in the prediction by a SAT-based combinatorial search. Next, we learn a compact decision tree for the correct behavior, including the correct behavior on the defect-inducing data. This tree suggests a plausible fix to the selection condition. We demonstrate the feasibility of our approach on real-world SAP-ABAP programs.
