List of Topics to be covered (8/29/2015)

(Note ** papers are classics; papers to be presented are indicated by a ## prefix and are listed first in each category,; all IEEE and ACM conferences and journals available through VT library electronic subscription)

---*Foundations of Dataflow Analysis (lectures 1+2)*

CFGs, call graphs, classical dataflow equations, Lattice theory and its relation to dataflow analysis, worklist algorithm as a solution procedure; fixed point iteration and dataflow transfer functions to define a monotone framework,

References:

Matthew S. Hecht, Flow Analysis of Computer Programs, Ch 9, Monotone Dataflow Frameworks** (book is out of print, but will be on reserve in the library on campus)

Thomas Marlowe & Barbara G. Ryder, "Properties of Data Flow Frameworks: A Unified Model", *Acta Informatica*, Vol. 28, pp 121-163, 1990** (a reference work about terms and definitions in dataflow analysis; lots of examples too) available from: http://prolangs.cs.vt.edu/refs/fdns.php

Or any standard compiler textbook, chapter on optimization technology, look at definitions of the classical dataflow problems: reaching definitions, live uses of variables, available expressions

---*Advanced Foundations of Dataflow Analysis (lecture 3)*

dataflow solution safety and what it means; meet-over-all-paths solution and what it means for static analysis;

---*Data dependence and control dependence analysis (lecture 3)*

data dependence and control dependence

Jeanne Ferrante, Karl Ottenstein, & Joe Warren, "The Program Dependence Graph and its Use in Optimization", ACM TOPLAS vol 9, no 3, July 1987 ** (first sections of the paper have good definitions of data and control dependence. PDG was used to parallelize programs)

S. Horwitz, T. Reps, & D. Binkley, "Interprocedural Slicing Using Dependence Graphs", PLDI 1988, pp 35-46. (same authors and title, ACM TOPLAS, vol 12, no 1, Jan 1990, pp 26-60).

---*Reference Analysis and Call Graph Construction for Statically Typed OOPLs (e.g., Java, C++)*

Type-based techniques – Class Hierarchy Analysis (CHA), Rapid Type Analysis (RTA)

##Jeff Dean, Dave Grove, & Craig Chambers, "Optimization of object-oriented Program using Static Class Hierarchy Analysis", ECOOP 1995, pp 77-101.** (CHA)

##David Bacon & Peter Sweeney, "Fast Static Analysis of C++ Virtual Function Calls", OOPSLA 1996, pp 324-341,(RTA) **

---*Flow-based techniques – flow-insensitive, context-insensitive approach to points-to analysis (i.e., reference analysis)*

##Ana Milanova, Atanas Rountev, Barbara G. Ryder, "Points-to Analysis for Java Using Annotated Constraints", OOPSLA 2001.

Other sources:

Lars Anderson, "Program Analysis and Specialization for the C Programming Language", Ph.D. thesis, 1994**

M. Sridharan and S.F. Fink, "The Complexity of Andersen's Analysis in Practice", SAS 2009

Bjarne Steensgaard, "Points-to Analysis in Almost Linear Time", POPL 1996, pp32- 41.(C pointers)

V. Sundaresen et al. "Practical Virtual Method Call Resolution for Java", OOPSLA 2000, pp 264-279.

O. Lhotak & L. Hendren, "Scaling Java Points-to Analysis using SPARK", CC 2003

---*Adding calling context sensitivity (Java)*

Reference: M. Sharir & A. Pneuli, "Two Approaches to Interprocedural Dataflow Analysis", pp 189-231, in Program Flow Analysis – Theory and Applications, edited by Steve Muchnick and Neil Jones.***

##Dave Grove & Craig Chambers, "Call graph Construction in OO Languages", OOPSLA 1997.; D. Grove & C. Chambers, "A Framework for Call Graph Construction Algorithms", ACM TOPLAS, vol 23 no 6, pp 685-7467, Nov 2001.

##A. Milanova, A. Rountev, B. G. Ryder, "Parameterized Object Sensitivity for Points-to Analysis for Java", ISSTA 2002. (Reference: A. Milnova, A. Rountev, B.G. Ryder, "Parameterized Object Sensitivity for Points-to Analysis for Java", ACM Transactions on Software Engineering Methodology, vol 14, no 1, pp 1-414. January 2005**)

O. Lhotak & L. Hendren, "Context-sensitive Points-to Analysis: is it worth it?", CC 2006

Other sources:

M.Sridharan & R. Bodik, "Refinement-based Context-sensitive Points-to Analysis for Java", PLDI 2006

Y. Smaragdakis, M. Bravenboer, & O. Lhotak, "Pick your contexts well: understanding object sensitivity", POPL 2011

G. Kastrinis, Y. Smaragdakis, "Hybrid Context Sensitivity for Points-to Analysis", PLDI 2013

---*Reference Analysis and Call Graph Construction for JavaScript (an example of a dynamically typed language, related to webpage information flow security problems) Dealing with dynamic code construction, dynamic object types, objects whose properties can vary at runtime (adds and deletes of properties are possible)*

1st set of papers to present:

##S. H. Jensen, A. Moeller, & P Thiemann, "Type Analysis for JavaScript" in SAS 2009.

##M. Sridharan, J. Dolby, S. Chandra, M. Schaefer, F. Tip, "Correlation Tracking for Points-to Analysis of JavaScript", ECOOP 2012.

##V. Kashyap et al, "JSAI: A Static Analysis Platform for JavaScript", FSE 2014

2nd set of papers to present:

##M. Madsen, B. Livshits, M. Fanning, "Practical Static Analysis of JavaScript Applications in the Presence of Frameworks and Libraries", ESEC/FSE 2013

##Shiyi Wei and Barbara G. Ryder, "State-sensitive Points-to Analysis for the Dynamic Behavior of JavaScript Objects", *Proceedings of the European Conference on Object-oriented Programming (ECOOP),* July, 2014.

##Y. Smaragdakis, G. Kastrinis, G. Balatsouras, "Introspective Analysis: Context-sensitivity, Across the Board", PLDI 2014. (Java)

Other soures:

R. Chugh, JA. Meister, R. Jhala, & S. Lerner, " Staged Information Flow for JavaScript", PLDI 2009.

J. H. Jensen, P. A. Jonsson, & A. Moeller, "Remedying the Eval That Men Do" , ISSTA 2012.

V. Kashhyap et al, "Type Refinement for Static Analysis of JavaScript", DLS 2013

Hybrid analysis approaches – applying mixes of different sorts of context sensitivity during analysis

V. Kashyap et al, "JSAI: A Static Analysis Platform for JavaScript", FSE 2014

Shiyi Wei and Barbara G. Ryder, "Adaptive Context-sensitive Analysis for JavaScript", *Proceedings of the European Conference on Object-oriented Programming (ECOOP),* July, 2015. (JavaScript)

---*Dynamic analysis – profiling and sampling*

##Ball, Thomas, and James R. Larus. *Efficient path profiling*. Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture. pp. 46-57, 1996.**
(selective instrumentation, profiling procedural acyclic paths)

##James R. Larus, "Whole Program Path", PLDI 1999**

##Richards, G., Lebresne, S., Burg, B., & Vitek, J. *An analysis of the dynamic behavior of JavaScript programs*. PLDI, pp. 1-12, 2010 (an empirical study of the dynamic behavior of JavaScript programs using JS tracing, focusing on the dynamism of this language; tool used: instrumented version of WebKit, tracingSafari, traceAnalyzer)

Other sources:

Arnold, Matthew, and Barbara G. Ryder. *A framework for reducing the cost of instrumented code*. PLDI pp. 168-179, 2001.(code-duplication and counter-based instrumentation sampling)

Liblit, B., Aiken, A., Zheng, A. X., & Jordan, M. I. *Bug isolation via remote program sampling*. PLDI, pp. 141-154, 2003. (statistical sampling using sampled instrumentation)

J. Clause, W. Li, a& Al Orso, "DYTAN: A Generic Dynamic Taint Analysis Framework", ISSTA 2007.

P. Ratanaworabhan, B. Livshits, & B. Zorn, "JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications", USENIX Conf on WinApp Development, 2010.

G. Richards et al, " The Eval that Men Do", ECOOP 2011 (dynamic study of Javascript codes).

F. Meawad, G. Richzrdds, F. Morandat & J. Vitek, "Eval Begone! Semi-automatic Removal of Eval for JavaScript Programs", OOPDLA 2012

Wei, S., Xhakaj, F., & Ryder, B. G. *Empirical study of the dynamic behavior of JavaScript objects*. Software: Practice and Experience, 2015. (more focused empirical study on the dynamism of JS with respect to the objects used in JS programs, using traces of JS executions; tools used: tracingSafari, traceAnalyzer)

Zhang, Xiangyu, and Rajiv Gupta. *Whole execution traces and their applications*. ACM Transactions on Architecture and Code Optimization (TACO). pp. 301-334, 2005. (whole-execution tracing that records fine-grained runtime artifacts including control flow, values, addresses, and data/control

dependence histories; the trace is represented by a labeled graph)

---*Static and Dynamic Slicing techniques – related to dependences*

Static:

##Mark Weiser, *Program Slicing*. TSE, 1984; Mark Weiser, "Programmers use Slices when Debugging", CACM vol 25, no 7, pp 446-452, July 1982.** (the original static slicing, intraprocedurally precise but interprocedurally imprecise because it does not model calling contexts)

##M. Sridharan, S. Fink, R. Bodik, "Thin Slicing", PLDI 2007.

Other sources:

Frank Tip, "A Survey of Program Slicing Techniques", Journal of Programming Languages, 1995.

Gupta, R., Soffa, M. L., & Howard, J. *Hybrid slicing: integrating dynamic information with static analysis*. TOSEM,1997. (hybrid slicing, breakpoints + dynamic call graph + static slice)

D. Binkley, N. Gold, M. Harman, "An Empirical Study of static program slice size", TOSEM, vol 16, no 2, 2007.

Noah Johnson, Juan Caballero, Kevin Zhijie Chen, Stephen McCamant, Pongsin Poosankam, Daniel Reynaud, and Dawn Song. "Differential Slicing: identifying causal execution differences for security applications", *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, page(s): 347-362. May 2011.

Dynami:c
##Zhang, Xiangyu, Rajiv Gupta, and Youtao Zhang. *Precise dynamic slicing algorithms*. ICSE, 2003.(best paper award, trace-based dynamic slicing with enhanced precision; three algorithms of same precision but different efficiency)

Other sources:

H. Agrawal & J. Horgan, "Dynamic Program Slicing", PLDI 1990. (the original dynamic slicing, three algorithms of different degrees of precision and costs)

Gyimóthy, Tibor, Árpád Beszédes, and István Forgács. *An efficient relevant slicing method for debugging.* FSE, 1999. (relevant slicing, dynamic slice + statements that actually did not affect the variable but could have affected it had they been evaluated differently)

Zhang, Xiangyu, and Rajiv Gupta. *Cost effective dynamic program slicing*. PLDI, 2004. (improving efficiency of dynamic slicing via optimizing dynamic dependence graph on which dynamic slices are computed; this seems to be the fastest backward computation algorithm for backward dynamic slicing)

X. Zhang, R. Gupta, & N. Gupta "Locating Faults through Automated Predicate Switching", ICSE 2006.

Specialized applications of slicing:

DeMillo, R. A., Pan, H., & Spafford, E. H. *Critical slicing for software fault localization*. ISSTA, 1996. (critical slicing, deletion-based slicing, including critical statements in slice only --- a statement is critical if deleting it results in changed observed behaviour for the slicing criterion)

Binkley, D., Gold, N., Harman, M., Islam, S., Krinke, J., & Yoo, S. *Orbs: Language-independent program slicing.* FSE, 2014. (observation-based slicing multiple-language programs --- e.g., programs including C, Java, and Python code)

Beszédes, Á., Faragó, C., Szabó, Z. M., Csirik, J., & Gyimóthy, T. Union slices for program maintenance. ICSM, 2002. (union slicing, unionizing traditional dynamic slices of multiple inputs, combined with static slices for software maintenance tasks)

Hall, Robert J. *Automatic extraction of executable program subsets by simultaneous dynamic program slicing*. Automated Software Engineering, 1995.
(simultaneous dynamic slicing, computing dynamic slices of multiple inputs simultaneously to address the unsoundness/incorrectness of union slicing)

---*Information flow -- integrity & confidentiality of information (static and dynamic approaches)*

##D. Denning & P.J. Denning, "Certification of Programs for Secure Information Flow", CACM jluy 1997, vol 20, no 7. **

##A. Sabelfeld & A. Myers, "Language-based Information-flow Security", IEEE Journal on Selected Areas of Communication, Vo 21, No 1, Jan 2003**

##W. G. Halfond & A. Orso, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-aware Evaluation", IEEE TSE, Vol 34, no 1, jan/feb 2008.

##O. Tripp et al. "TAJ: Effective Taint Analysis of Web Applications", PLDI 2009

Other Sources:

D. Wagner & D. Dean, "Intrusion Detection Via Static Analysis", Oakland Conf 2001.

W. Kleiber et al, "Android Taint Glow Analysis for App Sets", SOAP 2014???

Damien Octeau, Daniel Luchaup, Matthew Dering, Somesh Jha, and Patrick McDaniel. "Composite Constant Propagation: Application to Android Inter-Component Communication Analysis.", *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, May 2015.

W. Halfond, A. Orso, & P. Manolios, "Using Positive Tainting and Syntax-aware Evaluation to Protect Web Applications", FSE 2006.

J. Clause, W. Li, & Alex Orso "Dytan" A Generic Dynamic Taint Analysis Framework", ISSTA 2007.

J. Clause & A. Orso, "PENUMBRA: Automatically identifying Failure-relevant Inputs Using Dynamic Taint", ISSTA 2009.

---*Android security – dealing with a Java-family implementation language*

##Arzt, Steven, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps*. PLDI, pp. 259-269, 2014.
(taint analysis framework for android apps)

##Feng, Y., Anand, S., Dillig, I., & Aiken, A. *Apposcopy: Semantics-based detection of android malware*

through static analysis. FSE, pp. 576-587, 2014. (identifying Android malware that steals private user information, using static taint analysis and Inter-component call graph)

##William Enck et al., "TaintDroid: An Informaiotn –flow tracking System for Realtime Privacy Monitoring on Smartphones", OSDI, 2010.

##Dynamic Detection of Inter-application Communication Vulnerabilities in Android. Roee Hay, Omer Tripp and Marco Pistoia (IBM, Israel; IBM Research, USA), ISSTA 2015.

##S. Yang, D. Yan, H. Wu, Y, Wang, A. Rountev, "Static Control-flow Analysis of User-driven Callbacks in Android Applications", ICSE 2015.

Other sources:

Li, Li, Alexandre Bartel, Tegawendé François D. Assise Bissyande, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. *IccTA: detecting inter-component privacy leaks in android apps*. ICSE. 2015. (static taint analyzer detecting privacy leaks during ICC in android apps)

Grace, M. C., Zhou, Y., Wang, Z., & Jiang, X. *Systematic Detection of Capability Leaks in Stock Android Smartphones*. NDSS, 2012. m(known as Woodpecker, accidental capability leak detection for android apps, not implemented in a program-analysis framework but a tool-based approach mixing Java, Shell, and Python code atop an off-the-shelf dissembler; a forward context-insensitive analysis)

Lu, Long, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. *Chex: statically vetting android apps for component hijacking vulnerabilities*. CCS, pp. 229-240. 2012.(similar to Woodpecker in aims but implemented on WALA, context-sensitive forward analysis)

Octeau, D., McDaniel, P., Jha, S., Bartel, A., Bodden, E., Klein, J., & Le Traon, Y. *Effective inter-component communication mapping in android with Epicc*: *An essential step towards holistic security analysis*. USENIX Security, *2013*. (detecting ICC call targets in android apps)

'Chin, Erika, Adrienne Porter Felt, Kate Greenwood, and David Wagner. *Analyzing inter-application communication in Android*. MobiSys, pp. 239-252. 2011. (detecting ICC vulnerabilities based on flow-sensitive interprocedural static analysis on translated Dex code -- using existing dissembler dedexer)

A. Rountev & D. Yan, "Static Reference Analysis for GUI Objects in Android Software, CGO 2014.

Scalable and Precise Taint Analysis for Android. Wei Huang, Yao Dong, Ana Milanova and Julian Dolby (Google, USA; Rensselaer Polytechnic Institute, USA; IBM Research, USA) , ISSTA 2015.

---*Unsafe analysis techniques (e.g*., Symbolic execution and Concolic Testing)

Concolic Testing

##J. King, "Symbolic Execution and Program Testing", CACM 1976**

##P. Godefroid, Nils Klarlund, K. Sen "DART: Directed Automated Random Testing", PLDI 2005

##Cadar et al, "Symbolic Execution for Software Testing in Practice – a Preliminary Assessment", ICSE 2011. C. Cadar & K. Sen, "Symbolic Execution for Software Testing: Three Decades Later", CACM, Feb 2013, p 82-90. Read together as they seem very related.

Other sources:

K. Sen, D. Marinov, G. Agha, "CUTE: A Concolic Testing Engine for C", FSE 2006?

R. Majumdar, K. Sen, "Hybrid Concolic Testing" ICSE 2007

L. Clarke & D. Richardson, "Symbolic Evaluation Methods for Program Analysis", pp 264-299, in Program Flow Analysis – Theory and Applications, edited by Steve Muchnick and Neil Jones.

P. Godefroid, "Compositional Dynamic Test Generation", POPL 2007,pp 47-54.

Combinations of dynamic and static analyses and soundyness.

##B. Dufour, B.G. Ryder, and G. Sevitsky,"A Scalable Technique for Characterizing the Usage of Temporaries in Framework-intensive Java Applications", *FSE,* November 2008, pp 59-70. (Java enterprise software to statically find bloat)

##S. Wei and B.G. Ryder, "Practical Blended Taint Analysis for JavaScript", in *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pp 336-346, doi:10.1145/2483760.2483788, July 2013.

##O. Tripp, P. Ferrara, M. Pistoia, "Hybrid Security Anlaysis of Web JavaScript Code via Dynamic Partial Evaluation, ISSTA 2014.

##Ben Livshits et al., "Viewpoint: In Defense of Soundiness: A Manifesto", CACM vol 58, no 2, Feb 2015.

Other sources:

B. Dufour, B.G. Ryder, and G. Sevitskym "Blended Analysis for Performance Understanding of Framework-based Applications", in the *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, July 2007, pp 118-128.

Prateek Saxena, Devdatta Akhawe, Steve Hanna, Stephen McCamant, Feng Mao, Dawn Song. "A Symbolic Execution Framework for JavaScript", *Proceedings of IEEE Symposium on Security and Privacy*, page(s): 513-528. May 2010. (bytecode level)

--- *Change impact analysis & program understanding –a SE client of dynamic and static analyses*

##Ren, X., Shah, F., Tip, F., Ryder, B. G., & Chesley, O. *Chianti: a tool for change impact analysis of java programs*. OOPSLA, 2004. (descriptive test-case-level static impact analysis, finding impacted and impacting test cases based on change-type information)

##Law, J., & Rothermel, G. *Whole program path-based dynamic impact analysis*. ICSE, 2003. (predictive dynamic method-level impact analysis based on method-execution order)

##Orso, A., Apiwattanapong, T., & Harrold, M. J. *Leveraging field data for impact analysis and regression testing*. FSE, 2003. (predictive coverage-based dynamic method-level impact analysis; intersect approximate method-level static forward slice and method-level coverage)

M. Kim, D. Notkin, D. Grossman, "Automatic Inference of Structural Changes for Matching Acrss Program Versions", ICSE 2007.
L. Zhang, M. Kim, S. Khurshid, "Localizing Failure-inducing Program edits Based on Spectrum Information", ICSM 2011.

Other sources:

Badri, L., Badri, M., & St-Yves, D. *Supporting predictive change impact analysis: a control call graph based technique*. APSEC, 2005. (predictive static impact analysis based on call graph)

Ramanathan MK, Grama A, Jagannathan S. *Sieve: a tool for automatically detecting variations across program versions*. ASE, 2006. (descriptive statement-level dynamic impact analysis using longest-common-sequence based execution history differencing)

Gethers, M., Dit, B., Kagdi, H., & Poshyvanyk, D. *Integrated impact analysis for managing software changes*. ICSE, 2012. (change-request-level impact analysis combining information retrieval, dynamic analysis, and data mining techniques; can work in both predictive and descriptive settings, depending on the classes of information provided)

Garcia, J., Popescu, D., Safi, G., Halfond, W. G., & Medvidovic, N. *Identifying message flow in distributed event-based systems*. FSE, 2013. (static analysis of distributed event-based systems based on message interface type matching)

Kim, M., Notkin, D., Grossman, D., & Wilson Jr, G. *Identifying and summarizing systematic code changes via rule inference*. TSE, 2013. (rule inference based program differencing, potentially suitable for descriptive impact analysis)