

# Automated Verification of Operating System Code

— A Literature Review —

Jan Tobias Mühlberg  
<[muehlber@cs.york.ac.uk](mailto:muehlber@cs.york.ac.uk)>

York, December 19, 2005

# Motivation

- software defects are costly
- likewise is the process of revealing, locating and correcting them
- pure testing doesn't establish much trust in the code since it's never exhaustive
- having a trustworthy operating system (OS) is important since all application software needs to rely on it

# Outline

1. Overview on Software Verification
2. Specific Requirements for Verifying Operating Systems
3. Recent Approaches in OS Verification
4. High Level Languages versus Object Code or Intermediate Code

# 1. Software Verification

- strong method of establishing correctness of a program
- shows that the program satisfies its specification by providing a mathematical proof or by exhaustive search of its state space
- approaches mainly focus on small applications in highly safety-critical domains

# 1. Software Verification

- **Implicit Model Checking** (Bérard et al., 2001)
  - uses symbolic representation of the state space of a target program
  - prove temporal properties of the system by exhaustive search on the state space
  - produce counter examples and related paths in the program

# 1. Software Verification

- **Abstract Interpretation** (Cousot and Cousot, 2002)
  - semantical approximation used for semantics-based program analysis
  - can be used for debugging, optimisation and correctness proofs
  - basis for constructing abstractions of programs and discover predicates

# 1. Software Verification

- **Abstraction and Predicate Discovery**
  - state explosion is a problem of model checking; can be tackled by abstraction ([Das et al., 1999](#))
  - abstract systems need to be refined on demand ([Clarke et al., 2003](#))
  - automatic predicate discovery is based on detection of program invariants; verification requires manual definition of properties ([BLAST, 2005](#))

## 2. Problems with OS

- in general difficult to develop and to test
- high concurrency and no save operating environment ([Corbet et al., 2005](#))
- most serious bugs are found in device drivers ([Chou et al., 2001](#))
- main sources of bugs: improper handling of locks and dereferencing of invalid pointers ([Chou et al., 2001](#))

## 3. Approaches in OS Verification

- **SLAM/SDV**
  - industry-strength tools for checking API compliance of Windows device drivers  
([Microsoft Corporation, 2004](#))
  - uses predicate abstraction to derive a boolean program from C code  
([Ball et al., 2001](#)), ([Ball and Rajamani, 2001b](#))
  - applies BDD based model checking and abstraction refinement ([Ball and Rajamani, 2001a](#))

## 3. Approaches in OS Verification

- **BLAST**

- similar approach as SLAM: generate instrumented program from C code; compute abstraction; model check and refine the abstraction ([Henzinger et al., 2002a](#))
- not restricted to Windows code; has been applied to Linux drivers as well ([Henzinger et al., 2002a](#)), ([Beyer et al., 2005](#))
- uses Lazy Abstraction algorithm ([Henzinger et al., 2002b](#))

## 3. Approaches in OS Verification

- **VFiasco**
  - attempt to verify the Fiasco micro kernel OS ([Hohmuth and Tews, 2005](#))
  - pragmatic: using semantic analysis, theorem proving, and model checking ([Hohmuth and Tews, 2005](#)), ([Endrawaty, 2005](#))
  - provide semantics for goto and computed jumps in C++ ([Tews, 2004](#))

## 4. What language?

- high level languages usually have no formally defined semantics and most verification approaches support only subsets of the language  
([Beyer et al., 2005](#)), ([Tews, 2004](#)), ([Mauborgne, 2004](#)), ([Hohmuth and Tews, 2003](#))
- pointer analysis (esp. finding aliases and dealing with arithmetic) is difficult to perform for real-world applications  
([Beyer et al., 2005](#)), ([Wilson and Lam, 1995](#))

## 4. What language?

- dealing with aliases and pointer arithmetics is easier on object code or three address code  
([Gupta and Sharma, 2003](#)), ([Fernández and Espasa, 2002](#)), ([Debray et al., 1998](#))
- object code has a clearly defined semantic that can be used for verification ([Wahab, 1998](#))
- proving the correctness of the program in source code does not automatically prove the correctness of the compiled program for any platform  
([Buttle, 2001](#)), ([Wahab, 1998](#))

Thank you!

# References

- Ball, T., Majumdar, R., Todd Millstein, T., and Rajamani, S. K.: 2001, Automatic predicate abstraction of c programs, in *PLDI '01: Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation, Snowbird, Utah, USA*, pp 203 – 213, ACM Press, New York, NY, USA
- Ball, T. and Rajamani, S. K.: 2001a, Automatically validating temporal safety properties of interfaces, in *SPIN '01: Proceedings of the 8th international SPIN workshop on Model checking of software, Toronto, Ontario, Canada*, pp 103 – 122, Springer-Verlag New York, Inc., New York, NY, USA
- Ball, T. and Rajamani, S. K.: 2001b, The slam toolkit, in *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*, pp 260–264, Springer-Verlag, London, UK
- Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebeln, P., and McKenzie, P.: 2001, *Systems and Software Verification*, Springer-Verlag, Berlin and Heidelberg, Germany
- Beyer, D., Henzinger, T. A., Jhala, R., and Majumdar, R.: 2005, Checking memory safety with blast, in M.Cerioli (ed.), *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE 2005), Edinburgh, April 2-10, volume 3442 of Lecture Notes in Computer Science*, pp 2 – 18, Springer-Verlag, Berlin, Germany
- BLAST: 2005, *BLAST User's Manual*, Available online at <http://embedded.eecs.berkeley.edu/blast/doc/>; visited 14 December 2005.
- Buttle, D. L.: 2001, *Verification of Compiled Code, Ph.D. thesis*, University of York
- Chou, A., Yang, J., Chelf, B., Hallem, S., and Engler, D. R.: 2001, An empirical study of operating system errors, in *Symposium on Operating Systems Principles*, pp 73 – 88
- Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H.: 2003, *Journal of the ACM* **50(5)**, 752

- Corbet, J., Rubini, A., and Kroah-Hartmann, G.: 2005, *Linux Device Drivers*, O'Reilly, Sebastopol, CA, USA, 3rd edition
- Cousot, P. and Cousot, R.: 2002, On abstraction in software verification, in E. Brinksma and K. Larsen (eds.), *Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002), Copenhagen, Denmark*, Vol. 2404 of *Lecture Notes in Computer Science*, pp 37 – 56, Springer-Verlag, Berlin, Germany
- Das, S., Dill, D. L., and Park, S.: 1999, Experience with predicate abstraction, in *CAV '99: Proceedings of the 11th International Conference on Computer Aided Verification*, pp 160 – 171, Springer-Verlag, London, UK
- Debray, S., Muth, R., and Weippert, M.: 1998, Alias analysis of executable code, in *POPL '98: Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, San Diego, California, United States*, pp 12 – 24, ACM Press, New York, NY, USA
- Endrawaty: 2005, Verification of the fiasco ipc implementation, Thesis report, Dresden University of Technology
- Fernández, M. and Espasa, R.: 2002, Speculative alias analysis for executable code, in *PACT 2002: Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques*, pp 222 – 231, IEEE Computer Society, Washington, DC, USA
- Gupta, S. K. and Sharma, N.: 2003, Alias analysis for intermediate code, in *Proceedings of the First Annual GCC Developers' Summit, May 25 – 27, 2003, Ottawa, Canada*, Available online at <http://www.linux.org.uk/~ajh/gcc/gccsummit-2003-proceedings.pdf>; visited 3rd November 2005.
- Henzinger, T. A., Jhala, R., Majumdar, R., Necula, G. C., Sutre, G., and Weimer, W.: 2002a, Temporal-safety proofs for systems code, in *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pp 526 – 538, Springer-Verlag, London, UK
- Henzinger, T. A., Jhala, R., Majumdar, R., and Sutre, G.: 2002b, Lazy abstraction, in *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, Portland, Oregon*, pp 58 – 70, ACM Press, New York, NY, USA
- Hohmuth, M. and Tews, H.: 2003, The Semantics of C++ Data Types: Towards Verifying low-level System Components, in *Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics*

- Hohmuth, M. and Tews, H.: 2005, The VFiasco approach for a verified operating system, in *Proceedings of the 2nd ECOOP Workshop on Programm Languages and Operating Systems (ECOOP-PLOS'05)*, Available online at <http://wwwtcs.inf.tu-dresden.de/~tews/Plos-2005/ecoop-plos-05-a4.ps>; visited 3rd November 2005.
- Mauborgne, L.: 2004, ASTRÉE: Verification of absence of run-time error, in R. Jacquart (ed.), *Building the information Society (18th IFIP World Computer Congress)*, pp 384 – 392, The International Federation for Information Processing, Kluwer Academic Publishers
- Microsoft Corporation: 2004, *Static Driver Verifier: Finding Bugs in Device Drivers at Compile-Time*, Website, Available online at <http://www.microsoft.com/whdc/devtools/tools/SDV.aspx>; visited 3rd November 2005.
- Tews, H.: 2004, *Verifying Duff's device – A simple compositional denotational semantics for Goto and computed jumps*, Technical report, Dresden Technical University, Germany
- Wahab, M.: 1998, *Object Code Verification*, Dissertation, University of Warwick, UK
- Wilson, R. P. and Lam, M. S.: 1995, Efficient context-sensitive pointer analysis for c programs, in *PLDI '95: Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation, La Jolla, CA, USA*, pp 1 – 12, ACM Press, New York, NY, USA