Tallinn University of Technology
Department of Computer Science

Urmas Repinski

Ph.D. Dissertation Defense

Tallinn, 2016

Ph.D. Dissertation Title

Model-Based Error Localization and Mutation-Based Error Correction Algorithms and their Implementation for C Designs

Ph.D. Supervisors

- prof. Tanel Tammet
  - Department of Computer Science, Tallinn University of Technology

- dots. Juhan-Peep Ernits
  - Department of Computer Science, Tallinn University of Technology

- prof. Jaan Raik
  - Department of Computer Engineering, Tallinn University of Technology

Ph.D. Opponents

- prof. Erik G. Larsson
  - Department of Department of Electrical Engineering, Linköping University
- prof. Varmo Vene
  - Institute of Mathematics and Statistics, University of Tartu

List of Publications (1)

- Raik, Jaan; **Repinski, Urmas;** Hantson, Hanno; Jenihhin, Maksim; Di Guglielmo, Giuseppe; Pravadelli, Graziano; Fummi, Franco (2012). **Combining Dynamic Slicing and Mutation Operators for ESL Correction.** 17th IEEE European Test Symposium (1−6). IEEE.
- **Repinski, Urmas;** Raik, Jaan (2012). **Comparison of Model-Based Error Localization Algorithms for C Designs.** Proc. of 10th East-West Design & Test Symposium, Kharkov, Ukraine, September 14-17, 2012 (1−4). IEEE Computer Society Press.
- Hantson, H.; **Repinski, U.;** Raik, J.; Jenihhin, M.; Ubar, R. (2012). **Diagnosis and Correction of Multiple Design Errors Using Critical Path Tracing and Mutation Analysis.** In: 13th IEEE Latin-American Test Workshop Proceedings (27−32). IEEE Computer Society Press.

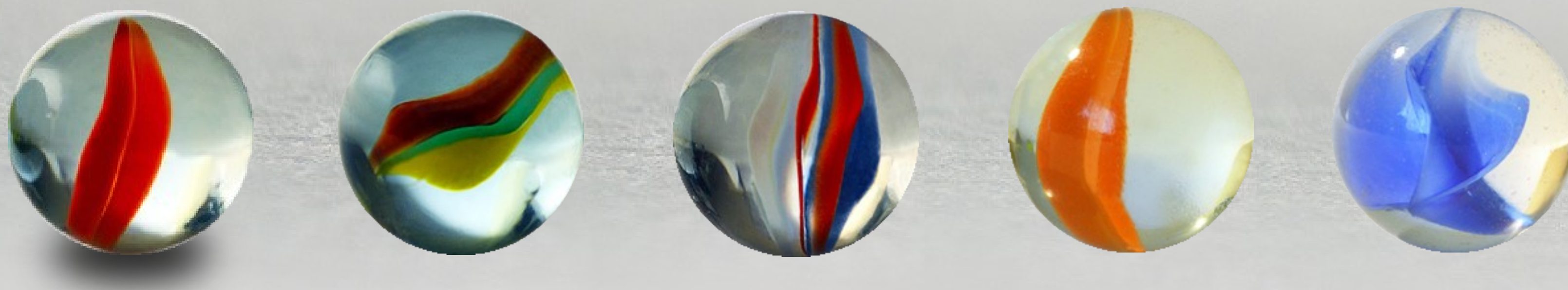
List of Publications (2)

- Raik, Jaan; **Repinski, Urmas;** Jenihhin, Maksim; Chepurov, Anton (2011). **High-Level Decision Diagram Simulation for Diagnosis and Soft-Error Analysis.** Design and Test Technology for Dependable Systems-on-Chip (294−309). IGI Publishing.
- Raik, Jaan; **Repinski, Urmas;** Ubar, Raimund; Jenihhin, Maksim; Chepurov, Anton (2010). **High-level design error diagnosis using backtrace on decision diagrams.** 28th Norchip Conference 15-16 November 2010, Tampere, Finland, IEEE.

Other Related Publications

- **Урмас Репинский** (2012), **Верификация на основе симуляции с нахождением и исправлением ошибок для с-дизайнов,** Програмные продукты и системы, № 100, 2012 год, стр. 229-237
- **Urmas Repinski** (2012). **Model-based Verification with Error Localization and Error Correction for C Designs,** Програмные продукты и системы (international journal Software and Systems), № 100, 2012, pp. 221-229.
- Bloem, Roderick; Drechsler, Rolf; Fey, Goerschwin; Finder, Alexander; Hofferek, Georg; Koenighofer, Robert; Raik, Jaan; **Repinski, Urmas;** Suelflow, Andre (2012). **FoREnSiC - An Automatic Debugging Environment for C Programs,** Haifa Verification Conference (HVC'2012), pp 1-6, IBM Research Labs, Haifa, Israel: IBM.

Outline



Introduction

- **Problem Formulation**
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

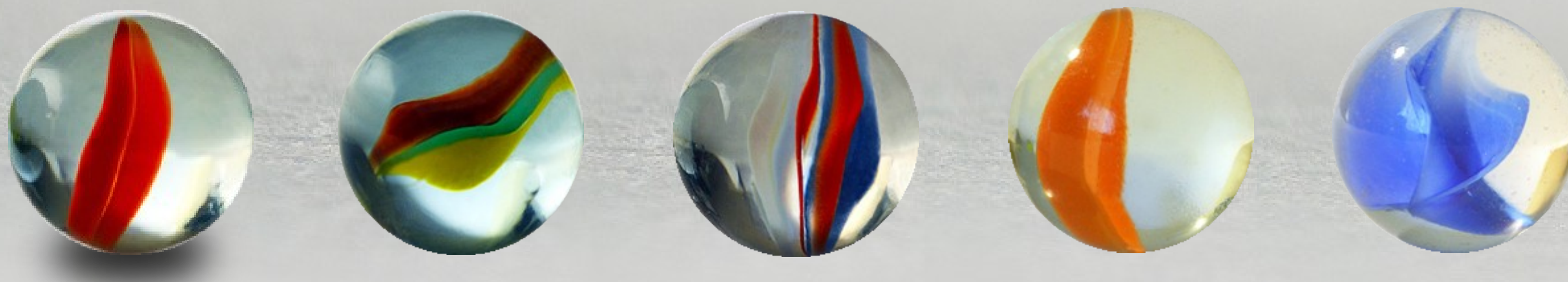
Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Problem Formulation

- “Black Box” → “White Box”
- The Model
- Algorithms (and Mutation-Based Error Correction)
  - Model-Based Error Localization
  - Mutation-Based Error Correction
- Specification Format
  - Vera
  - E
  - Java
  - C/C++

Outline



Introduction

- Problem Formulation
- **Requirements for the tool**
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

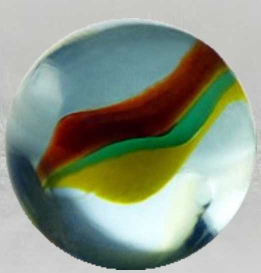
Requirements for the Tool

- Organized and Structured
  - Documentation:
    - Specification
    - Flow Graphs
    - Publications
  - Tutorial
- Should Process any kind of Design
- Inputs and Outputs in Clear Format
- Should have a name - “FORENSIC” tool

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Contribution and Novel Approaches (1)

- Novel Rankings for Model-Based Error Localization
  - Consecutive
  - Simple
- Compiler's functionality for
  - Model's simulation
  - Specification animation
  - Dynamic Slicing Algorithm's Implementation
- Number of applied mutations instead of a number of statements as a measure of the error localization accuracy.

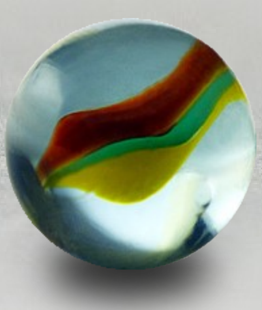
Contribution and Novel Approaches (2)

- Implementation of
  - Model-Based Error Localization Algorithm
  - Dynamic Slicing Algorithm
  - Mutation-Based Error Correction Algorithm
- Specification Format
- Tool can be used in the process of Software Development.

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- **The Model**
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

The Model (1)

- Valid processed Design's “White Box”
- Is Hammock Graph, a special case FlowGraph:

Definition 1: *Digraph* is a directed graph structure $\langle N, E\rangle$, where N is a set of *nodes and edges* in $N \times N$. If (n, m) is in E then n is an *immediate predecessor* of m and m is an *immediate successor* n.

Definition 2: *Flow graph* is a structure $<N, E, n_0>$, where $<N, E>$ is a digraph, n_0 is in N and there exists a path from n_0 to all other edges in N. n_0 is called *the initial node*.

Definition 3: **_Hammock graph_** is a structure $H=<N, E, n_0, n_e>$ where both $<N, E, n_0>$ and $<N, E^{-1}, n_e>$ are *flow graphs*. Note that, as usual, $E^{-1}=\{(a,b)|(b,a) \text{ is in } E\}$. n_e is called *the end node*. [28]

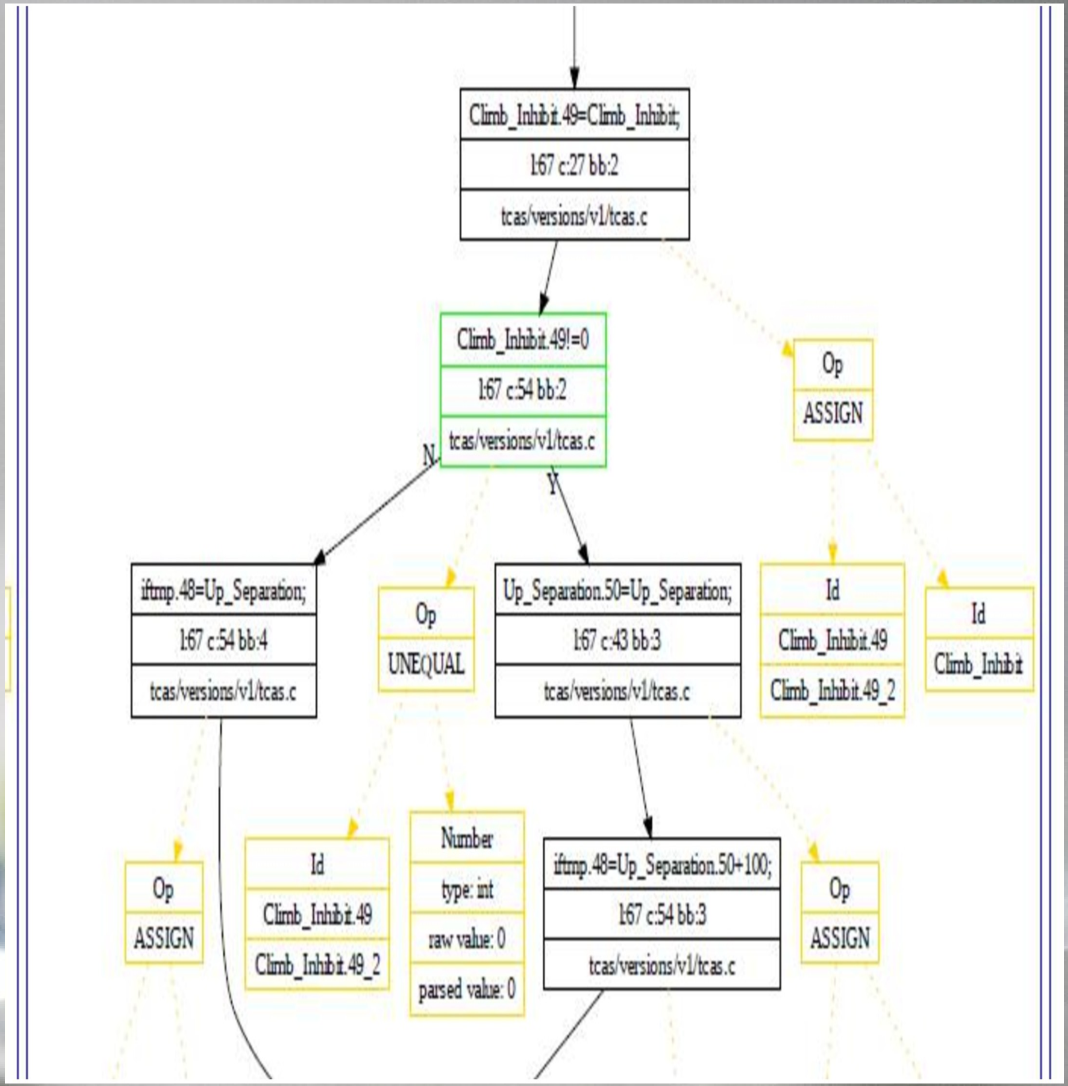
Definition 4: Undirected graph is a structure $U=<N, E>$ where N is a set of nodes amd E is a set of simple edges in $N \times N$. There exists a simple path from an arbitrary node n in N to all the other nodes in N. A path is a list of the nodes $p_0, p_1, ..., p_k$ such that $p_0 = n_i, p_1 = n_{i+1}, \ldots, p_k = n_k$, and for all i, $0 \le i \le k-1$, (p_i, p_{i+1}) is in E.

Definition 5: Abstract syntax tree $A=<N, E, n_0>$ is a rooted undirected graph $U=<N, E^*>$ with the root n_0 where two nodes are connected exactly by one simple path.

Developed at the **Graz University of Technology** by **Robert Könighofer** and colleagues from the Institute for Applied Information Processing and Communications – IAIK.

The Model (2)

- Black and green boxes are hammock graph nodes, yellow boxes are abstract syntax tree nodes.
- Mathematical structure of the FORENSIC tool's model is published in [45].

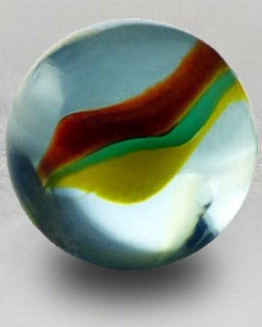


Developed at the **Graz University of Technology** by **Robert Könighofer** and colleagues from the Institute for Applied Information Processing and Communications – IAIK.

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- The Model
- **Animation and Simulation**
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Animation and Simulation

- Are the most time-consuming phases of the simulation-based verification
- a) direct simulation
- b) simulation using C functionality (novel approach)
- Same principles can be applied to Designs, written in arbitrary programming languages like Java/PHP/Fortran.

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- **Algorithm**
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Model-Based Error Localization

- Uses the simulation trace of the model
- Assigns *passed* or *failed* counters to nodes that belong to every simulation trace
- Rankings are applied to the counters
- Nodes in the model are sorted according to the rankings
- Mathematical Definitions are in the Dissertation

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- **Rankings**
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

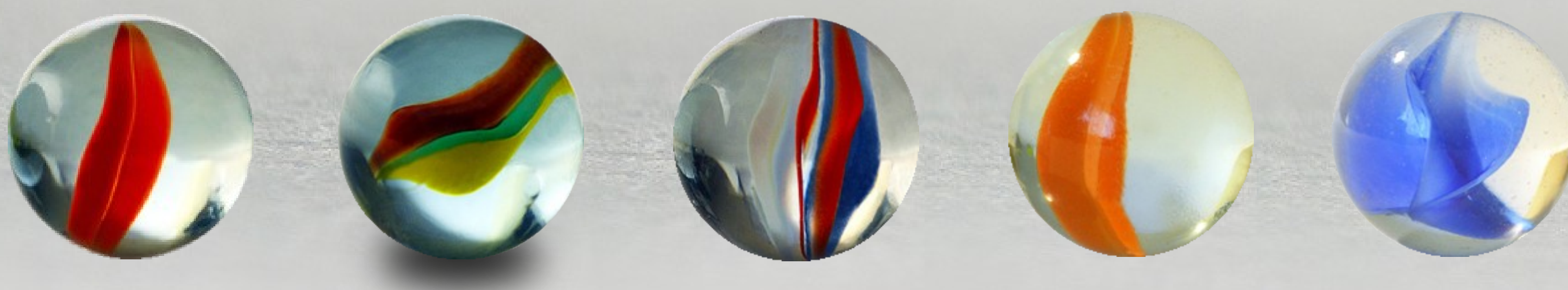
- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Rankings for the Model-Based Error Localization

- FORENSIC tool's Rankings
  - Ample tool $rank(n) = \left| \frac{failed(n)}{totalfailed} - \frac{passed(n)}{totalpassed} \right|$
  - Consecutive (Novel) $rank1(s) = \frac{failed(n)}{totalfailed}$ followed by $rank2(s) = \frac{failed(n)}{\sqrt{totalfailed \cdot (passed(n) + failed(n))}}$

    Jaccard Coefficient $rank(s) = \frac{failed(n)}{passed(n) + totalfailed}$
  - Ociai $rank(s) = \frac{failed(n)}{\sqrt{totalfailed \cdot (passed(n) + failed(n))}}$
  - Simple (Novel) $rank(s) = \frac{failed(n)}{totalfailed}$
  - Tarantula Tool $rank(s) = \frac{\frac{failed(n)}{totalfailed}}{\frac{passed(n)}{totalpassed} + \frac{failed(n)}{totalfailed}}$

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- **Dynamic Slicing Algorithm**
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Dynamic Slicing Algorithm (1)

- Is the technique that is applied to the activated nodes obtained during the simulation phase of model-based error localization
- Allows to reduce the number of nodes in the error localization result
- Some amount of nodes is activated during simulation of error localization, but do not have any influence on the simulation result
  - Constant declarations
  - Assignments
- Algorithm Discards those Nodes
- Mathematical Definitions are in the Dissertation

Dynamic Slicing Algorithm (2) *

- Example

	Design	Activated Operators $n \in N_a$	*DEF(n)*	*REF(n)*	*DV*(n)*	Line number
	int a, b, c;	int a, b, c;				1
	a=0;	a=0;	a			2
	b=a+1;	b=a+1;	b	a	a	3
	c=a+b;	c=a+b;	c	a, b	a, b	4
	if (c>0) {	if (c>0)		c	b, c	5
Not in slice	**a=0;**	**a=0;**	**a**		**b, c**	**6**
	a=b+c;	a=b+c;	a	b, c	b, c	7
	} else {					8
	a=b-c;					9
	}					10
Failed	assert(a==1);	assert(a==1);		a	a	11

Figure 4.1. An example of the dynamic slicing algorithm's execution.

* Can be described in detail if will be asked

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- **Related Work**

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Related Work

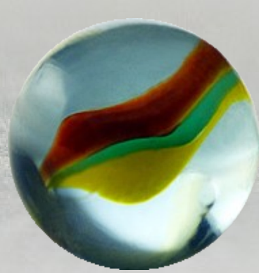
- Four independent ranking algorithms were proposed in [1].
- Program Spectrum

$$M\ Spectra \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix}$$

- Rankings Definitions have same meaning, but different mathematical form
- Experimental Results are Compared Later

[1] Abreu Rui, Zoeteweij Peter and van Gemund Arjan J.C. 2006. An Evaluation of Similarity Coefficients for Software Fault Localization. Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium on, 39-46.

Outline


Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- **The Model**
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

The Model

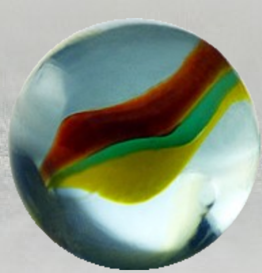
- Is Hammock Graph, where every node is parsed into Abstract Syntax Tree (AST)
  - Mutations in the AST Nodes *
  - Mutations in the edges of the hammock graph
  - Mutations in the edges of AST

* Implemented in the FORENSIC tool

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- **Error Classes**
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

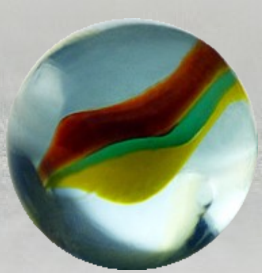
Error Classes

- Errors, that can be discovered in the processed design
- Algorithm should repair as much of them, as possible
- Possible Errors in the Siemens Benchmarks are in the Dissertation

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- **Algorithm**
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Mutation-Based Error Correction Algorithm (1)

- Implemented Mutations
  - Operator mutations: Integer variable mutation (“*I → I+1”*), (“*I → I-1”), (“I → -I”),* Mutations of the variables of primitive types (“*I → F()*”), General mutations of constants (“*C → C+1”*), (“*C → C-1”*), (*“C → 0”*), (*“C → -C”), (“C → C*2”), (“C → C/2”),* Mutations in the digits of a numeric constant (“axb.c → ayb.c”), (“ab.c → xab.c”), Floating point number rounding mutations.
  - Mutations in operators: Arithmetical operators (+), (-), (*), (/), (%), Assignment operators (=), (+=), (-=), (*=), (/=), (%=), Comparison operators (<), (>), (==), (!=), (<=), (>=), Logical operators and bit shift (&&), (||), (<<), (>>), (&), (|), (^), Bit shift with assignments operators (<<=), (>>=), (&=), (|=), (^=), Unary arithmetical operators (+), (-), Increment and decrement operators (++x), (x++), (--x), (x--), Unary logical operators (!), (~).

Mutation-Based Error Correction Algorithm (2)

- One Error Assumption
- Multiple Error Assumption
- Error Classes Are Defined
- Algorithm is fast, simple, reliable
- Mathematical Definitions are in the Dissertation

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- **Properties**

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

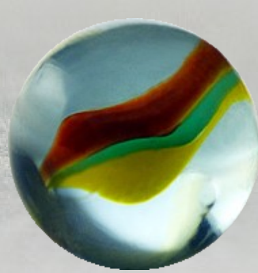
- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Properties*

- Can be used for Measuring of the Error Localization Accuracy
- Can Correct Error Using Alternative Mutation
- If Correctional Mutation is found Depends on the entire Model Structure

* Examples can be shown is will be asked

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- **Siemens Benchmarks**
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

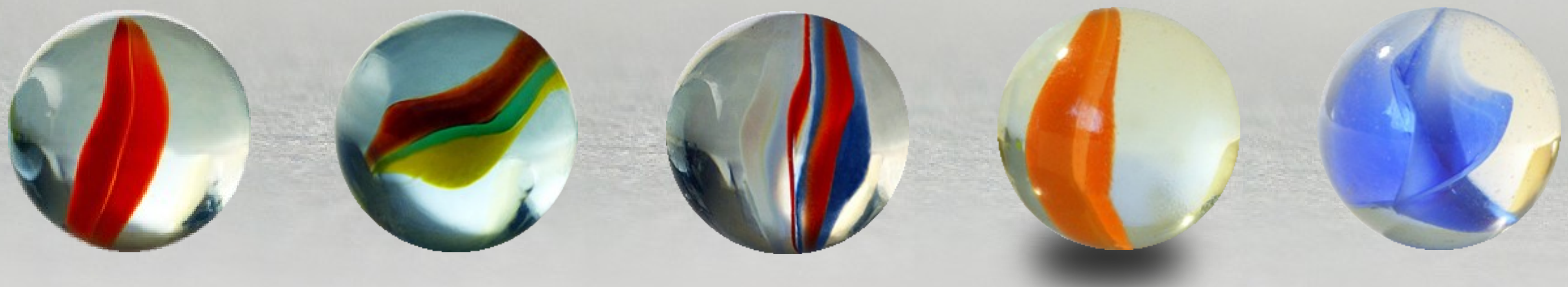
- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Siemens Benchmarks*

Design	Number of Lines of Code	Number of Inputs	Number of Designs with Errors
print_tokens	569	4130	7
print_tokens2	515	4115	10
replace	558	5543	31
schedule	419	2650	9
schedule2	312	2710	10
tcas	186	1605	41
tot_info	413	1052	22

* Bug Aristotle Analysis System -- Siemens Programs, HR Variants <http://pleuma.cc.gatech.edu/aristotle/Tools/subjects/>

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- **Error Localization**
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Model-Based Error Localization (1)

- In [1] a number of rankings for the model-based error localization are proposed

Design	Ample	Jaccard	Ochiai	Tarantula
print_tokens	2	7,3	1	10,5
print_tokens2	16,4	17,5	13,9	20
replace	12,7	11,6	7,6	12,2
schedule	11,3	2,9	1,6	3
schedule2	34,7	31,1	25,1	31,3
tcas	9,8	8,8	7,9	8,8
tot_info	13,2	9,6	7,1	11
Mean	14,3	12,69	9,17	12,69

Percentage of statements to be inspected to reach error.

[1] Abreu Rui, Zoeteweij Peter and van Gemund Arjan J.C. 2006. An Evaluation of Similarity Coefficients for Software Fault Localization. Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium on, 39-46.

Model-Based Error Localization (2)

- In FORENSIC tool

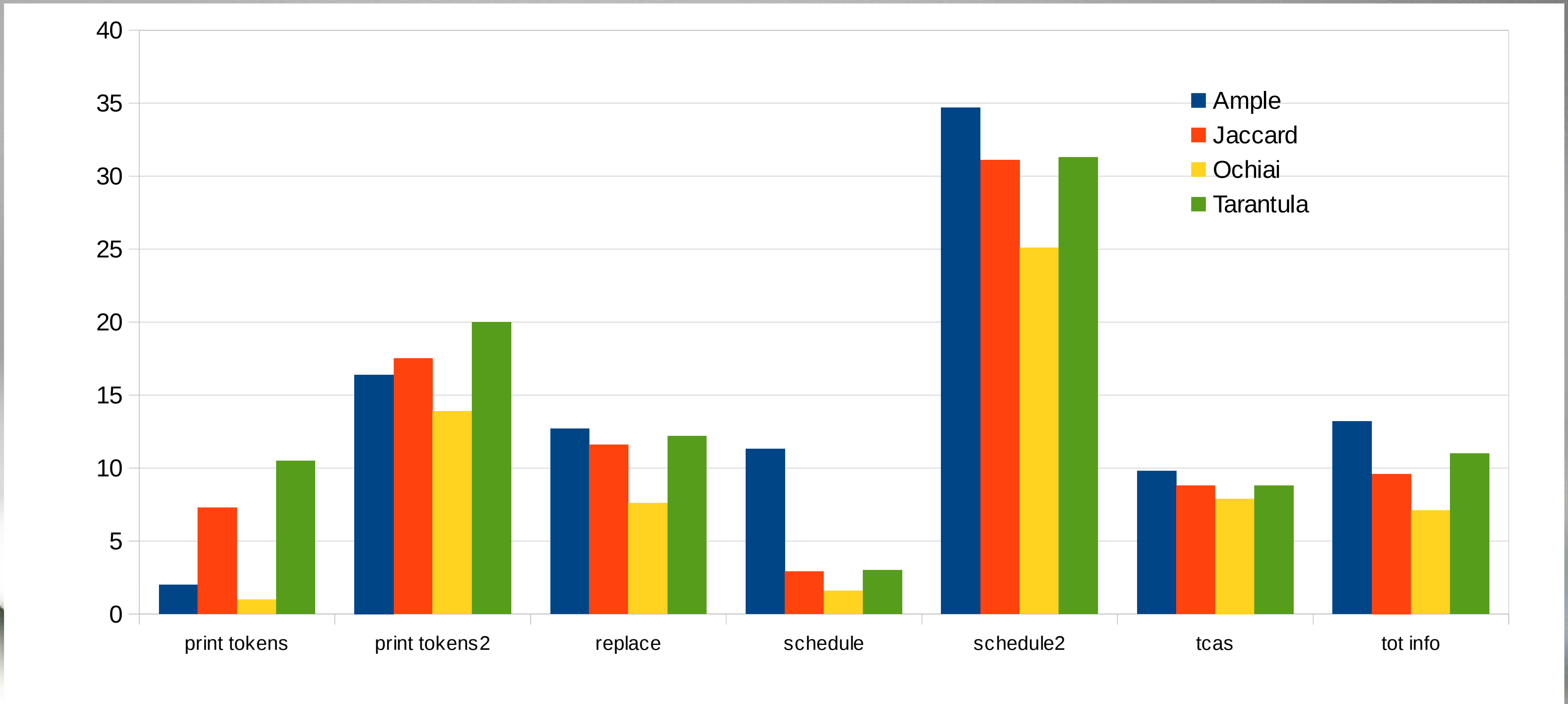
Design	Ochiai	Ample	Consecutive	Jaccard	Ochiai	Simple	Tarantula	*
print_tokens	1	2	7,3	1	10,5	1	10,5	1
print_tokens2	13,9	16,4	17,5	13,9	20	13,9	20	13,9
replace	7,6	12,7	11,6	7,6	12,2	7,6	12,2	7,6
schedule	1,6	11,3	2,9	1,6	3	1,6	3	1,6
schedule2	25,1	34,7	31,1	25,1	31,3	25,1	31,3	25,1
tcas	7,9	9,8	8,8	7,9	8,8	7,9	8,8	7,9
tot_info	7,1	13,2	9,6	7,1	11	7,1	11	7,1
Mean	9,17	14,3	12,69	9,17	12,69	9,17	12,69	9,17

* Number of Corrected Designs

Percentage of mutations to be applied to correct the error in the design.

Model-Based Error Localization (3)

- Graphical representation of experiments in [1]

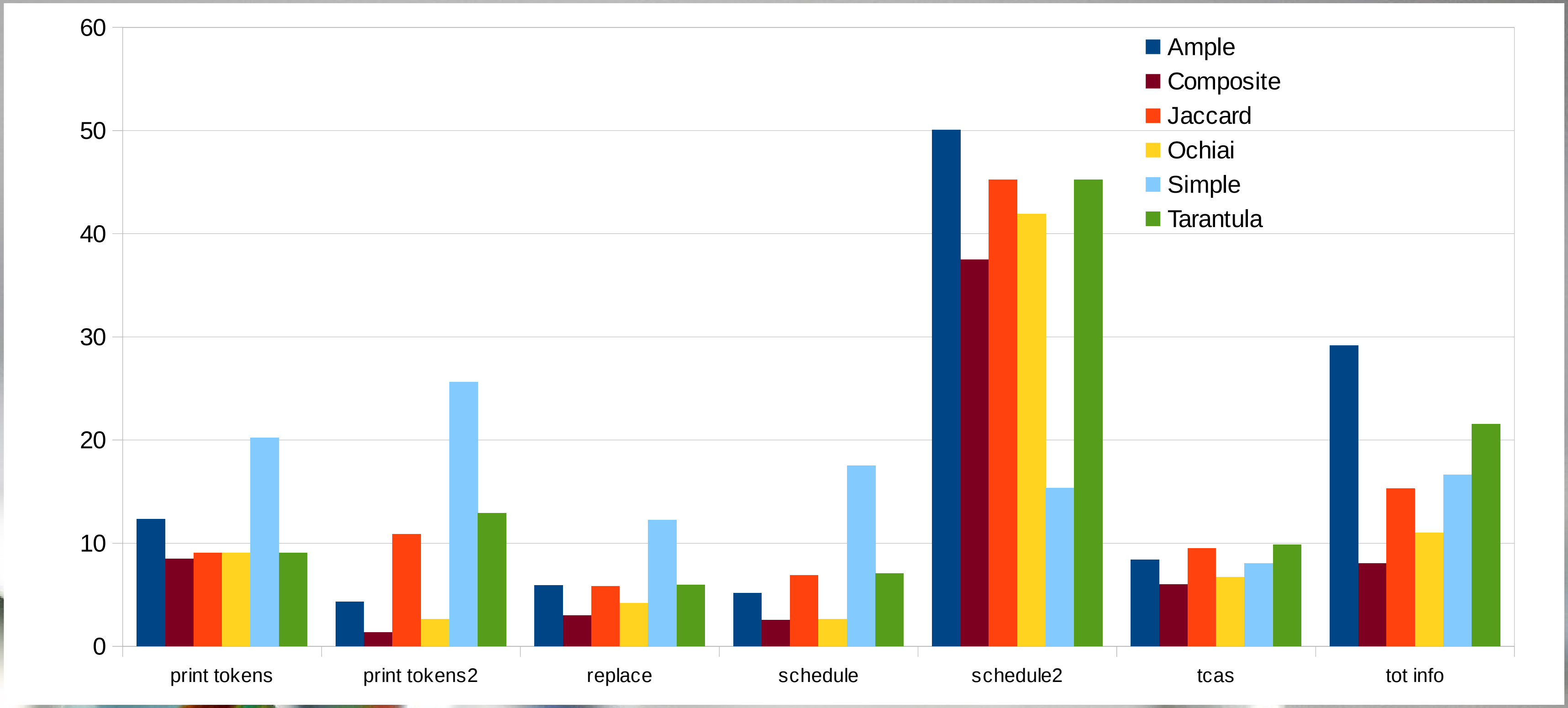


Percentage of statements to be inspected to reach error.

[1] Abreu Rui, Zoeteweij Peter and van Gemund Arjan J.C. 2006. An Evaluation of Similarity Coefficients for Software Fault Localization. Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium on, 39-46.

Model-Based Error Localization (4)

- Graphical representation FORENSIC experiments



Percentage of mutations to be applied to correct the error in the design.

Model-Based Error Localization (5)

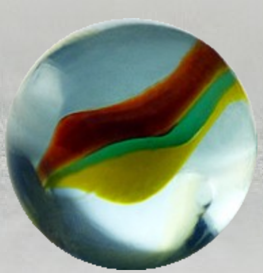
- Can be used for Measuring of the Error Localization Accuracy
- In [1] Ochiai ranking decreases the percentage of blocks of code to be inspected by *5%* comparing with worst case – the Ample tool ranking
- In FORENSIC tool by 10%, that is much better

[1] Abreu Rui, Zoeteweij Peter and van Gemund Arjan J.C. 2006. An Evaluation of Similarity Coefficients for Software Fault Localization. Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium on, 39-46.

Outline

Introduction

- Problem Formulation
- Requirements for the tool
- Contributions



Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- **Dynamic Slicing**
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Dynamic Slicing (1)

- In FORENSIC tool

Design	Number of Mutations, required to correct the error		Change, $\lvert\frac{A-B}{B}\cdot 100\,\%\rvert$
	(A) If Dynamic Slicing is used	(B) If Dynamic Slicing is not used	
print_tokens	1954,5	2106,0	7,75
print_tokens2	2141,14	2254,29	5,02
replace	1635,88	1683,06	2,8
schedule	564,0	715,33	21,16
schedule2	648,25	713,25	9,11
tcas	304,0	553,71	45,1
tot_info	1338,5	1363,94	1,87

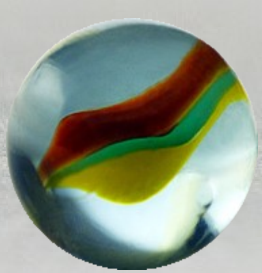
Dynamic Slicing (2)

- In [60] Experimental Evaluation of using Dynamic Slicing for same Benchmark Designs were performed

Design	# of Proc. Designs in FORENSIC	Change in FORENSIC	# of Proc. Designs in [60]	Change in [60]
print_tokens	2	7,75	5	32
print_tokens2	7	5,02	8	49
replace	16	2,8	19	42
schedule	3	21,16	6	47
schedule2	4	9,11	3	53
tcas	31	45,1	-	-
tot_info	18	1,87	-	-

[60] Zhang Xiangyu, He Haifeng, Gupta Neelam, Gupta Rajiv. 2005. Experimental evaluation of using dynamic slices for fault location. ADEBUG - Proceedings of the Third International Workshop on Automatic Debugging, Linköping, Sweden, 33-42.

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- **Error Correction**

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- Future Extensions

Error Correction (1)

- One Error Assumption
- Dynamic Slicing is used for Model-Based Error Localization
- Simple Ranking is used for Model-Based Error Localization

Design	% of Errors Corrected	# of Mutations to correct Error	Processing time, s (1000 inputs)
print_tokens	2/7=28,57	1954,5	330
print_tokens2	7/10=70,0	2124,14	502
replace	16/32=50,0	1635,88	342
schedule	3/9=33,33	564	130
schedule2	4/10=40,0	648,25	184
tcas	31/41=75,61	304	50
tot_info	18/32=56,25	1338,5	312

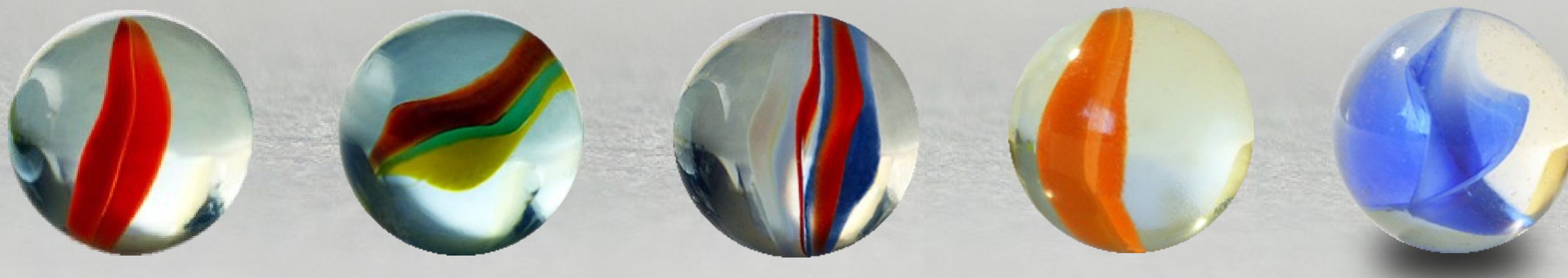
Error Correction (2)

- In [12] Mutation-Based Error Correction experiments on the same Siemens designs is presented

Design	# of Errors Corrected in [12] (A)	# Errors Corrected in FORENSIC (B)	Total Designs (C)	$\frac{B-A}{C} \cdot 100\,\%$
print_tokens	0	2	7	28,57
print_tokens2	0	7	10	70,0
replace	3	16	32	40,63
schedule	0	3	9	33,33
schedule2	1	4	10	30,0
tcas	9	31	43	53,66
tot_info	8	18	23	43,48

[12] Debroy Vidroha, and Wong W. Eric. 2010. Using Mutation to Automatically Suggest Fixes for Faulty Programs. Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, 65–74

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- **Contributions and Novel Approaches**
- Experimental Results
- Future Extensions

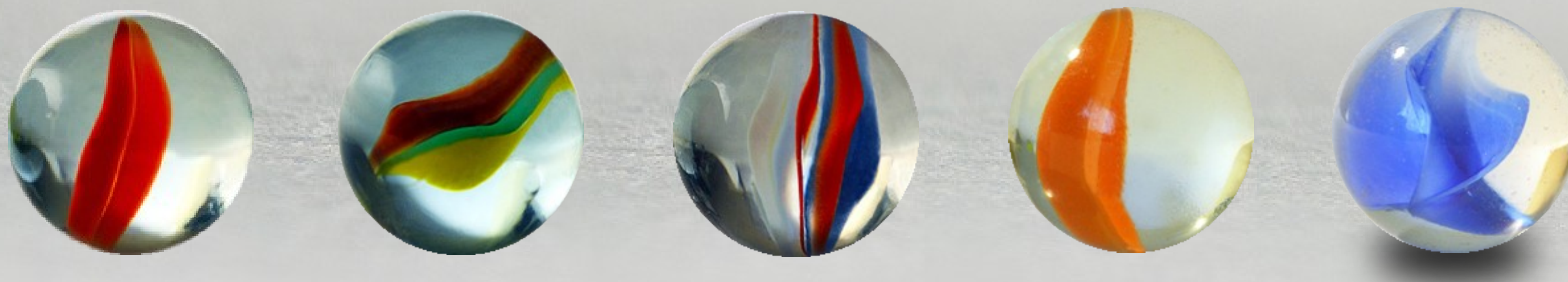
Contributions and Novel Approaches

- Introduced two novel rankings for model-based error localization – Consecutive and Simple rankings.
- Usage of the compiler functionality for the model simulation task, the specification animation and the implementation of the dynamic slicing algorithm.
- Usage of the number of processed mutations instead of a number of statements required to reach the error as a measure of the error localization accuracy.

Contributions and Novel Approaches

- Implementation of the model-based error localization, mutation-based error correction and dynamic slicing algorithms in the corresponding tool.
- Definition of the specification format for simulation-based verification.
- Finally, based on the development and evaluation performed during writing the dissertation, can be claimed that the FORENSIC tool can be utilized in a useful way in the process of software development.

Outline



Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- **Experimental Results**
- Future Extensions

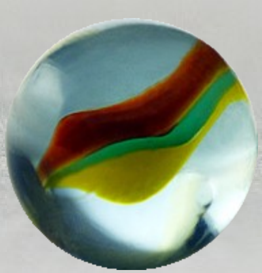
Experimental Results (1)

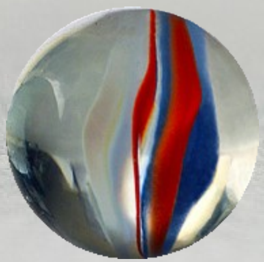
- *Simple* ranking for the model-based error localization is more stable than other rankings
- *Ochiai* ranking decreases the number of required mutations compared to the worst case *Ample* ranking by 10%
- *Consecutive* ranking that is simple ranking plus some secondary ranking, for example Ochiai, results the best results, 1.25% better than Ochiai ranking.

Experimental Results (2)

- For tcas design the dynamic slicing algorithm will reduce the number of required mutations for correction by *45,1%* on average, for schedule – by 21,16%, for schedule2 by 9,11%, and for print_tokens by 7.19%, but less for replace, tot_info and print_tokens2 designs.
- Mutation-based error correction algorithm can suggest corrections for significantly more designs than the approach presented in [12], *70%* additionally corrected errors for *print_tokens2* design, *54%* additionally corrected for *tcas* design.

Outline





Introduction

- Problem Formulation
- Requirements for the tool
- Contributions

Model-Based Error Localization Algorithm

- The Model
- Animation and Simulation
- Algorithm
- Rankings
- Dynamic Slicing Algorithm
- Related Work

Mutation-Based Error Correction Algorithm

- The Model
- Error Classes
- Algorithm
- Properties

Experimental Results

- Siemens Benchmarks
- Error Localization
- Dynamic Slicing
- Error Correction

Conclusion

- Contributions and Novel Approaches
- Experimental Results
- **Future Extensions**

Future Extensions

- In the future, the FORENSIC tool can be extended with C++/SystemC support and the algorithms presented in the dissertation can be carried over for C++/SystemC error localization and error correction.
- It is possible to implement an input generator for the tool while the reference outputs can be achieved by simulating the specification with the generated inputs.

Tallinn University of Technology
Department of Computer Science

Thank You

Urmas Repinski

Ph.D. Dissertation Defense

Tallinn, 2016