

Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Model-based Testing: Techniques and Industrial Applications

Lukas Brügger

Burkhart Wolff

Information Security Group, ETH Zürich LRI, Université Paris-Sud

BT Ipswich, May 2009

Abstract

Model-based testing has seen a wider range of industrial applications recently – enabling to systematically *generate* test-cases instead of speculate them or analyse post-hoc system traces or memory dumps.

Test-case generation techniques vitally depend on symbolic computation and constraint-solving techniques. Their limits therefore represent limits for model-based testing as a whole. The HOL-TestGen system is designed as plug-in into the state-of-the-art theorem proving environment Isabelle/HOL. Thus, powerful modeling languages as well as powerful automated and interactive proof methods for constraint resolution are available.

The talk is going to be a guided tour through theory, pragmatics, and recent industrial applications. "

Intro: Definition and Summary

WHAT IS MODEL-BASED TESTING ?

Model-Based Testing is the automatic generation of efficient test procedures/vectors using models of system requirements and specified functionality.

Models of Systems for Tests



Models of Systems for Tests



BT Ipswich, May 2009

Models of Systems for Tests



BT Ipswich, May 2009

Modeling ...

- ... aims at "blueprints" that can be analysed BEFORE the system is actually build
- ... does not guarantee the absense of any error (only the conformance between a model and the "system")
- ... can (and must) be integrated into the software development cycle ...

Modeling ...

• ... can be done post-hoc; significant projects "reverse engeneer" the model of a legacy system

• ... can help system integration processes by assuring that third-party components are in fact usable in a larger system.

The model gets the role of a "contract" in this scenario.

Workflow



BT Ipswich, May 2009

Vision

- Model-development should be integrated into the classical software development process; thus into:
 - Requirements documents; Design documents ...
 - Test-Cases should be used early for Animation and "Reverse Engineering" ...
 - ... in some cases, a combination with verification techniques might be useful ...

HOL-TestGen: A Solution

- HOL-TestGen is a Model-based TestCase Generation System
- Unlike e.g. Spec-Explorer (by Microsoft, available as VisualStudio Plugin), it emphasizes (*well, we are academic ;-)*):
 - logical cleaness and an expressivness.
 Modeling Language HOL instead of, say, an OO-language with quantifiers
 - symbolic computations having their roots in Theorem Proving instead of plain enumeration and model-checking

Agenda

- TestGen and its Method by Example
- Overview on Symbolic Test Case Generation
- Own Case Studies
- Industrial Applications
- Conclusion

- Step I in the TestGen method:
 - write Test Document containing HOL Definitions

```
text{* We include the TestGen system and
start with a litte example *}
Triangle = Testing +
text{* The result type is defined by: *}
datatype triangle = equilateral | scalene |
isosceles | error
constdefs triangle :: "[nat,nat,nat] => bool"
"triangle x y z == (0<x ^ 0<y ^ 0<z ^
(z<x+y) ^ (x<y+z) ^ (y<x+z))"</pre>
```

BT Ipswich, May 2009 • •

- Step II in the TestGen method:
 - containing a Test Specification TS in HOL ... (ctd'd):

```
testspec TS:
"prog(x, y, z) =
if triangle x y z
then if x = y
then if y = z then equilateral
else isosceles
else if y = z then isosceles
else if x = z then isosceles
else scalene
```

• where prog is the program under test

• • •

- Step III in the TestGen method:
 - fire generate cases tactic and get proof-state:

apply(gen_test_cases 3 1 simp: add_commute)

. . .

- Step III in the TestGen method:
 - fire generate cases tactic and get proof-state:

$$\begin{bmatrix} 0 < z; z < z + z \end{bmatrix} \implies \\ prog(z, z, z) = equilateral \\ \begin{bmatrix} x < z; 0 < x; 0 < z; z < x + z; x < z + z \end{bmatrix} \implies \\ prog(x, z, z) = isosceles \\ \begin{bmatrix} y < z; z & y; \neg z < z + y \end{bmatrix} \implies \\ prog(z, y, z) = error \end{bmatrix}$$

A Step Back: Test-Theorem

- Step III in the TestGen method:
 - consisting of 26 test cases C₁ to C₂₆
 (having the form of Horn clauses, where the premises are called constraints)
 - where the proof state corresponds to an equivalent test theorem of the form:

$$C_1 \longrightarrow \dots (C_{26} \longrightarrow TS) \quad (written: \llbracket C_1; \dots; C_{26} \rrbracket \implies TS)$$

- Step V in the TestGen method:
 - fire generate cases tactic and get proof-state and produce test statements (i.e. premises of the form):

...
gen_test_data "Triangle"

- Step V in the TestGen method:
 - fire generate cases tactic and get proof-state and produce test statements (i.e. premises of the form):

prog(3, 3, 3) = equilateralprog(4, 6, 0) = error

- Step VI in the TestGen method:
 - Convert test-data automatically into a test driver.

```
. . .
gen_test_script "Triangle"
```

In our case, this is an SML program that fires the test-harness, which can be linked to any .o file containing the program under test... (so, the SUT must not be SML, rather C, Java, ...)

Symbolic Computations Involved

- Basis for TestGen package (comprising Test Case and Test Data Generation tactics)
 - Isabelle/HOL library: 10000 derived rules . . .
 - about 500 are organized in larger data-structures used by Isabelles proof procedures . . .
- How are tactics organized?
 - Rewriting Normal Form Computation (RNF)
 - Tableaux Normal Form Computation (HCNF)
 - Testing Normal Form Computation (TNF)
 - Testing Normal Form Minimization (MTNF)
 - Generating and Using Test Hypothesis

BT Ipswich, May 2009

Own Case Study: Red Black Trees Red-Black-Trees: Test Specification

```
testspec :
(redinv t ∧ blackinv t)
→
  (redinv (delete x t) ∧
     blackinv (delete x t))
```

where delete is the program under test.

Own Case Study: Red Black Trees

• Statistics:

348 test cases were generated, within 2 min.

- one Error in the SML library was found, that makes crucial violation against redblackinvariants; makes lookup linear
- ... error not found within 12 years ...
- ... reproduced meanwhile by random test tool

Own Case Study: Firewalls

• Statistics:

10000 test cases were generated, within 8 h.

- ... realistic scenarios of analysis require quite advanced techniques for case-splitting and deduction
- ... uses real theorem proving

- Windows 98-Server Protocol: the story so far
- 2000 : EU and US administration ruled Microsoft is a Monopoly in the Server Market (applying older Antitrust rules in the Telecommunication market)
- 2002 : EU required the "specification" of the server protocols in order to allow third-party vendors acces to the market
- Polished internal documents of Mocrosoft were considered "insufficient" by the EU referees ...

- 2003: Microsoft legally contested this ruling, considering protocols as protected being IPR
- 2005: Microsoft lost the legal battle, was fined by 700 mio €, and forced to produce a document which:
 - also provides a formal specification
 - provides evidence that the model is actually compliant to the implemented system.

Since then, a team of 200 people started to reverse engineer the Protocol (developed in 1995), essentially using a tool-family on the basis of Spec-Explorer

🐢 smb2 - Microsoft Visual Studio					15 11	13 – 🛛 X
<u>File Edit View P</u> roject <u>B</u> uild	<u>D</u> ebug D <u>a</u> ta <u>T</u> ools	Te <u>s</u> t <u>W</u> indow <u>C</u> ommunity <u>H</u> elp				
1 🛐 - 🖽 - 📂 🔒 🥔 X 🖻 🛍	19 - 19 - 📮 - 🖳	Debug - Mixed Platforms	- 🖄	ContractModel	- 💀 🚰 🕺 🛠 🛃 🖸 - 📮	
:::::::::::::::::::::::::::::::::::::::	i ta 🕫 🔒 💷 🗞 🕅	L & 律律 1 1 1 1 1	a a a a			
Config.cord [design] Smb2Adapter.cs Model.cs			SlicedStateMachine exploration - config.cord			
Machine		•	• • •	Lu vik 🙆 🗖 🖡	8	
State Machine		Lises Confine				
Set un Scenario		Askees	1		(so)	
SlicedStateMachine	Actions				Y	
TestSuite	Hemove	Behavior			EnsureShareExists(1,DISK)	
<new machine=""></new>	Validate	(Satus Saappia():) State Machine	-		±	
	Emlans (In annumentally)	(Setup Scenario(),) Statemachine			(\$2)	
					Ť	
	Test Generate tests				SetupConnectionAndSession()	
					(54)	
					TreeConnectRequest(0,1,1)	
					V	
					anant TraaConnactRomonac(0.1.0.DISK)	
Config					event freeconnectivesponse(0,1,0,010K)	
	Switches Types Act	tions Constraints Exceptions			(57)	
Actions (new config)			-		Y	
chew conlig>					CreateRequest(1,1,0,Create, "test1") CreateRequest(1,1,0,Open, "test1")	
	Exploration		_		大大大	
	BoundPath	inherited			< <u>\$10</u> (\$11)	
	BoundStates	inherited				
	BoundSteps	1024			event CresteResponse(0,1,0) event ErrorResponse(CREATE,0,1)	
	CodeGenerationTim	eout inherited				
	DepthFirst	inherited		/		
	Explorer	inherited		Class Partient (2 1 0 0)	Pand Parameter 2 1 0 (b) White Parameter 2 1 0 0 Sar (1 2 2))	
	StepsUntilSuspend	inherited		Closerequest(2,1,0,0)	Coaucellesi(2,1,0,0) WhiteCollesi(2,1,0,0,004(1,5,2))	
	StopAtError	inherited		\$10	S18 (S17)	
	Solver			Y		
	Solver			event CloseResponse(0,]	1) event ReadResponse(0.1.Seq{}) event WriteResponse(0.1)	
	Testing				\mathbf{T} \mathbf{T} \mathbf{T} \mathbf{T}	
	ObservationBound	inherited		(\$22) ((S21) (S20)	
	ProceedControlTime	out inherited			$\gamma \gamma \gamma \sim$	
	QuiescenceTimeout	inherited		CloseRequest(3,1,0,0)	WriteRequest(3,1,0,0,Seq{1,3,2}) ReadRequest(3,1,0,0) CloseRequest(3,1,0,0)	
	ReRuns	inherited		\sim		
				<\$30	< <u> <529 <527 <528 </u>	
	CollapseLabels	inherited				
	CollapseSteps	inhented		event CloseResponse((0,1) event WriteResponse(0,1,Seq{1,3,2}) event Clos	eKesponse(0,1)
	Display Requirement	s innented	-	(\$34)		
	Graph Timeout	inherited		3.4	<u>پ</u>	
	Maxi abalaParAra	1C	-		CloseRequest(4,1,0,0)	
	MaxLabelsFerArc NoViewer	01 inherited	-			
	ShowErroreOnly	inhented			\$36	
	ShowEndisofily	intented	=		Y	
Remove BoundSteps					event CloseResponse(0,1)	
Extends Configs Defines a bound on the total number of steps to explorer.						
					(\$37)	
P			Finished		36 states, 37 steps, 0 errors	00:00:01.8220000
				hare evoloration 🔂 Calu	ution Evolorer 16 SlicedStateMachine evoloration	
	, , , , , , , , , , , , , , , , , , ,	*		mare exploration - 2000	auon explored we procedulate via chine exploration m Properties	
🛃 Error List 🛌 SlicedStateMachine sta	tes 📃 Output 🙀 Find Syr	mbol Results 🔚 Test Results				

Ready

Windows Server 98 Protocol :

Wolfgang Grieskamp[2008]:

Using Model-Based Testing for Quality Assurance of Protocol Documentation

Invited Talk MBT 2008, Budapest. http://research.microsoft.com/users/wrwg /MBTETAPS.pdf

٠

Conclusion

- Nowadays, model-based Testing is viable Technology
 - ... for systematic Testing (unit, sequence, reactive sequence, protocol testing)
 - ... for reverse-engineering Systems and integrating components of third-parties
 - ... to comply with future, legally required documentation standards

Conclusion

- HOL-TestGen
 - Specs were written in HOL
 - proof-state explosion controllable by abstraction
 - although logically puristic, systematic test of a "real" library code or network components security policies has been shown feasible ...
 - besides: HOL-TestGen is a verified tool inside a (well-known) theorem prover