#### **PROGRAMMING WITH TRIGGERS**

#### **Michał Moskal**



**Microsoft**<sup>•</sup> Innovation Center Europe

University of Wrocław Wrocław, Poland Aachen, Germany

Satisfiability Modulo Theories Workshop August 2<sup>nd</sup>, 2009, McGill University, Montreal, Canada

# OUTLINE

- case-study: an SMT-powered software verifier applied to a commercial operating system
- tools and methods to make this work
  - trigger-engineering
  - tools:
    - Axiom Profiler postmortem analysis of the search
    - Model Viewers analysis of counter examples
    - Z3 Inspector live view of Z3 operation

### WINDOWS HYPERVISOR

- virtualization platform
  - thin layer of software between guest operating systems and the hardware
- essentially a small operating system
  - small by OS standards: 100kloc of C, 5kloc x64 asm
- scheduler, memory allocator, etc
  - lock-free data structures
- shipping with Windows Server 2008

#### HYPERVISOR VERIFICATION (2007 – 2010)

Partners:

- European Microsoft Innovation Center
- Microsoft Research Redmond
- Microsoft's Windows Div.
- Universität des Saarlandes
- German Research Center for Artificial Inteligence

**erisoft** 

co-funded by the German Ministry of Education and Research http://www.verisoftxt.de

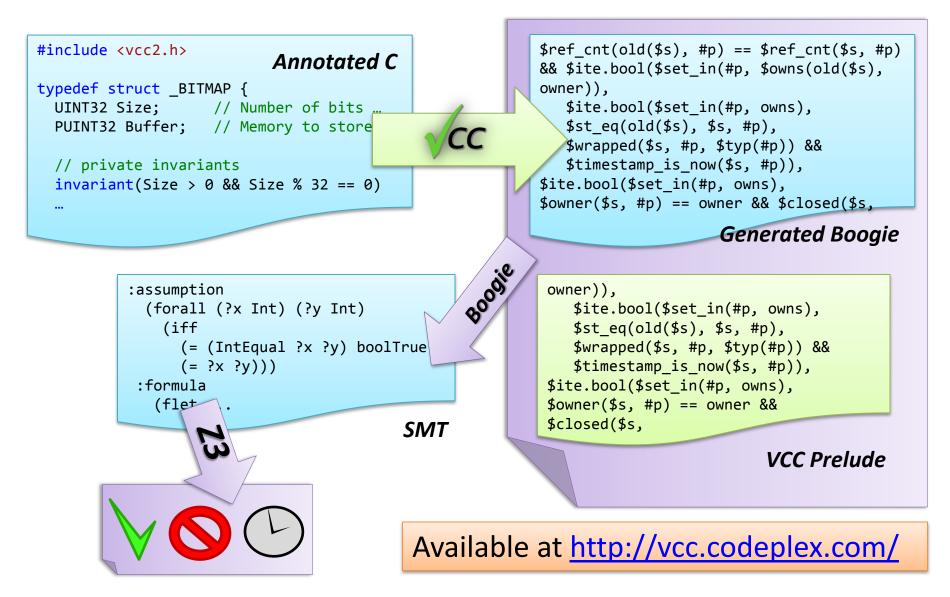
### GOAL: FULL-BLOWN VERIFICATION FOR EVERYONE

- functional properties
  - but even memory safety depends on functional correctness of complex data structures and concurrency protocols
- automatic
- exercised on real code
  - scalable modular
  - concurrency
  - not changing existing code
- necessary tool support

# VCC

- a deductive verifier for C
- verification methodology centered around
  - two-state invariants
  - ownership system
  - concurrency
- uses Boogie and Z3 (or other Boogiesupported provers)

### VCC ARCHITECTURE



# A VERIFICATION METHODOLOGY

- annotation language
  - e.g., first-order logic, higher-order logic, separation logic; + specific features
- specification concepts
  - ownership, type invariants, permissions
- modeling of the programming language semantics

how precise, what assumptions, etc.

• specification idioms

### VERIFICATION METHODOLOGY AS AN SMT THEORY

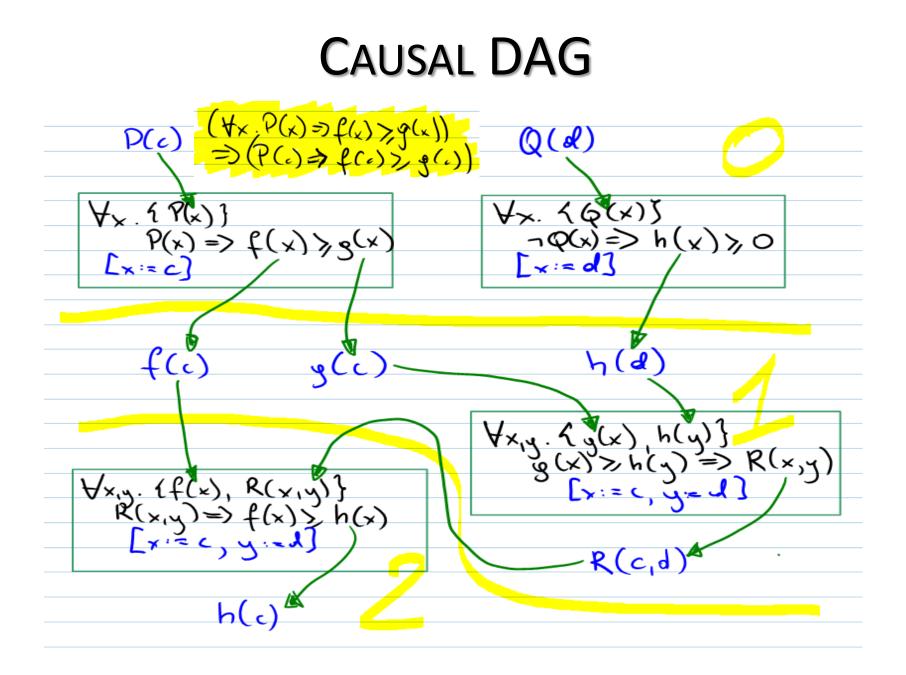
- complex
  - all input language + specification language constructions
- evolving with the verification tool
- not practical to implement as part of SMT solver
- instead encoded using first-order logic

### PROGRAMMING WITH TRIGGERS

- SMT formulas with quantifiers handled with instantiation
  - guided by E-matching, controlled by trigger annotations
- SMT theory is programmed using triggers

### Triggers

- (usually) subterms of the quantified formula, with free variables
- matched against active terms
  - terms with interpretation in the current partial model considered by the SMT solver

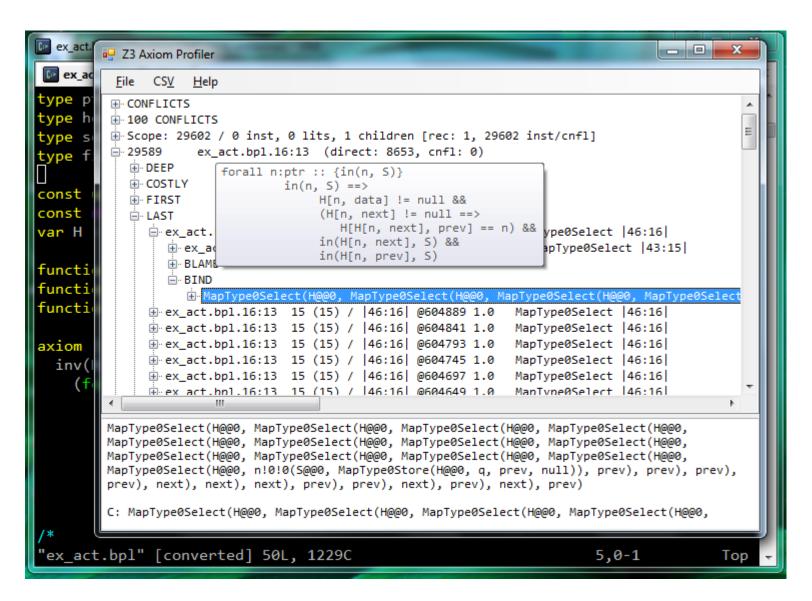


### **A LIST INVARIANT**

- inv(H, S): nodes in set S form a doubly-linked list in heap H
  - data is non-null
  - prev link in the next node points back here
  - S is next- and prev-closed

#### **DEMO OF THE AXIOM PROFILER**

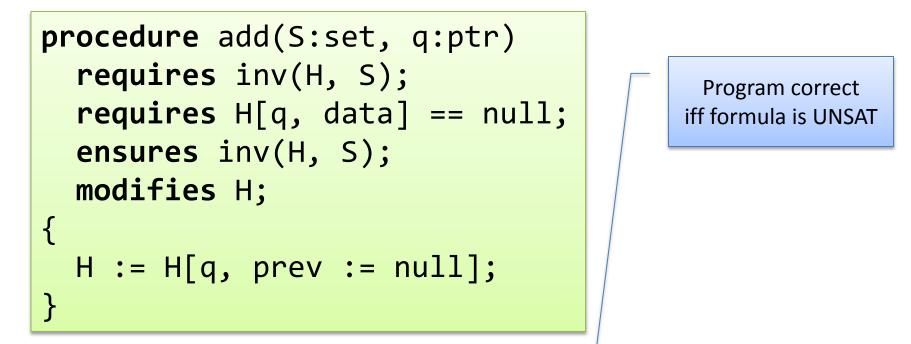
#### THE AXIOM PROFILER



#### PREVENT LOOP BY SPLITTING NEXT-CLOSEDNESS

```
(forall H:heap, S:set :: {inv(H, S)}
 inv(H, S) <==>
       (forall n:ptr :: {in(n, S)}
         in(n, S) =>
            H[n, data] != null \&\&
            (H[n, next] != null ==>
                H[H[n, next], prev] == n))
   && (forall n:ptr :: {in(H[n, next], S)}
          in(n, S) ==> in(H[n, next], S))
   && (forall n:ptr :: {in(H[n, prev], S)}
          in(n, S) ==> in(H[n, prev], S)))
```

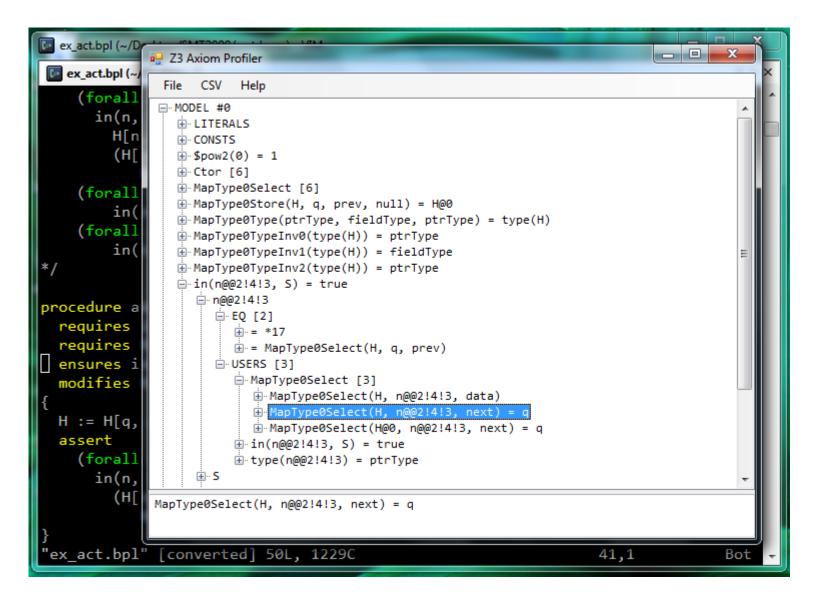
#### CHECK A PROGRAM



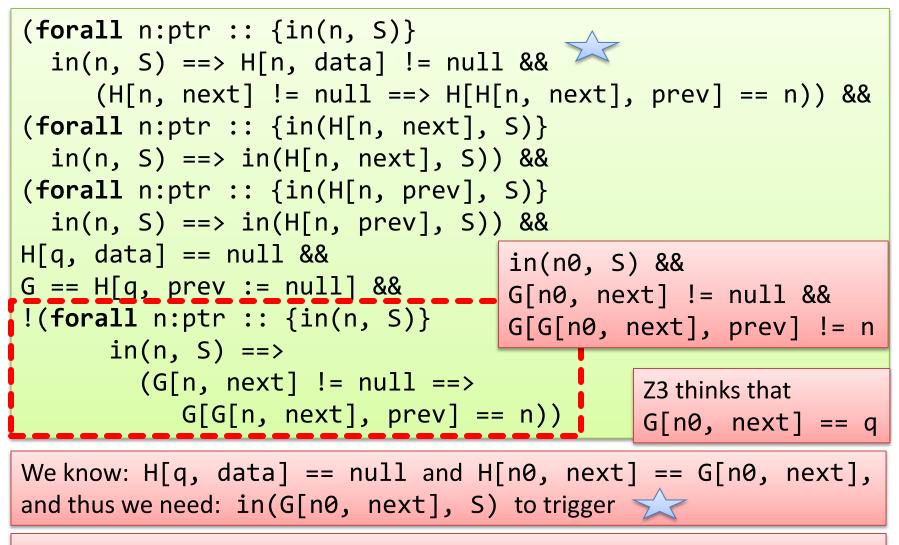
```
inv(H, S) &&
H[q, data] == null &&
G == H[q, prev := null] &&
!inv(G, S)
```

#### **DEMO OF THE MODEL VIEWER**

#### THE MODEL VIEWER



#### WITNESSES



But n0 is a skolem constant, so it's hard for the user to introduce it.

#### **ONE STEP AHEAD**

```
!(forall n:ptr :: {in(n, S)}
    in(n, S) ==>
    (G[n, next] != null ==>
        G[G[n, next], prev] == n))
```

Whenever proving this thing, look one step ahead, or: i.e., get in(G[n0, next], S) activated but don't loop. Please.

Hack the SMT solver to do it :-)

### **EXISTENTIAL ACTIVATION**

When proving this

```
quantifier, use lemma
(forall H:heap, S:set :: {inv(H,
                                       trigged by this
  inv(H, S) <==>
       (forall n:ptr :: {in(n, S)}
         {ex act(in(H[n, next], S))}
         in(n, S) ==>
            H[n, data] != null \&\&
             (H[n, next] != null ==>
                 H[H[n, next], prev] == n))
    && (forall n:ptr :: {in(H[n, next], S)}
          in(n, S) ==> in(H[n, next], S))
    && (forall n:ptr :: {in(H[n, prev], S)}
          in(n, S) ==> in(H[n, prev], S)))
```

#### **VCC IN ACTION**

### VCC VISUAL STUDIO PLUGIN

VccDemo - Microsoft Visual S	tudio	
<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	A <u>n</u> alyze <u>W</u> indow Verif <u>y</u> <u>H</u> elp
i 🛅 • 🛅 • 💕 🛃 🕔 👗	🗈 🛍   9 + 🔍 + 🚚 + 🖳   🕨	Debug 🔹 realc 🔹 💀 🙄 💷 🙄
BitMap.c SpinLock.r List.c		- × 🕎
(Global Scope)	•	■ RemoveEntryList(PLIST_ENTRY Entry) 🔹 🗸
<pre>requires(s requires(E ensures(!s maintains( ensures(En ensures(se ensures(wr writes(Ent ensures(E</pre>	str       PLIST ENTRY Entrue         Verify function 'RemoveEntryList'         Verify file 'List.c'         Custom verify file 'List.c'         More VCC Commands         Call Browser         Call Browser         Create Unit Tests         Go To Definition         Go To Declaration         Find All References         Go To Header File         Breakpoint         Run To Cursor         Cut         Copy         Paste	<pre>Properties Properties Proper</pre>
Ready	Outlining •	Ln 180 Col 42 Ch 42 INS

## VCC: FAILED VERIFICATION ATTEMPT

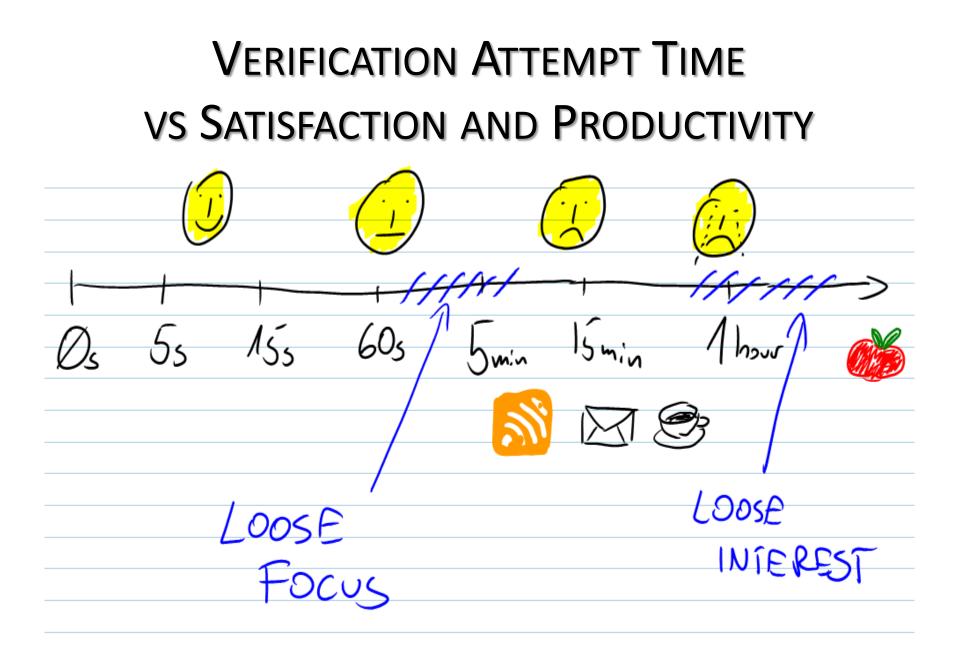
VccDemo - Microsoft Visual Studio	_	- 1		
File Edit View Project Build Debug Data Tools Test Anal	yze Window	Verify Hel	р	
🗄 🐨 🕶 🐨 🕞 🛃 🌽 📥 🛍 🖄 🔊 🔹 🖓 🖛 🖏 🕨 Debug	j <mark>-</mark> rea	alc	- 🟹 😤	
BitMap.c SpinLock. List.c				- × 🛐
	moveEntryList(	PLIST_ENTRY Er	ntry)	▼ rope
<pre>maintains(wrapped(Entry-&gt;Manager)) ensures(Entry-&gt;Manager-&gt;size == old(Entry-&gt;Mensures(set_equal(owns(Entry-&gt;Manager),set_densures(wrapped(Entry)) writes(Entry-&gt;Manager) { PLIST_ENTRY Blink, Flink; spec(PLIST_MANAGER ListManager = Entry-&gt;Manager); assert(in_domain(Entry,ListManager)); assert(in_domain(Entry-&gt;Flink,ListManager)); }</pre>	ifference( ger;)		Entry->Manager)),	Properties Solution Explorer
Blink = Entry->Blink; Flink = Entry->Blink;				
expose(ListManager) {				
<pre>invariant(forall(PLIST_ENTRY p; {_vcc_set_in0(p-&gt;Flink,_vcc_owns( (_vcc_set_in0(p-&gt;Flink,_vcc_owns(this)) &amp;&amp; (p-&gt;Flink-&gt;Blink == p))))</pre>	this))} {_vcc_sk of _LIST_MANA	_hack(_vcc_set_i GER fails on wrap	in0(p->Flink,_vcc_owns( ).	this)))} _vcc_set_
<pre></pre>				<b>-</b>
Error List 🗐 Output 🔜 Find Results 1 🔝 Find Symbol Results				
Verification failed.	Ln 197	Col 25	Ch 25	INS

### VCC-Specific Model Viewer

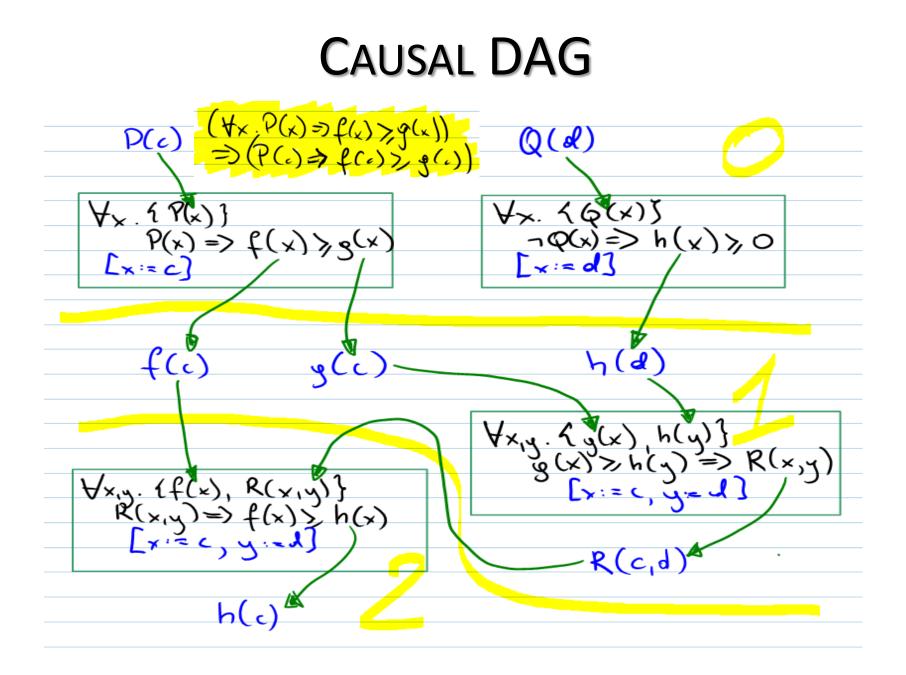
*	/ccDemo - Microsoft Visual Studio				×
F	VCC Model Viewer	-		×	
: 2	Current Model: List.vccmodel			🐟 🐼 🍻	•
	List.c (200:23) : State \$s@10, Timestamp 8910 -	Blink			-
F	Entry: LIST_ENTRY	Line	St	Value	
(0		List.c (180:1)	\$s	(_LIST_ENTRY, 657)	Properties
				(_LIST_ENTRY, 657) (_LIST_ENTRY, 657)	
	i≘ ♀ Blink : _LIST_ENTRY (Aliases: Blink->Blink, Blink->Manager->ListHead, Flink, p @ 			(_LIST_ENTRY, 657)	2
		List.c (201:27) List.c (204:27)		(_LIST_ENTRY, 657) (_LIST_ENTRY, 657)	Solut
	<ul> <li>Blink : _LIST_ENTRY (Aliases: Blink-&gt;Blink, Blink-&gt;Manager-&gt;ListHead, Fl</li> <li>ListManager : _LIST_MANAGER (Aliases: Blink-&gt;Manager, p @ List.c (38)</li> </ul>	List.c (205:13)		(_LIST_ENTRY, 657)	lion E
		List.c (204:27)		(_LIST_ENTRY, 657)	Solution Explorer
	ie ♀ ListManager : _LIST_MANAGER (Aliases: Blink->Manager, p @ List.c (38:0)->Mar Ield Addresses	List.c (209:9) List.c (210:9)		(_LIST_ENTRY, 657) (_LIST_ENTRY, 657)	orer
	ListManager : _LIST_MANAGER (Aliases: Blink->Manager, p @ List.c (38:0)->Manager)	List.c (211:9)	\$s	(_LIST_ENTRY, 657)	
	i ⊕… 🏈 p @ List.c (38:0) : _LIST_ENTRY	List.c (200:23)	\$s	(_LIST_ENTRY, 657)	
(Q					
					J
2					
Re	ady		-		æ

# WORKING WITH VCC

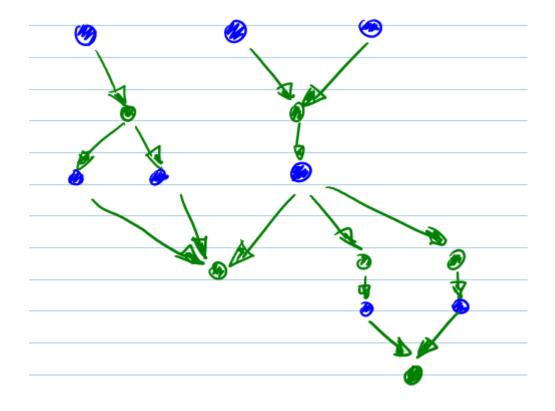
- write a version of the spec
- verify, fail
- add assertions or look at the model to see why it failed
  - for bigger functions, do it a couple of lines at a time, moving focus window down
- repeat



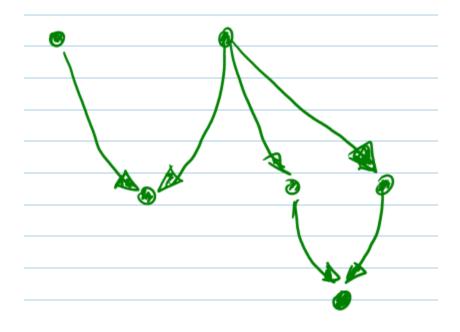
### WANT IT TO GO FASTER? PROFILE!



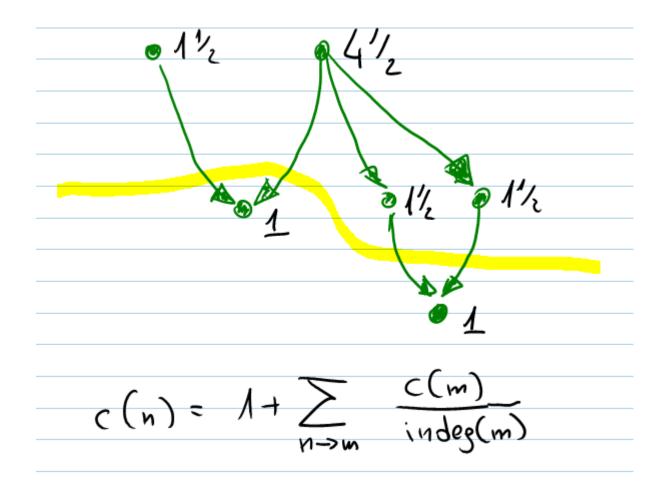
#### COST IN THE CAUSAL DAG



#### COST IN THE CAUSAL DAG



### COST IN THE CAUSAL DAG



### BROWSING THE CAUSAL DAG

<pre>File CSV Help CONFLICTS Quant. by last conflict Scope: 44403 / 0 inst, 0 lits, 1 children [rec: 896, 49 inst/cnfl] G0042 VccPrelude.bpl.240:15 (direct: 5208, cnfl: 362) CVcCPrelude.bpl.240:15 (direct: 5208, cnfl: 362) CVcCPrelude.bpl.240:15 l0 (0) / [9:3] @257401 244.8 \$memory [2:2] call30702formal@#1@0 i VccPrelude.bpl.240:15 l0 (0) / [17:4] @257327 301.3 \$store.mem [5:3] \$dot [5: VcCPrelude.bpl.240:15 l0 (0) / [19:8] @153441 1443.3 \$memory [2:2] call3702formal VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot BLAME BIND VcCPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot Store.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$s@2), call30702formal@#1@0, #x@311451264 Sptr(^_LIST_ENTRY, &amp;select.mem(\$memory(\$s@2), call30702formal@#1@0, #x@311451264 Sptr(^_LIST_ENTRY, &amp;select.mem(\$store.mem(\$memory(\$s@2), call30702formal@#100, #x@311451264 Sptr(^_LIST_ENTRY, &amp;select.mem(\$store.mem(\$memo</pre>	*	Z3 Axiom Profiler	
<pre>i00 CONFLICTS Quant. by last conflict Guant. by l</pre>		File CSV Help	
<pre> • Quant. by last conflict • Scope: 44403 / 0 inst, 0 lits, 1 children [rec: 896, 49 inst/cnf1] • G0842 VccPrelude.bp1.240:15 (direct: 5208, cnf1: 362) • VccPrelude.bp1.240:15 (direct: 5208, cnf1: 362) • VccPrelude.bp1.240:15 (direct: 5208, cnf1: 362) • VccPrelude.bp1 • VccPrelude.bp1 • VccPrelude.bp1 • VccPrelude.bp1.240:15 10 (0) / 9:3] @257401 244.8 \$memory [2:2] call30630formal@#1@0 • VccPrelude.bp1.240:15 10 (0) / 17:4] @257364 245.8 \$store.mem [3:3] \$dot [5: • VccPrelude.bp1.240:15 9 (0) / [64:7] @257327 301.3 \$store.mem [3:5] \$ptr • VccPrelude.bp1.240:15 8 (0) / 19:8] @153444 1443.3 \$memory [2:2] call36 • VccPrelude.bp1.240:15 8 (0) / 19:8] @153444 1443.3 \$memory [2:2] call36 • VccPrelude.bp1.240:15 8 (0) / 19:8] @257277 200.6 \$store.mem [5:3] \$dot • Store.mem(\$store.mem(\$memory(\$sg2), call30702formal@#1@0, #x@311451264 • \$ptr(^_LIST_ENTRY, \$select.mem(\$memory(\$sg2), call30702formal@#1451264 • \$store(.mem(\$store.mem(\$store.mem(\$memory(\$sg2), call30702formal@#1451264 • Store(.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$sg2), call30702formal@#1451264 • Stelect.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$sg2), call30702form • VTELDS • BLND • VTELDS •</pre>		E- CONFLICTS	A .
<pre>     Scope: 44403 / 0 inst, 0 lits, 1 children [rec: 896, 49 inst/cnf1]     Go042 VccPrelude.bpl.240:15 (direct: 5208, cnf1: 362)     DEEP     VccPrelude.bpl.241 forall M:Smemory_t, p:Sptr, q:Sptr, v:int :: {:weight 0}     VccPrelude.bpl.241 forall M:Smemory_t, p:Sptr, q:Sptr, v:int :: {:weight 0}     VccPrelude.bpl.241 forall M:Smemory_t, p:Sptr, q:Sptr, v:int :: {:weight 0}     VccPrelude.bpl.244     VccPrelude.bpl.244     VccPrelude.bpl.244     VccPrelude.bpl.240:15 11 (0) / 9:3] @257401 244.8 Smemory [2:2] call30702formal@#1@0     all30630formal@#1@0     VccPrelude.bpl.240:15 10 (0) / [17:4] @257364 245.8 Sstore.mem [3:5] Sptr     VccPrelude.bpl.240:15 9 (0) / [64:7] @257327 301.3 Sstore.mem [3:5] Sptr     VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 Sstore.mem [5:3] Sdot     BLAME     BIND     Sstore.mem(Sstore.mem(Smemory(Ss@2), call30702formal@#1@0, #x@0311451264)     Sselect.mem(Sstore.mem(Sstore.mem(Sstore.mem(Smemory(Ss@2), call30702formal@#100, #x@0311451264)     Sselect.mem(Sstore.mem(Sstore.mem(Sstore.mem(Sstore.mem(Sstore.mem(Smemory(Ss@2), call30702formal@#100, #x@0311451264)     Sselect.mem(Sstore.mem(Sstore.mem(Sstore.mem(Smemory(Ss@2), call30702formal@#100, #x@0311451264)     Sselect.mem(Sstore.</pre>			=
<pre></pre>			
<pre>     DEEP</pre>			
<pre></pre>		· · · · · · · · · · · · · · · · · · ·	
<pre>Select.mem(\$store.mem(M, p, v), q) == \$select.mem(M, q) fall30630formal@p(</pre>			
<pre>     VccPrelude.bpl.240:15 11 (0) /  9:3  @257401 244.8 \$memory  2:2  call30702formal     ··VccPrelude.bpl.240:15 10 (0) /  17:4  @257364 245.8 \$store.mem  5:3  \$dot  5:     ··VccPrelude.bpl.240:15 9 (0) /  64:7  @257327 301.3 \$store.mem  13:5  \$ptr     ··VccPrelude.bpl.240:15 8 (0) /  19:8  @153444 1443.3 \$memory  2:2  call30     ··VccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$dot     ···VccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$dot     ····VccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$dot     ·································</pre>			mal@#1@0 0formal@p(
<pre>     VccPrelude.bpl.240:15 9 (0) / [64:7] @257327 301.3 \$store.mem [13:5] \$ptr     ··VccPrelude.bpl.240:15 8 (0) / [19:8] @153444 1443.3 \$memory [2:2] call3(     ··VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot     ···VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot     ····VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot     ·································</pre>			
<pre>     VccPrelude.bpl.240:15 8 (0) /  19:8  @153444 1443.3 \$memory  2:2  call3(     VccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:6  @257277 200.6 \$store.mem  5:3  \$do1     OccPrelude.bpl.240:15 8 (0) /  30:</pre>			
<pre>     VccPrelude.bpl.240:15 8 (0) / [30:6] @257277 200.6 \$store.mem [5:3] \$dot     BLAME     BIND     \$store.mem(\$store.mem(\$memory(\$s@2), call30702formal@#1@0, #x@03!1451264)     \$sptr(^_LIST_ENTRY, \$select.mem(\$memory(\$s@9), \$dot(\$ptr(^_LIST_MANAGER, \$     \$ \$dot(\$ptr(^_LIST_ENTRY, Q#p\$1^38.594#dtl@@1!498!275), _LIST_ENTRY.Flink)     * \$select.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$s@2), call30702form</pre>			
<pre>     BLAME     BIND     Sstore.mem(\$store.mem(\$memory(\$s@2), call30702formal@#1@0, #x@@3!145!264)</pre>			
<pre>     Store.mem(Store.mem(Smemory(Ss@2), call30702formal@#1@0, #x@@3!145!264)</pre>			5151 2001
<pre>     \$ptr(^_LIST_ENTRY, \$select.mem(\$memory(\$s@9), \$dot(\$ptr(^_LIST_MANAGER, S     \$dot(\$ptr(^_LIST_ENTRY, Q#p\$1^38.594#dt1@@1!498!275), _LIST_ENTRY.Flink)     \$select.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$s@2), call30702form     \$YIELDS     \$BIND     \$YTFIDS     * </pre>			
<pre>     \$dot(\$ptr(^_LIST_ENTRY, Q#p\$1^38.594#dt1@@1!498!275), _LIST_ENTRY.Flink)     \$select.mem(\$store.mem(\$store.mem(\$store.mem(\$memory(\$s@2), call30702form     \$YIELDS     \$BLAME     \$BIND     \$YTFLDS     \$YTFLD</pre>			
<pre>     \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$ \$\$</pre>			
YIELDS      ································			
BIND     WIFI DS			
VTFLDS			
Image:			-
			•
\$1 \$store.mem(\$store.mem(\$memory(\$s@2), call30702formal@#1@0, #x@@3!145!264), \$dot(call30702formal@#1@0.	4 9	store.mem(\$store.mem(\$memory(\$s@2), call30702formal@#1@0, #x@@3!145!264), \$dot(call30702forma	al@#1@0,
<pre>\$owns_set_field(\$typ(call30702formal@#1@0))), \$ptrset_to_int(owns#22@0))</pre>			
Verification succeeded. Ln 196 Col 1 Ch 1 INS	Vent	cation succeeded. Ln 196 Col 1 Ch 1	INS

### THE INSPECTOR: CONTROL TO THE PEOPLE

### THE INSPECTOR

 ask Z3 to state the current model from time to time

in forms of labels

- translate such models to error messages
- display all possible error message and blink the current one

#### Screenshot

Output	Error Con	sole				
Status	Line#	Source	*	Property	Value	4
	196			num conflicts	1024	
	197	Blink = Entry->Blink;		num decisions	13047	
		>>> (VCC error 8511) Entry->Blink is typed		num propagations	90350	
		>>> (VCC error 8512) Entry->Blink is thread local		num restarts	1024 13047	
	198	Flink = Entry->Flink;		num final checks	0	
		>>> (VCC error 8511) Entry->Flink is typed	-	num theory conflicts	928	
		>>> (VCC error 8512) Entry->Flink is thread local	=	num interface eqs	0	
	199			max generation	6	=
	200	expose(ListManager) {		num mk clause	74383	
		>>> (VCC error 8014) invariant() of _LIST_MANAGER fails on wrap		num mk bin clause	0	
		>>> (VCC error 8014) invariant() of _LIST_MANAGER fails on wrap		num mk lits	242866	
		>>> (VCC error 8014) invariant(forall(_vcc_obj_t p; {_vcc_set_in0(p	0	num min lits	4102	
2		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY e; {_vcc_set_	i	num bs	0	
5		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY e1, e2; {_vcc.		num bsr	0	
3		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY p; {_vcc_set_	i	num fsr	0	
12		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY p; {_vcc_set_	i	num quant inst	41900	
16		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY p; {_vcc_set_	i	num lazy quant inst	0	
11		>>> (VCC error 8014) invariant(forall(PLIST_ENTRY p; {_vcc_set_	i	num missed quant inst	0	
		>>> (VCC error 8014) invariant(_vcc_set_in0(ListHead,_vcc_owns(_		min missed quant cost	0	
		>>> (VCC error 8014) invariant(index[ListHead] == 0) of _LIST_MANA		max missed quant cost	0	
		>>> (VCC error 8014) invariant(index[ListHead->Blink] == size - 1) of _		num arith conflicts	74	
		>>> (VCC error 8016) 'ListManager' is not wrapped before unwrap		num add rows	85413	
		N/CC empr 8510) was owned is writeble in cell to wrendl int Man	Ψ.	num pivots	10570	
<				num arith accarte	<u>81284</u>	

#### **SOME NUMBERS**

## LIMITS ON MATCHING DEPTH

- matching depth on one of the list functions
  - 10 heap related (10 heap updates in the function)
  - 6 user-defined (different levels of expansion of the invariant)
  - total of 17
- the looping example shows that number of instances can be easily exponential with depth

### Scale of Problems

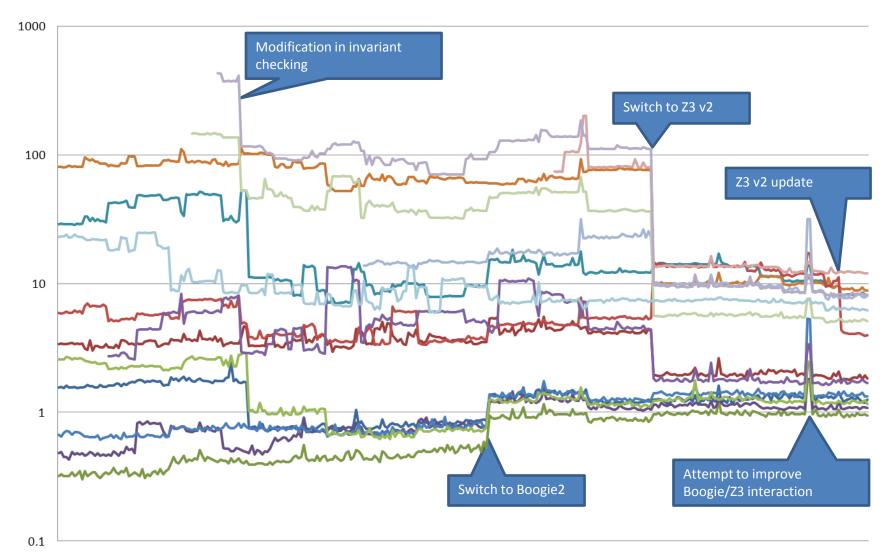
- prelude: 300 quantifiers, 50 multi-triggers
- Hypervisor program-specific background predicate:
  - 300 types, 1500 fields => 13000 axioms
  - type description dwarfs VC for the function itself
- even Z3 E-matching indices didn't take that lightly
- without proper guidance any SMT solver is likely to be lost

#### PERFORMANCE, PERFORMANCE, PERFORMANCE

Experience from the Hyper-V verification

- **successful** verifications:
  - typical: 0.5–500s, average 25s
  - current max: 2 500s
  - all time max: 50 000s (down to 1 000s with Z3v2)
- acceptable time for interactive work: < 30s</li>
- annotations (since Nov 2008):
  - 15 000 lines
  - 400 functions
  - ca. 20% of codebase verified

#### VCC PERFORMANCE TRENDS NOV 08 – MAR 09



#### SUMMARY

## SUMMARY

- program a custom theory for the SMT solver
- like for a "normal" programming language
  - debug models (model viewers)
  - profile traces (axiom profiler)
  - profile with sampling (the inspector)
- but:
  - lack of clear semantics
  - possibly not the best programming model

# VCC IS AVAILABLE!

- source code available for non-commercial purposes at <a href="http://vcc.codeplex.com/">http://vcc.codeplex.com/</a>
   includes the SMT tools!
- further information linked from there

THAMK YOUR