# Software Model Checking: automating the search for abstractions

Thomas Ball Testing, Verification and Measurement Microsoft Research

#### **People Behind SLAM**

Microsoft Research

- Thomas Ball and Sriram Rajamani

Summer interns

- Sagar Chaki, Todd Millstein, Rupak Majumdar (2000)
- Satyaki Das, Wes Weimer, Robby (2001)
- Jakob Lichtenberg, Mayur Naik (2002)
- Georg Weissenbacher, Fei Xie (2003)

Visitors

- Giorgio Delzanno, Andreas Podelski, Stefan Schwoon

#### Windows Partners

- Byron Cook -> MSR Cambridge
- Vladimir Levin, Abdullah Ustuner, Con McGarvey, Bohus Ondrusek
- Jakob Lichtenberg

# Thanks Also to Friends of SLAM

- BLAST
  - Thomas Henzinger
  - Ranjit Jhala
  - Rupak Majumdar
  - Gregoire Sutre
- MOPED
  - Stefan Schwoon
- MAGIC

   Sagar Chaki

# Outline

- Lecture 1
  - automating the search for program abstractions
- Lecture 2
  - predicate abstraction with procedures + pointers
- Lecture 3
  - predicate discovery via interpolants
- Lecture 4
  - relative completeness of abstraction refinement with respect to widening
- Lecture 5
  - predicate abstraction and testing

#### Name as many examples/types of software as you can

Name as many examples/types of software as you can

operating system network protocols document processing

games financial / business

• • •

What major inventions have improved software development in the past 50 years?

What major inventions have improved software development in the past 50 years?

structured programming abstract data types high-level prog. languages version control tools type systems garbage collection

How does a researcher demonstrate that an invention is a good idea?

#### Lessons

#### • Software products are varied, so is development

- Niche: desktop, net, consumer device, command & control
- Relation to other software: first vs nth version, member of family
- Seriousness of purpose: safety critical, prototype, one-use script
- Installation base: all consumers, all PC owners, company-specific

- .

#### • SE researchers produce many research products

- Formalisms, tools and algorithms, yes, but also...
- Processes, methodologies
- Guidance, recipes, patterns, distilled experience
- Formulas for scheduling, cost estimation, quality assessment, ...
- Notations, languages, descriptive tools

#### • Validating a SE invention often harder than inventing it

- True cost effectiveness typically too hard to measure
- Controlled experiments often impossible or too expensive
- Ideas need time to develop before validation stage

#### Automating Verification of Software

- Remains a "grand challenge" of computer science but a "minor player" in practice
- Behavioral abstraction is central to this effort
- Abstractions simplify our view of program behavior
- Proofs over the abstractions carry over to proofs over the program

#### How many program abstractions can you list?

How many program abstractions can you list?



4 3 9

# No "Silver Bullet"

- According to Frederick Brooks, there is no "silver bullet" that will improve software production by an order of magnitude.
- A corollary is that there is no "gold abstraction"
- Development of abstractions is dependent on
  - class of programs
  - class of properties

### The Usefulness of Abstractions

- Prove a theorem and write a paper
- Experimentation
  - Efficiency
    - run-time
    - memory consumption
  - Precision
    - # spurious counterexamples / total # of counterexamples
  - Termination
    - sometimes hard to distinguish from efficiency (or lack thereof)

#### Abstraction Refinement: PLDI'03 Case Study of Blanchet et al.

- "... the initial design phase is an iterative manual refinement of the analyzer."
- "Each refinement step starts with a static analysis of the program, which yields false alarms. Then a manual backward inspection of the program starting from sample false alarms leads to the understanding of the origin of the imprecision of the analysis."
- "There can be two different reasons for the lack of precision:
  - some local invariants are expressible in the current version of the abstract domain but were missed
  - some local invariants are necessary in the correctness proof but are not expressible in the current version of the abstract domain."

#### Software Verification: Search for the Right Abstraction



- A complex search space with a fitness function
- Can a machine beat a human at search?
- Deep Blue beat Kasparov

#### Automating the Search for Abstractions

- A knowledge base of useful abstractions
- A way to generate, combine and refine abstractions

• A fitness function

• A brute force search engine

### Puzzle Pieces

- Application Programming Interfaces (APIs)
- Model checking
- Theorem proving

• Program analysis

### A Brief History of Microsoft



# Model Checking

- Algorithmic exploration of state space of a (finite state) system
- Advances in the past decades:
  - symbolic model checking based on BDDs
    - [Bryant, 1986]
    - [Burch, Clarke, McMillan, Dill, Hwang, 1992]
  - predicate abstraction (parametric analysis)
    - [Graf,Saidi, 1997]
  - symmetry reductions
  - partial order reductions
  - compositional model checking
  - bounded model checking using SAT solvers
- Most hardware companies use a model checker in the validation cycle

# Model Checking

- Strengths
  - Fully automatic (when it works)
  - Computes inductive invariants
    - I such that  $F(I) \Rightarrow I$
  - Provides error traces
- Weaknesses
  - Scale
  - Operates only on models, usually provided by humans

### Theorem proving

- Early theorem provers were proof checkers
  - built to support assertional reasoning
  - cumbersome and hard to use
- Greg Nelson's thesis in early 80s paved the way for automatic theorem provers
  - theories of equality with uninterpreted functions, lists, linear arithmetic
  - combination of the above !
- Automatic theorem provers based on Nelson's work are widely used
  - SAL/ICS, ESC/Java, Proof Carrying Code
- Makes predicate abstraction possible

# Automatic theorem proving

- Strengths
  - Handles unbounded domains naturally
  - Good implementations for
    - equality with uninterpreted functions
    - linear inequalities
    - combination of theories
- Weaknesses
  - Hard to compute fixpoints (no abstraction)
  - Requires inductive invariants
    - Pre and post conditions
    - Loop invariants

# Program analysis

- Originated in optimizing compilers
  - constant propagation
  - live variable analysis
  - dead code elimination
  - loop index optimization
- Type systems use similar analysis – are the type annotations consistent?
- Theory of abstraction interpretation

# Program analysis

- Strengths
  - Works on code
  - Pointer aware
  - Integrated into compilers
  - Precision/efficiency tradeoffs well studied
    - flow (in)sensitive
    - context (in)sensitive
- Weaknesses
  - Abstraction is hardwired and done by the designer of the analysis
  - Not targeted at property checking (traditionally)

# Model Checking, Theorem Proving and Program Analysis

- Very related to each other
- Different histories
  - different emphasis
  - different tradeoffs
- Complementary, in some ways
- Combination can be extremely powerful

#### Stretch!

### **APIs and Usage Rules**



- •Rules in documentation
  - Incomplete, unenforced, wordy
  - Order of ops. & data access
  - Resource management
- •Breaking rules has bad effects
  - System crash or deadlock
  - Unexpected exceptions
  - Failed runtime checks
- •No compile-time checking

#### Socket API

the "communication domain" in which communication is to take place; see protocols(5).

Sockets of type SOCK\_STREAM are full-duplex byte streams, similar to pipes. A stream socket must be in a connected state before any data may be sent or received on it. A connection to another socket is created with a connect(2) call. Once connected, data may be transferred using read(2V) and write(2V) calls or some variant of the send(2) and recv(2) calls. When a session has been completed a close(2V), may be performed. Out-of-band data may also be transmitted as described in send(2) and received as described in recv(2).

The communications protocols used to implement a SOCK\_STREAM insure that data is not lost or duplicated. If a piece of



# The Windows Driver Problem

- Device drivers
  - glue between OS and devices
  - many are kernel plug-ins
  - huge part of PC ecosystem
- Windows Driver Model
  - complex legacy API
  - direct access to Windows kernel
  - low-level binary debugging



# State Machine for Locking





```
state {
   enum {Locked,Unlocked}
   s = Unlocked;
}
```

```
KeAcquireSpinLock.entry {
    if (s==Locked) abort;
    else s = Locked;
}
```

KeReleaseSpinLock.entry {
 if (s==Unlocked) abort;
 else s = Unlocked;



### Example

Does this code obey the locking rule?

```
do {
    KeAcquireSpinLock();
```

```
nPacketsOld = nPackets;
```

```
if(request){
    request = request->Next;
    KeReleaseSpinLock();
    nPackets++;
  }
} while (nPackets != nPacketsOld);
```

KeReleaseSpinLock();

#### Example

Reachability in boolean program

model checker


### Example

Is error path feasible in C program?

theorem prover







**b** : (nPacketsOld == nPackets)

Model checking refined boolean program



## Example

**b** : (nPacketsOld == nPackets)

Model checking refined boolean program



## **Observations about SLAM**

- Automatic discovery of invariants
  - driven by property and a finite set of (false) execution paths
  - predicates are *not* invariants, but observations
  - abstraction + model checking computes inductive invariants (boolean combinations of observations)
- A hybrid dynamic/static analysis that
  - "executes" a finite set of "concrete" paths symbolically
  - explores all paths through abstraction
- A new form of program slicing
  - program code and data not relevant to property are dropped
  - non-determinism allows slices to have more behaviors

SDV Report - Microsoft Internet Explorer																							
Eile Edit View Favorites Tools Help			-	-		15-1																	1
Search 🔶 🛛 📓 😭 🔎 Search 🤺 Favorites	🔊 Media	1	3-		2	•		-23															
Address 🛃 K:\ITP_24\sdv_original\sdv\report.htm																					<b>~</b> E	So 🤇	Links
0					SI	DV I	Repo	ort															
Summary										D	r	V	e	r	S								
Drivers Rules Potential Checks Breakdown - 1167 √ 847 Solve 28 Store 28 Store 28 Store 28 Store 28 Store 28 Store 20 St	26 82 2132 0 28	<u>src/general/cancel/startio</u> Specialization	src/general/ioctl/sys	src/general/toaster/bus	src/general/toaster/func	<u>src/onput/mouser</u> src/general/toaster/toastmon	src/input/pnpi8042/daytona	src/kernel/mca/imca/sys	src/kemel/parport	<u>src/kernel/serial</u> src/kernel/serenum	src/network/modem/fakemodem	<u>src/general/tracedrv/tracedrv</u>	src/input/kbdclass	src/input/kbfiltr	src/input/moufiltr	src/storage/fdc/fdc	<u>src/storage/fdc/flpydisk</u>	src/storage/filters/diskperf	src/wdm/1394/driver/1594diag src/vdd/dosioctl/kinldryr	src/wdm/1394/driver/1394vdev	src/wdm/hid/gameenum	src/general/cancel/sys	ere/general/event/sys
cancelSpinLockstartIoCanceladdDevicelowerDriverReturnTargetRelationNeedsRefDoubleCompletionPrematureSkipKeWaitDeadlockWmiCompleteWmiForwardIrpProcessingCompleteMarkIrpPendingPendedCompletedRequest				$\checkmark - \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark} \mathrel{\checkmark}$		× -		<b>1 1 1 2 1 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2</b> 2				>     >   >     > > > > >					>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	× -	$\checkmark - \checkmark = \checkmark = \checkmark = \checkmark = \checkmark $		> I > > > > > > > > > > > > > > > > > > >		

🗿 SDV Report - Microsoft	Internet Explorer		
<u>File Edit View Favorites</u>	Iools Help		<b>A</b> <u></u>
G Back • 🕑 • 💌	📓 🎧 🔎 Search 🤺 Favorites 🜒 Media	🛛 🖉 🖓 🔄 · 🛄 🗱 🦓	
Address E K:\ITP_24\sdv_orig	ginal\sdv\report.htm	· · · · · · · · · · · · · · · · · · ·	Go Link
0		SDV Report	
S	ummary	Drivers	
Drivers Bules	• Driver: Par	allel port device driver	
Potential Checks	• Rule: Cheo	cks that driver dispatch routines	.c/gen
Breakdown	do not call l	loCompleteRequest() twice on	eral/ eral/
			ever
Chaoler pat starts	the I/O requ	lest packet passed to it by the	el/sy
Errors found	OS or anot	her driver	
cancelSp:		i de la companya de l	
addDevice		v se	/
lowerDriv		🕑 🔪	
TargetRe:		la de la companya de	/ /
DoubleCor		la de la companya de	/
<u>Premature</u>			/
KeWaitDea			/
WmiComple	ete		
TrpProces	ssingComplete		
MarkIrpPe	ending		3 🗸 🗸
PendedCom	npletedRequest		1 1 1 .
<	•		
e		Sector Contraction Contractico	ntranet

#### sdvdefect File Font Trace Tree Source Code Help -Trace Tree - - Source Code DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c init1 ~ init32 ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c init31 116: return PptFdoPower( DevObj, Irp ); p devobj = 117: } else { p devobj t 118: return PptPdoPower( DevObj, Irp ); devobj.Dev 119: devobj two 120: } irp = &har 121: 0 irp->Tail. 122: NTSTATUS sdv main 123: PptDispatchCreateOpen( PDEVICE OBJECT DevObj, PIRP Irp ) { : stub dri 124: PFDO EXTENSION fdx = DevObj->DeviceExtension; 1: if (SLAM 125: P5TraceIrpArrival( DevObj, Irp ); : MakeChoi 126: if ( DevTvpeFdo == fdx->DevTvpe ) { ): switch ( 127: return PptFdoCreateOpen( DevObj, Irp ); : RunDispa 128: } else { 6: sdv IoG 129: return PptPdoCreateOpen( DevObj, Irp ); 56: PIO STA -130: 1 9: end inf 131: } 9: end inf 132: 0 4: SetStat 133: NTSTATUS 9: pirp->C 134: PptDispatchClose ( PDEVICE OBJECT DevObj, PIRP Irp ) { 3: ps->Min 135: PFDO EXTENSION fdx = DevObj->DeviceExtension; 9: stub di P5TraceIrpArrival( DevObj, Irp ); 136: 11: switch 137: if( DevTypeFdo == fdx->DevType ) { |3: ps->Maj 138: return PptFdoClose( DevObj, Irp ); 5: PptDisp 139: } else | 135: PFDO return PptPdoClose( DevObj, Irp ); 140: < 141: Step: 1322 $-142: \}$ State 143: 🛛 -144: NTSTATUS (!(G 145. Phthisnatcheleanun/ PDEVICE ORJECT Devoli PIRP Irn ) ( (completio File: ../../.././dispatchredirect.c, Line: 135, Function 'PptDispatchClose' Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

#### sdvdefect File Font Trace Tree Source Code Help Trace Tree - Source Code Lnit32 DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ~ Lnit31 ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c p devobj = 116: return PptFdoPower( DevObj, Irp ); p devobj t 117: } else { devobj.Dev 118: return PptPdoPower( DevObj, Irp ); devobj two 119: irp = &har 120: } irp->Tail. 121: 0 sdv main 122: NTSTATUS : stub dri 123: PptDispatchCreateOpen( PDEVICE OBJECT DevObj, PIRP Irp ) { 1: if (SLAM 124: PFDO EXTENSION fdx = DevObj->DeviceExtension; 8: MakeChoi 125: P5TraceIrpArrival( DevObj, Irp ); ): switch ( 126: if ( DevTvpeFdo == fdx->DevTvpe ) { : RunDispa 127: return PptFdoCreateOpen( DevObj, Irp ); 6: sdv IoG 128: } else { 56: PIO STA 129: return PptPdoCreateOpen( DevObj, Irp ); 9: end inf -130: 1 9: end inf 131: } 4: SetStat 132: 0 9: pirp->C 133: NTSTATUS 3: ps->Min 134: PptDispatchClose( PDEVICE OBJECT DevObj, PIRP Irp ) { 9: stub di PFDO EXTENSION fdx = DevObj->DeviceExtension; 135: 1: switch P5TraceIrpArrival( DevObj, Irp ); 136: 13: ps->Maj 137: if( DevTypeFdo == fdx->DevType ) { 5: PptDisp 138: return PptFdoClose( DevObj, Irp ); 135: PFDO 139: } else { 137: if( D return PptPdoClose( DevObj, Irp ); 140: < 1 > 141: Step: 1323 $-142: \}$ State 143: 🛛 -144: NTSTATUS (!(G 145. Phthisnatcheleanun/ PDEVICE ORJECT Devoli PIRP Irn ) ( (completio File: ../../.././dispatchredirect.c, Line: 137, Function 'PptDispatchClose' Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

#### sdvdefect File Font Trace Tree Source Code Help Trace Tree - Source Code init31 DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ~ p devobj = ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c p devobj t 116: return PptFdoPower( DevObj, Irp ); devobj.Dev 117: } else { devobj two 118: return PptPdoPower( DevObj, Irp ); irp = &har 119: irp->Tail. 120: } sdv main 121: 0 : stub dri 122: NTSTATUS 1: if (SLAM 123: PptDispatchCreateOpen( PDEVICE OBJECT DevObj, PIRP Irp ) { 8: MakeChoi 124: PFDO EXTENSION fdx = DevObj->DeviceExtension; : switch ( 125: P5TraceIrpArrival( DevObj, Irp ); : RunDispa 126: if ( DevTvpeFdo == fdx->DevTvpe ) { 6: sdv IoG 127: return PptFdoCreateOpen( DevObj, Irp ); 56: PIO STA 128: } else { 59: end inf 129: return PptPdoCreateOpen( DevObj, Irp ); 9: end inf -130: 1 4: SetStat 131: } 9: pirp->C 132: 0 3: ps->Min 133: NTSTATUS 9: stub di 134: PptDispatchClose( PDEVICE OBJECT DevObj, PIRP Irp ) { 1: switch PFDO EXTENSION fdx = DevObj->DeviceExtension; 135: |3: ps->Maj P5TraceIrpArrival( DevObj, Irp ); 136: 5: PptDisp 137: if ( DevTypeFdo == fdx->DevType ) { 135: PFDO : 138: return PptFdoClose( DevObj, Irp ); 137: if( D 139: } else | 138: PptFd 140: return PptPdoClose( DevObj, Irp ); < > 141: Step: 1324 $-142: \}$ State 143: 🛛 -144: NTSTATUS (!(G 145. Phthisnatcheleanun/ PDEVICE ORJECT Devoli PIRP Irn ) ( (completio File: ../../.././dispatchredirect.c, Line: 138, Function 'PptDispatchClose' Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

sdvdefect	
e Font Trace Tree Source Code Help	
Trace Tree Source Code	
RunDispate DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c sdv_IoGet ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
<pre>1: #include "pch.h" end_info end_info 2: end_info 3: NTSTATUS SetStatus 4: PptFdoClose( pirp-&gt;Can ps-&gt;Minor 6: IN PIRP Irp stub_disp 7: ) switch (x 8: {     ps-&gt;Major     PptDispat     10: NTSTATUS     status;     if ( Dev</pre>	
<pre>12: PAGED_CODE(); 13: 14: // 12: do_pag 15: // Verify that our device has not been SUPRISE_REMOVED. Generally 19: if( fd 16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and 17: // parallel ports in docking stations will be surprise removed. 1782: Iri 1783: Iri 19: if( fdx-&gt;PnpState &amp; PPT_DEVICE_SURPRISE_REMOVED ) { 1784: sd 20: // 1784: sd 21: // Our device has been SURPRISE removed, but since this is only 1785: ref 22: // a CLOSE, SUCCEED anyway. 1785: Ref 23: P4Comp 24: status = P4CompleteRequest( Irp, STATUS_SUCCESS, 0 ); 1797: P4 25: 26: goto target_exit; 28:</pre>	
(!(G (completio)) File://./fdoclose.c, Line: 9, Function 'PptFdoClose'	~
river: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling loCompleteRequest twice	

	×
File Font Trace Tree Source Code Help	
-Trace TreeSource Code	_
RunDispate       DoubleCompletion.slic       parallel.h       pdopnp.c       datalink.c       debug.c       sdv-harness.c       fdowmi.c         sdv_IoGet       ieee1284.c       fdopnp.c       wdmguid.h       ntddpar.h       parport.c       dispatchredirect.c       fdoclose.c       utils.c	_ 1
end_info 1: #include "pch.h"	
end_info 3: NTSTATUS	
SetStatus 4: PptFdoClose(	
pirp->Can 5: IN PDEVICE OBJECT DeviceObject,	
ps->Minor 6: IN PIRP Irp	
stub_alsp 7: )	
ps->Major	
PptDispat 9: PFDO_EXTENSION fdx = DeviceObject->DeviceExtension;	
S: PFDO EX	
7: if ( Dev 12. PAGED CODE ().	
18: PptFdoC 13:	
9: PFDO_EX' 14: //	
12: do_page 15: // Verify that our device has not been SUPRISE REMOVED. Generally	
19: if ( fd:	
17: // parallel ports in docking stations will be surprise removed.	
-1783: Tr	
<pre># 1784: SL:</pre>	
-1784: sd	
-1785: re 22. // a CLOSE SUCCEED anyway	
1785: Re 23: //	
63: P4Comp24: status = P4CompleteRequest(Irp, STATUS SUCCESS, 0);	
□ 1797: P4( 25:	
C 26: goto target exit;	
Step: 1328 27: }	
State 28:	
(!(G	
(completio	
File://./fdoclose.c, Line: 12, Function 'PptFdoClose'	

🔜 sdvdefect
File Font Trace Tree Source Code Help
Trace Tree Source Code
end_info       DoubleCompletion.slic       parallel.h       pdopnp.c       datalink.c       debug.c       sdv-harness.c       fdowmi.c         end_info       ieee1284.c       fdopnp.c       wdmguid.h       ntddpar.h       parport.c       dispatchredirect.c       fdoclose.c       utils.c         setStatus       1: #include "pch.h"       2:       3: NTSTATUS       3: NTSTATUS       4: PptFdoClose(
Switcen (x)       5: IN PDEVICE_OBJECT DeviceObject,         ps->Major       6: IN PIRP         PptDispat       7: )         S: PFDO_EX       8: (
<pre>:7: if( Dev 9: PFDO_EXTENSION fdx = DeviceObject-&gt;DeviceExtension; :8: PptFdoC 10: NTSTATUS status; 9: PFDO_EX' 11: 12: do_pag' 12: PAGED_CODE(); 19: if( fd 12:</pre>
24: P4Comp       14: //         -1782: Ir]       14: //         -1783: Ir]       15: // Verify that our device has not been SUPRISE_REMOVED. Generally         -1783: Ir]       16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and         -1784: SL       17: // parallel ports in docking stations will be surprise removed.         -1785: re       19: if(fdx->PnpState & PPT_DEVICE_SURPRISE_REMOVED) {         -1785: Re       20: //
<pre>b3: P4Comp 1797: P4C 1797: P4C 1782: 22: // a CLOSE, SUCCEED anyway. 1783: 23: // 1784: 24: status = P4CompleteRequest( Irp, STATUS_SUCCESS, 0 ); 1784: 25:</pre>
26:       goto target_exit;         Step: 1334       27:         State       28:         29:
(completio File:///./fdoclose.c, Line: 19, Function 'PptFdoClose' Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.



🔜 sdydefect	
File Font Trace Tree Source Code Help	
-Trace Tree Code	
end_info = DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c end_info =ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
1775: P4CompleteRequest (	^
pirp->Cance 1776: IN PIRP Irp,	
ps->MinorFu 1777: IN NTSTATUS Status,	
stub_dispat 1778: IN ULONG PTR Information	
switch (x) 1779: )	
bs->MajorFu 1780: {	
PptDispatch 1781: P5TraceIrpCompletion(Irp);	
PFDO_EXTE 1782: Irp->IoStatus.Status = Status;	
if ( DevTy 1783: Irp->IoStatus.Information = Information;	
PptFdoClo 1784: IoCompleteRequest(Irp, IO NO INCREMENT);	
PFDO_EXTE 1785: return Status;	
: do_paged1786: }	
: if ( fdx- 1787:	
1: P4Comple 1788: D	
1782: Irp-1 1789: NTSTATUS	
1783: Irp-: 1790: P4CompleteRequestReleaseRemLock (	
1784: SLIC 1791: IN PIRP Irp.	
1784: sdv1792: IN NTSTATUS Status.	
1785: retu 1793. IN HIGH PTR Information.	
1785: Retu 1794. IN PIO REMOVE LOCK RemLock	
: P4Comple	
1797: P4Col 1796.	
-1782: Irj 1797. PACompletePequest (Irp Status Information):	
-1783: Irj 1798. PhtPeleesePemovelock/ Pemlock Trp ):	
1784: SL: 1790. return Status.	
Inn.	
Step: 1338 1001.	
- State 1001.	
1902.	
(!(G	>
(completio	
Driver: grokkernelkparpent	vice

🖶 sdvdefect		
File Font Trace Tree Sou	irce Code Help	
Trace Tree	Source Code	
end_info = 🙆 end_info =	DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
BetStatus	1775: P4CompleteRequest(	~
pirp->Cance	1776: IN PIRP Irp,	
ps->MinorFu	1777: IN NTSTATUS Status,	
stub_dispat	1778: IN ULONG PTR Information	
switch (x)	1779.	
ps->MajorFu	1780: {	
PptDispatch	1781: P5TraceTrnCompletion( Trn ):	
PFDO_EXTE	$1782: Trp_NToStatus Status = Status:$	
if( DevTy	1783: Trp->ToStatus Information = Information:	
PptFdoClo	1784: ToCompleteRequest ( Irp. TO NO INCREMENT ):	
PFDO_EXTE	1785: return Status:	
: do_paged		
: if( fdx-	1797.	
1: P4Comple	1700· Π	
1782: Irp-:		
1783: Irp-1	1709. NISIAIUS	_
1784: SLIC	1790: P4CompileLeRequestReleaseRemLock (	
1784: sdv 📃	1791. IN PIRP IEP,	
1785: retu	1792: IN NTSTATUS Status,	
1785: Retu	1793: IN OLONG_PTR Information,	
: P4Comple	1794: IN PIO_REMOVE_LOCK REMLOCK	
1797: P4Co	1795: )	
-1782: Iri	1796: {	
-1783: Iri	1797: P4CompleteRequest( Irp, Status, Information );	
⊕ 1784: SL	1798: PptReleaseRemoveLock( RemLock, Irp );	
	1799: return Status;	
Stop: 1220	1800: }	
acep. 1339	1801:	
State	1802:	
(!(G 🔥	1803: // pcutil.c follows:	×
(completio 🚍		
J ı	File:////utils.c, Line: 1783, Function 'P4CompleteRequest'	
Driver: src\ker	nel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twi	ce.

🖶 sdvdefect		
File Font Trace Tree Sou	irce Code Help	
Trace Tree	-Source Code	
end_info = 🙆 end_info =	DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
BetStatus	1775: P4CompleteRequest(	~
pirp->Cance	1776: IN PIRP Irp,	
ps->MinorFu	1777: IN NTSTATUS Status,	
stub_dispat	1778: IN ULONG PTR Information	
switch (x)	1779:	
þs-≻MajorFu	1780 - {	
PptDispatch	1781: P5TraceIrpCompletion(Irp):	
PFDO_EXTE	1782. Trp->ToStatus Status = Status:	
if(DevTy	1783: Irp->IoStatus Information = Information:	
PptFdoClo	1784: Information Information	
PFDO_EXTE	1785: return Status:	
: do_paged	1786: 1	
: if( fdx-	1787.	
1: P4Comple	1700- П	
1782: Irp-:	1799. H	
1783: Irp-:	1700. R/CompletePequestPeleasePemLock/	-
1784: SLIC	1790. FACOMPIECEREQUESTREIEdSEREMLOCK	
1784: sdv	1791. IN PIRP ILP,	
1785: retu	1792. IN NISTATOS Status,	
1785: Retu	1793. IN OLONG_FIR INFORMATION,	
: P4Comple	1794: IN PIO_REMOVE_LOCK REMLOCK	
1797: P4Co	1795: )	
-1782: Ir	1790: {	
-1783: Ir	1797: P4CompleteRequest( irp, Status, information );	
⊕ 1784: SL	1798: PptReleaseRemoveLock(RemLock, Irp);	
	1799: return Status;	
Sten: 1341		
300p. 1341	1801:	
State	1802:	1200
(!(G 🔼	1803: // pcutil.c follows:	×
(completio 🥃		
J 1 1 1 O 🖾	File:////utils.c, Line: 1/84, Function 'P4CompleteRequest'	
Driver: src\ker	nel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twic	ce. /

🔚 sdydefect	×
File Font Trace Tree Source Code Help	
Trace Tree Source Code	-
end_info = DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c end_info =ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
SetStatus	1
pirp->Cance 1776: IN PIRP Irp.	1
ps->MinorFu 1777 IN NTSTATUS Status.	
stub_dispat 1778. IN HIGHIGS Endeal,	
switch (x)	
ps->MajorFu	
PptDispatch 1781. P5TraceIrpCompletion(Irp.):	
PFDO_EXTE 1782: Trp->ToStatus Status = Status:	
if ( DevTy 1783: Trp->ToStatus Information = Information:	
PptFdoClo 1784: ToCompleteRequest ( Trp. TO NO INCREMENT ):	
PFDO_EXTE 1785: return Status:	
: do_paged	
: if( fdx- 1787.	
: P4Comple	
1782: Irp-: 1789. MTSTATUS	
1783: Irp-: 1790. PACompleteRequestReleaseRemLock/	1
1784: SLIC	
1784: sdvI792. IN NUSURATUS Status	
1785: retu 1793. IN MISIATOS Status,	
1785: Retu 1794. IN DIO REMOVE LOCK RemLock	
: P4Comple	
1797: P4Col 1796.	
-1782: Irj 1797. P4CompleteRequest (Irp Status Information):	
-1783: Irj 1798. PrtReleaseRemoveLock (RemLock Trp.):	
1784: SL: 1799. return Status:	
Step: 1356	
-State 1802.	
1802: 1803: // poutil o follows:	
(G C C C C C C C C C C C C C C C C C C C	1
(completio File:///////	
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling ToCompleteRequest twice.	

🖶 sdvdefect	
File Font Trace Tree Sou	irce Code Help
Trace Tree	Source Code
end_info 🧕 end_info	DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c
SetStatus	24: status = P4CompleteRequest( Irp, STATUS SUCCESS, 0 );
pirp->Can	25:
ps->Minor	26: goto target exit;
stub_disp	27:
switch (x	28:
ps->Major	29.
PptDispat	30. //
5: PFDO_EX	31: // Try to acquire RemoveLock to prevent the device object from going
7: if( Dev	32: // away while we're using it
8: PptFdoC	22. // dwdy while we le using it.
9: PFDO_EX'	24. status - PotAcquirePemoveLock/(fdv_>PemoveLock Irp).
12: do_pag	25. if/ NTT SUCCESS ( status ) ) (
19: if( fd:	26: // Our douido had been neweyed but dinde this is only a CLOSE SUCCEED anymous
24: P4Comp	27. atotuc - Smamuc success.
-1782: Irj	20. rota target ouit.
-1783: Irj	38: goto target_exit;
± 1784: SL	39: )
-1784: sd	
-1785: re	
-1785: Re	42: // We have the RemoveLock
63: P4Comp	43: 77
⊨ 1797: P4	44:
-1782:	45: ExAcquireFastMutex(&fdx->OpenCloseMutex);
1783	46: if(fdx->OpenCloseRefCount > 0 ) {
±1784·	47: //
	48: // prevent rollover - strange as it may seem, it is perfectly
Chara 1005	49: // legal for us to receive more closes than creates - this
step: 1335	50: // info came directly from Mr. PnP himself
State	51: //
(!(G	52: if(((LONG)InterlockedDecrement(&fdx->OpenCloseRefCount)) < 0) {
(completio	
J	File:///./fdoclose.c, Line: 24, Function 'PptFdoClose'
Driver: src\ke	nel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

🖶 sdvdefect	
File Font Trace Tree Sou	rce Code Help
-Trace Tree-	-Source Code
if (SLAM NA	DoubleCompletion slic parallel h ndoppp c datalink c debug c sdy-harness c fdowmi c
MakeChoice	joss1204 al fdenne aludravid bletddeen bleeneett al digestebedieset al fdogloge glutila al
switch (ch	Teee1284.c  Tdophp.c  wdmguId.n  htddpar.n  parport.c  dispatchredirect.c Tdociose.c utiis.c
RunDispatc	38: goto target_exit;
sdy ToGet	39: }
PTO STACK	40:
end info	41: //
end_info	42: // We have the RemoveLock
SetStatus	43: //
pirp->Can	44:
prep / can	45: ExAcquireFastMutex(&fdx->OpenCloseMutex);
stub disp	46: if(fdx->OpenCloseRefCount > 0 ) {
switch (x	47: //
ns->Maior	48: // prevent rollover - strange as it may seem, it is perfectly
PotDispat	49: // legal for us to receive more closes than creates - this
15. PEDO EX	50: // info came directly from Mr. PnP himself
7: if/ Dev	51: //
18: PptFdoC	52: if( ((LONG)InterlockedDecrement(&fdx->OpenCloseRefCount)) < 0 ) {
Q. PEDO EV	53: // handle underflow
12 do page	54: InterlockedIncrement(&fdx->OpenCloseRefCount);
$12 \cdot d0_pag$	55: }
24: B/Comp	56: }
52: PAComp	57: ExReleaseFastMutex(&fdx->OpenCloseMutex);
± 1797 · ₽/(	58:
1797	59: target_exit:
1702.	60:
- 1703	61: DD((PCE)fdx,DDT,"PptFdoClose - OpenCloseRefCount after close = %d\n",fdx->OpenCloseRe
± 1/04.	62:
	63: return P4CompleteRequestReleaseRemLock( Irp, STATUS_SUCCESS, 0, &fdx->RemoveLock );
Step: 1359	64: }
State	65:
(G	
(completio	
J	File:///./fdoclose.c, Line: 63, Function 'PptFdoClose'
Driver: src\ker	nel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

🖶 sdvdefect		
File Font Trace Tree Sou	urce Code Help	
Trace Tree	Source Code	
E (SLAM NT 📥	DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c	
akeChoice	ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c	
witch (choi	1775: P4CompleteRequest(	~
unDispatchF	1776: IN PIRP Trp.	
sdv_IoGetCu	1777: IN NTSTATUS Status.	
PIO_STACK_L	1778: IN ULONG PTR Information	
end_info =	1779: )	
end_info =	1780: {	
βetStatus	1781: P5TraceTrpCompletion(Trp);	
pirp->Cance	1782: Irp->ToStatus.Status = Status:	
⊳s->MinorFu	1783: Trp->ToStatus.Information = Information:	
stub_dispat	1784: ToCompleteRequest(Irp, TO NO INCREMENT):	
switch (x)	1785: return Status:	
ps->MajorFu	1786: }	
PptDispatch	1787:	
PFDO_EXTE	1788: П	
if( DevTy	1789: NTSTATUS	
PptFdoClo	1790: P4CompleteRequestReleaseRemLock (	
PFDO_EXTE	1791: IN PIRP Irp.	
: do_paged	1792: IN NTSTATUS Status.	
: if( fdx-	1793. IN HIGHING PTR Information.	
1: P4Comple	1794: IN PTO REMOVE LOCK RemLock	
: P4Comple	1795· )	
1797: P4Co	1796: {	
-1782: Irj	1797: P4CompleteRequest( Irp, Status, Information ):	
-1783: Irj	1798: PptReleaseRemoveLock(RemLock, Trp.):	
🗄 1784: SL:	1799: return Status;	
	1800: }	
Step: 1362	1801:	
State	1802:	
	1803: // pcutil.c follows:	~
(G		>
	File:////utils.c, Line: 1797, Function 'P4CompleteRequestReleaseRemLock'	
Driver: src\ker	rnel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twi	ce.

🔜 sdydefect				
File Font Trace Tree Source Code Help				
-Trace Tree	Source Code			
(SLAM_NT_SLA eChoice tch (choice	DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c			
DispatchFur /_IoGetCurr D_STACK_LOC d_info = st d_info = st cStatus cp->CancelR ->MinorFunc ub_dispatch itch (x) { ->MajorFunc	<pre>1776: IN PIRP Irp, 1776: IN NTSTATUS Status, 1777: IN NTSTATUS Status, 1778: IN ULONG_PTR Information 1779: ) 1780: { 1781: P5TraceIrpCompletion(Irp); 1782: Irp-&gt;IoStatus.Status = Status; 1783: Irp-&gt;IoStatus.Information = Information; 1784: IoCompleteRequest(Irp, IO_NO_INCREMENT); 1785: return Status;</pre>			
DispatchCl PFDO_EXTENS If(DevType PptFdoClose PFDO_EXTENS do_paged_c if(fdx->P P4Complete P4Complete 97: P4Comp	<pre>1786: } 1787: 1787: 1788: D 1789: NTSTATUS 1790: P4CompleteRequestReleaseRemLock( 1791: IN PIRP Irp, 1792: IN NTSTATUS Status, 1793: IN ULONG_PTR Information, 1794: IN PIO_REMOVE_LOCK RemLock 1795: ) 1796: {</pre>			
1782: Irp- 1783: Irp- 1784: SLIC. Step: 1365 State (G (completio	<pre>1797: P4CompleteRequest(Irp, Status, Information); 1798: PptReleaseRemoveLock(RemLock, Irp); 1799: return Status; 1800: } 1801: 1802: 1803: // pcutil.c follows:</pre>			
Driver: src\ker	File://///utils.c, Line: 1782, Function 'P4CompleteRequest' mel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.			

🔜 sdydefect				
File Font Trace Tree Source Code Help				
-Trace Tree Source Code				
(SLAM_NT_SLA       DoubleCompletion.slic       parallel.h       pdopnp.c       datalink.c       debug.c       sdv-harness.c       fdowmi.c         eChoice       ieee1284.c       fdopnp.c       wdmguid.h       ntddpar.h       parport.c       dispatchredirect.c       fdoclose.c       utils.c				
<pre>tch (choice 1775: P4CompleteRequest( 1776: IN PIRP Irp, 1776: IN NTSTATUS Status, 1777: IN NTSTATUS Status, 1777: IN NTSTATUS Status, 1777: IN ULONG_PTR Information 1.info = st 1779: ) 1.info = st 1779: ) 1.info = st 1780: { 1.Status 1781: P5TraceIrpCompletion(Irp); 1.rp-&gt;CancelR 1782: Irp-&gt;IoStatus.Status = Status; -&gt;MinorFunc 1783: Irp-&gt;IoStatus.Information = Information; 1.d_dispatch 1784: IoCompleteRequest(Irp, IO_NO_INCREMENT); 1.reth (x) { 1.785: return Status; 1.787:</pre>				
1788: U1789: NTSTATUSPptFdoClose1790: F4CompleteRequestReleaseRemLock(FD0_EXTENS1791: IN PIRP1792: IN NTSTATUSstatus,if (fdx->P1793: IN ULONG_PTR1794: IN PIO_REMOVE_LOCK RemLockP4Complete1795: )97: P4Comp1796: {1782: Irp-:1797: P4CompleteRequest(Irp, Status, Information);1783: Irp-1798: PptReleaseRemoveLock(RemLock, Irp);1784: SLIC.1800: }Step: 13661801:				
State (G (completio) File:///utils.c, Line: 1783, Function 'P4CompleteRequest' Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling ToCompleteRequest twice				

🔜 sdydefect				
File Font Trace Tree Source Code Help				
-Trace Tree Source Code				
(SLAM_NT_SLA       DoubleCompletion.slic       parallel.h       pdopnp.c       datalink.c       debug.c       sdv-harness.c         eChoice       ieee1284.c       fdopnp.c       wdmguid.h       ntddpar.h       parport.c       dispatchredirect.c       fdocloc         tch       (choice       1775:       p4CompleteRequest (	fdowmi.c  ose.c utils.c			
DispatchFur1775: P4CompleteRequest(pispatchFur1776: IN PIRP Irp,r_IoGetCurr1777: IN NTSTATUS Status,p_STACK_LOC1778: IN ULONG_PTR Informationd_info = st1779: )d_info = st1779: )d_info = st1780: {cStatus1781: P5TraceIrpCompletion(Irp);cp->CancelR1782: Irp->IoStatus.Status = Status;>MinorFunc1783: Irp->IoStatus.Information = Information;ub_dispatch1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);itch (x) {1785: return Status;->MajorFunc1786: )tDispatchCl1786: )				
1787:PFDO_EXTENS1788: □If ( DevType1789: NTSTATUSPptFdoClose1790: P4CompleteRequestReleaseRemLock(PFDO_EXTENS1791: IN PIRPIrp,do_paged_c1792: IN NTSTATUSStatus,if ( fdx->P1793: IN ULONG_PTRInformation,P4Complete1794: IN PIO_REMOVE_LOCKP4Complete1795: )97: P4Comp1796: {1782: Irp-:1797: P4CompleteRequest( Irp, Status, Information );				
1783: Irp-:       1798: PptReleaseRemoveLock(RemLock, Irp);         1784: SLIC       1799: return Status;         Step: 1368       1800: }         State       1802:         (G       1803: // pcutil.c follows:         (File:///////utils.c, Line: 1784, Function 'P4CompleteRequest'         Driver: src/kernel/parport       Rule: DoubleCompletion				

## **SLAM Results**

- Boolean program model has proved itself
- Successful for device driver contracts
  - control-dominated safety properties
  - few boolean variables needed to do proof or find real errors
- Counterexample-driven refinement
  - terminates in practice
  - incompleteness of theorem prover not an issue

### SLAMming on the shoulders of ...

- Model checking
  - predicate abstraction
  - counterexample-driven refinement
  - BDDs and symbolic model checking
- Program analysis
  - abstract interpretation
  - points-to analysis
  - dataflow via CFLreachability

- Automated deduction
  - weakest preconditions
  - theorem proving
- Software
  - AST toolkit
  - Das's Golf
  - CU and CMU BDD
  - Simplify
  - OCaml

# SLAM/SDV History

#### • 2000-2001

- foundations, algorithms, prototyping
- papers in CAV, PLDI, POPL, SPIN, TACAS
- March 2002
  - Bill Gates review

### • May 2002

- Windows committed to hire two Ph.D.s in model checking to support Static Driver Verifier
- July 2002
  - running SLAM on 100+ drivers, 20+ properties

- September 3, 2002
  - made initial release of SDV to Windows (friends and family)

#### • April 1, 2003

 made wide release of SDV to Windows (any internal driver developer)

#### • September, 2003

- team of six in Windows working on SDV
- researchers moving into "consultant" role
- November, 2003
  - demonstration at Driver Developer Conference

Release on DDK in late 2004!

# Summary

• Use APIs and properties to guide search for appropriate abstractions

• Predicate abstraction provides parametric abstraction algorithm

 Predicates generated by analysis of spurious counterexamples











# Glossary

Model checking	Checking properties by systematic exploration of the state-space of a model. Properties are usually specified as state machines, or using temporal logics
Safety properties	Properties whose violation can be witnessed by a finite run of the system. The most common safety properties are invariants
Reachability	Specialization of model checking to invariant checking. Properties are specified as invariants. Most common use of model checking. Safety properties can be reduced to reachability.
Boolean programs	"C"-like programs with only boolean variables. Invariant checking and reachability is decidable for boolean programs.
Predicate	A Boolean expression over the state-space of the program eg. (x < 5)
Predicate abstraction	A technique to construct a boolean model from a system using a given set of predicates. Each predicate is represented by a boolean variable in the model.
Weakest precondition	The weakest precondition of a set of states S with respect to a statement T is the largest set of states from which executing T, when terminating, always results in a state in S.




13 790 -7 395 -7 1186 -7 593 -7 1780 526  $\rightarrow 263$ 890 -7 445 -7 1336 -7 668 -7 334 -7 167 -7 502 ~ 377-71132 ~ 566-7283-> -7 2BI -7 754 19 20 ~7850 -7 425 -7 1276 -7 638 -7 319 -7956 -7 -> 289-7868-7434-7217-7652-) -7 478 -> 163 -> 490 -> 245 -> 736 -> 368 -> 189 326 92 7 46 -7 23 27