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?

control	numeric	string	heap	...
FSM	odd/even	reg. exp.	...	
PDA	{−, 0, +}			
...				

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

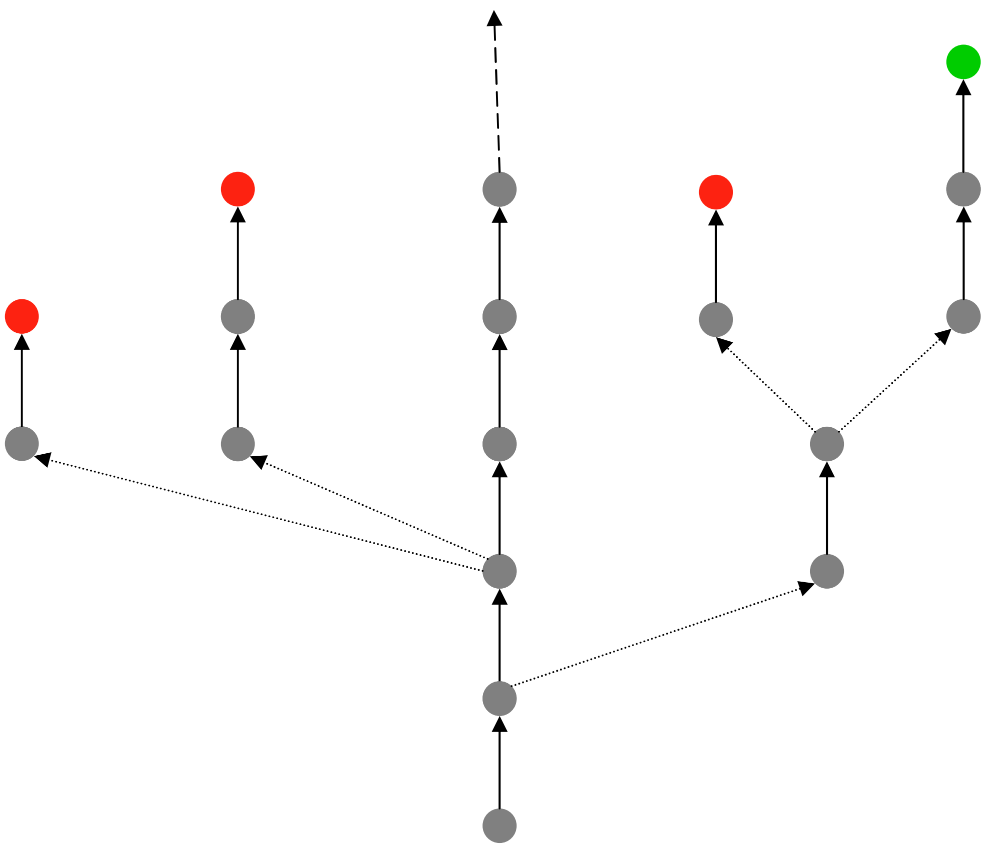
  - Precision
    - # spurious counterexamples / total # of counterexamples

  - 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

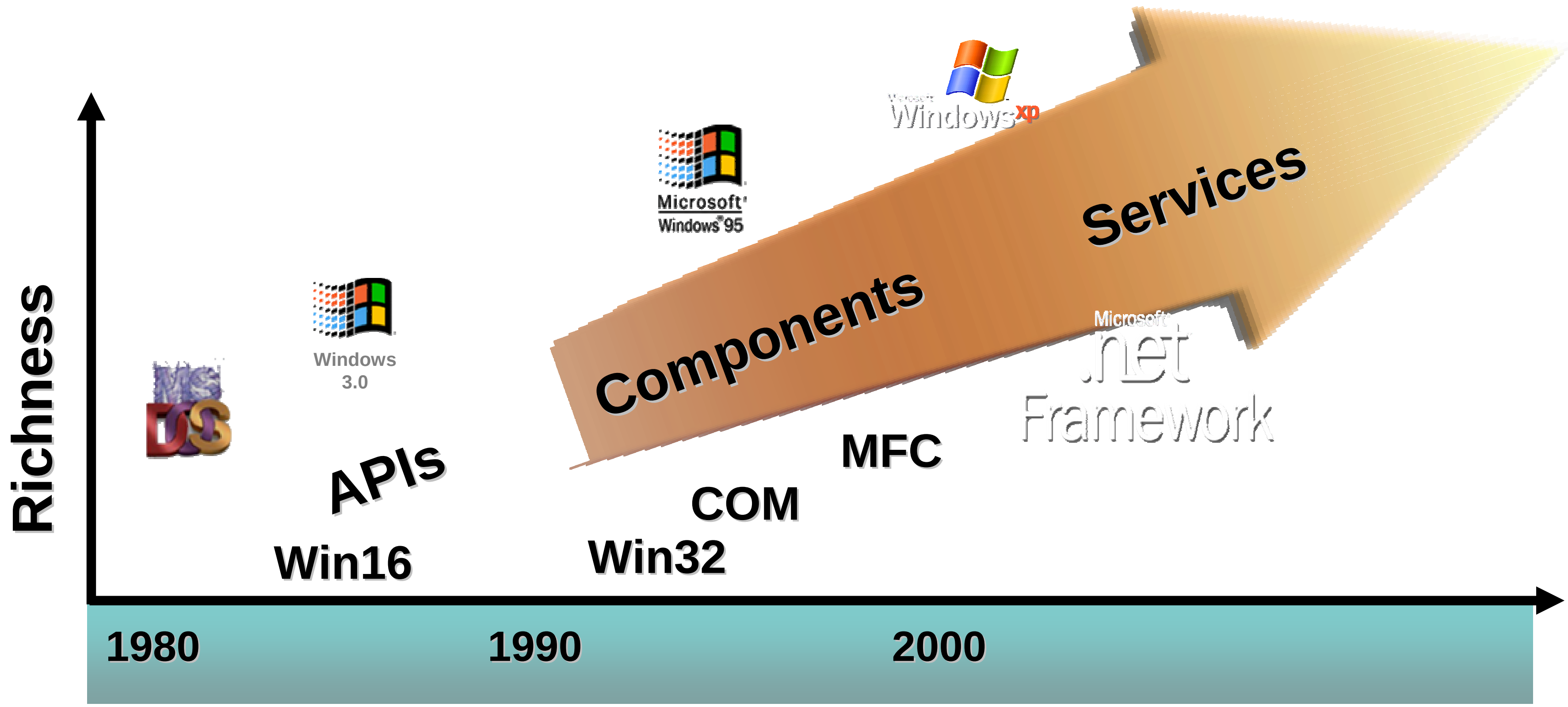
Automating the Search for Abstractions

- 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

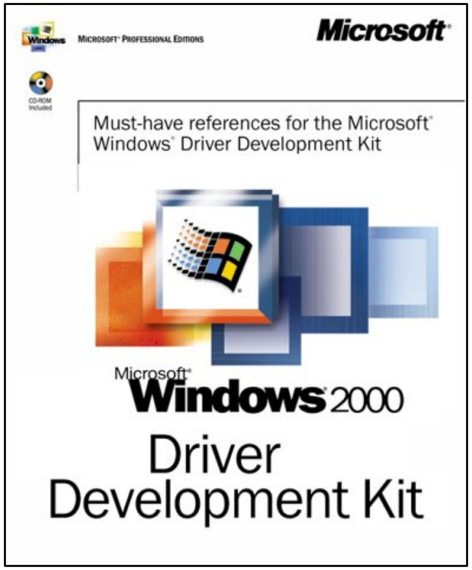
- Strengths
  - Works on code
  - Pointer aware
  - Integrated into compilers
  - Precision/efficiency tradeoffs well studied
    - flow (in)sensitive
    - context (in)sensitive
- Weaknesses
  - Abstraction is hardwired and done by the designer of the analysis
  - Not targeted at property checking (traditionally)

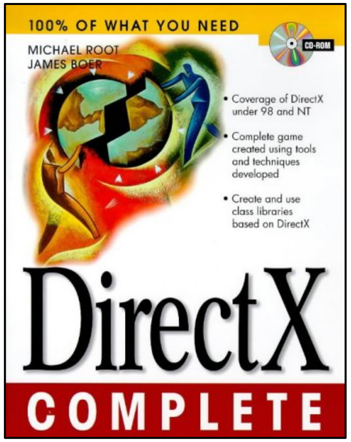
Model Checking, Theorem Proving and Program Analysis

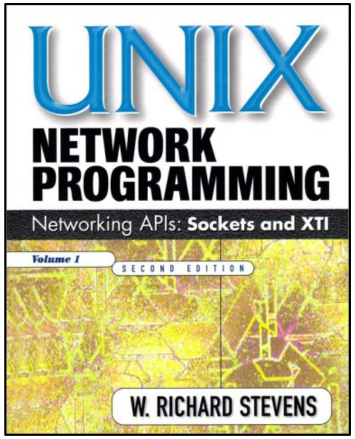
- Very related to each other
- Different histories
  - different emphasis
  - different tradeoffs
- Complementary, in some ways
- Combination can be extremely powerful

Stretch!

APIs and Usage Rules

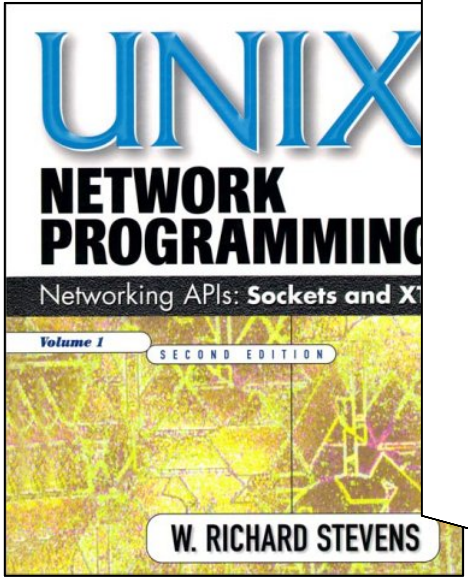






- Rules in documentation
  - Incomplete, unenforced, wordy
  - Order of ops. & data access
  - Resource management
- Breaking rules has bad effects
  - System crash or deadlock
  - Unexpected exceptions
  - Failed runtime checks
- *No compile-time checking*

Socket API

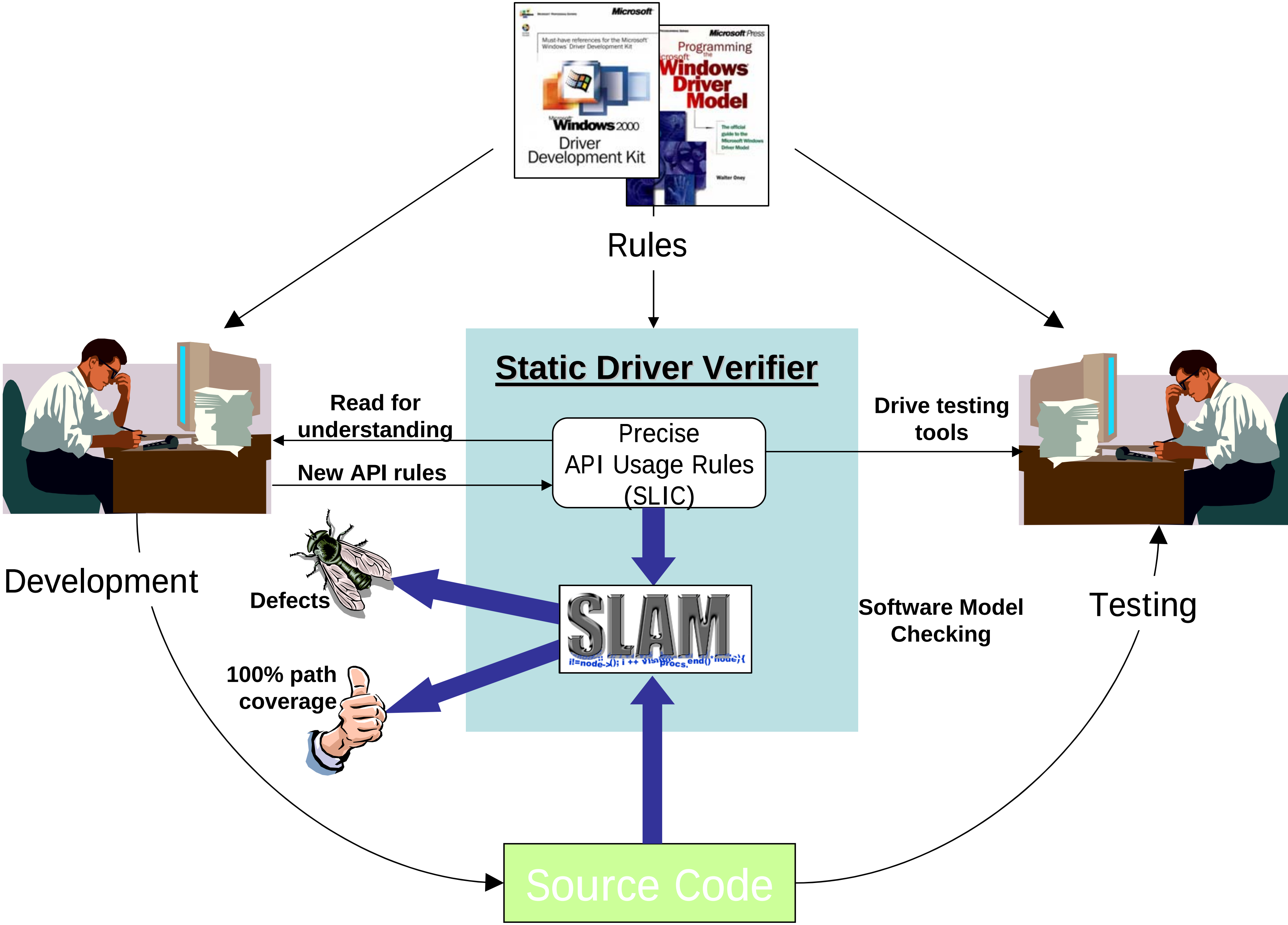



```
the "communication domain" in which communication is to take
place; see protocols(5).

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 con-
nection 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
```

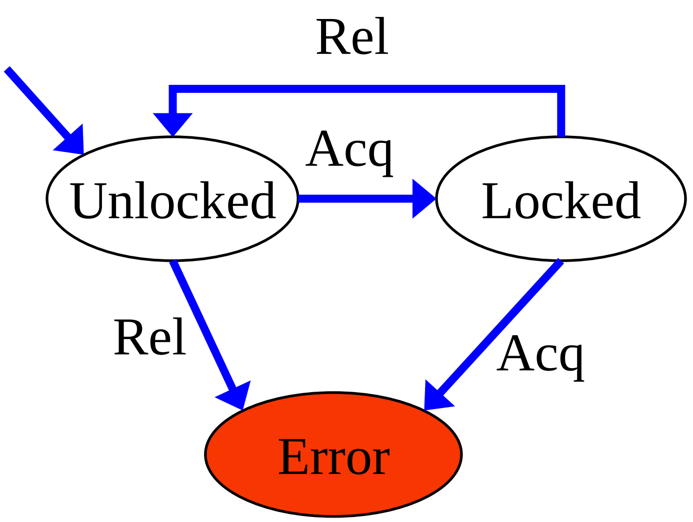
The Windows Driver Problem

- 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


Rules
Static Driver Verifier
Read for understanding
New API rules
Precise API Usage Rules (SLIC)
Drive testing tools
Development
Defects
SLAM
100% path coverage
Software Model Checking
Testing
Source Code

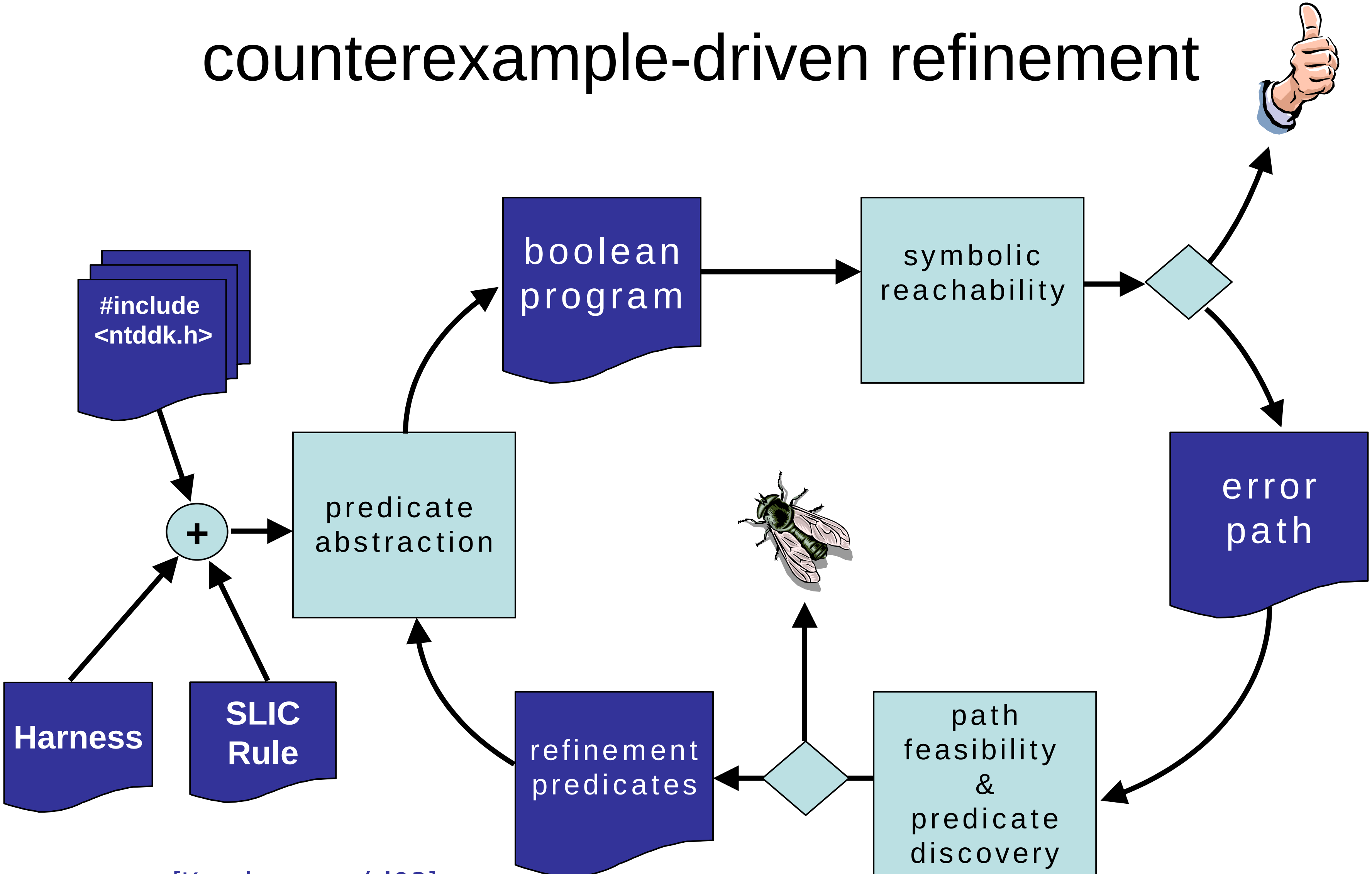
State Machine for Locking



Locking Rule in SLIC

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

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

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

The SLAM Process:
counterexample-driven refinement



[Kurshan *et al.* '93]
[Clarke *et al.* '00]
[Ball, Rajamani '00]

Example

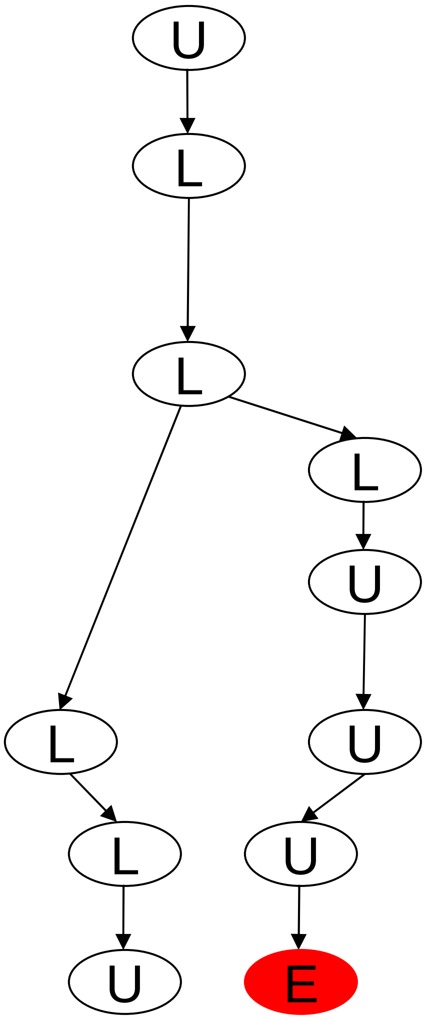
Does this code obey the locking rule?

```
do {
   KeAcquireSpinLock();

   nPacketsOld = nPackets;

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

KeReleaseSpinLock();
```

Example

Reachability in boolean program

model checker

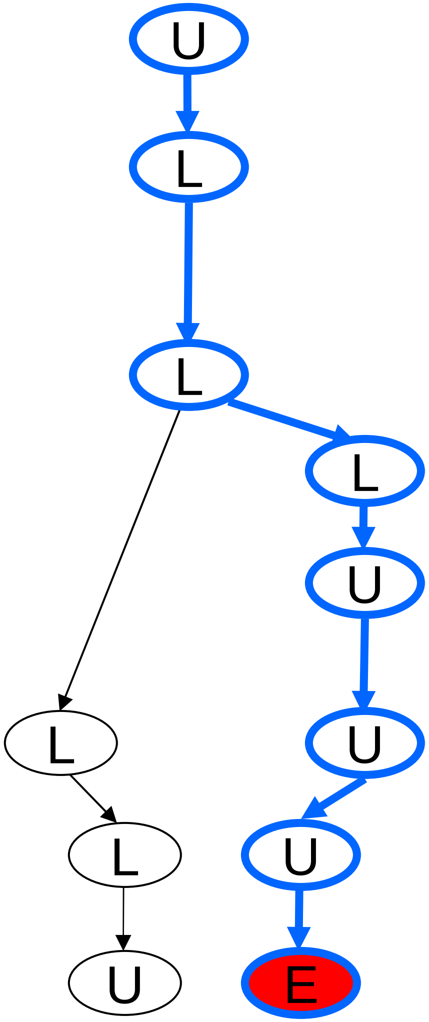



```
do {
   KeAcquireSpinLock();

   if(*){

      KeReleaseSpinLock();

   }
} while (*);

KeReleaseSpinLock();
```

Example

Is error path feasible in C program?

theorem prover



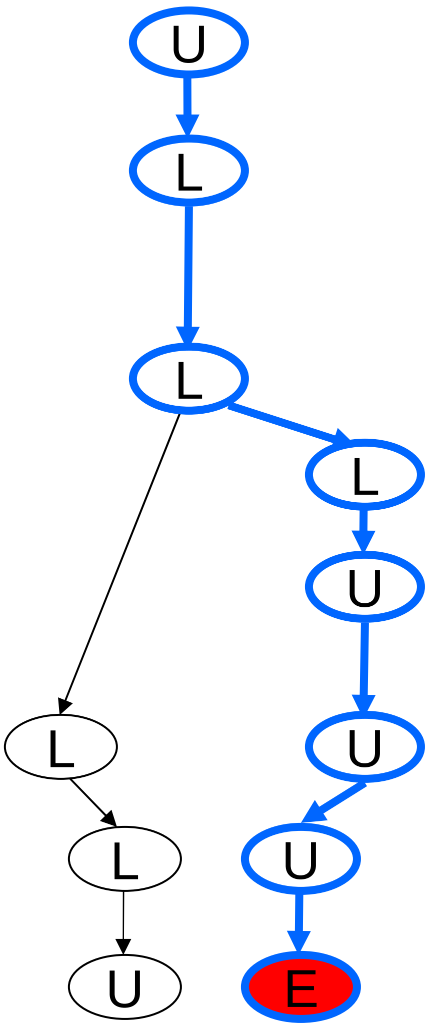

```
do {
   KeAcquireSpinLock();

   nPacketsOld = nPackets;

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

KeReleaseSpinLock();
```

Example

Add new predicate to boolean program

predicate abstraction theorem prover

b : (nPacketsOld == nPackets)

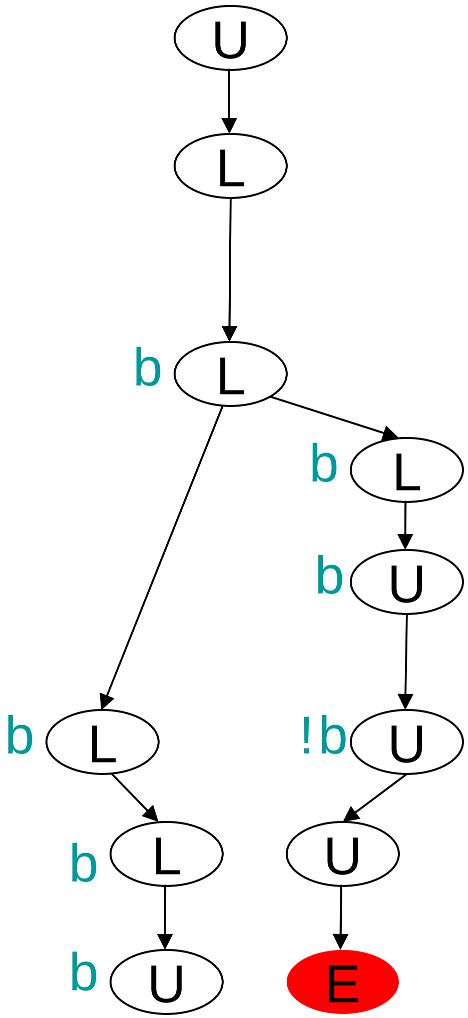



```
do {
   KeAcquireSpinLock();

   nPacketsOld = nPackets; b = true;

   if(request){
       request = request->Next;
       KeReleaseSpinLock();
       nPackets++; b = b ? false : *;
   }
} while (nPackets != nPacketsOld);  !b

KeReleaseSpinLock();
```

Example

b : (nPacketsOld == nPackets)

Model checking refined boolean program

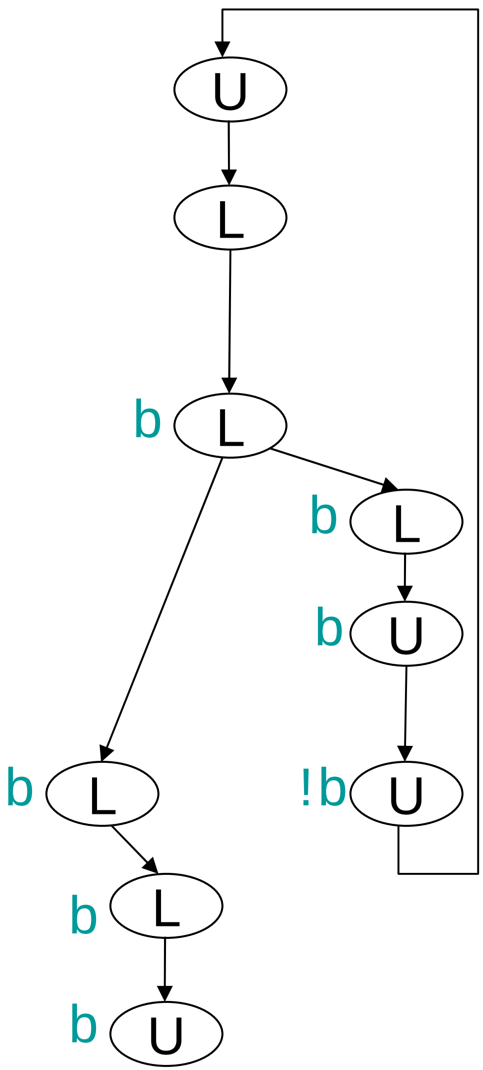



```
do {
   KeAcquireSpinLock();

   b = true;

   if(*){

      KeReleaseSpinLock();
      b = b ? false : *;
   }
} while ( !b );

KeReleaseSpinLock();
```

Example

b : (nPacketsOld == nPackets)

Model checking refined boolean program



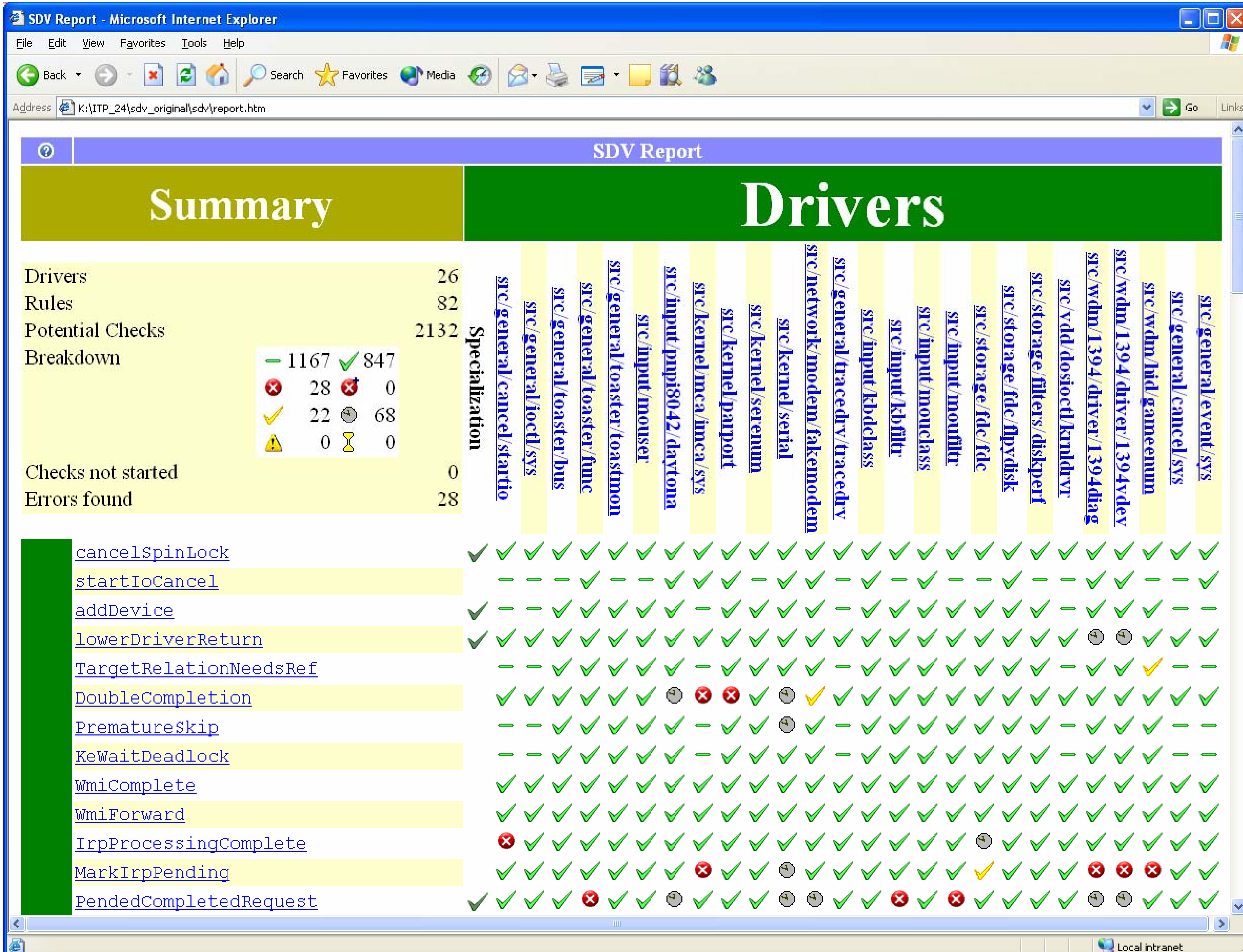

```
do {
   KeAcquireSpinLock();

   b = true;

   if(*){

      KeReleaseSpinLock();
      b = b ? false : *;
   }
} while ( !b );

KeReleaseSpinLock();
```

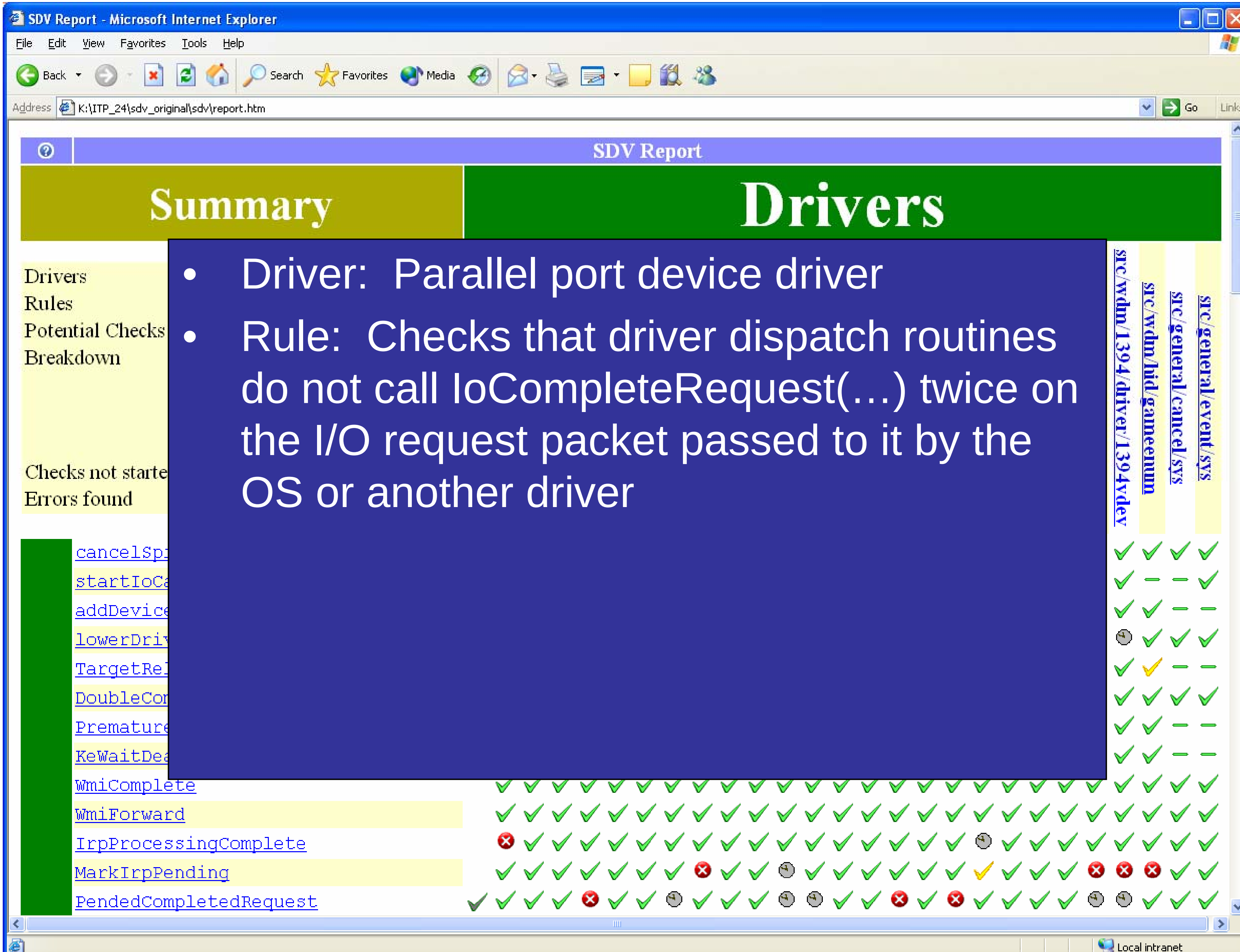
Observations about SLAM

- 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)

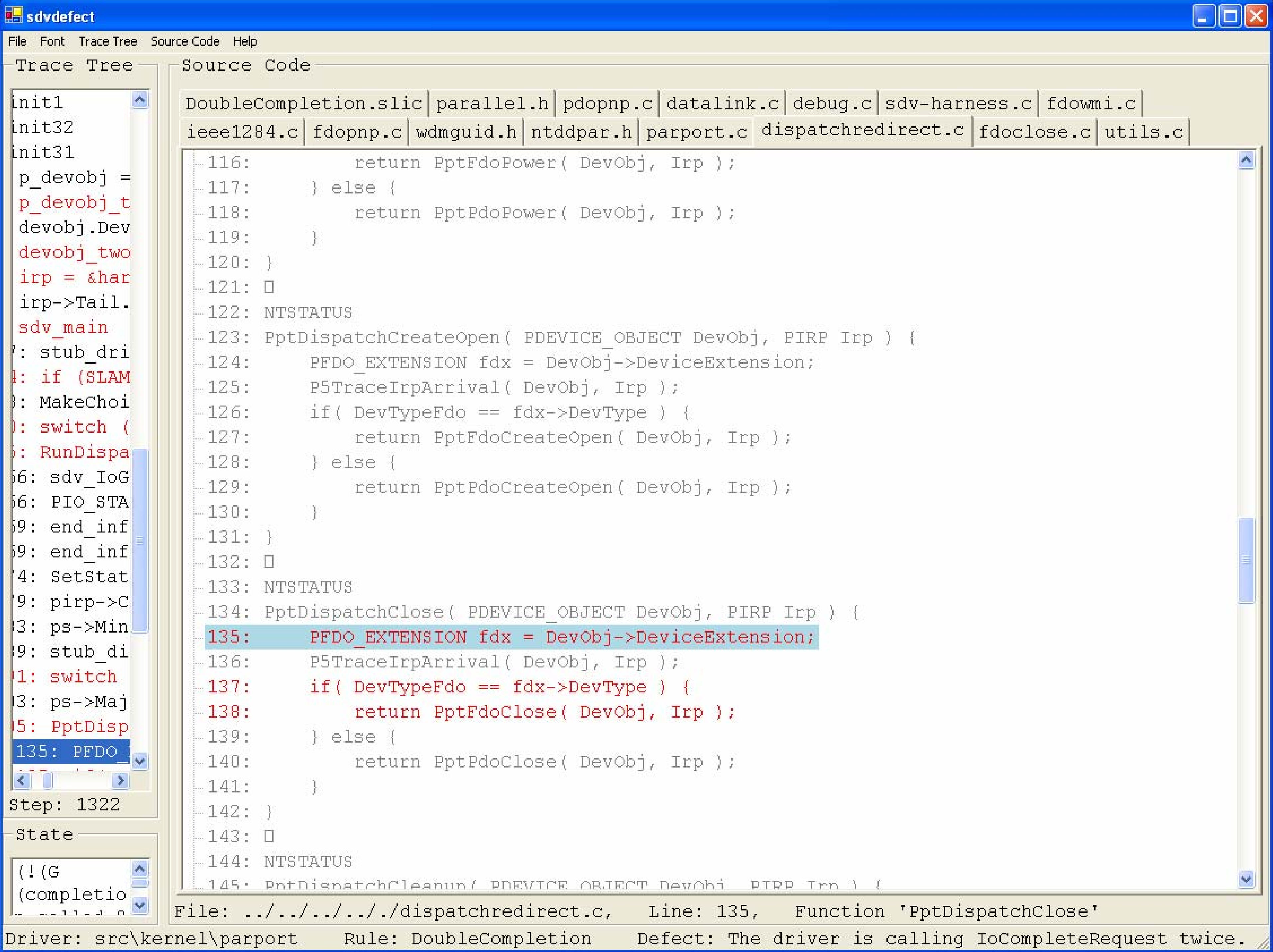
- A hybrid dynamic/static analysis that
  - "executes" a finite set of "concrete" paths symbolically
  - explores all paths through abstraction

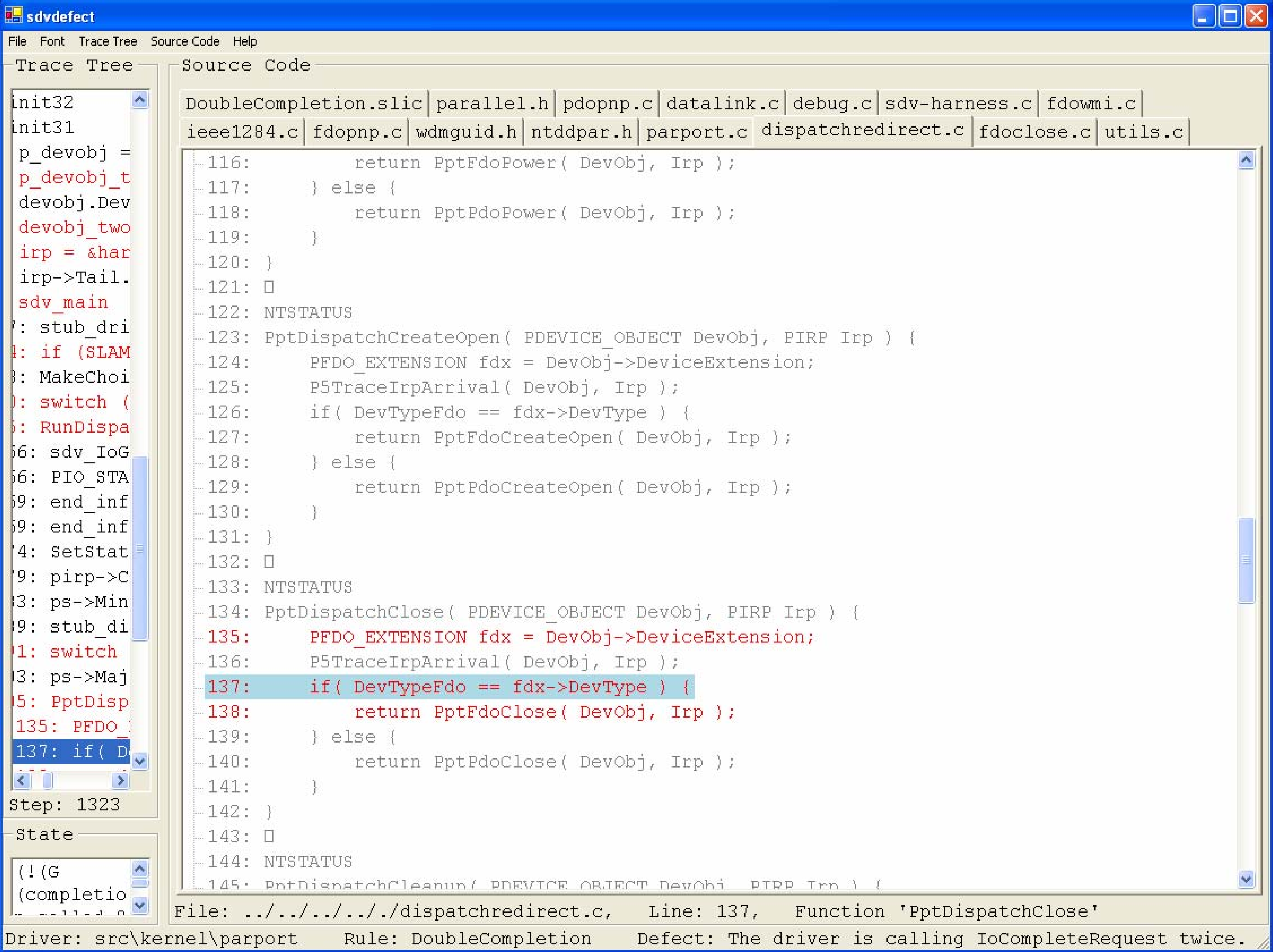
- A new form of program slicing
  - program code and data not relevant to property are dropped
  - non-determinism allows slices to have more behaviors

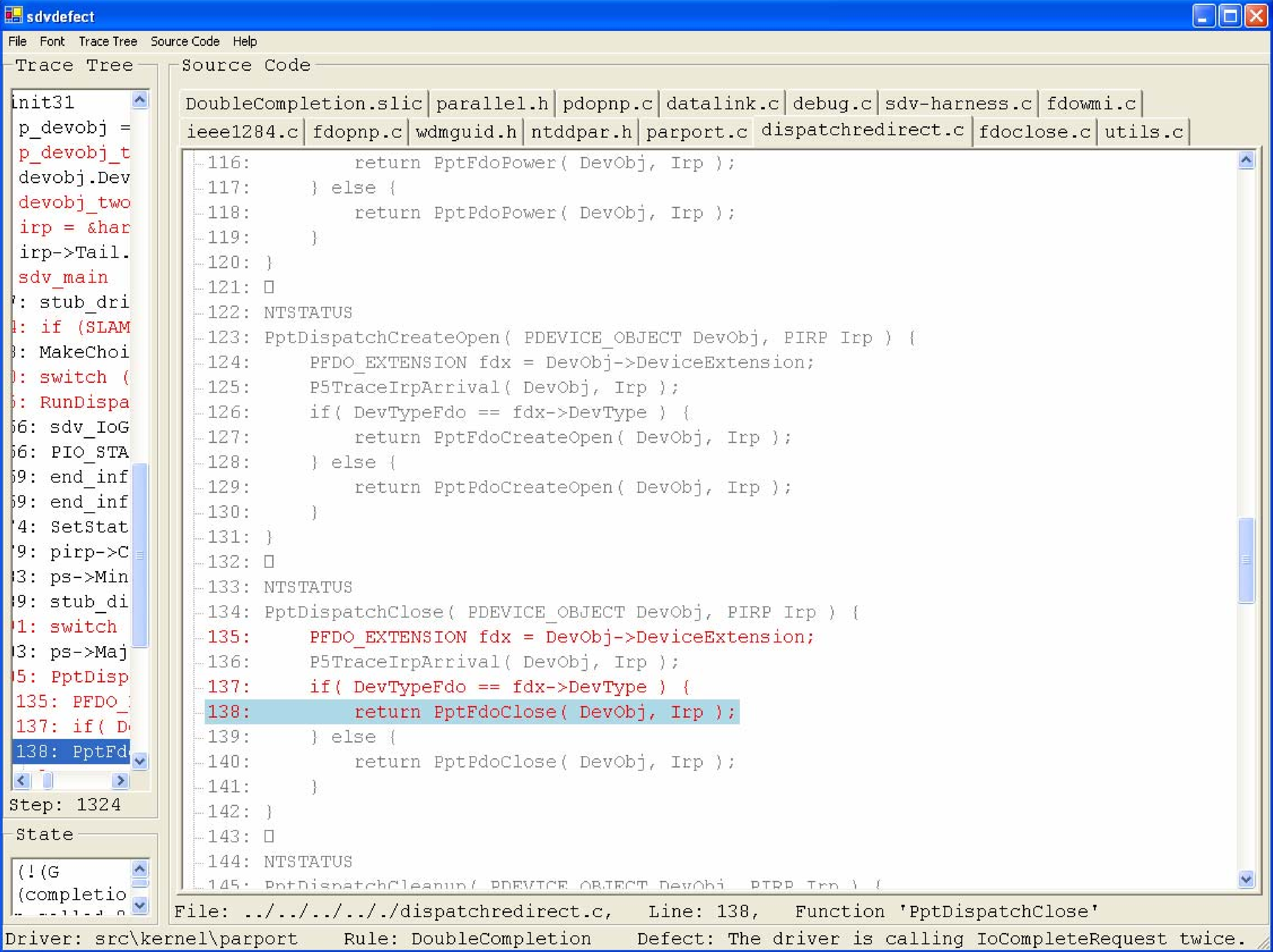

SDV Report - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
K:\ITP_24\sdv_original\sdv\report.htm
Go
Links
SDV Report
Summary
Drivers
Drivers 26
Rules 82
Potential Checks 2132
Breakdown
1167 847
28 0
22 68
0 0
Checks not started 0
Errors found 28
Specialization
src/general/cancel/startio
src/general/ioctl/sys
src/general/toaster/bus
src/general/toaster/func
src/general/toaster/toastmon
src/input/mouser
src/input/pnpi8042/daytona
src/kernel/mca/imca/sys
src/kernel/parport
src/kernel/serenum
src/kernel/serial
src/network/modem/fakemodem
src/general/tracedrv/tracedrv
src/input/kbdclass
src/input/kbfiltr
src/input/mouclass
src/input/moufiltr
src/storage/fdc/fdc
src/storage/fdc/flpydisk
src/storage/filters/diskperf
src/vdd/dosioctl/krnldrvr
src/wdm/1394/driver/1394diag
src/wdm/1394/driver/1394vdev
src/wdm/hid/gameenum
src/general/cancel/sys
src/general/event/sys
cancelSpinLock
startIoCancel
addDevice
lowerDriverReturn
TargetRelationNeedsRef
DoubleCompletion
PrematureSkip
KeWaitDeadlock
WmiComplete
WmiForward
IrpProcessingComplete
MarkIrpPending
PendedCompletedRequest
Local intranet

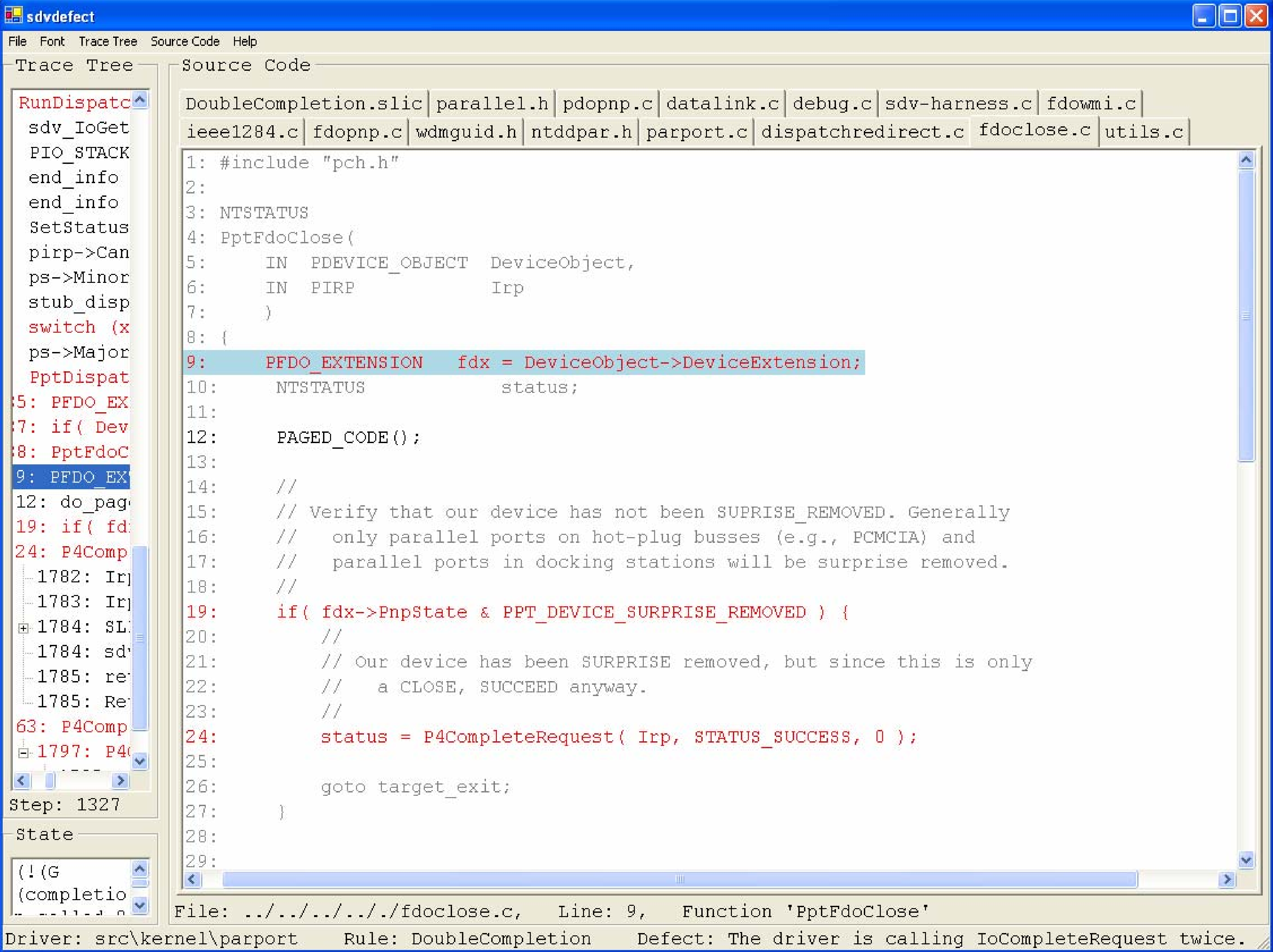


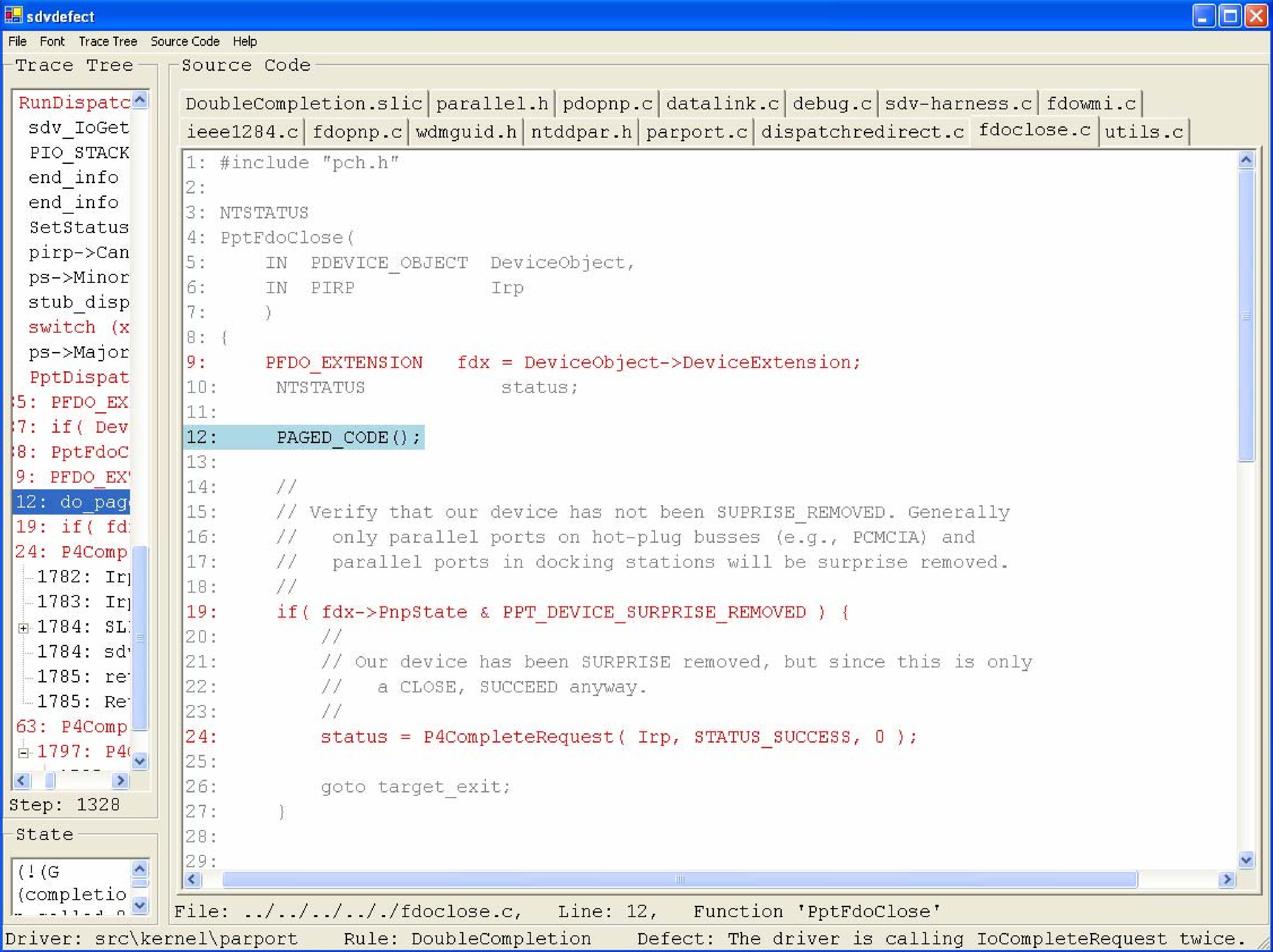
- Driver: Parallel port device driver
- Rule: Checks that driver dispatch routines do not call IoCompleteRequest(…) twice on the I/O request packet passed to it by the OS or another driver

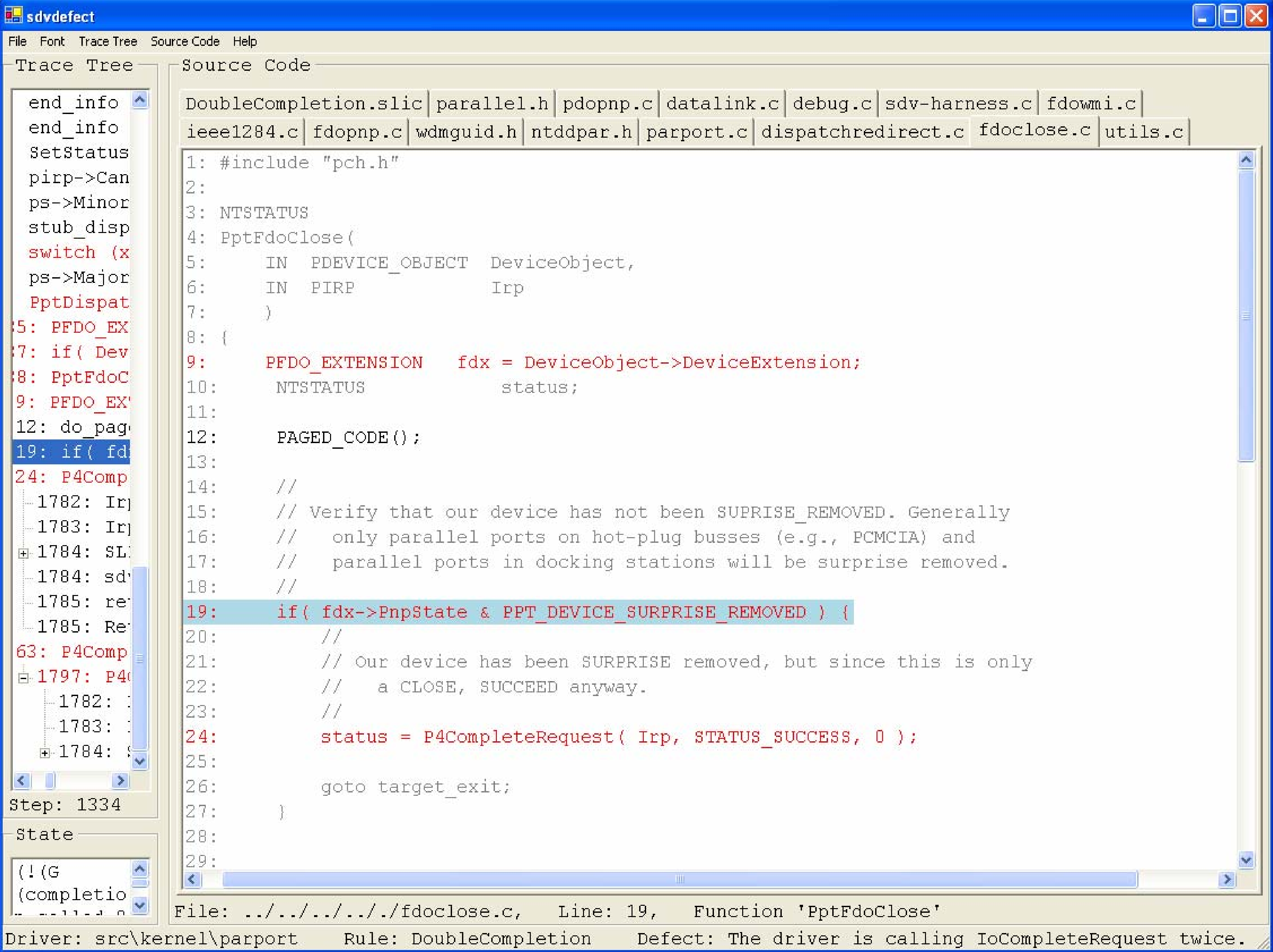

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
init1
init32
init31
p_devobj =
p_devobj_t
devobj.Dev
devobj_two
irp = &har
irp->Tail.
sdv_main
: stub_dri
: if (SLAM
: MakeChoi
: switch (
: RunDispa
56: sdv_IoG
56: PIO_STA
59: end_inf
59: end_inf
4: SetStat
9: pirp->C
3: ps->Min
9: stub_di
1: switch
3: ps->Maj
5: PptDisp
135: PFDO_
Step: 1322
State
(!(G
(completio
Source Code
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
116: return PptFdoPower(DevObj, Irp);
117: } else {
118: return PptPdoPower(DevObj, Irp);
119: }
120: }
121: □
122: NTSTATUS
123: PptDispatchCreateOpen(PDEVICE_OBJECT DevObj, PIRP Irp) {
124: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
125: P5TraceIrpArrival(DevObj, Irp);
126: if(DevTypeFdo == fdx->DevType) {
127: return PptFdoCreateOpen(DevObj, Irp);
128: } else {
129: return PptPdoCreateOpen(DevObj, Irp);
130: }
131: }
132: □
133: NTSTATUS
134: PptDispatchClose(PDEVICE_OBJECT DevObj, PIRP Irp) {
135: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
136: P5TraceIrpArrival(DevObj, Irp);
137: if(DevTypeFdo == fdx->DevType) {
138: return PptFdoClose(DevObj, Irp);
139: } else {
140: return PptPdoClose(DevObj, Irp);
141: }
142: }
143: □
144: NTSTATUS
File: ../../../../dispatchredirect.c, Line: 135, Function 'PptDispatchClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

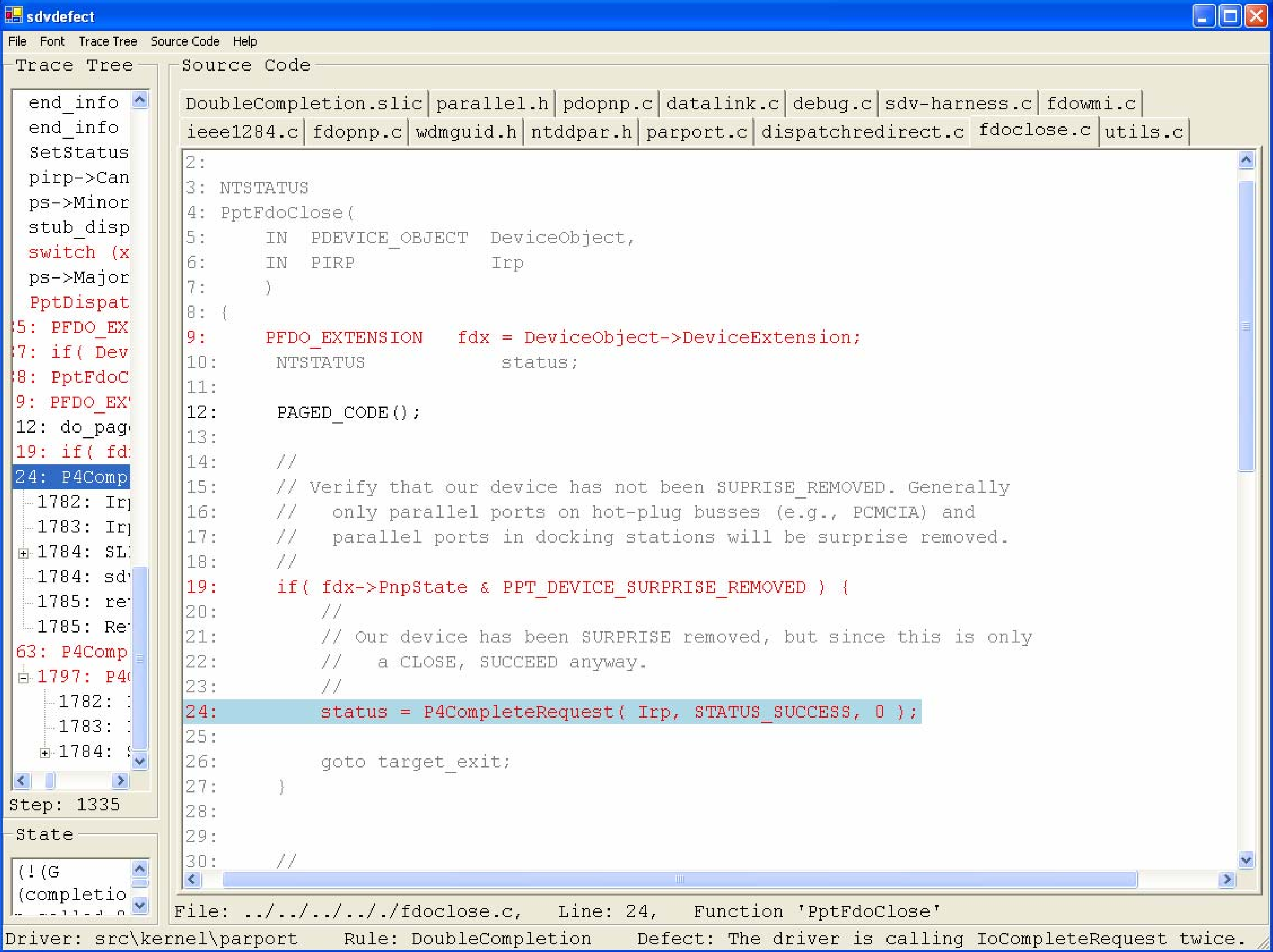

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c
ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c
init32
init31
p_devobj =
p_devobj_t
devobj.Dev
devobj_two
irp = &har
irp->Tail.
sdv_main
stub_dri
if (SLAM
MakeChoi
switch (
RunDispa
sdv_IoG
PIO_STA
end_inf
end_inf
SetStat
pirp->C
ps->Min
stub_di
switch
ps->Maj
PptDisp
135: PFDO_
137: if(D
Step: 1323
State
(!(G
(completio
116: return PptFdoPower(DevObj, Irp);
117: } else {
118: return PptPdoPower(DevObj, Irp);
119: }
120: }
121: □
122: NTSTATUS
123: PptDispatchCreateOpen(PDEVICE_OBJECT DevObj, PIRP Irp) {
124: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
125: P5TraceIrpArrival(DevObj, Irp);
126: if(DevTypeFdo == fdx->DevType) {
127: return PptFdoCreateOpen(DevObj, Irp);
128: } else {
129: return PptPdoCreateOpen(DevObj, Irp);
130: }
131: }
132: □
133: NTSTATUS
134: PptDispatchClose(PDEVICE_OBJECT DevObj, PIRP Irp) {
135: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
136: P5TraceIrpArrival(DevObj, Irp);
137: if(DevTypeFdo == fdx->DevType) {
138: return PptFdoClose(DevObj, Irp);
139: } else {
140: return PptPdoClose(DevObj, Irp);
141: }
142: }
143: □
144: NTSTATUS
File: ../../../../dispatchredirect.c, Line: 137, Function 'PptDispatchClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

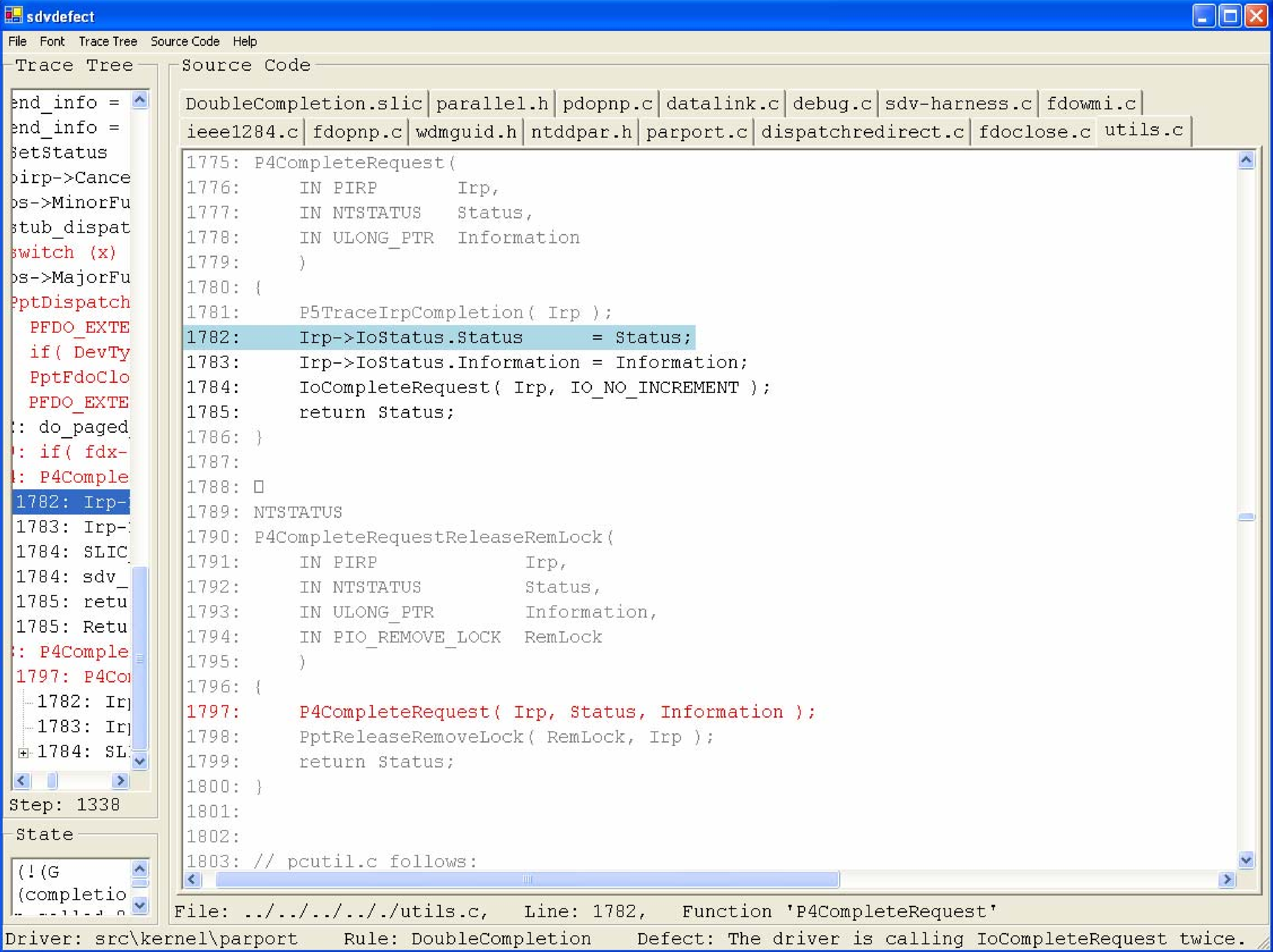

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
DoubleCompletion.slic
parallel.h
pdopnp.c
datalink.c
debug.c
sdv-harness.c
fdowmi.c
ieee1284.c
fdopnp.c
wdmguid.h
ntddpar.h
parport.c
dispatchredirect.c
fdoclose.c
utils.c
init31
p_devobj =
p_devobj_t
devobj.Dev
devobj_two
irp = &har
irp->Tail.
sdv_main
stub_dri
if (SLAM
MakeChoi
switch (
RunDispa
sdv_IoG
PIO_STA
end_inf
end_inf
SetStat
pirp->C
ps->Min
stub_di
switch
ps->Maj
PptDisp
135: PFDO_
137: if(D
138: PptFd
116: return PptFdoPower(DevObj, Irp);
117: } else {
118: return PptPdoPower(DevObj, Irp);
119: }
120: }
122: NTSTATUS
123: PptDispatchCreateOpen(PDEVICE_OBJECT DevObj, PIRP Irp) {
124: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
125: P5TraceIrpArrival(DevObj, Irp);
126: if(DevTypeFdo == fdx->DevType) {
127: return PptFdoCreateOpen(DevObj, Irp);
128: } else {
129: return PptPdoCreateOpen(DevObj, Irp);
130: }
131: }
133: NTSTATUS
134: PptDispatchClose(PDEVICE_OBJECT DevObj, PIRP Irp) {
135: PFDO_EXTENSION fdx = DevObj->DeviceExtension;
136: P5TraceIrpArrival(DevObj, Irp);
137: if(DevTypeFdo == fdx->DevType) {
138: return PptFdoClose(DevObj, Irp);
139: } else {
140: return PptPdoClose(DevObj, Irp);
141: }
142: }
144: NTSTATUS
Step: 1324
State
(!(G
(completio
File: ../../../../dispatchredirect.c, Line: 138, Function 'PptDispatchClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

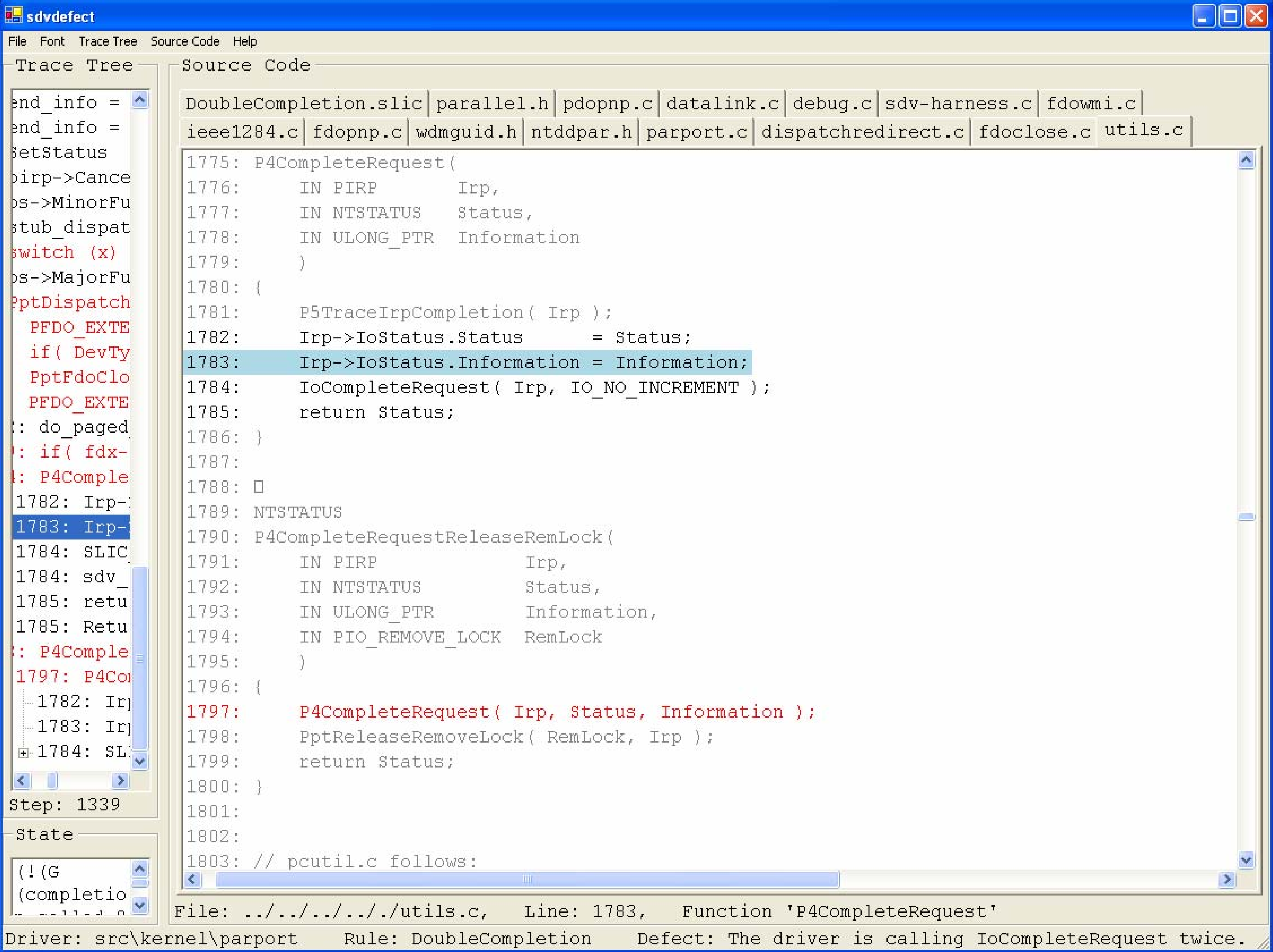

sdvdefect
File
Font
Trace Tree
Source Code
Help
Trace Tree
Source Code
DoubleCompletion.slic
parallel.h
pdopnp.c
datalink.c
debug.c
sdv-harness.c
fdowmi.c
ieee1284.c
fdopnp.c
wdmguid.h
ntddpar.h
parport.c
dispatchredirect.c
fdoclose.c
utils.c
RunDispatc
sdv_IoGet
PIO_STACK
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
1782: Ir
1783: Ir
1784: SL
1784: sd
1785: re
1785: Re
63: P4Comp
1797: P4
Step: 1327
State
(!(G
(completio
1: #include "pch.h"
2:
3: NTSTATUS
4: PptFdoClose(
5: IN PDEVICE_OBJECT DeviceObject,
6: IN PIRP Irp
7:)
8: {
9: PFDO_EXTENSION fdx = DeviceObject->DeviceExtension;
10: NTSTATUS status;
11:
12: PAGED_CODE();
13:
14: //
15: // Verify that our device has not been SUPRISE_REMOVED. Generally
16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and
17: // parallel ports in docking stations will be surprise removed.
18: //
19: if(fdx->PnpState & PPT_DEVICE_SURPRISE_REMOVED) {
20: //
21: // Our device has been SURPRISE removed, but since this is only
22: // a CLOSE, SUCCEED anyway.
23: //
24: status = P4CompleteRequest(Irp, STATUS_SUCCESS, 0);
25:
26: goto target_exit;
27: }
28:
29:
File: ../../../.././fdoclose.c, Line: 9, Function 'PptFdoClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

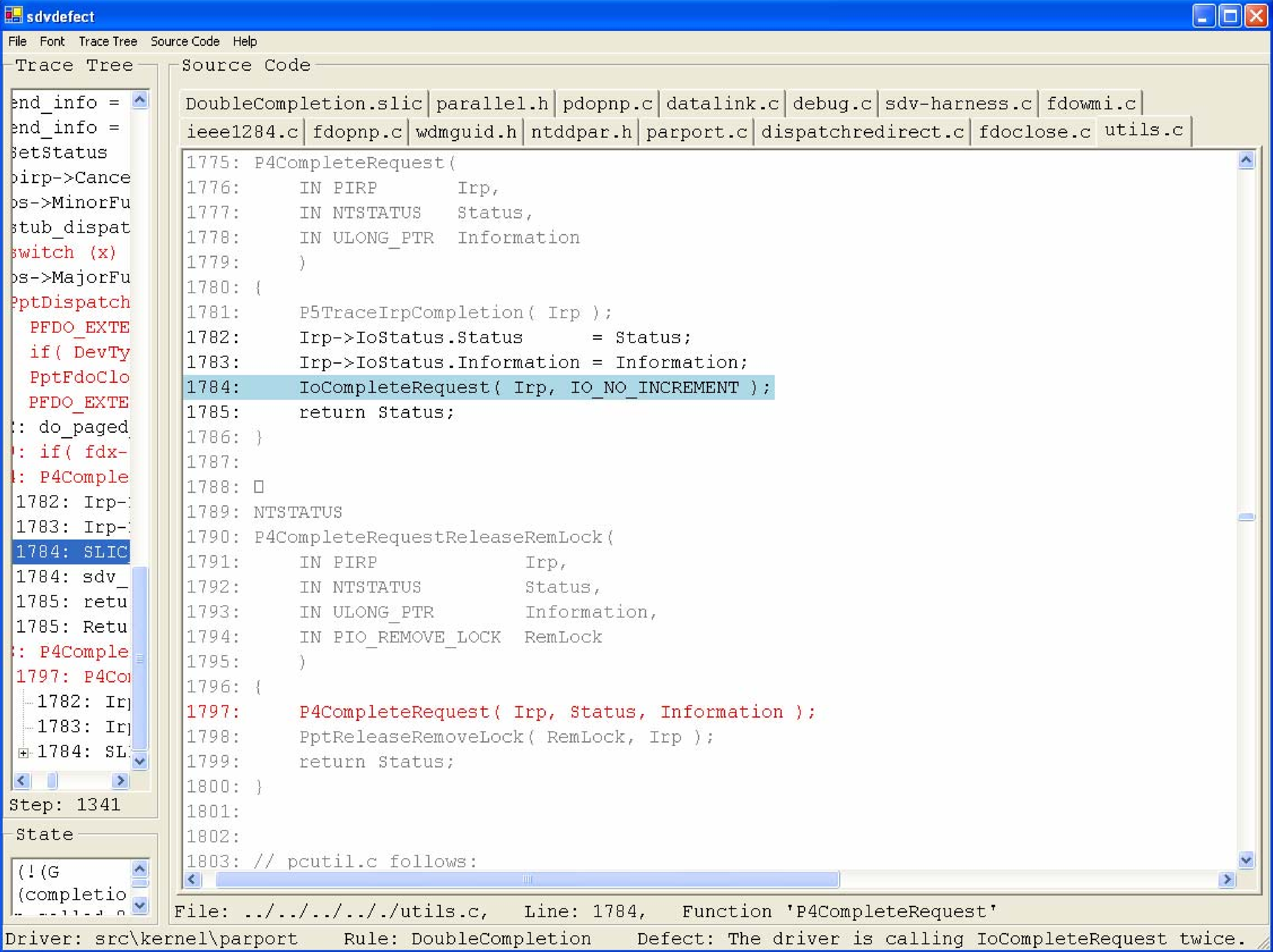

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
RunDispatc
sdv_IoGet
PIO_STACK
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
1782: Ir
1783: Ir
1784: SL
1784: sd
1785: re
1785: Re
63: P4Comp
1797: P4
Step: 1328
State
(!(G
(completio
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
1: #include "pch.h"
2:
3: NTSTATUS
4: PptFdoClose(
5: IN PDEVICE_OBJECT DeviceObject,
6: IN PIRP Irp
7:)
8: {
9: PFDO_EXTENSION fdx = DeviceObject->DeviceExtension;
10: NTSTATUS status;
11:
12: PAGED_CODE();
13:
14: //
15: // Verify that our device has not been SUPRISE_REMOVED. Generally
16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and
17: // parallel ports in docking stations will be surprise removed.
18: //
19: if(fdx->PnpState & PPT_DEVICE_SURPRISE_REMOVED) {
20: //
21: // Our device has been SURPRISE removed, but since this is only
22: // a CLOSE, SUCCEED anyway.
23: //
24: status = P4CompleteRequest(Irp, STATUS_SUCCESS, 0);
25:
26: goto target_exit;
27: }
28:
29:
File: ../../../.././fdoclose.c, Line: 12, Function 'PptFdoClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

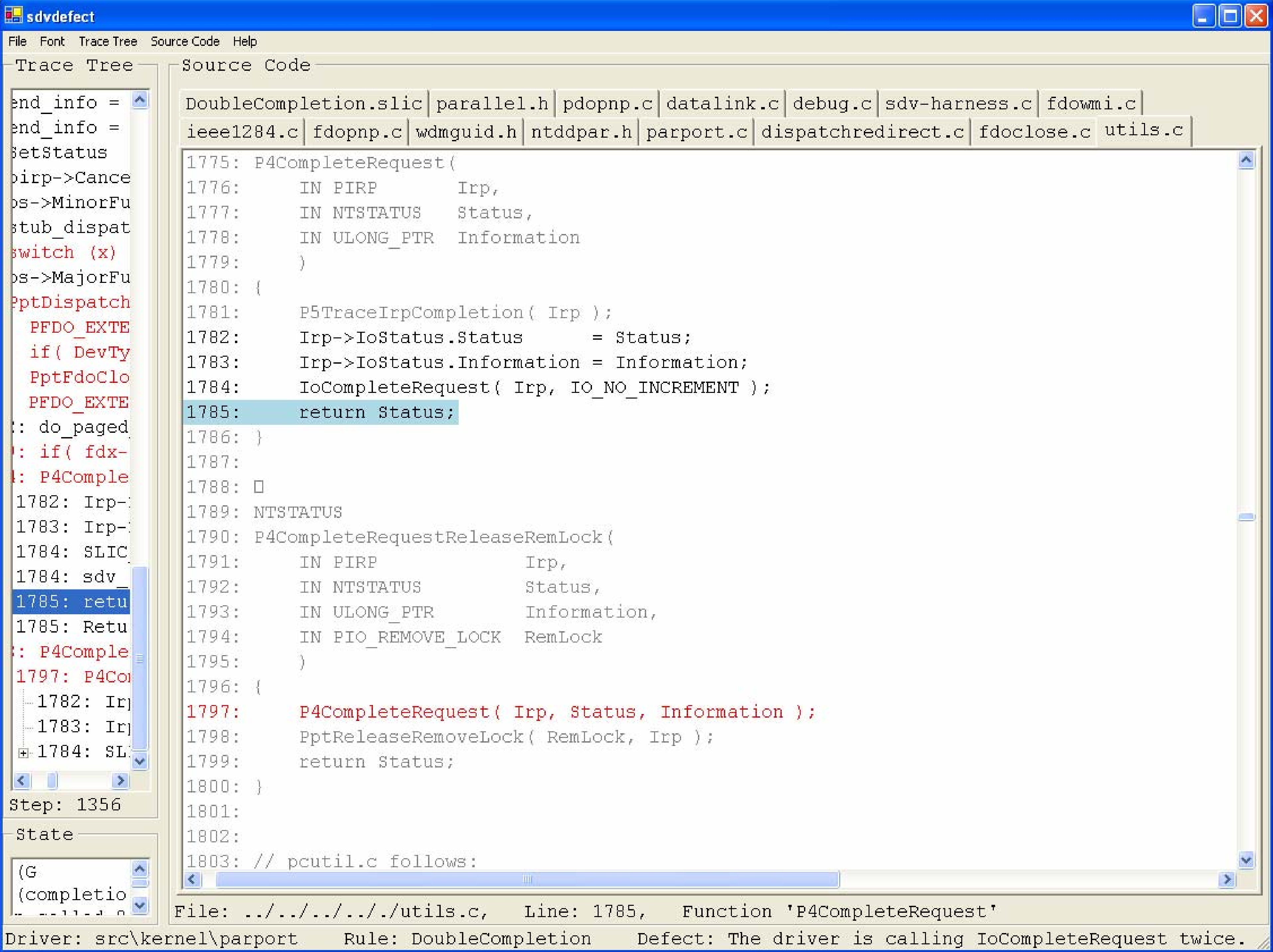

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
1782: Ir
1783: Ir
1784: SL
1784: sd
1785: re
1785: Re
63: P4Comp
1797: P4
1782:
1783:
1784:
Step: 1334
State
(!(G
(completio
Source Code
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
1: #include "pch.h"
2:
3: NTSTATUS
4: PptFdoClose(
5: IN PDEVICE_OBJECT DeviceObject,
6: IN PIRP Irp
7:)
8: {
9: PFDO_EXTENSION fdx = DeviceObject->DeviceExtension;
10: NTSTATUS status;
11:
12: PAGED_CODE();
13:
14: //
15: // Verify that our device has not been SUPRISE_REMOVED. Generally
16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and
17: // parallel ports in docking stations will be surprise removed.
18: //
19: if(fdx->PnpState & PPT_DEVICE_SURPRISE_REMOVED) {
20: //
21: // Our device has been SURPRISE removed, but since this is only
22: // a CLOSE, SUCCEED anyway.
23: //
24: status = P4CompleteRequest(Irp, STATUS_SUCCESS, 0);
25:
26: goto target_exit;
27: }
28:
29:
File: ../../../.././fdoclose.c, Line: 19, Function 'PptFdoClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

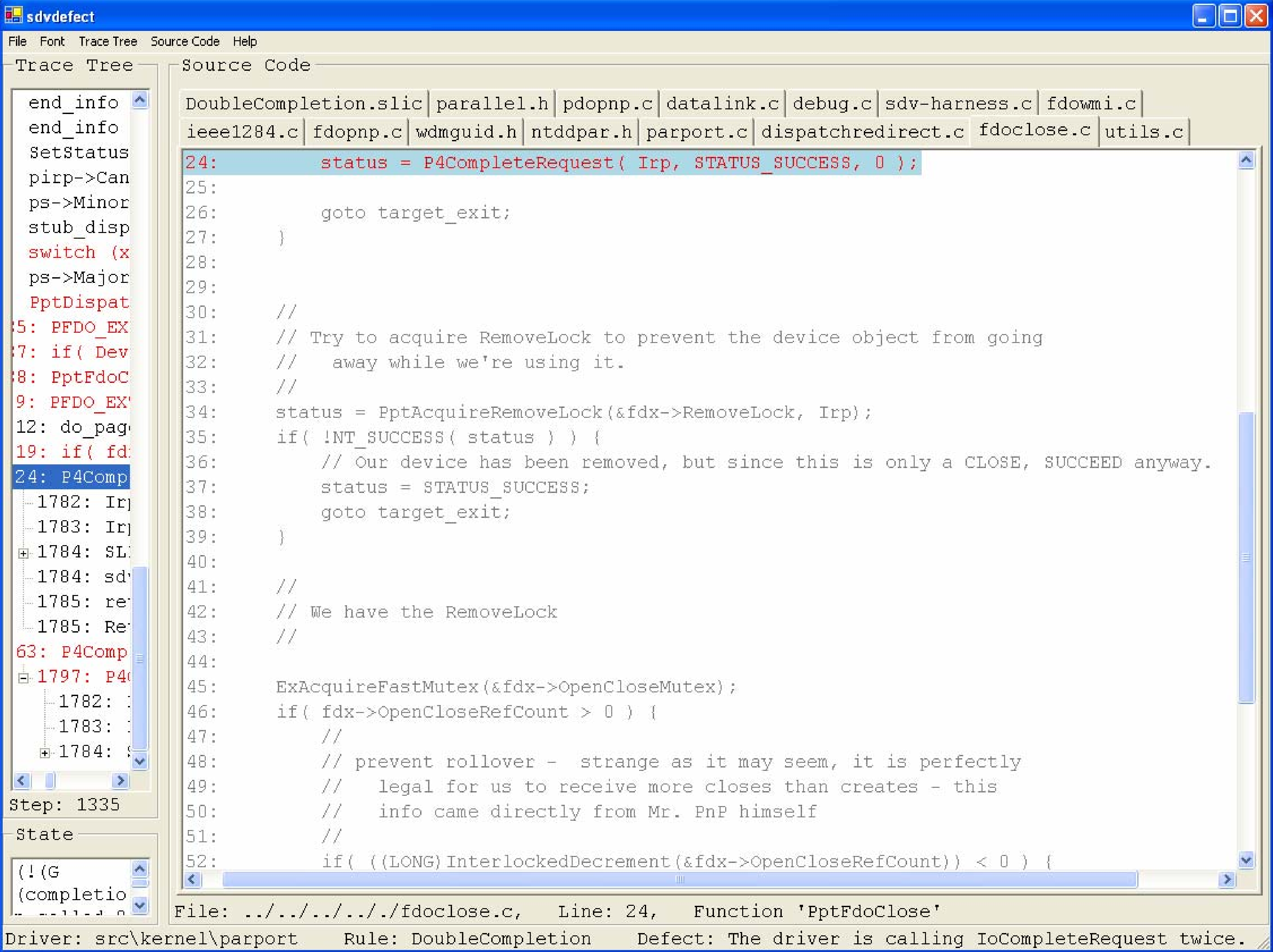

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
1782: Ir
1783: Ir
1784: SL
1784: sd
1785: re
1785: Re
63: P4Comp
1797: P4
1782:
1783:
1784:
Step: 1335
State
(!(G
(completio
Source Code
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
2:
3: NTSTATUS
4: PptFdoClose(
5: IN PDEVICE_OBJECT DeviceObject,
6: IN PIRP Irp
7:)
8: {
9: PFDO_EXTENSION fdx = DeviceObject->DeviceExtension;
10: NTSTATUS status;
11:
12: PAGED_CODE();
13:
14: //
15: // Verify that our device has not been SUPRISE_REMOVED. Generally
16: // only parallel ports on hot-plug busses (e.g., PCMCIA) and
17: // parallel ports in docking stations will be surprise removed.
18: //
19: if(fdx->PnpState & PPT_DEVICE_SURPRISE_REMOVED) {
20: //
21: // Our device has been SURPRISE removed, but since this is only
22: // a CLOSE, SUCCEED anyway.
23: //
24: status = P4CompleteRequest(Irp, STATUS_SUCCESS, 0);
25:
26: goto target_exit;
27: }
28:
29:
30: //
File: ../../../.././fdoclose.c, Line: 24, Function 'PptFdoClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

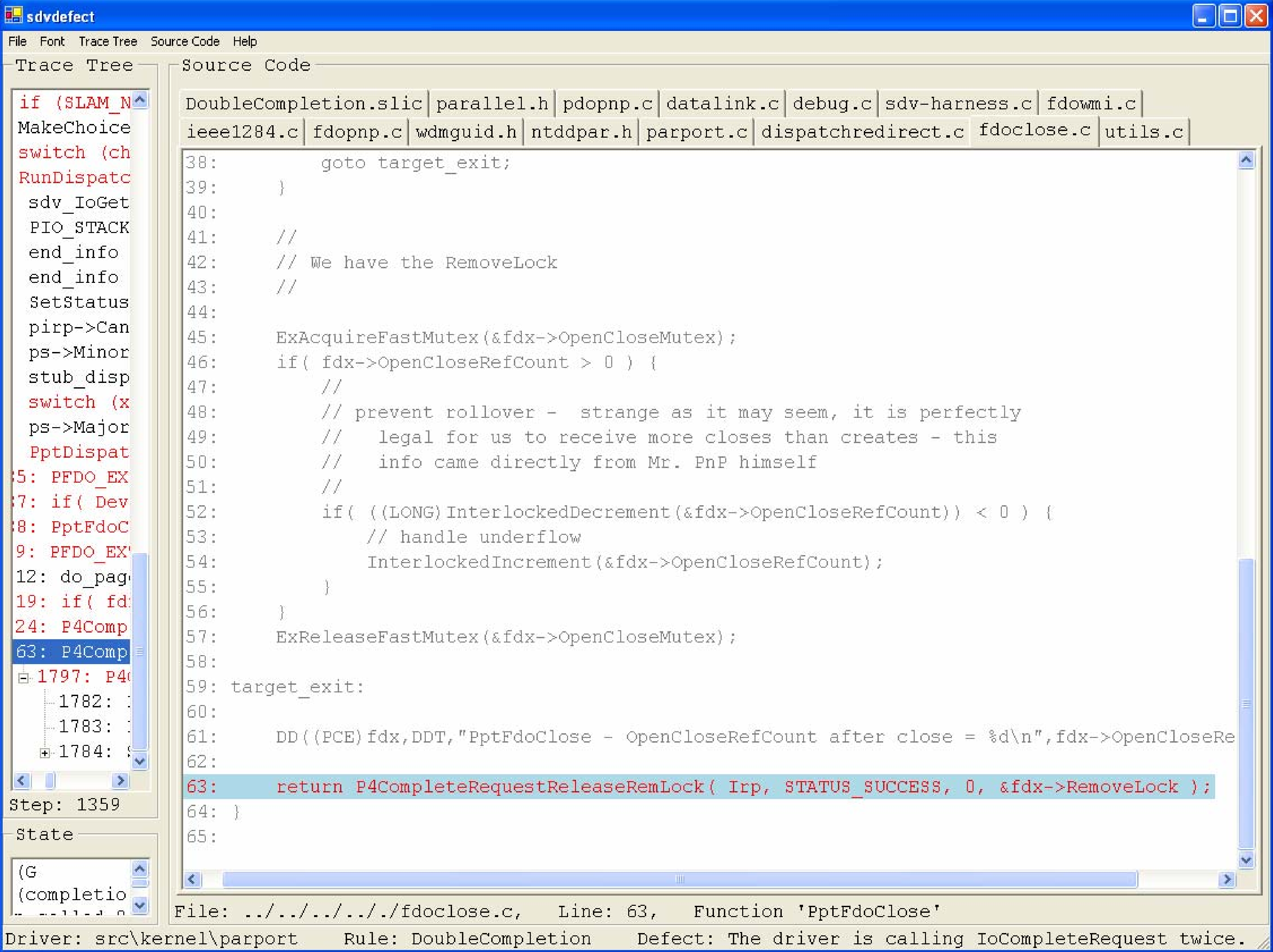

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c
ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1338
State
File: ../../../.././utils.c, Line: 1782, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

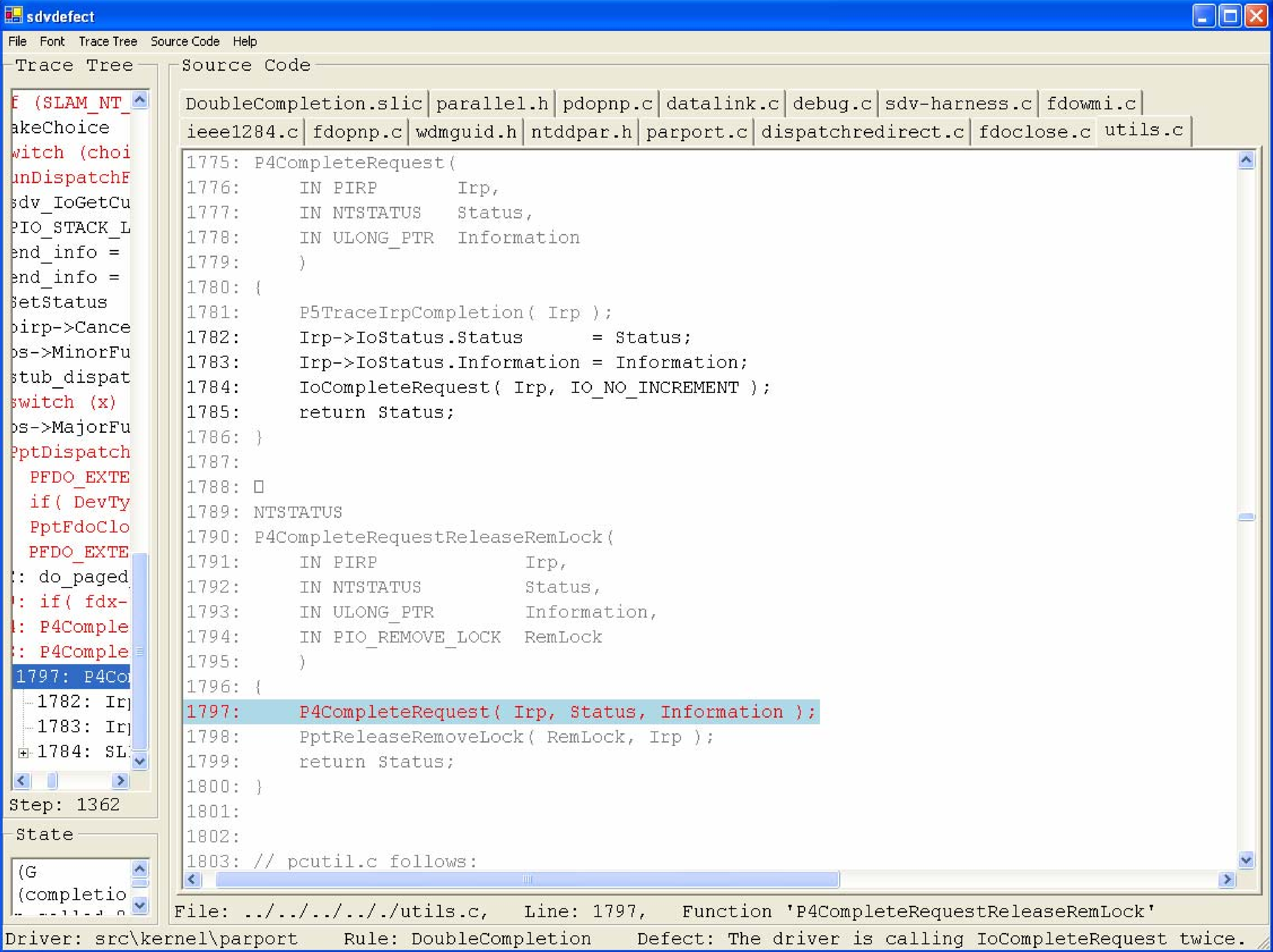

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c
ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1339
State
File: ../../../.././utils.c, Line: 1783, Function 'P4CompleteRequest'
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
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
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1341
State
File: ../../../.././utils.c, Line: 1784, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

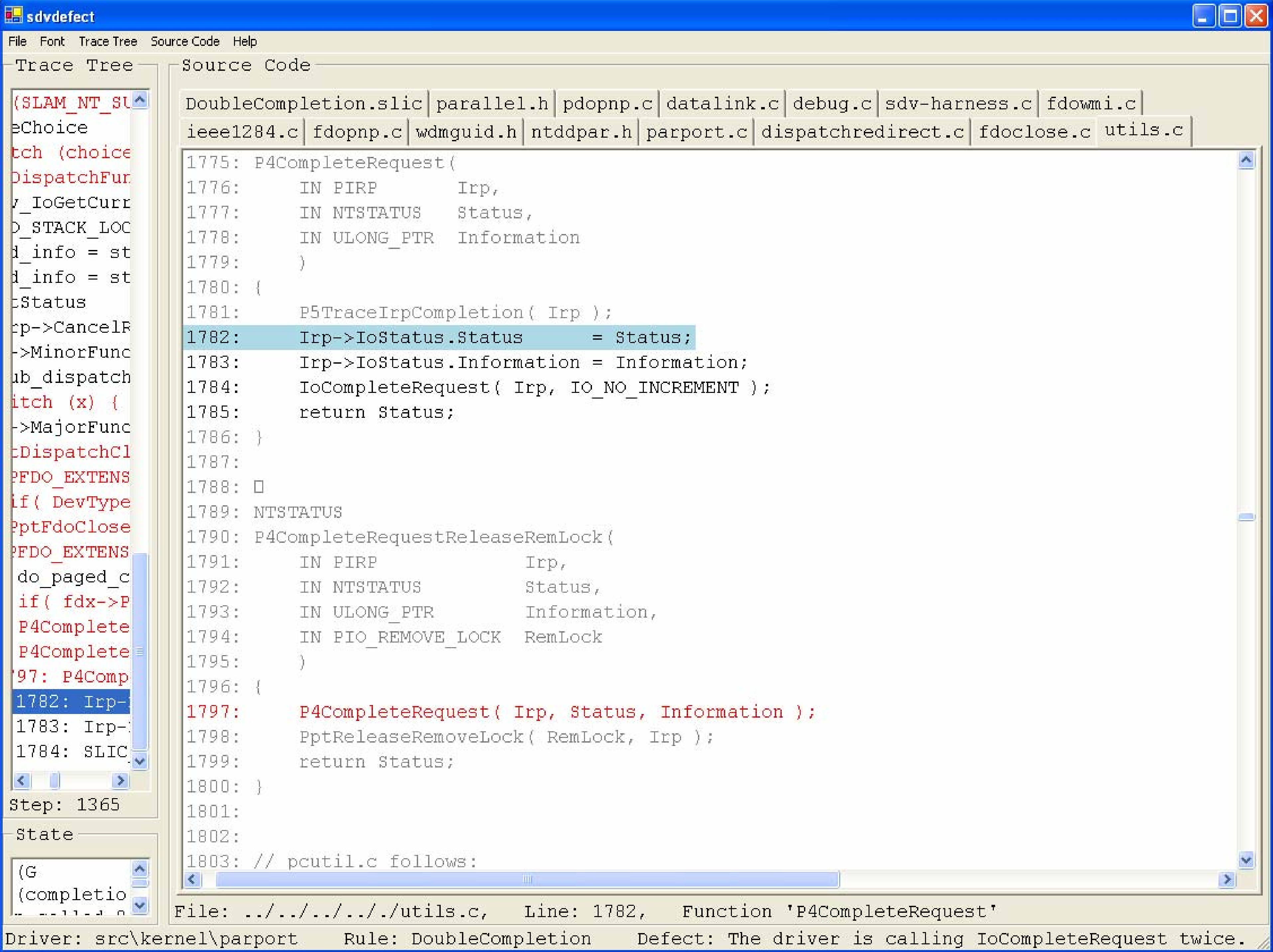

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
DoubleCompletion.slic parallel.h pdopnp.c datalink.c debug.c sdv-harness.c fdowmi.c
ieee1284.c fdopnp.c wdmguid.h ntddpar.h parport.c dispatchredirect.c fdoclose.c utils.c
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1356
State
File: ../../../.././utils.c, Line: 1785, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

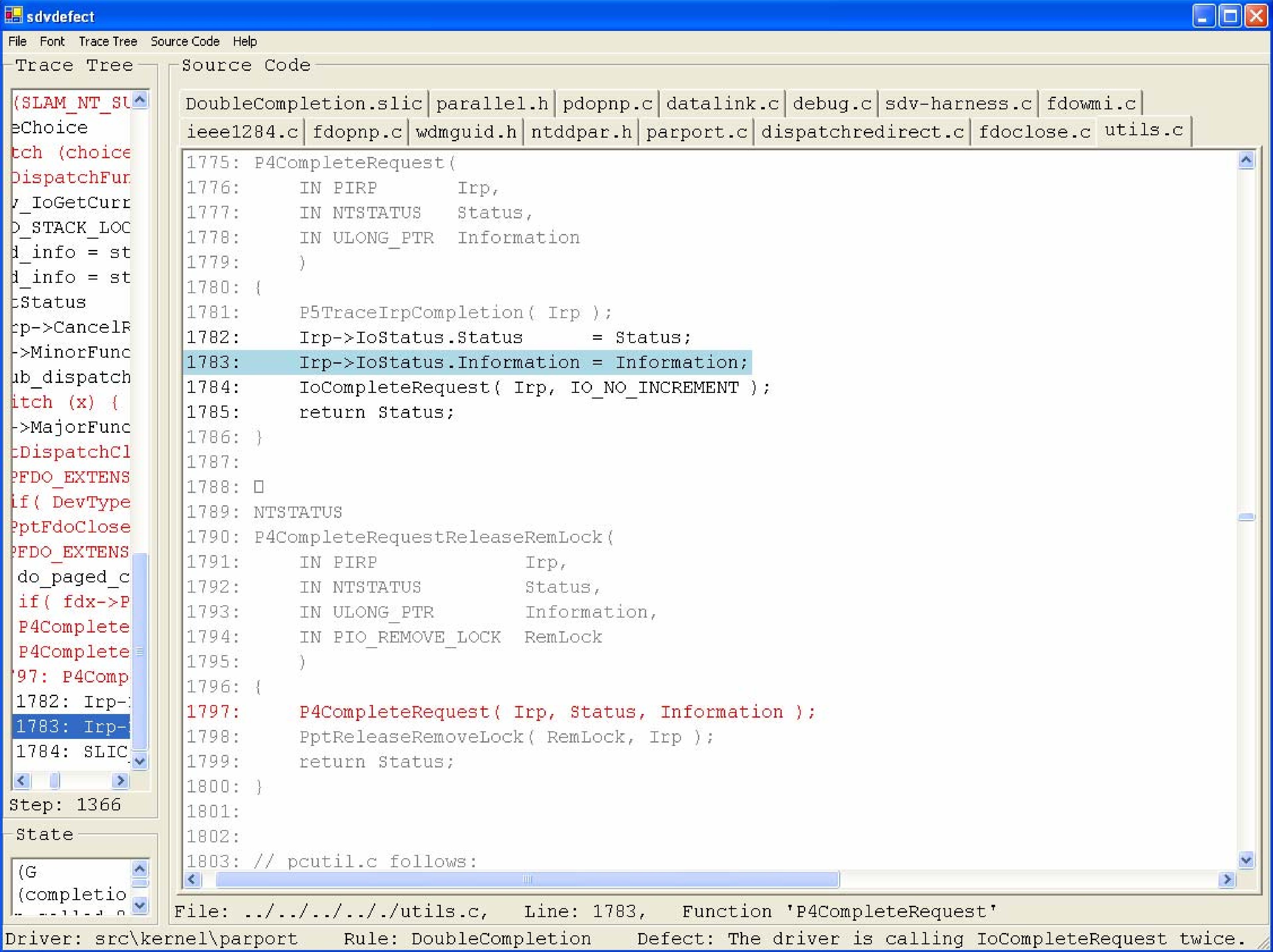

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
Source Code
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
1782: Ir
1783: Ir
1784: SL
1784: sd
1785: re
1785: Re
63: P4Comp
1797: P4
1782:
1783:
1784:
Step: 1335
State
(!(G
(completio
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
24: status = P4CompleteRequest(Irp, STATUS_SUCCESS, 0);
25:
26: goto target_exit;
27: }
28:
29:
30: //
31: // Try to acquire RemoveLock to prevent the device object from going
32: // away while we're using it.
33: //
34: status = PptAcquireRemoveLock(&fdx->RemoveLock, Irp);
35: if(!NT_SUCCESS(status)) {
36: // Our device has been removed, but since this is only a CLOSE, SUCCEED anyway.
37: status = STATUS_SUCCESS;
38: goto target_exit;
39: }
40:
41: //
42: // We have the RemoveLock
43: //
44:
45: ExAcquireFastMutex(&fdx->OpenCloseMutex);
46: if(fdx->OpenCloseRefCount > 0) {
47: //
48: // prevent rollover - strange as it may seem, it is perfectly
49: // legal for us to receive more closes than creates - this
50: // info came directly from Mr. PnP himself
51: //
52: if(((LONG)InterlockedDecrement(&fdx->OpenCloseRefCount)) < 0) {
File: ../../../../fdoclose.c, Line: 24, Function 'PptFdoClose'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

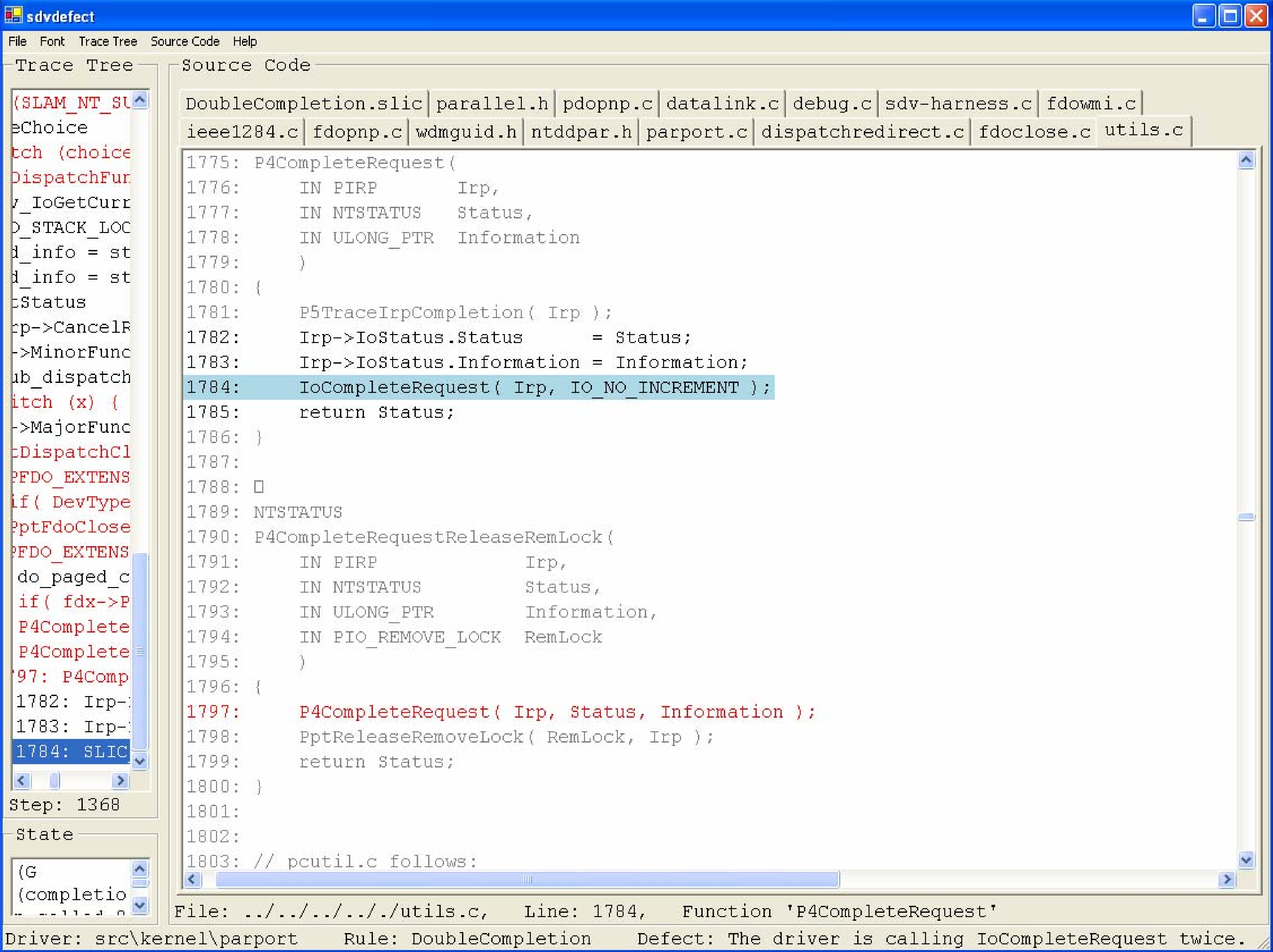

sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
if (SLAM_N
MakeChoice
switch (ch
RunDispatc
sdv_IoGet
PIO_STACK
end_info
end_info
SetStatus
pirp->Can
ps->Minor
stub_disp
switch (x
ps->Major
PptDispat
5: PFDO_EX
7: if(Dev
8: PptFdoC
9: PFDO_EX
12: do_pag
19: if(fd
24: P4Comp
63: P4Comp
1797: P4
1782:
1783:
1784:
Step: 1359
State
(G
(completio
Source Code
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
38: goto target_exit;
39: }
40:
41: //
42: // We have the RemoveLock
43: //
44:
45: ExAcquireFastMutex(&fdx->OpenCloseMutex);
46: if(fdx->OpenCloseRefCount > 0) {
47: //
48: // prevent rollover - strange as it may seem, it is perfectly
49: // legal for us to receive more closes than creates - this
50: // info came directly from Mr. PnP himself
51: //
52: if(((LONG)InterlockedDecrement(&fdx->OpenCloseRefCount)) < 0) {
53: // handle underflow
54: InterlockedIncrement(&fdx->OpenCloseRefCount);
55: }
56: }
57: ExReleaseFastMutex(&fdx->OpenCloseMutex);
58:
59: target_exit:
60:
61: DD((PCE)fdx,DDT,"PptFdoClose - OpenCloseRefCount after close = %d\n",fdx->OpenCloseRe
62:
63: return P4CompleteRequestReleaseRemLock(Irp, STATUS_SUCCESS, 0, &fdx->RemoveLock);
64: }
65:
File: ../../../.././fdoclose.c, Line: 63, Function 'PptFdoClose'
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
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
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1362
State
File: ../../../.././utils.c, Line: 1797, Function 'P4CompleteRequestReleaseRemLock'
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
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
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788:
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
Step: 1365
State
File: ../../../.././utils.c, Line: 1782, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.


sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
(SLAM_NT_SU
eChoice
tch (choice
DispatchFun
v_IoGetCurr
O_STACK_LOC
d_info = st
d_info = st
tStatus
rp->CancelR
->MinorFunc
ub_dispatch
itch (x) {
->MajorFunc
tDispatchCl
PFDO_EXTENS
if(DevType
PptFdoClose
PFDO_EXTENS
do_paged_c
if(fdx->P
P4Complete
P4Complete
'97: P4Comp
1782: Irp-
1783: Irp-
1784: SLIC
Step: 1366
State
(G
(completio
Source Code
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
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788: □
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
File: ../../../.././utils.c, Line: 1783, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.


sdvdefect
File Font Trace Tree Source Code Help
Trace Tree
(SLAM_NT_SU
eChoice
tch (choice
DispatchFun
v_IoGetCurr
D_STACK_LOC
d_info = st
d_info = st
tStatus
rp->CancelR
->MinorFunc
ub_dispatch
itch (x) {
->MajorFunc
tDispatchCl
PFDO_EXTENS
if(DevType
PptFdoClose
PFDO_EXTENS
do_paged_c
if(fdx->P
P4Complete
P4Complete
'97: P4Comp
1782: Irp-
1783: Irp-
1784: SLIC
Step: 1368
State
(G
(completio
Source Code
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
1775: P4CompleteRequest(
1776: IN PIRP Irp,
1777: IN NTSTATUS Status,
1778: IN ULONG_PTR Information
1779:)
1780: {
1781: P5TraceIrpCompletion(Irp);
1782: Irp->IoStatus.Status = Status;
1783: Irp->IoStatus.Information = Information;
1784: IoCompleteRequest(Irp, IO_NO_INCREMENT);
1785: return Status;
1786: }
1787:
1788: □
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: {
1797: P4CompleteRequest(Irp, Status, Information);
1798: PptReleaseRemoveLock(RemLock, Irp);
1799: return Status;
1800: }
1801:
1802:
1803: // pcutil.c follows:
File: ../../../.././utils.c, Line: 1784, Function 'P4CompleteRequest'
Driver: src\kernel\parport Rule: DoubleCompletion Defect: The driver is calling IoCompleteRequest twice.

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

SLAM/SDV History

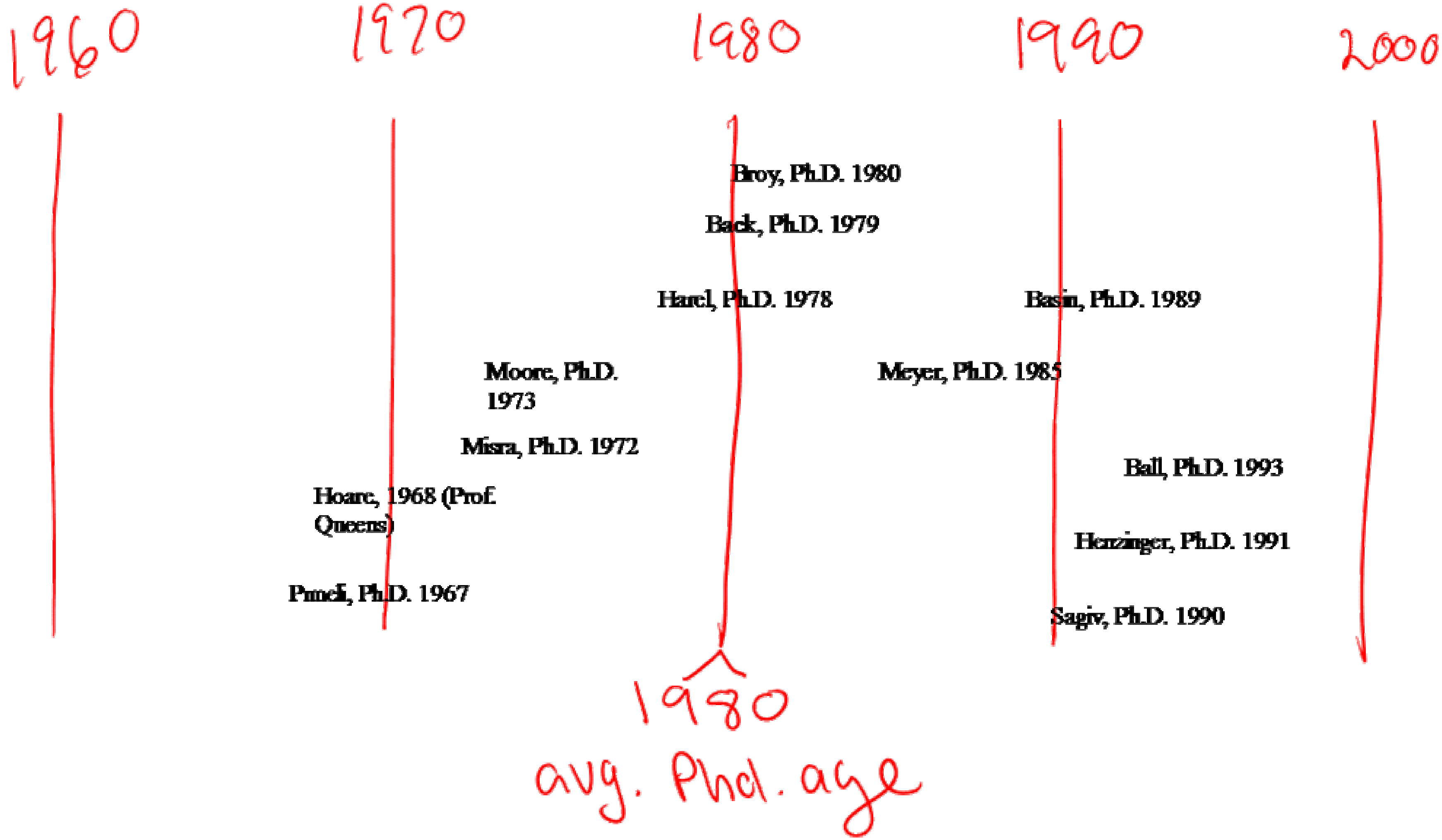
- **2000-2001**
  - foundations, algorithms, prototyping
  - papers in CAV, PLDI, POPL, SPIN, TACAS
- **March 2002**
  - Bill Gates review
- **May 2002**
  - Windows committed to hire two Ph.D.s in model checking to support Static Driver Verifier
- **July 2002**
  - running SLAM on 100+ drivers, 20+ 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

A Brief History of Verification



A Brief History of Verification

1960 1970 1980 1990 2000

Broy, Ph.D. 1980

Back, Ph.D. 1979

Harel, Ph.D. 1978

Basin, Ph.D. 1989

Moore, Ph.D. 1973

Meyer, Ph.D. 1985

Misra, Ph.D. 1972

Ball, Ph.D. 1993

Hoare, 1968 (Prof. Queens)

Henzinger, Ph.D. 1991

Pnueli, Ph.D. 1967

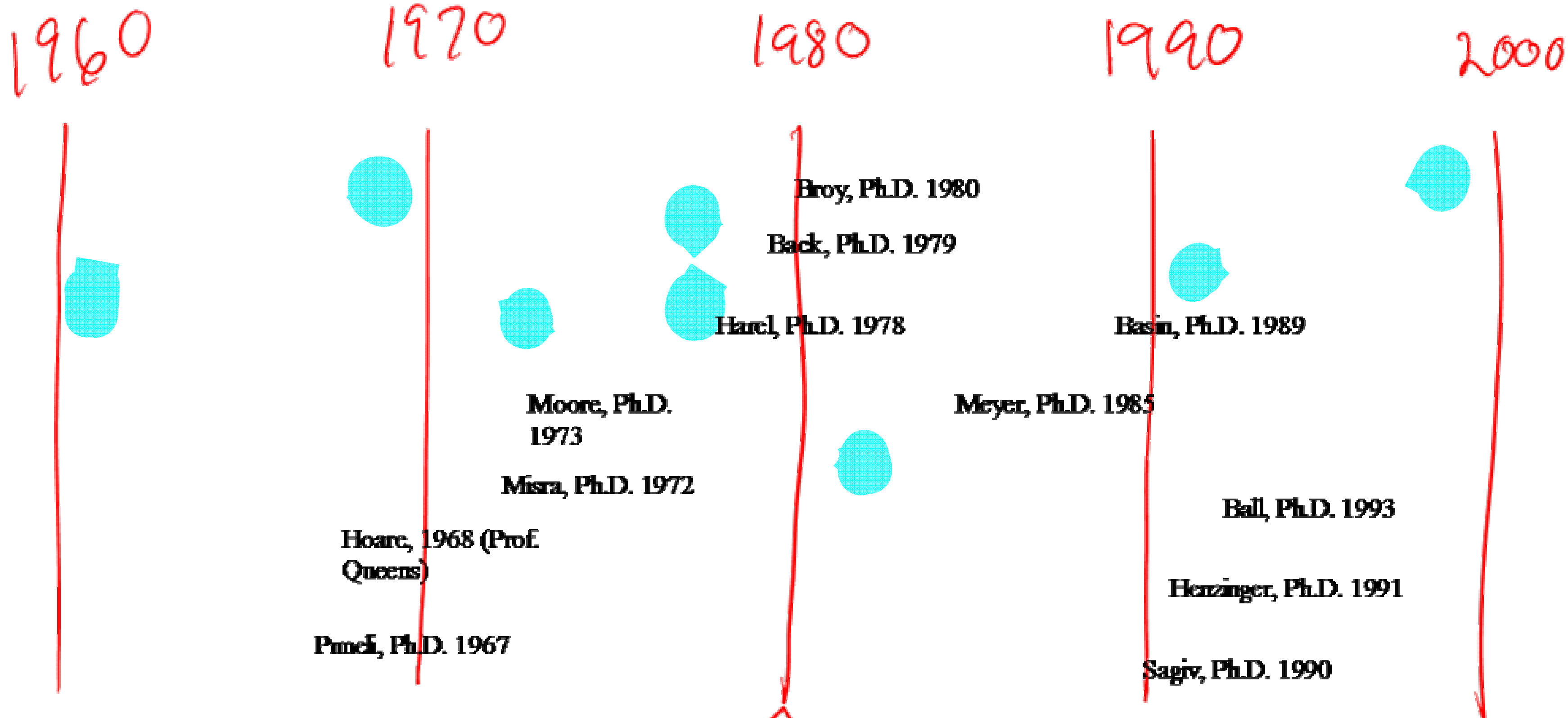
Sagiv, Ph.D. 1990

Program analysis
Hoare logic
Temporal logic
Abstract interpretation

1980
avg. PhD. age

Model checking
Auto. theorem proving
Symbolic model checking
Predicate abstraction

A Brief History of Verification

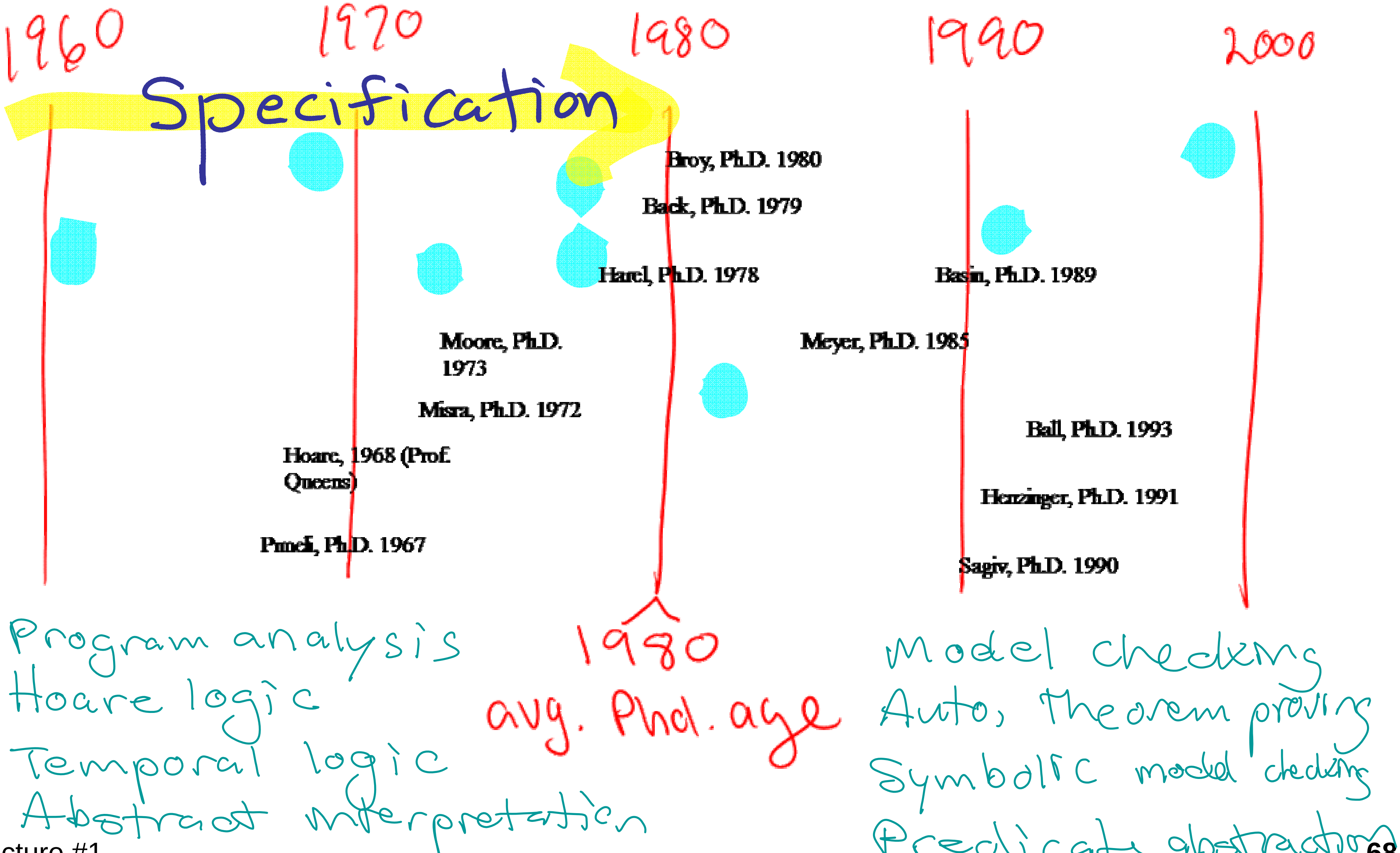


Program analysis
Hoare logic
Temporal logic
Abstract interpretation

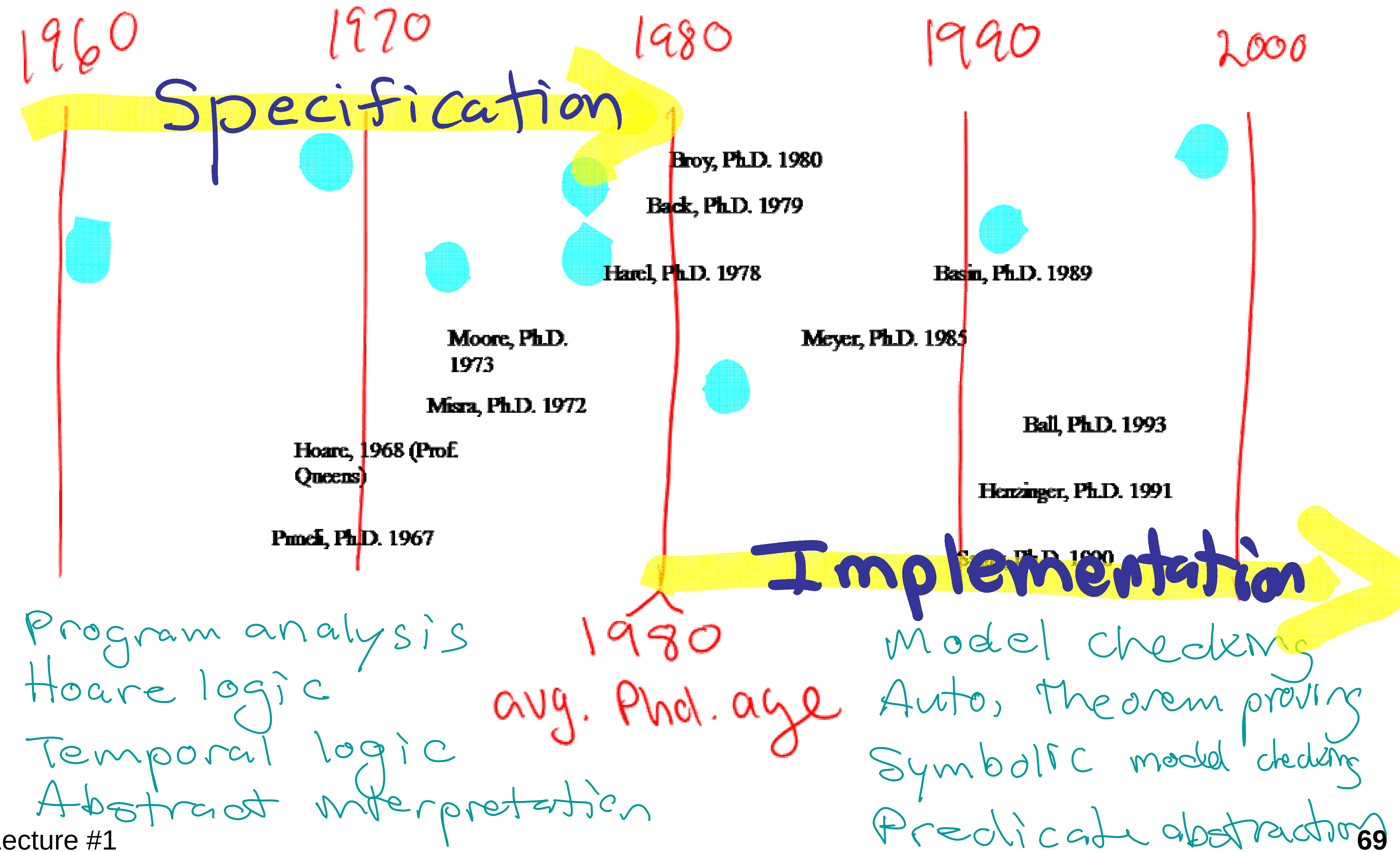
1980
avg. PhD. age

Model checking
Auto. theorem proving
Symbolic model checking
Predicate abstraction

A Brief History of Verification



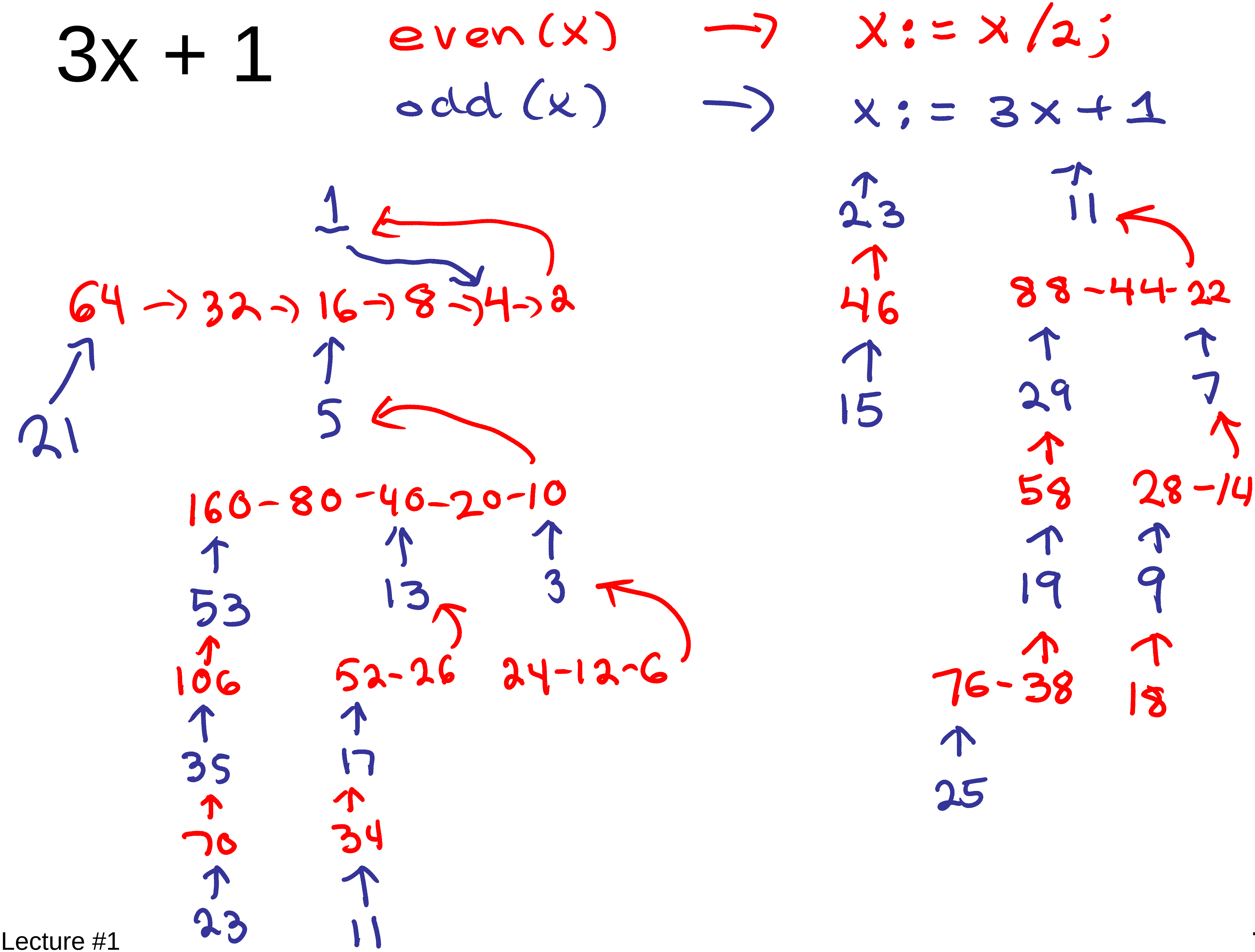
A Brief History of Verification



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.

3x + 1

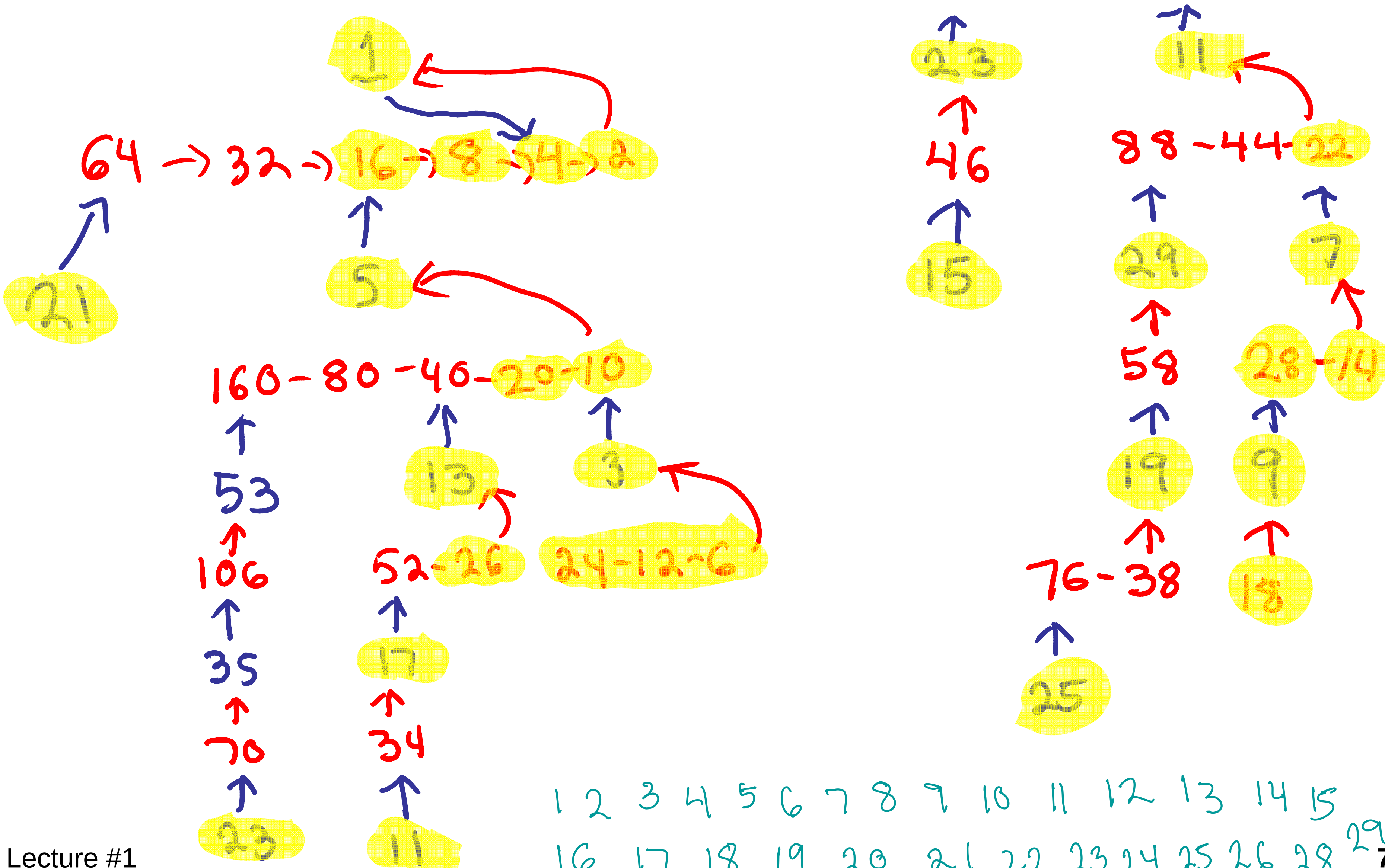


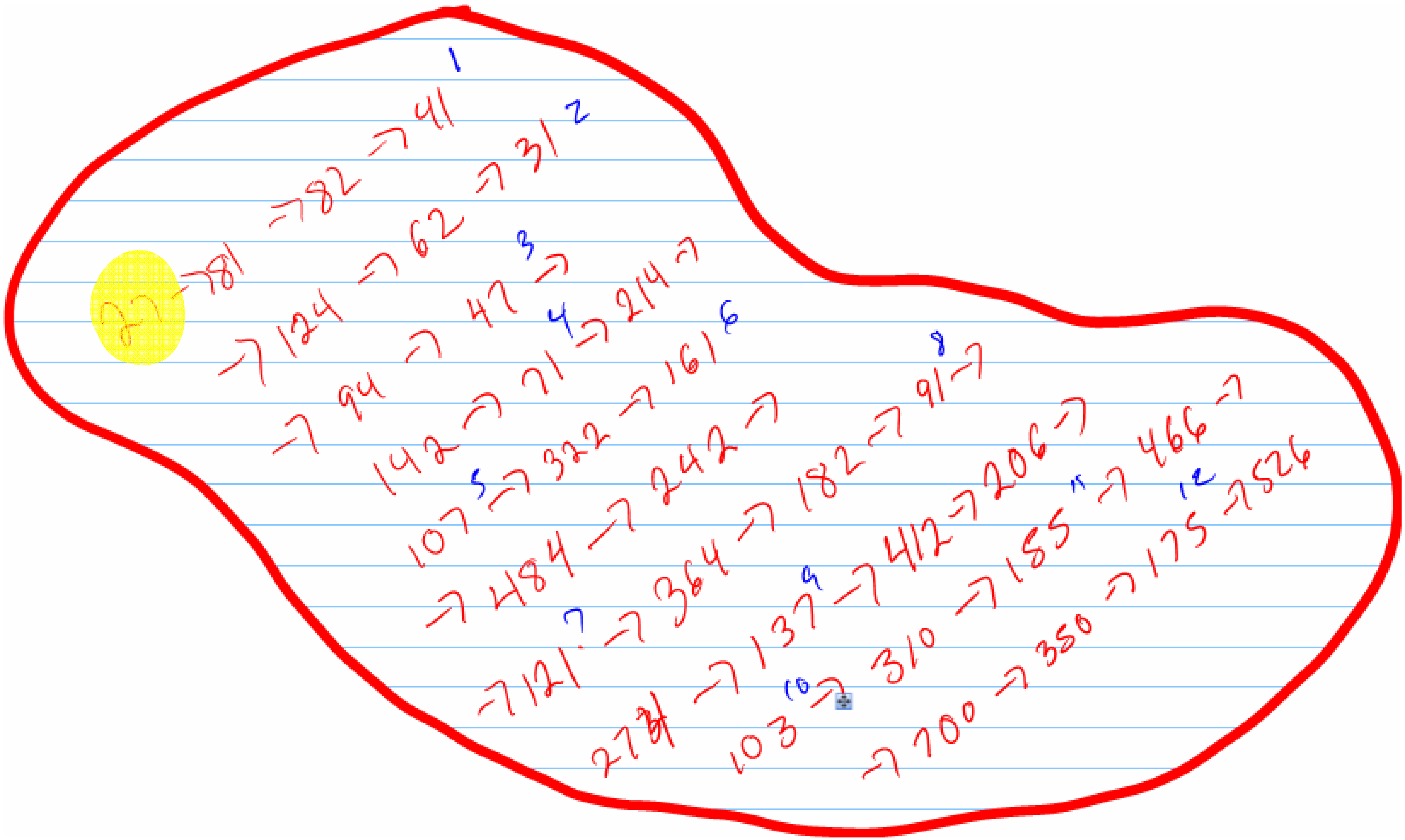
even(x) → x := x/2;
odd(x) → x := 3x+1
1
64 → 32 → 16 → 8 → 4 → 2
21
5
160-80-40-20-10
53
13
3
106
52-26
24-12-6
35
17
70
34
23
11
23
46
15
11
88-44-22
29
7
58
28-14
19
9
76-38
18
25

3x + 1

even(x) → $x := x/2;$

odd(x) → $x := 3x + 1$





526 → 263 [13] → 790 → 395 [14] → 1186 → 593 [15] → 1780

→ 890 → 445 [16] → 1336 → 668 → 334 → 167 [17] → 502

→ 281 [18] → 754 → 377 [19] → 1132 → 566 → 283 [20] →

→ 850 → 425 [21] → 1276 → 638 → 319 [22] → 956 →

→ 478 → 289 [23] → 868 → 434 → 217 [24] → 652 →

→ 326 → 163 [25] → 490 → 245 [26] → 736 → 368 → 184

→ 92 → 46 → 23 [27]