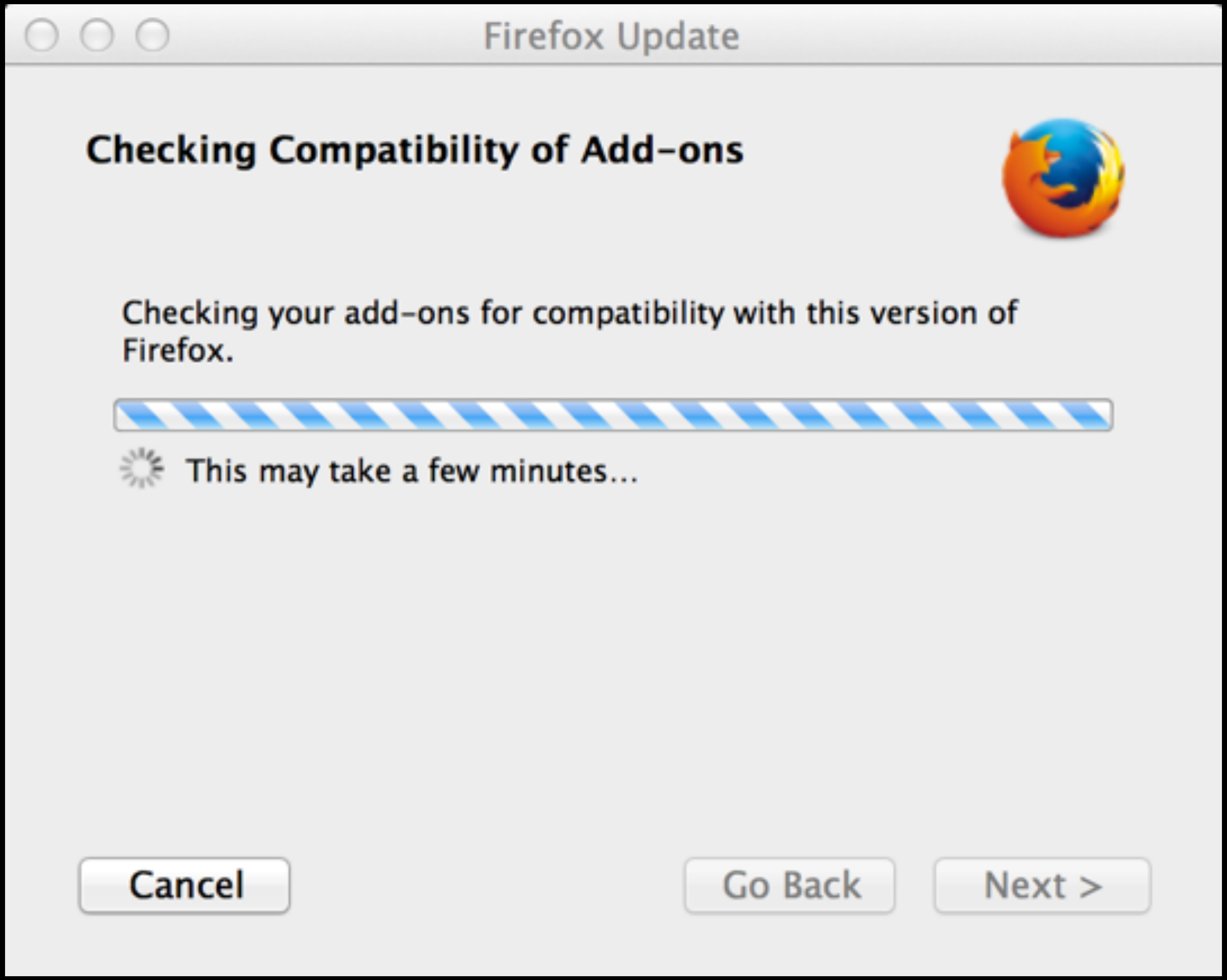
Compatibility Testing using Patterns-based Trace Comparison

Venkatesh-Prasad Ranganath, Kansas State University, USA
Pradip Vallathol, University of Wisconsin-Madison, USA
Pankaj Gupta, Microsoft Corporation, USA

*29th IEEE/ACM International Conference on
Automated Software Engineering (ASE 2014)
September 15 - 19, 2014
Västerås, Sweden*

Thanks to Microsoft Research (India) and USB team (at Microsoft) for supporting this effort.

Compatibility Testing



Firefox Update

Checking Compatibility of Add-ons

Checking your add-ons for compatibility with this version of Firefox.

This may take a few minutes...

Compatibility Testing: Syntactic Changes

client.java

```
service.getProperty(“name”)
```

service.java

```
void getProperty(PropertyId id) {
    …
}
```

Compatibility Testing: Semantic Changes

client1.c

```
s.q = c;
f(&s);
s.q = c;
g(&s);
```

client2.c

```
s.q = c;
f(&s);

g(&s);
```

serviceV1.c

```
@pre s.q == c
void f(Record *s) {
   …
   // no changes to s.q
}


@pre s.q == c
void g(Record *s) {
   …
}
```

Compatibility Testing: Semantic Changes

client1.c

```
s.q = c;
f(&s);
s.q = c;
g(&s);
```

client2.c

```
s.q = c;
f(&s);

g(&s);
```

Incorrect!!

serviceV2.c

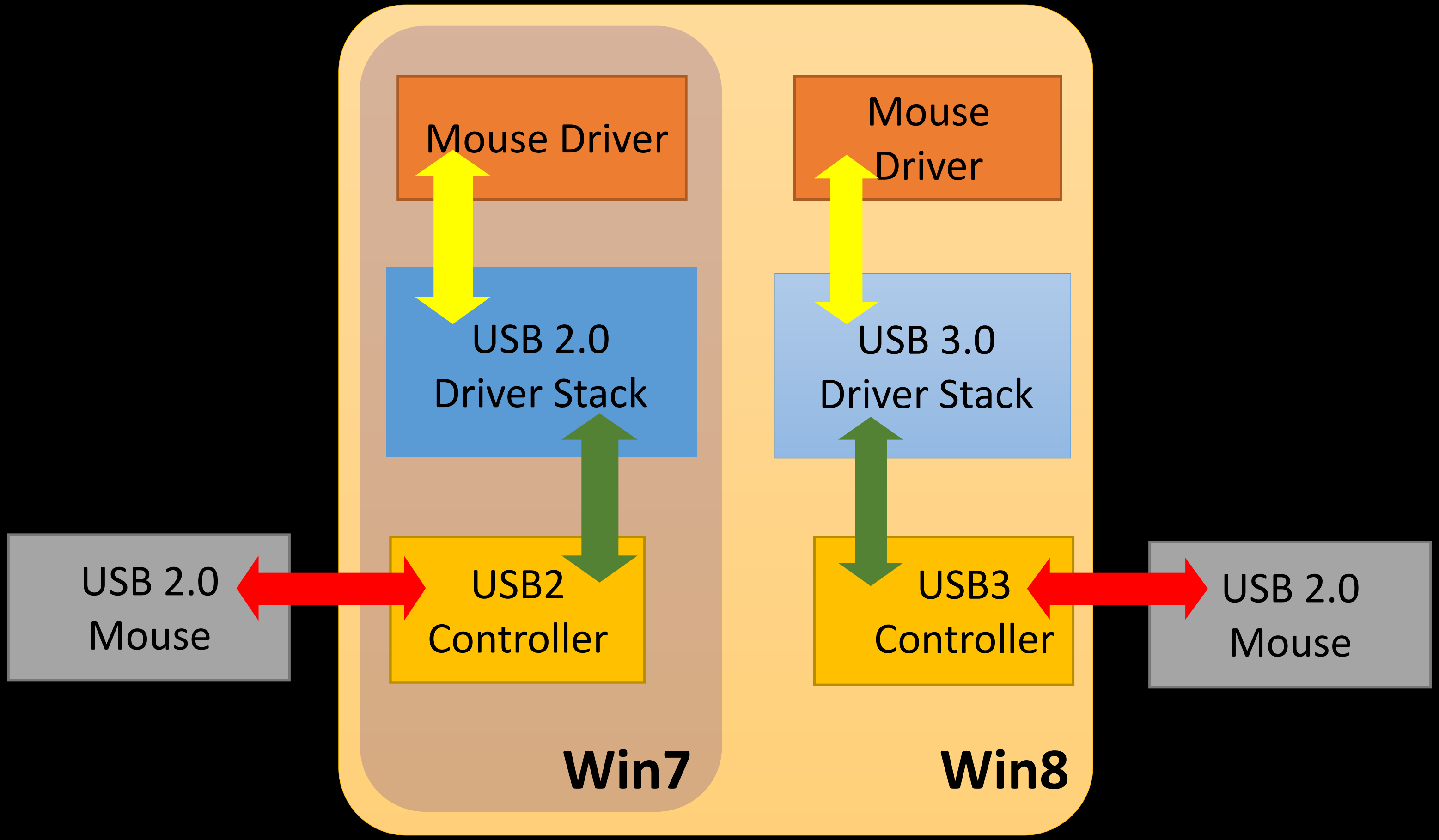
```
@pre s.q == c
void f(Record *s) {
    …
    s.q = 0;
}


@pre s.q == c
void g(Record *s) {
    …
}
```

Common Reasons for Semantic Incompatibilities

- Breaking semantic changes
- Observational dependences and influences
- Weak specifications
- Assumptions

Compatibility Testing of Windows USB drivers

When a USB 2.0 device is plugged into a USB 3.0 port on Win8, will USB 3.0 driver in Win8 behave similar to the USB 2.0 stack in Win7 (along both software and hardware interfaces)?

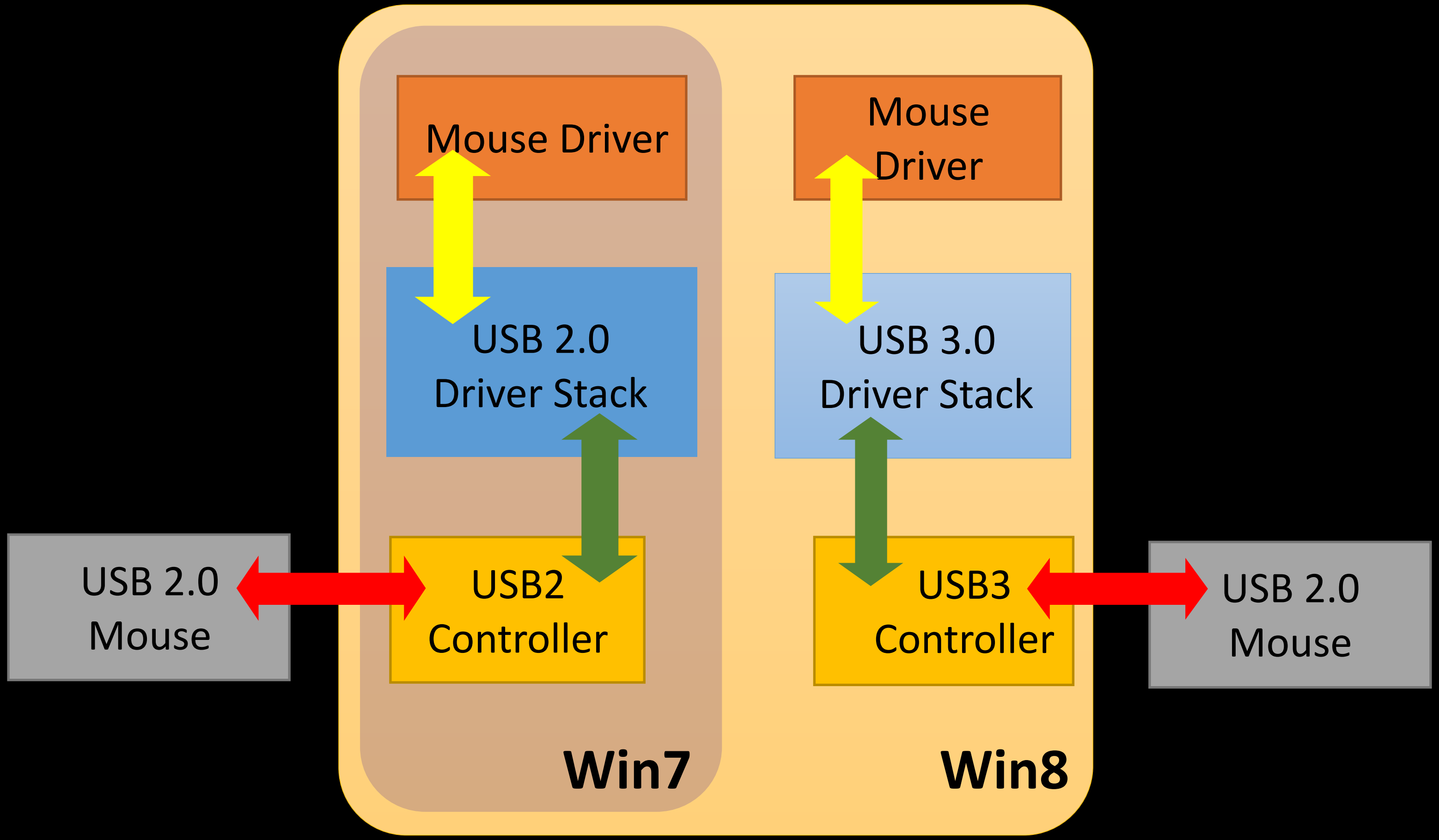


Why is it hard?

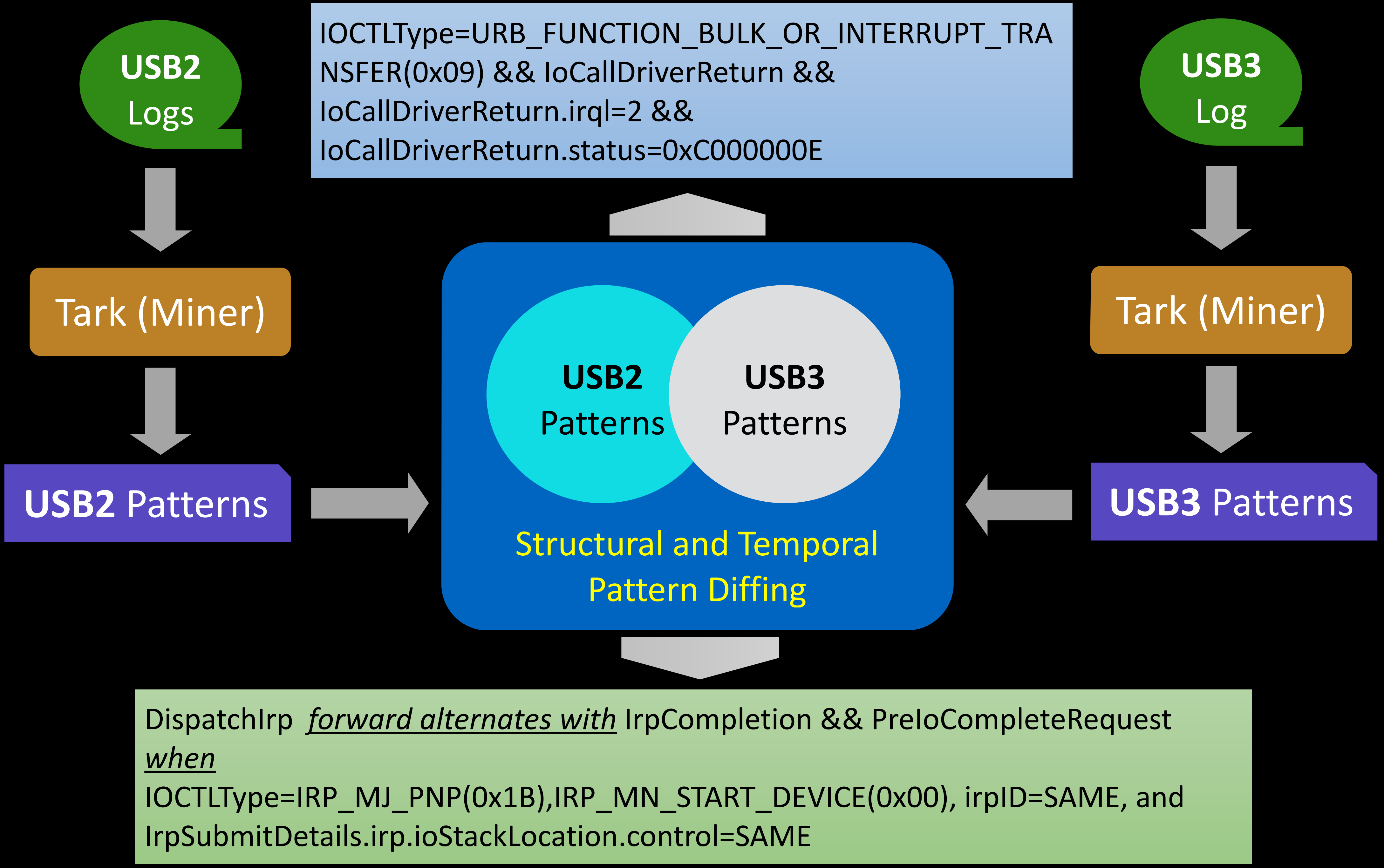
- Clean room implementation of USB 3 driver
  - No part of USB 2 driver was reused
- Regression tests were insufficient
- Large testing surface
  - Number of unique USB devices
  - Possibilities in USB protocol
- Multiple layers of variability
  - Device drivers, Controllers, & ASIC in devices

Compatibility Testing of Windows USB drivers

When a USB 2.0 device is plugged into a USB 3.0 port on Win8, will USB 3.0 driver in Win8 behave similar to the USB 2.0 stack in Win7 (along both software and hardware interfaces)?



Compatibility Testing using Patterns-based Trace Comparison



USB2 Logs

Tark (Miner)

USB2 Patterns

IOCTLType=URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER(0x09) && IoCallDriverReturn && IoCallDriverReturn.irql=2 && IoCallDriverReturn.status=0xC000000E

USB2 Patterns

USB3 Patterns

Structural and Temporal Pattern Diffing

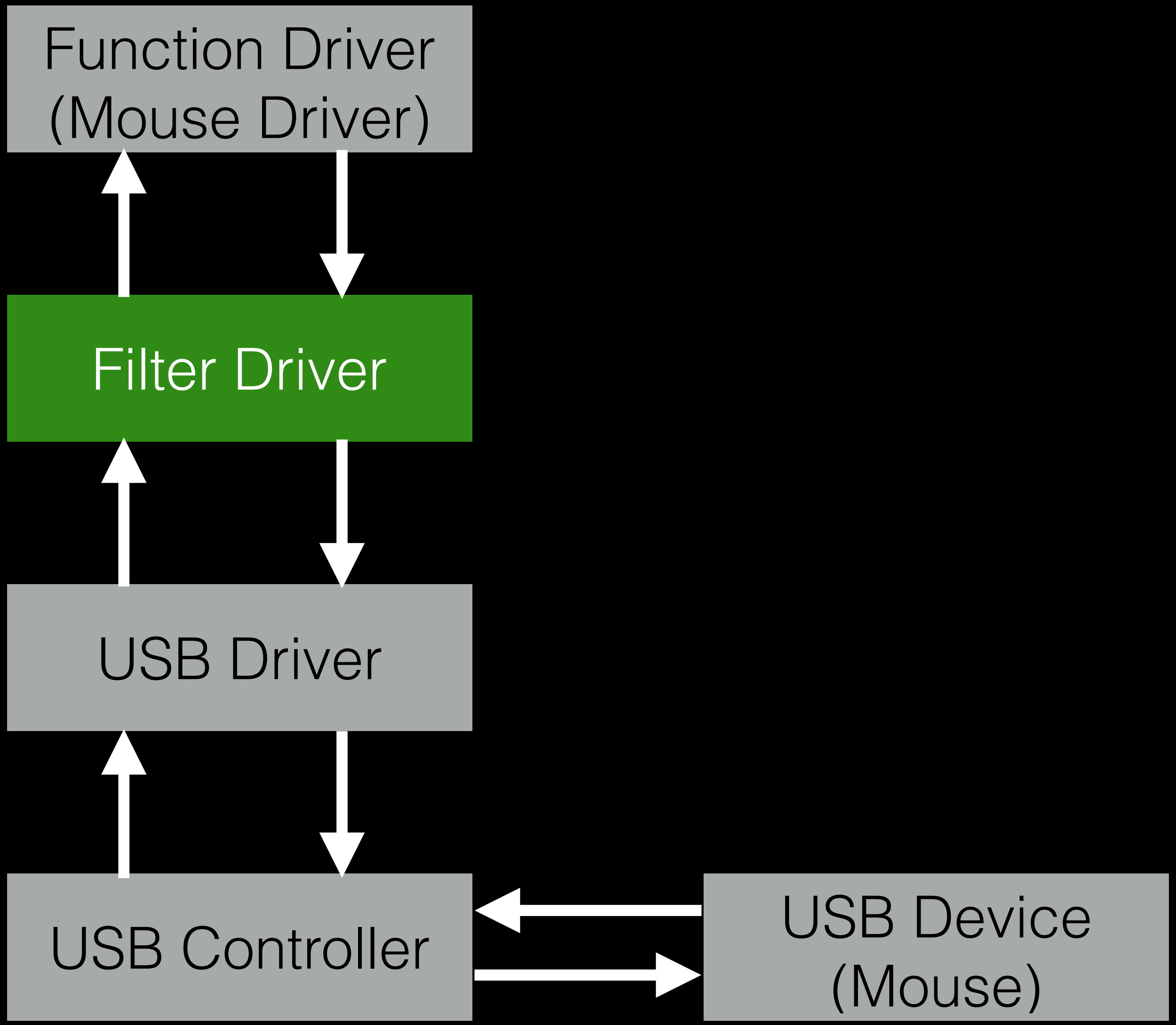
DispatchIrp *forward alternates with* IrpCompletion && PreIoCompleteRequest *when* IOCTLType=IRP_MJ_PNP(0x1B),IRP_MN_START_DEVICE(0x00), irpID=SAME, and IrpSubmitDetails.irp.ioStackLocation.control=SAME

USB3 Log

Tark (Miner)

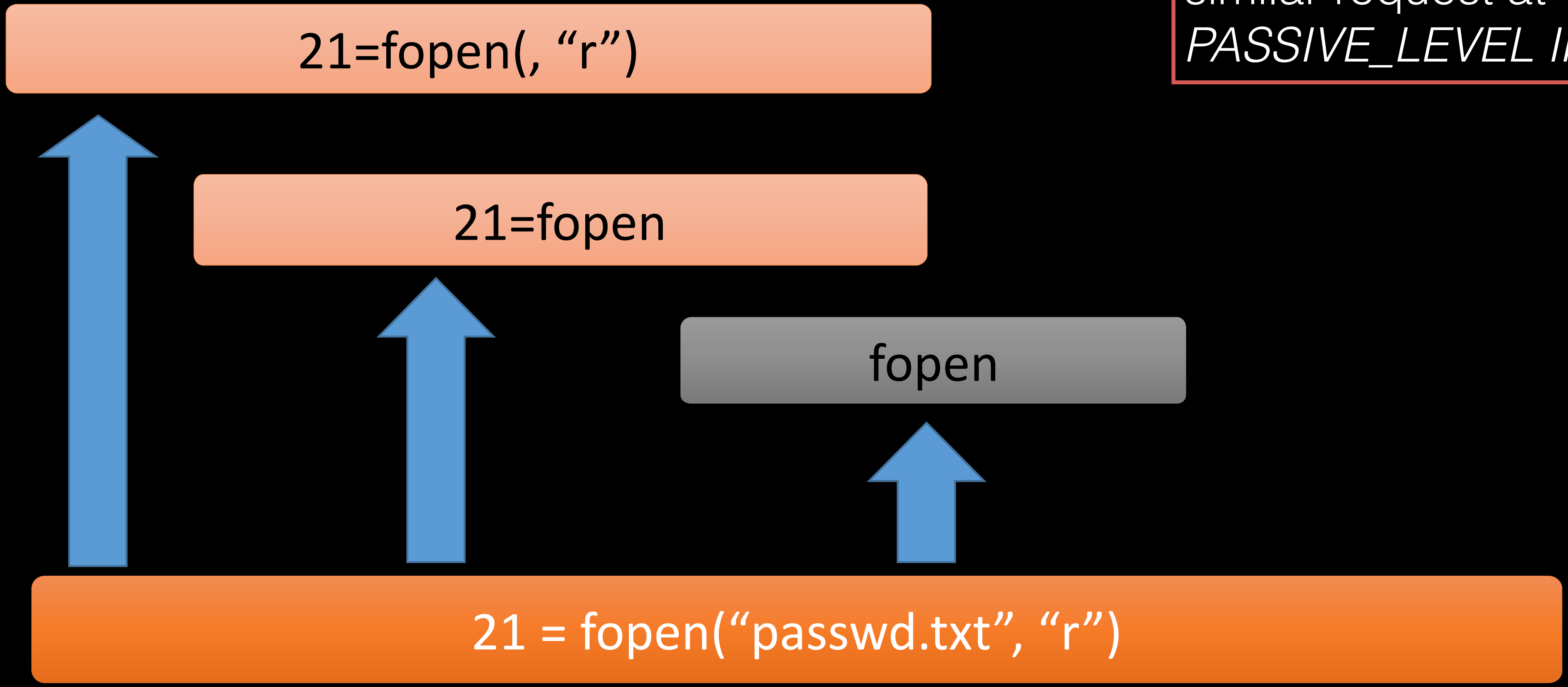
USB3 Patterns

Trace Collection



Structural Patterns

USB 2.0 driver completed *isochronous requests* at *DISPATCH_LEVEL IRQ* while USB 3.0 driver completed similar request at *PASSIVE_LEVEL IRQ*.



21=fopen(, “r”)

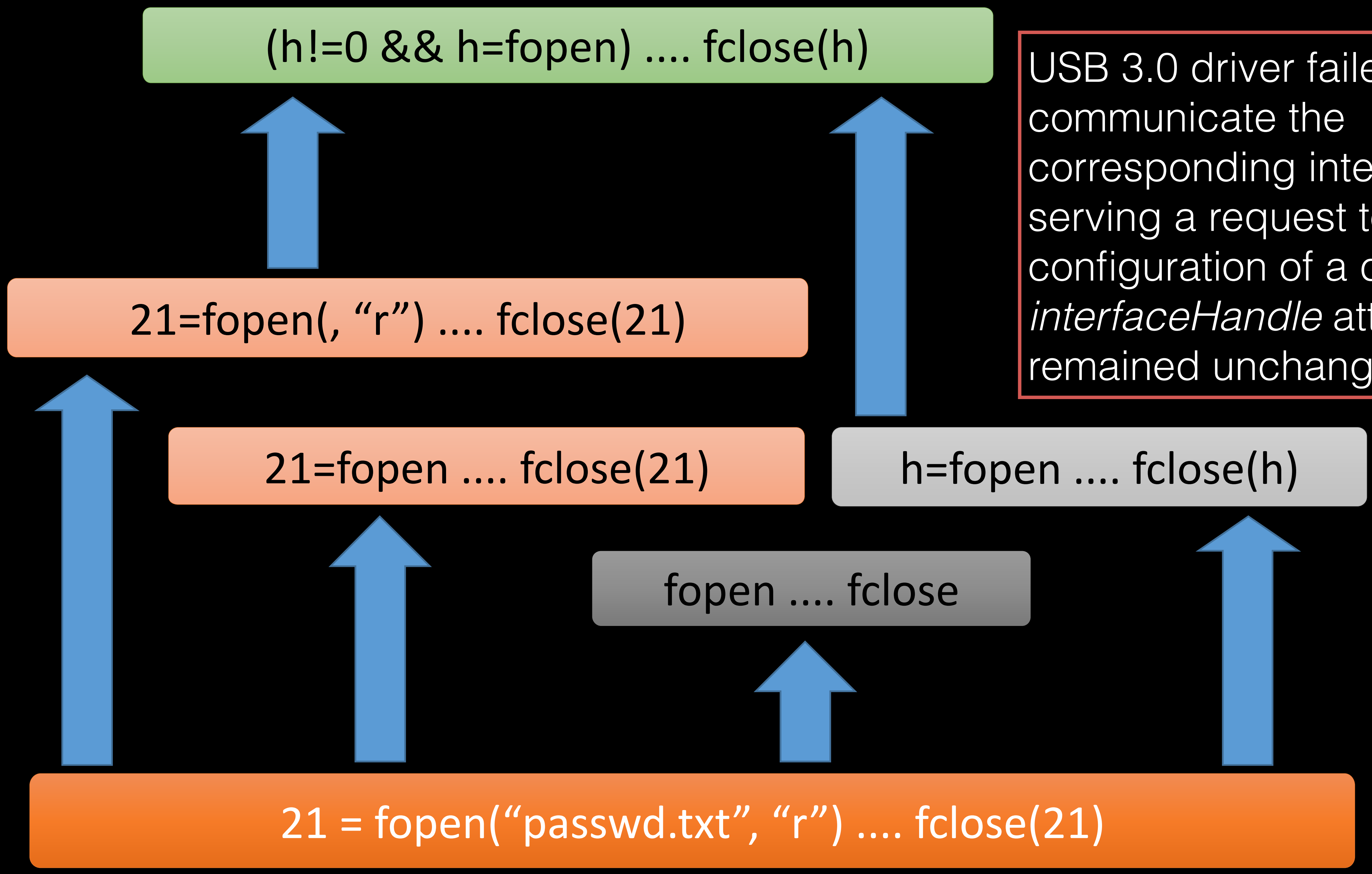
21=fopen

fopen

21 = fopen(“passwd.txt”, “r”)

Mining Quantified Temporal Rules: Formalism, Algorithms, and Evaluation, WCRE’09

Temporal Patterns with Data Flow



(h!=0 && h=fopen) .... fclose(h)

21=fopen(, “r”) .... fclose(21)

21=fopen .... fclose(21)

h=fopen .... fclose(h)

fopen .... fclose

21 = fopen(“passwd.txt”, “r”) .... fclose(21)

USB 3.0 driver failed to communicate the corresponding interface when serving a request to select a configuration of a device. *interfaceHandle* attribute remained unchanged.

Mining Quantified Temporal Rules: Formalism, Algorithms, and Evaluation, WCRE’09

What is reported?

Presence of previously unobserved patterns

$$USB3(dev_k) - \bigcup_i USB2(dev_i)$$

Absence of previously observed patterns

$$\bigcup_i USB2(dev_i) - USB3(dev_k)$$

Comment: This should be intersection

Is it effective?

We detected 14 unique bugs (25 bugs) by testing 14 devices with regression tested USB 3.0 driver.

Is it expensive?

- Worst case mining time was 115 minutes
- Worst case diffing time was 48 minutes
- *Non-empty* reports analysis took ~2 hours
    - Few reports required 24 hours

Domain Knowledge

# of attributes: 361

# of ignored attributes: 108 (361 - 108 = 253)

# of necessary attributes: 29 (253 - 29 = 224)

# of NULL abstracted attributes: 23

# of unquantifiable attributes: 75

# of quantifiable attributes: 150

# of data flows: 17 (between 26 attributes)

User Feedback

Device	Known	Detected	Simplified	Compacted	Reported	False +ve	Structural	Temporal
1	0	9844	932	478	478	11 + 454	6 / 9	4 / 4
2*	932	2545	121	63	15	0 + 11	1 / 1	1 / 3
3	965	743	41	21	4	1 + 0	0 / 0	1 / 3
4	965	1372	67	34	2	1 + 1	0 / 0	0 / 0
5*	2141	26118	1114	571	55	26 + 29	0 / 0	0 / 0
6	2141	26126	1054	541	0	0 + 0	0 / 0	0 / 0
7	2141	2320	84	44	0	0 + 0	0 / 0	0 / 0
8	2141	27804	1185	608	2	1 + 0	1 / 1	0 / 0
9	2141	34985	413	217	115	2 + 96	2 / 14	2 / 3
10	2141	51556	429	231	59	15 + 41	1 / 1	2 / 2
11	2141	695	35	18	0	0 + 0	0 / 0	0 / 0
12	2141	1372	67	34	0	0 + 0	0 / 0	0 / 0
13	2141	3315	122	72	24	19 + 4	1 / 1	0 / 0
14*	2141	9299	103	54	3	0 + 0	2 / 3	0 / 0

Smart Presentation

Device	Known	Detected	Simplified	Compacted	Reported	False +ve	Structural	Temporal
1	0	9844	932	478	478	11 + 454	6 / 9	4 / 4
2*	932	2545	121	63	15	0 + 11	1 / 1	1 / 3
3	965	743	41	21	4	1 + 0	0 / 0	1 / 3
4	965	1372	67	34	2	1 + 1	0 / 0	0 / 0
5*	2141	26118	1114	571	55	26 + 29	0 / 0	0 / 0
6	2141	26126	1054	541	0	0 + 0	0 / 0	0 / 0
7	2141	2320	84	44	0	0 + 0	0 / 0	0 / 0
8	2141	27804	1185	608	2	1 + 0	1 / 1	0 / 0
9	2141	34985	413	217	115	2 + 96	2 / 14	2 / 3
10	2141	51556	429	231	59	15 + 41	1 / 1	2 / 2
11	2141	695	35	18	0	0 + 0	0 / 0	0 / 0
12	2141	1372	67	34	0	0 + 0	0 / 0	0 / 0
13	2141	3315	122	72	24	19 + 4	1 / 1	0 / 0
14*	2141	9299	103	54	3	0 + 0	2 / 3	0 / 0

Lessons Learned

- If domain knowledge is available, use it
- If a feedback loop can be established, set it up
- Presentation matters
- Embrace the unorthodox

Limitations

- Detects a class of incompatibilities

Threats to Validity

- Generalization needs more experiments
- Effect of latent factors need to be studied

Key Takeaways

- An approach to compatibility testing via patterns-based trace comparison.

- The use of structural and temporal patterns as trace abstractions to enable software engineering and maintenance tasks.

- Of course, the lessons learned :)