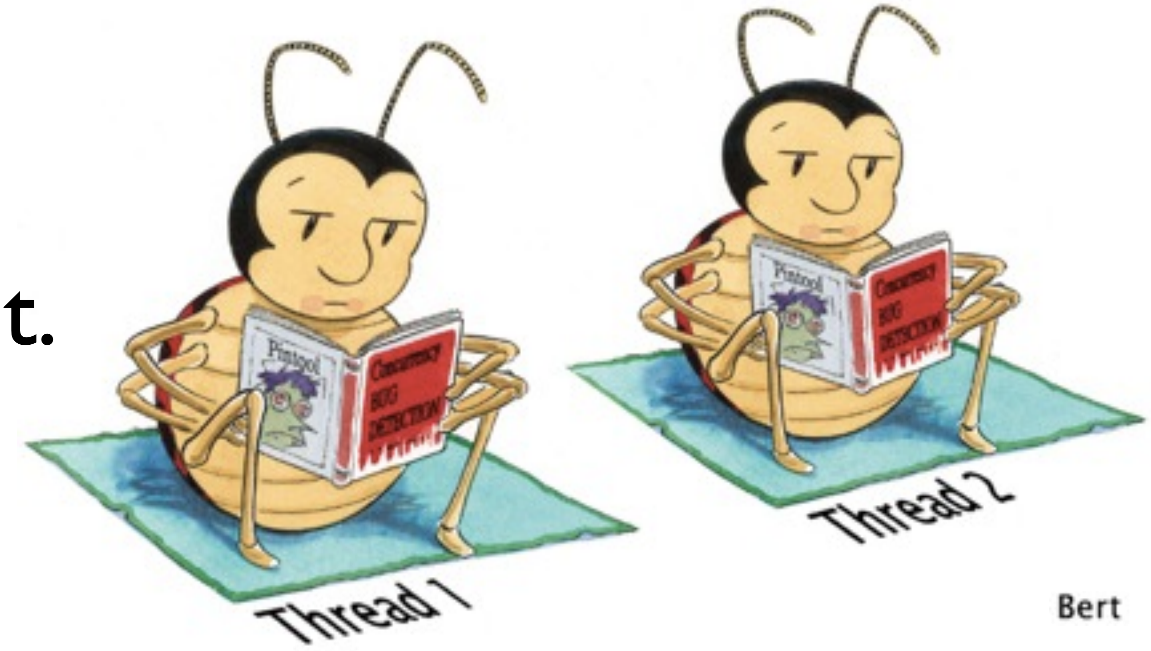
aComment:
Mining Annotations from Comments and Code to Detect Interrupt-Related Concurrency Bugs

Lin Tan, University of Waterloo, lintan@uwaterloo.ca
Yuanyuan (YY) Zhou, University of California, San Diego
Yoann Padioleau, Facebook Inc.

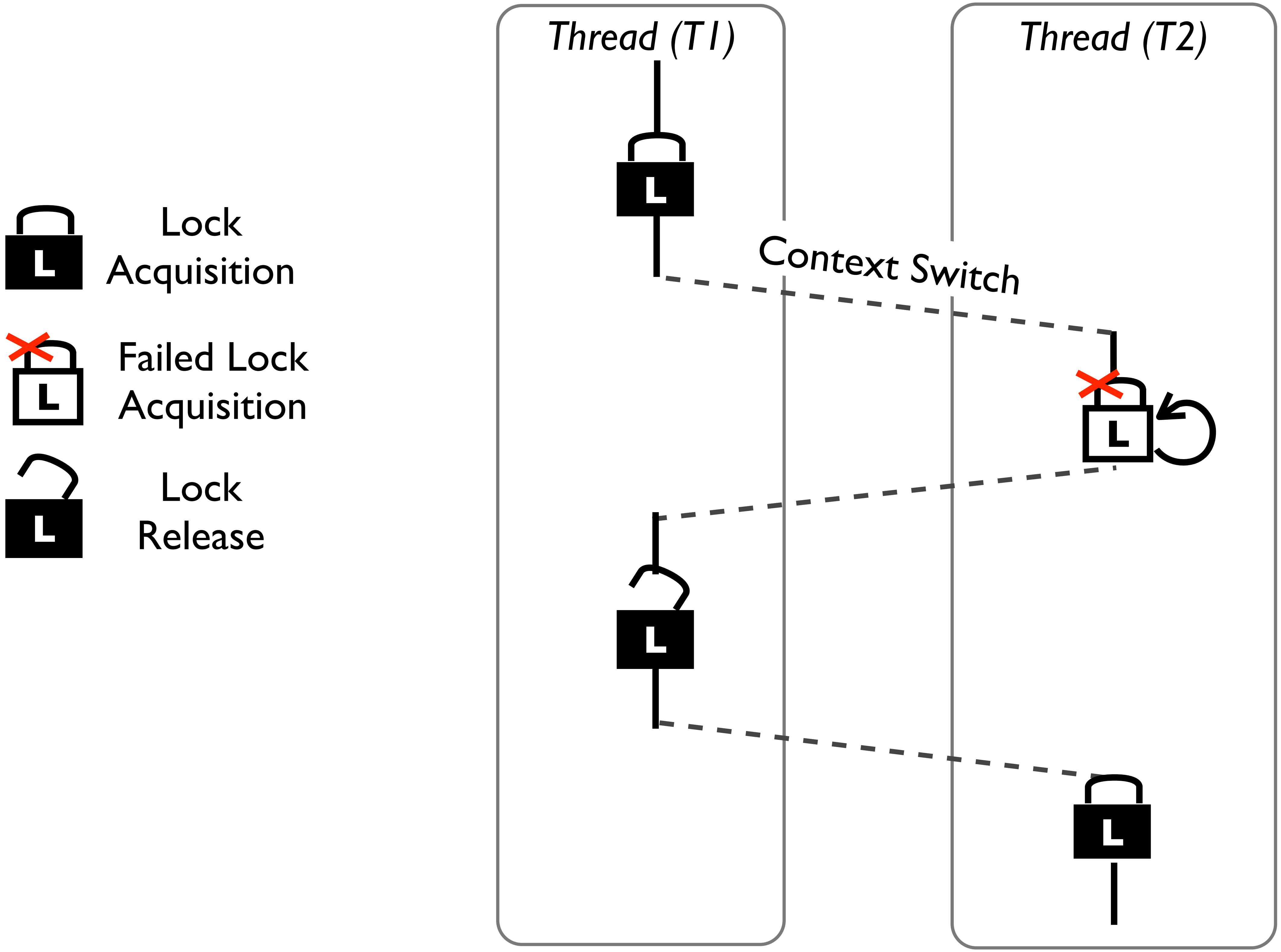
OS Concurrency Bugs are a Problem

- Concurrency bugs are pervasive and hard-to-detect.



- Operating System (OS) concurrency bugs can bring down all applications running on top of it.
- OS has a higher percentage of concurrency bugs than application software. [TanTechReport'11]
- 19% of OS driver bugs are concurrency bugs. [RyzhykEuroSys'09]

Interrupts Complicate OS Synchronization



Lock Acquisition

Failed Lock Acquisition

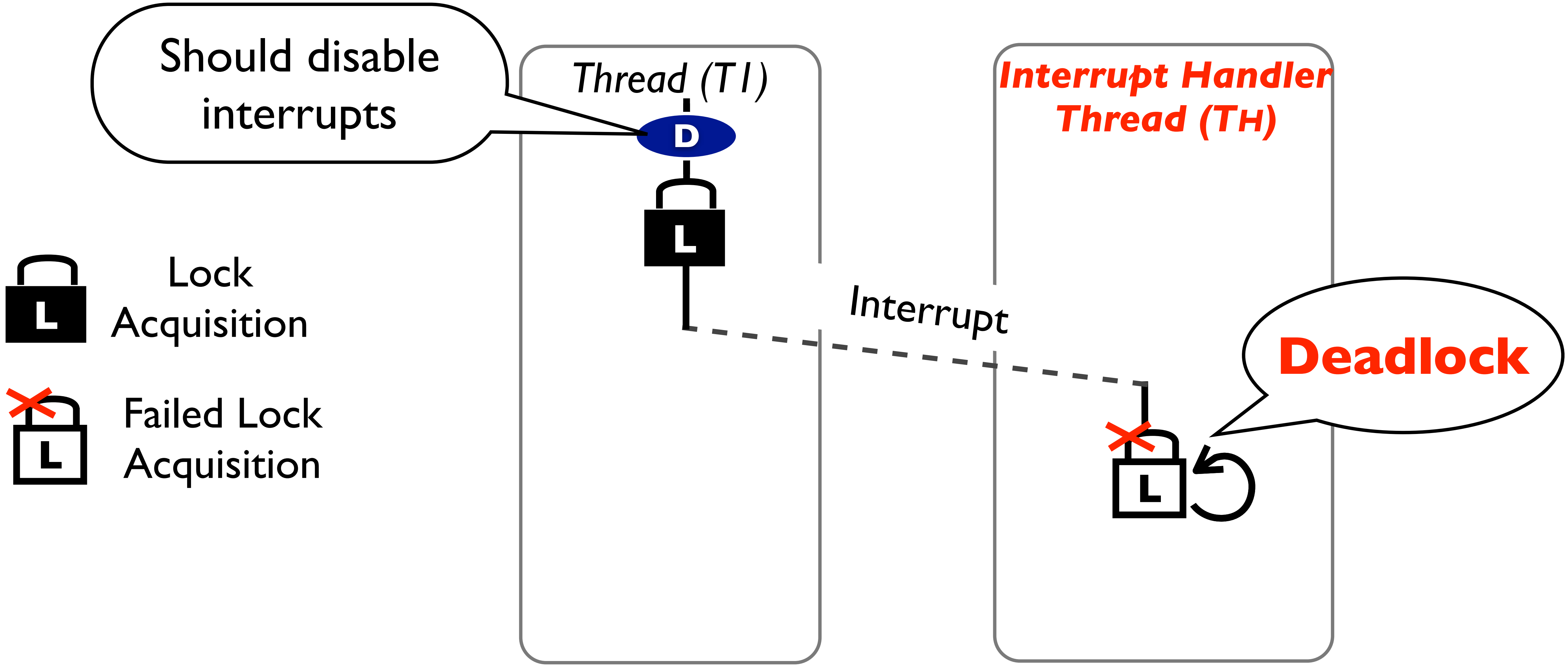
Lock Release

Thread (T1)

Thread (T2)

Context Switch

Interrupts Complicate OS Synchronization



- Interrupts can also cause other concurrency bugs.
- Hard to reason about interrupts because
  - Interrupts can happen at anytime.
  - Interrupts are relatively infrequent.
  - OS contains many interrupt handlers.

State-of-Art & Our Solution

- Most effective concurrency bug detection tools [SavageTOCS'97, ChoiPLDI'02, LuASPLOS'06, LuSOSP'07, HammerICSE'08, JulaOSDI'08, NaikICSE'09, BurnimICSE'10, LaiICSE'10]
    - do not consider interrupts
    - are dynamic tools designed for user-level applications.
- Dynamic approaches are cumbersome for OS:
    - difficult to instrument OS, low level, many drivers, large code sizes, complexity, ...

- **Our Solution**: **Static approach with interrupts in mind**

Goal

- Infer

  Annotations

  - **Precondition**: If interrupts should have already been disabled or enabled upon entry to a function, and
  - **Postcondition**: If interrupts should have already been disabled or enabled upon exit from the function
  - From **comments and code**

- Detect violations to these annotations statically

Inferring Annotations from Comments & Code

```
linux/kernel/time/tick-oneshot.c:
/* … Called with interrupts disabled. */
int  /*@IRQ(D, X)*/  tick_init_highres(void) {…}
```

```
linux/kernel/posix-cpu-timers.c:
void  /*@IRQ(D, X)*/  run_posix_cpu_timers(…)
{  BUG_ON(!irqs_disabled()); … }
```

Our Contributions

- ✦ **Feasible to extract annotations from comments & code**
  - Designed new interrupt-related annotations
  - Generated 96,821 **interrupt-related** annotations & automatically detected **9** true bugs in the Linux kernel
  - These annotations can help developers avoid bugs.
- ✦ **Combining comments & code help extract more annotations and detect more bugs than using comments or code alone.**

Outline

- Motivation & Contributions
- Annotation Design
- Annotation Extraction
  - From comments
  - From code
- Annotation Propagation & Bug Detection
- Results: Bug Detection & Annotation Extraction
- Related Work
- Conclusions

Annotation Language Design

@IRQ (Precondition, Postcondition)

D/E/X D/E/X

Value	Meaning
D	Interrupts are disabled.
E	Interrupts are enabled.
X	Don’t care

Example	Meaning
@IRQ (D, D)	Interrupts are disabled on entry and remain disabled on exit.
@IRQ (X, E)	Don’t-care on entry and interrupts are enabled on exit.
@IRQ (X, X)	Our design choice: Either @IRQ (D, D) or @IRQ (E, E)

Read our paper for the meaning of value ‘P’.

Annotation Extraction From Comments

Software	LOC	Sentence	IRQSent
Linux	5.2M	1,024,624	23,662
FreeBSD	2.4M	420,013	11,117
NetBSD	3.3M	680,650	23,942
OpenSolaris	3.7M	535,073	8,074
Total	**14.6M**	**2,660,360**	**66,795**

- Millions of lines of comments exist in OSs.
- We analyze comments as is: No need to rewrite comments.

Annotation Extraction From Comments

Contains no annotations

- /* Neither are the interrupt status bits */ (Linux)
- /* Called with interrupts disabled. */ (OpenSolaris)
- /* Disables interrupts before calling this function */ (NetBSD)
- /* Must be called with interrupts locked out */ (FreeBSD)

ID	Heuristics
1	<call> & <with> & <interrupt> (ordered)
2	<before> & <disable/enable> & <interrupt> (orderless)
3	<assume> & <**disable**/enable> & <interrupt> (orderless)

“disable”, “turn off”, “block”, “lock out”, ...

- Automatically extract function names and the preconditions (D or E).

Annotation Extraction From Code Assertions

Seed function

```
linux/kernel/posix-cpu-timers.c:
void   /*@IRQ(D, X)*/   run_posix_cpu_timers(…)
{ BUG_ON(!irqs_disabled()); … }
```

- Learn from dynamic assertions
- Can learn invariants from the majority of code
  [ErnstICSE'00], [EnglerSOSP'01], [HangalICSE'02], [LiFSE'05], [LivshitsFSE'05], [TanSecurity'08] …

- We directly extract annotations from seed functions' code and comments.
- Challenge: Scarceness of seed functions

Annotation Propagation

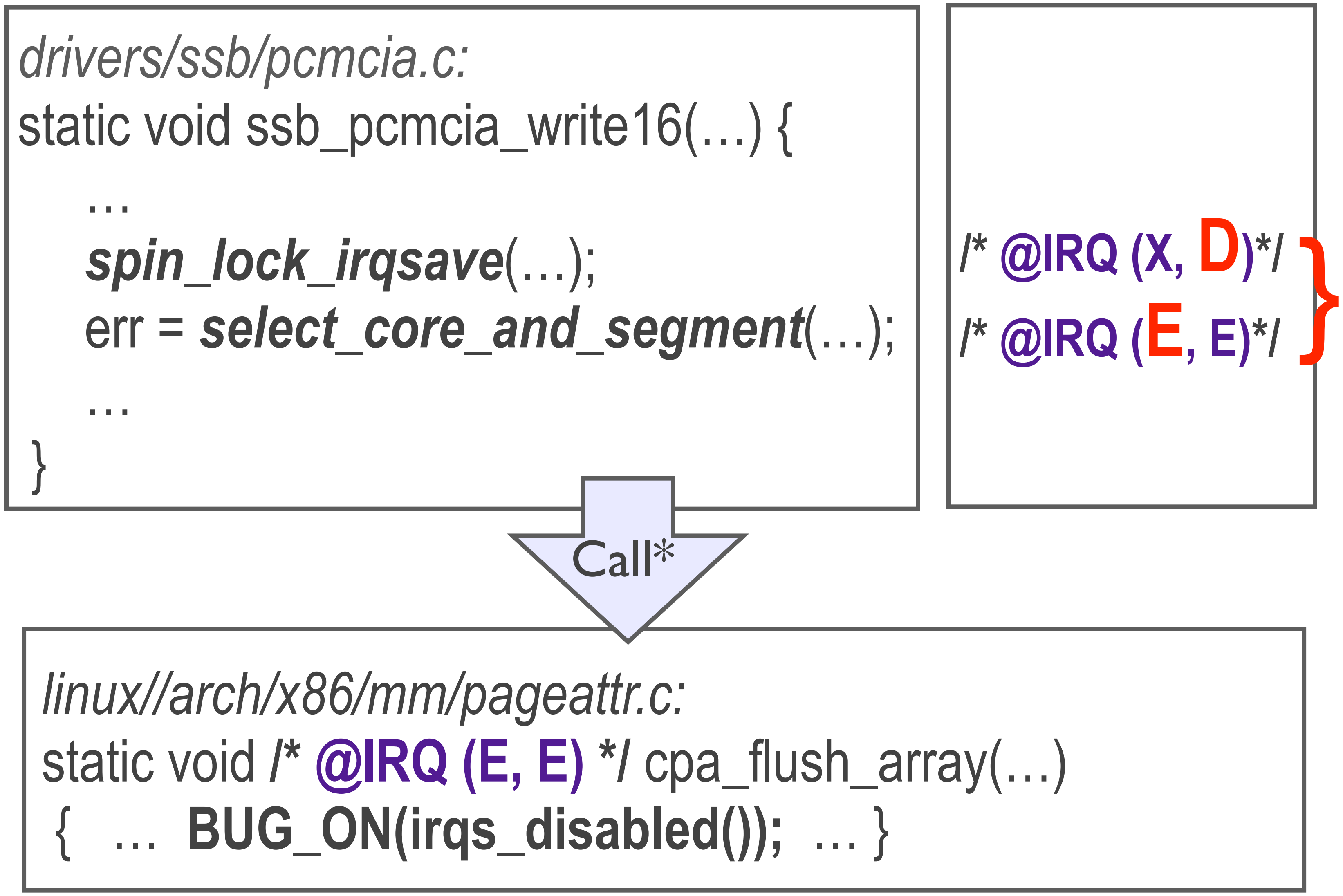
linux/kernel/timer.c:

```
void update_process_times(int user_tick)           @IRQ(D, D)
{
    struct task_struct p = get_current();          @IRQ(X, X)    ([illegible], [illegible])
    ...

    account_process_tick(p, user_tick);            @IRQ(D, D)    (D, D)
    run_local_timers();                            @IRQ(D, D)    (D, D)
    if (rcu_pending(cpu))                          @IRQ(X, X)    (D, D)
        rcu_check_callbacks(cpu, user_tick);       @IRQ(X, X)    (D, D)
    scheduler_tick();                              @IRQ(D, D)    (D, D)
    run_posix_cpu_timers(p);                       @IRQ(D, D)    (D, D)
}
```

- Initialize
  - only 8 **IRQ functions** (e.g., local_irq_disable) with (X, E), (X, D), etc.
  - **seed functions** with annotations extracted from comments and code

Bug Detection - Unsatisfiable Annotations



```
drivers/ssb/pcmcia.c:
static void ssb_pcmcia_write16(…) {
     …
     spin_lock_irqsave(…);
     err = select_core_and_segment(…);
     …
}
```

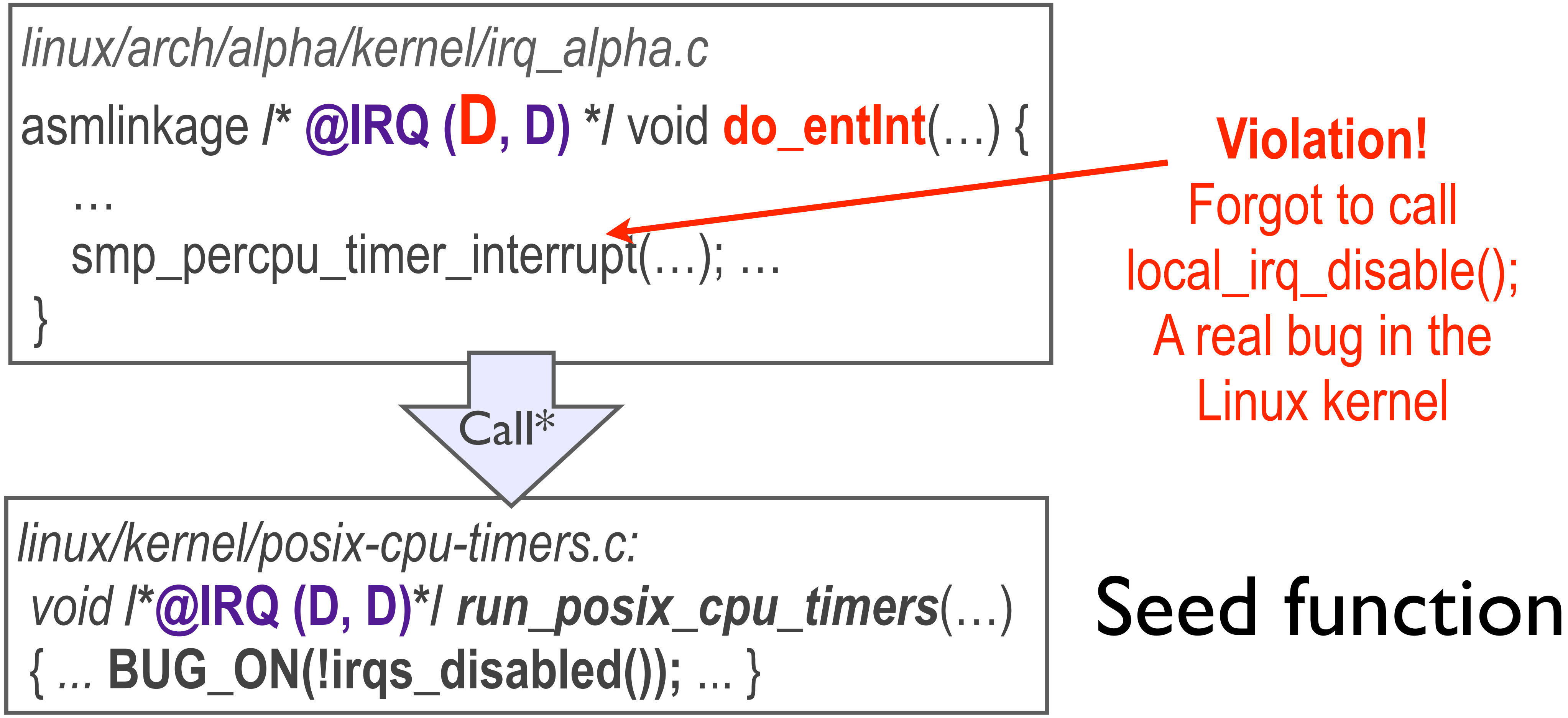
```
/* @IRQ (X, D)*/
/* @IRQ (E, E)*/
```

Violation!
A real bug
in the Linux
kernel

```
linux//arch/x86/mm/pageattr.c:
static void /* @IRQ (E, E) */ cpa_flush_array(…)
 {   …  BUG_ON(irqs_disabled());  … }
```

Seed function

Bug Detection - Root Function Annotations



linux/arch/alpha/kernel/irq_alpha.c

```
asmlinkage /* @IRQ (D, D) */ void do_entInt(…) {
   …
   smp_percpu_timer_interrupt(…); …
}
```

Violation!
Forgot to call
local_irq_disable();
A real bug in the
Linux kernel

linux/kernel/posix-cpu-timers.c:

```
 void /*@IRQ (D, D)*/ run_posix_cpu_timers(…)
{ ... BUG_ON(!irqs_disabled()); ... }
```

Seed function

- Root function do_entInt has no callers within a module.
  - No guaranteed that external callers will disable interrupts.

Outline

- Motivation & Contributions
- Annotation Design
- Annotation Extraction
  - From comments
  - From code
- Annotation Propagation & Bug Detection
- Results: Bug Detection & Annotation Extraction
- Related Work
- Conclusions

Overall Results On Linux

Source	Seed Annotation	Seed Checked	True Bugs	False Positives
Comment	226	119	7	2
Assertion	24	17	3	1
Total	**245**	**133**	**9**	**3**

- Annotations can help detect and avoid bugs.
- Comments and code complement each other for annotation extraction and bug detection.
- We propagate seed annotations to generate 96,821 annotations.

Annotation Extraction Results

Software	LOC	Sentence	IRQSent	HeuSent	Annotation
Linux	5.2M	1,024,624	23,662	423	226
FreeBSD	2.4M	420,013	11,117	80	43
NetBSD	3.3M	680,650	23,942	108	62
OpenSolaris	3.7M	535,073	8,074	71	24
Total	**14.6M**	**2,660,360**	**66,795**	**682**	**355**

- Reduce the # of annotations to be manually read from 66,795 to 682.
- The annotation generation accuracy is 90-100%.

Limitations & Future Work

- Automatically learn paraphrases, e.g., "disable" = "block"
  - Promising preliminary results [LinNLE'01, GlickmanRANLP'03, HillMSR'08]
- Consider different types of interrupts, different interrupt context, and conditional annotations
- Send annotations to developers for confirmation
  - To detect annotations extracted from outdated comments

Conclusions

- ✦ **Feasible to extract annotations from comments & code**
  - Generated 96,821 **interrupt-related** annotations & automatically detected **9** bugs in the Linux kernel
  - These annotations can help developers avoid bugs.
- ✦ **Combining comments & code help extract more annotations and detect more bugs than using comments or code alone.**
- Apply to non-OS code and for extracting other types of annotations