




# Static Analysis of Device Drivers: We Can Do Better!

**Sidney Amani** Leonid Ryzhyk Alastair Donaldson  
Gernot Heiser Alexander Legg Yanjin Zhu

Australian Government  
Department of Broadband, Communications  
and the Digital Economy  
Australian Research Council

NICTA leading and Supporting Members and Partners  
Australian University  
UNSW  
NSW  
Victoria  
University of Queensland  
University of Western Australia  
University of Technology Sydney  
University of New South Wales  
University of Melbourne  
University of Queensland  
University of Western Australia  
University of Technology Sydney  
University of New South Wales  
University of Melbourne




## Example

```
int suspend() {
    ...
    free(p);
}

void remove() {
    ...
    p->data = 0;
}
```

NICTA Copyright 2011 From Imagination to Impact




## Drivers are buggy

The driver reliability problem:

- 70% of OS code is in device drivers
- Drivers contain 3-7 times more bugs per loc than the rest of the kernel
- 70% of OS failures are caused by driver bugs

NICTA Copyright 2011 From Imagination to Impact




## Example

```
int suspend() {
    ...
    free(p);
}

void remove() {
    ...
    p->data = 0;
}
```

- Automatic verification:
  - Requires a model of the OS
  - `int os_main() {`
    - ...  
`suspend();`
    - ...  
`remove();`
    - }
  - Relies on pointer analysis (undecidable)


NICTA Copyright 2011 From Imagination to Impact



## Automatic verification of device drivers

- Modern model checkers successfully find many driver bugs
  - SLAM, SatAbs, Blast
- Verifying low-level driver code is tricky
- How can we make drivers easier to automatically verify?

NICTA Copyright 2011 From Imagination to Impact



## Example

```
int suspend() {
    ...
    free(p);
}

void remove() {
    ...
    p->data = 0;
}
```

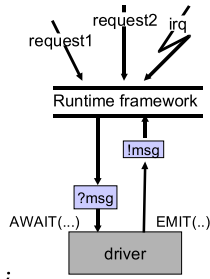
- What is the root cause of the bug?
  - The driver is not prepared to handle `remove` after `suspend`
  - This property cannot be checked directly as it is implicit in the driver code
  - Detecting indirect consequences is hard

NICTA Copyright 2011 From Imagination to Impact

## Solution: Active Device Drivers



- Active drivers:
  - Each driver has its own thread of control
  - Communication with the OS via message passing



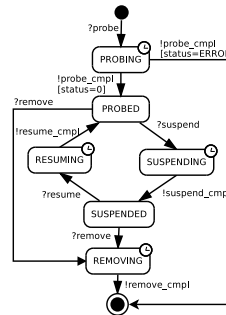
```
void driver_main() {
    ...
    EMIT(msg);
    ...
    AWAIT(msg1, ..., msgn);
}
```

NICTA Copyright 2011

From imagination to impact

7

## Active Drivers: Example



```
mb=AWAIT(suspend,remove);
if(mb == suspend){
    ...
    free(p);
    EMIT(suspend_cmpl);
    mb=AWAIT(resume);
    ...
} else if (mb == remove){
    p->data = 0;
    ...
}
```

NICTA Copyright 2011

From imagination to impact

10

## Solution: Active Device Drivers



- Active drivers:
  - Each driver has its own thread of control
  - Communication with the OS via message passing

- Advantages
  - Explicit control flow
  - No concurrency
  - Allows formal specification and verification of driver-OS protocols

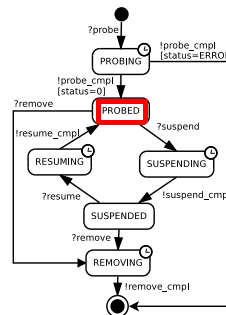
```
void driver_main() {
    ...
    EMIT(msg);
    ...
    AWAIT(msg1, ..., msgn);
}
```

NICTA Copyright 2011

From imagination to impact

8

## Active Drivers: Example



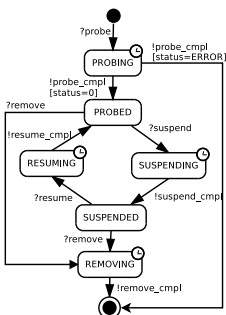
```
mb=AWAIT(suspend,remove);
if(mb == suspend){
    ...
    free(p);
    EMIT(suspend_cmpl);
    mb=AWAIT(resume);
    ...
} else if (mb == remove){
    p->data = 0;
    ...
}
```

NICTA Copyright 2011

From imagination to impact

11

## Active Drivers: Example

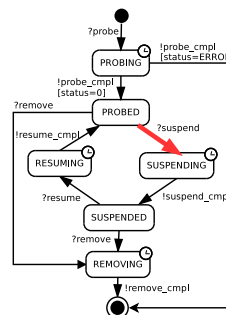


NICTA Copyright 2011

From imagination to impact

9

## Active Drivers: Example



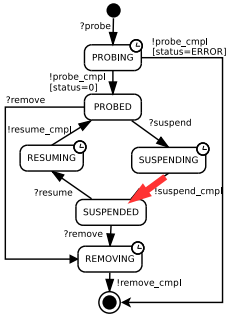
```
mb=AWAIT(suspend,remove);
if(mb == suspend){
    ...
    free(p);
    EMIT(suspend_cmpl);
    mb=AWAIT(resume);
    ...
} else if (mb == remove){
    p->data = 0;
    ...
}
```

NICTA Copyright 2011

From imagination to impact

12

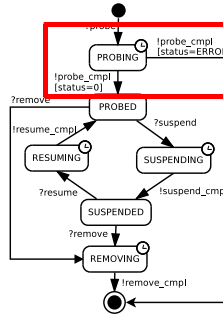
## Active Drivers: Example



```

mb=AWAIT (suspend,remove) ;
if (mb == suspend) {
    ...
    free (p) ;
    EMIT (suspend_cmpl) ;
    mb=AWAIT (resume) ;
    ...
} else if (mb == remove) {
    p->data = 0 ;
    ...
}
    
```

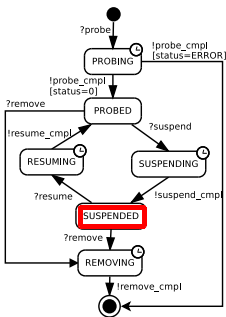
## Automatic verification



We define 3 rules for protocol compliance:

- EMIT: Allowed iff it triggers a valid state transition

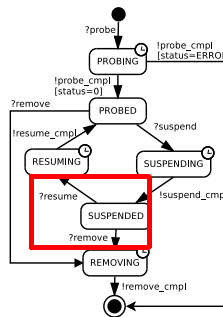
## Active Drivers: Example



```

mb=AWAIT (suspend,remove) ;
if (mb == suspend) {
    ...
    free (p) ;
    EMIT (suspend_cmpl) ;
    mb=AWAIT (resume) ;
    ...
} else if (mb == remove) {
    p->data = 0 ;
    ...
}
    
```

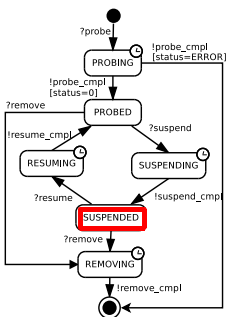
## Automatic verification



We define 3 rules for protocol compliance:

- EMIT: Allowed iff it triggers a valid state transition.
- AWAIT: must wait on all incoming messages enabled in the current state.

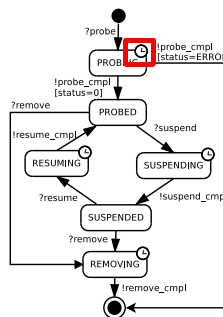
## Active Drivers: Example



```

mb=AWAIT (suspend,remove) ;
if (mb == suspend) {
    ...
    free (p) ;
    EMIT (suspend_cmpl) ;
    mb=AWAIT (resume,remove) ;
    ...
} else if (mb == remove) {
    p->data = 0 ;
    ...
}
    
```

## Automatic verification



We define 3 rules for protocol compliance:

- EMIT: Allowed iff it triggers a valid state transition.
- AWAIT: must wait on all incoming messages enabled in the current state.
- Timed: The driver must not remain in a timed state forever.

## Assertion based verification



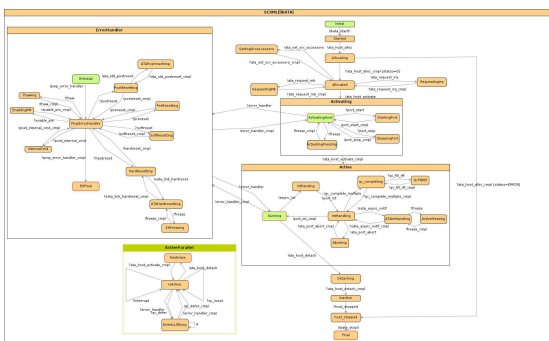
- We used SatAbs: a CEGAR model checker for C.
- EMIT and AWAIT rules can easily be encoded as C assertions.
- Timed states describe liveness properties, they can not be verified using assertions. We are working on this issue.
- We had to tune SatAbs' internal heuristics.
- We managed to verify a complex protocol.

## Conclusion



- Current driver architectures make drivers difficult to write
- It also makes them hard to verify using model checkers
- Active drivers help model checkers to find bugs

## LibATA protocol



## Performance



The active driver for AHCI SATA controllers achieves the same I/O throughput with slightly more CPU utilisation.

