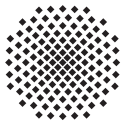
Tuning Static Data Race Analysis for Automotive Control Software

Steffen Keul

Dept. of Programming Languages and Compilers
Institute of Software Technology
University of Stuttgart

11th International Working Conference on Source Code Analysis and Manipulation



University of Stuttgart
Germany

Motivation

- Research Cooperation with the Department of Software Development for Passive Safety Systems at Robert Bosch GmbH, Germany
- Safety-critical Software, absence of Race Conditions should be demonstrated
- Evaluation of the Bauhaus static data race detector

Definition (Data Race)

A data race occurs if two threads access a common storage location without ordering constraints, and one of the accesses modifies the storage contents.

Example: Real-World Data Race

```
int control()
{
  ...
  if (sensor_valid &&
      sensor >= min_threshold &&
      sensor <= max_threshold)
  {
    control_car(sensor);
    ...
  }
}
```

Example: Real-World Data Race

```
int sensor;

int control()
{
  ...
  if (sensor_valid &&
      sensor >= min_threshold &&
      sensor <= max_threshold)
  {
    control_car(sensor);
    ...
  }
}
```

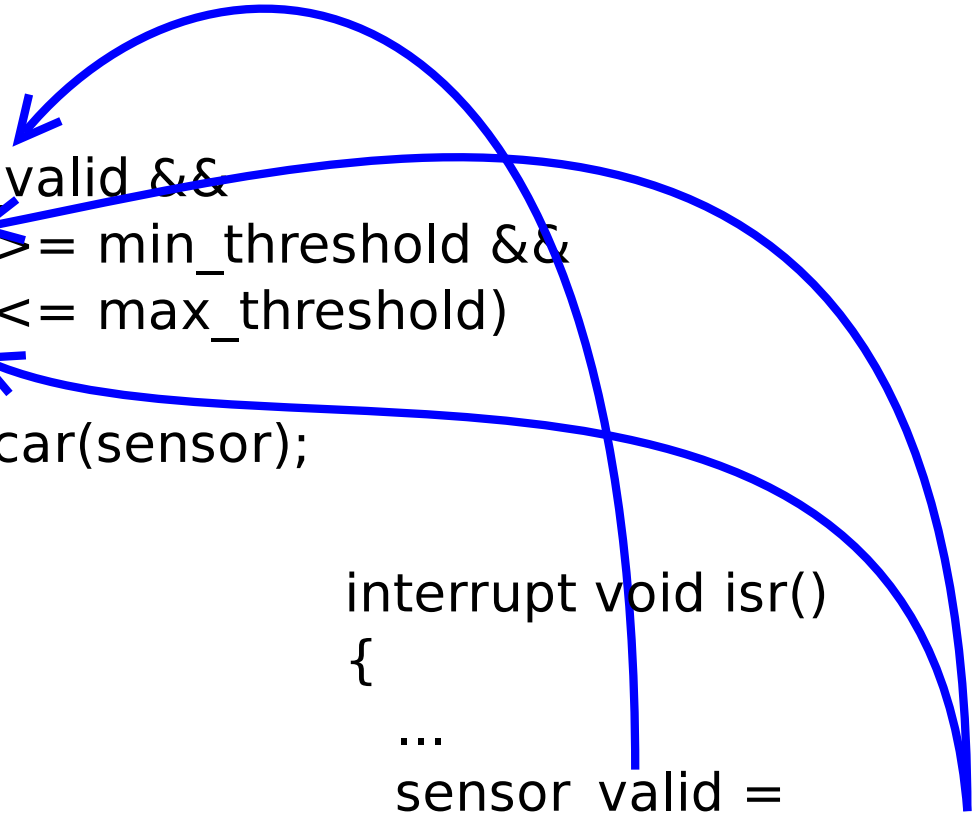
```
interrupt void isr()
{
  ...
  sensor_valid =
    read_port(0x23, &sensor);
}
```

Example: Real-World Data Race

```
int sensor;

int control()
{
  ...
  if (sensor_valid &&
      sensor >= min_threshold &&
      sensor <= max_threshold)
  {
    control_car(sensor);
    ...
  }
}
```

```
interrupt void isr()
{
  ...
  sensor_valid =
    read_port(0x23, &sensor);
}
```

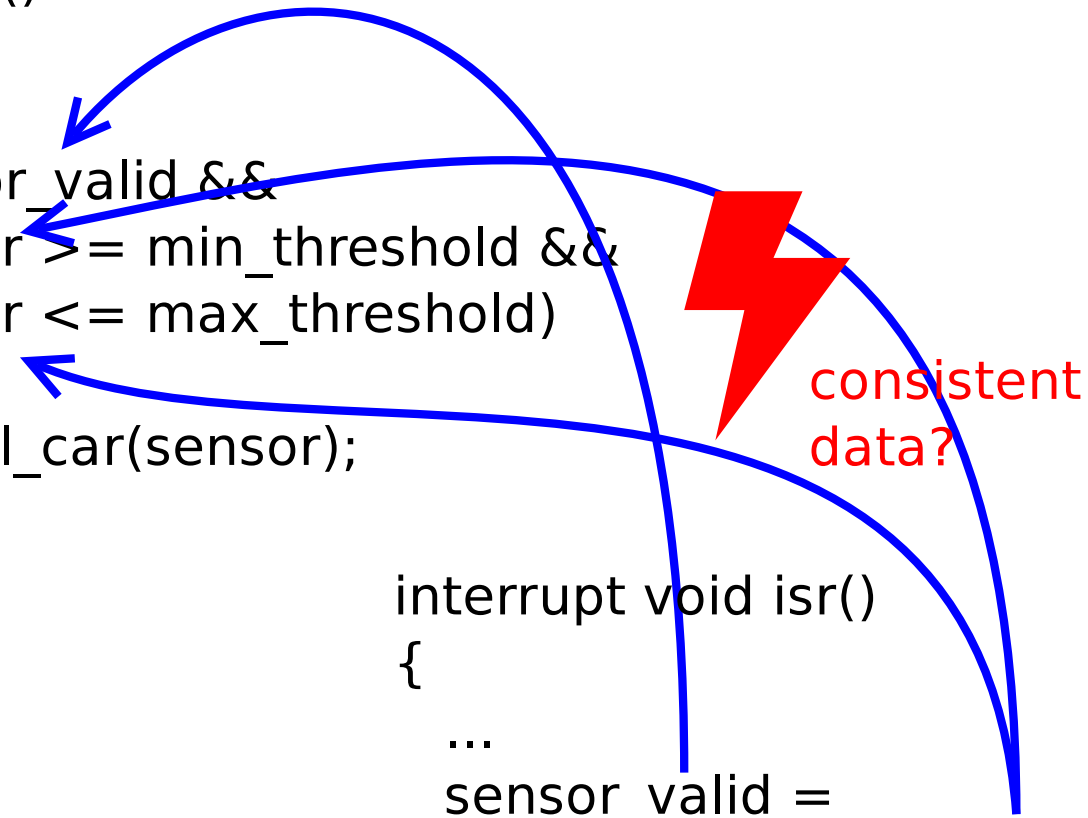


Example: Real-World Data Race

```
int sensor;

int control()
{
  ...
  if (sensor_valid &&
      sensor >= min_threshold &&
      sensor <= max_threshold)
  {
    control_car(sensor);
    ...
  }
}
```

```
interrupt void isr()
{
  ...
  sensor_valid =
    read_port(0x23, &sensor);
}
```



Reference System

- Control loop implementation
- Development version, roughly 1,750 functions in size
- Structure: 3 layers
  - Background operations implemented in periodically scheduled tasks: e.g. hardware checks, data logging, ...
  - Interrupt handlers used to respond to real-time events
  - High-priority timer interrupt used to compute and output control factors
- Communication through shared memory
- Currently single-core CPUs are used
- Disabling of interrupts used to achieve synchronization

Project Outline

1. Adapt static lockset analysis tool to application specific requirements
   - support interrupt disabling
   - support thread priorities on single-core CPUs
2. Apply data race detector to embedded software
3. Evaluate a sample of 70 warnings manually
4. Identify sources of imprecision in the sample, design and implement 3 extensions to the basic data race detector
5. Run the extended analysis tool
6. Measure improvement on whole system compared to initial results

Escape Analysis, example

```
void f(int *a, int *b)
{   ++(*a);
    ++(*b);
}

int global;

void isr()
{   int isr_local = 0;
    f(&global, &isr_local);
}

void fiq()
{   int fiq_local = 0;
    f(&global, &fiq_local);
}
```

Escape Analysis, example

```
void f(int *a, int *b)
{   ++(*a);   /* pt: global */ ⇒ Race!
    ++(*b);   /* pt: isr_local, fiq_local */ ⇒ Race!
}

int global;

void isr()
{   int isr_local = 0;
    f(&global, &isr_local);
}

void fiq()
{   int fiq_local = 0;
    f(&global, &fiq_local);
}
```

Escape Analysis, example

```
void f(int *a, int *b) /* *a, *b do not escape*/
{   ++(*a);  /*pt: global*/ ⇒ Race!
    ++(*b);  /*pt: isr_local, fiq_local: Thread local*/
}

int global;

void isr()
{   int isr_local = 0;
    f(&global, &isr_local); /*global escapes*/
}                           /*isr_local does not escape*/

void fiq()
{   int fiq_local = 0;
    f(&global, &fiq_local); /*global escapes*/
}                           /*fiq_local does not escape*/
```

Simple Path Exclusions, example

```
enum { Initial, Normal, Exceptional} State;

void ISR1(void)
{   if (State == Initial)
    {   A;
        State = Normal;
}   }

void ISR2(void)
{   if (State == Normal)
    {   B;
        State = Exceptional;
}   }

int main(void)
{   State = Initial;
    StartOS();
}
```

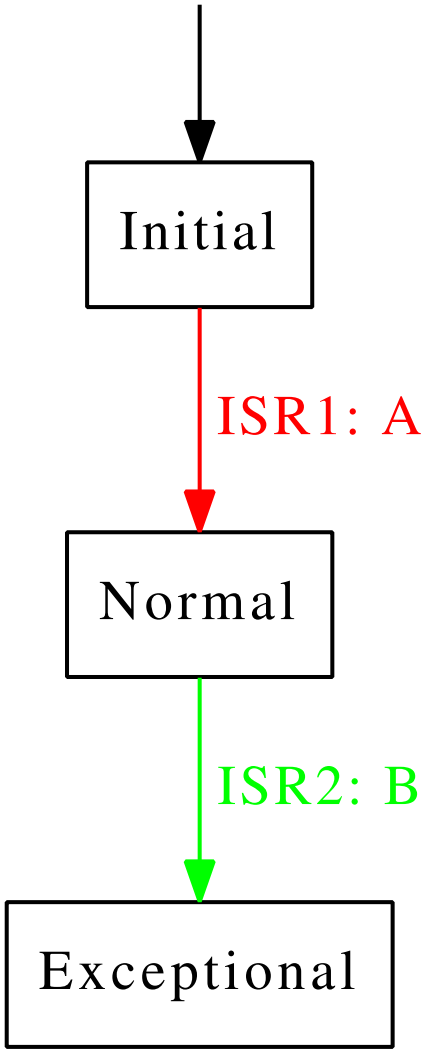
Simple Path Exclusions, example

```
enum { Initial, Normal, Exceptional} State;

void ISR1(void)
{   if (State == Initial)
    {   A;
        State = Normal;
}   }

void ISR2(void)
{   if (State == Normal)
    {   B;
        State = Exceptional;
}   }

int main(void)
{   State = Initial;
    StartOS();
}
```



Call Graph Computation, example

```
int G_isr = 0, G_fiq = 0;

void act_ALL(void (*helper_fpr)(void))
{   ...
    (*helper_fpr)();
}

void helper_ISR(void)
{ G_isr++; }
void isr()
{ act_ALL(helper_ISR); }

void helper_FIQ(void)
{ G_fiq++; }
void fiq()
{ act_ALL(helper_FIQ); }
```

Call Graph Computation, example

```
int G_isr = 0, G_fiq = 0;

void act_ALL(void (*helper_fpr)(void))
{   ...                    /*Union of all caller contexts*/
    (*helper_fpr)();       /*isr, fiq: helper_ISR, helper_FIQ*/
}

void helper_ISR(void)
{ G_isr++; }                    /* called by isr, fiq */
void isr()
{ act_ALL(helper_ISR); }   /* in thread isr: helper_ISR */

void helper_FIQ(void)
{ G_fiq++; }                    /* called by isr, fiq */
void fiq()
{ act_ALL(helper_FIQ); }   /* in thread fiq: helper_FIQ */
```

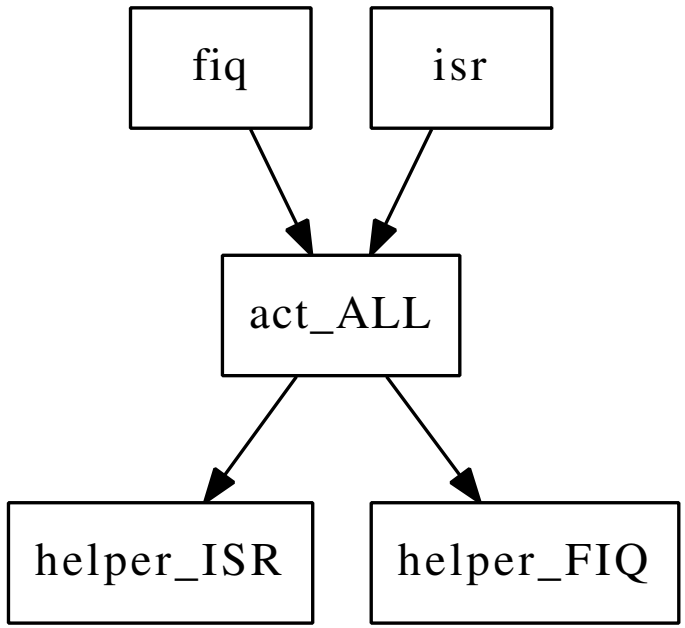
Call Graph Computation, example

```
int G_isr = 0, G_fiq = 0;

void act_ALL(void (*helper_fpr)(void))
{   ...
    (*helper_fpr)();
}

void helper_ISR(void)
{ G_isr++; }
void isr()
{ act_ALL(helper_ISR); }

void helper_FIQ(void)
{ G_fiq++; }
void fiq()
{ act_ALL(helper_FIQ); }
```



Call Graph Computation, example

```
int G_isr = 0, G_fiq = 0;

void act_ALL(void (*helper_fpr)(void))
{   ...                          /*distinguish contexts*/
    (*helper_fpr)();             /* thread isr: helper_ISR */
}                                /* thread fiq: helper_FIQ */

void helper_ISR(void)
{ G_isr++; }                     /* called by isr */
void isr()
{ act_ALL(helper_ISR); }         /* in thread isr: helper_ISR */

void helper_FIQ(void)
{ G_fiq++; }                     /* called by fiq */
void fiq()
{ act_ALL(helper_FIQ); }         /* in thread fiq: helper_FIQ */
```

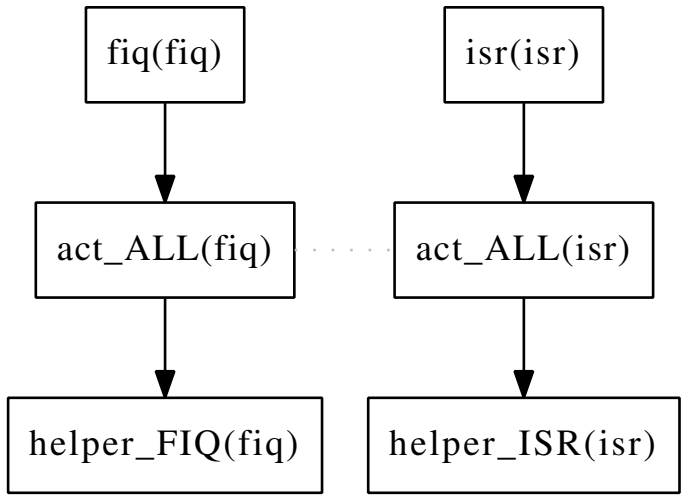
Call Graph Computation, example

```
int G_isr = 0, G_fiq = 0;

void act_ALL(void (*helper_fpr)(void))
{   ...
    (*helper_fpr)();
}

void helper_ISR(void)
{ G_isr++; }
void isr()
{ act_ALL(helper_ISR); }

void helper_FIQ(void)
{ G_fiq++; }
void fiq()
{ act_ALL(helper_FIQ); }
```



Measurements

extension			objects		fields	
simple path exclusion	c-s call graph	escape	warnings	reduction	warnings	reduction
			310	0%	366	0%
✓			302	3%	358	2%
	✓		232	25%	285	22%
		✓	207	33%	263	28%
✓	✓	✓	180	42%	233	36%

Conclusion

- Found true positive warnings, identified bugs in an early development version
- Designed and implemented three extensions to the basic algorithm
- Achieved significant improvement of precision
- Proposed DFA-based implementation pattern that is easy to analyze
- Created infrastructure for further research on embedded control software