Model Checking Race-Freeness
MEMICS’09

Frédéric Haziza <daz@it.uu.se>

Department of Computer Systems
Uppsala University

Friday, November 13th, 2009
Race Condition

Situation in which one process changes a variable which another process has previously read and the other process does not get notified of the change.
Race Condition

Situation in which one process changes a variable which another process has previously read and the other process does not get notified of the change.

Classic

Write

Read

Write
Race Condition

Situation in which one process changes a variable which another process has previously read and the other process does not get notified of the change.

**Classic**

- Write
- Read
- Write

**Check-then-act**

- \( \text{if (check)} \{
- \quad \text{act}
- \}

i.e. getting the right answer relies on lucky timing.
Race Condition

Situation in which one process changes a variable which another process has previously read and the other process does not get notified of the change.

**Classic**

- Write
- Read
- Write

**Check-then-act**

```
if (check){
    act
}
Update
```
Race Condition

Situation in which one process changes a variable which another process has previously read and the other process does not get notified of the change.

Classic

Write

Read

Write

Check-then-act

if (check){

act

} Update

i.e. getting the right answer relies on lucky timing.
Related Work

On-the-fly
- dynamic tools

Ahead-of-time
- static analysis
- compile-time heuristics

Post-mortem
Combination of static and dynamic techniques
Related Work

- Feasable races vs. False positives
- Coverage
- Expressiveness
- Annotations

On-the-fly
- dynamic tools

Ahead-of-time
- static analysis
- compile-time heuristics

Post-mortem
Combination of static and dynamic techniques
Outline

- Pthreaded C code
- Race Detection
Outline

- Pthreaded C code
- Extraction
- Model
- Race Detection

MEMICS'09 Dept. of Information Technology - 4 - Frédéric Haziza | daz@it.uu.se
Race Detection

Pthreaded C code

Extraction

Model

Method

Extraction

Conclusion

Outline

Hard: limited reproducability, non-determinism
Coverage vs False positives

MEMICS'09  Dept. of Information Technology  - 4 -  Frédéric Haziza | daz@it.uu.se
Outline

- Pthreaded C code
- Extraction
- Model
- Method
  - Symbolic Backward Reachability Analysis
- Race Detection
  - Hard: limited reproducability, non-determinism
  - Coverage vs False positives
Race Detection

- Hard: limited reproducability, non-determinism
- Coverage vs False positives

Method

Symbolic Backward Reachability Analysis

Extraction

PETRI Net

Model

Pthreaded C code

Outline
Race Detection

Hard: limited reproducability, non-determinism
Coverage vs False positives

Extraction

CIL: Parsing → AST
Pointer Analysis

Pthreaded C code

Method

Symbolic Backward Reachability Analysis

Model

Petri Net

Conclusion

MEMICS'09 Dept. of Information Technology - 4 - Frédéric Haziza | daz@it.uu.se
Race Detection

Method
Symbolic Backward Reachability Analysis

Extraction
CIL: Parsing → AST
Pointer Analysis

Model
Petri Net

Conclusion
Hard: limited reproducibility, non-determinism
Coverage vs False positives

Pthreaded C code
No recursion
No complex pointer arithmetics
No dynamic heap management
Language

Programs written in C

using POSIX threads
Language

Programs written in C using POSIX threads

- discard CPU operations (add, sub, ...)
Language

- Programs written in C using POSIX threads

- discard CPU operations (add, sub, · · ·)

- mover operations (load, store, · · ·)

- thread synchronization and bookkeeping
Language

Thread 1

\[ L_1 \]
\[ L_2 \]
\[ \vdots \]

Thread 2

\[ L_3 \]
\[ L_4 \]
\[ \vdots \]

Thread 3

Thread 4
Language

\[X, Y, Z, \ldots\]

\[L_1\]
\[L_2\]
\[
\]
Thread 1

\[L_3\]
\[L_4\]
\[
\]
Thread 2

\[\ldots\]
\[\ldots\]
\[
\]
Thread 3

\[\ldots\]
\[\ldots\]
\[
\]
Thread 4
Language

Control statements

- if
- if-then-else
- while
- for-loop

↔ branch and label
Language

Control statements
- if
- if-then-else
- while
- for-loop

\[ \text{branch and label} \]

Movers
- Read X
- Write X
Language

Control statements
- if
- if-then-else
- while
- for-loop

↔ branch and label

Movers
- Read X
- Write X

Lock – mutex
- acquire
- release
Language

Control statements
- if
- if-then-else
- while
- for-loop

→ branch and label

Movers
- Read X
- Write X

Lock – mutex
- acquire
- release

Condition Variable
- wait
- signal
Language

X, Y, Z, ..., M₁, M₂, ..., CV₁, CV₂, ...

L₁
L₂
...

Thread 1

L₃
L₄
...

Thread 2

...

Thread 3

...

Thread 4
Language – examples

```c
int counter;
pthread_mutex_t L;

pthread_mutex_lock(L);

counter++;

pthread_mutex_unlock(L);
```
Language – examples

```c
int counter;
pthread_mutex_t L;

pthread_mutex_lock(L);
counter++;
 pthread_mutex_unlock(L);

shared counter;
shared L;

acquire L;
read counter;
write counter;
release L;
```
Model – Petri Nets

**Model**

- **Method**
  - **Extraction**
  - **Conclusion**

---

**MEMICS’09**

Dept. of Information Technology

- 10 -

Frédéric Haziza | daz@it.uu.se
Reading and Writing a shared variable
Acquiring and releasing a lock

Model

Method

Extraction

Conclusion

MEMICS'09
Dept. of Information Technology

Frédéric Haziza | daz@it.uu.se
Waiting on a condition variable
Waiting on a condition variable

Model
Method
Extraction
Conclusion
Waiting on a condition variable
Waiting on a condition variable
Waiting on a condition variable

Model
Method
Extraction
Conclusion
Waiting on a condition variable

Model

Method

Extraction

Conclusion

MEMICS'09

Dept. of Information Technology

Frédéric Haziza | daz@it.uu.se
Waiting on a condition variable
Signaling a condition variable
Signaling a condition variable

Model
Method
Extraction
Conclusion

Condition Variable

cond signal

tr

ready

in ts

out

conflict
Branching and Jumps

Model
Method
Extraction
Conclusion

MEMICS’09 Dept. of Information Technology - 15 - Frédéric Haziza | daz@it.uu.se
Creating a new thread
Outline

- Pthreaded C code
- Race Detection
- Petri Net
- Model
Outline

Model

Method

Petri Net

Race Detection

Pthreaded C code

Model

Method

Symbolic Backward Reachability Analysis

MEMICS'09 Dept. of Information Technology - 17 - Frédéric Haziza | daz@it.uu.se
Method

Safety Property

MEMICS'09  Dept. of Information Technology - 18 - Frédéric Haziza | daz@it.uu.se
Method

Safety Property

Reachability Analysis
Method

Safety Property

Reachability Analysis

Can a set of bad states be reachable from the initial states?
Data race in our model

- Read\_v
- Write\_v
- Read\_v
- Write\_v

and
Interesting properties

A bad configuration:

\[ \text{Write}_v \]
Interesting properties

A bad configuration:

\[ \text{Write}_v \]
Interesting properties

A bad configuration:

$\text{Write}_v$
Interesting properties

A bad configuration:

\[ Write_v \]

Bad state:

\[ Write_v \]
Ordering
Verification Method: Reachability
Verification Method: Reachability
Verification Method: Reachability
Verification Method: Reachability

Safety property
Verification Method: Reachability

- Model
- Method
- Extraction
- Conclusion
Verification Method: Reachability

Init: 

Reachable?

Safety property
Verification Method: Reachability

Reachable?

Safety property

Init:
Verification Method: Reachability

Model
Method
Extraction
Conclusion

Init:

Safety property

Backward Reachable?

 Verification Method: Reachability
Bad states

- Make sure we can go backwards
- Prove termination
Is $\text{Pre}(\cdot)$ an upward-closed set?
Is Pre(\text{?}) an upward-closed set?

Is that enough to take to pre of the generator?
Monotonicity
Monotonicity

Model

Method

Extraction

Conclusion
Monotonicity
Monotonicity
Answer!

Yes!
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis
Backward reachability analysis

Pre*( )
Backward reachability analysis

\[ \text{Pre}^*( ) \]

\[ \text{init}_a \]

\[ \text{init}_b \]

\[ \text{init}_c \]
Termination and Correctness

- WQO
- Over-approximation
- Termination
- Monotonicity
- Pre* is computable
- Correctness
Termination and Correctness

- WQO
- Over-approximation
- Termination
- Monotonicity
- Pre* is computable
- Correctness
Outline

**Pthreaded C code**

**Race Detection**

**Model**

- Petri Net
- Symbolic Backward Reachability Analysis

**Method**
Outline

Model
Method
Extraction
Race Detection

Pthreaded C code

Extraction

Model
Petri Net
Symbolic Backward Reachability Analysis

Method
Extraction

C code → AST → Petri-Net

Optimizations
Transformations
Collecting information
Pointers, Arrays
Recursive calls
Pointers
Callgraph
Simplifications
Extraction

Model
Method
Extraction
Conclusion

C code \rightarrow CIL \rightarrow AST \rightarrow Petri-Net

Optimizations
Transformations
Collecting information
Pointers, Arrays
Recursive calls
Pointers
Callgraph
Simplifications
Extraction

C code → AST → Petri-Net

Collecting information:
- Pointers, Arrays
- Recursive calls

Optimizations
Transformations

Model
Method
Extraction
Conclusion

MEMICS’09
Dept. of Information Technology

Frédéric Haziza | daz@it.uu.se
Extraction

- C code
- AST
- Petri-Net

CIL
Optimizations
Transformations
Collecting information
Extraction

C code → CIL → AST

Optimizations → Transformations → Collecting information

Petri-Net

Pointers, Arrays
Recursive calls
Extraction

C code → CIL → AST

Pointers, Arrays
Recursive calls

Pointers
Callgraph
Simplifications

Optimizations
Transformations
Collecting information

Petri-Net
Some transformations

\[ x = y \ op \ z \]
Some transformations

\[ a = s.\text{field} \]
Some transformations

\[ a = s.body \]

\[ \text{tmp} = \&s + \text{shift}; \]
\[ a = \*\text{tmp} \]
Some transformations

\[ a = \text{arr}[i] \]
Some transformations

\[ a = \text{arr}[i] \]

\[ \downarrow \]

\[ \text{tmp} = \&\text{arr} + i * \text{cell\_size}; \]
\[ a = *\text{tmp} \]
Pointer Analysis → Limitations

- No recursion
- No complex pointer arithmetics
- No dynamic heap management
Conclusion

Race Detection

Method

Extraction

Model

Pthreaded C code

Petri Net

Backward Reachability Analysis

MEMICS'09 Dept. of Information Technology - 33 - Frédéric Haziza | daz@it.uu.se
Conclusion

- Pthreaded C code
- Extraction
- Petri Net
- Model
- Method
- Backward Reachability Analysis
- Race Detection
- Analyze the race conditions themselves
- No annotations

MEMICS'09
Dept. of Information Technology
Frédéric Haziza | daz@it.uu.se
Conclusion

Race Detection
- Analyze the race conditions themselves
- No annotations

Extraction

Model
- Petri Net

Method
- Backward Reachability Analysis
  - Simple and efficient
  - Powerful symbolic representation

Pthreaded C code
Conclusion

Race Detection

- Analyze the race conditions themselves
- No annotations

Limitations

Extraction

- Pthreaded C code

Method

- Backward Reachability Analysis
- Simple and efficient
- Powerful symbolic representation

Model

- Petri Net

Conclusion

MEMICS'09  Dept. of Information Technology

Frédéric Haziza  daz@it.uu.se
To Do

- More pthreads constructs and maybe other libraries
- Better approximation to handle limitations
- Usefulness of the method:
  - False positives and coverage
  - Compare with other tools
- Better Error reporting when a race is found