

Static analysis of the determinism of multithreaded programs

Pietro Ferrara

Ecole Polytechnique
Paris, France

Università Ca' Foscari
Venice, Italy

SEFM 2008, Cape Town, South Africa


Motivations

- Multicore revolution
- Applications with explicit parallelism
 - Multithreading
- Difficult to reason about them
 - Random interleaving
 - Memory model
- Debug
 - Testing
 - Not all the executions may be exposed
 - Hard to reproduce an execution
 - Static analysis

An example

Thread Deposit1

Variable	Value
tempt1	
...	


 `int tempt1=a.amount;`
`tempt1=tempt1+1000$;`
`a.amount=tempt1;`

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

 `int tempt2=a.amount;`
`tempt2=tempt2+1000$;`
`a.amount=tempt2;`

An example

Thread Deposit1

Variable	Value
tempt1	10.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

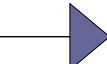
```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

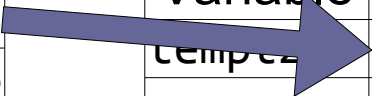


Shared Memory

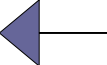
Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	11.000\$
...	



```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

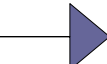


An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```



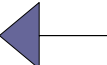
Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	12.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```



An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	12.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	12.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```


At the end:

`a.amount=12.000$`

An example

Thread Deposit1

Variable	Value
tempt1	
...	


 `int tempt1=a.amount;`
`tempt1=tempt1+1000$;`
`a.amount=tempt1;`

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

 `int tempt2=a.amount;`
`tempt2=tempt2+1000$;`
`a.amount=tempt2;`

An example

Thread Deposit1

Variable	Value
tempt1	10.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	10.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	10.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	10.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	10.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	10.000\$
...	

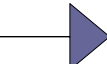
```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```



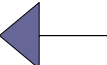
Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	11.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```



An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	11.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;  
tempt1=tempt1+1000$;  
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	11.000\$
...	

```
int tempt2=a.amount;  
tempt2=tempt2+1000$;  
a.amount=tempt2;
```

At the end:

`a.amount=11.000$`

An example

Thread Deposit1

Variable	Value
tempt1	11.000\$
...	

```
int tempt1=a.amount;
```

```
tempt1=tempt1+1000$;
```

```
a.amount=tempt1;
```

Shared Memory

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Thread Deposit2

Variable	Value
tempt2	11.000\$
...	

```
int tempt2=a.amount;
```

```
tempt2=tempt2+1000$;
```

```
a.amount=tempt2;
```

1st execution: a.amount=12.000\$

2nd execution: a.amount=11.000\$

Our solution

- **Statically** analyze the determinism
 - Focused on communications on shared memory
 - Via abstract interpretation
 - **Generic** w.r.t.
 - Programming language
 - Numerical domain
 - Memory model
- Advantages
 - Deal **directly** with the effects of random interleaving
 - **Flexible**

Concrete domain

$$S : [\text{Var} \rightarrow (V \times T)]$$

- S: shared memories
- Var: variables
- V: values
- T: thread identifiers
- Thread-partitioned trace domain
 - Relates each thread to its trace of execution

$$\Psi : [T \rightarrow S^{\vec{+}}]$$

- Concrete semantics: set of functions
 - All the possible executions

An example – Concrete semantics

Thread Deposit1:

Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1

Thread Deposit2:

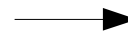
Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1



Obj.	Field	Value	Thread
a	amount	12.000\$	Deposit2

Thread Deposit1:

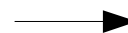
Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1

Thread Deposit2:

Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit2

First abstraction

$$S^\# : [\text{Var} \rightarrow [T \rightarrow V^\#]]$$

- Parameterized on an abstract domain $V^\#$
 - One value for each thread

$$\Psi^\# : [T \rightarrow S^{\#\vec{\tau}}]$$

- Abstract semantics: one function
 - Approximates all the possible executions
- **Sound** w.r.t. concrete domain

$$\langle \wp(\Psi), \sqsubseteq \rangle \begin{array}{c} \xleftarrow{\gamma_\Psi} \\ \xrightarrow{\alpha_\Psi} \end{array} \langle \Psi^\#, \sqsubseteq_{\Psi^\#} \rangle$$

An example – 1st abstraction

Thread Deposit1:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit1	[11.000..11.000]\$

Thread Deposit2:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$
		Deposit1	[11.000..11.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit2	[11.000..12.000]\$

Second abstraction

$$S^\circ : [\text{Var} \rightarrow (V^\# \times \wp(T))]$$

- Trace
 - One abstract value
 - The set of threads that may have written it

$$\Psi^\circ : [T \rightarrow S^{\circ\vec{+}}]$$

- Sound

$$\langle \wp(\Psi), \sqsubseteq \rangle \begin{array}{c} \xleftarrow{\gamma_\Psi} \\ \xrightarrow{\alpha_\Psi} \end{array} \langle \Psi^\#, \sqsubseteq_{\Psi^\#} \rangle \begin{array}{c} \xleftarrow{\gamma_{\Psi^\#}} \\ \xrightarrow{\alpha_{\Psi^\#}} \end{array} \langle \Psi^\circ, \sqsubseteq_{\Psi^\circ} \rangle$$

An example – 2nd abstraction

Thread Deposit1:

Obj.	Field	Value	Threads
a	amount	[10.000..10.000]\$	{System}

 →

Obj.	Field	Value	Threads
a	amount	[11.000..11.000]\$	{Deposit1}

Thread Deposit2:

Obj.	Field	Value	Threads
a	amount	[10.000..11.000]\$	{System, Deposit1}

 →

Obj.	Field	Value	Threads
a	amount	[11.000..12.000]\$	{Deposit2}

Determinism on concrete states

$$DS(s_1, s_2) = \text{false}$$



$$\exists \text{var} \in \text{dom}(s_1) \cap \text{dom}(s_2) : s_1(\text{var}) = (\text{val}_1, t_1), \\ s_2(\text{var}) = (\text{val}_2, t_2), t_1 \neq t_2$$

- A program is not deterministic iff
 - two executions
 - of the **same thread**
 - in the **same position** of the traces of execution
 - contains two shared memories that relate
 - the **same variable**
 - to **values written by different threads**

An example – Concrete semantics

Thread Deposit1:

Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1

Thread Deposit2:

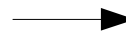
Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1



Obj.	Field	Value	Thread
a	amount	12.000\$	Deposit2

Thread Deposit1:

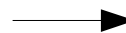
Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit1

Thread Deposit2:

Obj.	Field	Value	Thread
a	amount	10.000\$	System



Obj.	Field	Value	Thread
a	amount	11.000\$	Deposit2

Determinism on abstract states

- First abstraction

$$DS^\#(s^\#) = \text{false}$$



$$\exists \text{var} \in \text{dom}(s^\#) : |\text{dom}(s^\#(\text{var}))| > 1$$

- Second abstraction

$$DS^\circ(s^\circ) = \text{false} \Leftrightarrow \exists \text{var} \in \text{dom}(s^\circ) : |\pi_2(s^\circ(\text{var}))| > 1$$

- Soundness

$$\forall \theta \in \wp(\Psi) : D(\theta) = \text{false} \Rightarrow D^\#(\alpha_\Psi(\theta)) = \text{false}$$

$$\forall f^\# \in \Psi^\# : D^\#(f^\#) = \text{false} \Rightarrow D^\circ(\alpha_{\Psi^\#}(f^\#)) = \text{false}$$

An example – 1st abstraction

Thread Deposit1:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit1	[11.000..11.000]\$

Thread Deposit2:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$
		Deposit1	[11.000..11.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit2	[11.000..12.000]\$

An example – 2nd abstraction

Thread Deposit1:

Obj.	Field	Value	Threads
a	amount	[10.000..10.000]\$	{System}

 →

Obj.	Field	Value	Threads
a	amount	[11.000..11.000]\$	{Deposit1}

Thread Deposit2:

Obj.	Field	Value	Threads
a	amount	[10.000..11.000]\$	{System, Deposit1}

 →

Obj.	Field	Value	Threads
a	amount	[11.000..12.000]\$	{Deposit2}

Weak determinism

$$ADS^\# : [S^\# \rightarrow \{\text{true}, \text{false}\}]$$
$$ADS^\# (s^\#) = \text{false}$$

$$\exists \text{var} \in \text{dom}(s^\#) : |\text{dom}(s^\#(\text{var}))| > 1$$
$$\wedge \exists t_1, t_2 \in \text{dom}(s^\#(\text{var})) : s^\#(\text{var})(t_1) \neq s^\#(\text{var})(t_2)$$

- **Relax** the full determinism
- On the first level of abstraction
- Rely on a **numerical abstract domain**

An example – 1st abstraction

Thread Deposit1:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit1	[11.000..11.000]\$

Thread Deposit2:

Obj.	Field	Thread	Value
a	amount	System	[10.000..10.000]\$
		Deposit1	[11.000..11.000]\$



Obj.	Field	Thread	Value
a	amount	Deposit2	[11.000..12.000]\$

An example – 1st abstraction

Thread Deposit1:

Obj.	Field	Thread	Value
a	amount	System	+



Obj.	Field	Thread	Value
a	amount	Deposit1	+

Thread Deposit2:

Obj.	Field	Thread	Value
a	amount	System	+
		Deposit1	+



Obj.	Field	Thread	Value
a	amount	Deposit2	+

Projecting traces and states

- Check the determinism only
 - On a **subset of the shared variables**, e.g.
 - Only the amount of the bank account
 - On a **subset of the trace**
 - Only the actions that deposit and withdraw money
- Can be **composed**, e.g. check the determinism
 - On a . amount
 - When withdrawing
 - If the values written are always positive

Semi-automatic parallelization

- Given a sequential program
 - **Divide** it in two partitions - Input of the developer
 - **Analyze** them as executed in parallel
 - **Check** if the determinism is respected
 - Or one of its relaxations

Semi-automatic parallelization

- Given a sequential program
 - **Divide** it in two partitions - Input of the developer
 - **Analyze** them as executed in parallel
 - **Check** if the determinism is respected
 - Or one of its relaxations

Thread 1 → `acc.deposit(1.000$);`
`acc.deposit(1.000$);` ← Thread 2

Non-determinism on acc.amount

Semi-automatic parallelization

- Given a sequential program
 - **Divide** it in two partitions - Input of the developer
 - **Analyze** them as executed in parallel
 - **Check** if the determinism is respected
 - Or one of its relaxations

Thread 1 → `acc.deposit(1.000$);`
`acc.printAmount();` ← Thread 2



Non-determinism on stdout

Orthogonal issues

- We focused on the property
 - Approximate all the executions w.r.t. memory model

P. Ferrara. “*Static analysis via abstract interpretation of the happens-before memory model*”. In Springer, editor, Proceedings of TAP '08, LNCS, 2008.

- Apply it to a real programming language, e.g. Java
 - Threads are objects, i.e. identified by reference
 - Shared memory is the heap, i.e. accessed by reference

P. Ferrara. “*A fast and precise analysis for data race detection*”. In Elsevier, editor, Proceedings of Bytecode '08, volume ENTCS, 2008.

Experimental results

Program	# St.	# Th.	An.	Det.	Weak Det.
philo	213	2	< 1''	< 1''	< 1''
tsp	1936	2	24''	< 1''	< 1''
elevator	1829	2	15''	< 1''	< 1''
barrier	363	3	< 1''	< 1''	< 1''
forkjoin	170	2	< 1''	< 1''	< 1''
sync	320	3	< 1''	< 1''	< 1''
sor	1121	2	4''	< 1''	< 1''
crypt	2636	3	4''	< 1''	< 1''
lufact	3732	2	26''	< 1''	< 1''
montecarlo	3864	2	46''	4''	4''
molodyn	9029	2	13'51''	50''	1'01''

Conclusion

- We present a novel deterministic property
 - Generic
 - Flexible
- Concrete and two levels of abstract semantics
 - Sound
- Many ways of **relaxing** it
 - On the information → Weak determinism
 - On the states
 - On the traces
- First step to semi-automatic **parallelization**
- **Applied** to a set of well-known benchmarks

Question time

Thank you!

“Wrong” program with data race

Thread Deposit1

```
int tempt1=a.amount;  
  
tempt1=tempt1+1000$;  
  
a.amount=tempt1;
```

Shared Memory

Thread Deposit2

```
int tempt2=a.amount;  
  
tempt2=tempt2+1000$;  
  
a.amount=tempt2;
```

Variable	Value
tempt1	11.000\$
...	

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	12.000\$
...	

Variable	Value
tempt1	11.000\$
...	

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	11.000\$
...	

“Correct” program without data race

Thread Deposit1

```
lock(a);  
int tempt1=a.amount;  
  
tempt1=tempt1+1000$;  
  
a.amount=tempt1;  
unlock(a);
```

Shared Memory

Thread Deposit2

```
lock(a);  
int tempt2=a.amount;  
  
tempt2=tempt2+1000$;  
  
a.amount=tempt2;  
unlock(a);
```

Variable	Value
tempt1	11.000\$
...	

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	12.000\$
...	

Variable	Value
tempt1	12.000\$
...	

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	11.000\$
...	

“Wrong” program without data race

Thread Deposit1

```
lock(a);  
int tempt1=a.amount;  
unlock(a);  
tempt1=tempt1+1000$;  
lock(a);  
a.amount=tempt1;  
unlock(a);
```

Shared Memory

Thread Deposit2

```
lock(a);  
int tempt2=a.amount;  
unlock(a);  
tempt2=tempt2+1000$;  
lock(a);  
a.amount=tempt2;  
unlock(a);
```

Variable	Value
tempt1	11.000\$
...	

Object	Field	Value
a	amount	12.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	12.000\$
...	

Variable	Value
tempt1	11.000\$
...	

Object	Field	Value
a	amount	11.000\$
	ID	58656
	...	
...		

Variable	Value
tempt2	11.000\$
...	