### The C1x and C++11 concurrency model

#### Mark Batty

University of Cambridge

Sequential consistency

Sequential consistency

**Pthreads** 

Sequential consistency

**Pthreads** 

Java

Sequential consistency

**Pthreads** 

Java

Expose hardware model (e.g. ClightTSO)

Sequential consistency

Pthreads

Java

Expose hardware model (e.g. ClightTSO)

C++11/C1x: SC for data race free programs, almost...

C++11: the next C++

1300 page prose specification defined by the ISO.

The design is a detailed compromise:

- hardware/compiler implementability
- useful abstractions
- broad spectrum of programmers

C++11: the next C++

1300 page prose specification defined by the ISO.

The design is a detailed compromise:

- hardware/compiler implementability
- useful abstractions
- broad spectrum of programmers

We fixed serious problems in both C++11 and C1x, both now finalised.

The C1x/C++11 memory model

#### The C1x/C++11 memory model

- top level
- sequential execution
- simple concurrency
- expert concurrency
- very expert concurrency

How may a program execute?

The memory model is factored out from a symbolic operational semantics.

1.  $P \mapsto E_1, ..., E_n$ 

How may a program execute?

The memory model is factored out from a symbolic operational semantics.

- 1.  $P \mapsto E_1, ..., E_n$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$

# How may a program execute?

The memory model is factored out from a symbolic operational semantics.

- 1.  $P \mapsto E_1, ..., E_n$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$
- 3. is there an  $X_{ij}$  with a race? (actually, several kinds...)

### The relations of a pre-execution

Each symbolic execution,  $E_i$ , contains:

**sb** – *sequenced before* 

asw - additional synchronizes with

**dd** – data-dependence

# The relations of a pre-execution

Each symbolic execution,  $E_i$ , contains:

sb – sequenced beforeasw – additional synchronizes withdd – data-dependence

Each full execution,  $X_{ij}$ , also has:

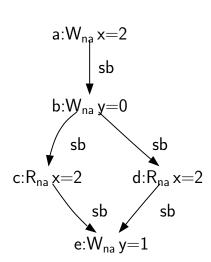
**rf** – reads from

**sc** – *SC* order

**mo** – modification order

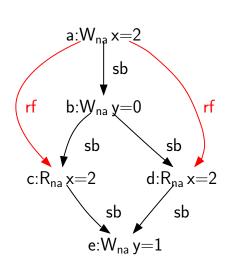
# A single threaded program

```
int main() {
  int x = 2;
  int y = 0;
  y = (x==x);
  return 0; }
```



# A single threaded program

```
int main() {
  int x = 2;
  int y = 0;
  y = (x==x);
  return 0; }
```



#### A data race

#### A data race

int y, x = 2;  

$$x = 3$$
;  $y = (x==3)$   
 $a:W_{na} x=2$   
 $b:W_{na} x=3$   $c:R_{na} x=2$   
 $sb$   
 $d:W_{na} y=0$ 

```
atomic_int x = 0;
atomic_int y = 0;
x.store(1, seq_cst); | y.store(1, seq_cst);
y.load(seq_cst);
                    x.load(seq_cst);
                                    e:W<sub>sc</sub> x=1
sb \sqrt{f:R_{sc}} y=0
```

```
atomic_int x = 0;
atomic_int y = 0;
x.store(1, seq_cst); | y.store(1, seq_cst);
```

```
atomic_int x = 0;
atomic_int y = 0;
x.store(1, seq_cst); | y.store(1, seq_cst);
y.load(seq_cst);
                   x.load(seq_cst);
      c:W_{sc}y=1
                              e:W_{sc}x=1
        SC
                        SC
       d:R_{sc}x=0
                               f:R_{sc} v=1
```

## An example rule

```
let sc\_reads\_restricted actions rf sc mo hb = \forall (a, b) \in rf.

is_seq\_cst b \rightarrow ((adjacent\_less\_than\_such\_that

(fun c \rightarrow is\_write c \land same\_location b c)

sc actions a b)

\lor ...)
```

Using only seq\_cst reads and writes gives SC.

(Initialization is not seq\_cst though...)

```
d:R_n x=1
```

```
// sender
                        // receiver
                     while (0 == y.load(acquire));
x = \dots
y.store(1, release); r = x;
                          d:R_n x=1
```

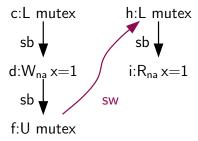
$$\frac{\textit{simple-happens-before}}{\longrightarrow} = \\ \left( \frac{\textit{sequenced-before}}{\longrightarrow} \cup \frac{\textit{synchronizes-with}}{\longrightarrow} \right)^{+}$$

```
int x, r;
mutex m;
m.lock();
x = ...
m.unlock();
m.lock();
```

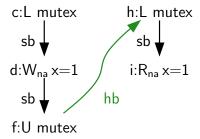
```
int x, r;
mutex m;
m.lock();
                   m.lock();
                   | r = x;
x = \dots
m.unlock();
     c:L mutex
                    h:L mutex
                  i:R_{na}x=1
    d:W_{na} x=1
     f:U mutex
```

```
int x, r;
mutex m;
m.lock();
                    m.lock();
                     r = x;
x = \dots
m.unlock();
     c:L mutex
                      h:L mutex
                       sb,
     d:W_{na}x=1
                      i:R_{na}x=1
     f:U mutex
```

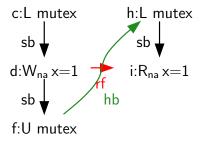
```
int x, r;
mutex m;
m.lock();
x = ...
m.unlock();
m.lock();
```



```
int x, r;
mutex m;
m.lock();
x = ...
m.unlock();
m.lock();
```



```
int x, r;
mutex m;
m.lock();
x = ...
m.unlock();
m.lock();
```



# Happens before is key to the model

Non-atomic loads read the most recent write in happens before. (This is unique in DRF programs)

The story is more complex for atomics, as we shall see.

Data races are defined as an absence of happens before.

#### A data race

int y, x = 2;  

$$x = 3$$
;  $y = (x==3)$   
 $a:W_{na} x=2$   
 $b:W_{na} x=3$   $c:R_{na} x=2$   
 $sb$   
 $d:W_{na} y=0$ 

#### Data race definition

```
let data\_races actions hb =  { (a, b) \mid \forall \ a \in actions \ b \in actions \mid  \neg \ (a = b) \land  same\_location a \ b \land  (is_write a \lor  is_write b) \land  \neg \ (same\_thread \ a \ b) \land  \neg \ (is\_atomic\_action \ a \land \ is\_atomic\_action \ b) \land  \neg \ ((a, b) \in hb \lor (b, a) \in hb)  }
```

A program with a data race has undefined behaviour.

## Relaxed writes: load buffering

No synchronisation cost, but weakly ordered.

### Relaxed writes: independent reads, independent writes

## Expert concurrency: fences avoid excess synchronisation

## Expert concurrency: fences avoid excess synchronisation

```
// receiver
while (0 == y.load(relaxed));
// sender
x = \dots
```

```
// sender
x = ...
y.store(1, release);
// receiver
while (0 == y.load(relaxed));
fence(acquire);
r = x;
```

```
// receiver
// sender
                  while (0 == y.load(relaxed));
x = \dots
r = x;
        d:W_{rel} y=1
                      g:R_n \times 1
```

```
// receiver
// sender
               while (0 == y.load(relaxed));
x = \dots
r = x;
```

```
// receiver
// sender
                        while (0 == y.load(relaxed));
x = \dots
                        fence(acquire);
y.store(1, release);
                         r = x;
```

## Expert concurrency: modification order

Modification order is a per-location total order over atomic writes of any memory order.

## Expert concurrency: modification order

Modification order is a per-location total order over atomic writes of any memory order.

```
b:W<sub>rlx</sub> x=1 d:R<sub>rlx</sub> x=1

sb \downarrow c:W<sub>rlx</sub> x=2 e:R<sub>rlx</sub> x=2
```

## Expert concurrency: modification order

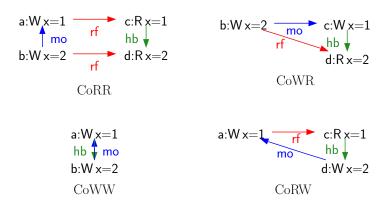
Modification order is a per-location total order over atomic writes of any memory order.

```
b:W<sub>rlx</sub> x=1 d:R<sub>rlx</sub> x=1

mo sb c:W<sub>rlx</sub> x=2 e:R<sub>rlx</sub> x=2
```

## Coherence and atomic reads

## All forbidden!



Atomics cannot read from later writes in happens before.

A successful compare\_exchange is a read-modify-write.

```
x.store(1, relaxed); | compare_exchange(&x, 2, 3, relaxed, relaxed);
x.store(2, relaxed); |
x.store(4, relaxed); |
```

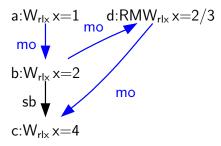
A successful compare\_exchange is a read-modify-write.

x.store(1, relaxed); | compare\_exchange(&x, 2, 3, relaxed, relaxed);

```
x.store(2, relaxed);
x.store(4, relaxed);
              a:W_{rlx} x=1 d:RMW_{rlx} x=2/3 sb
```

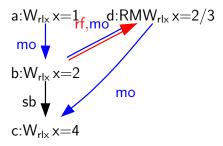
A successful compare\_exchange is a read-modify-write.

```
x.store(1, relaxed); | compare_exchange(&x, 2, 3, relaxed, relaxed);
x.store(2, relaxed); |
x.store(4, relaxed); |
```



A successful compare\_exchange is a read-modify-write.

```
x.store(1, relaxed); | compare_exchange(&x, 2, 3, relaxed, relaxed);
x.store(2, relaxed); |
x.store(4, relaxed); |
```



# Very expert concurrency: consume

Weaker than acquire

Stronger than relaxed

Non-transitive happens before! (only fully transitive through data dependence,  $\operatorname{dd}$ )

C1x and C++11 support many modes of programming:

sequential

- sequential
- concurrent with locks

- sequential
- concurrent with locks
- with seq\_cst atomics

- sequential
- concurrent with locks
- with seq\_cst atomics
- with release and acquire

- sequential
- concurrent with locks
- with seq\_cst atomics
- with release and acquire
- with relaxed, fences and the rest

- sequential
- concurrent with locks
- with seq\_cst atomics
- with release and acquire
- with relaxed, fences and the rest
- with all of the above plus consume

## The full model

| Second State   |                              |
|--|------------------------------|
| March   Marc   |                              |
| Section   Sect   |                              |
| Section   Sect   |                              |
| Section   Sect   |                              |
| Section of the control of the contro |                              |
| The content of the    |                              |
| The content of the    |                              |
| Manual Content   |                              |
| Manual Content   |                              |
| The content of the    |                              |
| The content of the    |                              |
| The content of the    |                              |
| The content of the    |                              |
| The content of the    |                              |
| The content of the    |                              |
| The state of the   |                              |
| March   Marc   |                              |
| The state of the   |                              |
| March   Marc   |                              |
| The state of the   |                              |
| The state of the   |                              |
| The state of the   |                              |
| The content of the    |                              |
| The content of the    |                              |
| The content of the    |                              |
| The state of the   |                              |
| Part   |                              |
| The content of the    |                              |
|  |                              |
| March   Marc   |                              |
| The state of the s |                              |
| The state of the s |                              |
| Service of the control of the contro |                              |
| Service of the control of the contro |                              |
| March   Marc   |                              |
| March   Marc   |                              |
| March   Marc   |                              |
| March   Marc   |                              |
| The state of the   |                              |
| The state of the   |                              |
| The state of the   |                              |
|  |                              |
| The state of the s |                              |
| The state of the s |                              |
| The second of th |                              |
| The properties of the properti |                              |
| The segretary of the contract  |                              |
| Aug   Participated    |                              |
| Management   Application   A   |                              |
|  | , n. m.                      |
| Company of the Comp   |                              |
| The Annual Process of the Control    |                              |
| Section 1 and 1 an |                              |
| Section (Section (Sec |                              |
|  |                              |
|  |                              |
| The property of the control of the c |                              |
| Design of the control |                              |
| Section of the control of the contro |                              |
|  |                              |
| The contract   The    |                              |
| The state of the s |                              |
| Section   Sect   |                              |
| Management   Man   |                              |
| H depther - (1) response - (1) respo | ) A B                        |
|  |                              |
| Contribute the skylanded an income. 1  |                              |
| Control of the Cont   | and a second contract of the |
|  |                              |
|  |                              |
|  |                              |
|  |                              |

## The full model

|  |  | with an white plan after the contract the co    |
|--|--|---|
|  | regions rejectors<br>regions rejectors/proceptor s   | contracting the artists from a boson and experient before extracting contracting the dependency contracting properties and  |
|  |  |   |
|  | Steman (Sec.) Sec. Sec. Sec. Sec. Sec. Sec. Sec. Sec.  | The second of th    |
|  | September 11   | in section  |
|  | From .   | To compare the contract of the    |
|  | (a)    | to the beautiful and  |
|  | See City   |   |
|  |  |   |
|  | Management when having the first bearing and supposed to the sufficiency of control and also depending control depending |   |
|  | The property of the Thirds States and September of the Associated Advantage Associated Advantage Associated Advantage Associated Advantage Associated Advantage Associated Advantage Associated Associ | - Mary construction of the    |
|  |  | Considered to a produce to the<br>(Put traphology and the part of the<br>(management plants and traphology and the part of the<br>consequence of parts and the parts of the parts |
|  | Auditorymana y Spirite America.  |   |
|  |  | management of part of the confidence of the conf    |
| Project C.  State and State C.  State and State C.  Stat | Direct   D   |   |
|  |  | A 18th page may from a fields of a three three to the contract contract posterior in the contract of the dependency contracts and the tage contract and the dependency contracts and the tage contracts and the contract of th    |
|  |  |   |
|  | Application data reported of the Treat Association applicabilities of the Application and Appl |   |
|  | And the statement while the A substitution of the Andrews Statement and Andrews Statement and Statement of Statement and Stateme |   |
|  |  |   |
|  | Contract of the Contract of th |   |
|  |  |   |
|  | · · · · · · · · · · · · · · · · · · ·  | [20] [and of Linguistic distance for an experiment of the proof of th     |
| ***************************************  | Suppose straig attention of an analysis of an analysis of the straightful and an analysis of the straightful analysis of the straightful and an analysis of  |   |
|  | System or type a track of  |   |
|  | Company actions of   |   |
| The consigning of the constraint of the constr   |  | Figure 2. The second se    |
|  | System at Specific 81 consplicad a 81<br>(i.e. a discount a diagnosis  |   |
|  |  |   |
|  | Communication and Communicatio |   |
| horaco i   |  |   |
|  |  |   |
|  |  |   |
|  |  |   |
|  |  |   |
| The form of the company of the compa | Opinion entitipation of the control  | - Sur State Companies on Transparence on Trans    |
| (max more series)  | the second of th |   |
|  |  | Spirit & Company of the Spirit    |
|  | By primary the extragation (a) (a) (b) (b) (b) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c  |   |
|  |  | (9.6) 6 spanger s   |
|  |  |   |
|  |  | (Sec) Simples   |
|  | and the same that the design of the same and | to properly the company of the advance of the company of the compa    |
|  |  |   |
|  | (substituting the state three bottom by the different policies and Advantage day (substitution and a state agrees by the first other agrees of   | Character 5   |
| Toponium or and a second   | And Andrewson Communication Co |   |
|  |  |   |
|  |  |   |
|  |  |   |
|  | may pertugged the true true to the common approximate and the second the depotes; and depotes; or indepotes of the   | Post 17 Temperature (Control of the Control of the     |
|  |  |   |
|  |  |   |
|  |  | Penarty   |
|  |  |   |
|  |  |   |
|  | Signatury principalities in the Managements of the Control of the  | Ferry Control     |
|  |  |   |
|  | Appello gradient delicitati e chica di contra della della contra della |   |
|  |  |   |
|  |  |   |
|  | employees : TOTAL :  |   |
|  |  |   |
|  |  |   |
| 7.7.   |  |   |
|  |  | August and     |
| Section of Control o   | Control of the Contro | Applications of the second of     |
|  | mAm (M)  | angle distance of the control of the    |
| The second secon | un produce plant a materialmente, a  | With distance of the control of the     |
|  | un produce plant a materialmente, a  | Section Control of Con    |
|  | Undergraphy (Section 2) and the section (Section 2) and th | Continue (Continue (Contin    |
|  | mAm (M)  | Control (Control Control Contr    |
|  | School (1)  We consider the constraint of the co | Control (Control Control Contr    |
|  | School (1)  We consider the constraint of the co | antificial production of the control    |
|  | Section (Control of Control of Co | International Control     |
|  | Section (Control of Control of Co | Control (Control Control Contr    |
|  | School (1)  We consider the constraint of the co | International Control     |
|  | Section (Control of Control of Co | International Control     |
| Information of the second of t |  | International Control     |
| Information of the second of t | Takenda Andreas Andrea | International Control     |
| Information of the second of t | Takenda Andreas Andrea | Service of the control of the contro    |
| The second secon | Takenda Andreas Andrea | Service of the control of the contro    |
| The second secon | Takenda Andreas Andrea | International Control     |
| The second secon | Takenda Andreas Andrea | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon | The state of the s |   |
| The second secon | The state of the s | The second secon    |
| The second secon | The state of the s |   |
| The second secon | The state of the s |   |
| The second secon | The state of the s | The second secon    |
| The second secon |  |   |
| The second secon | The state of the s | Section of the control of the contro    |
| The second secon | The state of the s | Section of the control of the contro    |
| The second secon | The state of the s |   |
| The second secon | The state of the s |   |
| The second secon |  | The second secon    |
| The second secon | The state of the s |   |
| The second secon |  | The second secon    |
| The second secon |  | The second secon    |
| The second secon |  | Section of the content of the conten    |



## Are C1x and C++11 hopelessly complicated?

Programmers cannot be given this model!

With a formal definition, we can do proof, and even mechanise it.

What do we need to prove?

## Are C1x and C++11 hopelessly complicated?

Programmers cannot be given this model!

With a formal definition, we can do proof, and even mechanise it.

What do we need to prove?

- implementability
- simplifications
- libraries

# Implementability

Can we compile to x86?

## Implementability

## Can we compile to x86?

| x86 Implementation |
|--------------------|
| mov                |
| lock xadd(0)       |
| mov                |
| lock xchg          |
| no-op              |
|                    |

x86-TSO is stronger and simpler.

Recall the C/C++ semantics for program P:

1.  $P \mapsto E_1, ..., E_n$ ,

Recall the C/C++ semantics for program P:

1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\mathrm{opsem}}$ 

Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\mathrm{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ ,

Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\text{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ , collectively  $X_{\text{witness}}$

Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\text{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ , collectively  $X_{\text{witness}}$
- 3. is there an  $X_{ij}$  with a race? (actually, several kinds...)

# Top level comparison

Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\text{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ , collectively  $X_{\text{witness}}$
- 3. is there an  $X_{ij}$  with a race? (actually, several kinds...)

## In x86-TSO:

Events and dependencies,  $E_{\rm x86}$  are analogous to  $E_{\rm opsem}$ .

# Top level comparison

Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\text{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ , collectively  $X_{\text{witness}}$
- 3. is there an  $X_{ij}$  with a race? (actually, several kinds...)

## In x86-TSO:

Events and dependencies,  $E_{x86}$  are analogous to  $E_{opsem}$ . Execution witnesses,  $X_{x86}$  are analogous to  $X_{witness}$ .

# Top level comparison

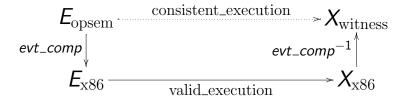
Recall the C/C++ semantics for program P:

- 1.  $P \mapsto E_1, ..., E_n$ , each an  $E_{\text{opsem}}$
- 2.  $E_i \mapsto X_{i1}, ..., X_{im}$ , collectively  $X_{\text{witness}}$
- 3. is there an  $X_{ij}$  with a race? (actually, several kinds...)

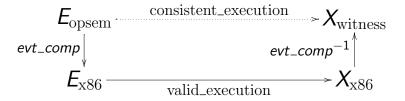
## In x86-TSO:

Events and dependencies,  $E_{x86}$  are analogous to  $E_{opsem}$ . Execution witnesses,  $X_{x86}$  are analogous to  $X_{witness}$ . There is not a DRF semantics.

## **Theorem**



## **Theorem**



We have a mechanised proof that C1x/C++11 behaviour is preserved.

# Implementability

Can we compile to IBM Power?

# Implementability

Can we compile to IBM Power?

| C++0x Operation  | POWER Implementation         |
|------------------|------------------------------|
| Non-atomic Load  | ld                           |
| Load Relaxed     | ld                           |
| Load Consume     | ld (and preserve dependency) |
| Load Acquire     | ld; cmp; bc; isync           |
| Load Seq Cst     | sync; ld; cmp; bc; isync     |
| Non-atomic Store | st                           |
| Store Relaxed    | st                           |
| Store Release    | lwsync; st                   |
| Store Seq Cst    | sync; st                     |

We have a hand proof that C1x/C++11 behaviour is preserved.

# Simplifications

Full model – visible sequences of side effects are unneeded (HOL4).

# Simplifications

Full model – visible sequences of side effects are unneeded (HOL4).

#### Derivative models:

- without consume, happens-before is transitive (HOL4).
- DRF programs using only seq\_cst atomics are SC (false).

# Simplifications

Full model – visible sequences of side effects are unneeded (HOL4).

#### Derivative models:

- without consume, happens-before is transitive (HOL4).
- DRF programs using only seq\_cst atomics are SC (false).

 $atomic_init$  is a non-atomic write, and in C1x/C++11 they race...

# Usability

Provide simplified models for higher level constructs.

Formal description of mutual exclusion in terms of happens-before.

We need libraries that provide a simpler model to programmers.

# $\begin{array}{c} {\rm CPPMEM} \\ {\rm helps\ explore\ and\ understand\ the\ model} \end{array}$

## Сррмем

Code in, all executions out

Confidence and speed

Communication

# How may a program execute in CPPMEM?

1.  $P \mapsto E_1, ..., E_n$  — tracking constraints

2.  $E_i \mapsto X_{i1}, ..., X_{im}$  — automatically uses formal model

3. is there an  $X_{ij}$  with a race?

Refinements to the standards

# The current state of the standard

## Fixed:

- Happens-before
- Coherence
- seq\_cst atomics were more broken

# The current state of the standard

## Fixed:

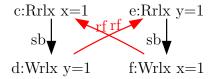
- Happens-before
- Coherence
- seq\_cst atomics were more broken

### Not fixed:

- Self satisfying conditionals
- seq\_cst atomics are still not SC

# Self-satisfying conditionals

```
r1 = x.load(mo_relaxed);
if (r1 == 42)
    y.store(r1, mo_relaxed);
    r2 = y.load(mo_relaxed);
    if (r2 == 42)
        x.store(42, mo_relaxed);
```



## Conclusion

It's OK to like the C++0x memory model design

Our formal model lets us make fun things (go use it!)

- Optimized compilation?
- Static analysis?
- Dynamic analysis?
- Observational congruence?
- Program logics?