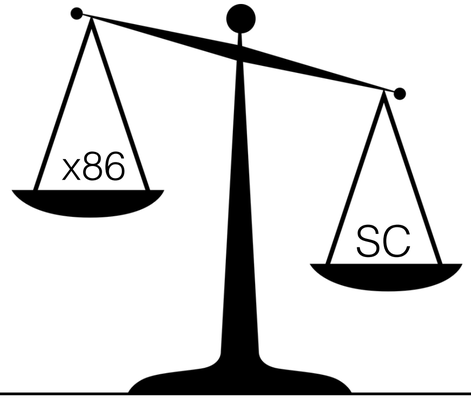



# Automatically Comparing Memory Consistency Models

**John Wickerson**
Imperial College

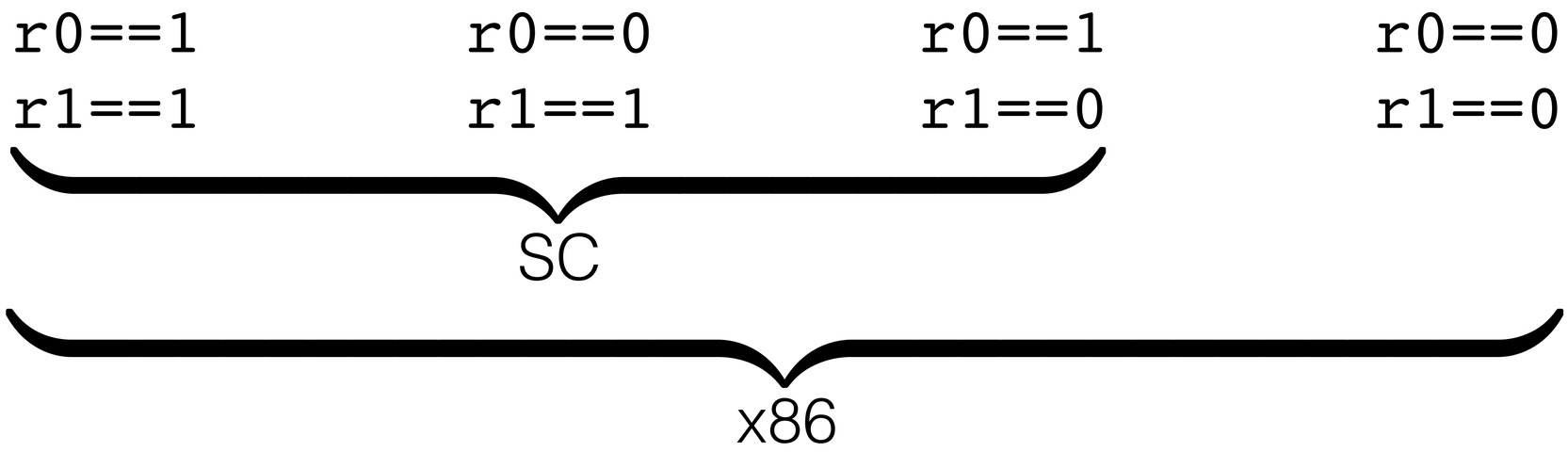
**Mark Batty**
University of Kent

**Tyler Sorensen**
Imperial College

**George Constantinides**
Imperial College

# Memory Models

```
x = 1;  ||  y = 1;
r0 = y; ||  r1 = x;
```



# Summary

- **The problem is** that memory models are complex and counterintuitive. Misunderstanding them has led to lots of bugs.
- **Our contribution is** a technique for automatically generating litmus tests that distinguish two models.
- **This is useful because** we can compare two variants of a model, and we can debug compilers.

# Three tasks

1. To show that **M** is **stronger** than **N**:
   - find P,σ where $\sigma \notin [\![P]\!]_{M}$ and $\sigma \in [\![P]\!]_{N}$.

2. To show that a **compiler optimisation** on **M** is unsound:
   - find P,Q,σ where $\sigma \notin [\![P]\!]_{M}$, P optimises to Q, and $\sigma \in [\![Q]\!]_{M}$.

3. To show that a **compiler mapping** from **M** to **N** is unsound:
   - find P,Q,σ where $\sigma \notin [\![P]\!]_{M}$, P maps to Q, and $\sigma \in [\![Q]\!]_{N}$.

# Related work

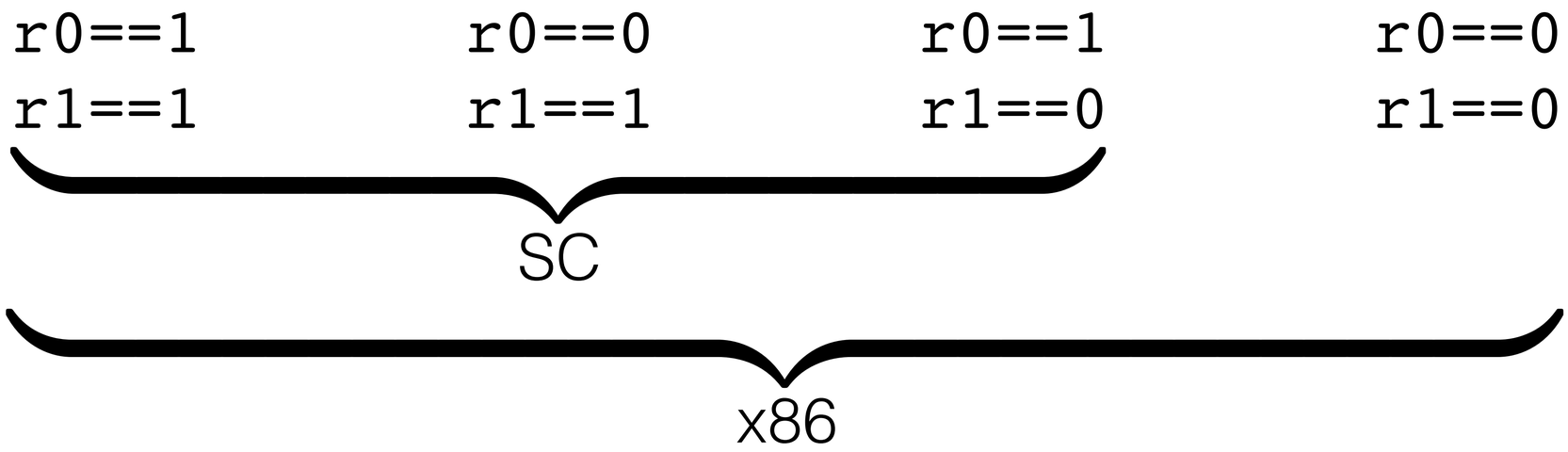
| | **Manual c'examples** | **Manual proofs** | **Automatically** |
|---|---|---|---|
| **Comparing memory models** | [Batty et al. *POPL'16*]<br>[Nienhuis et al. *OOPSLA'16*]<br>[Lahav et al. *POPL'16*] | [Owens et al. *TPHOLs'09*]<br>[Mador-Haim et al. *CAV'12*] | [Mador-Haim et al. *CAV'10*]<br>**THIS WORK** |
| **Checking compiler optimisations** | [Vafeiadis et al. *POPL'15*] | [Sevcik *PLDI'11*] | [Burckhardt et al. *CC'10*]<br>[Chakraborty et al. *CGO'16*]<br>**THIS WORK** |
| **Checking compiler mappings** | [Wickerson et al. *OOPSLA'15*] | [Batty et al. *POPL'11*] | [Lustig et al. *MICRO'14*]<br>**THIS WORK** |

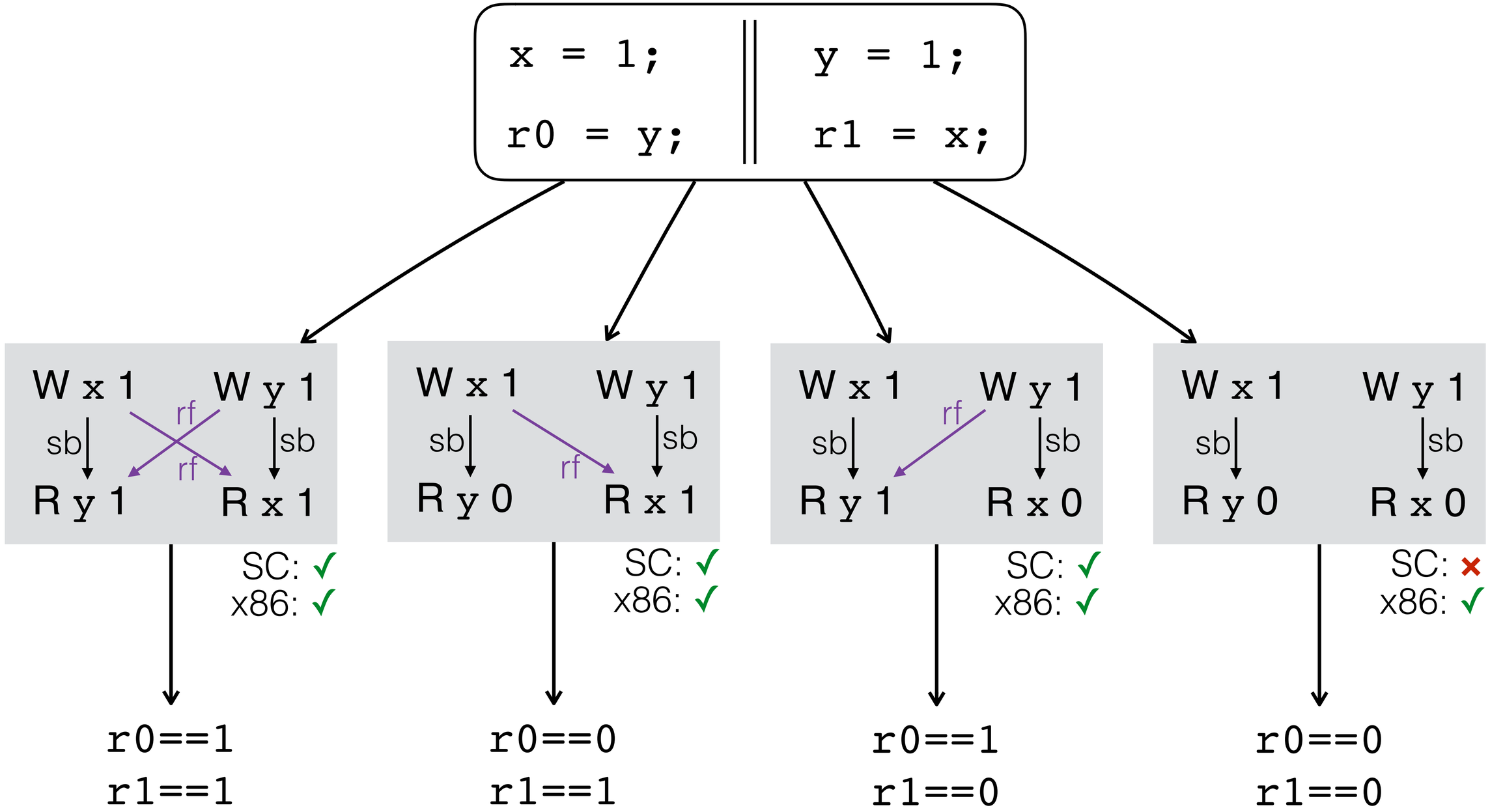
# Results

| | Previous work | Our tool (automatic) | Time taken |
|---|---|---|---|
| **Comparison:** Batty et al.'s C++ vs. original | 7 | 5 | <1 sec |
| **Comparison:** Lahav et al.'s C++ vs. original | 10 | 6 | 1 sec |
| **Comparison:** Nienhuis et al.'s C++ vs. original | 12 | 12 | 4 min |
| **Optimisation:** strengthening memory order in C++ | 8 | 7 | 1 min |
| **Optimisation:** thread linearisation in C++ | 6 | 6 | 5 sec |
| **Compiler mapping:** from OpenCL to AMD | 12 | 11 | 22 min |
| **Compiler mapping:** from C++ to Power | 14 | 14 | 2 min |

# Memory Models

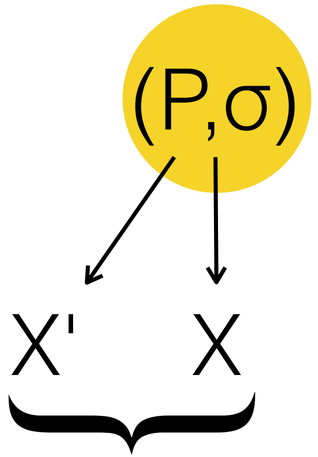
```
x = 1;  ||  y = 1;
r0 = y; ||  r1 = x;
```




x = 1;
r0 = y;
y = 1;
r1 = x;
W x 1
W y 1
sb
rf
rf
sb
R y 1
R x 1
SC: ✓
x86: ✓
r0==1
r1==1
W x 1
W y 1
sb
rf
sb
R y 0
R x 1
SC: ✓
x86: ✓
r0==0
r1==1
W x 1
W y 1
sb
rf
sb
R y 1
R x 0
SC: ✓
x86: ✓
r0==1
r1==0
W x 1
W y 1
sb
sb
R y 0
R x 0
SC: ✗
x86: ✓
r0==0
r1==0

# Why is this tricky?

$$\{(P,\sigma) \mid \sigma \notin [\![P]\!]_M \wedge \sigma \in [\![P]\!]_N\}$$

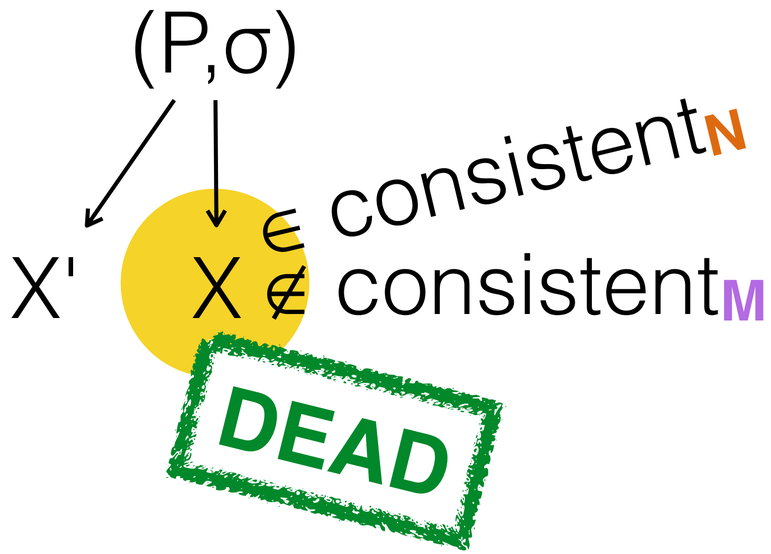



**all** are inconsistent under **M**

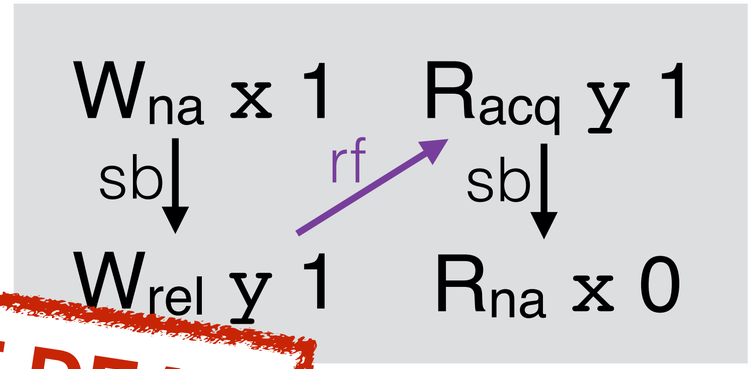
**some** are consistent under **N**

# Our solution

$$\{(P,\sigma) \mid \sigma \notin [\![P]\!]_M \wedge \sigma \in [\![P]\!]_N\}$$



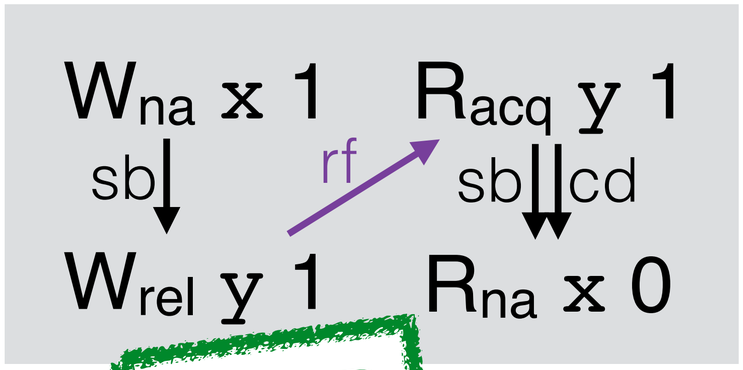
# Generating useful tests




NOT DEAD

C++: ×

```
x=1;                    ||  r0=ld(y,ACQ);
st(y,1,REL);            ||  r1=x;
```




DEAD

C++: ×

```
x=1;                    ||  r0=ld(y,ACQ);
st(y,1,REL);            ||  if(r0) r1=x;
```

# Conclusion

- **In summary:** we use Alloy to compare memory consistency models, and check for compiler bugs.
- **Future work:** how to extend our work to handle more recent styles of model like Promising model.