Data Race Freedom for Weakestmo: mechanization in COQ

Ilya Kaysin
ilya.s.kaysin@gmail.com

Advisor: Anton Podkopaev,
anton@podkopaev.net

MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

HIGHER SCHOOL OF ECONOMICS NATIONAL RESEARCH UNIVERSITY

JET BRAINS RESEARCH

Weak memory

```
          [data] = b = null
          [flag] = a = False

                          ||
[data] := "hello"         ||   a := [flag]
[flag] := True            ||   if a:
                          ||      b := [data]
                          ||   print(a, b)
```

🤔 ...what are the possible outputs?

Sequentially **C**onsistent outcomes:

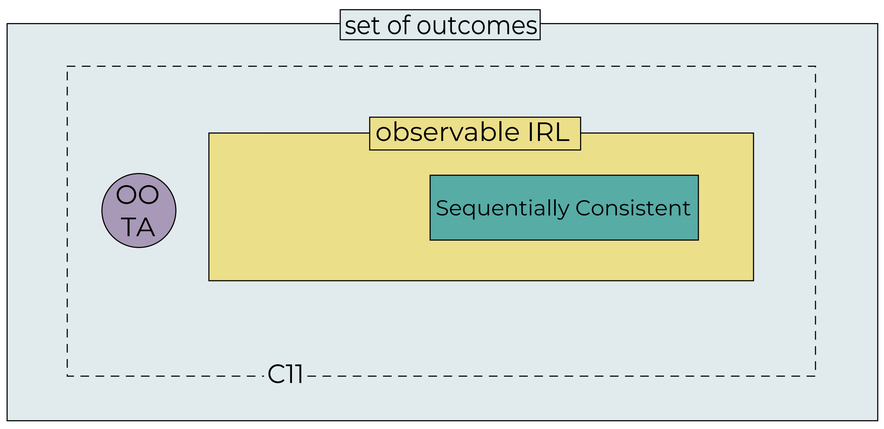
a = True, b = "hello"

a = False, b = null

Weak outcome is also observable!

😮 a = True, b = null

Weak memory models



☝ Weak memory models formalize the **semantics** (behavior) **of multithreading programs**

☝ There're several models enhancing **C11**: RC11[3], Promising semantics[2], and **Weakestmo**[1]

😮 **BUT** weak memory models are complicated!

😟 It would be hard for programmers to understand their own code if it weren't for the...

DRF property

NO DATA RACES => NO WEAK BEHAVIORS

😌 If a memory model satisfies DRF, avoiding data races ensures sequential consistency

☝ DRF has several modifications, not all of them are true for C11

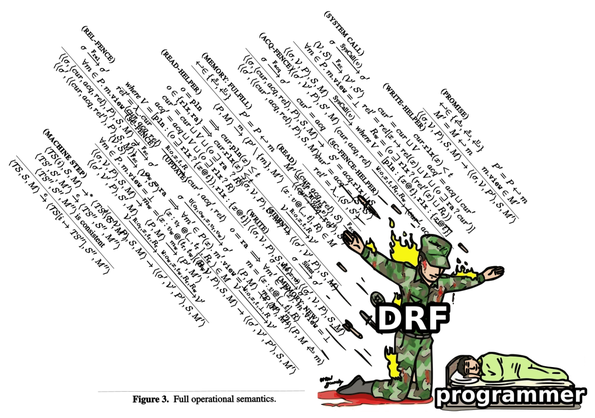


Figure 3. Full operational semantics.

Our project

GOAL: To verify DRF theorem proofs for Weakestmo memory model in COQ

🐥 ~7k LOC in COQ

🐥 335 lemmas proved

🐥 Issues found in Weakestmo[1]

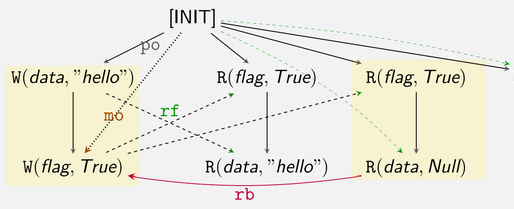
🐥 Work in progress...

DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER EXPERT ZONE DANGER

Execution as a graph

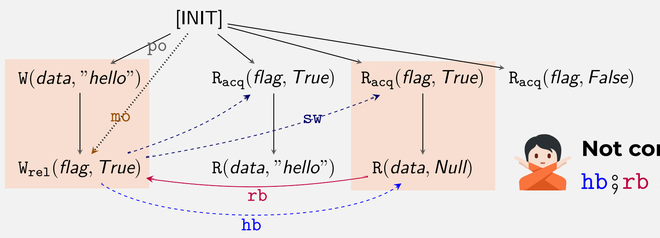
1. Build an Event Structure
   - **Nodes = Events:** Read, Write or Fence
   - **Edges = Relations**
2. Extract Execution as a **subgraph**
3. Check the Weakestmo **consistency**

po (program order)
rf (reads from)
mo (memory order)
rb (reads before)
sw (synchronized with)
hb (happens before)
...



☝ $\mathtt{rb} = (\mathtt{rf}^{-1} \,;\, \mathtt{mo})$
$\mathtt{hb} = (\mathtt{sw} \cup \mathtt{po})^{+}$

Release-Aquire modifiers ensure **synchronization** and **rule out** the weak execution:



🤦 **Not consistent:**
$\mathtt{hb} \,;\, \mathtt{rb}$ should be irreflexive!

DRF in detail

(Race)

Two events are racy iff:
(1) They access the **same location**
(2) At least one of them is **Write**
(3) They're not related with **hb**

(DRF-SC)

All racy events in all SC-consistent executions have SC access modifiers => No weak behaviors

☝ Weakestmo[1] also satisfies **DRF-RA** (Release-Aquire) and **DRF-RLX** (Relaxed)

Out-Of-Thin-Air

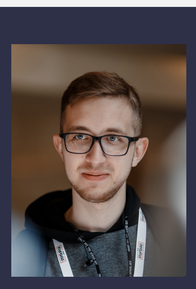
```
     [x] = [y] = a = b = 0

a := [x]          ||   b := [y]
if a:             ||   if b:
   [y] := a       ||      [x] := b
```

😌 a = 0, b = 0

😮 a = 1, b = 1

😮 a = 42, b = 42

🤦 The C11 model allows reading values non-existing in the program



ILYA KAYSIN

ilya.s.kaysin@gmail.com
facebook.com/demarkok
+7 911 131-13-89

Contact me

[1] Chakraborty, S., and Vafeiadis, V. Grounding thin-air reads with event structures. PACMPL 3, POPL (2019), 70:1–70:28.
[2] Kang, J., Hur, C.-K., Lahav, O., Vafeiadis, V., and Dreyer, D. A promising semantics for relaxed-memory concurrency. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017 (2017), pp. 175–189.
[3] Lahav, O., Vafeiadis, V., Kang, J., Hur, C.-K., and Dreyer, D. Repairing sequential consistency in C/C++11. In Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017 (2017), pp. 618–632.

Refferences

github.com/weakmemory/weakestmoToImm

Project repo