Exercises - C11, C++11

In the exercices below, you should use the CPPMem tool, available online at:

`http://svr-pes20-cppmem.cl.cam.ac.uk/cppmem/`

1. Load CPPMem, use the left-most drop down box to select the *examples/IRIW* test, change all memory orders to `memory_order_relaxed`, and press the run button. CPPMem will print an execution graph of the relaxed behaviour that this test allows.

   (a) Play with the tick boxes on the left to turn on and off the printing of relations. Make sure you look at *hb*, *rf*, *sc* and *mo*.

   (b) Use the following table to "compile" the program to a Power program:

C++0x Operation	POWER Implementation
Non-atomic Load	`ld`
Load Relaxed	`ld`
Load Seq Cst	`sync; ld; cmp; bc; isync`
Non-atomic Store	`st`
Store Relaxed	`st`
Store Seq Cst	`sync; st`

   Will the new program produce the relaxed behaviour on the Power abstract machine?

   (c) Recall that on Power, placing a `sync` barrier between each pair of reads was sufficient to forbid this behaviour. Press *reset*, and then alter the C program so that the last load on each load thread has order `memory_order_seq_cst`. How does the "compiled" program change, and is the relaxed behaviour still allowed on the Power abstract machine? Why is the relaxed behaviour still allowed in C?

   (d) By choosing different memory orders for each load or store in the C program, forbid the relaxed behaviour in C. Why does your solution work?

2. Use the left-most drop down box to select the *examples/LB* test and press run.

   (a) Could the SC machine reproduce the execution that CPPMem prints?

   (b) Press the *next consistent* button and inspect each consistent execution. Can all of the executions be seen on an SC machine?

   (c) Use the *previous candidate* and *next candidate* buttons to explore the candidate executions. Find the one with the relaxed behaviour. What predicates in the model fail?

   (d) Alter the memory orders in the program so that load-buffering relaxed behaviour can be witnessed.

3. For each program, enumerate the values that can be read at the commented line, given the commented constraints, and say why:

   (a)

   ```
   atomic_int x = 0;
   x.store(1,relaxed);  | x.load(relaxed); \\ this reads 1
                        | x.load(relaxed); \\ What values can be read?
   ```

   (b)

   ```
   atomic_int x = 0;
   x.store(1,relaxed);
   x.load(relaxed); \\ What values can be read?
   x.store(2,relaxed);
   ```