C++ Memory Order revision

Ex 1

Initial state:
Atomic_int x{0};
Atomic_int y{0};
Int RES = 0;

T1:

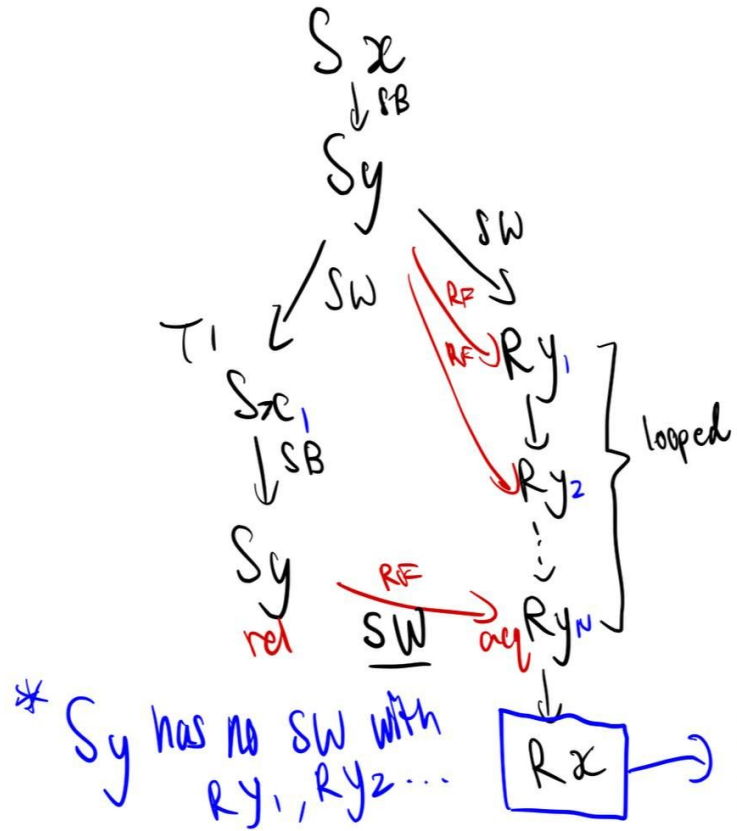
X.store(1, relaxed)

y.store(1, release)

T2:

while (y.load(acquire) != 1) {}

RES = x.load(relaxed)



* Sy has no SW with $Ry_1, Ry_2 \ldots$ Rx → can only RF Rx_1

Hence, RES is 1

Ex 2

Initial state:
Atomic_int x{0};
Atomic_int y{0};
Int RES1 = 0;
Int RES2 = 0;

T1:

X.store(1, relaxed)

y.store(1, release)

T2:

while (y.load(acq) != 1) {}

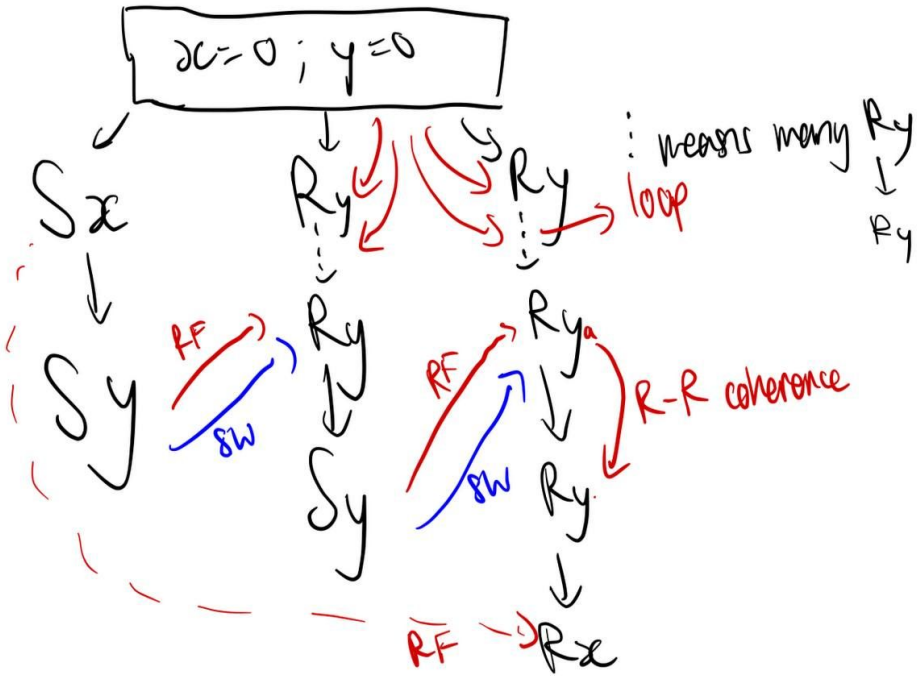
y.store(2, release)

T3:

while (y.load(acq) != 2) {}

RES1 = y.load(relaxed)

RES2 = x.load(relaxed)



$Rx = \{1\}$

$Ry = \{2\}$: By R-R coherence, Ry must be $= Ry_a$

Ex 3

Initial state:
Atomic_int x{0};
Atomic_int y{0};
Int RES1 = 0;
Int RES2 = 0;

T1:

X.store(1, relaxed)

y.store(1, release)

T2:

while (y.load(**relaxed**) != 1) {}

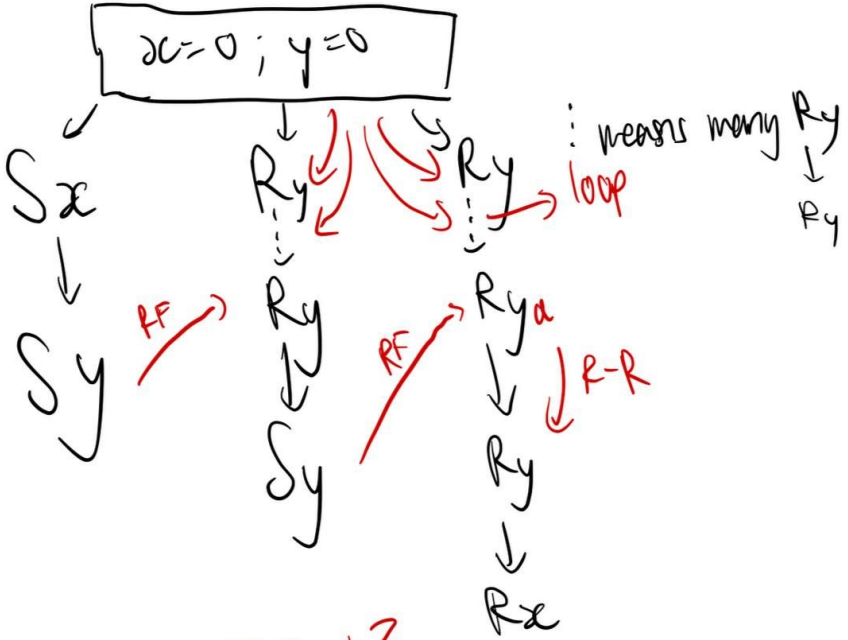
y.store(2, **relaxed**)

T3:

while (y.load(acquire) != 2) {}

RES1 = y.load(relaxed)

RES2 = x.load(relaxed)



$R_x = \{0, 1\}$

$R_y = \{2\}$: R_{y_a} must read 2, be it relaxed or not

Ex 4

Initial state:
Atomic_int x{0};
Atomic_int y{0};
Int RES1 = 0;
Int RES2 = 0;

T1:

X.store(1, relaxed)

y.store(1, release)

y.store(3, relaxed)

T2:

while (y.load(relaxed) != 1) {}

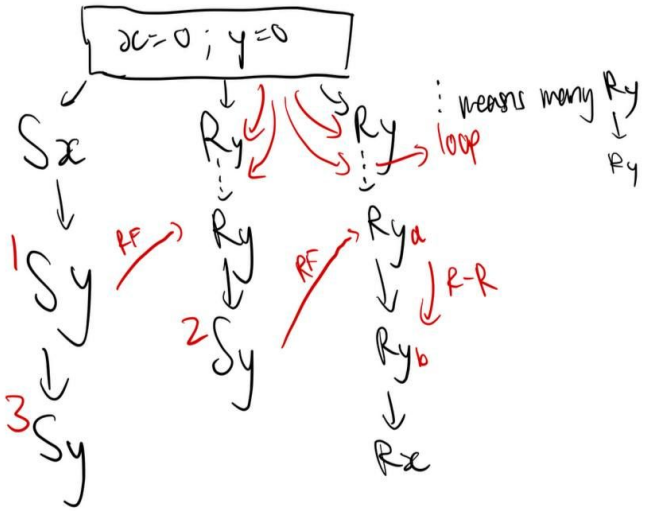
y.store(2, relaxed)

T3:

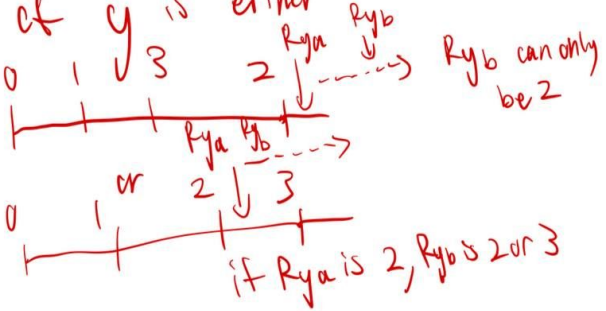
while (y.load(acquire) != 2) {}

RES1 = y.load(relaxed)

RES2 = x.load(relaxed)



mod order of y is either



Note: we did not consider case of ∞ loop here.

$R_x = \{0, 1\}$

Ex 5 - Modification order

Initial state:
Atomic_int x{0};
Int RES1 = 0;
Int RES2 = 0;
Int RES3 = 0;

T1:

X.store(1, relaxed)

T2:

x.store(2, relaxed)

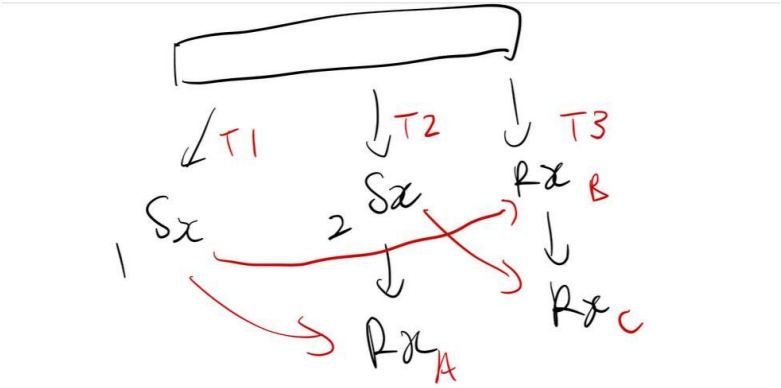
RES1 = x.load(relaxed);

T3:

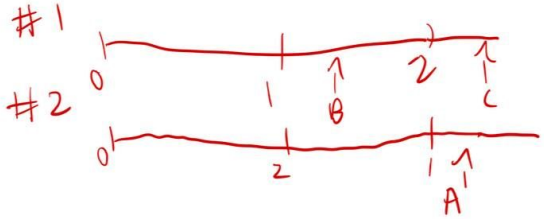
RES2 = x.load(relaxed)

RES3 = x.load(relaxed)

RES1 = [2], RES2 = [0,1]



2 possible mod orders of x.



A = 1, B = 1, C = 2 is invalid.

B = 1, C = 2 → T2 sees B seq before C mod order #1

For A = 1, T1 must see mod order #2.

But we can only have 1 mod order for each var (contradiction)

For more interesting examples, https://gracefu.neocities.org/cpp-atomic-puzzles-1