FSL: A Logic for Reasoning about Memory Fences

Marko Doko, Viktor Vafeiadis

Max Planck Institute for Software Systems

1. Strong vs. weak memory

- Memory models describe all possible behaviors resulting from concurrent accesses to shared memory locations.
- Most verification work assumes a strong memory model (i.e. interleaving semantics).
- In practice, hardware behaves weakly (x86-TSO, POWER, ARM, . . .).
- The C11 memory model unifies various existing hardware models.

2. Examples of weak behavior in C11

1

```
       int a = 0;
       int x = 0;
a = 42; || if (x == 1){
x = 1;  ||   print(a);
        || }
```

race

2

```
         int a = 0;
      atomic_int x = 0;
a = 42;      || if (x_rlx == 1){
x_rlx = 1;   ||   print(a);
             || }
```

race

3

```
      atomic_int a = 0;
      atomic_int x = 0;
a_rlx = 42;  || if (x_rlx == 1){
x_rlx = 1;   ||   print(a_rlx);
             || }
```

rf rf

☺ no races ☺ ☹ can print 0 ☹

3. Fences in C11

Fences can be used to achieve synchronization:

```
       int a = 0;
    atomic_int x = 0;
a = 42;        || if (x_rlx == 1){
fence_rel;     ||   fence_acq;
x_rlx = 1;     ||   print(a);
               || }
```

rf sync

☺ no races ☺ ☺ always prints 42 ☺

Other synchronization primitives, such as *release writes* and *acquire reads*, can be implemented using fences.

4. Fenced separation logic (FSL)

- Extension of relaxed separation logic (RSL).
- Direct reasoning about *release writes* and *acquire reads*.
- Simple inference rules for fences and atomic accesses.
- Proofs of memory safety and race freedom.

5. Inference rules

Atomic allocation:

$$\frac{\mathcal{Q} : \mathsf{Val} \to \mathsf{Assn}}{\{\mathcal{Q}(v)\}\ \texttt{atomic x = v}\ \{\mathsf{W}_{\mathcal{Q}}(x) * \mathsf{R}_{\mathcal{Q}}(x)\}}$$

Release fence:

$$\{P\}\ \texttt{fence}_{rel}\ \{\triangle P\}$$

Atomic write:

$$\{\triangle \mathcal{Q}(v) * \mathsf{W}_{\mathcal{Q}}(x)\}\ \texttt{x}_{rlx} = \texttt{v}\ \{\mathsf{W}_{\mathcal{Q}}(x)\}$$

Atomic read:

$$\{\mathsf{R}_{\mathcal{Q}}(x)\}\ \texttt{t = x}_{rlx}\ \{\nabla \mathcal{Q}(t)\}$$

Acquire fence:

$$\{\nabla P\}\ \texttt{fence}_{acq}\ \{P\}$$

6. Example proof

$\mathcal{Q}(v) \stackrel{def}{=} (v = 0 \lor \&a \mapsto 42)$

```
                 {true}
                int a = 0;
                {&a ↦ 0}
             atomic_int x = 0;
      {&a ↦ 0 * W_Q(x) * R_Q(x)}
{&a ↦ 0 * W_Q(x)}          || {R_Q(x)}
a = 42;                    || if (x_rlx == 1){
{&a ↦ 42 * W_Q(x)}         ||   {∇(&a ↦ 42)}
fence_rel;                 ||   fence_acq;
{△(&a ↦ 42) * W_Q(x)}      ||   {&a ↦ 42}
x_rlx = 1;                 ||   print(a);
{true}                     ||   {true}
                           || }
                 {true}
```