Data-Race-Freedom (DRF) Theorem

Recap: C/C++ Concurrency

C11 semantics

C11 compiler transformations

C11 mappings to processors

Observation: Weak accesses allows more transformations

Questions

Q: Should we weaken the model as far as possible?

A:

(1) Mapping to other model is efficient.

(2) May not always be beneficial for program transformations. (why?)

(3) Programmability is affected.

Programmability Concerns

Can developers get rid of relaxed accesses?

Solution:
Do not use weak accesses.

Cons: compilation is expensive.

Another approach:

Use strong accesses for synchronization.

Use weak accesses otherwise.

Programmability Concerns

No synchronization $\Longrightarrow$ no communication among threads

Synchronization $\Longrightarrow$ data race

Strong accesses for synchronization $\Longrightarrow$ data race on strong accesses

Do programmers have any guarantee?

Program behaviors should be dictated by stronger accesses involved in data race.

Example

Program with lock/unlock.

Global order on lock and unlock operations in an execution.

Sequential consistency/interleaving execution.

$$\begin{array}{l||l} \text{lock}(m) & \text{lock}(m) \\ X = 1; & a = X; \\ \text{unlock}(m) & \text{unlock}(m) \end{array}$$

(i) $a = 0, X = 1$ (ii) $X = 1, a = 1$

C11 Programmability Concerns

$$\begin{array}{l|l} \textbf{if}(X) & \textbf{if}(Y) \\ \quad Y = 1; & \quad X = 1; \end{array}$$

The program is race-free.

C11 allows racy execution $X = Y = 1$.

No guarantee for programmer.

Motivation for the DRF theorem

WMC is complicated:

- Most programmers "do not understand" WMC.
- Leads to subtle bugs $\rightsquigarrow$ hard to debug and fix.

Define programming disciplines that:

- Avoid weak behaviors.
- Can be understood without referring to the WMM.

The DRF discipline:

- Do not have any data races.
- Just use locks for synchronization.

DRF: A minimal requirement for memory models

Definition (DRF property)

A memory model M satisfies the *DRF property* if
for every program that is race-free under SC semantics,
its allowed outcomes under M are the same as under SC.

- A programming discipline to avoid weak behavior.
- The premise requires us to establish race-freedom under SC.
- So a defensive programmer does not need to understand WMM.

For specific memory models, one can establish more permissive programming disciplines that ensure the absence of weak behaviors.

What constitutes a race under SC? (declaratively)

Given an execution graph G and a relation $R \subseteq G.\mathsf{E} \times G.\mathsf{E}$, we say that two events a, b *R-race* in G if the following hold:

- $a \neq b$
- $a.\mathsf{loc} = b.\mathsf{loc}$
- $\{a.\mathsf{op}, b.\mathsf{op}\} \cap \{\mathsf{W}, \mathsf{U}\} \neq \emptyset$
- $\langle a, b \rangle \notin R^+$ and $\langle b, a \rangle \notin R^+$

An execution graph G is called *R-racy* if there are two events that R-race in G.

P is called *racy under SC* if there exists an execution graph G such that the following hold:

- G is a $(\mathsf{po} \cup \mathsf{rf})^+$-prefix of an execution of P
- G is SC-consistent
- G is $(\mathsf{po} \cup \mathsf{rf})$-racy

What constitutes a race under SC? (declaratively)

DRF theorem proof taken from
`https://people.mpi-sws.org/~viktor/wmc/`

Proof outline (1)

- Let G be a the an RA-consistent $(\mathtt{po} \cup \mathtt{rf})$-racy execution graph of P.
- Let G' be a minimal $(\mathtt{po} \cup \mathtt{rf})$-prefix of G that is $(\mathtt{po} \cup \mathtt{rf})$-racy.
- NB: This prefix might not be unique (*e.g.*, SB).
- Let a, b be two events that $(\mathtt{po} \cup \mathtt{rf})$-race in G'.
- Let $x = \mathtt{loc}(a) = \mathtt{loc}(b)$.
- G' is RA-consistent. (why?)
- Let $G'' \triangleq G' \setminus \{a, b\}$.
  - G'' is RA-consistent.
  - G'' is not $(\mathtt{po} \cup \mathtt{rf})$-racy.

  Therefore, G'' is SC-consistent.

Proof outline (2)

- Possible cases:
  - $\mathtt{typ}(a) = \mathtt{W}$ and $\mathtt{typ}(b) = \mathtt{W}$
  - $\mathtt{typ}(a) \in \{\mathtt{R}, \mathtt{RMW}\}$ and $\mathtt{typ}(b) = \mathtt{W}$
  - $\mathtt{typ}(a) = \mathtt{W}$ and $\mathtt{typ}(b) \in \{\mathtt{R}, \mathtt{RMW}\}$ (symmetric)
  - $\mathtt{typ}(a) = \mathtt{R}$ and $\mathtt{typ}(b) = \mathtt{RMW}$
  - $\mathtt{typ}(a) = \mathtt{RMW}$ and $\mathtt{typ}(b) = \mathtt{R}$ (symmetric)
- We cannot have $\mathtt{typ}(a) = \mathtt{RMW}$ and $\mathtt{typ}(b) = \mathtt{RMW}$. (why?)

Proof outline (3)

CASE 1: $\mathtt{typ}(a) = \mathtt{W}$ and $\mathtt{typ}(b) = \mathtt{W}$

- G' is SC-consistent.
  (Take an sc-order for G'' and add a and b at the end)

Proof outline (4)

CASE 2: $\mathtt{typ}(a) \in \{\mathtt{R}, \mathtt{RMW}\}$ and $\mathtt{typ}(b) = \mathtt{W}$

- There exists $a' \sim a$ (a and a' are identical except for the read value, and a' may be a read if a is an $\mathtt{RMW}$) such that some $G_a \in \mathsf{Add}(G'', a')$ is SC-consistent.
  (read from the last write to x in the $\mathtt{sc}$-order for G'')
- Let $G_{ab} \in \mathsf{Add}(G_a, b)$.
- G_{ab} is SC-consistent and $(\mathrm{po} \cup \mathtt{rf})$-racy.

Proof outline (5)

CASE 3: $\mathtt{typ}(a) = \mathtt{R}$ and $\mathtt{typ}(b) = \mathtt{RMW}$

- Let $G_b \triangleq G' \setminus \{a\}$.
- G_b is SC-consistent. (why?)
- b is the $(\mathrm{po} \cup \mathtt{rf})^+$-maximal write to x in G_b.
- There exists $a' \sim a$ (a and a' are identical except for the read value) such that some $G_{ba} \in \mathrm{Add}(G_b, a')$ is SC-consistent and $\langle b, a' \rangle \notin G_{ba}.\mathtt{rf}$.
  (read from the $(\mathrm{po} \cup \mathtt{rf})^+$-maximal write to x in G'')
- G_{ba} is $(\mathrm{po} \cup \mathtt{rf})$-racy.