Correctness of compilation under weak memory models

Ori Lahav Viktor Vafeiadis

29 August 2017

Correctness of compilation under weak memory models

What does "correct compilation" mean?

Compilation correctness

$$[\![\text{compile}(P)]\!]_{\textit{target memory model}} \subseteq [\![P]\!]_{\textit{source memory model}}$$

i.e.,

every outcome that is allowed compile(P) under the target memory model is also an allowed outcome for P under the source memory model

- The compiler is allowed to "lose" behaviors.
- Focusing on the weak memory concurrency aspect, we use high-level programming languages in both sides.
- We are also ignoring non-terminating programs.

Compilation schemes

C/C++11 to various architectures:

`https://www.cl.cam.ac.uk/~pes20/cpp/cpp0xmappings.html`

Compilation correctness

$$[\![\text{compile}_{\text{SC}\to\text{TSO}}(P)]\!]_{\text{TSO}} \quad \subseteq \quad [\![P]\!]_{\text{SC}}$$

Two possible compilation schemes:

- place a TSO fence after every write; or
- place a TSO fence before every read

Both schemes ensure there is a fence between a write and a subsequent read.

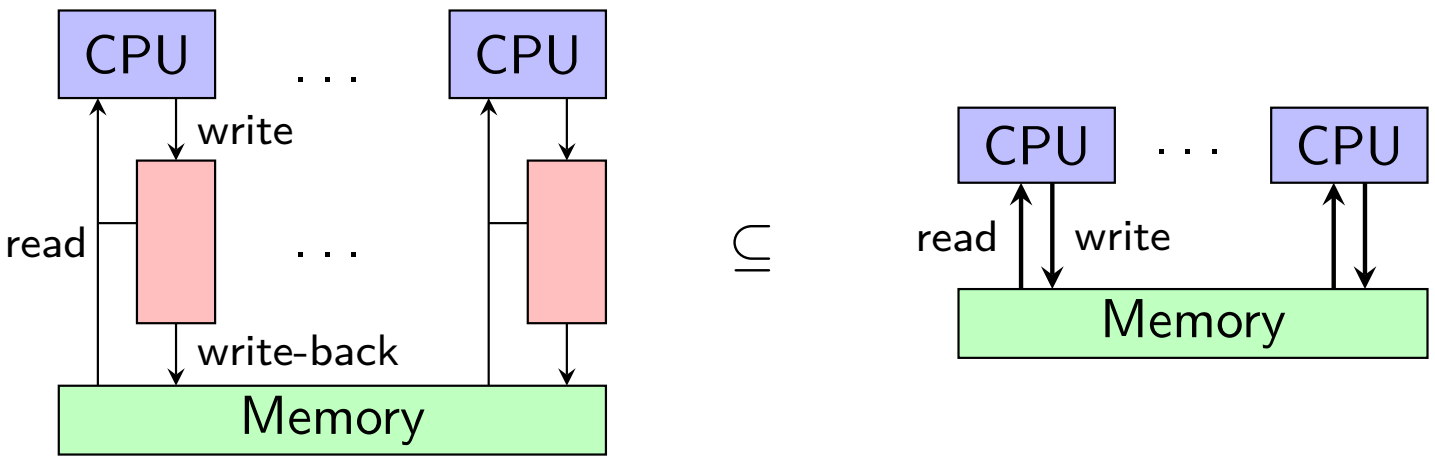
Example: Compilation of SB

$$\text{compile}_{\text{SC}\to\text{TSO}} \begin{pmatrix} x := 1; \\ a := y \end{pmatrix} \Bigg\| \begin{pmatrix} y := 1; \\ b := x \end{pmatrix} \quad = \quad \begin{matrix} x := 1; \\ \textbf{fence}; \\ a := y \end{matrix} \Bigg\| \begin{matrix} y := 1; \\ \textbf{fence}; \\ b := x \end{matrix}$$

Correctness of compilation of SC to TSO

Consider the first scheme (fence after writes).

- We show that every outcome of $\text{compile}_{\text{SC}\to\text{TSO}}(P)$ under TSO is an outcome of P under SC.
- We will use the operational semantics of SC and TSO.



Correctness of compilation of SC to TSO

- To show: $[\![\text{compile}_{\text{SC}\to\text{TSO}}(P)]\!]_{\text{TSO}} \quad \subseteq \quad [\![P]\!]_{\text{SC}}$
- To simplify the argument, we split the proof in two steps.
- First, just insert a fence after each write and show:

$$[\![\text{compile}_{\text{SC}\to\text{TSO}}(P)]\!]_{\text{SC}} \quad \subseteq \quad [\![P]\!]_{\text{SC}}$$

  This is trivial because the SC machine simply ignores fences:

$$M \xrightarrow{i:\mathtt{F}} M$$

- Then, we show that $[\![P_0]\!]_{\text{TSO}} \quad \subseteq \quad [\![P_0]\!]_{\text{SC}}$ for every program P_0 that has a fence after every write.

Simulation for the second step

Assumption on the program

For every $i \in \mathrm{Tid}$, if $P_0(i), s_0 \to^* \xrightarrow{\mathtt{W}(x,v)} c, s$, then $c, s \xrightarrow{\varepsilon}{}^* \xrightarrow{\mathtt{F}} \_$.

(notation: $q \xrightarrow{I} \_ \triangleq \exists q'.\ q \xrightarrow{I} q'$)

A simulation relation:

$\langle P, S, M, B\rangle\ \mathcal{R}\ \langle P_{\mathrm{SC}}, S_{\mathrm{SC}}, M_{\mathrm{SC}}\rangle$ if the following hold:

- for every $i \in \mathrm{Tid}$, one of the following holds:
  - $P(i) = P_{\mathrm{SC}}(i)$, $S(i) = S_{\mathrm{SC}}(i)$, and $B(i) = \epsilon$
  - $P(i), S(i) \xrightarrow{\varepsilon}{}^* \xrightarrow{\mathtt{F}} \_$ and there exist x, v such that $B(i) = \langle x, v\rangle$ and $P_{\mathrm{SC}}(i), S_{\mathrm{SC}}(i) \xrightarrow{\mathtt{W}(x,v)} \xrightarrow{\varepsilon}{}^* P(i), S(i)$
- $M = M_{\mathrm{SC}}$

- Show that $\mathcal{R}$ is a simulation relation.
- Deduce that every outcome of P_0 under TSO is also an outcome under SC.

Correctness of compilation of SC to TSO

The correctness of the alternative compilation scheme (fence before reads) can be proved similarly.

In fact, it suffices to ensure the following property:

Assumption on the program

$\neg(P_0(i), s_0 \rightarrow^* \xrightarrow{\mathtt{W}(x,v)} \xrightarrow{\varepsilon}^* \xrightarrow{\mathtt{R}(x',v')} \_)$ for every i, x, v, x', v'.

Exercise

Let P_0 be a program satisfying the assumption above.

- Devise a simulation relation relating executions of P_0 under TSO with those under SC.
- Show that every outcome of P_0 under TSO is also an outcome under SC.

Compilation correctness via a declarative semantics

Compilation correctness

$$[\![\text{compile}(P)]\!]_{\textit{target memory model}} \subseteq [\![P]\!]_{\textit{source memory model}}$$

Alternatively, compilation correctness can be proven using declarative semantics.

Definition (Allowed outcome under a declarative model)

An outcome O is *allowed* for a program P under X if there exists an execution graph G such that:

- G is an execution graph of P with outcome O.
- G is X-consistent.

Given an execution graph of compile(P) which is consistent according to the target memory model, we have to show an execution graph of P with the same outcome which is consistent according to the source memory model.

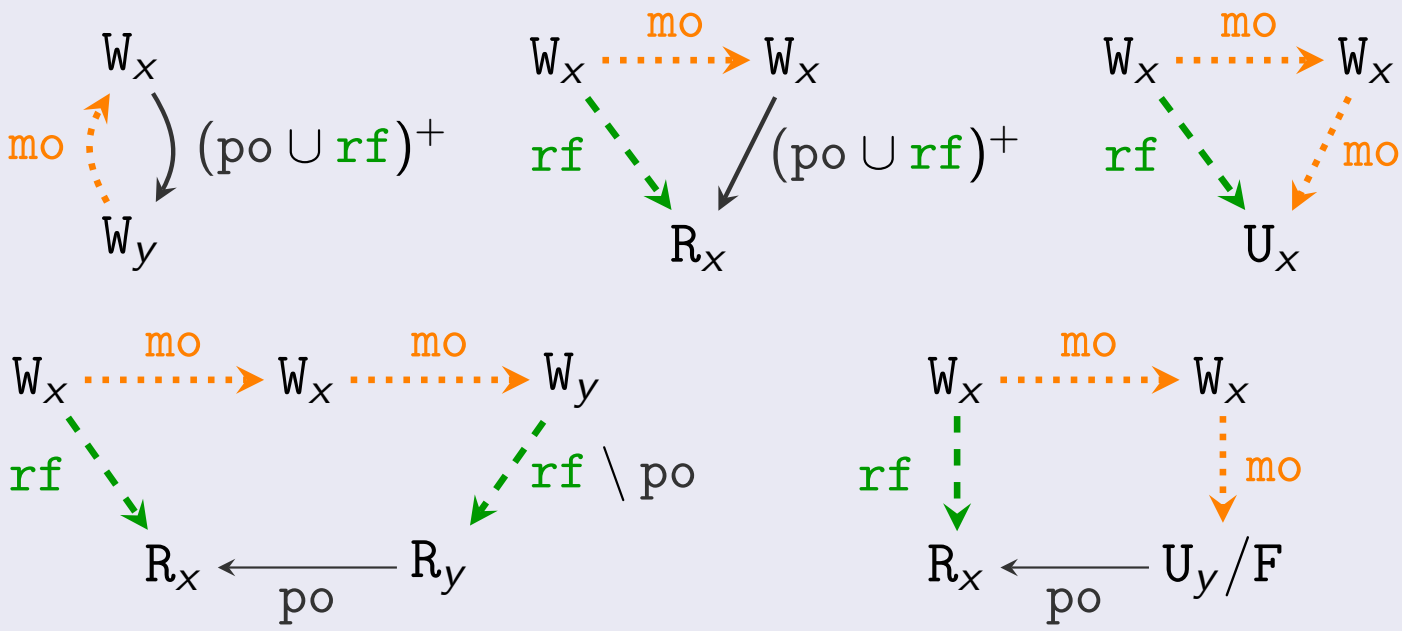
Declarative semantics for TSO

Owens et al. [TPHOLs '09] provided a declarative semantics for TSO, which is equivalent to the following:

Definition

An execution graph is TSO-*consistent* if it is complete,

- $(\mathtt{po} \cup \mathtt{rf})^+$ is irreflexive, and
- there exists a total order $\mathtt{mo}$ on $\mathtt{W} \cup \mathtt{F}$, such that *none* of the following patterns occurs:



Declarative semantics for TSO

Theorem

An outcome O is allowed for a program P under TSO *if there exists a* TSO*-consistent execution graph of P with outcome O.*

Compilation of SC to TSO via the declarative semantics

- As before we may assume that the source program has fences and the SC declarative semantics simply ignores them.

Assumption on the program

For every c, s such that $P_0(i), S_0 \to^* c, s$:
if $c, s \xrightarrow{\mathtt{W}(x,v)} \to^* \xrightarrow{\mathtt{R}(x',v')} c', s'$ then
$c, s \xrightarrow{\mathtt{W}(x,v)} \to^* \xrightarrow{\mathtt{F}} \to^* \xrightarrow{\mathtt{R}(x',v')} c', s'$

- Given a TSO-consistent execution graph G of P_0, we have to construct an SC-consistent execution graph of P_0 with the same outcome.
- We will take the same execution graph G.
- We have $[\mathtt{W}]; G.\mathtt{po}; [\mathtt{R}] \subseteq G.\mathtt{po}; [\mathtt{F}]; G.\mathtt{po}$.
- It remains to show that G is SC-consistent.

Definition (Alternative SC definition)

An execution graph G is SC-*consistent* if the following hold:

- G is complete
- There exists a modification order $\mathtt{mo}$ for G such that $G.\mathrm{po} \cup G.\mathtt{rf} \cup \mathtt{mo} \cup \mathtt{rb}$ is acyclic where:
  - $\mathtt{rb} \triangleq G.\mathtt{rf}^{-1}; \mathtt{mo} \setminus \mathrm{id}$ (from-reads / reads-before)

- Take $\mathtt{mo}_{\mathsf{SC}} = \bigcup_{x \in \mathsf{Loc}} [\mathtt{W}_x]; \mathtt{mo}_{\mathsf{TSO}}; [\mathtt{W}_x]$.
- Suppose that $G.\mathrm{po} \cup G.\mathtt{rf} \cup \mathtt{mo}_{\mathsf{SC}} \cup \mathtt{rb}$ is cyclic.
- Then, $G.\mathrm{po} \cup G.\mathtt{rf} \cup \mathtt{mo}_{\mathsf{TSO}} \cup \mathtt{rb}$ is cyclic.
- Consider a cycle of minimal length.
- In a minimal cycle, there are at most two events in $\mathtt{W} \cup \mathtt{F}$.
- Analyze all such cycles. . .

Compilation of RA to TSO

Compilation correctness

$$[\![\text{compile}_{\text{RA}\to\text{TSO}}(P)]\!]_{\text{TSO}} \quad \subseteq \quad [\![P]\!]_{\text{RA}}$$

Trivial:

- Compilation is identity.
- It suffices to note that TSO-consistency implies RA-consistency!

Theorem (Reminder)

Let $\texttt{mo}$ *be a modification order for an execution graph* G*.* G *is* RA*-consistent wrt* $\texttt{mo}$ *iff the following hold:*

- $(\texttt{po} \cup \texttt{rf})^+$ *is irreflexive.* *(no-future-read)*
- $\texttt{mo}; (\texttt{po} \cup \texttt{rf})^+$ *is irreflexive.* *(coherence-ww)*
- $\texttt{rf}^{-1}; \texttt{mo}; (\texttt{po} \cup \texttt{rf})^+$ *is irreflexive.* *(coherence-wr)*
- $\texttt{rf}^{-1}; \texttt{mo}; \texttt{mo}$ *is irreflexive.* *(rmw-atomicity)*

Compilation to Power/ARM

Power and ARM are given by both declarative and operational models. For the compilation proofs, the declarative models are more convenient.

Power Herding cats: modelling, simulation, testing, and data mining for weak memory. Jade Alglave, Luc Maranget, Michael Tautschnig, TOPLAS 2017.

ARMv8.2 aarch64.cat on GitHub

Since, however, the models are much more complex, compilation proofs are error-prone. See, *e.g.*,

- Synchronising C/C++ and POWER. Susmit Sarkar, Kayvan Memarian, Scott Owens, Mark Batty, Peter Sewell, Luc Maranget, Jade Alglave, Derek Williams, PLDI 2012.
- Repairing sequential consistency in C/C++11. Ori Lahav, Viktor Vafeiadis, Jeehoon Kang, Chung-Kil Hur, Derek Dreyer, PLDI 2017.

Power declarative model

An execution G is Power-*consistent* if there exists a modification order $\texttt{mo}$ for G such that the following hold:

1. $\texttt{hb}$ is acyclic. (*no-thin-air*)
2. $\texttt{po}|_{\texttt{loc}} \cup \texttt{rf} \cup \texttt{rb} \cup \texttt{mo}$ is acyclic. (*SC-per-loc*)
3. $(\texttt{rb} \setminus \texttt{po}); \texttt{prop}; \texttt{hb}^*$ is irreflexive. (*observation*)
4. $\texttt{mo} \cup \texttt{prop}$ is acyclic. (*propagation*)
5. $\texttt{mo}; [\texttt{RMW}]; \texttt{po}; [\texttt{RMW}]$ is acyclic.

where:

- $\texttt{sync} = \texttt{po}; [\texttt{F}^{\texttt{sync}}]; \texttt{po}$ and $\texttt{lwsync} = \texttt{po}; [\texttt{F}^{\texttt{lwsync}}]; \texttt{po}$
- $\texttt{fence} = \texttt{sync} \cup ([\texttt{R}]; \texttt{lwsync}; [\texttt{RW}] \cup ([\texttt{W}]; \texttt{lwsync}; [\texttt{W}]))$ (*fence order*)
- $\texttt{rb} = (\texttt{rf}^{-1}; \texttt{mo}) \setminus [\texttt{E}]$ (*read before*)
- $\texttt{rfe} = \texttt{rf} \setminus \texttt{po}$ (*external reads-from*)
- $\texttt{ppo} = \ldots$ (*preserved program order*)
- $\texttt{hb} = \texttt{ppo} \cup \texttt{fence} \cup \texttt{rfe}$ (*happens-before*)
- $\texttt{prop}_1 = [\texttt{W}]; \texttt{rfe}^?; \texttt{fence}; \texttt{hb}^*; [\texttt{W}]$
- $\texttt{prop}_2 = ((\texttt{mo} \cup \texttt{rb}) \setminus \texttt{po})^?; \texttt{rfe}^?; (\texttt{fence}; \texttt{hb}^*)^?; \texttt{sync}; \texttt{hb}^*$
- $\texttt{prop} = \texttt{prop}_1 \cup \texttt{prop}_2$ (*propagation relation*)

Exercise: Correctness of compilation of SC to TSO

Use a simulation argument to prove the correctness of compilation from SC to TSO, for a compilation scheme that places a TSO fence between every write and read.

That is, let P_0 be a program such that $\neg(P_0(i), s_0 \rightarrow^* \xrightarrow{\mathtt{W}(x,v)} \xrightarrow{\varepsilon}^* \xrightarrow{\mathtt{R}(x',v')} \_)$ for every i, x, v, x', v'.

- Devise a simulation relation relating executions of P_0 under TSO with those under SC.
- Conclude that every outcome of P_0 under TSO is also an outcome under SC.

Exercise: Correctness of compilation of RA to Power

The compilation scheme of RA to Power that places a *lightweight fence* before every write and after every read.
Assuming no RMW's, prove the correctness of this compilation scheme.
You may prefer to use the following declarative model for WeakPower (which is clearly weaker than the model for Power presented before):

Definition

An execution G is WeakPower-*consistent* if there exists a modification order $\texttt{mo}$ for G such that the following hold:

1. $\texttt{hb}$ is acyclic. (*no-thin-air*)
2. $\texttt{po}|_{\texttt{loc}} \cup \texttt{rf} \cup \texttt{rb} \cup \texttt{mo}$ is acyclic. (*SC-per-loc*)
3. $(\texttt{rb} \setminus \texttt{po}); \texttt{prop}; \texttt{hb}^*$ is irreflexive. (*observation*)
4. $\texttt{mo} \cup \texttt{prop}$ is acyclic. (*propagation*)

where:

- $\texttt{fence} = [\texttt{R}]; \texttt{po}; [\texttt{F}^{\texttt{lwsync}}]; \texttt{po}; [\texttt{RW}] \cup [\texttt{W}]; \texttt{po}; [\texttt{F}^{\texttt{lwsync}}]; \texttt{po}; [\texttt{W}]$ (*fence order*)
- $\texttt{rb} = (\texttt{rf}^{-1}; \texttt{mo}) \setminus \mathrm{id}$ (*reads before*)
- $\texttt{rfe} = \texttt{rf} \setminus \texttt{po}$ (*external reads-from*)
- $\texttt{hb} = \texttt{fence} \cup \texttt{rfe}$ (*(weak) happens-before*)
- $\texttt{prop} = [\texttt{W}]; \texttt{rfe}^?; \texttt{fence}; \texttt{hb}^*; [\texttt{W}]$ (*(weak) propagation relation*)