Bridging the Gap between Programming Languages and Hardware Weak Memory Models

Anton Podkopaev  Ori Lahav  Viktor Vafeiadis
Bridging the Gap between PL and Hardware Weak MMs
Bridging the **Gap** between **PL** and **Hardware** Weak MM

| Programming Language | Hardware |
|----------------------|----------|

∀ \( P \in \text{Syntax(PL)} \). 

\( J \) \( P \) \( K \) PL \( J \) compile \( P \) \( K \) HW \( J - K \) \{ PL, HW \} is Memory Model
Bridging the Gap between PL and Hardware Weak MMs
**Bridging the Gap between PL and Hardware** Weak MMs

![Diagram](image)

- Programming Language
- Correct Compiler
- Hardware

∀ \( P \in \text{Syntax (PL)} \). \( J \_P \_K \_PL \) compile \( (P) \_K \_HW \) \( J \_K \{-PL, HW\} \) is Memory Model
Bridging the Gap between PL and Hardware Weak MMs

∀\(P \in \text{Syntax}(PL)\).

\[[\text{compile}(P)]_{HW} \subseteq [P]_{PL}\]
Bridging the Gap between PL and Hardware Weak MMs

∀P ∈ Syntax(PL).

\[ [[\text{compile}(P)]_{HW}] \subseteq [[P]_{PL}] \]

\[ [\_\_\_]_{\{PL,HW\}} \text{ is Memory Model} \]
Bridging the Gap between PL and Hardware Weak MMs

∀P ∈ Syntax(PL).

\[
\begin{align*}
\text{compile}(P)_{HW} & \subseteq [P]_{PL} \\
\end{align*}
\]

\([-\_]\)\(_{\{PL,HW\}}\) is Memory Model
Strong (SC) MM allows $a = b = 1$

Values

\[
\begin{align*}
a &= \bot; \\
b &= \bot; \\
\end{align*}
\]

Memory

\[
\begin{align*}
[x] &\leftarrow 0; \\
[y] &\leftarrow 0
\end{align*}
\]
\[ a := \left[ x \right]; \quad [y] := 1; \quad b := \left[ y \right]; \quad [x] := b; \]

**Values**

\[ a = \bot; \quad b = \bot \]

**Memory**

\[ [x] \leftarrow 0; \quad [y] \leftarrow 0 \]
\[
\begin{align*}
a & := \llbracket x \rrbracket; \\
\llbracket y \rrbracket & := 1; \\
b & := \llbracket y \rrbracket; \\
\llbracket x \rrbracket & := \llbracket b \rrbracket;
\end{align*}
\]

**Values**

\[a = 0; \quad b = \bot\]

**Memory**

\[\llbracket x \rrbracket \leftarrow 0; \quad \llbracket y \rrbracket \leftarrow 0\]
$a := [x];$

$[y] := 1;$

$b := [y];$

$x := b;$

Values

\[ a = 0; \quad b = \bot \]

Memory

\[ [x] \leftarrow 0; \quad [y] \leftarrow 1 \]
```
a := [x];  b := [y];
[y] := 1;  [y] := 1;
[x] := b;
[x] := b;
```

Values
\[a = 0; \ b = 1\]

Memory
\[[x] \leftarrow 0; \ [y] \leftarrow 1\]
Strong (SC) MM disallows $a = b = 1$!

Values
\[
a = 0; \quad b = 1
\]

Memory
\[
[x] \leftarrow 1; \quad [y] \leftarrow 1
\]
Strong (SC) MM

\[
\begin{align*}
    a & := \lfloor x \rfloor; \quad & b & := \lfloor y \rfloor; \\
    \lfloor y \rfloor & := 1; \quad & \lfloor x \rfloor & := b;
\end{align*}
\]

Values
\[
a = 0; \quad b = 1
\]

Memory
\[
\lfloor x \rfloor \leftarrow 1; \quad \lfloor y \rfloor \leftarrow 1
\]

ARM and POWER weak MMs allow \( a = b = 1! \)
Strong (SC) MM disallows $a = b = 1$

\[a := [x]; \quad [y] := 1;\]
\[b := [y]; \quad [x] := b;\]

Values
\[a = 0; \quad b = 1\]

Memory
\[[x] \leftarrow 1; \quad [y] \leftarrow 1\]
Strong (SC) MM disallows $a = b = 1$

\[
\begin{align*}
  a & := \langle x \rangle; \\
  [y] & := 1; \\
  b & := \langle y \rangle; \\
  [x] & := b;
\end{align*}
\]

ARM and POWER weak MMs allow $a = b = 1!$

**Values**

\[
\begin{align*}
  a & = 0; \\
  b & = 1
\end{align*}
\]

**Memory**

\[
\begin{align*}
  [x] & \leftarrow 1; \\
  [y] & \leftarrow 1
\end{align*}
\]
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

plv.mpi-sws.org/imm/

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

1. Declarative
2. Preserves syntactic dependencies \( (\text{deps} \cup \text{rf}) \) is acyclic
3. Uses C11-style coherence \( (\text{hb}; \text{eco}^? \) is irreflexive
4. Non-multicopy-atomic w/o mutually recursive relations
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

plv.mpi-sws.org/imm/

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

- Declarative
- Preserves syntactic dependencies ($\text{deps} \cup \text{rf}$ is acyclic)
- Uses C11-style coherence ($\text{hb} \cup \text{eco}$ is irreflexive)
- Non-multicopy-atomic w/o mutually recursive relations

plv.mpi-sws.org/imm/

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

IMM

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

1. Declarative
2. Preserves syntactic dependencies \((\text{deps} \cup \text{rf})\) is acyclic

3. Uses C11-style coherence \((\text{hb} \cap \text{eco}\) is irreflexive

4. Non-multicopy-atomic w/o mutually recursive relations

(R)C11

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
(Declarative) Executions in IMM

\[ a := [x]; \quad b := [y]; \]
\[ [y] := 1; \quad [x] := b; \]
(Declarative) Executions in IMM

\[
\begin{align*}
  a & := [x]; & b & := [y]; \\
  [y] & := 1; & [x] & := b;
\end{align*}
\]
(Declarative) Executions in IMM

\[
a := [x]; \quad b := [y];
\]
\[
[y] := 1; \quad [x] := b;
\]
(Declarative) Executions in IMM

\[ a := [x]; \quad \quad b := [y]; \]
\[ [y] := 1; \quad \quad [x] := b; \]

Axioms:

1. \( \text{data} \cup \text{rf} \) is acyclic

...
Bridging the Gap between PL and Hardware Weak MMs

1. Declarative
2. Preserves syntactic dependencies ($\text{deps} \cup \text{rf}$ is acyclic)
3. Uses C11-style coherence ($\text{hb}; \text{eco}$ is irreflexive)
4. Non-multicopy-atomic w/o mutually recursive relations

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

1. Declarative
2. Preserves syntactic dependencies
\[(\text{deps} \cup \text{rf}) \text{ is acyclic}\]
3. Uses C11-style coherence
\[(\text{hb}; \text{eco}?) \text{ is irreflexive}\]
4. Non-multicopy-atomic

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

1. Declarative
2. Preserves syntactic dependencies \((\text{deps} \cup \text{rf})\) is acyclic
3. Uses C11-style coherence \((\text{hb}; \text{eco}^?\) is irreflexive)

(R)C11

IMM

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

1. Declarative
2. Preserves syntactic dependencies 
   \((\text{deps} \cup \text{rf})\) is acyclic \)
3. Uses C11-style coherence 
   \((\text{hb}; \text{eco}^?\) is irreflexive \)
4. Non-multicopy-atomic 
   w/o mutually recursive relations

Promise

(R)C11

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

IMM

x86-TSO
ARMv7
ARMv8.3
RISC-V
POWER

1. Declarative
2. Preserves syntactic dependencies (\(\text{deps} \cup \text{rf}\) is acyclic)
3. Uses C11-style coherence (\(\text{hb}; \text{eco}\) is irreflexive)
4. Non-multicopy-atomic w/o mutually recursive relations
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

IMM

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

IMM

- x86-TSO
- ARMv7
- ARMv8.3
- RISC-V
- POWER

Thank you!
(Operational) Execution in Promise

\[
\begin{align*}
a & := [x]; \\
[y] & := 1; \\
b & := [y]; \\
x & := b;
\end{align*}
\]

Values
\[
a = \bot; \quad b = \bot
\]
(Operational) Execution in Promise

\[ a := [x]; \quad b := [y]; \]
\[ [y] := 1; \quad [x] := b; \]

Values
\[ a = \bot; \quad b = \bot \]
(Operational) Execution in Promise

\[
a := [x]; \\
[y] := 1;
\]

Promised

\[
b := [y]; \\
x := b;
\]

Values
\[
a = \bot; b = \bot
\]
(Operational) Execution in Promise

\[
\begin{align*}
\text{Requires certification} & \quad a := [x]; \quad b := [y]; \\
\text{Promised} & \quad [y] := 1; \quad [x] := b;
\end{align*}
\]

Values
\[
a = \bot; \quad b = \bot
\]
(Operational) Execution in Promise

\[
\begin{align*}
  a & := [x]; \\
  [y] & := 1; \\
  b & := [y]; \\
  [x] & := b;
\end{align*}
\]

Promised

Values
\[
  a = \bot; b = 1
\]
(Operational) Execution in Promise

\[ a := [x]; \]
\[ [y] := 1; \]
\[ b := [y]; \]
\[ [x] := b; \]

Promised

Values
\[ a = \bot; \quad b = 1 \]
(Operational) Execution in Promise

\[
\begin{align*}
  a & := [x]; \\
  [y] & := 1; \\
  b & := [y]; \\
  [x] & := b;
\end{align*}
\]

Promised

Values

\[
a = 1; \quad b = 1
\]
(Operational) Execution in Promise

\[a := [x]; \quad [y] := 1; \quad [x] := [y]; \quad b := [y]; \quad [x] := b;\]

\[\text{Values}\]
\[a = 1; b = 1\]
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER
How to prove correctness of compilation?
How to prove correctness of compilation? Simulation
How to prove correctness of compilation? *Simulation*

How to *simulate* graphs?
How to prove correctness of compilation? **Simulation**

How to simulate graphs? **Traverse** in proper order!
Traversal of IMM execution

\[ a := [x]; \]
\[ [y] := 1; \]

Promised

\[ b := [y]; \]
\[ [x] := b; \]
Promise → IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:

\[ G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle \]
Promise \rightarrow IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:

\[ G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle \]

2. Completeness of traversal:

\[ \forall G \in \llbracket P \rrbracket_{IMM}. \ G \vdash \text{init}_{Traverse} \rightarrow^* \langle G.\text{Events}, G.\text{Writes} \rangle \]
Promise $\rightarrow$ IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:

$$G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle$$

2. Completeness of traversal:

$$\forall G \in \llbracket P \rrbracket_{IMM}. \ G \vdash \text{init}_{\text{Traverse}} \rightarrow^* \langle G.\text{Events}, G.\text{Writes} \rangle$$

3. Simulation theorems:

- $\text{init}_{\text{Traverse}}$ simulated by $\text{init}_{\text{Promise}}$
- $\text{traverse}$ simulated by $\text{promise}$
- $\text{traverse'}$ simulated by $\exists \text{promise'}$
Promise → IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:

   \[ G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle \]

2. Completeness of traversal:

   \[ \forall G \in [P]_{IMM}. \ G \vdash \text{init}_{\text{Traverse}} \rightarrow^* \langle G.\text{Events}, G.\text{Writes} \rangle \]

3. Simulation theorems:

   \[ \text{init}_{\text{Traverse}} \quad \text{simulated by} \quad \text{init}_{\text{Promise}} \]

   \[ \text{traverse} \quad \text{simulated by} \quad \text{promise} \]

   \[ \text{traverse'} \quad \text{simulated by} \quad \exists \text{promise'} \]
Promise → IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:
   \[ G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle \]

2. Completeness of traversal:
   \[ \forall G \in \mathbb{[P]}_{IMM}. \ G \vdash \text{init}_{\text{Traverse}} \rightarrow^* \langle G.\text{Events}, G.\text{ Writes} \rangle \]

3. Simulation theorems:

   Promise’s certification

   \[
   \begin{array}{c}
   \text{traverse} \quad \overset{\text{simulated by}}{\longrightarrow} \quad \text{promise} \\
   \downarrow \\
   \text{traverse}' \quad \overset{\text{simulated by}}{\longrightarrow} \quad \exists \text{promise}'
   \end{array}
   \]
Promise → IMM compilation correctness proof

1. Operational semantics of IMM’s traversal:
   \[ G \vdash \langle C, I \rangle \rightarrow \langle C', I' \rangle \]

2. Completeness of traversal:
   \[ \forall G \in [P]_{IMM}. \ G \vdash \text{init}_{\text{Traverse}} \rightarrow^* \langle G.\text{Events}, G.\text{Writes} \rangle \]

3. Simulation theorems:
   Promise’s certification via traversal of certification graph
   \[ \text{traverse} \sim \text{promise} \sim \exists \text{promise}' \]
Traversal of IMM execution

\[ a := [x]; \quad | \quad b := [y]; \]

\[ [y] := 1; \quad | \quad [x] := b; \]

\[
\begin{array}{c}
R_x1 \\
W_x1
\end{array}
\quad \begin{array}{c}
R_y1 \\
W_y1
\end{array}
\]

Covered

Issued
Traversal of IMM execution

\[ a := [x]; \]
\[ [y] := 1; \]
\[ b := [y]; \]
\[ [x] := b; \]
Traversal of IMM execution

\[
\begin{align*}
  a &:= [x]; \\
  [y] &:= 1; \\
  b &:= [y]; \\
  [x] &:= b;
\end{align*}
\]
Traversal of IMM execution

\[
\begin{align*}
a &:= [x]; \\
[y] &:= 1; \\
[x] &:= b;
\end{align*}
\]
Traversal of IMM execution

\[ a := [x]; \]
\[ [y] := 1; \]
\[ b := [y]; \]
\[ [x] := b; \]

Promised

Covered

Issued
Traversal of IMM execution

\[ a := [x]; \quad | \quad b := [y]; \]
\[ [y] := 1; \quad | \quad [x] := b; \]

Promised

Covered

Issued
Traversal of IMM execution

\[ a := [x]; \quad b := [y]; \]
\[ y := 1; \quad x := b; \]
Bridging the Gap between PL and Hardware Weak MMs

Promise

(R)C11

IMM

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

Promise

WeakestMO

(R)C11

[Chakraborty and Vafeiadis, 2019]
Bridging the Gap between PL and Hardware Weak MMs

CompCert \[\rightarrow\] Weak MMs

Promise

WeakestMO

(R)C11

[Chakraborty and Vafeiadis, 2019]

\[\text{IMM} \]

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
Bridging the Gap between PL and Hardware Weak MMs

CompCert \rightarrow \text{Weak MMs}

Promise
WeakestMO \rightarrow \text{IMM}
(R)C11

\text{IMM} \rightarrow \text{x86-TSO}
\text{ARMv7}
\text{ARMv8.3}
\text{RISC-V}
\text{POWER}

[Chakraborty and Vafeiadis, 2019]

plv.mpi-sws.org/imm/ Thank you!
Chakraborty, S. and Vafeiadis, V. (2019). Grounding thin-air reads with event structures. In *POPL 2019*. ACM.

Kang, J., Hur, C.-K., Lahav, O., Vafeiadis, V., and Dreyer, D. (2017). A promising semantics for relaxed-memory concurrency. In *POPL 2017*. ACM.
Backup slides
Def. $G$ is called IMM-consistent if the following hold:

1. $\text{codom}(G.\text{rf}) = G.\text{R}$.  
2. For every location $\ell \in \text{Loc}$, $G.\text{co}$ totally orders $G.\text{W}_\ell$.  
3. $G.\text{rmw} \cap (G.\text{fre} ; G.\text{coe}) = \emptyset$.  
4. $G.\text{hb} ; G.\text{eco}$ is irreflexive.  
5. $G.\text{ar}$ is acyclic.

\[\text{ar} \triangleq \text{rfe} \cup \text{bob} \cup \text{ppo} \cup \text{detour} \cup \text{psc} \cup [\text{W}_{\text{strong}}] ; \text{po} ; [\text{W}]\]
\[\text{bob} \triangleq \text{po} ; [\text{W}_{\text{rel}}] \cup [\text{R}_{\text{acq}}] ; \text{po} \cup \text{po} ; [\text{F}] \cup [\text{F}] ; \text{po} \cup [\text{W}_{\text{rel}}] ; \text{po}_{\text{loc}} ; [\text{W}]\]
\[\text{ppo} \triangleq [\text{R}] ; (\text{deps} \cup \text{rfi})^+ ; [\text{W}]\]
\[\text{deps} \triangleq \text{data} \cup \text{ctrl} \cup \text{addr} ; \text{po}^? \cup \text{casdep} \cup [\text{R}_{\text{ex}}] ; \text{po}\]
Traversals definition

\[ a \in \text{Next}(G, C) \cap \text{Coverable}(G, C, I) \quad \Rightarrow \quad G \vdash \langle C, I \rangle \rightarrow \langle C \cup \{a\}, I \rangle \]

\[ w \in \text{Issuable}(G, C, I) \setminus I \quad \Rightarrow \quad G \vdash \langle C, I \rangle \rightarrow \langle C, I \cup \{w\} \rangle \]

**Def.** \( w \in \text{Issuable}(G, C, I) \) iff \( w \in G.W \) and the following hold:

- \( \text{dom}([G.W^{re}] ; G.po|_{G.loc} \cup [G.F] ; G.po) ; [w]) \subseteq C \)
- \( \text{dom}((G.detour \cup G.rf) ; G.ppo ; [w]) \subseteq I \)
- \( \text{dom}((G.detour \cup G.rf) ; [G.R^{acq}] ; G.po ; [w]) \subseteq I \)
- \( \text{dom}([G.W_{strong}] ; G.po ; [w]) \subseteq I \)

**Def.** \( e \in \text{Coverable}(G, C, I) \) iff \( e \in G.E, \text{dom}(G.po ; [e]) \subseteq C \) and either

(i) \( e \in G.W \cap I \);
(ii) \( e \in G.R \) and \( \text{dom}(G.rf ; [e]) \subseteq I \);
(iii) \( e \in G.F^{\neg sc} \);
(iv) \( e \in G.F^{sc} \) and \( \text{dom}(G.sc ; [e]) \subseteq C \).
Mistake in Kang et al.'s compilation to POWER correctness proof

Consistent in Strong-POWER.
Not consistent in the promise-free declarative model of [Kang et al., 2017].
Promise → IMM compilation of RMWs

\[
a := [y]^{rlx} // 1
\]
\[
[z]^{rlx} := a
\]
\[
b := [z]^{rlx} // 1
\]
\[
c := FADD^{rlx,rel}_{strong}(x, 1) // 0
\]
\[
[y]^{rlx} := c + 1
\]