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 MMs

| Programming Language | Hardware |
Bridging the Gap between PL and Hardware Weak MMs

Programming Language  Compiler  Hardware

∀ \text{PL} \in \text{Syntax} \implies \text{PL} \text{compile} \rightarrow \text{HW} \rightarrow \{\text{PL, HW}\} \text{is Memory Model}
Bridging the Gap between PL and Hardware Weak MMs
Bridging the Gap between PL and Hardware Weak MMs

∀\(P \in \text{Syntax}(PL)\).

\[
\begin{align*}
&[\text{compile}(P)]_{\text{HW}} \\
&[P]_{\text{PL}}
\end{align*}
\]
**Bridging the Gap between PL and Hardware Weak MMs**

∀ \( P \in Syntax(PL) \).

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

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

∀\(P \in Syntax(PL)\).

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

\([-]\)\(_{\{\text{PL}, \text{HW}\}}\) is Memory Model
\[
\begin{align*}
a & := \{x\}; & b & := \{y\}; \\
\{y\} & := 1; & \{x\} & := b;
\end{align*}
\]

Values: \( a = \bot; \ b = \bot \)

Memory: \( [x] \leftarrow 0; \ [y] \leftarrow 0 \)
\[
\begin{align*}
\overrightarrow{a} &:= [x]; & \overrightarrow{b} &:= [y]; \\
[y] &:= 1; & [x] &:= b;
\end{align*}
\]

Values

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

Memory

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

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

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

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

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

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

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

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

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

Values

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

Memory

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

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

Values

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

Memory

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

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

ARM and POWER weak MMs allow \( a = b = 1 \)!

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

- **Memory**
  \[
  [x] \leftarrow 1; [y] \leftarrow 1
  \]
Bridging the Gap between PL and Hardware Weak MMs

**Promise**

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

(R)C11
Bridging the Gap between PL and Hardware Weak MMs

**Promise**

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

**Technology Platforms**

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

- **Promise**
- **(R)C11**

**Features**:
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

θplv.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

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

(R)C11

\[
\text{hb; eco? is irreflexive}
\]

4. Non-multicopy-atomic

\[\text{w/o mutually recursive relations}\]

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!
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

Thank you!
(Declarative) Executions in IMM

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

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

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

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

Axioms:

1. data ∪ rf is acyclic

\[
\begin{array}{c}
\text{Rx1} & \text{Ry1} \\
\text{po} & \text{data} \\
Wx1 & Wy1
\end{array}
\]
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 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 \((\text{hb}; \text{eco}\) is irreflexive\)
4. Non-multicopy-atomic \(\text{w/o mutually recursive relations}\)

Promise

(R)C11

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

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

Promise

(R)C11

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

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

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

Plots:
- 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!
Bridging the Gap between PL and Hardware Weak MMs

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

hb; eco? is irreflexive

deps ∪ rf is acyclic

Declarative
Preserves syntactic dependencies
Uses C11-style coherence
Non-multicopy-atomic w/o mutually recursive relations
(Operational) Execution in Promise

\[ a := [x]; \quad \text{Values} \quad \begin{array}{c} a = \bot; \\ b = \bot \end{array} \]

\[ b := [y]; \quad \text{[y]} := 1; \quad \text{[x]} := b; \]
(Operational) Execution in Promise

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

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

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

Values
\[
\begin{align*}
a &= \bot; \\
b &= \bot
\end{align*}
\]
(Operational) Execution in Promise

Requires certification

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

Promised

Values
\[
\begin{align*}
a &= \bot; \\
b &= \bot
\end{align*}
\]
(Operational) Execution in Promise

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

Promised

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

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

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

Promised

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

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

Promised

Values

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

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

Values

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

Promise

(R)C11

IMM

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!
Traversals of IMM execution

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

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

3. Simulation theorems:
   \[ \text{init}_{\text{Traverse}} \sim \text{init}_{\text{Promise}} \]
   \[ \text{traverse} \sim \text{promise} \]
   \[ \text{traverse}' \sim \exists \text{promise}' \]
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:

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

3. Simulation theorems:

init_{\text{Traverse}} \quad \text{simulated by} \quad \text{init}_{\text{Promise}}

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

traverse' \quad \text{simulated by} \quad \exists \text{promise'}
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:

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

3. Simulation theorems:

   Promise’s certification

   traverse $\xrightarrow{\text{simulated by}}$ promise

   traverse $\downarrow$

   traverse’ $\xrightarrow{\text{simulated by}}$ $\exists$ promise’
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:

**Promise’s certification via traversal of certification graph**

- Traverse simulated by $\exists$ Promise'
- Traverse' simulated by $\exists$ Promise'
Traversal of IMM execution

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

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

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

Promised

Covered

Issued
Traversal of IMM execution

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

\[ a := [x]; \quad \text{Promised} \quad b := [y]; \]

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

Covered

Issued
Traversal of IMM execution

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

Promised

Covered

Issued
Traversal of IMM execution

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

WeakestMO

(R)C11

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

CompCert \longrightarrow Weak MMs

\begin{itemize}
\item [Promise]
\item [WeakestMO]
\item [(R)C11]
\end{itemize}

[Chakraborty and Vafeiadis, 2019]

\[
\begin{align*}
\text{IMM} & \quad \text{x86-TSO} \\
\text{IMM} & \quad \text{ARMv7} \\
\text{IMM} & \quad \text{ARMv8.3} \\
\text{IMM} & \quad \text{RISC-V} \\
\text{IMM} & \quad \text{POWER}
\end{align*}
\]
Bridging the Gap between PL and Hardware Weak MMs

CompCert \longrightarrow \text{Weak MMs}

Promise

WeakestMO \longrightarrow IMM

[R)C11

\[ \text{hb} \; ; \; \text{eco} \; ? \; \text{is irreflexive} \]

\[ \text{deps} \cup \text{rf} \; \text{is acyclic} \]

x86-TSO

ARMv7

ARMv8.3

RISC-V

POWER

Thank you!
Grounding thin-air reads with event structures.
In *POPL 2019*. ACM.

A promising semantics for relaxed-memory concurrency.
In *POPL 2017*. ACM.
Backup slides
IMM definition

Def. $G$ is called IMM-consistent if the following hold:

- $\text{codom}(G.rf) = G.\mathbb{R}$.
- For every location $\ell \in \text{Loc}$, $G.co$ totally orders $G.W_\ell$.
- $G.rmw \cap (G.fr e; G.coe) = \emptyset$.
- $G.hb ; G.eco?^?$ is irreflexive.
- $G.ar$ is acyclic.

$$
\begin{align*}
\text{ar} & \triangleq \text{rfe} \cup \text{bob} \cup \text{ppo} \cup \text{detour} \cup \text{psc} \cup [W_{\text{strong}}] ; \text{po} ; [W] \\
\text{bob} & \triangleq \text{po} ; [W_{\text{rel}}] \cup [R_{\text{acq}}] ; \text{po} \cup \text{po} ; [F] \cup [F] ; \text{po} \cup [W_{\text{rel}}] ; \text{po}|_{\text{loc}} ; [W] \\
\text{ppo} & \triangleq [R] ; (\text{deps} \cup \text{rfi})^+ ; [W] \\
\text{deps} & \triangleq \text{data} \cup \text{ctrl} \cup \text{addr} ; \text{po}^? \cup \text{casdep} \cup [R_{\text{ex}}] ; \text{po}
\end{align*}
$$
Traversal definition

\[
a \in \text{Next}(G, C) \cap \text{Coverable}(G, C, I) \quad \text{and} \quad w \in \text{Issuable}(G, C, I) \setminus I
\]

\[
G \vdash \langle C, I \rangle \rightarrow \langle C \cup \{a\}, I \rangle \\
G \vdash \langle C, I \rangle \rightarrow \langle C, I \cup \{w\} \rangle
\]

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

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

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

\( i \) \( e \in G.W \cap I \) \\
\( ii \) \( e \in G.R \) and \( \text{dom}(G.\text{rf} ; [e]) \subseteq I \) \\
\( iii \) \( e \in G.F^{\text{esc}} \) or \( iv \) \( e \in G.F^{\text{sc}} \) and \( \text{dom}(G.\text{sc} ; [e]) \subseteq C \).
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

\[
\begin{align*}
    a &:= [y]_{rlx} \parallel 1 \\
    [z]_{rlx} &:= a \\
    b &:= [z]_{rlx} \parallel 1 \\
    c &:= \text{FADD}_{\text{strong}}^{rlx,rel}(x, 1) \parallel 0 \\
    [y]_{rlx} &:= c + 1
\end{align*}
\]