Owicki-Gries Reasoning for Weak Memory Models

Ori Lahav and Viktor Vafeiadis

Max Planck Institute for Software Systems (MPI-SWS)

ICALP, July 2015

A mismatch

- Sequential consistency (a.k.a. “interleaving semantics") is the standard memory model for reasoning about concurrent programs.

- In the presence of data races, SC is invalidated by hardware implementations and compiler optimizations.

Examples of weak behaviors

Store buffering

Initially $x = y = 0$.

$$\begin{array}{l|l|l} & x := 1 & y := 1 \\ & \texttt{print}\ y & \texttt{print}\ x \end{array}$$

This program can print 00 (observed on x86/Power/ARM).

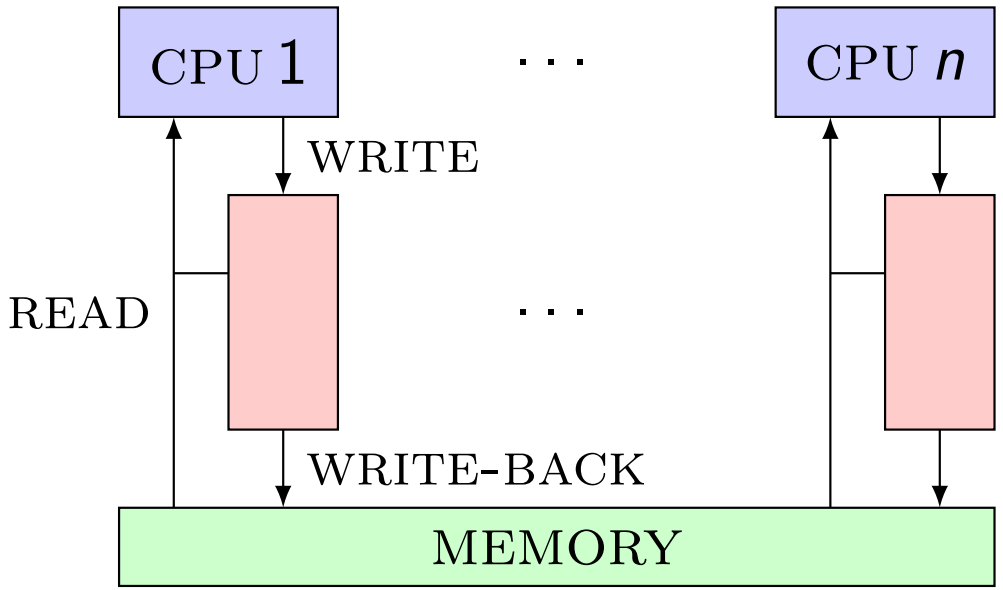
Examples of weak behaviors

Store buffering

Initially $x = y = 0$.

$$\begin{array}{l|l} x := 1 & y := 1 \\ \texttt{print}\ y & \texttt{print}\ x \end{array}$$

This program can print 00 (observed on x86/Power/ARM).



Basic X86-TSO Architecture

More examples of weak behaviors

Independent reads, independent writes

Initially, $x = y = 0$.

$$x := 1 \;\Big\|\; y := 1 \;\Big\|\; \begin{array}{l} \texttt{print}\ x \\ \texttt{print}\ y \end{array} \;\Big\|\; \begin{array}{l} \texttt{print}\ y \\ \texttt{print}\ x \end{array}$$

Both threads can print 10 (observed on Power/ARM).

Common sub-expression elimination

Initially, $x = y = 0$.

$$\begin{array}{l} x := 1 \\ y := 1 \end{array} \;\Big\|\; \begin{array}{l} \texttt{print}\ x \\ \texttt{print}\ y \\ \texttt{print}\ x \end{array}$$

This program can print 010 (observed with GCC compiler).

More examples of weak behaviors

Independent reads, independent writes

Initially, $x = y = 0$.

$$x := 1 \;\Big\|\; y := 1 \;\Big\|\; \begin{array}{l}\texttt{print}\ x\\ \texttt{print}\ y\end{array} \;\Big\|\; \begin{array}{l}\texttt{print}\ y\\ \texttt{print}\ x\end{array}$$

Both threads can print 10 (observed on Power/ARM).

Common sub-expression elimination

Initially, $x = y = 0$.

$$\begin{array}{l}x := 1\\ y := 1\end{array} \;\Big\|\; \begin{array}{l}\texttt{print}\ x\\ \texttt{print}\ y\\ \texttt{print}\ x\end{array} \quad \Rrightarrow \quad \begin{array}{l}t := x\\ \texttt{print}\ t\\ \texttt{print}\ y\\ \texttt{print}\ t\end{array}$$

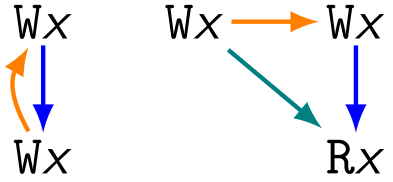
This program can print 010 (observed with GCC compiler).

Weak memory models provide formal sound semantics for realistic high-performance concurrency.

The C11 release/acquire memory model

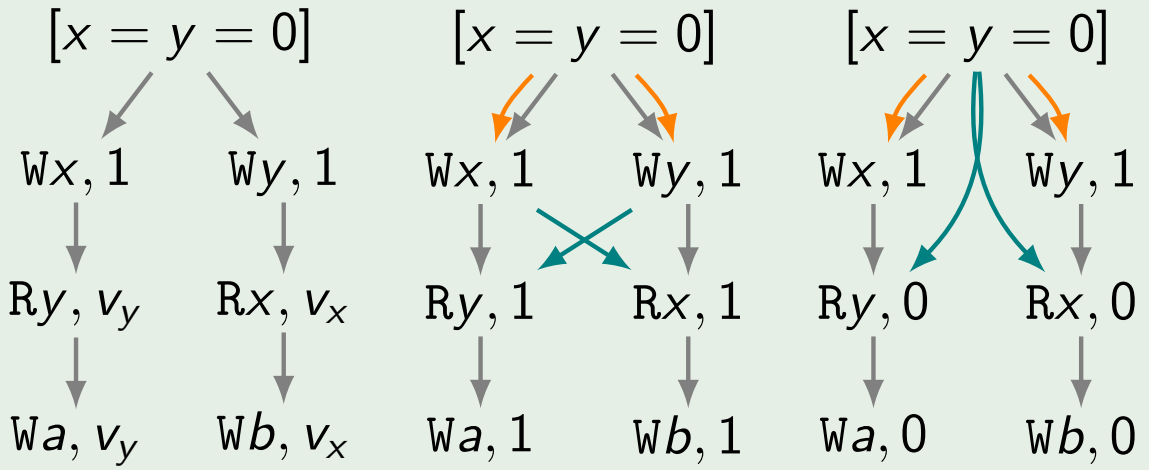
- A program $\rightsquigarrow$ a set of graphs (called: *executions*).
- An execution is *consistent* if it can be augmented with relations:
  - *reads-from*: associates each read with a corresponding write
    such that *happens-before* $= (\textit{prog-order} \cup \textit{reads-from})^+$ is irreflexive.
  - *modification-order*: total order on all writes to the same location

    such that none of the following occur:



Example (Store buffering)

$$\begin{array}{c} x = y = 0 \\ x := 1 \;\|\; y := 1 \\ a := y \;\|\; b := x \end{array}$$



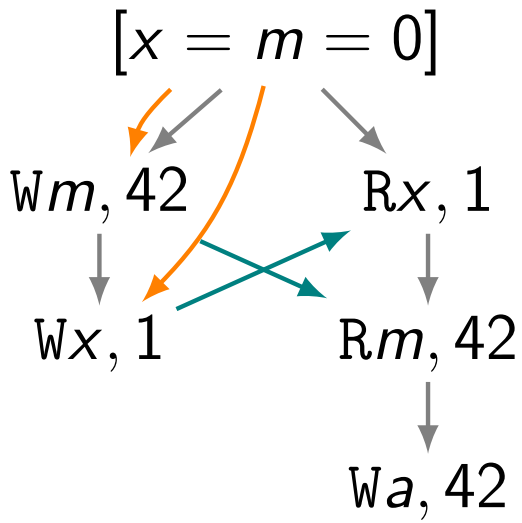
Message passing

$$
\begin{array}{c}
m = x = 0 \\
\begin{array}{l} m := 42 \\ x := 1 \end{array} \quad \Big\| \quad \begin{array}{l} \texttt{while } x = 0 \texttt{ do} \\ \quad \texttt{skip} \\ a := m \end{array}
\end{array}
$$

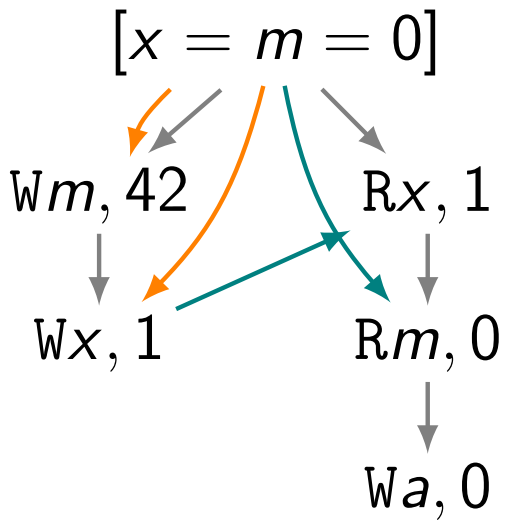
Message passing

$$m = x = 0$$

$$\begin{array}{l} m := 42 \\ x := 1 \end{array} \;\Big\|\; \begin{array}{l} \texttt{while } x = 0 \texttt{ do} \\ \quad \texttt{skip} \\ a := m \end{array}$$



Consistent



Inconsistent

Our work

Goals:

- Verify concurrent programs under WM.
- Investigate what program logics are sound under WM.

This paper:

- We show that Owicki-Gries is unsound for WM *(even without ghost variables and atomic blocks)*.
- We identify a simple weakening of OG that is sound for the release/acquire memory model.
- We demonstrate that this simple program logic is useful:
  - Verification of a simple RCU (read-copy-update) synchronization mechanism with release/acquire accesses.

Owicki-Gries method (1976)

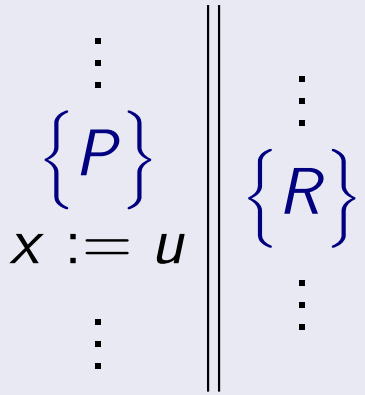
OG = Hoare logic + rule for parallel composition

$$\frac{\{P_1\}\, c_1\, \{Q_1\} \qquad \{P_2\}\, c_2\, \{Q_2\} \quad \text{the two proofs are } \textit{non-interfering}}{\{P_1 \wedge P_2\}\, c_1 \parallel c_2\, \{Q_1 \wedge Q_2\}}$$

Non-interference

$R \wedge P \vdash R\{u/x\}$ for every:

- assertion R in one proof outline
- assignment $x := u$ with precondition P in the other proof outline



Example SB: store buffering

$$
\begin{array}{c}
\{a \neq 0\} \\
\begin{array}{l|l}
\{a \neq 0\} & \{\top\} \\
x := 1 & y := 1 \\
\{x \neq 0\} & \{y \neq 0\} \\
a := y & b := x \\
\{x \neq 0\} & \{y \neq 0 \wedge (a \neq 0 \vee b = x)\}
\end{array} \\
\{a \neq 0 \vee b \neq 0\}
\end{array}
$$

Example SB: store buffering

$$
\begin{array}{c}
\{a \neq 0\} \\
\{a \neq 0\} \quad \| \quad \{\top\} \\
x := 1 \quad \| \quad y := 1 \\
\{x \neq 0\} \quad \| \quad \{y \neq 0\} \\
a := y \quad \| \quad b := x \\
\{x \neq 0\} \quad \| \quad \{y \neq 0 \wedge (a \neq 0 \vee b = x)\} \\
\{a \neq 0 \vee b \neq 0\}
\end{array}
$$

$$R \wedge P \vdash R\{y/a\}$$

Example SB: store buffering

$$
\begin{array}{c}
\{a \neq 0\} \\
\{a \neq 0\} \quad \| \quad \{\top\} \\
x := 1 \quad \| \quad y := 1 \\
\{x \neq 0\} \quad \| \quad \{y \neq 0\} \\
a := y \quad \| \quad b := x \\
\{x \neq 0\} \quad \| \quad \{y \neq 0 \wedge (a \neq 0 \vee b = x)\} \\
\{a \neq 0 \vee b \neq 0\}
\end{array}
$$

$$R \wedge P \vdash R\{y/a\}$$

Standard OG is **unsound** under weak memory!

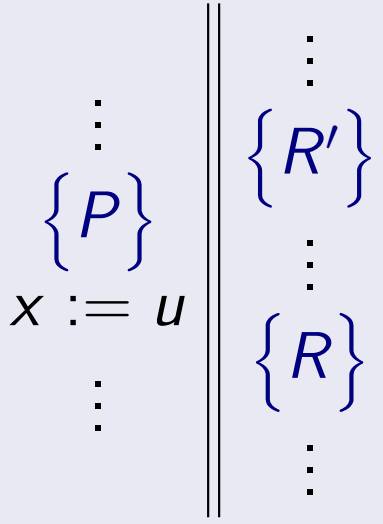
Stronger non-interference condition

$$\frac{\{P_1\}\, c_1\, \{Q_1\} \qquad \{P_2\}\, c_2\, \{Q_2\} \quad \text{the two proofs are non-interfering}}{\{P_1 \wedge P_2\}\, c_1 \parallel c_2\, \{Q_1 \wedge Q_2\}}$$

Strong non-interference

$R \wedge P \vdash R\{v/x\}$ for every:

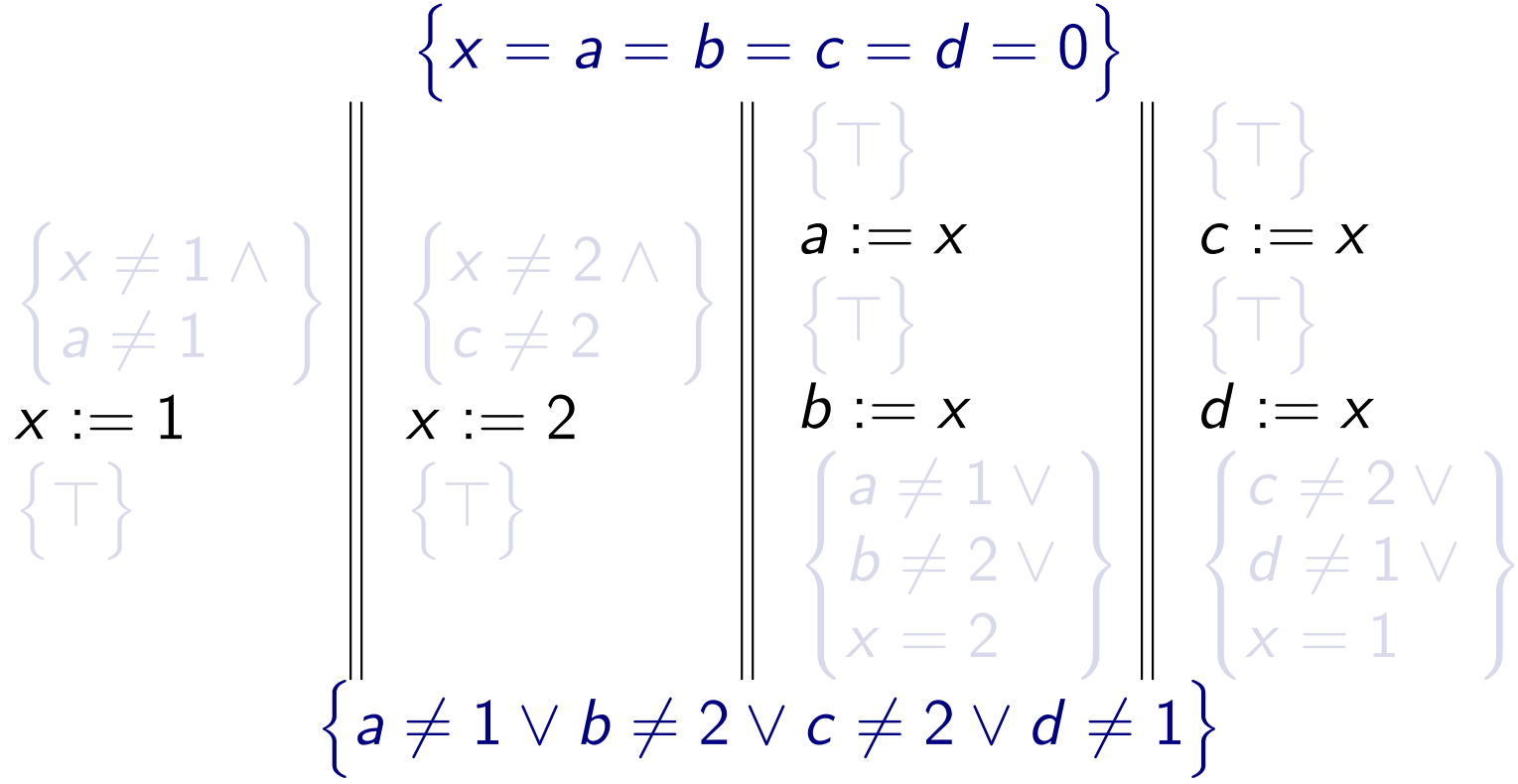
- assertion R in one proof outline
- assignment $x := u$ with precondition P in the other proof outline
- value v such that $P \wedge R' \wedge u = v$ is satisfiable for some R' above R



Example: message passing

$$
\begin{array}{c}
\{x = 0\} \\
\begin{array}{l|l}
\{\top\} & \{x \neq 0 \to m = 42\} \\
m := 42 & \texttt{while } x = 0 \texttt{ do skip} \\
\{m = 42\} & \{m = 42\} \\
x := 1 & a := m \\
\{\top\} & \{a = 42\}
\end{array} \\
\{a = 42\}
\end{array}
$$

Example: read-read coherence (CoRR2)



Example: read-read coherence (CoRR2)

$$\left\{x = a = b = c = d = 0\right\}$$

$$\begin{array}{l} \left\{\begin{array}{l} x \neq 1 \wedge \\ a \neq 1 \end{array}\right\} \\ x := 1 \\ \left\{\top\right\} \end{array} \;\Big\|\; \begin{array}{l} \left\{\begin{array}{l} x \neq 2 \wedge \\ c \neq 2 \end{array}\right\} \\ x := 2 \\ \left\{\top\right\} \end{array} \;\Big\|\; \begin{array}{l} \left\{\top\right\} \\ a := x \\ \left\{\top\right\} \\ b := x \\ \left\{\begin{array}{l} a \neq 1 \vee \\ b \neq 2 \vee \\ x = 2 \end{array}\right\} \end{array} \;\Big\|\; \begin{array}{l} \left\{\top\right\} \\ c := x \\ \left\{\top\right\} \\ d := x \\ \left\{\begin{array}{l} c \neq 2 \vee \\ d \neq 1 \vee \\ x = 1 \end{array}\right\} \end{array}$$

$$\left\{a \neq 1 \vee b \neq 2 \vee c \neq 2 \vee d \neq 1\right\}$$

Example: read-read coherence (CoRR2)

$$\left\{x = a = b = c = d = 0\right\}$$

$$\begin{array}{l|l|l|l}
\left\{\begin{array}{l} x \neq 1 \wedge \\ a \neq 1 \end{array}\right\} & \left\{\begin{array}{l} x \neq 2 \wedge \\ c \neq 2 \end{array}\right\} & \left\{\top\right\} & \left\{\top\right\} \\
x := 1 & x := 2 & a := x & c := x \\
\left\{\top\right\} & \left\{\top\right\} & \left\{\top\right\} & \left\{\top\right\} \\
 & & b := x & d := x \\
 & & \left\{\begin{array}{l} a \neq 1 \vee \\ b \neq 2 \vee \\ x = 2 \end{array}\right\} & \left\{\begin{array}{l} c \neq 2 \vee \\ d \neq 1 \vee \\ x = 1 \end{array}\right\}
\end{array}$$

$$\left\{a \neq 1 \vee b \neq 2 \vee c \neq 2 \vee d \neq 1\right\}$$

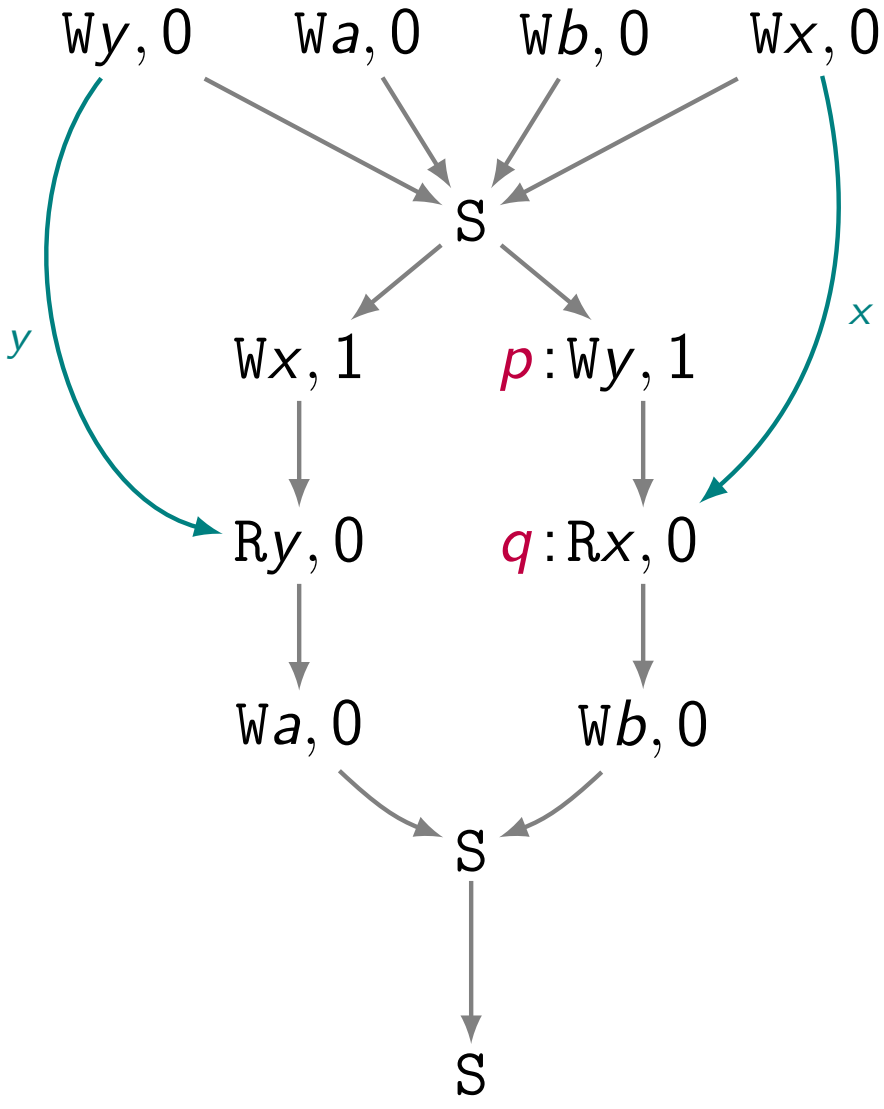
Soundness

Challenges in a weak memory setting:

- No intuitive operational semantics
- No notion of global state

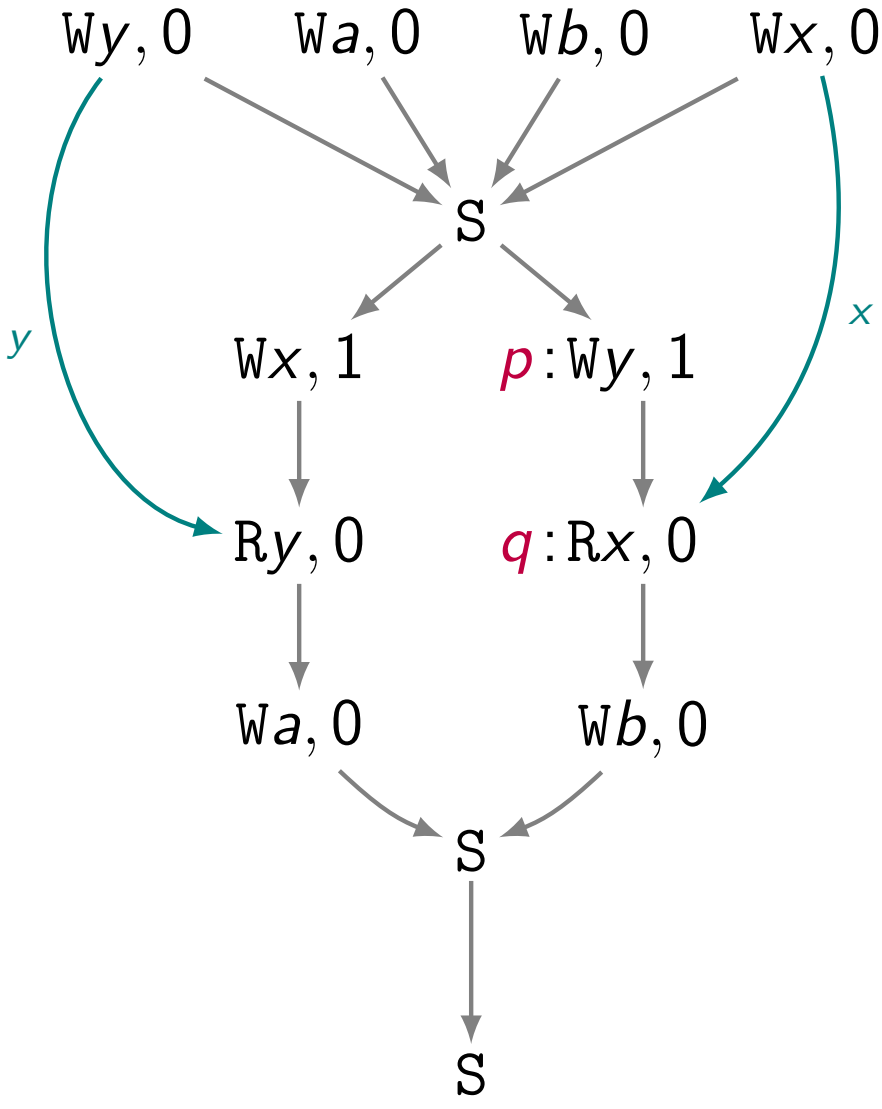
What does soundness exactly mean?

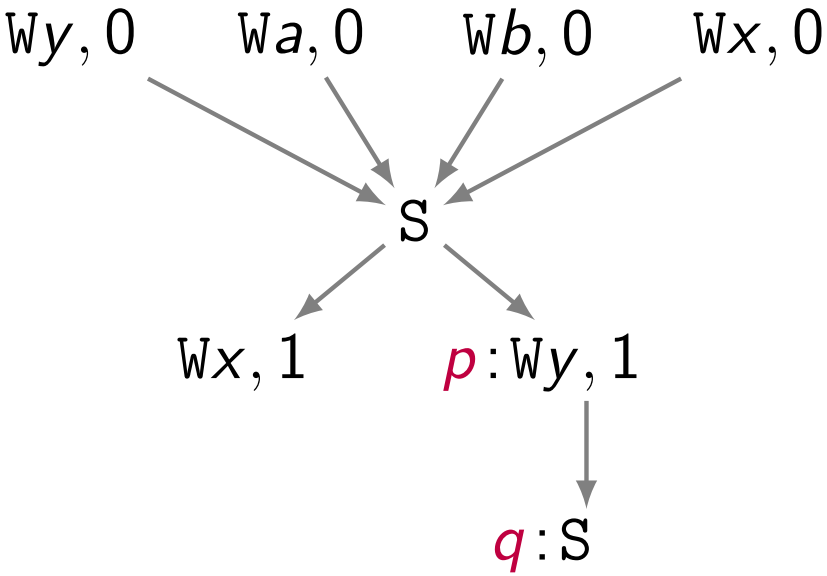
Visible states



$\sigma = \{x \mapsto 1, y \mapsto 1, a \mapsto 0, b \mapsto 0\}$ is visible at $\langle p, q \rangle$

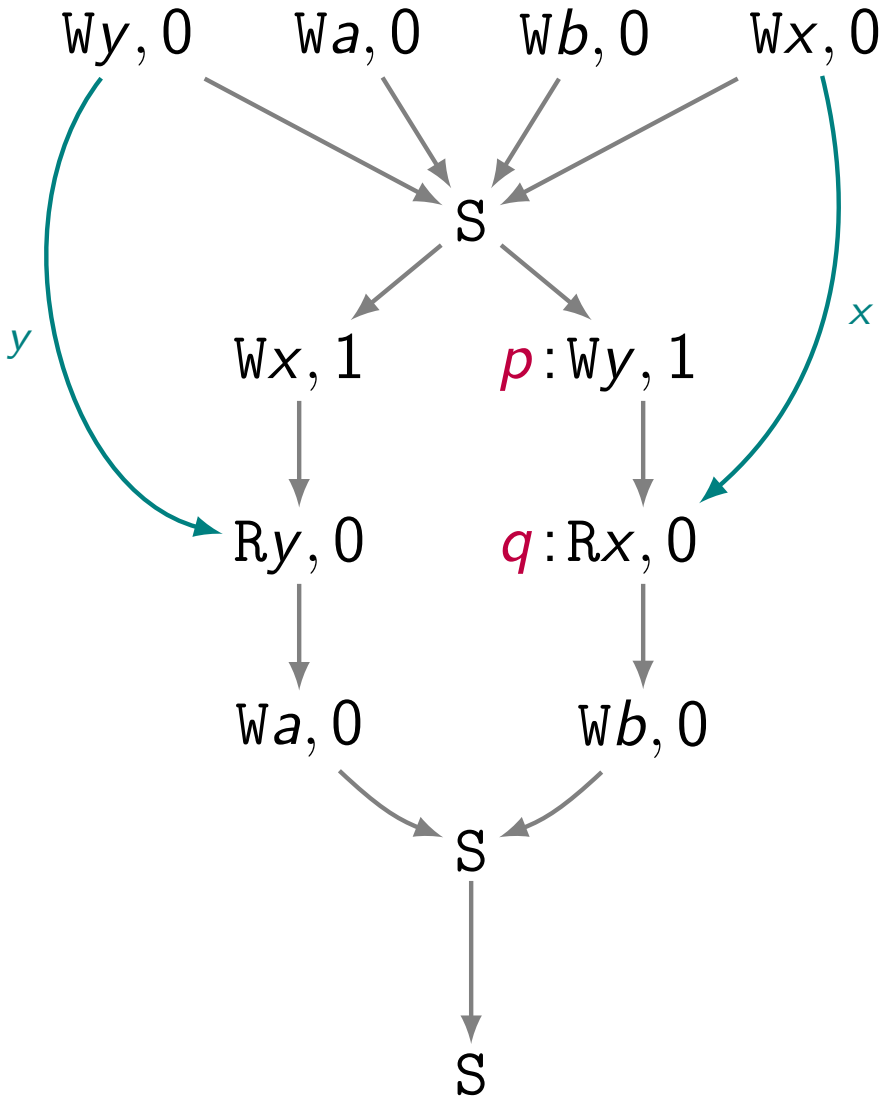
Visible states

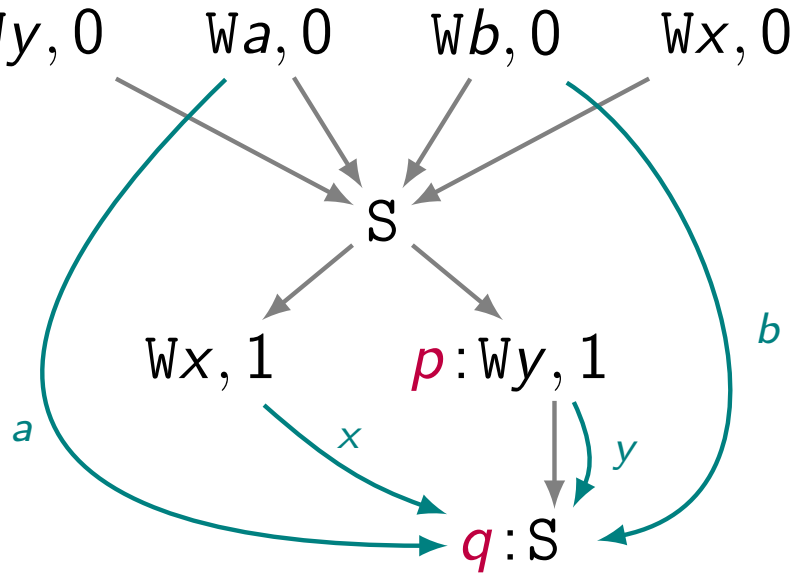




$\sigma = \{x \mapsto 1, y \mapsto 1, a \mapsto 0, b \mapsto 0\}$ is visible at $\langle p, q \rangle$

Visible states





$\sigma = \{x \mapsto 1, y \mapsto 1, a \mapsto 0, b \mapsto 0\}$ is visible at $\langle p, q \rangle$

Meaning of Hoare triples

Triple validity

$\{P\}\, c\, \{Q\}$ is *valid* if every state visible at the terminal edge of some consistent execution in $\mathcal{W}(P); [\![c]\!]$ satisfies Q.

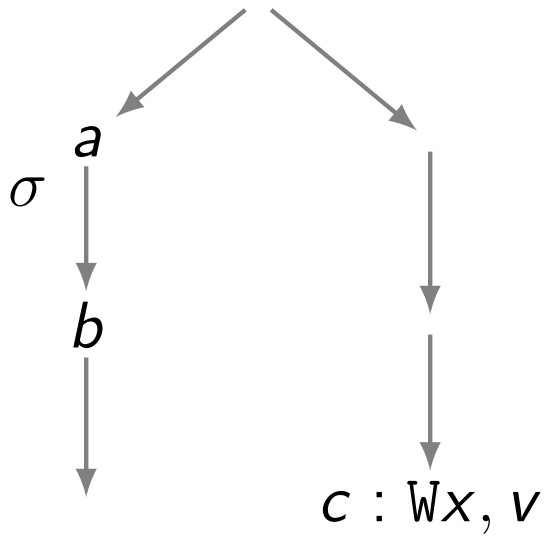
Main steps in soundness proof:

- Study properties of visibility under the RA model.
- Show that edges of consistent executions can be annotated with the assertions from the OG derivation such that every state visible at an edge satisfies its annotation.

Main visibility lemma (simplified)

Lemma

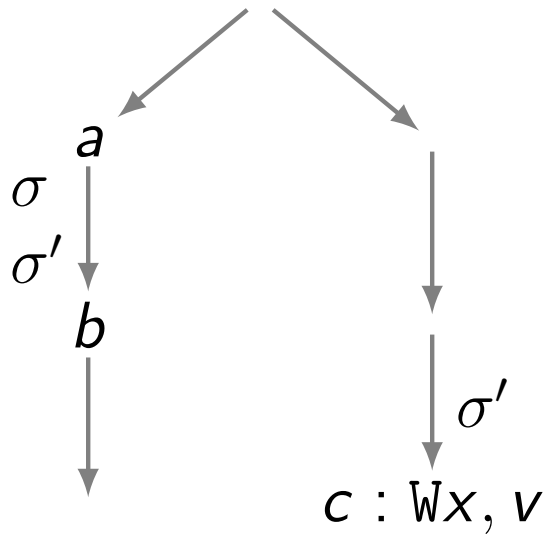
If a state σ becomes visible at $\langle a, b \rangle$ when adding a parallel node $c : \mathtt{W}x\ v$, then some x-variant of σ is visible both at $\langle a, b \rangle$ before adding c, and at every incoming edge to c.



Main visibility lemma (simplified)

Lemma

If a state σ becomes visible at $\langle a, b \rangle$ when adding a parallel node $c : \mathtt{W}x\ v$, then some x-variant of σ is visible both at $\langle a, b \rangle$ before adding c, and at every incoming edge to c.



Related work

Separation logics for C11:

- Relaxed separation logic (Vafeiadis & Narayan, OOPSLA'13)
- GPS (Turon et al., OOPSLA'14)

Other program logics:

- Rely/guarantee for TSO (Ridge, VSTTE'10)
- Verifying TSO programs (Jacobs, 2014)
- iCAP-TSO (Sieczkowski et al., ESOP'15)
- Coherent causal memory (Cohen, coRR 2014)

Conclusion

Summary of contributions:

- Owicki-Gries is unsound for WM.
- Stronger non-interference condition gives soundness for RA.
- Very basic auxiliary variables can be used ("ghost values").
- Rely/guarantee-style presentation of OG, that avoids non-modularity.
- This program logic is fairly useful and allows some automation.

Further work:

- Improve automation, apply to bigger examples
- Support more ghost variables
- Investigate completeness
- Revisit the separation logics for weak memory

Conclusion

Summary of contributions:

- Owicki-Gries is unsound for WM.
- Stronger non-interference condition gives soundness for RA.
- Very basic auxiliary variables can be used ("ghost values").
- Rely/guarantee-style presentation of OG, that avoids non-modularity.
- This program logic is fairly useful and allows some automation.

Further work:

- Improve automation, apply to bigger examples
- Support more ghost variables
- Investigate completeness
- Revisit the separation logics for weak memory

Thank you!

Rely/guarantee-style presentation of OG

OG judgments

$$\mathcal{R}; \mathcal{G} \Vdash \{P\}\,\{c\}\,\{Q\}$$

- $\mathcal{R} = \{R_1 \nearrow C_1, \ldots, R_n \nearrow C_n\}$ ("stable" assertions)
- $\mathcal{G} = \{\{P_1\}x_1 := u_1, \ldots, \{P_n\}x_n := u_n\}$ (guarded assignments)

Stability

$R \nearrow C$ is *stable* under $\{P\}x := y$ if $R \wedge P \vdash R[v_y/x]$ whenever $C \wedge P \wedge y = v_y$ is satisfiable.

Non-interference

$\mathcal{R}_1; \mathcal{G}_1$ and $\mathcal{R}_2; \mathcal{G}_2$ are *non-interfering* if every $R \nearrow C \in \mathcal{R}_i$ is stable under every $\{P\}c \in \mathcal{G}_j$ for $i \neq j$.

Some derivation rules

Example (Basic assignment rule)

$$\frac{P \vdash Q[y/x]}{\{P \nearrow P, Q \nearrow (P \vee Q)\}; \{\{P\} x := y\} \Vdash \{P\}\, x := y\, \{Q\}}$$

Example (Parallel composition rule)

$$\frac{\begin{array}{c}\mathcal{R}_1; \mathcal{G}_1 \Vdash \{P_1\}\, c_1\, \{Q_1\} \qquad \mathcal{R}_2; \mathcal{G}_2 \Vdash \{P_2\}\, c_2\, \{Q_2\} \\ Q_1 \wedge Q_2 \vdash Q \qquad \mathcal{R}_1; \mathcal{G}_1 \text{ and } \mathcal{R}_2; \mathcal{G}_2 \text{ are non-interfering}\end{array}}{\mathcal{R}_1 \cup \mathcal{R}_2 \cup \{Q \nearrow (\mathcal{R}_1^R \vee \mathcal{R}_2^R \vee Q)\}; \mathcal{G}_1 \cup \mathcal{G}_2 \Vdash \{P_1 \wedge P_2\}\, c_1 \parallel c_2\, \{Q\}}$$