Analysing Release-Acquire Consistency using Partial Orders

FM Update, July 09, 2021

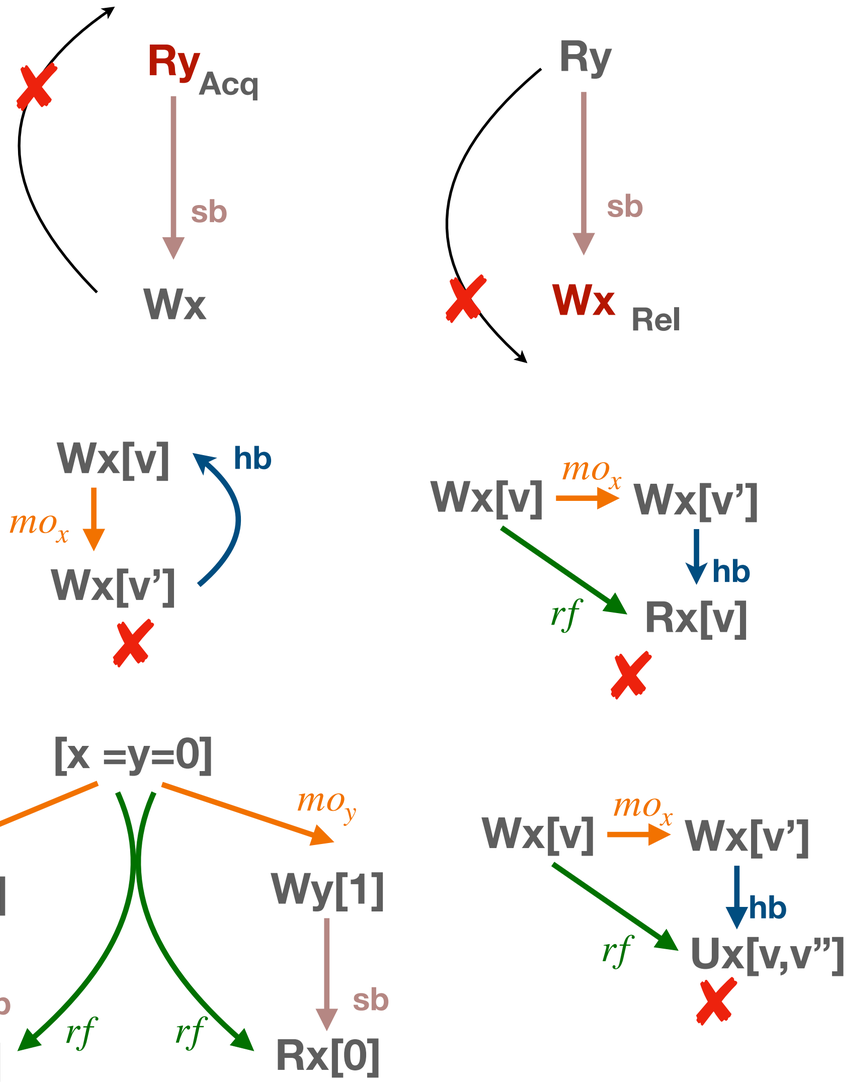
Divyanjali, Subodh Sharma
IIT Delhi

On Memory Consistency Models

- Sequential Consistency (SC): strong safety property; simple reasoning tool
  - interleaved execution semantics;
  - a total order on operations consistent with the order of operations of each thread.
- Weaker models: such as relaxed models for high performance
  - TSO: W-R reordering; exploiting store buffers
- Release-acquire: weaker than SC model but stricter than relaxed models

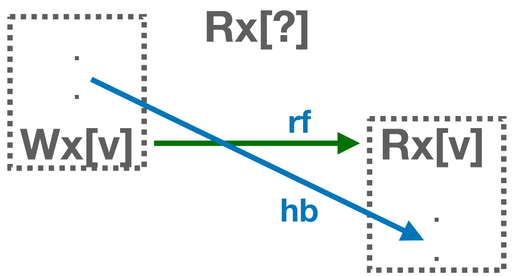
On Release-Acquire Model [BOSSW, POPL'11]

- All read are *acquires* and all writes are *release*
- All atomic updates are *acquire/release*
- Formal definition: In a *consistent* execution
  - Every read is justified by a corresponding write
    - Irreflexive *hb* = (*sequenced-before* ∪ *reads-from*)$^+$
  - Reads cannot observe overwritten values
    - Existence of per location total order on writes: *modification-order* s.t.
      - *hb* and *mo* are not inconsistent



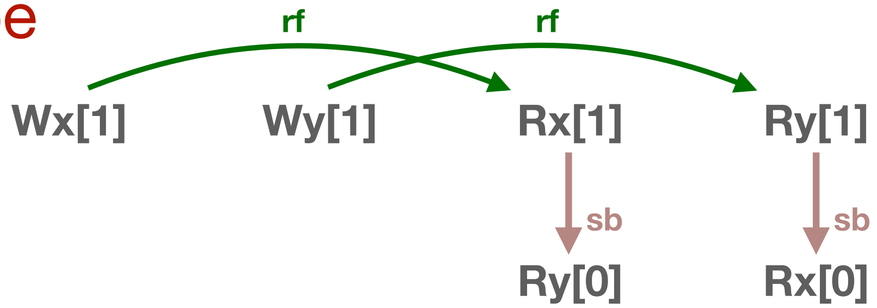
Positives w.r.t. Release-Acquire

- Verified compilation schemes on x86-TSO and POWER
- *Lack of global visibility of updates*
- *Stores prior to synchronising store also become observable to synchronising load's thread.*



Challenges with Release-Acquire Model

- Counter-intuitive outcomes (not explainable via interleaving semantics)
- Prior proposals for weak memory models may not be applicable
  - RA semantics cannot be modelled using store buffers [MM, VMCAI'17][TM, ASPLA'18]
  - Dependency involving more than two threads, use of interference combinations [KW, FSE'16,17]
- Control state reachability is undecidable [AAAK, PLDI'19]



Independent reads of independent writes

Objective: Efficient and sound verification of user assertions in programs under release-acquire memory model

Our Proposal

- Abstract domain: A new abstract domain that captures ordering dependencies (as a partial order) among instructions.
  - Upper approximation: Only the latest stores per thread per location are preserved.
- Thread-modular abstract interpretation: analyse each thread in isolation with an environment assumption.
- PRIORI: a prototype abstract interpreter; effective in refutation and verification of RA programs

Results

Comparison for Bug Hunting

Name	PRIORI		VBMC		CDS	TRACER	RCMC
	T	#It	T	VS			
dijkstra	0.05	2	0.18	2	0.01	0.01	0.03
bakery	0.18	2	0.10	2	0.01	0.01	0.03
burns	0.01	2	0.04	2	0.01	0.01	0.02
dekker	0.02	2	0.09	2	0.01	0.01	0.03
dekker_sim	0.01	2	0.03	2	0.01	0.01	0.03
lamport	0.02	2	0.20	2	0.01	0.02	0.03
peterson	0.01	2	0.05	2	0.01	0.01	0.03
peterson3	0.12	3	0.55	3	0.01	0.01	0.05
10R1W	0.02	2	3.99	10	0.01	0.01	0.03
15R1W	0.03	2	24.45	15	0.02	0.01	0.03
szymanski(7)	0.06	1	6.58	2	TO	TO	TO
fmax(2,7)	1.00	2	✗	-	0.15	0.05	TO

Results

Comparison for Proof of Correctness

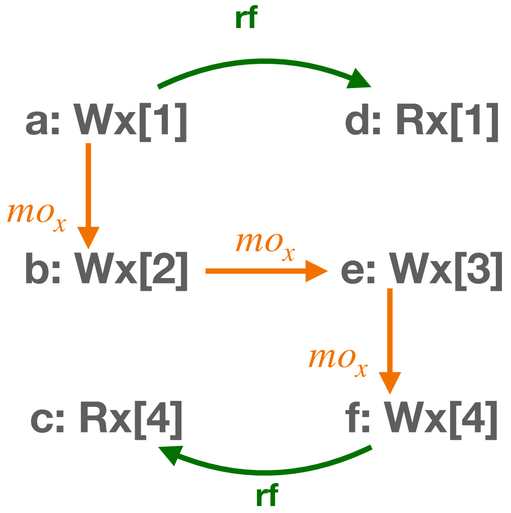
Name	PRIORI		VBMC	CDS	TRACER	RCMC
	T	#It	T			
CO-2+2W(5)	0.01	3	0.32	0.01	0.01	17.26
CO-2+2W(15)	0.02	3	1.29	0.02	0.01	TO
dijkstra_fen	0.10	5	206.70†	0.01	0.01	0.03
bakery_fen	0.38	7	171.62†	0.17	0.05	0.06
burns_fen	0.02	4	37.37†	0.02	0.01	0.02
peterson_fen	0.10	6	44.12†	0.02	0.01	0.03
tbar	0.04	6	18.58	0.02	0.01	0.14
hehner_c11	0.03	6	107.16†	0.07	0.02	0.04
red_co_20	0.04	3	31.47	23.32	0.13	TO
exp_bug_6	0.45	6	✗	97.13	0.96	37.82
exp_bug_9	0.57	6	✗	TO	2.98	437.47
stack_true(12)	0.06	4	✗	TO	589.81	TO
ib700wdt (1)	0.01	3	31.73	0.01	0.01	0.02
ib700wdt (20)	0.05	3	TO	0.01	0.01	TO
ib700wdt (40)	0.07	3	TO	0.01	0.01	TO
keybISR	0.01	4	0.01	0.01	0.01	0.03
fibonacci	0.11†	5	310.75	TO	56.4	20.61
lamport_fen	0.17†	4	431.40	0.09	0.03	0.04

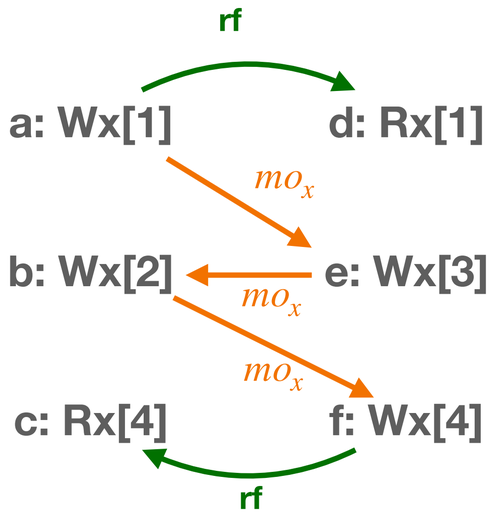
Collecting Semantics

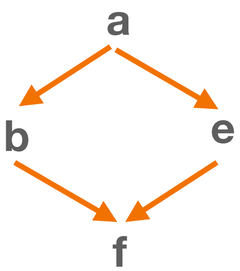
- Set of Modification orders as collecting semantics
  - Sound over- approximation [LV, ICALP'15]
  - Lemma 1: $(\mathcal{T}, \sqsubseteq)$ is a poset

$$t_1 := \underset{m_{11}, m_{12}}{\left\{ \begin{matrix} a & a \\ b & c \\ c & b \end{matrix} \right\}} \sqsubseteq t_2 := \underset{m_{21}, m_{22}, m_{23}}{\left\{ \begin{matrix} a & a & c \\ b & c & a \\ c & b & b \end{matrix} \right\}}$$

$$t_3 := \underset{m_{31}}{\left\{ \begin{matrix} a \\ b \end{matrix} \right\}} \quad \sqsubseteq \quad t_4 := \underset{m_{41}}{\{a\}}$$

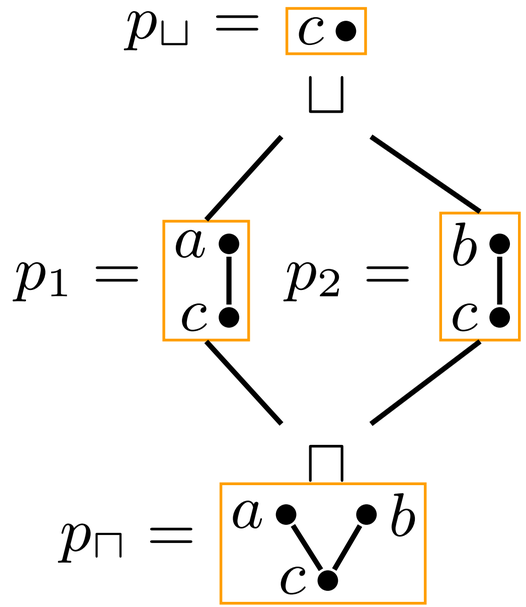






Abstract Semantics

- Lemma 2: $(P, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$ is a complete lattice
  - $p_1 \sqcap p_2$: the orderings of p_1 and p_2 are both present in the combination
  - $p_1 \sqcup p_2$: the common orderings of p_1 and p_2 on common elements are present in the combination.
  - Require additional checks:
    - For PO elements to remain acyclic
    - Do not have conflicting pair — ie, (a,b) and (b,a) cannot both be in the PO element.



Transfer Functions

$$\frac{(pre(\ell), mo, m) \in \mathcal{S} \qquad m' = m[x \to v] \qquad mo' = mo[x \to mo(x) \Diamond \ell]}{\mathcal{S} \xrightarrow{\ell:\mathtt{st}\ x\ v} \mathcal{S} \uplus (\ell, mo', m')}\ \text{STORE}$$

- For $x := v$, we update the map $x \mapsto v$, and augment l in the modification order of x so long as it is a valid extension.
- Rules for LOAD and RMW are similarly constructed.

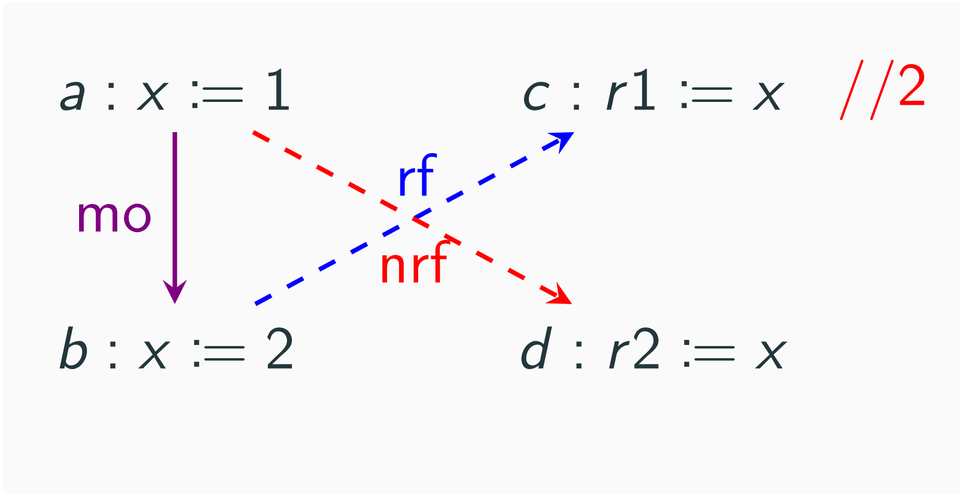
Theoretical Results

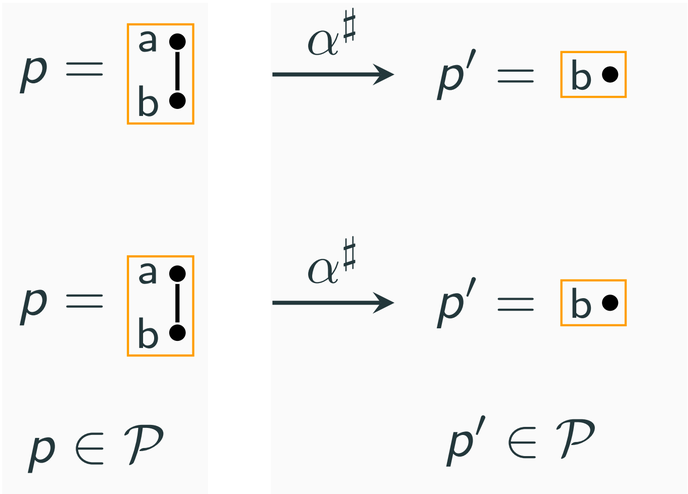
Theorem 1

$(\mathcal{T}, \trianglelefteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathcal{P}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$, where $\alpha : \mathcal{T} \to \mathcal{P}$ and $\gamma : \mathcal{P} \to \mathcal{T}$.

Upper Approximation

- Do we really need to store all the writes to the same location in the PO domain?





- Older sb-ordered writes to the same location can be forgotten!
- Key advantage: Potentially significant reduction in the number of states!

Theoretical Results

Theorem 1

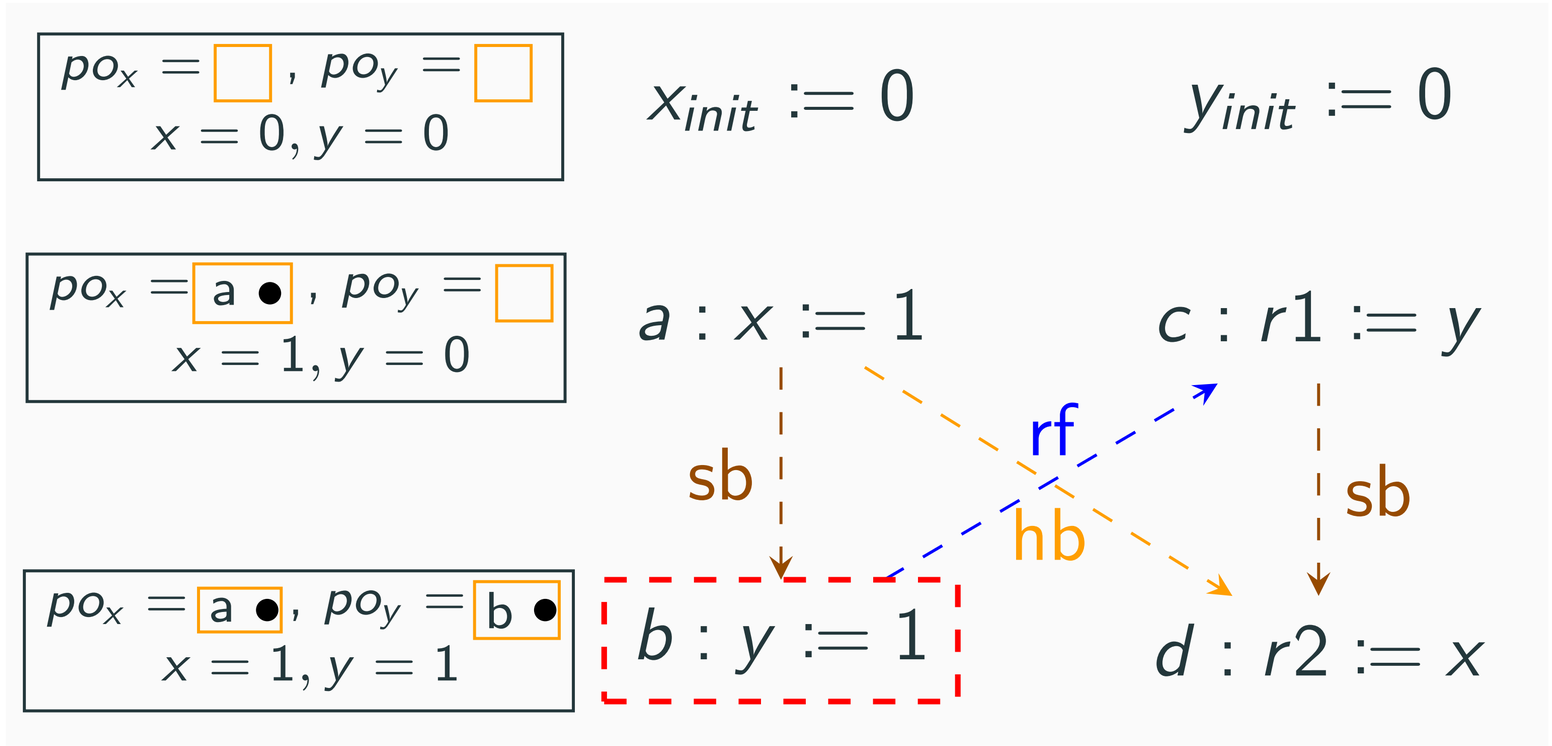
$(\mathcal{T}, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathcal{P}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$, where $\alpha : \mathcal{T} \to \mathcal{P}$ and $\gamma : \mathcal{P} \to \mathcal{T}$.

Theorem 2

Abstraction function $\alpha^\sharp : \mathcal{P} \to \mathcal{P}$ is a sound abstraction.

Thread Modular Analysis with PO Domain

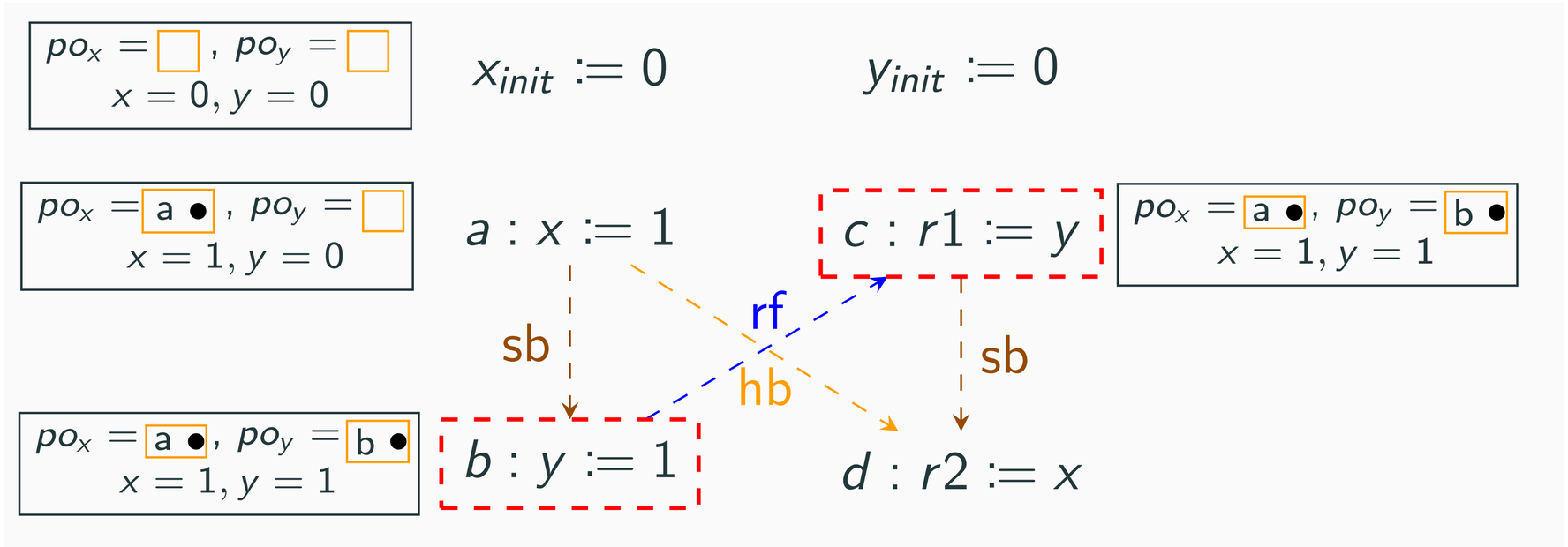
- Keep one $p \in P$ per memory location



Append stores

Thread Modular Analysis with PO Domain

- Keep one $p \in P$ per memory location



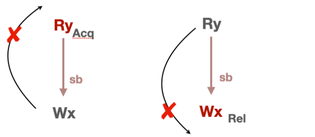
Applying the effects of the environment:
carry the program state

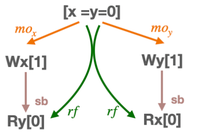
Some ideas

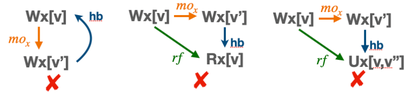
- Upper approximation can increase the false-alarm rate
  - Counter-example guided refinement?
  - Critical writes?
- Application to other relaxed memory models (PO domains are general)
  - TSO/PSO, RA+NA, RA+RLX

On Release-Acquire Model [Batty et al POPL 2011]

- All read are *acquires* and all writes are *release*
- All atomic updates are *acquire/release*
- Formal definition: In a *consistent* execution
  - Every read is justified by a corresponding write
    - Irreflexive *hb* = (*sequenced-before* ∪ *reads-from*)$^+$
  - Reads cannot observe overwritten values
    - Existence of per location total order on writes: *modification-order* s.t.
      - *hb* and *mo* are not inconsistent







Our Proposal

- Abstraction: A new abstract domain that captures ordering dependencies (as a partial order) among instructions.
  - Upper approximation: Only the latest stores per thread per location are preserved.
  - Meet, Join, Widening: Operators over elements of the abstract lattice.
- Thread-modular abstract interpretation: analyse each thread in isolation with an environment assumption.
- PRIORI: a prototype abstract interpreter; effective in refutation and verification of RA programs

Thank You!

svs@cse.iitd.ac.in

Theoretical Results

Theorem 1

$(\mathcal{T}, \preceq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathcal{P}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$, where $\alpha : \mathcal{T} \to \mathcal{P}$ and $\gamma : \mathcal{P} \to \mathcal{T}$.

Theorem 2

Abstraction function $\alpha^\sharp : \mathcal{P} \to \mathcal{P}$ is a sound abstraction.

Results

Comparison for Proof of Correctness

Name	PRIORI		VBMC	CDS	TRACER	RCMC
	T	#It	T			
CO-2+2W(5)	0.01	3	0.32	0.01	0.01	17.26
CO-2+2W(15)	0.02	3	1.29	0.02	0.01	TO
dijkstra_fen	0.10	5	206.70†	0.01	0.01	0.03
bakery_fen	0.38	7	171.62†	0.17	0.05	0.06
burns_fen	0.02	4	37.37†	0.02	0.01	0.02
peterson_fen	0.10	6	44.12†	0.02	0.01	0.03
tbar	0.04	6	18.58	0.02	0.01	0.14
hehner_c11	0.03	6	107.16†	0.07	0.02	0.04
red_co_20	0.04	3	31.47	23.32	0.13	TO
exp_bug_6	0.45	6	✗	97.13	0.96	37.82
exp_bug_9	0.57	6	✗	TO	2.98	437.47
stack_true(12)	0.06	4	✗	TO	589.81	TO
ib700wdt (1)	0.01	3	31.73	0.01	0.01	0.02
ib700wdt (20)	0.05	3	TO	0.01	0.01	TO
ib700wdt (40)	0.07	3	TO	0.01	0.01	TO
keybISR	0.01	4	0.01	0.01	0.01	0.03
fibonacci	0.11†	5	310.75	TO	56.4	20.61
lamport_fen	0.17†	4	431.40	0.09	0.03	0.04