Decidable Verification under Localized Release-Acquire Concurrency

Abhishek Singh

Ori Lahav
Outline

• Causal Consistency Models

• The State Reachability Problem

• Overview of the known results, their implications, and key observations

• A new memory model (LRA) to improve the existing results

• Decidability of State Reachability for LRA using a lossy loLRA semantics
Causal Consistency Models

› Distributed and Concurrent Computing
  › Relies on access to Shared Data
  › No unique global view for shared data

› Most modern architecture (non SC)
  › x86-TSO
  › POWER
  › ARM
  › RISC-V
  › …

C/C++11 Memory Model
(Release-Acquire Fragment)
State Reachability

(Sequential Consistency)

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

Sequential Consistency Models
State Reachability

(Sequential Consistency)

SB

x := 0

y := 0

x := 1; y := 1;
a := y; // 0
b := x; // 0

Sequential Consistency Models
State Reachability

(Causal Consistency)

\[ x := 0 \quad y := 0 \]

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

\[ x := o \quad y := o \]

Sequential Consistency Models

Causal Consistency Models
State Reachability

(Causal Consistency)

Program Order ‘po’

SB

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

✓ Causal Consistency Models
State Reachability

(Causal Consistency)

x := 1;
a := y;
y := 1;
b := x;

SB

Reads from relation ‘rf’

✓ Causal Consistency Models
State Reachability

(Causal Consistency)

\[ x := 1; \quad \text{// 0} \]
\[ a := y; \quad \text{// 0} \]
\[ y := 1; \]
\[ b := x; \quad \text{// 0} \]

\( \text{SB} \)
\[ x := o \quad \text{y := o} \]

\( \text{po} \cup \text{rf} \) is acyclic

\( \text{irr-hb} \): the relation hb is irreflexive

\( \text{hb} = (\text{po} \cup \text{rf})^+ \)

WRA
Weak Release-Acquire

RA
Release-Acquire

SRA
Strong Release-Acquire
State Reachability

(Causal Consistency)

\[ po \cup rf \text{ is acyclic} \]

irr-hb: the relation hb is irreflexive

\[ hb = (po \cup rf)^+ \]

Load Buffer (LB)

WRA
Weak Release-Acquire

RA
Release-Acquire

SRA
Strong Release-Acquire
State Reachability

(Causal Consistency)

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

\[ \text{x:= o} \]
\[ \text{y:= o} \]

\[ \text{weak-coherence} : \text{read from the last aware write} \]

\[ \text{WRA} \quad \text{irr-hb} \]
\[ \text{RA} \quad \text{irr-hb} \]
\[ \text{SRA} \quad \text{irr-hb} \]
State Reachability

(Causal Consistency)

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

\( x := o \)
\( y := o \)

\( W_x \)
\( Rx \)
\( \text{weak-coherence} \)

\( \text{WRA} \)
irr-hb

\( \text{RA} \)
irr-hb

\( \text{SRA} \)
irr-hb

\( \times \) weak-coherence: read from the last aware write
State Reachability

(Causal Consistency)

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

**MP**
\[ x := o \]
\[ y := o \]

**read-coherence**

\[ W_x \xrightarrow{\text{mo}} W_x \]
\[ R_x \xleftarrow{\text{rf} x} \]
\[ \text{mo} = \bigcup \{m_{ox}\}_{x \in \text{Loc}} \]

**weak-coherence**: read from the last aware write

**WRA**
mir-hb
weak-coherence

**RA**
mir-hb
read-coherence

**SRA**
mir-hb
read-coherence

**mo; hb** is irreflexive
"mo agrees with hb"
State Reachability

(Causal Consistency)

\[ x := 1; \]
\[ y := 1; \]
\[ a := y; \quad \text{// 1} \]
\[ b := x; \quad \text{// 0} \]

\[ \text{x:= o} \]
\[ \text{y:= o} \]

\[ \text{weak-coherence} : \text{read from the last aware write} \]

\[ \text{WRA} \]
\[ \text{irr-hb} \]
\[ \text{weak-coherence} \]

\[ \text{RA} \]
\[ \text{irr-hb} \]
\[ \text{read-coherence} \]

\[ \text{SRA} \]
\[ \text{irr-hb} \]
\[ \text{read-coherence} \]

\[ \text{mo} \cup \text{hb} \text{ is acyclic} \]

\[ \text{mo} = \bigcup \{ \text{mo}_x \}_{x \in \text{Loc}} \]

\[ \text{mo;hb} \text{ is irreflexive} \]

"mo agrees with hb"
State Reachability Problem

Load Buffer (LB)

Store Buffer (SB)

Message Passing (MP)

x := 0
y := 0

a := y; // 1
b := x; // 1

x := 1;
y := 1;

// 0
// 1

x := o
y := o

x := 1;
a := y; // 0
b := x; // 0

y := 1;

// 1
// 0

x := o
y := o

x := 1;
a := y; // 1
b := x; // 0

WRA RA SRA

WRA RA SRA

WRA RA SRA

irr-hb

E

hb

hb

W_x

hf

R_x

W_x
Q- Can any execution of the above program reach the annotated program state under a given memory model $M$?
State Reachability Problem

Annotated Program

Q- Can any execution of the above program reach the annotated program state under a given memory model M?
State Reachability Problem

Q- Can any execution of the given program $P$ reach the annotated program state under a given memory model $M$?
Q- Can any execution of the given program $P$ reach the annotated program state under a given memory model $M$?
Known Results

[Podhorski and Ganninger. 2020]

\[ \text{RA} \]  Undecidable (PCP)

\[ \text{WRA} \]  Decidable (WSTS)

\[ \text{SRA} \]

\[ \text{P} \]

Annotated Program

\[ \text{WRA} \]  ?  \[ \text{RA} \]  ?  \[ \text{SRA} \]  ?

\[ Q \] Can any execution of the given program \( P \) reach the annotated program state under a given memory model \( M \)?
Decidability of WRA and SRA

- Decidability results established using the framework of **well-structured transition systems** (WSTS)
- New Operational Semantics is based on the concept of thread potentials which is lossy in nature.
- Lossy semantics induces labeled transition systems (LTS) equipped with a well quasi-order.
- WSTS admits **effective initialisation** and **effective pred-basis**.
Implications

If a bad state is reachable in RA then it is also reachable in WRA
Implications

Annotated Program

If a bad state is reachable in RA then it is also reachable in WRA

SOUND

Bad state not reached in WRA => not reachable in RA
Implications

If a bad state is reachable in RA then it is also reachable in WRA

SOUND

Bad state not reached in WRA => not reachable in RA
Implications

Annotated Program

If a bad state is reachable in RA then it is also reachable in WRA

INCOMPLETE
Bad state reachable in WRA => ???
If a bad state is reachable in RA then it is also reachable in WRA

Bad state reachable in WRA => ???
Implications

Annotated Program

If a bad state is reachable in RA then it is also reachable in WRA

**P**

\[
x := 1;
\]

\[
a := x; // 2
\]

\[
x := 2;
\]

\[
b := x; // 1
\]

WRA

WWrace

RA

No global order $m_{ox}$ between the racy writes can explain the annotated behaviour

False Alarm!

INCOMPLETE

Bad state reachable in WRA => ???
If a bad state is reachable in RA then it is also reachable in WRA

Thread: 2 changes its mind about the order of the racy writes between the writes.

False Alarm!

Oscillation 1

Thread: 1

Thread: 2

\[ x := 1; \]
\[ x := 2; \]
\[ b := x; //2 \]
\[ c := x; //1 \]
Oscillations

If a bad state is reachable in RA then it is also reachable in WRA

WRA is too weak to be useful for the sound verification of program under RA

Thread: 2 changes its mind about the order $mo_x$ between the racy writes

False Alarm!

WRA

RA

INCOMPLETE

Bad state reachable in WRA => ???
A single thread can't change it's mind about the order $m_0x$ between the racy writes.

Oscillatory Patterns

Axiomatic Characterisation + Effective Identification Procedure

OUTCOME:
A stronger memory model than WRA that can be used for sound verification of program under RA
Our Contributions

- Axiomatic Characterisation of Oscillations
- Localised Release-Acquire Memory Model (LRA)

- $\text{WRA} < \text{LRA} < \text{RA}$
  - LRA is more effective than WRA for sound verification of program under RA

- State Reachability is decidable under LRA
  - Using an equivalent lossy semantics $\text{loLRA}$ based on thread potentials
Axiomatising Oscillations (LRA)

Oscillation 1

Thread: 1

x := 1;

Thread: 2

x := 2;

b := x; // 2

c := x; // 1

Oscillation 2

Thread: 1

x := 2;
a := x; // 1

Thread: 2

b := x; // 2

c := x; // 1

Thread: 3

x := 1;

Oscillation 3

Thread: 1

x := 2;
a := y; // 1

Thread: 2

b := x; // 2

c := x; // 1

Thread: 3

x := 1;
y := 1
Axiomatising Oscillations (LRA)

Thread: 1

\[ x := 1; \]
\[ b := x;/\ 2 \]
\[ c := x;/\ 1 \]

Thread: 2

\[ x := 2; \]
\[ b := x;/\ 2 \]
\[ c := x;/\ 1 \]

Thread: 3

\[ a := x;/\ 1 \]
\[ b := x;/\ 2 \]
\[ c := x;/\ 1 \]

Thread: 1

\[ x := 2; \]
\[ b := x;/\ 2 \]
\[ c := x;/\ 1 \]

Thread: 2

\[ x := 1; \]
\[ a := y;/\ 1 \]
\[ b := x;/\ 2 \]
\[ c := x;/\ 1 \]
\[ y := 1 \]

Oscillation 1

Oscillation 2

Oscillation 3

\( W_x \xrightarrow{hb} W_x \)

\( W_x \xrightarrow{rf} W_x \)

\( W_x \xrightarrow{hb/\ rf} R_x \)

\( W_x \xrightarrow{rf} R_x \)

\( W_x \xrightarrow{hb} R_x \)

\( W_x \xrightarrow{rf} R_x \)

\( \text{irr-hb} \)

\( \text{local-read-coherence} \)

\( \text{weak-coherence} \)
LRA forbids Oscillations

Oscillation 1

Thread: 1
\(x := 1;\)
\(b := x; /\ 2\)
\(c := x; /\ 1\)

Thread: 2
\(x := 2;\)

Oscillation 2

Thread: 1
\(a := x; /\ 1\)
\(b := x; /\ 2\)
\(c := x; /\ 1\)

Thread: 2
\(x := 2;\)

Oscillation 3

Thread: 1
\(a := y; /\ 1\)
\(b := x; /\ 2\)
\(c := x; /\ 1\)

Thread: 2
\(x := 1;\)
\(y := 1\)

\(E\)

\(\text{irr-hb}\)

\(\text{weak-coherence}\)

\(\text{local-read-coherence}\)
LRA forbids Oscillations

Oscillation 1

Thread: 1
x := 1;
b := x; //2
c := x; //1

Thread: 2
x := 2;
x := 1;
b := x; //2
c := x; //1

Oscillation 2

Thread: 1
x := 1;

Thread: 2
a := x; //1
b := x; //2
c := x; //1

Oscillation 3

Thread: 1

Thread: 2
a := y; //1
b := x; //2
c := x; //1

Thread: 3
x := 1;
y := 1

irr-hb

hb

weak-coherence

hb

local-read-coherence
LRA forbids Oscillations

Oscillation 1

Thread: 1
\[ x := 1; \]
\[ b := x; /\ 2 \]
\[ c := x; /\ 1 \]

Thread: 2
\[ x := 2; \]
\[ b := x; /\ 2 \]
\[ c := x; /\ 1 \]

Oscillation 2

Thread: 1
\[ a := x; /\ 1 \]
\[ b := x; /\ 2 \]
\[ c := x; /\ 1 \]

Thread: 2
\[ x := 2; \]
\[ b := x; /\ 2 \]
\[ c := x; /\ 1 \]

Oscillation 3

Thread: 3
\[ a := y; /\ 1 \]
\[ b := x; /\ 2 \]
\[ c := x; /\ 1 \]

Thread: 1
\[ y := 1 \]

Thread: 2
\[ x := 1; \]

Thread: 3
\[ y := 1 \]

\[ W_x \xrightarrow{hb} W_x \]
\[ W_x \xrightarrow{rf} W_x \]
\[ W_x \xrightarrow{hb \setminus rf} R_x \]

Local-read-coherence

weak-coherence

irr-hb

E

hb

hb

hb

rf

rf

hb
LRA forbids Oscillations

Oscillation 1

Thread: 1
x := 1;
b := x; // 2
c := x; // 1

Thread: 2
x := 2;
b := x; // 2
c := x; // 1

Oscillation 2

Thread: 1
a := x; // 1
b := x; // 2
c := x; // 1

Thread: 2
x := 1;
b := x; // 2
c := x; // 1

Thread: 3
x := 2;
b := x; // 2
c := x; // 1

Oscillation 3

Thread: 1
x := 1;
b := x; // 2
c := x; // 1

Thread: 2
y := 1

Thread: 3
x := 1;
y := 1

weak-coherence

irr-hb

local-read-coherence
LRA Memory Model

Axiomatic Characterisation of Oscillations
Localised Release-Acquire Memory Model (LRA)

irr-hb

weak-coherence

local-read-coherence
LRA weaker than RA

LRA is weaker than RA
If a program state is reachable under RA, then it is also reachable under LRA

WRA < LRA < RA
LRA is more effective than WRA for sound verification of program under RA
Novel Lossy Semantics: IoLRA

Every thread $\tau$ has a **thread potential** $B(\tau)$ indicating the possible sequence of future reads and writes.

$$\begin{align*}
x &:= 0; \\
x &:= 1; \\
Y &:= 1; \\
a &:= y; \\
b &:= x; //0
\end{align*}$$

(Thread Potentials)

- $O_w(x)$ → write option
- $o(x)$ → read option
- $O_w(y)$ → write option
- $o(x)$ → read option
Novel Lossy Semantics: IoLRA

Initial thread potential $B_0(\tau)$
Only contains **write options**
Indicating possible future writes

**Initial Thread Potentials**

```
x := 0;
x := 1;
Y := 1;
a := y;
b := x; //0
```
Novel Lossy Semantics: IoLRA

\[ B_0(1) \]
\[ O_w(x) \]
\[ O_w(x) \]
\[ O_w(y) \]

\[ B_0(2) \]
\[ ... \]
\[ ... \]
\[ ... \]

These write options are guessed non-deterministically in the Initial state.
Novel Lossy Semantics: IoLRA

No future step of IoLRA can introduce new write options

These write options are guessed non-deterministically in the initial state.
Novel Lossy Semantics: IoLRA

\[ x := 0; \quad | \quad a := y; \]
\[ x := 1; \quad | \quad b := x; //0 \]

\[ O_w(x) \]
\[ O_w(y) \]

\[ \square \]

Can consume one or more read/write options
Novel Lossy Semantics: IoLRA

Thread: 1

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

Thread: 2

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

READ STEP

Requires the presence of \( o(y) \) read option at the beginning
Novel Lossy Semantics: IoLRA

The presence of \( o(y) \) can only be ensured by a future write step

\[
\begin{align*}
\text{Thread: 1} & \\
x &:= 0; \\
x &:= 1; \\
y &:= 1; \\
\text{Thread: 2} & \\
a &:= y; \quad //1 \\
b &:= x; \quad //0
\end{align*}
\]
Novel Lossy Semantics: IoLRA

**Thread: 1**

- `x := 0;`
- `x := 1;`
- `y := 1;`

**Thread: 2**

- `a := y; /*1`
- `b := x; /*0`

This write option is consumed by the current write step.

**B(1)**

- `O_w(x)`
- `O_w(x)`
- `O_w(y)`

**B(2)**

- ...

**WRITE STEP**

Marks the location of last read on the other thread that reads from the write and is replaced with `o(x)` by the current write step.
Novel Lossy Semantics: IoLRA

This write option is consumed by the current write step

Thread: 1
\[\begin{align*}
  x &:= 0; \\
  x &:= 1; \\
  y &:= 1;
\end{align*}\]

Thread: 2
\[\begin{align*}
  a &:= y;  \\
  b &:= x;
\end{align*}\]

WRITE STEP
Novel Lossy Semantics: IoLRA

Thread: 1

\[
x := 0; \\
x := 1; \\
y := 1;
\]

Thread: 2

\[
a := y; //1 \\
b := x; //0
\]

WRITE STEP

This write option is consumed by the current write step

Will be replaced with \( o(y) \) by the current write step
Novel Lossy Semantics: IoLRA

This write option is consumed by the current write step

Will be replaced with $o(y)$ by the current write step

The suffix $o(x)$ is present in the writing thread’s potential

Shared Memory Causality Principle
Novel Lossy Semantics: IoLRA

When this write option was being consumed by the first write step

Will be replaced with \( o(y) \) by the future write \( y := 1 \)

The suffix \( o(x) \) is present in the writing thread's potential
Novel Lossy Semantics: IoLRA

Oscillations

Thread: 1

\[ x := 2; \]

Thread: 2

\[ x := 1; \]

\[ b := x; \quad //2 \]

\[ c := x; \quad //1 \]

Initial State

Initial State will be replaced with \( o(x) \) by the write step \( x := 2 \)
Novel Lossy Semantics: IoLRA

Oscillations

CASE-1
x := 2 happens before x := 1

Thread: 1

Thread: 2

x := 2;

x := 1;
b := x; //2
c := x; //1

WRITE STEP
x := 2

The write option is consumed by the write step x := 2

Will be replaced with o(x) by the write step x := 2
Novel Lossy Semantics: IoLRA

**CASE-1**

\[x := 2 \text{ happens before } x := 1\]

**Thread: 1**

\[x := 2;\]

**Thread: 2**

\[x := 1;\]

\[b := x; \quad //2\]

\[c := x; \quad //1\]

**Oscillations**

\[\square\]

This write option consumed

**B(1)**

**B(2)**

\[O_w(x)\]

\[o(x)\]

replaced with \(o(x)\)

**WRITE STEP**

\[x := 2\]
Novel Lossy Semantics: IoLRA

Oscillations

CASE-1
x := 2 happens before x := 1

Thread: 1
x := 2;

Thread: 2
x := 1;
b := x; // 2
c := x; // 1

WRITE STEP
x := 1

This write option will be consumed
Novel Lossy Semantics: IoLRA

Oscillations

Thread: 1
x := 2;

Thread: 2
x := 1;
b := x; //2
c := x; //1

B(1)

O_w(x)

B(2)

O_w(x) → O_w(x)

Initial State

Will be replaced with o(x) by the write step x := 2
Novel Lossy Semantics: IoLRA

CASE-2
x := 1 happens before x := 2

Thread: 1

x := 2;

Thread: 2

x := 1;

b := x; //2

c := x; //1

Oscillations

WRITE STEP
x := 1

B(1)

O_w(x)

B(2)

O_w(x)

O_w(x)

o(x)
LRA equivalent to IoLRA

program state reachable under LRA

IFF

program state reachable under IoLRA

Declarative Semantics

Potential based Lossy Semantics
State Reachability is Decidable

**THEOREM**

State Reachability is decidable for loLRA

For a given program $P$, the LTS $P \bowtie \text{loLRA}$ is equipped with a well-quasi-ordering $\subseteq$, such that $P \bowtie \text{loLRA}$ is a WSTS that admits effective initialisation and effective pred-basis.

State Reachability is decidable under LRA
Using an equivalent lossy semantics loLRA based on thread potentials
Conclusion

• Introduced Localised Release-Acquire Memory Model which forbids oscillations.
• Decidability of LRA using an equivalent lossy Operational Semantics (loLRA)
• LRA is a better memory model for sound verification of programs under RA
• Novel lossy semantics is based on thread potential used for WRA and SRA
• Potential based techniques may be effective in exploring other decidable fragments

Thank you!