C/C++ Concurrency and Program Transformations
Recap: Shared Memory Accesses

Non-atomic accesses: Read (R), Write (W)
Atomic accesses = operation + memory order

Operations:
- Read (R)
- Write (W)
- Atomic update (U)
- Fence (F)

Memory orders:
- Relaxed (rlx)
- Release (rel)
- Acquire (acq)
- Acquire-Release (acq_rel)
- Sequentially consistent (sc)

Example:
- X.load(memory order)
- X.store(val, memory order)
- X.CAS(oldval, nwval, success mem order)
- atomic_thread_fence(memory order)
Recap: Access Types

Non-atomic read. e.g. \( t = X; \)
- \( R_{na} \)

Non-atomic write. e.g. \( X := v; \)
- \( W_{na} \)

Atomic read. e.g. \( X.\text{load(memory order)} \)
- \( R_{rlx}, R_{acq}, R_{sc} \)

Atomic write. e.g. \( X.\text{store(val, memory order)} \)
- \( W_{rlx}, W_{rel}, W_{sc} \)

Atomic update. e.g. \( X.\text{CAS(oldval, nwval, success memory order)} \)
- \( U_{rlx}, U_{rel}, U_{acq}, U_{acq\_rel}, U_{sc} \)

Fence. e.g. \( \text{atomic\_thread\_fence(memory order)} \)
- \( F_{rel}, F_{acq}, F_{acq\_rel}, F_{sc} \)
Axioms

Happens-before (hb) is irreflexive. \hspace{1cm} (irrHB)

Execution is coherent.

- rf; hb is irreflexive \hspace{1cm} (CohHBRF)
- mo; rf; hb is irreflexive. \hspace{1cm} (Coh-RW)
- mo; hb is irreflexive. \hspace{1cm} (Coh-WW)
- mo; hb; rf\(^{-1}\) is irreflexive. \hspace{1cm} (Coh-WR)
- mo; rf; hb; rf\(^{-1}\) is irreflexive. \hspace{1cm} (Coh-RR)

\[ [U] \cap (fr; mo) = \emptyset \] \hspace{1cm} (Atomicity)

\( (hb \cup mo) \cap ([SC] \times [SC]) \subseteq sc \) \hspace{1cm} (ConsSC)

\[
\forall a, b. \ (a, b) \in rf; [RU_{sc}] \implies
sc_{imm}(a, b) \lor a \not\in SC \land \exists x. \ hbloc(a, x) \land x \in WU_{sc} \land sc(x, b)
\] \hspace{1cm} (SCread)
Challenges

\[ a = X; \]
\[ \text{if}(a == 1) \]
\[ Y = 1; \]
\[ b = Y; \]
\[ \text{if}(b == 1) \]
\[ X = 1; \]

(A) \( a = b = 1 \) is not possible.

(B) \( a = b = 1 \) is possible.

Two programs share an execution.

Causality cycle: \((\text{po} \cup \text{rf})\).
Axioms

Causality cycle: \((po \cup rf)\).

Allow causality cycle:
- Justifies outcome in (A) which is not possible.
- Violates data-race-freedom guarantee.
- Justifies (B) outcome.
- Allows read-write reordering.

Disallow causality cycle:
- Restrict undesired outcome in (A).
- Provides data-race-freedom guarantee.
- Restricts (B) outcome.
- Restrict read-write reordering in compiler and architectures.

Tradeoff: Data-race-freedom guarantee vs efficient compilation.
Compilation.

Outcome of execution:

\[ O(X) \triangleq \{(e.\text{Loc}, e.\text{wval}) \mid e \in X.E \cap WU. \{e\}; X.\text{mo} = \emptyset\} \]

Outcome of program P:

\[ O(P) = \{O(X) \mid X \in [[P]]\} \]

Given a compilation

\[ P_{C11} \Rightarrow P_1 \Rightarrow P_2 \Rightarrow \cdots \Rightarrow P_{\text{arch}} \]

Correctness requires:

\[ O(P_{C11}) \supseteq O(P_1) \supseteq O(P_2) \supseteq \cdots \supseteq O(P_{\text{arch}}) \]
Correct Compilation

For each consistent execution of the target program there exists a corresponding consistent execution of the source program with same outcome.

\[ X \in \llbracket P_{tgt} \rrbracket. \exists X' \in \llbracket P_{src} \rrbracket. O(X) = O(X') \]
Transformations

Redundant access elimination such as

- $Y_{rlx} = v; Y_{rlx} = v'; \sim \rightarrow Y_{rlx} = v'$; (OW)
- $Y_{rlx} = v; a = Y_{rlx}; \sim \rightarrow Y_{rlx} = v; a = v$; (RAW)
- $a = Y_{rlx}; a = Y_{rlx}; \sim \rightarrow a = Y_{rlx}$; (RAR)

Fence Insertion such as $C_1; C_2; C_1; \{rel, acq, sc\}; C_2$

Access strengthening such as $Y_{rlx} = v; Y_{rel} = v; a = Y_{rlx}; Y_{acq} = Y_{rlx}$

Reordering of accesses such as $a = Y_{rlx}; Z_{rlx} = v; Z_{rlx} = v; a = Y_{rlx}$;
Transformations

Redundant access elimination such as

- \( Y_{rlx} = v; Y_{rlx} = v'; \sim Y_{rlx} = v' \);  
- \( Y_{rlx} = v; a = Y_{rlx}; \sim Y_{rlx} = v; a = v \);  
- \( a = Y_{rlx}; a = Y_{rlx}; \sim a = Y_{rlx} \);

(OW)  
(RAW)  
(RAR)

Fence Insertion such as

- \( C_1; C_2 \sim C_1; F_{\{rel,acq,sc\}}; C_2 \)
Transformations

Redundant access elimination such as

- $Y_{rlx} = v; Y_{rlx} = v'; \rightsquigarrow Y_{rlx} = v'$;  \hspace{1cm} (OW)
- $Y_{rlx} = v; a = Y_{rlx}; \rightsquigarrow Y_{rlx} = v; a = v$;  \hspace{1cm} (RAW)
- $a = Y_{rlx}; a = Y_{rlx}; \rightsquigarrow a = Y_{rlx}$;  \hspace{1cm} (RAR)

Fence Insertion such as

- $C_1; C_2 \rightsquigarrow C_1; F_{\{\text{rel,acq,sc}\}}; C_2$

Access strengthening such as

- $Y_{rlx} = v \rightsquigarrow Y_{rel} = v$
- $a = Y_{rlx} \rightsquigarrow a = Y_{acq}$
Transformations

Redundant access elimination such as

- $Y_{rlx} = v; Y_{rlx} = v' \leadsto Y_{rlx} = v'$; (OW)
- $Y_{rlx} = v; a = Y_{rlx}; \leadsto Y_{rlx} = v; a = v$; (RAW)
- $a = Y_{rlx}; a = Y_{rlx}; \leadsto a = Y_{rlx}$; (RAR)

Fence Insertion such as

- $C_1; C_2 \leadsto C_1; F_{\{rel,acq,sc\}}; C_2$

Access strengthening such as

- $Y_{rlx} = v \leadsto Y_{rel} = v$
- $a = Y_{rlx} \leadsto a = Y_{acq}$

Reordering of accesses such as

- $a = Y_{rlx}; Z_{rlx} = v \leadsto Z_{rlx} = v; a = Y_{rlx}$
Can we prove the correctness of an access elimination?
Can we prove the correctness of an access elimination?

Remember the theorem:

\[ X \in [P_{tgt}]. \exists X' \in [P_{src}]. \, O(X) = O(X') \]
Can we prove the correctness of an access elimination?

Remember the theorem:

\[ X \in \mathbb{P}_{tgt}. \exists X' \in \mathbb{P}_{src}. O(X) = O(X') \]

Proof Strategy:

1. For each consistent execution \( X' \) of target program \( \mathbb{P}_{tgt} \), define an execution \( X \) of \( \mathbb{P}_{src} \).
2. Show that \( X \) is consistent.
3. Show that \( O(X) = O(X') \)
Let's consider OW transformations i.e.

\[ Y_o = v; \quad Y_{o'} = v' \quad \leadsto \quad Y_{o'} = v' \]

where \( o, o' \in \{\text{na, rlx, rel, sc}\} \)

**Theorem**

\((OW)\) is correct when \( o \subseteq o' \).
Remember the proof strategy steps:

1. For each consistent execution $X'$ of target program $P_{tgt}$, define an execution $X$ of $P_{src}$.
Remember the proof strategy steps:

1. For each consistent execution $X'$ of target program $P_{tgt}$, define an execution $X$ of $P_{src}$.

Let $X'$ be a target execution where $Y_{o'} = v'$ results in event $e' = W_{o'}(Y, v')$.

We define source execution $X$ where $Y_o = v$ results in event $e = W_o(Y, v)$ and

- $X.E = X'.E \cup \{e\}$
- $X.po = X'.po \cup \{(e, e')\} \cup \{(a, e) \mid X'.po(a, e')\} \cup \{(e, a) \mid X'.po(e', a)\}$
- $X.rf = X'.rf$
- $X.mo = X'.mo \cup \{(e, e')\} \cup \{(a, e) \mid X'.mo(a, e')\} \cup \{(e, a) \mid X'.mo(e', a)\}$
- $X.sc = X'.sc \cup \{(e, e') \mid o = sc\} \cup \{(a, e) \mid X'.(a, e') \land o = sc\} \cup \{(e, a) \mid X'.sc(e', a) \land o = sc\}$
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$$X'.hb = X'.hb \cup \{(e, e')\} \cup \{(a, e) \mid X'.hb(a, e')\} \cup \{(e, a) \mid X'.hb(e', a)\}$$
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$X.hb = X'.hb \cup \{(e, e')\} \cup \{(a, e) \mid X'.hb(a, e')\} \cup \{(e, a) \mid X'.hb(e', a)\}$

(irrHB)

Assume there exists a $X.hb$ cycle.

It implies that there exists a $(X.po \cup X.sw)^+$ cycle.

It implies that the $(X.po \cup X.sw)^+$ cycle passes through event $e$.

However, from definition, there is no incoming/outgoing $X.sw$ edge to/from event $e$.

Therefore $X.hb$ through $e$ contains $X.hb(\neg, e)$ and $X.hb(e, \neg)$.

It implies $X'.hb$ is a cycle which is a contradiction.

Therefore $X.hb$ satisfies (irrHB).
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$$X . hb = X' . hb \cup \{(e, e')\} \cup \{(a, e) | X' . hb(a, e')\} \cup \{(e, a) | X' . hb(e', a)\}$$

Assume $X$ is not coherent.
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$$X.hb = X'.hb \cup \{(e, e')\} \cup \{(a, e) \mid X'.hb(a, e')\} \cup \{(e, a) \mid X'.hb(e', a)\}$$

Assume $X$ is not coherent.
Assume there exists a $X.rf; X.hb$ cycle.
It involves event $e$.
However, from definition, we know that $X.rf = X.rf$.
Therefore a $X.rf; X.hb$ cycle implies that there already exists a a $X'.rf; X'.hb$ cycle which is a contradiction.
Therefore $X$ satisfies (CohHBRF).
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

\[ X.hb = X'.hb \cup \{(e, e')\} \cup \{(a, e) \mid X'.hb(a, e')\} \cup \{(e, a) \mid X'.hb(e', a)\} \]

Assume $X$ is not coherent.
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$$X.\text{hb} = X'.\text{hb} \cup \{(e, e')\} \cup \{(a, e) \mid X'.\text{hb}(a, e')\} \cup \{(e, a) \mid X'.\text{hb}(e', a)\}$$

Assume $X$ is not coherent.

Assume there exists a $X.\text{mo}; X.\text{rf}; X.\text{hb}$ cycle.

It involves event $e$.

However, from definition, we know that

$$X.\text{mo}(a, e) \implies X.\text{mo}(a, e'),$$

$$X.\text{mo}(e, a) \implies a = e' \lor X'.\text{mo}(e', a),$$

$$X.\text{po}(a, e) \implies X.\text{po}(a, e'), X.\text{po}(e, a) \implies a = e' \lor X'.\text{po}(e', a).$$

Therefore a $X.\text{mo}; X.\text{rf}; X.\text{hb}$ cycle implies that there already exists a $X'.\text{mo}; X'.\text{rf}; X'.\text{hb}$ cycle which is a contradiction.

Therefore $X$ satisfies (Coh-RW).
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

Note that

$$X \cdot \text{hb} = X' \cdot \text{hb} \cup \{(e, e')\} \cup \{(a, e) \mid X' \cdot \text{hb}(a, e')\} \cup \{(e, a) \mid X' \cdot \text{hb}(e', a)\}$$

Assume $X$ is not coherent.

Exercise: Prove rest of the coherence axioms.
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

(Atomicity)
Assume $X'$ does not preserve (Atomicity).
However, from definition, it implies $X'$ violates (Atomicity) which is a contradiction.
Therefore $X'$ preserves atomicity.
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

(ConsSC)

Two possibilities:

Case $X.sc = X'.sc$ when $o \neq sc$

In this case $X$ preserves (ConsSC) as $X'$ preserves (ConsSC).

Case $X'.sc \subseteq X.sc$ when $o = sc$

Assume $X'$ does not preserve (ConsSC).

However, from definition, it implies $X'$ violates (ConsSC) which is a contradiction.

Therefore $X$ preserves (ConsSC).
Remember the proof strategy steps:

1. Show that $X$ is consistent.

We show $X$ is consistent by contradiction.

(SCread)
Assume $X$ does not preserve (SCread).
Two possibilities:

Case $o = sc$
Suppose $(a, b) \in X.rf; [RU_{sc}]$ and $X.sc_{imm}(e, b)$ holds.
However, from definition we know that $sc_{imm}(e, e')$ and $e' \notin RU_{sc}$.
Considering the other condition if $e$ violates the condition in $X$
then $e'$ already violates the condition in $X'$.
Hence a contradiction and (SCread) holds.

Case $o \neq sc$
From the definition $X$ preserves (ConsSC).

Therefore source execution $X$ is consistent.
Remember the proof strategy steps:

1. Show that $O(X) = O(X')$

From definition, $\{e\}; X.\text{mo} \neq \emptyset$ as $X.\text{mo}(e, e')$ holds.
Hence $O(X) = O(X')$.

Therefore the transformation is correct.
Show that the following transformation is not safe.

\[ Y_{sc} = \nu; \text{CAS}_{sc}(Y, v_1, v') \sim \text{CAS}_{sc}(Y, v_1, v'). \]

Hint. Check the \((SCread)\) axiom.