A OTHER LOOPING CONSTRUCTS

Using the primitive looping construct \( \text{loop } e \), the loop exiting construct \( \text{break}_n e \), and a simple register library, we can implement other types of loops as follows:

\[
\begin{align*}
\text{repeat } e & \triangleq \\
\text{loop} & \\
\text{let } x = e \text{ in} & \\
\text{if } x = 0 \text{ then } 0 \text{ else } \text{break}_1 e
\end{align*}
\]

for \( i = e_1 \) to \( e_2 \) do \( e_3 \triangleq \\
\text{let } \text{ivar} = \text{alloc}(1) \text{ in} & \\
\text{store}(\text{ivar}, e_1); & \\
\text{loop} & \\
\text{let } i = \text{load}(\text{ivar}) \text{ in} & \\
\text{store}(\text{ivar}, i + 1); & \\
\text{if } i \leq e_2 \text{ then } e_3 \text{ else } \text{break}_1 0
\]

B PROOFS OF §5 THEOREMS

B.1 Correctness of Theorem 2

Given an execution \( G \), let us write \( \text{isMX}(G) \) for the following:

- \( G.lhb \) is a strict total order on \( G.E; \)
- \( G.lhb \subseteq (C^x \times L^x) \cup (L^x \times U^x) \cup (U^x \times L^x); \) and
- \( [L^x]; (G.lhb \cap G.po); [L^x \cup U^x] \subseteq G.po; [U^x]; G.lhb. \)

We are then required to show that for all \( x \) and \( G \), if \( G \in G_c^{\text{MX}} \cap G_w^{\text{MX}} \), then \( \text{isMX}(G_x) \) holds.

Proof. Pick an arbitrary \( x \) and \( G \in G_c^{\text{MX}} \cap G_w^{\text{MX}} \) with \( G_x = (E, po, com, so, lhb) \). Pick an event \( e_0 \) from \( E \) such that it is minimal in \( lhb \); that is, \( \forall e \in E \setminus \{e_0\} \). \( (e, e_0) \neq lhb. \) We first demonstrate that \( e_0 \in C^x \). As \( e_0 \) is minimal with respect to \( lhb \), \( po \subseteq lhb \) and since from the well-formedness of \( G_x \) we know that the first event (in po order) in each thread is either a constructor or lock event, we know that \( e_0 \in C^x \cup L^x \). Let us assume \( e_0 \in L^x \). From the consistency of \( G_x \) we know that there exists \( u \) such that \( (u, e_0) \in \text{com} \cap \text{so} \subseteq lhb. \) This however contradicts our assumption that \( e_0 \) is minimal with respect to \( lhb \) and we thus know \( e_0 \in C^x \).

Let \( E = E_0 \cup E_2 \) with \( E_0 \triangleq \{e_0\} \) and \( E_2 \triangleq E \setminus \{e_0\} \). Let us write \( \text{numL}(S) \) to denote the number of lock events in the set of events \( S \); and write \( \text{numU}(S) \) to denote the number of unlock events in the set of events \( S \). Note that from the well-formedness of \( G_x \) we know that \( \text{numL}(E_0^r) \leq \text{numL}(E_2^r) \). Moreover, from the well-formedness of \( G_x \) (namely the uniqueness of the constructor event) we know that \( E_2^r \cap C^x = \emptyset \). Without loss of generality, let \( m \) denote the number of lock events in \( E_2^r \); i.e. \( \text{numL}(E_0^r) = m. \) We next demonstrate that:

\[
\forall n \in \mathbb{N}. \ \forall E_r, E_o. \\
E = E_o \cup E_r \land \text{numL}(E_r) = n \land \text{numU}(E_r) \leq \text{numL}(E_r) \land E_r \cap C = \emptyset \\
\land [E_r]; lhb; [E_o] = \emptyset \land \text{isMX}(G_x \upharpoonright E_o) \land G_x \upharpoonright E_r \text{ is MX-well-formed on } x \\
\land \exists! e_m. \ e_m = \text{max}(E_o, lhb) \land e_m \in C^x \cup U^x \land \forall e \in E_o \setminus \{e_m\}. \ \text{com}(e) \in E_o \\
\land (\forall l \in L^x \cap E_r. \ (l, e_m) \in (lhb|_{E_o})|_{\text{imm}} \Rightarrow (l, e_m) \in \text{po}) \Rightarrow \text{isMX}(G_x)
\]

(1)

where \( \text{max}(E_o, lhb) \) denotes the maximal elements in \( E_o \) with respect to \( lhb \) – note that \( lhb \) is total on \( E_o \) due to \( \text{isMX}(G_x|_{E_o}) \).

The desired result then follows immediately from (1) and the definitions of \( E_0^r \) and \( E_2^r \). To show (1) we proceed by induction on \( n \).

Base case \( n = 0 \)

From the assumptions of the left-hand side we have \( E_r = \emptyset \) and thus \( E = E_o. \) As such, from
isMX$(G_x|E_x)$ we have isMX$(G_x)$, as required.

**Inductive case** $n=k+1$

$$\forall j \in \mathbb{N}, \forall E_x, E_o,
        j \leq k \land E=E_o \cup E_r \land \text{numL}(E_r)=n \land \text{numU}(E_r) \leq \text{numL}(E_r) \land E_r \cap C = \emptyset
$$

$$\land [E_r]; lhb; [E_o]=\emptyset \land \text{isMX}(G_x|E_x) \land G_x|E_x \text{ is MX-well-formed on } x
$$

$$\land \exists ! e_m. e_m=\text{max}(E_o, lhb) \land e_m \in C^x \cup U^x \land \forall e \in E_o \setminus \{e_m\}. \text{com}(e) \in E_o
$$

$$\land (\forall l \in L^x \cap E_o. (l, e_m) \in (lhb|E_o)|_{\text{limm}} \Rightarrow (l, e_m) \in po)
$$

(I.H.)

Pick an arbitrary $E_r, E_o, e_m$ such that $E=E_o \cup E_r, \text{numL}(E_r)=n, \text{numU}(E_r) \leq \text{numL}(E_r), E_r \cap C = \emptyset,$

$[E_r]; lhb; [E_o]=\emptyset, \text{isMX}(G_x|E_x), G_x|E_x \text{ is MX-well-formed on } x, e_m \text{ is unique and } e_m = \text{max}(E_o, lhb),

$e_m \in C^x \cup U^x, \forall e \in E_o \setminus \{e_m\}. \text{com}(e) \in E_o, \text{ and } (l, e_m) \in (lhb|E_o)|_{\text{limm}} \Rightarrow (l, e_m) \in po.$

Pick an event $e_t \in E_r$ such that it is minimal in lhb; that is, $\forall e \in E_r \setminus \{e_t\}. (e, e_t) \notin lhb$. As $e_t$ is minimal with respect to lhb, $po \subseteq lhb$, and since from the MX-well-formedness of $G_x|E_x$ we know

that the first event (in po order) in each thread is either a constructor or lock event, we know that

$e_t \in C^x \cup L^x$. Moreover, since $E_r \cap C^x = \emptyset$, we know that $e_t \in L^x$. We next demonstrate that

$(e_m, e_t) \in \text{com} = so \subseteq lhb.$

From the consistency of $G_x$ we know there exists $u \in C^x \cup U^x$ such that $(u, e_t) \in \text{com}$. There are then three cases to consider: a) $u = e_m$; or b) $u \in E_o \setminus \{e_m\}$; or c) $u \in E_r$. In case (a) we then have $(e_m, e_t) \in \text{com}$ as required. In case (b) from $\forall e \in E_o \setminus \{e_m\}. \text{com}(e) \in E_o$ we know

there exists another $l \in E_o$ such that $(u, l) \in \text{com}$. Since $l \in E_o$ and $e_t \in E_r$ we know $l \neq e_t$. Consequently, we have $(u, l) \in (u, e_t) \in \text{com},$ contradicting the assumption that $\text{com}$ is functional (since $G_x$ is consistent). In case (c), from the well-formedness of $G_x|E_x$ we know there exists $l$

such that $(l, u) \in po_{\text{limm}}$. We then have $l \to u \text{ com} \to e_t$; that is we have $l \to lhb$

the assumption that $e_t$ is a minimal element of $E_r$ with respect to lhb.

From the MX-well-formedness of $G_x|E_x$, we know there exists $e_u \in U^x$ such that $(e_t, e_u) \in po_{\text{limm}}.$

As $po \subseteq lhb$ and $[E_r]; lhb; [E_o]=\emptyset$, we know that $e_u \in E_r$. Let us then define $E'_r = E_r \cup \{e_t, e_u\}$ and

$E'_r = E_r \setminus \{e_t, e_u\}; i.e. (1) \ E = E'_o \cup E'_r, \text{numL}(E'_r) = -1 = k, \text{numU}(E'_r) \leq \text{numL}(E'_r), E'_r \cap C = \emptyset.$

We next demonstrate that $[E'_r]; lhb; [E'_o]=\emptyset$. As we already have $[E_r]; lhb; [E_o]=\emptyset$ from the assumption, it suffices to show: $\forall e \in E'_r. (e, e_t) \notin lhb$ and $\forall e \in E'_r. (e, e_u) \notin lhb$. The former follows from

the fact that $e_t$ is a minimal element of $E_r$ with respect to lhb. For the latter, we use proceed by

contradiction and assume there exists $e \in E'_r$ such that $(e, e_u) \in lhb$. As $e_u$ is an unlock event without incoming so edges and $(e_t, e_u) \in po_{\text{limm}}$, we know $(e_t, e_u) \in lhb$. This however contradicts our

assumption that $e_t$ is a minimal element of $E_r$ with respect to lhb. We thus have: (2) $[E'_r]; lhb; [E'_o]=\emptyset.$

We next show that isMX$(G_x|E'_o)$ holds. As isMX$(G_x|E_o)$ holds, we know:

lhb is a strict total order on $E_o$;

$G_x|E_o.lhb|_{\text{limm}} \subseteq (C^x \cup L^x) \cup (L^x \cup U^x) \cup (U^x \cup L^x)$; and

$[L^x]; (G_x|E_o.lhb \cup G_x|E_o.po); [L^x \cup U^x] \subseteq G_x|E_o.po; [U^x); G_x|E_o.lhb.$

As $e_m = \text{max}(E_o, lhb)$ and $e_m \to e_t \to e_u$, we know:

lhb is a strict total order on $E'_o$ and

$G_x|E'_o.lhb|_{\text{limm}} \subseteq (C^x \cup L^x) \cup (L^x \cup U^x) \cup (U^x \cup L^x).$

Lastly, since $\forall l \in L^x \cap E_o. (l, e_m) \in (lhb|E_o)|_{\text{limm}} \Rightarrow (l, e_m) \in po$, we have

$[L^x]; (G_x|E_o.lhb \cup G_x|E_o.po); [L^x \cup U^x] \subseteq G_x|E_o.po; [U^x); G_x|E_o.lhb.$

We thus know: (3) isMX$(G_x|E'_o)$ holds.

Since $G_x|E'_o$ is MX-well-formed on $x$, from the definition of $E'_o$ we have (4) $G_x|E'_o$ is MX-well-formed

on $x$. As $e_m = \text{max}(E_o, lhb)$ and $e_m \to e_t \to e_u$, from the definition of $E'_o$ and since $e_u \in U^x$ we
have: (5) $e_u = \max(E'_o, lhb)$ and $e_u \in C^x \cup U^x$. Moreover, as $\forall e \in E_o \setminus \{e_m\}$. com(e) $\in E_o$, and $(e_m, e_l) \in \text{com}$, from the definition of $E'_o$ we have (6) $\forall e \in E'_o \setminus \{e_u\}$. com(e) $\in E'_o$. Lastly, since $e_m = \max(E_o, lhb)$ and $e_m \leftarrow e_l \rightarrow e_u$, and $(e_l, e_u) \in \text{po}$, we know that:

(7) $\forall l \in L^x \cap E'_o$ (l, e_u) $\in (lhb|E'_o)_{\text{lim}} \Rightarrow (l, e_u) \in \text{po}$.

Consequently, from (1)-(7) and (I.H.) we have isMX(G) as required.

\[\square\]

B.2 Correctness of Theorem 3

Given a relation $r$, let us define the following relations for for all $n, m > 0$ and $k \geq 0$:

\[A(r) \triangleq \text{com}^{-1}; r\]

\[B(r) \triangleq \text{com}; r\]

\[C^{n,m}(r) \triangleq (A(r); C^{n-1,m}) \cup (B(r); C^{n,m-1})\]

\[C^k \triangleq C^{k,0} \triangleq \text{com}; r\]

\[C^{n,k} \triangleq C^k \cup (A(r); C^{n-1,k}) \cup (B(r); C^{n,k-1})\]

(2)

Pick an arbitrary execution $G = (E, \text{po}, \text{com}, \text{so}, lhb)$ of the queue library such that irreflexive($C^{n,n}$) holds for all $n > 0$. Let $\text{to}_0 \triangleq lhb$, $\text{TO}_0 = \{\text{to}_0\}$ and for all $i \geq 0$ let us define:

\[\text{TO}_{i+1} \triangleq \left\{ (\text{to}_i \cup \{(d_1, d_2)\}) \cap \{d_1, d_2 \in D^q \wedge (d_1, d_2) \notin \text{to}_i \cup \text{to}_i^{-1}; \exists e_1, e_2. \right\}

\[= \left\{ (\text{to}_i \cup \{(e_1, e_2)\}) \cap \{e_1, e_2 \in C \wedge (e_1, e_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \exists \breve{e}_1, \breve{e}_2. \right\}

\[= \left\{ (\text{to}_i \cup \{(e_2, e_2)\}) \cap \{e_2 \in \text{com} \wedge (e_2, e_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \forall n \in \mathbb{N}^+. \text{irreflexive}(C^{n,n}(\text{to}_{i+1})) \right. \}

\[= \left\{ (\text{to}_i \cup \{(e_1, e_2)\}) \cap \{e_1 \in \text{com} \wedge (e_1, e_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \exists d_1, d_2. \right\}

\[= \left\{ (\text{to}_i \cup \{(e_2, e_2)\}) \cap \{e_2 \in \text{com} \wedge (d_1, d_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \forall n \in \mathbb{N}^+. \text{irreflexive}(C^{n,n}(\text{to}_{i+1})) \right. \}

\[= \left\{ (\text{to}_i \cup \{(e_1, e_2)\}) \cap \{e_1 \in \text{com} \wedge (e_1, e_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \exists d_1, d_2. \right\}

\[= \left\{ (\text{to}_i \cup \{(e_2, e_2)\}) \cap \{e_2 \in \text{com} \wedge (d_1, d_2) \notin \text{to}_i \cup \text{to}_i^{-1} \wedge \forall n \in \mathbb{N}^+. \text{irreflexive}(C^{n,n}(\text{to}_{i+1})) \right. \}

In what follows, we write $A_i$ for $A(\text{to}_i)$, $B_i$ for $B(\text{to}_i)$ and $C^{n,m}_i \triangleq C^{n,m}(\text{to}_i)$, when the choice of $\text{to}_i$ is clear from the context.

We next demonstrate that:

\[\forall i \in \mathbb{N}. \forall \text{to}_i \in \text{TO}_i. \forall n \in \mathbb{N}^+. \text{irreflexive}(C^{n,n}_i)\]

(3)

PROOF. We proceed by induction on $i$.

Base case $i = 0$

Follows immediately from the definition of $\text{TO}_0 = \{\text{to}_0\}$ (as $\text{to}_0 \triangleq lhb$) and the assumption of the lemma.

Inductive case $i = j+1$

\[\forall k \leq j. \forall \text{to}_k \in \text{TO}_k. \forall n \in \mathbb{N}^+. \text{irreflexive}(C^{n,n}_k)\] (I.H.)
Pick an arbitrary $t_0 \in \mathcal{O}_i$. Let us proceed by contradiction and assume that there exists a $C_{i,k}$ cycle for some $k > 0$. Given the definition of $t_0$, there are now five cases to consider. The proof of the second and fourth cases follow immediately from the definition of $t_0$. The proof of the fifth case follows (I.H.).

**Case 1**

We then know there exist $t_0 \in \mathcal{O}_i$ and $e_1, e_2, d_1, d_2$ such that $t_0 = t_0 \cup \{(d_1, d_2)\}$, $d_1, d_2 \in \mathcal{D}^q$, $(d_1, d_2) \notin t_0 \cup t_0^{-1}$, $(e_1, d_1), (e_2, d_2) \in \text{com}$ and $(e_1, e_2) \in t_0$.

As $C_{i,n}^n$ is irreflexive for all $n > 0$, we know the $C_{i,k}$ cycle involves the newly added edge $d_1 \xrightarrow{t_0} d_2$.

That is, either i) $e_1 \rightarrow d_1 \rightarrow d_2 \rightarrow e_1$; or ii) there exist $a, b$ such that $a \rightarrow d_1 \rightarrow d_2 \rightarrow b \rightarrow a$.

In case (i), as $d_2$ is a dequeue event (and cannot have an outgoing $B_j$ edge), we know its outgoing edge is an $A_j$ edge. That is, there exists $a$ such that $d_2 \xrightarrow{\text{com}} e_2 \xrightarrow{t_0} a \xrightarrow{C_{j,k-1}} e_1$. We thus have $a \xrightarrow{t_0} e_1 \rightarrow e_2 \rightarrow a$. As $t_0$ is transitively closed, we have $a \xrightarrow{t_0} e_1 \rightarrow a$. From the definition of $C_{j,k-1}$ we know that it ends with $t_0$. As such, we have $a \rightarrow a$, contradicting (I.H.).

In case (ii), we also have $d_1 \xrightarrow{\text{com}} e_1 \rightarrow e_2 \rightarrow d_2$. We thus have $d_1 \xrightarrow{\text{com}} e_1 \rightarrow e_2 \xrightarrow{t_0} d_2 \xrightarrow{\text{com}} d_2 \xrightarrow{t_0} b \rightarrow a \rightarrow d_1$. That is, we have $d_1 \xrightarrow{A_j} e_2 \rightarrow b \rightarrow a \rightarrow d_1$. From the definition of $C_{j,k}$ we know that it ends with $t_0$. As $t_0$ is transitively closed and $a \rightarrow d_1$, we thus also have $b \rightarrow d_1$. We then have $d_1 \rightarrow e_2 \rightarrow b \rightarrow d_1$. That is, $d_1 \xrightarrow{C_{j,k}} d_1$, contradicting (I.H.).

**Case 3**

We then know there exist $t_0 \in \mathcal{O}_i$ and $e_1, e_2, d_1, d_2$ such that $t_0 = t_0 \cup \{(e_1, e_2)\}$, $e_1, e_2 \in \mathcal{D}^q$, $(e_1, e_2) \notin t_0 \cup t_0^{-1}$, $(e_1, d_1), (e_2, d_2) \in \text{com}$ and $(d_1, d_2) \in t_0$.

As $C_{j,n}^n$ is irreflexive for all $n > 0$, we know the $C_{i,k}$ cycle involves the newly added edge $e_1 \xrightarrow{t_0} e_2$.

That is, either i) $d_1 \rightarrow e_1 \rightarrow e_2 \rightarrow d_1$; or ii) there exist $a, b$ such that $a \rightarrow e_1 \rightarrow e_2 \rightarrow b \rightarrow a$.

In case (i), as $e_2$ is an enqueue event (and cannot have an outgoing $A_j$ edge), we know its outgoing edge is an $B_j$ edge. That is, there exists $a$ such that $e_2 \xrightarrow{\text{com}} d_2 \xrightarrow{t_0} a \xrightarrow{C_{j,k-1}} e_1$. We thus have $a \xrightarrow{t_0} d_1 \rightarrow d_2 \rightarrow a$. As $t_0$ is transitively closed, we have $a \xrightarrow{t_0} d_1 \rightarrow a$. From the definition of $C_{j,k-1}$ we know that it ends with $t_0$. As such, we have $a \rightarrow a$, contradicting (I.H.).

In case (ii), we also have $e_1 \rightarrow d_1 \rightarrow d_2 \rightarrow e_2$. We thus have $e_1 \rightarrow d_1 \rightarrow d_2 \rightarrow e_2 \xrightarrow{t_0} b \rightarrow a \rightarrow d_1$. That is, we have $e_1 \rightarrow e_2 \rightarrow b \rightarrow a \rightarrow e_1$. From the definition of $C_{j,k}$ we know that it ends with $t_0$. As $t_0$ is transitively closed and $a \rightarrow d_1$, we thus also have $b \rightarrow e_1$. We then have $e_1 \rightarrow d_2 \rightarrow b \rightarrow d_1$. That is, $e_1 \xrightarrow{C_{j,k}} e_1$, contradicting (I.H.).


We next demonstrate that:

\[
\forall i \in \mathbb{N}, \forall to_i \in TO_i. \forall to_{i+1} \in TO_{i+1}. to_i = to_{i+1} \Rightarrow \\
\forall d_1, d_2 \in \mathcal{D}^q, (d_1, d_2) \in to_i \cup to_i^{-1} \quad (4)
\]

**Proof.** Pick an arbitrary \( i \in \mathbb{N} \), \( to_i \in TO_i \) and \( to_{i+1} \in TO_{i+1} \) such that \( to_i = to_{i+1} \). We then proceed by contradiction. Let us assume there exist \( d_1, d_2 \in \mathcal{D}^q \) such that \( (d_1, d_2) \notin to_i \cup to_i^{-1} \). Let us write \( e_1 \) for \( \text{com}^{-1}(d_1) \) when it exists (i.e. when \((e_1, d_1) \in \text{com}\)) and write \( e_2 \) for \( \text{com}^{-1}(d_2) \) when it exists (i.e. when \((e_2, d_2) \in \text{com}\)). Let \( S_1 = to_i \cup \{(d_1, d_2)\} \) and \( S_2 = to_i \cup \{(d_2, d_1)\} \). From the definition of \( to_{i+1} \) we then know that \( (e_1, e_2) \notin to_i \) and there exist \( k, n \) such that \( \neg \text{irreflexive}(C^{k, k}(S_1)) \) and \( \neg \text{irreflexive}(C^{n, n}(S_2)) \).

As from (3) we know irreflexive\((C^{k, k}(S_1))\) holds, we know the cycle in \( C^{k, k}(S_1) \) is due to the \((d_1, d_2)\) edge. That is, either 1) \( d_1 \rightarrow d_2 \rightarrow d_1 \); or 2) there exist \( a, b \) such that \( a \rightarrow d_1 \rightarrow d_2 \rightarrow b \rightarrow a \). Similarly, as from (3) we know irreflexive\((C^{n, n}(S_2))\) holds, we know the cycle in \( C^{n, n}(S_2) \) is due to the \((d_2, d_1)\) edge: either a) \( d_2 \rightarrow d_1 \rightarrow d_2 \); or b) there exist \( f, g \) such that \( f \rightarrow d_1 \rightarrow d_1 \rightarrow g \rightarrow f \).

There are now four cases to consider. In case (1.a) we have \( d_1 \rightarrow d_2 \rightarrow d_1 \rightarrow d_1 \), contradicting our result in (3).

In case (1.b) we then have \( g \rightarrow f \rightarrow d_2 \rightarrow d_1 \rightarrow d_1 \rightarrow g \rightarrow g \). As from the definitions of \( C^{n, n}_{i, k} \) and \( C^{k, k}_{i, k} \) we know they end with \( to_i \), we then have \( g \rightarrow d_2 \rightarrow g \). That is, we have \( g \rightarrow g \), contradicting our result in (3).

Similarly, in (2.a) we have \( b \rightarrow a \rightarrow d_1 \rightarrow d_2 \rightarrow b \). As from the definitions of \( C^{n, n}_{i, n} \) and \( C^{k, k}_{i, k} \) we know they end with \( to_i \), we then have \( b \rightarrow d_1 \rightarrow b \). That is, we have \( b \rightarrow b \), contradicting our result in (3).

Lastly, in (2.b) we have \( b \rightarrow a \rightarrow d_1 \rightarrow g \rightarrow f \rightarrow d_2 \rightarrow b \). As from the definitions of \( C^{n, n}_{i, n} \) and \( C^{k, k}_{i, k} \) we know they end with \( to_i \) and \( to_i \) is transitively closed, we then have \( b \rightarrow g \rightarrow b \). That is, we have \( b \rightarrow b \), contradicting our result in (3).

\[\square\]

Similarly, we can demonstrate that:

\[
\forall i \in \mathbb{N}, \forall to_i \in TO_i. \forall to_{i+1} \in TO_{i+1}. to_i = to_{i+1} \Rightarrow \\
\forall e_1, e_2 \in \mathcal{E}^q, (e_1, e_2) \in to_i \cup to_i^{-1} \quad (5)
\]

**Theorem 7.** For a given tuple \((E, po, \text{com}, so, lhb)\), the \( C^{n, n} \) is irreflexive for all \( n \in \mathbb{N}^+ \) iff condition (7) on page 20 holds.

**Proof.** For the \( \Rightarrow \) direction, pick an arbitrary execution \( G = (E, po, \text{com}, so, lhb) \) of the queue library such that irreflexive\((C^{n, n})\) holds for all \( n > 0 \). Let \( m \in \mathbb{N} \) be the least natural number for which there exist \( to_m \in TO_m \) and \( to_{m+1} \in TO_{m+1} \) such that \( to_m = to_{m+1} \). Let us then define \( to \) as an arbitrary extension of \( to_m \) to a strict total order. From the definition of \( to_m \) we then know that \( to \) agrees with \( lhb \). Let \( H \) denote the enumeration of events in \( E \) according to \( to \). From (3), (4) and (5) it is then straightforward to demonstrate that \( \text{fifo}(\epsilon, H) \) holds.
For the $\Leftarrow$ direction pick an arbitrary execution $G = (E, po, com, so, lhb)$ with a sequential enumeration $H$ of $E \setminus C^q$ such that: (i) $H$ agrees with $\text{lhb}$; (ii) $\text{fifo}(e, H)$ holds. Let $to$ denote the strict total order on $E$ induced by $H$. We next demonstrate that:

$$\forall n \in \mathbb{N}^+. C^{n,n}(to) \subseteq to$$

We then have $C^{n,n}(\text{lhb}) \subseteq C^{n,n}(to)$. As such, the irreflexivity of $C^{n,n}$ for an arbitrary $n \in \mathbb{N}^+$ simply follows from above and the irreflexivity of the strict total order $to$.

To show that $\forall n \in \mathbb{N}^+. C^{n,n}(to) \subseteq to$, we proceed by induction on $n$.

**Base case $n = 1$**

Pick an arbitrary $(a, b) \in C^{1,1}(to)$. From the definition of $C^{1,1}(to)$ we then know that there exists $c$ such that either 1) $(a, c) \in \text{com}^{-1} \cap \text{to}$; $\text{com}$ and $(c, b) \in \text{to}$; or 2) $(a, c) \in \text{com} \cap \text{to}$; $\text{com}^{-1}$ and $(c, b) \in \text{to}$.

In both cases from $\text{fifo}(e, H)$ and the definition of $\text{to}$ we know that $(a, c) \in \text{to}$. As in both cases we have $(c, b) \in \text{to}$ and $\text{to}$ is transitively closed, we have $(a, b) \in \text{to}$ as required.

**Base case $n = k + 1$**

$$\forall m \in \mathbb{N}^+. m \leq k \Rightarrow C^{m.m}(to) \subseteq to$$  (I.H.)

Pick an arbitrary $(a, b) \in C^{n,n}(\text{to})$. From the definition of $C^{n,n}$ we know there exists at least one adjacent pair of $A(\text{to})$ and $B(\text{to})$ edges. That is, there exists $i, j, c, d$ such that: $a \xrightarrow{C^{i,j}(\text{to})} c \xrightarrow{C^{1,1}(\text{to})} d \xrightarrow{C^{k-i,k-j}(\text{to})} b$. As such, from the proof of the base case we know $a \xrightarrow{C^{i,j}(\text{to})} c \xrightarrow{to} d \xrightarrow{C^{k-i,k-j}(\text{to})} b$. As the $C^{i,j}(\text{to})$ path ends with a $\text{to}$ edge for all $i, j$ and $\text{to}$ is transitive, we have $a \xrightarrow{C^{i,j}(\text{to})} d \xrightarrow{C^{k-i,k-j}(\text{to})} b$.

That is, $a \xrightarrow{C^{k}(\text{to})} b$. Consequently, from (I.H.) we have $(a, b) \in \text{to}$, as required.

## ADDITIONAL SPECIFICATIONS

### C.1 Multiple-Readers-Single-Writer Lock Library Specification

We consider a multiple-readers-single-writer (MRSW) lock library with six methods: 1) $\text{new-MSRW}(\cdot)$, for constructing a lock; 2) $\text{wlock}(x)$, for acquiring $x$ in writer mode; 3) $\text{wunlock}(x)$, for releasing the writer lock on $x$; 4) $\text{rlock}(x)$, for acquiring $x$ in reader mode; 5) $\text{runlock}(x)$, for releasing a reader lock on $x$; and 6) $\text{plock}(x)$, for promoting a reader lock on $x$ to a writer one. A reader lock on $x$ is promoted once all reader locks on $x$ (except that of the promoter) are released.

The MRSW interface is $\langle \mathcal{M}^{\text{RW}}, \mathcal{M}^{\text{cRW}}, \mathcal{1\text{ocRW}} \rangle$, where $\mathcal{M}^{\text{cRW}} \equiv \bigcup_{x \in \text{Loc}} \mathcal{M}^{\text{c}}$ with $\mathcal{M}^{\text{c}} \equiv \{\text{new-MSRW}(x)\}$; $\mathcal{M}^{\text{RW}} \equiv \bigcup_{x \in \text{Loc}} \mathcal{M}^{x}$ with $\mathcal{M}^{x} \equiv \mathcal{M}^{\text{c}} \cup \{\text{wlock}(x), \text{wunlock}(x), \text{rlock}(x), \text{runlock}(x), \text{plock}(x)\}$; and $\forall l \in \mathcal{M}^{x}$. $\text{1oc}^{\mathcal{M}^{x}}(l) = \{x\}$. For an MRSW lock at location $x$, we define the following event sets:

$$\mathcal{WL}^{x} \equiv \{e \mid \text{lab}(e) = \text{wlock}(x)\} \quad \mathcal{WU}^{x} \equiv \{e \mid \text{lab}(e) = \text{wunlock}(x)\}$$

$$\mathcal{RL}^{x} \equiv \{e \mid \text{lab}(e) = \text{rlock}(x)\} \quad \mathcal{RU}^{x} \equiv \{e \mid \text{lab}(e) = \text{runlock}(x)\}$$

$$\mathcal{CL}^{x} \equiv \{e \mid \text{lab}(e) = \text{new-MSRW}(x)\} \quad \mathcal{PL}^{x} \equiv \{e \mid \text{lab}(e) = \text{plock}(x)\}$$

Let $\mathcal{L}^{x} \equiv \mathcal{WL}^{x} \cup \mathcal{RL}^{x} \cup \mathcal{PL}^{x}$ and $\mathcal{U}^{x} \equiv \mathcal{WU}^{x} \cup \mathcal{RU}^{x}$.

A tuple $(E, po, \text{com}, \text{so}, \text{lhb})$ is RW-consistent on $x$ iff:

1. $(1)$ there is at most one constructor event: $E^{c} = 0 \lor \exists c \in C^{x}, E^{c} = \{c\}$;
2. $(2)$ com relates matching lock events: $\text{com} = \text{com}_{w} \cup \text{com}_{r} \cup \text{com}_{p}$ with:

$$\text{com}_{w, \text{com}_{p}} \subseteq (\mathcal{C}^{x} \cup \mathcal{U}^{x}) \times \mathcal{WL}^{x} \quad \text{com}_{r} \subseteq (\mathcal{C}^{x} \cup \mathcal{WU}^{x}) \times \mathcal{RL}^{x}$$
(3) each event is matched by at most one lock except for reader locks, i.e. for all $e, e_1, e_2$:

$$e_1 \not= e_2 \land (e, e_1), (e, e_2) \in \text{com} \Rightarrow (e_1, e_2) \in \mathcal{RL}^x \lor (e_1 \in \mathcal{RL}^x \land e_2 \in \mathcal{PL}^x) \lor (e_1 \in \mathcal{PL}^x \land e_2 \in \mathcal{RL}^x)$$

(4) each lock is matched by at most one event: $\text{com}^{-1}$ is functional;

(5) all lock events are matched: $E \cap \mathcal{L}^x = \text{rng}(\text{com})$; and

(6) every matching edges is synchronising: $\text{so} = \text{com}$.

Intuitively, $\text{com}$ describes the order of lock acquisition. For each $l \in \mathcal{WL}^x$ with $(e, l) \in \text{com}_w$, when $e \in \mathcal{U}^x$ then $e$ denotes the unlock event releasing the lock immediately before it is acquired by $l$; when $e \in \mathcal{C}^x$ then $e$ denotes the constructor initialising the lock, and thus $l$ corresponds to the very first $\text{wlock}(x)$ call. As $l$ acquires the lock in the (exclusive) writer mode, no other lock may be matched with its predecessor $e$ (see (3)). For each $l \in \mathcal{RL}^x$ with $(e, l) \in \text{com}_r$, the case where $e \in \mathcal{C}^x$ can be described analogously; when $e \in \mathcal{WU}^x$, then $e$ denotes the last time the lock in writer mode was released. As MRSW locks allow for multiple reader locks simultaneously, multiple events in $\mathcal{RL}^x$ may be matched with the same event in $\mathcal{WL}^x$ (see (3)). Lastly, for each $l \in \mathcal{PL}^x$ with $(e, l) \in \text{com}_p$, when $e \in \mathcal{RU}^x$ then $e$ denotes the event releasing the last reader lock on $x$; when $e \in \mathcal{WU}^x$, then $e$ denotes the last time the lock in writer mode was released; when $e \in \mathcal{C}^x$, then $l$ denotes the first $\text{plock}(x)$ call, prior to any writer lock acquisition. In the latter two cases, at the time of promoting the lock via $l$, no other reader locks (other than that being promoted) are held on $x$ and thus $l$ follows the last writer lock release or the constructor. As such, the event acquiring a reader lock as well as its subsequent promotion may both be matched by $e$ (see (3)).

A tuple $(E, po, \text{com}, so, lhb)$ is $\text{RW}$-well-formed on $x$ iff:

- $\min(po) \subseteq \mathcal{C}^x \cup \mathcal{L}^x$ and $\text{po}|_{\text{limm}(E^c)} \subseteq \mathcal{L}^x$;
- $[\mathcal{RL}^x]: \text{po}|_{\text{limm}} = \text{po}|_{\text{limm}}; [\mathcal{PL}^x \cup \mathcal{RU}^x]$; and
- $[\mathcal{WL}^x \cup \mathcal{PL}^x]: \text{po}|_{\text{limm}} = \text{po}|_{\text{limm}}; [\mathcal{WU}^x]$.

Analogously to mutex well-formedness, MRSW well-formedness requires that the first call in each thread be either to the constructor or for lock acquisition, a constructor call be immediately followed (in $po$) by a lock acquisition; each reader unlock or lock promotion call be immediately preceded (in $po$) by a reader lock acquisition call and vice versa; and each writer unlock call be immediately preceded (in $po$) by a writer lock acquisition or lock promotion call and vice versa.

**Strong MRSW-Consistency.** Note that there are no $\text{com}$ edges between events of reader locks. Consequently, as $so = \text{com}$, reader lock events do not synchronise with one another. This is to keep the specification as general as possible and admit certain MRSW implementations with weaker guarantees such as those discussed in §E. Nevertheless, we can strengthen this specification by extending the domain of $\text{com}_p$ as: $\text{com}_p \subseteq (\mathcal{C}^x \cup \mathcal{WU}^x \cup \mathcal{RL}^x \cup \mathcal{RU}^x) \times \mathcal{RL}^x$, and the domain of $\text{com}_p$ as: $\text{com}_p \subseteq (\mathcal{RU}^x \cup \mathcal{RL}^x) \times \mathcal{PL}^x$. That is, for each $l \in \mathcal{RL}^x$ with $(e, l) \in \text{com}_p$, the case where $e \in \mathcal{C}^x$ can be described as before; when $e \in \mathcal{WU}^x \cup \mathcal{RU}^x$ then $e$ denotes the (reader or writer) unlock event releasing the lock immediately before it is acquired by $l$; when $e \in \mathcal{RL}^x$, then $e$ denotes the event acquiring a reader lock on $x$ immediately before it is also acquired by $l$. This is because MRSW locks allow for multiple reader locks simultaneously. Note that this is in contrast to the $\text{com}_p$ edges above. In particular, in the above (weaker) description, when $e \in \mathcal{WU}^x$, then $e$ denotes the last release of the writer lock on $x$ before its acquisition by $l$; i.e. $e$ may not be immediately preceding $l$ and may be interleaved by several reader (lock or unlock) events. Similarly, for each $l \in \mathcal{PL}^x$ with $(e, l) \in \text{com}_p$, when $e \in \mathcal{RU}^x$, then $e$ denotes the reader unlock event releasing the lock immediately before it is acquired by $l$; when $e \in \mathcal{RL}^x$, then $e$ denotes the very reader lock being promoted: no other reader locks on $x$ have been acquired between its acquisition ($e$) and its promotion ($l$). As such, in contrast to (3), we require that $\text{com}$ be functional.

We thus denote $(E, po, \text{com}, so, lhb)$ as strongly RW-consistent on $x$ iff: (1) as above; (2) $\text{com} = \text{com}$; (3) each event is matched by at most one lock except for reader locks, i.e. for all $e, e_1, e_2$:

$$e_1 \not= e_2 \land (e, e_1), (e, e_2) \in \text{com} \Rightarrow (e_1, e_2) \in \mathcal{RL}^x \lor (e_1 \in \mathcal{RL}^x \land e_2 \in \mathcal{PL}^x) \lor (e_1 \in \mathcal{PL}^x \land e_2 \in \mathcal{RL}^x)$$

(4) each lock is matched by at most one event: $\text{com}^{-1}$ is functional;

(5) all lock events are matched: $E \cap \mathcal{L}^x = \text{rng}(\text{com})$; and

(6) every matching edges is synchronising: $\text{so} = \text{com}$.
We consider a set library with four methods: new-set(), for constructing a new set; add(s, v) for adding v to the set at s; rem(s, v) for removing v from the set at s; and is-in(s, v) for checking the membership of v in the set at s. An add(s, v) call successfully adds v to s only if it does not already contain v; analogously, a rem(s, v) successfully removes v if s contains v. Similarly, the return value of is-in(s, v) indicates whether s contains v. As such, all three operations return a boolean reflecting their outcomes. We present a set specification in our framework below. Once again, we forgo a strong specification with a total execution order in the linearisability style, and opt instead for a weaker specification more suitable for the WMC setting.

We define the set interface as \( \langle M_{\text{Set}}, M_{\text{Set}}^c, 1_{\text{ocSet}} \rangle \), where \( \forall l \in M^S \cdot 1_{\text{ocMX}}(l) = \{s\} \)

\[
\begin{align*}
M_{\text{Set}}^c \triangleq & \bigcup_{s \in \text{Loc}} M_{\text{c}}^c \\
M_{\text{Set}} \triangleq & \bigcup_{s \in \text{Loc}} M^s \\
M^s \triangleq & M_{\text{c}}^s \cup \{\text{add}(s, v, o), \text{rem}(s, v, o), \text{is-in}(s, v, o) \mid v \in \text{Val} \land o \in \{\top, \bot\}\}
\end{align*}
\]

For a set at location s, we define the following sets of events:

\[
\begin{align*}
C^s \triangleq & \{e \mid \text{lab}(e) = \text{new-set}(s)\} \\
I^{s,v,o} \triangleq & \{e \mid \text{lab}(e) = \text{is-in}(s, v, o)\} \\
A^{s,v,o} \triangleq & \{e \mid \text{lab}(e) = \text{add}(s, v, o)\} \\
R^{s,v,o} \triangleq & \{e \mid \text{lab}(e) = \text{rem}(s, v, o)\}
\end{align*}
\]

Let \( A^{s,\top} \triangleq \bigcup_{v \in \text{Val}} A^{s,v,\top} \) and \( A^{s,\bot} \triangleq A^{s,\top} \cup A^{s,\bot} \); let us similarly define \( R^{s,\top}, R^{s,\bot}, I^{s,\top}, I^{s,\bot} \). A tuple \( \langle E, po, \text{com}, so, \text{lbh} \rangle \) is set-consistent on s iff:

1. there is at most one constructor event: \( E^c = \emptyset \lor \exists c \in C^s \cdot E^c = \{c\} \);
2. \text{com} relates matching events: \( \text{com} \triangleq \text{com}_r \cup \text{com}_f \cup \text{com}_j \), where

\[
\begin{align*}
\text{com}_r \subseteq & \bigcup_{v \in \text{Val}} A^{s,v,\top} \times R^{s,v,\top} \\
\text{com}_f \subseteq & \bigcup_{v \in \text{Val}} A^{s,v,\top} \times I^{s,v,\top} \\
\text{com}_j \subseteq & \bigcup_{v \in \text{Val}} A^{s,v,\top} \times A^{s,v,\bot}
\end{align*}
\]

3. every remove, membership and failed add is matched by at most one add: \( \text{com}^{-1} \) is functional;
4. every add is matched by at most one remove: \( \text{com}_r \) is functional;
5. every unmatched remove or membership returns \( \bot: (E \cap (R^{s,\top} \cup I^{s,\top})) \setminus \text{rng}(\text{com}) \subseteq R^{s,\bot} \cup I^{s,\bot} \);
6. every failed add event is matched: \( (E \cap A^{s,\bot}) \setminus \text{rng}(\text{com}) = \emptyset \);
7. every matching edge is synchronising: \( so = \text{com} \); and
8. value v cannot be added twice before being removed first; that is, for all v:

\[
[A^{s,v,\top}] ; \text{lbh} ; [A^{s,v,\top}] ; \text{lbh} ; \text{com}^{-1} \text{ is irreflexive} \quad \text{and} \quad [A^{s,v,\top}] \setminus \text{rng}(\text{com}_r) ; \text{lbh} ; [A^{s,v,\top}] = \emptyset
\]

9. adding a value must not fail when it is already removed: \( \text{com}_f ; \text{lbh} ; \text{com}^{-1} \) is irreflexive;
10. removing a value must not fail when the value is yet to be removed:

\[
\forall v. [R^{s,v,\top} \cup I^{s,v,\top}] ; \text{com}^{-1} \cup \text{com}_f ; \text{lbh} ; [R^{s,v,\top}] ; \text{lbh} \text{ is irreflexive};
\]
11. membership check for a value must not fail when the value is yet to be removed:

\[
\forall v. [R^{s,v,\top} \cup I^{s,v,\top}] ; \text{com}^{-1} \cup \text{com}_f ; \text{lbh} ; [I^{s,v,\bot}] ; \text{lbh} \text{ is irreflexive};
\]
12. a remove or membership with a previous unmatched add cannot return \( \bot:\)

\[
\forall v. [A^{s,v,\top} \setminus \text{dom}(\text{com})] ; \text{lbh} ; [R^{s,v,\top} \cup I^{s,v,\bot}] = \emptyset
\]
\begin{align*}
\text{new-mutex}() & \triangleq \\
\text{lock}(x) & \triangleq \\
\text{let } x = \text{alloc()} \text{ in } x & \text{loop} \\
\text{unlock}(x) & \triangleq \\
\text{if compare-set}(x, 0, 1, \text{acq}) \text{ then } & \text{store}(x, 0, \text{rel}); \\
& \text{break;} \\
\end{align*}

Fig. 7. A simple mutex implementation using a release acquire register

(13) successful remove events cannot match with adds which are already removed:
\[ [R^{s_i, r_T}]; \text{lhb}; [R^{s_i, r_T}]; \text{lhb}; \text{com}_r^{-1} \text{ is irreflexive}; \]

(14) successful membership events cannot match with adds which are already removed:
\[ \text{com}_i^{-1}; \text{com}_r; \text{lhb} \text{ is irreflexive}. \]

Intuitively, \((a, r) \in \text{com}_r\) denotes that \(r\) removes a value added by \(a\). As such, each successful remove is matched by exactly one add, and each successful add is matched by at most one remove.

Similarly, \((a, i) \in \text{com}_i\) denotes that \(i\) observes the value added by \(a\). Each successful membership is thus matched by exactly one add. However, as membership calls leave the set unchanged, multiple membership events may be matched by the same add. Lastly, \((a, f) \in \text{com}_f\) denotes that \(f\) fails to add its value to the set as it has been previously added by \(a\). Each failed add is hence matched by exactly one successful add, whilst each successful add may be matched by several failed adds.

**Definition C.2 (Set library).** The set library is \(L^{\text{Set}}\triangleq (M^{\text{Set}}, M^{\text{Set}}, \text{loc}^{\text{Set}}, G^{\text{Set}}, G^{\text{Set}}, G^{\text{Set}}), \text{where} \ G^{\text{Set}}_c \triangleq \{ G \in G^{\text{Set}} | \forall s, G_s \text{ is set-consistent on } s \} \).

**D A SOUND MUXTE IMPLEMENTATION**

In Fig. 7 we present a simple implementation of mutex locks using release-acquire registers. As we formalise in Thm. 8, this implementation is sound with respect to the mutex library \(L^{\text{MX}}\).

**Theorem 8.** The mutex implementation in Fig. 7 is a sound implementation of \(L^{\text{MX}}\).

**Proof.** The full proof is mechanised in the Coq proof assistant and is available as auxiliary material. \(\square\)
new-MSRW() ≜
let x = alloc() in x

lock(x) ≜
loop
  let c = load(x, rlx) in
  if is-even(c) then
    if compare-set(x, c, c+2, acqrel) then
      break \(_1\)
  else
    fetch-add(x, -2, rel)

unlock(x) ≜
store(x, 0, rel);

\(wlock(x) \triangleq\)
loop
  if compare-set(x, 0, 1, acq) then
    break \(_1\)

\(plock(x) \triangleq\)
loop
  let b = 0 in
  if b then
    let c = load(x, acq) in
    if c == 3 then break \(_1\)
  else
    let c = load(x, rlx) in
    if is-even(c) then
      if compare-set(x, c, c+1, acq) then
        b = 1
Fig. 8. An implementation of strong MRSW locks using a C11 register

E SOUND MRSW LOCK IMPLEMENTATIONS

E.1 A Sound Strong MRSW Lock Implementation

We present a strong MRSW lock implementation using the C11 registers: a lock at location \(x\) is represented as a C11 register at \(x\). The state of a lock \(x\) is represented by an integer value. A lock \(x\) may hold either:

1. Value 0, denoting that the lock is free (not held in read or write mode); or
2. Value 1, denoting that the lock is held (exclusively) in write mode; or
3. An even value \(2n\) with \(n > 0\), denoting that \(x\) is held in (shared) read mode by \(n\) readers; or
4. An odd value \(2n+3\) with \(n > 0\), denoting that the lock is currently being promoted, awaiting the release of \(n\) readers; or
5. Value 3, denoting that the lock is successfully promoted.

As such, the implementation of \(wlock(x)\) simply spins until it can atomically update (via atomic \(\text{compare-set}\)) the value of \(x\) from zero (free) to one (acquired in write mode). Dually, the implementation of \(unlock(x)\) simply releases the write lock by atomically assigning \(x\) to zero.

The implementation of \(plock(x)\) is more involved. As multiple readers may attempt to promote their reader locks simultaneously, promotion is granted on a ‘first-come-first-served’ bases. As such, the implementation of \(plock(x)\) first reads the value of \(x\) (the else branch). If \(x\) holds an odd value, then another reader is currently promoting \(x\) and thus promotion must be retried. On the other hand, if \(x\) holds an even value, then its value is atomically incremented (to an odd value) to signal the intention to promote. Moreover, \(b\) is set to 1 to indicate that the intention to promote has been successfully registered and promotion can enter the next waiting phase. The implementation then proceeds by spinning until all other readers have released their locks on \(x\) (i.e. \(x == 3\)), at which point \(x\) is successfully promoted and the implementation terminates. Note that once a reader has signalled its intention to promote \(x\) (by incrementing \(x\) to an odd value), any other such attempt to promote the lock on \(x\), as well as calls to acquire it in read mode will fail thereafter until such time that \(x\) is released by its current promotor.

The implementation of \(lock(x)\) is similar. It first checks whether \(x\) is odd (held in write mode or being promoted). If so then the implementation spins until \(x\) is even (free or held in read mode), at which point its value is incremented by two (to increase the number of readers by one) using the...
We must next find \( \text{com}'' \), so'' such that \( \langle E'', \text{po}'', \text{com}'', \text{so}'', \text{lhba}'' \rangle \in \text{L}^{\text{SRW}}_c \), where \( \text{lhba}'' \) is the same as \( \text{lhba} \) in Def. 11.

For each location \( x \), without loss of generality let us assume \( G \) contains \( n_x \) read lock calls on \( x \), \( m_x \) write lock calls on \( x \), \( p_x \) lock promotion calls on \( x \). Let us enumerate each of read lock calls, read unlock calls, write lock calls, write unlock calls and lock promotion calls arbitrarily. Note that:

- the MRSW constructor at location \( x \) contains a single event \( c_x \) where \( \text{lab}(c_x) = \text{alloc}(x, 0) \).
- For each ith read lock operation on \( x \), the \( G \) contains the trace \( \theta^r_i(x) = r f^{*} \rightarrow r r^{*} \rightarrow r l \), where \( r f^{*} \) denotes the events of those iterations that failed to acquire the reader lock, \( \text{lab}(rr) = \text{load}(rlx, x, rvl) \), \( \text{lab}(rl) = \text{compare-set}(\text{acqrel}, x, rvl, rvl+2) \), and \( rvl \) is an even value.
- For each ith read unlock operation on \( x \), the \( G \) contains the trace \( \theta^u_i(x) \) with a single event \( ru \), where \( \text{lab}(ru) = \text{fetch-add}(\text{rel}, x, v_i, v_i-2) \) for some \( v_i \).
- For each ith write lock operation on \( x \), the \( G \) contains the trace \( \theta^w_i(x) = w f^{*} \rightarrow w l \), where \( w f^{*} \) denotes the events of those iterations that failed to acquire the writer lock and \( \text{lab}(wl) = \text{compare-set}(\text{acq}, x, 0, 1) \).
- For each ith write unlock operation on \( x \), the \( G \) contains the trace \( \theta^u_i(x) \) with a single event \( wu \), where \( \text{lab}(wu) = \text{store}(\text{rel}, x, 0) \).
- For each ith read lock promotion operation on \( x \), the \( G \) contains the trace \( \theta^p_i(x) = p f^{*} \rightarrow p f^{2*} \rightarrow p l \), where \( p f^{*} \) denotes the events of those iterations that failed to indicate lock promotion, \( p f^{2*} \) denotes the events of those iterations that failed to promote the lock, \( \text{lab}(pr) = \text{load}(rlx, x, pv_i) \), \( \text{lab}(pi) = \text{compare-set}(\text{acqrel}, x, pv_i, pv_i+1) \), \( pv_i \) is an even value, and \( \text{lab}(pl) = \text{load}(\text{acq}, x, 3) \).

Let \( \text{imp}(...) : E'' \rightarrow E \) be defined as:

\[
\text{imp}(e) \triangleq \begin{cases} 
  c_x & \exists x. e = f(c_x) \\
  \theta^r_i(x).rl & \exists i. x. e = f(\theta^r_i(x).rl) \\
  \theta^u_i(x).ru & \exists i. x. e = f(\theta^u_i(x).ru) \\
  \theta^w_i(x).wl & \exists i. x. e = f(\theta^w_i(x).wl) \\
  \theta^u_i(x).wu & \exists i. x. e = f(\theta^u_i(x).wu) \\
  \theta^p_i(x).pl & \exists i. x. e = f(\theta^p_i(x).pl) 
\end{cases}
\]

Let us define: so'' = com'' with com'' defined as follows:

\[
\text{com}'' \triangleq \{ (e_1, c_x), (e_2, c_x), (e_3, \theta^h_k.pi), (e_4, \theta^h_k.pi) \} \quad \text{where} \quad \begin{cases} 
  x \in \text{Loc} \land (\text{imp}(e_1), \text{imp}(e')) \in \text{com} \land \exists i. \text{imp}(e'_i) \in \theta^r_i(x) \\
  \land (\text{imp}(e_2), \text{imp}(w_f^{*})) \in \text{com} \land \exists i. \text{imp}(e'_i) \in \theta^w_i(x) \\
  \land \exists k, e_3. (\theta^h_k.pi, \theta^h_k.pi) \in \text{com} \land (e_3, \theta^h_k.pi) \in \text{com} \\
  \land \exists k, e_4. (e_4, \theta^h_k.pi) \in \text{com} \land e_4 \neq \theta^h_k.pi 
\end{cases}
\]
To show that $G' = \langle E', \mathit{po}'', \mathit{com}'' \mathit{so}'', \mathit{lhb}'' \rangle \in L^{\mathit{RW}} G_c$, we are then required to show for all $x \in \mathit{Loc}$, $G'_x$ is RW-consistent on $x$. Pick an arbitrary $x \in \mathit{Loc}$ and let $G'_x = \langle E', \mathit{po}', \mathit{com}', \mathit{so}', \mathit{lhb}' \rangle$. We then need to show:

(1) $G'_x$ contains at most one constructor event;

(2) $\mathit{com}' = \mathit{com}_w \cup \mathit{com}_r \cup \mathit{com}_p$ with:

\[
\begin{align*}
\mathit{com}_w &\subseteq (C^x \cup U^x) \times W^L x; \quad \mathit{com}_r \subseteq (C^x \cup W U^x \cup RL^x \cup RU^x) \times RL^x; \quad \text{and} \quad \mathit{com}_p \subseteq (RL^x \cup RU^x) \times PL^x
\end{align*}
\]

(3) $\mathit{com}'$ is functional;

(4) $\mathit{com}^{-1}$ is functional;

(5) $E \cap L^x = \mathit{rng}(\mathit{com}')$; and

(6) $\mathit{so}' = \mathit{com}'$.

Parts (1), (5) and (6) follow immediately from the construction of $G'_x$. For part (2), pick an arbitrary $(a, b) \in \mathit{com}'$. From the definition of $\mathit{com}'$ we then know that there exists $i$ such that either $a) b = rl^x_i$ and $(\mathit{imp}(a), \mathit{imp}(rl^x_i)) \in \mathit{com};$ or $b) b = wl^x_i$ and there exists $e$ such that $a = f_a(e)$, $(\theta_i^{pl(x)} . pi, \mathit{imp}(pl^x_i)) \in \mathit{com}$ and $(e, \theta_i^{pl(x)} . pi) \in \mathit{com};$ or $c) b = pl^x_i$ and there exists $e$ such that $a = f_a(e)$, $(e, \mathit{imp}(pl^x_i)) \in \mathit{com}$ and $e \neq \theta_i^{pl(x)} . pi$.

In case (a), since the value read by $\mathit{imp}(rl^x_i)$ is even, from the implementation encapsulation (Thm. 1) we know there exists $j$ such that $\mathit{imp}(a) = c_x$ or $\mathit{imp}(a) = \theta_j^{ru(x)}.rl$ or $\mathit{imp}(a) = \theta_j^{ru(x)}.ru$ or $\mathit{imp}(a) = \theta_j^{ru(x)}.wu$. As such, we know that either $a \in C^x \cup RL^x \cup RU^x \cup WU^x$, as required.

Similarly in case (b), since the value read by $\mathit{imp}(wl^x_i)$ is zero, from the implementation encapsulation (Thm. 1) we know there exists $j$ such that $\mathit{imp}(a) = c_x$ or $\mathit{imp}(a) = \theta_j^{ru(x)}.ru$ or $\mathit{imp}(a) = \theta_j^{ru(x)}.wu$. As such, we know that either $a \in C^x \cup RU^x \cup WU^x$, as required.

In case (c) we then know that the value read by $\theta_i^{pl(x)} . pi$ is 2; and thus from the implementation encapsulation (Thm. 1) we know $e$ is either a read unlock event or a read lock event. That is, there exists $j$ such that $e = \theta_j^{ru(x)}.ru$ or $e = \theta_j^{ru(x)}.rl$, as required.

In case (d), since the value read by $\mathit{imp}(pl^x_i)$ is 3, from the implementation encapsulation (Thm. 1) we know there exists $j$ such that $e = \theta_j^{ru(x)}.ru$, as required.

For part (3) we proceed by contradiction. Let us assume there exists $e, e_1, e_2$ such that $(e, e_1)(e, e_2) \in \mathit{com}$. We then know there exists $i$ such that either a) $e_1 = rl^x_i$; or b) $e_1 = wl^x_i$; or c) $e_1 = pl^x_i$. Similarly, we know there exists $j$ such that either i) $e_1 = rl^x_j$; or ii) $e_2 = wl^x_i$; or iii) $e_2 = pl^x_j$.

In case (a–i), from the definition of $\mathit{com}'$ we know $(\mathit{imp}(e), \mathit{imp}(rl^x_j)), (\mathit{imp}(e), rl^x_j) \in \mathit{com}$. However, since both $rl^x_i$ and $rl^x_j$ are atomic update operations, from the C11 consistency we know that there exists $\mathit{mo}$ such that $(\mathit{imp}(e), \mathit{imp}(rl^x_j)), (\mathit{imp}(e), \mathit{imp}(rl^x_j)) \in \mathit{mo}_{\mathit{limm}}$, and that either $\mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j)$ or $\mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j)$. In the former case we have $\mathit{imp}(e) \rightarrow \mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j)$ and thus $(\mathit{imp}(e), \mathit{imp}(rl^x_j)) \in \mathit{mo}_{\mathit{limm}}$, leading to contradiction. Similarly, in the latter case we have $\mathit{imp}(e) \rightarrow \mathit{imp}(rl^x_j) \rightarrow \mathit{imp}(rl^x_j)$ and thus $(\mathit{imp}(e), \mathit{imp}(rl^x_j)) \not\in \mathit{mo}_{\mathit{limm}}$, leading to contradiction.

The proof of the remaining cases (a–ii)–(a–iii), (b–i)–(b–iii) and (c–i)–(c–iii) are analogous and are omitted here.

Part (4) follows from the definition of $\mathit{com}'$ and the functionality of $\mathit{com}^{-1}$ for C11 registers. □
\text{new-MSRW}(x) \triangleq \\
\text{let } x = \text{alloc}(K) \text{ in } x \\
\text{rlock}(x) \triangleq \\
\text{let } t = \text{get-tid}() \text{ in } \\
\text{loop} \\
\text{if } \text{compare-set}(x[t], 0, 2, \text{acq}) \text{ then } \\
\text{break}(); \\
\text{runlock}(x) \triangleq \\
\text{let } t = \text{get-tid}() \text{ in } \\
\text{store}(x[t], 0, \text{rel}) \\
\text{wunlock}(x) \triangleq \\
\text{for } i = 0 \text{ to } K \text{ do } \\
\text{store}(x[i], 0, \text{rel}); \\
\text{wlock}(x) \triangleq \\
\text{for } i = 0 \text{ to } K \text{ do } \\
\text{loop} \\
\text{if } \text{compare-set}(x[i], 0, 1, \text{acq}) \text{ then } \\
\text{break}(); \\
\text{plock}(x) \triangleq \\
\text{let } t = \text{get-tid}() \text{ in } \\
\text{let } c = \text{load}(x[t], \text{rlx}) \text{ in } \\
\text{if } c == 2 \text{ then } \\
\text{if } t == 0 \text{ then } \text{store}(x[t], 1, \text{rlx}) \\
\text{else} \\
\text{loop} \\
\text{if } \text{compare-set}(x[0], 0, 1, \text{acq}) \text{ then } \\
\text{store}(x[t], 1, \text{rel}); \text{break}(); \\
\text{for } i = 1 \text{ to } K \text{ do } \\
\text{if } (i \neq t) \text{ then } \\
\text{loop} \\
\text{if } \text{compare-set}(x[i], 0, 1, \text{acq}) \text{ then } \\
\text{break}();

\textbf{Fig. 9.} An implementation of weak MRSW locks (for } K \text{ threads) using a } K \text{-array of C11 registers}

\subsection{E.2 A Sound Weak MRSW Lock Implementation}

Our second MRSW lock implementation is similarly implemented using C11 registers and is given in Fig. 9. In this implementation, a lock at location } x \text{ is represented as an } \textit{ordered} \text{ map of size } K \text{ at location } x \text{. The map at } x \text{ contains one entry per thread (when there are } K \text{ threads present) as follows. For each thread with identifier } \tau \text{, the } x[\tau] \text{ map entry records the current locking privileges of } \tau \text{ on } x \text{. More concretely, when } x[\tau] = 0 \text{, then } \tau \text{ does not hold the } x \text{ lock; when } x[\tau] = 2 \text{, then } \tau \text{ holds } x \text{ in read mode; and when } x[\tau] = 1 \text{, then some thread (either } \tau \text{ or another thread) either holds } x \text{ in write mode, or it is in the process of acquiring } x \text{ in write mode. The } x \text{ lock is held in write mode only when all entries in } x \text{ are mapped to one. As we describe shortly, for thread } \tau \text{ to acquire } x \text{ in write mode, it must inspect each entry in } x \text{ (in order), wait for it be free (zero) and then set it to one. In our implementation, we assume that the thread identifier can be obtained by calling } \text{get-tid}(). \text{ We identify the top-most thread by } \tau = 0; \text{ as such, the entry of top-most thread in each map is ordered before all other threads.}

\text{We proceed with a more detailed explanation of our implementation after introducing our map notation.}

\textbf{Map notation.} \text{ We write } 1 \text{ to denote a map where all entries have value } 1; \text{ similarly, we write } 0 \text{ to denote a map where all entries have value } 0. \text{ Lastly, we write } S \subseteq x, \text{ to denote that the values held in map } x \text{ are a superset of } S. \text{ The lock map } x \text{ associated with location } x \text{ can be in one of the following states:}

\begin{itemize}
\item $x = 0$ \text{ when } x \text{ is free;}
\item $x = 1$ \text{ when } x \text{ is held in write mode;}
\item $\{2\} \subseteq x$ \text{ when } x \text{ is held in read mode (by those threads } \tau \text{ where } x[\tau] = 2).$
\end{itemize}
When thread $\tau$ calls $rlock(x)$, it simply spins until the lock is free ($x = 0$ and thus $x[\tau] = 0$), at which point it acquires it in read mode by setting $x[\tau]$ to two. Dually, when $\tau$ calls $runlock(x)$ it simply sets $x[\tau]$ to zero.

Analogously, when $\tau$ calls $wlock(x)$, it traverses the $x$ map in order, spinning on each entry until it is free (0) and subsequently acquiring it (by setting it to 1). Conversely, when $\tau$ calls $wunlock(x)$, it releases $x$ by traversing $x$ in order and setting each entry to one.

Recall that in order to promote a reader lock, the calling thread must already hold a reader lock on $x$. As such, the implementation of $plock(x)$ first check whether the calling thread $\tau$ currently holds a reader lock on $x$, i.e. $x[\tau] = 2$. If this is not the case then the implementation simply returns. To understand the remainder of the implementation, first consider the case where $plock(x)$ is called by $\tau \neq 0$, i.e. a thread other than the top-most thread. The implementation of $plock(x)$ then inspects the first entry in the map ($x[0]$), i.e. that of the top-most thread. If $x[0] = 1$, then $x$ is currently being acquired by another thread; the promotion must thus be retried. If on the other hand $x[0] \neq 1$ (i.e. $x[0] = 0$ or $x[0] = 2$), the implementation spins until it is zero and atomically updates it to one, signalling its intention to promote $x$. This pre-empts the promotion of $x$ by other threads: any such attempt would fail as now $x[0] = 1$. The implementation then sets its own entry ($x[\tau]$) to one, traverses the map in order, and spins on each entry until they too can be set to one. At this point the lock is successfully promoted and the implementation returns. Note that it is safe for $\tau$ to update its own entry $x[\tau]$ to one: at this point in execution no thread holds the writer lock on $x$, no thread can promote its lock on $x$, and those threads with a reader lock on $x$ never access the $x[\tau]$ entry – the read lock calls of another thread $\tau'$ solely accesses $x[\tau']$.

Let us now consider the case when the top-most thread with $\tau = 0$ calls can-promote $x$. Since prior to a $plock(x)$ call $\tau$ owns a reader lock on $x$, i.e. $x[\tau] = 2$, no other thread can promote its $x$ lock. As such, $\tau$ successfully sets $x[\tau]$ to one, signalling its intention to promote $x$. In other words, the promotion is skewed in favour of the top-most thread: if a thread races against the top-most thread to promote $x$, the top-most thread always wins. With the exception of the top-most thread, promotion is done on a ‘first-come-first-served’ basis. The rest of the implementation is then carried out as before: the map $x$ is traversed in turn and each entry is set to one.

**Implementation Correctness.** Let $I$ denote the weak MRSW lock implementation in Fig. 9. To show the soundness of $I$, we appeal to Thm. 1 and show that $I$ is locally sound on $L^{SRW}$.

Pick an arbitrary $\Lambda, \vec{f}, G = \langle E, \langle \vec{com}, \vec{so} \rangle, E'', \langle \vec{po''} \rangle \rangle$ such that $G$ is $\Lambda$-consistent and $\Lambda$-well-formed and $\text{abs}_{SRW, I}(\vec{f}, \langle E, \vec{po}, \langle E'', \vec{po''} \rangle \rangle)$. We must next find $\vec{com''}, \vec{so''}$ such that $(E'', \langle \vec{po''}, \vec{com''}, \vec{so''}, \vec{lab''} \rangle) \in L^{SRW}.G$, where $\vec{lab''}$ is the same as $\vec{lab'}$ in Def. 11.

For each location $x$, without loss of generality let us assume $G$ contains $n_x$ read lock calls on $x$, $n'_x$ read unlock calls on $x$, $m_x$ write lock calls on $x$, $m'_x$ write unlock calls on $x$ and $p_x$ lock promotion calls on $x$. Let us enumerate each of read lock calls, read unlock calls, write lock calls, write unlock calls and lock promotion calls arbitrarily. Note that:

- the MRSW constructor at location $x$ contains a single event $c_x$ where $\text{lab}(c_x) = \text{alloc}(x, 0)$.
- For each $i$th read lock operation on $x$, the $G$ contains the trace $\theta_i^{rl(x)} = t \xrightarrow{\text{po лим}} r^x \xrightarrow{\text{po лим}} rl$, where $r^x$ denotes the events of those iterations that failed to acquire the reader lock (failed CAS), $\text{lab}(t) = \text{get-tid}(\tau)$ for some $\tau$, $\text{lab}(rl) = \text{compare-set}(\text{acqrel}, x[\tau], 0, 2)$.
- for each $i$th read unlock operation on $x$, the $G$ contains the trace $\theta_i^{ru(x)} = t \xrightarrow{\text{po лим}} ru$, where $\text{lab}(t) = \text{get-tid}(\tau)$ for some $\tau$ and $\text{lab}(ru) = \text{store}(\text{rel}, x[\tau], 0)$.
Let us define: so" = com', with com" defined as follows:

\[
\text{com}" = \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{w(x)} \cdot w_l \land a = \theta_j^{w(x)} \cdot ru \right\}
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{w(x)} \cdot w_l \land a = \theta_j^{w(x)} \cdot w_u \right\}
\]

\[
\forall k, c. 0 \leq k \leq K \land (c, \theta_i^{w(x)} \cdot w_l) \in \text{com} \implies \neg \exists h. c = \theta_h^{w(x)} \cdot ru
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{w(x)} \cdot w_l \land a = c \land e \right\}
\]

\[
\forall k, c. 0 \leq k \leq K \land (c, \theta_i^{w(x)} \cdot w_l) \in \text{com} \implies \neg \exists h. c = \theta_h^{w(x)} \cdot ru
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{r(x)} \cdot r_l \land (a = c \land e) \right\}
\]

\[
\exists c, i, j, k, x, \tau. b = \theta_i^{r(x)} \cdot r_l \land c = \theta_j^{r(x)} \cdot ru
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land b = \theta_i^{r(x)} \cdot r_l \land c = \theta_j^{r(x)} \cdot ru \right\}
\]

\[
\land (a = c \land e) \land \exists d. a \rightarrow d \rightarrow c \implies \neg \exists h. (d = \theta_h^{w(x)} \cdot w_r \lor d = \theta_h^{w(x)} \cdot w_u) \lor (d = \theta_h^{b(x)} \cdot p_l \lor d = \theta_h^{b(x)} \cdot p_r)
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{b(x)} \cdot p_l \lor b = \theta_i^{b(x)} \cdot p_r \land a = \theta_j^{w(x)} \cdot ru \right\}
\]

\[
\bigcup \left\{ (f_a(a), f_b(b)) \mid (a, b) \in \text{com} \land \exists i, j, x, \tau. b = \theta_i^{r(x)} \cdot r_l \land (a = c \land e) \right\}
\]
We then need to show:

\[ \neg \Rightarrow \Rightarrow \text{ and } \Rightarrow \equiv \equiv \text{ to a contradiction.} \]

Similarly, in the latter case we then have

\[ \Rightarrow \equiv \equiv \text{ and let } G' = \langle E', \text{po'}, \text{com'}, \text{so'}, \text{lhbb'} \rangle. \]

We then need to show:

1. \( G' \) contains at most one constructor event;
2. \( \text{com}' = \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \cap \text{com}^\equiv \)

\[
\text{com}^\equiv \subseteq (C^x \cup U^x) \times WL^x \quad \text{com}^\equiv \subseteq (C^x \cup WU^x) \times RL^x \quad \text{com}^\equiv \subseteq (C^x \cup U^x) \times PL^x
\]

3. For all \( e, e_1, e_2 \):

\[
e_1 \neq e_2 \land (e, e_1, e_2) \in \text{com}' \Rightarrow (e_1, e_2) \in RL^x \lor (e_1 \in RL^x \land e_2 \in PL^x) \lor (e_1 \in PL^x \land e_2 \in RL^x)
\]

4. \( \text{com}^{-1} \) is functional;
5. \( E \cap L^x = \text{rng}(\text{com}') \); and
6. \( \text{so}' = \text{com}' \).

Parts (1), (5) and (6) follow immediately from the construction of \( G'_\equiv \). For part (2), pick an arbitrary \( (a, b) \in \text{com}' \). From the definition of \( \text{com}' \) we then know that there exists \( i, j \) such that either:

1. \( b = \text{wl}_j^x \land a = \text{ru}_j^x \lor a = \text{con}_x \); or
2. \( b = \text{rl}_j^x \land a = \text{ru}_j^x \lor a = \text{con}_x \); or
3. \( b = \text{pl}_j^x \land a = \text{ru}_j^x \lor a = \text{con}_x \).

In case (i), we then have \( (a, b) \in (C^x \cup U^x) \times WL^x \), as required. In case (ii) we have \( (a, b) \in (C^x \cup WU^x) \times RL^x \), as required. In case (iii), we have \( (a, b) \in (C^x \cup U^x) \times WL^x \), as required.

For part (3) we proceed by contradiction. Let us assume there exists \( e, e_1, e_2 \) such that \( e_1 \neq e_2 \), \( (e, e_1), (e, e_2) \in \text{com} \) and either: i) \( e_1, e_2 \in WL^x \); or ii) \( e_1 \in WL^x \) and \( e_2 \in RL^x \); or iii) \( e_1 \in WL^x \) and \( e_2 \in PL^x \); or iv) \( e_1, e_2 \in PL^x \).

In case (i), we know there exists \( a, b_1, b_2 \) such that \( e = f_a(a), e_1 = f_a(b_1), e_2 = f_a(b_2), (a, b_1), (a, b_2) \in \text{com} \) and \( b_1 \) and \( b_2 \) are both atomic CAS operations. As such, from the C11 consistency we know \( (a, b_1), (a, b_2) \in \text{mo}_{\text{imm}} \). Moreover, from C11 consistency we have either \( (b_1, b_2) \in \text{mo} \) or \( (b_2, b_1) \in \text{mo} \). In the former case we then have \( a \overset{\text{mo}}{\rightarrow} b_1 \overset{\text{mo}}{\rightarrow} b_2 \) and thus \( (a, b_2) \notin \text{mo}_{\text{imm}} \), leading to a contradiction. Similarly, in the latter case we then have \( a \overset{\text{mo}}{\rightarrow} b_2 \overset{\text{mo}}{\rightarrow} b_1 \) and thus \( (a, b_1) \notin \text{mo}_{\text{imm}} \), leading to a contradiction.

The proof of cases (ii) and (iii) are analogous to that of (i) and are omitted here.

In case (ii) we then know there exists \( a, b_1, b_2, c \) such that \( e = f_a(a), e_1 = f_a(b_1), e_2 = f_a(b_2), (a, b_1), (a, b_2) \in \text{com} \) and \( b_1 \) and \( b_2 \) are both atomic CAS operations, \( (a, b_1) \in \text{com} \), and either a) \( (a, b_2) \in \text{com} \) or b) \( (a, c) \in \text{mo} \), \( (c, b_2) \in \text{com} \) and \( (a, b_1) \in \text{mo}_{\text{imm}} \). The proof of case (a) is analogous to the proof of case (1) above. For part (b), as \( b_2 \) is an atomic CAS from the C11 consistency we have \( (c, b) \in \text{mo}_{\text{imm}} \). Moreover, from C11 consistency we have either \( (b_1, c) \in \text{mo} \) or \( (c, b_1) \in \text{mo} \). However, as we have \( (a, b_1) \in \text{mo}_{\text{imm}} \), we then have \( (c, b_1) \in \text{mo} \). As such, we have \( a \overset{\text{mo}}{\rightarrow} c \overset{\text{mo}}{\rightarrow} b_1 \) and thus \( (a, b_1) \notin \text{mo}_{\text{imm}} \), leading to a contradiction.

Part (4) follows from the definition of \( \text{com}' \) and the functionality of \( \text{com}^{-1} \) for C11 registers. □
new-queue() ≜
  let l = new-mutex() in
  let q = alloc(∞) in
  store(q, l, rlx); store(q+1, 2, rlx); q

deq(q) ≜
  let l = load(q, rlx) in
  lock(l); let range = load(q+1, rlx) in
  for i = 1 to range do
    let x = load(q + i, rlx) in
    if x ≠ 0 then
      unlock(l); break; x
  unlock(l); ⊥

Fig. 10. The locking queue implementation

F A SOUND STRONG QUEUE IMPLEMENTATION

In Fig. 10 we present a simple implementation of a strong queue using release-acquire registers and the mutex library. As we formalise in Thm. 9, this implementation is sound with respect to the strong queue library $L^S$.

Theorem 9. The queue implementation in Fig. 7 is a sound implementation of the strong queue library $L^S$.

Proof. The full proof is mechanised in the Coq proof assistant and is available as auxiliary material.

G THE SOUNDNESS OF EXCHANGER IMPLEMENTATION

Let $I$ denote the exchanger implementation in Fig. 5. To show the soundness of $I$, we appeal to Thm. 1 and show that $I$ is locally sound on $I^X$.

Pick an arbitrary $\Lambda, f, G=(E, po, com, so), E'$, $po'$ such that $G$ is $\Lambda$-consistent and $\Lambda$-well-formed and $\text{abs}_{I, f}(f, (E, po), (E', po'))$. We must next find $\text{com}'$, $so'$ such that $(E', po', \text{com}', so', \text{lhb}') \in L^X.\mathcal{G}_c$, where $\text{lhb}'$ is as defined in Def. 11.

Note that each exchange operation $\text{exchange}(g, v)$ either:

1. offers a value at index $g+k$ (for some $k \in \mathbb{N}^+$ where $k$ is an odd number) and returns unmatched due to a timeout; or
2. offers a value at index $g+k$ (for some $k \in \mathbb{N}^+$ where $k$ is an odd number) and matches with value $v'$ at index $g+k+1$; or
3. tries to offer a value at index $g+k$ (for some $k \in \mathbb{N}^+$ where $i$ is an even number) and returns unmatched as the slot at index $g+k$ is already taken; or
4. offers a value at index $g+k$ (for some $k \in \mathbb{N}^+$ where $k$ is an even number) and matches with value $v'$ at index $g+k-1$.

Without loss of generality, let us assume $e$ contains $n$ exchange calls. For each $i$th exchange operation of the form $\text{exchange}(g, v_i)$, the $G_i$ contains a trace of one of the following forms depending which of the four categories above it falls into:

- $\theta_{\text{po}, 0}(g)$ is of the form $s \xrightarrow{po_{\text{num}} \cdot 0} t$, with $\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(o) = \text{CAS}(rlx, g+j_i, 0, v_i)$, and $\text{lab}(t) = \text{CAS}(rlx, g+j_i+1, 0, \perp)$, for some $j_i$ and $v_i$ where $j_i$ is odd;
- $\theta_{\text{po}, 0}(g)$ is of the form $s \xrightarrow{po_{\text{num}} \cdot 0} a \xrightarrow{po_{\text{num}} \cdot r}$, with $\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(o) = \text{CAS}(rlx, g+j_i, 0, v_i), \text{lab}(a) = \text{load}(rlx, g+j_i+1, v'_i)$ with $v'_i ≠ 0$, $\text{lab}(r) = \text{load}(acq, g+j_i+1, v'_i)$, for some $j_i$, $v_i$ and $v'_i$ where $j_i$ is odd;
• \( \theta_i^r(g) \) is of the form \( s \xrightarrow{p_0|\text{imm}} f \xrightarrow{p_0|\text{imm}} o \xrightarrow{p_0|\text{imm}} c \), with \( \text{lab}(s) = \text{load}(rlx, g, j_i) \), \( \text{lab}(f) = \text{load}(rlx, g+j_i, w_i) \) with \( w_i \neq 0 \), \( \text{lab}(o) = \text{load}(rlx, g+j_i+1, w'_i) \) with \( w'_i \neq 0 \), and \( \text{lab}(c) = \text{CAS}(rlx, g, j_i, j_i+2) \) or \( \text{lab}(c) = \text{load}(rlx, g, u_i) \) with \( u_i \neq j_i \), for some \( j_i \) and \( v_i \) where \( j_i \) is odd;

• \( \theta_i^e(g) \) is of the form \( s \xrightarrow{p_0|\text{imm}} f \xrightarrow{p_0|\text{imm}} o \xrightarrow{p_0|\text{imm}} c \xrightarrow{p_0|\text{imm}} r \), with \( \text{lab}(s) = \text{load}(rlx, g, j_i) \), \( \text{lab}(f) = \text{load}(rlx, g+j_i, w_i) \) with \( w_i \neq 0 \), \( \text{lab}(o) = \text{CAS}(rel, g+j_i+1, 0, v_i) \), \( \text{lab}(c) = \text{CAS}(rlx, g, j_i, j_i+2) \) or \( \text{lab}(c) = \text{load}(rlx, g, u_i) \) with \( u_i \neq j_i \), \( \text{lab}(r) = \text{load}(acq, g+j_i, v'_i) \), for some \( j_i, v_i \) and \( v'_i \) where \( j_i \) is odd.

Let \( s' = \text{com}' \) with

\[
\text{com}' \triangleq \left\{ (a, b), (b, a) \left| \begin{array}{l}
\exists i, j, g, f(e_i^{p_0}(g), a) = a \land f(e_j^{p_0}(g), b) = b \\
\land (e_i^{p_0}(g), a, e_j^{p_0}(g), b) \in \text{com} \land (e_j^{p_0}(g), a, e_i^{p_0}(g), b) \in \text{com}
\end{array} \right. \right\}
\]

To show that \( G' = (E', \text{po}', \text{com}', s', \text{lhbc}) \in L^X G_c \), we are then required to show for all \( g \in \text{Loc} \), 
\( G'_g \) is exchanger-consistent on \( g \). Pick an arbitrary \( g \in \text{Loc} \), we then need to show:

1) \( E'^c = \emptyset \lor \exists c \in C^g. E'^c = \{ c \}; \)

2) \( \text{com}' \) is symmetric, irreflexive and \( \text{com}' \subseteq \bigcup_{\nu_1, \nu_2 \in \text{Val}} X^{\nu_1, \nu_2} \times X^{\nu_1, \nu_2} \setminus \text{id}; \)

3) \( \text{com}' \) is functional;

4) \( E' \cap X^g \setminus \text{dom}(\text{com}') \subseteq \bigcup_{\nu \in \text{Val}} X^{\nu, \nu^\perp}; \) and

5) \( s' = \text{com}' \).

Parts (1), (2), and (5) follow immediately from the construction of \( G_c \) and the consistency of the C library. The proof of parts (3) and (4) follows from the definition of \( \text{com}' \), the consistency of \( G_i \) and the definition of the C library.
H CORRECTNESS OF THE HERLIHY-WING QUEUE IMPLEMENTATIONS

H.1 Soundness of the Strong Herlihy-Wing Queue Implementation

Let I denote the strong Herlihy-Wing implementation in Fig. 2. To show the soundness of I, we appeal to Thm. 1 and show that I is locally sound on $L^{SO}$. Let $\Lambda, f, G = (E, po, com, so), E'', po''$ such that $G$ is $\Lambda$-consistent and $\Lambda$-well-formed and $\text{abs}_{\Lambda}(q, E, (E, po), (E'', po''))$. We must next find $\text{com}''$, $\text{so}''$ such that $(E'', po'', \text{com}''$, $\text{so}$$''$, $\text{lbh}''') \in L^{SO}.G_c$, where $\text{lbh}''$ is the same as $\text{lbh}$ in Def. 11.

For each queue at $q$, let us enumerate the enqueue operations on $q$ by their insertion index. For instance, the very first enqueue operation is that which inserts the new value at index $q+1$. That is, the $i$th enqueue operation is that which inserts its value at index $q+i$. Similarily, let us enumerate the dequeue operations by their removal index. That is, the $j$th dequeue operation is that which removes the element at index $q+j$. When program $e$ contains $n_q$ enqueue operations on $q$ and $m_q$ dequeue operations on $q$ then:

- the constructor of the queue at location $q$ contains a trace with a single event $c_q$ where $
abla (c_q) = \text{alloc}(q, 0)$;
- for each $i$th enqueue operation $enq(q, v_i)$ with $v_i \neq \bot$, $G$ contains a trace of the form $\theta_i^{enq} = e_i ^{enq} \rightarrow f_i ^{enq}$, where $\nabla (e_i) = \text{FAA}(\text{rel1}, q, i-1, 1)$ and $\nabla (e_i) = \text{store}(\text{rel1}, q+i, v_i)$;
- for each $j$th dequeue operation, $G$ contains a trace of the form $\theta_j^d = \theta_j^f \rightarrow d_j ^{po} \rightarrow f_j ^{po} \rightarrow \cdots \rightarrow f_j ^{enq}$, such that all events of $\theta_j^f$ are read events: $\forall a \in \theta_j^f. \nabla (a) = \text{load}(-, -, -)$; $\nabla (d_j) = \text{load}(\text{acq}, q, \text{len}_j)$; $\nabla (\text{lab}_k) = \text{load}(\text{acqr}l, q+k, 0)$ for all $k \in \{1 \cdots j-1\}$; and $\nabla (\text{lab}_j) = \text{AX}(\text{acqr}l, q+j, \text{w}_j, 0)$, for some $w_j \neq \bot$, range $\text{len}_j \in \mathbb{N}^+$ and $1 \leq j \leq \text{len}_j$.

Let us define: $\text{imp}(.): E'' \rightarrow E$ as follows:

$$\begin{align*}
\text{imp}(e) & \triangleq \begin{cases}
\text{c}_q & \exists q. f(c_q) = e \\
\theta_i^{enq} \cdot e_i^2 & \exists i. q. f(\theta_i^{enq} \cdot e_i^2) = e \\
\theta_i^{d(q)} \cdot d_i^2 & \exists i. q. f(\theta_i^{d(q)} \cdot d_i^2) = e
\end{cases}
\end{align*}$$

Let $\text{so}'' = \text{com}''$ with

$$\text{com}'' \triangleq \{(e, d) \mid \exists i, q. f(\theta_i^{enq} \cdot e_i^2) = e \land f(\theta_i^{d(q)} \cdot d_i^2) = d \land (\theta_i^{enq} \cdot e_i^2, \theta_i^{d(q)} \cdot d_i^2) \in \text{com}\}$$

Let $\text{lbh} = G.\text{lbh} = (po \cup so)'$. Note that from the definition of $\text{lbh}'$, $\text{lbh}$ and the construction of $so'$ above we know the (6) below holds. As such, from the irreflexivity of $\text{lbh}$, we also know (7) holds.

$$\begin{align*}
\forall a, b. (a, b) \in \text{lbh}' & \iff a \neq b \land (\text{imp}(a), \text{imp}(b)) \in \text{lbh} \tag{6} \\
\forall a. (a, a) & \notin \text{lbh}'' \tag{7}
\end{align*}$$

Moreover, it is straightforward to demonstrate that given the C11 memory model and the values written, the (8) property below holds. Consequently, since the “range” value read by each dequeue operation is greater than or equal the slot acquired by its matching enqueuing thread, the $\text{relq}$ mode of fetch-and-add operations in enqueue and the $\text{acq}$ mode of “range” reads in dequeue operations, thanks to the release sequences of C11 the (9) property below holds.

$$\begin{align*}
\forall a, b \in \{1 \cdots n\}. a < b & \Rightarrow (e_a^1, e_b^1) \in \text{mo} \tag{8} \\
\forall \pi. (e_\pi^2, d_\pi^2) \in \text{com} & \Rightarrow (e_\pi^1, d_\pi^1) \in \text{so} \tag{9}
\end{align*}$$

Given the encapsulation of $G$ and the definition of $I$, we know that for all $\pi \in \mathbb{N}$ and all queue locations $q, G.\pi \cap \{e \mid \text{loc}(e) = q+\pi\} = W_\pi$, where $W_\pi \triangleq \pi \in G.\pi \cap \{\pi \cdot \pi \cup W_\pi^f \cup W_\pi^d\}$, with $W_\pi^f \triangleq \{f_\pi^e \mid 1 \leq k \leq m\}$ and $W_\pi^d \triangleq \{d_\pi^e \}$. 

Given the definition of the C11 library, we then know that for each $\pi \in \mathbb{N}^+$, the $W_\pi$ is totally ordered by a strict total order $\text{mo}$. Consequently, given the release-acquire (acqrel) mode of update events in $W^{f}_\pi \cup W^{d}_\pi$, it is straightforward to demonstrate that:

$$\forall \pi \in \mathbb{N}. \forall w_1, w_2 \in W^{f}_\pi \cup W^{d}_\pi. 
((w_1, w_2) \in (\text{so} \cap \text{mo}) \cup (\text{so}^{-1} \cap \text{mo}^{-1})) \lor (w_1, e^2_\pi), (e^2_\pi, w_2) \in \text{mo} \lor (w_2, e^2_\pi), (e^2_\pi, w_1) \in \text{mo}$$

(10)

We next demonstrate that:

$$\forall a, b. a < b \Rightarrow (e^2_b, e^2_a) \notin \text{hb}$$

(11)

$$\forall \pi, \pi' \in \mathbb{N}^+. d^2_\pi \xrightarrow{\text{com}} e^2_\pi \xrightarrow{\text{hb}} e^2_{\pi'} \xrightarrow{\text{com}} d^2_{\pi'}. \Rightarrow d^2_\pi \xrightarrow{\text{hb}} d^2_{\pi'}$$

(12)

For part (11) we proceed by contradiction. Let us assume there exists $a, b$ such that $a < b$ and $(e^2_b, e^2_a) \in \text{hb}$. Since $G$ is consistent and $e^2_b, e^2_a$ are write events, we know that they do not have any incoming $\text{so}$ edges. As such, as $\text{hb} = (\text{po} \cup \text{so})^+$, we know that there exists $e$ such that $e^2_b \xrightarrow{\text{hb}} e \xrightarrow{\text{po}} e^2_a$.

Moreover, since $e^2_a \xrightarrow{\text{po}\text{imm}} e^2_a$, we also know that $e \xrightarrow{\text{po}} e^1_a$. That is, we have $(e^2_b, e^1_a) \in \text{hb}$. On the other hand, from (8) we know that $(e^1_b, e^1_a) \in \text{mo}$. We then have: $e^1_b \xrightarrow{\text{mo}} e^1_a \xrightarrow{\text{po}} e^2_b \xrightarrow{\text{hb}} e^1_a$. That is, we have $e^1_a \xrightarrow{\text{mo}} e^1_b \xrightarrow{\text{hb}} e^1_a$, contradicting the assumption that $G$ is consistent.

For part (12), as $e^2_\pi \xrightarrow{\text{hb}} e^2_{\pi'}$, from (11) we know that $\pi < \pi'$. Moreover, as $(e^2_\pi, d^2_{\pi'}) \in \text{com}$, from the consistency of the C library we know that $(e^2_\pi, d^2_{\pi'}) \in \text{mo}_{\text{limm}}$. As such, from (10) we know that either i) $(d^2_\pi, f^2_{\pi'}) \in \text{so}$; or ii) $(f^2_{\pi'}, e^2_\pi) \in \text{mo}$.

In case (i) we have $d^2_\pi \xrightarrow{\text{so}} f^2_{\pi'} \xrightarrow{\text{po}} d^2_{\pi'}, i.e. d^2_\pi \xrightarrow{\text{hb}} d^2_{\pi'}$, as required.

In case (ii), since $G$ is consistent and $e^2_{\pi'}, e^2_\pi$ are write events, we know that they do not have any incoming $\text{so}$ edges. As such, as $\text{hb} = (\text{po} \cup \text{so})^+$, we know that there exists $e$ such that $e^2_{\pi'} \xrightarrow{\text{hb}} e \xrightarrow{\text{po}} e^2_\pi$. Moreover, since $e^1_{\pi'}, \xrightarrow{\text{po}\text{limm}} e^1_{\pi'}$, we also know that $e \xrightarrow{\text{po}} e^1_{\pi'}$. That is, we have $(e^2_{\pi'}, e^1_{\pi'}) \in \text{hb}$. Moreover, from (9) we have $(e^1_{\pi'}, d^1_{\pi'}) \in \text{so} \subseteq \text{hb}$. As such, from the assumption of the case we have $e^2_{\pi'} \xrightarrow{\text{hb}} e^1_{\pi'} \xrightarrow{\text{hb}} d^1_{\pi'} \xrightarrow{\text{po}} f^1_{\pi'} \xrightarrow{\text{mo}} e^2_{\pi'}$. That is, we have $e^2_{\pi'} \xrightarrow{\text{hb}} f^1_{\pi'} \xrightarrow{\text{mo}} e^2_{\pi'}$, contradicting the assumption that $G$ is consistent.

To show that $G' = (E'', \text{po''}, \text{com''}, \text{so''}, \text{hb''}) \in L^{\text{seq}}.G_c$, we are then required to show that for all locations $q, G'_q$ is queue consistent on $q$. Pick an arbitrary location $q$ and let $G'_q = (E', \text{po'}, \text{com'}, \text{so'}, \text{hb'})$.

We then need to show:

1) $E'^c = \emptyset \lor \exists c \in C^q. E'^c = \{c\}$;
2) $\text{com'} \subseteq \bigcup_{u \in \text{Val}(\perp)} E^{q, u} \times D^{q, u}$
3) $\text{com'}$ \ (\text{com'})$^{-1}$ are functional
4) $E' \cap D^{q} \setminus \text{rng} \text{(com')}$ \ $D^{q, \perp}$
5) $\{E^{q} \setminus \text{dom} \text{(com')}; \text{hb'}; [D^{q, \perp}] = \emptyset$
6) $\text{so'} = \text{com'}$
7) there exists a sequential enumeration $S$ of the events in $E'$ such that: (i) $S$ respects $\text{hb'}$; and (ii) $\text{fifo}(e, S)$ holds.

Parts (1), (2) and (6) follow simply from the construction of $G'$. Part (5) holds trivially as $D^{q, \perp} = \emptyset$.

For part (4), we demonstrate that $E' \cap D^{q} \setminus \text{rng} \text{(com')} = \emptyset$. We proceed by contradiction. Let us assume there exists $d \in E' \cap D^{q}$ such that $d \notin \text{rng} \text{(com')}$. From the construction of $G'$ we then know...
there exists $\theta^{d(q)}_i$ and $d'_i$ such that $d'_i \in \theta^{d(q)}_i$ and $\lambda(d'_i) = \lambda(\text{acq}, q+s_j, w_j, 0)$ for some $s_j, w_j$ such that $s_j > 0$ and $w_j \neq \perp$. From the definition of consistency for the C library we know there exists a write event $a$ such that $(a, d'_i) \in \text{com}$. Given the assumption of the case (A), and the shape of the $\theta^e$ and $\theta^d$ traces, we then know that there exists $\theta^{e(q)}_i$ and $e'_i$ such that $e'_i \in \theta^{e(q)}_i$, $a = e'_i$; and $\lambda(e'_i) = \text{store}(\text{rel}, q+s_j, w_j)$. As such, from the construction of $G'$ we know there exists $e \in \mathcal{E}(q,w_j)$ such that $(e, d) \in \text{com}'$. This however contradicts our assumption that $d \notin \text{rng}(\text{com}')$

For part (3) we proceed by contradiction. Let us assume that $(\text{com}')^{-1}$ is not functional and there exist $e'_1, e'_2, e'_3 \in \mathcal{E}^q$ and $d'_1, d'_2 \in \mathcal{D}^q$ such that $e'_1 \neq e'_2$ and $(e'_1, d'_1), (e'_2, d'_2) \in \text{com}'$. From the construction of $G'$ we then know there exist $\theta^{e(q)}_1, \theta^{e(q)}_2, \theta^{d(q)}_1$ such that $\theta^{e(q)}_1 \neq \theta^{e(q)}_2$ and $e'_1, e'_2 \in \text{com}$. That is, $(e'_1, d'_1), (e'_2, d'_2) \in \text{com}^{-1}_C$. This however contradicts the assumption that $G$ is consistent with respect to the C library, i.e. the assumption $\text{com}^{-1}_C$ is functional.

Let us next assume that $\text{com}'$ is not functional and there exist $e' \in \mathcal{E}^q$ and $d'_1, d'_2 \in \mathcal{D}^q$ such that $d'_1 \neq d'_2$ and $(e', d'_1), (e', d'_2) \in \text{com}' \cup \text{io}$. From the construction of $G'$ we then know there exist $\theta^{d(q)}_1, \theta^{d(q)}_2, \theta^{e(q)}_1$ such that $\theta^{d(q)}_1 \neq \theta^{d(q)}_2$ and $(e'_1, d'_1), (e'_2, d'_2) \in \text{com}$. That is $(e'_1, d'_1), (e'_2, d'_2) \in \text{com}^{-1}_C$. As $d'_1, d'_2$ are atomic update events, from the definition of the C library and the consistency of $G$ we know they are ordered by a total modification order $\text{mo}$. Without loss of generality, let us assume that $(d'_1, d'_2) \in \text{mo}$. From the definition of the C library and the consistency of $G$, we then also have $(e'_1, d'_1), (e'_2, d'_2) \in \text{com}^{-1} \cup \text{mo}$. Consequently, we have $d'_1 \xrightarrow{\text{com}^{-1}_C} d'_2 \xrightarrow{\text{mo}} d'_1$, contradicting the assumption that $G$ is consistent.

For part (7), in what follows we demonstrate that for all $n \in \mathbb{N}^+$, the irreflexive($C^{n,n}$) holds for $G'$. The desired property of (7) then follows immediately from Thm. 7.

To demonstrate that $\forall n \in \mathbb{N}^+$. irreflexive($C^{n,n}$) holds, we proceed by induction on $n$.

**Base case $n = 1$**

We proceed by contradiction. Let us assume there exist $d_1, d_2, e_1, e_2$ such that $d_1 \rightarrow e_1 \rightarrow e_2 \rightarrow d_2 \rightarrow d_1$. From the definition of $\text{com}'$, the definition of $\text{imp}(.)$ and (6) we then have $\text{imp}(d_1) \rightarrow \text{com}^{-1} \rightarrow \text{imp}(e_1) \rightarrow \text{imp}(e_2) \rightarrow \text{imp}(d_2) \rightarrow \text{imp}(d_1)$. As such, from the transitivity of $\text{hb}$ we have $\text{imp}(d_1) \rightarrow \text{imp}(d_1)$, contradicting the assumption that $G$ is consistent.

**Inductive case $n = m+1$**

$$\forall k \in \mathbb{N}^+. k \leq m \Rightarrow \text{irreflexive}(C^{k,k}) \quad \text{(I.H.)}$$

We proceed by contradiction. Let us assume that there exist a $C^{n,n}$ cycle. As $n > 0$, we know there is at least one adjacent pair of $\text{com}'^{-1}$; $\text{hb}'$ and $\text{com}'$; $\text{hb}'$ edges. That is, there exist $a, b$ such that and $a \xrightarrow{\text{com}^{-1}_C} b \xrightarrow{\text{com}} a$. As such, from (6), (12), the transitivity of $\text{hb}$ and the functionality of $\text{imp}(.)$ we have $\text{imp}(a) \rightarrow \text{imp}(b)$ and $a \neq b$. Consequently, from (6) we have $(a, b) \in \text{hb}'$. Moreover, since by definition the $C^{m,m}$ edge ends with $\text{hb}'$ and $a \rightarrow b$, from the transitivity of $\text{hb}'$ and since $b \rightarrow a$ we have $b \xrightarrow{\text{com}^{-1}_C} b$. This however contradicts our assumption in (I.H.).
H.2 Soundness of the Weak Herlihy-Wing Queue Implementation

Let \( I \) denote the weak Herlihy-Wing implementation in Fig. 2. To show the soundness of \( I \), we appeal to Thm. 1 and show that \( I \) is locally sound on \( L^G \).

Pick an arbitrary \( \Lambda, f, G = \langle E, p, \text{com}, so \rangle, E'' \) such that \( G \) is \( \Lambda \)-consistent and \( \Lambda \)-well-formed and \( \text{abs}_{\text{com}}(f, \langle E, p, \text{com} \rangle, (E'', \text{po}'')) \).

We must next find \( \text{com}'' \), \( \text{so}'' \) such that \( \langle E'', \text{po}'', \text{com}'', \text{so}'' \rangle, \text{lhb}'' \rangle \in L^{G_0} G_c \), where \( \text{lhb}'' \) is the same as \( \text{lhb} \) in Def. 11.

In the remainder of this proof, we assume all identically-named definitions (e.g. \( G', \text{com}' \), \( \text{so}' \) and \( \text{imp}() \) and so forth) are as defined for the strong Herlihy-Wing queue implementation unless otherwise stated.

Note that due to the encapsulation of \( G \) and the definition of \( I \), we know that for all \( \pi \in \mathbb{N} \):

\[
G.E \cap \{ e \mid \text{loc}(e) = q + \pi \} = W_\pi
\]

As such, given the consistency of the C library \( L^C \), we then know that for each \( \pi \in \mathbb{N}^+ \), the \( W_\pi \) is totally ordered by a strict total order \( \text{mo} \).

Observe that the \((6), (7) (8), (9) \) and \((11) \) properties also hold for the weak implementation. We next demonstrate that:

\[
\forall \pi, \pi' \in \mathbb{N}^+. \; d^2_\pi \xrightarrow{\text{com}^{-1}} e^2_\pi \xrightarrow{\text{hb}} e^2_\pi \xrightarrow{\text{com}} d^2_\pi \Rightarrow (d^2_\pi, d^2_\pi) \notin \text{hb} \tag{13}
\]

We proceed by contradiction. Let us assume there exist \( \pi, \pi' \) such that \( (e^2_\pi, d^2_\pi) \in \text{com}, (e^2_\pi, e^2_\pi) \in \text{hb}, (e^2_\pi, d^2_\pi) \in \text{com} \) and \( (d^2_\pi, d^2_\pi) \in \text{hb} \). As \( e^2_\pi \xrightarrow{\text{hb}} e^2_\pi \), from (11) we know that \( \pi < \pi' \). Moreover, as \( (e^2_\pi, d^2_\pi) \in \text{com} \), from the consistency of the C library we know that \( (e^2_\pi, d^2_\pi) \in \text{mo}_{\text{imm}} \). As such, since the writes in \( W_\pi \) are totally ordered by \( \text{mo} \) (see above), we know that either i) \( (d^2_\pi, f^2_\pi) \in \text{mo} \); or ii) \( (f^2_\pi, e^2_\pi) \in \text{mo} \).

In case (i) we then have \( d^2_\pi \xrightarrow{\text{mo}} f^2_\pi \xrightarrow{\text{po}} d^2_\pi \xrightarrow{\text{hb}} d^2_\pi \). That is, we have \( d^2_\pi \xrightarrow{\text{mo}} f^2_\pi \xrightarrow{\text{hb}} d^2_\pi \), contradicting the assumption that \( G \) is consistent.

In case (ii), since \( G \) is consistent and \( e^2_\pi, e^2_\pi \) are write events, we know that they do not have any incoming \( \text{so} \) edges. As such, as \( \text{hb} = (\text{po} \cup \text{so})^* \), we know that there exists \( e \) such that \( e^2_\pi \xrightarrow{\text{hb}} e \xrightarrow{\text{po}} e^2_\pi \). Moreover, since \( e^1_\pi \xrightarrow{\text{po}_{\text{imm}}} e^2_\pi \), we also know that \( e \xrightarrow{\text{po}} e^1_\pi \). That is, we have \( (e^2_\pi, e^1_\pi) \in \text{hb} \). Moreover, from (9) we have \( (e^1_\pi, d^1_\pi) \in \text{so} \subseteq \text{hb} \). As such, from the assumption of the case we have \( e^2_\pi \xrightarrow{\text{hb}} e^1_\pi \xrightarrow{\text{hb}} d^1_\pi \xrightarrow{\text{po}} f^1_\pi \xrightarrow{\text{mo}} e^2_\pi \). That is, we have \( e^2_\pi \xrightarrow{\text{hb}} e^1_\pi \xrightarrow{\text{mo}} e^2_\pi \), contradicting the assumption that \( G \) is consistent.

To show that \( G' = (E'', \text{po}'', \text{com}'', \text{so}'', \text{lhb}'') \in L^{G_0} G_c \), we are then required to show that for all locations \( q \), \( G'_q \) is queue consistent on \( q \). Pick an arbitrary location \( q \) and let \( G'_q = (E', q, \text{po}', \text{com}', \text{so}', \text{lhb}') \).

We then need to show:
1) \( \text{com}' \subseteq \bigcup_{e \in \text{Val}(\lambda, s)} E^q, s \times D^{q, s} \)
2) \( \text{com}', (\text{com}')^{-1} \) are functional
3) \( E' \cap D^q \setminus \text{rng}(\text{com}') \subseteq D^q, -1 \)
4) \( E^q \setminus \text{dom}(\text{com}'); \text{lhb}''; [D^q, -1] = 0 \)
5) \( \text{so}' = \text{com}' \)
6) \( \text{com}^{-1}; \text{lhb}; \text{com}; \text{lhb} \) is irreflexive.
Proof of parts (1-5) are as in the case of the strong implementation. For part (6) we proceed by contradiction. Let us assume there exist \(d_1, d_2, e_1, e_2\) such that \(d_1 \xrightarrow{\text{com}'} e_1 \xrightarrow{\text{hb}} e_2 \xrightarrow{\text{com}'} d_2 \xrightarrow{\text{hb}} d_1\).

From the definition of \(\text{com}'\), the definition of \(\text{imp}(.)\) and (6) we then have \(\text{imp}(d_1) \xrightarrow{\text{com}^{-1}} \text{imp}(e_1) \xrightarrow{\text{hb}} \text{imp}(e_2) \xrightarrow{\text{hb}} \text{imp}(d_2) \xrightarrow{\text{hb}} \text{imp}(d_1)\). From (13) we then have \(\text{imp}(d_1) \xrightarrow{\text{hb}} \text{imp}(d_2) \xrightarrow{\text{hb}} \text{imp}(d_1)\). As such, from the transitivity of \(\text{hb}\) we have \(\text{imp}(d_1) \xrightarrow{\text{hb}} \text{imp}(d_1)\), contradicting the assumption that \(G\) is consistent.
I THE SOUNDNESS OF THE WEAK STACK IMPLEMENTATION

Let $I$ denote the weak stack implementation in Fig. 6. To show the soundness of $I$, we appeal to Thm. 1 and show that $I$ is locally sound on $L_{\text{WS}}$.

Pick an arbitrary $\Lambda, f, G=\langle E, \text{po}, \text{com}, \text{so} \rangle, E'$, $\text{po}'$ such that $G$ is $\Lambda$-consistent and $\Lambda$-well-formed and $\text{abs}_{\text{WS,1}}(f, \langle E, \text{po} \rangle, \langle E', \text{po}' \rangle)$.

We must next find $\text{com}''$, $\text{so}''$ such that $\langle E', \text{po}'', \text{com}''', \text{so}''', \text{lab}'' \rangle \in L_{\text{WS}} \cdot G_c$, where $\text{lab}''$ is the same as $\text{lab}^c'$ in Def. 11.

Let us assume without loss of generality that for each location $s$, the $G$ contains $n_s$ push operations and $m_s$ pop operations. We will shortly enumerate these operations in order of their lock acquisition.

Note that for each $i \in \{1 \cdots n_s\}$, the $i^{\text{th}}$ push operation try-push($s, v_i$) either:

- pushes $v_i$ on the stack at $s$ and thus $G$ contains the events in the trace: $\theta^a(s) = \{ r_i \xrightarrow{\text{po}} \text{c} \}
\text{lab}(l) = \text{CAS}(\text{acqrel}, s, 0, 1), \text{lab}(r_i) = \text{load}(rlx, s+1, t)$
for some top value $t, \text{lab}(a) = \text{store}(rlx, s+t+1, v_i), \text{lab}(w_i) = \text{store}(rlx, s+1, t+1)$, and
$\text{lab}(u) = \text{store}(rel, s, 0)$; or
- fails to push $v_i$ on the stack as it fails to acquire the lock at $s$, and thus $G$ contains the single-event trace: $\theta^a(s) = \{ f \}$, where $\text{lab}(f) = \text{load}(\text{acq}, s, 1)$.

Similarly, for each $i \in \{1 \cdots m_s\}$, the $i^{\text{th}}$ pop operation try-poop($s$) either:

- pops $w_i$ from the stack at $s$ and thus $G$ contains the events in the trace: $\theta^a(s) = \{ r_i \xrightarrow{\text{po}} \text{c} \}
\text{lab}(l) = \text{CAS}(\text{acqrel}, s, 0, 1), \text{lab}(r_i) = \text{load}(rlx, s+1, t)$
for some top value $t, \text{lab}(r) = \text{load}(rlx, s+t, v_i), \text{lab}(p) = \text{store}(rlx, s+t, 0), \text{lab}(w_i) = \text{store}(rlx, s+1, t-1)$, and
$\text{lab}(u) = \text{store}(rel, s, 0)$; or
- fails to pop from the stack as it fails to acquire the lock at $s$ and thus $G$ contains the single-event trace: $\theta^{\text{rel}}(s) = \{ f \}$, where $\text{lab}(f) = \text{load}(\text{acq}, s, 1)$.
- fails to pop from the stack array as it is empty, and thus $G$ contains the events in the trace: $\theta^{\text{rel}}(s) = \{ \text{lab}(l) = \text{CAS}(\text{acqrel}, s, 0, 1), \text{lab}(f) = \text{load}(rlx, s+1, 1)$, and
$\text{lab}(u) = \text{store}(rel, s, 0)$.

Moreover, the constructor of the stack at location $s$ contains a trace of the form $\theta^c(s) = \{ c \}
\text{lab}(c_s) = \text{alloc}(s, 0), \text{lab}(c_i) = \text{store}(rel, s, 0), \text{lab}(c_i) = \text{store}(rel, s+1, 1)$.

Let us define $\text{imp}(.) : E'' \rightarrow E$ as:

\[
\text{imp}(e) \triangleq \begin{cases} 
\theta^c(s) \cdot c_i & \exists s. \theta^c(s) \cdot c_i = e \\
\theta^a(s) \cdot a & \exists i, s. f(\theta^a(s) \cdot a) = e \\
\theta^a(s) \cdot f & \exists i, s. f(\theta^a(s) \cdot f) = e \\
\theta^a(s) \cdot r & \exists i, s. f(\theta^a(s) \cdot r) = e \\
\theta^{\text{rel}}(s) \cdot f & \exists i, s. f(\theta^{\text{rel}}(s) \cdot f) = e \\
\theta^{\text{rel}}(s) \cdot r & \exists i, s. f(\theta^{\text{rel}}(s) \cdot r) = e \\
\end{cases}
\]

Let $\text{so}'' = \text{com}'''$ with

\[
\text{com}''' \triangleq \{ (a, r) \mid \exists i, j, s. f(\theta^a(s) \cdot a) = a f(\theta^{\text{rel}}(s) \cdot r) = r \wedge (\theta^a(s) \cdot a, \theta^{\text{rel}}(s) \cdot r) \in \text{com} \}
\]
It is straightforward to demonstrate that given the acquire mode of the lock acquisitions, the release mode of lock releases, and the specification of the C library we have:

\[
\forall s. \forall i, j \in \{1 \ldots n+m\}. \ ((\theta_i^{as}(s).u, \theta_j^{rs}(s).l) \in \text{hb} \lor (\theta_j^{rs}(s).u, \theta_i^{as}(s).l) \in \text{hb})
\]

\[
\land \forall s. \forall i, j \in \{1 \ldots n+m\}. \ ((\theta_i^{as}(s).u, \theta_j^{as}(s).l) \in \text{hb} \lor (\theta_j^{as}(s).u, \theta_i^{as}(s).l) \in \text{hb})
\]

\[
\land \forall i, j \in \{1 \ldots n+m\}. \ ((\theta_i^{as}(s).u, \theta_j^{rs}(s).l) \in \text{hb} \lor (\theta_j^{rs}(s).u, \theta_i^{as}(s).l) \in \text{hb})
\]

For each location \(s\), let us then enumerate the successful push and pop operations (i.e., those with a \(\theta_i^{as}(s)\) or \(\theta_j^{rs}(s)\) trace) in order of their lock acquisition. That is, the first operation is either i) a successful push operation associated with trace \(\theta_i^{as}(s)\) such that for all \(i \neq 1\), \(\theta_i^{as}(s).u \rightarrow \theta_i^{as}(s).l\) and \(\theta_1^{as}(s).u \rightarrow \theta_1^{rs}(s).l\); or ii) a successful pop operation associated with trace \(\theta_i^{rs}(s)\) such that for all \(i \neq 1\), \(\theta_i^{rs}(s).u \rightarrow \theta_i^{rs}(s).l\) and \(\theta_1^{rs}(s).u \rightarrow \theta_1^{as}(s).l\).

Let us write \(\text{isPush}(s, i)\) when the \(i\)th operation on \(s\) (as ordered above) is a successful push operation with trace \(\theta_i^{as}(s)\). Similarly, let us write \(\text{isPop}(s, i)\) when the \(i\)th operation on \(s\) (as ordered above) is a successful pop operation with trace \(\theta_i^{rs}(s)\). We can then demonstrate that:

\[
\forall s. \forall i, j \in \{1 \ldots n+m\}. \ i < j \Rightarrow \exists s. \text{isPush}(s, i) \land \text{isPop}(s, j) \land (\theta_i^{as}(s).u, \theta_j^{rs}(s).l) \in \text{hb}
\]

(14)

Let us write \(\text{match}(s, i, j)\) when \(\text{isPush}(s, i) \land \text{isPop}(s, j) \land (\theta_i^{as}(s).u, \theta_j^{rs}(s).r) \in \text{com}\). Let us write \(\text{top}(s, i)\) for \(t\) when either: 1) \(\text{isPush}(s, i)\) and \(\text{lab}(\theta_i^{as}(s).w_t) = \text{store}(r1x, s+1, t)\); or 2) \(\text{isPop}(s, i)\) and \(\text{lab}(\theta_i^{rs}(s).r_t) = \text{load}(r1x, s+1, t)\). It is then straightforward to demonstrate that for all \(i, j\):

\[
\text{isPush}(s, i) \land \text{isPush}(s, i+1) \Rightarrow \text{top}(s, i+1) = \text{top}(s, i) + 1
\]

\[
\text{match}(s, i, j) \Rightarrow \text{top}(s, j) = \text{top}(s, i)
\]

(15)

It is then straightforward to demonstrate by induction that for all \(i, j\):

\[
i < j \land \text{isPush}(s, i) \land \text{isPush}(s, j) \Rightarrow \text{top}(s, i) < \text{top}(s, j) \lor \exists k. i < k < j \land \text{isPop}(s, k) \land \text{match}(s, i, k)
\]

\[
i < j \land \text{isPush}(s, i) \land \text{isPop}(s, j) \Rightarrow \text{top}(s, i) < \text{top}(s, j) \lor \exists k. i < k < j \land \text{isPop}(s, k) \land \text{match}(s, i, k)
\]

(16)

Let \(\text{hb} = G.\text{hb} = (\text{po} \cup \text{so})^+.\) Note that from the definition of \(\text{hb}''\), \(\text{hb}\) and the construction of \(\text{so}'\) above we know the (17) below holds. As such, from the irreflexivity of \(\text{hb}\), we also know (18) holds.

\[
\forall a, b. (a, b) \in \text{hb}' \Rightarrow (\text{im}(a), \text{im}(b)) \in \text{hb}
\]

(17)

\[
\forall a, (a, a) \notin \text{hb}'
\]

(18)

To show that \(G' = (E'', \text{po}'', \text{com}'', \text{so}'', \text{hb}'') \in L^{WS}.G_c\), we are then required to show that for all locations \(s\), \(G'_s\) is weak-stack consistent on \(s\). Pick an arbitrary location \(s\) and let \(G'_s = \langle E', \text{po}', \text{com}', \text{so}', \text{hb}' \rangle\). We then need to show:

1) \(E'^c = \emptyset \lor \exists c \in C^q. \ E'^c = \{c\};\)

2) \(\text{com}' \subseteq \bigcup_{\text{Val}(L)} \mathcal{A}^{k,v,T} \times \mathcal{R}^{k,v,T};\)

3) \(\text{com}'\), \(\text{com}'^{-1}\) are functional;

4) \(E' \cap \mathcal{R}^k \setminus \text{rng}(\text{com}') \subseteq \mathcal{R}^{k, L, L}^k\)
5) so’ = \text{com’}; and
6) \forall a_1, a_2, r_1, r_2, (a_1, r_1), (a_2, r_2) \in \text{com’} \land (a_1, a_2), (r_1, r_2) \in \text{llb’} \Rightarrow (a_2, r_1) \notin \text{llb’}.

Parts (1), (2) and (5) follow immediately from the construction of \( G’ \). Part (3) follows from the definition of \( \text{com’} \), the consistency of \( G \) and the definition of the C library and the fact that \( \text{com}^{-1} \) is functional for \( L^C \).

For part (4), we proceed by contradiction. Let us assume there exists \( r \in E' \cap R^s \) such that \( r \notin \text{rng}(\text{com'}) \) and \( r \notin R^{s,\perp,\perp} \). Let \( \text{lab}(r) = \text{try-pop}(s, v, \neg) \). From the construction of \( G’ \) we then know that there exists \( i \) and \( r' \) such that \( r' = \text{imp}(r) \) and either: i) \( r' = \theta_r^{s(s)} \cdot r \) and \( r = r_i^{rs(s)} \); or ii) \( r' = \theta_i^{rf1(s)} . f \) and \( r = r_i^{rf1(s)} \); or iii) \( r' = \theta_i^{rf2(s)} . f \) and \( r = r_i^{rf2(s)} \). In cases (ii) and (iii) from the construction of \( G’ \) we then know \( \text{lab}(r) = \text{try-pop}(s, \bot, \bot) \) and thus \( r \in R^{s,\perp,\perp} \), contradicting the assumption that \( r \notin R^{s,\perp,\perp} \). In case (i), from the shape of the \( \theta_i^{s(s)} \) we know there exists \( t \) such that \( \text{lab}(\theta_i^{s(s)} . r_i) = \text{load}(r \cdot x, s \cdot t, t) \), \( \text{lab}(r') = \text{load}(r \cdot x, s \cdot t, v) \). Moreover, from the consistency of the C library we know there exists \( w \) such that \( (w, r') \in \text{com} \) and \( \text{lab}(w) = \text{store}(\neg, s \cdot t, \neg) \). As such, from the well-formedness of \( G \) and the shape of the implementation traces we know that there exists \( j \) such that \( w = \theta_j^{as(s)} . a \). Consequently, since \( (w, r') \in \text{incom} \), from the construction of \( G’ \) we know that \( (a_j^{as(s)}, r_i^{rs(s)}) \in \text{com’} \). Since \( r = r_i^{rs(s)} \), this contradicts our assumption that \( r \notin \text{rng}(\text{com’}) \).

For part (6), we proceed by contradiction. Let us assume there exist \( a_1, a_2, r_1, r_2 \) such that \( (a_1, r_1), (a_2, r_2) \in \text{com’} \), \( (a_1, a_2), (r_1, r_2) \in \text{llb’} \) and \( (a_2, r_1) \in \text{llb’} \).

From the construction of \( G’ \) we then know there exist \( i, j, k, l \) such that \( \text{imp}(a_1) = \theta_l^{as(s)} . a \), \( \text{imp}(r_1) = \theta_i^{rs(s)} . r \), \( \text{imp}(a_2) = \theta_k^{as(s)} . a \), \( \text{imp}(r_2) = \theta_j^{rs(s)} . r \), and \( (\text{imp}(a_1), \text{imp}(r_1)), (\text{imp}(a_2), \text{imp}(r_2)) \in \text{com} \). That is we have match(s, i, j) and match(s, j, k).

From (19) we then know \( (\text{imp}(a_1), \text{imp}(a_2)), (\text{imp}(r_1), \text{imp}(r_2)), (\text{imp}(a_2), \text{imp}(r_1)) \) \( \in \text{lb} \). From (14) we then have \( i < k \), \( k < j \) and \( j < l \), since otherwise we would get an \( \text{hb} \) cycle contradicting the assumption that \( G \) is consistent. As such, since we also have match(s, i, j) and match(s, k, l), and match(s, ...,) is uniquely determined (due to the functionality of \( \text{com} \) and \( \text{com}^{-1} \)), from (16) we have \( \text{top}(s, i) < \text{top}(s, k) \) and \( \text{top}(s, k) < \text{top}(s, j) \). That is, we have \( \text{top}(s, i) < \text{top}(s, j) \). On the other hand, since match(s, i, j) holds, from (15) we have \( \text{top}(s, i) = \text{top}(s, j) \). This however leads to a contradiction as we both have \( \text{top}(s, i) < \text{top}(s, j) \) and \( \text{top}(s, i) = \text{top}(s, j) \).
THE SOUNDNESS OF THE ELIMINATION STACK IMPLEMENTATION

Let $I$ denote the elimination stack implementation in Fig. 6. To show the soundness of $I$, we appeal to Thm. 1 and show that $I$ is locally sound on $L^\mathcal{C}$.

Pick an arbitrary $\Lambda, f, G=(E, po, \text{com}, so), E''', po'''$ such that $G$ is $\Lambda$-consistent and $\Lambda$-well-formed and $\text{abs}_{I}(f, (E, po), (E''', po'''))$.

We must next find $\text{com}''$, so'' such that $(E''', po''', \text{com}''$, so'', $\text{lbh}''') \in L^\mathcal{C}_G$, where $\text{lbh}'''$ is the same as $\text{lbh}''$ in Def. 11.

Note that the constructor of the stack at $s$ contains a trace of the following form in $G$: $\theta^{(s)}_c = c_s \xrightarrow{\text{po}} c_w \xrightarrow{\text{po}} c_e \xrightarrow{\text{po}} c_w \xrightarrow{\text{po}} c_e \xrightarrow{\text{po}} \ldots \xrightarrow{\text{po}} c_e \xrightarrow{\text{po}} c_w$ where $\text{lab}(c_s) = \text{alloc}(s, 0), \text{lab}(c_w)=\text{alloc}(ws, 0), \text{lab}(c_e)=\text{alloc}(ea, 0), \text{lab}(c'_w)=\text{store}(rlx, s, ws), \text{lab}(c'_e)=\text{store}(rlx, s+1, ea)$, and for each $j \in \{0 \ldots k-1\}$, $c'_e=\text{of the form } j_nj \rightarrow w_j$, where $\text{lab}(n_j) = \text{new-exchanger}(x_j)$ for some $x_j$, and $\text{lab}(w_j) = \text{store}(rlx, ea+j, x_j)$.

When $G$ contains $n_s$ push operations on the stack at $s$ and $m_s$ pop operations, let us enumerate them arbitrarily. Note that for each $i \in \{1 \ldots n_s\}$, the $i^{th}$ push operation $push(s, v_i)$ either:

- pushes $v_i$ on the weak stack at $s$ and thus $G$ contains the events in the trace: $\theta^{as(i)}_s = r_s \xrightarrow{\text{po}} r_e \xrightarrow{\text{po}} f^* \xrightarrow{\text{po}} a$, where $\text{lab}(r_s) = 1\text{oad}(rlx, s, ws), \text{lab}(r_e) = 1\text{oad}(rlx, s+1, ea), f^*$ denotes the loop iterations that fail to push $v_i$, $\text{lab}(a) = \text{try-push}(ws, v_i, 1); or$
- fails to push $v_i$ on the weak stack and thus pushes it on the elimination array at $s+1$; as such $G$ contains the events in the trace: $\theta^{as(i)}_r = r_s \xrightarrow{\text{po}} r_e \xrightarrow{\text{po}} f^* \xrightarrow{\text{po}} a f \xrightarrow{\text{po}} a$, where $\text{lab}(r_s) = 1\text{oad}(rlx, s, ws), \text{lab}(r_e) = 1\text{oad}(rlx, s+1, ea), f^*$ denotes the loop iterations that fail to push $v_i, \text{lab}(a) = \text{try-push}(ws, v_i, 0), \text{lab}(a) = \text{exchange}(ea[j], v_i, \text{POP})$ for some $j \in \{0 \ldots k-1\}$.

Similarly, for each $i \in \{1 \ldots m_s\}$, the $i^{th}$ pop operation $pop(s)$ either:

- pops $w_i$ from the weak stack and thus $G$ contains the events in the trace: $\theta^{rs(i)}_r = r_s \xrightarrow{\text{po}} r_e \xrightarrow{\text{po}} f^* \xrightarrow{\text{po}} r$, where $\text{lab}(r_s)=1\text{oad}(rlx, s, ws), \text{lab}(r_e)=1\text{oad}(rlx, s+1, ea), f^*$ denotes loop iterations that fail to pop, and $\text{lab}(r) = \text{try-pop}(ws, w_i, 1); or$
- fails to pop from the weak stack and thus pops $w_i$ from the elimination array at $s+1$; as such, $G$ contains the events in the trace: $\theta^{re(i)}_r = r_s \xrightarrow{\text{po}} r_e \xrightarrow{\text{po}} f^* \xrightarrow{\text{po}} r f \xrightarrow{\text{po}} r$, where $\text{lab}(r_s)=1\text{oad}(rlx, s, ws), \text{lab}(r_e)=1\text{oad}(rlx, s+1, ea), f^*$ denotes the iterations that fail to pop, $\text{lab}(r_f)=\text{try-pop}(ws, 1, 0), and \text{lab}(r)=\text{exchange}(ea[j], \text{POP}, w_i)$, for some $j \in \{0 \ldots k-1\}$.

Let us define $\text{imp}(.) : E''' \rightarrow E$ as:

$$\text{imp}(e) \triangleq \begin{cases} \theta^{(s)}_c \cdot c_w \iff s. \theta^{(s)}_c \cdot c_w = e \\
\theta^{as(i)}_s \cdot a \iff i, s. \theta^{as(i)}_s \cdot a = e \\
\theta^{re(i)}_r \cdot a \iff i, s. \theta^{re(i)}_r \cdot a = e \\
\theta^{rs(i)}_r \cdot r \iff i, s. \theta^{rs(i)}_r \cdot r = e \\
\theta^{re(i)}_r \cdot r \iff i, s. \theta^{re(i)}_r \cdot r = e 
\end{cases}$$

Let so'' = com'' with

$$\text{com}'' \triangleq \{(a, r) \mid \exists i, j, s. \theta^{as(i)}_s \cdot a = af(\theta^{rs(i)}_r \cdot r) = r \land (\theta^{as(i)}_s \cdot a, \theta^{rs(i)}_r \cdot r) \in \text{com}\} \cup \{(a, r) \mid \exists i, j, s. \theta^{ae(i)}_a \cdot a = af(\theta^{re(i)}_r \cdot r) = r \land (\theta^{ae(i)}_a \cdot a, \theta^{re(i)}_r \cdot r) \in \text{com}\}$$
Let \( \text{hb} = G.\text{hb} = (\text{po} \cup \text{so})^+ \). Note that from the definition of \( \text{lhb}'' \), \( \text{hb} \) and the construction of \( \text{so}' \) above we know the (19) below holds. As such, from the irreflexivity of \( \text{hb} \), we also know (20) holds.

Let \( \text{hb}' = (\text{po}' \cup \text{so}')^+ \). We next demonstrate that:

\[
\forall a, b. (a, b) \in \text{hb}' \Rightarrow (\text{im}(a), \text{im}(b)) \in \text{hb} \quad (19)
\]

\[
\forall a. (a, a) \notin \text{hb}' \quad (20)
\]

To show that \( G' = (E'', \text{po}'', \text{com}'', \text{so}'', \text{lhb}'') \in L^S.G \), we are then required to show that for all locations \( s \), \( G'_s \) is stack consistent on \( s \). Pick an arbitrary location \( s \) and let \( G'_s = (E', \text{po}', \text{com}', \text{so}', \text{lhb}') \).

We then need to show:

1) \( E'^c = \emptyset \lor \exists c \in C^q. E'^c = \{c\} \);

2) \( \text{com}' \subseteq \bigcup_{v \in \text{Val}\{\bot\}} \mathcal{A}^s,v \times \mathcal{R}^s,v; \)

3) \( \text{com}', \text{com}'^{-1} \) are functional;

4) \( E' \cap \mathcal{R}^s \setminus \text{rng}(\text{com}') \subseteq \mathcal{R}^{s,\bot} \);

5) \( [\mathcal{A}^s \setminus \text{dom}(\text{com}')] \cap \text{hb}' \cap \mathcal{R}^{s,\bot} = \emptyset; \)

6) \( \text{so}' = \text{com}'; \) and

7) \( \forall a_1, a_2, r_1, r_2. (a_1, r_1), (a_2, r_2) \in \text{com}' \land (a_1, a_2), (r_1, r_2) \in \text{hb}' \Rightarrow (a_2, r_1) \notin \text{hb}'. \)

Parts (1), (2) and (6) follow immediately from the construction of \( G' \). Part (3) follows from the construction of \( G' \) and the fact that \( \text{com} \) and \( \text{com}^{-1} \) are functional for \( L^W.S \) and \( L^X \). Part (5) follows trivially as \( \mathcal{R}^{s,\bot} = \emptyset \).

For part (4), we demonstrate that \( E' \cap \mathcal{R}^s \setminus \text{rng}(\text{com}') = \emptyset \). We proceed by contradiction. Let us assume there exists \( r \in E' \cap \mathcal{R}^s \) such that \( r \notin \text{rng}(\text{com}') \). Let \( \text{lab}(r) = \text{pop}(s, v) \) for some value \( v \). From the construction of \( G' \) we then know that there exists \( i \) and \( r' \) such that \( r' = \text{imp}(r) \) and either: i) \( r' = \theta_i^{\text{fs}(s)} . r \); or ii) \( r' = \theta_i^{\text{pe}(s)} . r \).

To see (i) let \( \text{lab}(r') = \text{try}-\text{pop}(w_1, v, 1) \). As such, from the consistency of \( G \) and the specification of the weak stack library we know there exist \( w' \) such that \( \text{lab}(w') = \text{try}-\text{push}(w_1, v, 1) \) and \( (w', r') \in \text{com} \). Moreover, given the well-formedness of \( G \) and the shape of \( G \) traces we know that there exist \( j \) such that \( w' = \theta_j^{\text{as}(s)} . a \). Consequently, from the construction of \( G' \) we know there exist \( w \) such that \( \text{imp}(w) = w' \) and \( (w, r) \in \text{com}' \), contradicting our assumption that \( r \notin \text{rng}(\text{com}') \).

Similarly, in case (ii) we then know that \( \text{lab}(r') = \text{exchange}(a[j], \text{POP}, v) \) for some \( j \). As such, from the consistency of \( G \) and the specification of the exchanger library we know there exist \( w' \) such that \( \text{lab}(w') = \text{exchange}(a[j], v, \text{POP}) \) and \( (w', r') \in \text{com} \). Moreover, given the well-formedness of \( G \) and the shape of \( G \) traces we know that there exist \( j \) such that \( w' = \theta_j^{\text{ae}(s)} . a \). Consequently, from the construction of \( G' \) we know there exist \( w \) such that \( \text{imp}(w) = w' \) and \( (w, r) \in \text{com}' \), contradicting our assumption that \( r \notin \text{rng}(\text{com}') \).

For part (7) we proceed by contradiction. Let us assume there exist \( a_1, a_2, r_1, r_2 \) such that

\[
(a_1, r_1), (a_2, r_2) \in \text{com}', (a_1, a_2), (r_1, r_2) \in \text{hb}', \text{ and } (a_2, r_1) \in \text{hb}'. \quad (19)
\]

From (19) and the construction of \( G' \) we know \( \text{imp}(a_1), \text{imp}(r_1), (\text{imp}(a_2), \text{imp}(r_2)) \in \text{com}, \text{imp}(a_1), \text{imp}(a_2), \text{imp}(r_1), \text{imp}(r_2) \in \text{hb}, \) and \( (\text{imp}(a_2), \text{imp}(r_1)) \in \text{hb} \). There are now three cases to consider: i) \( \text{imp}(a_2) = \theta_i^{\text{fs}(s)} . a \), \( \text{imp}(r_2) = \theta_j^{\text{fs}(s)} . a \), \( \text{imp}(a_1) = \theta_k^{\text{as}(s)} . a \), \( \text{imp}(r_1) = \theta_l^{\text{fs}(s)} . a \) for some \( i, j, k, l \); or ii) \( \text{imp}(a_2) = \theta_i^{\text{ae}(s)} . a \) and \( \text{imp}(r_2) = \theta_j^{\text{ex}(s)} . a \) for some \( i, j, s \).

Case (i) however leads to contradiction as it contradicts the \( L^W.S \)-consistency of \( (G)_{L^W.S} \). In case (ii) from the definition of the \( L^X \) library and the symmetry of its \( \text{com} \) relation we know that

\[(\text{imp}(r_2), \text{imp}(a_2)) \in \text{com}.\] As such, we have \(\text{imp}(a_2) \xrightarrow{\text{hb}} \text{imp}(r_1) \xrightarrow{\text{hb}} \text{imp}(r_2) \xrightarrow{\text{com}} \text{imp}(a_2).\) That is, \(\text{imp}(a_2) \rightarrow \text{imp}(r_2) \rightarrow \text{imp}(a_2),\) contradicting the assumption that \(G\) is consistent.

Similarly, in case (iii) from the definition of the \(L^X\) library and the symmetry of its \text{com} relation we know that \(\text{imp}(r_1), \text{imp}(a_1)) \in \text{com}.\) As such, we have \(\text{imp}(a_1) \xrightarrow{\text{hb}} \text{imp}(a_2) \xrightarrow{\text{com}} \text{imp}(r_1) \xrightarrow{\text{hb}} \text{imp}(a_1).\) That is, \(\text{imp}(a_1) \rightarrow \text{imp}(r_1) \rightarrow \text{imp}(a_1),\) contradicting the assumption that \(G\) is consistent.

**K THE SOUNDNESS OF EXCHANGER IMPLEMENTATION**

Let \(I\) denote the exchanger implementation in Fig. 5. To show the soundness of \(I,\) we appeal to Thm. 1 and show that \(I\) is locally sound on \(L^X.\)

Pick an arbitrary \(\Lambda, f, G=\langle E, \text{po}, \text{com}, \text{so}, E', \text{po}' \rangle\) such that \(G\) is \(\Lambda\)-consistent and \(\Lambda\)-well-formed and \(\text{abs}_{L^X}(f, \langle E, \text{po}, (E, \text{po}') \rangle).\) We must next find \text{com}', \text{so}' such that \(\langle E', \text{po}', \text{com}', \text{so}', \text{lb}' \rangle \in L^X \cdot G_c,\) where \(\text{lb}'\) is as defined in Def. 11.

Note that each exchange operation \(\text{exchange}(g, v)\) either:

1. offers a value at index \(g+k\) (for some \(k \in \mathbb{N}^+\) where \(k\) is an odd number) and returns unmatched due to a timeout; or
2. offers a value at index \(g+k\) (for some \(k \in \mathbb{N}^+\) where \(k\) is an odd number) and matches with value \(v'\) at index \(g+k+1;\) or
3. tries to offer a value at index \(g+k\) (for some \(k \in \mathbb{N}^+\) where \(i\) is an even number) and returns unmatched as the slot at index \(g+k\) is already taken; or
4. offers a value at index \(g+k\) (for some \(k \in \mathbb{N}^+\) where \(k\) is an even number) and matches with value \(v'\) at index \(g+k-1;\)

Without loss of generality, let us assume \(c\) contains \(n\) exchange calls. For each \(i\)th exchange operation of the form \(\text{exchange}(g, v_i),\) the \(G_i\) contains a trace of one of the following forms depending which of the four categories above it falls into:

- \(\theta_i^{(g)}\) is of the form \(s \xrightarrow{\text{po}_i} a \xrightarrow{\text{po}_i} t,\) with \(\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(o) = \text{CAS}(rlx, g+j_i, 0, v),\) and \(\text{lab}(t) = \text{CAS}(rlx, g+j_i+1, 1, v')\), for some \(j_i\) and \(v\) where \(j_i\) is odd;
- \(\theta_i^{(g)}\) is of the form \(s \xrightarrow{\text{po}_i} o \xrightarrow{\text{po}_i} a \xrightarrow{\text{po}_i} r,\) with \(\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(a) = \text{CAS}(rlx, g+j_i, 0, v),\) for some \(j_i, v\);
- \(\theta_i^{(g)}\) is of the form \(s \xrightarrow{\text{po}_i} f \xrightarrow{\text{po}_i} o \xrightarrow{\text{po}_i} c,\) with \(\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(f) = \text{load}(rlx, g+j_i, w_i)\) with \(w_i \neq 0, \text{lab}(a) = \text{load}(rlx, g+j_i+1, w_i')\) with \(w_i' \neq 0,\) and \(\text{lab}(c) = \text{CAS}(rlx, g, j_i+1, 0),\) for some \(j_i, v\);
- \(\theta_i^{(g)}\) is of the form \(s \xrightarrow{\text{po}_i} f \xrightarrow{\text{po}_i} o \xrightarrow{\text{po}_i} c \xrightarrow{\text{po}_i} r,\) with \(\text{lab}(s) = \text{load}(rlx, g, j_i), \text{lab}(f) = \text{load}(rlx, g+j_i, w_i)\) with \(w_i \neq 0, \text{lab}(a) = \text{CAS}(rlx, g+j_i+1, 0),\) for some \(j_i, v\);

Let \(\text{so}' = \text{com}'\) with

\[\text{com}' \triangleq \left\{ (a, b), (b, a), \left| \exists i, j, g, f(e_i^{(g)} . a) = a \land f(e_j^{(g)} . o) = b \right\} \land \left( (\theta_i^{(g)} . a), \theta_j^{(g)} . c) \in \text{com} \land (\theta_i^{(g)} . o), \theta_j^{(g)} . r) \in \text{com} \right\} \]

To show that \(G' = (E', \text{po}', \text{com}', \text{so}', \text{lb}') \in L^X \cdot G_c,\) we are then required to show for all \(g \in \text{Loc},\) 
\(G_g\) is exchanger-consistent on \(g.\) Pick an arbitrary \(g \in \text{Loc},\) we then need to show:
1) \( E^c = \emptyset \lor \exists c \in C^g \). \( E^c = \{c\} \);

2) \( \text{com}' \) is symmetric, irreflexive and \( \text{com}' \subseteq \bigcup_{v_1, v_2 \in \text{Val}} X^{g, v_1, v_2} \times X^{g, v_2, v_1} \setminus \text{id} \);

3) \( \text{com}' \) is functional;

4) \( E' \cap X^g \setminus \text{dom} (\text{com}') \subseteq \bigcup_{v \in \text{Val}} X^{g, v, \bot} \); and

5) so' = \( \text{com}' \).

Parts (1), (2), and (5) follow immediately from the construction of \( G_i \) and the consistency of the C library. The proof of parts (3) and (4) follows from the definition of \( \text{com}' \), the consistency of \( G_i \) and the definition of the C library.