The Anchor Verifier for Blocking and Non-blocking Concurrent Software: Supplementary Appendix

CORMAC FLANAGAN, University of California, Santa Cruz, USA
STEPHEN N. FREUND, Williams College, USA

Verifying the correctness of concurrent software with subtle synchronization is notoriously challenging. We present the ANCHOR verifier, which is based on a new formalism for specifying synchronization disciplines that describes both (1) what memory accesses are permitted, and (2) how each permitted access commutes with concurrent operations of other threads (to facilitate reduction proofs). ANCHOR supports the verification of both lock-based blocking and cas-based non-blocking algorithms. Experiments on a variety concurrent data structures and algorithms show that ANCHOR significantly reduces the burden of concurrent verification.

CCS Concepts: • **Theory of computation** → **Program verification**; **Program specifications**; • **Software and its engineering** → **Concurrent programming languages**; **Formal software verification**;

Additional Key Words and Phrases: concurrent program verification, reduction, synchronization

ACM Reference Format:
Cormac Flanagan and Stephen N. Freund. 2020. The Anchor Verifier for Blocking and Non-blocking Concurrent Software: Supplementary Appendix. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 156 (November 2020), 34 pages. https://doi.org/10.1145/3428224

A SUPPLEMENTARY APPENDIX

A.1 Generic Reduction Theorem

We start by presenting a generic reduction theorem that extends Theorem 1 of [Flanagan and Qadeer 2003]. The key extension is that we now consider two operations to commute even if swapping them could introduce new errors (but never hide old errors), as any errors from a preemptive execution will still also be present in the corresponding cooperative execution obtained via repeated swapping.

Given a set of states $State$, with error states $E \subseteq State$, and transition relations $\rightarrow_i, \rightarrow_j \subseteq State \times State$, we say that $\rightarrow_i$ *commutes with* $\rightarrow_j$ *with respect to* E if whenever $s_1 \rightarrow_i s_2 \rightarrow_j s_3$ then either:

(1) $\exists e \in E.\ s_1 \rightarrow_j e$, or
(2) $\exists s_4 \in State, e \in E.\ s_1 \rightarrow_j s_4 \rightarrow_i e$, or
(3) $\exists s_4 \in State.\ s_1 \rightarrow_j s_4 \rightarrow_i s_3$.

Thus, swapping $\rightarrow_i$ and $\rightarrow_j$ transitions either does not affect the final state s_3 or introduces a new error state e.

Authors' addresses: Cormac Flanagan, University of California, Santa Cruz, Santa Cruz, CA, USA; Stephen N. Freund, Williams College, Williamstown, MA, USA.

2475-1421/2020/11-ART156
https://doi.org/10.1145/3428224

For $S \subseteq \mathit{State}$, we define

$$\begin{array}{rcl} S/{\rightarrow_i} & = & (\rightarrow_i) \cap (S \times \mathit{State}) \\ \rightarrow_i \backslash S & = & (\rightarrow_i) \cap (\mathit{State} \times S) \end{array}$$

Theorem 6 (Generic Reduction)**.** For all i, let R_i, L_i, and E_i be sets of states and $\rightharpoonup_i$ be a transition relation. Suppose, for all i,

(1) R_i, L_i, and E_i are pairwise disjoint.
(2) $(L_i/{\rightharpoonup_i}\backslash R_i)$ is false.

and for all $j \neq i$,

(3) $\rightharpoonup_i$ and $\rightharpoonup_j$ are disjoint.
(4) $(\rightharpoonup_i \backslash R_i)$ commutes with $\rightharpoonup_j$ with respect to E.
(5) $\rightharpoonup_j$ commutes with $(L_i/{\rightharpoonup_i})$ with respect to E.
(6) if $p \rightharpoonup_i q$, then $R_j(p) \Leftrightarrow R_j(q)$, $L_j(p) \Leftrightarrow L_j(q)$, $E_j(p) \Leftrightarrow E_j(q)$.
(7) $\forall q \in L_i.\ q \rightharpoonup_i^* q' \in N_i$, *i.e.*, q is non-blocking.
(8) $E_i/{\rightharpoonup_i} = \emptyset$.

Let

$$\begin{array}{rcl} N_i & = & \neg(R_i \vee L_i) \\ N & = & \forall i.\ N_i \\ E & = & \exists i.\ E_i \\ \rightharpoonup & = & \exists i.\ \rightharpoonup_i \\ \rightharpoonup\!\!\!\!\rightharpoonup & = & \exists i.\ (\forall j \neq i.\ N_j)/{\rightharpoonup_i} \end{array}$$

Suppose $p \in N$ with $p \not\rightharpoonup\!\!\!\!\rightharpoonup^* E$. Then

(1) $p \not\rightharpoonup^* E$, and
(2) For all $q \in N$, if $p \rightharpoonup^* q$ then $p \rightharpoonup\!\!\!\!\rightharpoonup^* q$.

Proof. The proof of this theorem follows the structure of the proof of Theorem 1 in [Flanagan and Qadeer 2003], extended to permit the reduced trace to go wrong more often than the original.

We begin by defining several terms.

A *pre-commit sequence* by thread i is a sequence of states starting in N_i where each transition is a right-mover. We summarize the beginning and ending states of a pre-commit sequence as

$$\mathit{Pre}_i \quad = \quad N_i/((\rightharpoonup_i \backslash R_i)^*)$$

A *post-commit sequence* by thread i is a pre-commit sequence followed by a committing transition $\rightharpoonup_i$, followed by left-movers $(L_i/{\rightharpoonup_i})$.

$$\mathit{Post}_i \quad = \quad \mathit{Pre}_i; (\rightharpoonup_i \backslash L_i); (L_i/{\rightharpoonup_i})^*$$

where $\cdot;\cdot$ denotes the composition of relations, *i.e.*:

$$\mathit{Pre}_i; \rightharpoonup_i \quad = \quad \{\, (s, s'') \mid \exists s'.(s, s') \in \mathit{Pre}_i \wedge (s', s'') \in \rightharpoonup_i \,\}$$

A *finished sequence* by thread i is a post-commit sequence that ends in N_i:

$$\mathit{Finish}_i \quad = \quad \mathit{Post}_i \backslash N_i$$

We note that $\mathit{Pre}_i, \mathit{Post}_i, \mathit{Finish}_i \subseteq \mathit{State} \times \mathit{State}$. Let

$$\begin{array}{rcl} \mathit{Pre} & = & \exists i.\ \mathit{Pre}_i \\ \mathit{Post} & = & \exists i.\ \mathit{Post}_i \\ \mathit{Finish} & = & \exists i.\ \mathit{Finish}_i \end{array}$$

We first prove that

$$\text{If } (p, q) \in \mathit{Post}^*; \mathit{Pre}^* \text{ then } q \notin E \tag{1}$$

Let $L(q) = \{\ i \mid q \in L_i\ \}$. Proof is by induction on $|L(q)|$.

- Base Case: Suppose $L(q) = \emptyset$, and assume $q \in E_i$ for some i. Then $(p, q) \in \mathit{Finish}^*; \mathit{Pre}_i$. Furthermore, since a Pre_i step cannot end in E_i, $\exists q'.\ (p, q') \in \mathit{Finish}^* \wedge q' \in E_i$. Hence, $p \rightharpoonup\!\!\!\rightharpoonup^* q'$, contradicting the assumption that $p \not\rightharpoonup\!\!\!\rightharpoonup^* q'$.
- Inductive Case: Assume $(p, q) \in \mathit{Post}^*; \mathit{Pre}^*$ and $j \in L(q)$, *i.e.* $q \in L_j$. By the non-blocking assumption, $q \rightharpoonup_j^* q' \in N_j$. Thus, we can left-commute these $q \rightharpoonup_j^* q'$ steps to the end of the last Post_j block to yield one of the following cases:
  - $(p, q') \in \mathit{Post}^*; \mathit{Pre}^*$ and $L(q') \subset L(q) \setminus \{j\}$ if no commuting steps yield an error. By induction, $q' \notin E$, and thus $q \notin E$.
  - If a commuting step yields an error state $q'' \in E$, then $L(q'') \subset L(q)$, but by induction this is impossible.

We next prove that

$$\text{If } p \rightharpoonup^* q \text{ then } (p, q) \in \mathit{Post}^*; \mathit{Pre}^* \tag{2}$$

The proof is by induction on the length of sequence $p \rightharpoonup^* q$.

- Base Case: If $p = q$, this is trivial.
- Inductive Case: Assume $p \rightharpoonup^* q \rightharpoonup_i q'$ and $(p, q) \in \mathit{Post}^*; \mathit{Pre}^*$. There are several cases to consider:
  - $(q, q') \in N_i \times (N_i \vee L_i)$. Then all Pre blocks right-commute over this $\rightharpoonup_i$ step, and we can add this $\rightharpoonup_i$ step as additional Post block, yielding $(p, q') \in \mathit{Post}^*; \mathit{Pre}^*$.
  - $(q, q') \in N_i \times R_i$. Then we can add this step as a new Pre block.
  - $q \in L_i$. Then the last i block must be a Post_i block, and $q \rightharpoonup_i q'$ left commutes to the end of this Post_i block.
  - $q \in R_i$. Then the last i block must be a Pre_i block, and the subsequent fsPre blocks right-commute over $q \rightharpoonup_i q'$. Thus we can add this step at the end of the Pre_i block.

We now prove the two parts of this theorem. To show part 1, assume $p \rightharpoonup^* q$. By (2), $(p, q) \in \mathit{Post}^*; \mathit{Pre}^*$. By (1), $q \notin E$.

To show part 2, assume $q \in N$ and $p \rightharpoonup^* q$. Then $(p, q) \in \mathit{Post}^*; \mathit{Pre}^*$ by (2). But since $q \in N$, $(p, q) \in \mathit{Post}^*$. Hence, each Post_i block must end in N_i, so $(p, q) \in \mathit{Finish}^*$, and $p \rightharpoonup\!\!\!\rightharpoonup^* q$.

□

A.2 Proofs for Section 6

We now leverage the generic reduction theorem to prove our central reduction argument for Anchor.

First, we prove that $\Rightarrow$-steps right-commute (or go wrong) in the Pre phase and left-commute (or go wrong) in the Post phase. Proving this property requires a notion like strict stability from Definition 5.2. That notion of stability is satisfied by many synchronization specifications. However, strict stability is sometimes overly restrictive. For example, it prohibits interleaved writes from increasing the access permissions of a thread. For maximal expressiveness, we instead derive a more relaxed notion of stability from the conditions necessary to prove the following commuting lemma for all pairs of possible execution steps.

Definition A.1 (Stability). *Specification R, W is stable if for all l, k t, H, v, w, and u where $u \neq t$:*

$$\begin{array}{cr@{\ \land\ }lcl}
\text{A.1} & W_l(t,H,v) \sqsubseteq \mathsf{R} & W_k(u,H[l:=v],w) \sqsubseteq \mathsf{E} & \Rightarrow & W_k(u,H,w) \in \{\, W_k(u,H[l:=v],w), \mathsf{E} \,\} \\
\text{A.2} & W_l(t,H,v) \neq \mathsf{E} & W_k(u,H[l:=v],w) \not\sqsubseteq \mathsf{L} & \Rightarrow & W_k(u,H,w) \not\sqsubseteq \mathsf{L} \\
\text{A.3} & k \neq l \land W_l(t,H,v) \neq \mathsf{E} & W_k(u,H[l:=v],w) \sqsubseteq \mathsf{L} & \Rightarrow & W_k(u,H,w) \in \{\, W_k(u,H[l:=v],w), \mathsf{E} \,\} \\
\text{C} & W_l(t,H,v) \sqsubseteq \mathsf{E} & W_k(u,H,w) \sqsubseteq \mathsf{L} & \Rightarrow & W_l(t,H[k:=w],v) \in \{\, W_l(t,H,v), \mathsf{E} \,\} \\
\text{D} & W_l(t,H,v) \sqsubseteq \mathsf{R} & W_k(u,H[l:=v],w) \sqsubseteq \mathsf{N} & \Rightarrow & W_l(t,H[k:=w],v) \in \{\, W_l(t,H,v), \mathsf{E} \,\} \\
\text{E} & W_l(t,H,v) \sqsubseteq \mathsf{N} & W_k(u,H[l:=v],w) \sqsubseteq \mathsf{L} & \Rightarrow & W_l(t,H[k:=w],v) \in \{\, W_l(t,H,v), \mathsf{E} \,\} \\
\text{F} & R_l(t,H) \sqsubseteq \mathsf{R} & W_k(u,H,w) \sqsubseteq \mathsf{N} & \Rightarrow & R_l(t,H[k:=w]) \in \{\, R_l(t,H), \mathsf{E} \,\} \\
\text{H} & R_l(t,H,v) \sqsubseteq \mathsf{E} & W_k(u,H,w) \sqsubseteq \mathsf{L} & \Rightarrow & R_l(t,H[k:=w]) \in \{\, R_l(t,H), \mathsf{E} \,\} \\
\text{I} & k \neq l \land R_l(t,H) \sqsubseteq \mathsf{N} & W_k(u,H,w) \sqsubseteq \mathsf{N} & \Rightarrow & R_l(t,H[k:=w]) \in \{\, R_l(t,H), \mathsf{E} \,\} \\
\text{J} & W_l(t,H,v) \sqsubseteq \mathsf{R} & R_k(u,H) \sqsubseteq \mathsf{E} & \Rightarrow & R_k(u,H) \in \{\, R_k(u,H[l:=v]), \mathsf{E} \,\} \\
\text{K} & W_l(t,H,v) \sqsubseteq \mathsf{N} & R_k(u,H[l:=v]) \sqsubseteq \mathsf{L} & \Rightarrow & R_k(u,H) \in \{\, R_k(u,H[l:=v]), \mathsf{E} \,\} \\
\text{L} & W_l(t,H,v) \sqsubseteq \mathsf{N} & R_k(u,H[l:=v]) \not\sqsubseteq \mathsf{L} & \Rightarrow & R_k(u,H) \not\sqsubseteq \mathsf{L} \\
\text{M} & W_l(t,H,v) \sqsubseteq \mathsf{N} & W_k(u,H,w_1) \sqsubseteq \mathsf{E} & \Rightarrow & W_k(u,H[k:=w_1],w_2) \in \{W_k(u,H[l:=v][k:=w_1],w_2), \mathsf{E}\} \\
\text{N} & W_l(t,H,v_1) \sqsubseteq \mathsf{E} & W_k(u,H,w) \sqsubseteq \mathsf{L} & \Rightarrow & W_l(t,H[k:=w][l:=v_1],v_2) \in \{\, W_l(t,H[l:=v_1],v_2), \mathsf{E} \,\}
\end{array}$$

Shaded terms are trivially true but included for symmetry among the rules. Rules M and N are only applicable if l or k is manipulated via a cas operation.

Lemma 1 (Commuting). Suppose R, W is a valid and stable synchronization discipline, $u \neq t$, and

$$\Pi_1 \Rightarrow_t \Pi_2 \Rightarrow_u \Pi_3$$

where $\Pi_2 = T_2 \cdot H_2 \cdot C_2 \cdot P_2$, and $P_2(t) = \textsc{Pre}$ or $P_2(u) = \textsc{Post}$. Then:

(1) $\exists \Pi_4.\ \Pi_1 \Rightarrow_u \Pi_4 \Rightarrow_t \Pi_3$, or
(2) $\exists \Pi_4, \Pi_5.\ \Pi_1 \Rightarrow_u \Pi_4 \Rightarrow_t \Pi_5$ and t is wrong in Π_5, or
(3) $\exists \Pi_4.\ \Pi_1 \Rightarrow_u \Pi_4$ and u is wrong in Π_4.

Proof. By case analysis on all pairs of possibly conflicting operations. □

Next, we define the predicates $\mathbb{R}_i, \mathbb{L}_i, \mathbb{E}_i, \mathbb{N}_i, \mathbb{N} \subseteq \Pi$ over an instrumented state $\Pi = T \cdot H \cdot C\dot{P}$:

$$\begin{array}{rcl}
\mathbb{R}_i(\Pi) & = & P(i) = \textsc{Pre} \land \neg \mathbb{N}_i(\Pi) \\
\mathbb{L}_i(\Pi) & = & P(i) = \textsc{Post} \land \neg \mathbb{N}_i(\Pi) \\
\mathbb{E}_i(\Pi) & = & \text{thread } i \text{ is wrong in } \Pi \\
\mathbb{N}_i(\Pi) & = & \text{thread } i \text{ is yielding in } \Pi \\
\mathbb{N}(\Pi) & = & \forall i.\ \mathbb{N}_i(\Pi)
\end{array}$$

These predicates characterize whether thread i is in the pre-commit, or right-mover, part of a reducible code sequence ($\mathbb{R}_i$); in the post-commit, or left-mover, part ($\mathbb{L}_i$); gone wrong ($\mathbb{E}_i$); or is yielding ($\mathbb{N}_i$).

We can now formalize the *nonblocking* requirement that any transaction that has passed its commit point must be able to terminate, i.e. if $\Pi_0 \Rightarrow^* \Pi'$ and $\mathbb{L}_i(\Pi')$ then $\Pi' \Rightarrow_i^* \Pi''$ such that $\mathbb{N}_i(\Pi'')$.

Restatement of Theorem 3 (Reduction). If $\Pi = T_0 \cdot H_0 \cdot C_0 \cdot P_0$ and Π_0 does not go wrong under $\Mapsto$, then:

(1) Π_0 does not go wrong under $\Rightarrow$.
(2) If $\Pi_0 \Rightarrow^* \Pi$ where Π is yielding, then $\Pi_0 \Mapsto^* \Pi$.

Proof. By Theorem 6, where the preconditions of that theorem are satisfied as follows. For all $i, j \in \mathit{Tid}$ where $i \neq j$:

(1) $\mathbb{R}_i$, $\mathbb{L}_i$, and $\mathbb{E}_i$ are pairwise disjoint by construction.

(2) $\mathbb{L}_i \,/ \Rightarrow_i \backslash \mathbb{R}_i$ is false. $\mathbb{L}_i$ says that $P(i) = \textsc{Post}$ and thread i is not yielding. $\mathbb{R}_i$ says that $P(i) = \textsc{Pre}$. However, the only transition of thread i from Post to Pre is for `yield`.
(3) $\Rightarrow_i$ and $\Rightarrow_j$ are disjoint.
(4) $\Rightarrow_i \,/\, \mathbb{R}_i$ commutes with $\Rightarrow_j$ with respect to E_i. By Lemma 1.
(5) $\Rightarrow_j$ commutes with $\mathbb{L}_i \backslash \Rightarrow_i$ with respect to E_i. By Lemma 1.
(6) $\Rightarrow_i$ does not change the phase or statements of other threads.
(7) By the non-blocking assumption.
(8) By definition of $\Rightarrow_i$.

□