On the Succinctness of Idioms for Concurrent Programming: Supplementary Material

David Harel[1], Guy Katz[1], Robby Lampert[2], Assaf Marron[1], and Gera Weiss[3]

1 **Weizmann Institute of Science, Rehovot, Israel**
2 **Mobileye Vision Technologies Ltd., Jerusalem, Israel**
3 **Ben Gurion University, Beer-Sheva, Israel**

A Proof of Proposition 1

We wish to show that $\mathcal{RWB} \xrightarrow{p} \mathcal{C}$.

Consider an $\mathcal{RWB}$-automaton A with threads $T_1, \ldots, T_n$, each described by a tuple $T_i = \langle Q^i, \Sigma^i, \delta^i, q_0^i, R^i, B^i \rangle$. This naturally gives rise to a $\mathcal{C}$-automaton M, in the following manner. The alphabet Σ of M is the set of all thread events, $\Sigma = \cup \Sigma^i$. M has n components $M^1, \ldots, M^n$, each corresponding to a single thread. Component M^i has the same states and initial state as its corresponding thread, Q^i and q_0^i.

The transition relation of M^i is defined as follows: For every $e \in \Sigma$, let

$$\mathfrak{R}(e) = \{q \mid \exists i, q \in Q^i, e \in R^i(q)\}, \quad \mathfrak{B}(e) = \{q \mid \exists i, q \in Q^i, e \in B^i(q)\}$$

denote the set of states in which individual threads request/block event e. For every event $e \in \Sigma$ and every thread T_i, for each state $q \in Q^i$ we define a transition in M^i by $\langle q, e, (\bigvee_{u \in \mathfrak{R}(e)} u) \wedge (\neg \bigvee_{v \in \mathfrak{B}(e)} v), \hat{q} \rangle$, where if $e \in \Sigma^i$ then $\hat{q} = \bar{\delta}^i(q, e)$, and if $e \notin \Sigma^i$ then $\hat{q} = q$. Thus, the transitions imitate the operation of the RWBA, while their guards guarantee that they are applicable iff the event is requested by at least one thread and is blocked by none. Recall that for $e \in \Sigma^i$, q_i and e uniquely determine $\hat{q}_i$ — and so the definition is sound. It is clear that both automata accept the same language. Further, the resulting $\mathcal{C}$-automaton is of size polynomial in the size of A. In particular, we introduce at most $|A|^2$ edges and at most $|A|$ guards, each of size at most $|A|$.

B Proof of Proposition 5

We show that for any two models $\mathcal{M}_1, \mathcal{M}_2 \in \{\mathcal{RWB}, \mathcal{WB}, \mathcal{RB}, \mathcal{RW}\}$, it holds that $\mathcal{M}_1 \dot{\rightarrow} \mathcal{M}_2$.

Every $\mathcal{RB}$- or $\mathcal{RW}$-automaton is also an $\mathcal{RWB}$-automaton, in which, at every synchronization point, the waited-for or blocked events set are empty, respectively. A $\mathcal{WB}$-automaton can also be regarded as an $\mathcal{RWB}$-automaton, where all threads request all events at each synchronization point. Hence, $\mathcal{M}_1 \xrightarrow{p} \mathcal{RWB}$.

By Proposition 1, we know that $\mathcal{RWB} \xrightarrow{p} \mathcal{C}$. By [1], we know that $\mathcal{C} \dot{\rightarrow}$ DLA. Finally, a DLA can be translated into an $\mathcal{M}_2$-automaton with a single thread that imitates the DLA, as follows. The thread has the same states and transitions as the DLA, and:

- For $\mathcal{M}_2 = \mathcal{WB}$, in each state the thread blocks any events for which there are no transitions, and waits for the events of all outgoing transitions.
- For $\mathcal{M}_2 = \mathcal{RB}$, $\mathcal{M}_2 = \mathcal{RW}$ or $\mathcal{M}_2 = \mathcal{RWB}$, in each state the thread only requests events for which there are outgoing transitions, blocking no events.

It follows that DLA $\xrightarrow{p} \mathcal{M}_2$. Thus, putting these together yields:

$$\mathcal{M}_1 \xrightarrow{p} \mathcal{C} \dot{\rightarrow} \text{DLA} \xrightarrow{p} \mathcal{M}_2$$

And so $\mathcal{M}_1 \stackrel{\cdot}{\to} \mathcal{M}_2$, as needed.

C Proof of Proposition 6

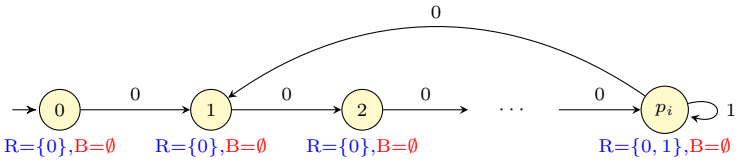
We wish to prove that $\mathcal{RWB} \not\stackrel{\cdot}{\to} \mathcal{WB}$.

Let $n \in \mathbb{N}$, and let $k \in \mathbb{N}$ be the smallest number such that the first k prime numbers, $p_1, \ldots, p_k$, satisfy $\prod_{i=1}^{k} p_i > n$. Define the family of languages $L_n = \{\ell_1 \ell_2 \ldots\}$ by:

$$\ell_j = \begin{cases} 0 \text{ or } 1 & ; \quad \exists i \text{ such that } p_i \mid \#_0(\ell_1 \ldots \ell_{j-1}) \\ 0 & ; \quad \text{otherwise} \end{cases}$$

Here $\#_0(\ell_1, \ldots, \ell_{j-1})$ is the number of 0s that have appeared in the word so far. The jth event can be either 0 or 1 if there is a p_i which divides this number.

In the $\mathcal{RWB}$ model, this language is accepted by an automaton with k threads, $T_1, \ldots, T_k$, where T_i is given by:



In state p_i the thread requests 0 and 1, and in all other states it only requests 0. The size analysis is the same as in the proof of Proposition 3, and the automaton is of size $O(\log^2 n \cdot \log\log n)$.

Consider a $\mathcal{WB}$-automaton that accepts this language and the word $\sigma = 0^\omega \in L_n$. Assume that all threads have less than n states. By the pigeonhole principle they are traversing cycles in their respective transition systems as the automaton reads σ.

There are infinitely many indices in which the triggering of 1 would result in a word (either finite or infinite) that is not in L_n being accepted by the automaton. Hence, triggering 1 has to be prevented in these indices, which, in the $\mathcal{WB}$ model, means it has to be blocked infinitely often. Therefore, there is at least one thread T_j whose cycle contains a state q in which 1 is blocked. Let β denote the length of that cycle. By our assumption, $\beta < n$. Consequently, there is at least one prime p_i such that p_i and β are coprime. By the Chinese Remainder Theorem, the automaton will eventually reach a point in the run where p_i divides the number of zeroes encountered so far, and so 1 should be allowed, but in which thread T_j is in state q, blocking 1 — thus preventing a word that is in the language from being accepted by the automaton. Hence, there must be at least one thread with n or more states, proving the claim.

D Proof of Proposition 7

We wish to prove that $\mathcal{RWB} \not\stackrel{\cdot}{\to} \mathcal{RB}$.

Note that in the $\mathcal{RB}$ model, since threads do not wait for anything unless they explicitly request it, if a thread has no requests at some synchronization point it remains in that state forever, either blocking some events forever, or doing nothing at all.

Let $n \in \mathbb{N}$, and consider the family of singleton languages $L_n = 0^n 1^\omega$. In Section 3.2 we saw that there exists an $\mathcal{RWB}$-automaton of size $O(\log^2 n \cdot \log\log n)$ that accepts L_n.

Now, suppose that an $\mathcal{RB}$-automaton accepts L_n, and that all its threads have less than n states. Consider the automaton after reading the input 0^n. As all threads have less than n states, they are all traversing cycles in their transition systems where all the edges are labeled 0. We call these cycles 0-cycles. In the $\mathcal{RB}$ model, these cycles can be of two kinds:

1. Single-state cycles, where the thread does not request 0; hence, a 0 event being triggered leaves the thread in that state.
2. Cycles in which all states request 0; in these cycles, the thread moves to a new state whenever 0 is triggered.

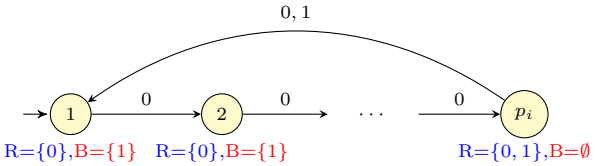
Neither kind of cycle can have a state that blocks 0 inside it; otherwise the 0^n prefix would already be rejected, although it is a prefix of a word in L_n. Also, there is at least one thread in a cycle that requests 0s; otherwise the automaton would be stuck — again, rejecting a word in L_n.

Let us see what happens when the automaton reads the word 0^ω. After reading the 0^n prefix, the threads are already traversing their cycles, in which 0 is never blocked. At least one thread is constantly requesting 0 in every state of its cycle. Hence, the threads will continue to traverse their cycles indefinitely as they read 0^ω, accepting the word although it is not in L_n.

E Proof of Proposition 8

We wish to prove that $\mathcal{RWB} \not\to \mathcal{RW}$.

Let $n \in \mathbb{N}$, and let $k \in \mathbb{N}$ be the smallest number such that the first k prime numbers, $p_1, \ldots, p_k$, satisfy $\prod_{i=1}^{k} p_i > n$. We prove the claim using the family of languages $L_n = (0^{N-1}(0+1))^\omega$, where $N = \prod_{i=1}^{k} p_i$. In the $\mathcal{RWB}$ model, this language is accepted by an automaton with k threads $T_1, \ldots, T_k$, where T_i is given by:



In state p_i the thread requests both 0 and 1, and in the other states it requests 0 but blocks 1. Thus, 0 may always be triggered, but 1 may be triggered only when all threads are in their respective p_i states, as required. The size analysis is the same as in the proof of Proposition 3, and the automaton is of size $O(\log^2 n \cdot \log\log n)$.

Assume we have an $\mathcal{RW}$-automaton that accepts this language, and consider the word $0^\omega \in L_n$. Again, by the pigeonhole principle, the threads will, during this run, indefinitely traverse cycles within their respective transition systems. Clearly, event 1 has to be requested infinitely often throughout the run, in order to allow all the words of L_n to be accepted. Therefore, in at least one of the threads' cycles, there will be a state in which 1 is requested. Denote the length of that cycle by α.

In the $\mathcal{RW}$ model, threads cannot block events. Thus, every time 1 is requested it may be triggered. Since there cannot exist two words in the language that have 1s in indices that are less than N steps apart, it follows that $\alpha \geq N > n$. Thus, the total size of the $\mathcal{RW}$-automaton is at least n.

F A Rigorous Comparison between $\mathcal{RB}$, $\mathcal{WB}$ and $\mathcal{RW}$

▶ Proposition I. $\mathcal{RB} \not\to \mathcal{WB}$ and $\mathcal{WB} \xrightarrow{p} \mathcal{RB}$

Proof. First, observe that the proof of Proposition 6, showing that $\mathcal{RWB} \not\to \mathcal{WB}$, used an $\mathcal{RWB}$-automaton that did not utilize the waiting idiom; hence, the same proof applies here.

For the other direction, note that any $\mathcal{WB}$-automaton can be trivially translated into an $\mathcal{RB}$-automaton. In $\mathcal{WB}$, a thread can be regarded as constantly requesting all events, and waiting for only some of them. Such a thread can be converted into an equivalent thread in the $\mathcal{RB}$ model: it continues to request all events, and reacts (changes state) only to events for which the original thread waited. The resulting automaton accepts the same language as the original automaton, and has the same size. ◀

In light of Proposition I, Proposition 7 is particularly surprising. It shows that although the $\mathcal{WB}$ model is a weaker than the $\mathcal{RB}$ model, their combination affords greater succinctness than either of them affords separately.

▶ Proposition II. $\mathcal{RW} \not\to \mathcal{WB}$ and $\mathcal{WB} \not\to \mathcal{RW}$

Proof. For the first direction: The proof of Proposition 6, showing that $\mathcal{RWB} \not\to \mathcal{WB}$, used an $\mathcal{RWB}$-automaton that did not utilize the blocking idiom; hence, the same proof applies here.

For the other direction, we use the proof of Proposition 8, showing that $\mathcal{RWB} \not\to \mathcal{RW}$. The original proof was based on an automaton in which all threads requested all the events at every state, except those that they blocked. Hence, moving them to the $\mathcal{WB}$ model would produce the same language. ◀

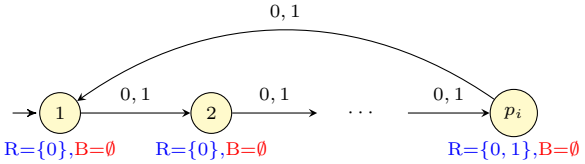
▶ Proposition III. $\mathcal{RB} \not\to \mathcal{RW}$ and $\mathcal{RW} \not\to \mathcal{RB}$

Proof. For the first direction: The proof of Proposition 8, showing that $\mathcal{RWB} \not\to \mathcal{RW}$, used an $\mathcal{RWB}$-automaton that did not wait for any events (i.e., the state-changes of each thread were caused solely by triggering events that the thread itself had requested). Hence, the same proof applies here.

We now prove that $\mathcal{RW} \not\to \mathcal{RB}$. Let $n \in \mathbb{N}$, and let $k \in \mathbb{N}$ be the smallest number such that the first k prime numbers $p_1, \ldots, p_k$ satisfy $\prod_{i=1}^{k} p_i > n$. Define the family of languages $L_n = \{\ell_1 \ell_2 \ldots\}$ by:

$$\ell_j = \begin{cases} 0 \text{ or } 1 & ; \quad \exists i \text{ such that } p_i \mid j \\ 0 & ; \quad \text{otherwise} \end{cases}$$

In the $\mathcal{RW}$ model, this language can be accepted by an automaton with k threads, $T_1, \ldots, T_k$. Thread T_i corresponds to prime p_i, and is given by:



Here, in every state the thread requests 0, and in state p_i (and only in that state) it also requests 1. Observe that 0 is always requested, and that 1 is requested if and only if the index is divisible by one of the primes. In the $\mathcal{RW}$ model there is no blocking, and so requested events can always be triggered — generating the desired language. As in previous proofs, the size of the automaton is $O(\log^2 n \cdot \log\log n)$.

Now, observe what happens in the $\mathcal{RB}$ model. Suppose towards contradiction that an $\mathcal{RB}$-automaton accepts the language L_n, and that all its threads have less than n states. Denote $N = \prod_{i=1}^{k} p_i$. We will show that there exists a sequence of N consecutive integers, $M, M+1, \ldots, M+N-1$, such that for every $M \leq i \leq M+N-1$, there exists a run of the automaton in which 1 is triggered at index i. This will form a contradiction to the definition of L_n, proving the claim. The rest of the proof shows how to construct these runs.

Observe the $\mathcal{RB}$-automaton as it reads the finite sequence 0^n. By the pigeonhole principle, this sequence causes all threads to traverse cycles where all the edges are labeled 0. (0-cycles). Note that this is true regardless of the starting configuration of the automaton — i.e., at any point during the run, reading a 0^n sequence causes all threads to traverse 0-cycles.

We begin with the following observation regarding 0-cycles:

▶ **Observation 1.** When all threads are traversing 0-cycles, 1 cannot be blocked in any of the cycles' states.

Proof. Observe some thread T_j and the 0-cycle it is traversing. Let α denote the length of that cycle. Since T_j has less than n states, and $\prod_{i=1}^{k} p_i > n$, it follows that there exists some i such that α and p_i are coprime. Suppose the remainder of the input word is just an infinite sequence of zeroes, 0^ω. By the Chinese remainder theorem, if 1 was blocked in any of the states of the 0-cycle, it would eventually get blocked at an index divisible by p_i — and the automaton would reject a word it is supposed to accept, which is a contradiction. Hence, 1 is never blocked in any of the 0-cycles. ◀

Clearly, when all threads are traversing their 0-cycles, 1 has to be requested infinitely often. This means that at least one thread has, within its cycle, a state that requests 1. We now make the following observation:

▶ **Observation 2.** Suppose all threads are traversing 0-cycles, and let T be one of the threads that request 1 infinitely often. Then eventually 1 will be requested by another thread when T is at a state in which it does not request 1.

Proof. Suppose towards contradiction that this is not the case; i.e. that T requests 1 whenever the index is divisible by one of the primes. Denote T's cycle length by α. Because T has less than n states, there is some i such that α and p_i are coprime. By the Chinese remainder theorem, if the stream of 0s continues, every state in T's cycle will eventually be reached when p_i divides the run index. Hence, since T must request 1 whenever the index is divisible by p_i, *every* state in its cycle must request 1. Consequently, as 1 is never blocked within the 0-cycles (Observation 1) there exists a finite index ℓ such that for every $\ell' > \ell$ the word $0^{\ell'}1$ is a prefix of a word accepted by the automaton — which is a contradiction. Hence, there must be a time in which 1 is requested, but T does not request it. ◀

We now use these two observations in order to construct the aforementioned N runs.

The first run we examine, denoted ρ_0, is the one associated with input word 0^ω. All threads eventually traverse 0-cycles. Observe a fixed thread T that requests 1 infinitely

often. Let β denote the first index, after entering its cycle, in which T requests 1, and let α denote its cycle length. Then T will request 1 at all indices $\beta + \alpha \cdot t$, for all $t \in \mathbb{N}$.

The second run, ρ_1, is constructed as follows. The input word starts with 0^n, in order to have all threads traverse 0-cycles. We now apply Observation 2: we "feed" the automaton more 0s, until reaching an index in which 1 is requested by some thread and not by T. We then trigger 1, and afterwards continue with 0^ω. Since there is no waiting in the $\mathcal{RB}$ model, and since T did not request the triggered 1, T is oblivious to the fact that 1 has been triggered. For the remainder of the run, the thread T is *delayed* by one index position. This implies that there is some constant T_1, such that, in run ρ_1, for every $t > T_1$, thread T requests 1s at indices $\beta + \alpha \cdot t + 1$.

We continue iteratively. Run ρ_2 is produced as follows. Up to the 1 triggered in ρ_1 the prefixes of the two runs are identical. We then feed more 0s, trigger 1 again at an index divisible by a prime when T does not request 1, and continue with 0^ω. This results in a constant T_2, such that in run ρ_2 for every $t > T_2$, thread T requests 1s at indices $\beta + \alpha \cdot t + 2$.

The process is repeated N times, where each ρ_k is identical to ρ_{k-1} up to the last 1, and a 1 is then added where T does not request it. We get a series of constants $T_{N-1} > T_{N-2} > \ldots > T_2 > T_1 > 0$, such that for every $t > T_i$, there exists a run in which T requests 1 at indices $\beta + \alpha \cdot t + i$, and 1 is not blocked.

Set $t = T_{N-1} + 1$, so that it is larger than all the constants $T_1, \ldots, T_{N-1}$ above. For every $1 \leq i \leq N$, there is some run of the automaton, in which thread T requests 1 at index $\beta + \alpha \cdot t + i$, and 1 is not blocked at that index. This implies that there are N consecutive integers that are each divisible by at least one of the primes $p_1, \ldots, p_n$, which is a contradiction. It follows that our initial assumption is false; i.e. that there is some thread with at least n states, proving the gap in succinctness as needed. ◀

Note that the above proof that $\mathcal{RW} \not\to \mathcal{RB}$ also constitutes a stronger proof for Proposition 7, i.e. that $\mathcal{RWB} \not\to \mathcal{RB}$.

From Propositions I, II and III we also obtain the following corollary:

▶ **Corollary 1.** *$\mathcal{WB} \not\to DLA$, $\mathcal{RB} \not\to DLA$ and $\mathcal{RW} \not\to DLA$.*

Proof. By using Propositions I, II and III, we get that for every $\mathcal{M}_1 \in \{\mathcal{WB}, \mathcal{RB}, \mathcal{RW}\}$ there exists a $\mathcal{M}_2 \in \{\mathcal{WB}, \mathcal{RB}, \mathcal{RW}\}$ such that $\mathcal{M}_1 \neq \mathcal{M}_2$ and that $\mathcal{M}_1 \not\to \mathcal{M}_2$. Further, the DFA model is directly embeddable in model $\mathcal{M}_2$ (see explanation in Appendix B). The claim follows. ◀

References

1 D. Drusinsky and D. Harel. On the Power of Bounded Concurrency I: Finite Automata. *J. Assoc. Comput. Mach.*, 41:517–539, 1994.