A Additional Material for Section 3: Concurrent Programs

Every statement $a \in \Sigma$ for a concurrent program P has as semantics $[\![a]\!]$ a binary relation over *program states*, i.e., valuations of the variables occurring in P. We extend the semantics to traces via relational composition. Validity of Hoare triples $\{\varphi\}\ \tau\ \{\psi\}$, where φ and ψ are first-order assertions over the program variables and τ is a trace, is defined as usual.

B Additional Material for Section 4: Reductions

As explained in 7, our final approach is based on an extension of Mazurkiewicz equivalence, where the commutativity relation is *conditional* (parametrized in the state of an automaton). In this appendix, we thus introduce this more general form now, and use it to formulate and prove slightly more general versions of our results. Unless otherwise noted, all definitions and results in this appendix also hold for conditional commutativity.

Let $A = (Q, \Sigma, \delta, q_{\text{init}}, F)$ be any DFA over alphabet Σ. A *conditional commutativity relation* is a mapping from each state q of A to a symmetric relation $\leftrightarrows_q \subseteq \Sigma \times \Sigma$. From such a commutativity relation on letters, we define the *(conditional) Mazurkiewicz equivalence* relation $\sim \subseteq \Sigma^* \times \Sigma^*$ as the smallest relation satisfying

1. For all $w \in \Sigma^*$, we have that $w \sim w$.
2. For all $w_1, w_2, w_3 \in \Sigma^*$ such that $w_1 \sim w_2$ and $w_2 \sim w_3$, then also $w_1 \sim w_2$.
3. For all $u, v \in \Sigma^*$ and $a, b \in \Sigma$, if either $\delta^*(q_{\text{init}}, u)$ is undefined or $a \leftrightarrows_{\delta^*(q_{\text{init}}, u)} b$, then we have $uabv \sim ubav$.

If we have $w \sim v$, we say that the word w *is equivalent to* the word v. Let $L \subseteq \Sigma^*$ be a language over our alphabet Σ. We define the *closure* of L as

$$cl(L) := \{\, w \in \Sigma^* \mid \exists v \in L \,.\, w \sim v \,\} \text{ or equivalently, } L = \bigcup_{w \in L} [w]$$

Observe that this operation is indeed a closure operation, i.e., it is extensive, monotonic and idempotent. We say that L is closed iff $L = cl(L)$.

We call L' a *reduction* of L iff $L' \subseteq L$, and we have $\forall w \in L \,.\, \exists v \in L' \,.\, w \sim v$. If L is closed, then this is equivalent to $cl(L') = L$.

A reduction L' of L is called *(language-) minimal* iff no strict subset of L' is a reduction of L. This corresponds to the fact that L' contains exactly one representative of each $\sim$-equivalence class present in L. For languages $L_1, L_2 \subseteq \Sigma^*$, the inclusion $L_1 \subseteq cl(L_2)$ is equivalent to the existence of a reduction L_1' of L_1 such that $L_1' \subseteq L_2$.

Here in the appendix we present a more general version of the reduction induced by a preference order, which is not specific to lexicographic orders. In general, preference orders may not be total, and thus an equivalence class might have multiple minimal elements. Further, we do not assume closedness of the language L.

Definition 4.1 (Reduction induced by Preference Order)**.** *Let L be a language, and let $\preceq$ be a preference order. The reduction of L induced by $\preceq$ is defined as:*

$$red_{\preceq}(L) := \bigcup_{w \in L} \min_{\preceq}\Big([w] \cap L\Big)$$

General (not necessarily lexicographic) preference orders fully characterize reductions:

Lemma B.1 (Preference Order Characterization)**.** *Let $L, L' \subseteq \Sigma^*$ be languages. Then L' is a reduction of L iff there exists a preference order $\preceq$ such that $L' = red_{\preceq}(L)$.*

Proof. We begin by showing that $red_{\preceq}(L)$ is a reduction. For each $w \in L$, the upwards closure $[w]$ (in case of symmetry, this corresponds to the equivalence class of w) is finite. Hence the set $[w] \cap L$ is also finite, and as it contains at least w, it is also nonempty. Hence it must contain at least one $\preceq$-minimal element v. Then $v \in red_{\preceq}(L)$ is a representative for w. As each word $w \in L$ has a representative in $red_{\preceq}(L)$, this language is a reduction of L.

Vice versa, for a given reduction L' consider the preference order $\preceq_{L'} := L' \times (\Sigma^* \setminus L') \cup id_{\Sigma^*}$. It is easy to see that this relation is reflexive, transitive and antisymmetric:

Reflexivity By definition, $id_{\Sigma^*} \subseteq \preceq_{L'}$.

Transitivity Let $u \preceq_{L'} v$ and $v \preceq_{L'} w$ for words $u, v, w \in \Sigma^*$. By definition of $\preceq_{L'}$, there are two cases: In the case that $(u, v) \in id_{\Sigma^*}$, i.e., $u = v$, it immediately follows from $v \preceq_{L'} w$ that $u \preceq_{L'} w$. In the case that $u \in L'$ and $v \notin L'$, we conclude from $v \preceq_{L'} w$ that indeed $v = w$, and hence $u \preceq_{L'} w$ follows.

Antisymmetry Let $u \preceq_{L'} v$ and $v \preceq_{L'} u$ for words $u, v \in \Sigma^*$. In the cases where $(u, v) \in id_{\Sigma^*}$ or $(v, u) \in id_{\Sigma^*}$, the conclusion $u = v$ is immediate. Hence we only need to consider the case that $u \in L$, $v \notin L$, $v \in L$ and $u \notin L$, and immediately arrive at a contradiction.

Hence $\preceq_{L'}$ is a preference order. Further, we observe that $L' = red_{\preceq_{L'}}(L)$: Any word $w \in L'$ is necessarily minimal in its equivalence class, as there exists no word in Σ^* that is strictly less than w. Hence it follows that $w \in red_{\preceq_{L'}}(L)$. Conversely, for any word $w \notin L'$, by the fact that L' is a reduction we know that there exists some word $v \in L'$ such that $w \sim v$. By construction, it is immediate that $v \preceq_{L'} w$. Hence w is not minimal in its equivalence class, and we conclude that $w \notin red_{\preceq_{L'}}(L)$. □

Definition 4.1 is fully general, yet interesting classes of reductions can be characterized by adding constraints about the preference order:

Observation B.2. *Let $L \subseteq \Sigma^*$ be a language, and let $\preceq$ be a* total *preference order. Then it follows that $red_{\preceq}(L)$ is a language-minimal reduction of L, i.e., no strict subset of $red_{\preceq}(L)$ is a reduction of L.*

Proof. This is straightforward: For a total order, every equivalence class has exactly one minimal element. Hence no element of the reduction can be removed. □

Next, we present a generalization of the *(upwards-)closedness* notion specific to a given preference order. This generalization is sufficient to prove our results, and we make use of it in the proofs of section 6.2.

Definition B.3 ($\preceq$-Closedness)**.** *Let $\preceq$ be a preference order. A language $L \subseteq \Sigma^*$ is $\preceq$-closed iff it holds that*

$$\forall w, v \in \Sigma^* . w \in L \wedge w \sim v \wedge v \preceq w \implies v \in L$$

Note that this is truly a strict generalization of the classic notion of closedness: If L is closed, then L is $\preceq$-closed for every preference order $\preceq$.

Let us make two more observations about $\preceq$-closedness that play an important role in section 6.2 in the soundness proof of our reduction algorithm. First, observe that the reduction induced by a preference order $\preceq$ preserves $\preceq$-closedness:

Lemma B.4. *If L is $\preceq$-closed, then $red_{\preceq}(L)$ is still $\preceq$-closed.*

Proof. Let $w, v \in \Sigma^*$ such that $w \in red_{\preceq}(L)$, $w \sim v$ and $v \preceq w$. By the definition of $red_{\preceq}(L)$, it follows that $w \in \min_{\preceq}\left([w] \cap L\right) \subseteq L$. From $\preceq$-closedness of L we conclude that $v \in L$, and specifically $v \in [w] \cap L$. But minimality of w and $v \preceq w$ imply that $w = v$, and hence $v \in red_{\preceq}(L)$. □

This is a significant difference from general closedness: Even if L is closed, the reduction $red_{\preceq}(L)$ is generally not closed.

Next, note that the notion of closedness wrt. a preference order is monotone in the underlying commutativity relation, as well as in the preference order:

Lemma B.5. *Let $\preceq_1, \preceq_2 \subseteq \Sigma^* \times \Sigma^*$ be preference orders, such that $\preceq_1 \subseteq \preceq_2$. Furthermore, let $\mathbin{\circlearrowright}^1_\cdot, \mathbin{\circlearrowright}^2_\cdot$ be (conditional) symmetric commutativity relations, such that for all q, $\mathbin{\circlearrowright}^1_q \subseteq \mathbin{\circlearrowright}^2_q$; and let $\sim_i$ for $i \in \{1, 2\}$ be the equivalence relation corresponding to $\mathbin{\circlearrowright}^i_\cdot$.*

Then we have that $\sim_1 \subseteq \sim_2$, and it follows that, if L is $\preceq_2$–closed wrt. $\sim_2$, L is also $\preceq_1$–closed wrt. $\sim_1$.

Proof. Assume that L is $\preceq_2$–closed wrt. $\sim_2$. Then let $w, v \in \Sigma^*$ such that $w \in L$, $w \sim_1 v$ and $v \preceq_1 w$. By assumption, it follows that $w \sim_2 v$ and $v \preceq_2 w$. Since L is $\preceq_2$–closed wrt. $\sim_2$, we conclude that $v \in L$. □

The assumption of $\preceq$-closedness, or in particular the classic notion of closedness, allows to simplify the definition of the reduction induced by a preference order, clarifying the role of the preference order in choosing representatives:

$$\begin{aligned} red_{\preceq}(L) &= \bigcup_{w \in L} \min_{\preceq}\left([w] \cap L\right) \\ &= \bigcup_{w \in L} \min_{\preceq}[w] \\ &= L \cap \left(\bigcup_{w \in \Sigma^*} \min_{\preceq}[w] \right) \\ &= L \cap red_{\preceq}(\Sigma^*) \end{aligned}$$

Thus we arrive at the observation stated in section 4:

Observation B.6. *If L is $\preceq$-closed, then the choice of representatives for each class is independent of L, and we have*

$$red_{\preceq}(L) = L \cap red_{\preceq}(\Sigma^*)$$

The results discussed in this appendix so far hold for any preference order, whether it is a lexicographic order or not. Some of these results will indeed be applied for another preference order in proofs in subsequent sections.

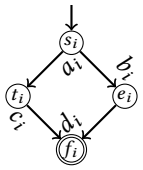
We now turn to the case of lexicographic preference orders and investigate their space complexity, i.e., the minimal number of states any DFA recognizing the reduction language must have. First, let us identify a subclass of lexicographic orders: We say that $\preceq$ is *thread-uniform*, if the underlying letter ordering treats all letters of a thread the same:

$$i \neq j,\ (\exists a \in \Sigma_i, b \in \Sigma_j \,.\, a < b) \implies \forall a \in \Sigma_i, b \in \Sigma_j \,.\, a < b$$

In other words, there are cases where less commutativity induces a smaller recognizer for a lexicographic reduction. In general, even full commutativity does not prevent exponential explosion:

Observation B.7. *There exists a concurrent program P and a lexicographic preference order $\preceq$, such that, even under full commutativity, the state complexity of the corresponding reduction $red_{\preceq}(\mathcal{L}(P))$ is exponential in size(P).*

Proof. Let each thread T_i (for $i = 1, \ldots, n$) be given by a single if/then/else-statement, with a control flow automaton as below:



Then let $P = T_1 \parallel \ldots \parallel T_n$ be the concurrent program in question. Let the letter order be given as $a_1 < b_1 < \ldots < a_n < b_n < c_1 < d_1 < \ldots < c_n < d_n$.

We denote the Nerode equivalence between words wr.t. the language $red_{\preceq}(\mathcal{L}(A))$ by $\equiv$, i.e., the index of $\equiv$ is the state complexity of $red_{\preceq}(\mathcal{L}(A))$. Then the 2^n words of the form $x_1 x_2 \ldots x_n$, where $x_i \in \{a_i, b_i\}$, are all pairwise non-equivalent w.r.t. $\equiv$. To see this, observe that the concatenation with the word $y_1 \ldots y_n$, where

$$y_i = \begin{cases} c_i & \textbf{if } x_i = a_i \\ d_i & \textbf{else} \end{cases}$$

is in the reduction $red_{\preceq}(\mathcal{L}(A))$: Firstly, it is straightforward to see that the word $w := x_1 \ldots x_n y_1 \ldots y_n$ is accepted by P. Secondly, the word w is sorted w.r.t. to the preference order and hence lexicographically minimal among all its permutations, including all equivalent words. However, for any other word $x'_1 x'_2 \ldots x'_n$, the concatenation $x'_1 x'_2 \ldots x'_n y_1 \ldots y_n$ is not in $\mathcal{L}(P)$, and thus also not in $red_{\preceq}(\mathcal{L}(P))$.

Hence, a minimal DFA for $red_{\preceq}(\mathcal{L}(P))$ must have at least 2^n states. □

In particular, if the lexicographic preference order not thread-uniform, branching in the control flow of the threads can lead to such an explosion. The DFA for the reduction, just like the full interleaving product, may have to keep track of all combinations of branches taken by the different threads, resulting in an exponential number of reachable states. The fact that this explosion is really due to branching is illustrated by the observation that, for straight-line thread programs, the reduction always has linear complexity, even for positional, not thread-uniform orders (as it contains only a single word). However, as the slightly more precise version of the following theorem shows, thread-uniform lexicographic preference orders do not suffer from exponential explosion (under full commutativity).

Theorem 4.2. *Let $\preceq$ be a thread-uniform lexicographic preference order. Under full commutativity, the state complexity of the induced reduction $red_{\preceq}(\mathcal{L}(P))$ is linear in the program size size(P).*

Proof. In section 6, we give a construction of a DFA for $red_{lex(\preccurlyeq)}(\mathcal{L}(P))$ and in theorem 7.2 we prove that, under the above assumptions, the constructed DFA has only $O(size(P))$ states. □

We note the following about the reduction in figure 1 and the claim in example 4.3:

Observation B.8. *No lexicographical preference order can induce the reduction in figure 1.*

Proof. Consider the word $w := a_1 a_2 b_1 b_2 a_1 a_2 b_1 b_2 c_1 c_2$, i.e., both threads loop once then exit. Since this word is in the reduction (it is accepted by the DFA in figure 1), it must be minimal within its equivalence class. We conclude for the underlying strict order on letters:

- We must have $a_1 < a_2$. Otherwise, the equivalent word $a_2 a_1 b_1 b_2 a_1 a_2 b_1 b_2 c_1 c_2$ is preferable to w.
- We must have $a_2 < b_1$. Otherwise, the equivalent word $a_1 b_1 a_2 b_2 a_1 a_2 b_1 b_2 c_1 c_2$ is preferable to w.

- We must have $b_1 < b_2$. Otherwise, the equivalent word $a_1a_2b_2b_1a_1a_2b_1b_2c_1c_2$ is preferable to w.
- We must have $b_2 < a_1$. Otherwise, the equivalent word $a_1a_2b_1a_1b_2a_2b_1b_2c_1c_2$ is preferable to w.

As a strict order is acyclic, this is a contradiction. □

We split theorem 4.6 on regularity and language-minimality in two results here:

Observation B.9. *The lexicographic reduction $red_{lex(\preccurlyeq)}(L)$ of a language L is language-minimal, i.e., no proper subset of $red_{lex(\preccurlyeq)}(L)$ is a reduction of L.*

Proof. This follows directly from observation B.2 and the fact that positional lexicographic preference orders are total. □

The regularity result is only proven for the more limited setting discussed in the main sections of this paper, i.e., non-conditional commutativity. Our algorithmic methods discussed later will, for the more general setting, only over-approximate this language.

Lemma B.10. *Assume that $\mathcal{L}(A)$ is $lex(\preccurlyeq)$-closed, and the commutativity relation is unconditional. The lexicographic reduction $red_{lex(\preccurlyeq)}(\mathcal{L}(A))$ is regular.*

Proof. We give a construction of a DFA for the reduction in section 5. Theorem 5.3 proves that this construction indeed recognizes the language $red_{lex(\preccurlyeq)}(\mathcal{L}(A))$. □

Theorem 4.6. *Assume that $\mathcal{L}(A)$ is $lex(\preccurlyeq)$-closed, and the commutativity relation is unconditional. The lexicographic reduction $red_{lex(\preccurlyeq)}(\mathcal{L}(A))$ is regular and language-minimal.*

Proof. Proven in observation B.9 and lemma B.10. □

For positional lexicographic preference orders, even thread-uniformity cannot prevent exponential explosion under full commutativity.

Observation B.11. *There exists a concurrent program P and a thread-uniform P-positional lexicographic order $lex(\preccurlyeq)$, such that, under full commutativity, the state complexity of the corresponding lexicographic reduction $red_{lex(\preccurlyeq)}(\mathcal{L}(P))$ is exponential in $size(P)$.*

Proof. Consider again the concurrent program P seen in the proof of theorem B.7. We define for each $k \in \{1, \dots, n\}$ the relation $<_k$ on $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$ as the total order such that

- we have $a_i <_k b_i <_k c_i <_k d_i$ for all $i \in \{1, \dots, n\}$;
- for i, j such that either $i, j \in \{1, \dots, k-1\}$ or $i, j \in \{k, \dots, n\}$, for all $x \in \Sigma_i$ and $y \in \Sigma_j$ we have that $x <_k y$ iff $i < j$;
- and for all $i \in \{k, \dots, n\}$, $j \in \{1, \dots, k-1\}$, $x \in \Sigma_i$ and $y \in \Sigma_j$, we have that $x < y$.

Intuitively, we fix an ordering *within* each thread (for sake of totality), and for increasing k we rotate the order *between* threads: For $k = 1$, we order threads by their index (T_1 first, T_n last); for $k = 2$ we shift T_1 to the end; and so on. We then associate with each state q of P one such total order:

- For a state $q = \langle \ell_1, \dots, \ell_{k-1}, s_k, \dots, s_n \rangle$ where $k \in \{1, \dots, n\}$ and $\ell_i \in \{t_i, e_i\}$ for all $i \in \{1, \dots, k-1\}$, we choose the order $<_q := <_k$.
- For a state $q = \langle f_1, \dots, f_{k-1}, \ell_k, \dots, \ell_n \rangle$ where $k \in \{1, \dots, n\}$ and $\ell_i \in \{t_i, e_i\}$ for all $i \in \{k, \dots, n\}$, we choose the order $<_q := <_k$.
- For all other states q, we choose the order $<_q := <_1$.

Once again, we consider the 2^n words of the form $x_1 \dots x_n$ with $x_i \in \{a_i, b_i\}$ and show that they are pairwise non-equivalent w.r.t. the Nerode equivalence $\equiv$ induced by the language $red_{lex(\preccurlyeq)}(\mathcal{L}(P))$. For one such word $x_1 \dots x_n$ we define the word $y_1 \dots y_n$ as above and once again observe that the concatenation $w := x_1 \dots x_n y_1 \dots y_n$ is accepted by P. Further, w is lexicographically minimal within its equivalence class. To see this, let us assume, for purposes of contradiction, some word $v = v_1 \dots v_{2n}$ with $v \neq w$, $w \sim v$ and $(v, w) \in lex(\preccurlyeq)$. Let u be the longest common prefix of w and v, and let $a, b \in \Sigma$, $w', v' \in \Sigma^*$ such that $w = uaw'$ and $v = ubv'$. We show that $a <_q b$, where $q = \delta^*(q_{\text{init}}, u)$. To this end, let us distinguish two cases:

1. If $|u| < n$, then $a = x_k$ for some $k \in \{1, \dots, n\}$. Observe that q has the form $\langle \ell_1 \dots, \ell_{k-1}, s_k, \dots, s_n \rangle$ where $\ell_i \in \{t_i, e_i\}$ for all $i \in \{1, \dots, k-1\}$. By definition of the preference order, we thus have that $<_q = <_k$. We distinguish two sub-cases:
   a. In the case that $b = x_j \in \Sigma_j$ for some $j > k$, and noting that $a = x_k \in \Sigma_k$ and both $k, j \in \{k, \dots, n\}$, we conclude from the definition of $<_k$ that $a <_q b$.

b. In the case that $b = y_j$ for some $j \in \{1, \dots, n\}$, we first observe that necessarily $j < k$: Were this not the case, then v would differ in the ordering between x_j and y_j, but since both these statements belong to the same thread and cannot commute, this would contradict our assumption that $w \sim v$. Hence we now know that $j \in \{1, \dots, k-1\}$ and conclude again from the definition of $<_k$ that $a <_q b$.

2. If $|u| \geq n$, then $a = y_k$ and $b = y_j$ for some $k \in \{1, \dots, n\}$ and $j > k$. Observe that q has the form $\langle f_1 \dots, f_{k-1}, \ell_k, \dots, \ell_n \rangle$ where $\ell_i \in \{t_i, e_i\}$ for all $i \in \{k, \dots, n\}$. By definition of the preference order, we thus have that $<_q = <_k$. We note that $a = y_k \in \Sigma_k$, $b = y_j \in \Sigma_j$ and both $k, j \in \{k, \dots, n\}$ and hence conclude that $a <_q b$.

Therefore, we know that $w \in red_{lex(\lessdot)}(\mathcal{L}(P))$. On the other hand, for any other word $x'_1 \dots x'_n$ with $x'_i \in \{a_i, b_i\}$, the concatenation $x'_1 \dots x'_n y_1 \dots y_n$ is not accepted by P and hence not in $red_{lex(\lessdot)}(\mathcal{L}(P))$. Thus, $\equiv$ has at least 2^n equivalence classes, and hence, a minimal DFA for $red_{lex(\lessdot)}(\mathcal{L}(P))$ must have at least 2^n states. □

C Additional Material for Section 5: Finite Representations

The definition of the sleep set automaton here differs only in the fact that conditional commutativity is used in the definition of the updated sleep set S'.

Definition 5.1 (Sleep Set Automaton). *We define the sleep set automaton* $\mathfrak{S}_{\lessdot}(A) := (Q \times 2^{\Sigma}, \Sigma, \delta_{\mathfrak{S}}, \langle q_{\text{init}}, \emptyset \rangle, F \times 2^{\Sigma})$, *where*

$$\delta_{\mathfrak{S}}(\langle q, S \rangle, a) := \begin{cases} \textit{undefined} & \textbf{\textit{if}}\ a \in S\ \textit{or}\ \delta(q,a)\ \textit{undefined} \\ \langle \delta(q,a), S' \rangle & \textbf{\textit{else}} \end{cases}$$

with $S' = \{\, b \in \mathit{enabled}(q) \mid (b \in S \vee b <_q a) \wedge a \leftrightarrows_q b \,\}$.

We prove the correctness of the sleep set automaton in two lemmata. First however, we define some notation used in these lemmata. First, we define a variant of the lexicographic preference order $lex(\lessdot)$ parametrized in a state q, namely we let $lex_q(\lessdot)$ be the smallest relation such that

- for all words w, v we have $(w, wv) \in lex_q(\lessdot)$;
- and for all words u, v, w and all letters a, b such that from state q, by reading u we reach or get stuck in state q' and $a <_{q'} b$, we have $(uav, ubw) \in lex_q(\lessdot)$.

Note that in particular $lex(\lessdot) = lex_{q_{\text{init}}}(\lessdot)$, and that for words u, v, w, if $q' = \delta^*(q, u)$ and $(v, w) \in lex_{q'}(\lessdot)$, it follows that $(uv, uw) \in lex_q(\lessdot)$.

Second, we define analogously a variant of the conditional Mazurkiewicz equivalence parametrized in q: Let $\sim_q$ be the least reflexive-transitive relation such that for all $u, v \in \Sigma^*$ and $a, b \in \Sigma$, if either $\delta^*(q, u)$ is undefined or $a \leftrightarrows_{\delta^*(q,u)} b$, then we have $uabv \sim_q ubav$. Similar to above, we have that $\sim = \sim_{q_{\text{init}}}$ and that for $q' = \delta^*(q, u)$, $v \sim_{q'} w$ implies $uv \sim_q uw$. By $cl_q(\cdot)$ we denote the closure up to this equivalence relation, and similarly $red^q_{lex_q(\lessdot)}(\cdot)$ refers to the reduction induced by the equivalence $\sim_q$.

We can now state the first invariant we need to prove about all states of the sleep set automaton. It is essentially a local variant of the result that the sleep set automaton recognizes a superset of the lexicographic reduction.

Lemma C.1. *For all* $w \in \Sigma^*$, *for all states* q *of* A *and all* $S \subseteq \Sigma$,

$$w \in red^q_{lex_q(\lessdot)}(\mathcal{L}_A(q)) \implies w \in \mathcal{L}(\langle q, S \rangle) \cup cl_q(S \cdot \Sigma^*)$$

Proof. We proceed by induction over the length of a word $w \in red^q_{lex_q(\lessdot)}(\mathcal{L}_A(q))^*$. The induction start is simple: If $w = \varepsilon$, then we must have $q \in F$, and thereby $\varepsilon \in \mathcal{L}(\langle q, S \rangle)$. For the induction step, let $w = av$ for some $a \in \Sigma$ and $v \in \Sigma^*$. If $a \in S$, then it immediately follows that $av \in S \cdot \Sigma^* \subseteq cl_q(S \cdot \Sigma^*)$, and we are done. Consider however the case where $a \notin S$, and therefore $\delta_{\mathfrak{S}_{\lessdot}(P)}(\langle q, S \rangle, a) = \langle q', S' \rangle$, where $q' := \delta_A(q, a)$, and S' is defined as in definition 5.1. We know that q' exists, because $av \in \mathcal{L}_A(q)$; and furthermore, we know that $v \in \mathcal{L}_A(q')$. Then it follows that $v \in red^{q'}_{lex_{q'}(\lessdot)}(\mathcal{L}_A(q'))$: If this were not the case, i.e., v not were minimal within its $\sim_{q'}$-equivalence class, there would have to exist some $v' \in \mathcal{L}_A(q')$ such that $v \sim_{q'} v'$ and, by totality, $(v', v) \in lex_{q'}(\lessdot)$. But from this it would follow that $av' \in \mathcal{L}_A(q)$, $av \sim_q av'$ and $(av', av) \in lex_q(\lessdot)$; this would contradict our assumption that $av \in red^q_{lex_q(\lessdot)}(\mathcal{L}_A(q))$.

Now, our induction hypothesis allows us to conclude that $v \in \mathcal{L}(\langle q', S' \rangle) \cup cl_{q'}(S' \cdot \Sigma^*)$. If specifically $v \in \mathcal{L}(\langle q', S' \rangle)$, we can indeed conclude that $w = av \in \mathcal{L}(\langle q, S \rangle)$, and we are done.

On the other hand, let us investigate the case that $v \in cl_{q'}(S' \cdot \Sigma^*)$, i.e, $v \sim_{q'} bx$ for some $b \in S'$ and $x \in \Sigma^*$. Recall that by definition, S' contains only letters that commute with a, and that are either already in S, or less than a, i.e.,

$$S' = \{\, b \in \mathit{enabled}(q) \mid (b \in S \vee b <_q a) \wedge a \leftrightarrows_q b \,\}$$

It follows that $w = av \sim_q abx \sim_q bax$. Hence, the case that $b <_q a$ yields an immediate contradiction to our assumption that $w = av \in red^q_{lex_q(\lessdot)}(\mathcal{L}_A(q))$, as there exists a strictly smaller equivalent word bax. If $b \in S$, we can conclude that $w \in cl_q(S \cdot \Sigma^*)$, as $w \sim_q bax \in S \cdot \Sigma^*$. □

The second result shows the converse of the above: A word accepted by the sleep set automaton (starting from a state q) is indeed in the lexicographic reduction. However, this only holds for unconditional commutativity.

Lemma C.2. *Assume that $\leftrightarrows$ is unconditional. For all $w \in \Sigma^*$, for all states q of A and all $S \subseteq \Sigma$,*

$$w \in \mathcal{L}(\langle q, S\rangle) \implies w \in red_{lex_q(\lessdot)}(\mathcal{L}_A(q)) \setminus cl(S \cdot \Sigma^*)$$

Proof. We proceed by induction over the length of a word $w \in \mathcal{L}(\langle q, S\rangle)$. The case $w = \varepsilon$ again reduces both sides to $q \in F$.

For the induction step, let $w = av$, and let $\delta_{\Xi_\lessdot(A)}(\langle q, S\rangle, a) = \langle q', S'\rangle$. Then we know by induction hypothesis that, since $v \in \mathcal{L}(\langle q', S'\rangle)$, we can conclude $v \in red_{lex_{q'}(\lessdot)}(\mathcal{L}_A(q')) \setminus cl(S \cdot \Sigma^*)$.

Let us first show that av is not equivalent to any word beginning with a letter in S. Assume that such a word cu with $c \in S$ and $av \sim cu$ existed. Clearly, $a \notin S$, otherwise we would have no transition from state $\langle q, S\rangle$. But then it follows that $a \neq c$ and indeed $a \leftrightarrows c$. From this we conclude that $c \in S'$. There exists some word u' such that $v \sim cu'$ (one commutation before c reaches the beginning of the word cu, we must have dcu' for some letter d). This contradicts the induction hypothesis that $v \notin cl(S' \cdot \Sigma^*)$.

To show that $av \in red_{lex_q(\lessdot)}(\mathcal{L}_A(q))$, let $b \in \Sigma$, $x \in \Sigma^*$ such that $av \sim bx$ and $(bx, av) \in lex_q(\lessdot)$; we must now show that $bx = av$ to prove minimality of av within its equivalence class. We distinguish two cases: If $a = b$, we conclude from $(ax, av) \in lex_q(\lessdot)$ that also $(x, v) \in lex_{q'}(\lessdot)$, and by minimality of v, this implies $x = v$. Hence we have shown $bx = ax = av$, and we are done with minimality.

In the case where $a \neq b$, we conclude from $(bx, av) \in lex_q(\lessdot)$ that $b <_q a$. Furthermore, since bx and av differ in their ordering of the letters a and b, but we have $av \sim bx$, we know that these letters commute, i.e., $a \leftrightarrows b$. Then, we must have $b \in S'$. There exists some word u such that $v \sim bu$ (one commutation before b reaches the beginning of the word bx, we must have cbu for some letter c). But this contradicts the induction hypothesis, specifically the part that $v \notin cl(S' \cdot \Sigma^*)$. □

Theorem 5.3. *Assume that $\leftrightarrows$ is unconditional. The sleep set automaton $\Xi_\lessdot(A)$ recognizes exactly the lexicographic reduction $red_{lex(\lessdot)}(\mathcal{L}(A))$ of $\mathcal{L}(A)$.*

Proof. We apply lemma C.1 and lemma C.2 to the initial state $\langle q_{\text{init}}, \emptyset\rangle$. Noting that $cl(\emptyset \cdot \Sigma^*) = cl(\emptyset) = \emptyset$, we arrive at our conclusion. □

The above theorem, and in particular lemma C.2, does not hold in case of conditional commutativity: The sleep set automaton recognizes only an over-approximation of the lexicographic reduction (lemma C.1). As an example, consider the case where a, b commute in the initial state, and c, d commute in the state reached by ba, but c, d do not commute in the state reached by ab. Then $abcd \sim bacd \sim badc \sim abdc$, but the reduction automaton accepts both $abcd$ and $abdc$.

Observation C.3. *If a state q is reachable in a concurrent program P, then $\langle q, S\rangle$ is reachable in $\Xi_\lessdot(P)$ for some $S \subseteq \Sigma$.*

Proof. Let w be a word that reaches state q. Among all the interleavings of w (permutations that preserve the order with in each thread), let v be the lexicographically minimal such interleaving. Then v also reaches state q. As only statements of different threads commute, all elements of $[v]$ must also be interleavings of v. Thus, v is minimal within its equivalence class, and either v or an extension of v to a word accepted by P is also accepted by the sleep set automaton. Hence the sleep set automaton must not get stuck while reading v, and so it reaches a state $\langle q, S\rangle$ for some sleep set S. □

D Additional Material for Section 6: Space-Efficient Representations

Observation D.1. *For a state q of a concurrent program P, every weakly persistent set in q is a membrane for q.*

Proof. If q is terminal, then $\mathcal{L}_P(q) \subseteq \{\varepsilon\}$. As $\mathcal{L}_P(q)$ contains no non-empty word, any M is trivially a membrane for q. If $q = \langle \ell_1, \ldots, \ell_n\rangle$ is not terminal, observe that $enabled(q) = \bigcup_{i=1..n, \ell_i \neq \ell^i_{\text{exit}}} enabled_{T_i}(\ell_i)$. To reach an accepting state, each thread i with $\ell_i \neq \ell^i_{\text{exit}}$ must make at least one step. Hence, $enabled_{T_i}(\ell_i)$ for any such i is a membrane for q. A weakly persistent set M for q must necessarily be a superset of at least one such set (see results for section 7.1) and thus also a membrane for q. □

Theorem 6.3 (Soundness of π-Reduction)**.** *Assume the language of A is closed, and $\pi(q)$ is a weakly persistent membrane for each state q of A. The π-reduced automaton $A_{\downarrow\pi}$ recognizes a reduction of $\mathcal{L}(A)$.*

Proof. Theorem D.2 proves a stronger result. □

In this appendix, we prove a more precise version of the above theorem. This more precise version is then used to argue for soundness of the combined construction in section 6.2. Specifically, we will here describe the preference order that induces the reduction recognized by the π-reduced automaton. Note that this preference order does not fall into the class of positional lexicographic preference orders described in section 4 (though it looks similar), as it is not total, and hence a class might have multiple representatives.

We define, for a set M of letters, the partial strict order $<_M$ over letters as follows: The letter a is smaller than the letter b if a lies in M, and b does not.

$$a \in M,\ b \notin M \ \Rightarrow\ a <_M b$$

Given the mapping π, we define the preference order $\ll_\pi^q$, a preference order on words, as the smallest relation such that for all words w, v and letters a, b we have $w \ll_\pi^q wv$, and if $a <_{\pi(q')} b$ for $q' = \delta_+^*(q, w)$, then $wau \ll_\pi^q wbv$. By $\ll_\pi$ we denote $\ll_\pi^{q_{\text{init}}}$.

We denote the language obtained by applying the corresponding reduction (i.e., the reduction induced by the π-preference order) to the language of A by

$$red_{\ll_\pi}(\mathcal{L}(A)).$$

The next statement links the persistent-set reduction of an automaton (see section 6.1) and the persistent-set reduction of its language.

Theorem D.2 (Soundness of Persistent Set Reduction)**.** *Assume unconditional independence. The automaton obtained by applying the π-reduction to the DFA A recognizes exactly the language obtain by applying the reduction induced by the π-preference order to the language recognized by A, i.e.,*

$$\mathcal{L}(A_{\downarrow\pi}) = red_{\ll_\pi}(\mathcal{L}(A))$$

if the language of A, i.e., $\mathcal{L}(A)$ is $\ll_\pi$-closed, and the mapping π always assigns to a state q a weakly persistent membrane for q.

Proof. We prove the more general statement that for every $w \in \Sigma^*$, it holds that

$$\forall q \in Q\,.\, w \in \mathcal{L}_{A_{\downarrow\pi}}(q) \iff w \in red_{\ll_\pi^q}(\mathcal{L}_A(q))$$

Proceed by induction over the length $|w|$ of w. In the case where $|w| = 0$ and $w = \varepsilon$, the result is immediate, as both sides of the above equivalence reduce to $q \in F$.

For the induction step, let $w = av$. For the inclusion, let now $av \in \mathcal{L}_{A_{\downarrow\pi}}(q)$. Then it is easy to see that $av \in \mathcal{L}_A(q)$, and $a \in \pi(q)$. Hence there can be no $x \in \Sigma^*$ such that $x \ll_\pi^q av$ unless $x = av$, and hence $av \in \min_{\ll_\pi^q}([av] \cap \mathcal{L}_A(q)) \subseteq red_{\ll_\pi^q}(\mathcal{L}_A(q))$.

For the reverse inclusion, let now $av \in red_{\ll_\pi^q}(\mathcal{L}_A(q))$. That means there exists some $x \in \mathcal{L}_A(q)$ such that $av \in \min_{\ll_\pi^q}([x] \cap \mathcal{L}_A(q))$. We want to show that $a \in \pi(q)$. For purposes of contradiction, suppose now that $a \notin \pi(q)$. By the fact that $av \in \mathcal{L}_A(q)$ and $\pi(q)$ is a membrane for q, there exists some (and hence also, a first) letter $b \in \pi(q)$ such that $av = av_1bv_2$ for some $v_1, v_2 \in \Sigma^*$. By the fact that $\pi(q)$ is weakly persistent, that then implies that $av \sim_q bav_1v_2$. But then $bav_1v_2 \ll_\pi^q av$ and $x \sim_q av \sim_q bav_1v_2$, and by $\ll_\pi^q$-closedness of $\mathcal{L}_A(q)$[6], $bav_1v_2 \in \mathcal{L}_A(q)$, which contradicts the fact that $av \in \min_{\ll_\pi^q}([x] \cap \mathcal{L}_A(q))$.

Hence we now know that $a \in \pi(q)$. Then let $q' := \delta_{\downarrow\pi}(q, a) = \delta(q, a)$. We then conclude that $v \in \mathcal{L}_A(q')$, $v \in red_{\ll_\pi^{q'}}(\mathcal{L}_A(q'))$ – if v wasn't minimal, av would not be minimal either –, and by induction hypothesis, $v \in \mathcal{L}_{A_{\downarrow\pi}}(q')$. Hence $av \in \mathcal{L}_{A_{\downarrow\pi}}(q)$. □

Proposition 6.4. *The subset of outgoing edges assigned to the state q by the mapping π must be a membrane for q if $\mathcal{L}_{A_{\downarrow\pi}}(q)$ is a reduction of $\mathcal{L}_A(q)$.*

Proof. If, for a proof by contraposition, $\pi(q)$ is not a membrane for q, then there exists a word $w \in \mathcal{L}_A(q)$ such that $w \neq \varepsilon$ and w does not contain any letter in $\pi(q)$. But every $v \in \mathcal{L}_{A_{\downarrow\pi}}(q)$ (where $v \neq \varepsilon$) begins with a letter in $\pi(q)$. Thus, there can be no such v with $w \sim v$; i.e., $\mathcal{L}_{A_{\downarrow\pi}}(q)$ is not a reduction of $\mathcal{L}_A(q)$. □

Note that, in the situation of proposition 6.4, when we take a state q and a word w such that $\delta^*(q_{\text{init}}, w) = q$, if the set $\pi(q)$ is not a membrane and the word $v \in \mathcal{L}_A(q)$ has no representative in $\mathcal{L}_{A_{\downarrow\pi}}(q)$, then it could still be that the composed word wv does have a representative in $\mathcal{L}(A_{\downarrow\pi})$ (because, "by chance", some word accepted by $A_{\downarrow\pi}$ is equivalent to wv). Theorem D.2 required the language of the input automaton to be $\ll_\pi$-closed. For the combination of sleep set and persistent reduction this is critical: The sleep set reduction automaton $\mathfrak{S}_{lex(<)}(A)$ does not recognize a closed language, but a reduction. We can not generally assume $\ll_\pi$-closedness either. However, compatibility resolves this issue. We state here a more general version of compatibility, that follows directly from the simple local criterion for (positional) lexicographic preference orders given in section 6.2.

[6]Strictly speaking, we need to restrict to reachable states q. Then closedness of $\mathcal{L}_A(q)$ follows from the assumption of closedness of $\mathcal{L}(A)$. Since we also need this for sleepsets, I should probably put it in a lemma.

Definition D.3 (Compatibility). *Let $\pi : Q \to 2^\Sigma$, and let $\preceq \subseteq \Sigma^* \times \Sigma^*$ be a preference order. We say that π and $\preceq$ are* compatible *if $\ll_\pi \subseteq \preceq$.*

The idea is that for each word w, if a previous (sleep set) reduction chooses the lexicographically minimal representative $v \in \min_{lex(\lessdot)}[w]$, then persistent set reduction must not remove this representative in favor of another representative already removed by the first reduction. The second obstacle we face is the fact that our variant of persistent set reduction is based on unconditional commutativity. However, we take advantage of conditional commutativity in sleep set reduction. To reconcile this difference, we identify the unconditional core of a conditional commutativity relation.

Definition D.4. *Let $\leftrightarrows_\cdot : Q \to 2^{\Sigma\times\Sigma}$ be a conditional commutativity relation. Then we define that letters $a, b \in \Sigma$ commute unconditionally, denoted $a \leftrightarrows b$, iff $a \leftrightarrows_q b$ for all conditions $q \in Q$.*

The following lemma shows that compatibility achieves the goal we set:

Lemma D.5. *Let $\pi : Q \to 2^\Sigma$, and let $lex(\lessdot)$ be a regular lexicographical order, If π and $lex(\lessdot)$ are compatible, then $\mathcal{L}(\mathfrak{S}_\lessdot(A))$ is $\ll_\pi$-closed wrt. the unconditional commutativity relation $\leftrightarrows$.*

Proof. $\mathcal{L}(\mathfrak{S}_\lessdot(A)) = red_{lex(\lessdot)}(A)$ is $lex(\lessdot)$-closed. By compatibility and lemma B.5, the result follows. □

We formalize and proof the observation made in the paper about weakly persistent membranes for the sleep set automaton.

Lemma D.6. *Let $q \in Q$ be a state, and let $S \subseteq \Sigma$ be a set of letters. If M is a weakly persistent membrane for q, then $M \setminus S$ is a weakly persistent membrane for the state $\langle q, S\rangle$ of $\mathfrak{S}_\lessdot(A)$.*

Proof. We begin by showing that $M \setminus S$ is a membrane for $\langle q, S\rangle$. As M is a membrane for q, and $\mathcal{L}_{\mathfrak{S}_\lessdot(A)}(\langle q, S\rangle) \subseteq \mathcal{L}_A(q)$, it follows that M is a membrane for $\langle q, S\rangle$. Let now $w \in \mathcal{L}(\langle q, S\rangle)$. We know that w contains some letter $b \in M$. If $b \notin S$, i.e., $b \in M \setminus S$, then we are done. Now consider the case that $b \in M \cap S$. Then clearly b cannot be the first letter of w, otherwise $w \notin \mathcal{L}(\langle q, S\rangle)$. In fact, there must occur a letter c in w before b such that $c \not\leftrightarrows b$, otherwise b is still in the sleep set. By the fact that M is a persistent set for q, we know that the first such c must be in M. Inductively, we arrive at the fact that the first letter in w that is in M must not be in S. Hence $M \setminus S$ is a membrane for $\langle q, S\rangle$.

To show that $M \setminus S$ is weakly persistent at $\langle q, S\rangle$, let us first observe that $M \subseteq enabled_A(q)$, and hence $M \setminus S \subseteq enabled_A(q) \setminus S = enabled_{\mathfrak{S}_\lessdot(A)}(\langle q, S\rangle)$. Now, let $a_1 \dots a_m \in \mathcal{L}(\langle q, S\rangle)$ such that there exists $b \in M \setminus S$ with $a_i \not\leftrightarrows b$. But then also $a_1 \dots a_m \in \mathcal{L}(q)$, and $b \in M$. Hence because M is weakly persistent at q, there exists $j \le i$ with $a_j \in M$. But as argued for membranes above, the first letter in $a_1 \dots a_m$ that is in M must not be in S. Hence there exists $k \le j \le i$ with $a_m \in M \setminus S$. □

Theorem 6.5 (Soundness of Combined Reduction). *The automaton $\left(\mathfrak{S}_\lessdot(A)\right)_{\downarrow\pi_\Leftrightarrow}$ recognizes the lexicographic reduction induced by the preference order $lex(\lessdot)$.*

$$\mathcal{L}(\left(\mathfrak{S}_\lessdot(A)\right)_{\downarrow\pi_\Leftrightarrow}) = red_{lex(\lessdot)}(\mathcal{L}(A))$$

Proof. By lemma D.5, lemma D.6 and theorem D.2, it follows that $L' := \mathcal{L}(\left(\mathfrak{S}_\lessdot(A)\right)_{\downarrow\pi})$ is indeed a reduction of $\mathcal{L}(A)$. Further, we clearly have $L' \subseteq \mathcal{L}(\mathfrak{S}_\lessdot(A)) = red_{lex(\lessdot)}(A)$. But since $red_{lex(\lessdot)}(A)$ is a minimal reduction, it follows that $L' = red_{lex(\lessdot)}(A)$. □

E Additional Material for Section 7: Proof Checking for Reductions

The following result says that the approach we take to computing persistent sets, i.e., picking a set E of threads and taking their enabled actions, is sound (and there is no alternative).

Proposition E.1. *M is weakly persistent at state q of a concurrent program P iff there exists a conflict-closed set of non-terminated threads $E \subseteq \{1, \dots, n\}$ such that $M = \bigcup_{i\in E} enabled_{T_i}(\ell_i)$. Further, E must only be empty if q has no outgoing edges.*

Proof. If M contains some transition $a \in enabled_{T_i}(\ell_i)$, then M must contain every enabled letter of the thread T_i (i.e., $enabled_{T_i}(\ell_i) \subseteq M$). Otherwise some $b \in enabled_{I_1}(\ell_i) \setminus M$ could be executed, which does not commute with all letters in M (at the very least, not with a), violating the definition of weakly persistent sets. Hence the set of letters are given by a set E of threads. Similarly, if some thread $i \in E$ has a conflict with a thread $j \notin E$, we consider the word w consisting only of letters of thread j until the conflicting location ℓ'_j is reached; followed by the conflicting outgoing edge of ℓ'_j; followed by more letters of all threads until an accepting state is reached. Then the first letter that does not commute with all letters in M is the one labeling the conflicting edge, which is not itself in M.

The proof that conversely, every set E as described yields a weakly persistent set is straightforward. □

Proposition 7.1. *π implemented by algorithm 1 is compatible with the preference order $lex(\lessdot)$, and maps states to weakly persistent membranes.*

Proof. Apply lemma **??** on the fact that a topologically maximal strongly connected component is non-empty and closed under the edge relation. For compatibility, note that if action $a <_q b$, a is an action of thread i and b an action of thread j, and $a \in \pi(q)$, then algorithm 1 creates an edge (ℓ_i, ℓ_j) in the graph. Since the set E of threads is a topologically maximal strongly connected component and $i \in E$, it follows that $j \in E$ and hence $b \in \pi(q)$. □

In order to prove our efficiency theorem, theorem 7.2, we show that all reachable states of the automaton in question, i.e., $\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}$, have a certain form. Specifically, let $\sigma_k(\ell_k)$ for $k \in \{1, \ldots, n\}$ and $\ell_k \in Q_k$ denote the state $\langle q, S\rangle$ with the program location $q = \langle \ell^1_{\text{exit}}, \ldots, \ell^{k-1}_{\text{exit}}, \ell_k, \ell^{k+1}_{\text{init}}, \ldots, \ell^n_{\text{init}}\rangle$ and the sleep set $S = \bigcup_{i=1}^{k-1} \mathit{enabled}(\ell^i_{\text{exit}})$. The following lemma describes the weakly persistent membranes our algorithm computes for such states.

Lemma E.2. *Let $\lessdot$ be non-positional and thread-uniform, and assume full commutativity. Let $k \in \{1, \ldots, n\}$, $\ell_k \in Q_k$, and $\sigma_k(\ell_k) = \langle q, S\rangle$ such that $q = \langle \ell_1, \ldots, \ell_n\rangle$. Let k' be the least index such that $k' \geq k$ and $\mathit{enabled}_{T_{k'}}(\ell_{k'}) \neq \emptyset$, if such an index exists. Then it follows that*

$$\pi(\sigma_k(\ell_k)) \subseteq \mathit{enabled}_{T_{k'}}(\ell_{k'})$$

Proof. Let us go through the computation of `CompatiblePersistentSet(q)` step-by-step.

We begin with the assignment of *active*, and note that $\ell_{k'} \in \mathit{active}$ by assumption. Next, let us consider the relation *conflicts*. By the assumption of full commutativity, a conflict $\ell_j \rightsquigarrow_{\ell_i}$ only occurs if $i = j$. Any conflicts (ℓ_i, ℓ_j) induced by the preference order must also satisfy $i \geq j$. In particular, for all $i \in \{k' + 1, \ldots, n\}$ such that $\ell_i \in \mathit{active}$, the preference order ensures that *conflicts* contains the pair $(\ell_i, \ell_{k'})$.

Since there are no conflicts (ℓ_i, ℓ_j) where $i < j$, a topologically maximal SCC cannot contain any ℓ_i with $i > k'$. Thereby, our algorithm, i.e., a call of `CompatiblePersistentSet(q)`, computes a weakly persistent membrane $M_q \subseteq \bigcup_{i=1}^{k'} \mathit{enabled}(\ell_i)$ for state q. But since k' is the least index greater or equal k that has an enabled action, we can refine the inclusion to

$$M_q \subseteq \bigcup_{i=1}^{k'} \mathit{enabled}(\ell_i) = \left(\bigcup_{i=1}^{k-1} \mathit{enabled}(\ell_i)\right) \cup \mathit{enabled}(\ell_{k'})$$

Finally, to arrive at a weakly persistent membrane for $\langle q, S\rangle$, we subtract S from M_q. But by the definition of $\sigma_k(\ell_k)$, we have that $S = \bigcup_{i=1}^{k-1} \mathit{enabled}(\ell_i)$, and hence $\pi(\langle q, S\rangle) \subseteq \mathit{enabled}(\ell_{k'})$. □

The efficiency theorem itself is once again stated more precisely here:

Theorem 7.2. *If $\lessdot$ is thread-uniform and non-positional, and we have full commutativity, the automaton $\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi_{\mathfrak{S}}}$ has $O(\mathit{size}(P))$ reachable states.*

Proof. Let us assume, for purposes of simplicity, that for the program $P = T_1 \parallel \ldots \parallel T_n$, our non-positional thread-uniform preference order $\lessdot$ orders actions of thread T_1 before those of thread T_2, and actions of thread T_2 before those of thread T_3, etc. If this is not the case, we renumber the threads to ensure their numbering conforms to the given preference order.

We now prove that all reachable states of $\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}$ have the form $\sigma_k(\ell_k)$ for some $k \in \{1, \ldots, n\}$ and $\ell_k \in Q_k$. To this end, we proceed by induction over the word w by which a state is reached. Hence, let $w \in \Sigma^*$ such that $\delta^*_{\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}}(\langle q_{\text{init}}, \emptyset\rangle, w) = \langle q, S\rangle$ for some program location $q \in Q_P$ and some sleep set $S \subseteq \Sigma$.

- **Induction Start** If $w = \varepsilon$, i.e., $\langle q, S\rangle = \langle q_{\text{init}}, \emptyset\rangle$, then it follows immediately that q has the described form for $k = 1$ and $\ell_k = \ell^1_{\text{init}}$. Hence, it follows that $\langle q, S\rangle = \sigma_1(\ell^1_{\text{init}})$.
- **Induction Step** Let now $w = va$ for some $v \in \Sigma^*$ and $a \in \Sigma_k$ for some $k \in \{1, \ldots, n\}$. There exists a state $\langle q', S'\rangle$ such that by reading the prefix v we reach $\delta^*_{\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}}(\langle q_{\text{init}}, \emptyset\rangle, v) = \langle q', S'\rangle$, and $\delta_{\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}}(\langle q', S'\rangle, a) = \langle q, S\rangle$. By induction hypothesis, we have some $k', \ell_{k'}$ such that $\langle q', S'\rangle = \sigma_{k'}(\ell_{k'})$.
  From the fact that $\langle q', S'\rangle$ has a transition labeled by the letter a, i.e., $a \in \pi(\langle q', S'\rangle)$, we conclude by lemma E.2 that k is the least index greater or equal than k' that is enabled in q'. Since only ℓ^i_{exit} may be terminal in thread i, we have that all

threads $i \in \{k, \ldots, k-1\}$ have reached ℓ^i_{exit} in state q'. Furthermore, the sleep set can be written as

$$\begin{aligned}
S &= \{\, b \in enabled(q') \mid (b \in S' \vee b <_{q'} a) \wedge a \leftrightsquigarrow_{q'} b \,\} && (\textit{definition of } \mathfrak{S}(\cdot)) \\
&= \{\, b \in enabled(q') \mid (b \in S' \wedge a \leftrightsquigarrow_{q'} b) \vee (b \notin S' \wedge b <_{q'} a \wedge a \leftrightsquigarrow_{q'} b) \,\} \\
&= \{\, b \in enabled(q') \mid b \in S' \vee (b \notin S' \wedge b <_{q'} a \wedge a \leftrightsquigarrow_{q'} b) \,\} && (\textit{full commutativity}) \\
&= \{\, b \in enabled(q') \mid b \in S' \vee (b \in \Sigma_k \wedge b <_{q'} a \wedge a \leftrightsquigarrow_{q'} b) \,\} && (\textit{assumption on } \lessdot) \\
&= \{\, b \in enabled(q') \mid b \in S' \,\} && (\textit{no commutativity within thread } T_k) \\
&= S' \\
&= enabled(q') \cap \bigcup_{i=1}^{k'-1} \Sigma_i && (\textit{induction hypothesis}) \\
&= enabled(q') \cap \bigcup_{i=1}^{k-1} \Sigma_i && (\textit{minimality of } k) \\
&= enabled(q) \cap \bigcup_{i=1}^{k-1} \Sigma_i && (\textit{only thread } k \textit{ changes location})
\end{aligned}$$

Thereby it follows for the successor state $\langle q, S\rangle$ that indeed $\langle q, S\rangle = \sigma_k(\ell_k)$, where ℓ_k is the thread location reached by executing a.

Finally, since there are at most $size(P) = \sum_{i=1}^{n} |T_i|$ distinct states of the form $\sigma_k(\ell_k)$, we conclude that the automaton $\mathfrak{S}_{\lessdot}(P)_{\downarrow\pi}$ has $O(size(P))$ reachable states. □

We give here the precise definition of Floyd/Hoare automata:

Definition E.3 (Floyd/Hoare automaton)**.** *A Floyd/Hoare automaton over alphabet Σ is a total DFA $A = (Q_A, \Sigma, \delta_A, q^A_{\text{init}}, F_A)$ such that*

- *Q_A is a set of formulae, whose free variables are a subset of the program variables,*
- *$q^A_{\text{init}} = pre$, and $F_A = \{post\}$,*
- *and for all $\varphi, \psi \in Q_A$ and $a \in \Sigma$ such that $\delta_A(\varphi, a) = \psi$, the Hoare triple $\{\varphi\}\, a\, \{\psi\}$ is valid.*

We formalize here the definition of proof-sensitive commutativity:

Definition 7.3 (Proof-Sensitive Commutativity)**.** *Let φ be an assertion of $\mathcal{A}$. Statements a and b commute under condition φ, denoted $a \leftrightsquigarrow_\varphi b$, iff the compositions ab and ba have the same semantics when starting from a state satisfying φ. Formally, let $[\![\varphi]\!]$ denote the set of all program states satisfying φ. Then we have $a \leftrightsquigarrow_\varphi b$ iff*

$$([\![\varphi]\!] \times [\![\top]\!]) \cap [\![ab]\!] = ([\![\varphi]\!] \times [\![\top]\!]) \cap [\![ba]\!]$$

Proof-sensitive commutativity is an instance of conditional commutativity [13]. Proof-sensitive commutativity satisfies our intuition that covering between traces preserves correctness:

Lemma E.4. *Proof-sensitive commutativity based on a Floyd/Hoare automaton preserves valid Hoare triples: Let τ_1, τ_2 be traces such that $\tau_1 \sim \tau_2$. If a Hoare triple $\{\varphi\}\, \tau_2\, \{\psi\}$ is valid, then $\{\varphi\}\, \tau_1\, \{\psi\}$ is also valid.*

Proof. Since equality is reflexive and transitive, we only have to prove the case that $\tau_1 = uabv$, $\tau_2 = ubav$ for some $u, v \in \Sigma^*$ and $a, b \in \Sigma$ such that $a \leftrightsquigarrow_\varphi b$, where $\delta^*_A(\top, u) = \varphi$. Then by inductivity of the Floyd/Hoare automaton, we have that $\{\top\}\, u\, \{\varphi\}$ is valid, or equivalently $[\![u]\!] \subseteq [\![\top]\!] \times [\![\varphi]\!]$. Hence

$$\begin{aligned}
[\![\tau_1]\!] &= [\![u]\!] \circ [\![ab]\!] \circ [\![v]\!] \\
&= [\![u]\!] \circ \Big(([\![\varphi]\!] \times [\![\top]\!]) \cap [\![ab]\!]\Big) \circ [\![v]\!] \\
&= [\![u]\!] \circ \Big(([\![\varphi]\!] \times [\![\top]\!]) \cap [\![ba]\!]\Big) \circ [\![v]\!] \\
&= [\![u]\!] \circ [\![ba]\!] \circ [\![v]\!] \\
&= [\![\tau_2]\!]
\end{aligned}$$

□

Theorem 7.4 (Soundness)**.** *If* `CheckProof`$(q_{init}, pre, \emptyset)$ *does not find a counterexample, the program is correct, i.e., P satisfies the pre/postcondition-pair* $(pre, post)$.

Proof. Let $P \times \mathcal{A}$ denote the product automaton of the interleaving product P and the Floyd/Hoare automaton $\mathcal{A}$, where states are accepting iff the P-component is accepting. Since $\mathcal{A}$ is total, the language of $P \times A$ is exactly the same as the language of P (and hence closed), only the states differ. We apply the combined reduction with conditional sleep set reduction (using proof-sensitive commutativity) and unconditional persistent set reduction to obtain the automaton

$$B := \Big(\mathfrak{S}_{\preccurlyeq}(P \times A) \Big)_{\downarrow \pi_{\mathfrak{S}}}$$

By theorem 6.5 and 7.1, the automaton B recognizes a reduction of $\mathcal{L}(P)$. Finally, consider the automata difference between B and the Floyd/Hoare automaton $\mathcal{A}$. This difference automaton is isomorphic to B, merely the accepting states change. The call `CheckProof`$(q_{init}, pre, \emptyset)$ amounts to an emptiness check of this difference automaton. If no counterexample is found, all words accepted by B (i.e., all words in the reduction) are also accepted by $\mathcal{A}$ (and thus satisfy the pre/postcondition pair $(pre, post)$. From lemma E.4 it follows that then all words of P satisfy this pre/postcondition pair, and P is correct. □

Theorem 7.5 (Efficiency)**.** *If the mapping $\preccurlyeq$ is thread-uniform and non-positional, and we have full commutativity, the time required by algorithm 2 is polynomial in size(P).*

Proof. By theorem 4.2, the size of the reduced automaton B from the proof of theorem 7.4 is linear. By our observations on the efficiency of algorithm 1, only polynomial time is needed to compute weakly persistent membranes. Putting these results together, it is easy to see that we can construct B in polynomial time. Hence, the inclusion check (algorithm 2) that constructs this automaton on-the-fly and checks its inclusion against $\mathcal{A}$ also terminates in polynomial time. Specifically, the time required is in $O(size(P)^2 + n^2 \cdot size(P))$, where n is the number of threads. □

Observation E.5. *If two statements commute under some condition, they also commute under all stronger conditions.*

Proof. Let a, b be statements that commute under a condition φ, and let ψ be a stronger condition. The result is easy to see: If ab and ba behave the same when starting in any state satisfying φ, they must also behave the same when starting in any state satisfying ψ (a subset of states). □