0.1 From Examples to Precise Rules

Notations

- T_i denotes thread number i.
- $T_i \equiv T_j$ means both threads have same code.
- w_i^j is the j^{th} event in thread i which is a write.
- r_i^j is the j^{th} event in thread i which is a read.

A few definitions for our use

Definition 1. *Program Order(*po*) Total order between events in the same thread. Respects the execution order between events in the same thread.*

Definition 2. *Symmetric Memory Order (*smo*) A strict partal order between writes in a set of symmetric threads. Consider a set of symmetric threads $T_1 \equiv T_2 \equiv ... \equiv T_n$. Each of these threads have exactly one read event, and multiple write events, all to the same memory, say x.*

Then each write in the above threads are involved in a symmetric order, such that.

$$\forall i \in [0, n-1] \ . \ w_i^j \xrightarrow[smo]{} w_{i+1}^j$$

Where j denotes the j^{th} event in any of the threads, which is a write.

Perhaps should put examples for the above defintion.

Definition 3. *Reads-From (*rf*) Binary relation that links a read to a write from which its value comes. Note that for our purpose, this relation is functional. For example, if a read r_i^j gets its read value from write w_k^l, then we have the relation.*

$$w_k^l \xrightarrow[rf]{} r_i^j$$

Main Rule Using the above setup, our intention is to explore lesser execution graphs leveraging the symmetry that can result due to swapping of thread identities. For this, we enforce a restriction on possible $\xrightarrow[rf]{}$ relations that are to be considered *valid*. A valid $\xrightarrow[rf]{}$ relation is one that respects the following **irreflexivity constraints**.

$$smo; rf; po$$
$$smo; po; rf^{-1}$$

Perhaps consider labelling this rule

We can add here more as we go about to prove completeness.

Recall examples to show how our analysis through examples satisfy the above irreflexivity constraint.

0.2 Soundness of the rules above

To prove soundness, we first define the following:

Definition 4. *Implied Write Order(*iwo*) Binary relation between any two* distinct *writes of symmetric threads, derived through the following two sequential conposition:*

$$w_i^j; po; rf^{-1}; w_k^j$$
$$w_i^j; rf; po; w_k^j$$

Property 0.1. $w_i^j; po; rf^{-1}; w_k^j$ *corresponds to an implied write order between writes above the read((po before read)) and* T_i*'s read being satisfied.*

Proof. Expanding the first sequential composition exposes the following relations involved.

$$w_i^j \xrightarrow[po]{} r_i$$
$$r_i \xrightarrow[rf^{-1}]{} w_k^j$$

By Def 1, w_i^j is above its read r_i. Because T_i and T_k are symmetric threads, we can infer $w_k^j \xrightarrow[po]{} r_k$, thus verifying that w_k^j is also above its read. By Def 3, we can infer $w_k^j \xrightarrow[rf]{} r_i$ implying T_i's read being satisfied. □

Property 0.2. $w_i^j; rf; po; w_k^j$ *corresponds to an implied write order between writes below the read (po after read) and* T_k*'s read being satisfied.*

Proof. Expanding the first sequential composition exposes the following relations involved.

$$w_i^j \xrightarrow[rf]{} r_k$$
$$r_k \xrightarrow[po]{} w_k^j$$

By Def1, w_k^j is below its read r_k. Because T_i and T_k are symmetric threads, we can infer $r_i \xrightarrow[po]{} w_i^j$, thus verifying that w_i^j is also below its read. By Def 3, we can infer T_k's read being satisfied. □

Property 0.3. *Simplified irreflexivity rule The irreflexivity constraint rule is equivalent to the following irreflexivity condition*

$$smo; iwo$$

Proof. Expanding for implied write order as per the definition, gives us the following two sequential compositions.

$$smo; w_i^j; po; rf^{-1}; w_k^j$$
$$smo; w_i^j; rf; po; w_k^j$$

If the above relations must be irreflexive, then so should the following:

$$w_k^j; smo; w_i^j; po; rf^{-1}$$
$$w_k^j; smo; w_i^j; rf; po$$

By Def 2, the above can be simplified to

$$smo; po; rf^{-1}$$
$$smo; rf; po$$

Thus, proving our property. □

Property 0.4. *No write order is implied when a read reads from its own thread's write*

Proof. If the read is from its own thread's write, then we can infer that $i = k$ in both the sequential compositions. Hence

$$w_i^j; rf; po; w_i^j$$
$$w_i^j; po; rf^{-1}; w_i^j$$

which gives us $w_i^j \xrightarrow[iwo]{} w_i^j$. Since implied write orders are only between distinct writes, the property is proven. □

Perhaps define implied write order as being irreflexive

Property 0.5. *Implied write orders between two symmetric threads are reversed when they are swapped.*

Proof. Considering first sequential composition, i.e. $w_i^j; rf; po; w_k^j$, expanding gives us the following binary relations involved:

$$w_i^j \xrightarrow[rf]{} r_k$$
$$r_k \xrightarrow[po]{} w_k^j$$

Swapping thread identities involves swapping the indices i and k for each event, thus giving us

$$w_k^j \xrightarrow[rf]{} r_i$$
$$r_i \xrightarrow[po]{} w_i^j$$

Through sequential composition of the above, we get $w_k^j; rf; po; w_i^j$, which by definition is $w_k^j \xrightarrow[iwo]{} w_i^j$.

For the second sequential composition, i.e. $w_i^j; po; rf^{-1}; w_k^j$, expanding gives us the following binary relations involved:

$$w_i^j \xrightarrow[po]{} r_i$$
$$r_i \xrightarrow[rf^{-1}]{} w_k^j$$

Swapping thread identities T_i and T_k gives us the following relations

$$w_k^j \xrightarrow[po]{} r_k$$
$$r_k \xrightarrow[rf^{-1}]{} w_i^j$$

Whose sequential compostiion gives us $w_k^j; po; rf^{-1}; w_i^j$, which by defintion is $w_k^j \xrightarrow[iwo]{} w_i^j$. □

Property 0.6. *There are at most two implied write orders between writes of two threads, with one between writes above the read event and one below.*

Proof. Consider two threads T_i and T_k. Suppose we have one implied write order between one of their writes, i.e.

$$w_i^j \xrightarrow[iwo]{} w_k^j.$$

If the above is derived by $w_i^j; po; rf^{-1}; w_k^j$ then from Prop 0.1, we can infer the two writes are above the read and T_i' read being satisfied.

Now suppose we have an implied write order between another set of writes, i.e.

$$w_i^l \xrightarrow[iwo]{} w_k^l.$$

If the above is derived by $w_i^l; po; rf^{-1}; w_k^l$, then by Prop 0.1 and Def 3, it violates $\xrightarrow[rf]{}$ functionality. Hence it can only be derived by $w_i^l; rf; po; w_k^l$. From Prop 0.2, we can infer that the two writes are below the read and T_k's read being satisfied.

Suppose a third impleid write order exists of the form

$$w_i^n \xrightarrow[iwo]{} w_k^n$$

If the above is derived by $w_i^n; po; rf^{-1}; w_k^n$, then by Prop 0.1 and Def 3, it violates $\xrightarrow[rf]{}$ functionality. If the above is derived by $w_i^n; rf; po; w_k^n$, then by Prop 0.2 and Def 3, it violates $\xrightarrow[rf]{}$ functionality.

Hence, we cannot have any more implied write orders between T_i and T_j, thus verifying our property. □

Property 0.7. *Implied write orders between two threads either all respect irreflexivity condition by Prop 0.3 or they all do not.*

Proof. Consider our two threads T_i and T_k.

For cases where implied write orders are established between wrties of T_i and T_j, let us first consider one between writes above the read which respect Prop 0.3;

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge \ w_i^j \xrightarrow[smo]{} w_k^j$$

By Prop 0.1, the implied write order satisfies T_i's read by a reads-from relation with w_k^j.

The other set of implied write order, by Prop 0.6 can only be between writes below the read. Suppose we have such an order but not respecting Prop 0.3:

$$w_k^l \xrightarrow[iwo]{} w_i^l \ \wedge \ w_i^l \xrightarrow[smo]{} w_k^l$$

By Prop 0.2, this implies another $\xrightarrow[rf]{}$ relation with T_i's read. By Def 3 this violates the functional property of $\xrightarrow[rf]{}$. Hence we can only have an implied write order respecting Prop 0.3.

$$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_i^l \xrightarrow[smo]{} w_k^l$$

By symmetry, if the implied write order between writes above the read did not respect Prop 0.3, then so would the writes below the read. □

Property 0.8. *Implied write orders are acyclic*

Proof. Suppose a cycle exists. Then without loss of generality, we can consider the cycle composed of 3 writes.

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge w_k^j \xrightarrow[iwo]{} w_l^j \ \wedge w_l^j \xrightarrow[iwo]{} w_i^j.$$

By Def 4 and Prop 0.1, 0.2, we can have just two cases; the set of writes involved in cycle are either above the read event or below.

If they are above the read, then we have the following relations that result in the cycle

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf^{-1}]{} w_k^j$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf^{-1}]{} w_l^j$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf^{-1}]{} w_i^j.$$

The above relations violate $po \cup rf^{-1}$ acyclic rule, thus violating coherence.

If the writes are below the read, then we have the following relations that result in the above cycle.

$$w_i^j \xrightarrow[po]{} r_i \ \wedge \ r_i \xrightarrow[rf]{} w_k^j$$
$$w_k^j \xrightarrow[po]{} r_k \ \wedge \ r_k \xrightarrow[rf]{} w_l^j$$
$$w_l^j \xrightarrow[po]{} r_l \ \wedge \ r_l \xrightarrow[rf]{} w_i^j$$

The above relations form a cycle thus violating $po \cup rf$ acyclic rule, thus violating coherence.

Because both cases violate coherence, we conclude that $\xrightarrow[iwo]{}$ is acylic. □

Property 0.9. *Writes w_i above read in each thread can only have one relation of the form $w_i \xrightarrow[iwo]{} w$ but can have several of the form $w \xrightarrow[iwo]{} w_i$.*

Proof. Suppose a write w_i^j above a read has a relation with w_k^j such that $w_i^j \xrightarrow[iwo]{} w_k^j$. By Prop 0.1, we can infer T_i's read has been satisfied. By Def 3 and our assumption of one read per thread, we cannot have any more implied write order relations of the form $w_i^j \xrightarrow[iwo]{} w^j$.

On the other hand, there can be many relations of the form $w_k^j \xrightarrow[iwo]{} w_i^j$ as k can be identity of any thread, whose read gets satisfied by the argument above. □

Property 0.10. *Writes w_i below read in each thread can only have one relation of the form $w \xrightarrow[iwo]{} w_i$ but can have several of the form $w_i \xrightarrow[iwo]{} w$.*

Proof. Suppose a write w_i^j below the read has a relation with w_k^j such that $w_k^j \xrightarrow[iwo]{} w_i^j$. By Prop 0.2, we can infer that T_i's read has been satisfied by write w_k^j. By Def 3 and our assumption of one read per thread, we cannot have any more implied write order relations of the form $w^j \xrightarrow[iwo]{} w_i^j$.

On the other hand, there can be many relations of the form $w_i^j \xrightarrow[iwo]{} w_k^j$ as k can be identity of any thread, whose read gets satisfied by the argument above. □

0.2.1 Soundness

To prove that our rules are sound, we need to show that for every execution graph that does not respect the irreflexivity constraint, there is a symmetric execution that does respect and hence covered by our rules.

By Prop 0.3, we can infer that an execution not respectiing our irreflexivity constraint will have implied write orders which are "incorrect".

Perhaps put Incorrect implied write orders as a definition??

By Prop 0.5, we can swap thread identities to reverse implied write orders, thus "fixing" them, which then can make the resultant execution respect Prop 0.3. If we can show this process of "fixing" implied write orders is terminating, our soundness proof is complete.

We also maintain the order in which we do such "fixing". We first address all implied write orders above the read and then go below. If any new implied write orders is introduced in the process, we fix them as per the above order, i.e. if the new relation is above the read, we fix that first, contrast to any new impled write order introduced below the read.

For each new implied write order introduced, if we show that it wasn't "fixed" before, it gives us guarantee that "fixing" is a terminating process.

However, before all this, we first need to ensure that implied write orders do not form a cycle, as "fixing" them may not terminate. By Prop 0.8, we have that $\xrightarrow[iwo]{}$ is acyclic, hence they do not form a cycle.

Given this, we show that for one given set of equal writes, swapping thread identities will always result in the implied write orders respecting Prop 0.3 (Lemma 1). We then show that this holds in general given multiple sets of equal writes (Lemma 2).

Lemma 1. *For one given set of equal writes, every implied write order that is "incorrect", can be fixed and will remain fixed. (A more formal statement required perhaps?)*

Proof. Suppose we have one implied write order which is correct / fixed between writes w_i^j and w_k^j.

$$w_i^j \xrightarrow[iwo]{} w_k^j$$

We divide our proof into two cases, one for writes above the read and one for writes below.

Case1 : The writes are above the read

- Part 1: There exists an implied write order between w_i^j and some w_m^j

  Because Prop 0.9 and $w_i^j \xrightarrow[iwo]{} w_k^j$, other implied write orders with w_i^j can be of the form:

  $$w_m^j \xrightarrow[iwo]{} w_i^j$$

  If this implied order is wrong, swapping thread identities will give us the following relations

  $$w_i^j \xrightarrow[iwo]{} w_m^j \ \wedge \ w_m^j \xrightarrow[iwo]{} w_k^j$$

  Here, there is no implied write order relation between w_i^j and w_k^j, hence we can consider them as remaining fixed. If the implied order between w_m^j and w_k^j is wrong, swapping it will result in the following relations

  $$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge \ w_k^j \xrightarrow[iwo]{} w_m^j$$

  Here, the implied write order between w_i^j and w_k^j remains the same. Thus, for this case, we can conclude that the relation remains "fixed".

- Part 2: There exists an implied write order between w_k^j and some w_m^j

  By Prop 0.9, implied write orders with w_k^j can be of the forms:

  $$w_k^j \xrightarrow[iwo]{} w_m^j$$
  $$w_m^j \xrightarrow[iwo]{} w_k^j$$

  The first form is symmetric to Case1, hence we only consider the second form.

  If this implied write order is wrong, swapping thread identities will give us the following relations.

  $$w_i^j \xrightarrow[iwo]{} w_m^j \ \wedge \ w_k^j \xrightarrow[iwo]{} w_m^j$$

  Here, there is no implied write order relation between w_i^j and w_k^j, hence we can consider them as remaining fixed.

  If the implied write order between w_m^j and w_i^j is wrong, swapping their threads will give us

  $$w_m^j \xrightarrow[iwo]{} w_i^j \ \wedge \ w_k^j \xrightarrow[iwo]{} w_i^j$$

  thus making our claim invalid. To show that this state is not possible, note firstly that from the initial configuration, we can infer by Def 2 and Prop 0.3:

  $$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^j \xrightarrow[smo]{} w_m^j$$

  Because *smo* is a total order w.r.t. one set of writes, we have by transitivity.

  $$w_i^j \xrightarrow[smo]{} w_m^j$$

  After swapping threads T_k and T_m, we get $w_i^j \xrightarrow[iwo]{} w_m^j$ which respects the irreflexivity constraint Prop 0.3. Hence, this implied write order is not wrong. Thus we cannot have the case which results in $w_k^j \xrightarrow[iwo]{} w_i^j$.

Case2: The writes are below the read

- Part 1: There exists an implied write order between w_i^j and some w_m^j

  By Prop 0.10, implied write order between w_i^j and w_m^j can be of two forms

  $$w_m^j \xrightarrow[iwo]{} w_i^j$$
  $$w_i^j \xrightarrow[iwo]{} w_m^j$$

  Considering the first form, if it is wrong, swapping it will result in the following relations

  $$w_i^j \xrightarrow[iwo]{} w_m^j \ \wedge \ w_m^j \xrightarrow[iwo]{} w_k^j$$

  Here, there is no implied write order relation between w_i^j and w_k^j, hence we can consider them as remaining fixed.

If the implied write order between w_m^j and w_k^j is wrong, swapping their threads will give us

$$w_i^j \xrightarrow[iwo]{} w_k^j \ \wedge \ w_k^j \xrightarrow[iwo]{} w_m^j$$

Here, the implied write order between w_i^j and w_k^j remains the same, while we fixed the relations they had with w_m^j. Thus, for this case, we can conclude that the implied write order remains "fixed".

Considering the second form, if it is wrong, swapping it will result in the following relations

$$w_m^j \xrightarrow[iwo]{} w_i^j \ \wedge \ w_m^j \xrightarrow[iwo]{} w_k^j$$

If the implied write order between w_m^j and w_i^j is wrong, swapping their threads will give us

$$w_k^j \xrightarrow[iwo]{} w_m^j \ \wedge \ w_k^j \xrightarrow[iwo]{} w_i^j$$

thus making our claim invalid. To show that this state is not possible, note firstly that from the initial configuration, we can infer by Def 2 and Prop 0.3 (given relation between w_i^j and w_m^j is wrong):

$$w_m^j \xrightarrow[smo]{} w_i^j \ \wedge \ w_i^j \xrightarrow[smo]{} w_k^j$$

Because *smo* is a total order w.r.t. one set of writes, we have by transitivity.

$$w_m^j \xrightarrow[smo]{} w_k^j$$

After swapping threads T_i and T_m, we get $w_m^j \xrightarrow[iwo]{} w_k^j$ which respects the irreflexivity constraint Prop 0.3. Hence, this implied write order is not wrong. Thus we cannot have the case which results in $w_k^j \xrightarrow[iwo]{} w_i^j$.

- Part 2: There exists an implied write order between w_k^j and some w_m^j

  By Prop 0.10 and $w_i^j \xrightarrow[iwo]{} w_k^j$, relation between w_k^j and w_m^j can only be of one form

  $$w_k^j \xrightarrow[iwo]{} w_m^j$$

  This case is symmetric to the first form of Part 1, hence this case is already proved.

Thus, for a given set of equal writes, once an implied write order is fixed, it remains fixed. □

Lemma 2. *For a given set of equal writes whose implied write orders respect Prop 0.3, new implied write order introduced among them by fixing other sets, if wrong, can be fixed and will remain fixed. (A more formal statement required? Discuss with Viktor)*

Proof. Main part We go about the proof by considering a set of writes fixed and another set that needs fixing. We then show that on fixing an implied write order, any new implied write order to the fixed set may either be right, or if wrong, was not something fixed before, and hence can be fixed by lemma 1. Since we fixed the order in which we "fix" these orders, we have to only consider the following cases:

1. Write orders above read fixed , write orders above still need fixing. Let us assume that the fixed set **above the read is the set of writes** w^j. Let us consider two threads T_i and T_k whose writes **above the read** w_i^l **and** w_k^l is going to be fixed. Without loss of generality, let us conisder the symmetric memory order between these two threads to be of the form $w_i \xrightarrow[smo]{} w_k$. By Prop 0.7, we then have the configuration as:

$$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^l \xrightarrow[iwo]{} w_i^l$$

We divide our analysis into two cases:

- There is an implied write order with w_i^j and some w^j

  By Prop 0.9, implied write orders between the two events can be of the form

  $$w^j \xrightarrow[iwo]{} w_i^j$$
  $$w_i^j \xrightarrow[iwo]{} w^j$$

  For the first case, note that by Def 2 and Prop 0.3, we have

  $$w * j \xrightarrow[smo]{} w_k^j$$

  By Prop 0.5, on swapping T_i and T_k, we have the following relations

  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w^j \xrightarrow[iwo]{} w_k^j$$

  The above relations do not violate Prop 0.3, hence we can stop here.

  For the second case, by Prop 0.5, on swapping T_i and T_k, we have the following relations

  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_k^j \xrightarrow[iwo]{} w^j$$

  The new relation $w_k^j \xrightarrow[iwo]{} w^j$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

  1. Case where $w^j \xrightarrow[iwo]{} x \ \wedge \ x \xrightarrow[iwo]{} w_k^j$
     The set of relations can exist in an execution graph, but results in $po \cup rf^{-1}$ cycle, thus violating coherence.
  2. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w_k^j$
     By Prop 0.9, x is an invalid event and thus such an execution graph cannot exist.
  3. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w_k^j$
     There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph an implied write order between w^j and w_k^j, which could have also been "fixed". By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. Such a configuration cannot exist, as $po \cup rf_{-1}$ is a cycle here, thus violating coherence.

4. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.9 and Prop 0.1, either x' or x is invalid, hence such an execution graph cannot exist.

- There is an implied write order with w^j_k and some w^j

  By Prop 0.9, implied write orders between the two events can be of the form

  $$w^j \xrightarrow[iwo]{} w^j_k$$

  For the above case, by Prop 0.5, on swapping T_i and T_k, we have the following relations

  $$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j \xrightarrow[iwo]{} w^j_i$$

  The new relation $w^j \xrightarrow[iwo]{} w^j_i$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

  1. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_i$
     By Prop 0.9, x is invalid and thus such an execution graph cannot exist.
  2. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ w^j_i \xrightarrow[iwo]{} x$
     The set of relations can exist in an execution graph, but results in $po \cup rf^{-1}$ cycle, thus violating coherence.
  3. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w^j_i$
     There could exist events w' and x' above the read program ordered with w^j and x respectively, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_i, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.9 and Prop 0.1, either w' or w^j is invalid, hence such a configuration cannot exist.
  4. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_i$
     There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_i, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.9 and Prop 0.1, either x' or x is invalid, hence such a configuration cannot exist.
  5. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ w^j_i \xrightarrow[iwo]{} x$
     There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_i, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. Though such an exeuction graph cann exist, it has $po \cup rf^{-1}$ as a cycle, thus violating coherence

6. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ w_i^j \xrightarrow[iwo]{} x$
   There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.9 and Prop 0.1, either w' or w^j is invalid, hence such a configuration cannot exist.

2. Write orders above read fixed , write orders below still need fixing. Let us assume that the fixed set **above the read is the set of writes** w^j. Let us consider two threads T_i and T_k whose writes **below the read** w_i^l **and** w_k^l is going to be fixed. Without loss of generality, let us conisder the symmetric memory order between these two threads to be of the form $w_i \xrightarrow[smo]{} w_k$. By Prop 0.7, we then have the configuration as:

$$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^l \xrightarrow[iwo]{} w_i^l$$

We divide our analysis into two cases

- There is an implied write order with w_i^j and some w^j
  By Prop 0.9 and Prop 0.2, implied write orders between the two events can be of the form
  $$w^j \xrightarrow[iwo]{} w_i^j$$
  By Def 2 and Prop 0.3, we have
  $$w^j \xrightarrow[smo]{} w_k^j$$
  By Prop 0.5, on swapping T_i and T_k, we have
  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w^j \xrightarrow[iwo]{} w_k^j$$
  THe above relations do not violate Prop 0.3, hence we can end this case here.
- There is an implied write order with w_k^j and some w^j By Prop 0.9 and Prop 0.2, implied write orders between the two events can be of the form
  $$w_k^j \xrightarrow[iwo]{} w^j$$
  $$w^j \xrightarrow[iwo]{} w_k^j$$
  For the first case, by Def 2 and Prop 0.3, we have
  $$w_i^j \xrightarrow[smo]{} w^j$$
  By Prop 0.5, on swapping T_i and T_k, we have
  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_i^j \xrightarrow[iwo]{} w^j$$
  The above relations do not violate Prop 0.3, hence we can end this case here.

For the second case, by Prop 0.5, on swapping, we have

$$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w^j \xrightarrow[iwo]{} w_i^j$$

The new relation $w^j \xrightarrow[iwo]{} w_i^j$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

1. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w_i^j$
   By Prop 0.9, x is invalid and thus such an execution cannot exist.
2. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w_i^j$
   There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.9 and Prop 0.1, either w' or w^j is invalid and thus such an execution graph cannot exist.
3. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w_i^j$
   There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.9 and Prop 0.1, either x' or x is invalid and thus such an execution graph cannot exist.

3. Write orders below read fixed , write orders below still need fixing. Let us assume that the fixed set **below the read is the set of writes** w^j. Let us consider two threads T_i and T_k whose writes **below the read** w_i^l **and** w_k^l is going to be fixed. Without loss of generality, let us conisder the symmetric memory order between these two threads to be of the form $w_i \xrightarrow[smo]{} w_k$. By Prop 0.7, we then have the configuration as:

$$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^l \xrightarrow[iwo]{} w_i^l$$

We divide our analysis into two cases

- There is an implied write order with w_i^j and some w^j
  By Prop 0.10 and Prop 0.2, implied write orders between the two events can be of the form

  $$w_i^j \xrightarrow[iwo]{} w^j$$

  By Prop 0.5, on swapping, we have

  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_k^j \xrightarrow[iwo]{} w^j$$

  The new relation $w_k^j \xrightarrow[iwo]{} w^j$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

1. Case where $w^j \xrightarrow[iwo]{} x \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   Though such an execution graph can exist, it has $po \cup rf$ cycle, thus violating coherence.
2. Case where $w^j \xrightarrow[iwo]{} x \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   By Prop 0.10, x is invalid.
3. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.2 and Prop 0.9, either w' or w^j are invalid and thus such an execution graph cannot exist.
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. Though such an execution graph can exist, it has $po \cup rf$ as a cycle, thus violating coherence.
4. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$.
     The only way to get a configuration with $w^j_k \xrightarrow[iwo]{} w^j$ before would be to swap T_x and T' first, followed by swapping T' and T_k. But this swapping order is not possible, as the "fixing" order we have is to address implied write orders above read first and then below. The "de-swapping" here can only be implied write orders below read and then those above. Since this is not possible in the above configuration, we can conclude that $w^j_k \xrightarrow[iwo]{} w^j$ did not exist before and hence could not have been fixed.
     The reader should note that though it was possible for the correct relation $w^j \xrightarrow[iwo]{} w^j_k$ to exist, it wasn't something that was "fixed" by swapping. This is different from those implied write orders which were fixed by swapping thread identities. (Discuss with Viktor on this)
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2 and Prop 0.10, either w' or w^j are invalid and thus such an execution graph cannot exist.
5. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.2 and Prop 0.9, either w' or w^j are invalid.
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been

fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.10 nad Prop 0.2, either x' or x are invalid and thus such an execution graph cannot exist.

6. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2 and Prop 0.9, either x' or x are invalid and thus such an execution graph cannot exist.
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2 and Prop 0.10, either w' or w^j are invalid and thus such an execution graph cannot exist.

- There is an implied write order with w^j_k and some w^j

  By Prop 0.10 and Prop 0.2, implied write orders between the two events can be of the form

  $$w^j_k \xrightarrow[iwo]{} w^j$$
  $$w^j \xrightarrow[iwo]{} w^j_k$$

  For the first case, by Def 2 and Prop 0.3, we have

  $$w^j_i \xrightarrow[smo]{} w^j$$

  By Prop 0.5, on swapping T_i and T_k, we have

  $$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j_i \xrightarrow[iwo]{} w^j$$

  The above relations do not violate Prop 0.3, hence we can end this case here.

  For the second case, by Prop 0.5, on swapping, we have

  $$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j \xrightarrow[iwo]{} w^j_i$$

  The new relation $w^j \xrightarrow[iwo]{} w^j_i$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

  1. Case where $x \xrightarrow[iwo]{} w^j \ \wedge \ w^j_i \xrightarrow[iwo]{} x$
     Though such an execution graph can exist, it has $po \cup rf$ as a cycle, thus violating coherence.
  2. Case where $w^j \xrightarrow[iwo]{} x \ \wedge \ w^j_i \xrightarrow[iwo]{} x$
     By Prop 0.10, x is invalid and thus such an execution graph cannot exist.
  3. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ w^j_i \xrightarrow[iwo]{} x$

- There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2 and Prop 0.9, either x' or x are invalid.
- There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. Such an execution graph has $po \cup rf$ cycle, thus violating coherence.

4. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ w_i^j \xrightarrow[iwo]{} x$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$.
     The only way to get a configuration with $w^j \xrightarrow[iwo]{} w_i^j$ before would be to swap T_x and T' first, followed by swapping T' and T_i. But this swapping order is not possible, as the "fixing" process we have is to fix implied write orders above read first and then below. The de-swapping here can only be implied write orders below read and then above read. Since this is not possible in the above configuration, we can conclude that $w^j \xrightarrow[iwo]{} w_i^j$ did not exist before and hence could not have been fixed.
     The reader should note that though it was possible for the correct relation $w^j \xrightarrow[iwo]{} w_k^j$ to exist, it wasn't something that was "fixed" by swapping. This is different from those implied write orders which were fixed by swapping thread identities. (Discuss with Viktor on this)
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.2 and Prop 0.10, either x' or x are invalid.

We still will have another case for any new implied write order introduced, (Consider this cases in the end. We need to divide it into two cases.)

4. Write orders below and above read fixed , write orders above need fixing. Let us assume that the fixed set **below the read is the set of writes** w^j. Let us consider two threads T_i and T_k whose writes **above the read** w_i^l **and** w_k^l is going to be fixed. Without loss of generality, let us conisder the symmetric memory order between these two threads to be of the form $w_i \xrightarrow[smo]{} w_k$. By Prop 0.7, we then have the configuration as:

$$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^l \xrightarrow[iwo]{} w_i^l$$

We divide our analysis into two cases

- There is an implied write order with w_i^j and some w^j By Prop 0.10 and Prop 0.1, implied write orders between the two events can be of the form

$$w^j \xrightarrow[iwo]{} w_i^j$$
$$w_i^j \xrightarrow[iwo]{} w^j$$

For the first case by Def 2 and Prop 0.3, we have

$$w^j \xrightarrow[smo]{} w^j_k$$

By Prop 0.5, on swapping T_i and T_k, we have

$$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j \xrightarrow[iwo]{} w^j_k$$

The above relations do not violate Prop 0.3, hence we can end this case here.

For the second case, by Prop 0.5, on swapping, we have

$$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j_k \xrightarrow[iwo]{} w^j$$

The new relation $w^j_k \xrightarrow[iwo]{} w^j$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. For this, we consdier another event x, and the following cases:

1. Case where $w^j \xrightarrow[iwo]{} x \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   By Prop 0.10, x is invalid and thus such an execution graph cannot exist.
2. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.2, Prop 0.1 and Prop 0.9, either w' or w^j are invalid.
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$. By Prop 0.2 and Prop 0.9, either x' or x are invalid.
3. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ w^j_k \xrightarrow[iwo]{} x$
   - There could exist events w' and x' above the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2, Prop 0.1 and Prop 0.9, either x' or x are invalid.
   - There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$. By Prop 0.2 and Prop 0.9, either w' or w^j are invalid.

- There is an implied write order with w^j_k and some w^j By Prop 0.10 and Prop 0.1, implied write orders between the two events can be of the form

$$w^j_k \xrightarrow[iwo]{} w^j$$

For the first case by Def 2 and Prop 0.3, we have

$$w_i^j \xrightarrow[smo]{} w^j$$

By Prop 0.5, on swapping T_i and T_k, we have

$$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_i^j \xrightarrow[iwo]{} w^j$$

The above relations do not violate Prop 0.3, hence we can end this case here.

Let us assume that the fixed set **above the read is the set of writes** w^j. Let us consider two threads T_i and T_k whose writes **above the read** w_i^l **and** w_k^l is going to be fixed. Without loss of generality, let us conisder the symmetric memory order between these two threads to be of the form $w_i \xrightarrow[smo]{} w_k$. By Prop 0.7, we then have the configuration as:

$$w_i^j \xrightarrow[smo]{} w_k^j \ \wedge \ w_k^l \xrightarrow[iwo]{} w_i^l$$

We divide our analysis into two cases

- There is an implied write order with w_i^j and some w^j

  By Prop 0.9 and Prop 0.1, implied write orders between the two events can be of the form

  $$w^j \xrightarrow[iwo]{} w_i^j$$
  $$w_i^j \xrightarrow[iwo]{} w^j$$

  For the first case by Def 2 and Prop 0.3, we have

  $$w^j \xrightarrow[smo]{} w_k^j$$

  By Prop 0.5, on swapping T_i and T_k, we have

  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w^j \xrightarrow[iwo]{} w_k^j$$

  The above relations do not violate Prop 0.3, hence we can end this case here.

  For the second case, by Prop 0.5, on swapping, we have

  $$w_i^l \xrightarrow[iwo]{} w_k^l \ \wedge \ w_k^j \xrightarrow[iwo]{} w^j$$

  The new relation $w_k^j \xrightarrow[iwo]{} w^j$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. Since most of these cases have already been covered, we will only address those cases which aren't; i.e. those where certain implied write orders between writes below the read have been "fixed". For this, we consdier another event x, and the following cases:

1. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$
   The only way to get a configuration with $w^j_k \xrightarrow[iwo]{} w^j$ before would be to swap T_x and T' first, followed by swapping T' and T_i. On swapping T_x and T', we have the following configuration

   $$x' \xrightarrow[iwo]{} w' \ \wedge \ w^l_k \xrightarrow[iwo]{} w^l_i \ \wedge \ w^j \xrightarrow[iwo]{} w^j_k \ \wedge \ x \xrightarrow[iwo]{} w^j_i$$

   If this was the original configuration of execution, we can note that $w^l_k \xrightarrow[iwo]{} w^l_i$ would be fixed first as it is between writes above the read, in contrast to $x' \xrightarrow[iwo]{} w'$. Hence, it is not possible to have gotten our intended execution graph from this configuration. Thus, we can conclude that $w^j_k \xrightarrow[iwo]{} w^j$ could not have been fixed before.
2. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_k$
   There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_k, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$ By Prop 0.10 and Prop 0.2, either w' or w^j is invalid.

- There is an implied write order with w^j_k and some w^j

  By Prop 0.9 and Prop 0.1, implied write orders between the two events can be of the form

  $$w^j \xrightarrow[iwo]{} w^j_k$$

  By Prop 0.5, on swapping, we have

  $$w^l_i \xrightarrow[iwo]{} w^l_k \ \wedge \ w^j \xrightarrow[iwo]{} w^j_i$$

  The new relation $w^j \xrightarrow[iwo]{} w^j_i$ may be wrong. To show that this relation was not fixed before, we consider cases where it could have been fixed. We then show that such a configuration cannot exist. Since most of these cases have already been covered, we will only address those cases which aren't; i.e. those where certain implied write orders between writes below the read have been "fixed". For this, we consdier another event x, and the following cases:

  1. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ w^j_i \xrightarrow[iwo]{} x$
     There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w^j_i, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$
     The only way to get a configuration with $w^j \xrightarrow[iwo]{} w^j_i$ before would be to swap T_x and T' first, followed by swapping T' and T_i. On swapping T_x and T', we have the following configuration

     $$x' \xrightarrow[iwo]{} w' \ \wedge \ w^l_k \xrightarrow[iwo]{} w^l_i \ \wedge \ w^j_i \xrightarrow[iwo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w^j_i$$

If this was the original configuration of execution, we can note that $w_k^l \xrightarrow[iwo]{} w_i^l$ would be fixed first as it is between writes above the read, in contrast to $x' \xrightarrow[iwo]{} w'$ which is between writes below the read. Hence, it is not possible to have gotten our intended execution graph from this configuration. Thus, we can conclude that $w^j \xrightarrow[iwo]{} w_i^j$ could not have been fixed before.

2. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ w_i^j \xrightarrow[iwo]{} x$

   There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$ By Prop 0.10 and Prop 0.2, either w' or w^j is invalid.

3. Case where $w^j \xrightarrow[smo]{} x \ \wedge \ x \xrightarrow[iwo]{} w_i^j$

   There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $w' \xrightarrow[iwo]{} x'$ By Prop 0.10 and Prop 0.2, either x' or x is invalid.

4. Case where $x \xrightarrow[smo]{} w^j \ \wedge \ x \xrightarrow[iwo]{} w_i^j$

   There could exist events w' and x' below the read program ordered with w^j and x resp, with an implied write order. This could have been fixed before, in an execution graph having an implied write order between w^j and w_i^j, which could have also been fixed. By Def 2 and Prop 0.3, we have $x' \xrightarrow[iwo]{} w'$ By Prop 0.10 and Prop 0.2, either w' or w^j is invalid.

In summary, we can conclude that new implied write orders among other set of writes introduced while fixing an implied write order, if wrong cannot have been resolved earlier, and thus, by Lemma **??** can be fixed and will remain fixed.

□