Week 9

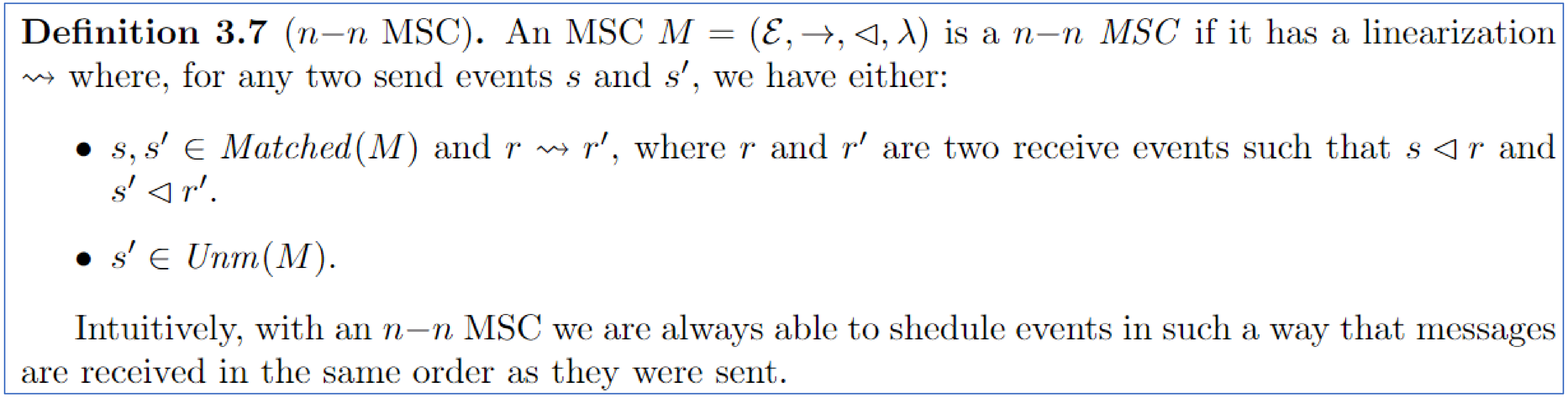
Overview

- $1 - n$ MSCs:
  - MSO-definability and STW-boundness does not imply decidability of synchronizability, because the preliminary results given for p2p and mailbox do not translate to 1-n ☹
  - Rewritten preliminary results for 1-n
- MSO-definability of n-n!

informally

MSO-definability of n-n

$n - n$ MSC - definition

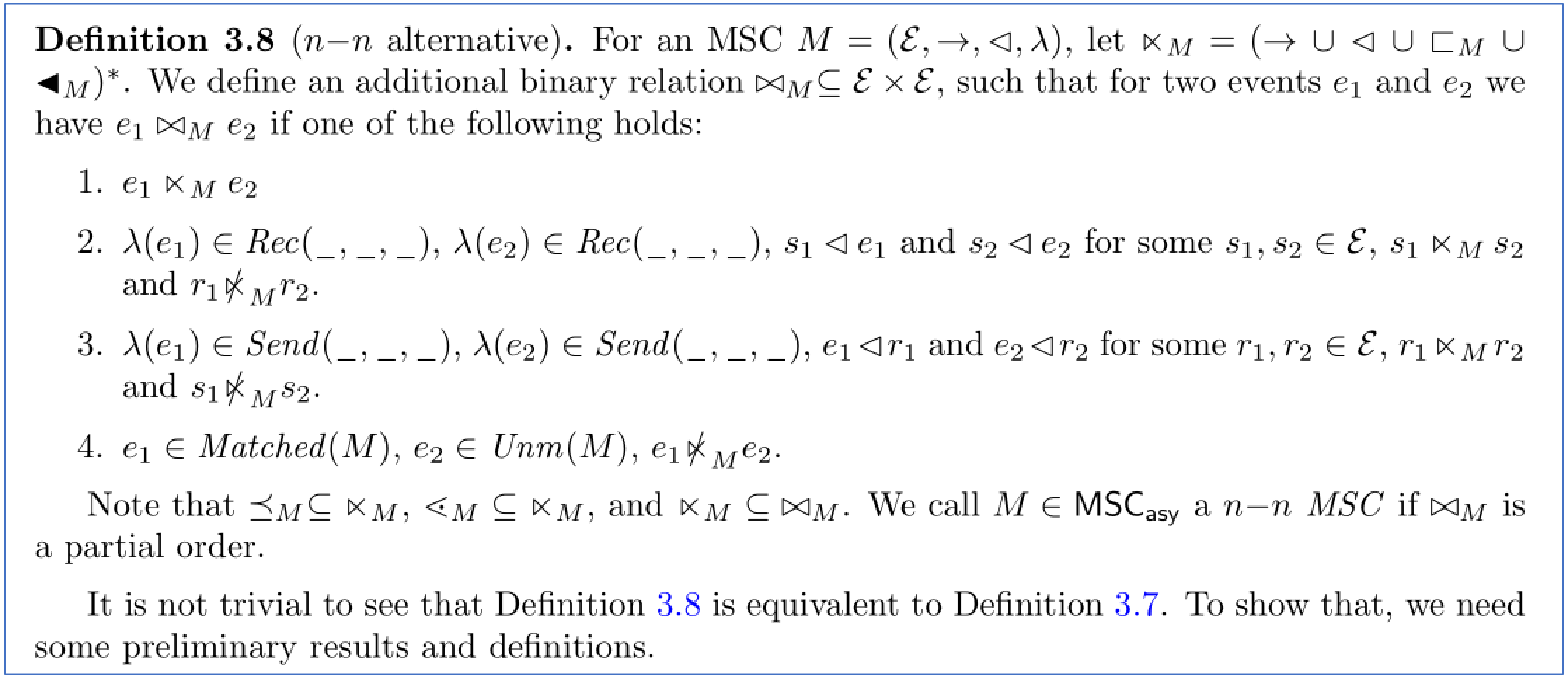


Definition 3.7 (n–n MSC)**.** An MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$ is a n–n *MSC* if it has a linearization $\rightsquigarrow$ where, for any two send events s and s', we have either:

- $s, s' \in \mathit{Matched}(M)$ and $r \rightsquigarrow r'$, where r and r' are two receive events such that $s \lhd r$ and $s' \lhd r'$.
- $s' \in \mathit{Unm}(M)$.

Intuitively, with an n–n MSC we are always able to shedule events in such a way that messages are received in the same order as they were sent.

$n - n$ MSC – alternative definition



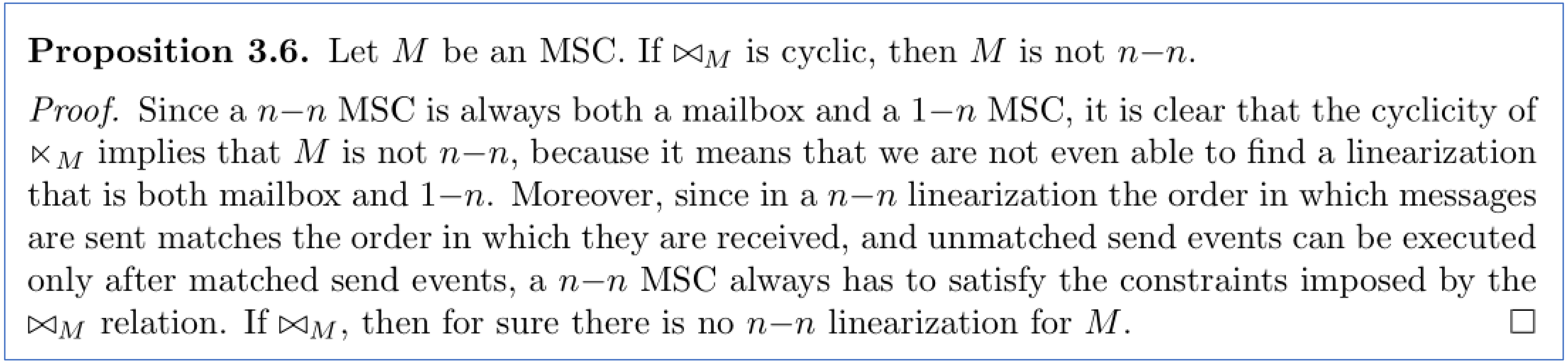
Definition 3.8 ($n{-}n$ alternative)**.** For an MSC $M = (\mathcal{E}, \rightarrow, \lhd, \lambda)$, let $\ltimes_M = (\rightarrow \cup \lhd \cup \sqsubset_M \cup \blacktriangleleft_M)^*$. We define an additional binary relation $\bowtie_M \subseteq \mathcal{E} \times \mathcal{E}$, such that for two events e_1 and e_2 we have $e_1 \bowtie_M e_2$ if one of the following holds:

1. $e_1 \ltimes_M e_2$
2. $\lambda(e_1) \in \mathit{Rec}(\_, \_, \_)$, $\lambda(e_2) \in \mathit{Rec}(\_, \_, \_)$, $s_1 \lhd e_1$ and $s_2 \lhd e_2$ for some $s_1, s_2 \in \mathcal{E}$, $s_1 \ltimes_M s_2$ and $r_1 \not\ltimes_M r_2$.
3. $\lambda(e_1) \in \mathit{Send}(\_, \_, \_)$, $\lambda(e_2) \in \mathit{Send}(\_, \_, \_)$, $e_1 \lhd r_1$ and $e_2 \lhd r_2$ for some $r_1, r_2 \in \mathcal{E}$, $r_1 \ltimes_M r_2$ and $s_1 \not\ltimes_M s_2$.
4. $e_1 \in \mathit{Matched}(M)$, $e_2 \in \mathit{Unm}(M)$, $e_1 \not\ltimes_M e_2$.

Note that $\preceq_M \subseteq \ltimes_M$, $\lessdot_M \subseteq \ltimes_M$, and $\ltimes_M \subseteq \bowtie_M$. We call $M \in \mathsf{MSC}_{\mathsf{asy}}$ a *$n{-}n$ MSC* if $\bowtie_M$ is a partial order.

It is not trivial to see that Definition 3.8 is equivalent to Definition 3.7. To show that, we need some preliminary results and definitions.

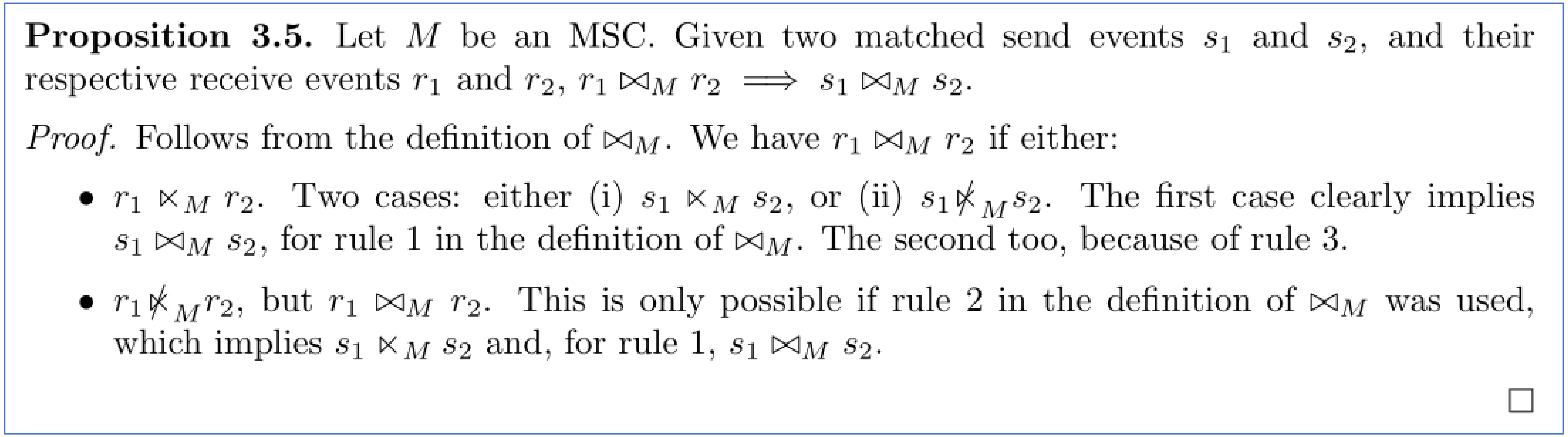
$n - n$ MSC



Proposition 3.6. Let M be an MSC. If $\bowtie_M$ is cyclic, then M is not $n{-}n$.

Proof. Since a $n{-}n$ MSC is always both a mailbox and a $1{-}n$ MSC, it is clear that the cyclicity of $\ltimes_M$ implies that M is not $n{-}n$, because it means that we are not even able to find a linearization that is both mailbox and $1{-}n$. Moreover, since in a $n{-}n$ linearization the order in which messages are sent matches the order in which they are received, and unmatched send events can be executed only after matched send events, a $n{-}n$ MSC always has to satisfy the constraints imposed by the $\bowtie_M$ relation. If $\bowtie_M$, then for sure there is no $n{-}n$ linearization for M. $\square$

$n - n$ MSC



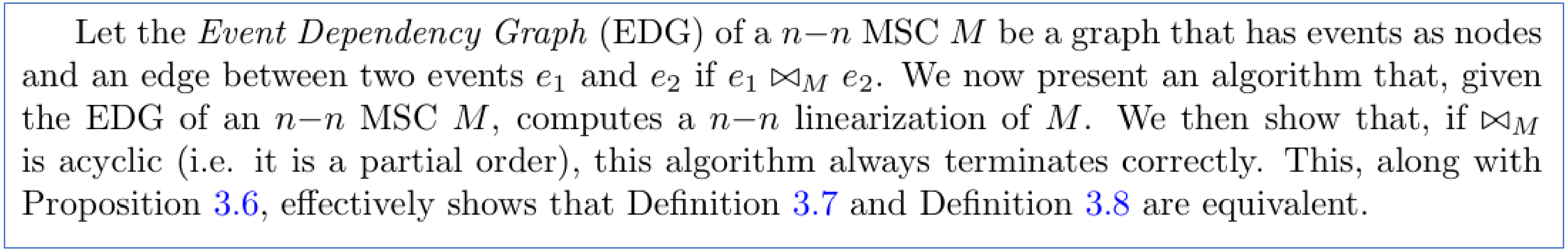
Proposition 3.5. Let M be an MSC. Given two matched send events s_1 and s_2, and their respective receive events r_1 and r_2, $r_1 \bowtie_M r_2 \implies s_1 \bowtie_M s_2$.

Proof. Follows from the definition of $\bowtie_M$. We have $r_1 \bowtie_M r_2$ if either:

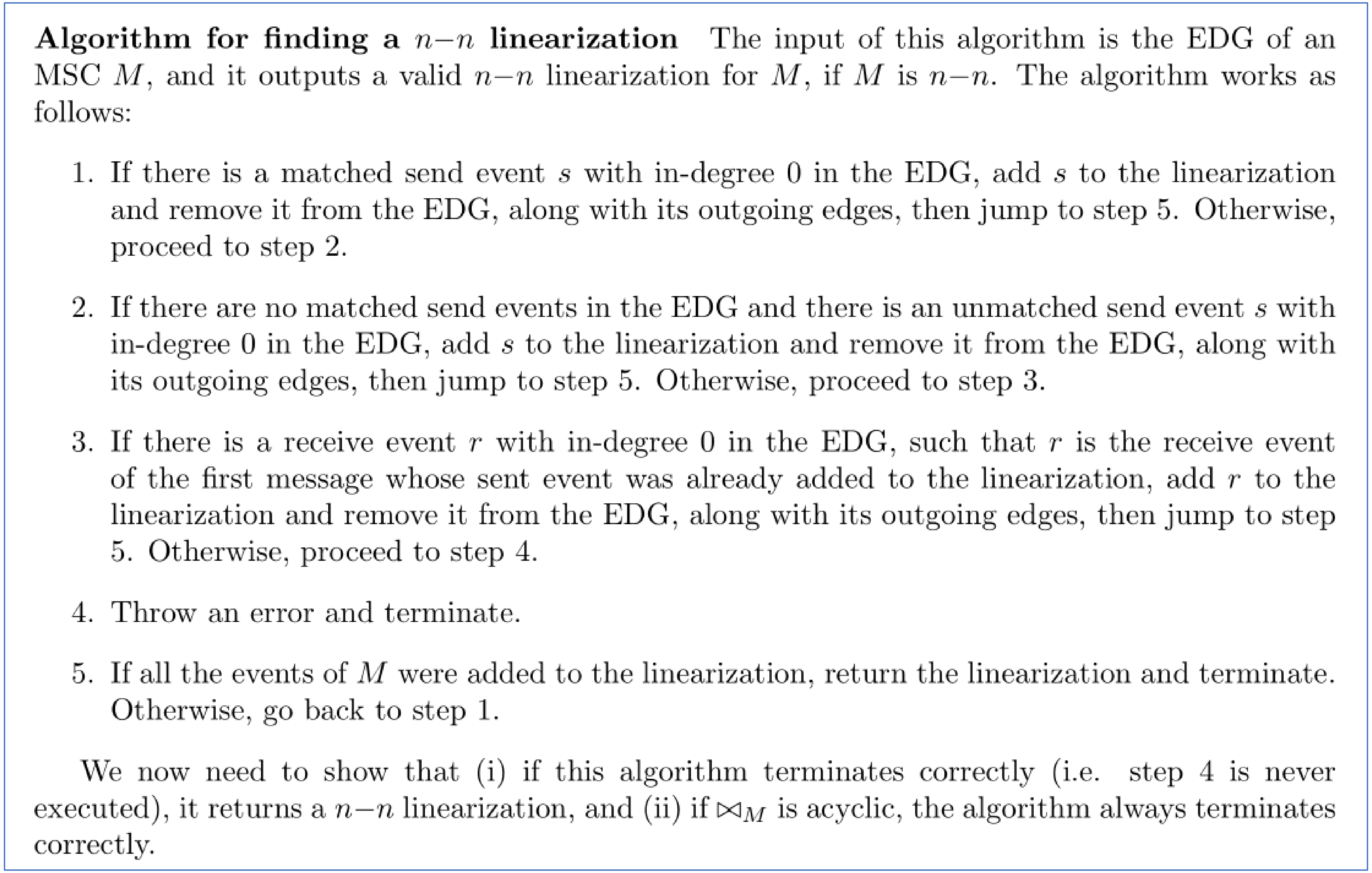
- $r_1 \ltimes_M r_2$. Two cases: either (i) $s_1 \ltimes_M s_2$, or (ii) $s_1 \not\ltimes_M s_2$. The first case clearly implies $s_1 \bowtie_M s_2$, for rule 1 in the definition of $\bowtie_M$. The second too, because of rule 3.
- $r_1 \not\ltimes_M r_2$, but $r_1 \bowtie_M r_2$. This is only possible if rule 2 in the definition of $\bowtie_M$ was used, which implies $s_1 \ltimes_M s_2$ and, for rule 1, $s_1 \bowtie_M s_2$.

$\square$

$n - n$ MSC - EDG



Let the *Event Dependency Graph* (EDG) of a $n{-}n$ MSC M be a graph that has events as nodes and an edge between two events e_1 and e_2 if $e_1 \bowtie_M e_2$. We now present an algorithm that, given the EDG of an $n{-}n$ MSC M, computes a $n{-}n$ linearization of M. We then show that, if $\bowtie_M$ is acyclic (i.e. it is a partial order), this algorithm always terminates correctly. This, along with Proposition 3.6, effectively shows that Definition 3.7 and Definition 3.8 are equivalent.



Algorithm for finding a $n{-}n$ linearization The input of this algorithm is the EDG of an MSC M, and it outputs a valid $n{-}n$ linearization for M, if M is $n{-}n$. The algorithm works as follows:

1. If there is a matched send event s with in-degree 0 in the EDG, add s to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 2.

2. If there are no matched send events in the EDG and there is an unmatched send event s with in-degree 0 in the EDG, add s to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 3.

3. If there is a receive event r with in-degree 0 in the EDG, such that r is the receive event of the first message whose sent event was already added to the linearization, add r to the linearization and remove it from the EDG, along with its outgoing edges, then jump to step 5. Otherwise, proceed to step 4.

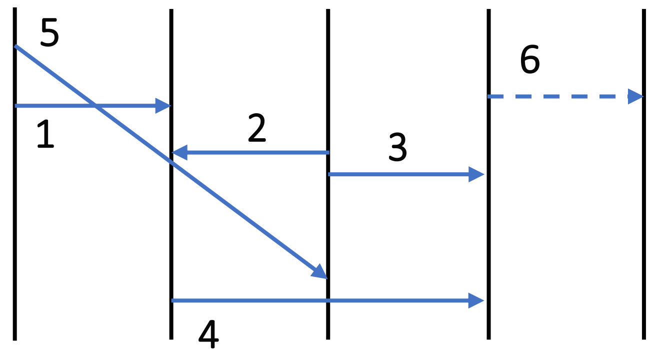
4. Throw an error and terminate.

5. If all the events of M were added to the linearization, return the linearization and terminate. Otherwise, go back to step 1.

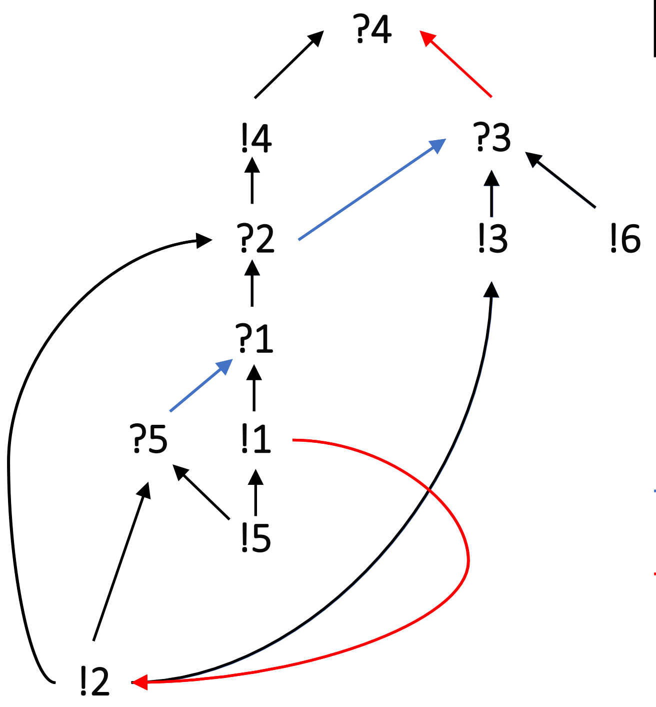
We now need to show that (i) if this algorithm terminates correctly (i.e. step 4 is never executed), it returns a $n{-}n$ linearization, and (ii) if $\bowtie_M$ is acyclic, the algorithm always terminates correctly.

Algorithm for $n - n$

[*] arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity



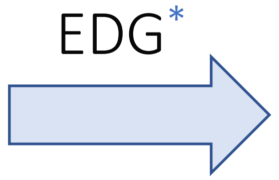
This is a n-n MSC

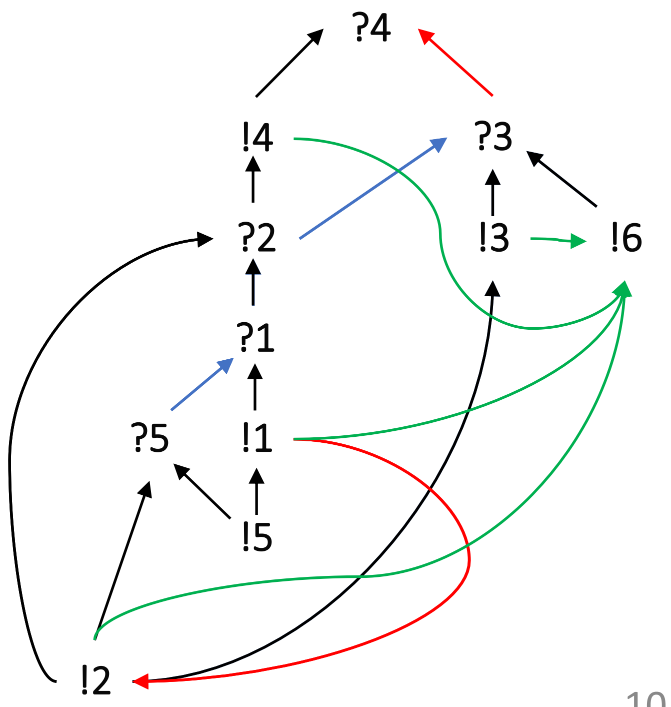


$\longrightarrow$ = ($\rightarrow$ / $\lhd$)

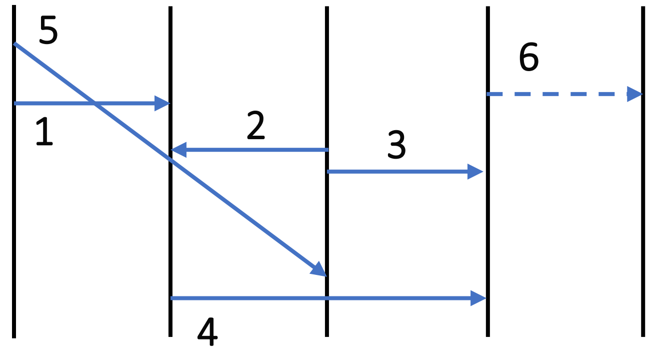
$\longrightarrow$ = ($\blacktriangleleft$)

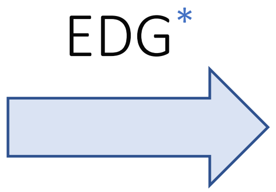
$\longrightarrow$ = ($\sqsubset$)

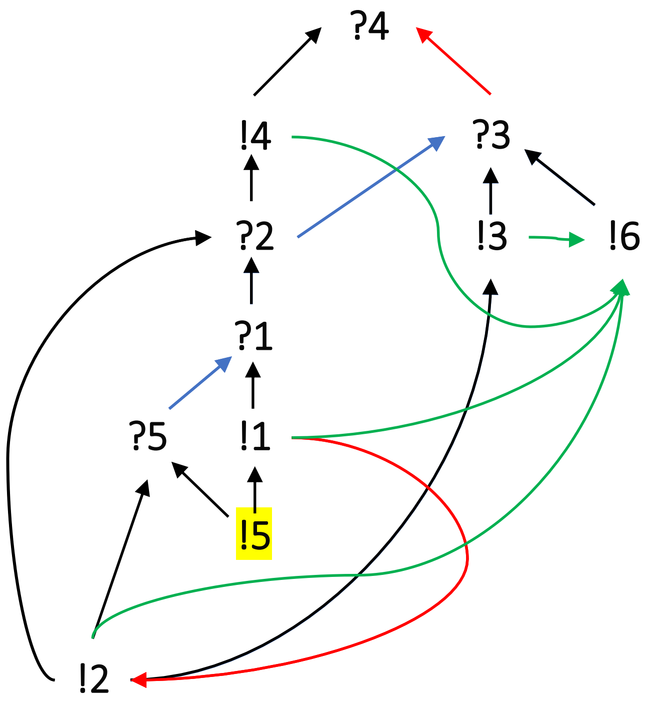




Algorithm for $n - n$





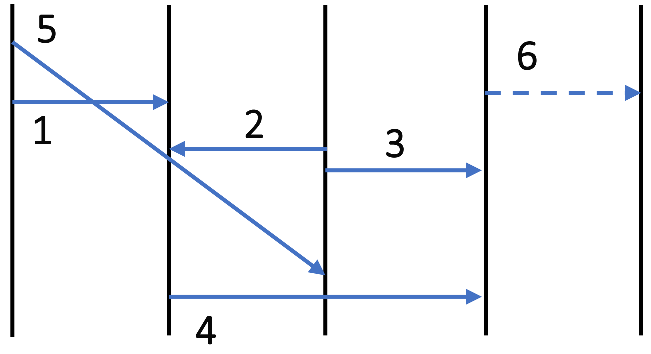


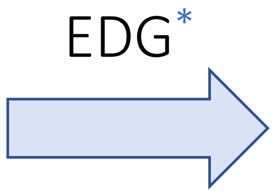
Rule 1

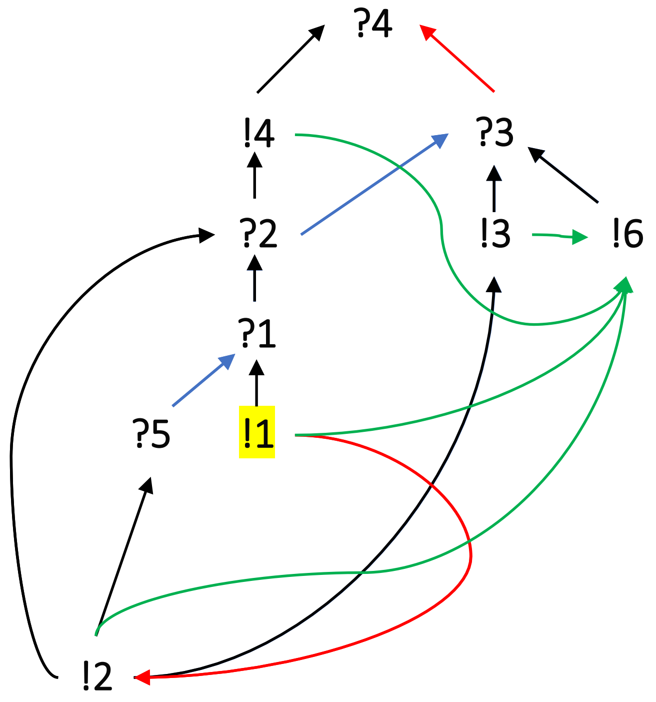
Linearization: $!\mathbf{5}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





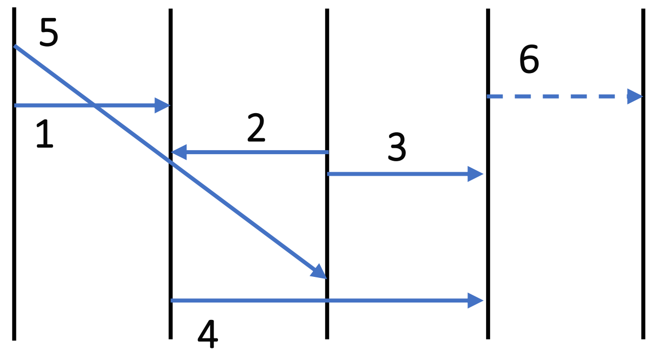


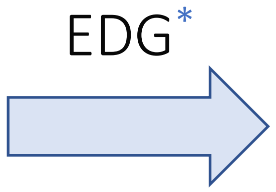
Rule 1

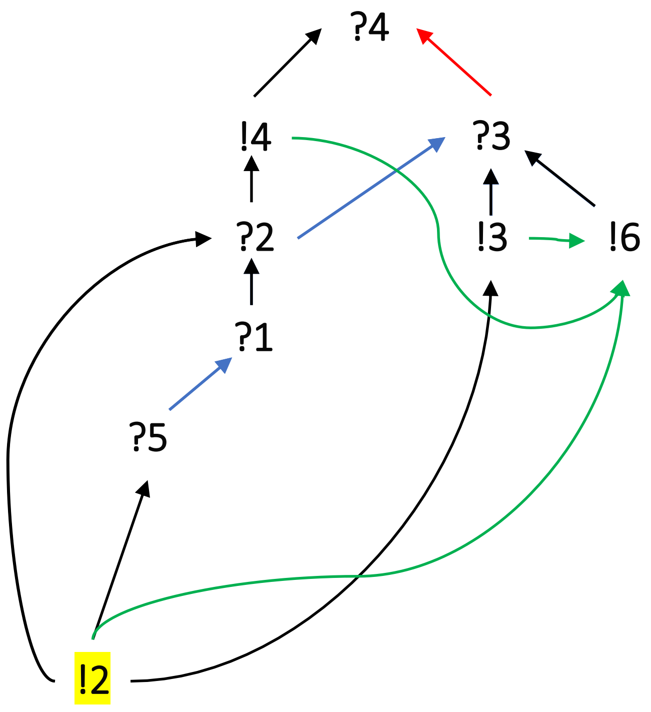
Linearization: $!5$ $\mathbf{!1}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





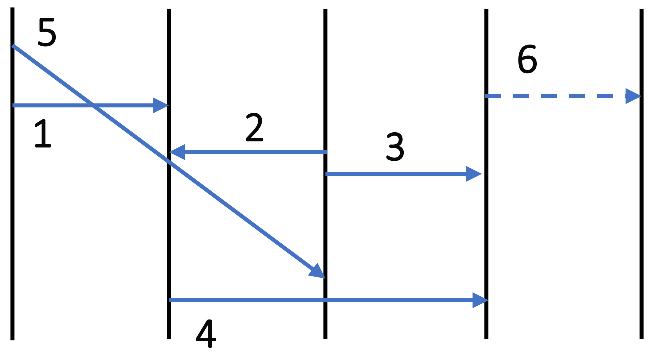


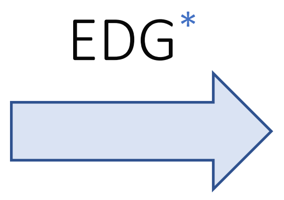
Rule 1

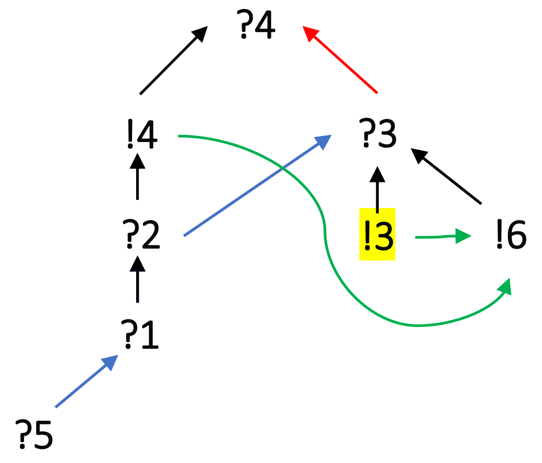
Linearization: ! 5 ! 1 **! 2**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





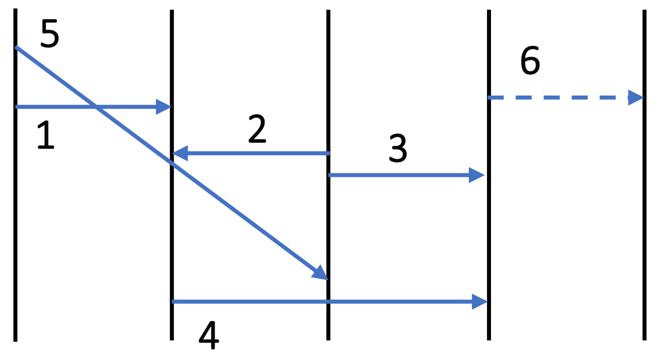


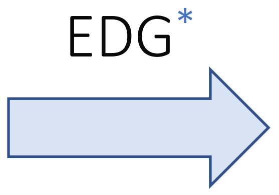
Rule 1

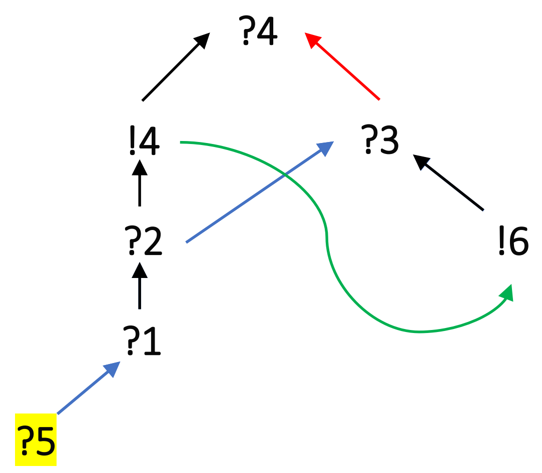
Linearization: $!5\ !1\ !2\ \mathbf{!3}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





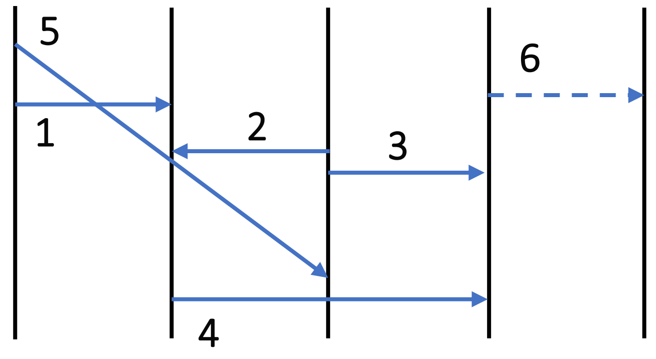


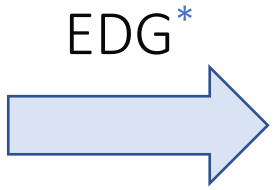
Rule 3

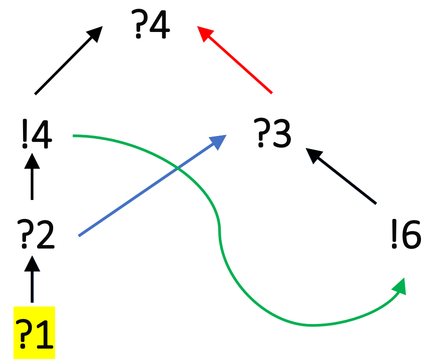
Linearization: !5 !1 !2 !3 **?5**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





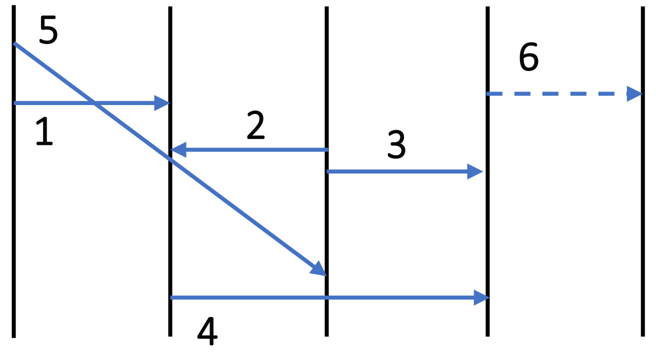


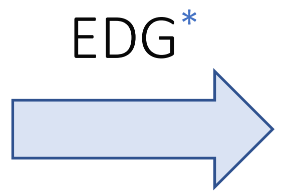
Rule 3

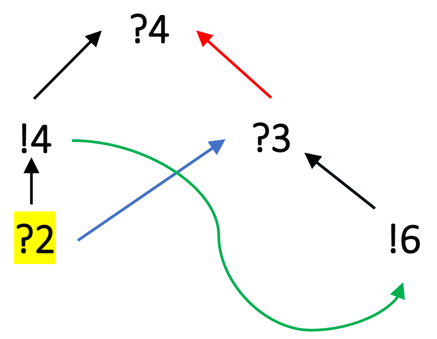
Linearization: $!5\ !1\ !2\ !3\ ?5\ \mathbf{?1}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





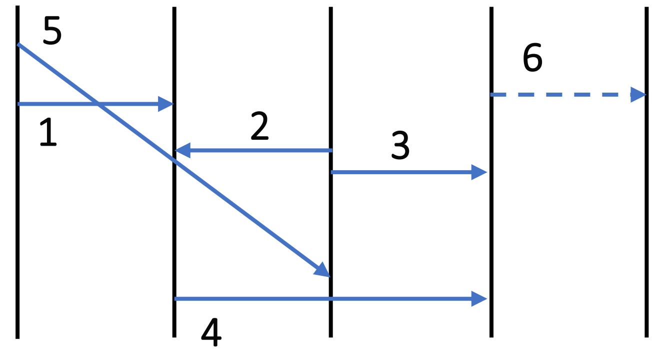


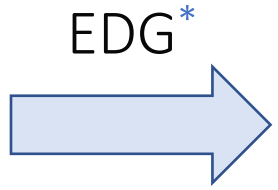
Rule 3

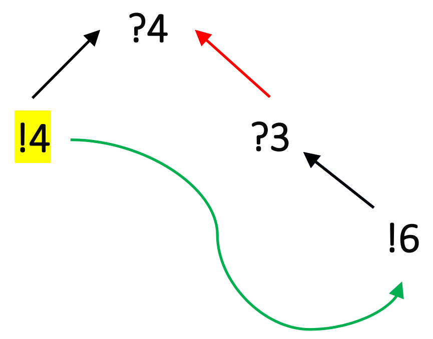
Linearization: $!5 \; !1 \; !2 \; !3 \; ?5 \; ?1 \; \mathbf{?2}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





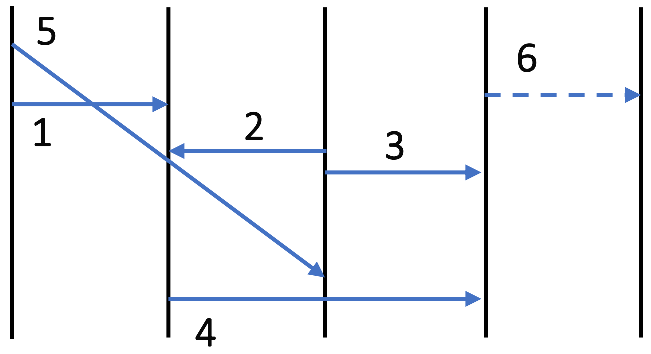


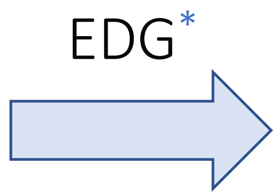
Rule 1

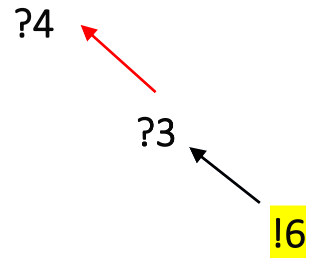
Linearization: $!5 \; !1 \; !2 \; !3 \; ?5 \; ?1 \; ?2 \; \mathbf{!4}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





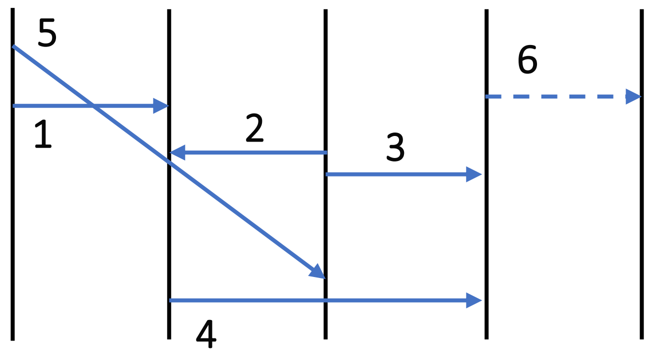


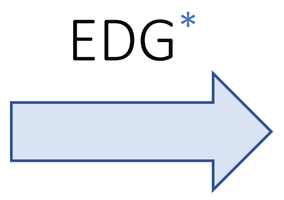
Rule 2

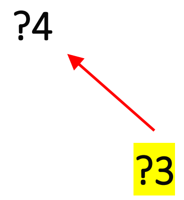
Linearization: !5 !1 !2 !3 ?5 ?1 ?2 !4 **!6**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





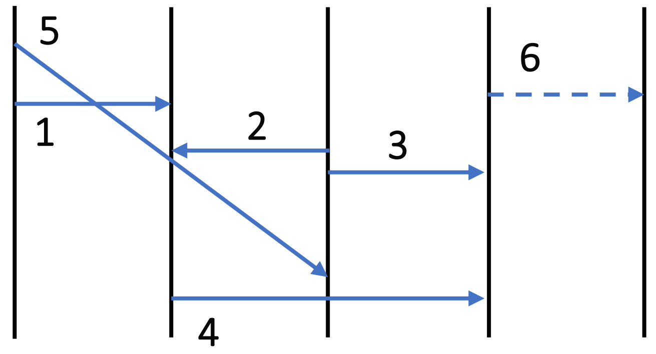


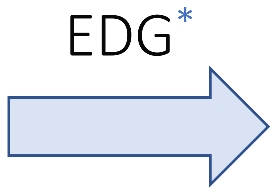
Rule 3

Linearization: $!5 \ !1 \ !2 \ !3 \ ?5 \ ?1 \ ?2 \ !4 \ !6 \ \mathbf{?3}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





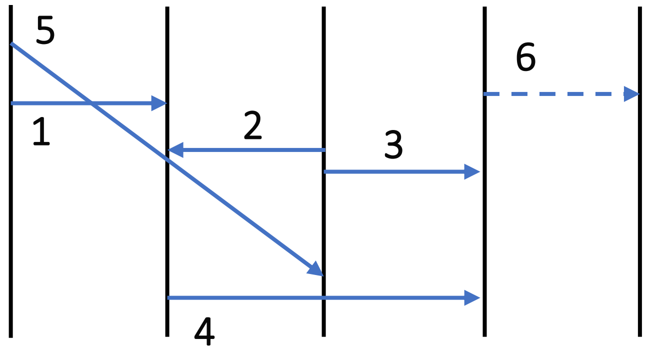
?4

Rule 3

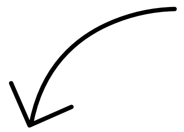
Linearization: !5 !1 !2 !3 ?5 ?1 ?2 !4 !6 ?3 **?4**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$



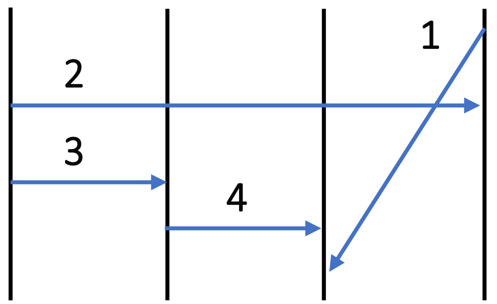
This is a n-n linearization ☺



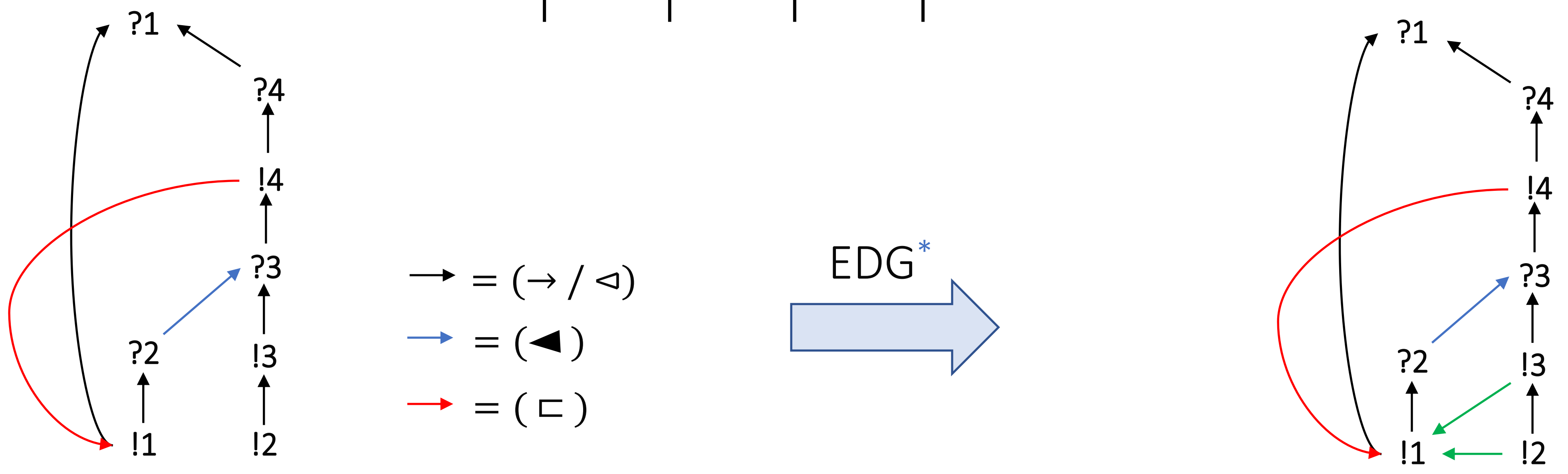
Linearization: $!5\ !1\ !2\ !3\ ?5\ ?1\ ?2\ !4\ !6\ ?3\ ?4$

Algorithm for $n - n$

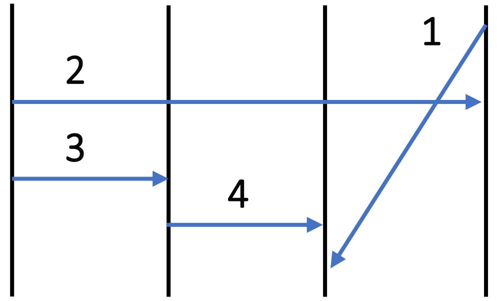
* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

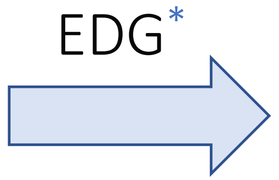


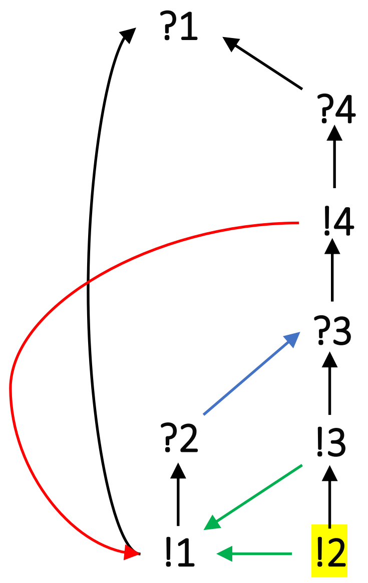
This is **not** a n-n MSC



Algorithm for $n - n$





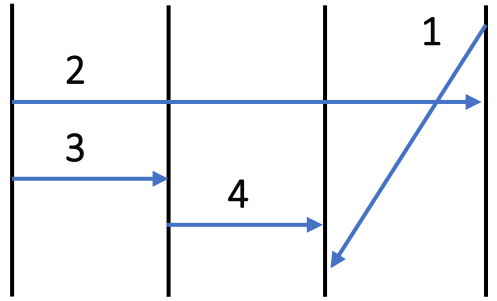


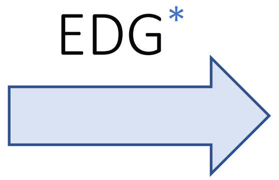
Rule 1

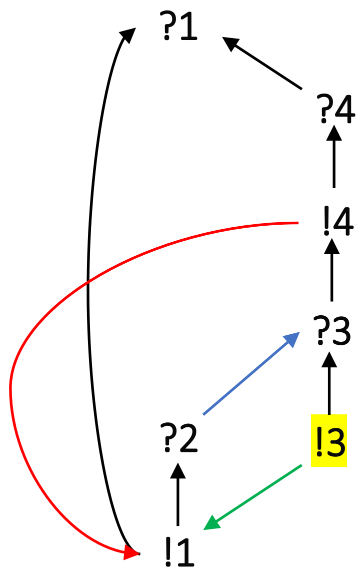
Linearization: $!\mathbf{2}$

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





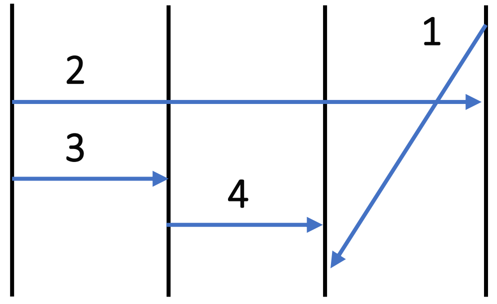


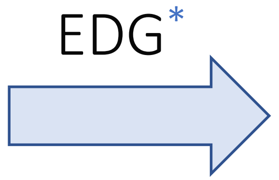
Rule 1

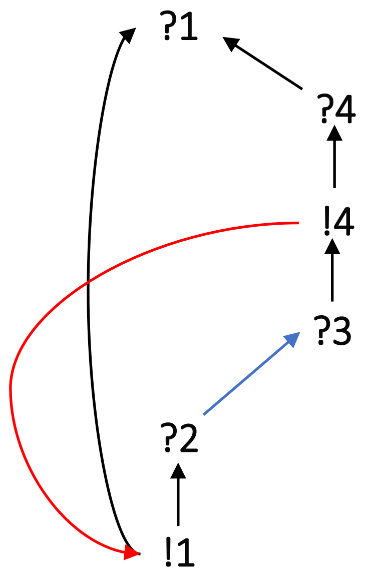
Linearization: !2 **!3**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

Algorithm for $n - n$





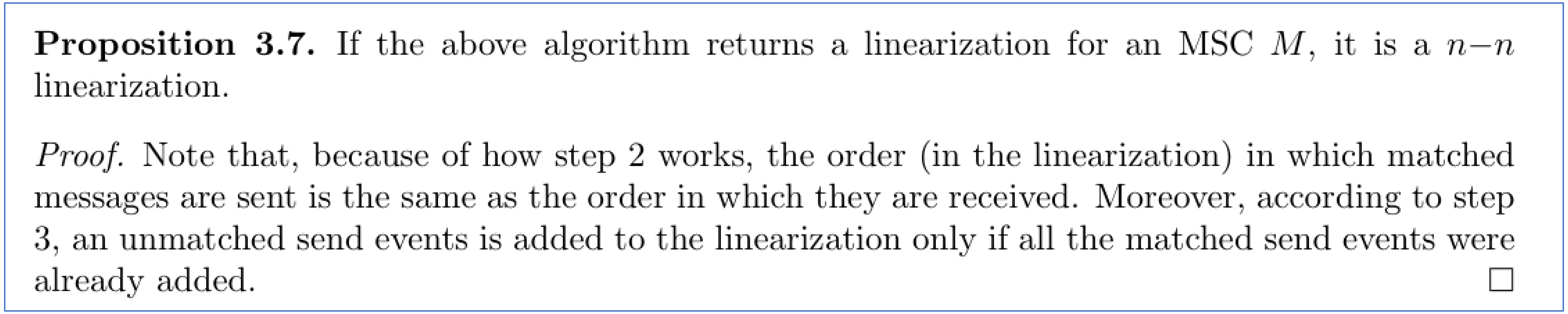


Linearization: !2 !3

Rules 1,2,3 cannot be applied →
Rule 4, **ERROR**

* arrows for $\ltimes = (\rightarrow \cup \lhd \cup \blacktriangleleft \cup \sqsubset)^*$ are not all shown for the sake of clarity

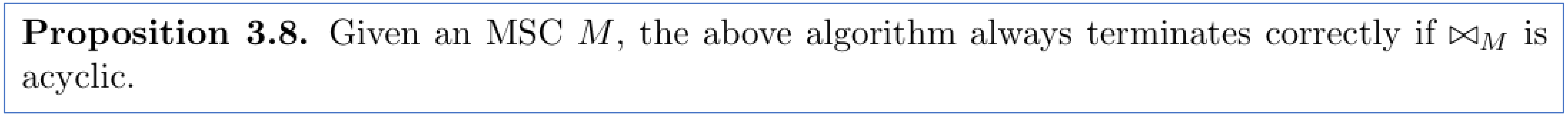
Algorithm for $n - n$ - correctness



Proposition 3.7. If the above algorithm returns a linearization for an MSC M, it is a $n{-}n$ linearization.

Proof. Note that, because of how step 2 works, the order (in the linearization) in which matched messages are sent is the same as the order in which they are received. Moreover, according to step 3, an unmatched send events is added to the linearization only if all the matched send events were already added. $\square$

Algorithm for $n - n$ - completeness

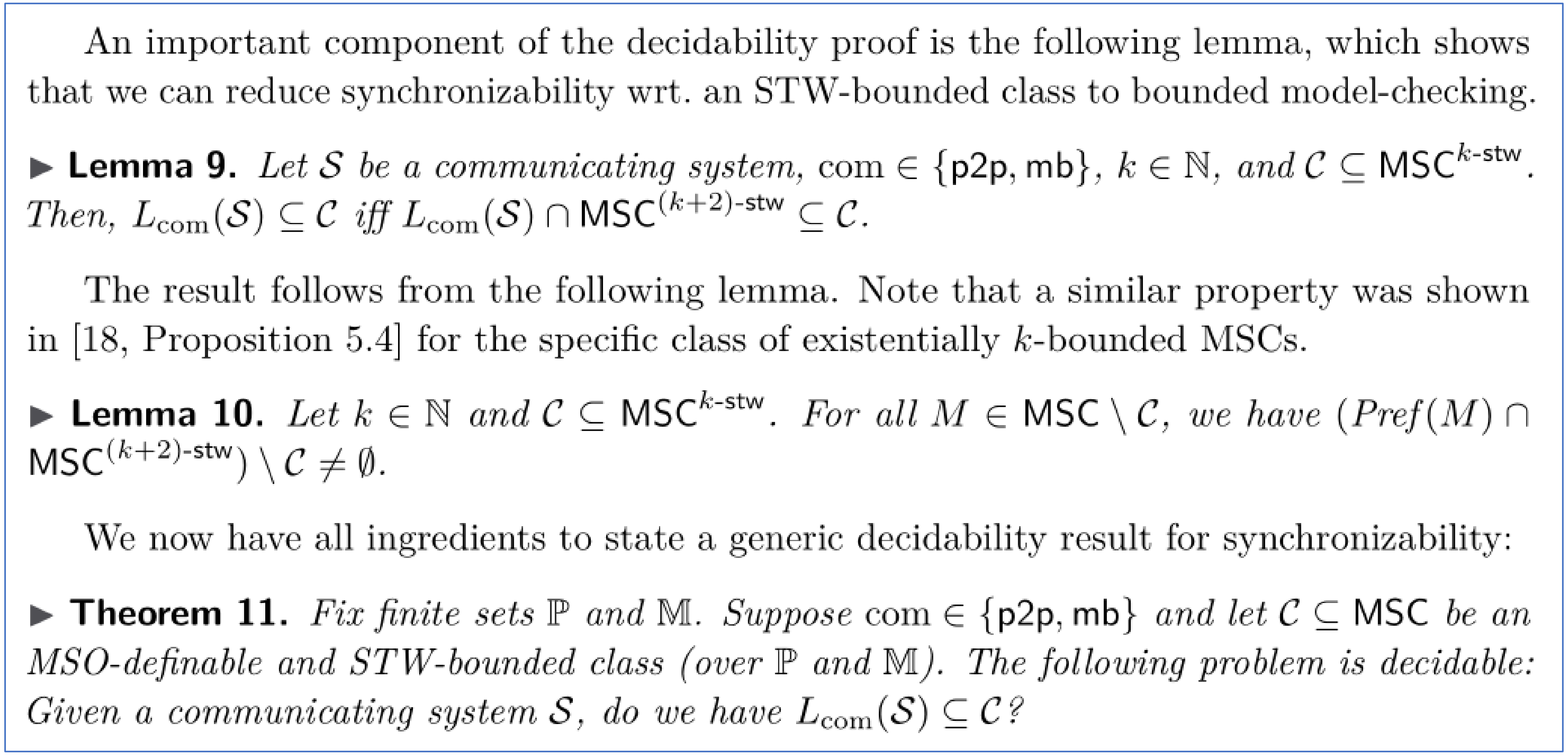


Proposition 3.8. Given an MSC M, the above algorithm always terminates correctly if $\bowtie_M$ is acyclic.

Proof in the report! Too long…

1-n

Preliminary results



An important component of the decidability proof is the following lemma, which shows that we can reduce synchronizability wrt. an STW-bounded class to bounded model-checking.

▶ **Lemma 9.** *Let $\mathcal{S}$ be a communicating system,* com $\in \{\mathsf{p2p}, \mathsf{mb}\}$*, $k \in \mathbb{N}$, and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. Then, $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ iff $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.*

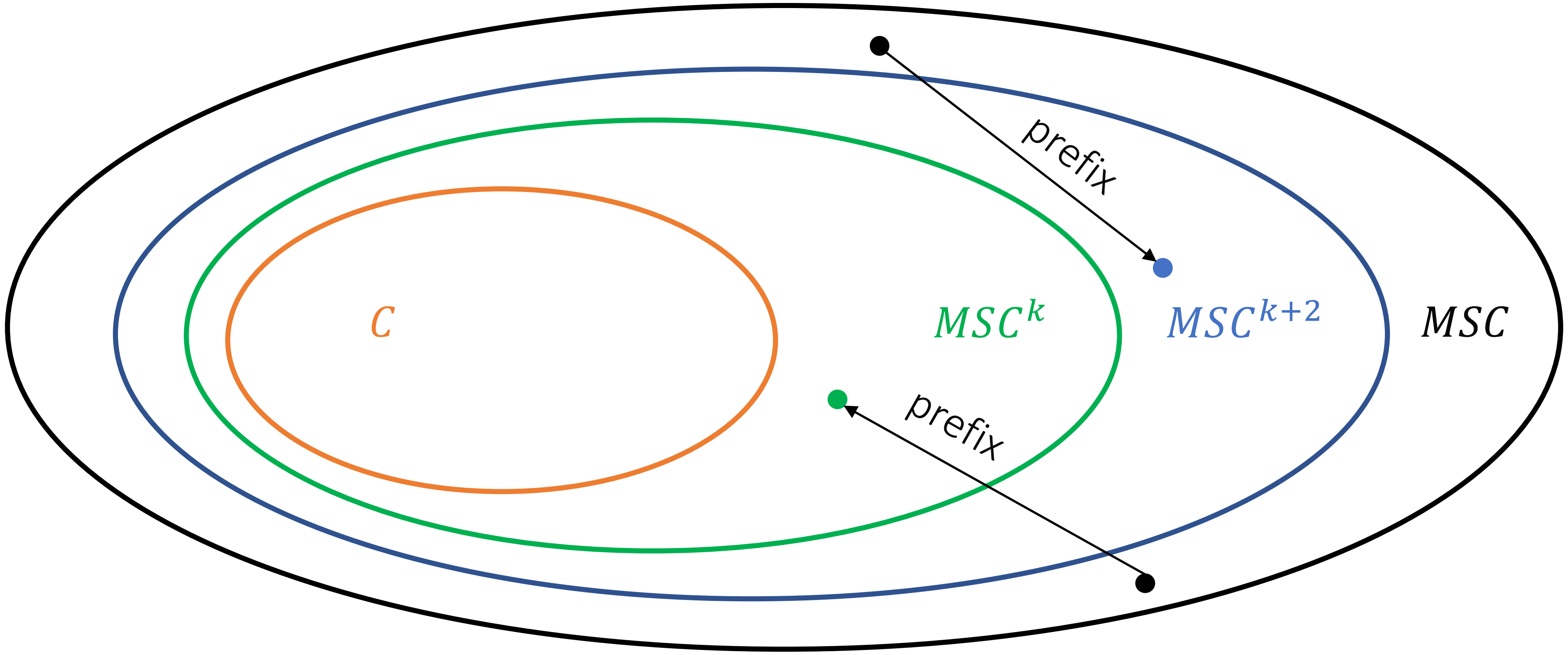
The result follows from the following lemma. Note that a similar property was shown in [18, Proposition 5.4] for the specific class of existentially k-bounded MSCs.

▶ **Lemma 10.** *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC} \setminus \mathcal{C}$, we have $(\mathit{Pref}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.*

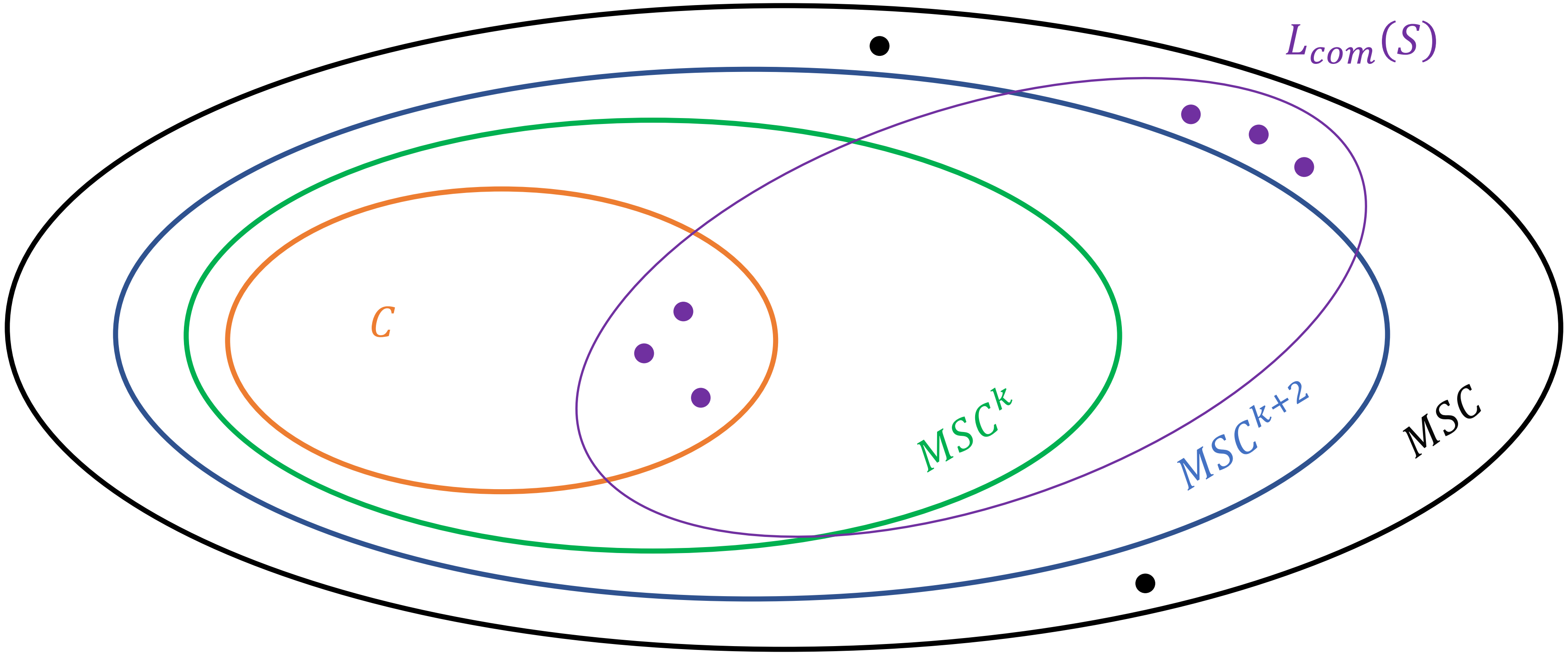
We now have all ingredients to state a generic decidability result for synchronizability:

▶ **Theorem 11.** *Fix finite sets $\mathbb{P}$ and $\mathbb{M}$. Suppose* com $\in \{\mathsf{p2p}, \mathsf{mb}\}$ *and let $\mathcal{C} \subseteq \mathsf{MSC}$ be an MSO-definable and STW-bounded class (over $\mathbb{P}$ and $\mathbb{M}$). The following problem is decidable: Given a communicating system $\mathcal{S}$, do we have $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$?*

▶ **Lemma 10.** *Let $k \in \mathbb{N}$ and $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. For all $M \in \mathsf{MSC} \setminus \mathcal{C}$, we have* $(\mathit{Pref}(M) \cap \mathsf{MSC}^{(k+2)\text{-stw}}) \setminus \mathcal{C} \neq \emptyset$.

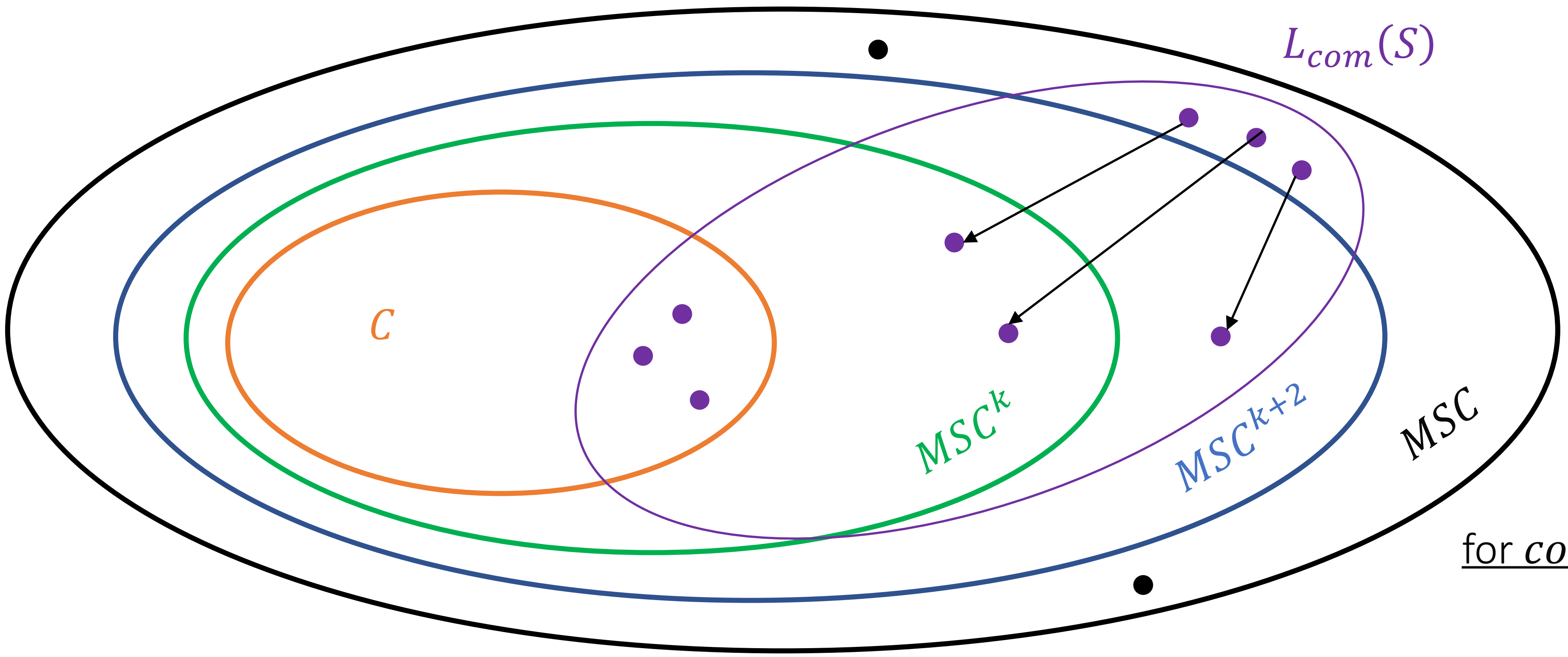


▶ **Lemma 9.** *Let $\mathcal{S}$ be a communicating system,* $\text{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$, $k \in \mathbb{N}$, *and* $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-}\mathsf{stw}}$. *Then,* $L_{\text{com}}(\mathcal{S}) \subseteq \mathcal{C}$ *iff* $L_{\text{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-}\mathsf{stw}} \subseteq \mathcal{C}$.



Is this scenario possible?

▶ **Lemma 9.** *Let $\mathcal{S}$ be a communicating system,* com $\in \{\mathsf{p2p}, \mathsf{mb}\}$, $k \in \mathbb{N}$, *and* $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-stw}}$. *Then,* $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ *iff* $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-stw}} \subseteq \mathcal{C}$.



for $com \in \{p2p, mailbox\}$

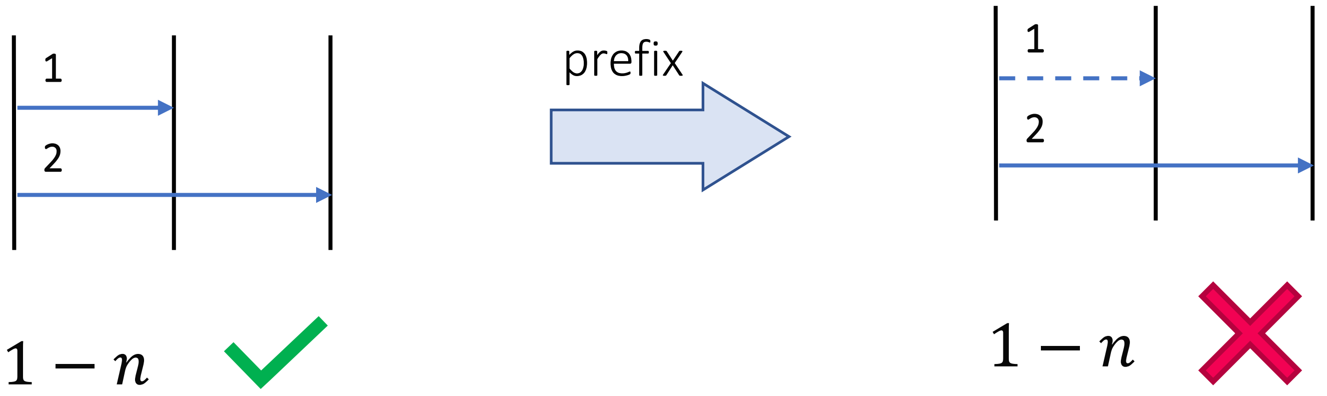
Is this scenario possible? **NO**, because of Lemma 10 and prefix-closure of $L_{com}(S)$

▶ **Lemma 9.** *Let $\mathcal{S}$ be a communicating system,* $\mathrm{com} \in \{\mathsf{p2p}, \mathsf{mb}\}$, $k \in \mathbb{N}$, *and* $\mathcal{C} \subseteq \mathsf{MSC}^{k\text{-}\mathsf{stw}}$. *Then,* $L_{\mathrm{com}}(\mathcal{S}) \subseteq \mathcal{C}$ *iff* $L_{\mathrm{com}}(\mathcal{S}) \cap \mathsf{MSC}^{(k+2)\text{-}\mathsf{stw}} \subseteq \mathcal{C}$.

Proof based on Lemma 10 and prefix-closure of $L_{com}(S)$

for $com \in \{p2p, mailbox\}$

Problem: $L_{1-n}(S)$ is not prefix-closed ☹


1
2
$1 - n$ ✓
prefix
1
2
$1 - n$ ✗