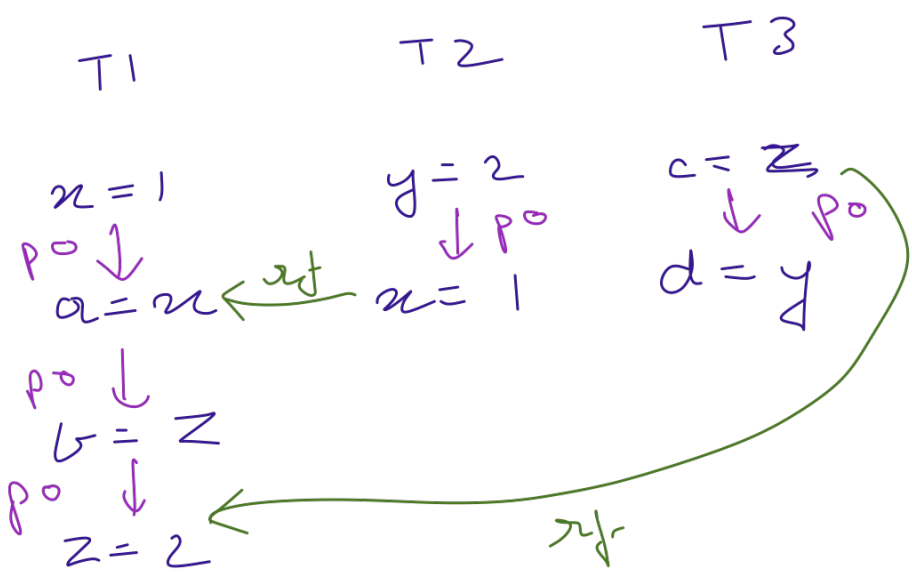
Model Checking Notes! :)

MPI - Internship
with
Viktor V

Model Checking for Weak Memory!

#1: An idea is that, extra information is not always essential.

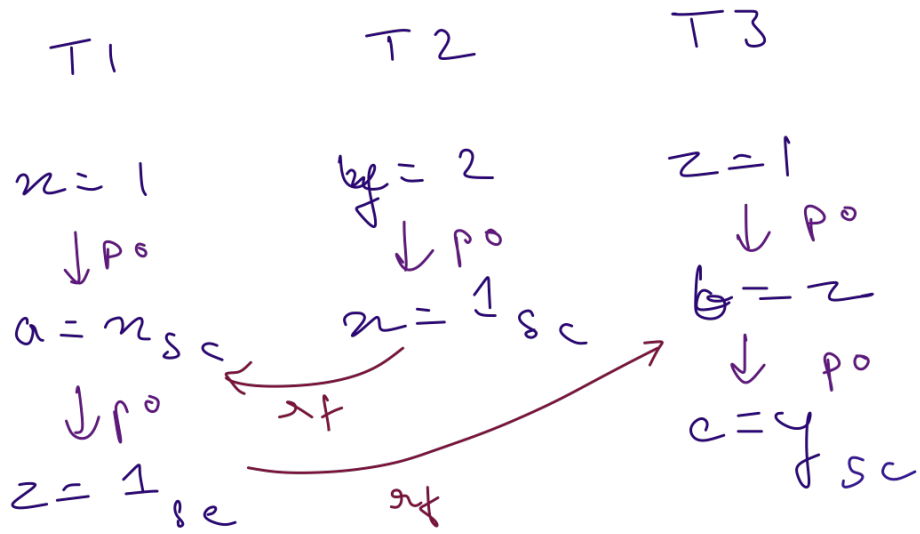
Less information meaning less synchronization or less cache updates.



$a = 1 \wedge c = 2 \wedge d = 2$ or 0 ?

* $a = 1$ implies T1 have seen what T2 has to offer
* $c = 2$ implies T3 has seen what T2 has seen.....

#2: Sometimes when someone sees something, it also means it is public (no private chats)



$$\boxed{a = 1 \;\wedge\; b = 1 \;\wedge\; c = 2.}$$

T1/T2 T1/T3 T2/init.

* We cannot identify which one of these is the exact way the current execution is.

#3: Let's take a simple example, when do we not need some info from another thread?

T1	T2
$y=2$	$y=2$
↓po	↓po
$x=1$	$x=1$ rel
↓po	
$a=x$ acq	
↓po	
$b=y$	

rf

Same info

* If we do not get some new information, it is not necessary to check

T1	T2	T3
$z=1$	$x=1$	$y=2$
$y=2$	$a=x$ acq	$x=1$ rel
$x=1$ rel	$b=y$	$c=2$
⋮		

Redundant info addition:

* Extra info $z=1$ may not even be needed any T2 or any of its connectors.

* As a recap :-

① Threads can gain info about what other threads have done, by using sw (or with acq-rel / sc-sc) relations between each other.

② Is the info shared private or is it public (is it acq-rel / sc-sc) is another thing that concerns me.

③ If it is private, then we need to concern ourselves with the fact that any thread having private comm with some thread is bound to get info about other threads prev having private comm.. The question is then, whether this info is in some way useful to anyone.

#4: If we do not req redundant information, which executions would be equivalent?

T1	T2	T3	T4
$z=1$	$y=1$	$z=1$	$x=1$ rel
↓ po	↓ po	↓ po	
$y=1$	$x=1$ rel	$x=1$ rel	
↓ po			
$x=1$			
↓ po			
$a=x$ acq.			

* Any of these, we do not need to explore of relations with acq x & $x=1$ of T2/T3/T4 so we only get redundant information.

#5 But what if this redundant info may not be so redundant?

T1	T2
$x=1$	$b=y$
↓ po	↓ po
$a=x$ acq	$x=1$ rel
↓ po	
$y=2$	

$a=1 \wedge (b=0 \vee b=2)$

This can come from just its own thread's write to x.

* Does this mean, we need to explore both executions? one where $b=y \xrightarrow{hb} y=2$ & one where there isn't any hb with it?

Could we view the redundant problem in a different way?

T1	T2		T0	T1	T2
$x=1$	$x=1$	$\equiv$	$x=1$	$a=x$	$x=$
$a=x$					

* No we cannot b/c happens before relations do not seem to exist beforehand & no scope of getting them back..

What Viktor claims is that for model checking we need just soundness. Be that so, shouldn't all possible executions be checked to see if the memory model itself is not creating problems?

Why all this symmetric reduction & what not when nobody is bothered about the model?

"Litmus for Sym"

① T1 T2

a1: x = 1 a2: x = 1

↓ po ↓ po

b1: a = x b2: b = x

b1: a1 → b2: {a1, a2}

② T1 T2

a1: x = 1 a2: x = 1

b1: a = x b2: b = x

c1: x = 2 c2: x = 2

b1: a1 → b2: {a1, c1, a2}

b1: a2 → b2 {c1}

③ T1 T2

a1: x = 1 a2: x = 1

↓ ↓

b1: x = 2 b2: x = 2

↓ ↓

c1: a = x c2: b = x

c1: b1 → c2: {b1, b2}

c1: a2 → c2: {b2}

(4)

T1	T2
a1 : x=1	a2 : x=1
↓ po	↓ po
b1 : x=2	b2 : x=2
↓ po	↓ po
c1 : a=x	c2 : b=x
↓ po	↓ po
d1 : x=3	d2 : x=3

c1 : b1 → c2 : {b1, d1, b2}

c1 : a2 → c2 : {d1, b2}

c1 : b2 → c2 : {d1}

* Coherence Rule is visible.

(5)

T1	T2	T3
a1 : x=1	a2 : x=1	a3 : x=1
↓ po	↓ po	↓ po
b1 : a=x	b2 : b=x	b3 : c=x

b1 : a1 → b2 : a1 → b3 : {a1, a2}

b1 : a1 → b2 : a2 → b3 : {a2, a3}

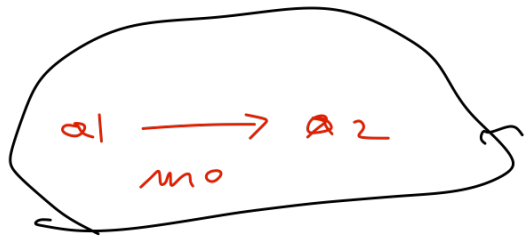
Random idea !! (Thanks to Vikta)

* Write a mutency order of all writes that are equal in each thread.

T1 T2

a1: x=1 a2: x=1

b1: a=x b2: b=x



b1 has choice to read from a1 or a2

b2 has choice to read based on what b1 reads from, but which also ensures no change in predef $\xrightarrow{mo}$.

b1 : a1 → b2: {a2}

* cuz if a1 then $a2 \xrightarrow{mo} a1$ which is countering the pre-def order

b1 : a2 → b2 : {a2}.

* Its good.....

T1 T2

a1 x=1 a2 x=1

b1 a=x b2 b=x

c1 x=2 c2 : x=2

b1 : a1 → b2 : {c1, a2}

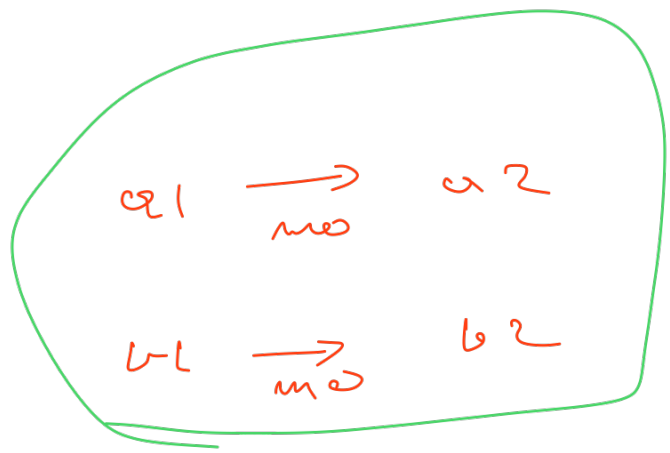
b1 : a2 → b2 : {a2, c1}

T1 T2

a1 : x=1 a2 : x=1

b1 : x=2 b2 : x=2

c1 : a=x c2 : b=x



c1 : b1 → c2 : {b2}

c1 : a2 → c2 : {b2}

c1 : b2 → c2 : {b2}.

T1 T2 T3

a1 x=1 a2 x=1 a3 x=1

b1 a=x b2 b=x b3 c=x

a1 → a2 → a3
mo mo.

b1: a1 → b2: a2 → b3: a3

b1: a2 → b2: a2 → b3: a3

b1: a3 → b2: a3 → b3: a3

b1: a2 → b2: a3 → b3: a3.

~~b1: a1~~ b1: a1 b2: a3 b3: a3

* okay so this way looking at it seems to work.

a1 → a2 → x.

T1 T2 T3

a1 x=1 a2 x=1 x: x=2

b1 a=x b2: b=x

a1 → a2
mo

a1 → x

writes ordered after x

b1: a1 → b2: {a2, x}

b1: a2 → b2: {a2, x}

b1: x → b2: {a2, x}

a2

x

a1 → x → a2
mo mo

T1	T2	T3	T4
a1: $x=1$	a2: $x=1$	$x=3$	$x=4$
b1: $x=x$	b2: $0=x$		a1 $\xrightarrow[mo]{}$ a2
c1: $x=2$	c2: $x=2$		c1 $\xrightarrow[mo]{}$ c2

b1 : a1 $\longrightarrow$ b2: $\{a2, c1, x\}$

b1 : a2 $\longrightarrow$ b2: $\{a2, c1, x\}$

b1 : x $\longrightarrow$ b2: $\{c1, x\}$.

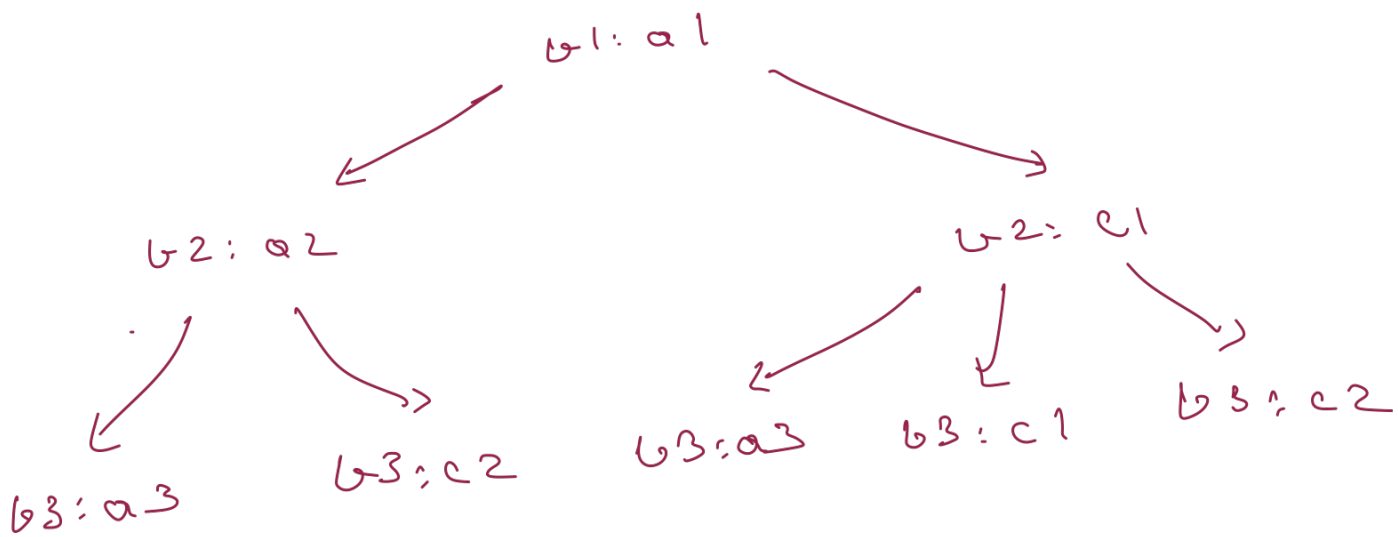
If b1 reads from c2, it means that b2 has taken place before it?

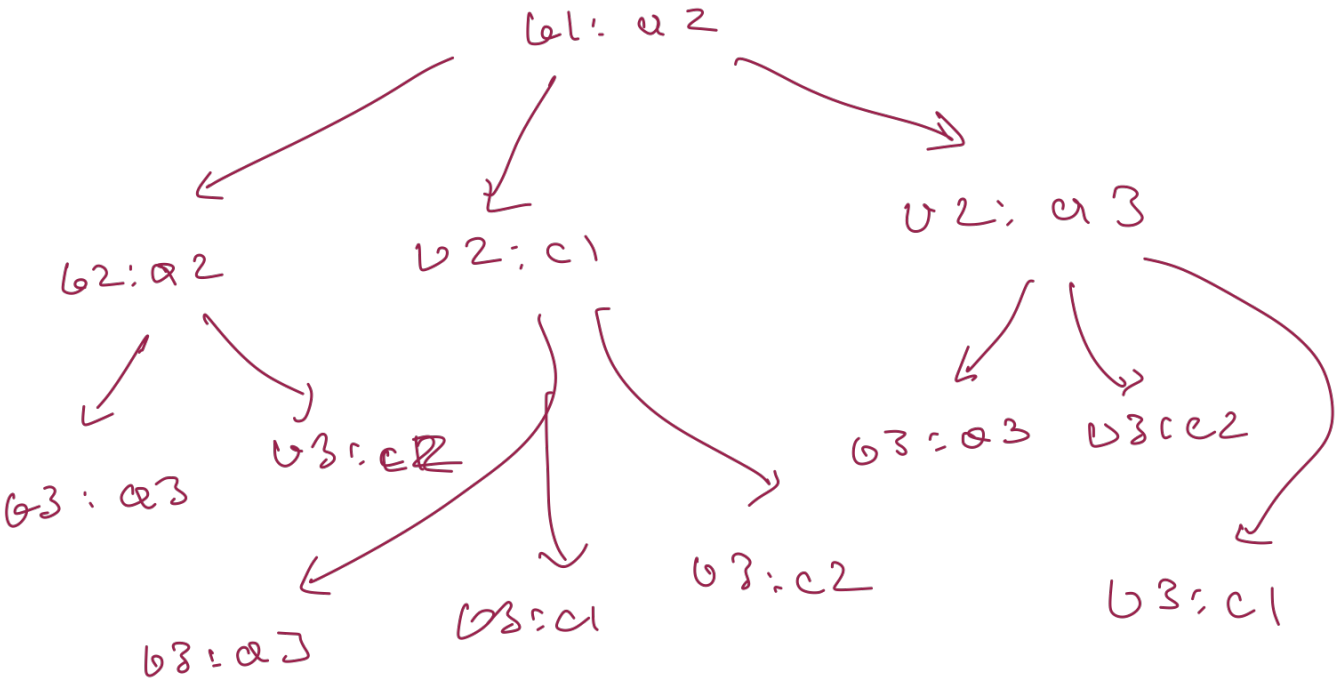
Somehow by ensuring a tot exists among events, it is possible to avoid symmetric explorations ....

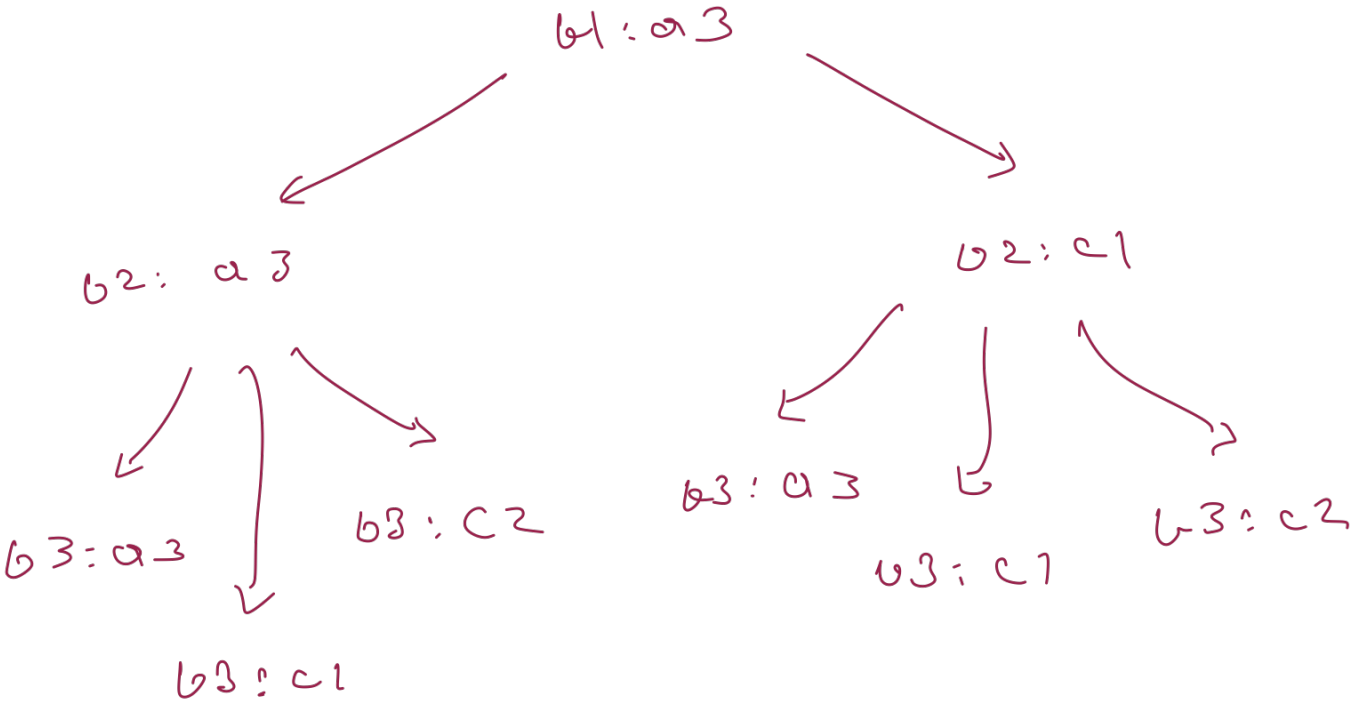
T1	T2
$x=1$	$x=1$
$a=x$	$b=x$
$x=2$	$x=2$

* The execution graphs will start a little differently, b.c if there are dependancies, the graph might change based on description.

T1	T2	T3
a1: $x=1$	a2: $x=2$	a3: $x=1$
↓	↓	↓
b1: $a=x$	b2: $b=x$	b3: $c=x$
↓	↓	↓
c1: $x=2$	c2: $x=2$	c3: $x=2$




b1: a2
b2: a2
b2: c1
b2: a3
b3: a3
b3: c2
b3: a3
b3: c1
b3: c2
b3: a3
b3: c2
b3: c1


b1: a3
b2: a3
b2: c1
b3: a3
b3: c2
b3: c1
b3: a3
b3: c1
b3: c2

T1 T2

a1: x=1 a2: x=1

b1: a=x b2: b=x

b1: a1 → b2: a2

b1: a2 → b2: a2

b1: a1 → b2: a1

sym

b2: a1 → b1: a1

T1 T2 T3

a1: x=1 a2: x=1 a3: x=1

b1: a=x b2: b=x b3: c=x

b1: a1 → b2: a3 → b3: a3

b1: a2 → b2: a2 → b3: a3

b3: a3 → b2: a1 → b1: a1

b3: a3 → b1: a2 → b2: a2

a

a

a & a are different b.c of the fact between the writes that exist.....

T1 T2

a1 x=1 a2 x=1

b1 a=x b2 a=x

c1 x=2 c2 x=2

b1 : a1 → b2 : a2

b1 : a1 → b2 : c1

b1 : a2 → b2 : a2

b1 : a2 → b2 : c1

Proof Argument for Soundness.

* Proof by contradiction:-

Suppose there is an execution graph violating the order between writes.

Then:-

→ Either one order is violated, other is open

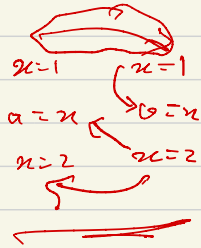
→ Both orders violated.

Ans: In both these cases, swapping thread identities, we have an execution graph covered by not violating orders....

** The intuition is that this will also hold for more than 2 threads. But catch is that every symmetric thread must have exactly one read.

Zoom Meeting with VIKTOR.

(1) Proof :- Idea by contradiction.



Two threads :-

→ Order between writes

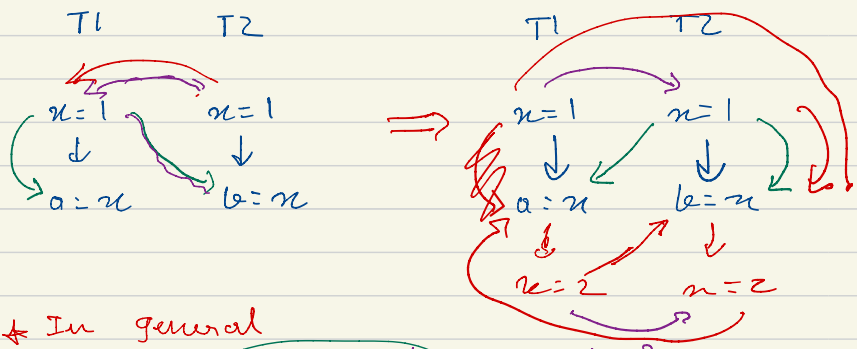
→ Free & can be any

→ Implied by a read

→ A read can

→ Read its own write

→ Read outside write



* In general

Wrong order?

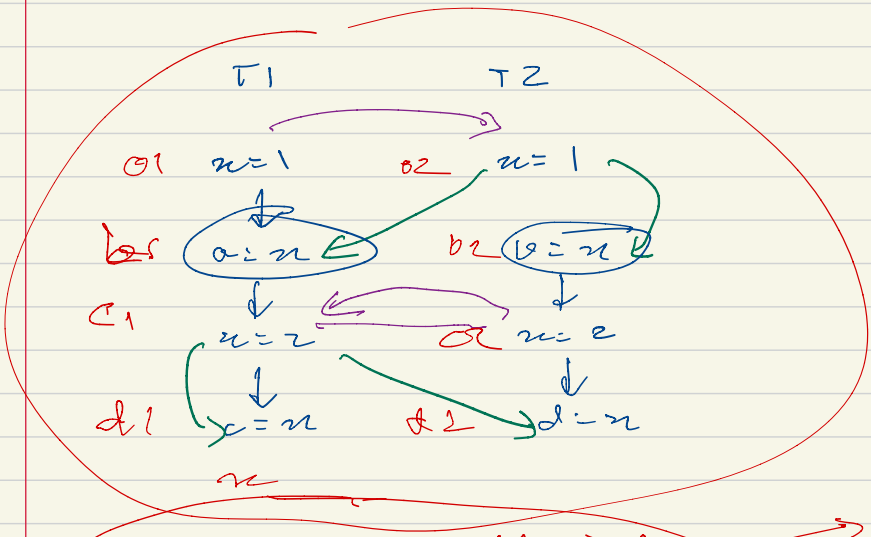
w1 → w2 → w3 . . . . . .

swap.

w1 → w3 → w2 → w3 → w1 → w2

② Limitation of Proof & whole idea :-

→ Does not work for more than one read.....



x

$b1 \to b2$ $\quad$ $d1 \to d2$

$b1 \to b2$ $\quad$ $d2 \to d1$

$a1 \to a2$ $\quad$ $c2 \to c1$

$a1 \to a2$ $\quad$ $c1 \to c2$

③ Implicit limitation to Rules too

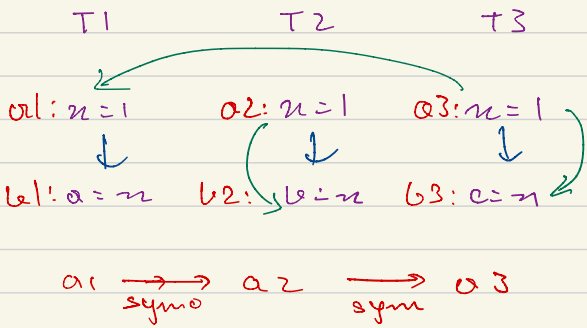
(4) What's to be done?

a) Either look at it with new insight, totally changing everything

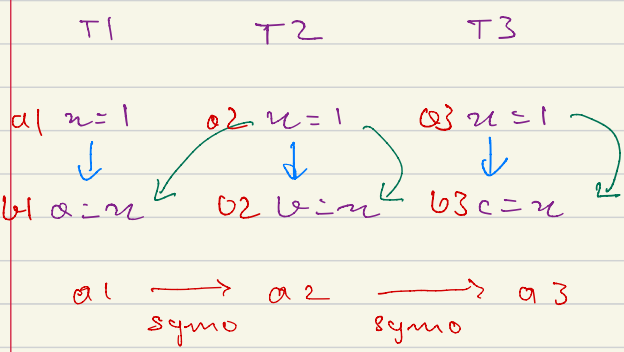
b) Make it iterative, proving along the way removing certain assumptions ... ...

Revisiting the Proof

* We said that order between both, reads & writes are important, but why is it so?

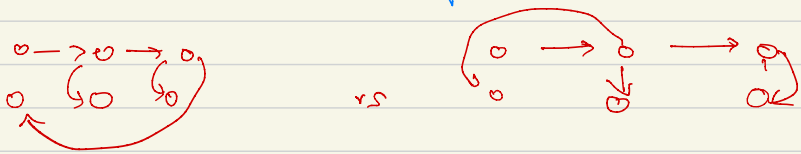


Is this a valid non-sym execution?

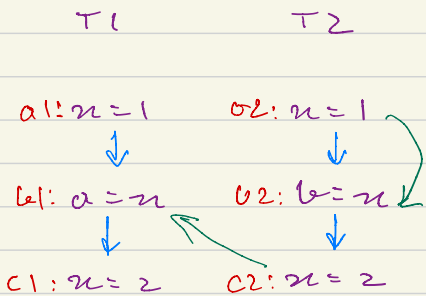


Can we get something of above with this?

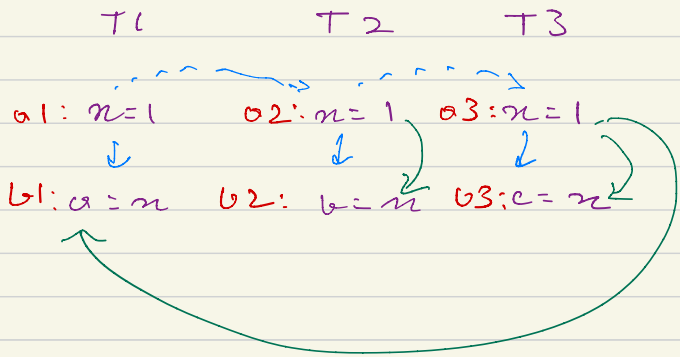
* If we swap identities of T2 & T3 we can get what is the first execution, however, something concerns me.....



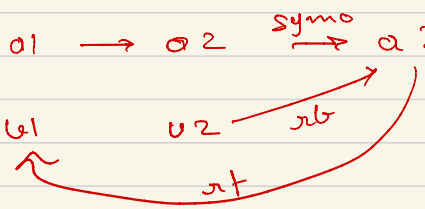
* The fact that writes must be ordered is clear, but that of reads I am still not sure.....



One thing is that here both order of reads & writes matter.



b1: a3 b2: a2 b3: a3



* So here we can infer that b2 occurs before b1.

* Back to work :-

T1	T2			
		b1 : a1	b2 : a2	1
		b1 : a1	b2 : a1	2
a1 : x = 1	a2 : x = 1	b1 : a2	b2 : a2	3
↓	↓	b1 : a2	b2 : a1	4
b1 : a = x	b2 : b = x			

1 is a unique execution & its symmetry cannot be observed on any other execution.

2 & 3 are symmetric to each other as swapping thread identities.

4 is not possible under coherence.

T1	T2			
a1 : x = 1	a2 : x = 1	b1 : a1	b2 : a2	1
↓	↓	b1 : a1	b2 : c1	2
b1 : a = x	b2 : b = x	b1 : a2	b2 : a2	3
↓	↓	b1 : a2	b2 : c1	4
c1 : x = 2	c2 : x = 2			

1, 2, 3, 4 is explored keeping in mind that $a1 \rightarrow a2 \wedge a1 \rightarrow c2$ hold.

* The idea is that the writes may be ordered, but the reads can still occur out of order.....

* What does it mean to swap thread identities? What changes occur in execution graph??

→ The following changes occur:-

a) rf relations ....

Those reads reading from a part sym thread's write will now read from the other thread.

b) sym o between writes:

Any sym order to be enforced are swapped, meaning the exact reverse order needs to be enforced...

* I think its not about the order of reads, but order of writes that is at play here..

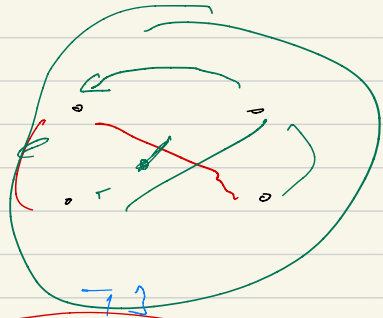
* Any write order implied by a reads-from relation can be reversely implied if the thread identities are swapped...

Meeting Points.

T1 $\longrightarrow$ T2

$x = 1$ $\quad$ $x = 1$

$a = x$ $\quad$ $b = x$



T1 $\quad$ T2 $\quad$ T3

$w_1 \longrightarrow w_2 \longrightarrow w_3$

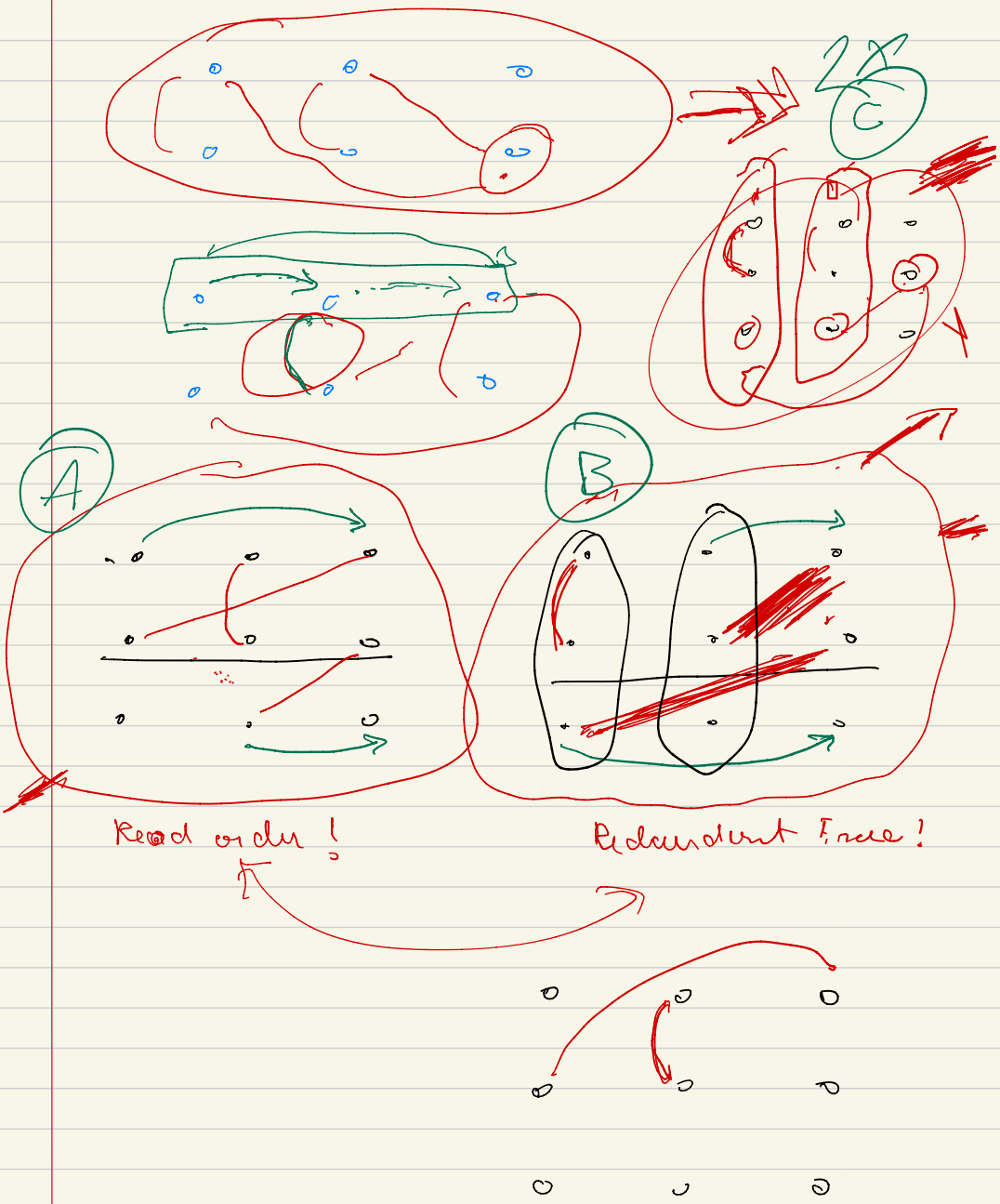
$w_1 \longrightarrow w_3 \longrightarrow w_2$

$w_3 \longrightarrow w_1 \longrightarrow w_2$

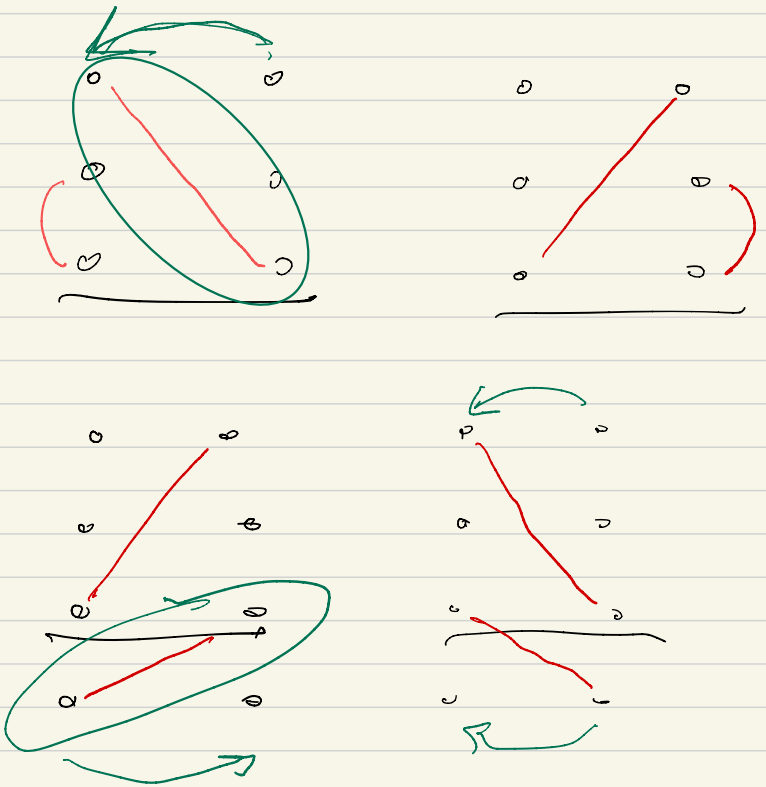
$w_3 \longrightarrow w_2 \longrightarrow w_1$

$w_1 \xrightarrow[mo]{} w_2 \xrightarrow[mo]{} w_3$

$w_1 \xrightarrow[mo]{} w_3 \xrightarrow[mo]{} w_2$


A
B
C
Read order !
Redundant Trace !

2 Threads. → simple. (Base)

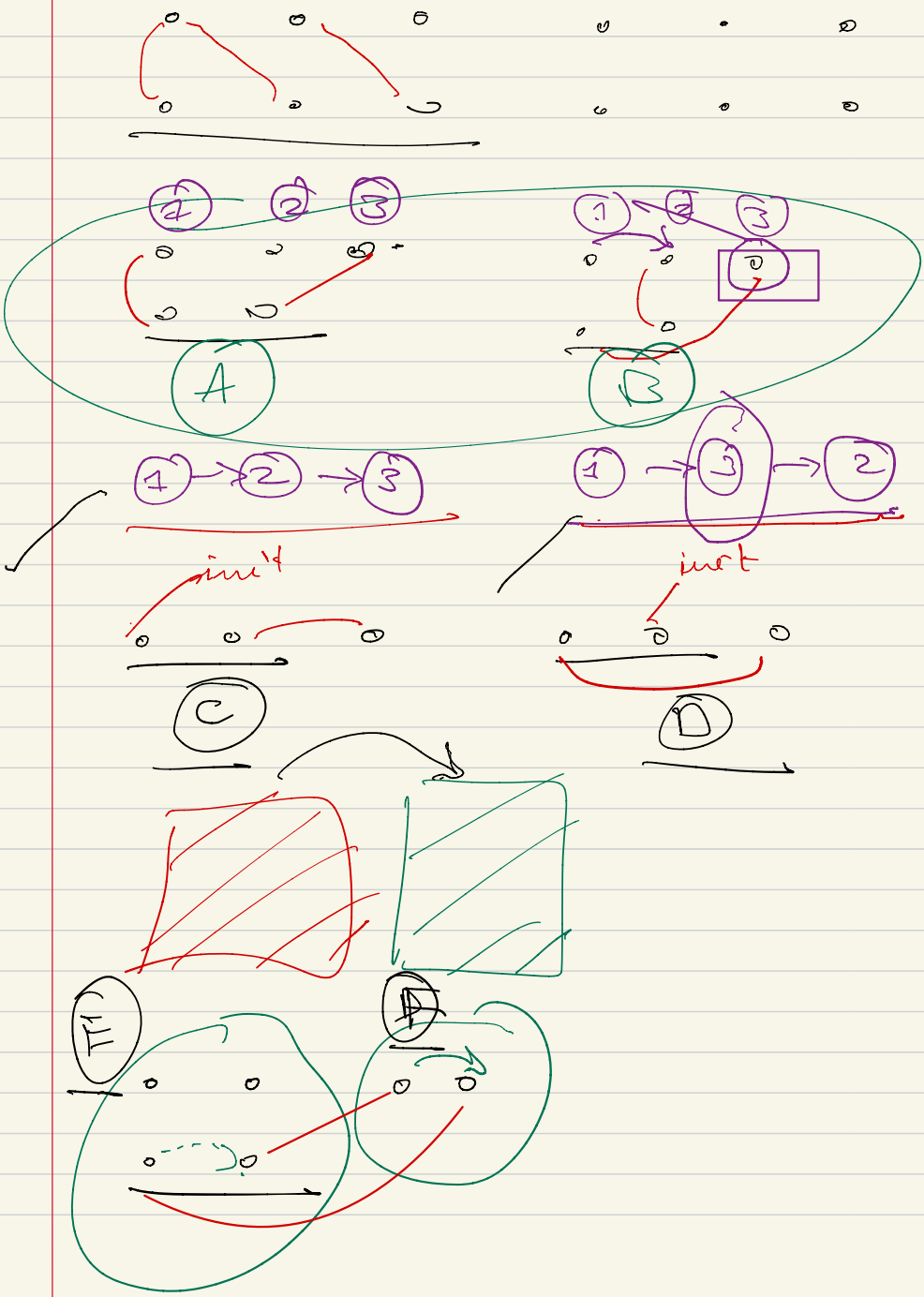


★ 1st Thread's read :-

Own write
Any write above ≡ read

★ 2nd Thread read

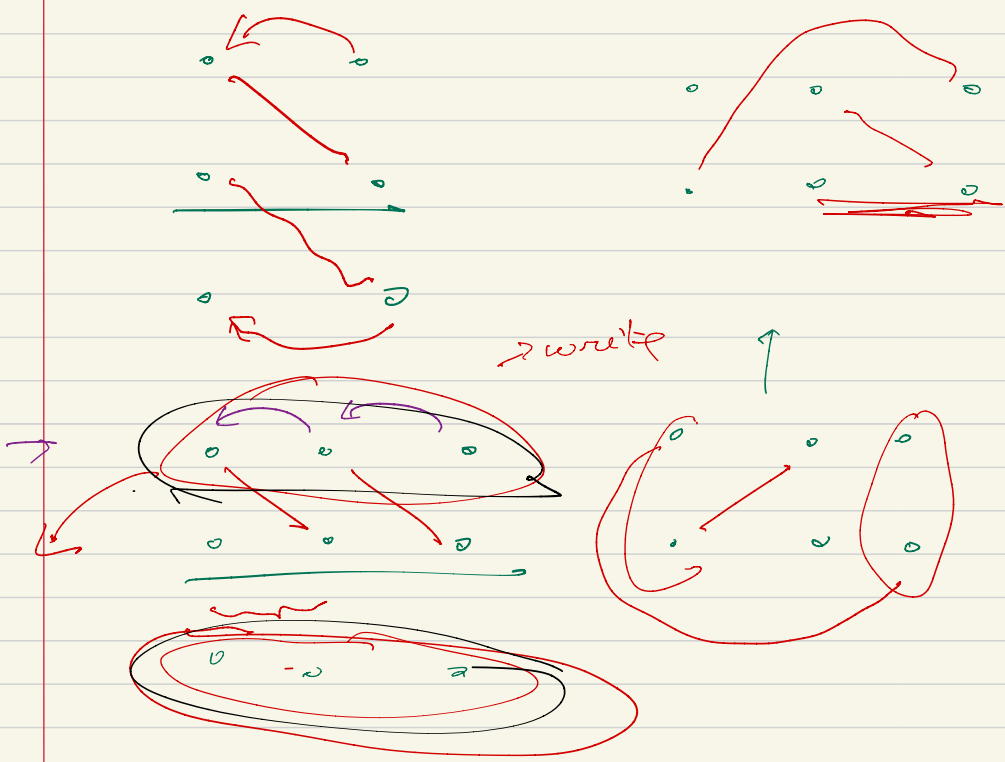
Own W
Any write below row ≡ read.


A
B
C
D
insert
insert

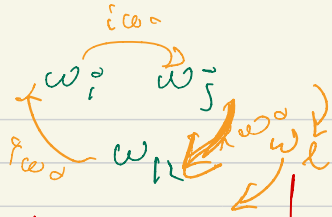
Meeting Viktor.

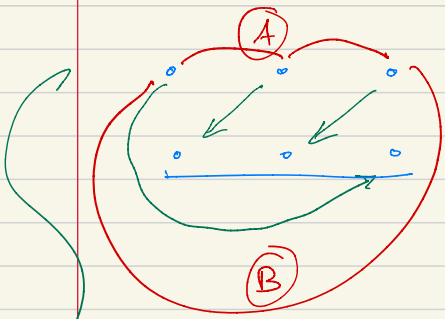
- Implied write order between 2 threads is reversed when swapping.
- Either all implied write orders between 2 threads are correct, or none.

→ Base Case – 2 threads.



Meeting!

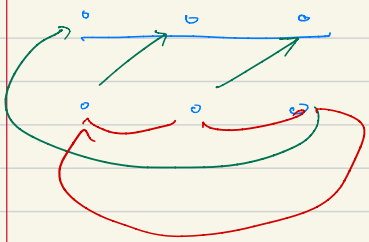




$(po \cup rf^{-1})_{loc}$ cycle.

Coherence Violating

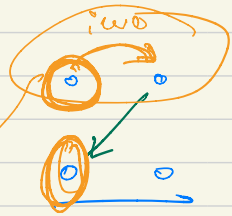
A



$(po \cup rf)_{loc}$ cycle

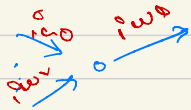
Coherence Violating.

B



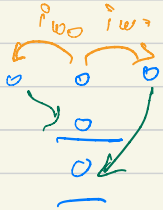
* Two Observations:-

* → writes above read:-

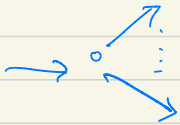


∞ incoming

1 outgoing



→ writes below read



∞ outgoing

1 incoming.

Given a set of writes

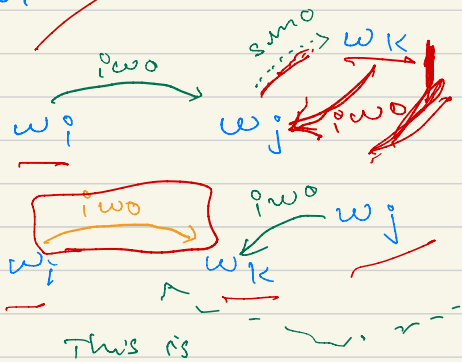
* Write orders implied, once fixed remain fixed:-

→ writes above

(1)

$w_i \xrightarrow{iwo} w_j$ remains the same.

(2)



$\Rightarrow w_i \xrightarrow{smo} w_k$.

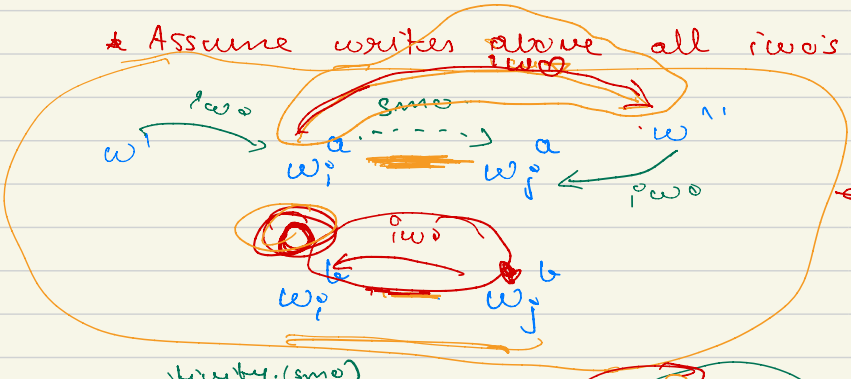
This is correct.

Hence for writes above once iwo is corrected, remains corrected.

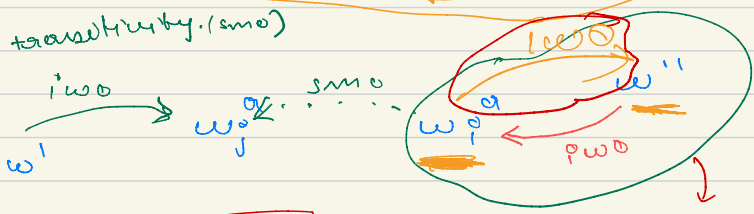
* Sym org for writes below

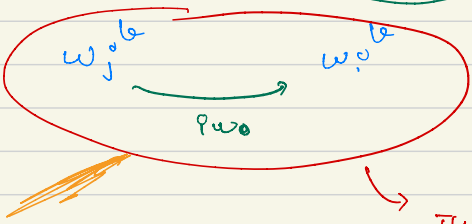
* General case where writes above & below are present.

* Assume writes above all iwo's fixed.



transitivity. (smo)

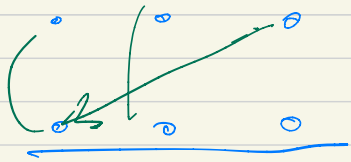


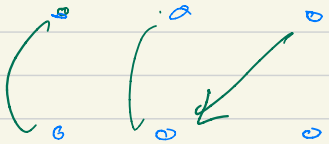


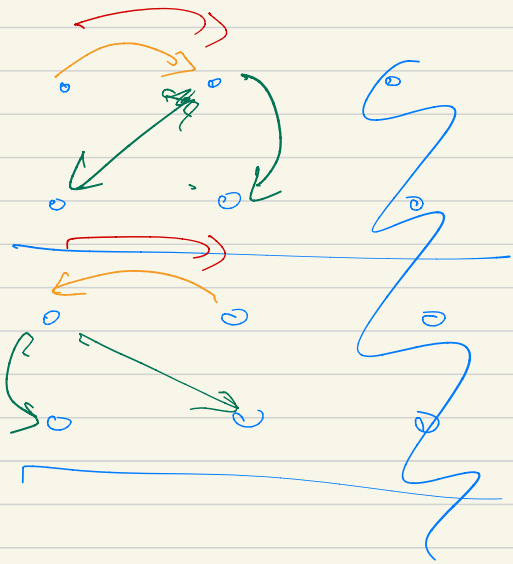
Thus once fixed will remain fixed

This since fixed will remain fixed.

** Hence, we show that since we fix all orders & they don't change, we can get an execution covered by our rules.







Meeting Points

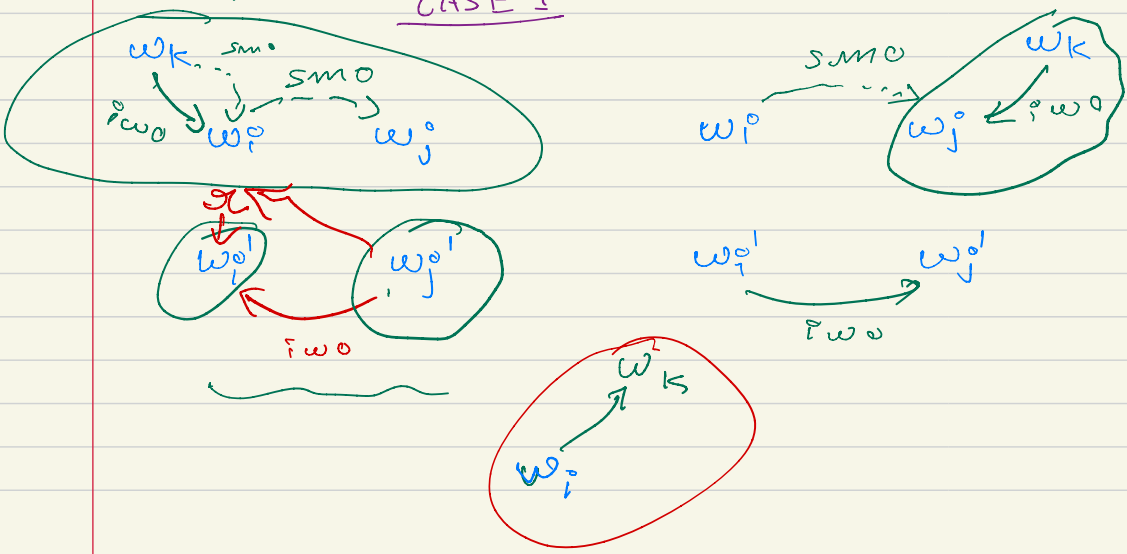
→ Soundness General Case :-

Given a fixed set of implied write orders, fixing another set may result in new implied write orders which can be fixed as per our previous result.

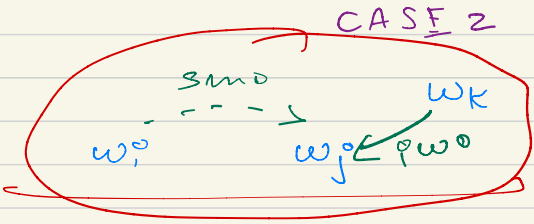
★★ The idea is to ensure that this new relation never occurred or was never present originally while fixing implied write orders

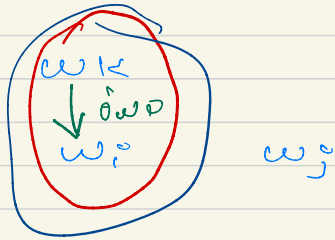
Part 1: Fixed set above read, fixing set below read

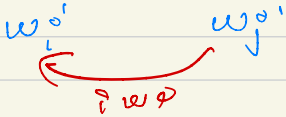
CASE 1

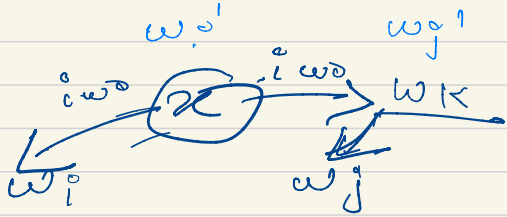


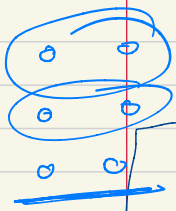
CASE 2





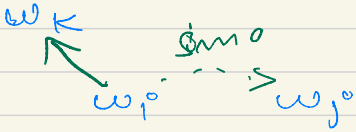


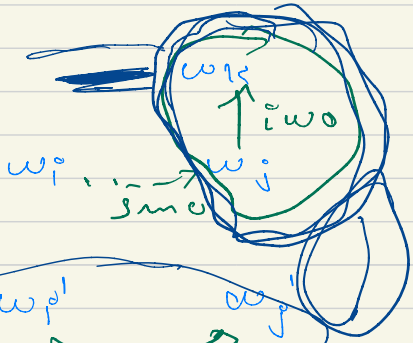


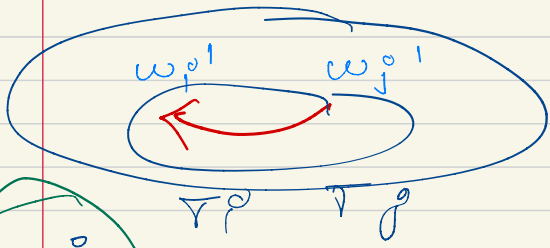


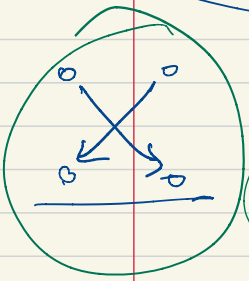
Part 2: Fixed set & fixing set above reach.

CASE 1

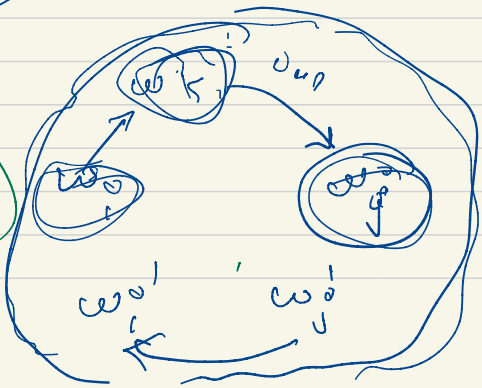








$(p \circ \cup \circ f^{-1})$ cyclic

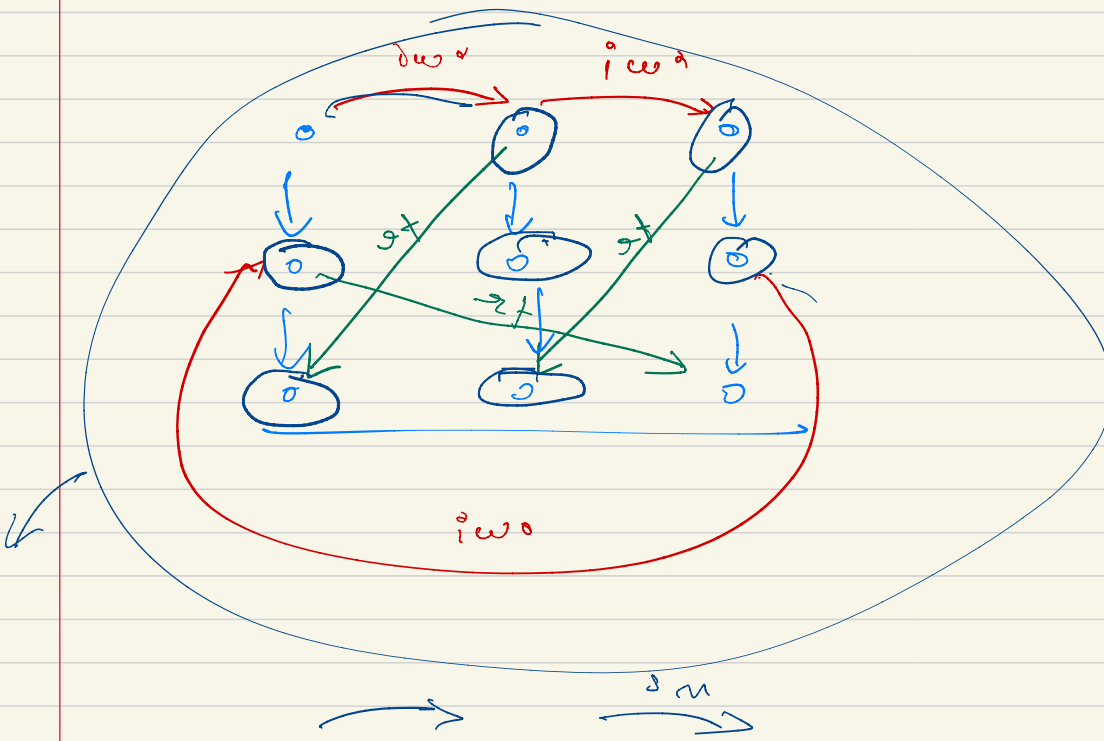


CASE 2.

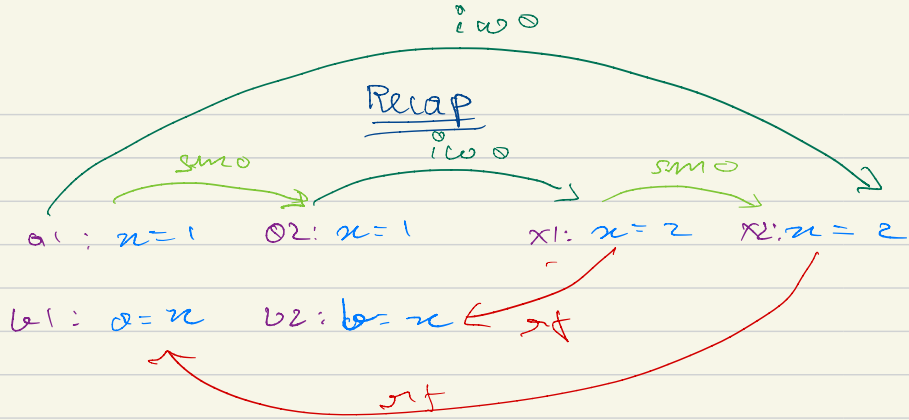
w_k $\qquad$ w_k

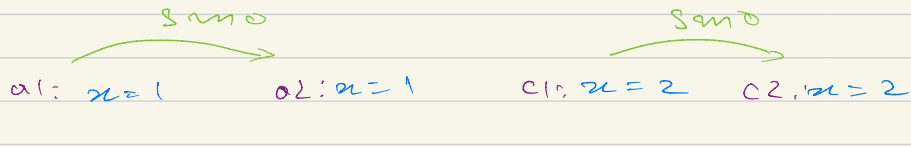
w_i $\qquad$ w_j $\qquad$ w_i $\qquad$ w_j

w_i' $\qquad$ w_j' $\qquad$ w_i' $\qquad$ w_j'

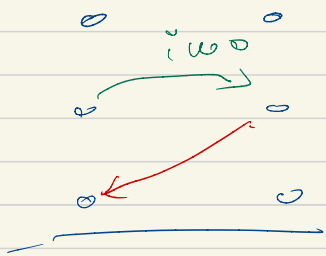


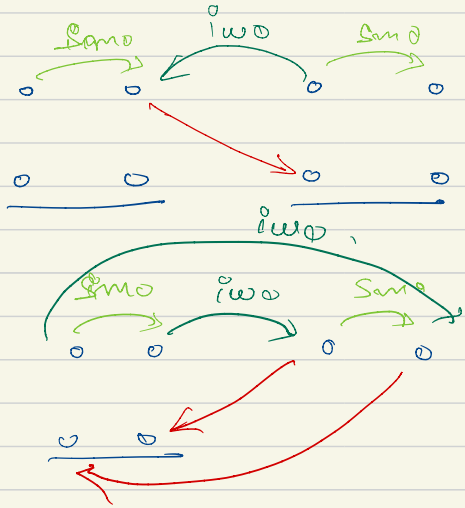
Recap



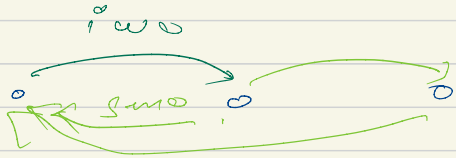


b1: a=x b2: b=x





* It seems to me that iwo should only be between writes of symmetric threads.

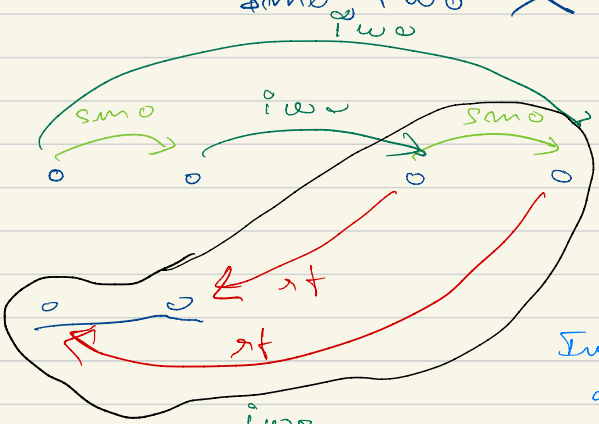


* This case cannot be possible.

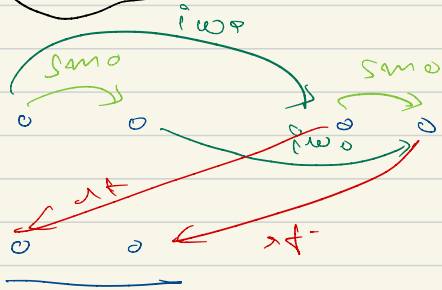
$\Downarrow$

iwo $\cup$ smo acyclic. ✓

smo; iwo ✗ i.e iwo is not transitive.



Implied read order can exist among reads, & can be easily derived?



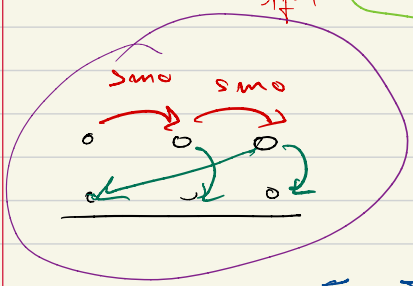
$\boxed{rf^{-1}; smo; rf}$

IRO

* We need to show implied read order does not conflict with soundness argument. How do we show that?

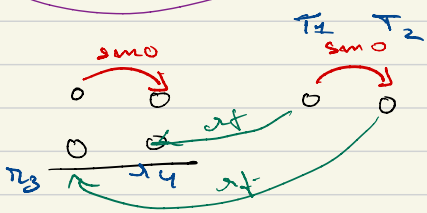
→ By definition? $rf^{-1}; smo; rf$?

$$r \xrightarrow[rf^{-1}]{} w_j^0 \xrightarrow[smo]{} w_k^j \xrightarrow[rf]{} r'$$



$$\boxed{rf^{-1}; smo; rf}$$

This set may have elements to



$r_4 \xrightarrow[rf^{-1}]{} w_1 \quad w_1 \xrightarrow[smo]{} w_2$

$w_2 \xrightarrow[rf]{} r_3$.

$r_4 \xrightarrow[rf^{-1}]{} w_2 \quad w_1 \xrightarrow[rf]{} r_3$.

$\boxed{w_1 \xrightarrow[rf]{} r_3} \quad w_1 \xrightarrow[smo]{} w_2 \quad w_2 \xrightarrow[rf]{} r_4$

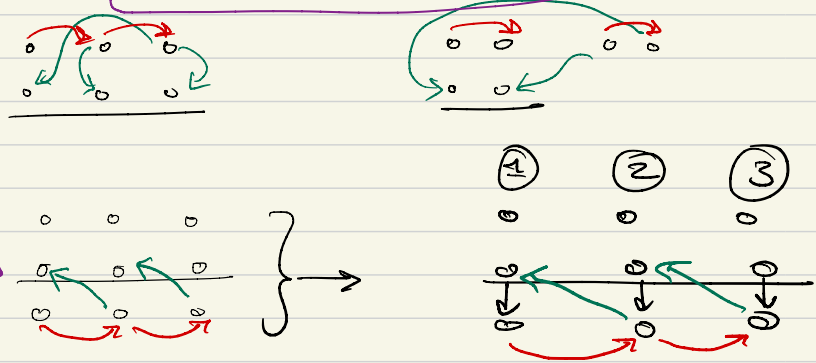
$r_3 \xrightarrow[rf^{-1}]{} w_1$

★ Implied Read Order.

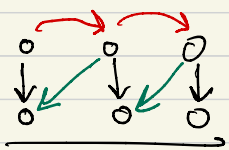
- Certain implied read orders are incorrect & may not be covered by implied write order.
- We only select those patterns to define our version of implied read order.

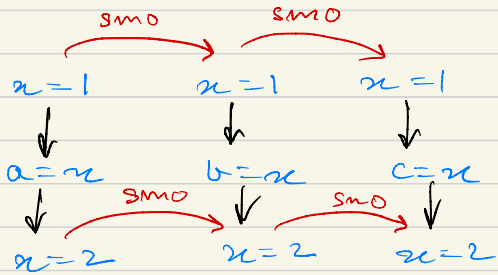
cannot explain this

$\boxed{rf^{-1} ; smo ; rf_e}$



$\boxed{po ; smo ; rf}$

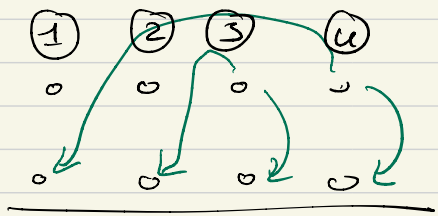


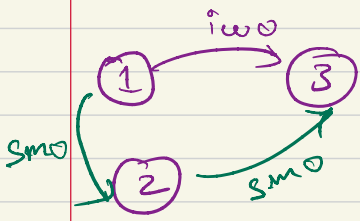


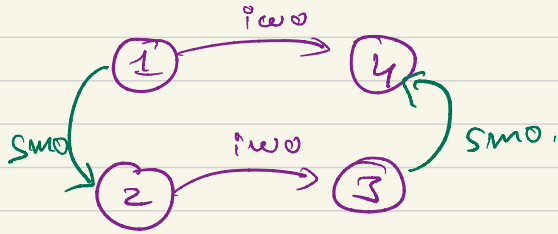
$\boxed{po ; rf^{-1}}$

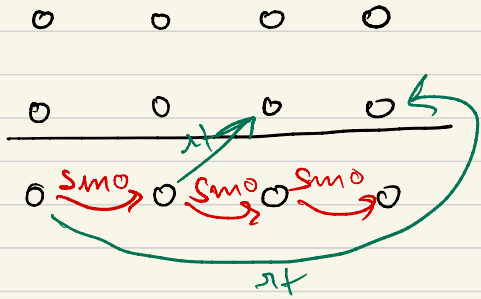
~~po;~~ rf ; po

Short Meeting (Viktor)



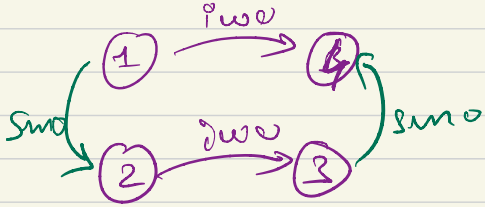






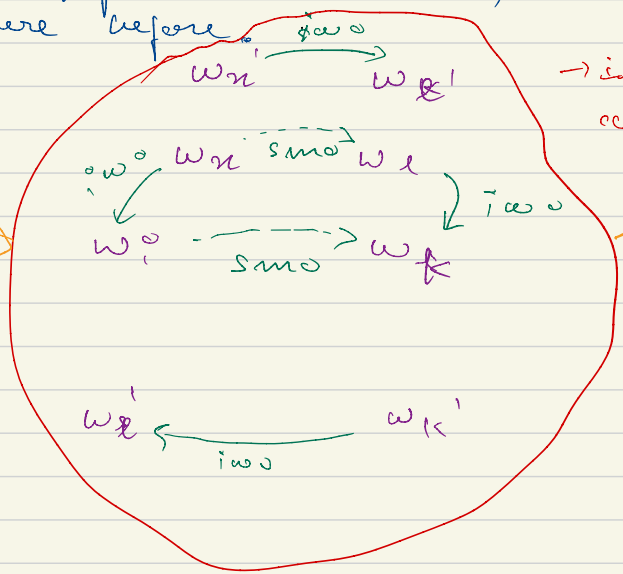
$iwo; smo; iwo^{-1}$

$smo; iwo^{-1}$



Soundness (Lemma 2)

* Any new implied write order, could not have been there before.

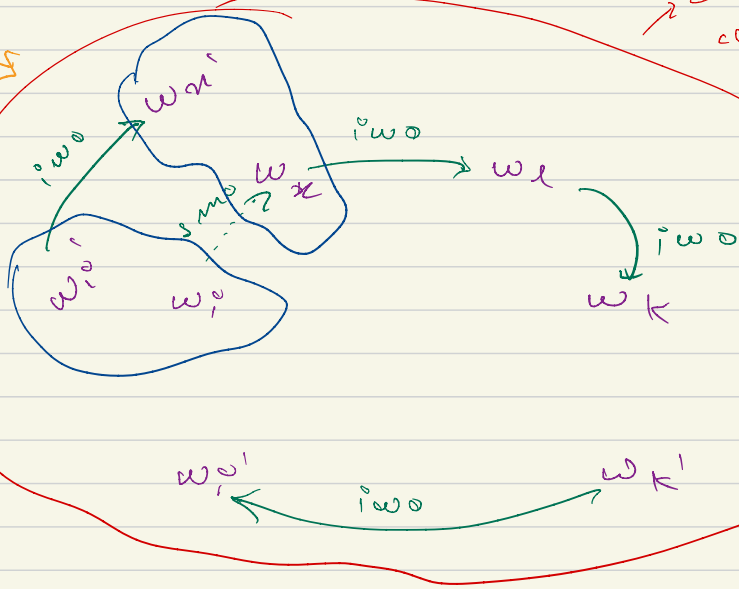


→ incorrect configuration

w_x' w_l' above/below we get either w_x / w_l invalid.

Case where w_i w_k above read

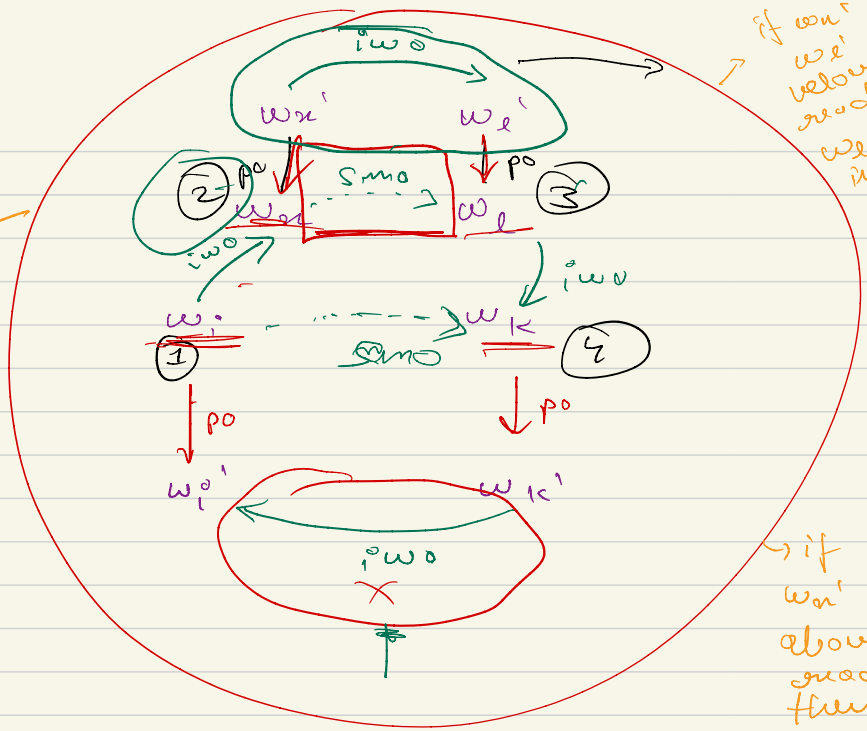
w_i' w_k' below read



incorrect config.

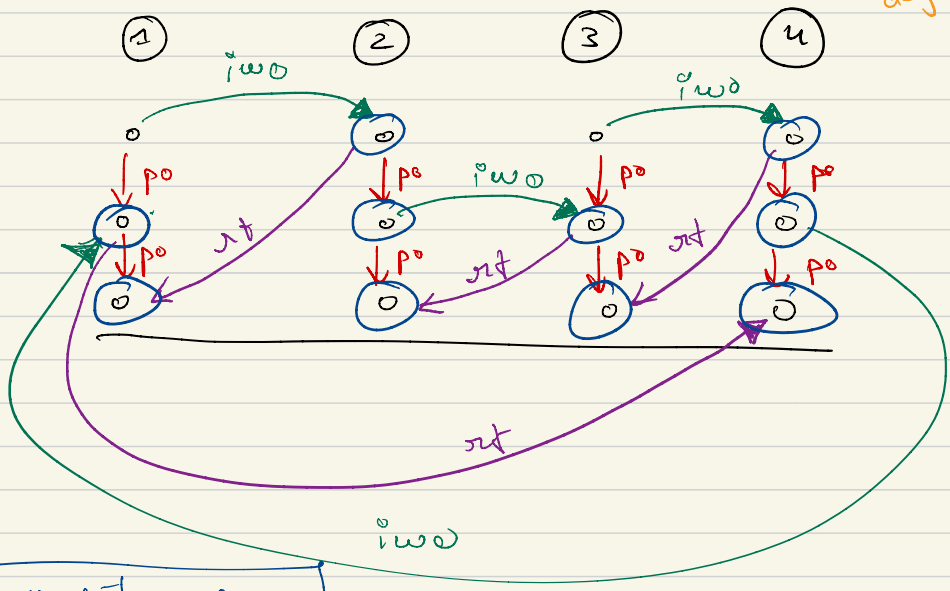
w_x' w_i' above/below we get either w_i or w_x invalid

Case where w_0, w_k above read & w_0', w_k' above read

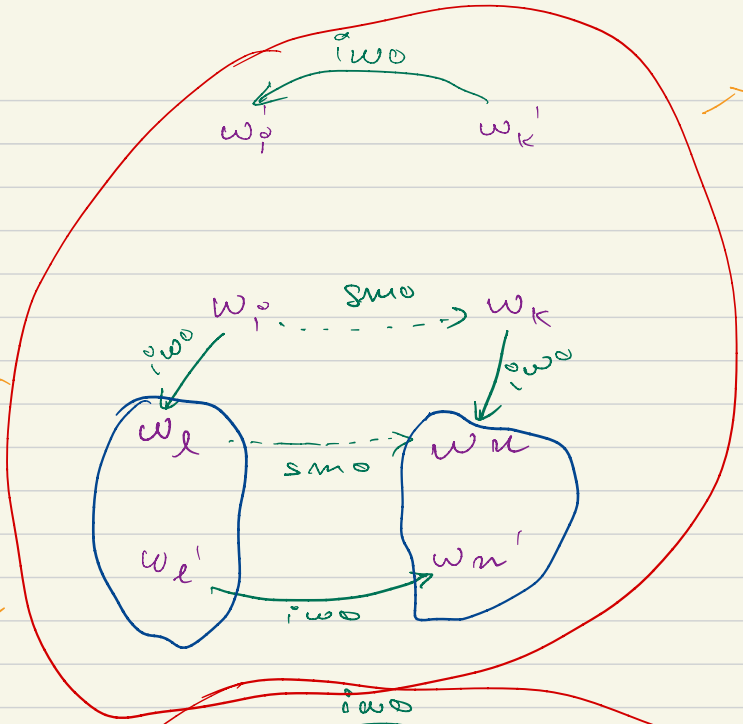


if w_x' w_l' below read w_l is initial

if w_x' w_l' above read, then $pour^{-1}$ acyclic

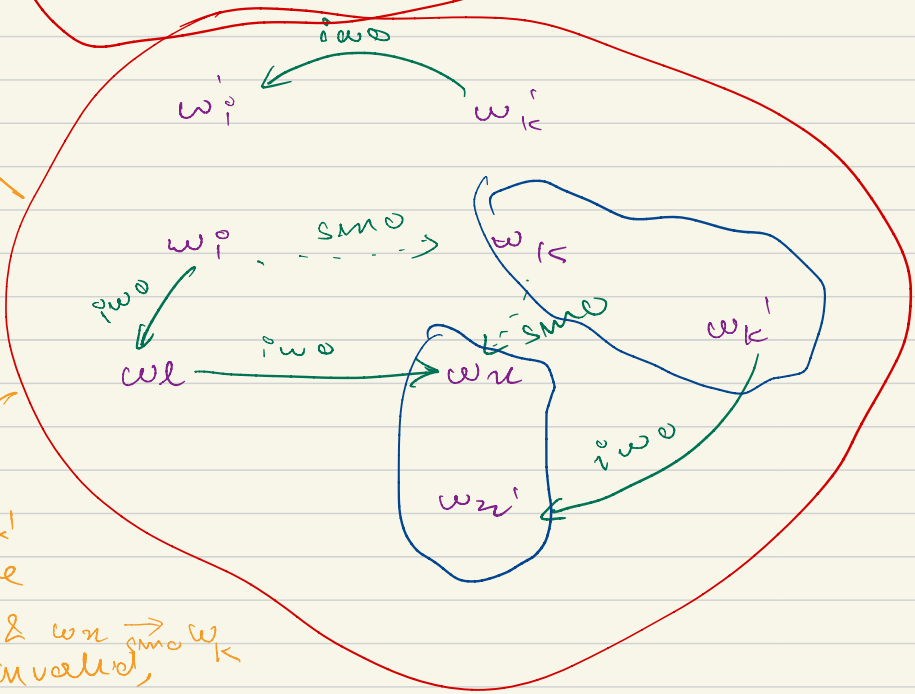


$\boxed{pour^{-1} \text{ cycle.}}$

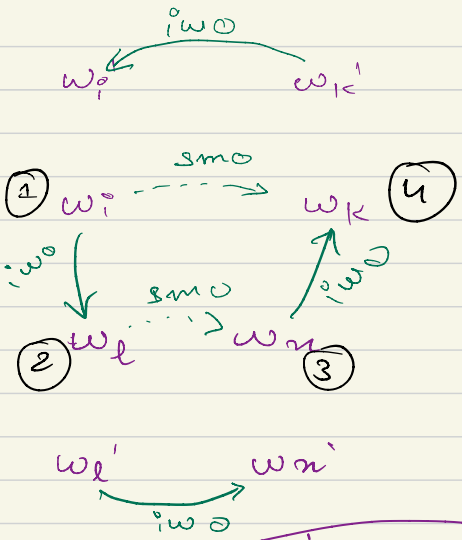


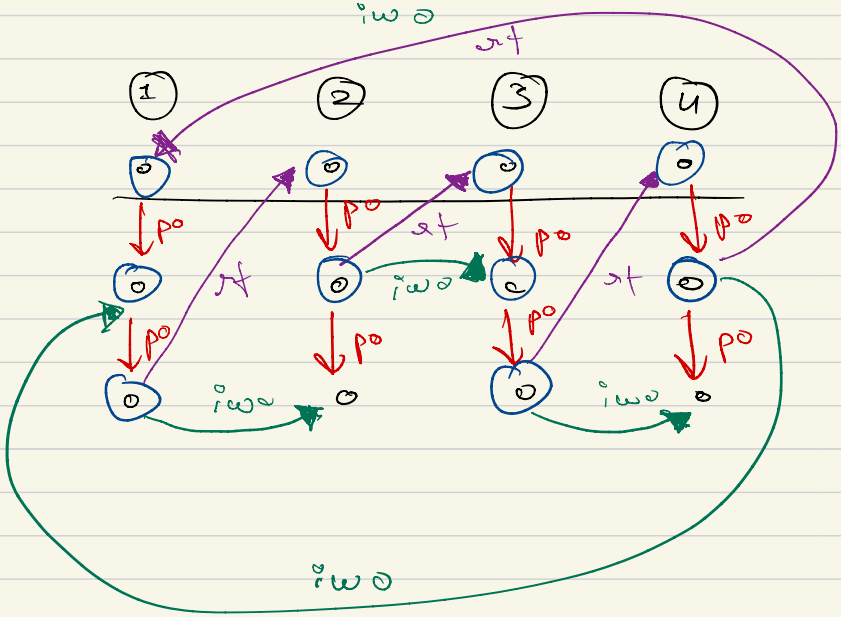
When w_n' & w_l' are above read, then w_l is invalid, while if they are below, w_n is invalid.

Case when w_i, w_k below read & w_i', w_k' above



When w_n' & w_k' are above read & $w_n \xrightarrow{smo} w_k$ w_k' is invalid, while if $w_k \xrightarrow{smo} w_n$ w_n' is invalid.





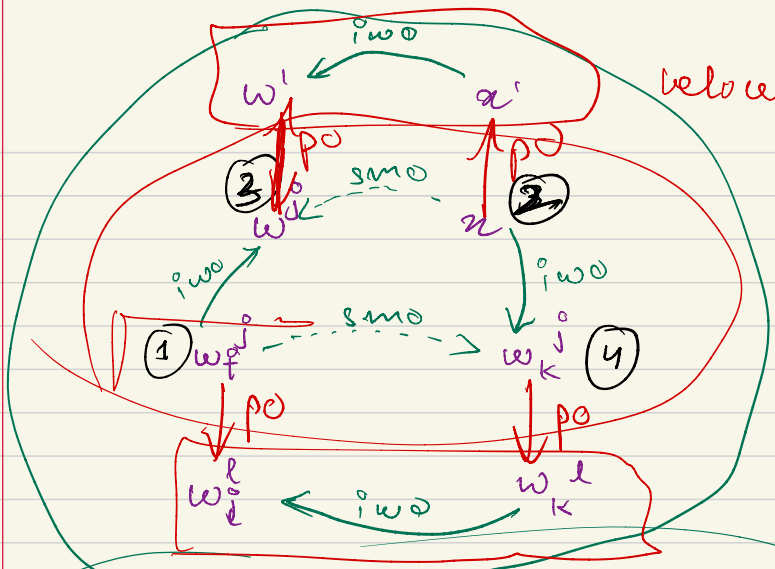
$\boxed{\text{pourt cyluc}}$

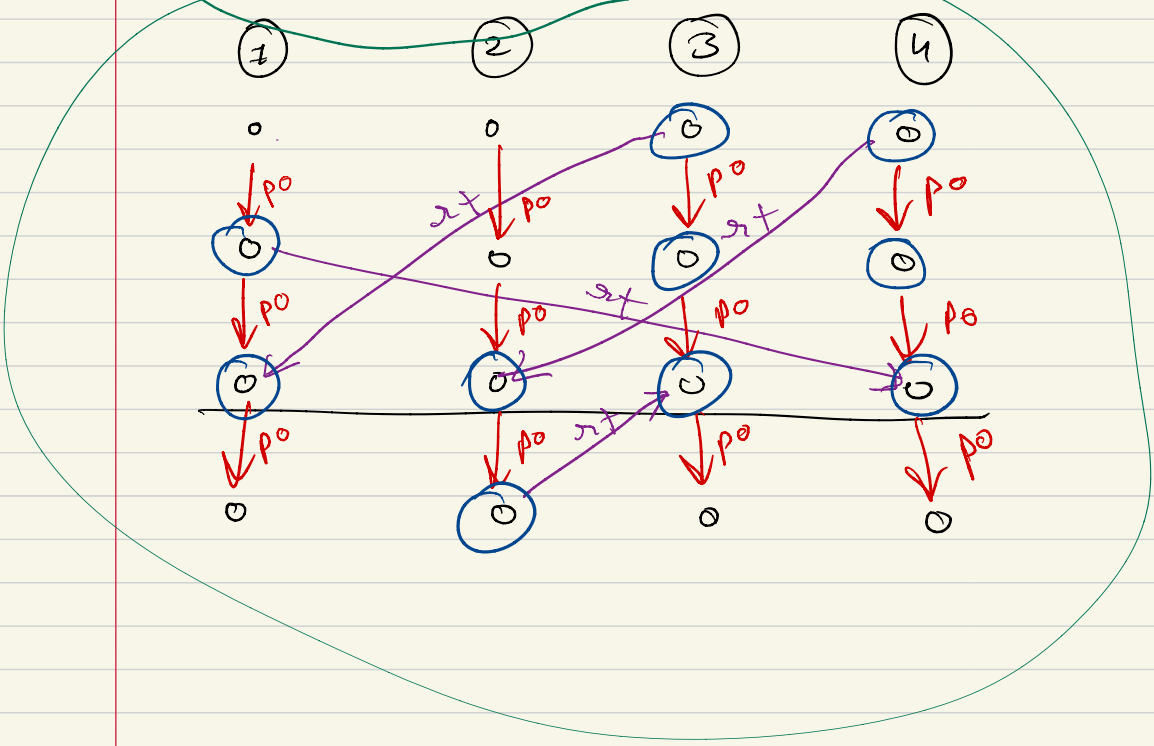
$$
\begin{tikzcd}
w' \arrow[r, "iwo"] & x' \\
w_i^j \arrow[r, dashed, "smo"] & x \arrow[d, "iwo"] \\
w_i^j \arrow[u, "iwo"] \arrow[r, dashed, "smo"] & w_k^j \\
w_i^l & w_k^l \arrow[l, "iwo"']
\end{tikzcd}
$$

case when w_i^j, w_k^j & w_i^l, w_k^l are above read.

if w', x' above read, pop rf^{-1} cycle, else

below read





$$\omega \xleftarrow{\text{iwo}} x'$$

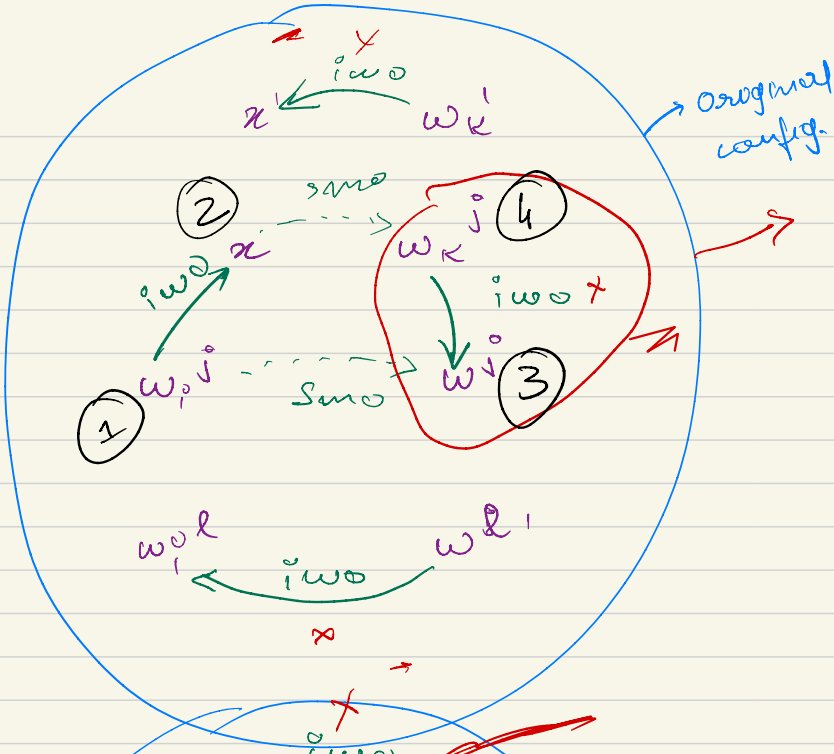
$$\begin{tikzcd}
 & \omega_j & x \arrow[l, dashed, "\text{smo}"'] \arrow[d, "\text{iwo}"] \\
\omega_k^j \arrow[ur, "\text{iwo}"] & & \omega_i^j \arrow[ll, dashed, "\text{smo}"]
\end{tikzcd}$$

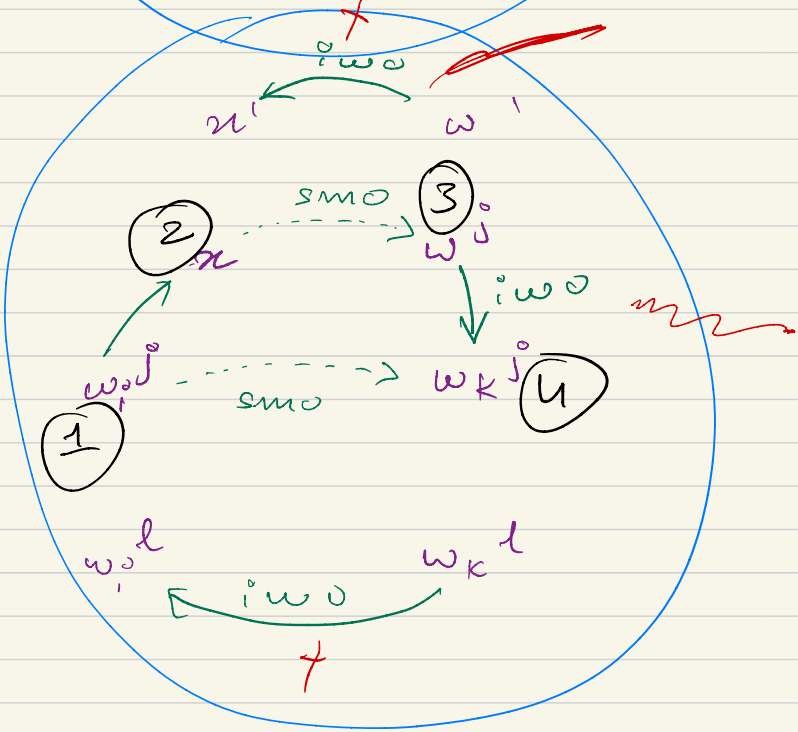
$$\omega_k^l \xleftarrow{\text{iwo}} \omega_i^l$$

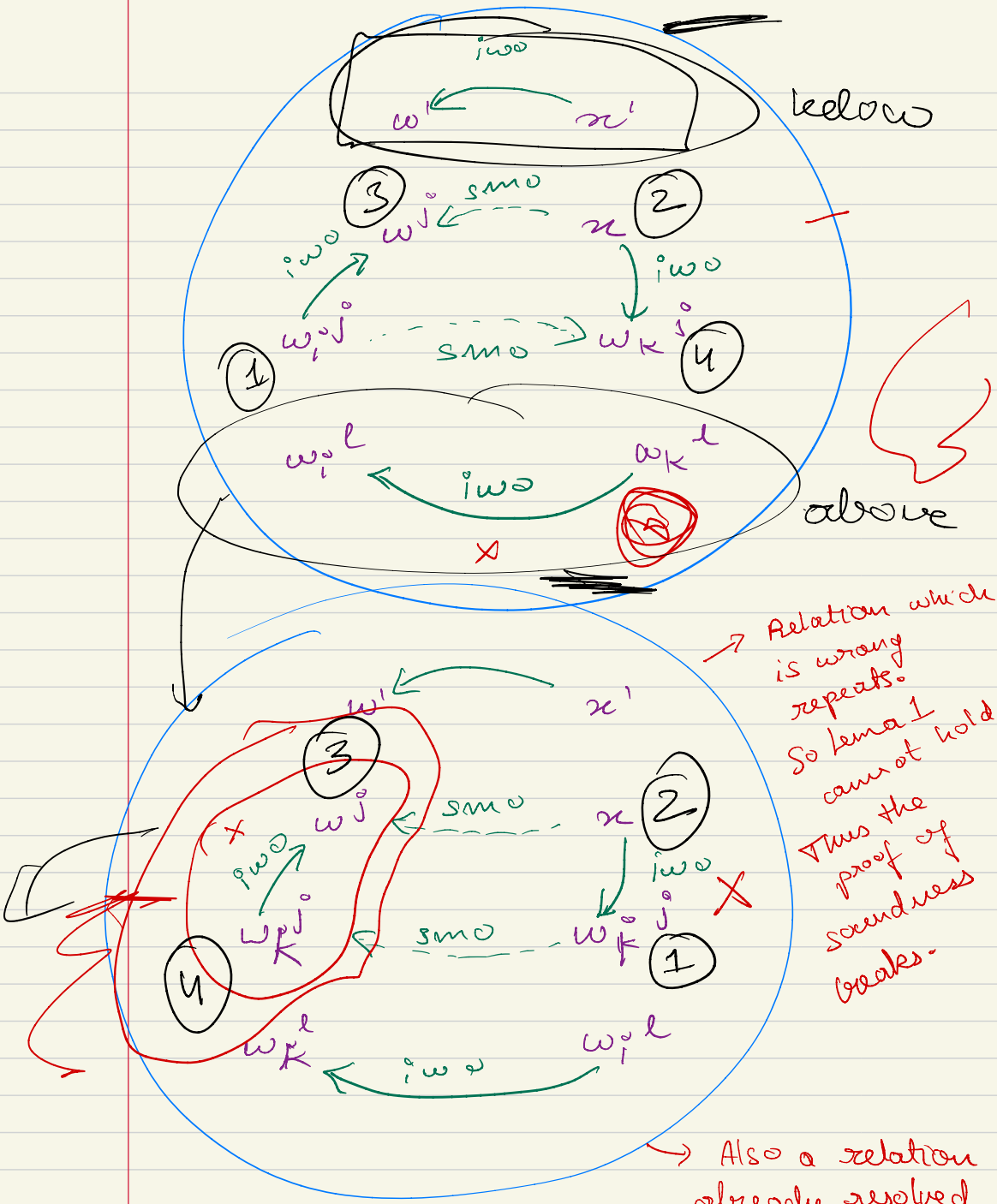
$$\omega_k' \xleftarrow{\text{iwo}} x'$$

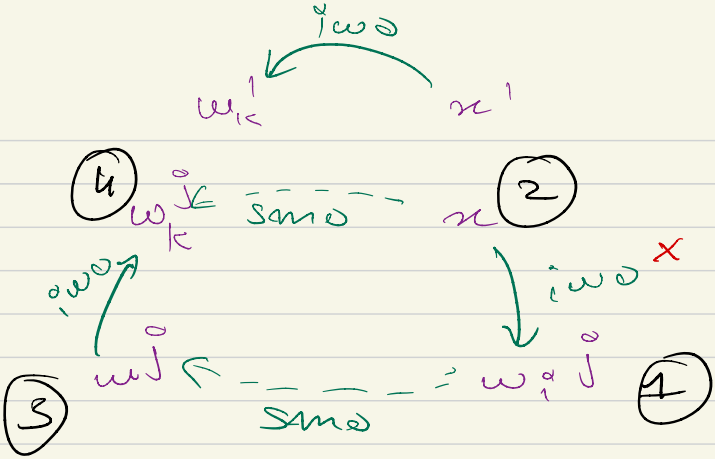
$$\begin{tikzcd}
\omega_k^j & & x \arrow[ll, dashed, "\text{smo}"'] \arrow[d, "\text{iwo}"] \\
\omega^j \arrow[u] & & \omega_i^j \arrow[ll, dashed, "\text{smo}"]
\end{tikzcd}$$

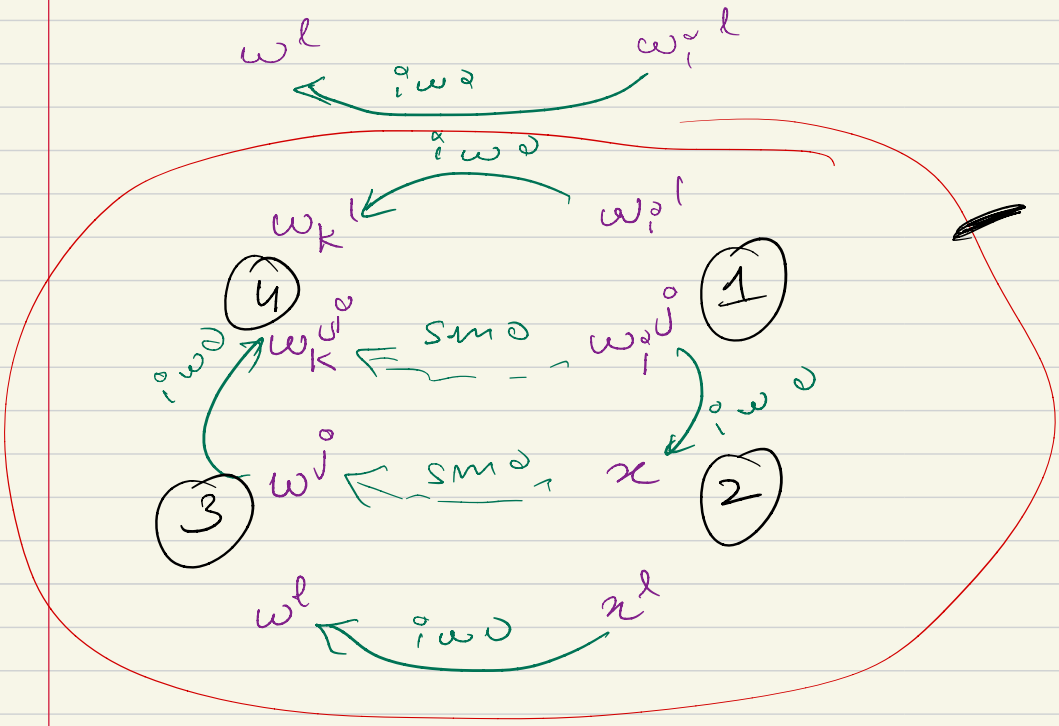
$$\omega^j \xleftarrow{\text{iwo}} \omega_i^l$$


original config.
iwo
smo
iwo
smo
iwo
1
2
3
4

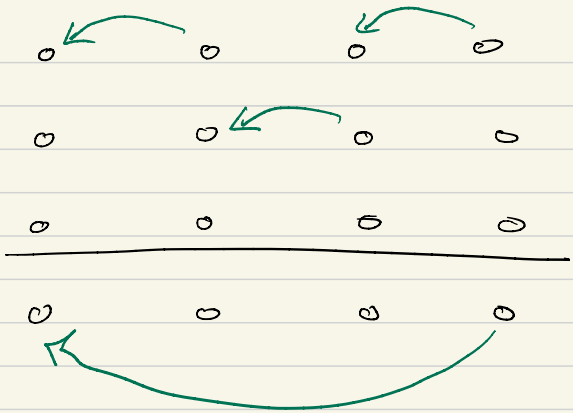

iwo
smo
iwo
smo
iwo
1
2
3
4

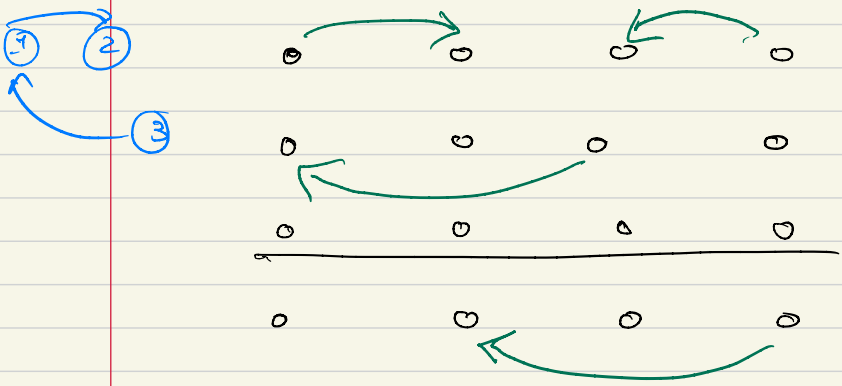

below
above
iwo
smo
Relation which is wrong repeats. So lemma 1 cannot hold. Thus the proof of soundness breaks.
Also a relation already resolved comes up.

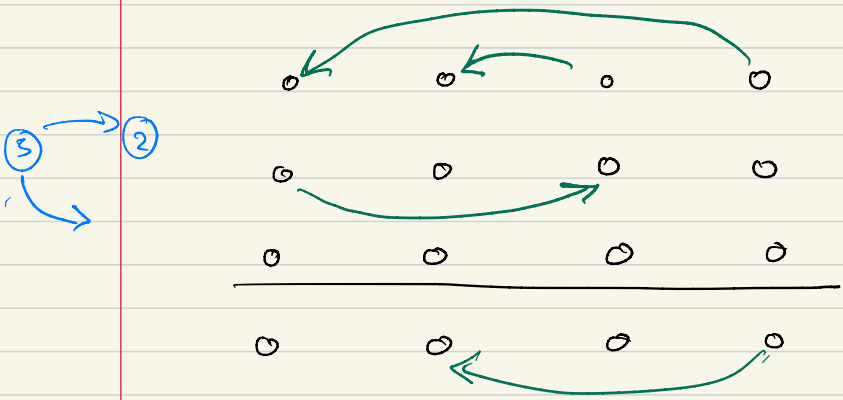


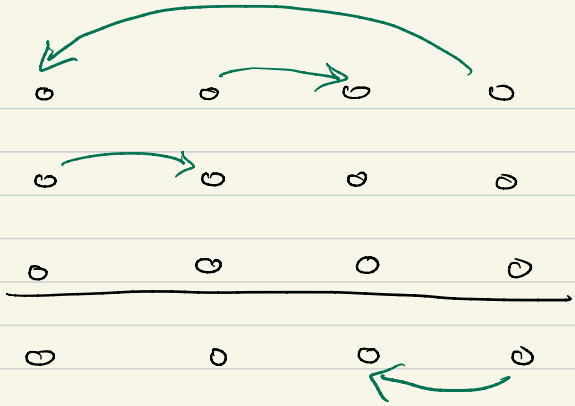


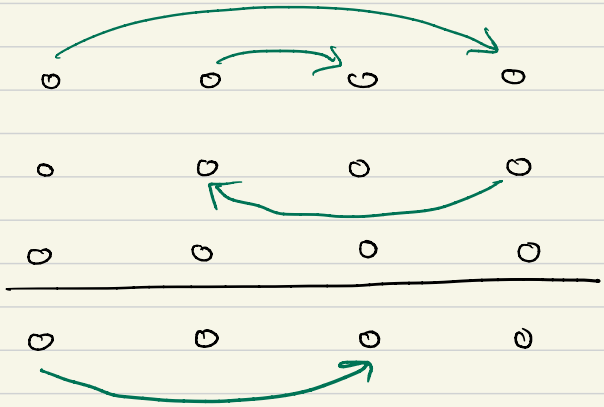
* Problem is it still resolves itself fine. So most probably my proof method is faulty.....

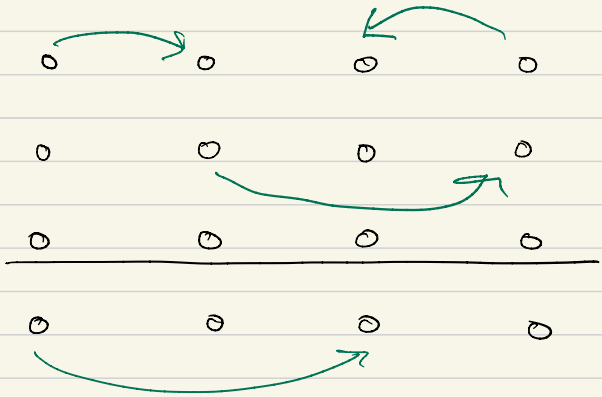



1
2
3


5
2







* What if we only concern ourself to fixing above set of writes first. Then going to set of writes below?

→ Lemma 1 holds nevertheless. So sorting one set of writes can be done.

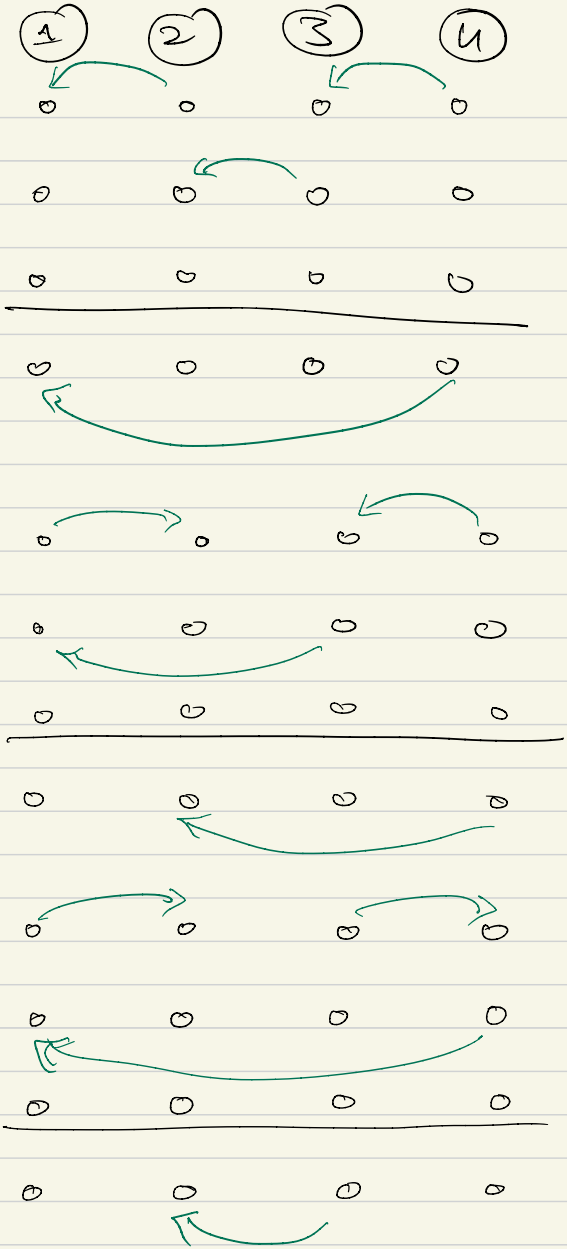
→ <u>Further Issues:-</u>

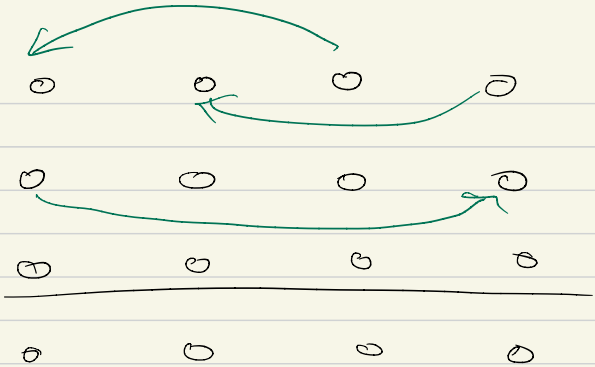
→ Considering one set above sorted, sort next set above

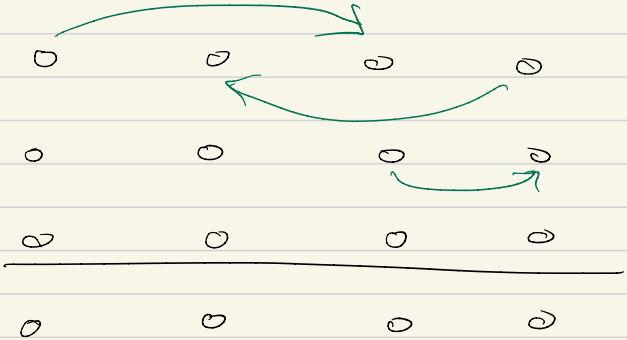
→ Considering above sorted, sort one set below.

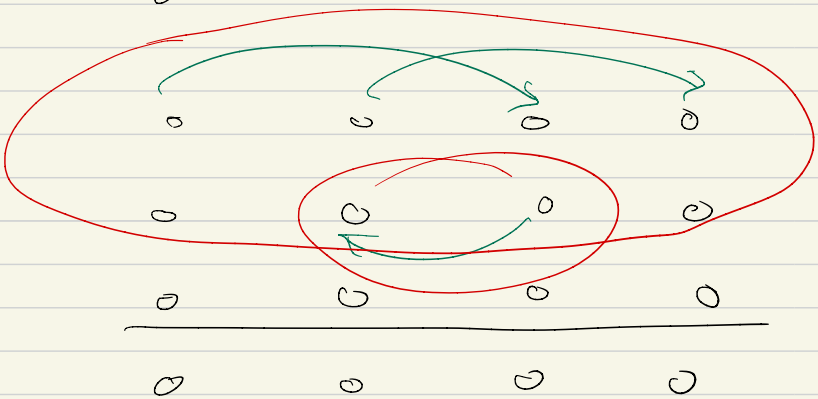
→ Considering below one set sorted, sort next set below.

If I fix two betw writes above read


1
2
3
4





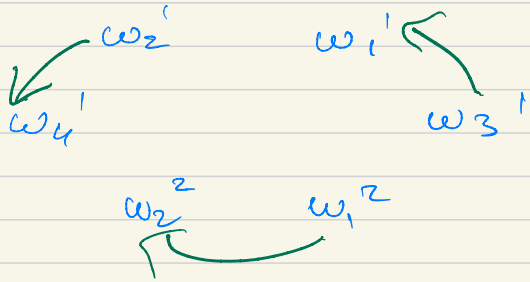


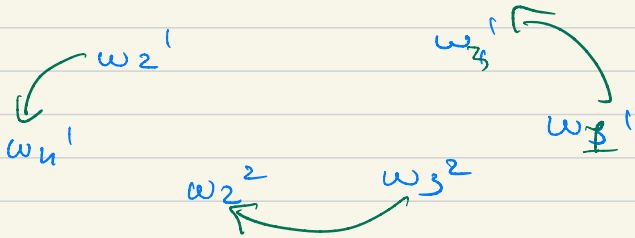
$$\begin{tikzcd}
\omega_2^1 \arrow[d] & \omega_3^1 \\
\omega_1^1 & \omega_4^1 \arrow[u] \\
\omega_2^2 & \omega_3^2 \arrow[l]
\end{tikzcd}$$

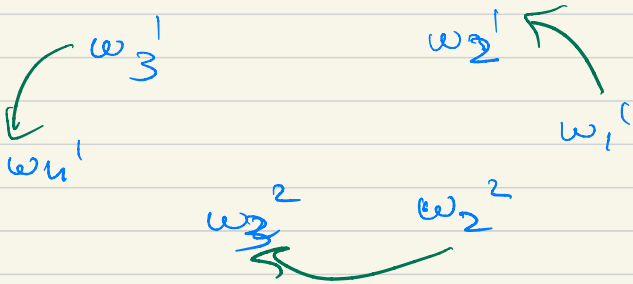
$$\begin{tikzcd}
\omega_1^1 \arrow[d] & \omega_3^1 & \omega_4^1 \arrow[l, "i\omega_0"'] \\
\omega_2^1 & & \\
\omega_1^2 & \omega_3^2 \arrow[l, "i\omega_2"]
\end{tikzcd}$$

$$\begin{tikzcd}
\omega_1^1 \arrow[d] & \omega_4^1 & \omega_3^1 \arrow[l] \\
\omega_2^1 & & \\
\omega_1^2 & \omega_4^2 \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}
\omega_4^1 \arrow[d] & \omega_1^1 & \omega_3^1 \arrow[l] \\
\omega_2^1 & & \\
\omega_4^2 & \omega_1^2 \arrow[l]
\end{tikzcd}$$







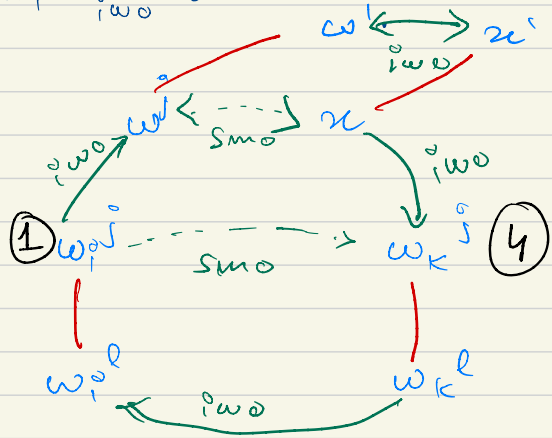
★ Once fixed by swapping, it is fixed.

Choose even algo to sort these orders.

Proof Cases (Soundness Last Part)

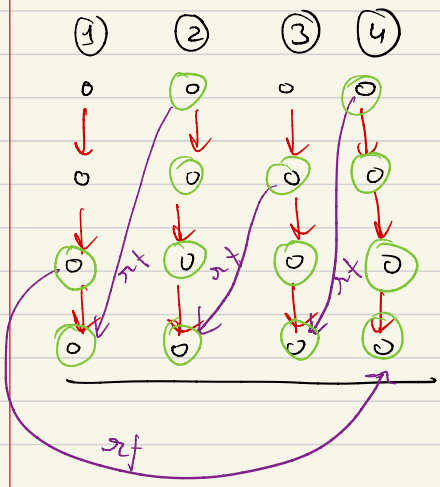
(1) Both writes above read :-

(A) $w_i^j \xrightarrow{iwo} w^j$



(a) $w^j \xrightarrow{smo} x$
(2) (3)

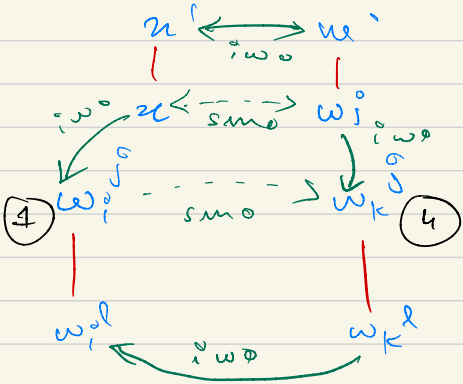
(b) $x \xrightarrow{smo} w^j$
(2) (3)



pour rf⁻¹ cycle.

x' invalid by (Prop 7)

(B) $w^j \xrightarrow[smo]{} w_k^j \quad \wedge \quad x \xrightarrow[iwo]{} w_i^j$



(a) $x \xrightarrow[smo]{} w^j$

2 3

x' invalid by (prop 7)

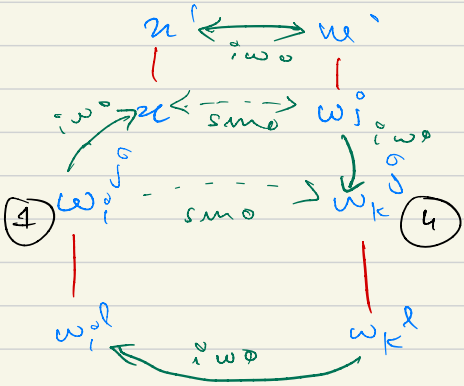
(4) $w^j \xrightarrow[smo]{} x$

2 3

w' invalid due to (prop 7)

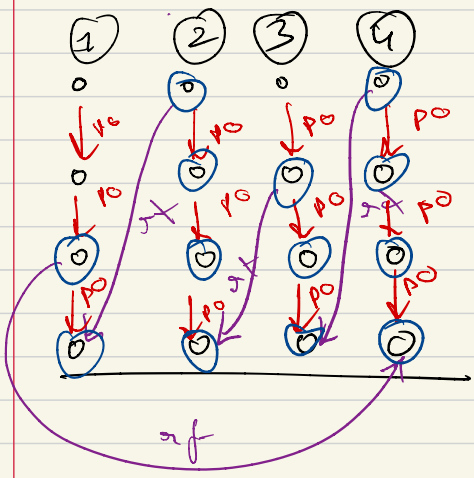
* pourf^y cycle..

(B) $w^i \xrightarrow[smo]{} w_k^j \quad \wedge \quad w_i^j \xrightarrow[iwo]{} x$



(a) $x \xrightarrow[smo]{} w^j$
2 3

(4) $w^i \xrightarrow[smo]{} x$
2 3



w' invalid due to (prop 7)

(2) Fixed writes Above read, to be fixed below.

(A) $w^j \xrightarrow[iwo]{} w_k^j$

$$\begin{tikzcd}
 & x' \arrow[r, leftrightarrow, "iwo"] \arrow[d, dash] & w_j^{i\,\prime} \arrow[d, dash] & \\
 & x \arrow[r, leftrightarrow, dashed, "smo"] & w_j^i \arrow[d, "iwo"] & \\
(1)\ w_i^j \arrow[d] \arrow[rr, dashed, "smo"'] \arrow[uur, "iwo"] & & w_k^j\ (4) \arrow[d] & \\
w_i^l & & w_k^l \arrow[ll, "iwo"] &
\end{tikzcd}$$

(a) $w^j \xrightarrow[smo]{} x$
(2) (3)

x' invalid due to prop 7

(b) $x \xrightarrow[smo]{} w^j$
(2) (3)

x' invalid due to prop 7

(3) Fixed writes above & below read. To be fixed Below

(A) $w_i^j \xrightarrow[iwo]{} w^j \wedge$ w', x' above read.

$$w' \xleftrightarrow[iwo]{} x'$$

$$\begin{tikzcd}
 & w^j \arrow[r, dashed, leftrightarrow, "smo"'] & x \\
w_i^j \arrow[ur, "iwo"] \arrow[rr, dashed, "smo"'] & & w_k^j \arrow[u, "iwo"']
\end{tikzcd}$$

$$\begin{tikzcd}
w_i^l & & w_k^l \arrow[ll, bend left, "iwo"]
\end{tikzcd}$$

(a) $w^j \xrightarrow[smo]{} x$

w' invalid by prop 8

(u) $x \xrightarrow[smo]{} w^j$

x' invalid by prop 8,

(B) $w_i^0 \xrightarrow[iwo]{} w^i \wedge x', w'$ below read

$$w' \xleftrightarrow[iwo]{} x'$$

$$\begin{tikzcd}
 & w^i \arrow[r, dashed, leftrightarrow, "smo"'] & x \\
w_i^0 \arrow[ur, "iwo"] \arrow[rr, dashed, "smo"'] & & w_k^i \arrow[u, "iwo"']
\end{tikzcd}$$

$$\begin{tikzcd}
w_i^l & & w_k^l \arrow[ll, bend left, "iwo"]
\end{tikzcd}$$

(a) $w_i^0 \xrightarrow[smo]{} x$

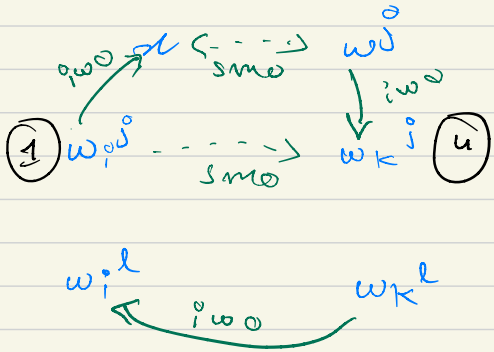
x' invalid by prop 8

(b) $x \xrightarrow[smo]{} w^i$

w' invalid by prop 8

(c) $w_i^j \xrightarrow[\text{iwo}]{} w_k^j \wedge x', w'$ below read

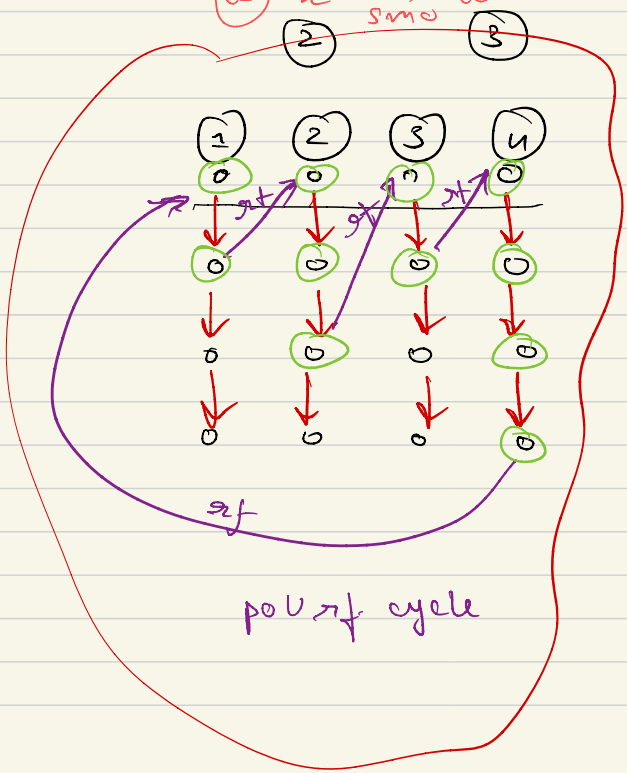
$$x' \xleftrightarrow[\text{iwo}]{} w'$$



(a) $w_i^j \xrightarrow[\text{smo}]{} x$

x' invalid by prop 8

(b) $x \xrightarrow[\text{smo}]{} w_i^j$



(D) $w^j \xrightarrow[\text{iwo}]{} w_k^j \wedge x', w'$ above read

$$x' \xleftrightarrow[\text{iwo}]{} w'$$

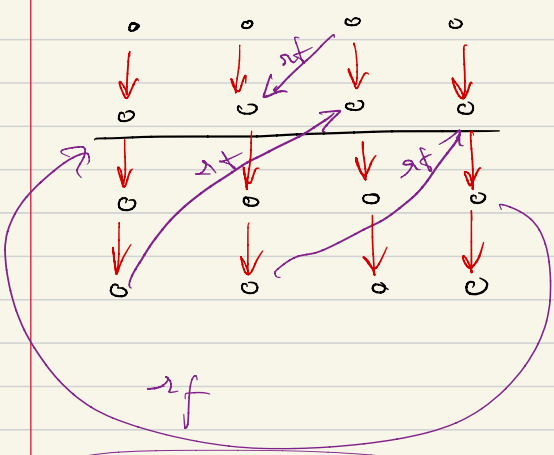
$$\begin{tikzcd}
 & x \arrow[r, dashed, leftrightarrow, "\text{smo}"'] & w^j \arrow[d, "\text{iwo}"] \\
1 \; w_i^j \arrow[ur, "\text{iwo}"] \arrow[rr, dashed, "\text{smo}"'] & & w_k^j \; u
\end{tikzcd}$$

$$w_i^l \xleftarrow{\text{iwo}} w_k^l$$

(a) $w^j \xrightarrow[\text{smo}]{} x$ (2) (3)

(b) $x \xrightarrow[\text{smo}]{} w^j$ (2) (3)

x' invalid by prop 7



* We cannot have resolved $w^j \to w_i^j$ before.

$$\begin{tikzcd}
 & & x' \arrow[d, "i\omega_0"] & w' \arrow[l] \\
 & x \arrow[r, dashed, "s\omega_0"] & w_0^0 \arrow[d, "i\omega_0"] & \\
1 \quad w_i^0 \arrow[ur, "i\omega_0"] \arrow[r, dashed, "s\omega_0"] & & w_k^j & \quad u \\
w_i^l & & w_k^l \arrow[ll, "i\omega_0"]
\end{tikzcd}$$

Above

Below.

Below

Above

$$\begin{tikzcd}
w_0^0 \arrow[r] & w_i^j
\end{tikzcd}$$

$$\begin{tikzcd}
 & & x' \arrow[d, "i\omega_0"] & w' \arrow[l] \\
 & x \arrow[r, dashed, "s\omega_0"] & w_0^0 \arrow[d, "i\omega_0"] & \\
1 \quad w_i^0 \arrow[ur, "i\omega_0"] \arrow[r, dashed, "s\omega_0"] & & w_k^j & \quad u \\
w_i^l & & w_k^l \arrow[ll, "i\omega_0"]
\end{tikzcd}$$

$$\begin{tikzcd}
 & & x' \arrow[d, "i\omega_0"] & w' \arrow[l] \\
 & x \arrow[r, dashed, "s\omega_0"] & w_0^0 \arrow[d, "i\omega_0"] & \\
1 \quad w_i^0 \arrow[ur, "i\omega_0"] \arrow[r, dashed, "s\omega_0"] & & w_k^j & \quad u \\
w_i^l & & w_k^l \arrow[ll, "i\omega_0"]
\end{tikzcd}$$

Below

(E) w^i above read, $w^i \xrightarrow[iwo]{} w_k^j$, x & w' below read.

$$x^l \xleftrightarrow{iwo} w'^l$$

$$\begin{tikzcd}
 & x \arrow[r, dashed, leftrightarrow, "smo"'] \arrow[dl, "iwo"'] & w'^l \arrow[dr, "iwo"] & \\
w_i^j \arrow[rr, dashed, "smo"'] & & & w_k^j
\end{tikzcd}$$

$$w_i^l \xleftarrow{iwo} w_k^l$$

(a) $w^i \xrightarrow[smo]{} x$

x invalid by prop 8

(b) $x \xrightarrow[smo]{} w^i$

w^i invalid by prop 8.

(F) w^j above read, $w^j \xrightarrow[iwo]{} w_k^j$, $x^i \wedge w^i$ above read.

$$x^i \xleftrightarrow[iwo]{} w^i$$

$$\begin{tikzcd}
 & x \arrow[dl, "iwo"'] & w^j \arrow[l, dashed, "smo"'] \arrow[dr, "iwo"] & \\
w_i^j \arrow[rr, dashed, "smo"'] & & & w_k^j
\end{tikzcd}$$

$$\begin{tikzcd}
w_i^l & & w_k^l \arrow[ll, "iwo"']
\end{tikzcd}$$

(a) $w^j \xrightarrow[smo]{} x$

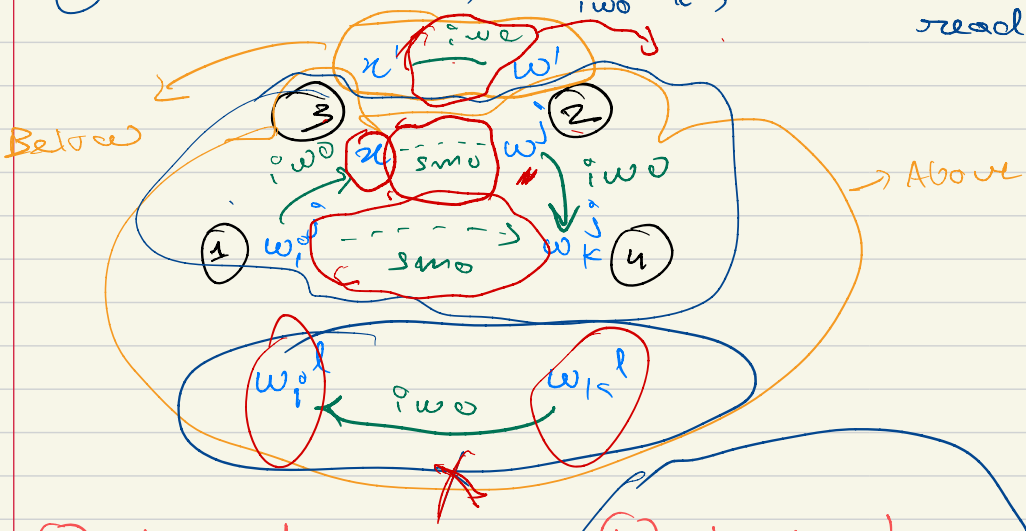
w^i invalid by prop 7

(b) $x \xrightarrow[smo]{} w^j$

x^i invalid by prop 7.

(4) Fixed writes above & some below, to be fixed above. (Can happen if new iwo est after fixing another iwo)

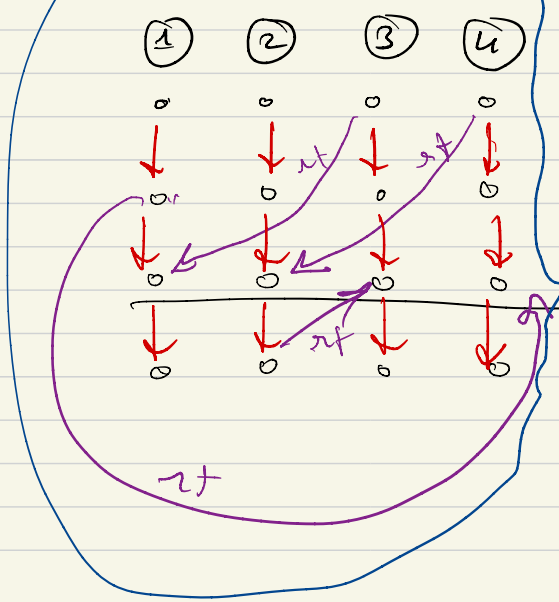
(A) w^i above read, $w^i \xrightarrow[iwo]{} w^j_k$, $x' \wr w'$ below read.

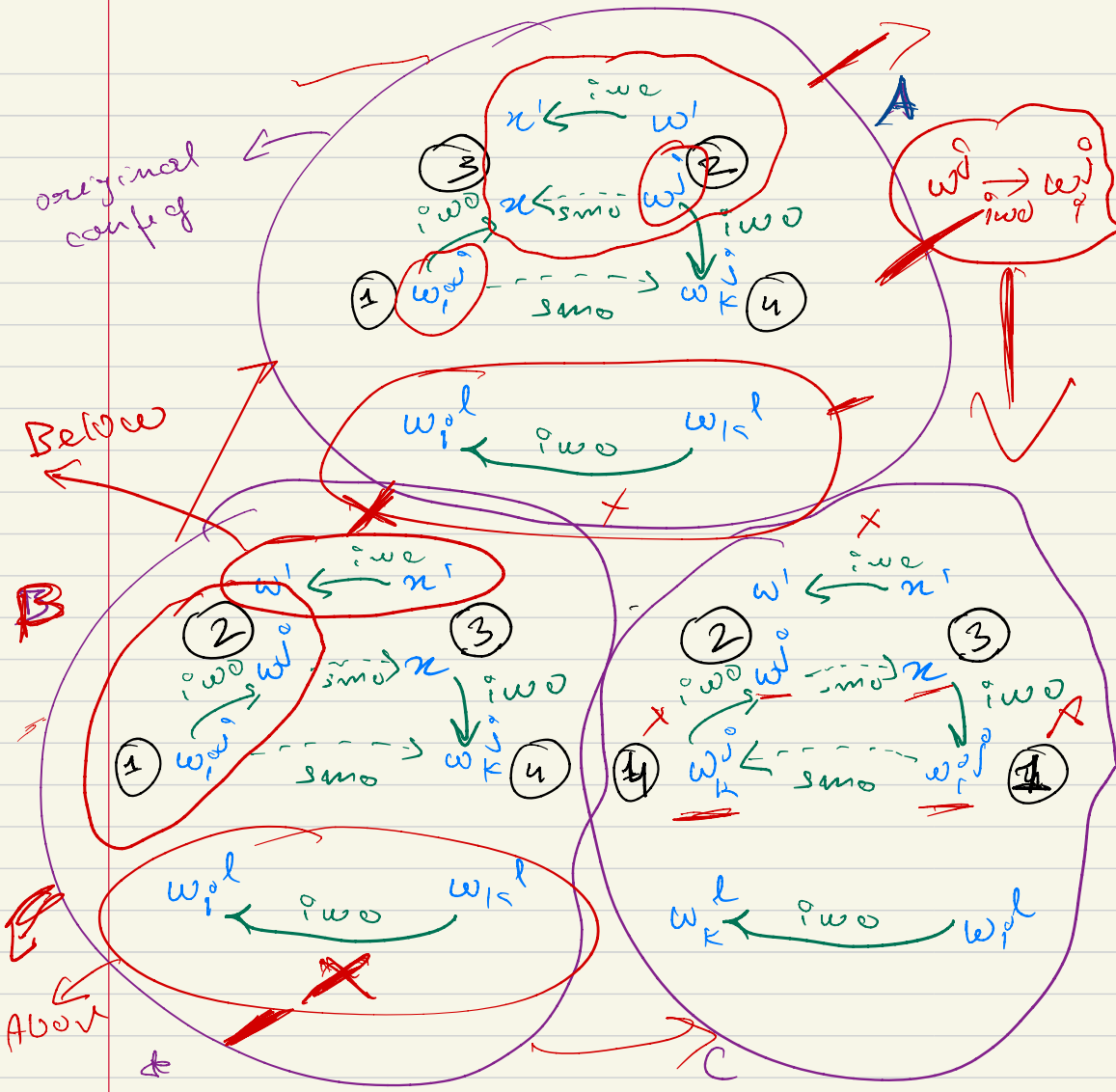


(a) $x' \xrightarrow[smo]{} w'$

w' invalid by prop 8,

(b) $w' \xrightarrow[smo]{} x'$





* From config B, we would only go to config C, as x', w' below and x & w^l above reach.

(B) w^j below read, x', w' above read,
$w_i^j \xrightarrow[iwo]{} w^j$

$$w' \overset{iwo}{\text{———}} x'$$

$$\begin{tikzcd}
 & w^j \arrow[r, dashed, "smo"] & x \\
w_i^j \arrow[ur, "iwo"] \arrow[rr, dashed, "smo"'] & & w_k^j \arrow[u, "iwo"']
\end{tikzcd}$$

$$w_i^l \xleftarrow{iwo} w_k^l$$

(a) $w^j \xrightarrow[smo]{} x$

w' invoked by prop 8

(b) $x \xrightarrow[smo]{} w^j$

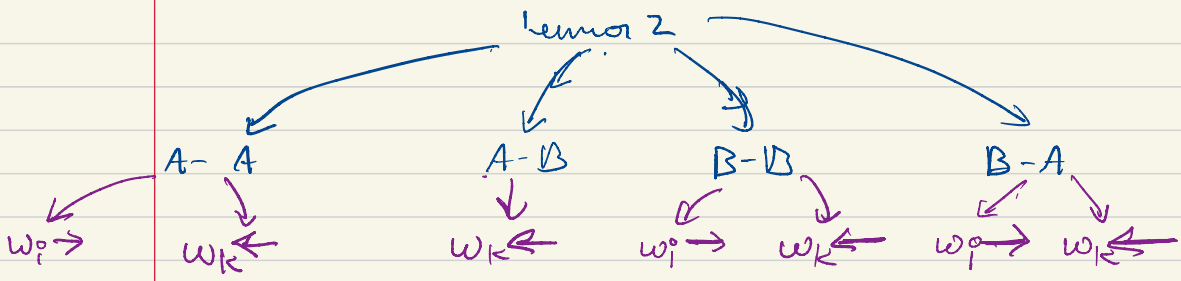
x' invoked by prop 7

THE END OF
ALL POSSIBLE
CASES OF
SOUNDNESS.

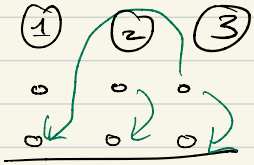
* What if we get a new iwo & then we fix that. Won't the order change then? (Fair point)

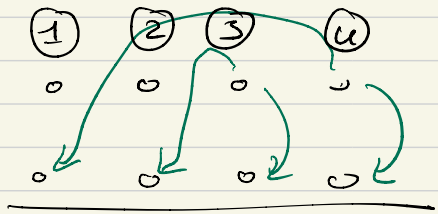
$$\begin{tikzcd}
 & x' \arrow[d, "p^\circ"] & w' \arrow[l, bend right, "{}^\circ iwp"'] \arrow[d, "p^\circ"] \\
 & x & w_j^\circ \arrow[l, dashed, "smo"'] \arrow[d, bend left, "iwo"] \\
w_i^{\circ j} \arrow[uu, bend left, "{}^\circ iwo"] \arrow[r, dashed, "smo"] \arrow[d, "p^\circ"] & w_k^{\circ j} \arrow[d, "p^\circ"] & \\
w_i^{\circ l} & w_k^{\circ l} \arrow[l, bend left, "iwo"] &
\end{tikzcd}$$

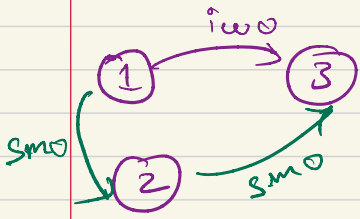
A) We can show for each case that there will be no problem.

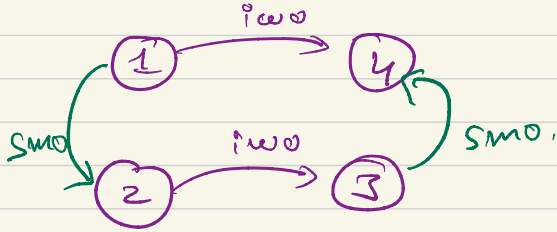

Lemma 2
A-A
A-B
B-B
B-A
w_i°
w_k
w_k
w_i°
w_k
w_i°
w_k

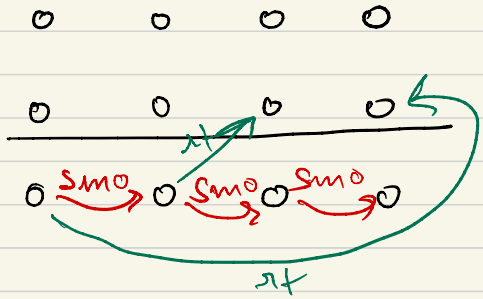
Short Meeting (Viktor)

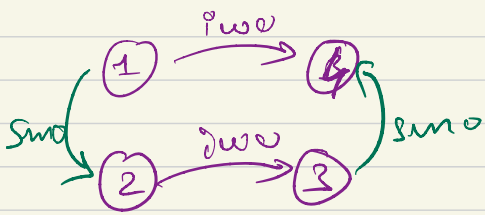












iwo above read

iwo; smo; iwo^{-1}

smo; iwo^{-1}

iwo^{-1}; smo; iwo

iwo below read

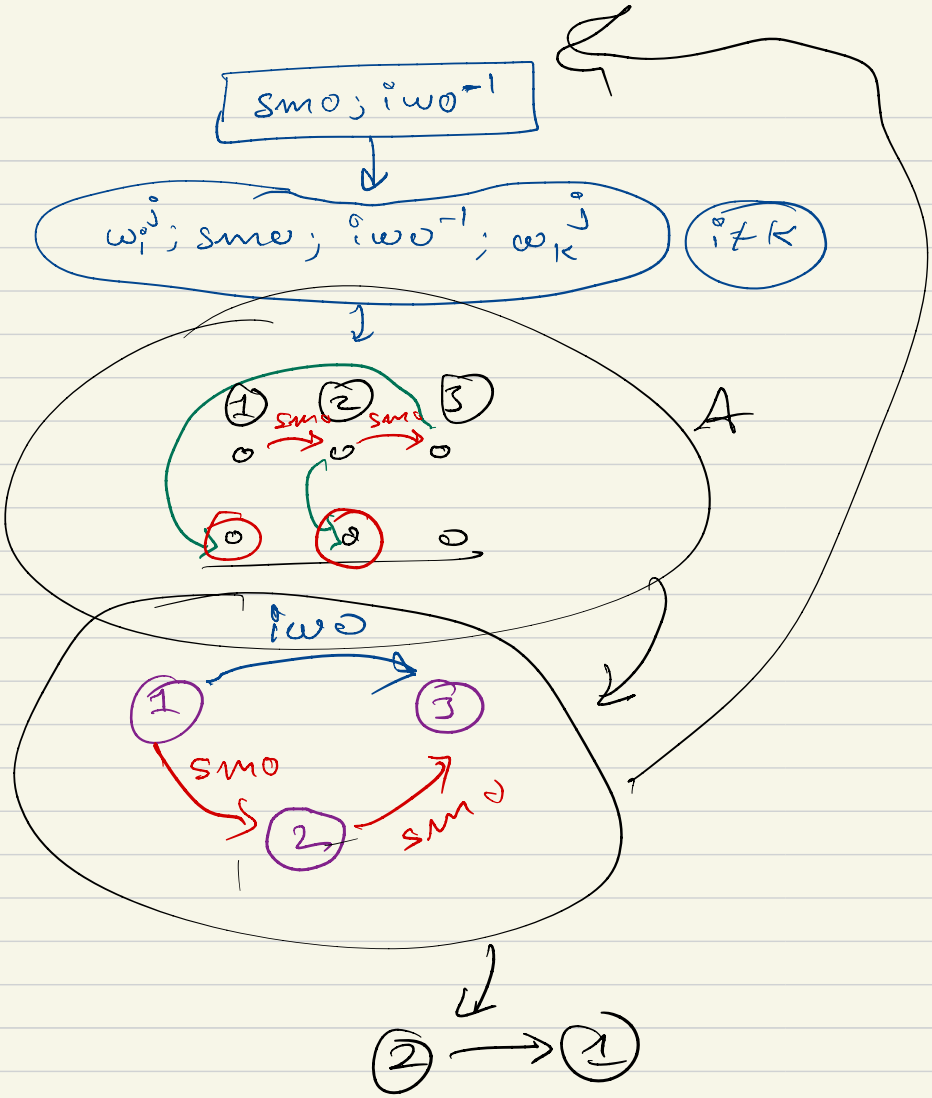
Reorders

1. How long will it take for the admission process?

2. Azalea's meeting was informative about scholarships .. at UK (not good option for me)

3. what's your feedback on the proof if you have got time to look at it?

4. Read order defn & approach to go

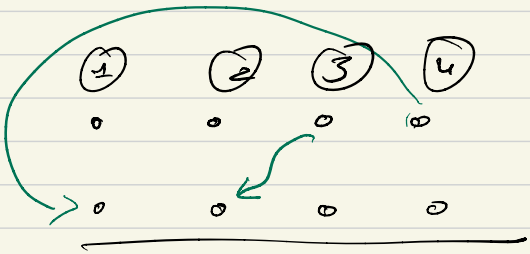

smo; iwo^-1
w_i^j; smo; iwo^-1; w_k^j
i≠k
A
1
2
3
smo
smo
iwo
1
3
smo
2
smo
2
1

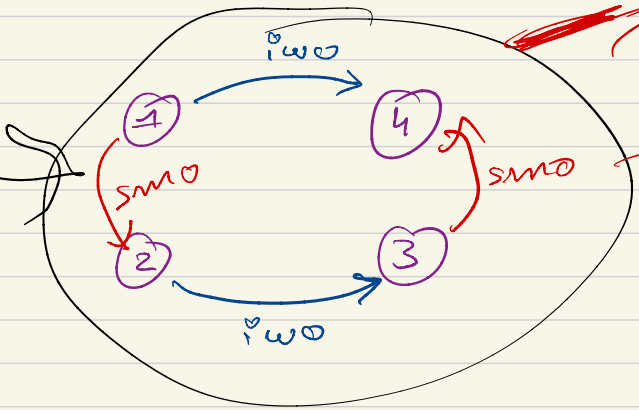
$i\omega\sigma ; s\omega\sigma ; i\omega\sigma^{-1}$

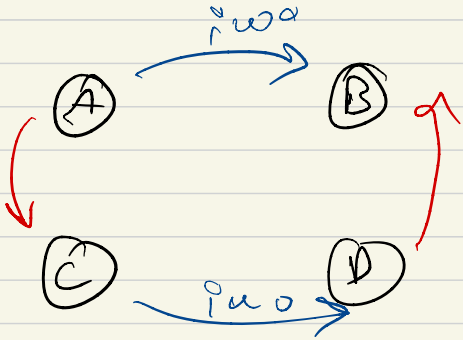
$\downarrow$

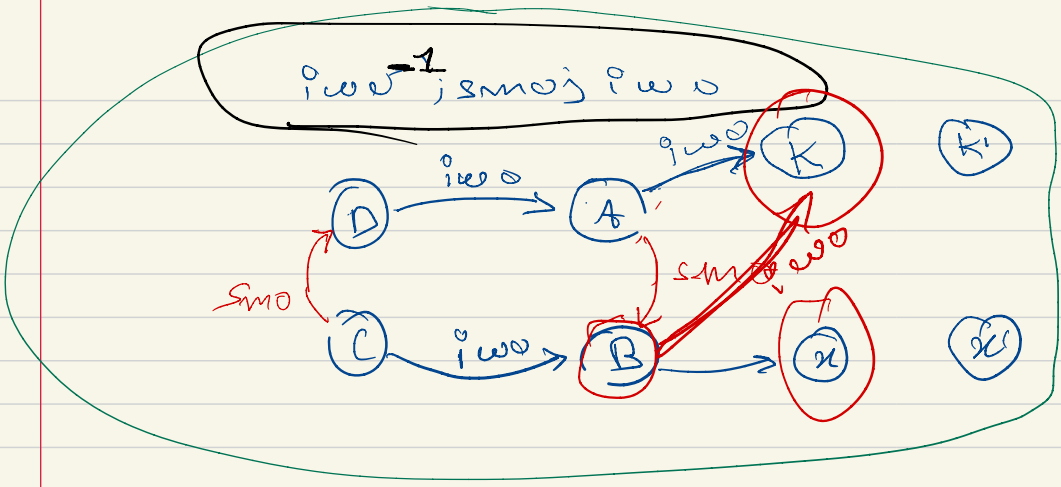
$\omega_l^{\sigma} ; i\omega\sigma^{-1} ; \omega_k^{\sigma} ; s\omega\sigma ; i\omega\sigma ; \omega_i^{\sigma}$

$i \neq k \neq l.$





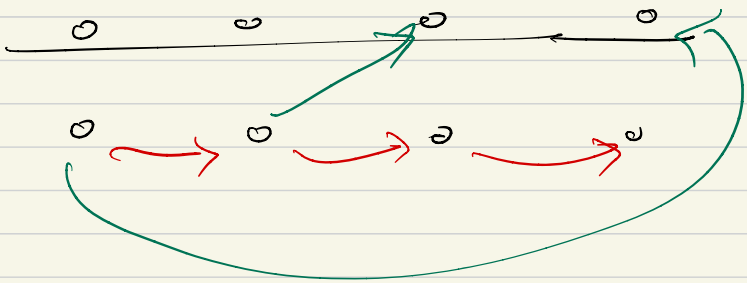




* Conclusion is that the new relation can never have existed before.

* What if we need to fix it?

Will it introduce some new implied mod orders?

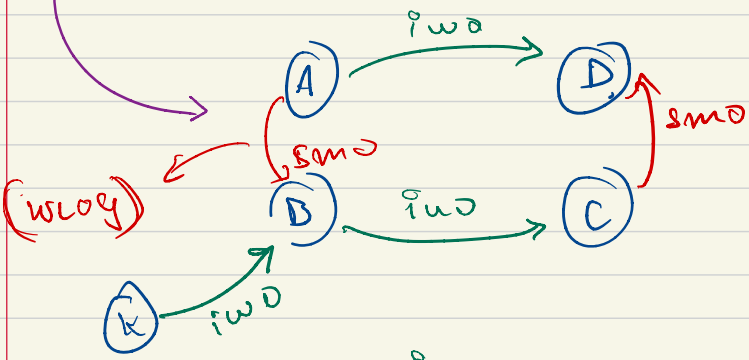


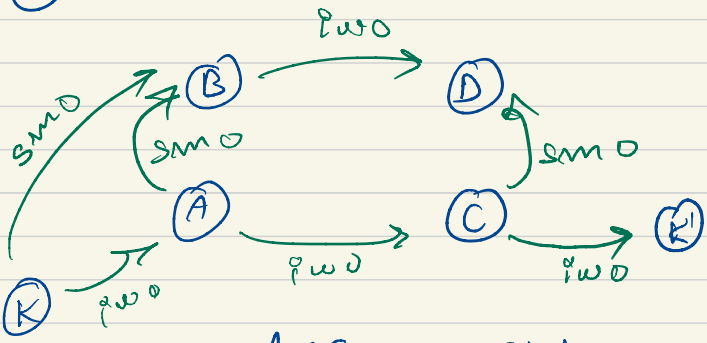
READ ORDER STABILITY

(A) smo ; iwo^{-1}

(B) iwo ; smo ; iwo^{-1}

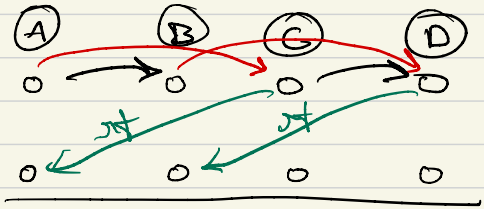
(C) iwo^{-1} ; smo ; iwo

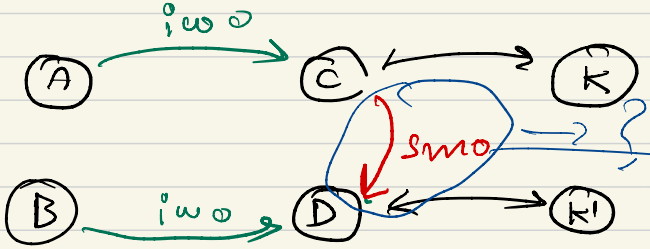


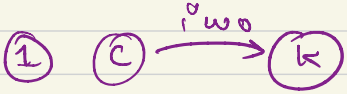


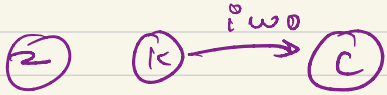
A < C C > K

Can I have a set of transformations that can reverse this?









$A \xrightarrow[iwo]{} K \xrightarrow[iwo]{} C$

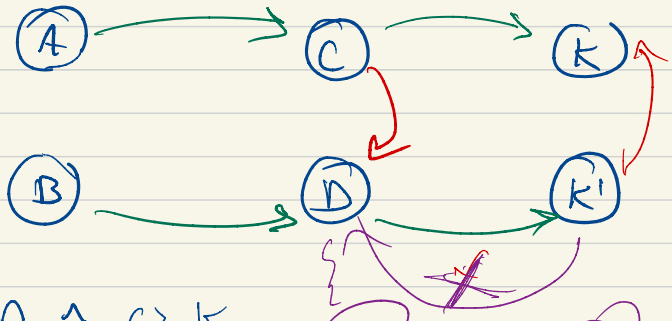
$K > C > A$

$A < K$

if wrong we fine, then $A \xrightarrow[iwo]{} B$ lost.

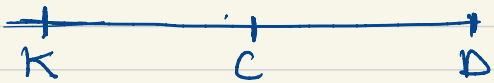
or

Whether this SMO is reversed? ??? ??

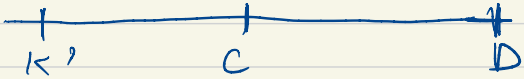


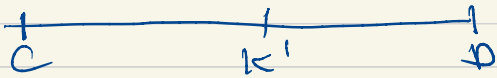
$C < D \wedge C > K$

$\boxed{C < D \wedge C > K}$ $\longrightarrow$ $K < C < D$



$D > C \wedge D > K'$



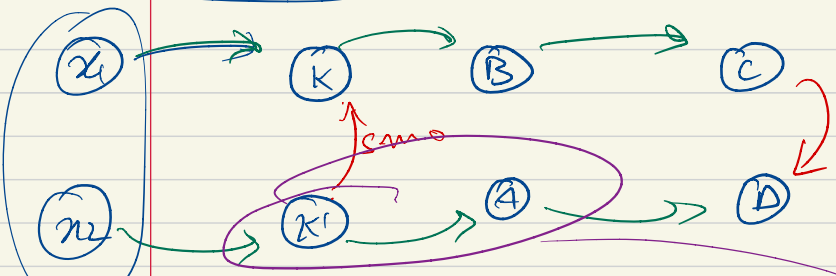


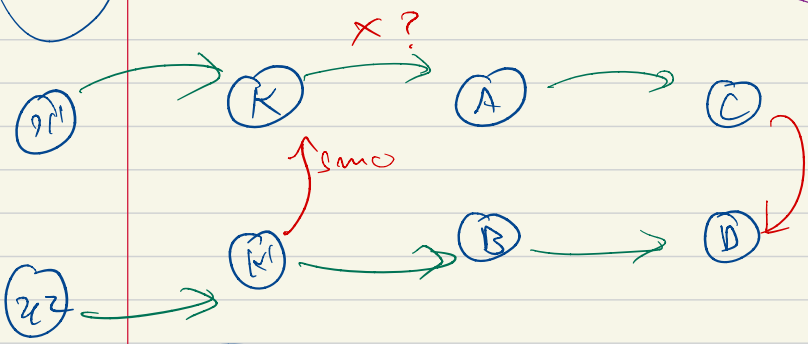
$C \xrightarrow{\text{sm}} K$

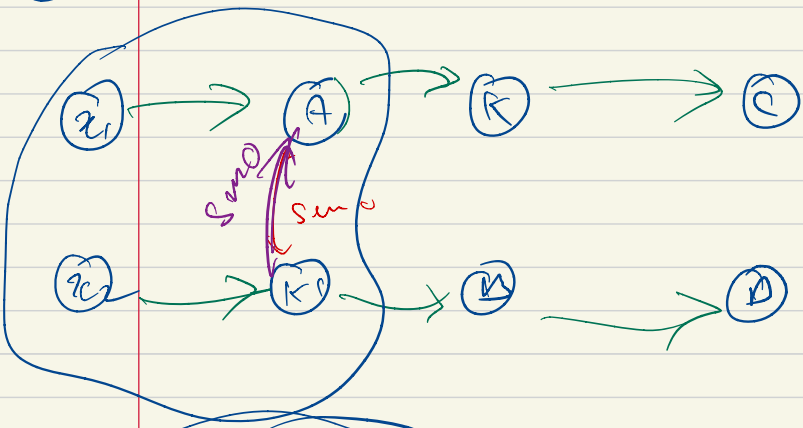
$D > K' \wedge D > C$

$K' < C$

$x_2 \xrightarrow[\text{iso}]{} x_1$ ✓





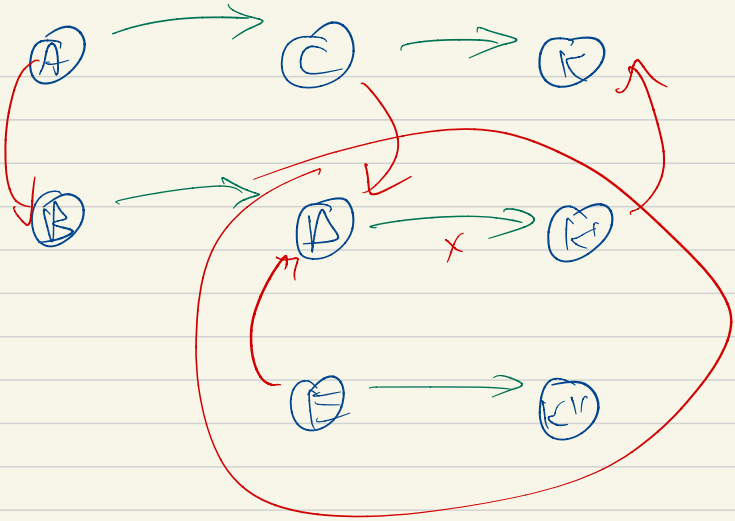


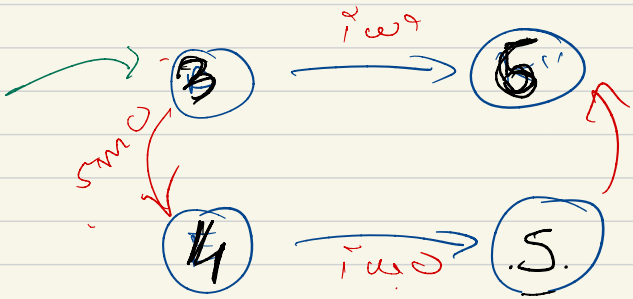
$x_1 \xrightarrow[\text{iso}]{} x_2$ ✗ Not Stable!!!

$x_2 < x_1$

$A > K'$ $\quad$ $A < B$

$K > A$ $\quad$ $K' < B$


A
C
K
B
D
K'
x
E
K''


3
6
4
5

Counter Example to Show Implied Read Orders are Not Stable.

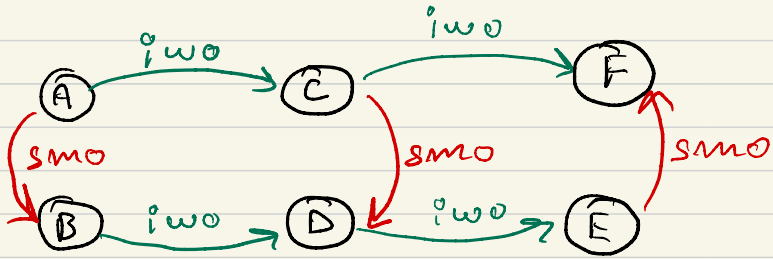
1. Implied read orders are constructed using implied write orders with some other external write.

2. If the set of external write involved changes, the implied read order may change.

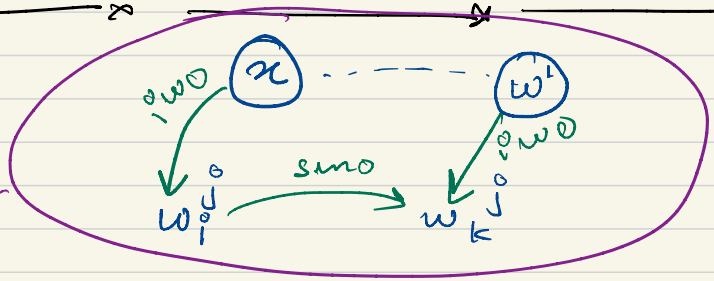
3. This is why one can easily construct a case when implied read orders may not be stable, because the implied write orders used to derive it changes.

4. So in essence, the implied read order does change in terms of its components.

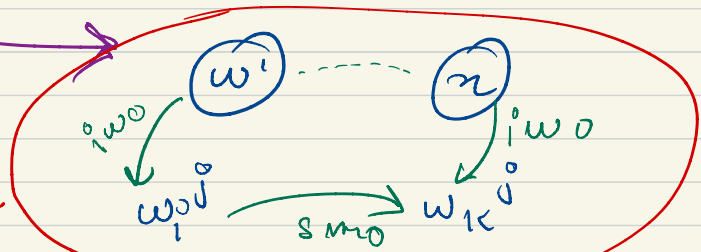
5. In turn, what we can show is that the external writes involved in implied read order are not swopped themselves, which would result in reversal of implied read order with the same external writes involved.



$$A \xrightarrow[iro]{} B \xrightarrow[iwo]{} D \xrightarrow[iro]{} C$$

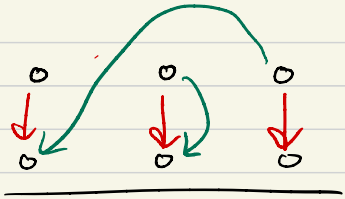


$$w_i^l \xleftarrow{iwo} w_k^l$$



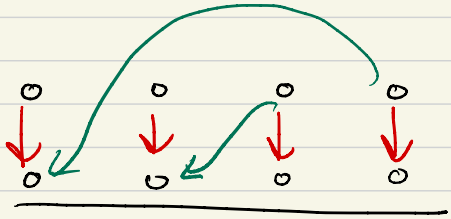
If $w' \xrightarrow[iwo]{} w_i^j$ was incorrect, it must be fixed first before swapping w' & x to fix implied read order btw them.

$T_i, smo; iwo^{-1}$



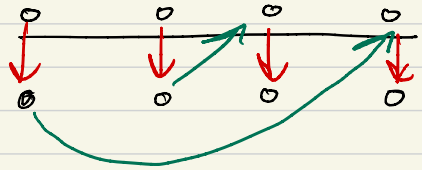
$af^{-1}; smo; af$

$iwo; smo; iwo^{-1}$

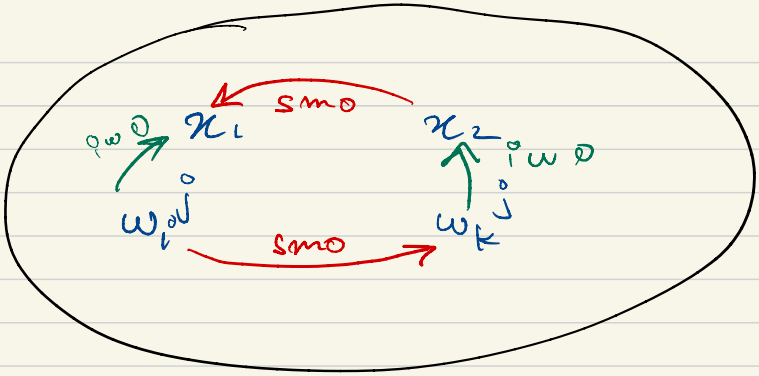


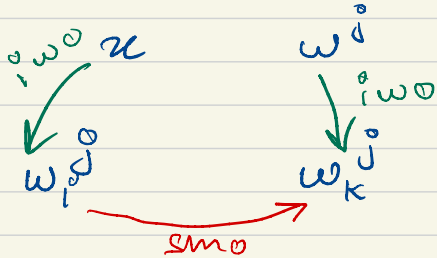
$af^{-1}; smo; af$

$iwo^{-1}; smo; iwo$



$af^{-1}; smo; af$

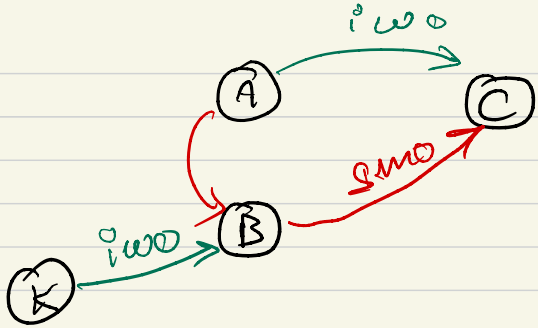


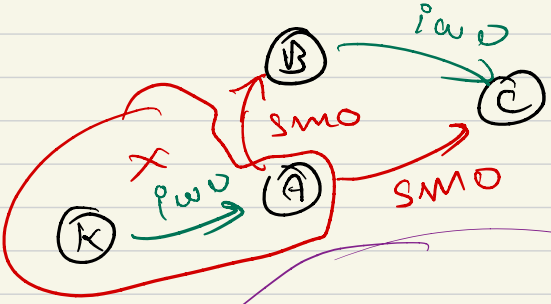


1. x & w^j cannot have an implied order between them.

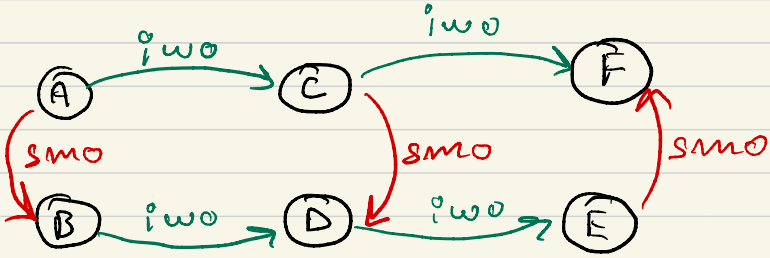
2. There cannot exist any iwo between x & w^j simply because both have their reads satisfied by w_i^j & w_k^j

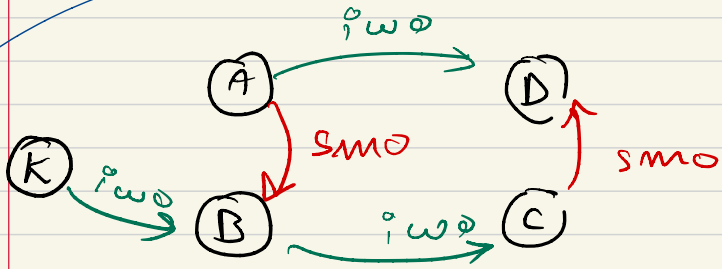
3. If they were swapped to fix an implied read order, then the original config would have had, $w^j \xrightarrow{iwo} w_i^j$ & since this is wrong, it should have been fixed before swapping x & w^j & hence we can come by contradiction that it wasn't fixed.





This pattern is symmetric to our previous case, hence proved by contradiction that $w_i \xrightarrow{iwo} w_j$ was never fixed in the first place.



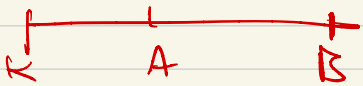


$B \xrightarrow[iro]{} A$ ✗

① $K < A$

$K \xrightarrow[smo]{} A$

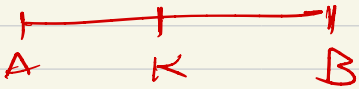
$K \xrightarrow[iro]{} A$



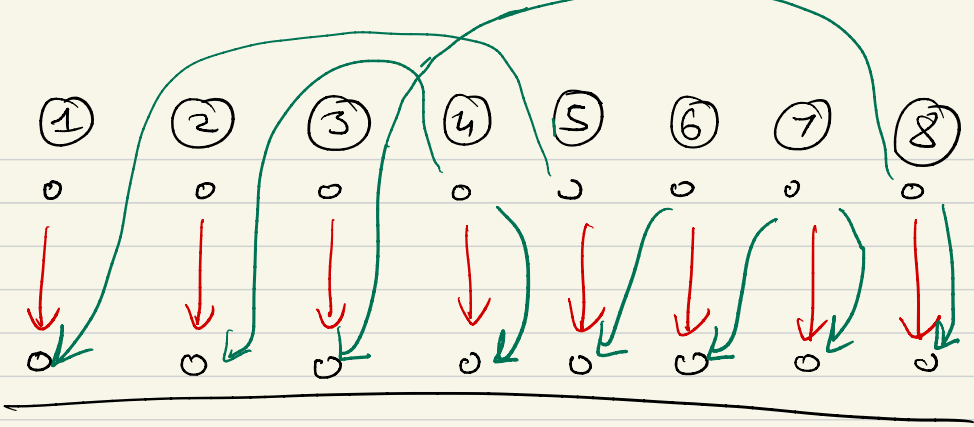
② $K \geq A$

$A \xrightarrow[smo]{} K.$

$K \xrightarrow[iro]{} A$ ✗



Ans that no implied order introduced can be wrong.



1 IRO with 1

$4 \xrightarrow[\text{iro}]{} 1$

$2 \xrightarrow[\text{iro}]{} 1$

2 IRO with 3

$1 \xrightarrow[\text{iro}]{} 3$

$2 \xrightarrow[\text{iro}]{} 3$

$4 \xrightarrow[\text{iro}]{} 3$

$5 \xrightarrow[\text{iro}]{} 3$

$6 \xrightarrow[\text{iro}]{} 3$

$7 \xrightarrow[\text{iro}]{} 3$

* KEY IDEA IS THAT FIXING Implied Read Orders BY FIXING ORDERS OF THE FORM

$$\text{MIN}(x) \xrightarrow[\text{iro}]{} \text{MIN}(y)$$

$$\text{s.t} \quad x > y$$

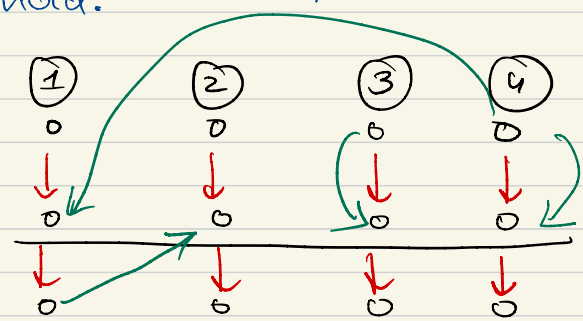
FOR GIVEN CONFIGURATION SEEMS TO BRING STABILITY IN READ ORDERS.

* We can show that implied read orders while being fixed, those writes involved above the read, will not have any new implied write orders which are incorrect.

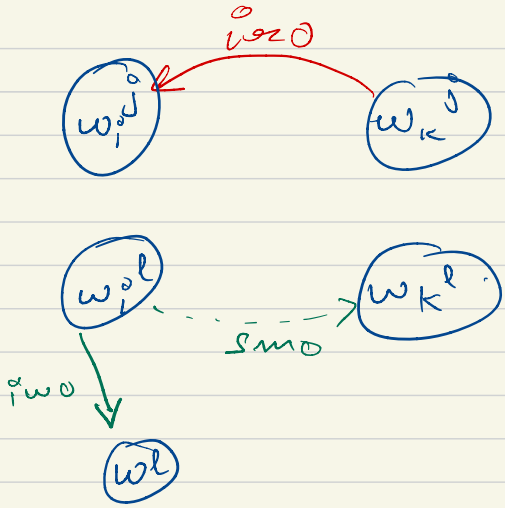
CHECK FIGURE TWO PAGES BEFORE.

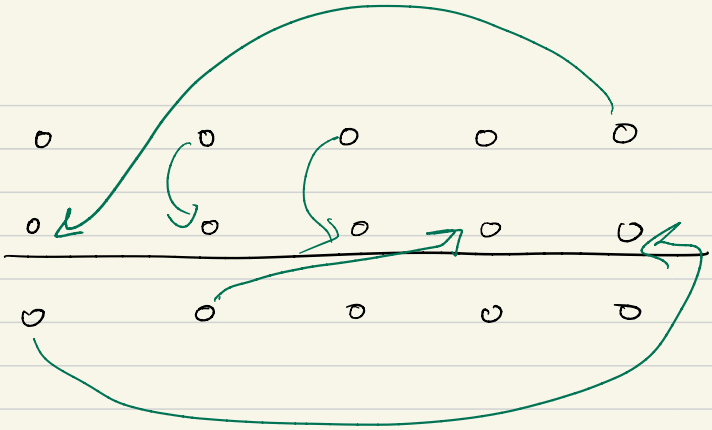
* Similar proof for when writes are only below the read.

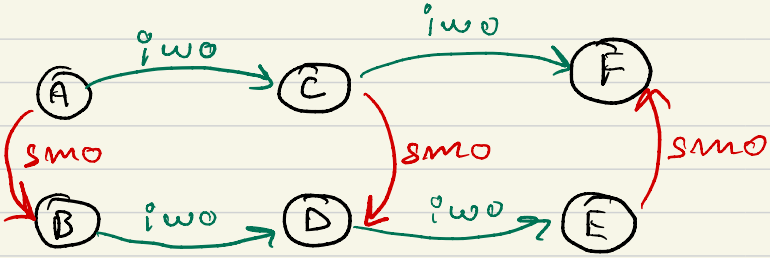
*** Now for a combination of writes both above & below? We have a counterexample when it does not hold.

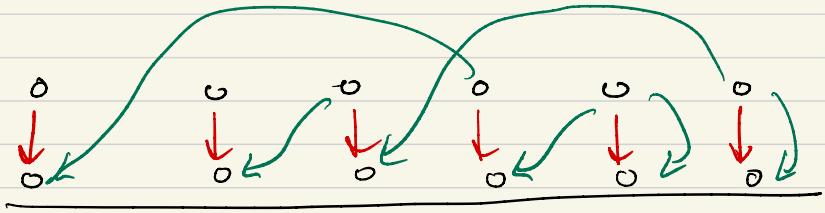


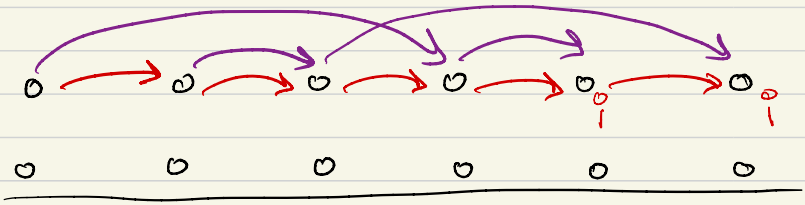
On swapping 1 & 3, we get 3 $\xrightarrow{\text{rwo}}$ 2 between writes below the read.

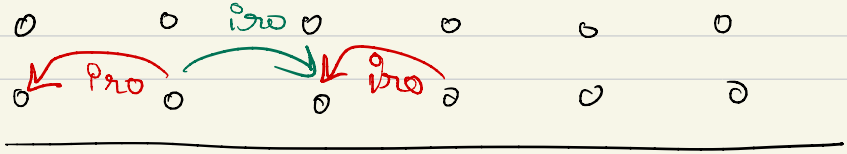

$i \geq 0$
w_i^0
w_k^0
w_i^l
w_k^l
$s \geq 0$
$i \geq 0$
w^l

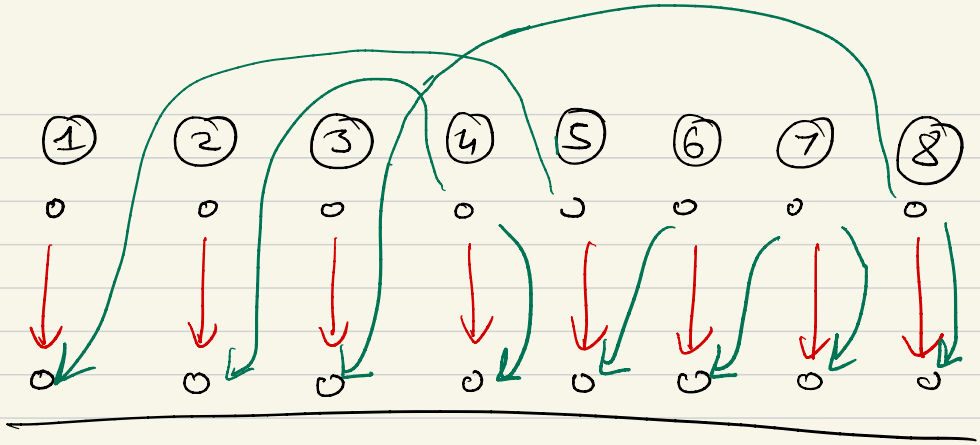



iwo
iwo
A
C
F
smo
smo
smo
B
iwo
D
iwo
E






iro
pro
iro



1 IRO with 1

4 $\xrightarrow[\text{IRO}]{}$ 1

2 $\xrightarrow[\text{IRO}]{}$ 1

2 IRO with 3

1 $\xrightarrow[\text{IRO}]{}$ 3

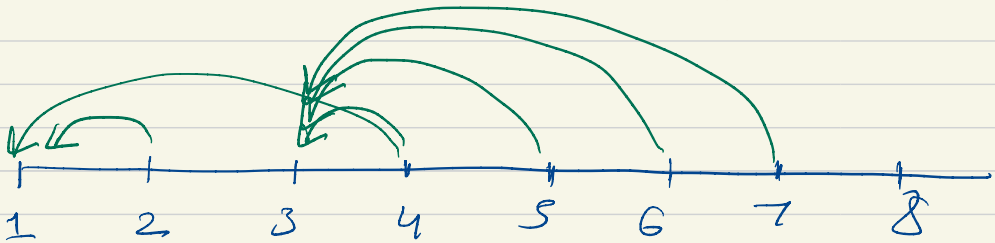
2 $\xrightarrow[\text{IRO}]{}$ 3

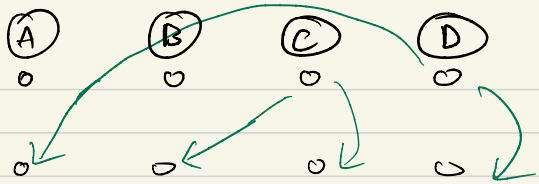
4 $\xrightarrow[\text{IRO}]{}$ 3

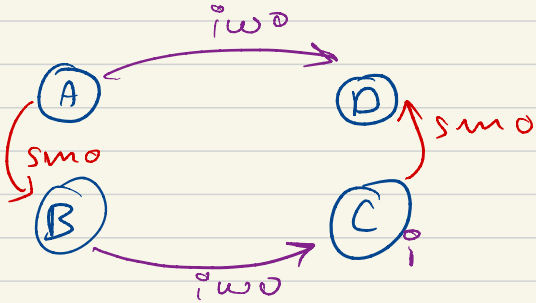
5 $\xrightarrow[\text{IRO}]{}$ 3

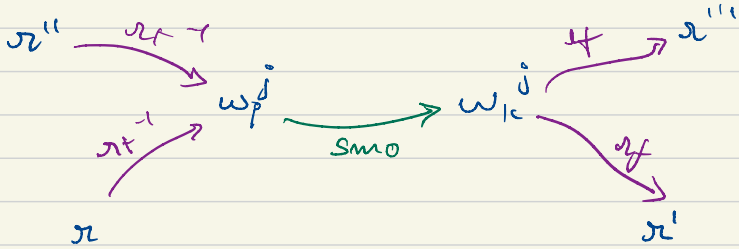
6 $\xrightarrow[\text{IRO}]{}$ 3

7 $\xrightarrow[\text{IRO}]{}$ 3









r, r', r'', r''' are in a total order.

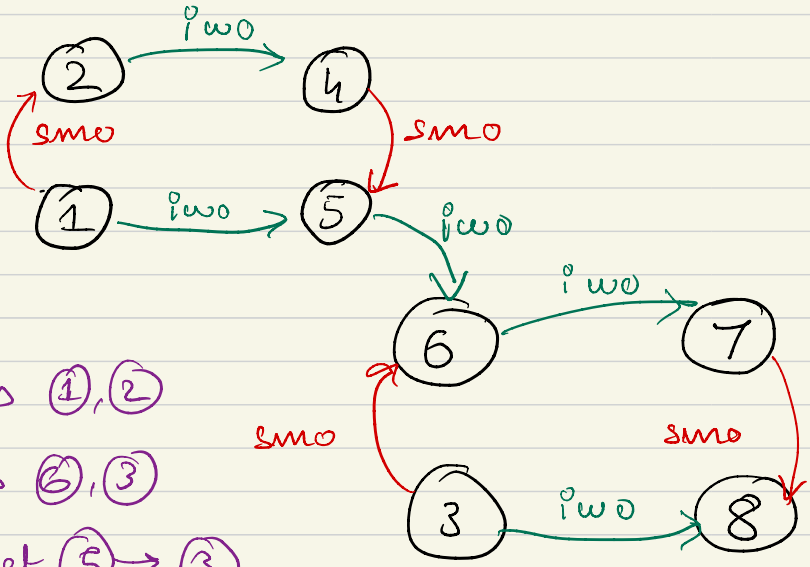
$r < r''$ $\qquad$ $r, r'' < i < k$

$r > r' \Rightarrow r'' > r'$ $\qquad$ $r > r''' \Rightarrow r'' > r'''$

Summary of Approaches.

1. Read order sorting may introduce new violating write orders, which we can fix using Lemma 1 & 2. (Prove that the new relation was not fixed in any previous config)

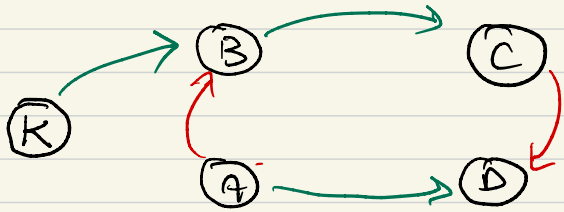
2. What remained is Read Order stability, which wasn't possible, Here's the counter example.



* Swap 1, 2
* Swap 6, 3
* We get $5 \xrightarrow{iwo} 3$
* Swap 5, 3
* We get again $2 \xrightarrow{iro} 1$

(3) How about we fix order in which we resolve read orders?

Take 1:- $\min(X) \xrightarrow[\text{iso}]{} \min(Y)$



$B \xrightarrow[\text{iso}]{} A$ exists.

Case 1:-

$K \xrightarrow[\text{smo}]{} A$

New order is correct

Case 2:-

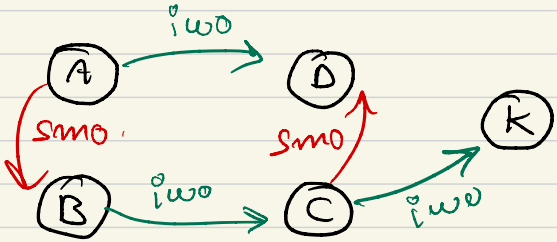
$A \xrightarrow[\text{smo}]{} K$

New order is incorrect.

But we have $K \xrightarrow[\text{iso}]{} A$ in current config.

$K < B$, hence we fix $K \xrightarrow[\text{iso}]{} A$ first.

(4) (3) does not work for writes below the read.



Case 1:-

$D \xrightarrow[smo]{} K$

No problem as new relation is correct

Case 2:-

$K \xrightarrow[smo]{} D$

Problem as new iwo rel is incorrect

Can resolve $max(x) \xrightarrow[iso]{} min(y)$ for those orders derived by writes below read.

We have then $D \xrightarrow[iso]{} K$

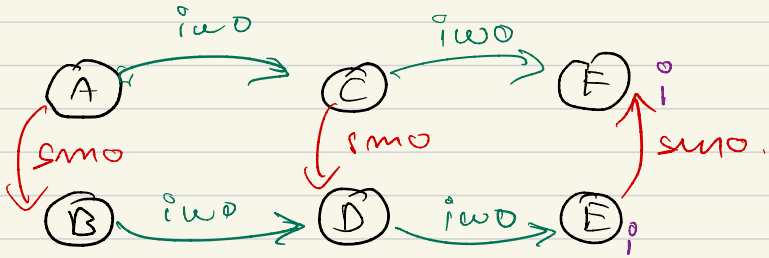
But is not $min(x) \xrightarrow[iso]{} min(y)$

So we do not resolve it first.

(5) (2), (3) only work if writes above or below exclusively, but not both.

But more than that, read orders are not stable either ways.....

Here's a counter example.

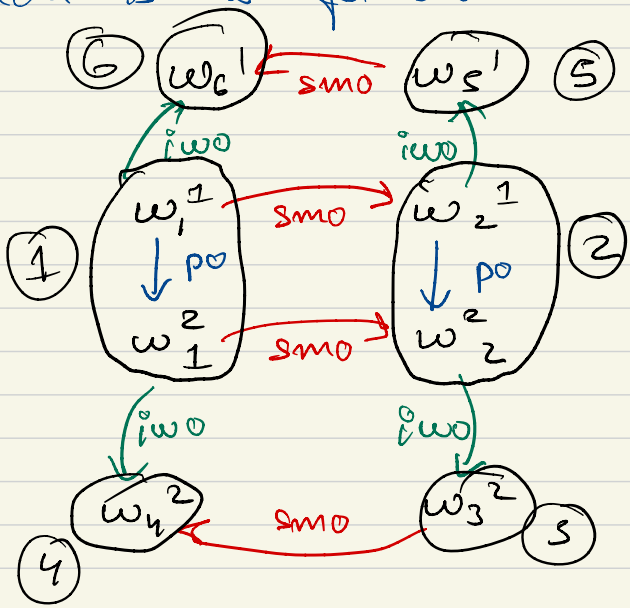


$A \to B \to C \to D \to E \to F$

$D \xrightarrow[iso]{} E$, so fixing $min(x) \xrightarrow[iso]{} min(y)$ only.

Fixing this will give us $B \xrightarrow[iso]{} A$ which was correct before.

(6) A more general example involving both writes above & below the read is as follows.



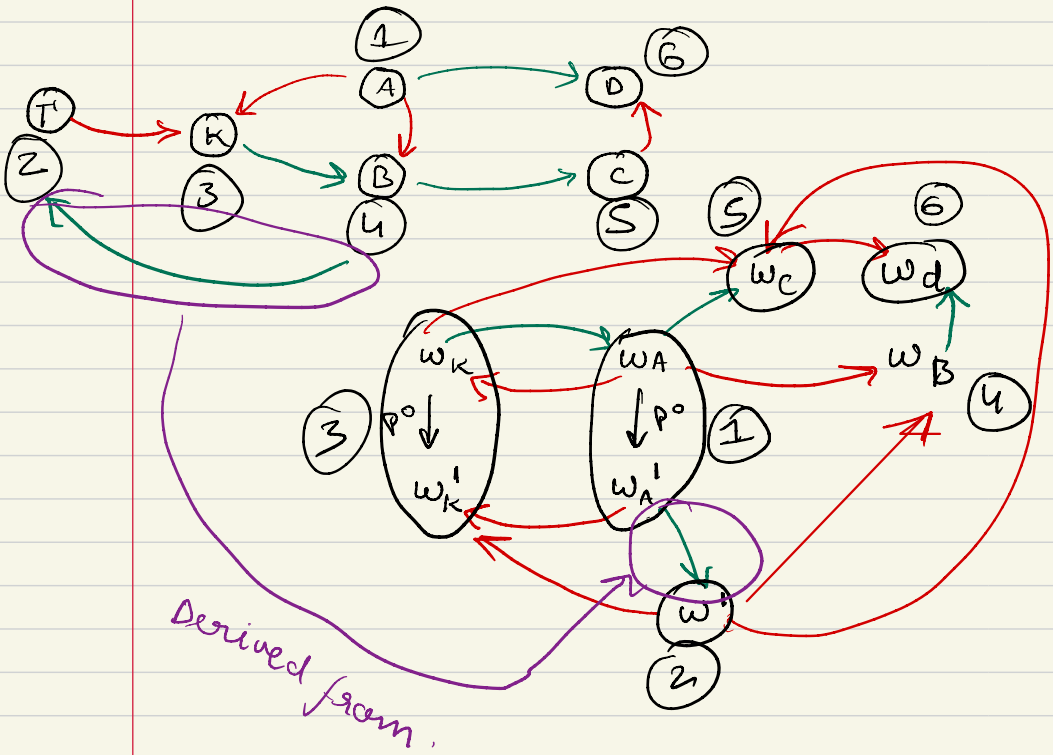
This has left me confused

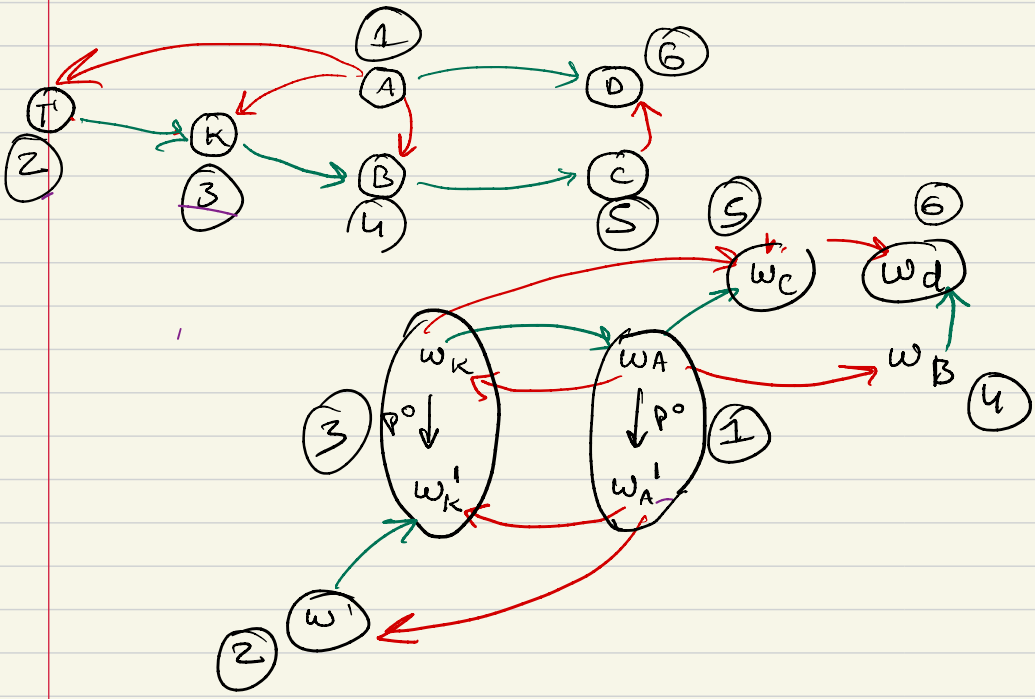
☹

* (2) $\xrightarrow[\text{iro}]{}$ (1) & (4) $\xrightarrow[\text{iro}]{}$ (3)
* If we swap (1) & (2), I fix even iro between (4) & (1)
* If I swap (4) & (3), then I still have to resolve (1) & (2).
* But doing so makes (4) $\xrightarrow[\text{iro}]{}$ (3) again.

> What's to be done?

- Lets go back to basic reasoning with partial orders. (Transitivity)




1
A
6
D
T
2
K
3
B
4
C
5
5
6
ω_c
ω_d
ω_K
ω_A
ω_B
4
3
p^0
p^0
1
ω_K'
ω_A'
ω'
2

WAY TOWARDS FORWARD PROG

→ Showing read order stability does not work, there are many counterexamples to it.

→ Forcing the order in which we resolve read orders differ for writes above & below the read

→ The difference is that new implied write orders may not be introduced if writes are only above or only below read.

→ Together order does not seem to guarantee anything; including read order stability.

→ Hence we settle for a weaker guarantee, that of Forward Progress

* If the state defines the set of implied write & read orders, forward progress would guarantee, we do not reach the same state again.

* How to show Forward Progress?

While fixing of read orders.

1. We can show any new implied write order, if wrong, wasn't fixed before.

2. Since there could be just finitely many (two), fixing them is a finite process.

3. This means on fixing an implied read order, if our implied write order is wrong, then we fix it.

Which implies, we may never come to the original config wherein on swapping, the same implied write order is wrong.

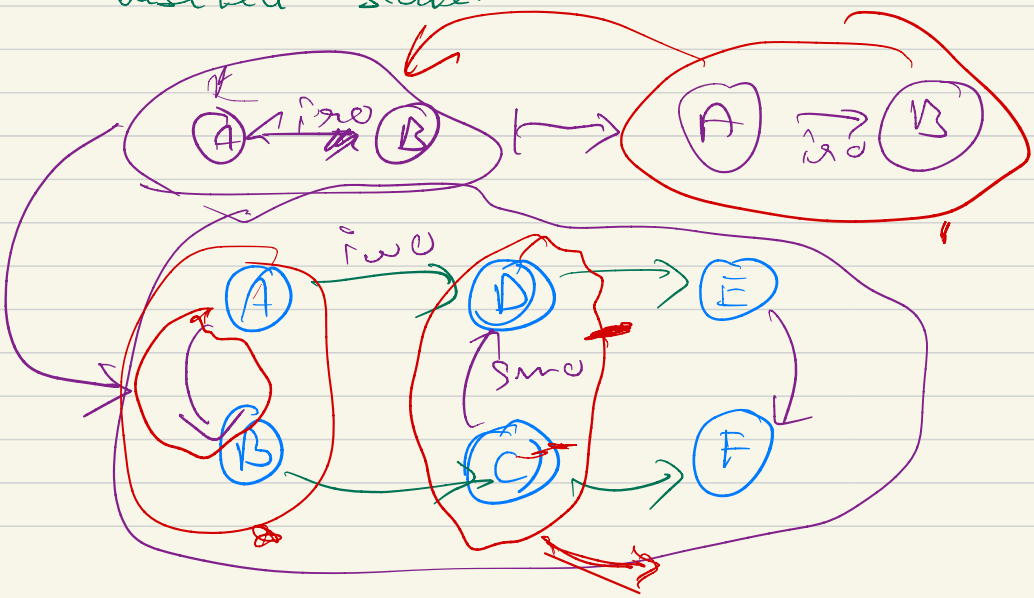
In the process of resolving read orders, if I come across any new implied write order that is wrong, then we can conclude that we will never reach the initial state from which we started resolving read orders.

(4) Suppose no (iwo) is wrong while fixing implied read orders.

(5) Then to come to initial state, note that the only swaps we can make are between threads having no iwo but whose iro is wrong.

If iro is wrong & resolved which leads to same previous iro inc again, it still is not the same state

To add to this, an iro is resolved. Now fixing this iro, will again not result in the prev iro incorrect as it would mean going back to some previously visited state.



(1) Definition of implied read order.

$iwo; smo; iwo^{-1}$
$smo; iwo^{-1}$ } Above read writes.

if a thread's read is satisfied by its own write.

$iwo^{-1}; smo; iwo$ } Below read writes.

(2) Implied read order is reversed when resp threads are swapped.

- Swap two threads & show seq comp.

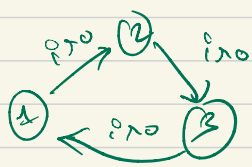
(3) If two threads have an implied read order between them, no implied write order exists between them.

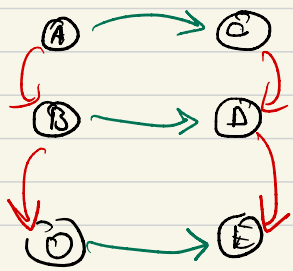
- By contradiction.

(4) Implied read order is acyclic.

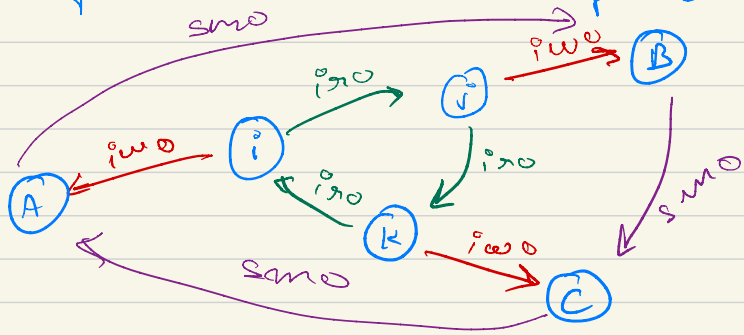
- $iwo \subseteq smo$
- smo acyclic $\wedge$ iwo acyclic.

Interesting





(w) Proof that iro's are acyclic.

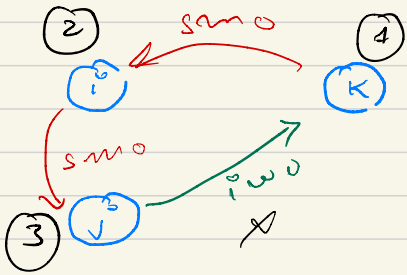


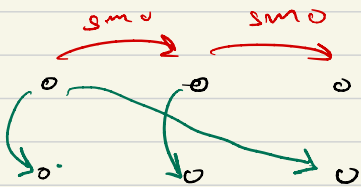
* But $\xrightarrow{smo}$ is a total order.

* Thus by contradiction a cycle must not exist.

* Similar case for writes below the read.

Final argument for Fwd progress.

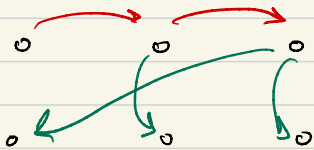




$iwo^{-1}; smo$

Not correct.

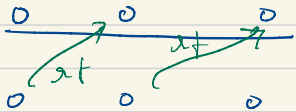
$iwo; smo$

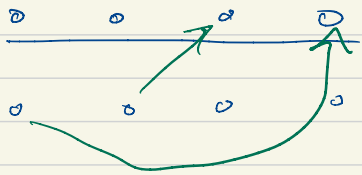


$smo; iwo^{-1}$

→ "True" definition of read order?

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





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

$po; rf^{-1} \rightarrow iwo$

$iwo \leftarrow po; rf^{-1}; smo; (rf; po^{-1}) \rightarrow iwo^{-1}$

$iwo; po^{-1}$

$iwo; \cancel{po^{-1}}; smo; \cancel{po}; iwo^{-1}$

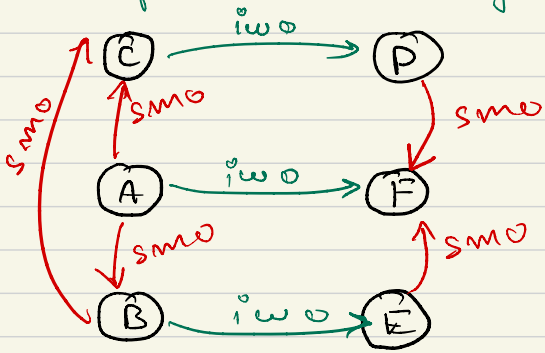
* When no $\overset{\rightarrow}{iwo}$ that is incorrect is introduced can we still fix all read orders?

(1) Show that when $\overset{\rightarrow}{iro}$ fixed between (A), (B) → getting it unfixed would involve fixing another relation. K

* (2) Show that relation K remains the same before & after fixing $\overset{\rightarrow}{Pro}$ betn (A), (B)

This is imp.

(3) Because $\overset{\rightarrow}{iro}$ is finite set, by (1), (2) we can infer Fwd Progress.

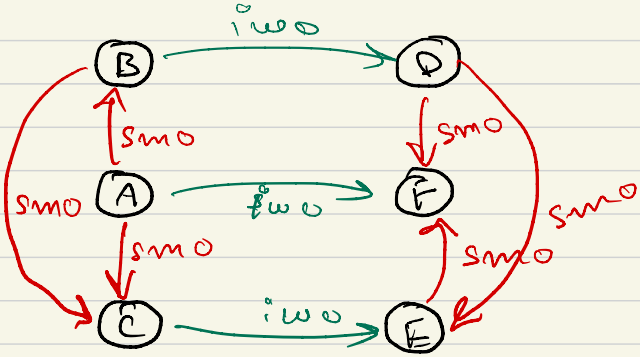


$B \overset{\rightarrow}{iro} A$ $\quad C \overset{\rightarrow}{iro} B$ $\quad C \overset{\rightarrow}{iro} A$

Before fixing $B \overset{\rightarrow}{iro} A$

$A \overset{\rightarrow}{iro} B$ $\quad C \overset{\rightarrow}{iro} B$ $\quad C \overset{\rightarrow}{iro} A$

After fixing $B \overset{\rightarrow}{iro} A$



$B \xrightarrow[\text{iro}]{} A$ $\quad$ $C \xrightarrow[\text{iro}]{} A$ $\quad$ $B \xrightarrow[\text{iro}]{} C$

Before fixing

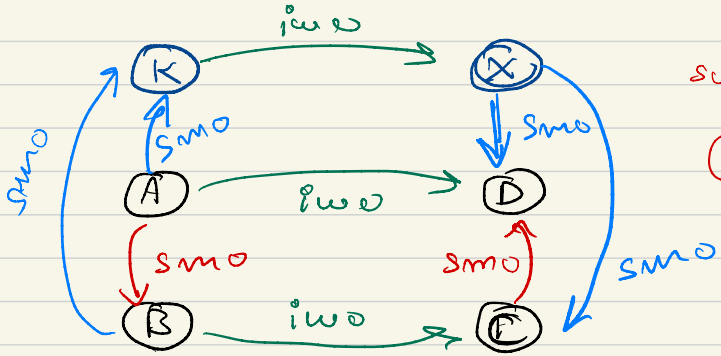
$C \xrightarrow[\text{iro}]{} A$

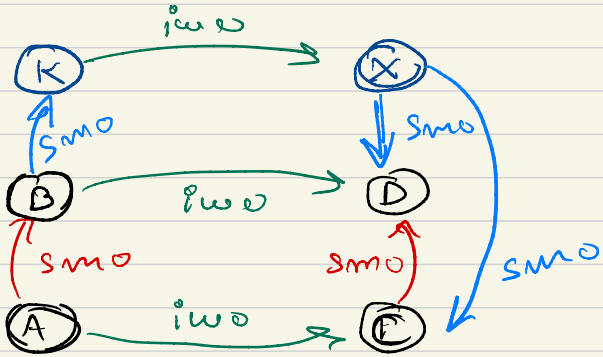
$B \xrightarrow[\text{iro}]{} A$ $\quad$ $A \xrightarrow[\text{iro}]{} C$ $\quad$ $B \xrightarrow[\text{iro}]{} C$

After fixing

$C \xrightarrow[\text{iro}]{} A$

We can resolve $B \xrightarrow[iso]{} A$ using blue relations & swapping K & A





$K \xrightarrow[iso]{} B$ remains same before & after swapping.

Let the state represent the set of relations iwo & iro.

Let Sound (S) represent the state where $\overrightarrow{iwo}$ respects over irreflexivity constraints that are sound!

A] Lemma 1 :-

If $\overset{iwo}{correct}(S) \wedge S \rightarrow S'$ after resolving an $\overrightarrow{iro}$ as per irreflexivity condition on it, then

If $\neg \overset{iwo}{correct}(S')$ true

The new $\overrightarrow{iwo}$ relations which are incorrect could not have been resolved in any previous state S'' s.t $S'' \rightarrow S$

Proof :- Req :- $\overrightarrow{iro}$ inc no $\overrightarrow{iwo}$, $\overrightarrow{iro}$ resolved by swapping threads that have their reads $\overrightarrow{iro}$ with the

→ Use contradiction

→ Take case wise analyses

→ Mind the order in which we resolve.

Lemma 2:-

$$\underset{\text{correct}}{\text{iro}}(S) \wedge \boxed{B \underset{\text{iro}}{\rightarrow} A} \in S'$$

$A \rightarrow B \in S'$, S''

then

$$S' \text{ s.t } S \xrightarrow{B,A} S' \wedge \exists \boxed{Y \underset{\text{iro}}{\rightarrow} X} \in S'$$

then

If

$$S' \xrightarrow{(Y,X)} S'' \text{ s.t } \boxed{B \underset{\text{iro}}{\rightarrow} A} \in S''$$

$$\Rightarrow$$

$$\boxed{Y \underset{\text{iro}}{\rightarrow} X} \in S$$

$X \rightarrow Y$, $A \rightarrow B$, S'''

Any iro s.t on resolving introduces previously resolved incorrect iro, must have been stable before and after forming previous iro

Proof:-

→ Go case wise analyses

→ Show that after transition the relations that exist are preserved or not.

Theorem of Optimality :-

Resolving $\overrightarrow{iro}$ is a terminating process such that the end state S is

sound (S) $\wedge$ optimal (S)

Proof :-

→ ~~By Lemma 1, if any incorrect $\overrightarrow{iro}$ is new, then~~ by Soundness, they can be resolved. ($\overrightarrow{iro}$)

By Lemma 1, $\overrightarrow{iro}$ inc is new only. Hence, → If resolved, we cannot come back to the state from which transition results in the incorrect relation.

⇒ If no incorrect $\overrightarrow{iro}$ is introduced then by Lemma 2, to come to original state where resolved $\overrightarrow{iro}$ is incorrect, another $\overrightarrow{iro}$ is resolved.

By Lemma 2, contradiction shows that we can never reach the original state.

⇒ Since $\overrightarrow{iro}$ is finite, we can resolve all of them, hence proving that it is a terminating process.

Finite argument needed?

Lemma 1 proof (detailed)

① We have :-

req irreflex Prop $\longleftarrow$ Sound (S) $\rightarrow \nexists\, w \xrightarrow[iwo]{} w' \wedge w' \xrightarrow[smo]{} w$

req Prop $\overset{a}{iro} \equiv_{iwo,smo}$ comp. $\longleftarrow$ $r_j \xrightarrow[iro]{} r_i$

read sat. by writes above read

$\hookrightarrow \exists\, w_k$ s.t $w_i \xrightarrow[iwo]{} w_k \wedge w_j \xrightarrow[smo]{} w_k$ (a)

$\hookrightarrow \exists\, w_k, w_l$ s.t $w_i \xrightarrow[iwo]{} w_l \wedge w_j \xrightarrow[iwo]{} w_k$ (b)

$\wedge\ w_k \xrightarrow[smo]{} w_l$.

read sat. by writes below read. $\longleftarrow$

$\hookrightarrow \exists\, w_k, w_l$ s.t $w_l \xrightarrow[iwo]{} w_i \wedge w_k \xrightarrow[iwo]{} w_j$ (c)

$\wedge\ w_k \xrightarrow[smo]{} w_l$.

② case (a) $\rightarrow$ $w_i \xrightarrow[iwo]{} w_k \wedge w_j \xrightarrow[smo]{} w_k$

$\hookrightarrow$ iwo with w_i & some w'

i) $w' \xrightarrow[iwo]{} w_i$ (other case not possible)

$$\begin{tikzcd}
w' \arrow[r, "iwo"] \arrow[dr, bend right] & w_i \arrow[d, "smo"] \arrow[dr, "iwo"] & \\
& w_j \arrow[r, "smo"] & w_k
\end{tikzcd}$$

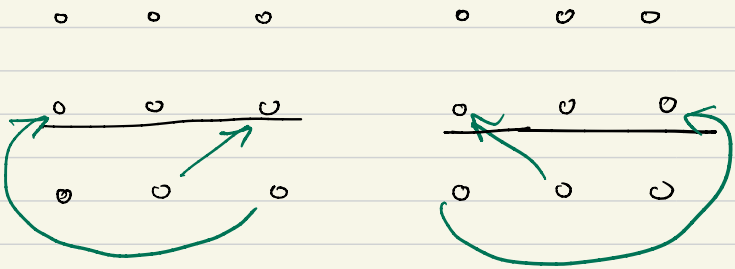
$w' \xrightarrow[smo]{} w_j$

$\hookrightarrow$ Aft swapping T_i, T_j

$w' \xrightarrow[iwo]{} w_j$ which is not incorrect.

- Why wasn't this case considered for soundness?

$$\begin{tikzcd}
 & w'' \arrow[dr] & \\
 & w' \arrow[u, leftrightarrow] \arrow[r, "smo"] & w_k \arrow[d, leftrightarrow] \\
w_i \arrow[ur, "smo"] \arrow[rr, "smo"] & & w_k \\
w_i' & & w_k' \arrow[ll, "iwo"']
\end{tikzcd}$$



- Why was this case NOT CONSIDERED?

- Because you can reduce the above case to the 2 cases we considered.

$$\begin{tikzcd}
 & w'' \arrow[dr] & \\
w_i'' \arrow[ur, "smo"] \arrow[rr, "smo"] \arrow[d, leftrightarrow] & w' \arrow[u, leftrightarrow] \arrow[r, "smo"] & w_k \arrow[d, leftrightarrow] \\
w_i \arrow[ur, "smo"] \arrow[rr, "smo"] & & w_k \\
w_i' & & w_k' \arrow[ll, "iwo"']
\end{tikzcd}$$

$$\begin{tikzcd}
w_i'' \arrow[r, "smo"] & w'' \arrow[d, "iwo"] \\
 & w_k'' \arrow[l, "smo"]
\end{tikzcd}$$

→ $\overrightarrow{iwo}$ with w_j and some w'

$$\begin{tikzcd}
 & w_i \arrow[d, "smo"] \arrow[dr, "iwo"] & \\
w' \arrow[r, "iwo"'] & w_j \arrow[r, "smo"'] & w_k
\end{tikzcd}$$

i) $w' \xrightarrow[smo]{} w_i$

After swopping T_i & T_j $w' \xrightarrow[iwo]{} w_i$ is correct.

ii) $w_i \xrightarrow[smo]{} w'$

After swopping T_i, T_j $w' \xrightarrow[iwo]{} w_i$ is incorrect.

$$\begin{tikzcd}
 & x \arrow[dr, "iwo"] \arrow[dl, bend right, "smo"'] & \\
w' \arrow[dr, "iwo"'] & w_i \arrow[l, "smo"'] \arrow[d, "smo"] \arrow[dr, "iwo"] & \\
 & w_j \arrow[r, "smo"'] & w_k
\end{tikzcd}$$

Next Pg.

what all w_k?

$x \xrightarrow[smo]{} w'$

or

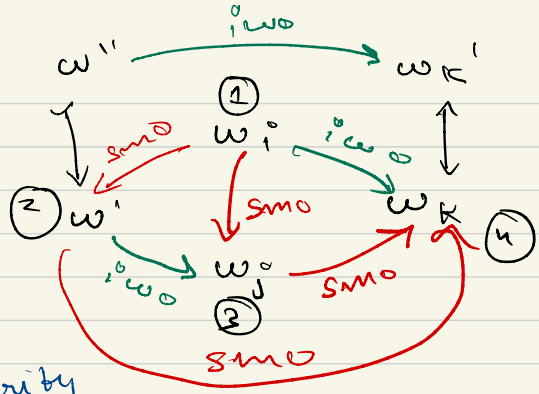
$w' \xrightarrow[smo]{} x$

↓

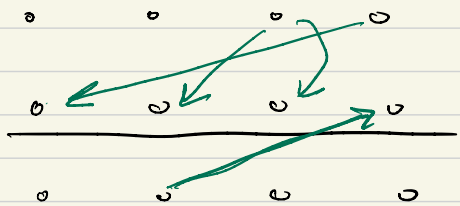
No $\overrightarrow{iwo}$ exists between them, so swopping cannot have happened.

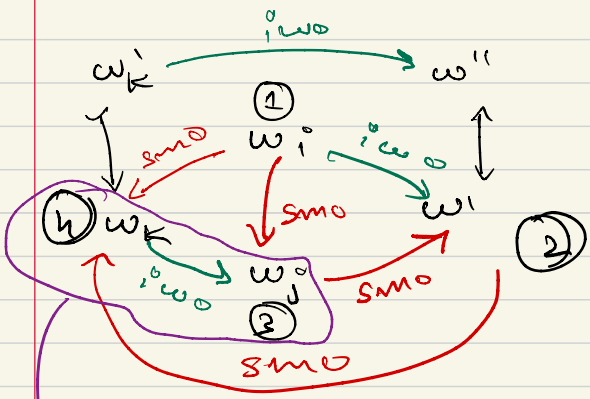
T_i, T_j may not have been swopped before due to "iso one no iwo" property.

next 1d



* what I am looking is clarity to show that those two cases are covering every other case.





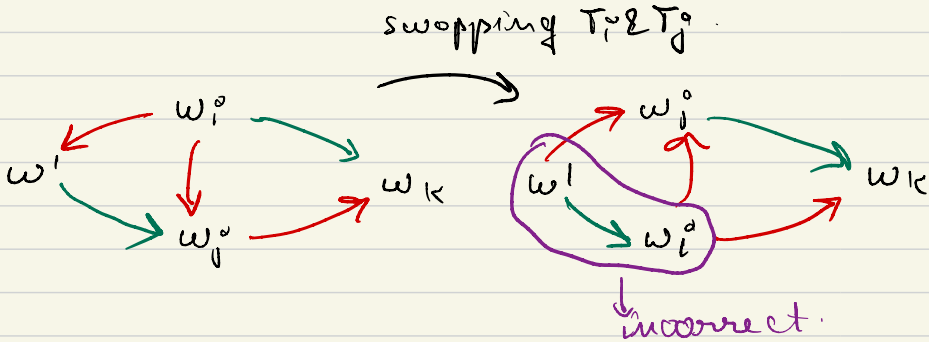
* So we either take some se or see if we can swap ω_k & ω'

* Both cases can be proven as not possible.

This should have been sorted first as per our rules of resolving iwo alone first & then below.

* Getting too complicated. (Lemma proof)

* What do we want to prove :-
  - → For any incorrect relation, it wasn't resolved before
  - → Configurations that we have cannot have come thru resolving certain incorrect relations.

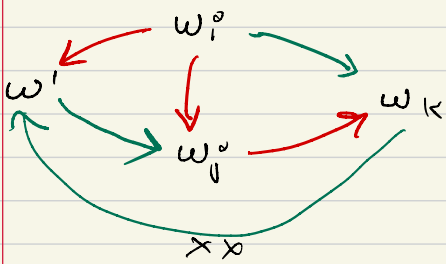


* What possible ways can we get $\xrightarrow{iwo}$ between w_i & w' without swapping T_i & T_j.

  - Swap w' & w_k → T', T_k
    - (A1) ∘ $w_k \xrightarrow[iwo]{} w'$
    - (A2) ∘ $w'' \xrightarrow[iwo]{} w_k'$
    - (A3) ∘ $x' \xrightarrow[iwo]{} x_k$.

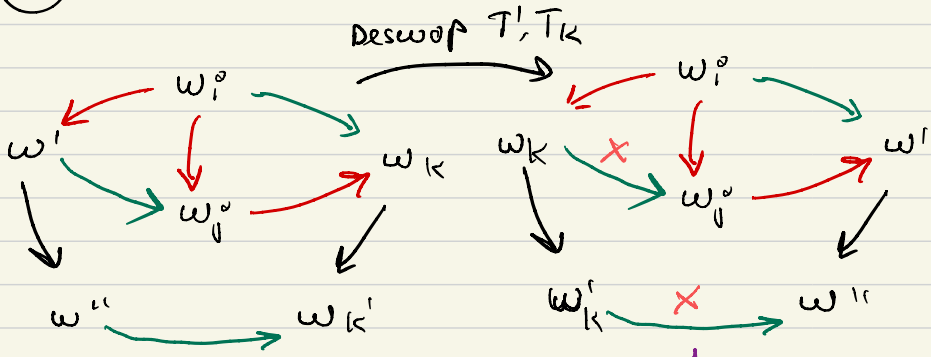
  - Swap w' & some x s.t $x \xrightarrow{iwo} w_i$ → T_x, T'
    - (B1) ∘ $w' \xrightarrow[smo]{} x$ Want swap to fix iwo & invalid iwo would exist
    - (B2) ∘ $x \xrightarrow[smo]{} w'$ Would have swapped to fix iwo but old config had incorrect $\xrightarrow{iwo}$
    - (B3) ∘ $x \xrightarrow[iwo]{} w'$ (Invalid case)

(A1) $w_k \xrightarrow[\text{iwo}]{} w'$



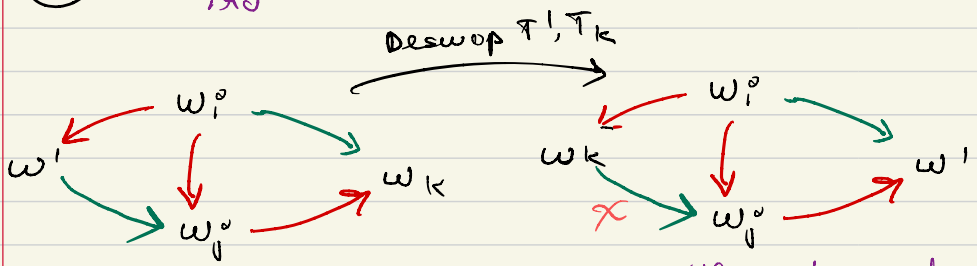
We would not resolve iwo btw w_i^0, w_j^0 bcz $w_k \xrightarrow[\text{iwo}]{} w'$ is already incorrect in current config.

(A2) $w'' \xrightarrow[\text{iwo}]{} w_k'$ (below read)



would swap T_k, T_j, not T', T_k.

(A3) $\sigma' \xrightarrow[\text{iso}]{} \sigma_k$.



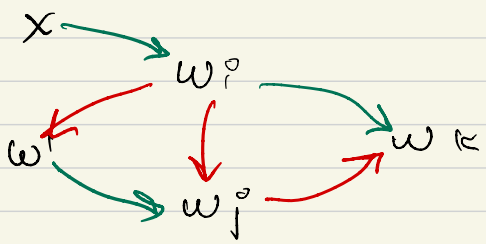
will not resolve iso as $w_k \xrightarrow[\text{iwo}]{} w_i^0$ is incorrect.

* There are 2 possible reasons to swap, two threads → iwo & iro resolving.

→ iwo → smo relations are also swapped.

→ iro → smo relations are involved

* If we can ensure that in the end the last swapping involved will still result in a contradiction or invalid config, we are done.

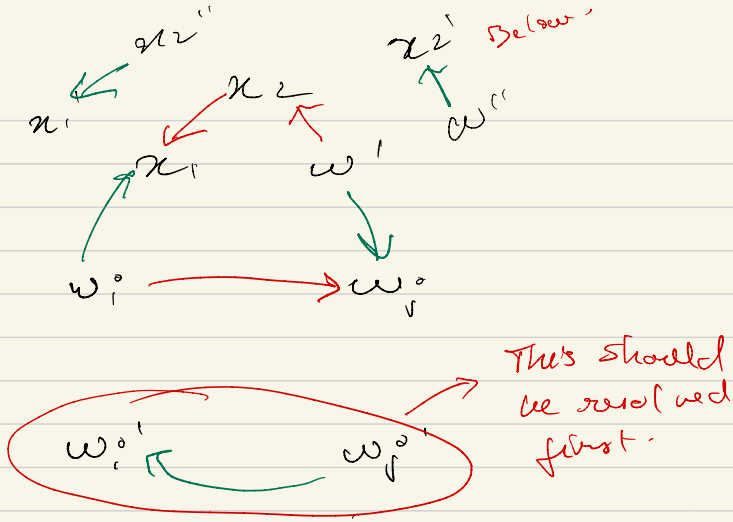


① Swap w' & w_k on grounds of prev inc
iwo

② Swap w' & w_k on grounds of prev inc
iro

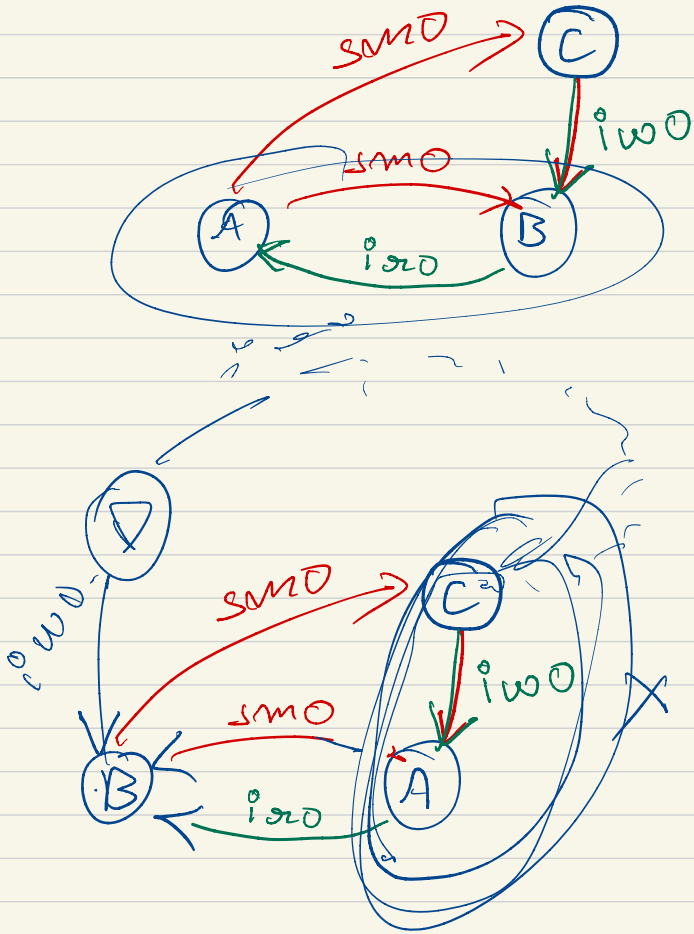
③ Swap x & w on grounds of prev inc
iwo

④ Swap x & w on grounds of prev inc
iro.



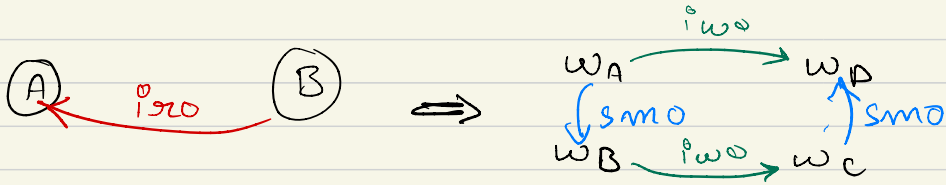
* What's the stopping point?

* How do we conclude that there are minimal swappings to show that relations are new?

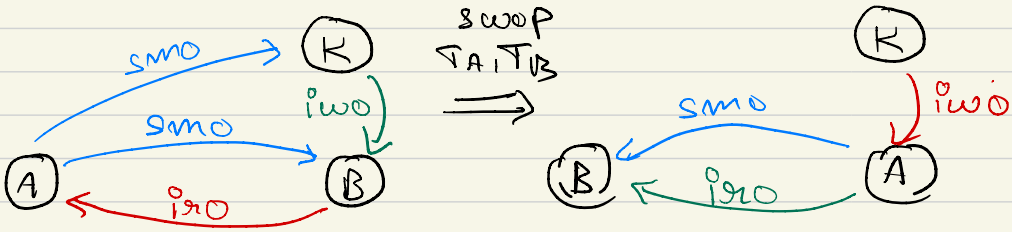

smo
C
iwo
smo
A
B
iro
D
iwo
smo
C
iwo
smo
B
A
iro

New Simpler Method for Mind Authors

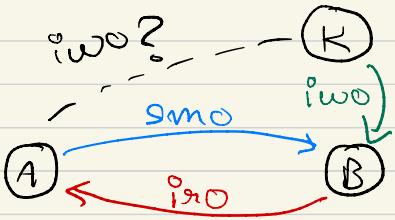
ABOVE READ



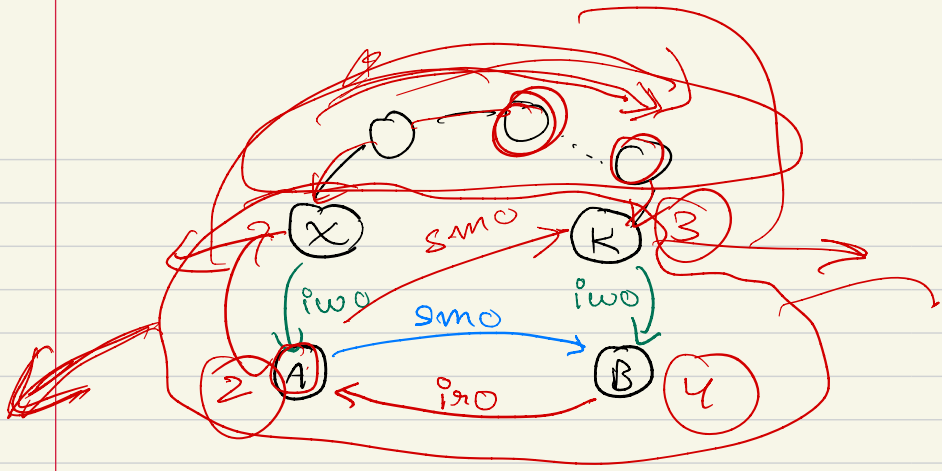
① Both reads are satisfied already.



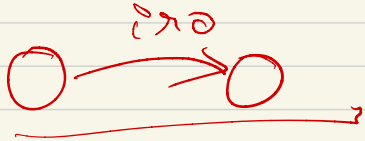
I) Base Cases:-

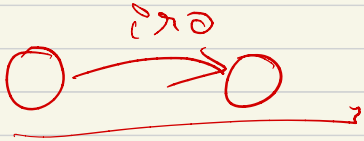


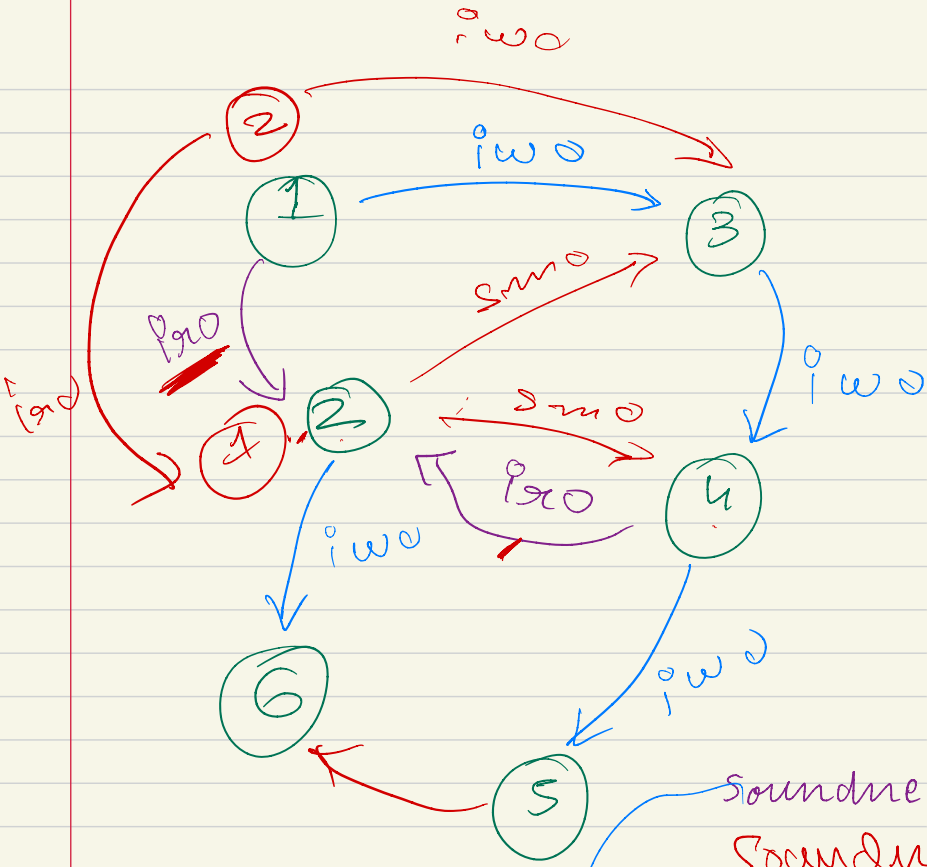
* By prop of writes above read, read-from func, no iwo can exist.



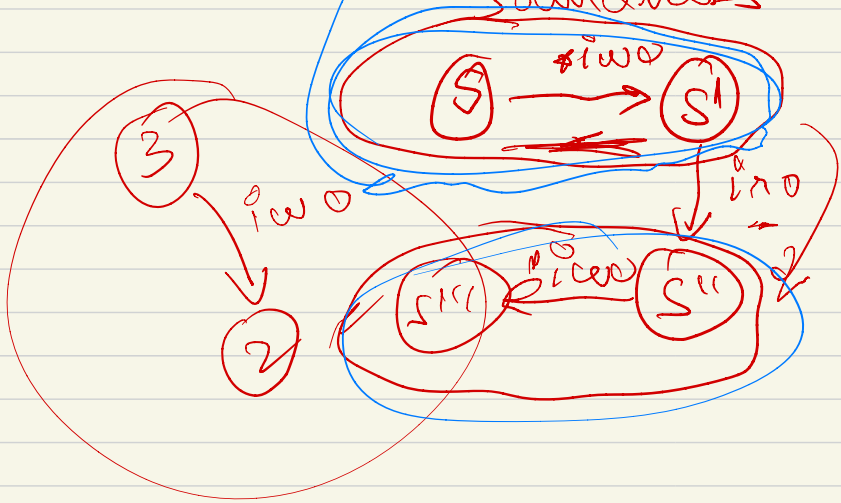
Can there be a sequence of swoppings such that



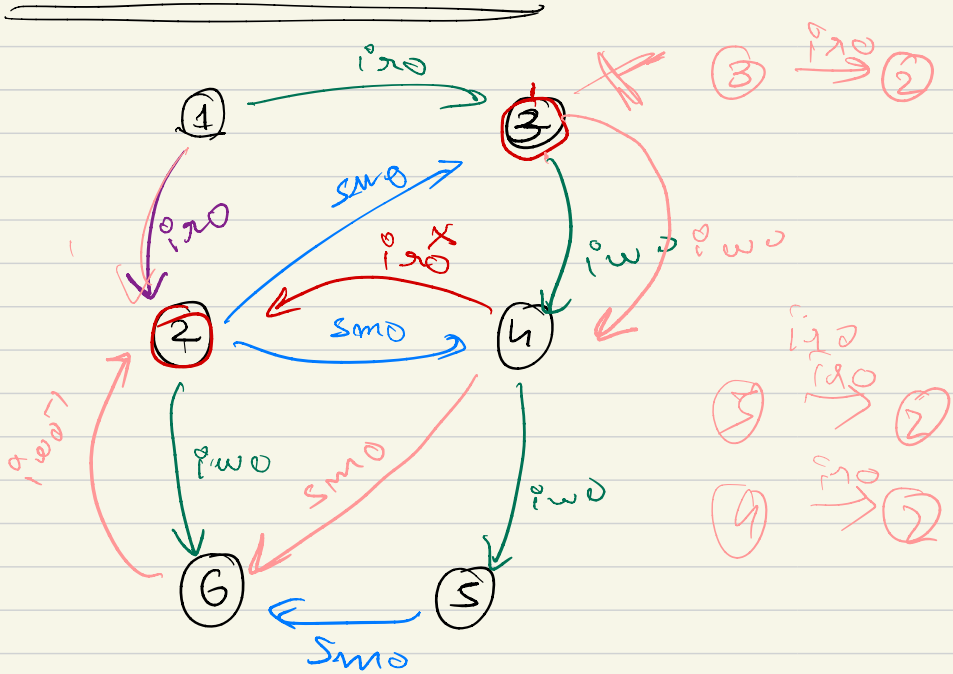



iwo
iwo
smo
fro
iro
smo
iwo
fro
iwo
iwo
iwo

Soundness


Soundness
iwo
iro
iwo
iwo

ABV READ WRITES



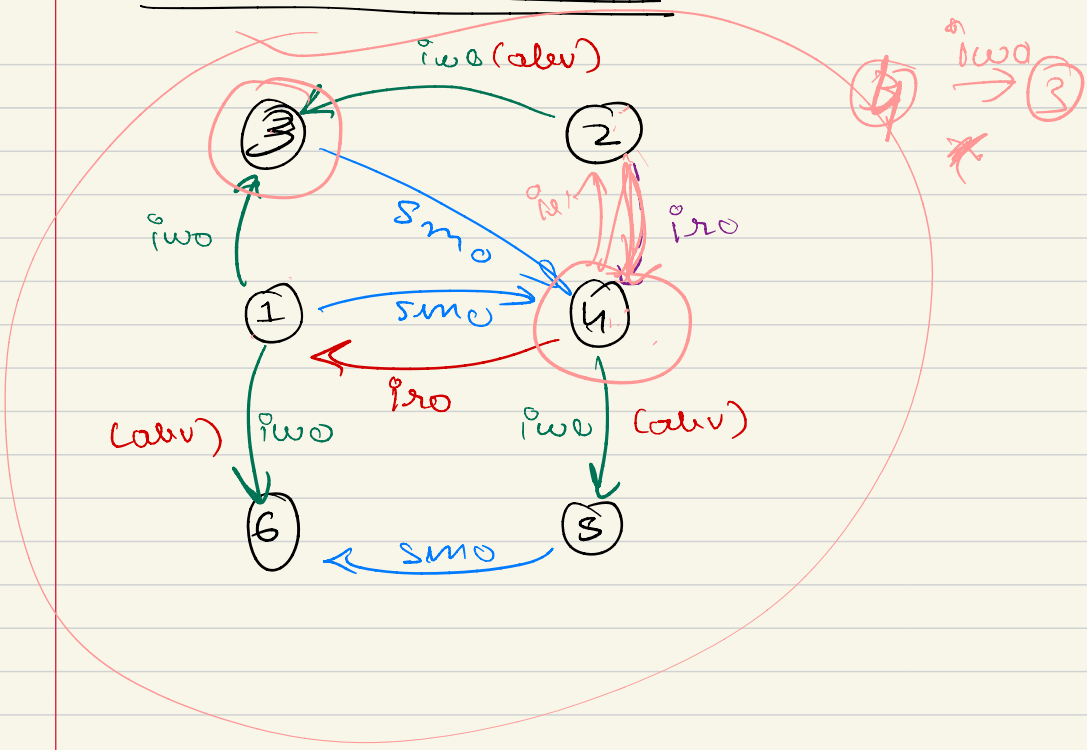
$$\min(y) \xrightarrow[\text{iro}]{} \min(x)$$

$y > x$

$s \longrightarrow 1$

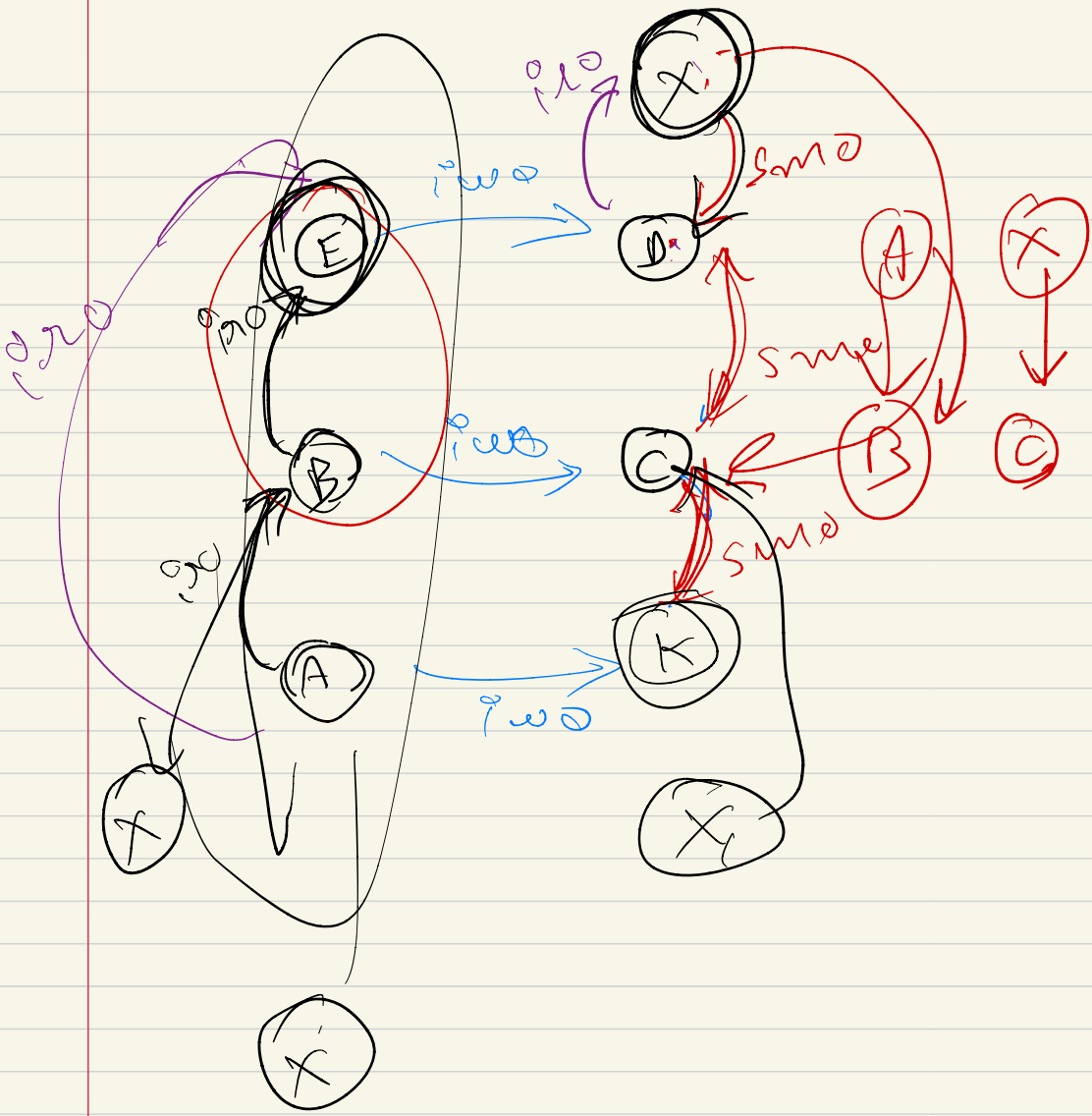
$x \longrightarrow 2$

BEL READ WRITES.



$$\text{more}(x) \xrightarrow[\text{iro}]{} \text{more}(x)$$

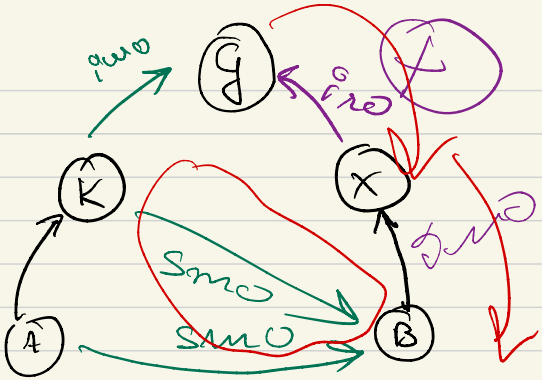
$$y > x$$


E
B
A
X
X'
D
C
K
X
A
B
C
X

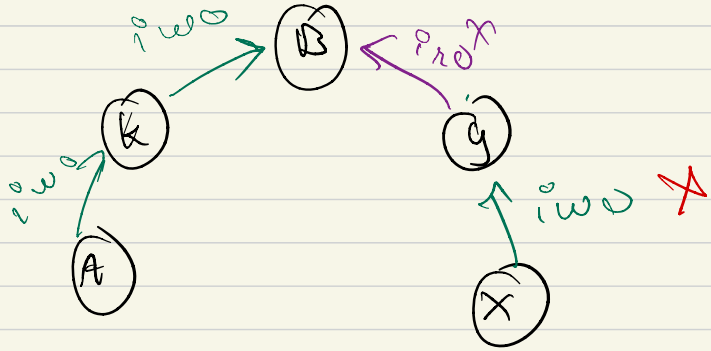
$$\begin{tikzcd} A \arrow[r, "i_A \circ"] & B \arrow[d, "i_B \circ"] \\ & i \end{tikzcd}$$

$$\begin{tikzcd} B \arrow[r, "i_B \circ"] & A \arrow[d, "i_A \circ"] \\ & i \end{tikzcd}$$

$$\begin{tikzcd} & \omega_k^j \arrow[dr, bend left, "s_{k\ell} \circ"] & \\ \omega_i^j \arrow[ur, "i_{ik} \circ"] \arrow[rr, "s_{i\ell} \circ"'] & & \omega_\ell^j \end{tikzcd}$$


iwo
G
iro
X
K
X
SMO
SMO
A
B

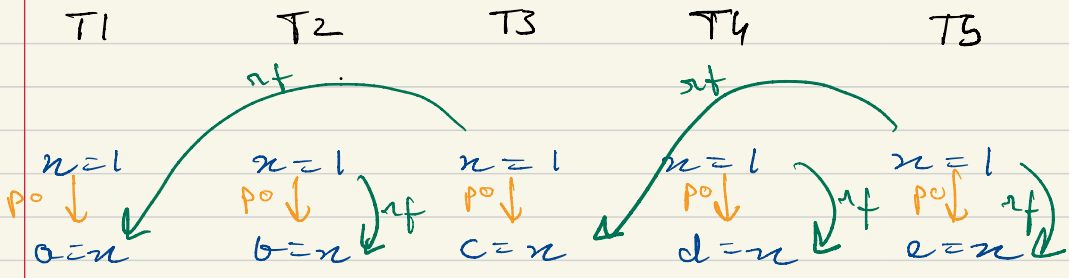
G
K
B
A
X


iwo
B
iro
K
G
iwo
iwo
A
X

T1	T2	T3
a1: $x=1$	a2: $x=1$	a3: $x=1$
b1: $a=x$	b2: $b=x$	b3: $c=x$

$$\begin{tikzcd}
x=1 \arrow[d, "po"] & x=1 \arrow[d, "po"] \arrow[dl, bend left, "rf"] & x=1 \arrow[d, "po"] \arrow[dl, bend left, "rf"] \arrow[dll, bend right, "rf"'] \\
a=x & b=x & c=x
\end{tikzcd}$$

$$\begin{tikzcd}
x=1 \arrow[d, "po"] \arrow[dr, bend left, "rf"] & x=1 \arrow[d, "po"] & x=1 \arrow[d, "po"] \arrow[dl, bend left, "rf"] \arrow[dll, bend right, "rf"'] \\
a=x & b=x & c=x
\end{tikzcd}$$


T1
T2
T3
T4
T5
rf
rf
x=1
x=1
x=1
x=1
x=1
po
po
po
po
po
rf
rf
rf
a=x
b=x
c=x
d=x
e=x

Lemma 2:-

S : $B \xrightarrow[\text{iso}]{} A \in S$

$S \xrightarrow[A, B]{\text{Swap}} S'$ s.t $Y \xrightarrow[\text{iso}]{} X \in S'$

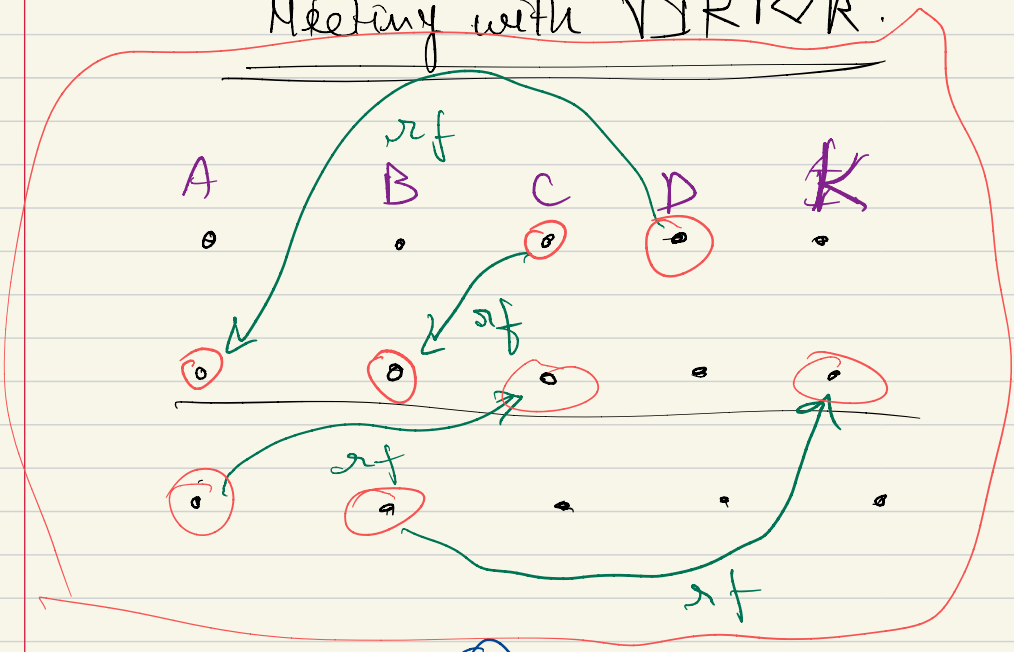
$S' \xrightarrow[X, Y]{\text{Swap}} S''$ s.t $B \xrightarrow[\text{iso}]{} A \in S''$

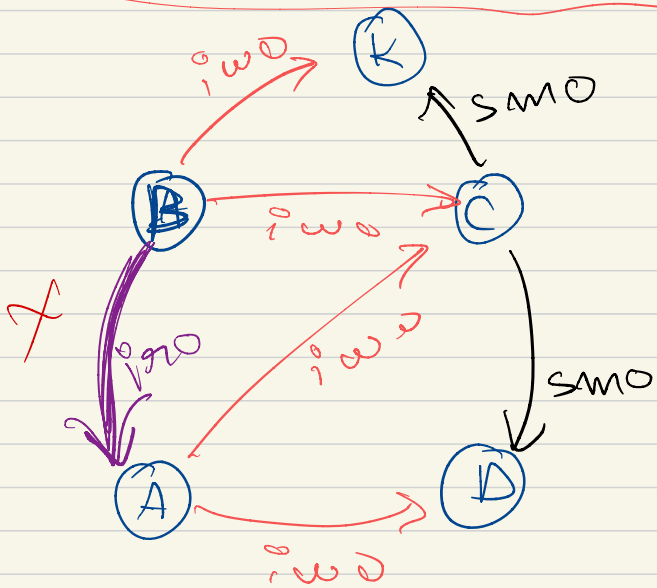
then

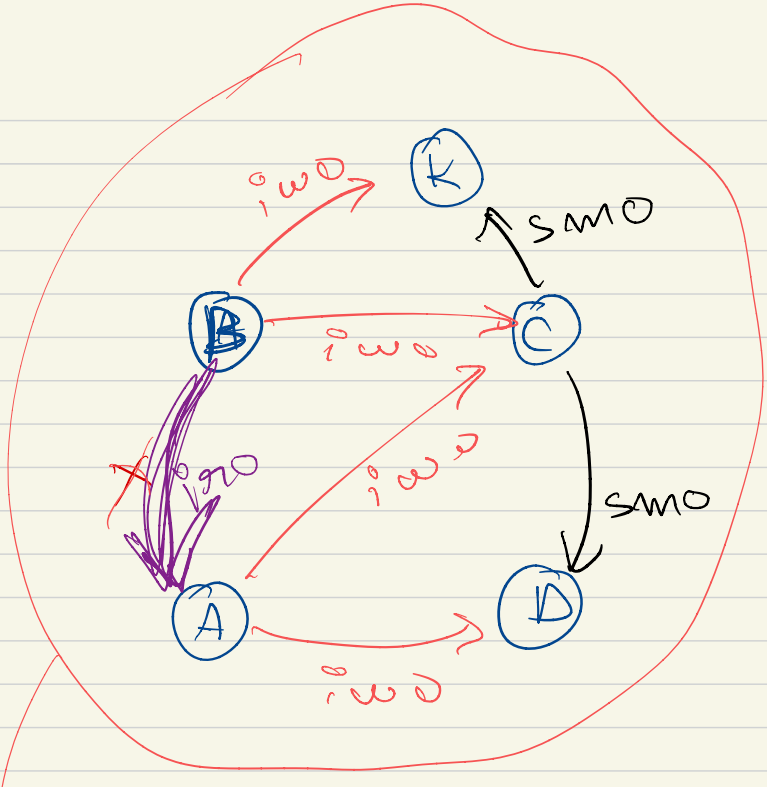
$Y \xrightarrow[\text{iso}]{} X \in S$

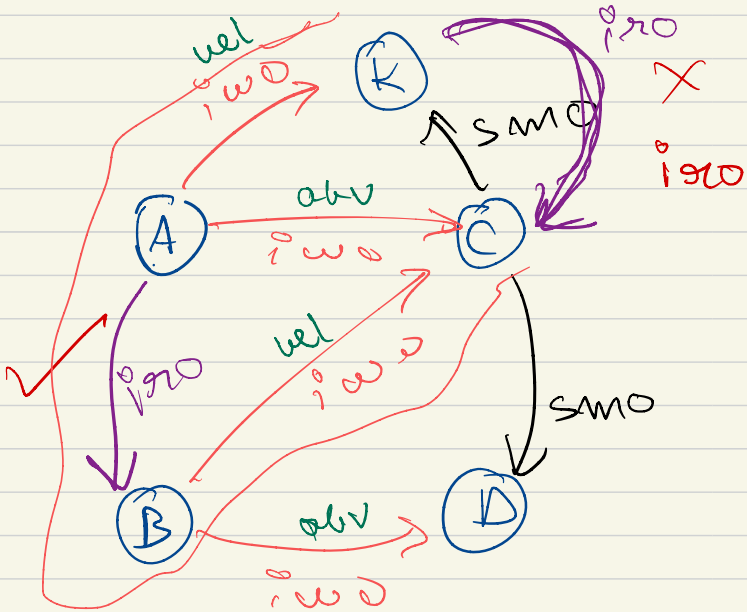
Lemma 2 does not hold.

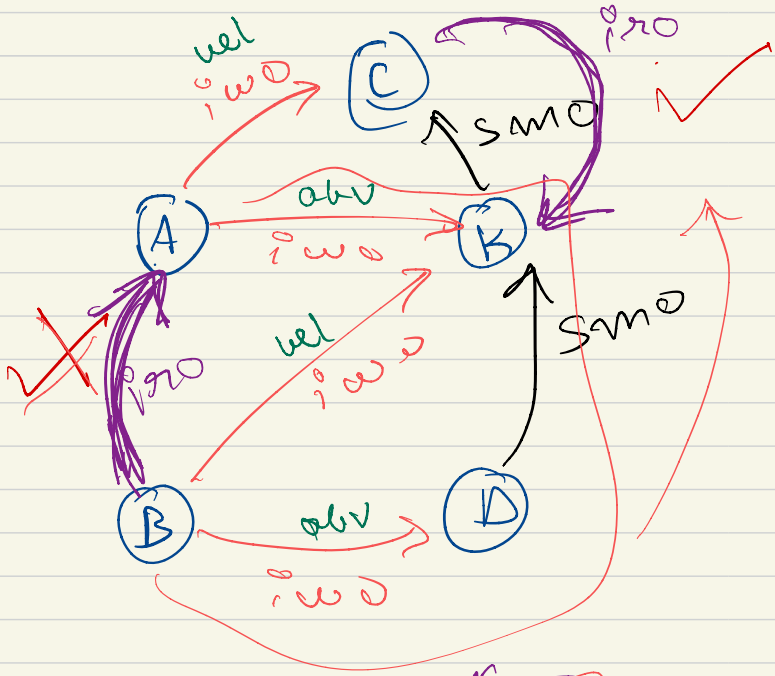
Meeting with VIKTOR.

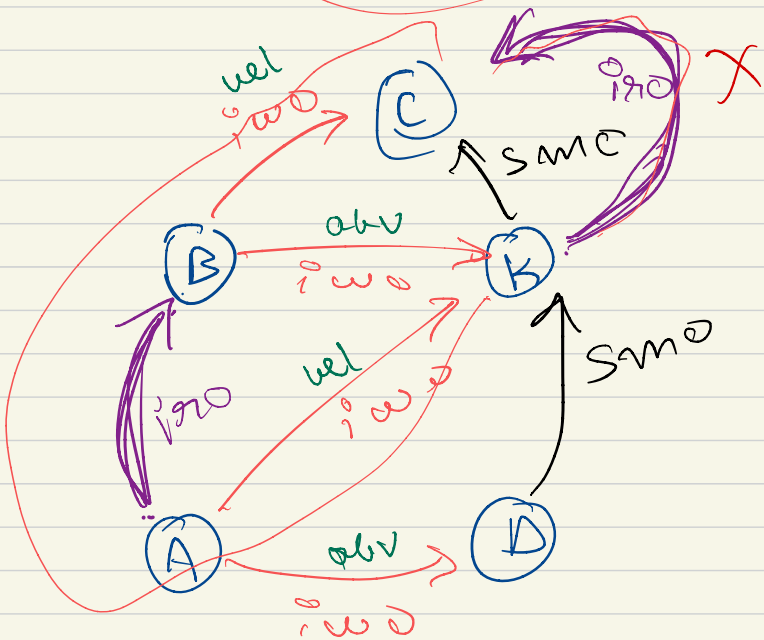


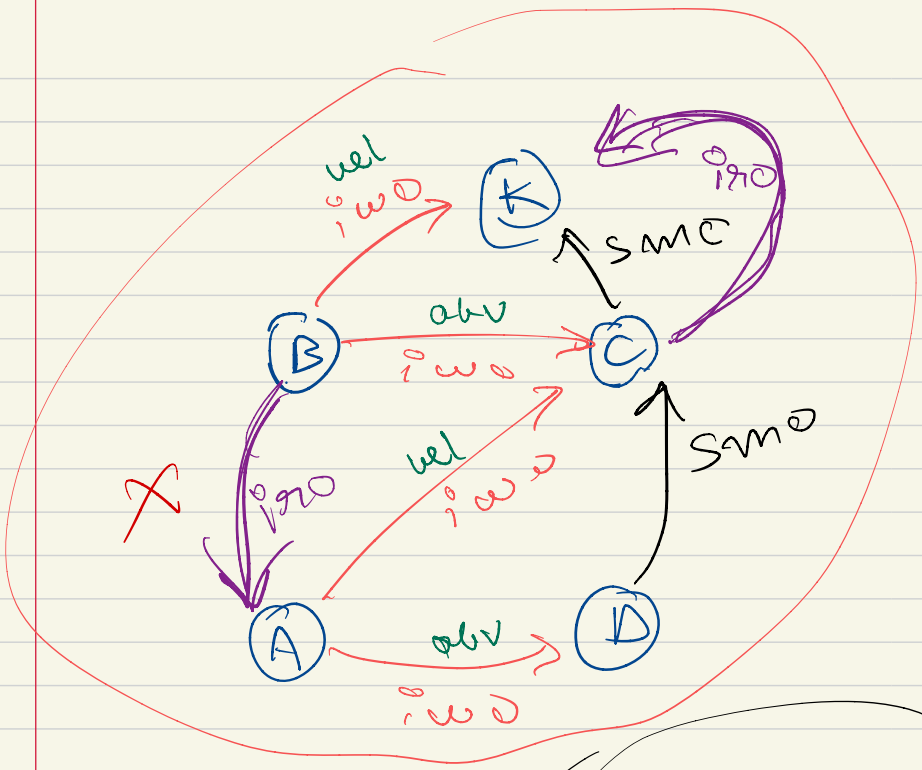



K
B
C
A
D
smo
smo
iso
iso
iso
iso
iso


K
A
C
B
D
vel
iso
obv
iso
iso
smo
vel
iso
smo
obv
iso


A
B
C
D
K
vel
iwo
abv
iwo
vel
iwo
iro
iro
smo
smo
abv
iwo


A
B
C
D
K
vel
iwo
abv
iwo
vel
iwo
iro
iro
smo
smo
abv
iwo



cycle !!

Same config as before!

Done &

Downvoted.

Thanks Victor!