ELLE

Practical Serializability Verification

Kyle Kingsbury
&
Peter Alvaro

HIGH PERFORMANCE TRANSACTION SYSTEMS

HIGH PERFORMANCE

TRANSACITON

SYSTEMS

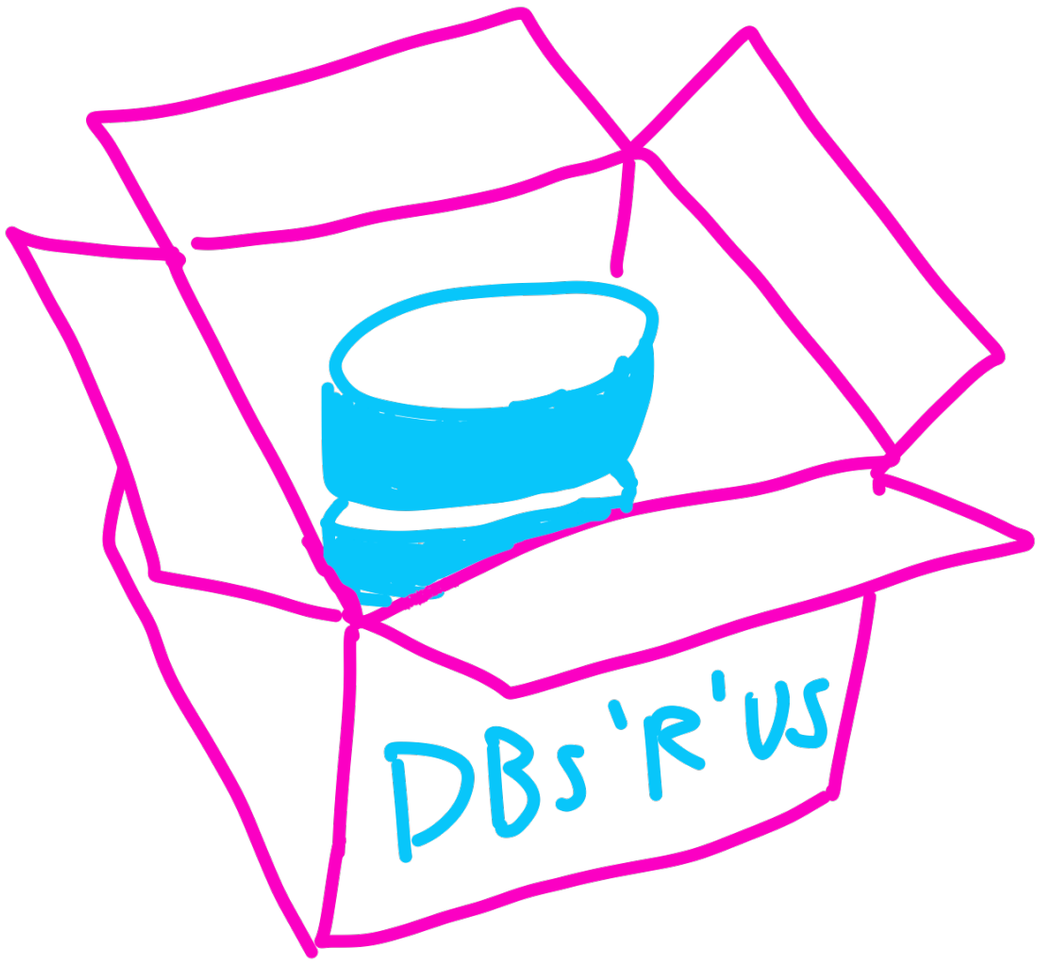
HIGH PERFORMANCE

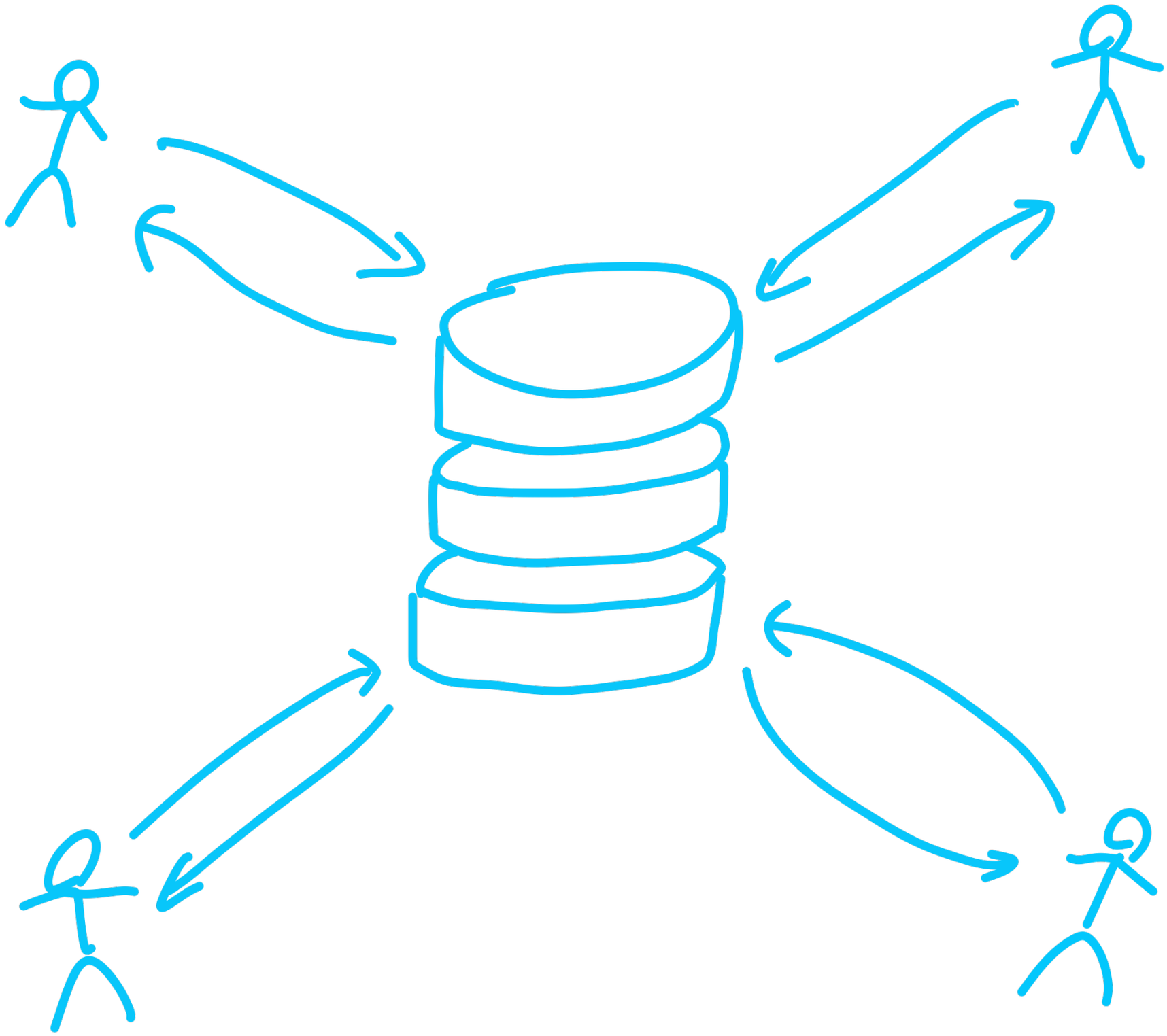
TRANSACTION

?

SYSTEMS

So, is it Serializable?





time

T_1

T_3

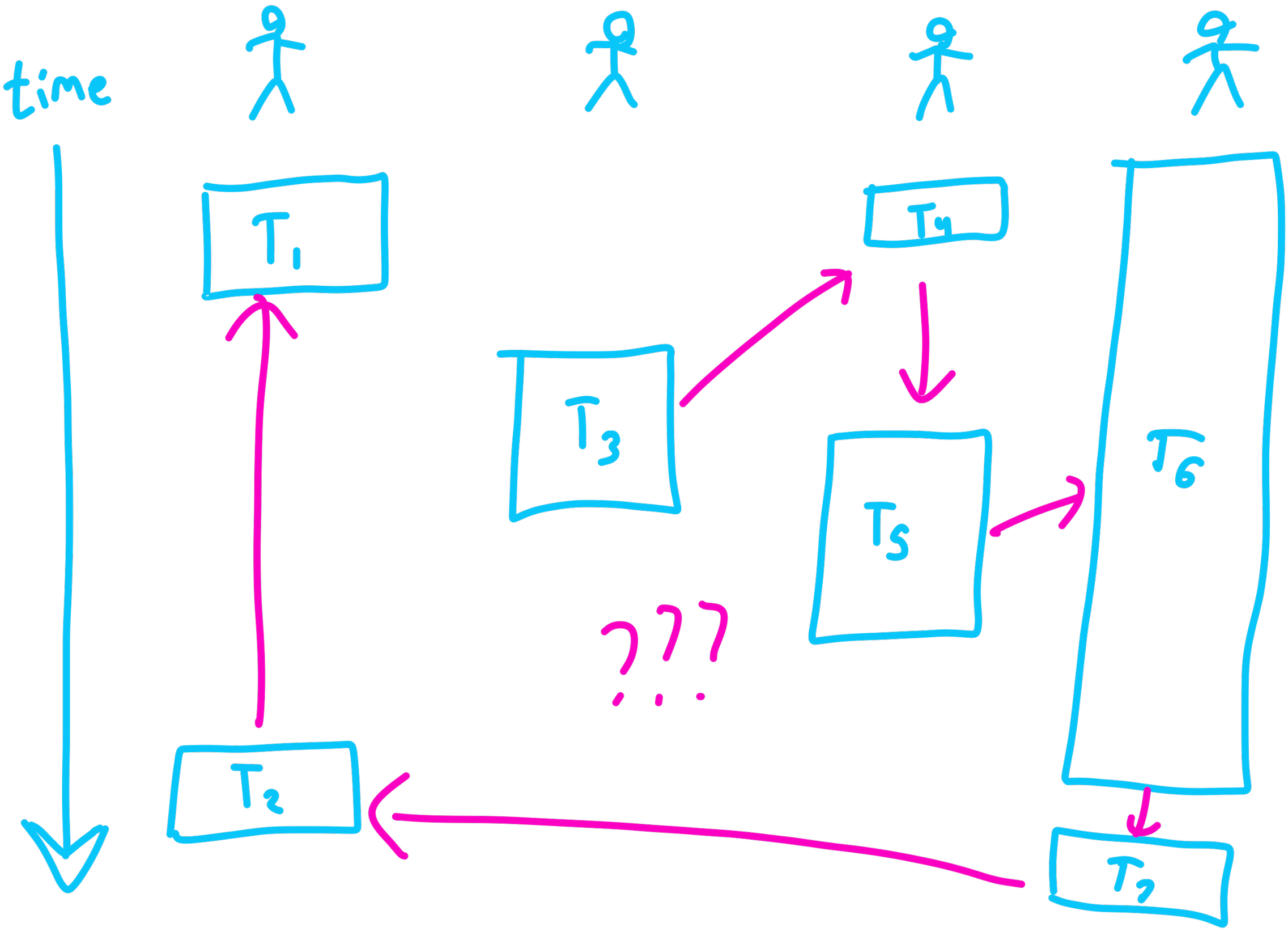
T_4

T_5

T_6

T_2

T_7


time
T1
T2
T3
T4
T5
T6
T7
???

~~$T_1\ T_2\ T_3\ T_4\ T_5\ T_6\ T_7$~~

~~$T_2\ T_1\ T_3\ T_4\ T_5\ T_6\ T_7$~~

~~$T_1\ T_3\ T_2\ T_4\ T_5\ T_6\ T_7$~~

$T_2\ T_3\ T_1\ T_4\ T_5\ T_6\ T_7$

~~$T_3\ T_1\ T_2\ T_4\ T_5\ T_6\ T_7$~~

$T_3\ T_2\ T_1\ T_4\ T_5\ T_6\ T_7$

~~$T_1\ T_2\ T_4\ T_3\ T_5\ T_6\ T_7$~~

~~$T_1\ T_4\ T_2\ T_3\ T_5\ T_6\ T_7$~~

~~$T_1\ T_4\ T_3\ T_2\ T_5\ T_6\ T_7$~~

Aha! Serializable!

$T_1\ T_2\ T_3\ T_4\ T_5\ T_6\ T_7$

$T_2\ T_1\ T_3\ T_4\ T_5\ T_6\ T_7$

$T_1\ T_3\ T_2\ T_4\ T_5\ T_6\ T_7$

$T_2\ T_3\ T_1\ T_4\ T_5\ T_6\ T_7$

$T_3\ T_1\ T_2\ T_4\ T_5\ T_6\ T_7$

$T_3\ T_2\ T_1\ T_4\ T_5\ T_6\ T_7$

$T_1\ T_2\ T_4\ T_3\ T_5\ T_6\ T_7$

$T_1\ T_4\ T_2\ T_3\ T_5\ T_6\ T_7$

$T_1\ T_4\ T_3\ T_2\ T_5\ T_6\ T_7$

$N!$

Gretchen

github.com/aphyr/gretchen

(Cerone, Bernardi, Gotsman 2015)

Serializability

INT

Check in each txn separately

EXT

Build read dependency Constraint problem in CNF

TOTALVIS

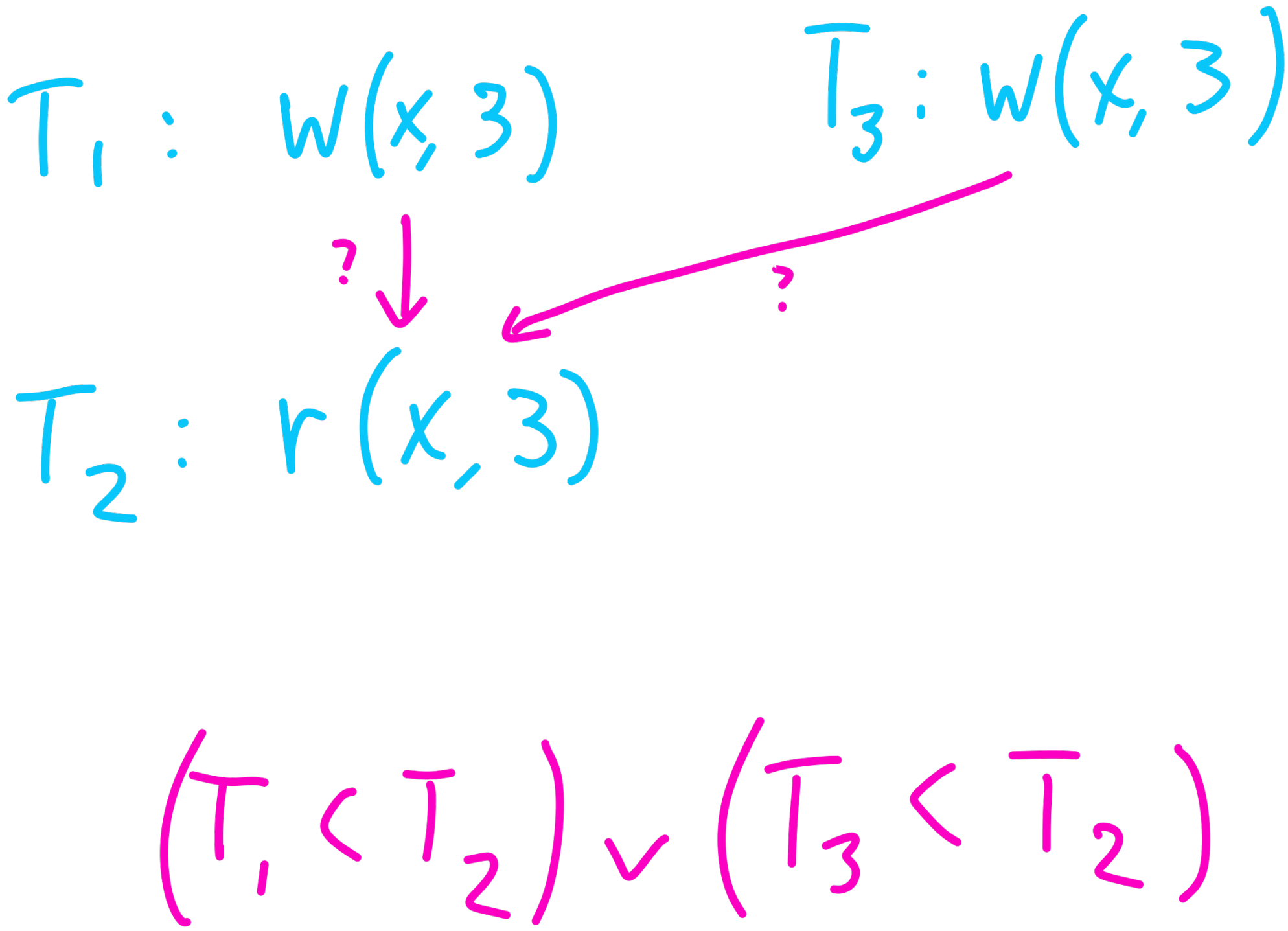
Solve for order using integer constraint solver

T_1 : $w(x, 3)$

↓

T_2 : $r(x, 3)$

$T_1 < T_2$

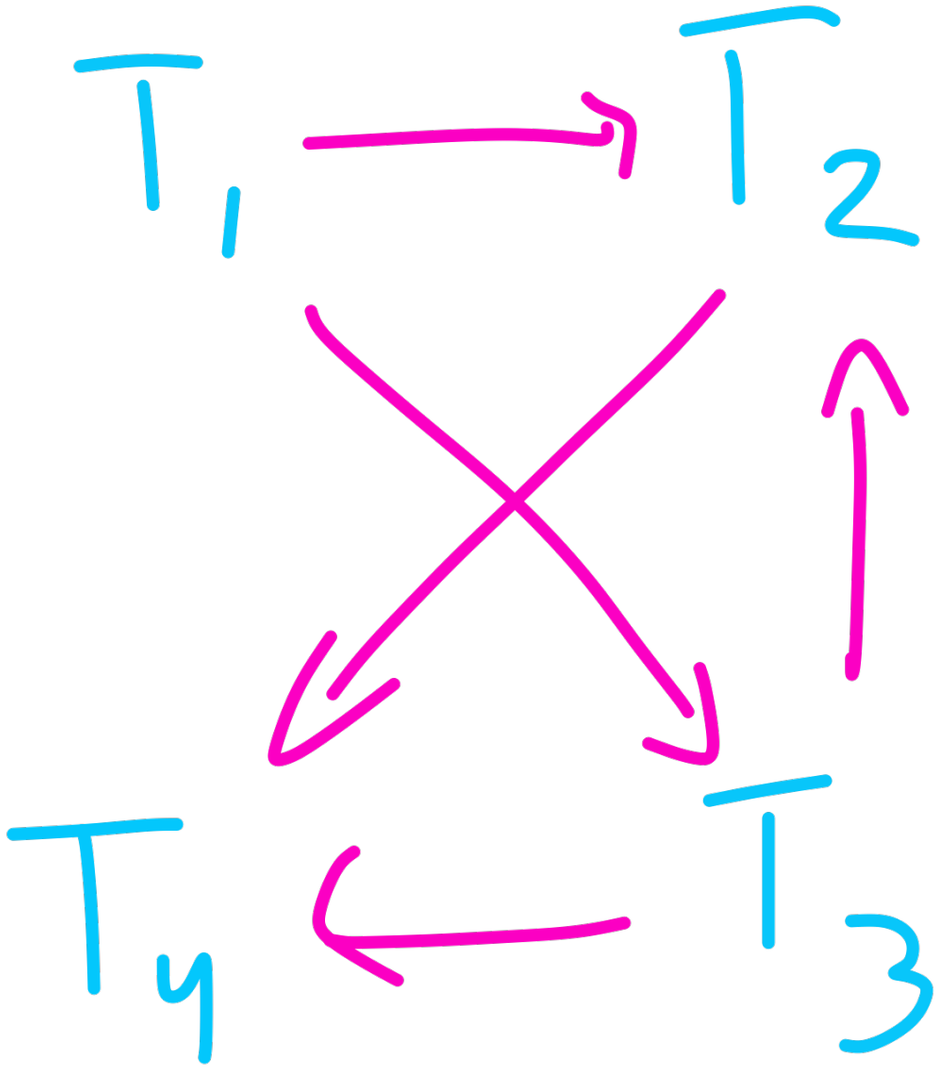


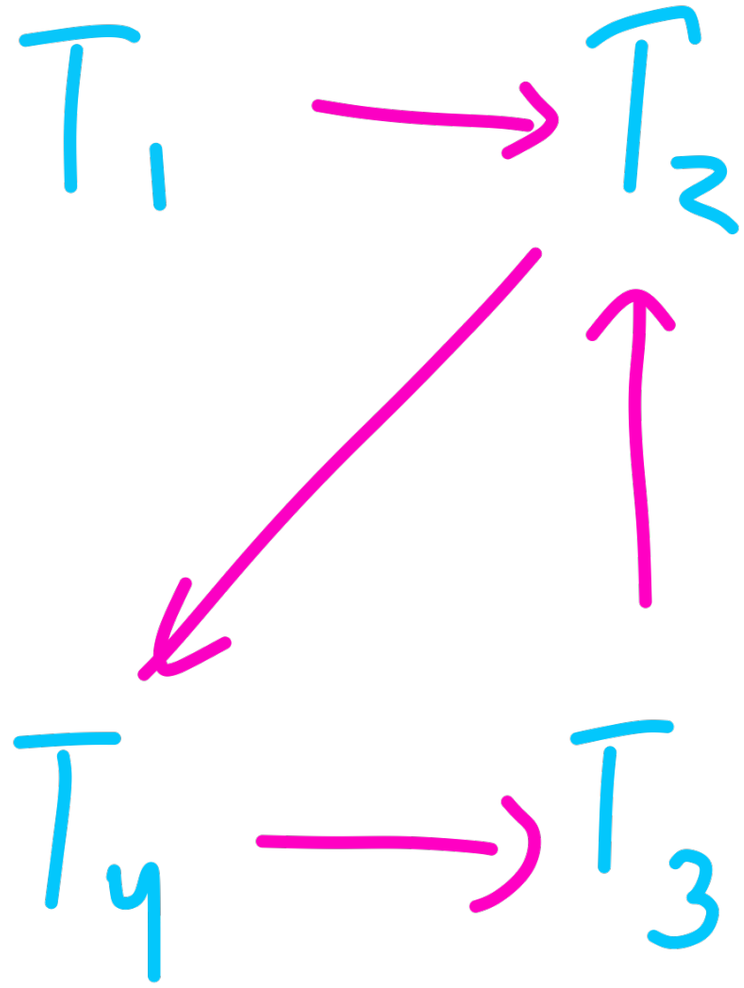
$$(T_1 < T_2) \vee (T_3 < T_2)$$

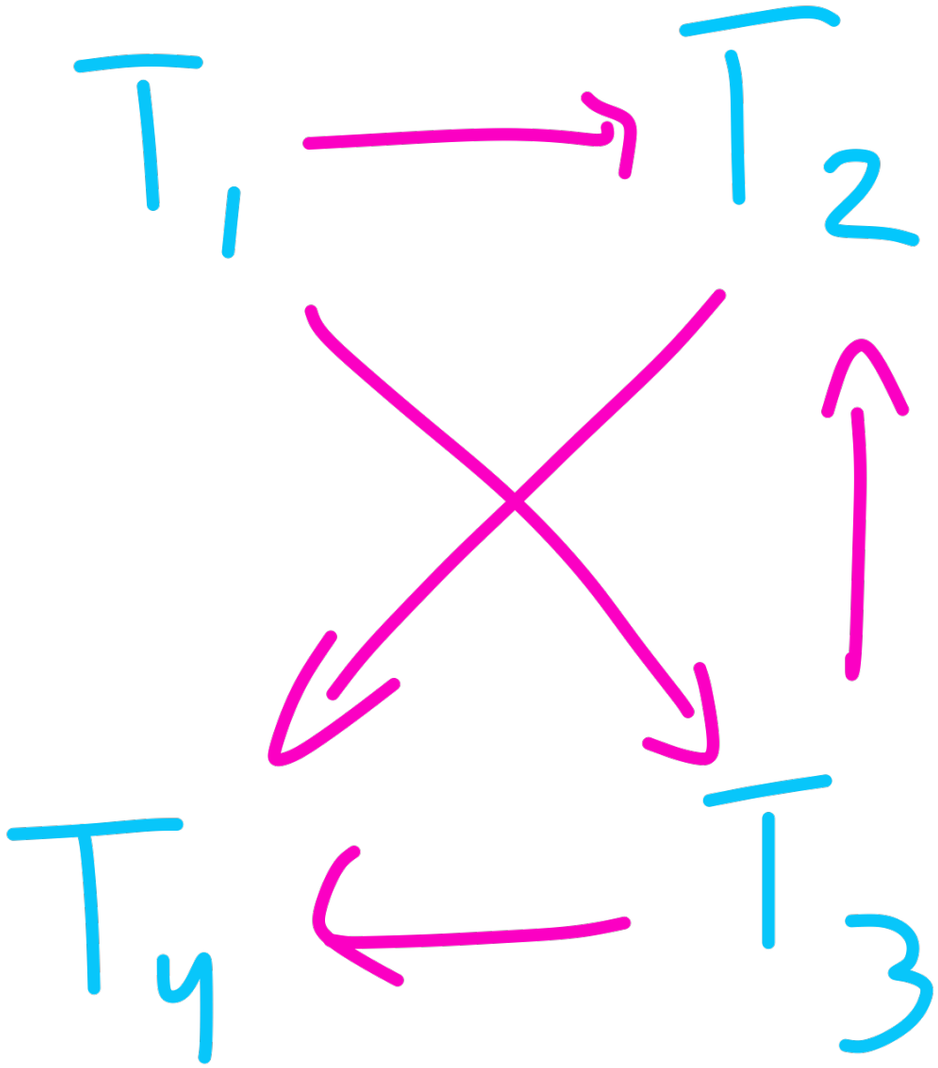
- SAT is ALSO NP-Complete
- Breaks down after ~100 txns
- Doesn't explain why no soln!

Cycle Detection

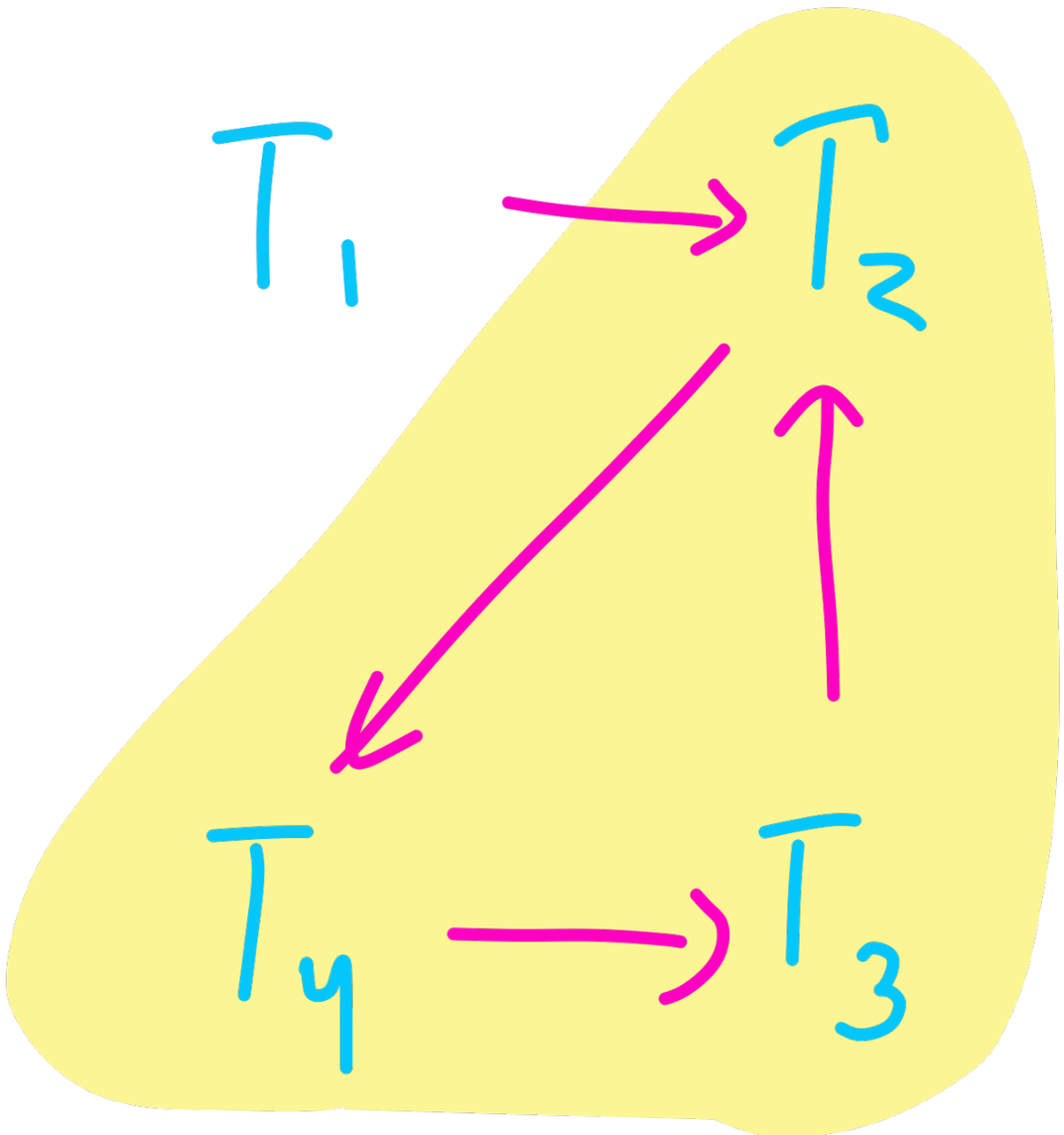
Possibility of a Total Order


T1
T2
T3
T4


T1
T2
T3
T4



Serializable



Not Serializable!

Adya 1995

Isolation Levels in terms of Dependency Cycles

Write – Read Dependency

$T_1: w(x_i) \longrightarrow T_2: r(x_i)$

Write — Write Dependency

$T_1: w(x_i) \longrightarrow T_2: w(x_j)$

$x_i \ll x_j$

Version Order $\ll$

Total order of installed xs, ys, etc.

Read — Write Dependency

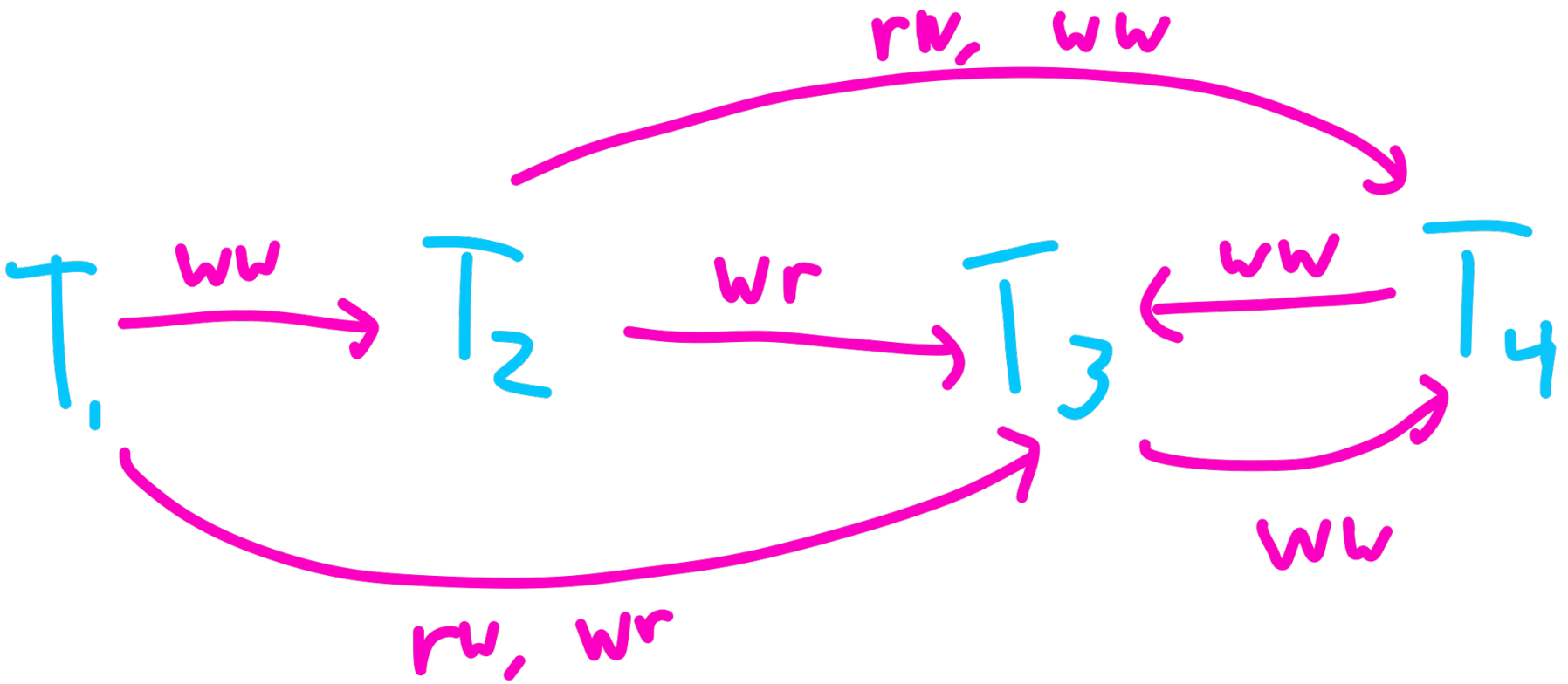
$T_1: r(x_i) \longrightarrow T_2: w(x_j)$

$x_i \ll x_j$

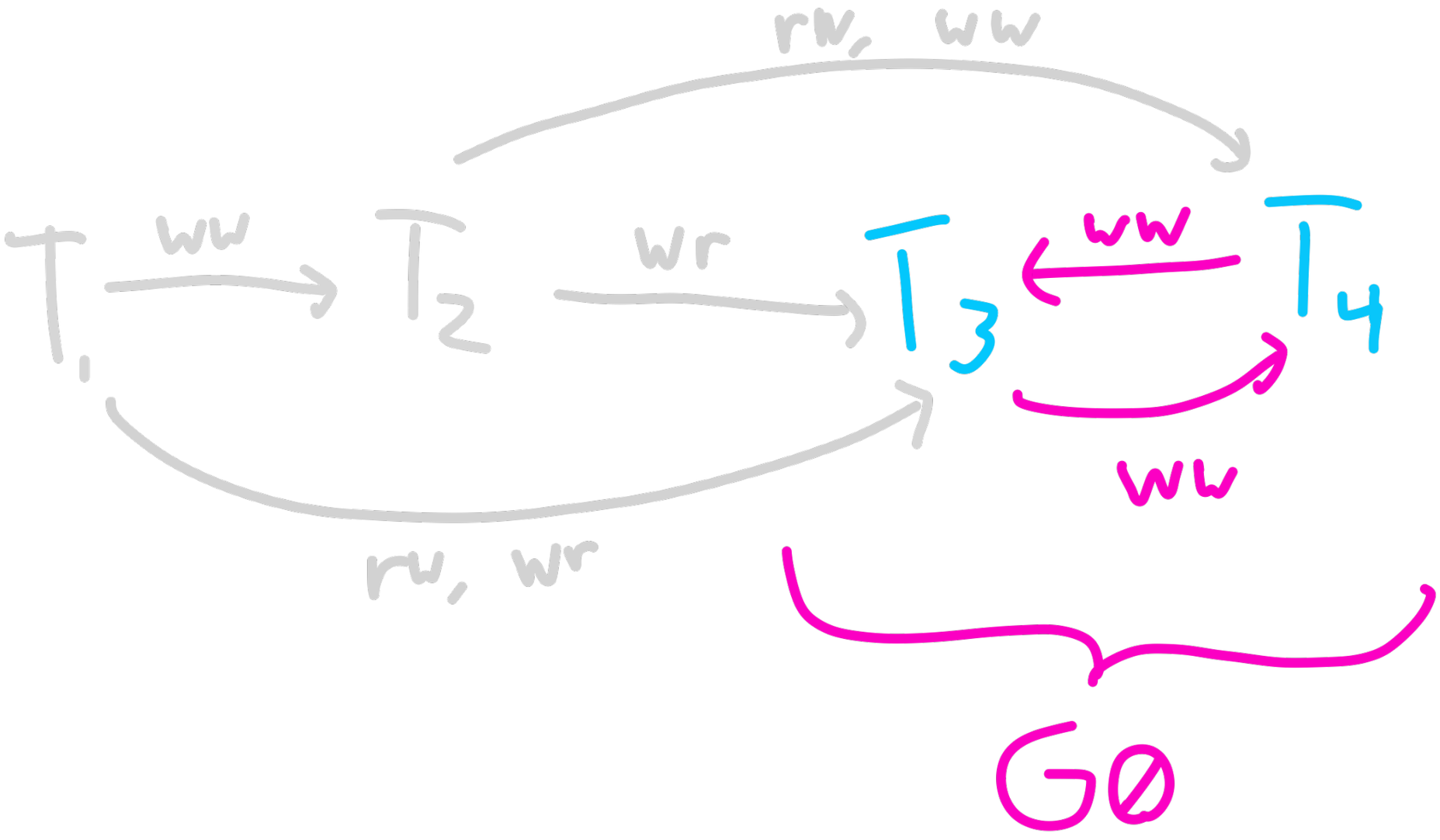
Version Order $\ll$

Total order of installed xs, ys, etc.

Direct Serialization Graph



Direct Serialization Graph



G0	ww cycle
G1a	aborted read
G1b	intermediate read
G1c	ww + wr cycle
G-single	ww+wr+1rw cycle
G2	ww+wr+rw cycle

Great! Where do we get an Adya History?

T_1: $w(x_1)$, $w(y_1)$

T_2: $r(x_1)$, $w(y_2)$

Great! Where do we get an Adya History?

T_1: $w(x_1)$, $w(y_1)$

↓? ↓? ↑?

T_2: $r(x_1)$, $w(y_2)$

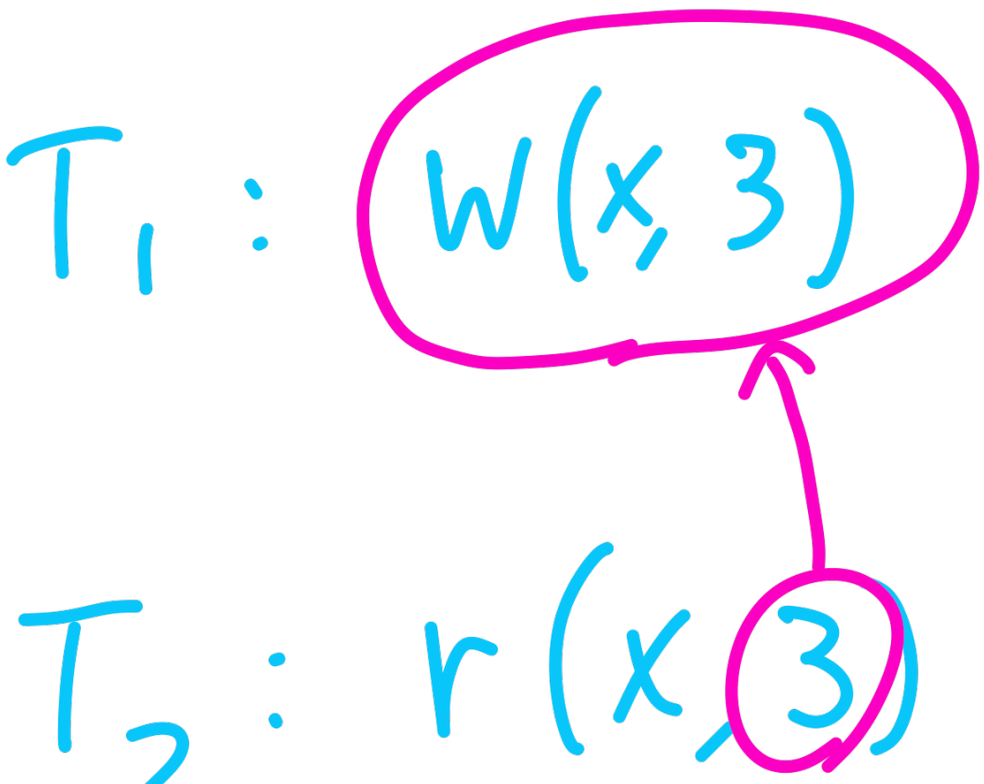
Only write of x_1?

Missing

$y_1 \ll y_2$

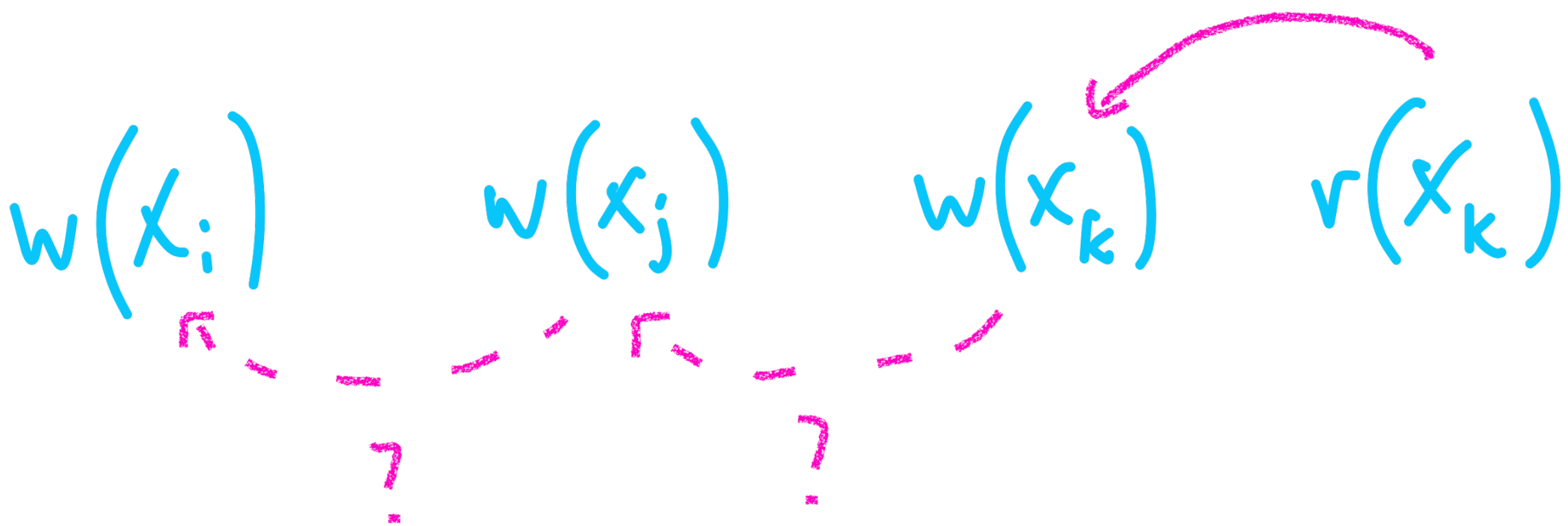
or

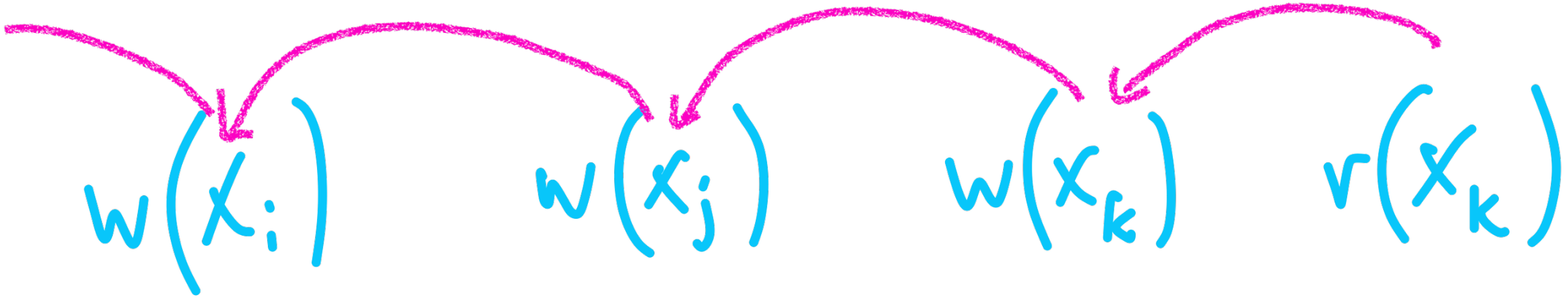
$y_2 \ll y_1$



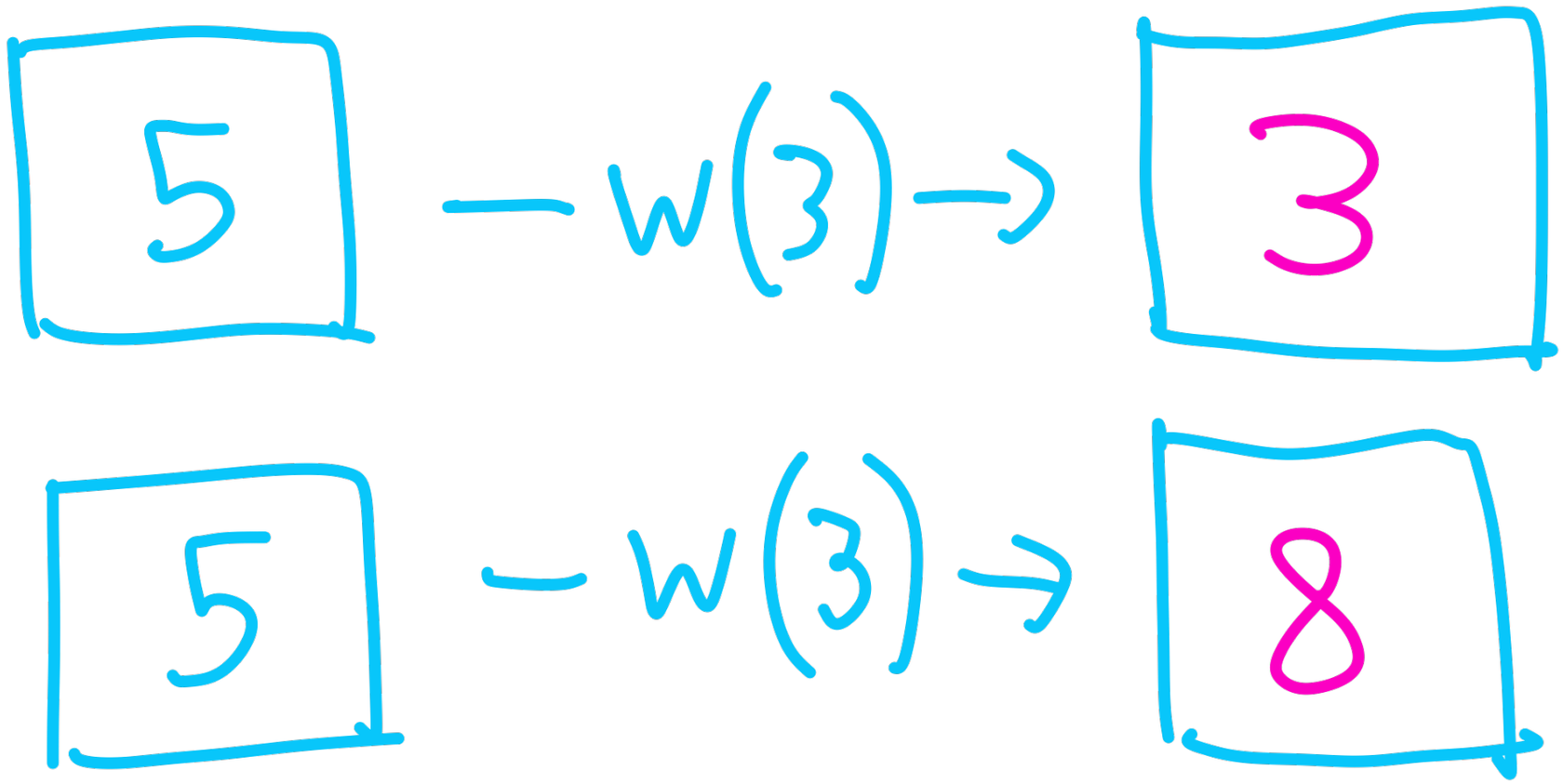
Recoverability

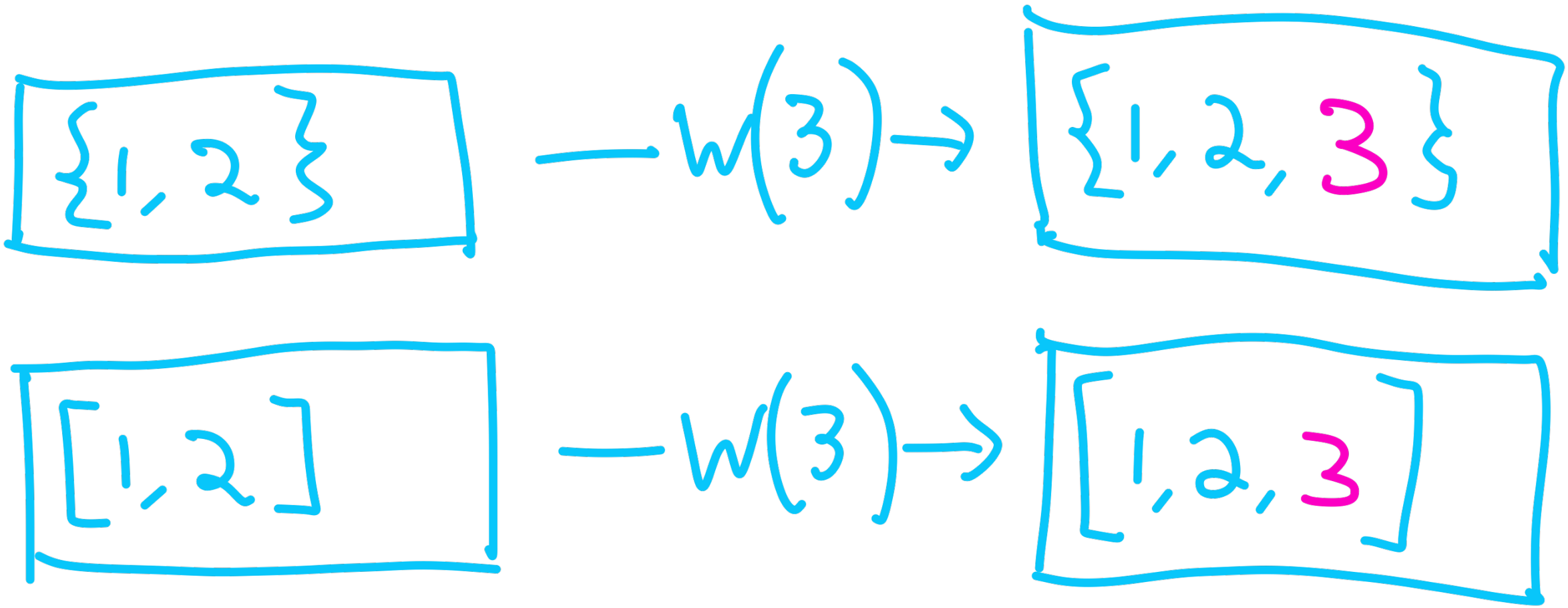
Blind Writes Destroy History



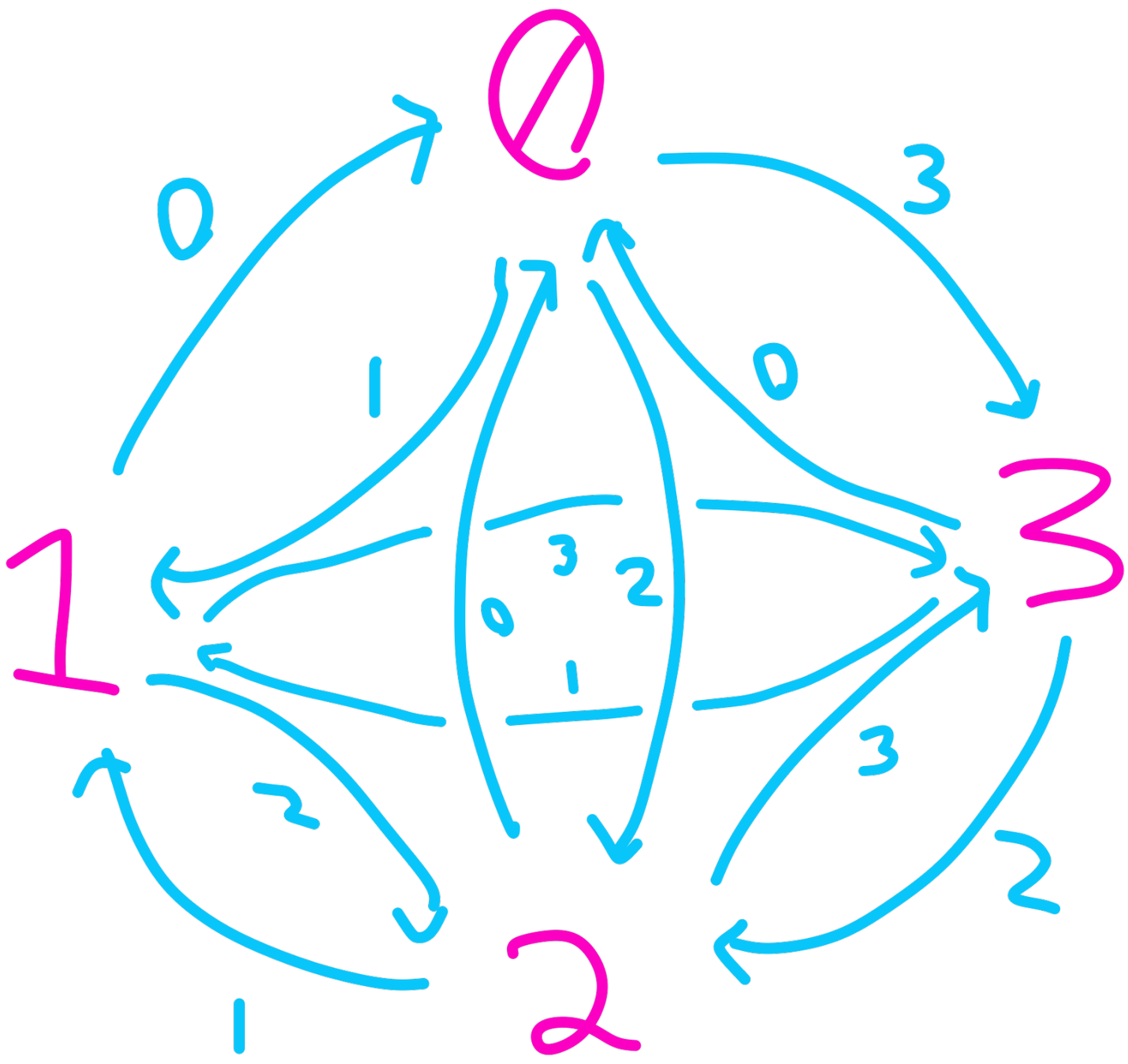


Traceability

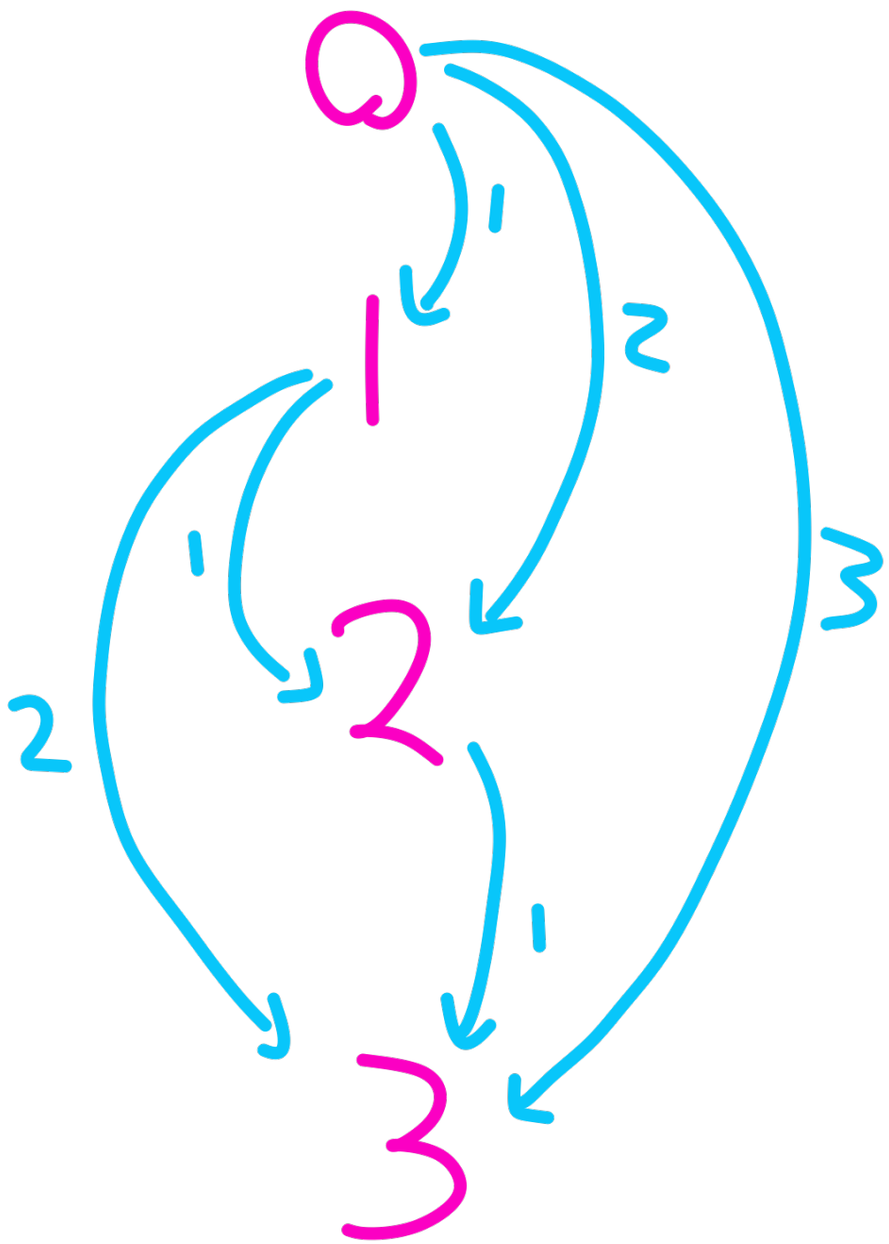

5
w(3)
3
5
w(3)
8


{1,2}
w(3)
{1,2,3}
[1,2]
w(3)
[1,2,3]

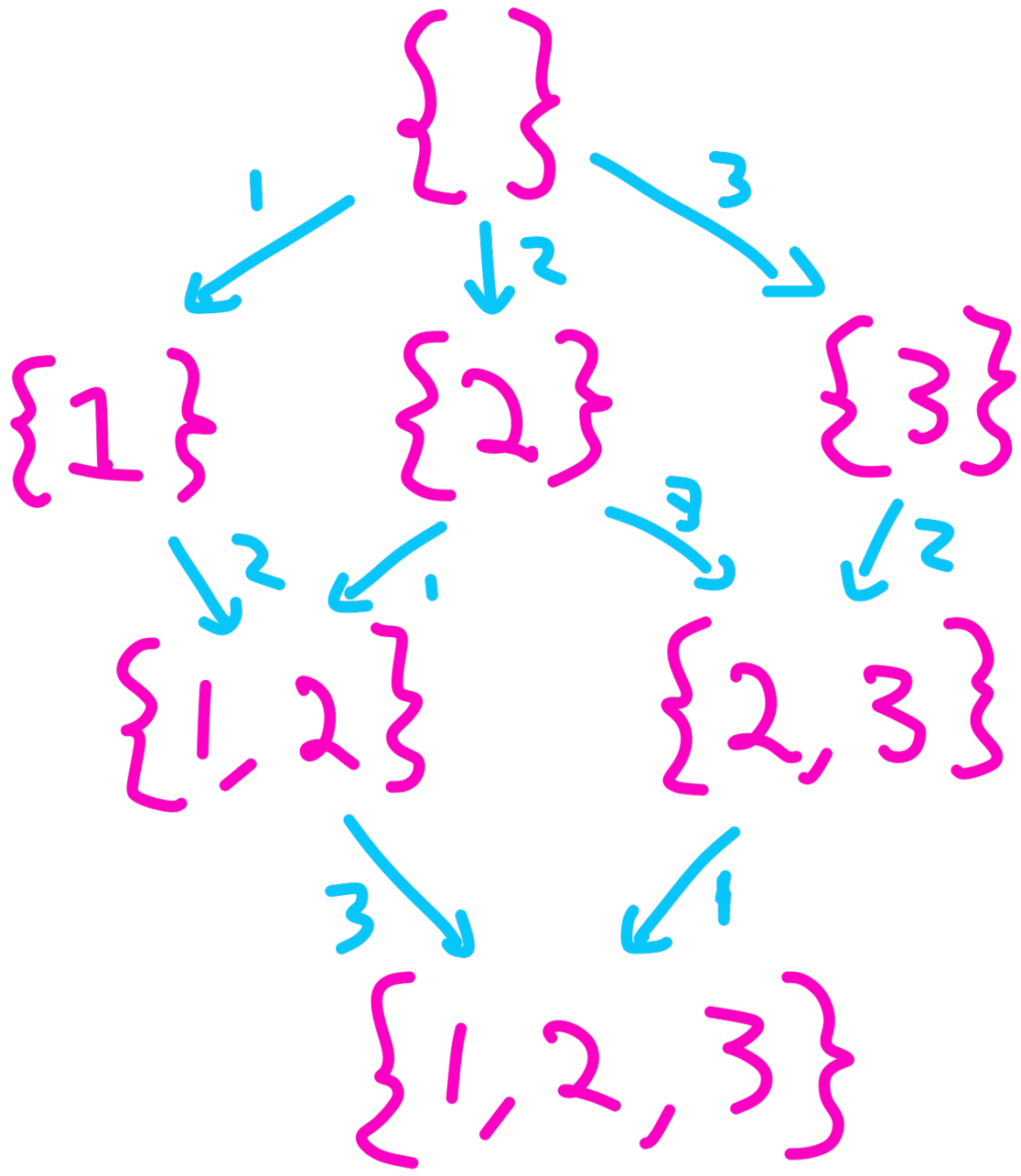
Registers



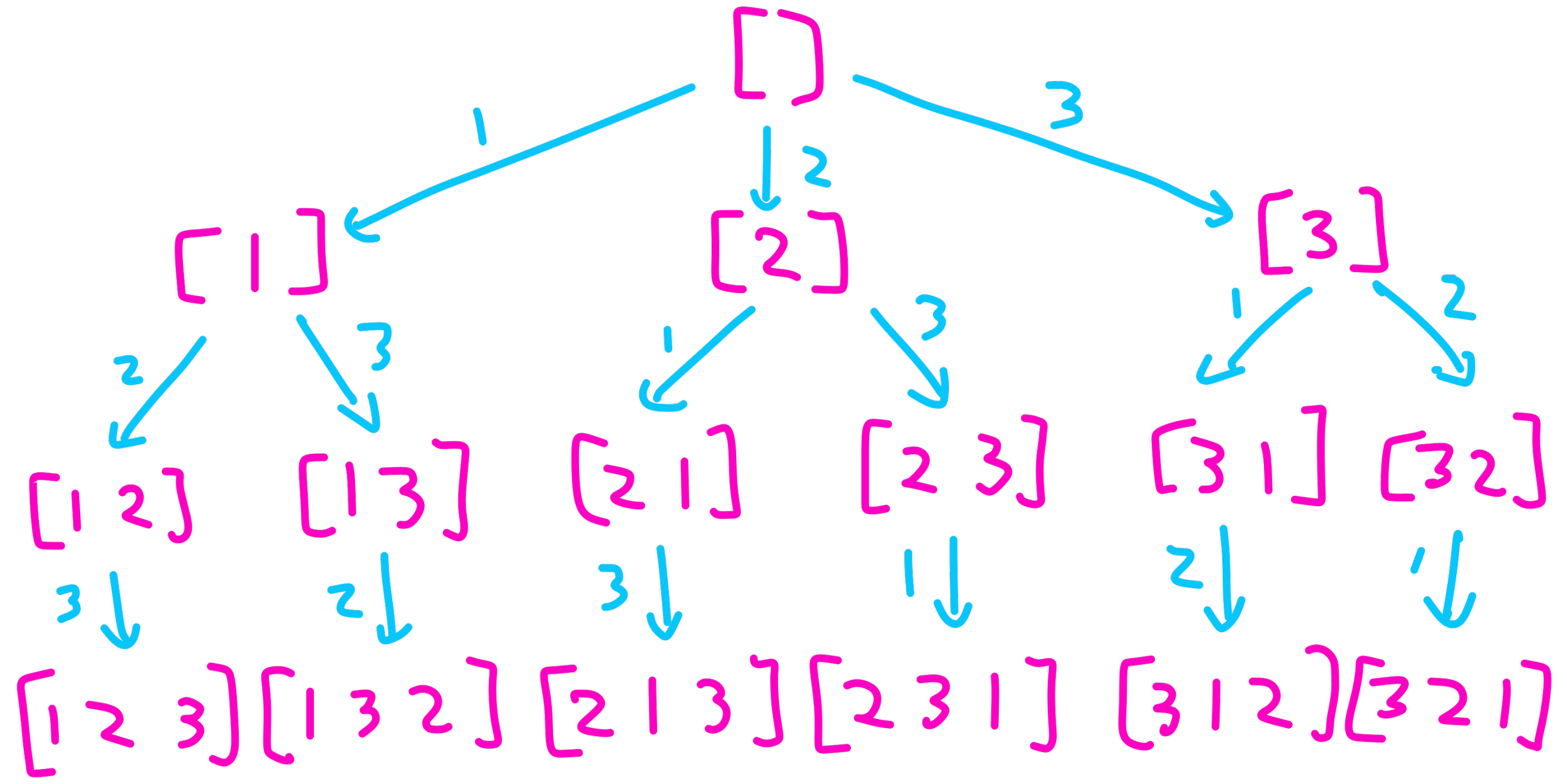
Counters



Sets



Lists



Inferred { [ ] → 1 → [1] → 3 → [1 3] → 2

Given { [1 3 2]

Traceability

- Lists w/ append!
- Logs w/ insert!
- Strings w/ concat!

It's in the SQL standard!

EXAMPLES

T_1: $w(x,1)$, $w(y,1)$

T_2: $w(x,2)$, $w(y,2)$

T_1: $w(x,1)$, $w(y,1)$

ww ↓ ↑ ww

T_2: $w(x,2)$, $w(y,2)$

GO

T_3: $r(x,[1,2])$, $r(y,[2,1])$

T_1: $r(x,[0,\underline{1}])$, $w(x,2)$

wr ↑ ↙ ww

T_2: $w(x,\underline{1})$, $w(x,3)$

] G1c

T_3: $r(x,[0,1,\underline{2},\underline{3}])$,

⋮

T_1: r(x, [ ]), w(y, 1)

rw ↓

T_2: w(x, 2), r(y, [ ])

T_3: r(x, [1, 2])

↑ rw

G2

Well yes, but

HOW

T_1

T_2

T_3

T_4

T_5

T_6

T_7

T_8

T_9

T_{10}

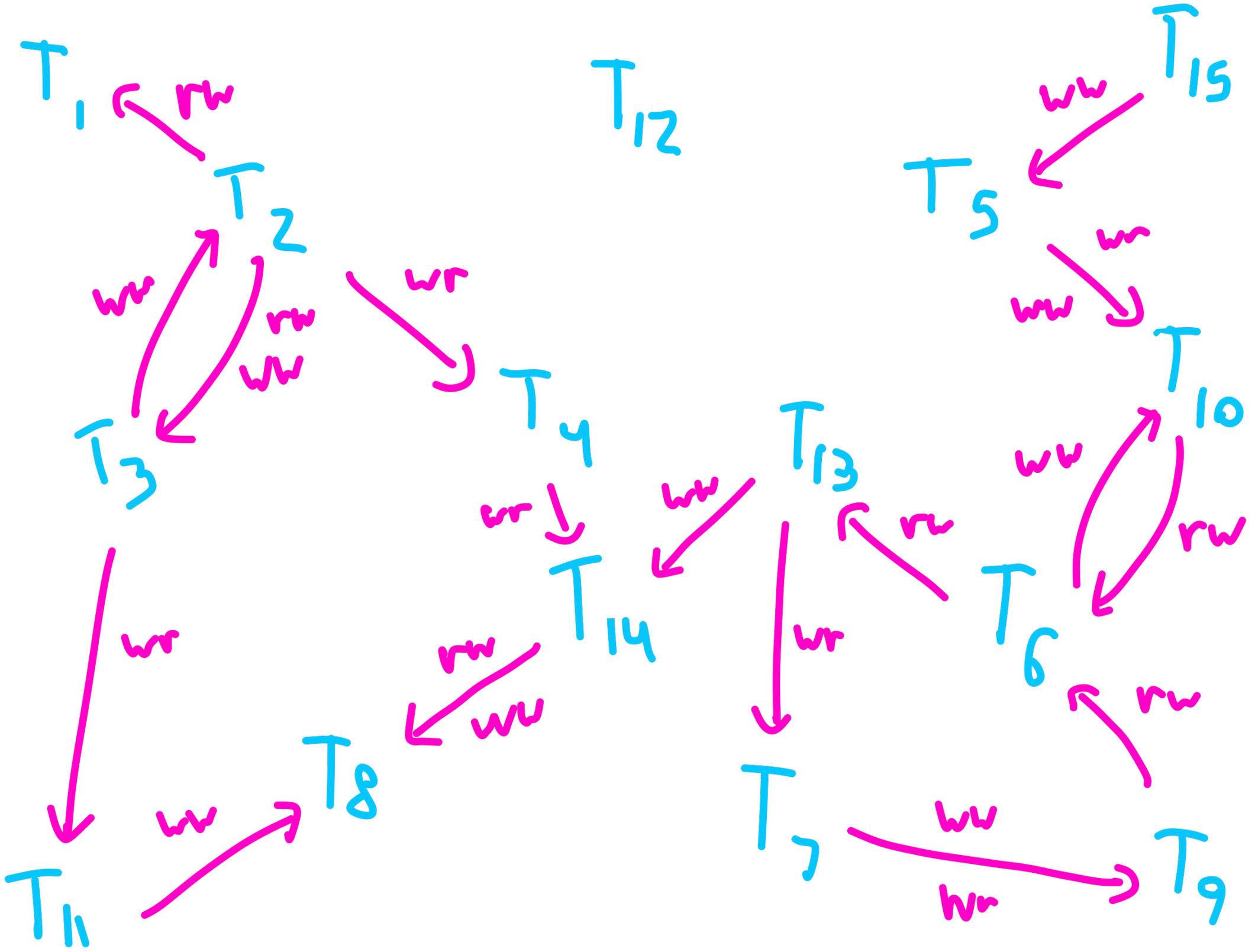
T_{11}

T_{12}

T_{13}

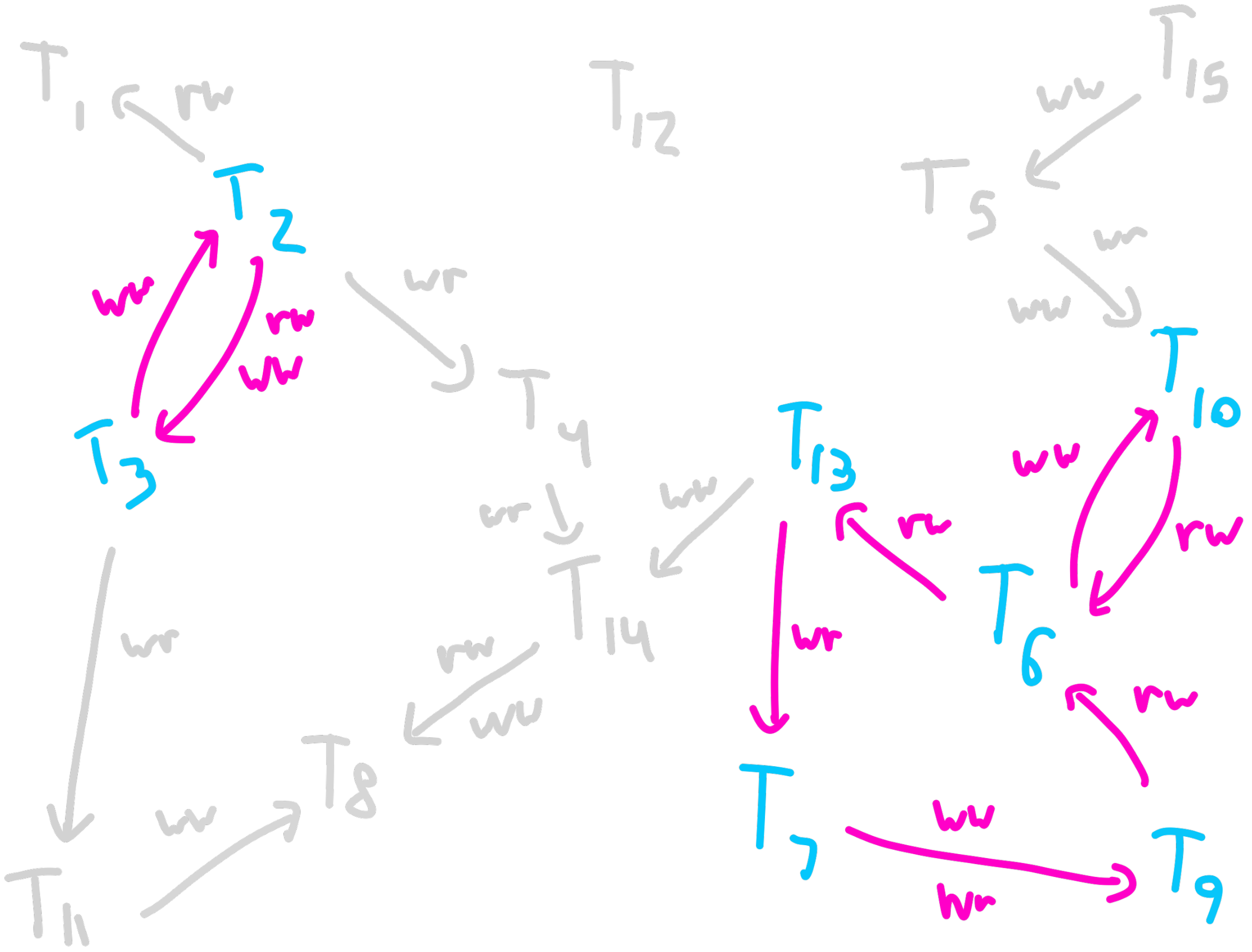
T_{14}

T_{15}


T1
rw
T2
ww
rw
ww
wr
T4
T12
T13
wr
ww
wr
T14
rw
ww
T3
wr
T11
ww
T8
T15
ww
T5
ww
T10
rw
T6
rw
T7
ww
T9

Tarjan 1972

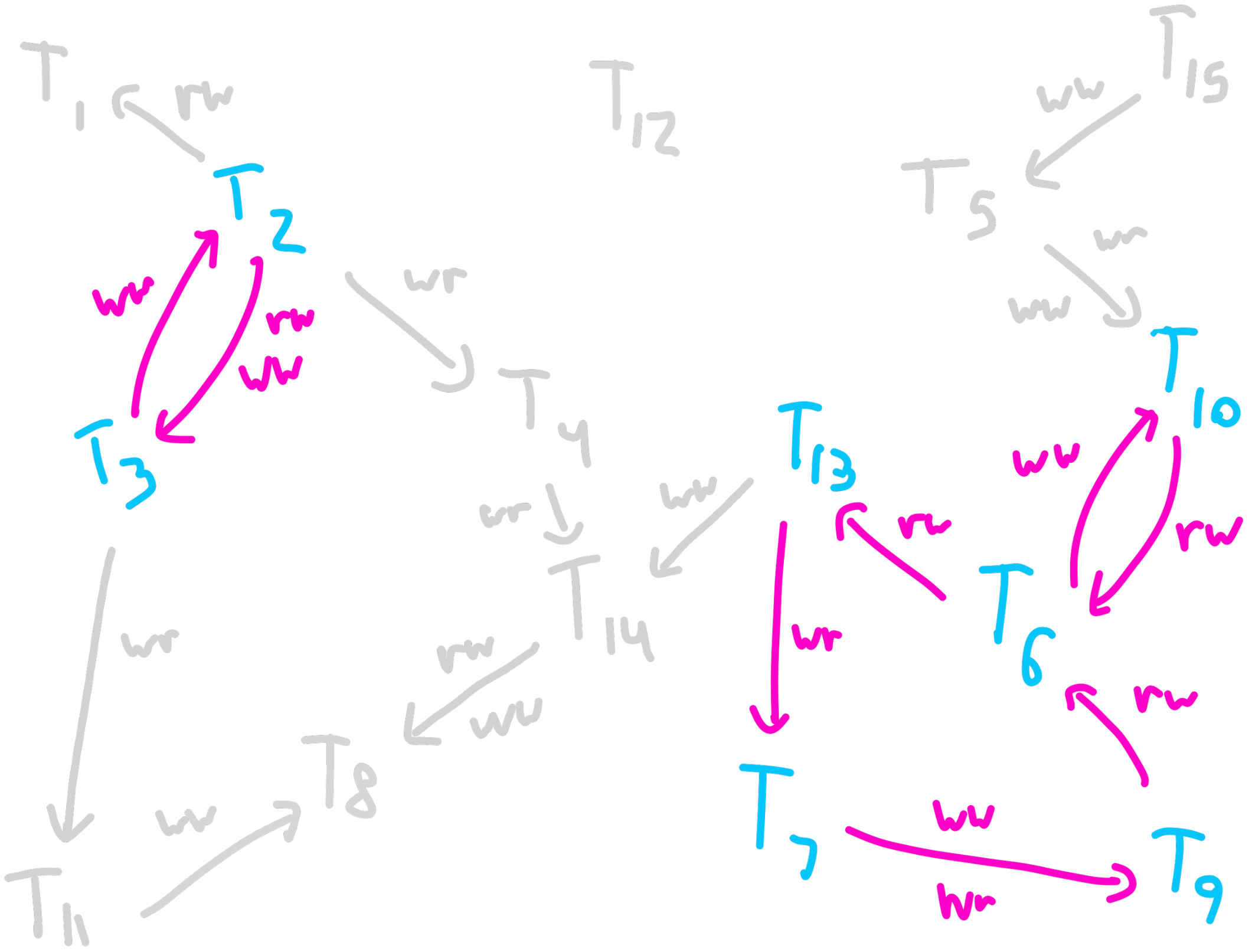
Strongly Connected Components in O(vertices + edges)


T1
rw
T2
ww
rw
ww
wr
T3
wr
T4
T12
T13
ww
wr
T14
rw
ww
T8
ww
T11
T15
ww
T5
ww
ww
T10
rw
rw
T6
wr
rw
T7
ww
wr
T9

$T_2 \xrightarrow{ww} T_3 \xrightarrow{ww} T_2$: G0

$T_2 \xrightarrow{rw} T_3 \xrightarrow{ww} T_2$: G-single

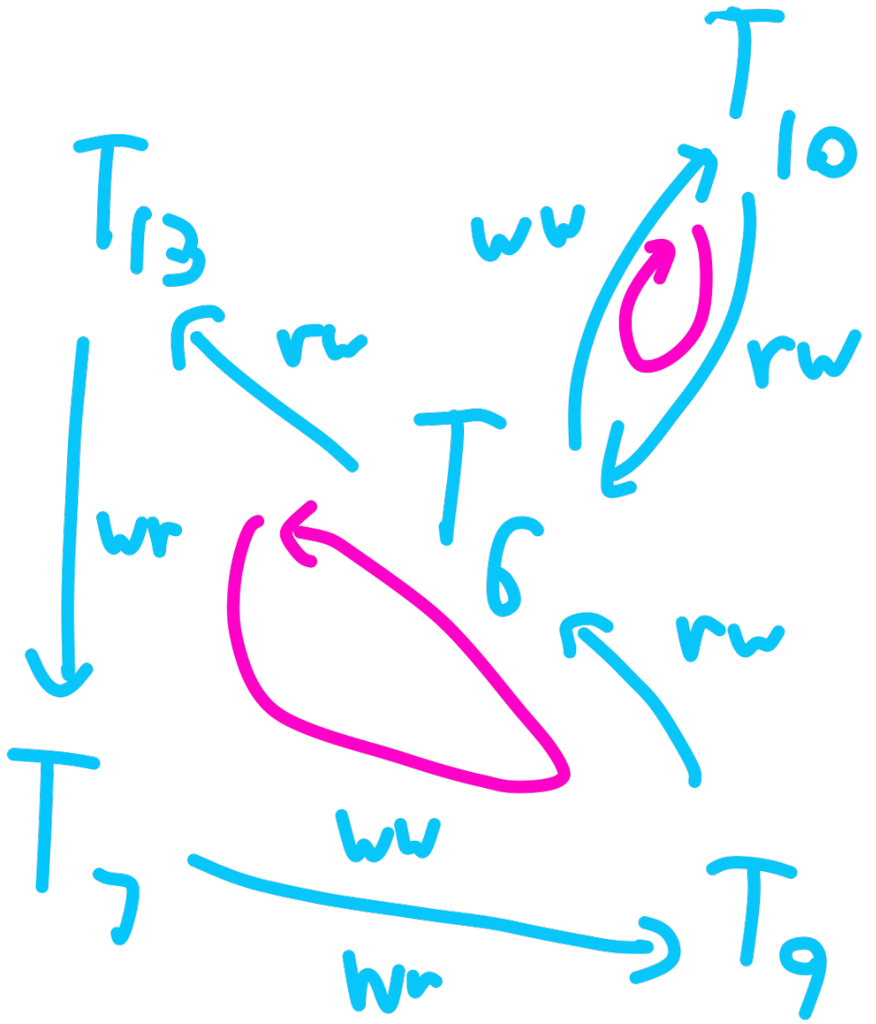
BFS


T1
rw
T2
ww
rw
ww
wr
T3
wr
T4
T12
T13
ww
wr
T14
rw
ww
T8
ww
T11
T15
ww
T5
ww
ww
T10
rw
rw
T6
wr
rw
T7
ww
wr
T9

$T_{10} \xrightarrow{rw} T_6 \xrightarrow{ww} T_{10}$: G-Single

$T_{13} \xrightarrow{wr} T_7 \xrightarrow{ww} T_9 \xrightarrow{rw} T_6 \xrightarrow{rw} T_{13}$: G2

BFS



Automatic Explanations

$T_1 < T_2$ because T_2 observed T_1's write of 3 to x.

$T_2 < T_3$ because T_2 did not observe T_3's write of 5 to y.

$T_3 < T_1$ because T_1 appended 3 to z before T_3 appended 7.

A contradiction! □

— more —

CONSTRAINTS

time

T_1

T_2

T_3

T_4

T_5

T_6

T_7

Process Order

time

T_1

T_4

T_3

T_5

T_6

T_2

T_7

Realtime Order

time

1 3 T_1

1 2 T_4

3 4 T_3

3 5 T_5

1 6 T_6

5 7 T_2

6 8 T_7

Start-Ordered Serialization Graph

time

T_1 2

T_4 1

T_3 3

T_5 4

T_6 5

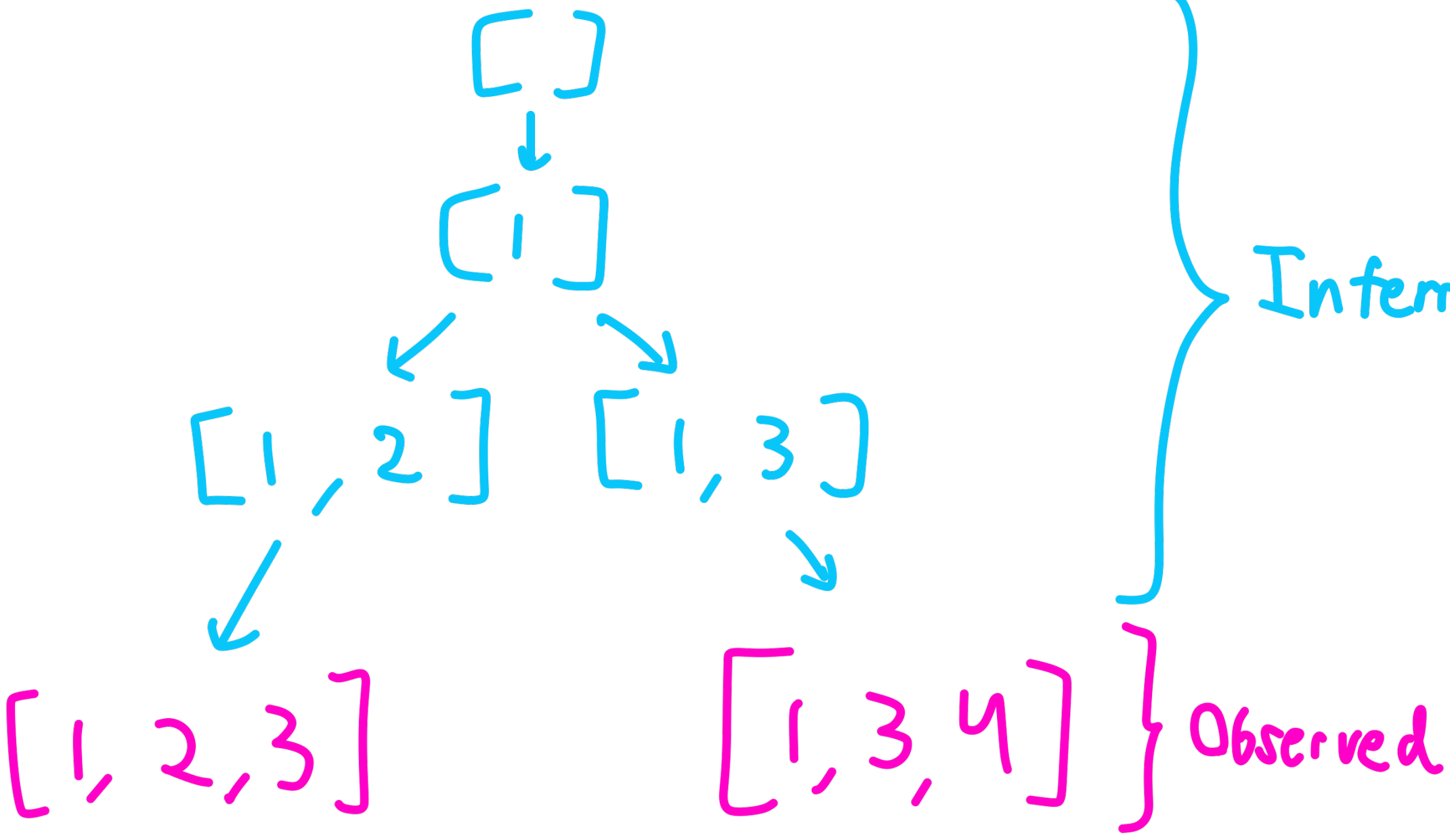
T_2 6

T_7 7

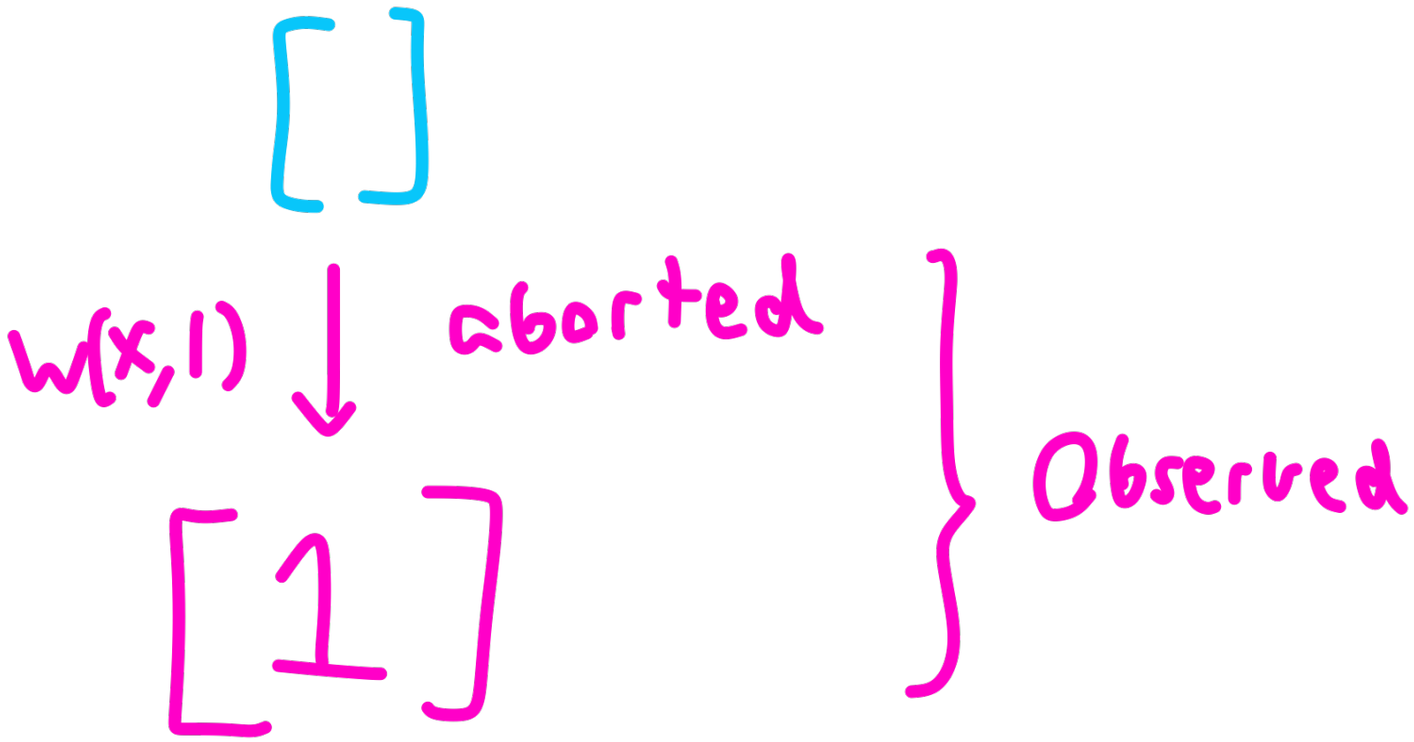
Explicit Version Order

Other

ANOMALIES



Aborted Read



Aborted Read

[]

w(x,1) ↓ aborted } Observed

[1]

w(x,2) ↓ committed } Observed

[1,2]

Dirty Update

[1, 1, 2]

Duplicate Element

[1,  , 3]

Garbage Element

T_i: w(x,3), ..., r(x, [..., 2])

Internal Inconsistency

Does it

WORK?

TiDB 2.1.7 ~ 3.0.0-beta1

- Claim: SI
- Found: G2

G-single

Lost Updates

Aborted Reads

TiDB 2.1.7 ~ 3.0.0-beta1

- Claim: select... for update

  ⇒ No write skew

- Found: Write skew on new rows

Yugabyte DB 1.3.1

- Claim: Serializable
- Found: G2-item when master nodes paused/crashed

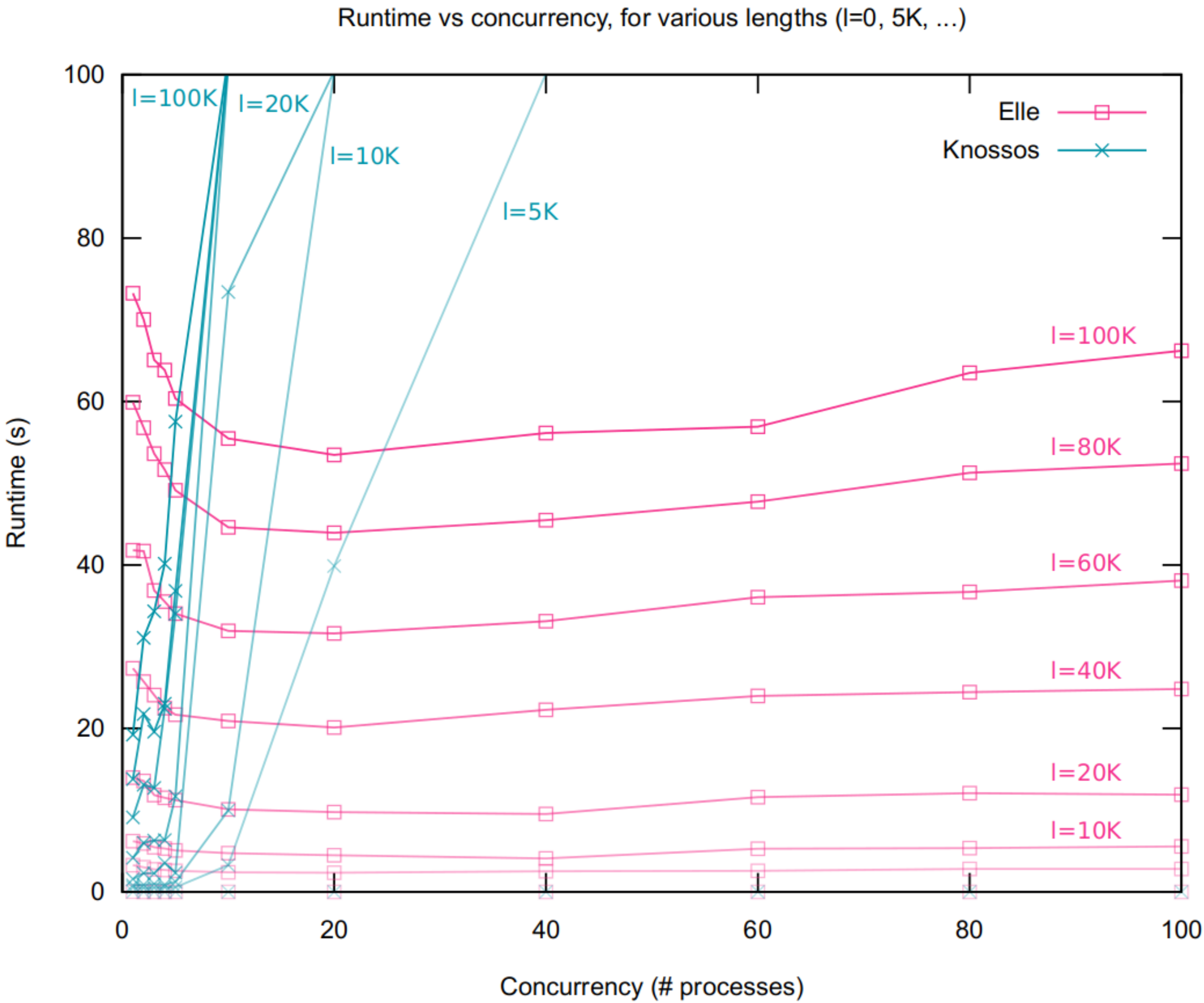
FaunaDB 2.6.0

- Claim: SI → Strict-1SR
- Found: Internal Inconsistencies in index reads: did not reflect txn's prev writes

And it's

Fast

too!


Runtime vs concurrency, for various lengths (l=0, 5K, ...)
Runtime (s)
Concurrency (# processes)
0
20
40
60
80
100
Elle
Knossos
l=100K
l=20K
l=10K
l=5K
l=100K
l=80K
l=60K
l=40K
l=20K
l=10K

Runtime vs history length, for various concurrencies (c=1, 2, ...)

Elle
Knossos

c=40
c=20
c=10
c=1
c=100
c=5
c=20
c=5
c=1

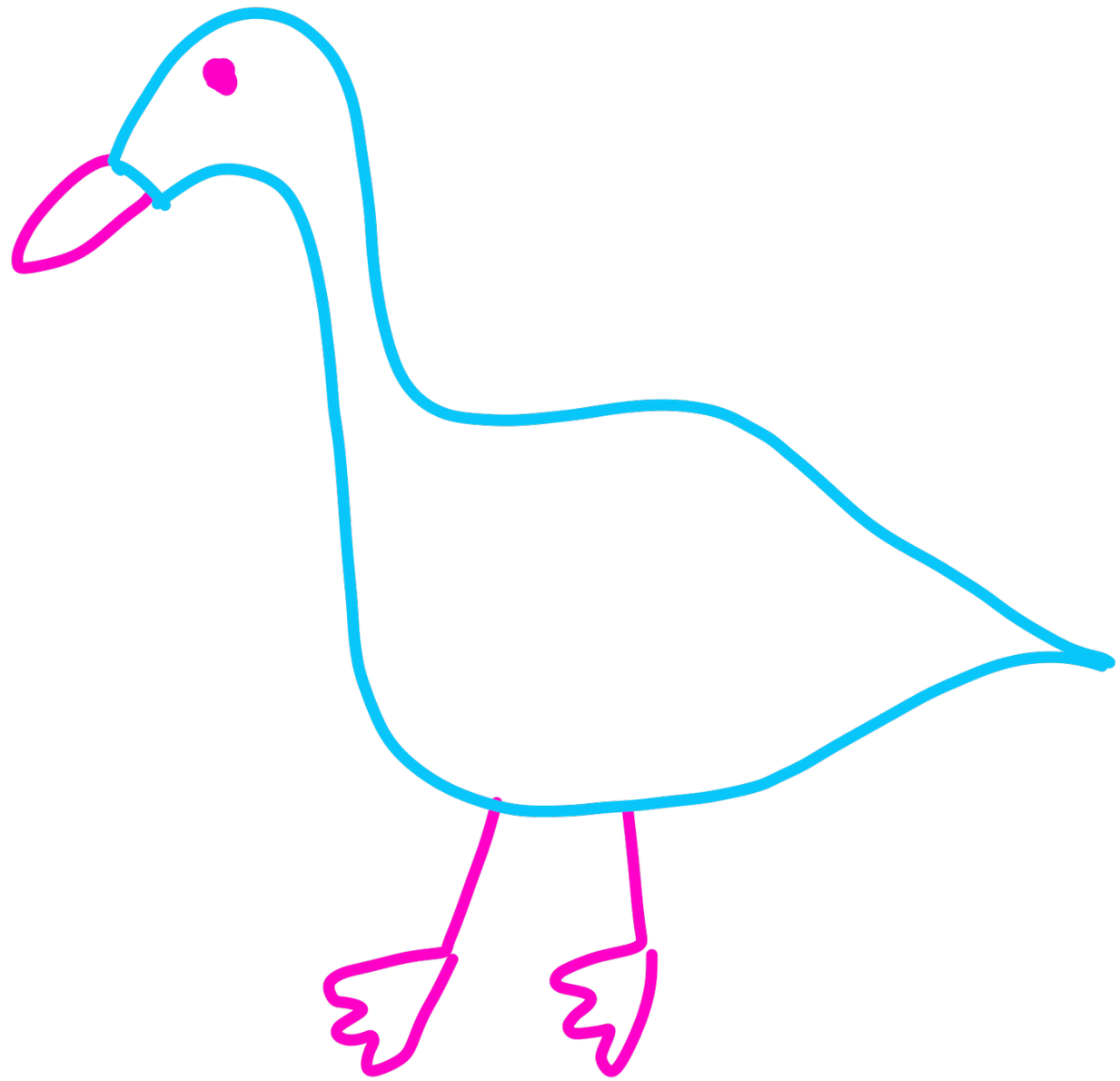
Runtime (s)
0
20
40
60
80
100

0
20000
40000
60000
80000
100000

Operations

ELLE

- General
- Effective
- Efficient
- Sound
- Explainable



Thanks

Peter Alvaro

Asha Karim

Kit Patella