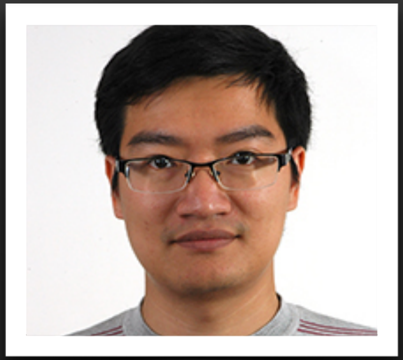
Verifying Weakly Consistent Systems (TSO as an Example)

Parosh Aziz Abdulla[1]

Mohamed Faouzi Atig[1]

Ahmed Bouajjani[2]



Tuan Phong Ngo[1]

[1]Uppsala University

[2]IRIF, Université Paris Diderot & IUF

Outline

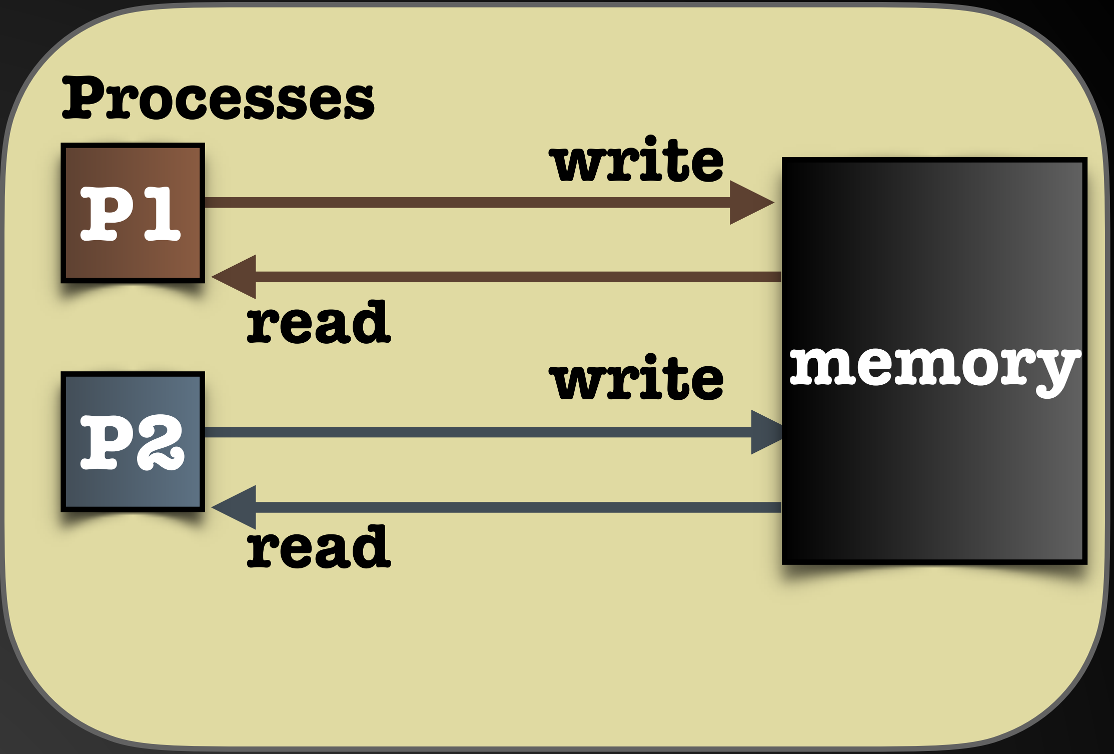
- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Monitors
- Synthesis

Outline

- **Weak Consistency**
- **Total Store Order (TSO)**
- **Dual TSO**
- **Verification**
- **Monitors**
- **Synthesis**

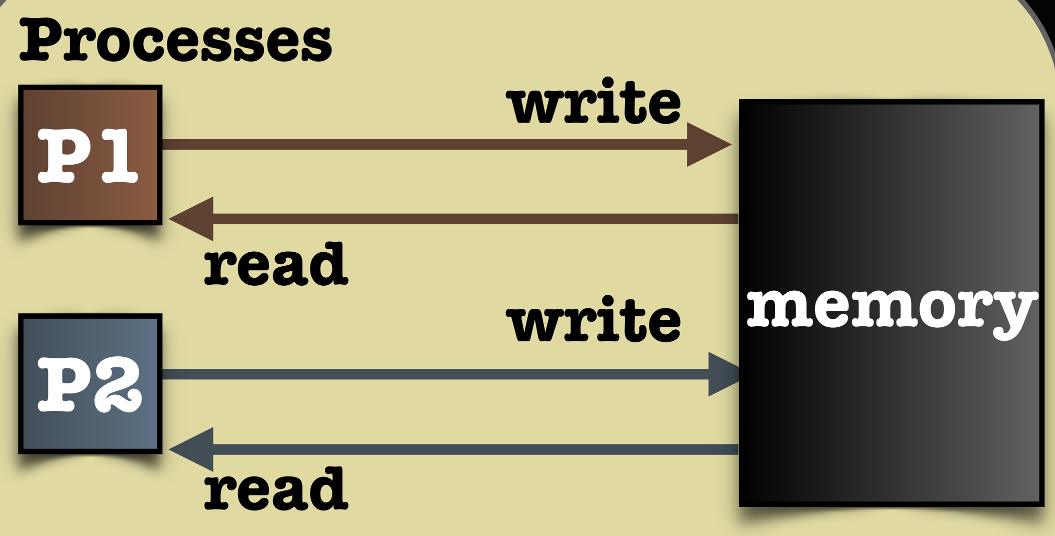
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations



Sequential Consistency (SC)

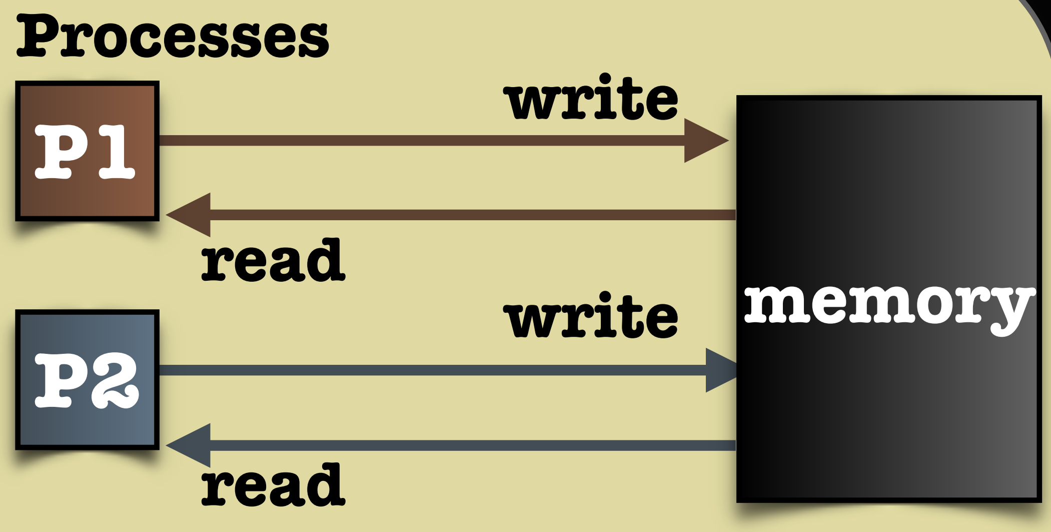
- Shared memory
- Processes: atomic read/write
- Interleaving of the operations



Execution

Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

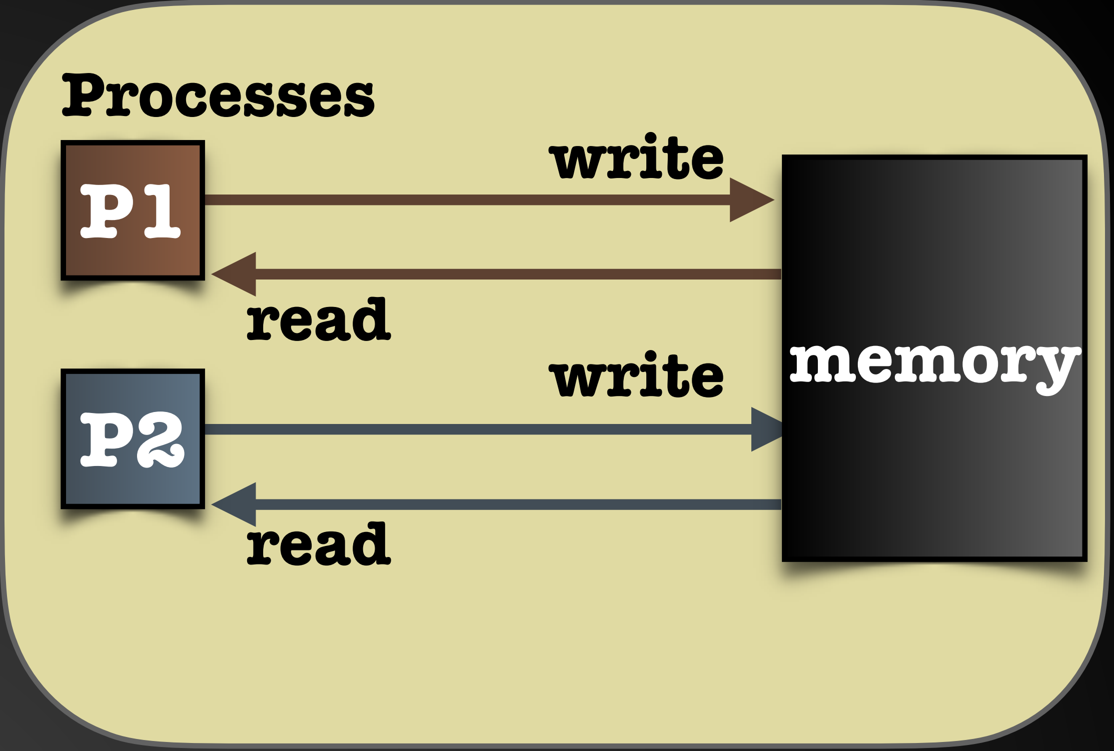


P1: w(x,1)

Execution

Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations



Processes

P1

write

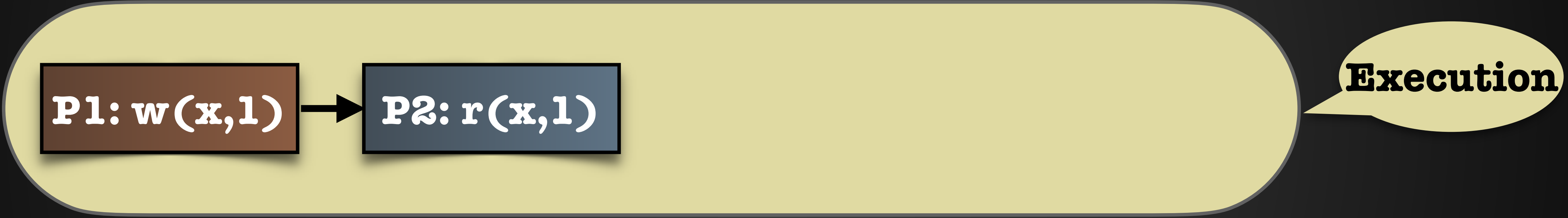
read

P2

write

read

memory

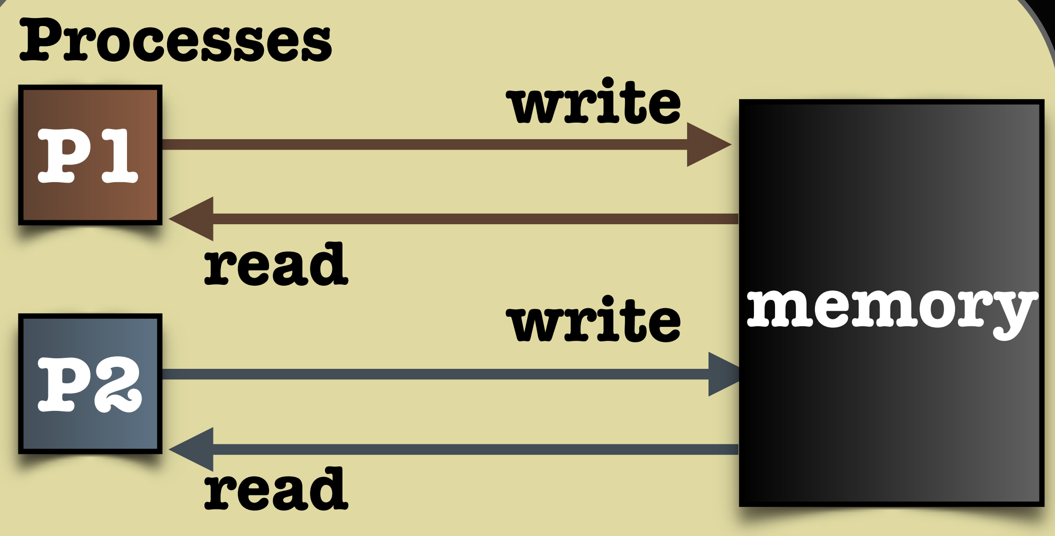


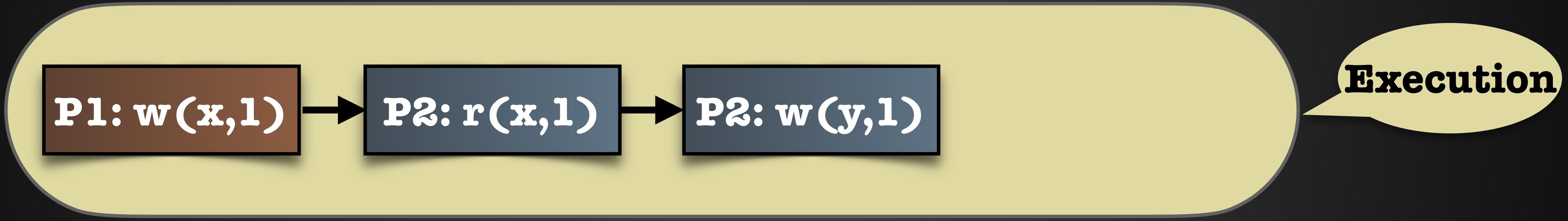
P1: w(x,1) → P2: r(x,1)

Execution

Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

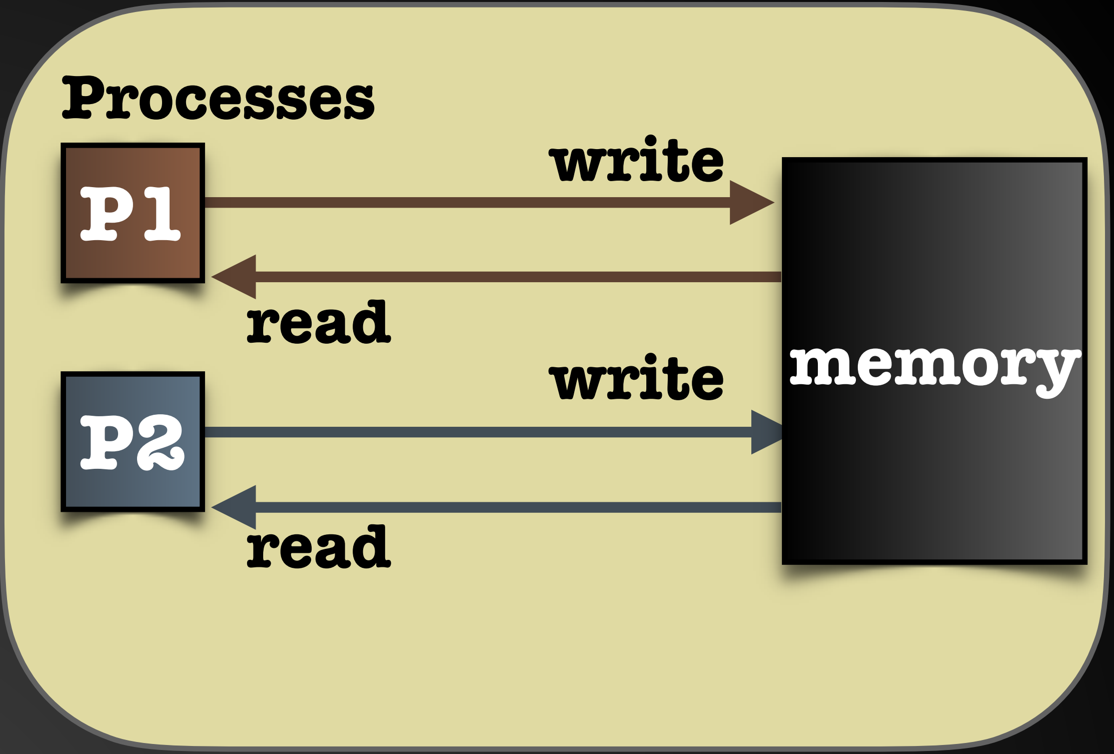



P1: w(x,1) → P2: r(x,1) → P2: w(y,1)

Execution

Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations



Processes

P1

write

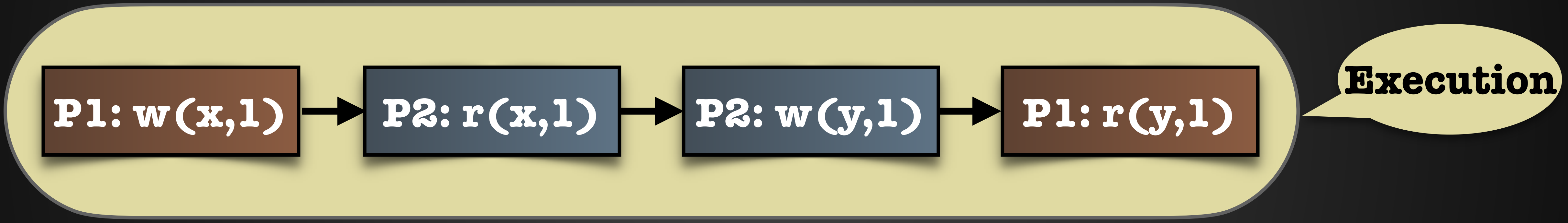
read

P2

write

read

memory



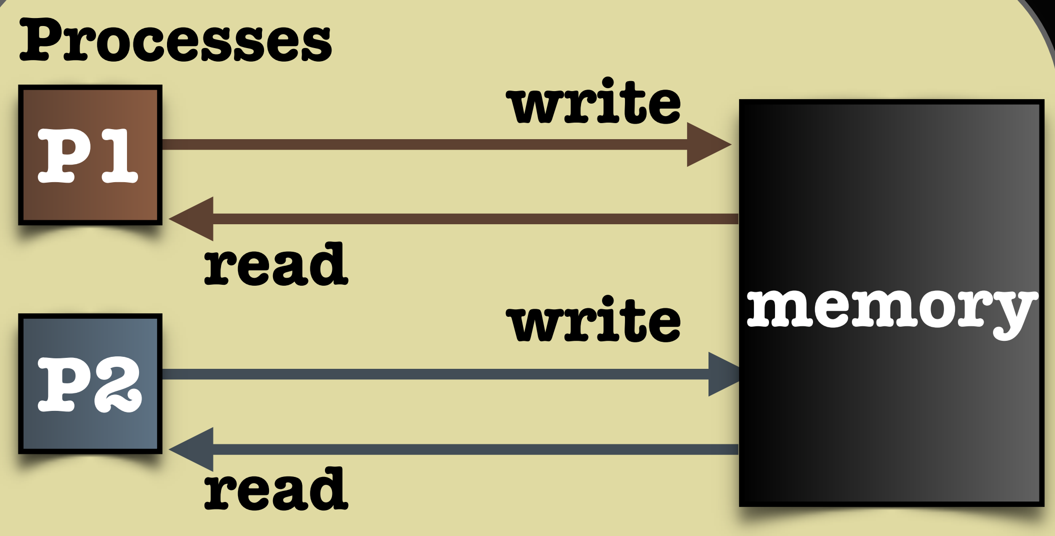
P1: w(x,1) → P2: r(x,1) → P2: w(y,1) → P1: r(y,1)

Execution

Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

+ Simple and intuitive

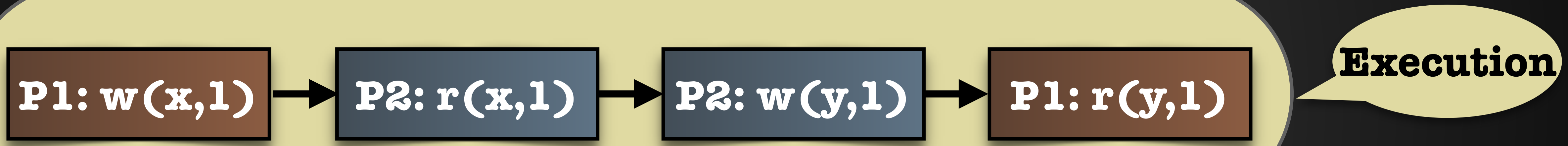


Processes

P1 write read

P2 write read

memory



P1: w(x,1) → P2: r(x,1) → P2: w(y,1) → P1: r(y,1)

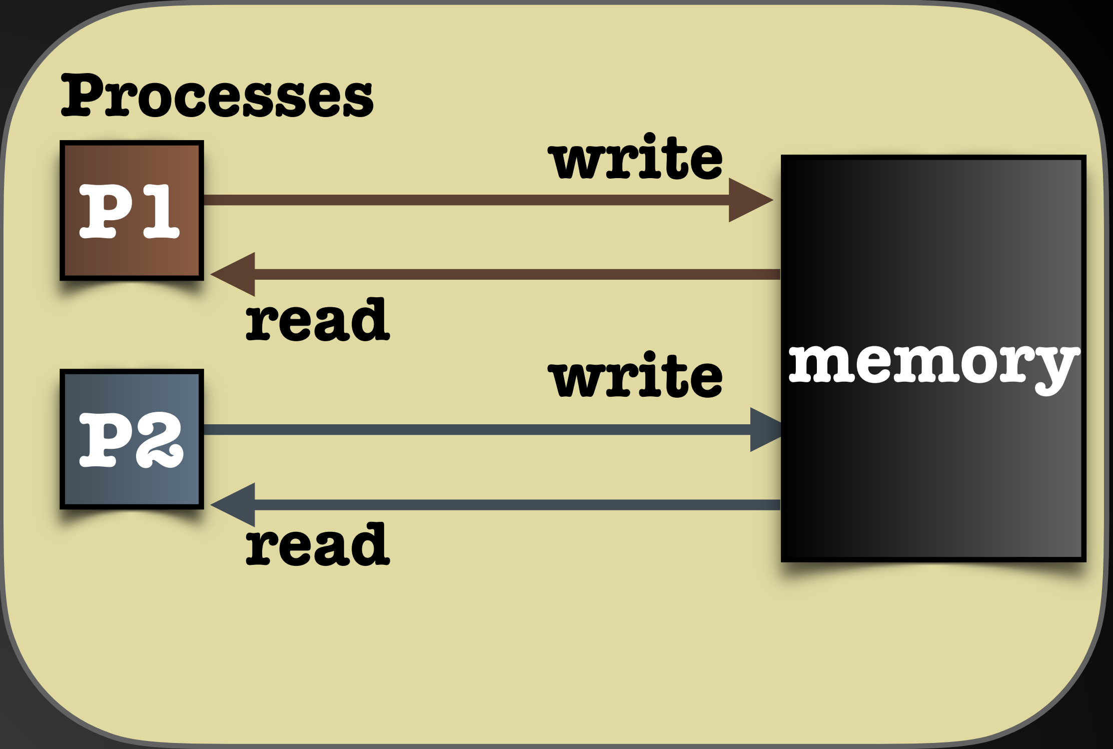
Execution

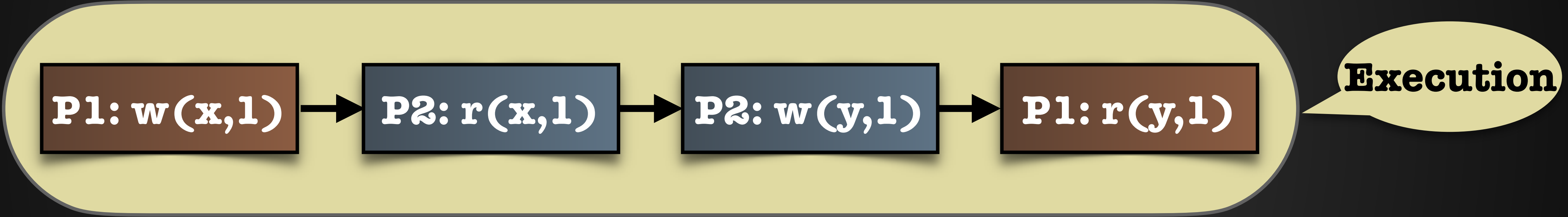
Sequential Consistency (SC)

- Shared memory
- Processes: atomic read/write
- Interleaving of the operations

+ Simple and intuitive

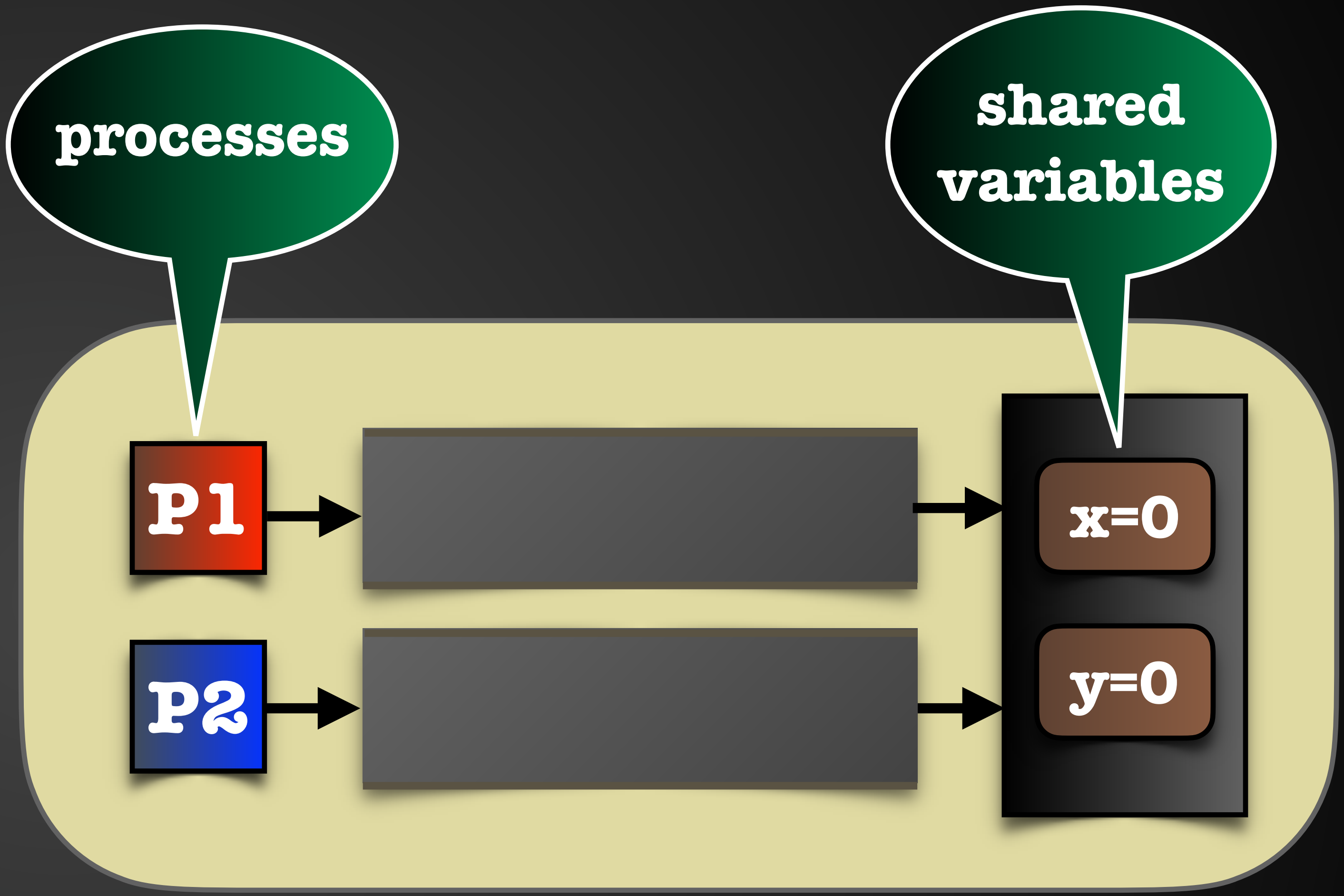
- Too strong





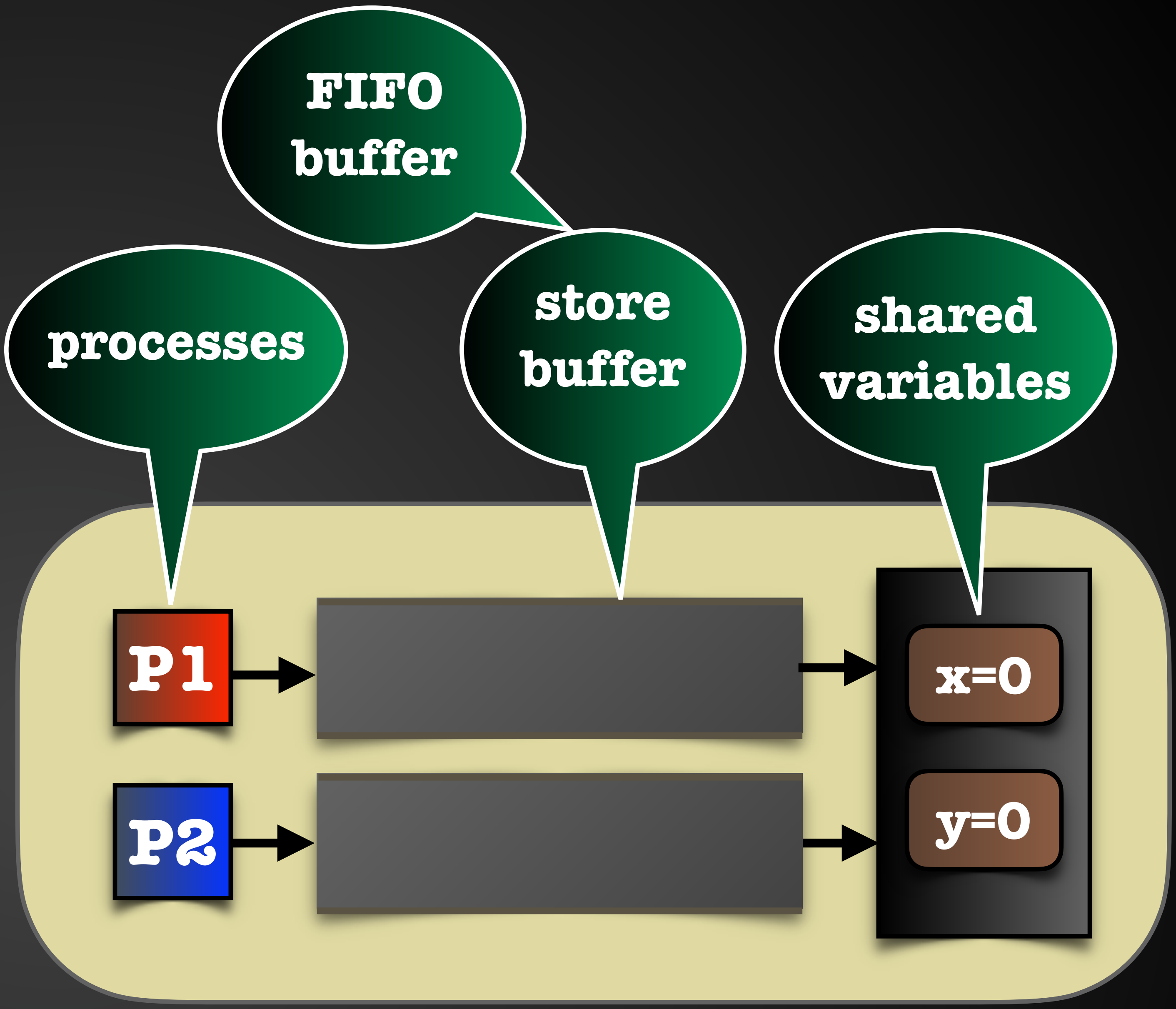
TSO - Total Store Order

- Widely used:
  - Used by Sun SPARCv9
  - Formalization of Intel x86
- Memory access optimization:
  - Write operations are slow
  - Introduce store buffers



TSO - Total Store Order

- Widely used:
  - Used by Sun SPARCv9
  - Formalization of Intel x86
- Memory access optimization:
  - Write operations are slow
  - Introduce store buffers



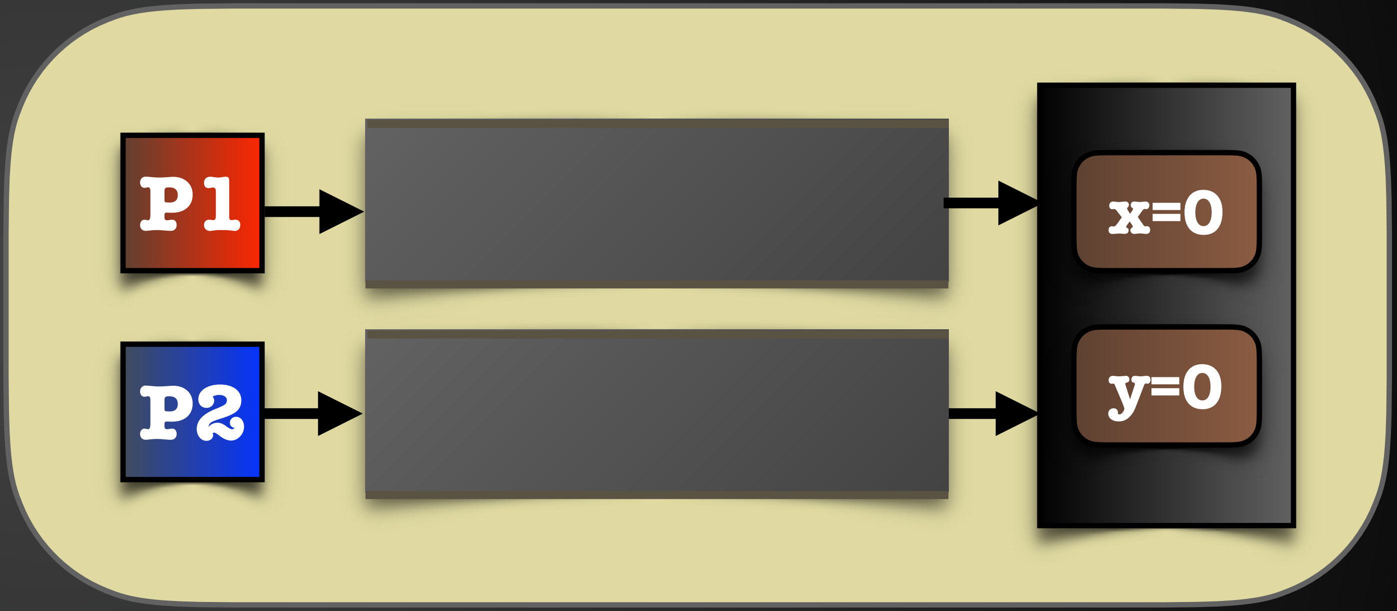
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



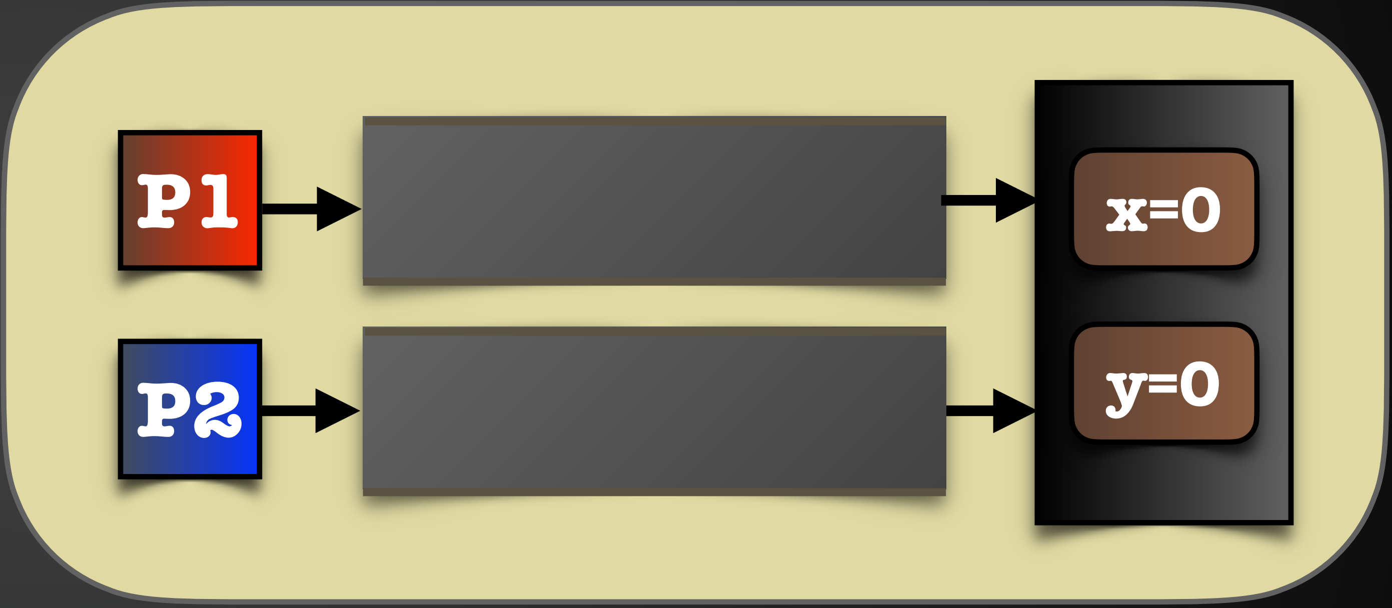
TSO - Classical Semantics

P1: write: x=1

P1: write: x = 2

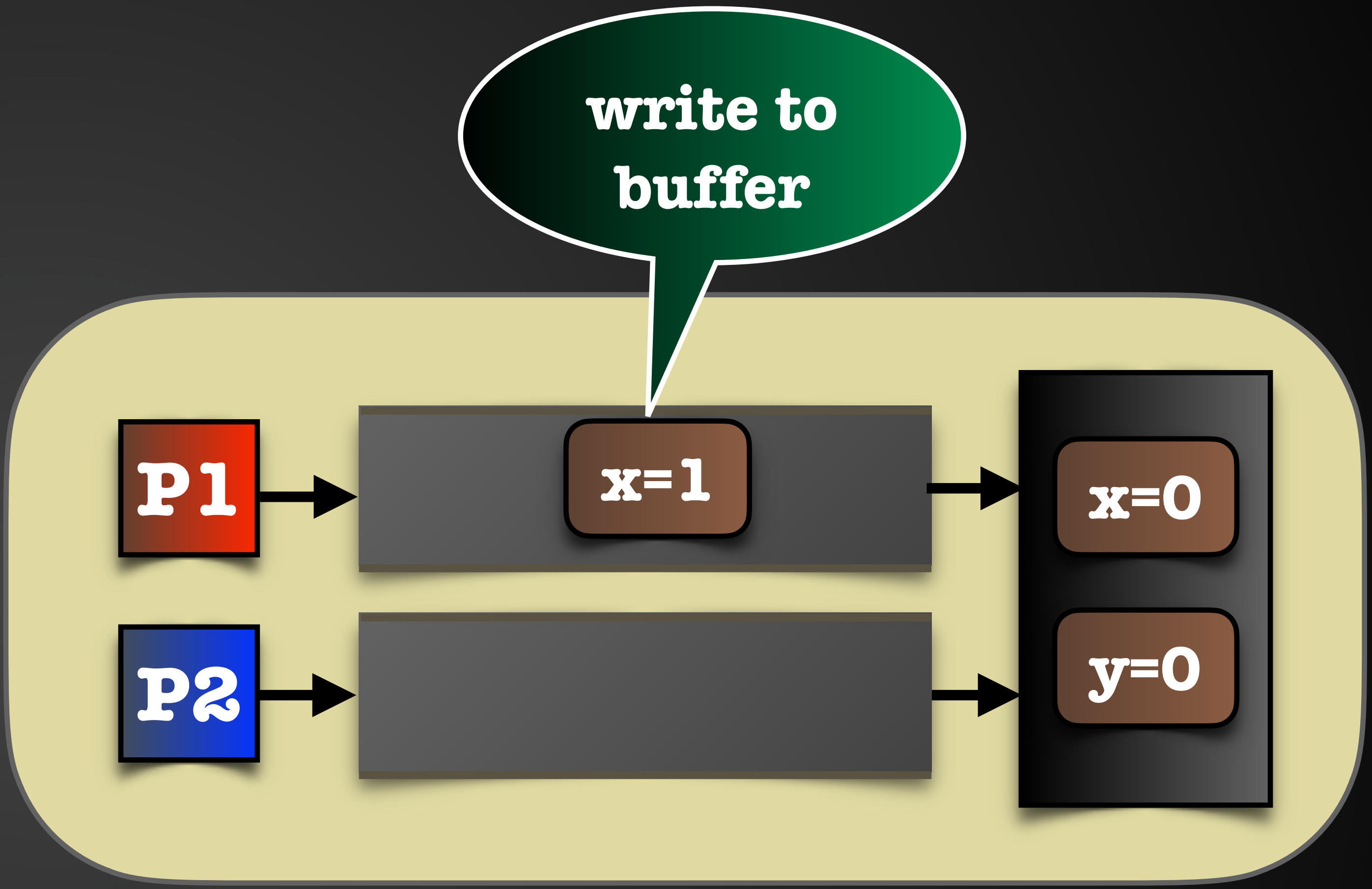
P1: read: x = 2

P1: read: y = 0



TSO - Classical Semantics

```
  P1:  write:  x = 1
▶ P1:  write:  x = 2
  P1:  read: x = 2
  P1:  read: y = 0
```



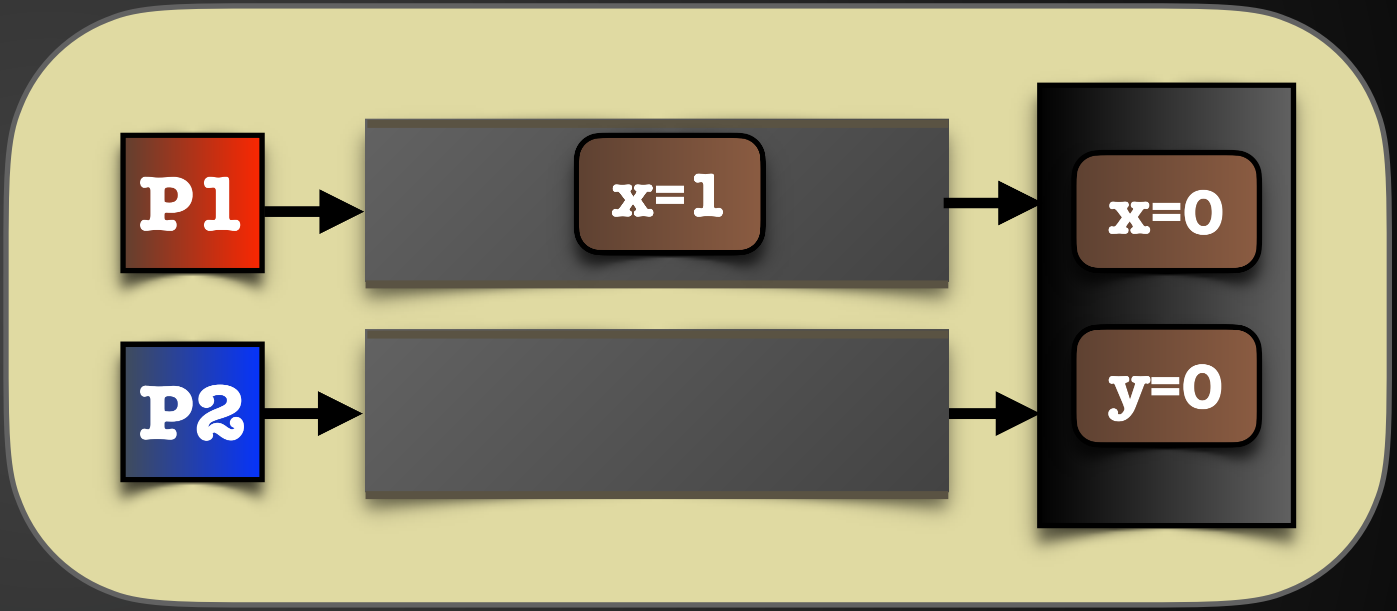
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

▶ P1: read: x = 2

P1: read: y = 0



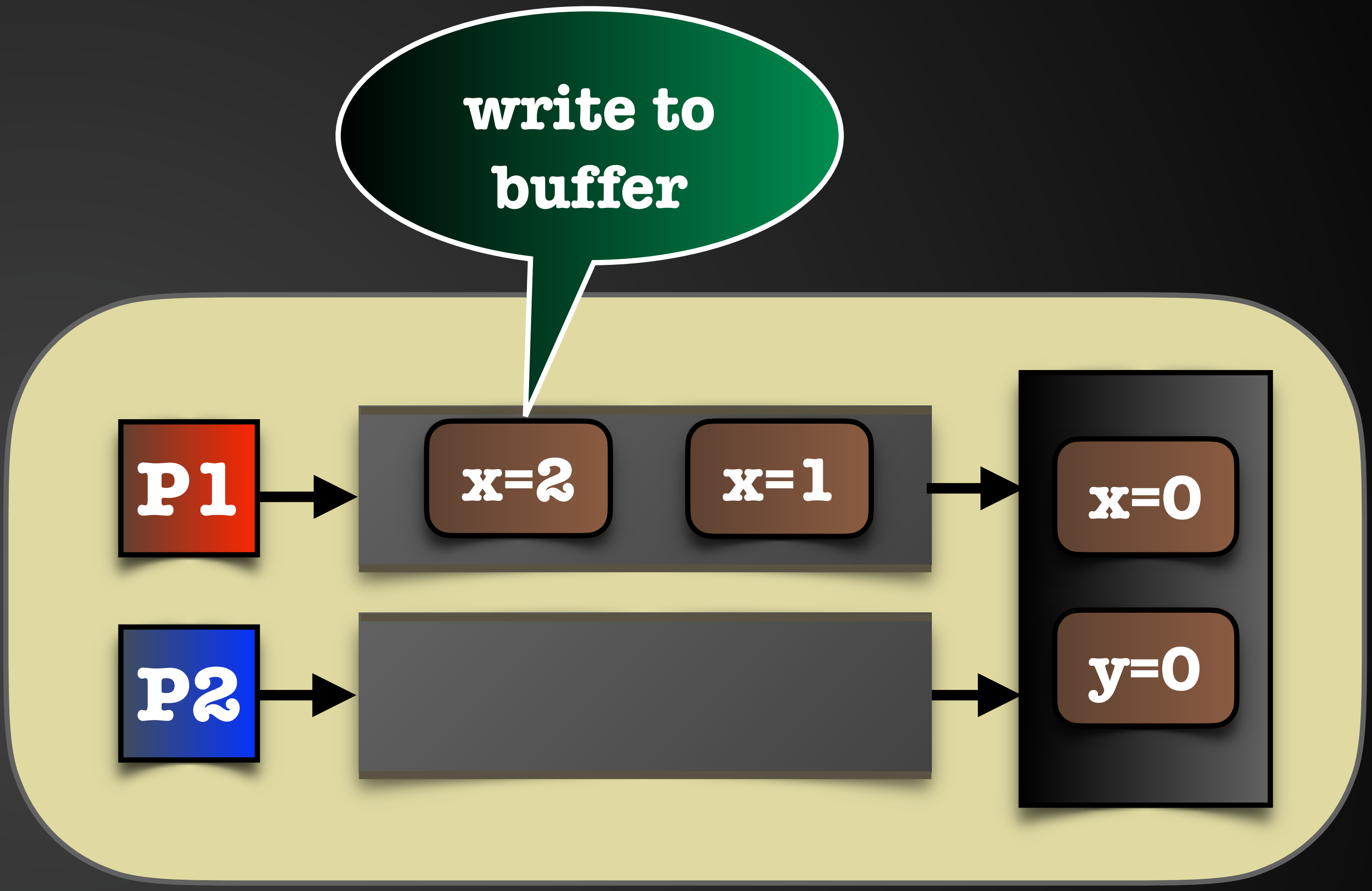
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

▶ P1: read: x = 2

P1: read: y = 0



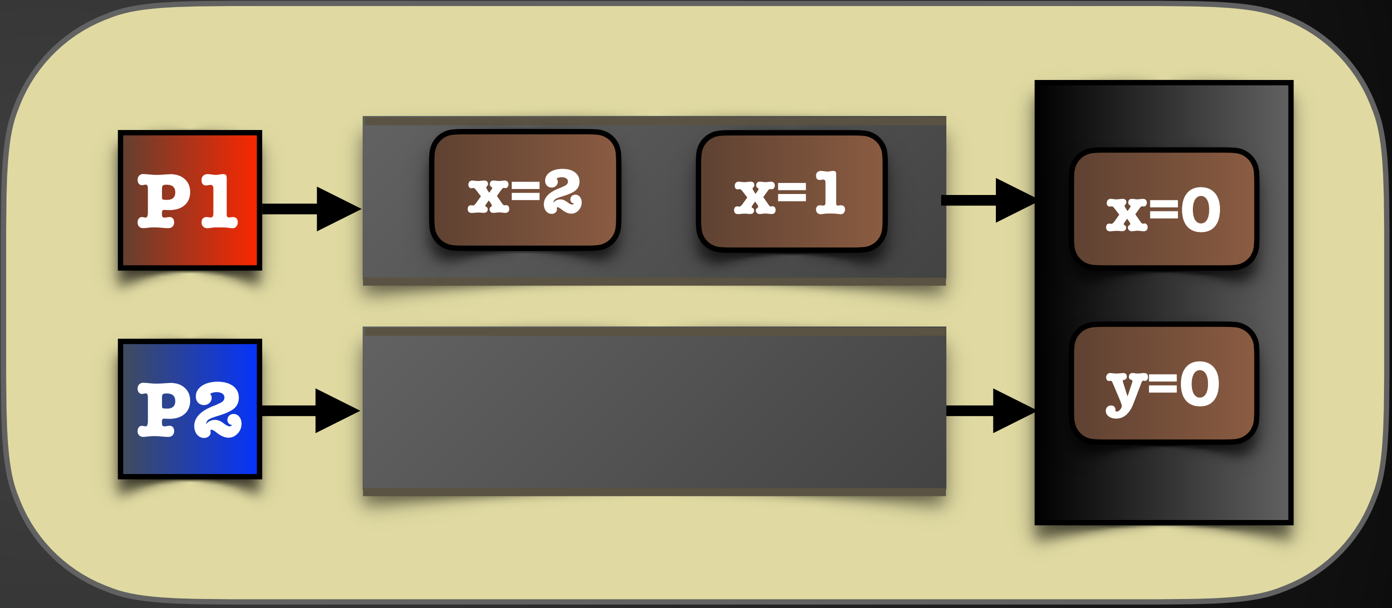
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

▶ P1: read: y = 0



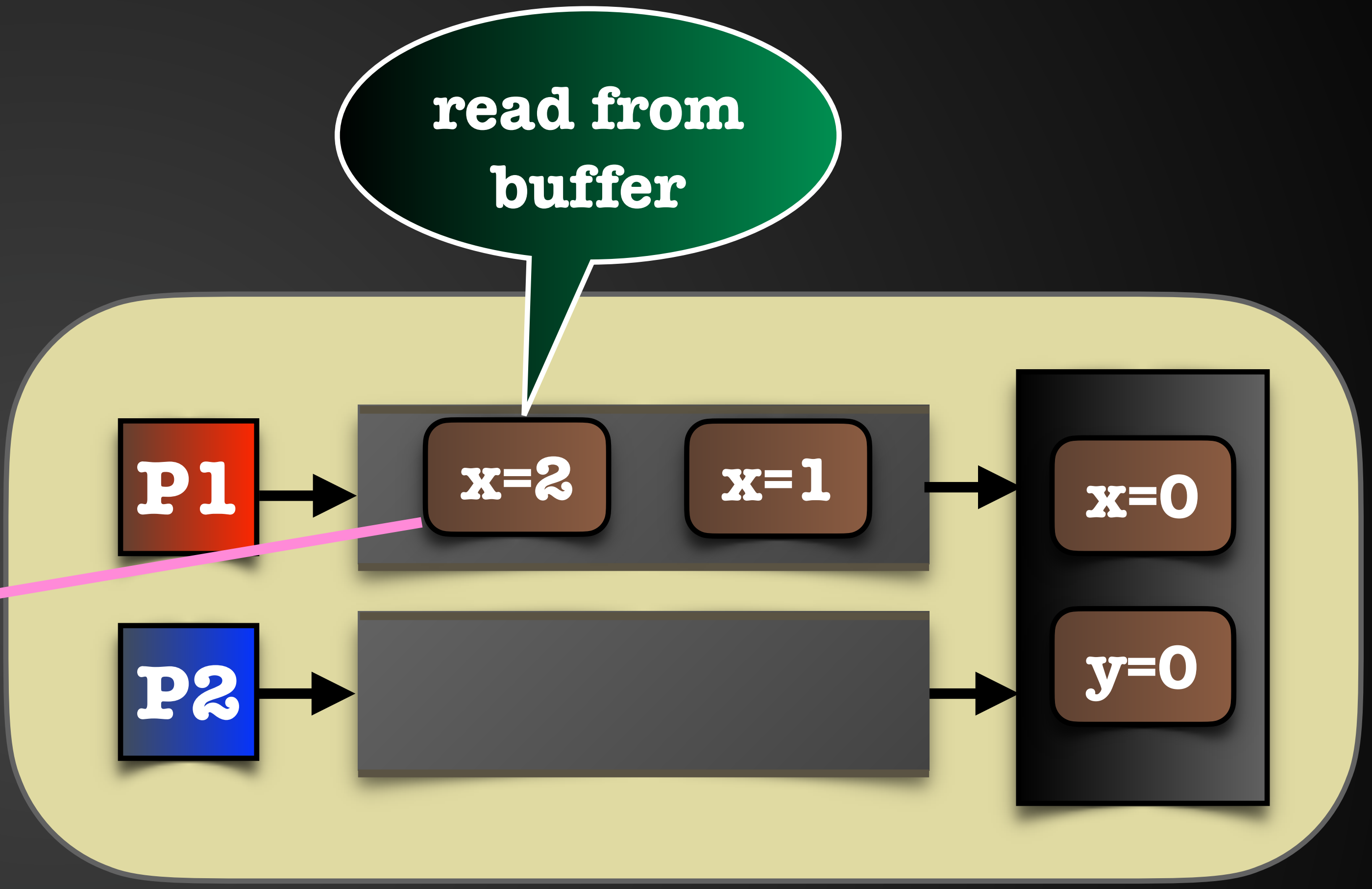
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



read from buffer

P1

x=2

x=1

P2

x=0

y=0

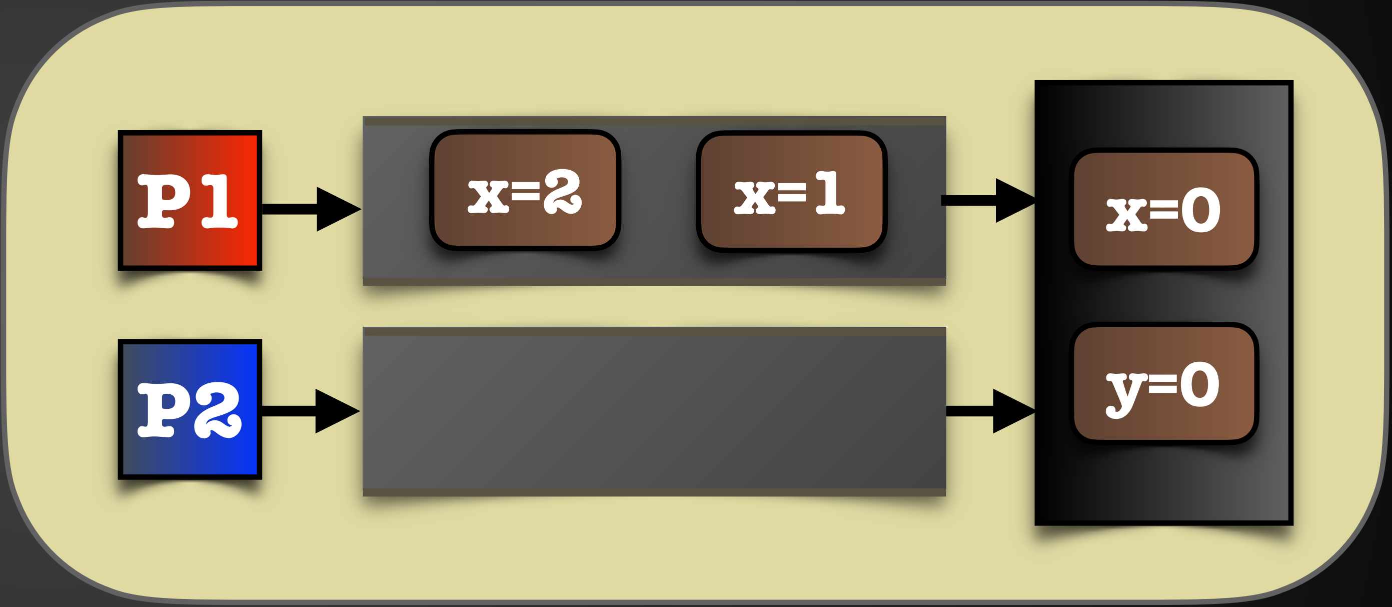
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

▶ P1: read: y = 0



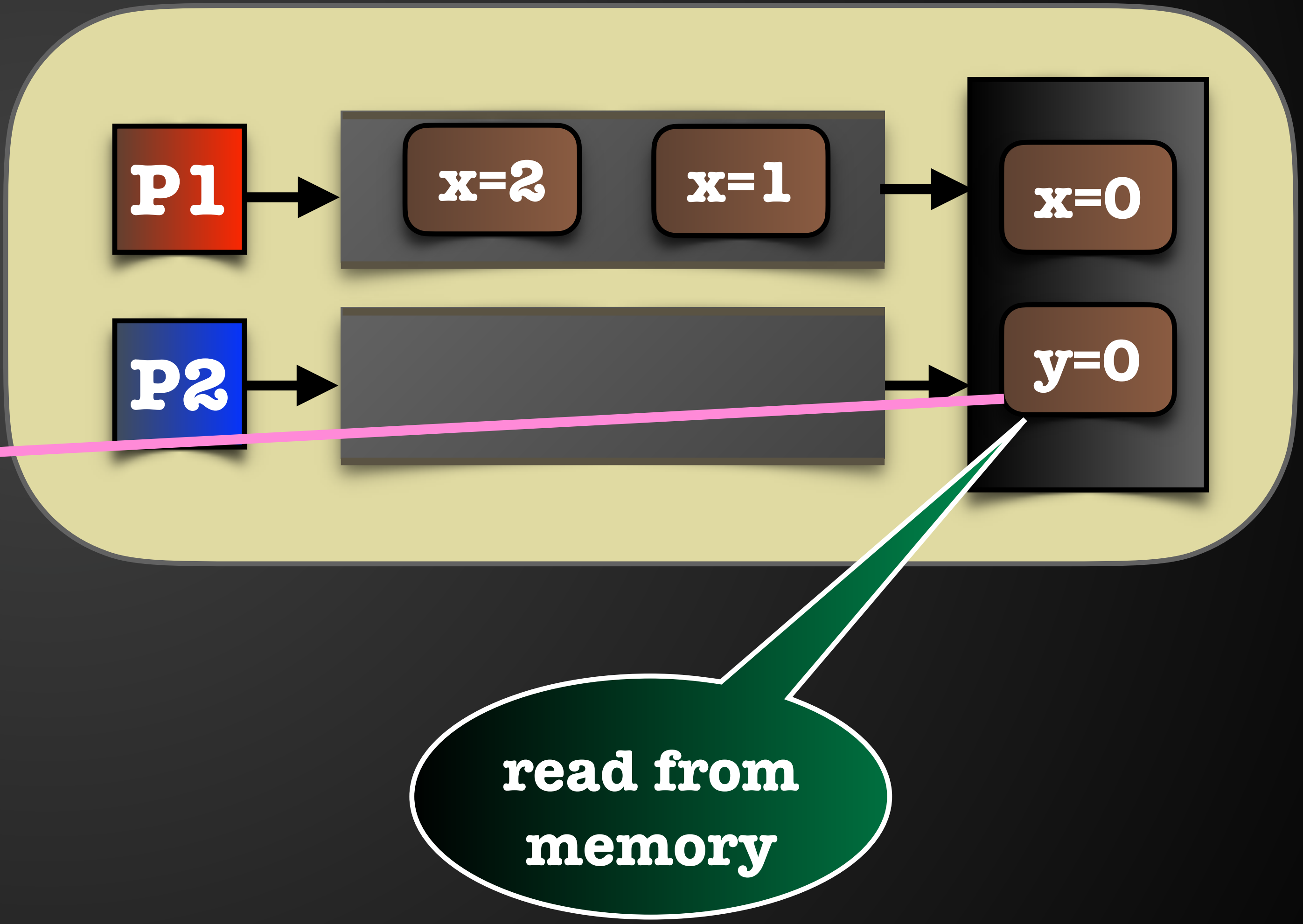
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



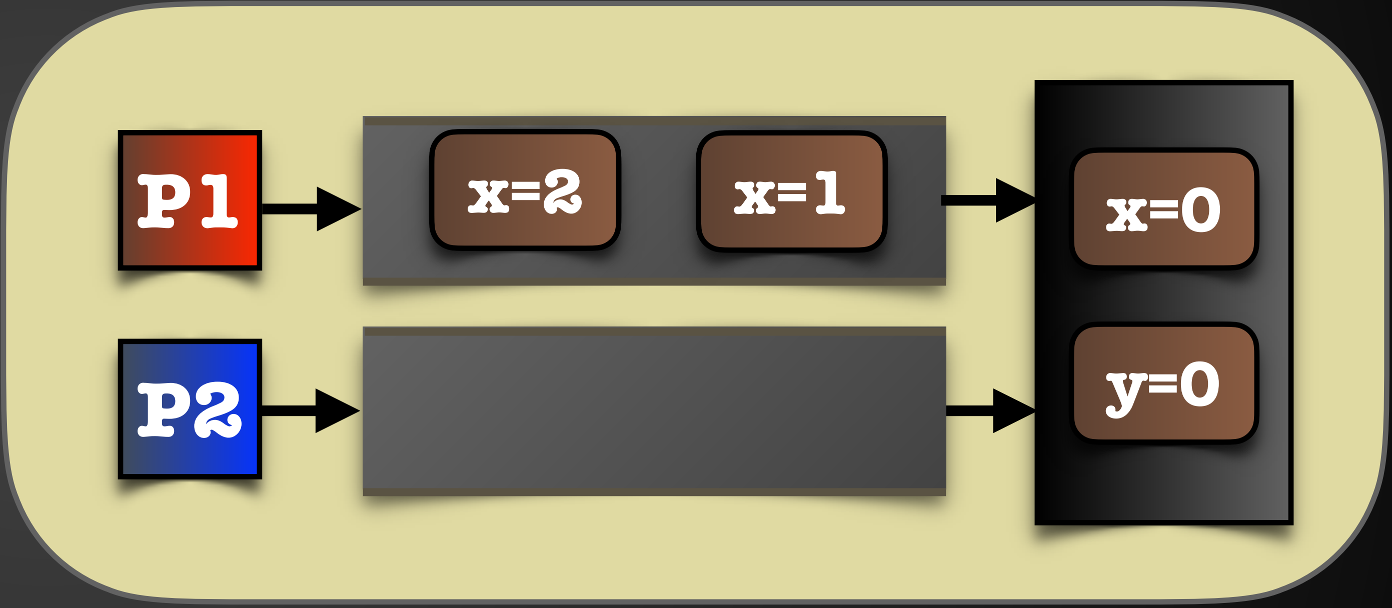
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

▶ P1: read: y = 0



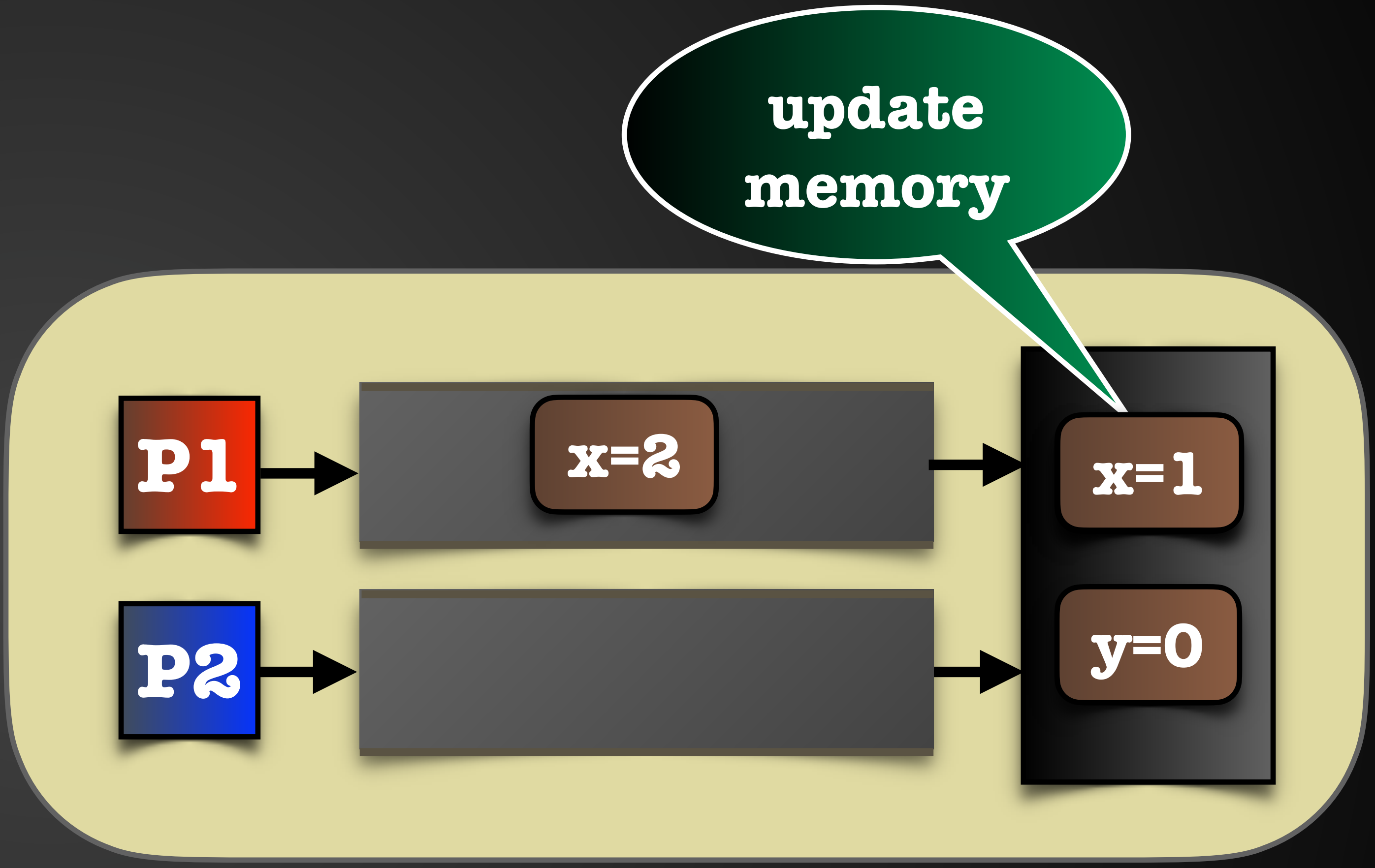
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



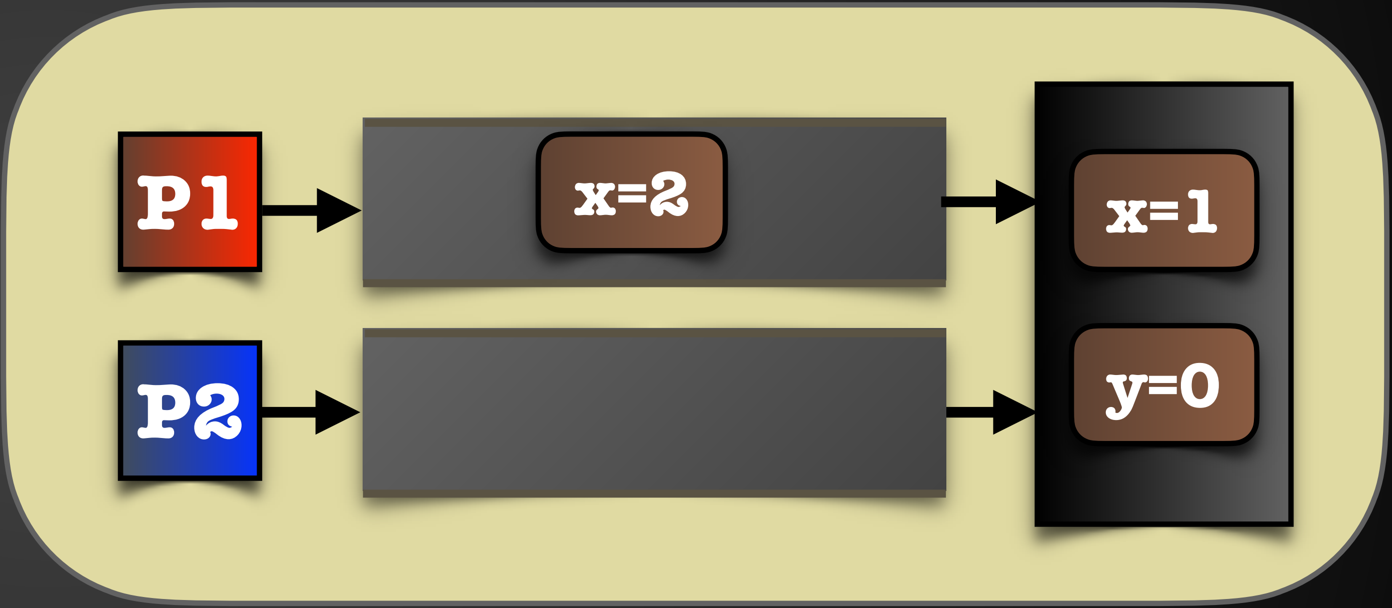
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



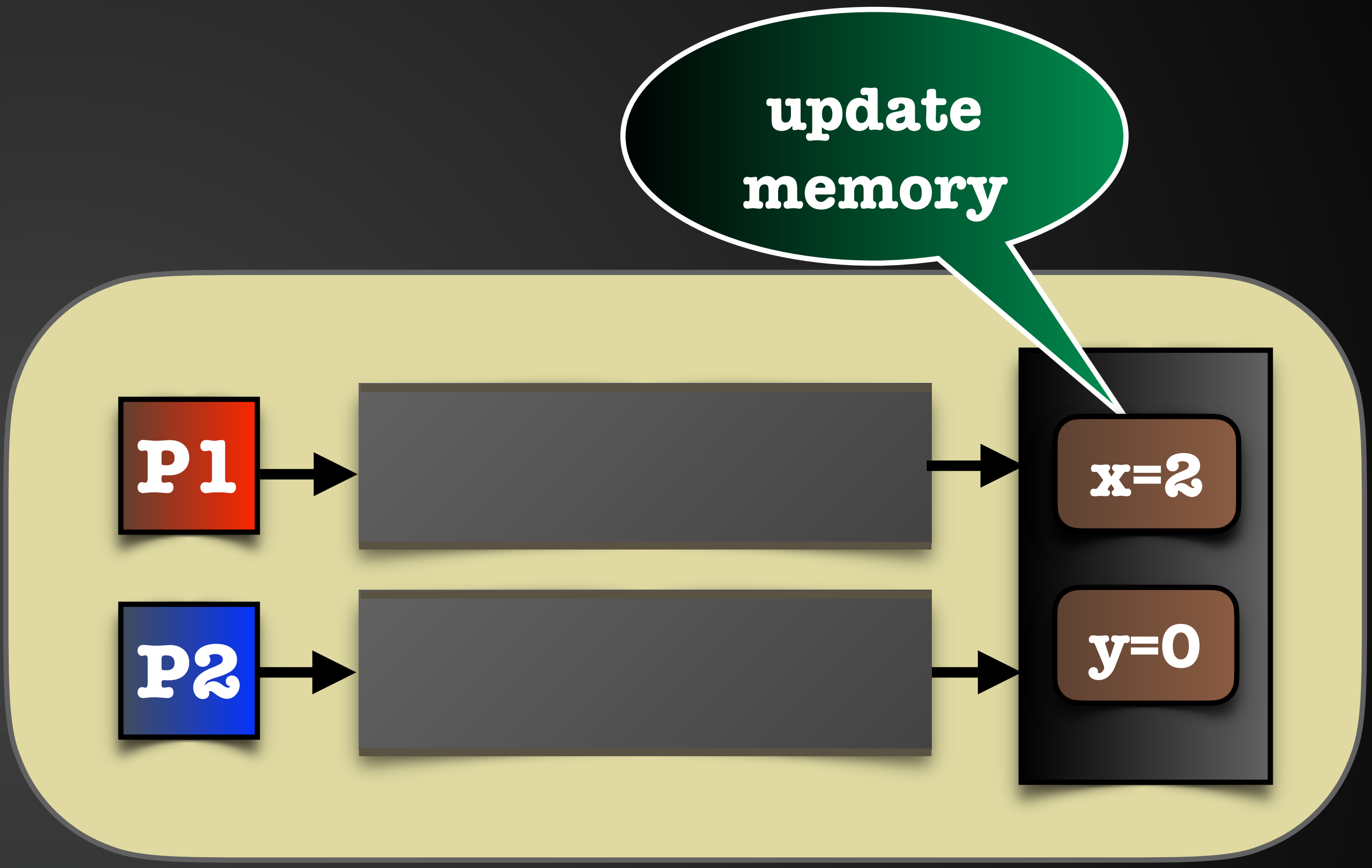
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

▶ P1: read: y = 0



TSO - Classical Semantics

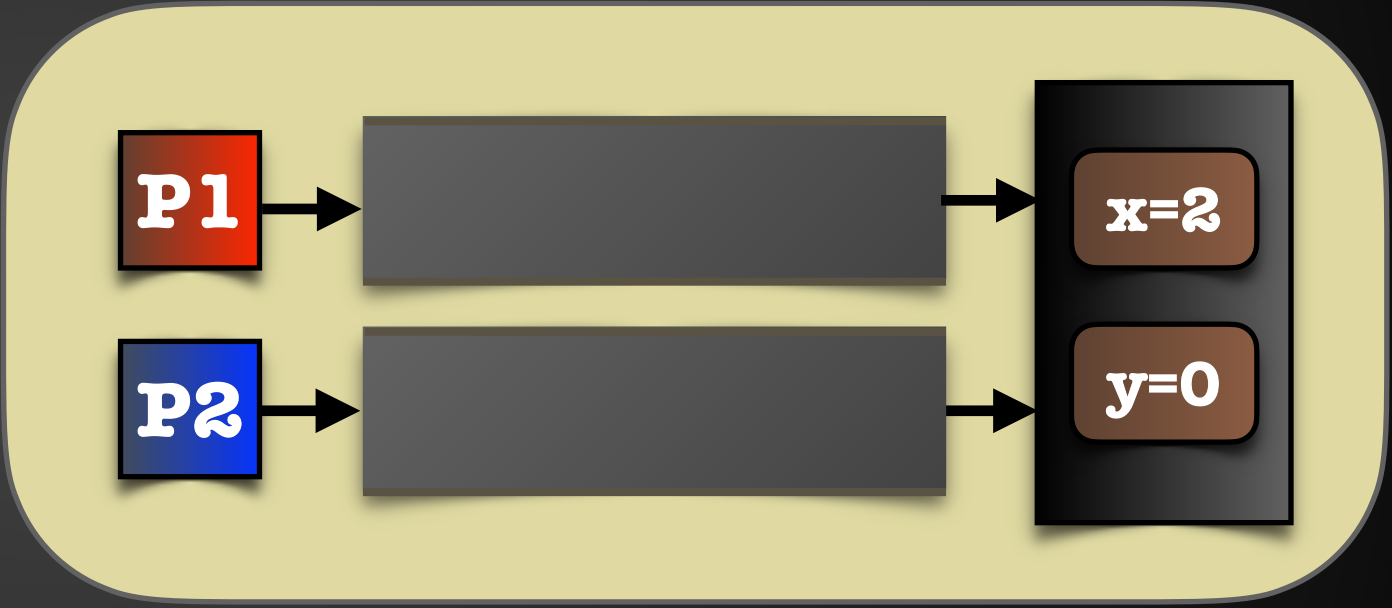
P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0

- write to buffer
- read from buffer
- read from memory
- update memory



TSO - Classical Semantics

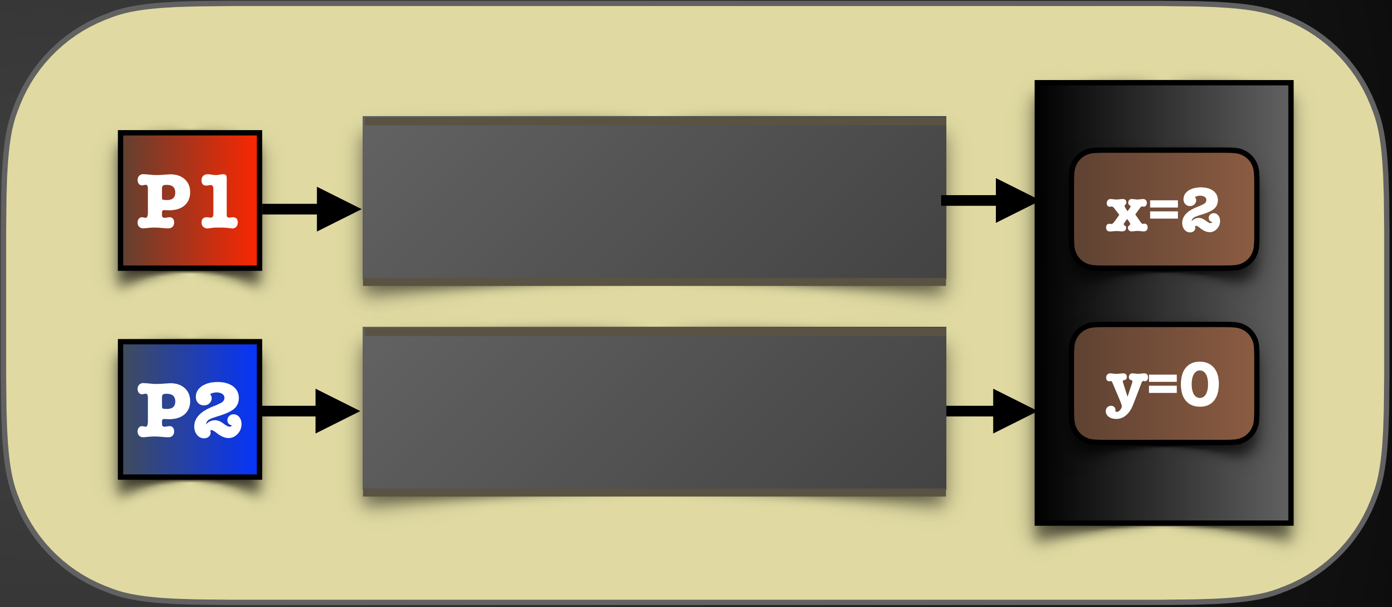
P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0

- write to buffer
- read from buffer
- read from memory
- update memory



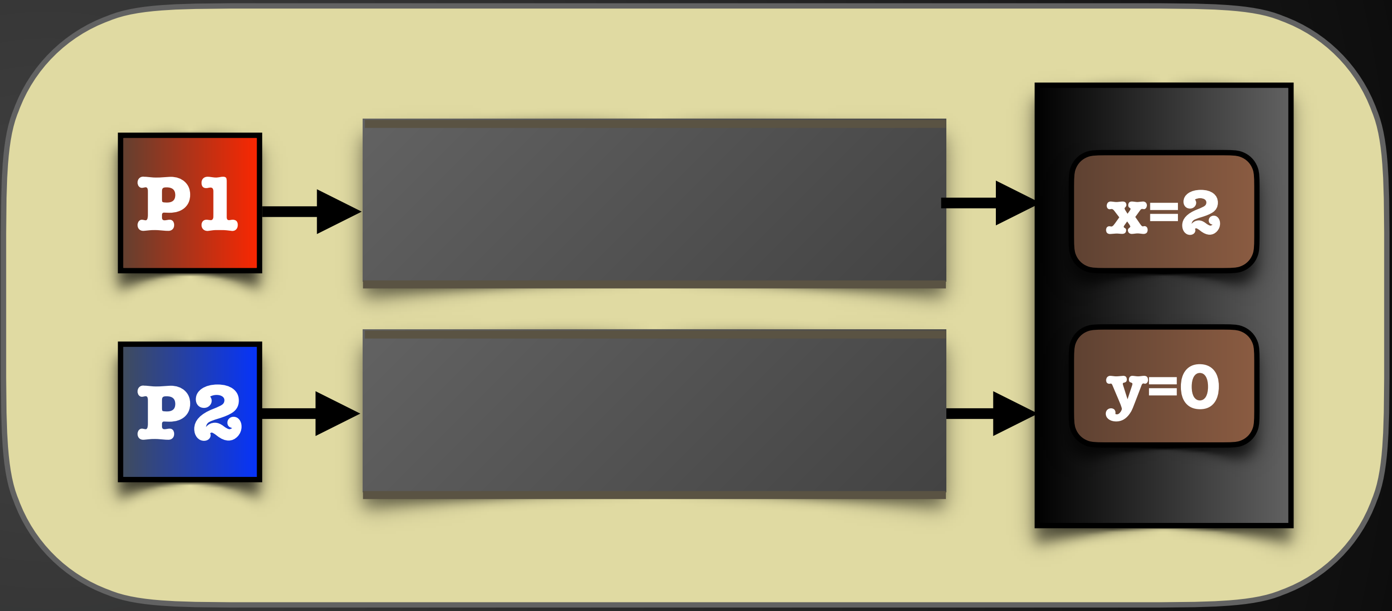
TSO - Classical Semantics

P1: write: x = 1

P1: write: x = 2

P1: read: x = 2

P1: read: y = 0



- write to buffer
- read from buffer
- read from memory
- update memory

TSO

- Extra behaviors
- Potentially bad behaviors

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

P1

P2

x = 0

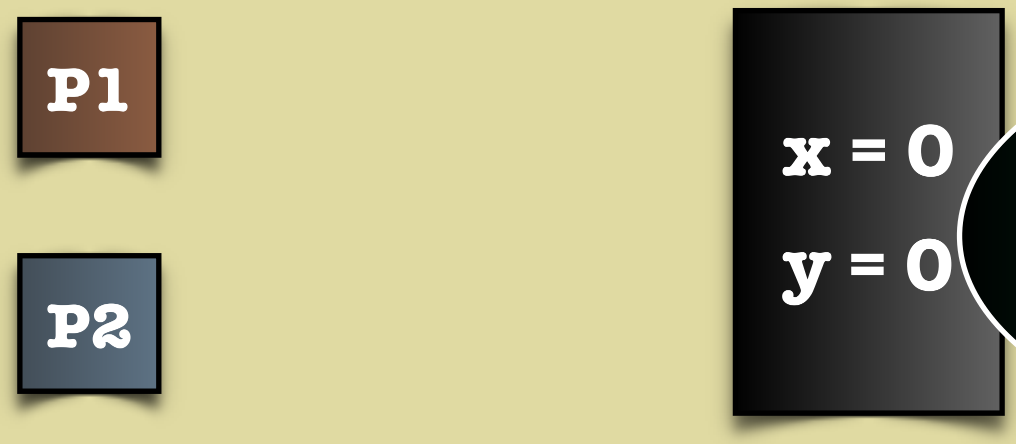
y = 0

Sequential Consistency = Interleaving

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



P1

P2

x = 0

y = 0

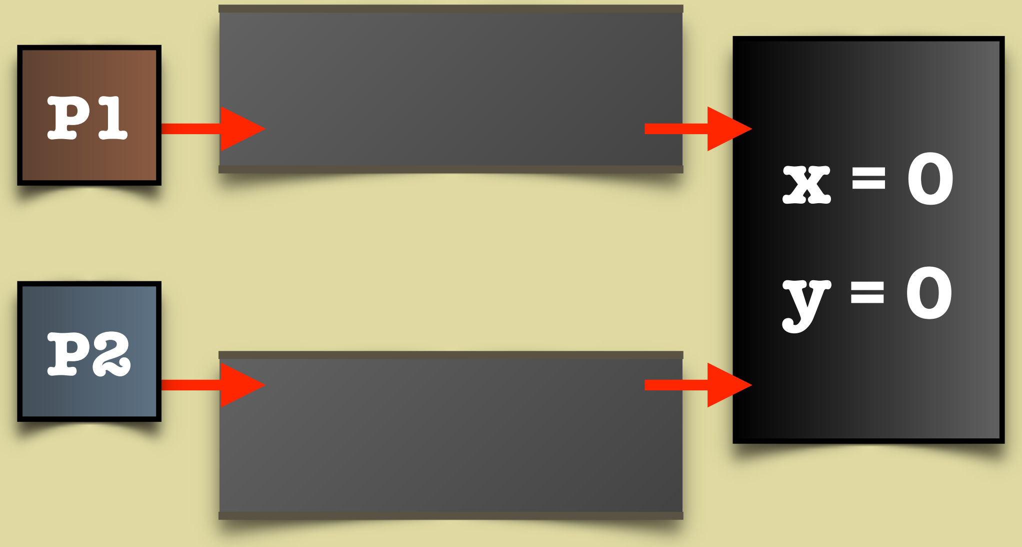
At most one process at its CS at any time

Sequential Consistency = Interleaving

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

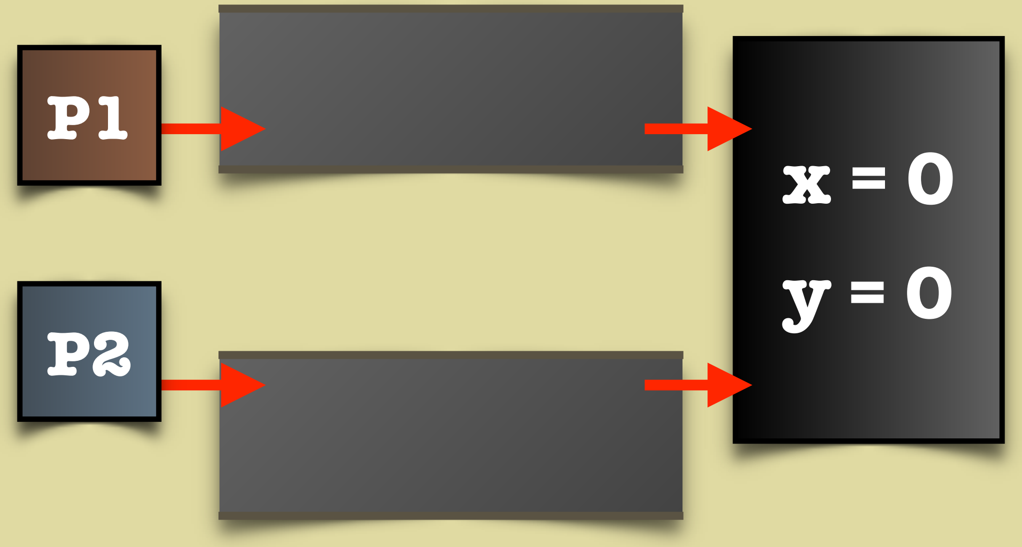


TSO

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

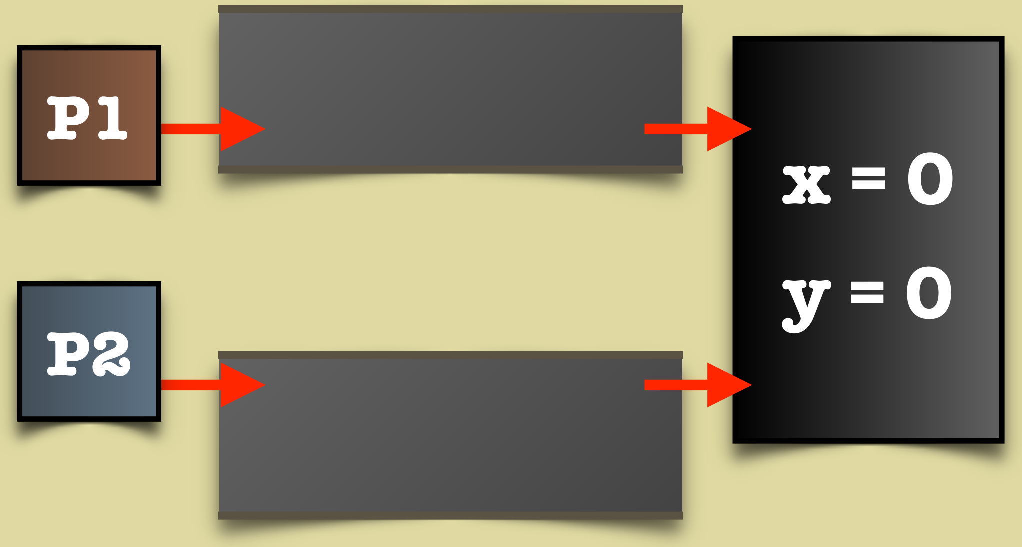


TSO

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

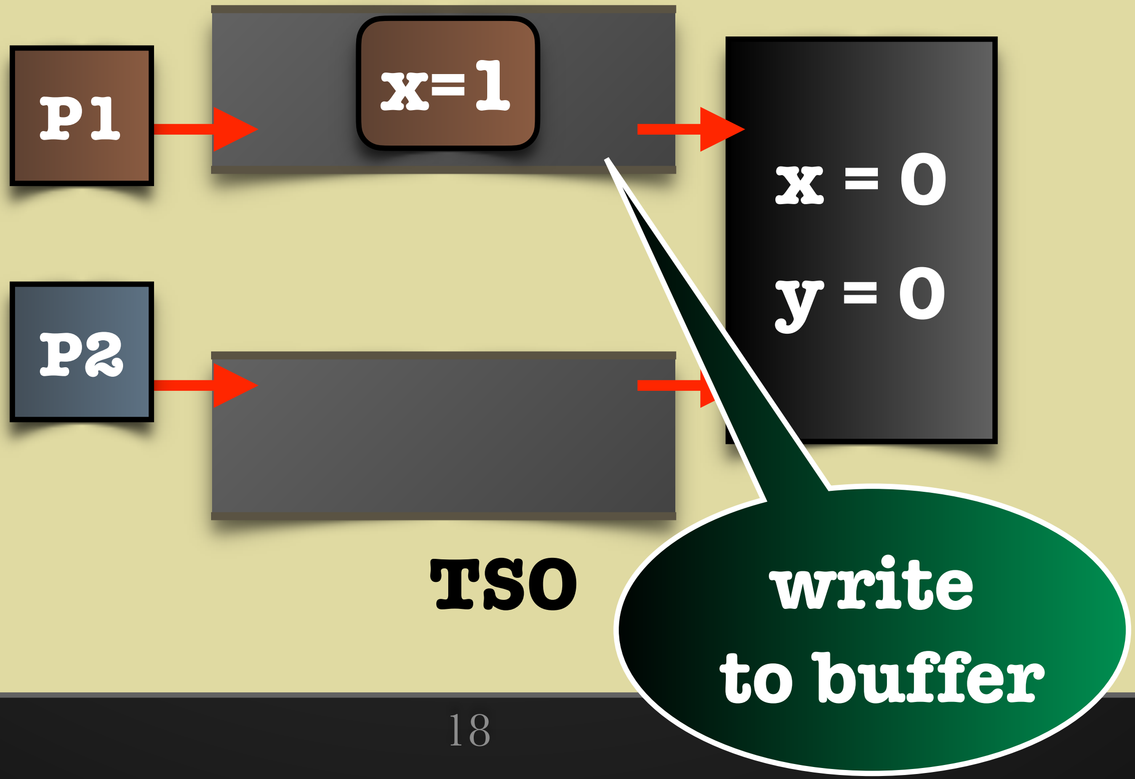


TSO

Dekker Protocol

Initially: x = y = 0

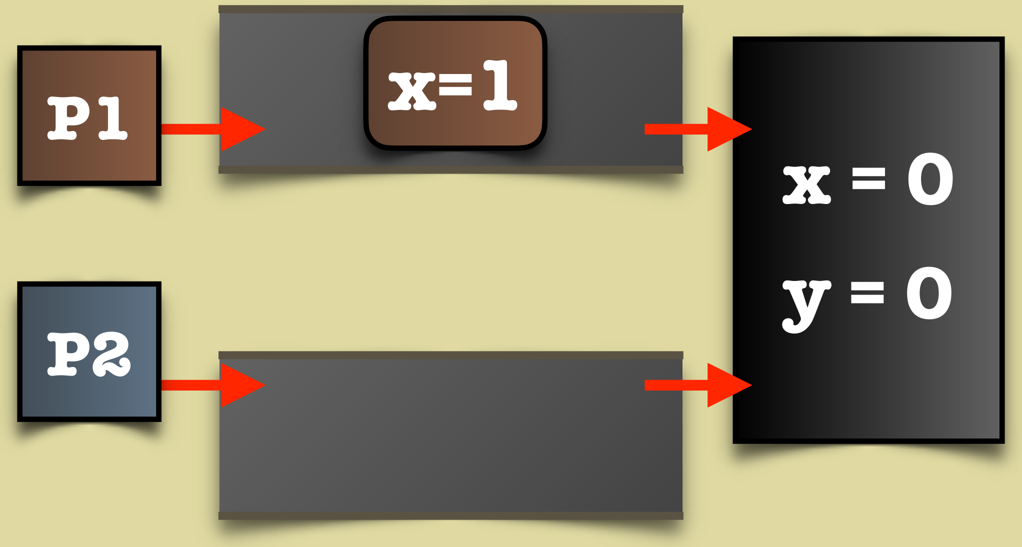
P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



Dekker Protocol

Initially: x = y = 0

P1	P2
	▶
write: x = 1	write: y = 1
▶	
read: y = 0	read: x = 0
critical section	critical section



TSO

Dekker Protocol

Initially: x = y = 0

P1

write: x = 1

read: y = 0

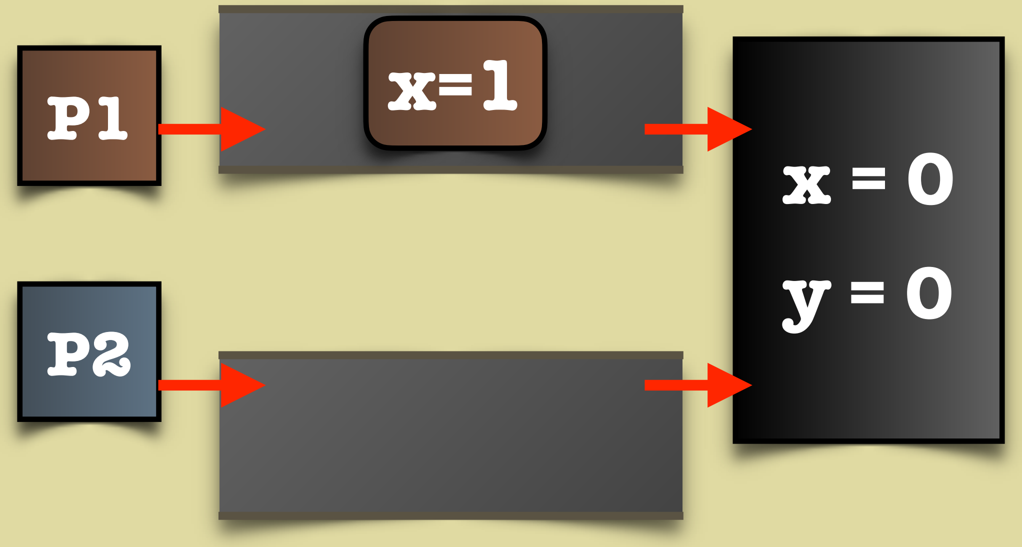
▶ critical section

P2

▶ write: y = 1

read: x = 0

critical section

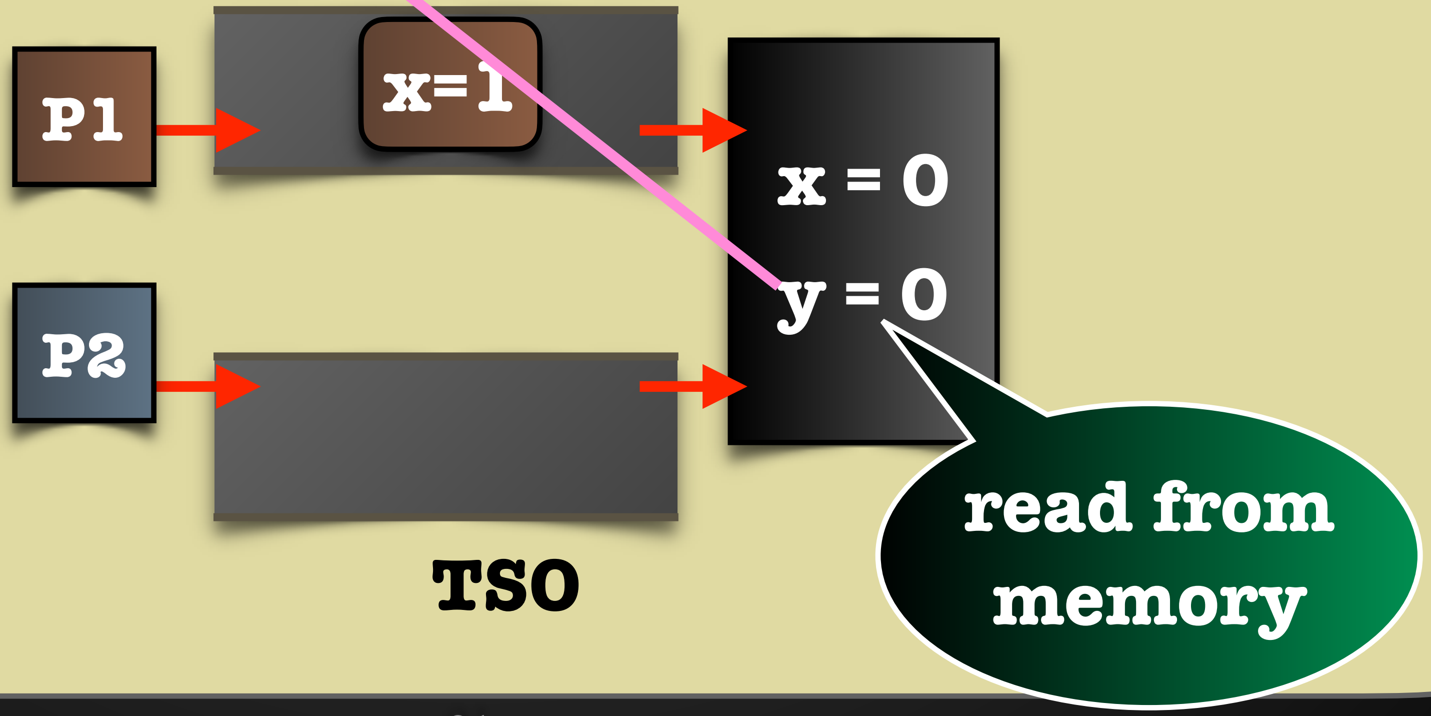


TSO

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

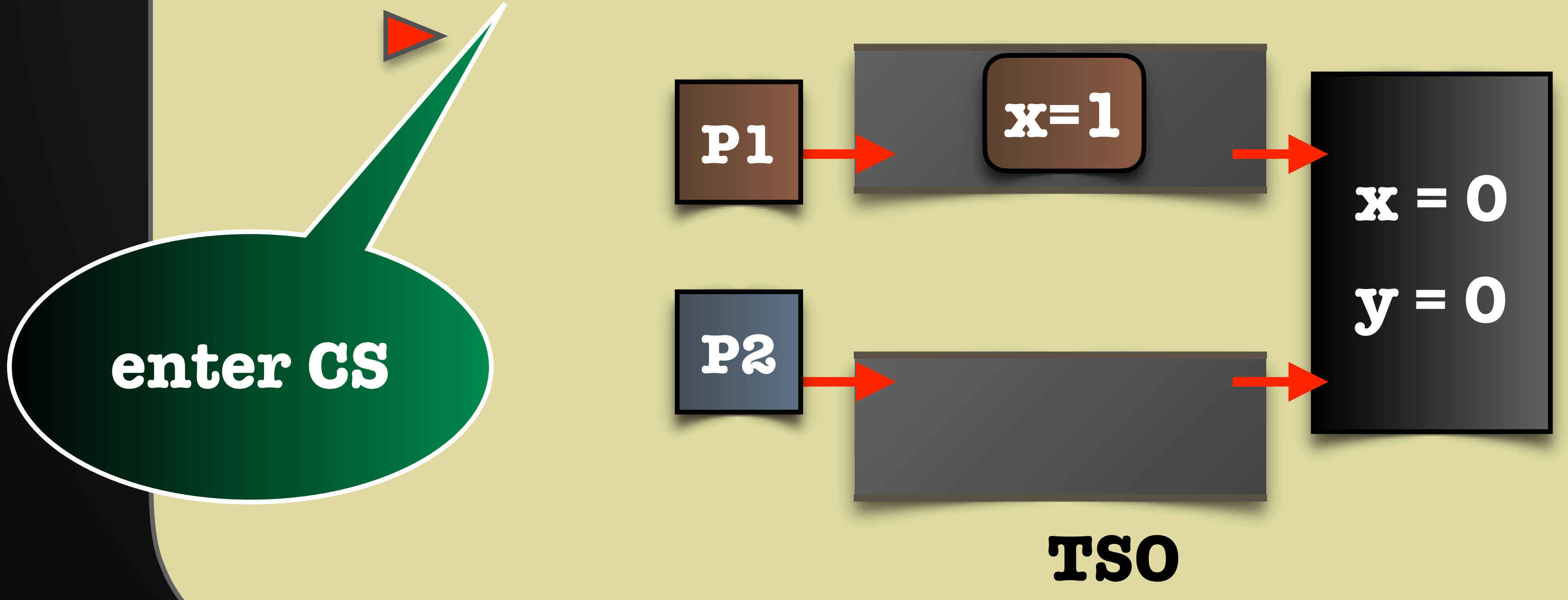


read from memory

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



Dekker Protocol

Initially: x = y = 0

P1

write: x = 1

read: y = 0

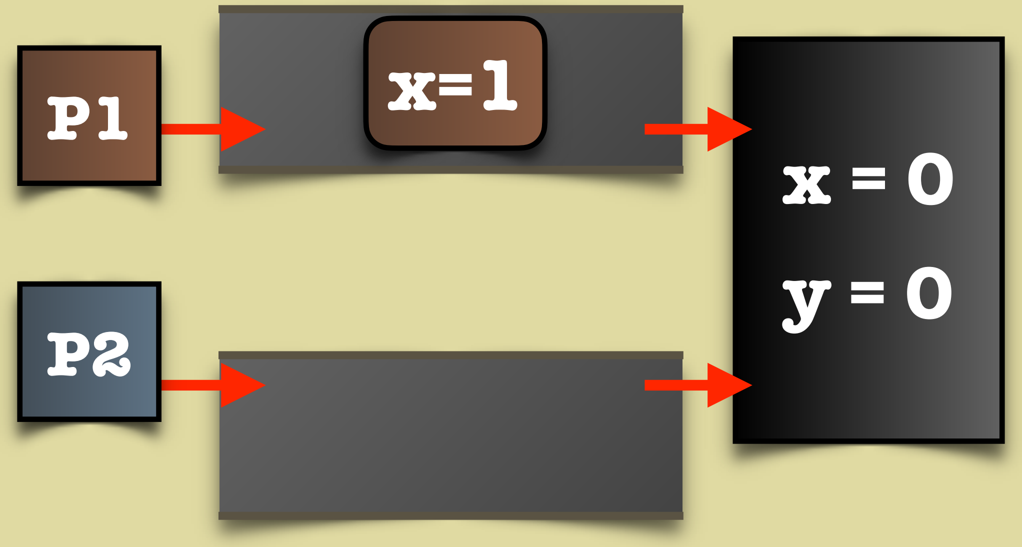
critical section

P2

write: y = 1

read: x = 0

critical section

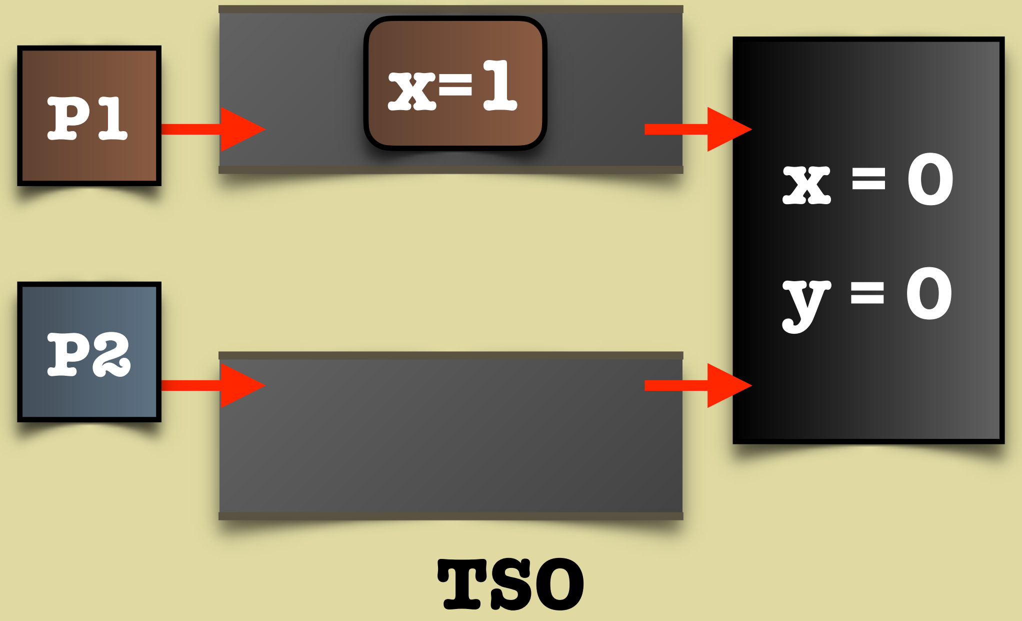


TSO

Dekker Protocol

Initially: x = y = 0

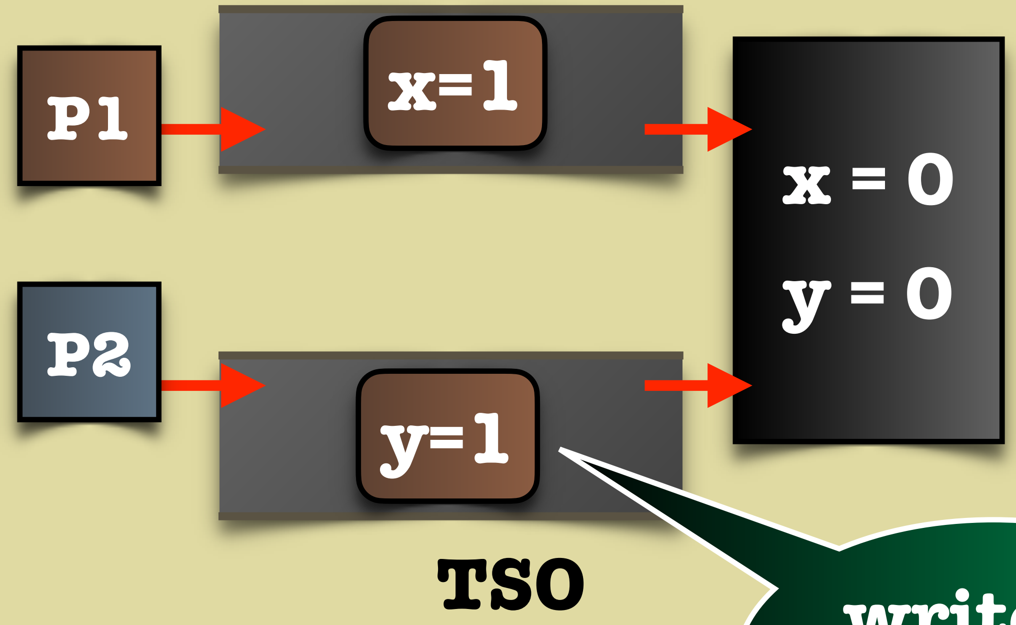
P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



write to buffer

Dekker Protocol

Initially: x = y = 0

P1

write: x = 1

read: y = 0

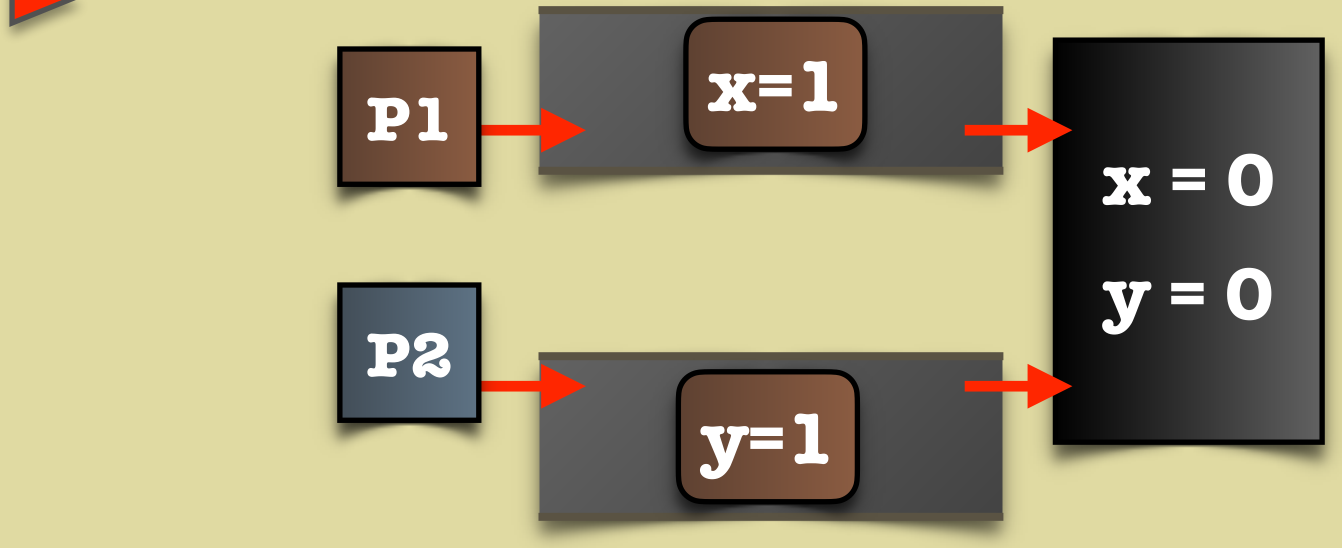
critical section

P2

write: y = 1

read: x = 0

critical section

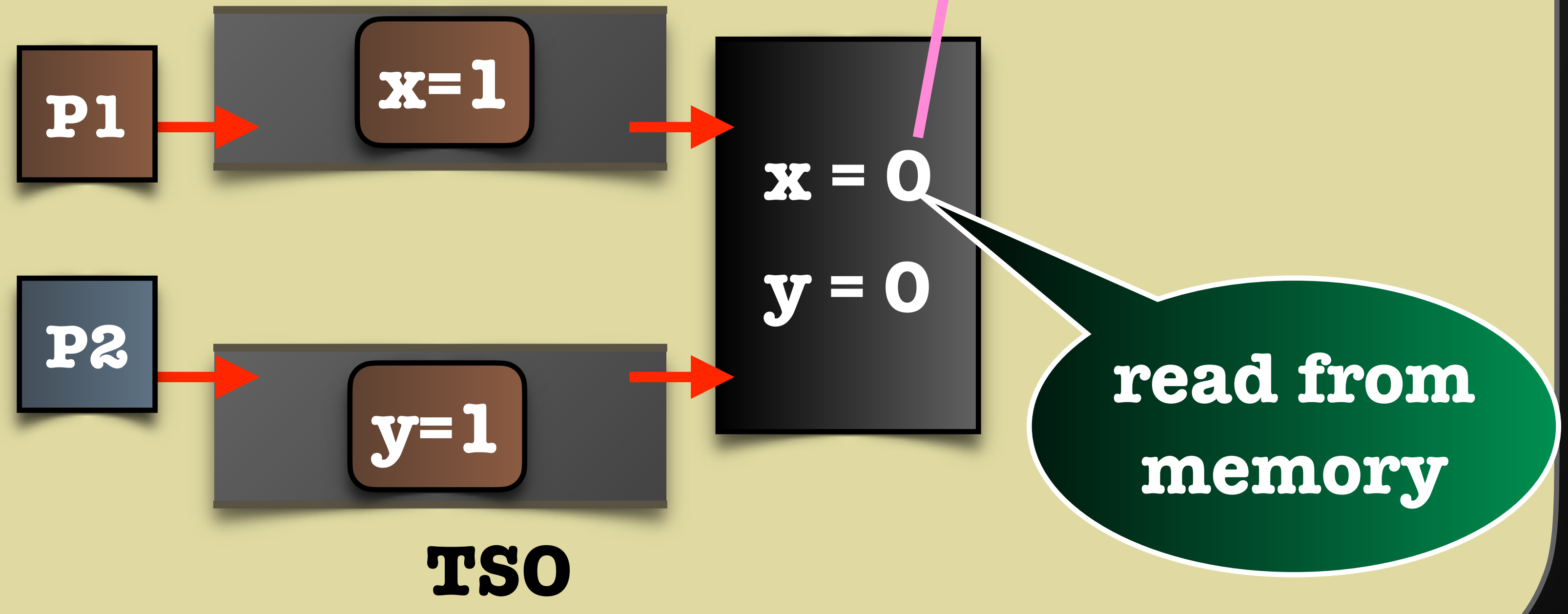


TSO

Dekker Protocol

Initially: x = y = 0

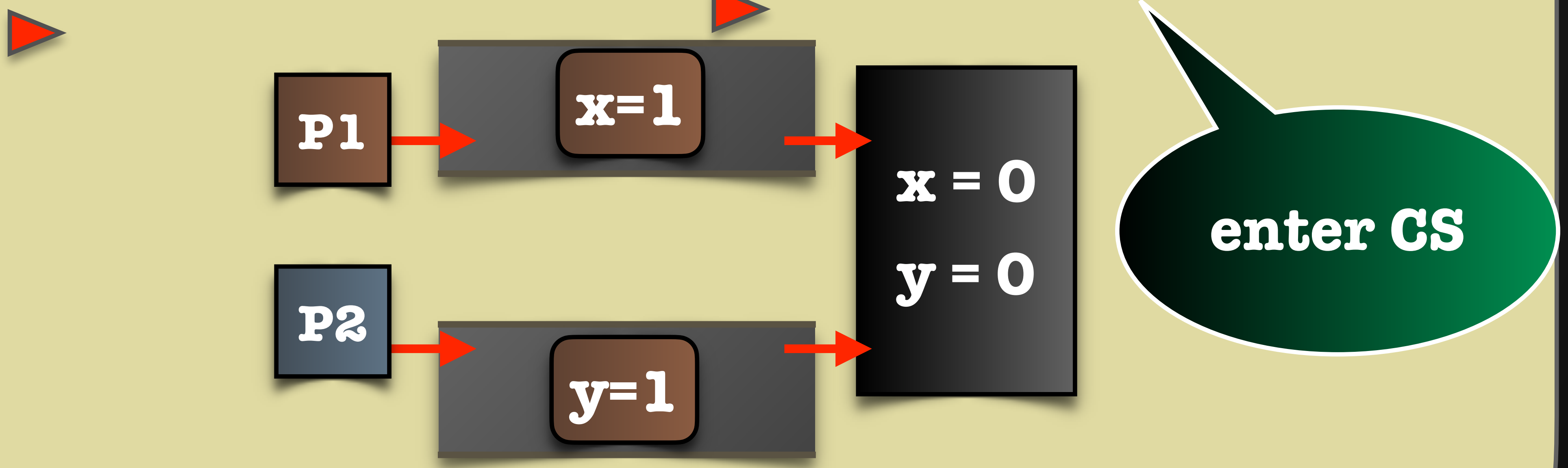
P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



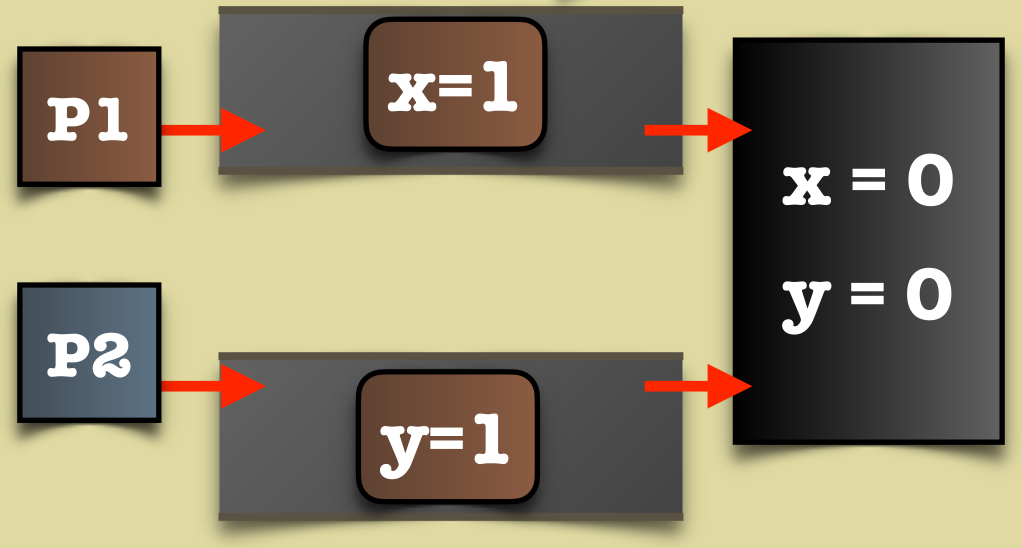
TSO

Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section

2 processes in CS
at the same time

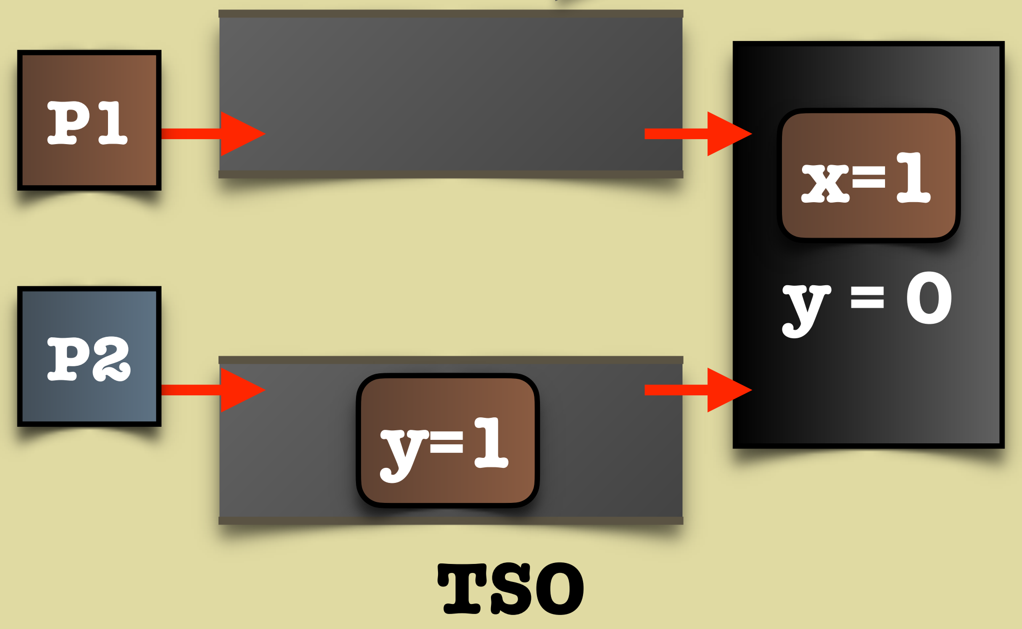


TSO

Dekker Protocol

Initially: x = y = 0

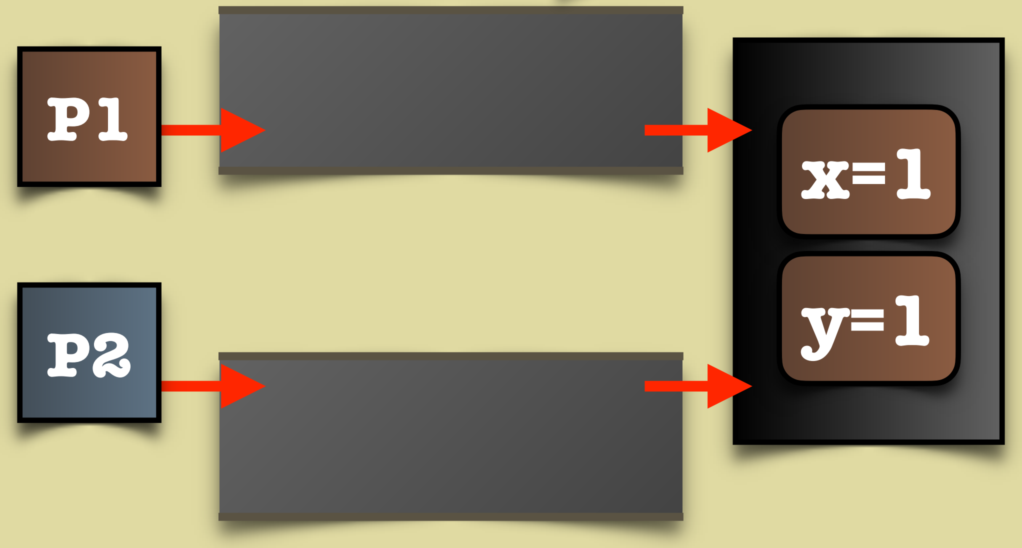
P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



Dekker Protocol

Initially: x = y = 0

P1	P2
write: x = 1	write: y = 1
read: y = 0	read: x = 0
critical section	critical section



TSO

Dekker Protocol

“read overtaking write”

Initially: x = y = 0

P1

write: x = 1

read: y = 0

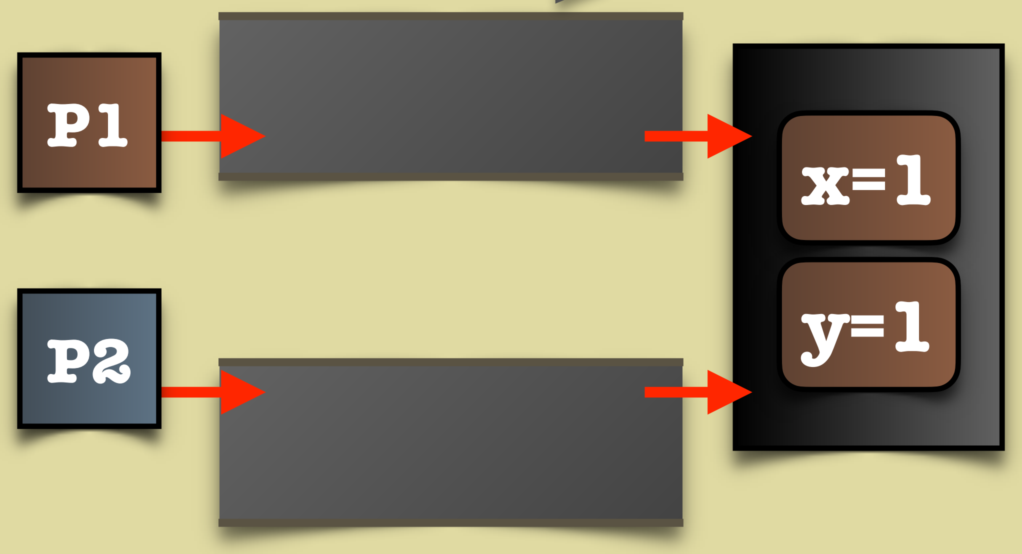
critical section

P2

write: y = 1

read: x = 0

critical section



TSO

Dekker Protocol

"read overtaking write"

Initially: x = y = 0

P1

write: x = 1

read: y = 0

critical section

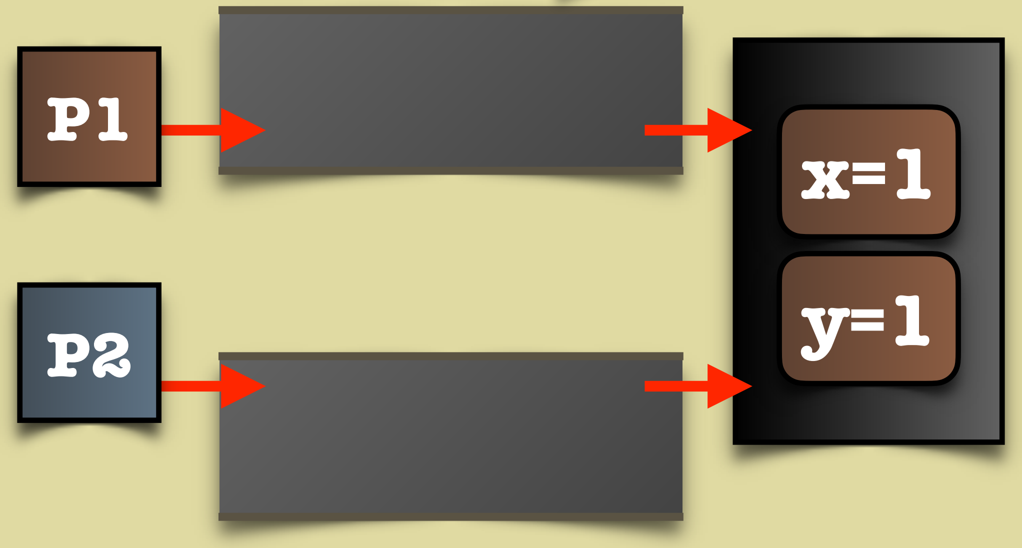
P2

write: y = 1

read: x = 0

critical section

"read overtaking write"



TSO

Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- Languages: C11

+ Efficiency

- Non-intuitive behaviours

Weakly Consistent Systems

- Cloud
- Weak memories
- Weak cache protocols
- Languages: C11

+ Efficiency

- Non-intuitive behaviours

- Semantics
- Correctness analysis: simulation, testing, verification, synthesis
- Methods and tools: decidability, complexity, algorithms
- Monitoring

Outline

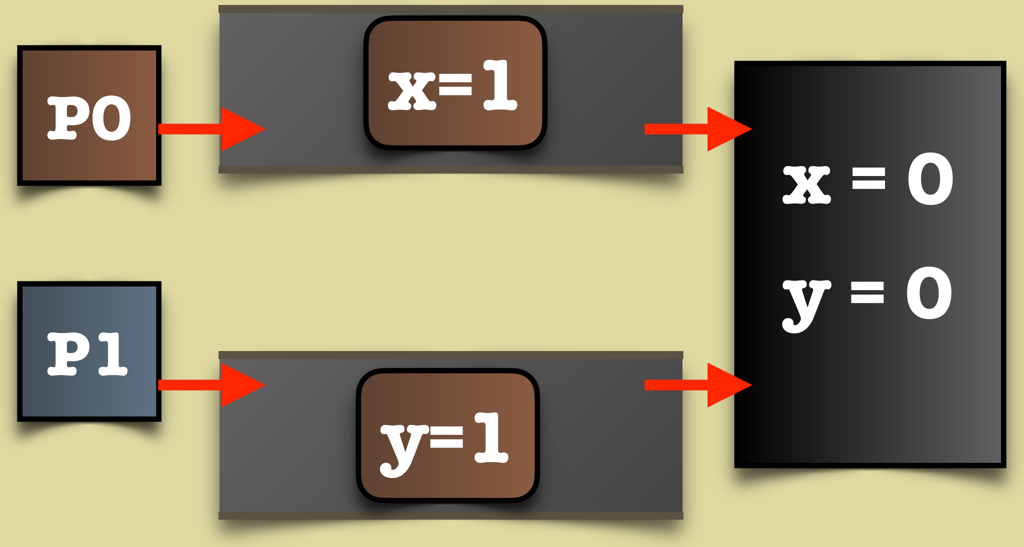
- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- Specification
- **Synthesis**

Potential Bad Behaviour - Dekker

Initially: x = y = 0

P0	P1
write: x = 1	write: y = 1
mfence	mfence
read: y = 0	read: x = 0
critical section	critical section

fence instruction



TSO

Potential Bad Behaviour - Dekker

Initially: x = y = 0

P0

write: x = 1

mfence

read: y = 0

critical section

P1

write: y = 1

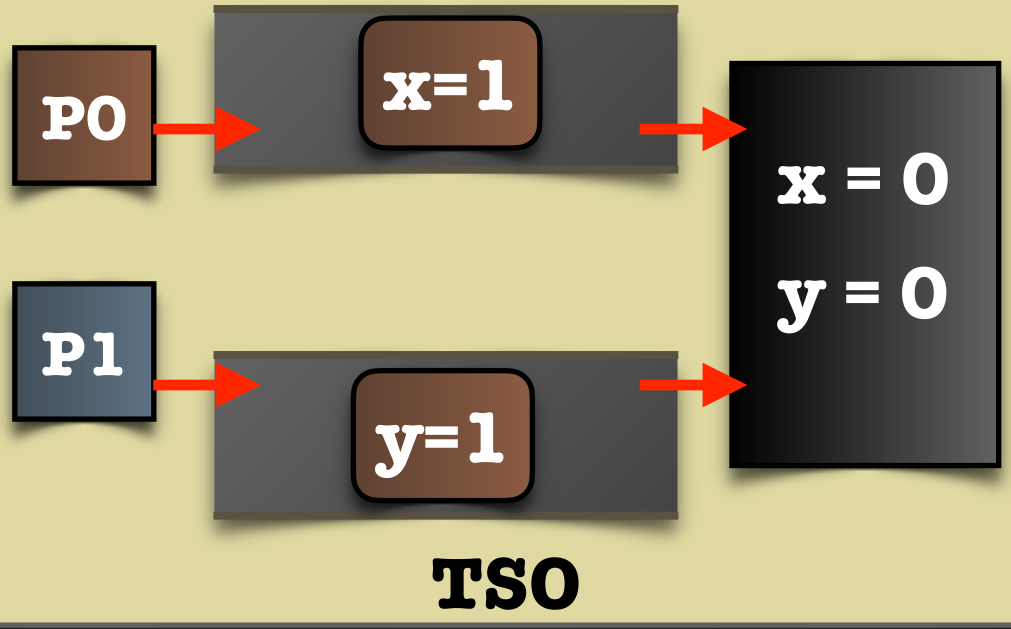
mfence

read: x = 0

critical section

fence instruction

flushes the buffer



Potential Bad Behaviour - Dekker

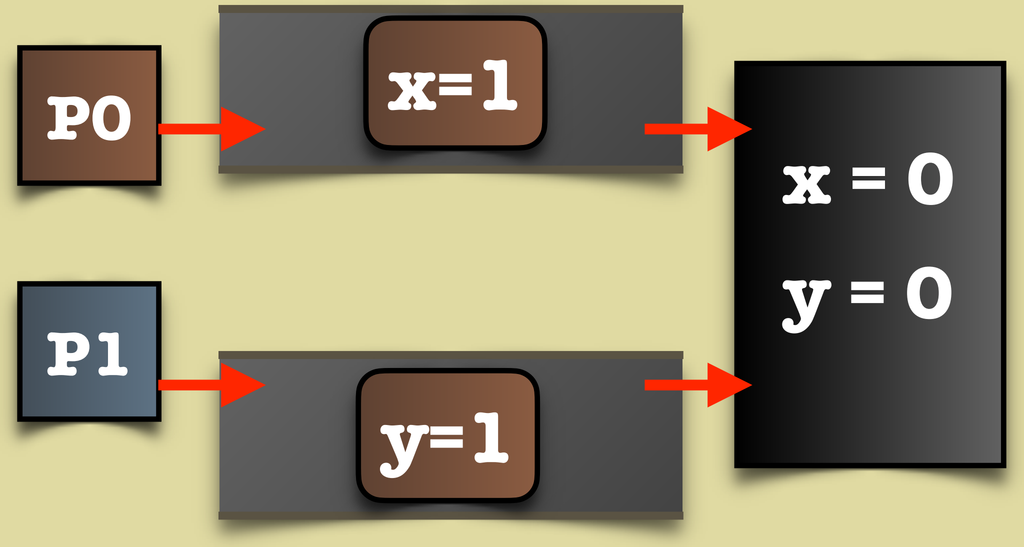
Initially: x = y = 0

P0	**P1**
write: x = 1	write: y = 1
mfence	mfence
read: y = 0	read: x = 0
critical section	critical section

fence instruction

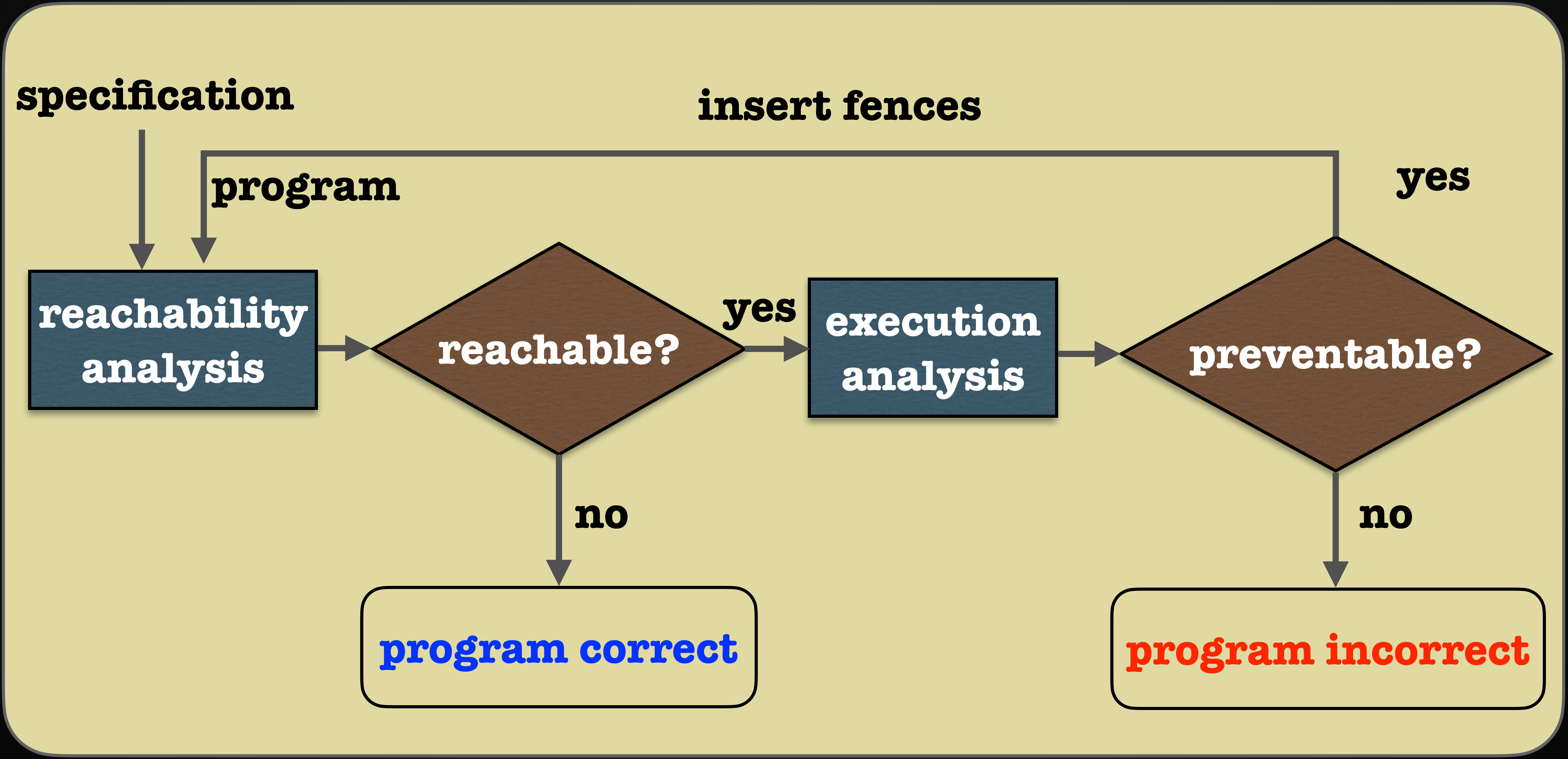
flushes the buffer

prevents re-ordeirng

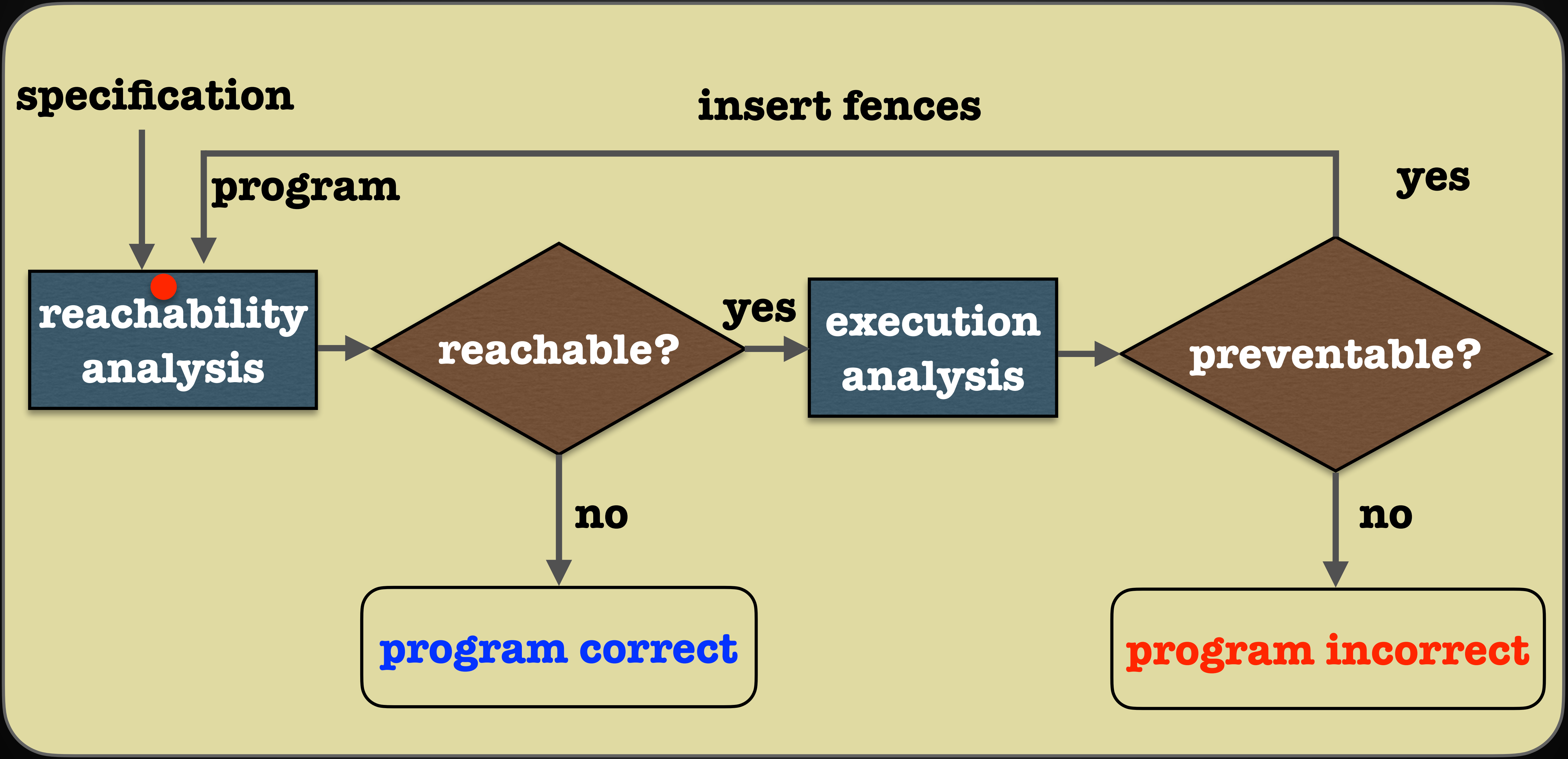


TSO

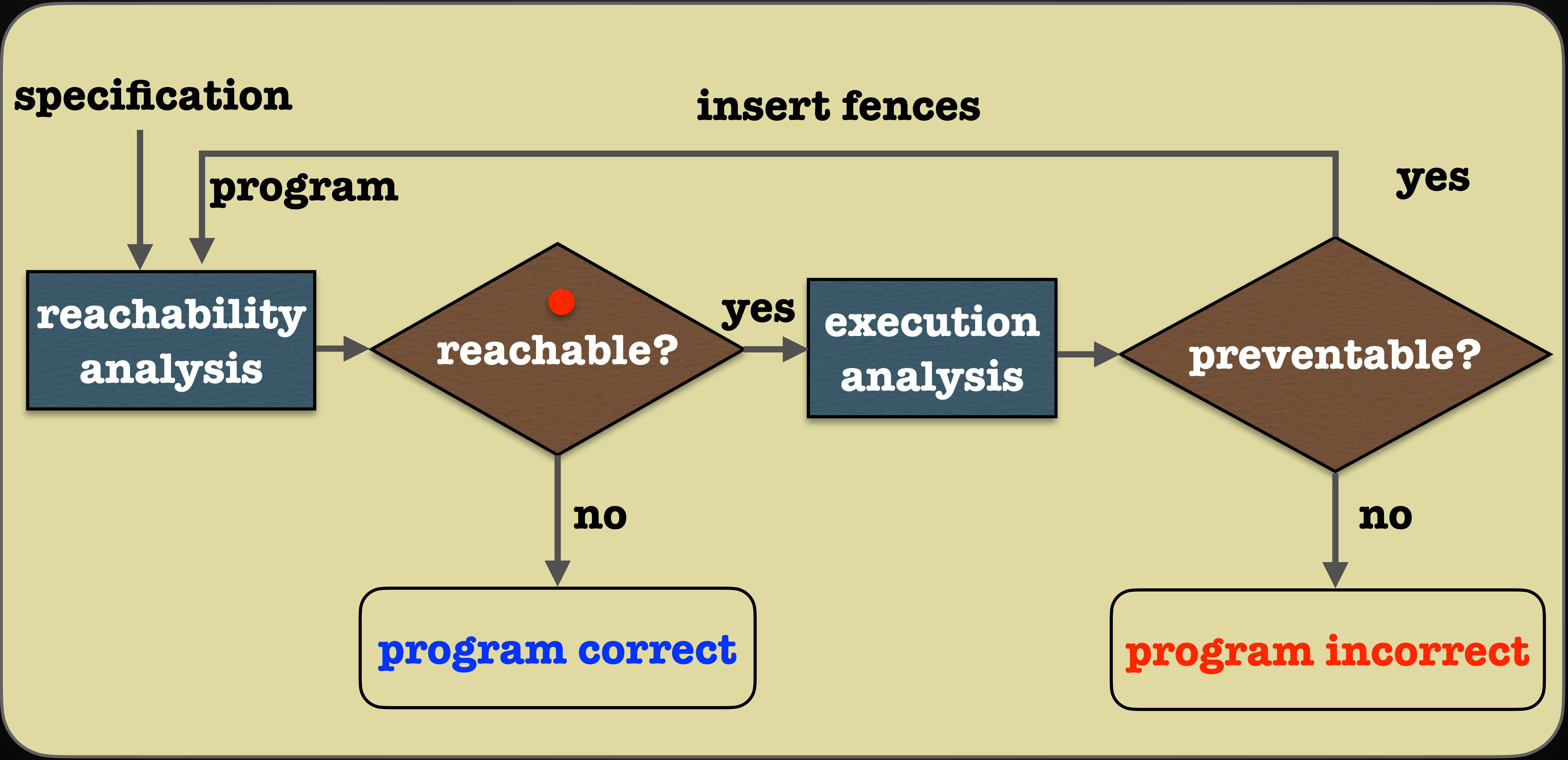
Verification and Correction



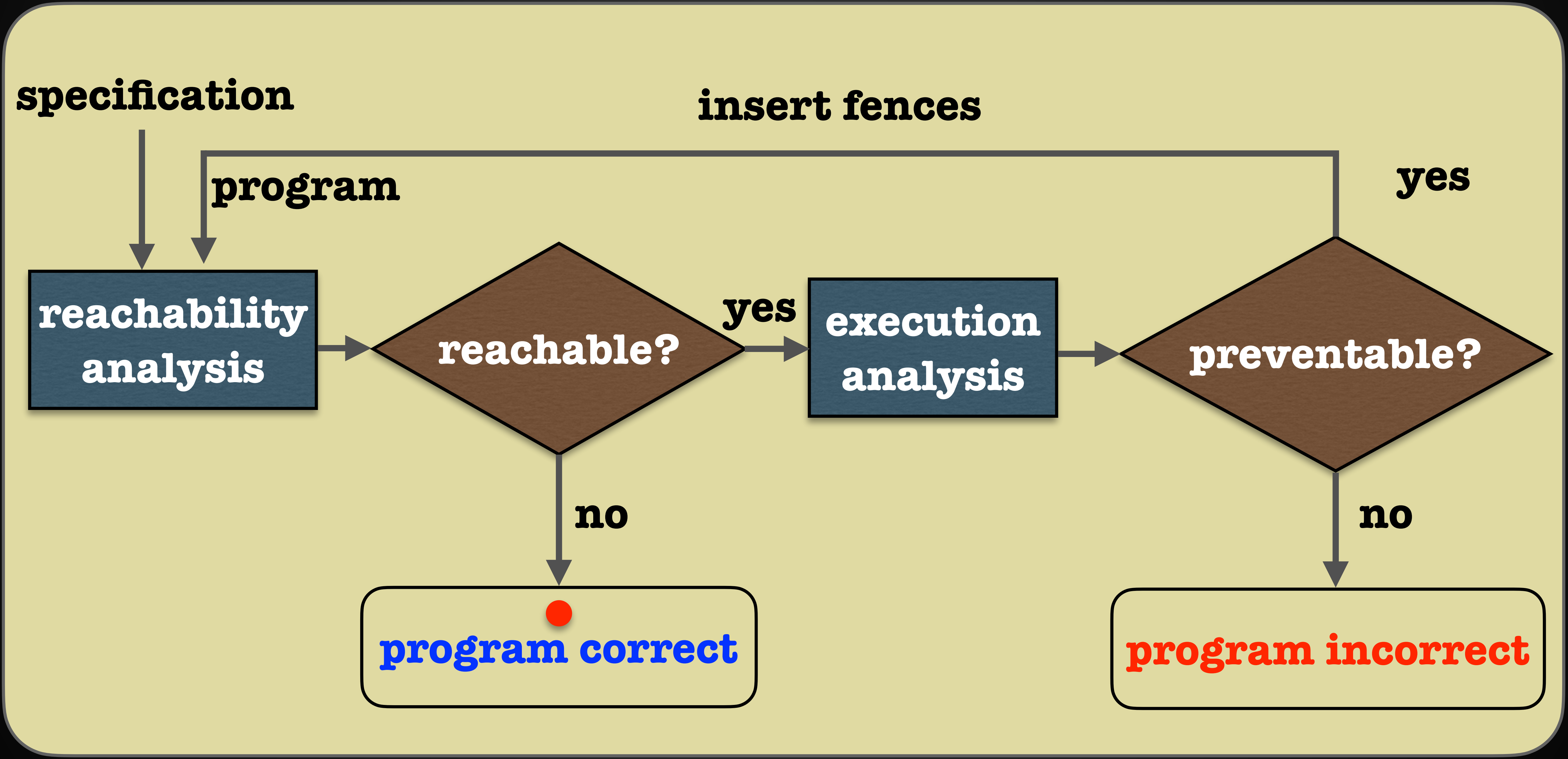
Verification and Correction



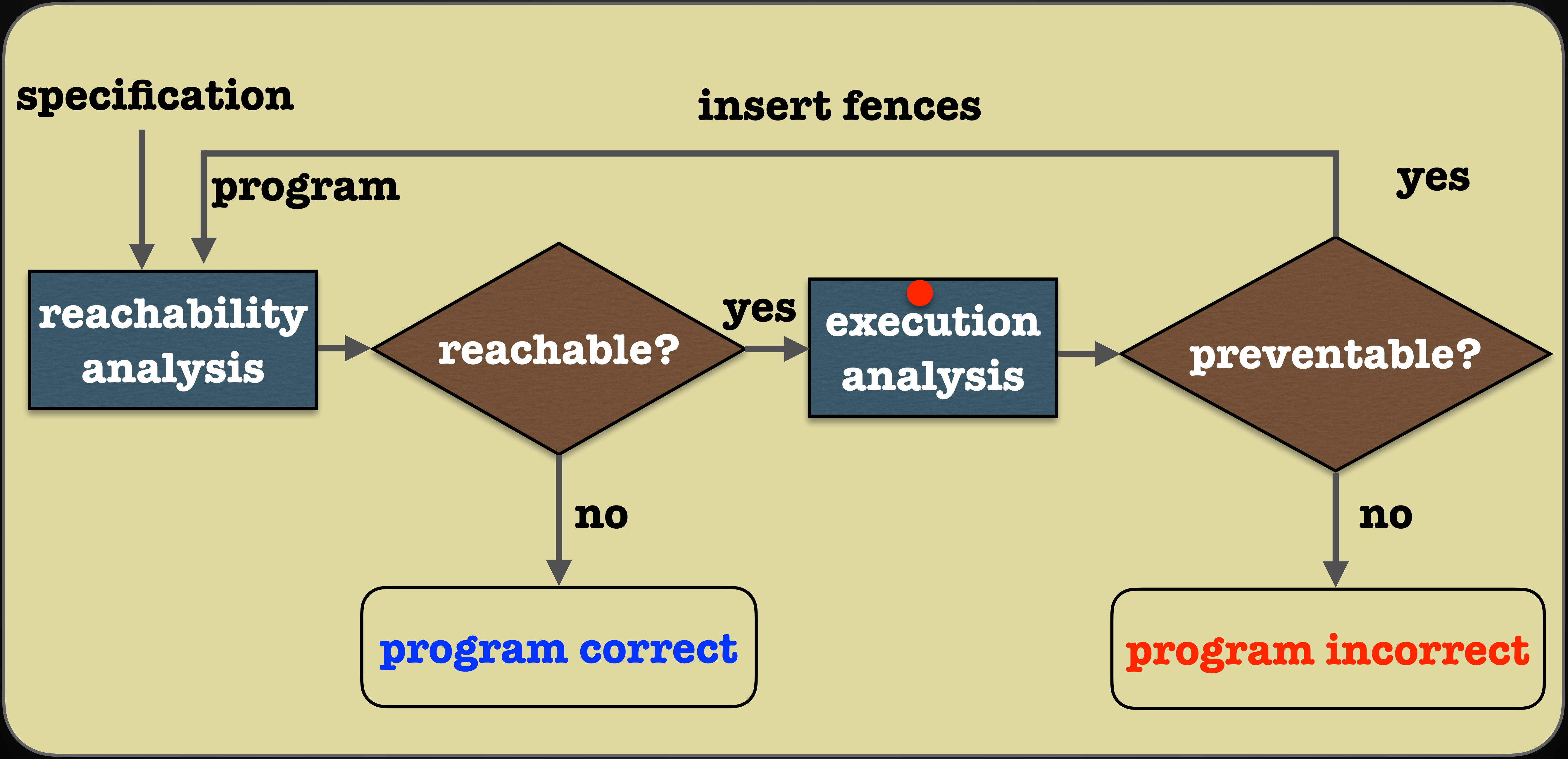
Verification and Correction



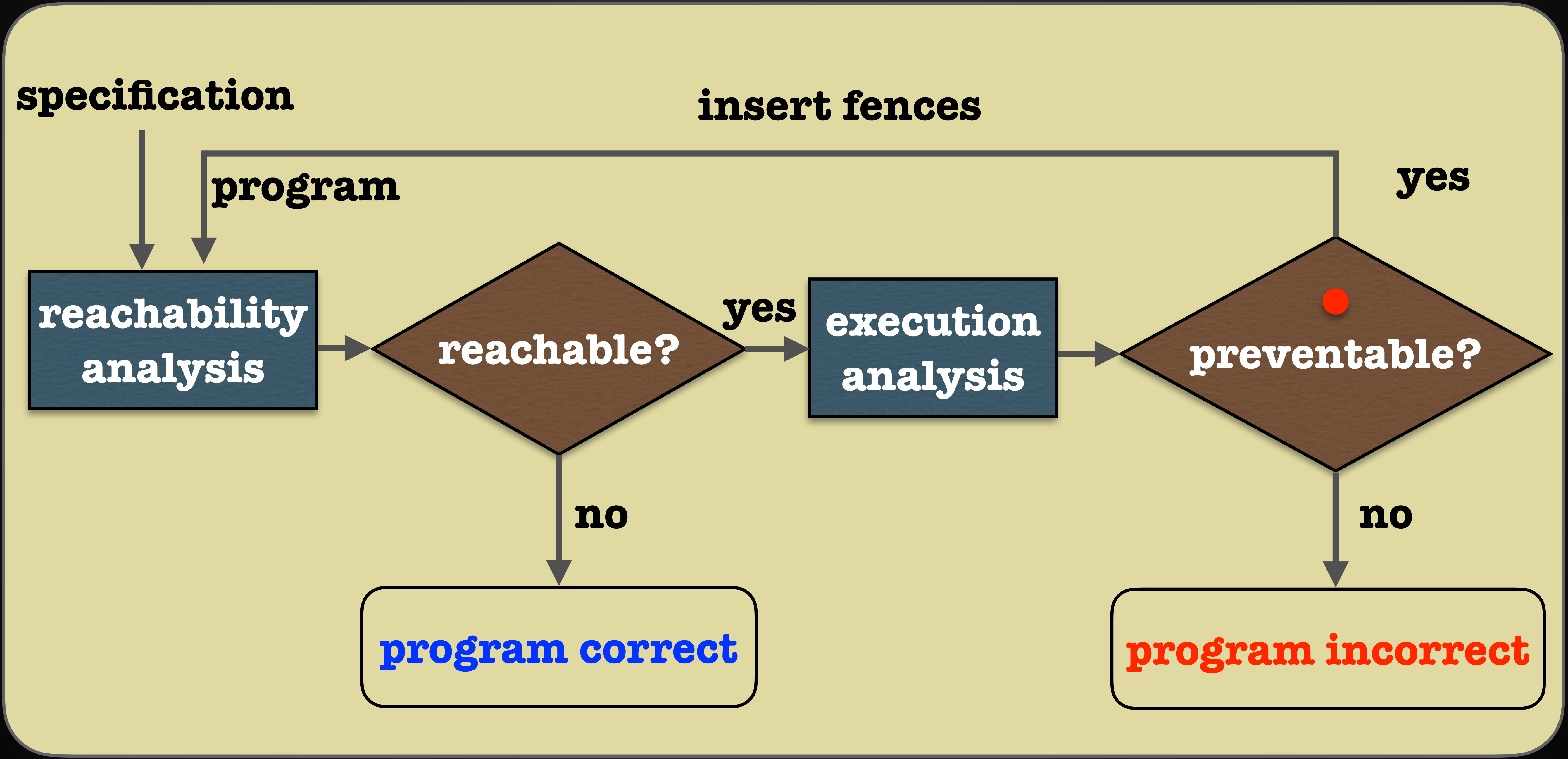
Verification and Correction



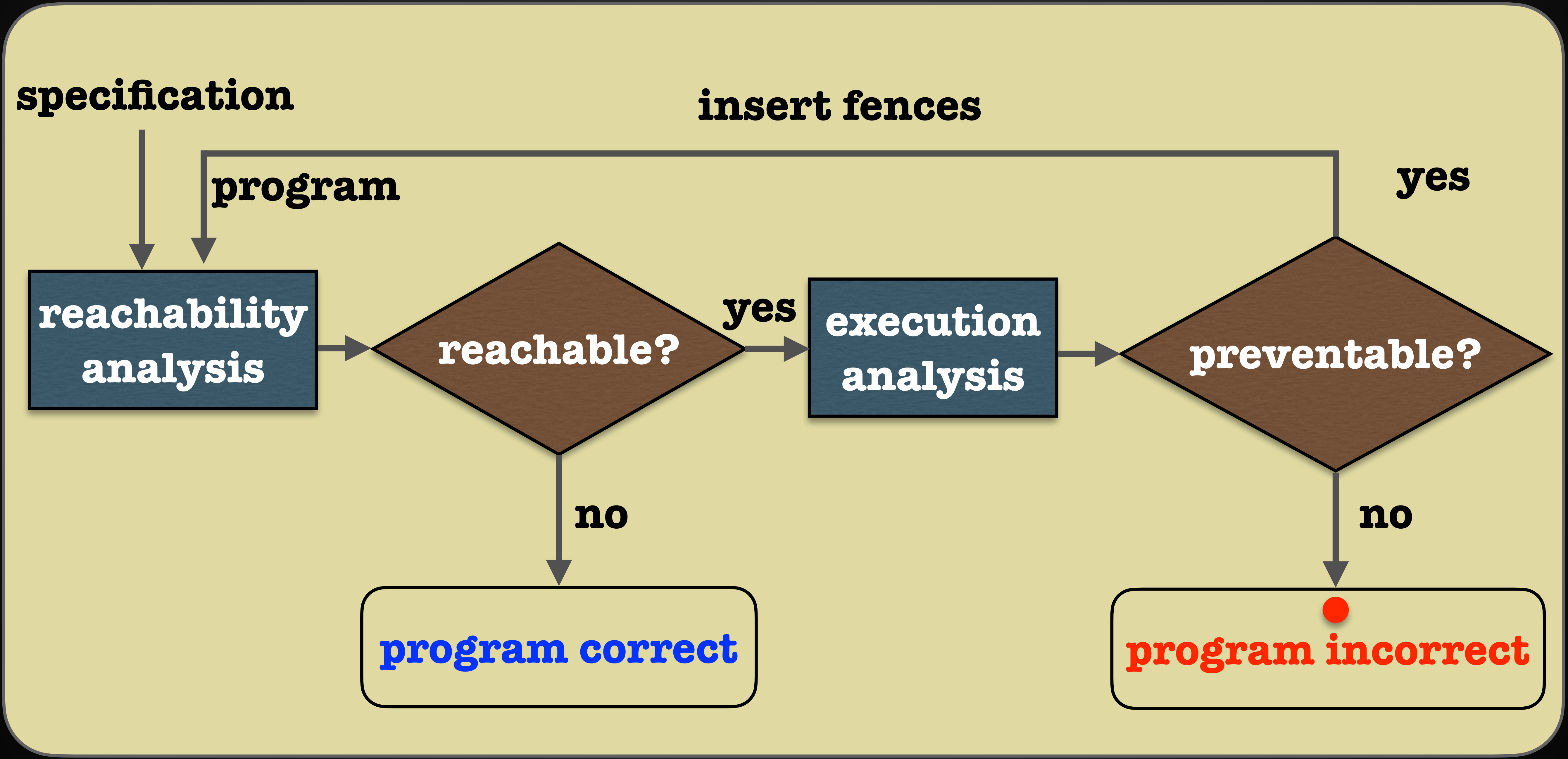
Verification and Correction



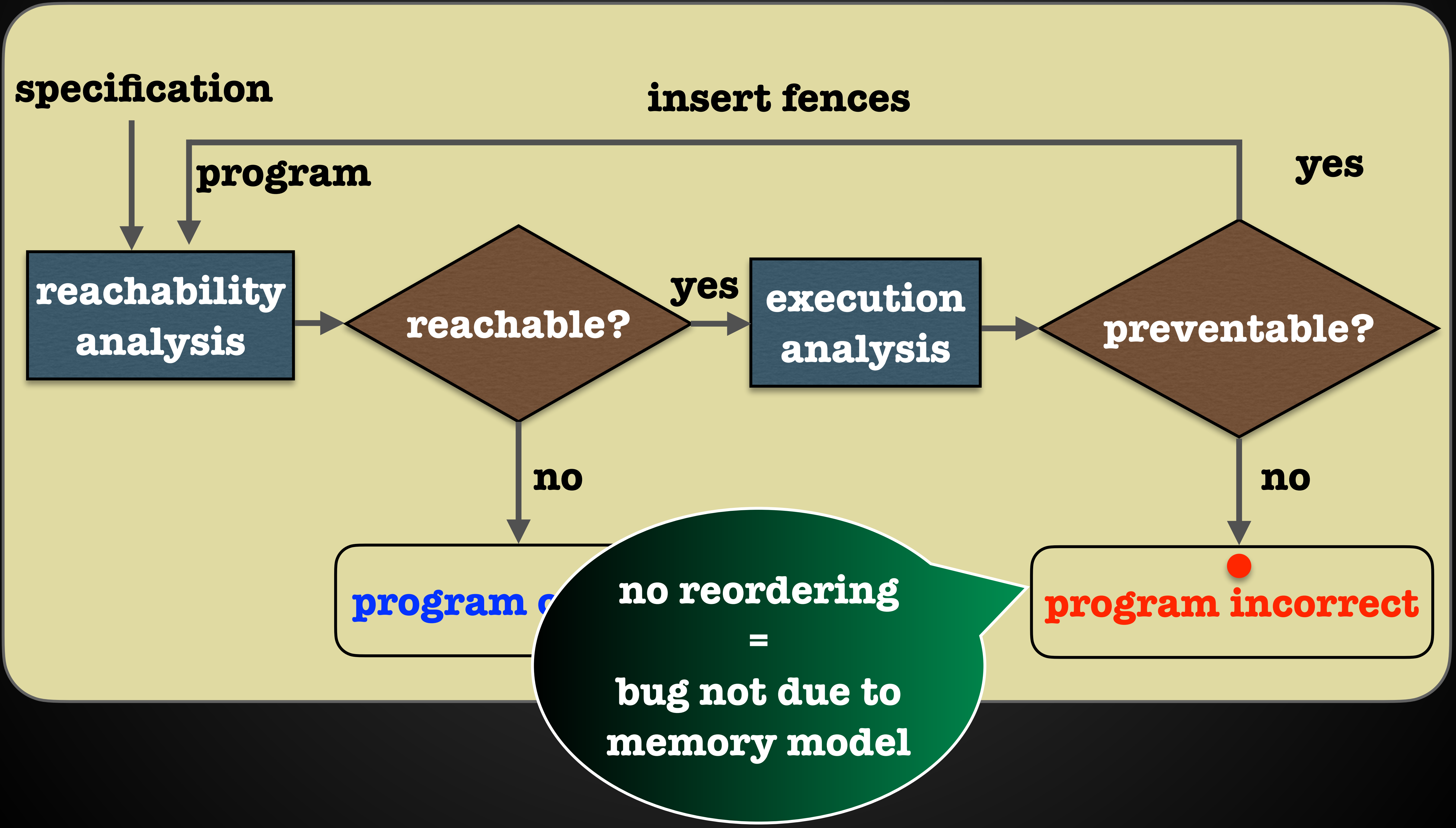
Verification and Correction



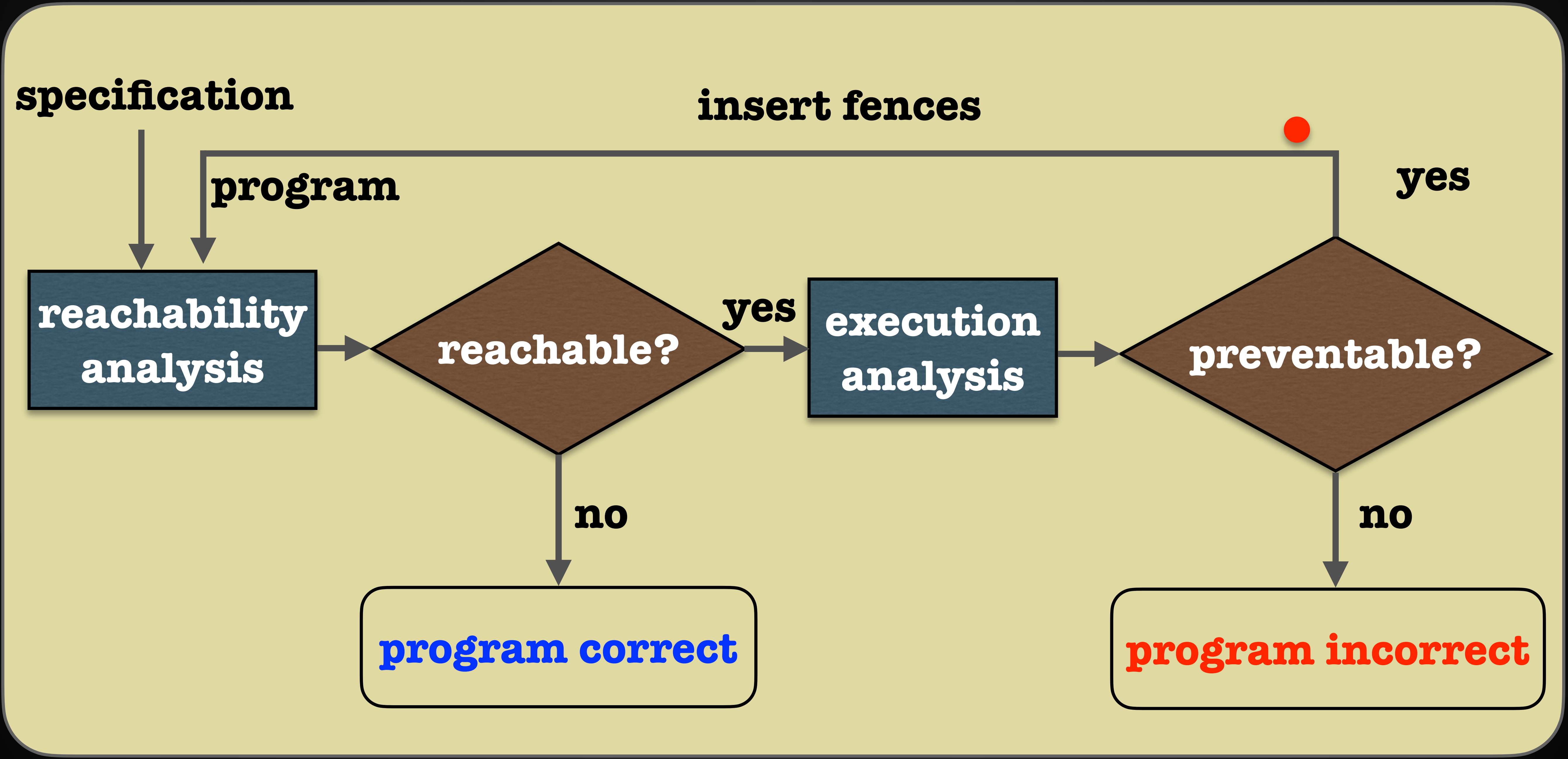
Verification and Correction



Verification and Correction

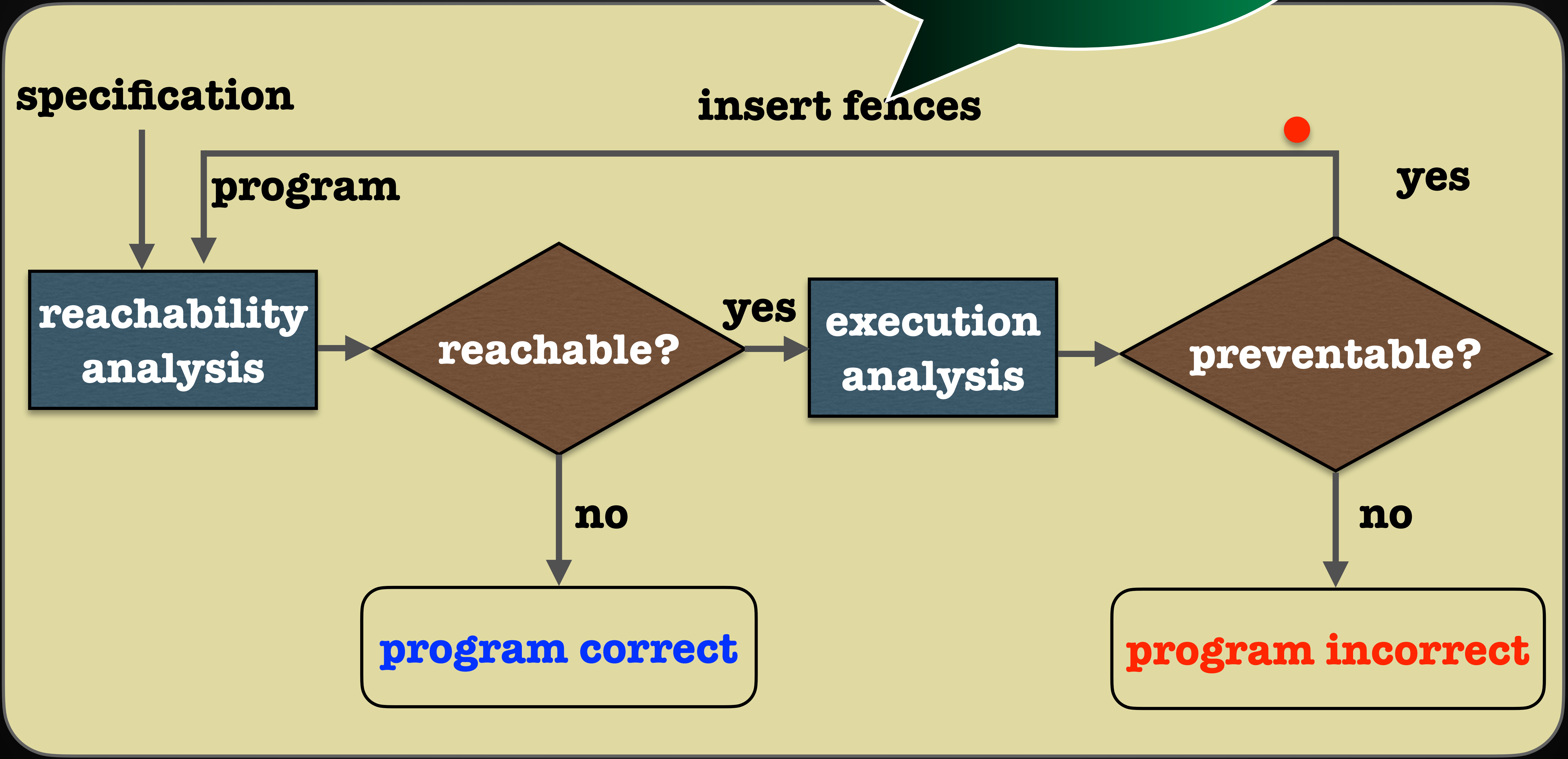


Verification and Correction

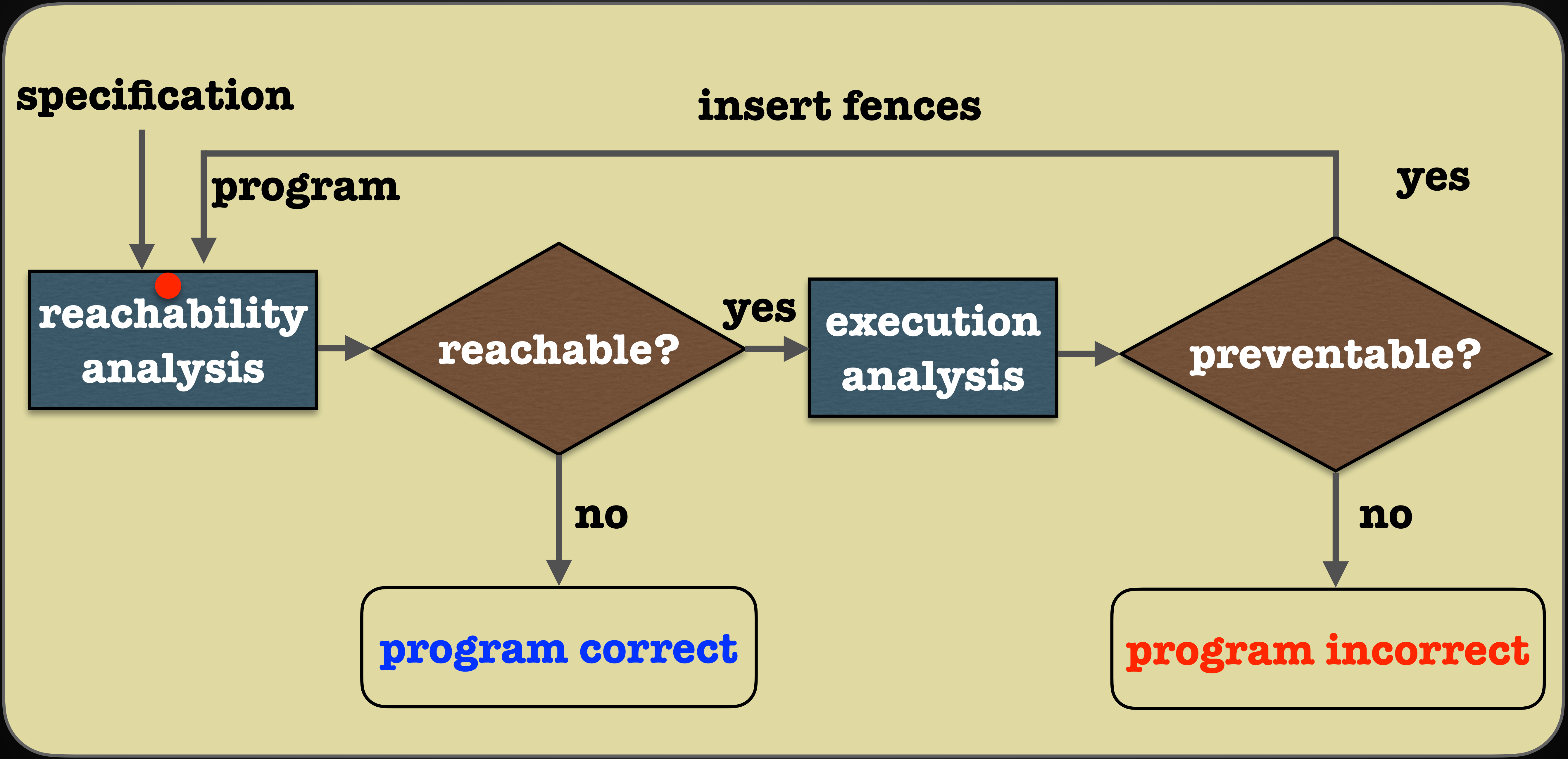


Verification an n

find reordering
and
prevent it



Verification and Correction



specification

insert fences

program

yes

reachability analysis

reachable?

yes

execution analysis

preventable?

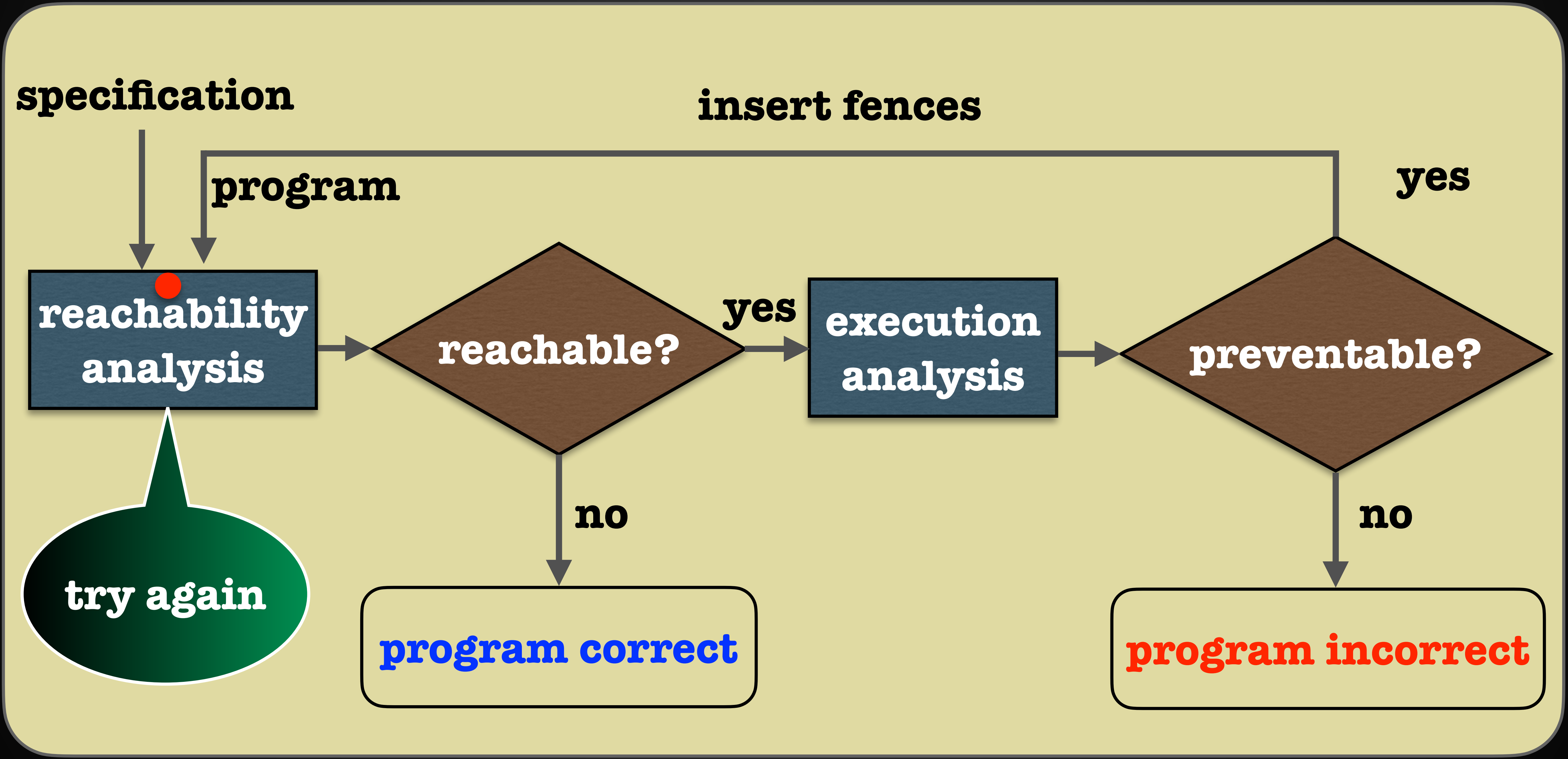
no

no

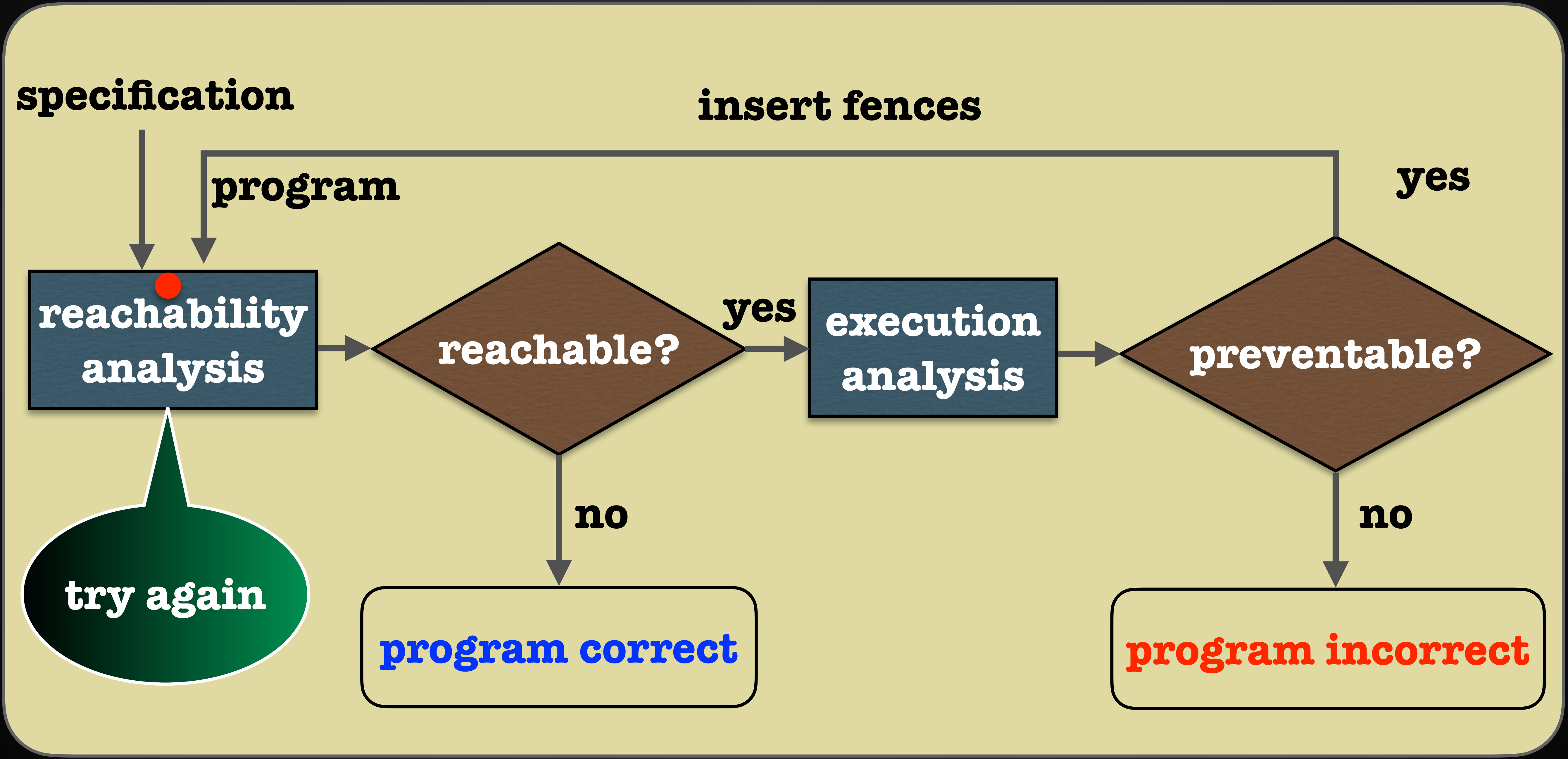
program correct

program incorrect

Verification and Correction

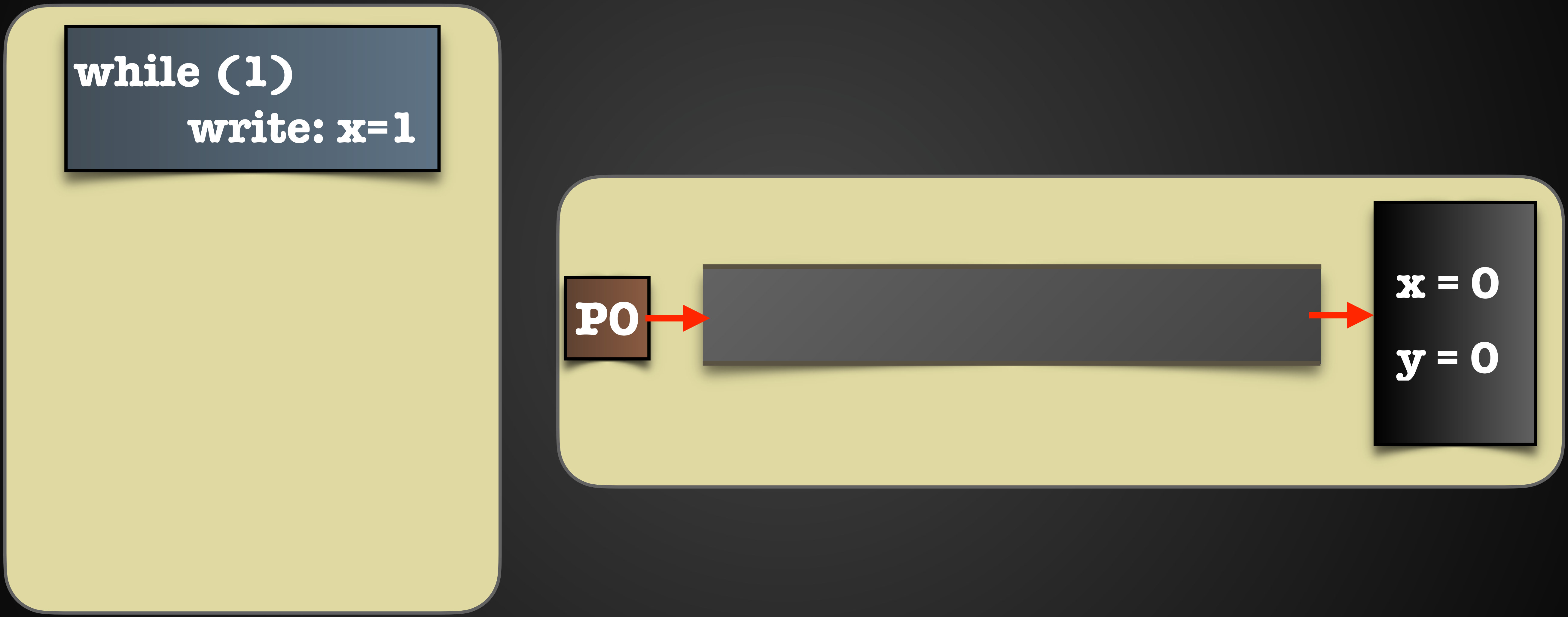


Verification and Correction



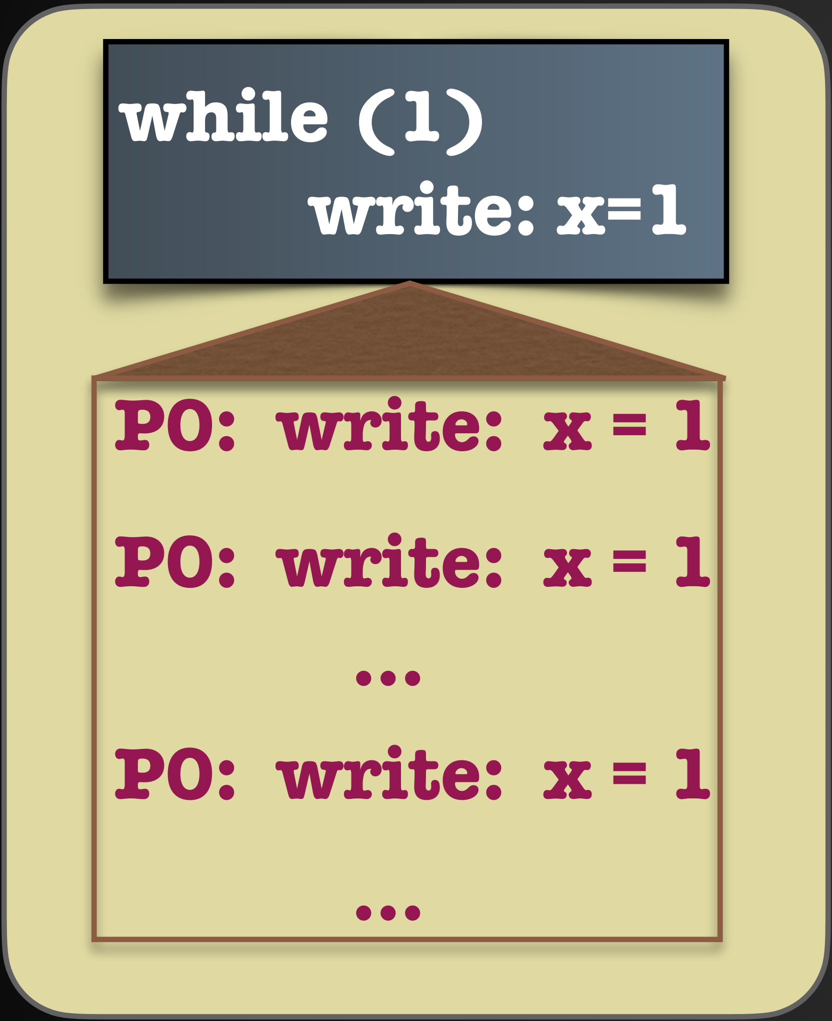
optimality = smallest set of fences needed for correctness

Verification under TSO is Difficult

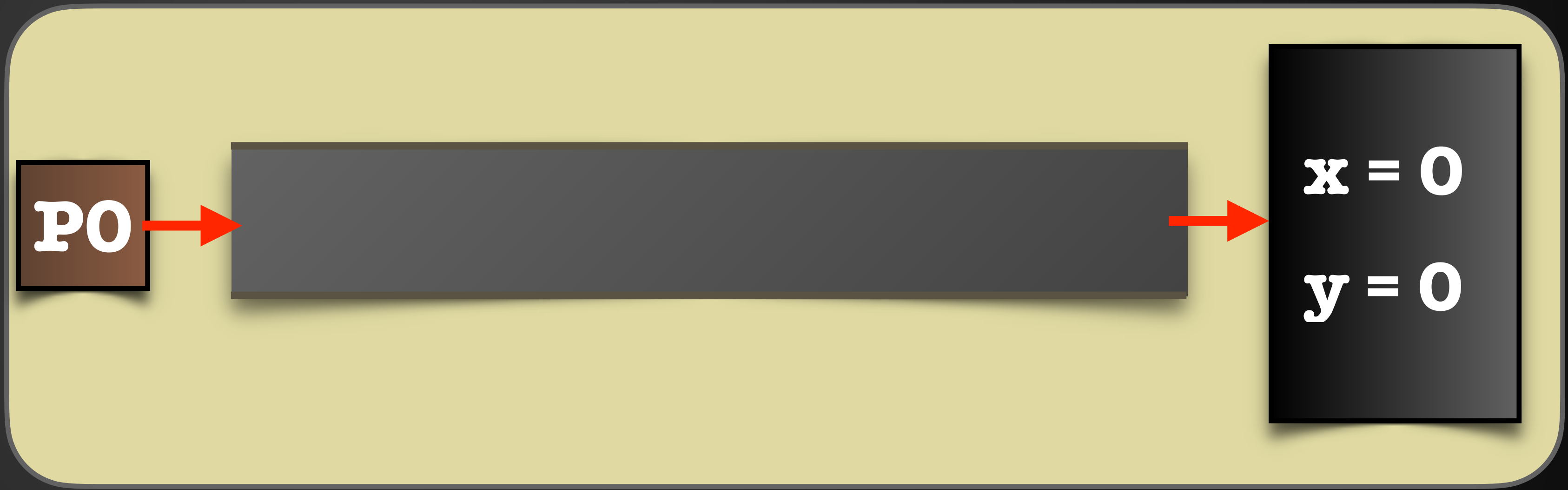


```
while (1)
        write: x=1
```

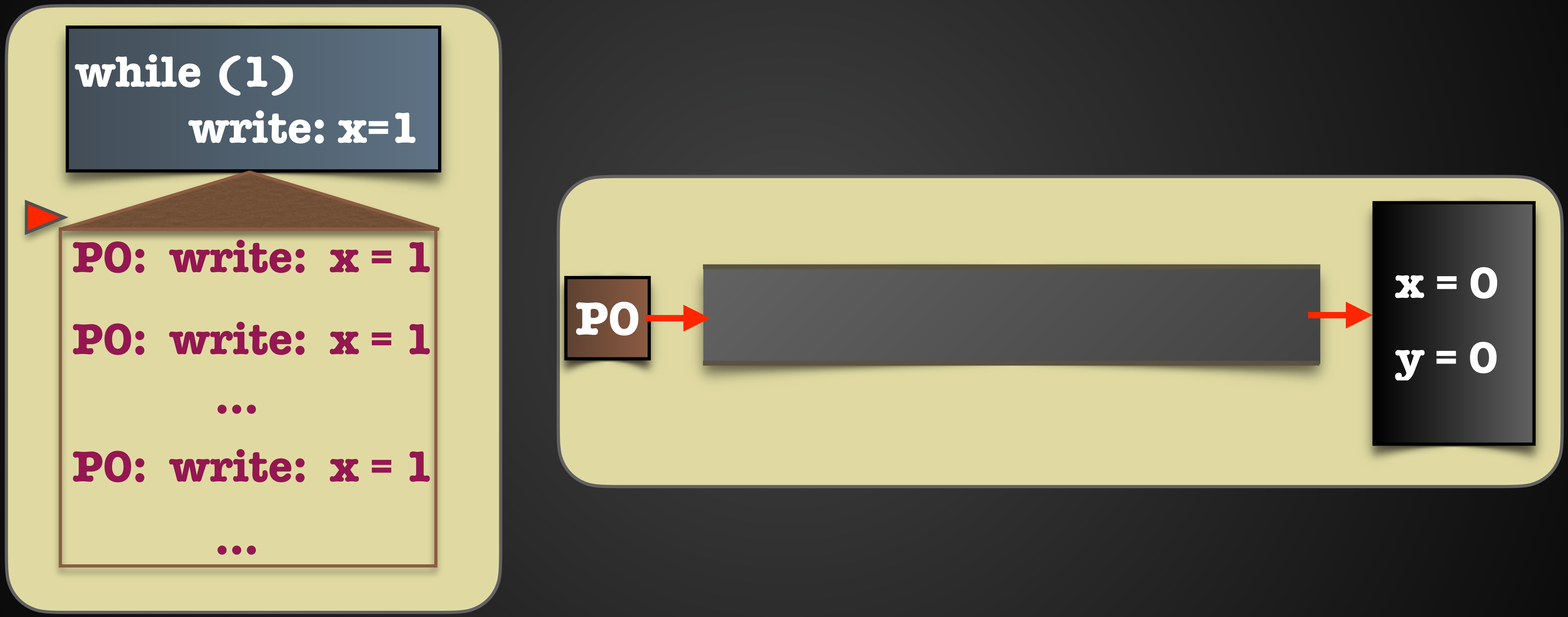
Verification under TSO is Difficult



```
while (1)
        write: x=1
```



Verification under TSO is Difficult



```
while (1)
          write: x=1
```

PO: write: x = 1

PO: write: x = 1

...

PO: write: x = 1

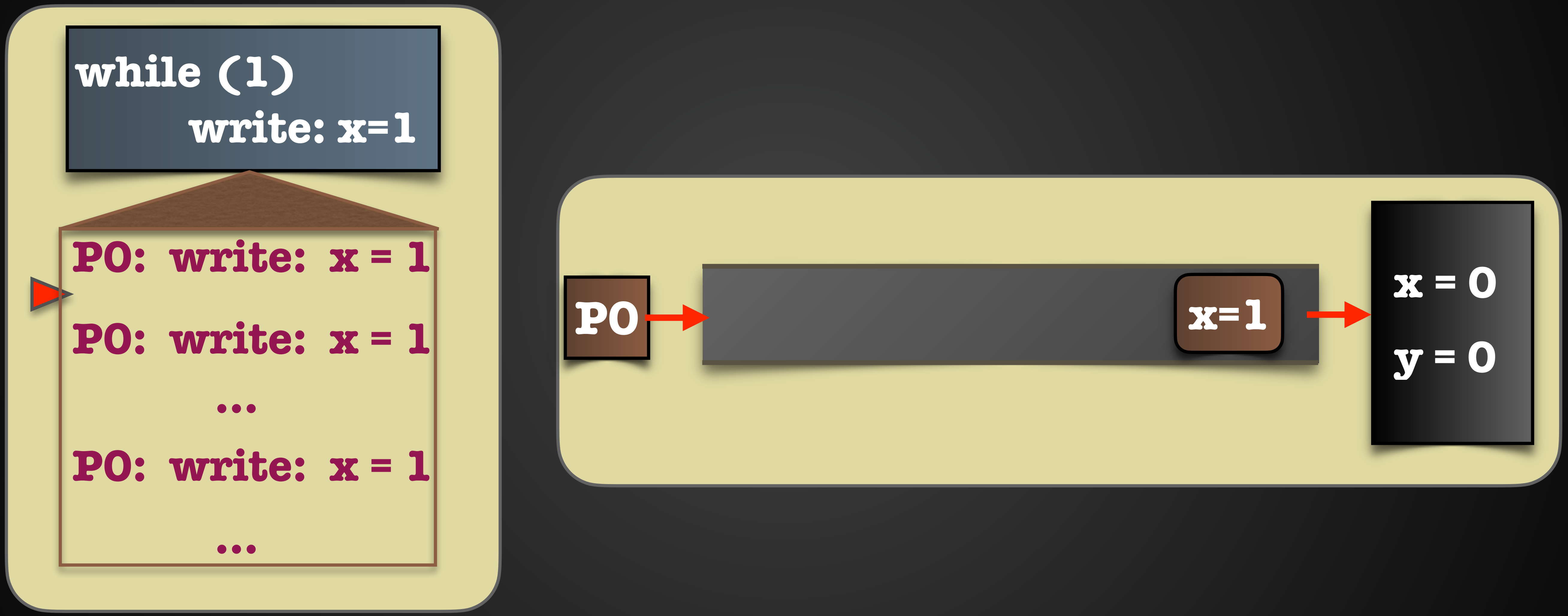
...

PO

x = 0

y = 0

Verification under TSO is Difficult



```
while (1)
        write: x=1
```

P0: write: x = 1

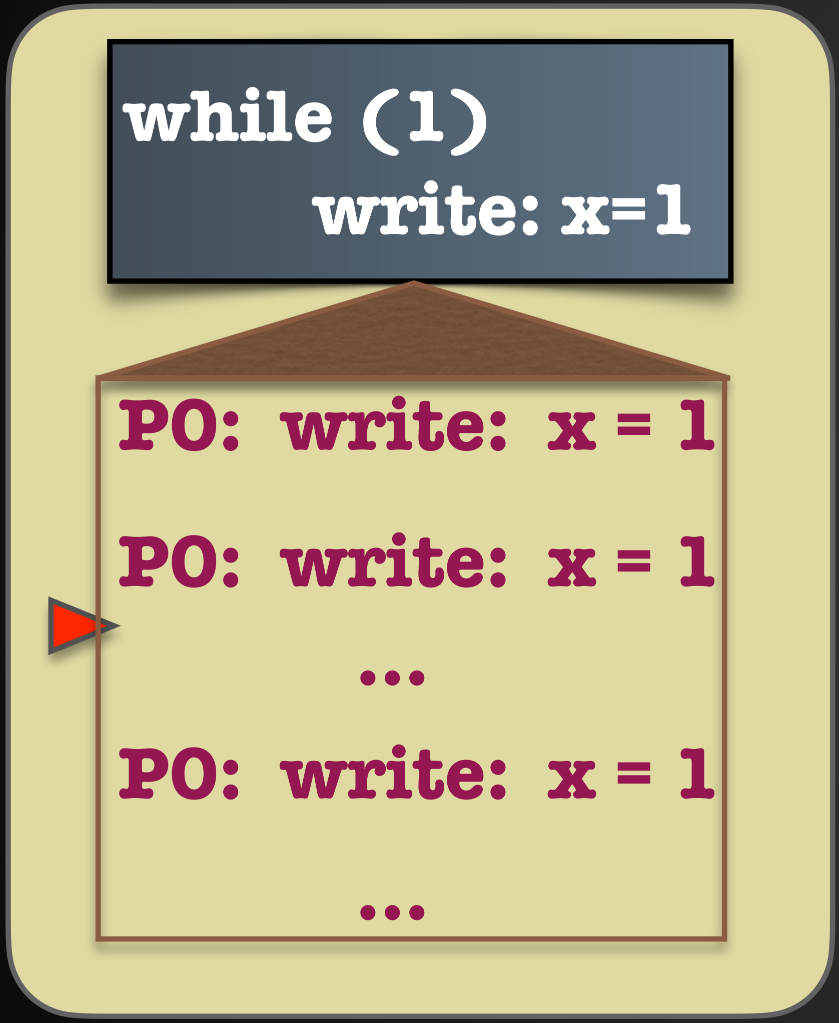
P0: write: x = 1

...

P0: write: x = 1

...

Verification under TSO is Difficult



```
while (1)
        write: x=1
```

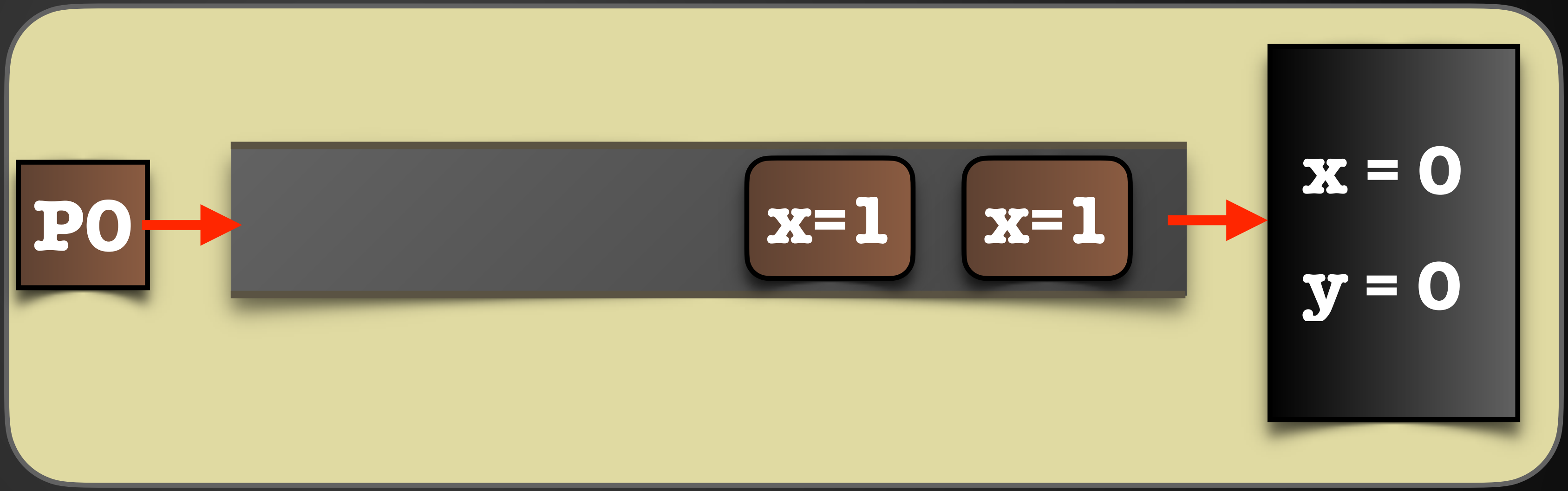
PO: write: x = 1

PO: write: x = 1

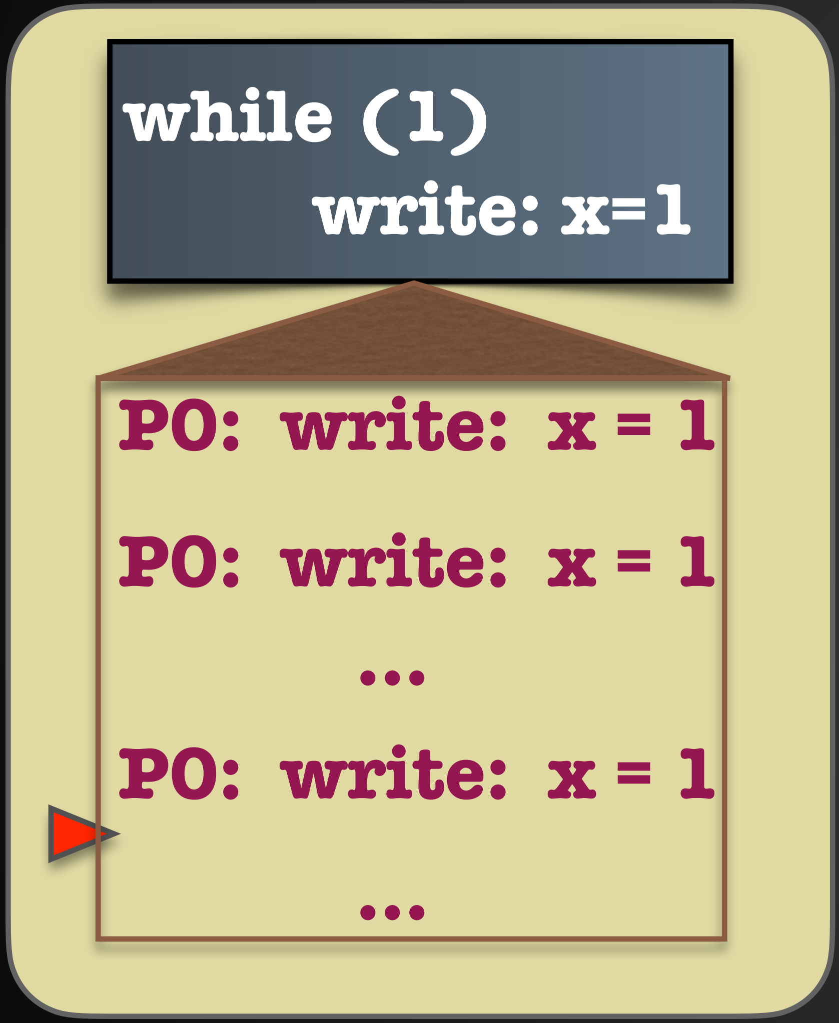
...

PO: write: x = 1

...



Verification under TSO is Difficult



```
while (1)
        write: x=1
```

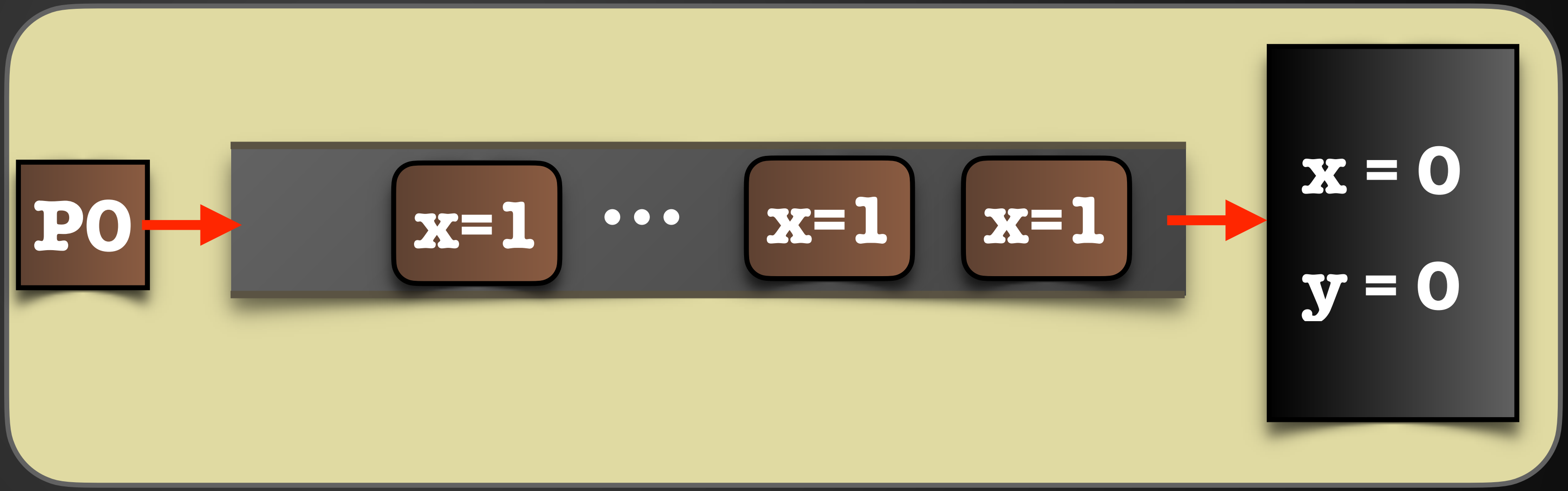
PO: write: x = 1

PO: write: x = 1

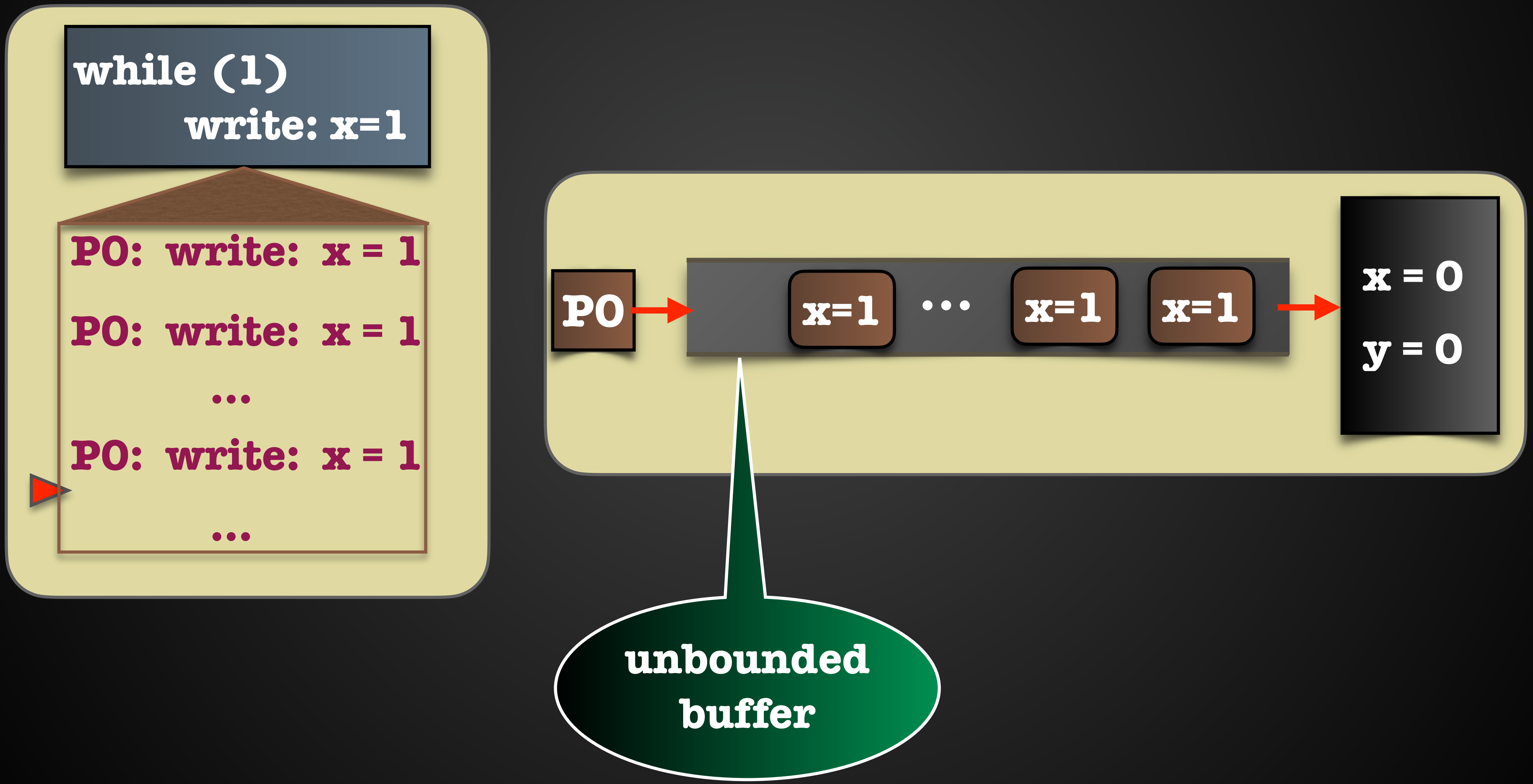
...

PO: write: x = 1

...



Verification under TSO is Difficult



```
while (1)
        write: x=1
```

PO: write: x = 1

PO: write: x = 1

...

PO: write: x = 1

...

PO

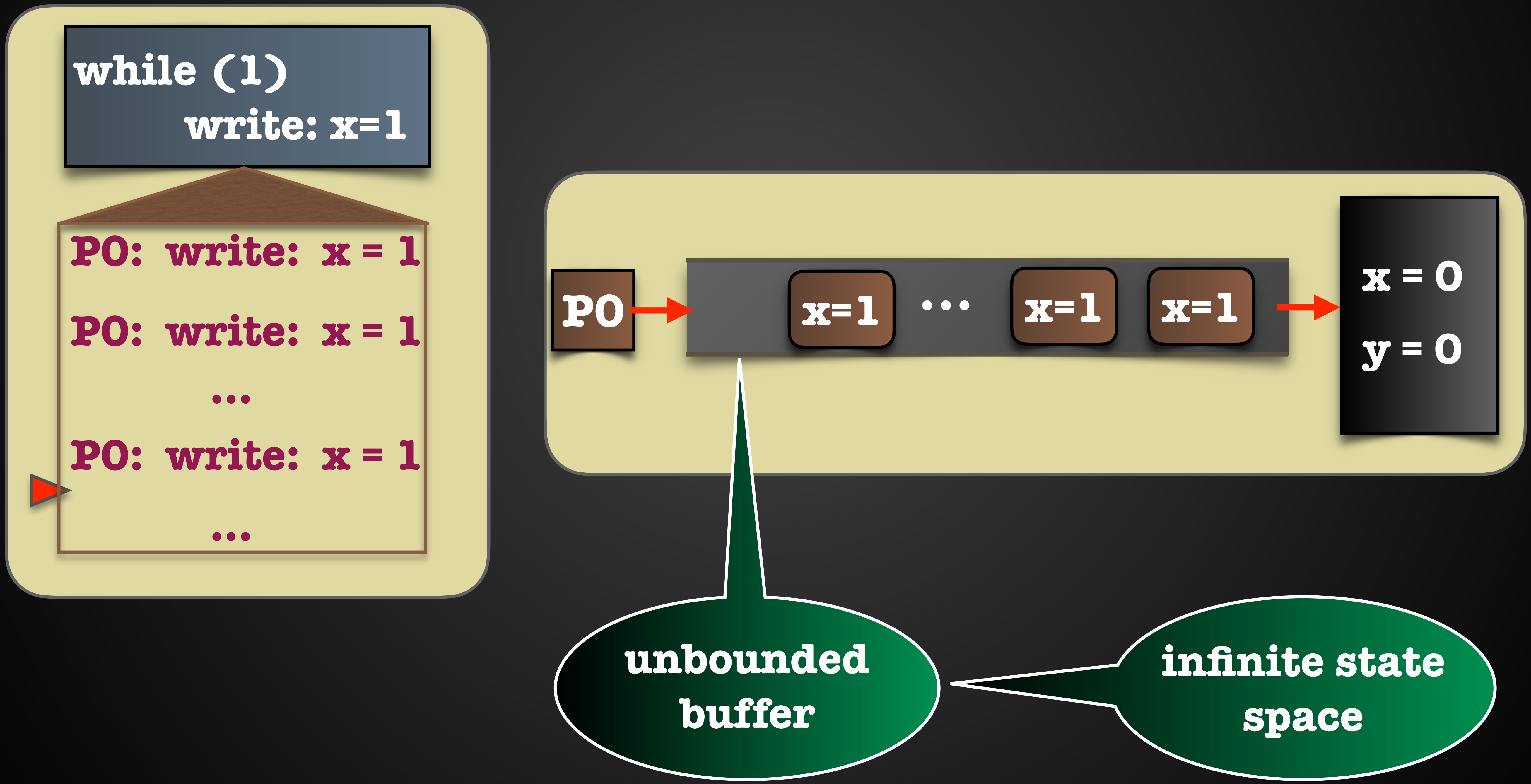
x=1 ... x=1 x=1

x = 0

y = 0

unbounded buffer

Verification under TSO is Difficult



```
while (1)
        write: x=1
```

```
PO:  write:  x = 1
PO:  write:  x = 1
        …
PO:  write:  x = 1
        …
```

unbounded buffer

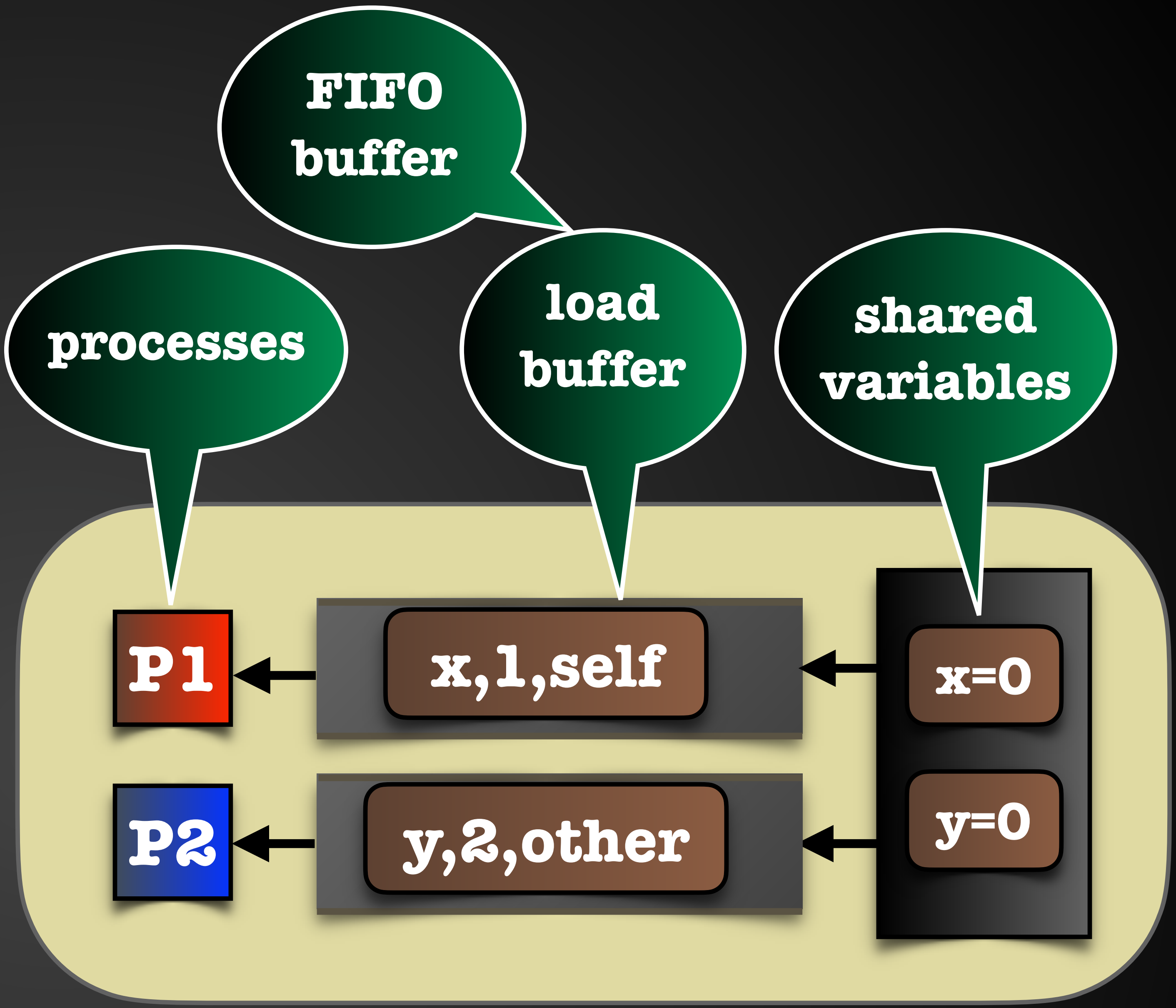
infinite state space

Outline

- Weak Consistency
- Total Store Order (TSO)
- **Dual TSO**
- Verification
- Monitors
- Synthesis

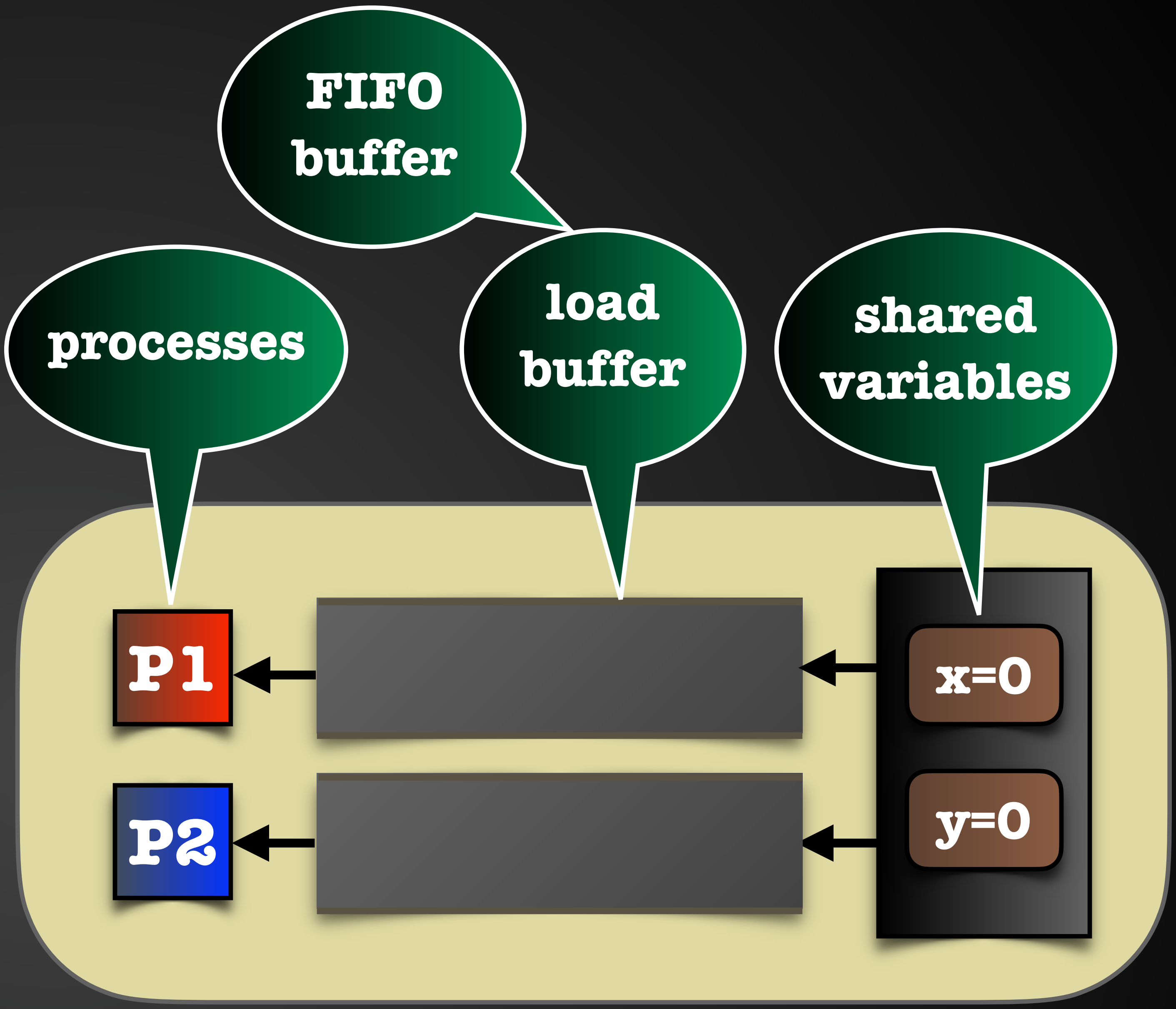
Dual TSO

- store buffer → load buffer
- write **immediately updates** memory
- buffers contain **expected reads**
- messages: **self, other**



Dual TSO

- store buffer ↷ load buffer
- write **immediately updates** memory
- buffers contain **expected reads**
- messages: **self, other**

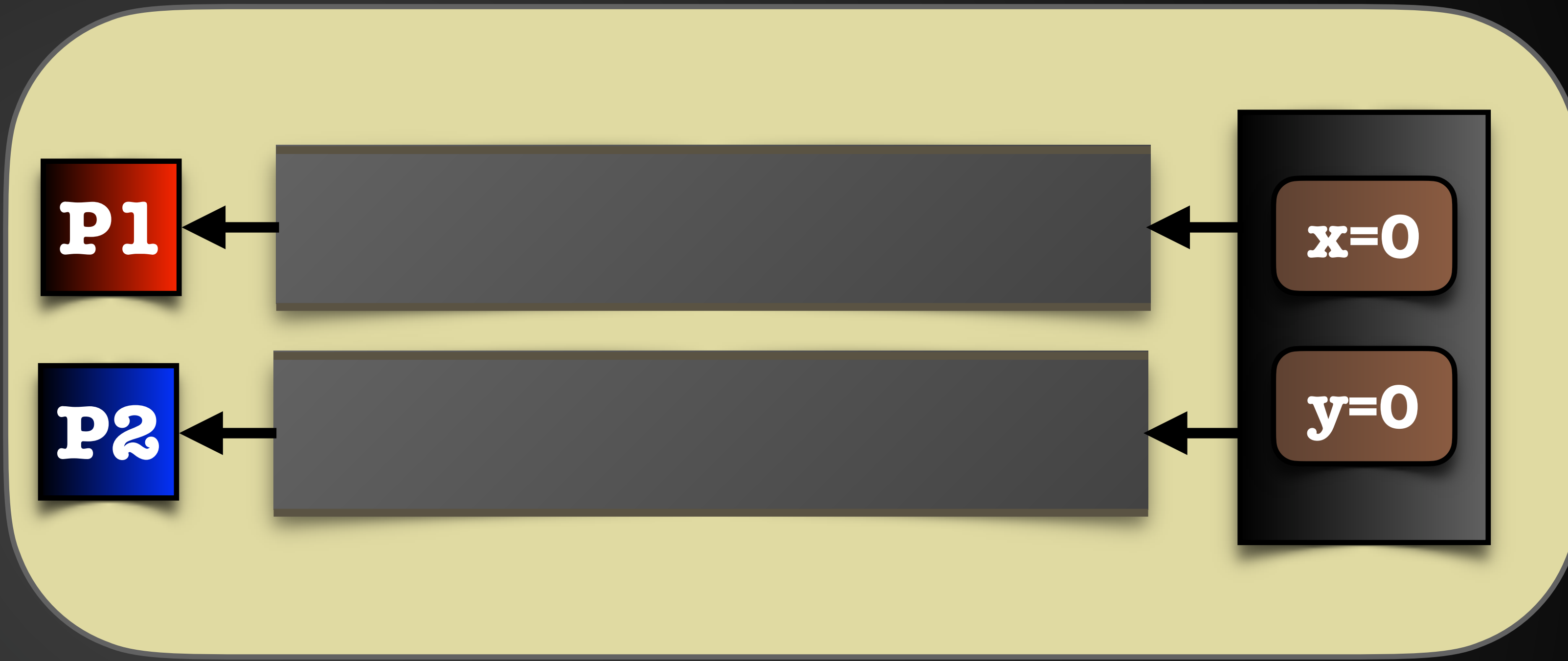


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0

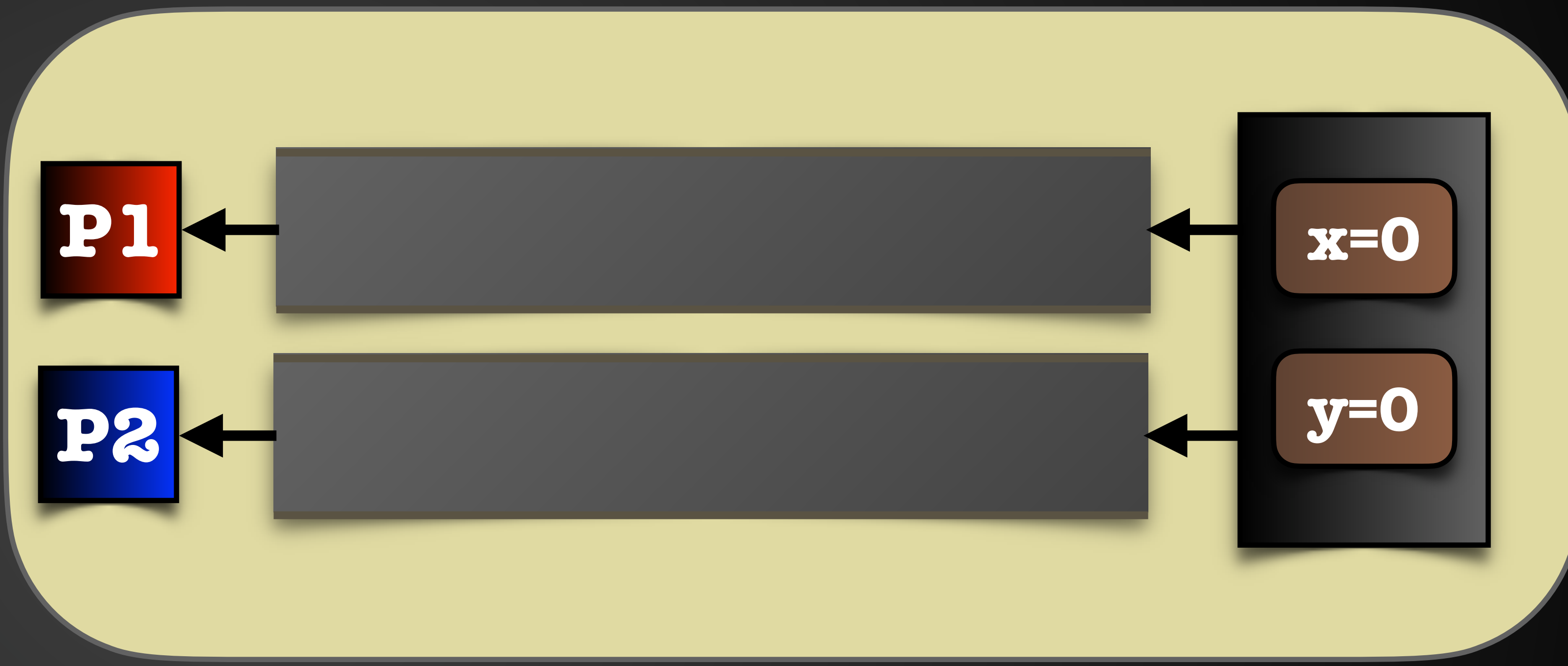


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0

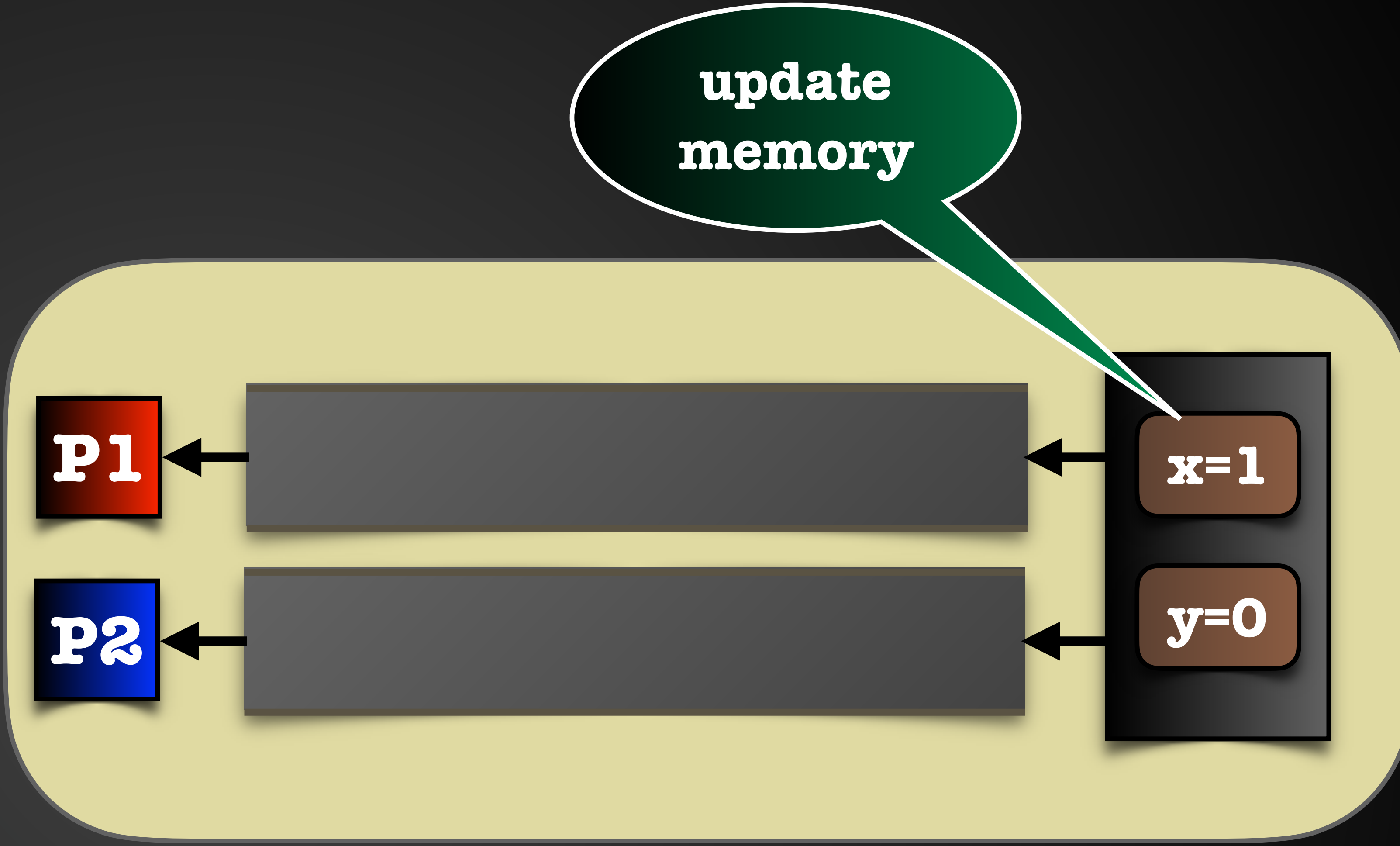


Dual TSO

P1: write: x = 1

▶ P1: read: x = 1

P1: read: y = 0

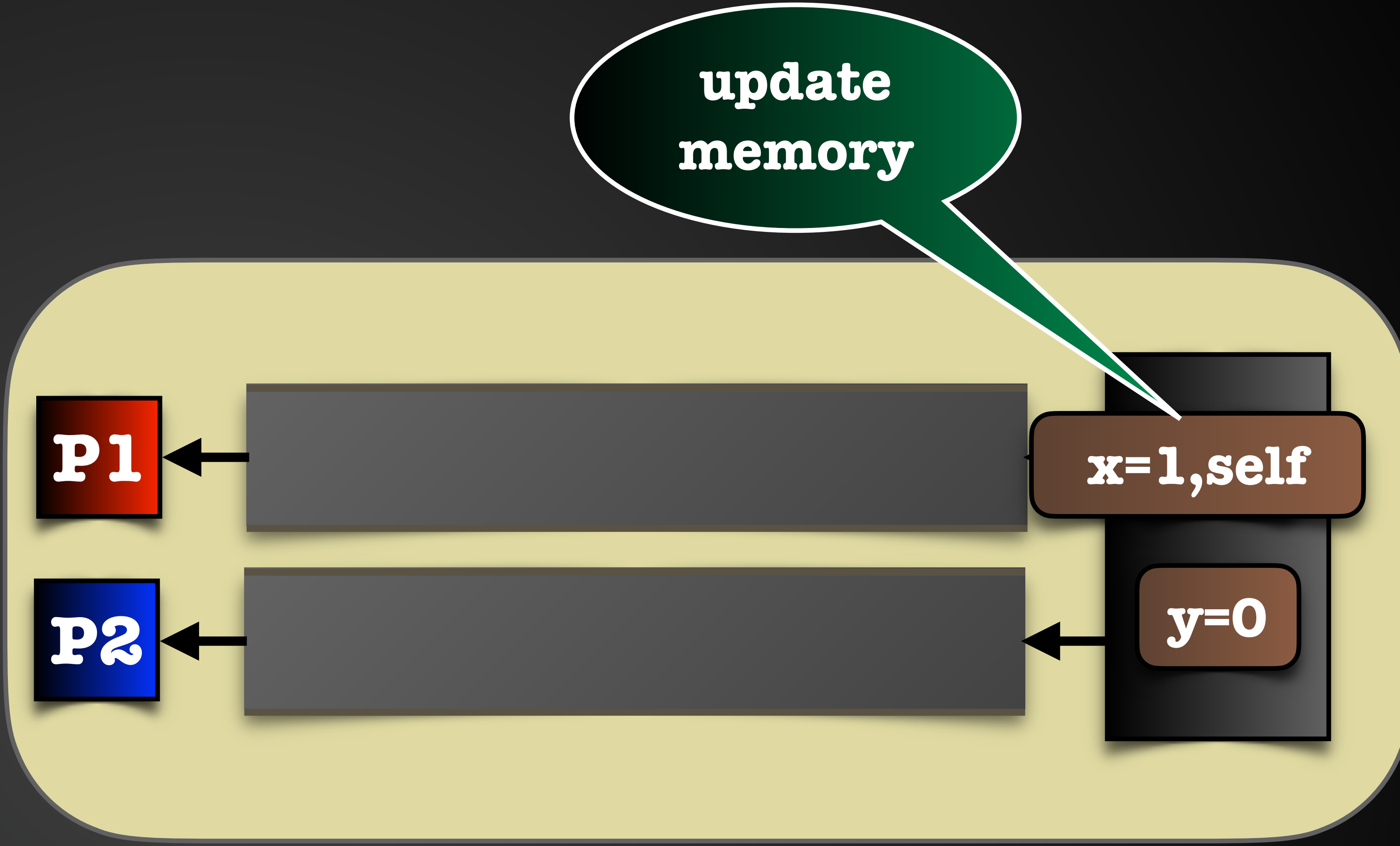


Dual TSO

P1: write: x = 1

▶ P1: read: x = 1

P1: read: y = 0

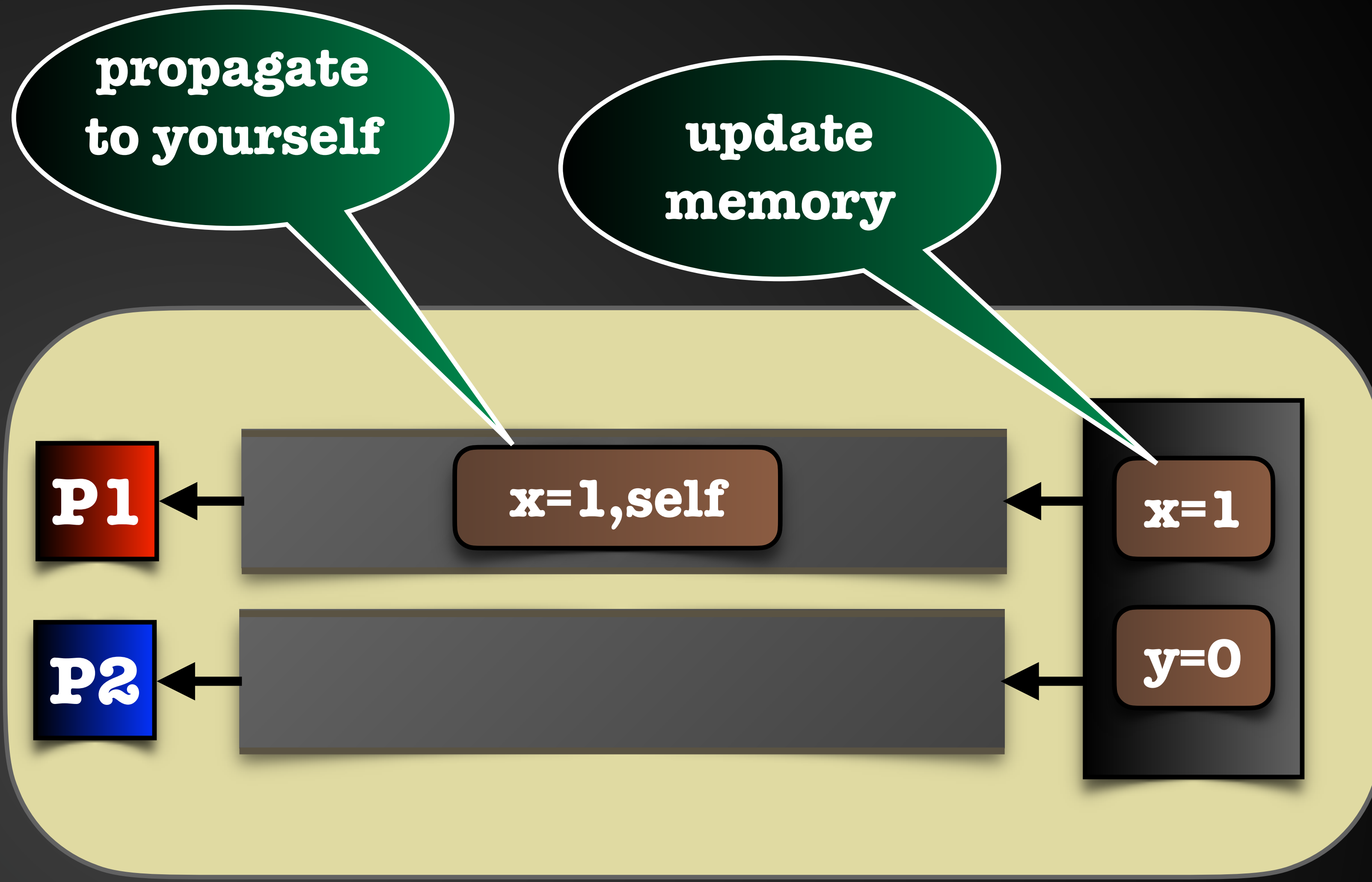


Dual TSO

P1: write: x = 1

▶ P1: read: x = 1

P1: read: y = 0

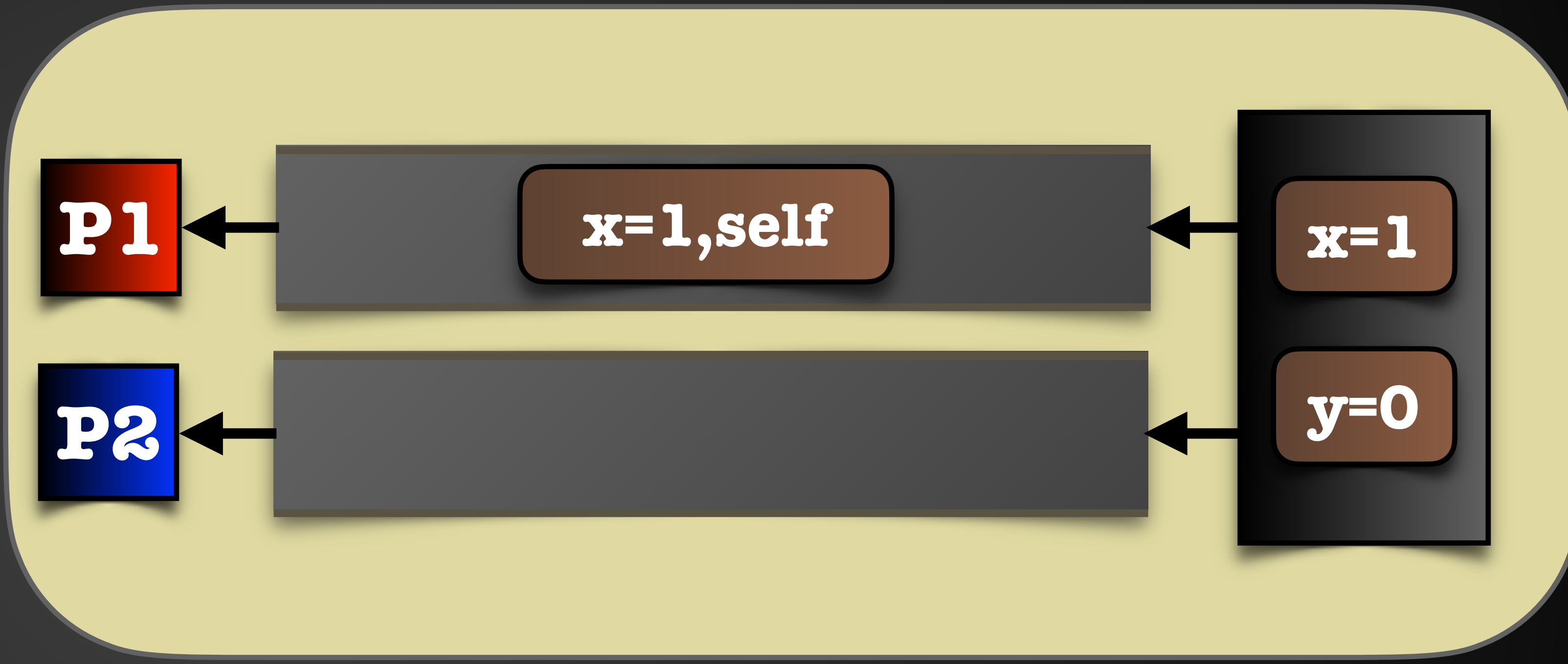


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0

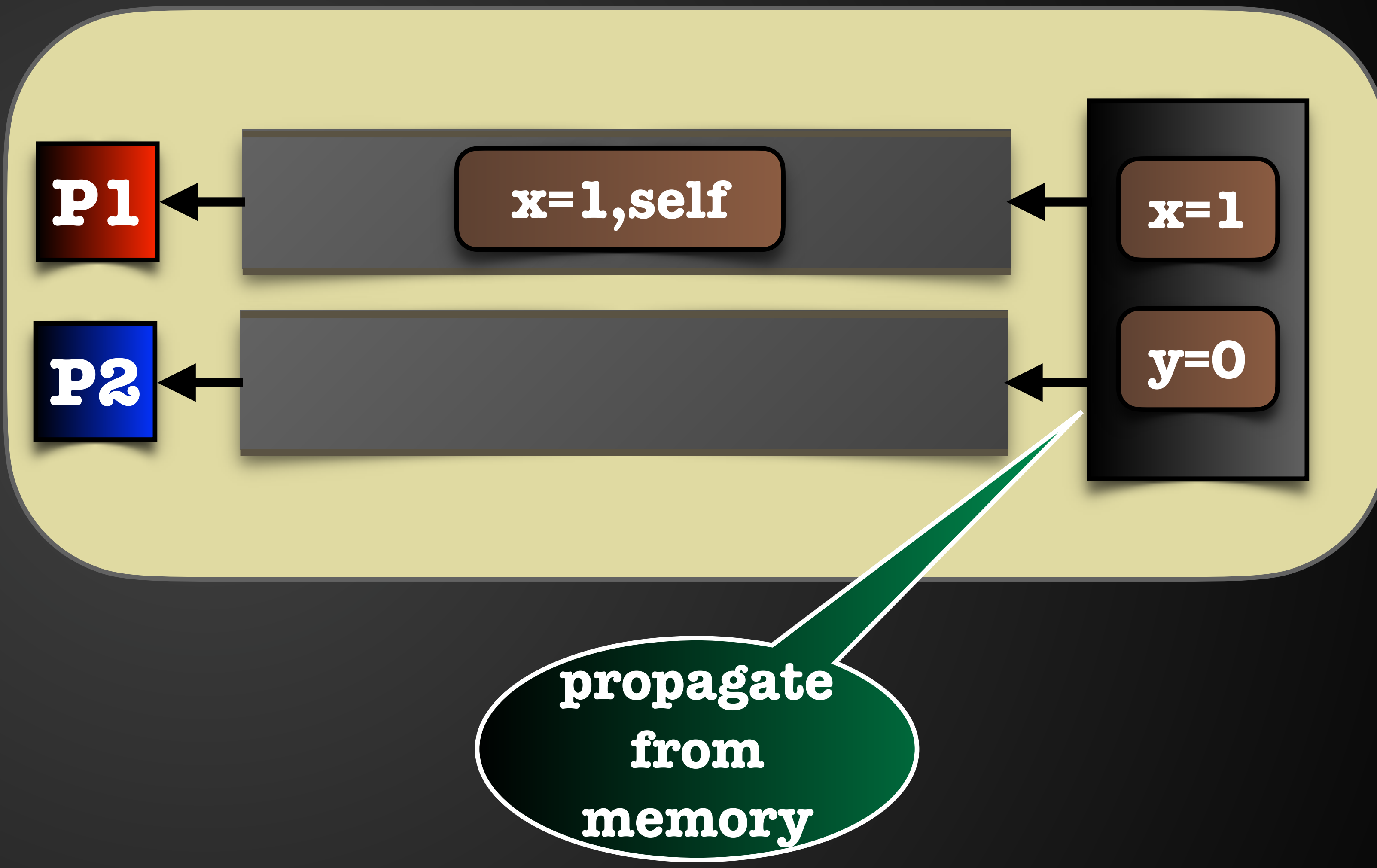


Dual TSO

P1: write: x = 1

▶ P1: read: x = 1

P1: read: y = 0

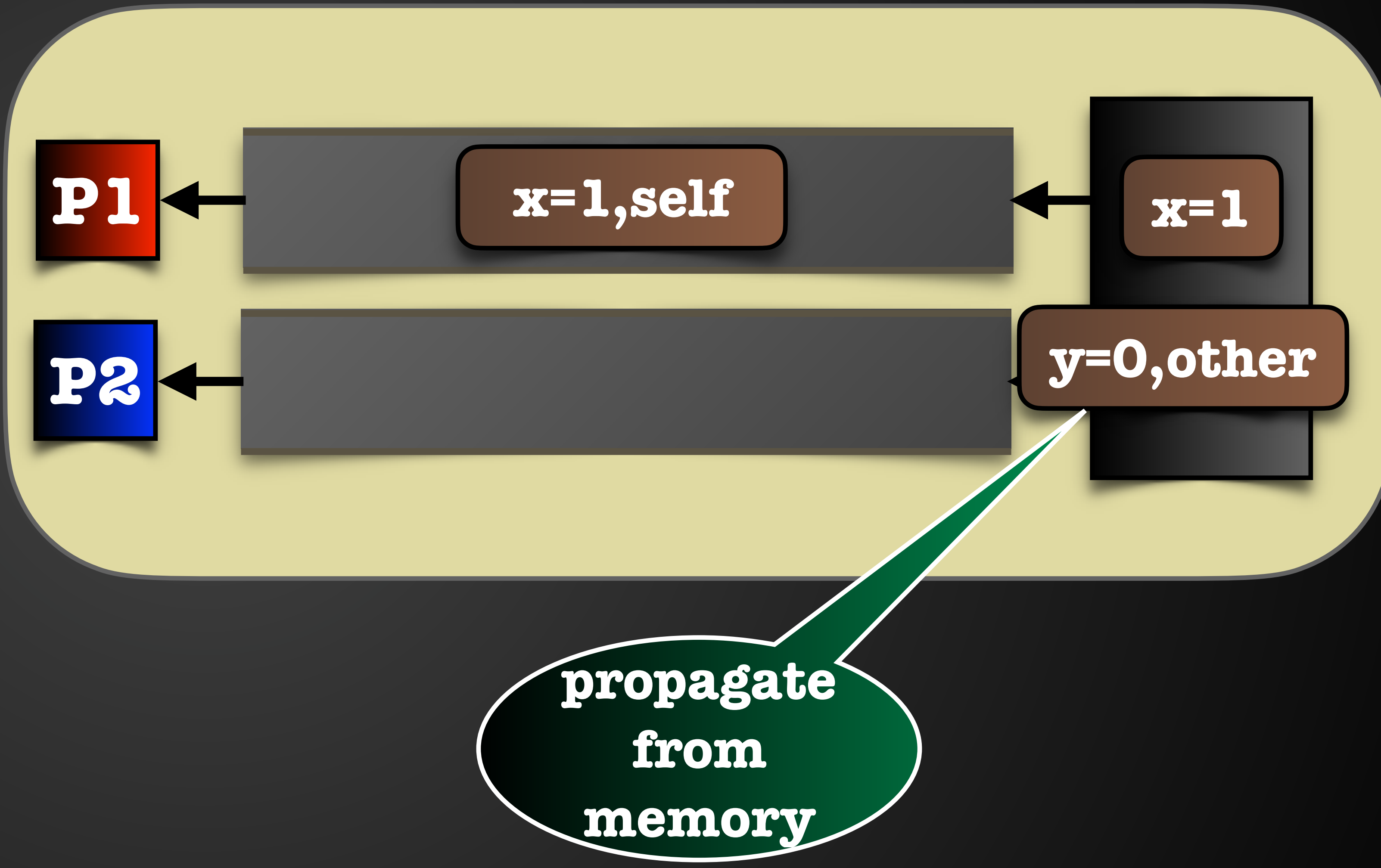


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



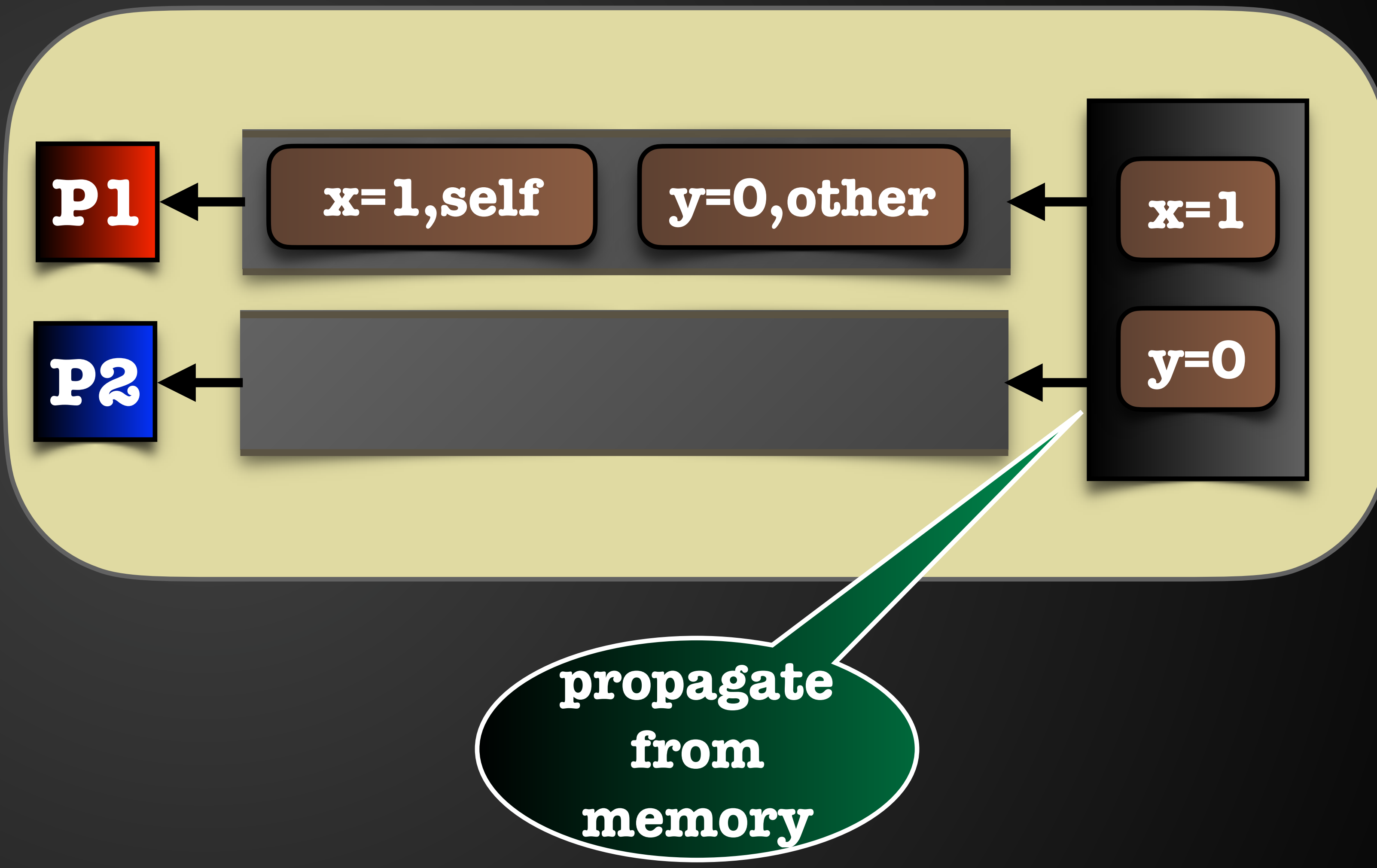
propagate from memory

Dual TSO

P1: write: x = 1

▶ P1: read: x = 1

P1: read: y = 0

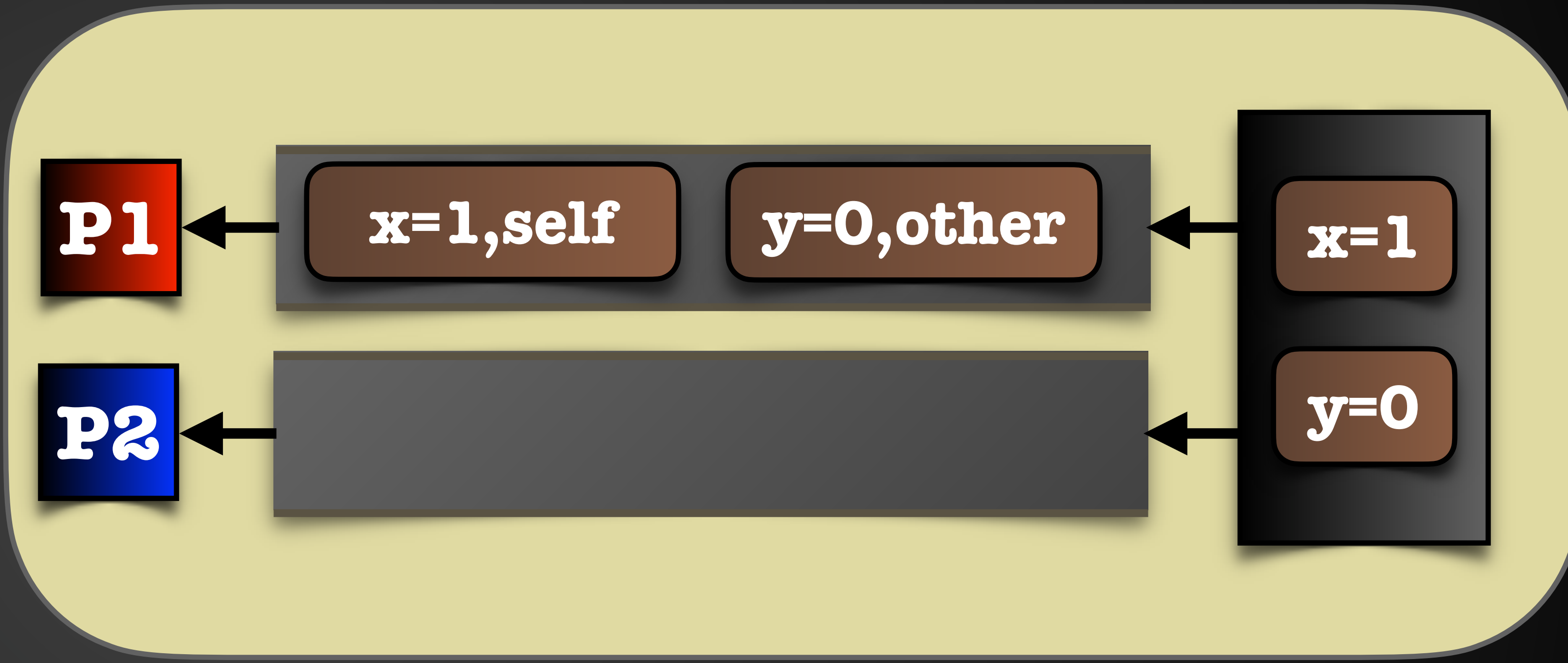


Dual TSO

P1: write: x = 1

P1: read: x = 1

▶ P1: read: y = 0

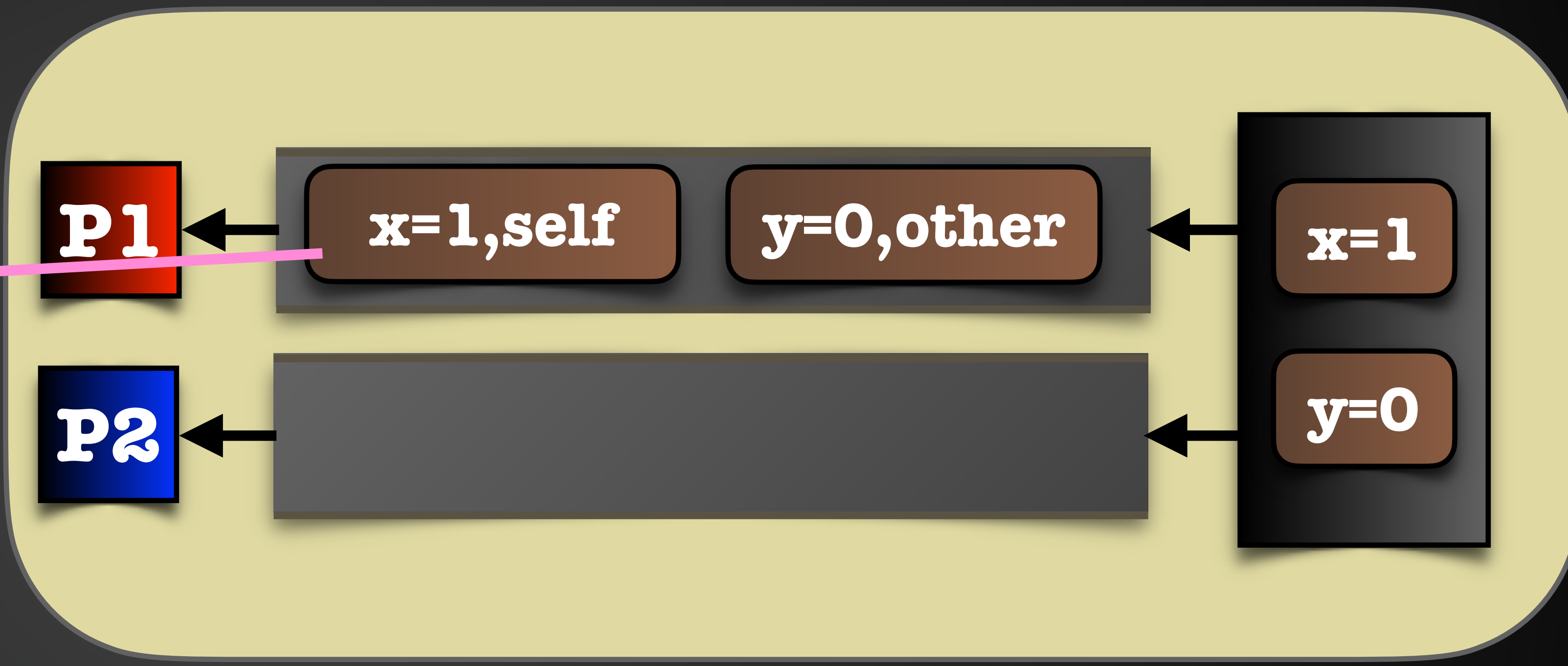


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



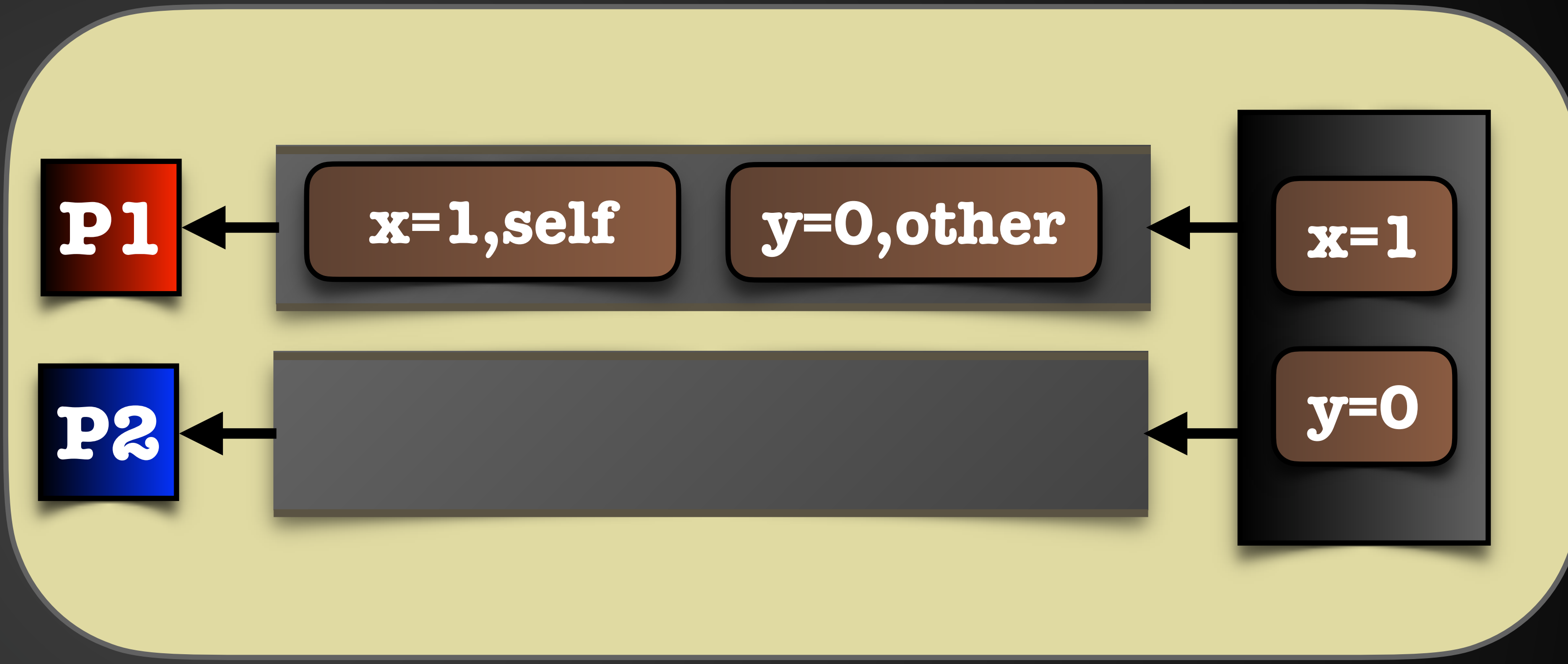
read own write

Dual TSO

P1: write: x = 1

P1: read: x = 1

▶ P1: read: y = 0

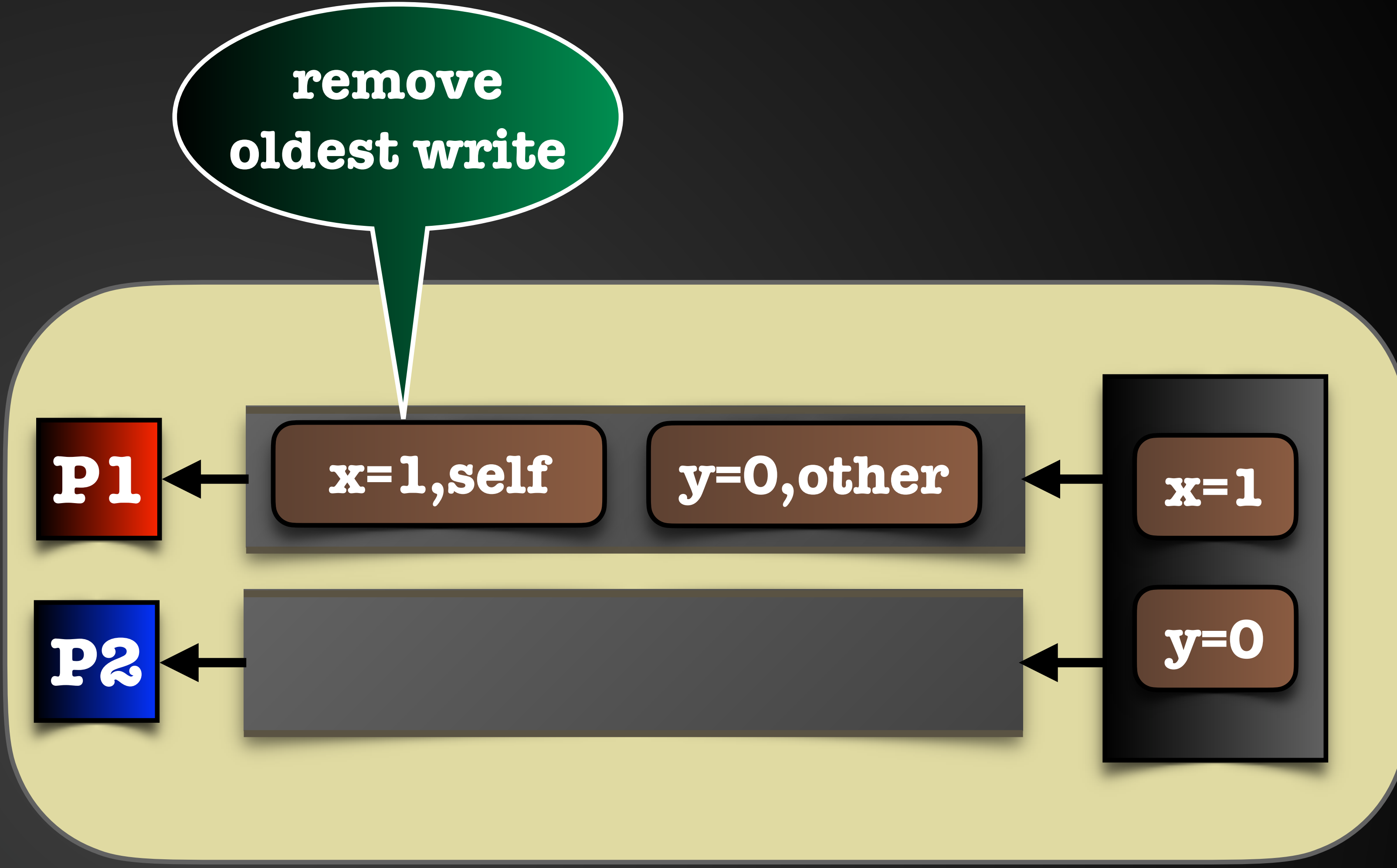


Dual TSO

P1: write: x = 1

P1: read: x = 1

▶ P1: read: y = 0

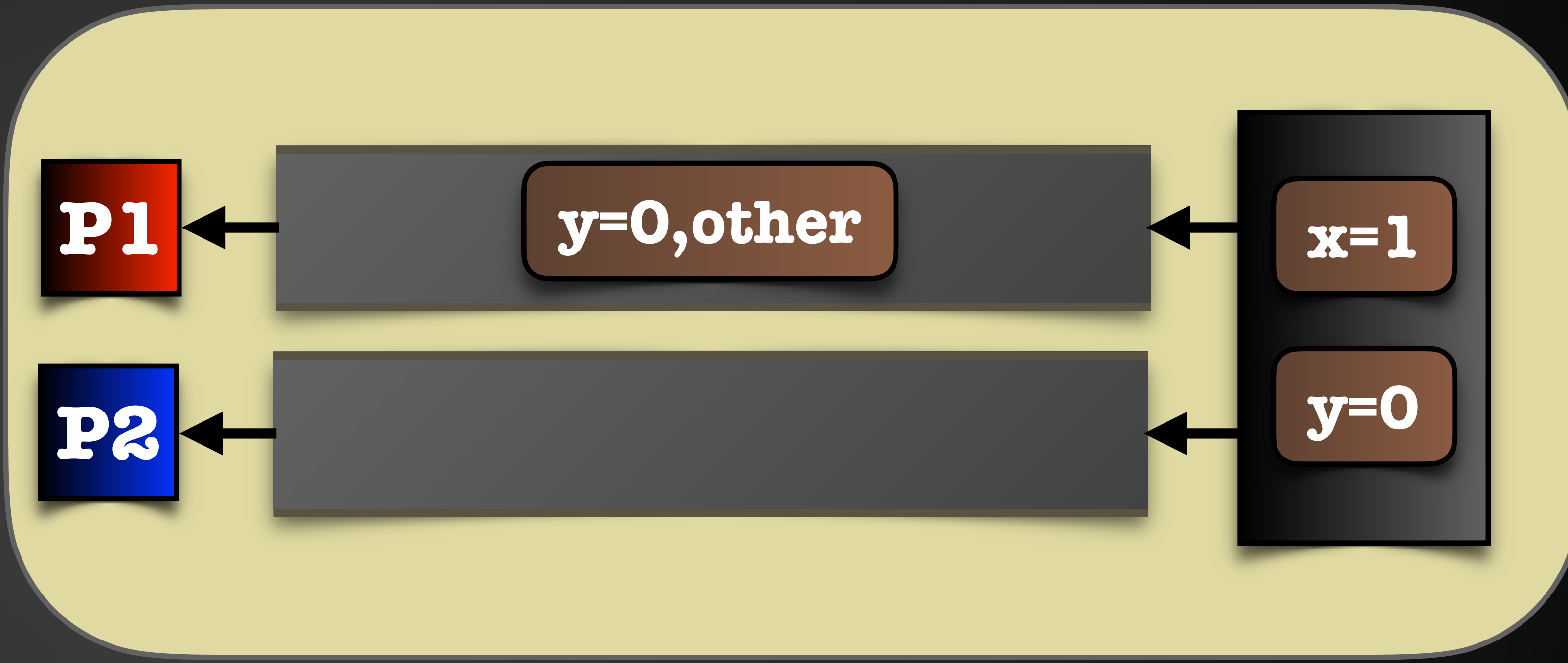


Dual TSO

P1: write: x = 1

P1: read: x = 1

▶ P1: read: y = 0

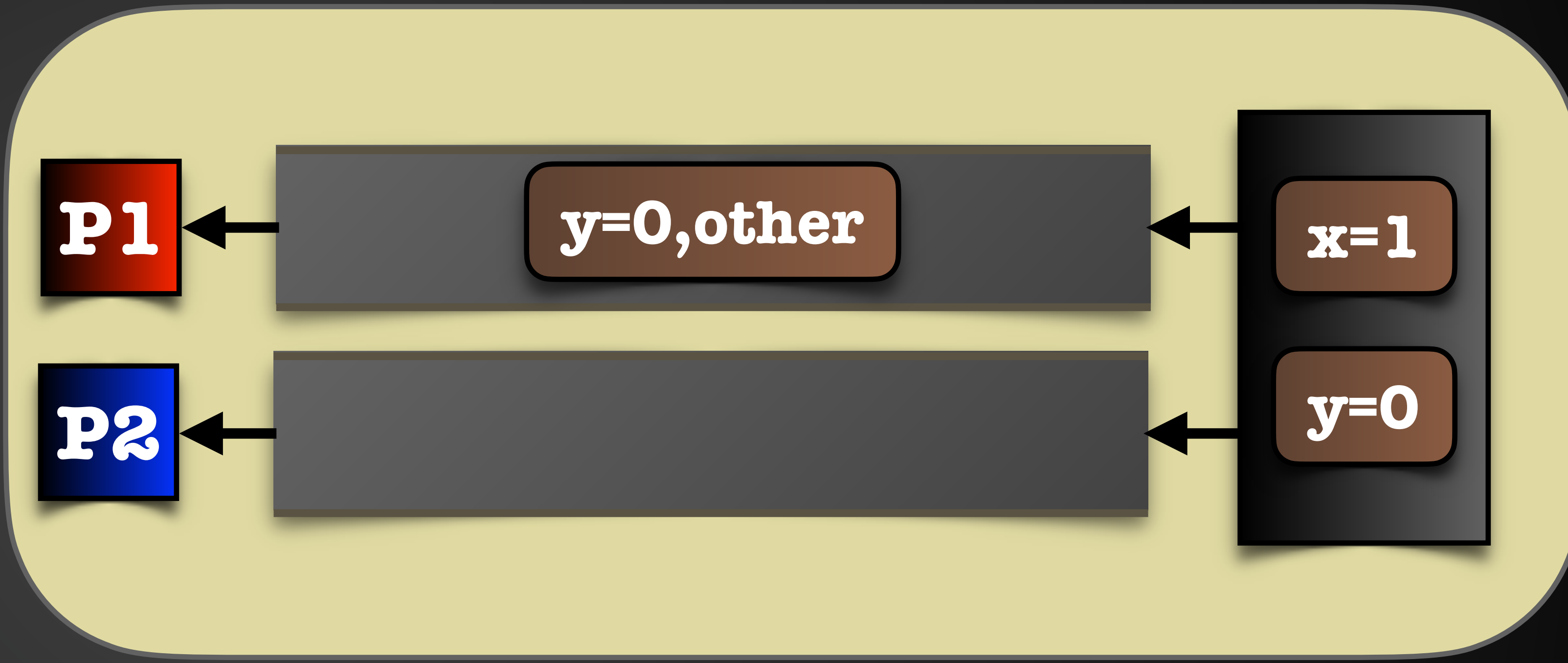


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



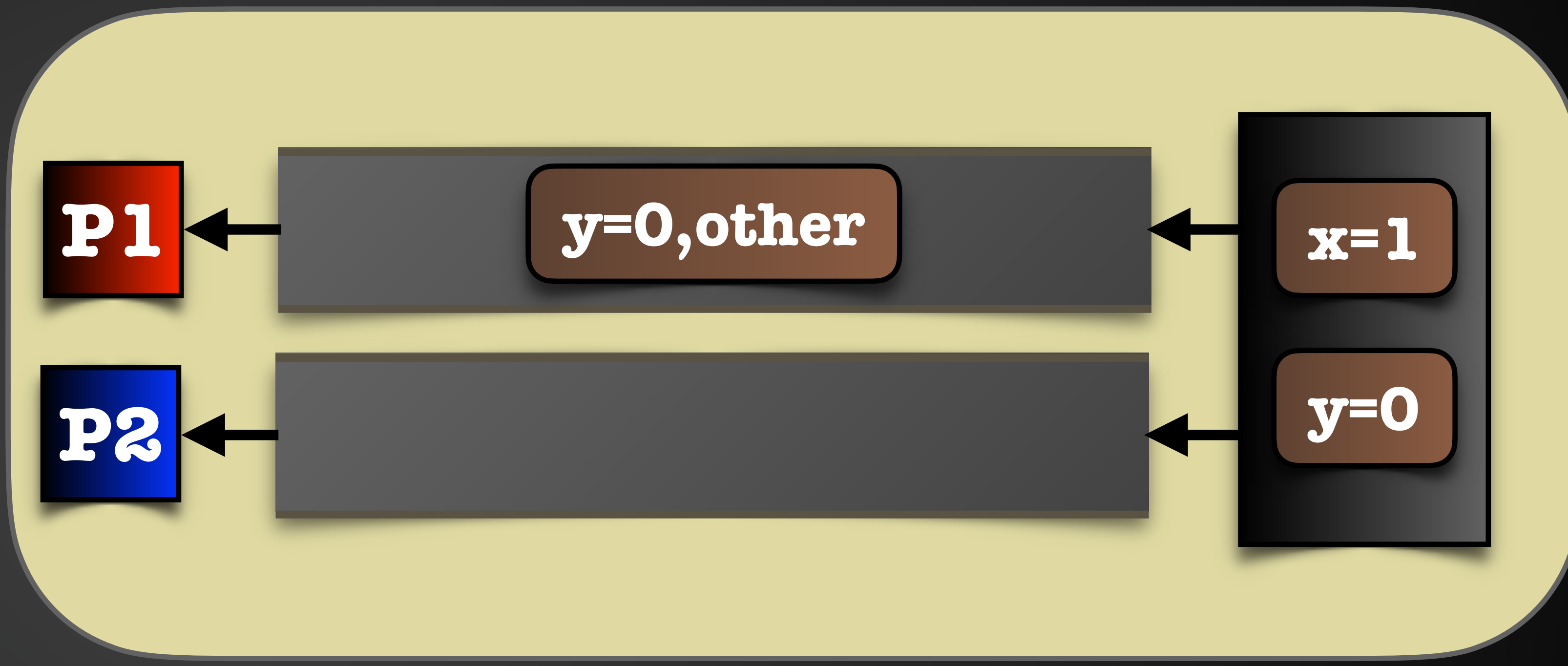
read oldest write

Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



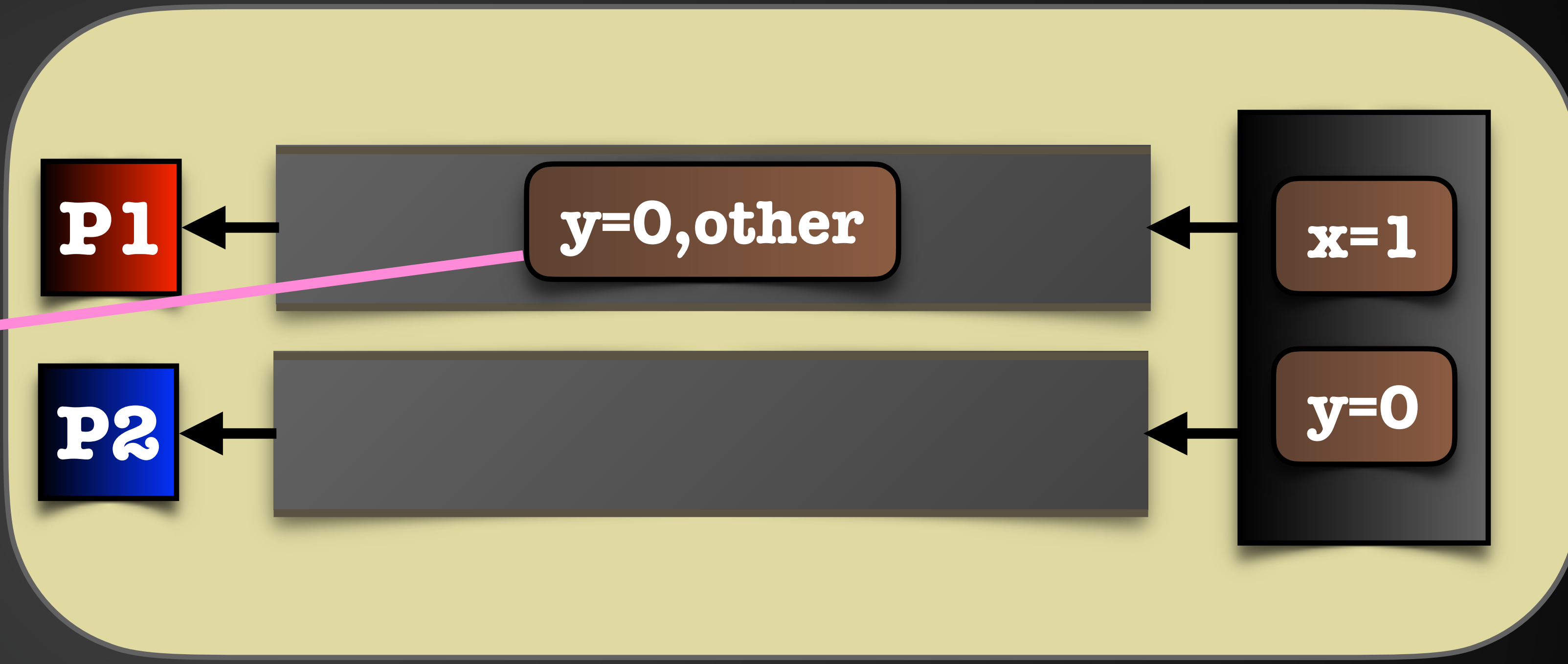
read oldest write

Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0

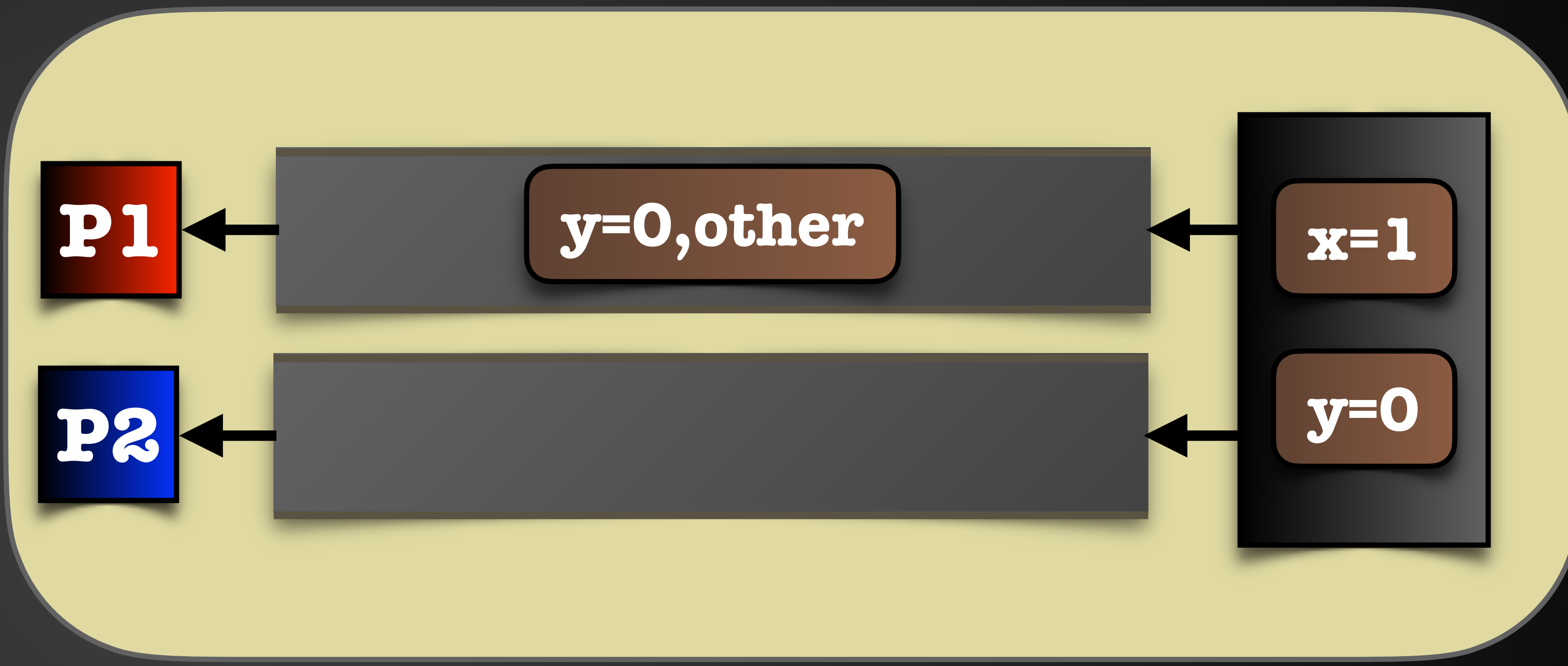


Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



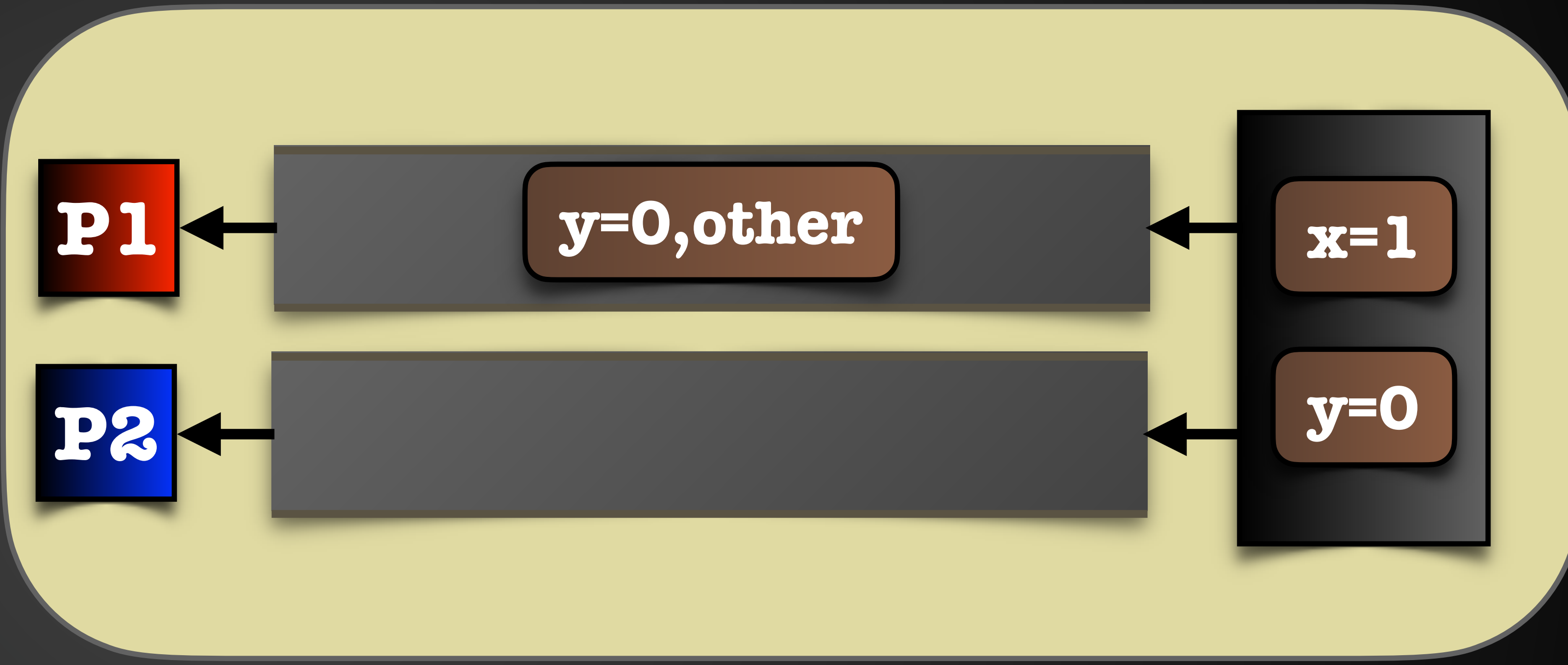
- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



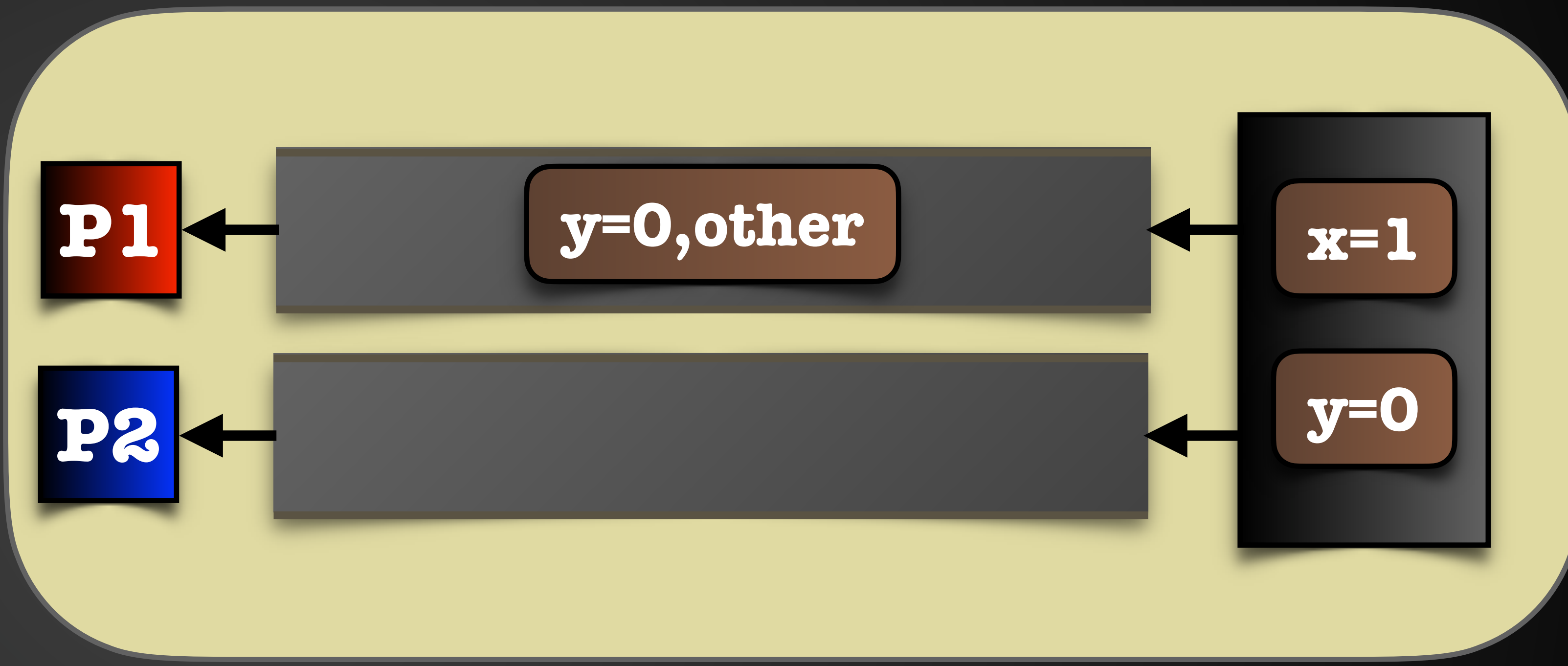
- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

Dual TSO

P1: write: x = 1

P1: read: x = 1

P1: read: y = 0



- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

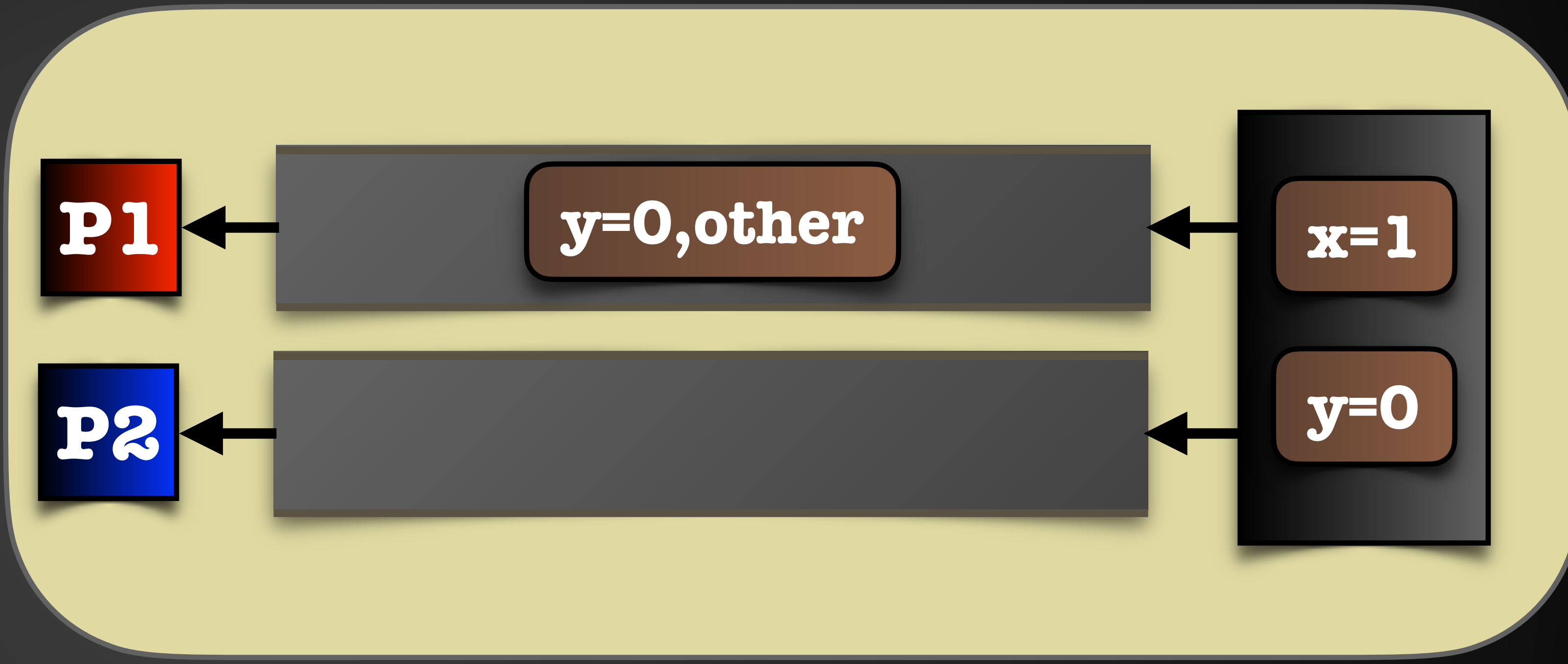
TSO ≡ Dual-TSO

Dual TSO

P1: write: x = 1

P1: read: x = 1

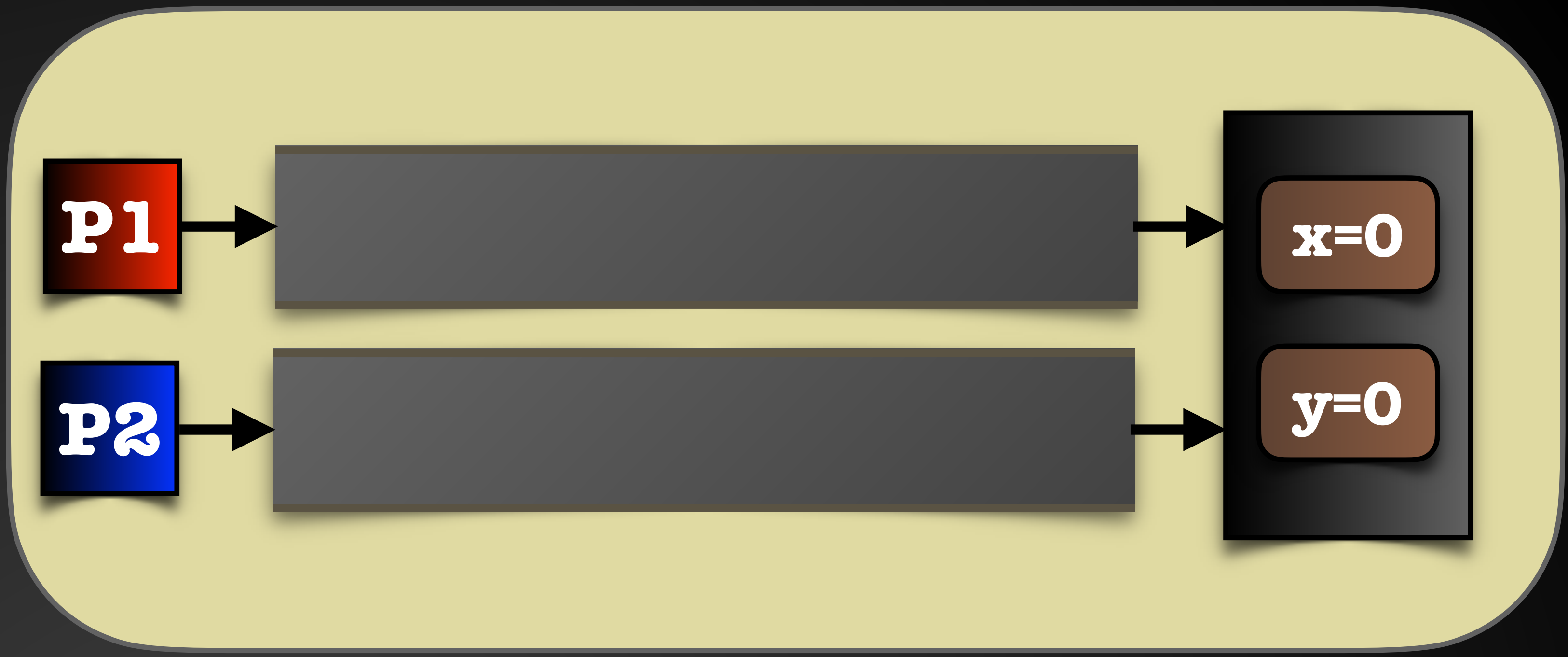
P1: read: y = 0



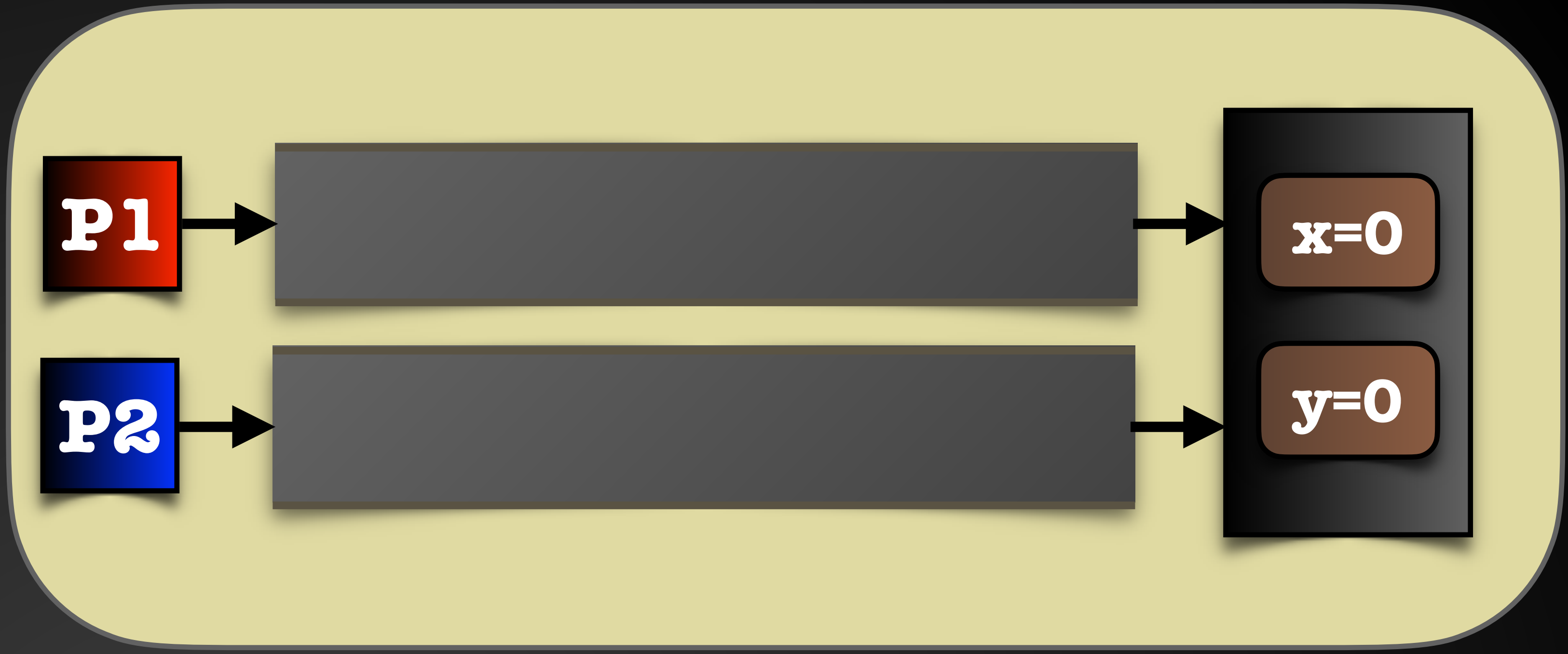
- write + self-propagation
- propagate from memory
- read own-writes
- read oldest write
- remove oldest write

TSO ≡ Dual-TSO

reachability

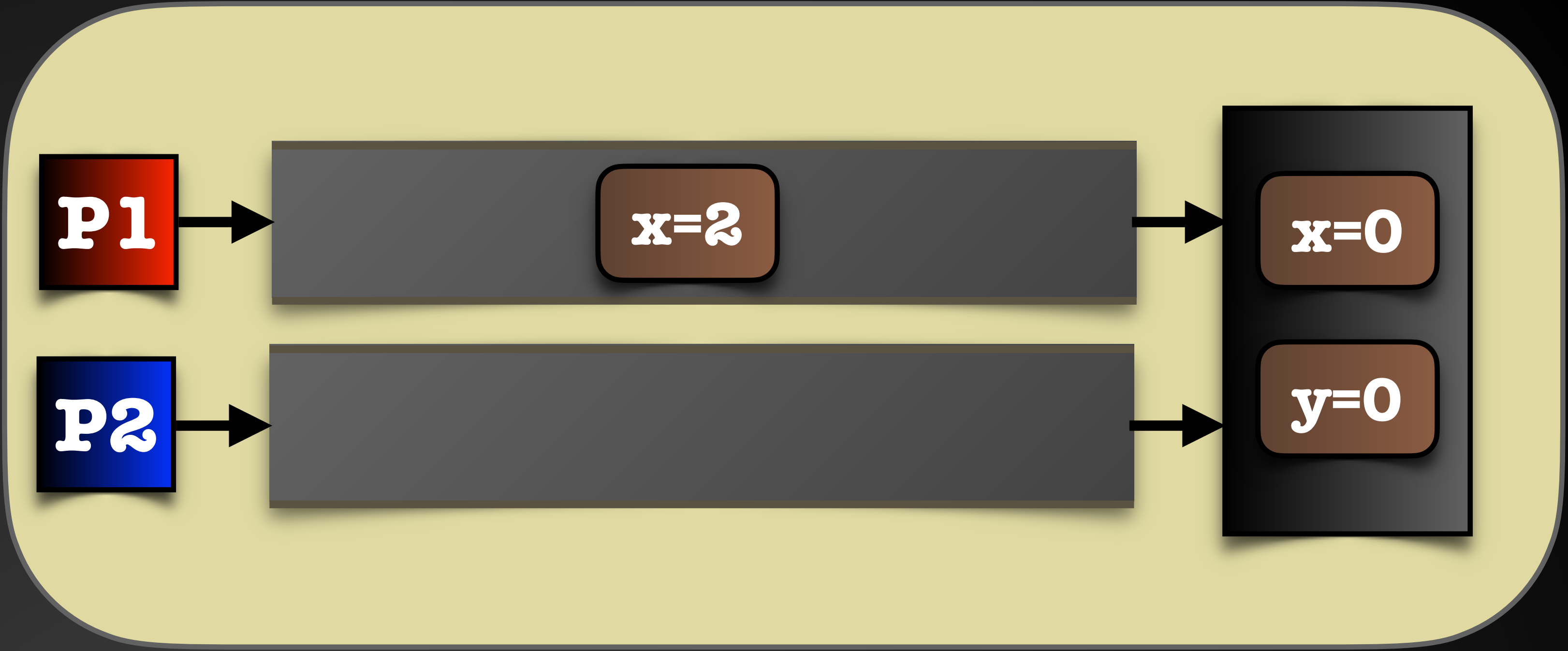


Classical TSO



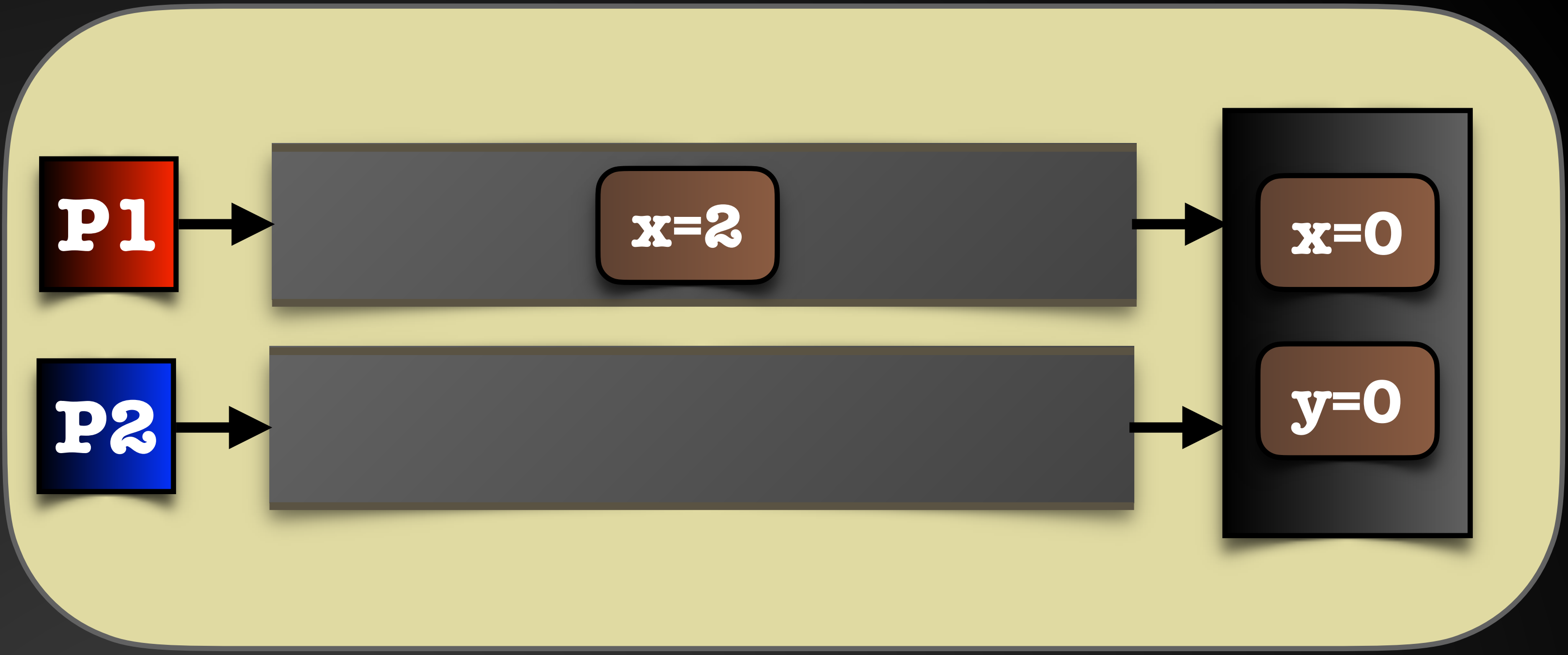
P1: w(x,2)

Classical TSO



P1: w(x,2)

Classical TSO



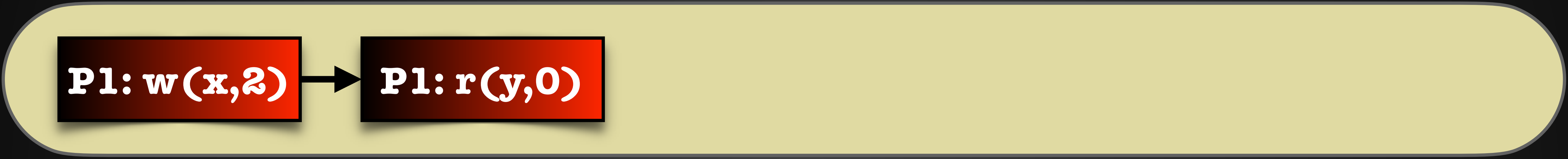
P1

x=2

P2

x=0

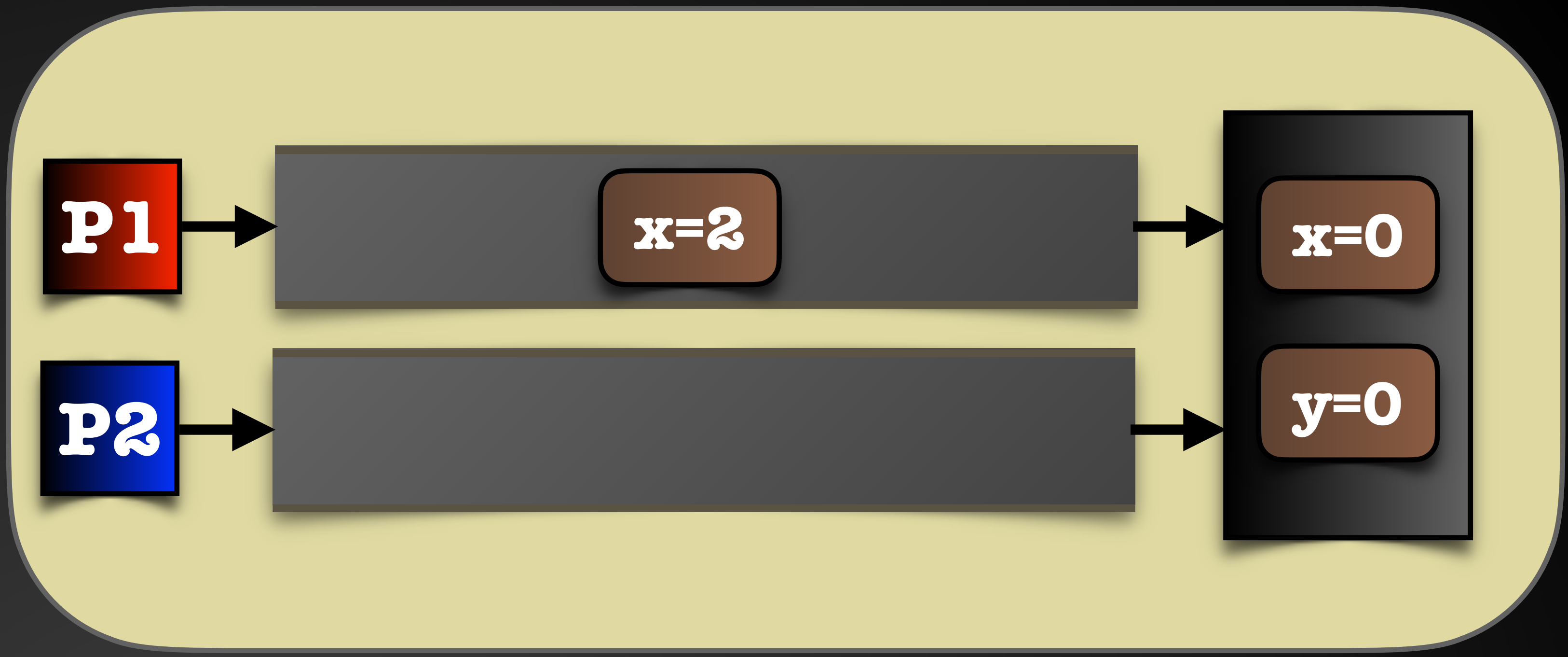
y=0



P1: w(x,2)

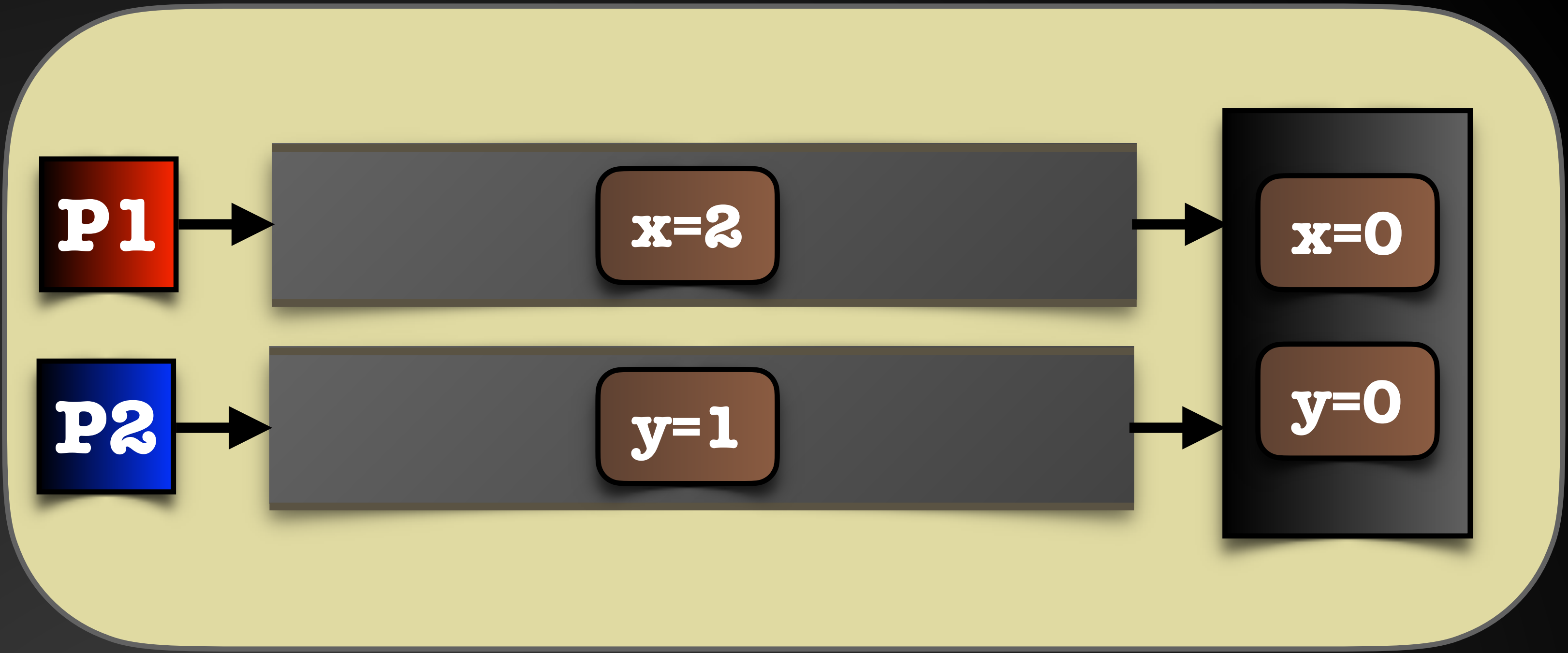
P1: r(y,0)

Classical TSO



P1: w(x,2) → P1: r(y,0) → P2: w(y,1)

Classical TSO



P1

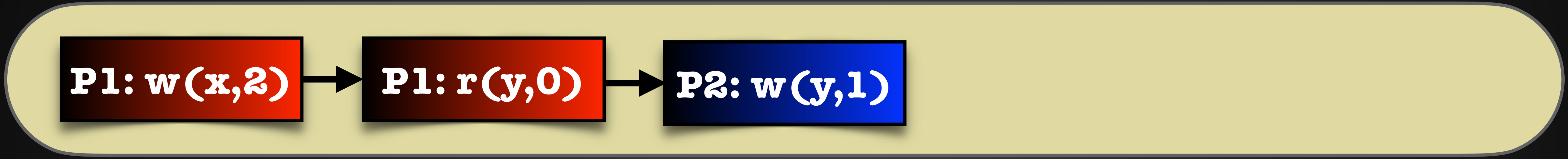
x=2

P2

y=1

x=0

y=0

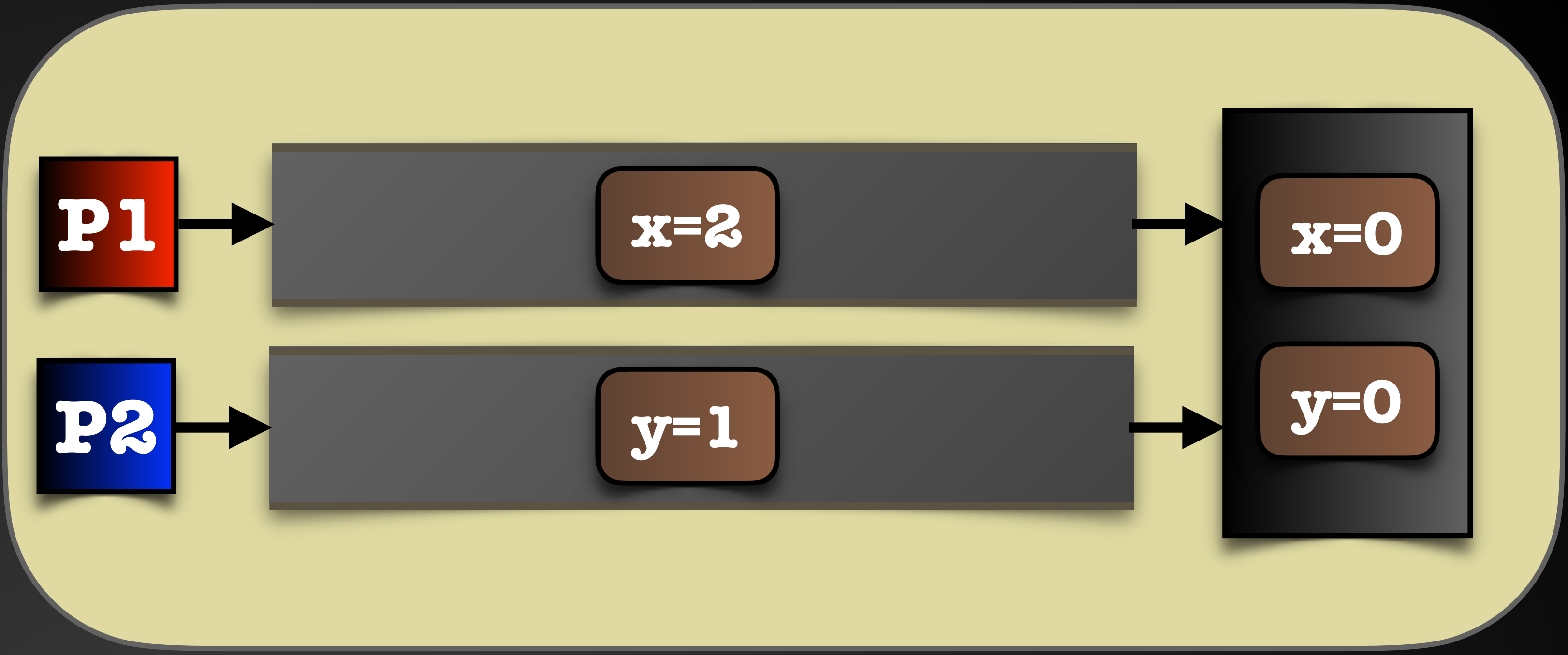


P1: w(x,2)

P1: r(y,0)

P2: w(y,1)

Classical TSO



P1

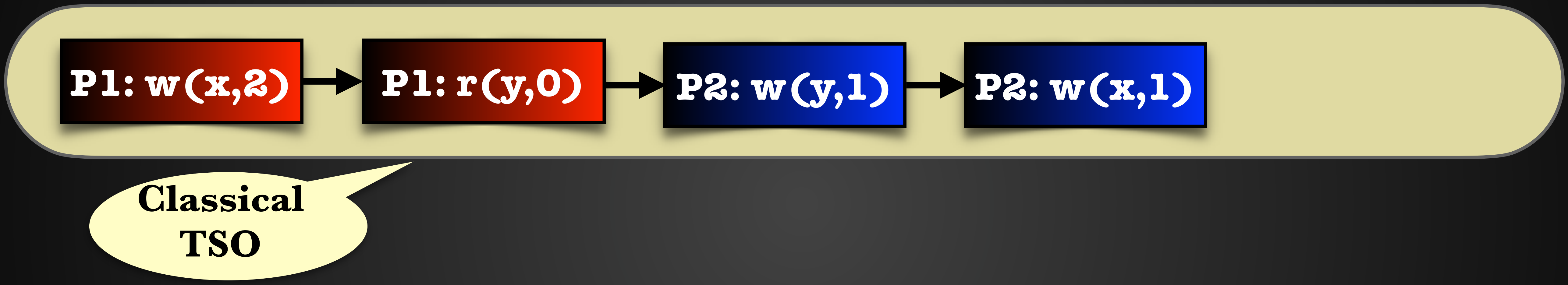
x=2

x=0

P2

y=1

y=0



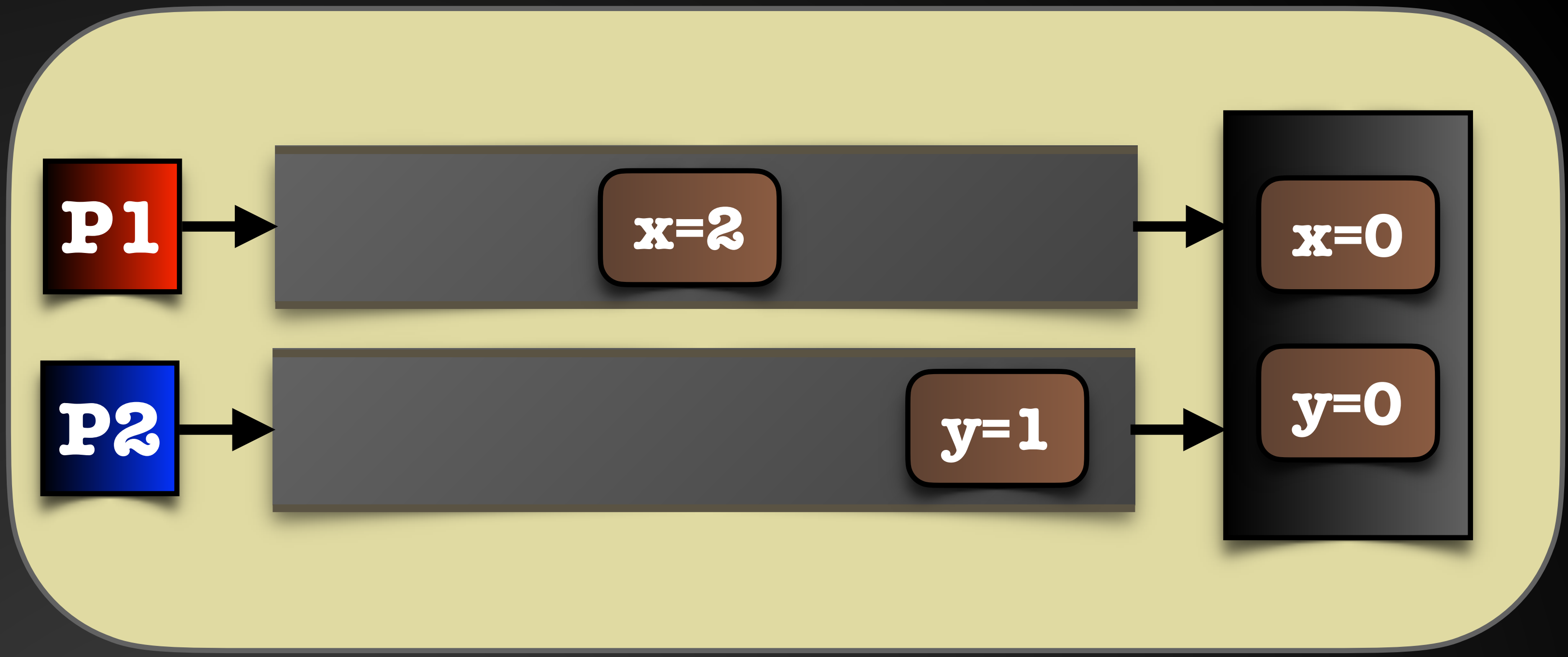
P1: w(x,2)

P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

Classical TSO



P1

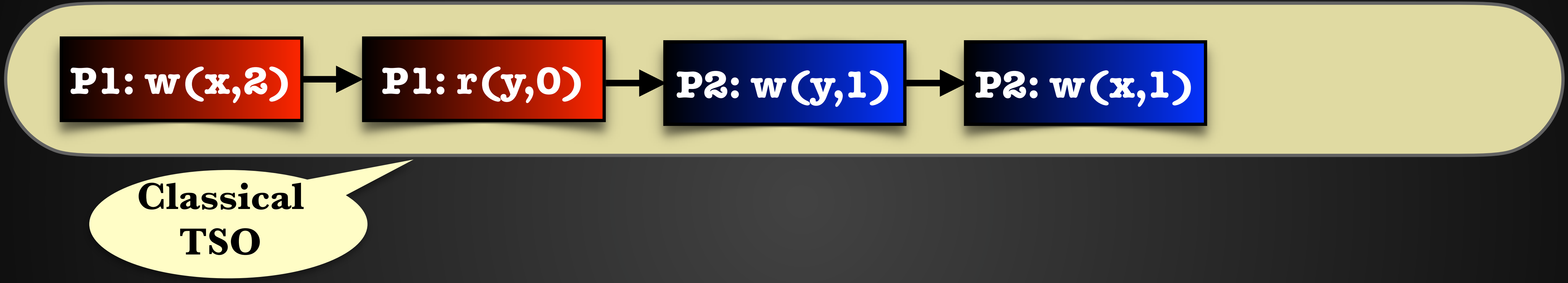
x=2

x=0

P2

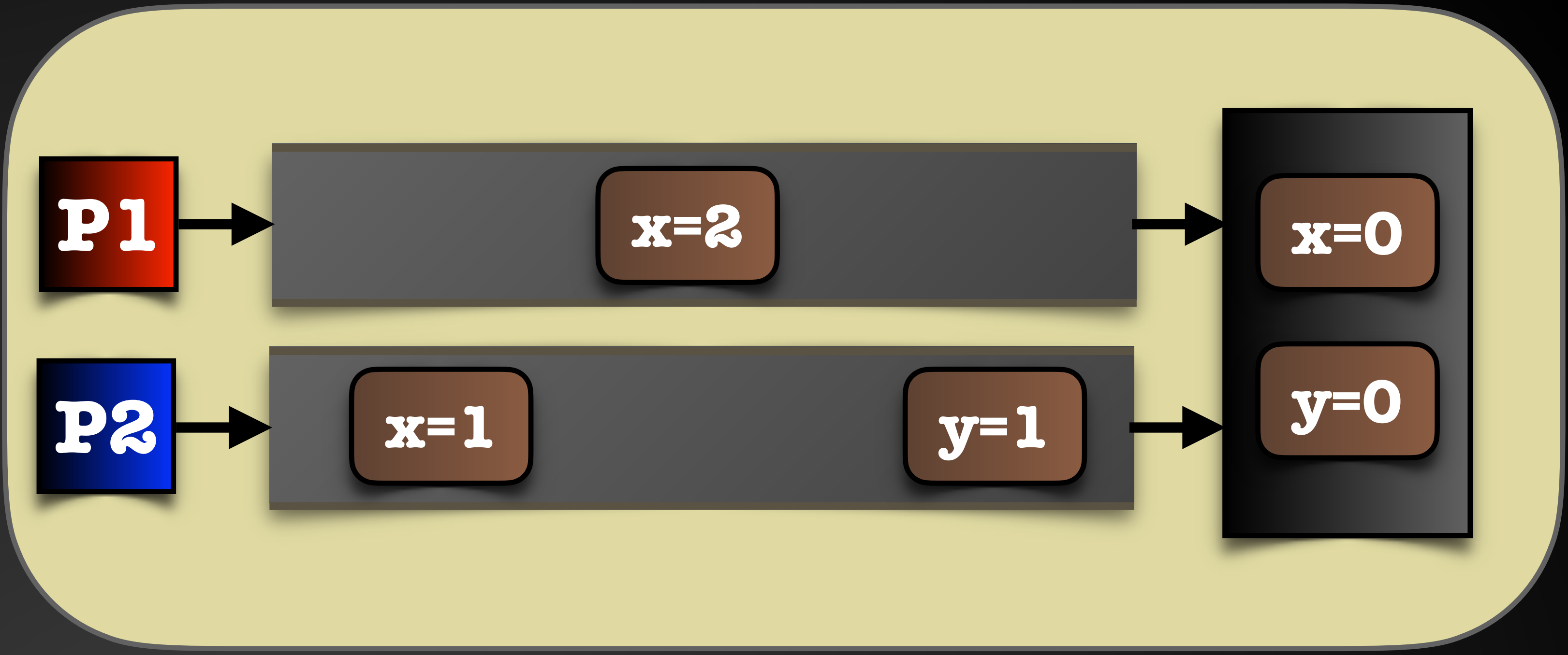
y=1

y=0



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1)

Classical TSO



P1

x=2

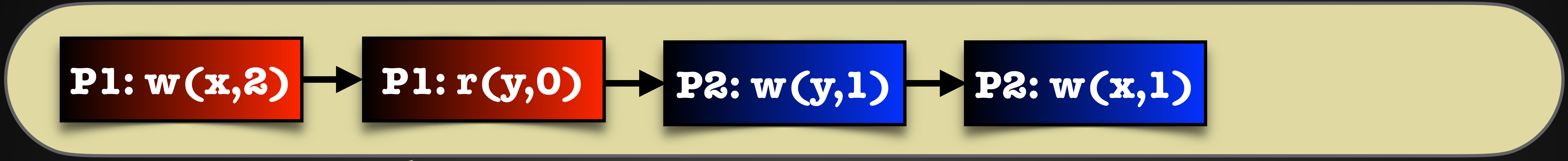
x=0

P2

x=1

y=1

y=0



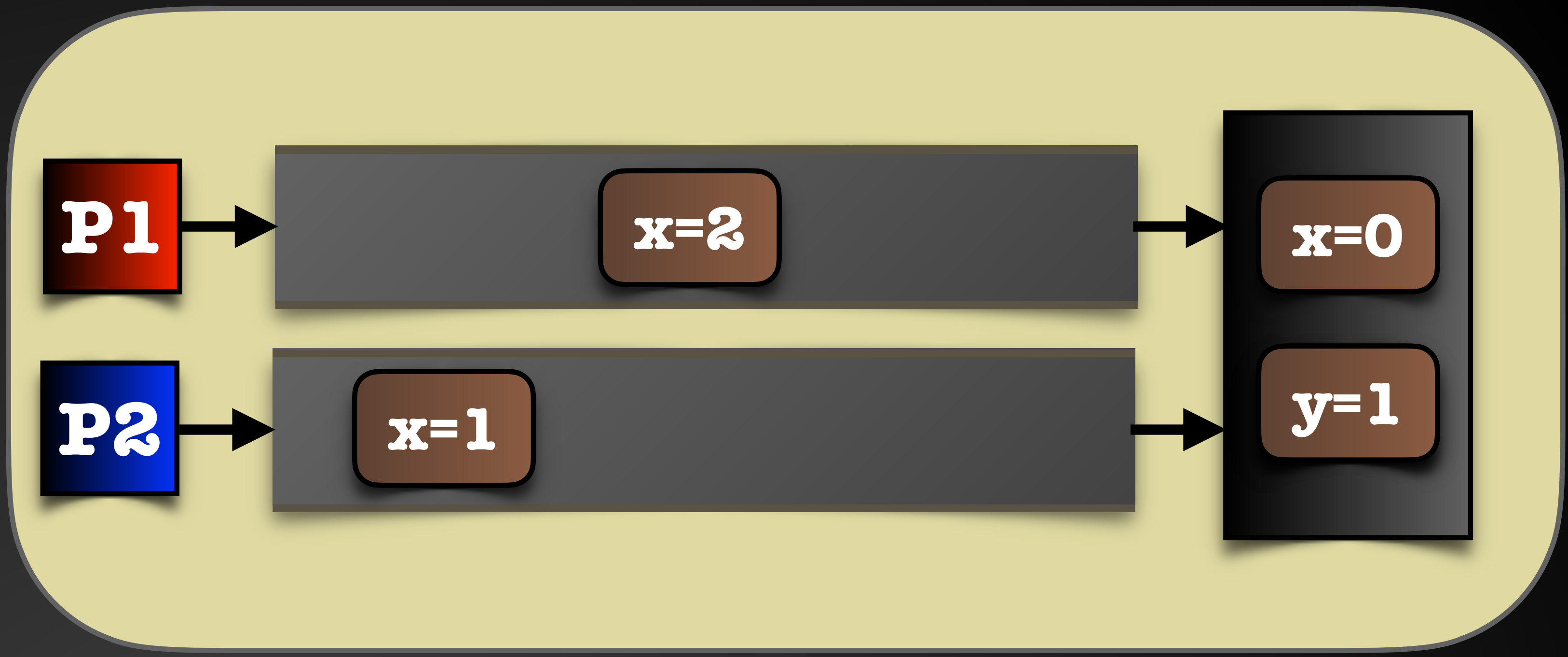
P1: w(x,2)

P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

Classical TSO



P1

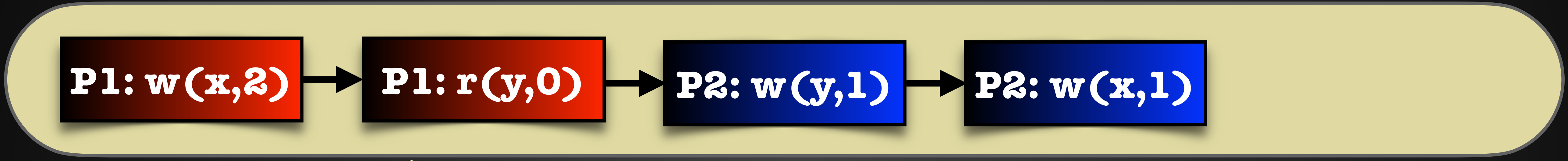
x=2

x=0

P2

x=1

y=1



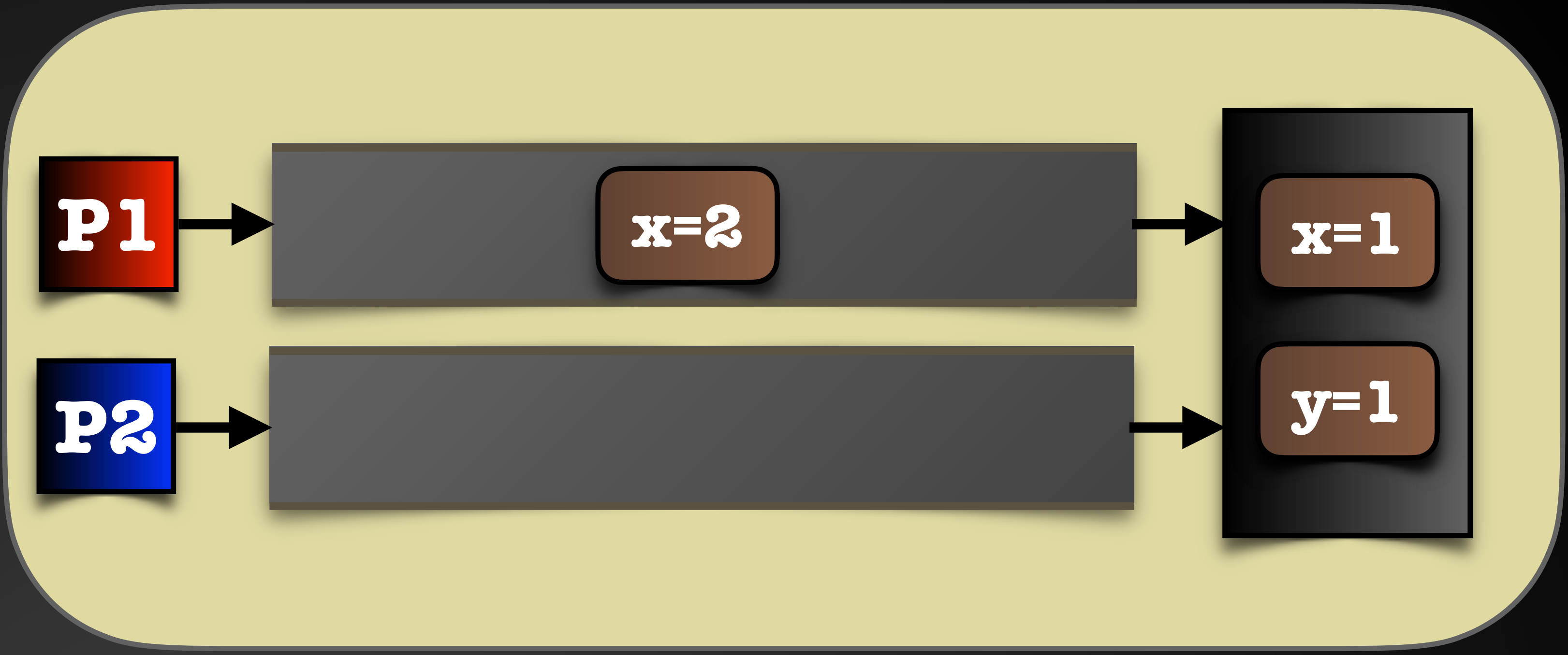
P1: w(x,2)

P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

Classical TSO



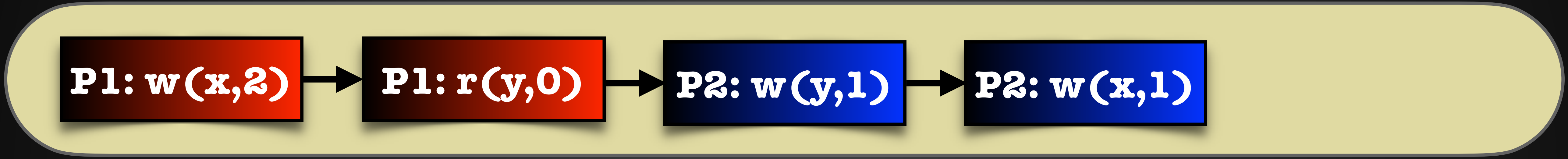
P1

x=2

P2

x=1

y=1



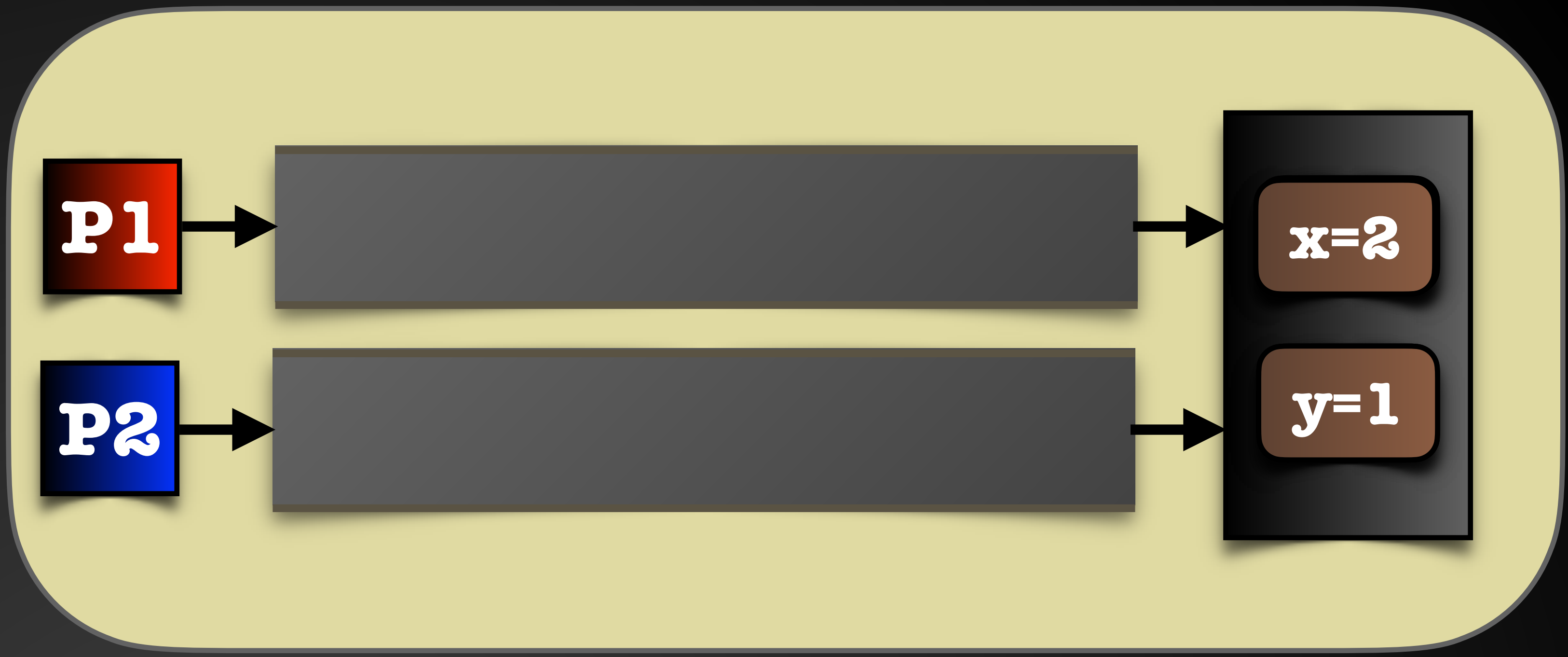
P1: w(x,2)

P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

Classical TSO

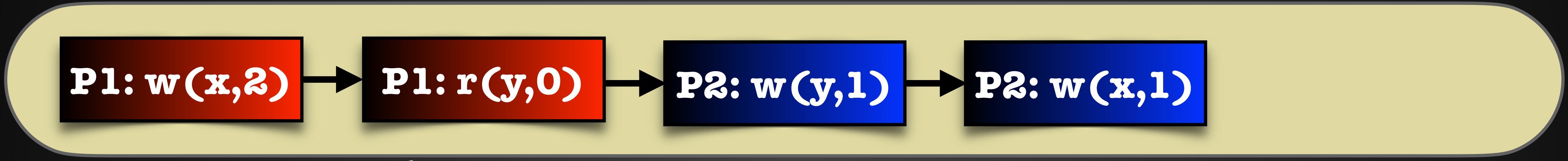


P1

P2

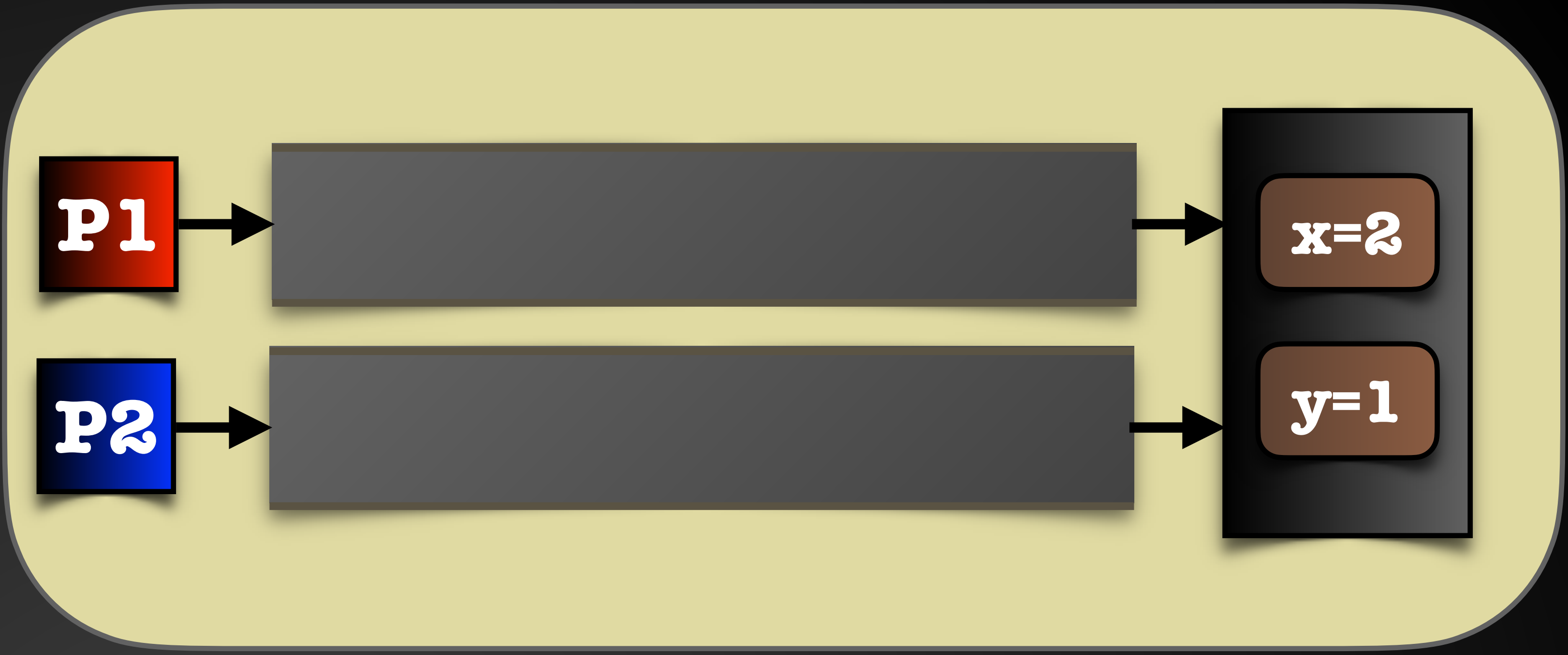
x=2

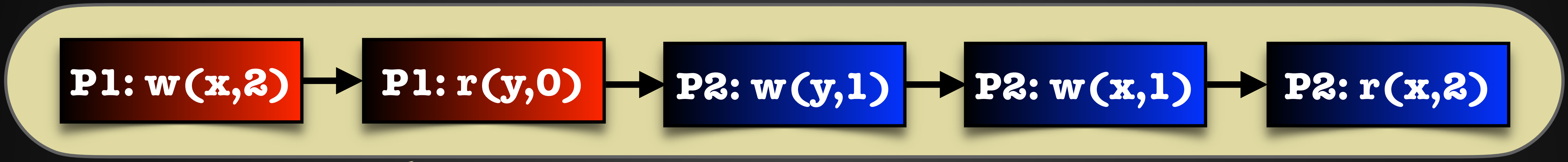
y=1



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1)

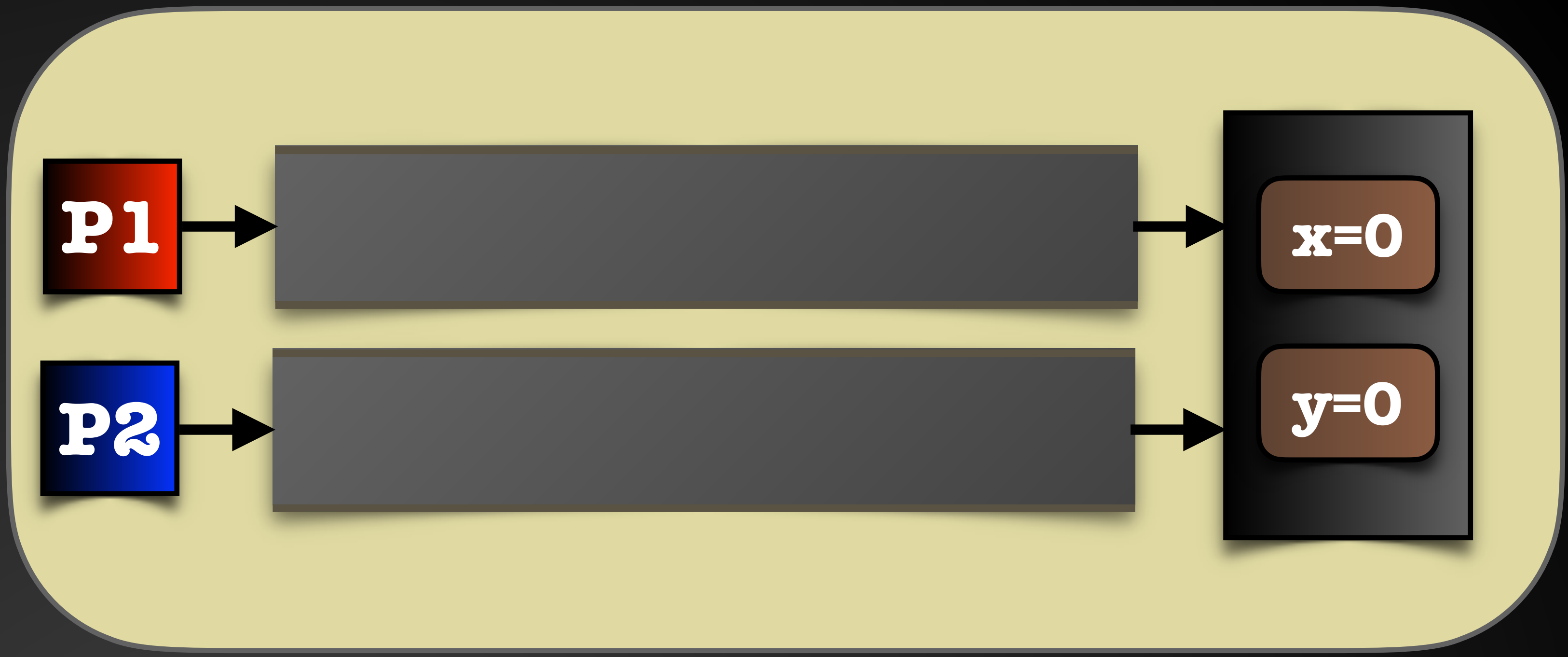
Classical TSO

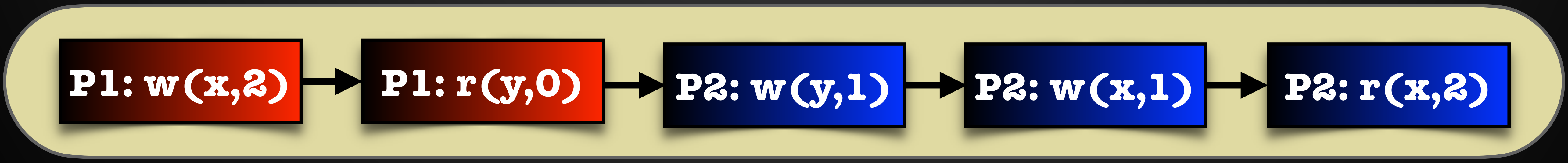




P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

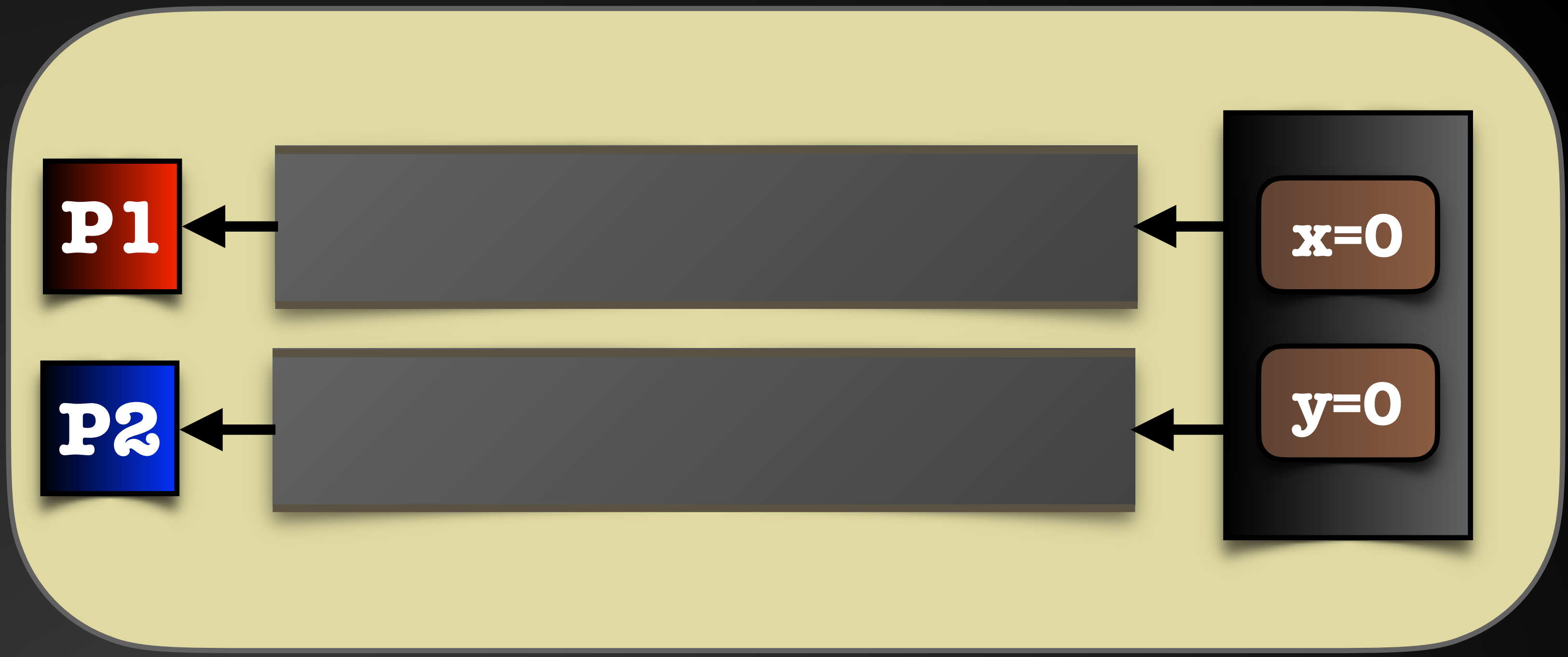
Classical TSO



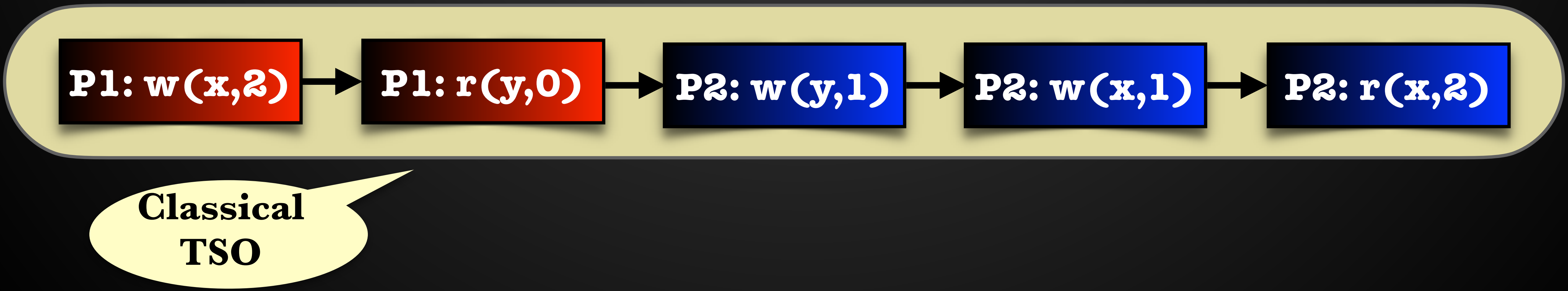


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

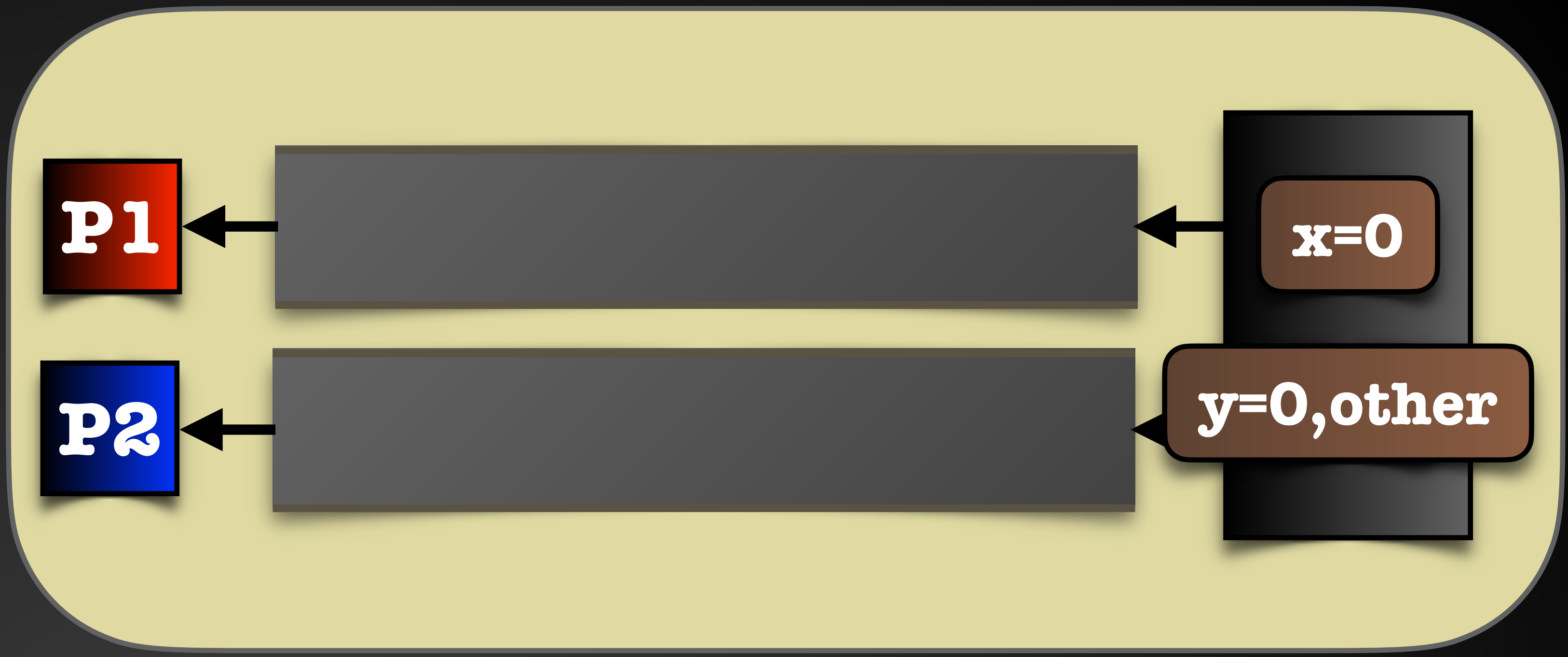
Classical TSO



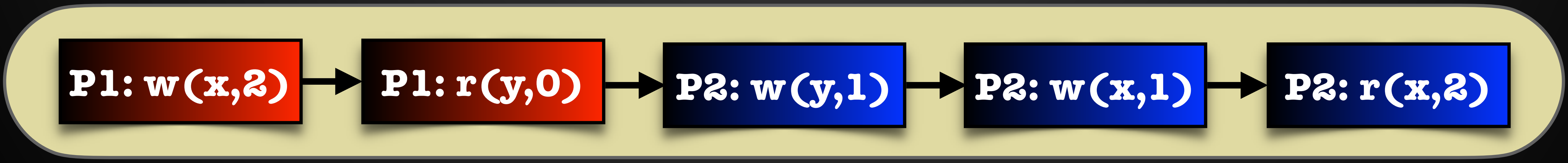
Dual TSO


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

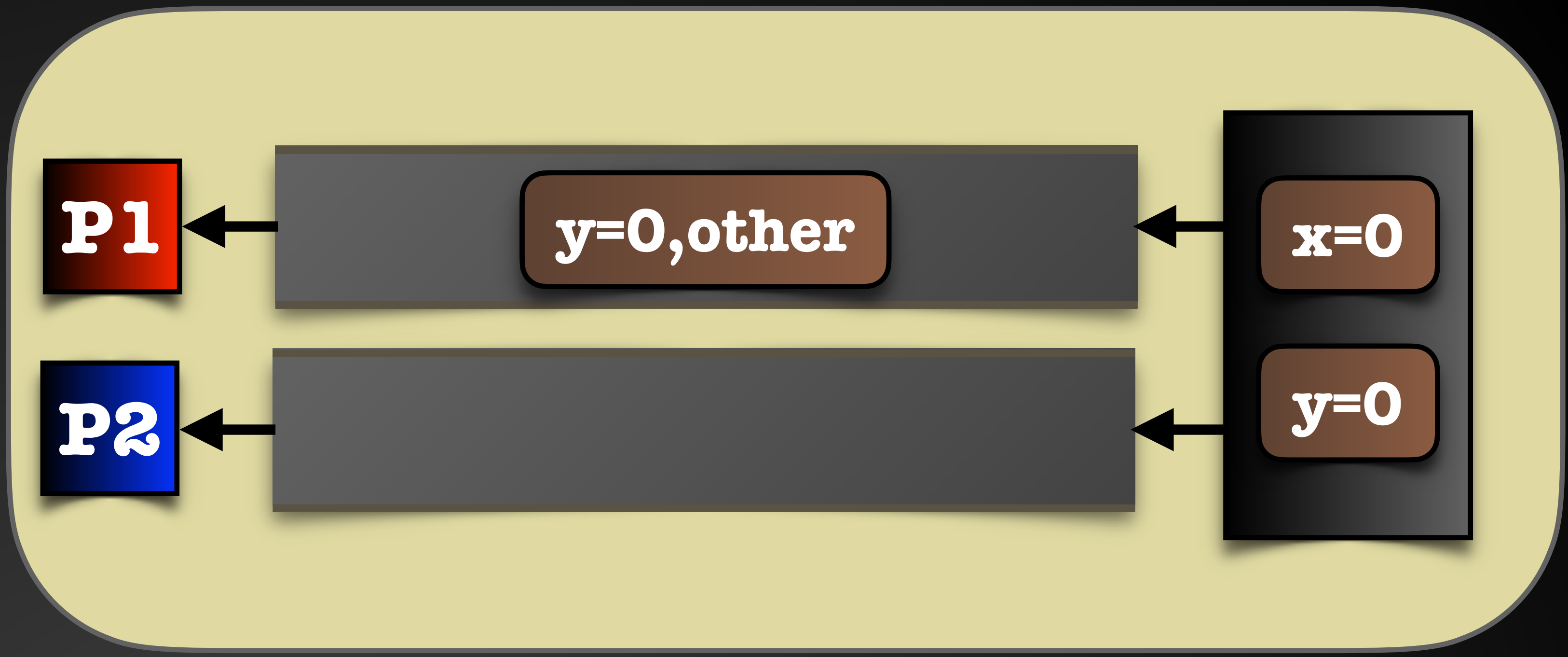


Dual TSO



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

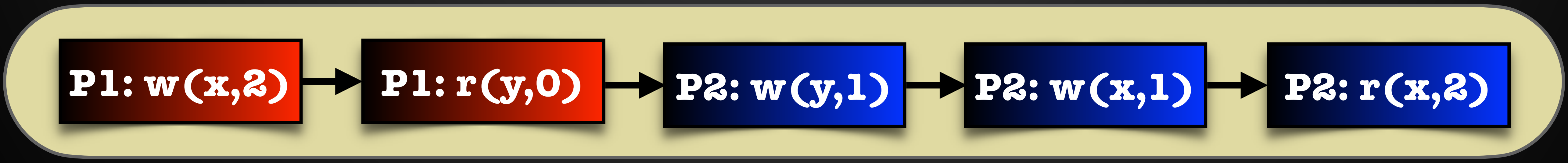
y=0,other

x=0

P2

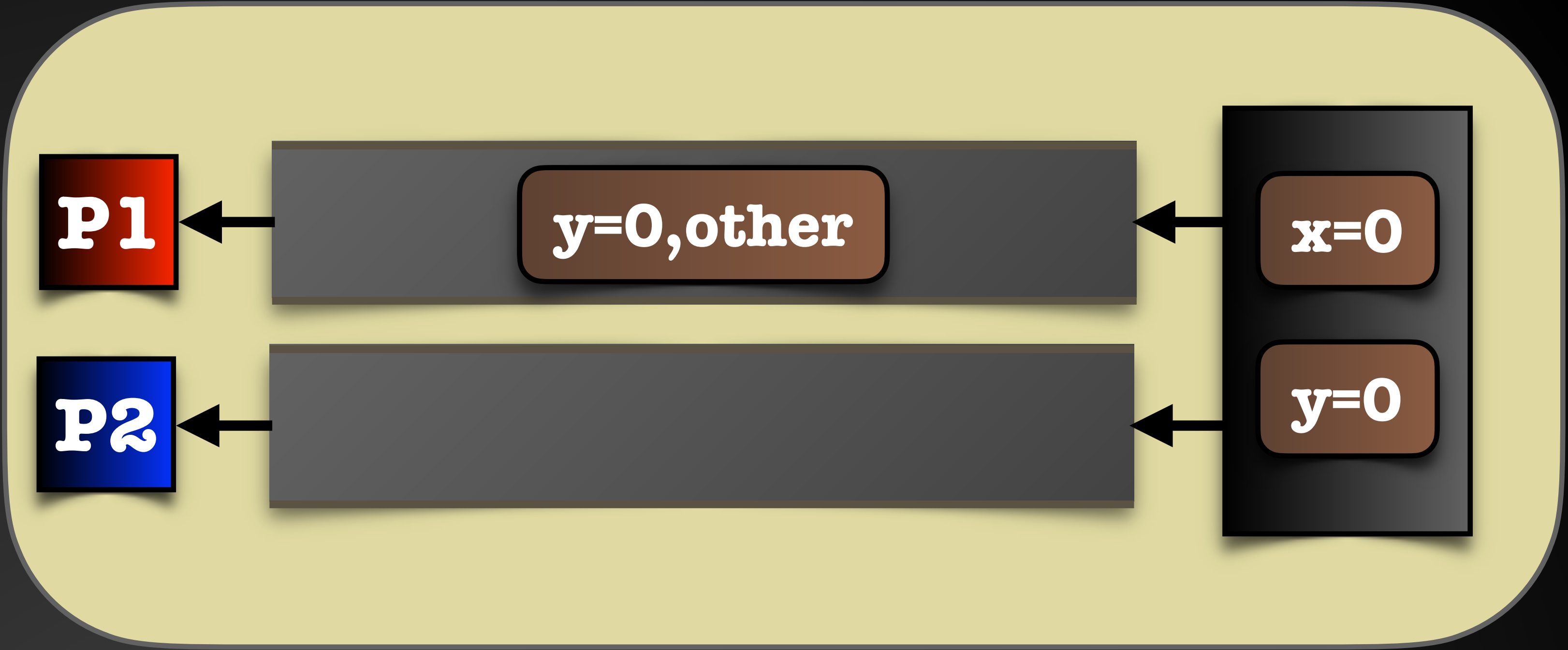
y=0

Dual TSO



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO


P1

y=0,other

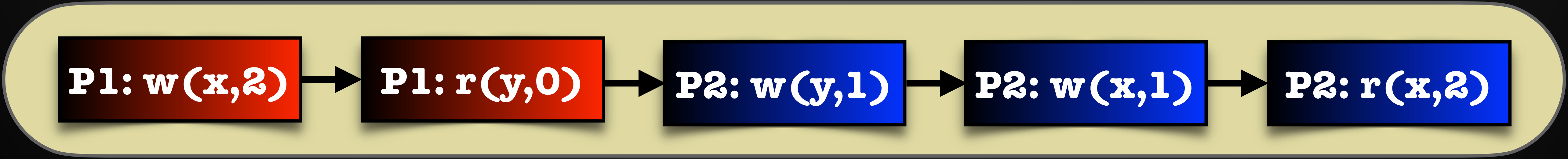
x=0

P2

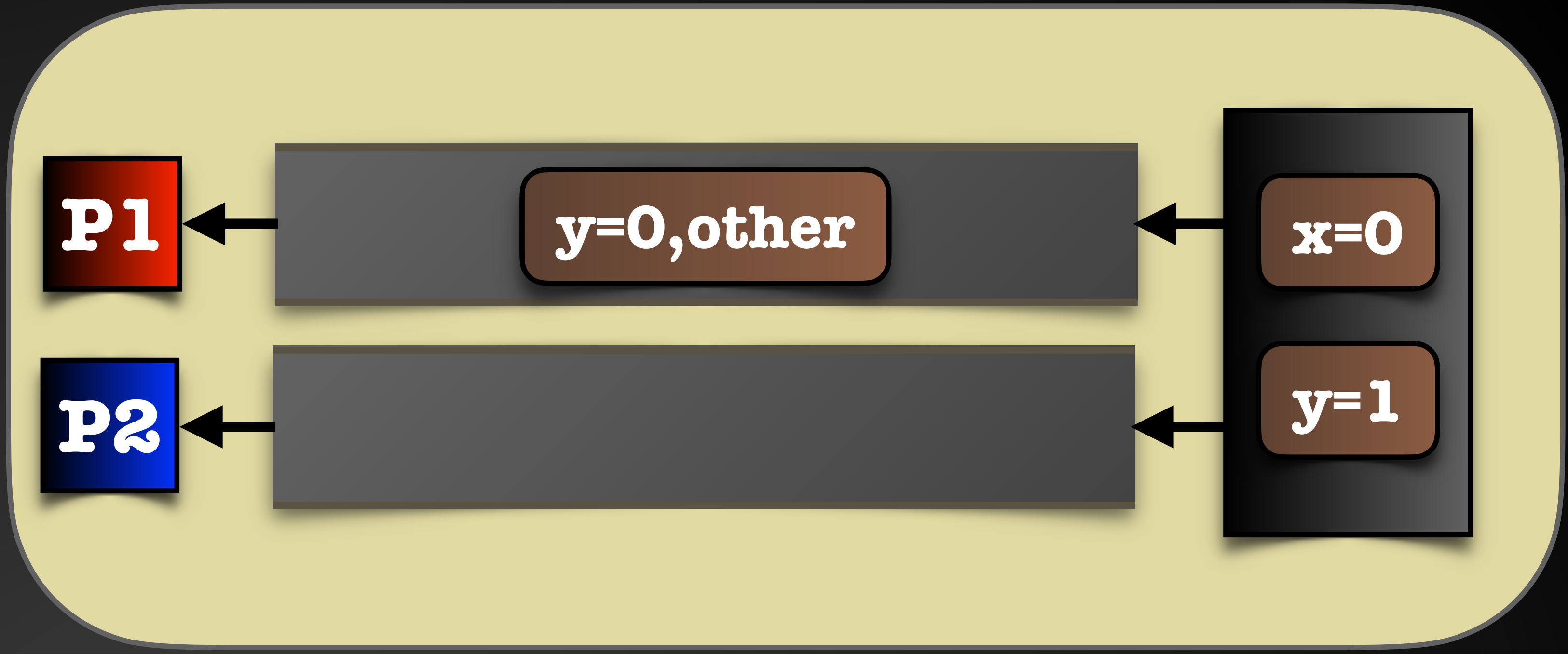
y=0

Dual TSO

P2: w(y,1)


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

x=0

P2

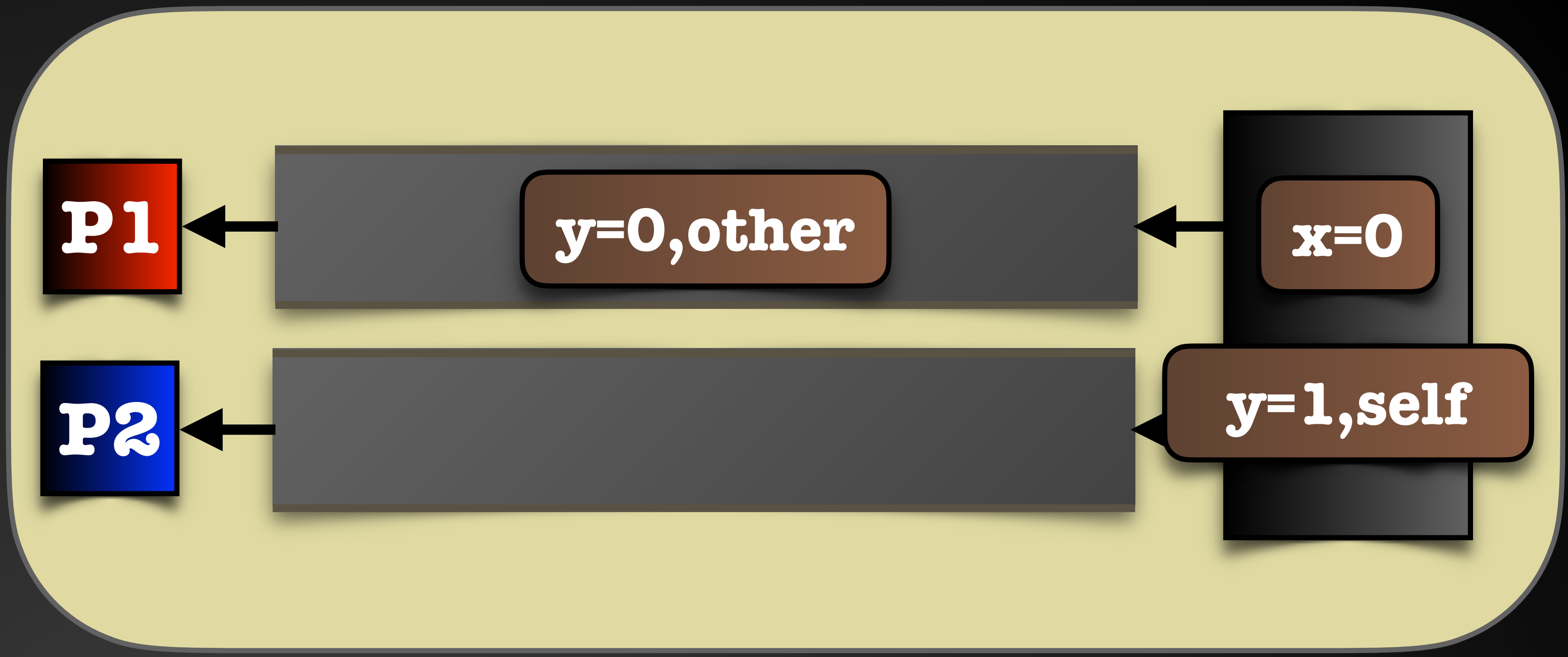
y=1

Dual TSO

P2: w(y,1)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

x=0

P2

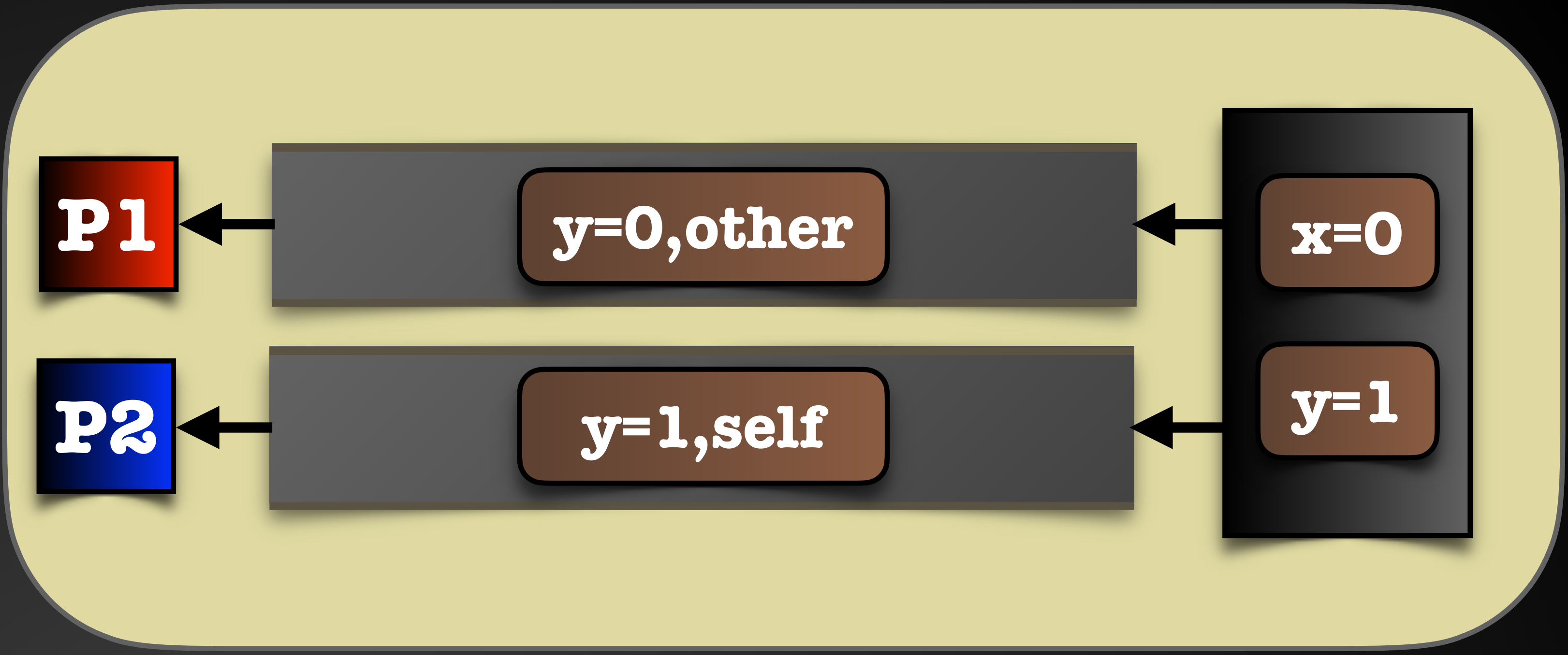
y=1,self

Dual TSO

P2: w(y,1)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

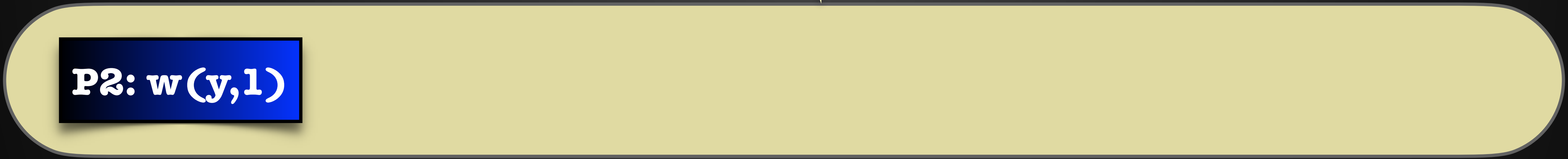
P2

y=1,self

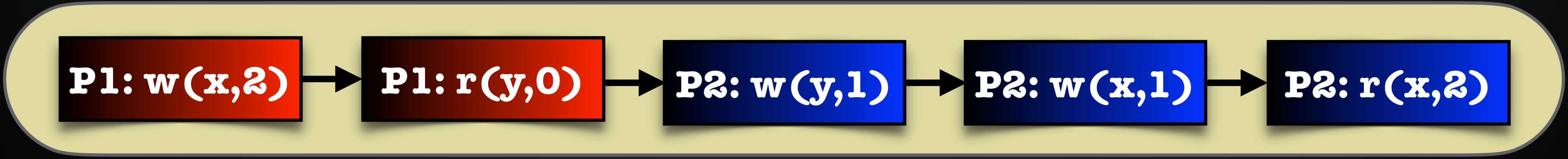
x=0

y=1

Dual TSO



P2: w(y,1)



P1: w(x,2)

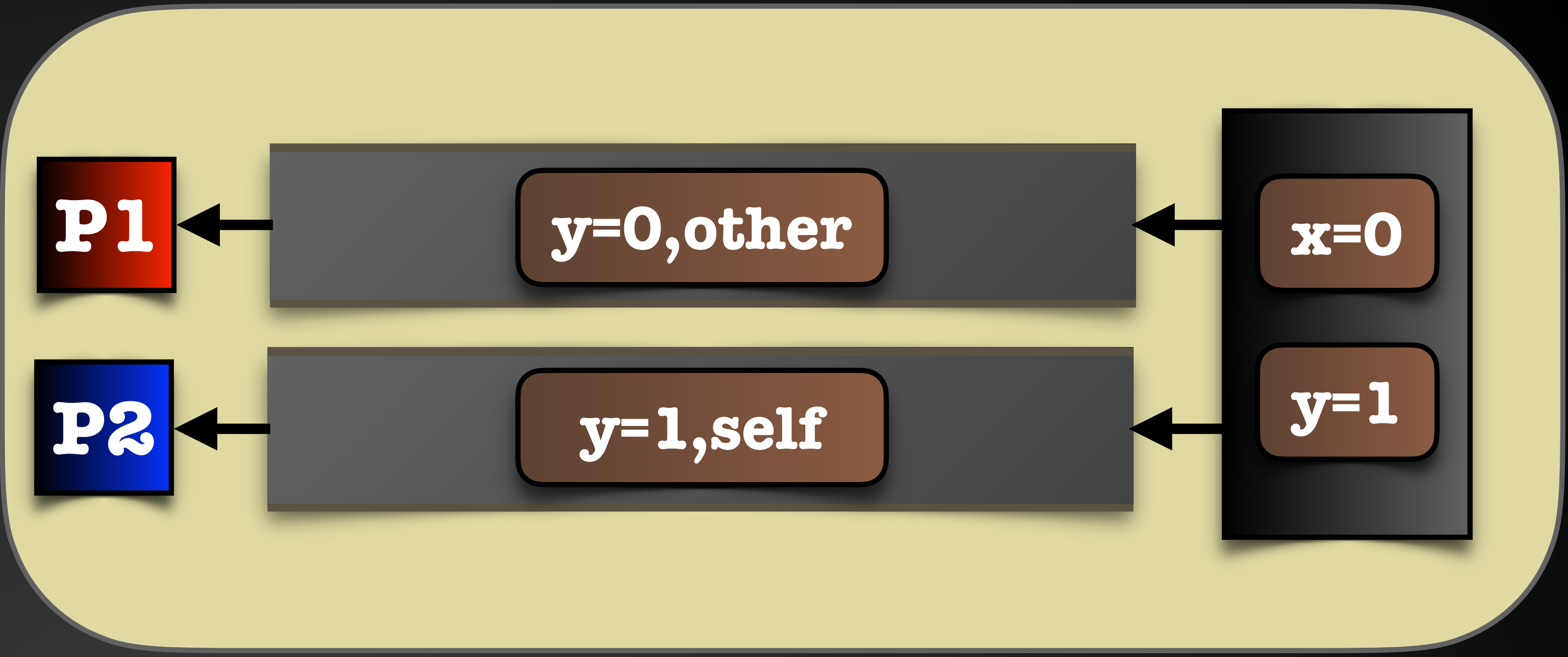
P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

P2: r(x,2)

Classical TSO



P1

y=0,other

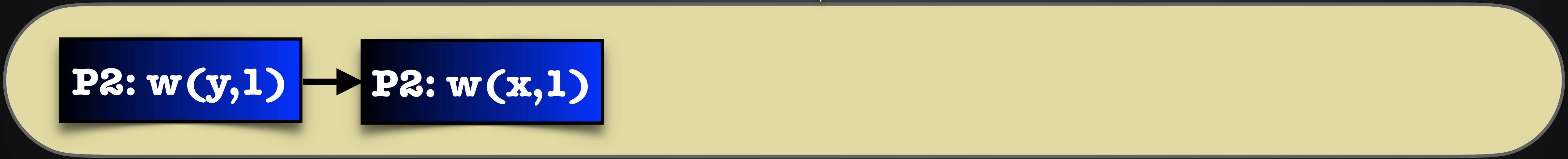
x=0

P2

y=1,self

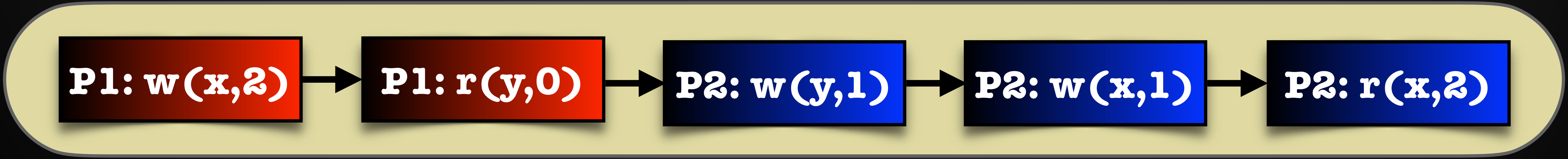
y=1

Dual TSO



P2: w(y,1)

P2: w(x,1)



P1: w(x,2)

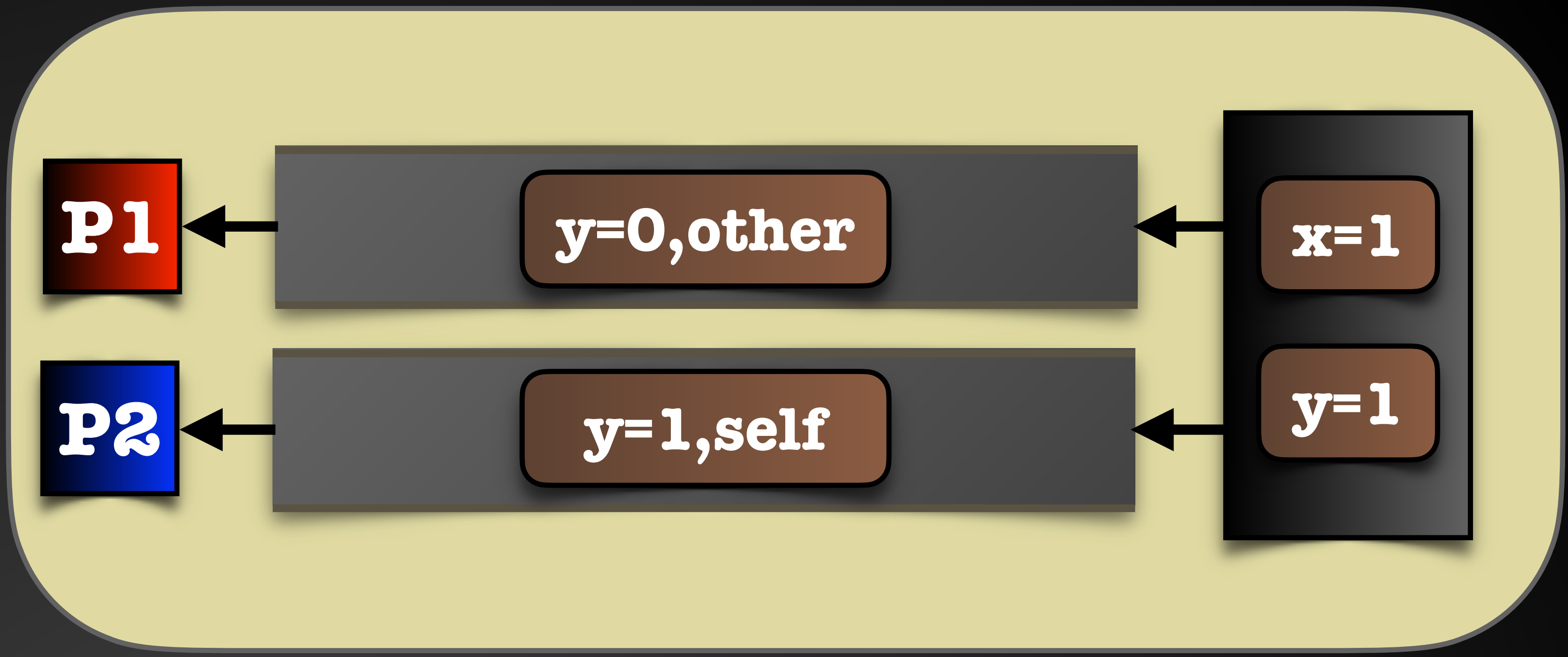
P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

P2: r(x,2)

Classical TSO

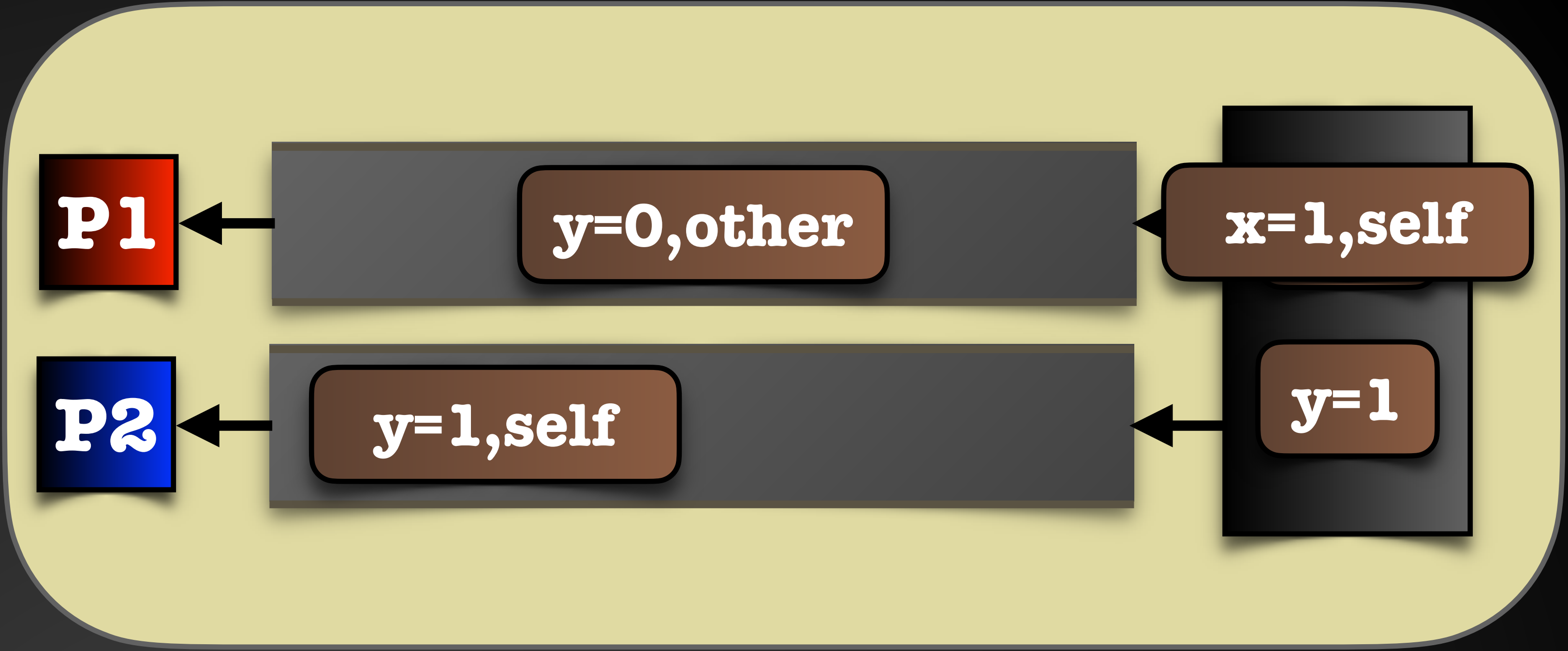


Dual TSO

P2: w(y,1) → P2: w(x,1)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

x=1,self

P2

y=1,self

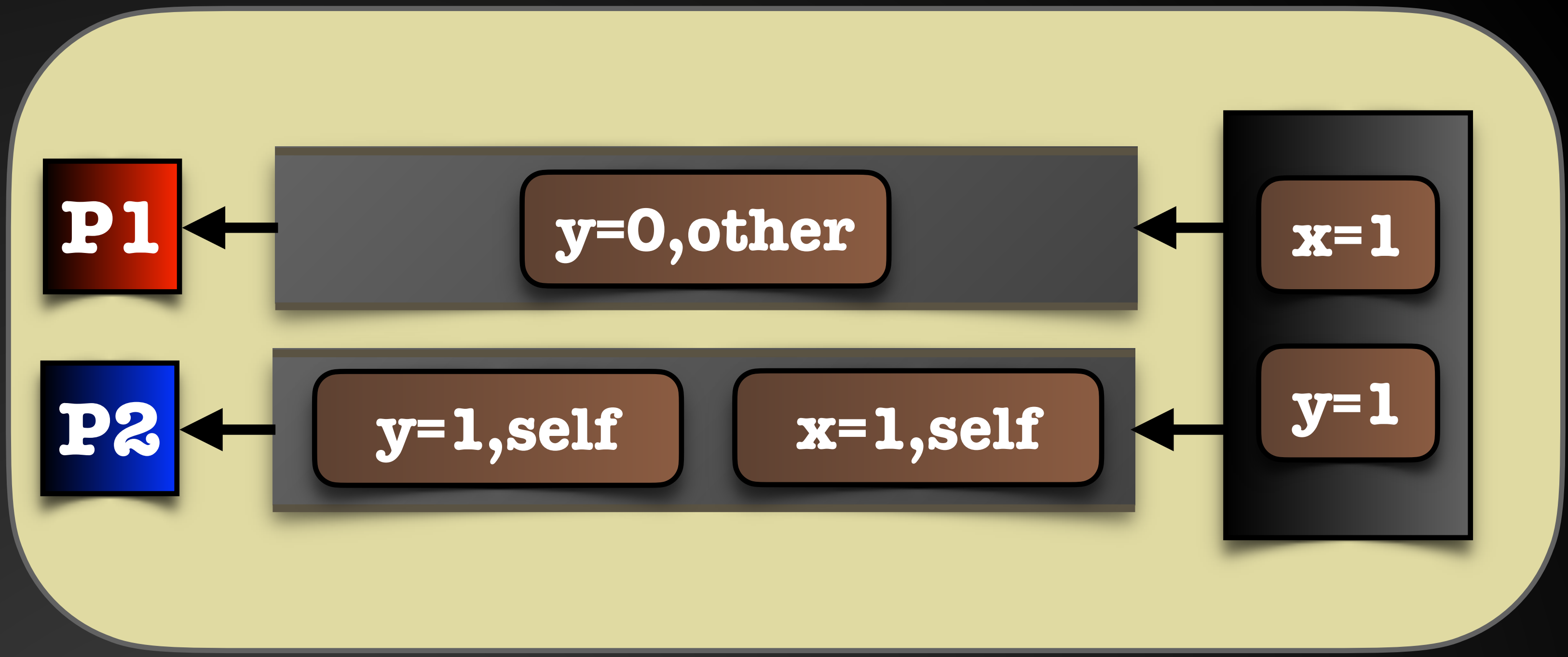
y=1

Dual TSO

P2: w(y,1) → P2: w(x,1)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

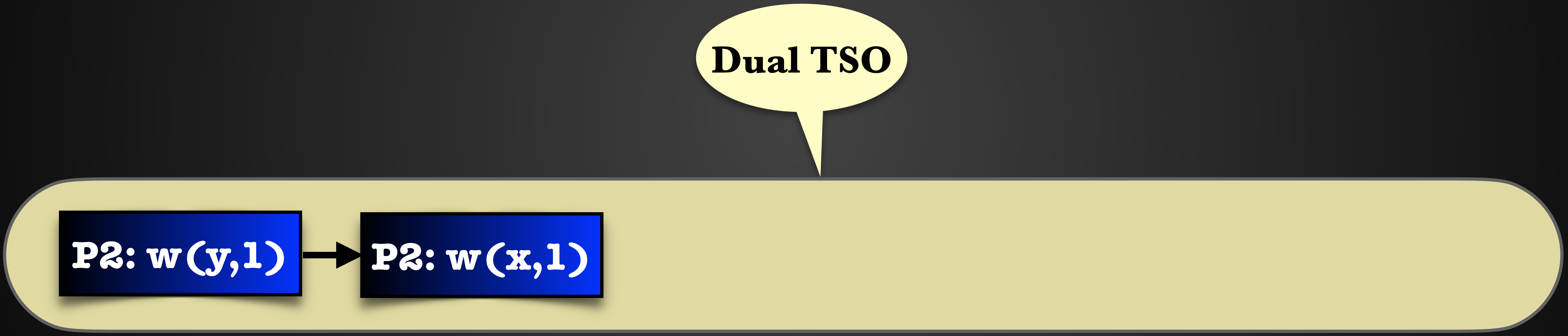
x=1

P2

y=1,self

x=1,self

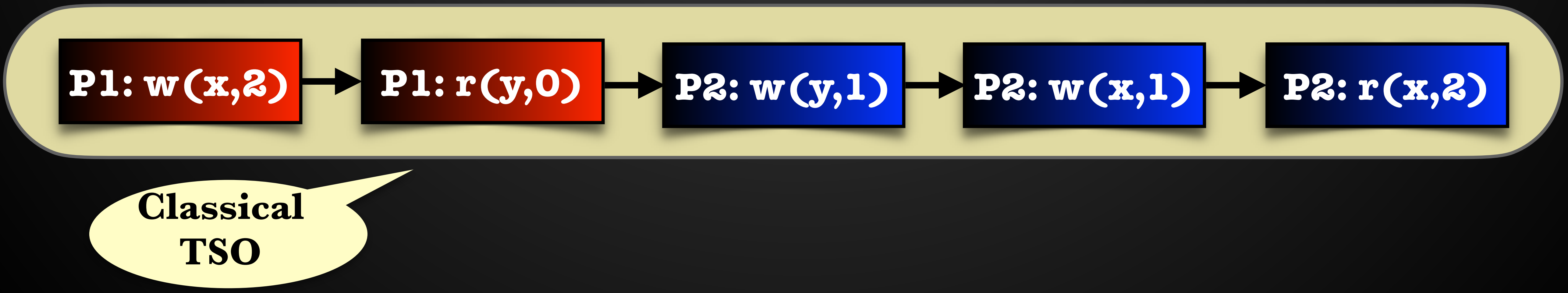
y=1



Dual TSO

P2: w(y,1)

P2: w(x,1)



P1: w(x,2)

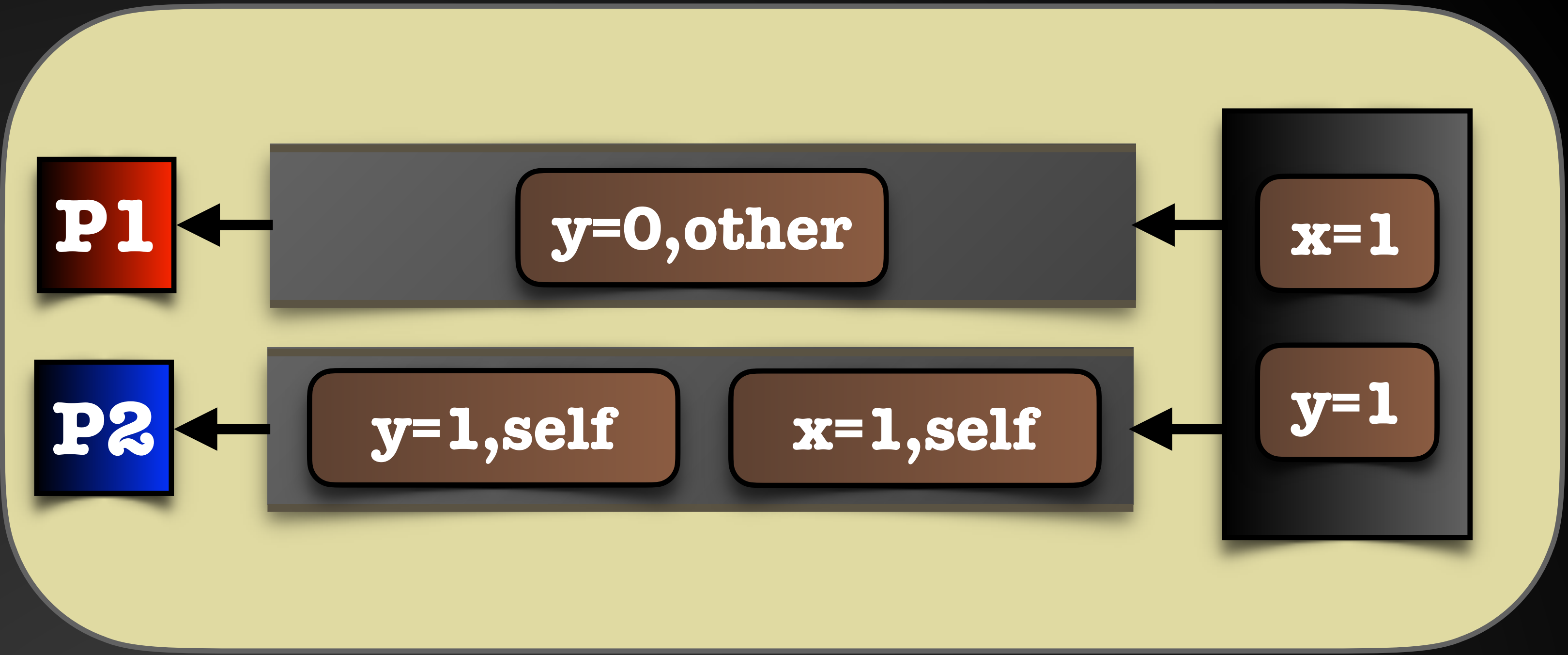
P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

P2: r(x,2)

Classical TSO



P1

y=0,other

x=1

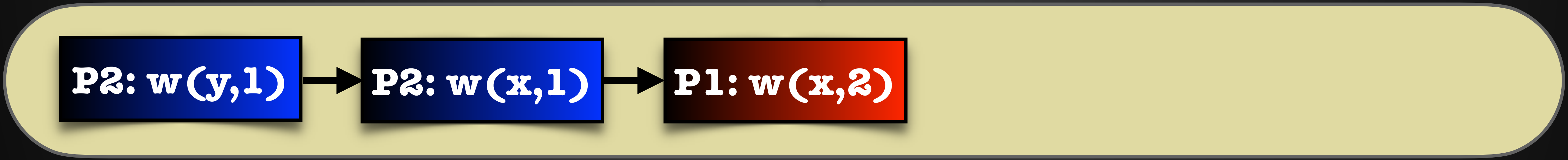
P2

y=1,self

x=1,self

y=1

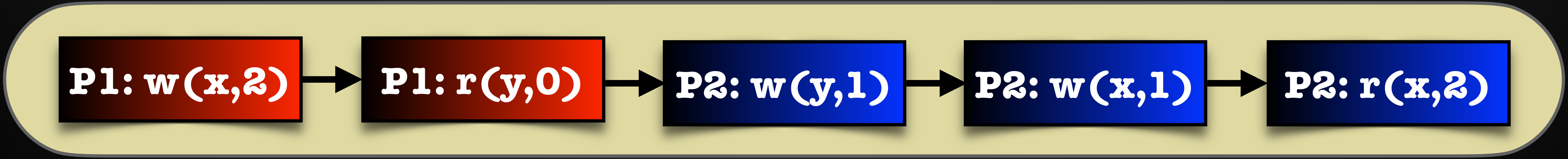
Dual TSO



P2: w(y,1)

P2: w(x,1)

P1: w(x,2)



P1: w(x,2)

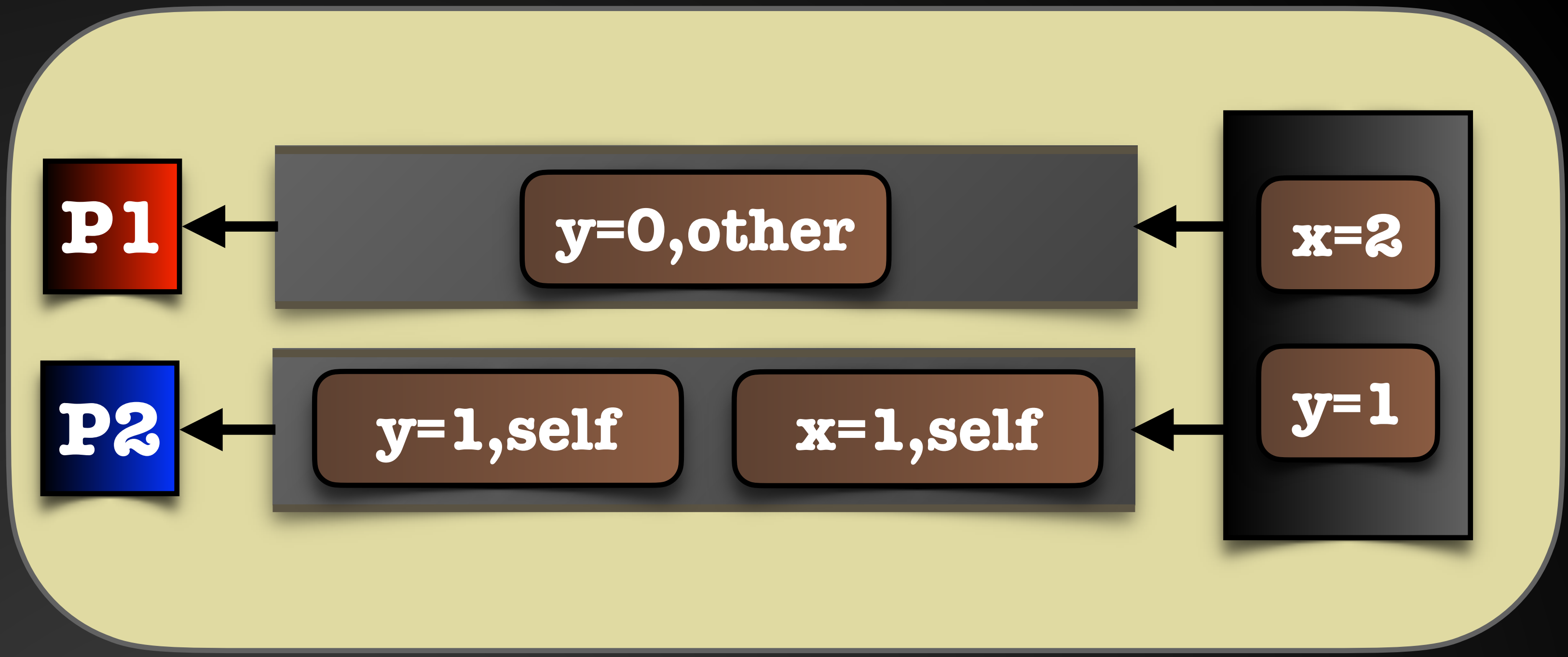
P1: r(y,0)

P2: w(y,1)

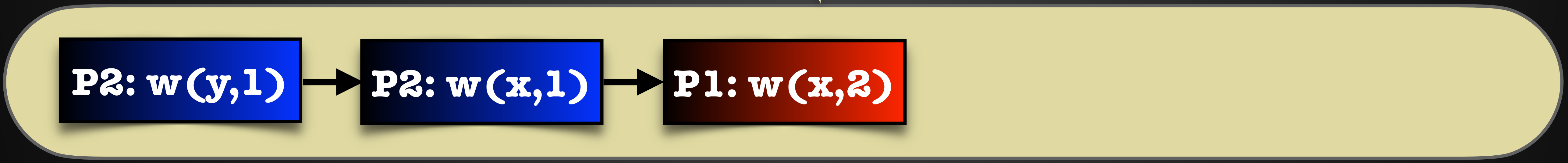
P2: w(x,1)

P2: r(x,2)

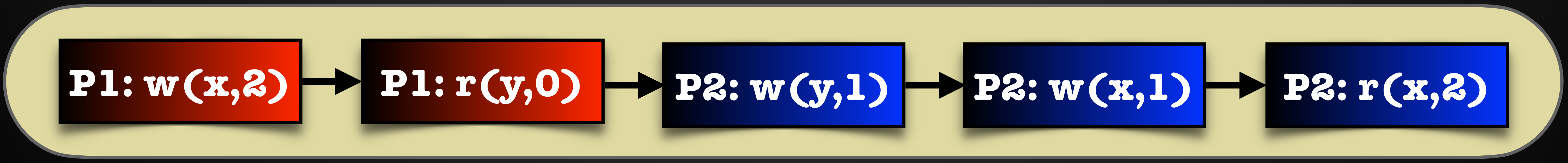
Classical TSO



Dual TSO

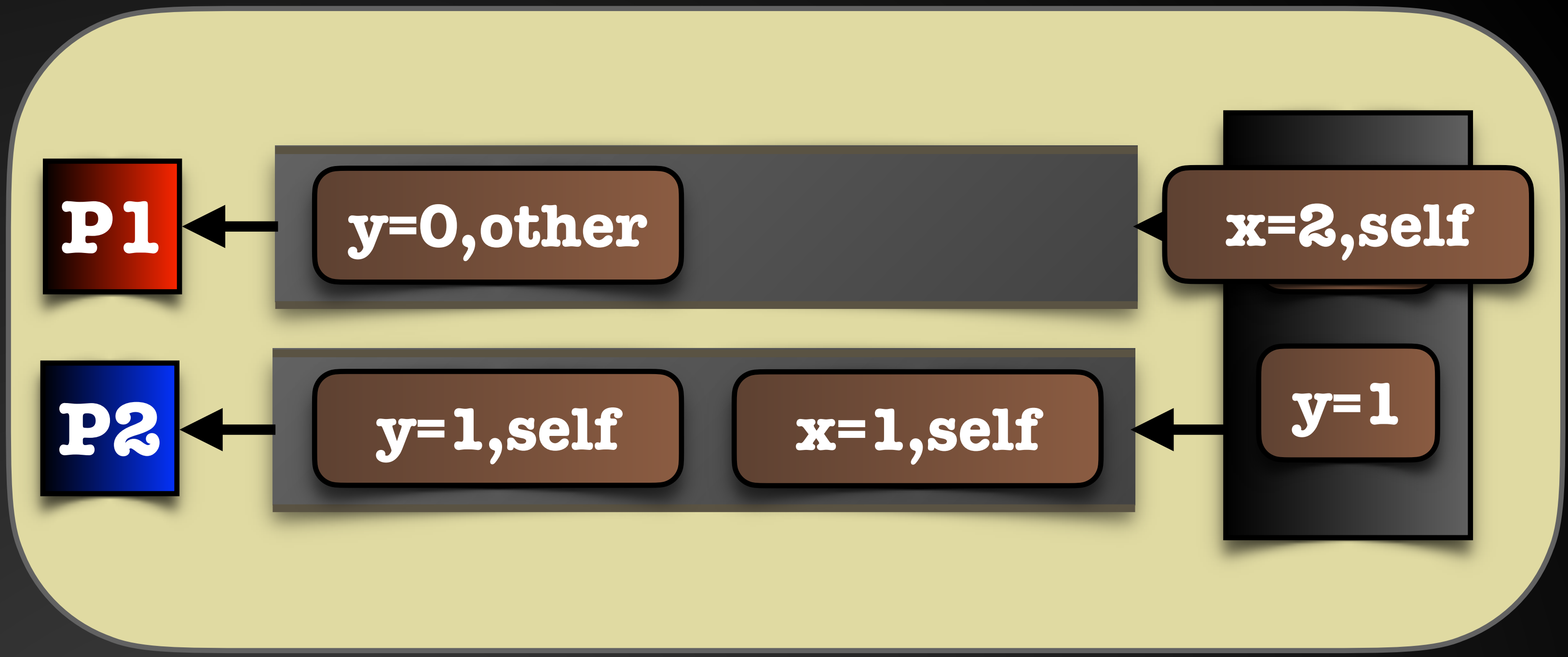


P2: w(y,1) → P2: w(x,1) → P1: w(x,2)

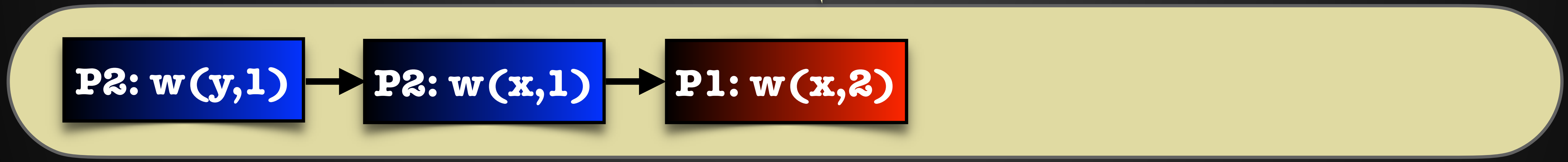


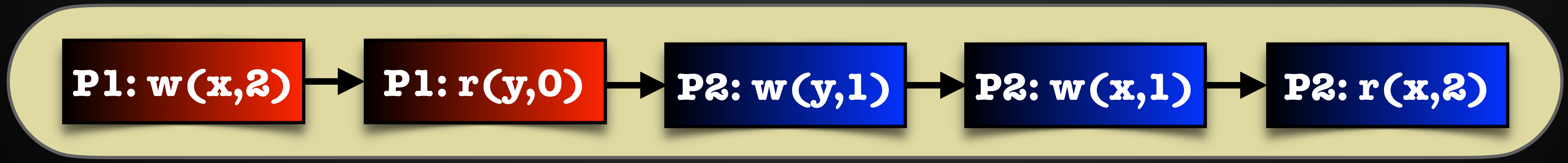
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

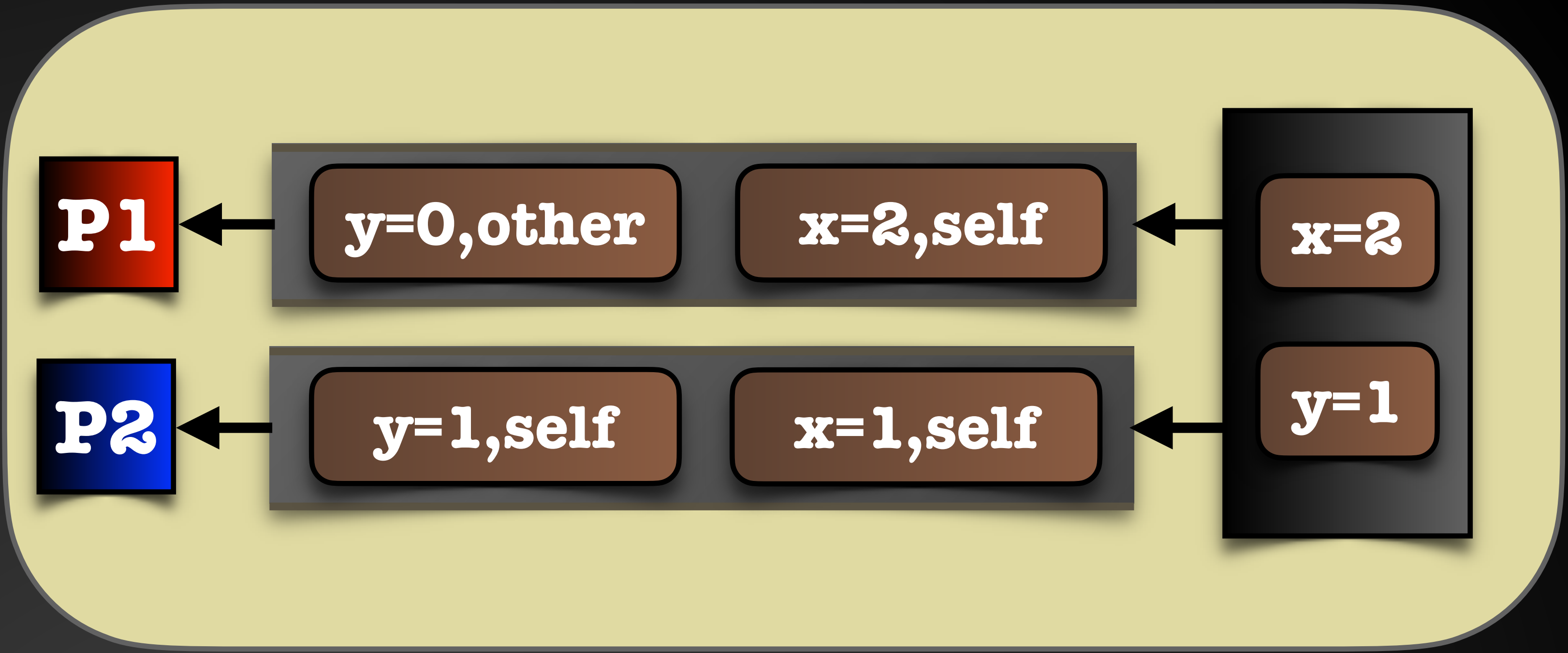


Dual TSO





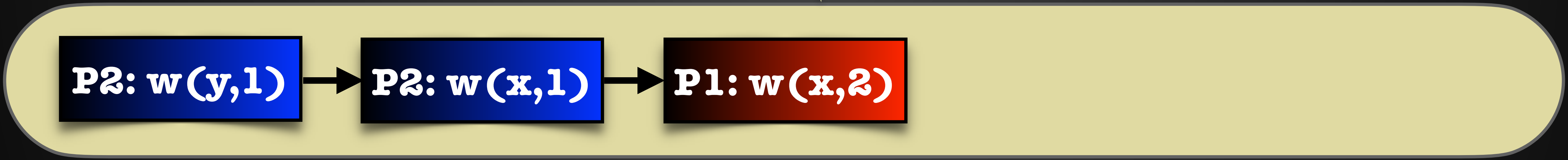
Classical TSO



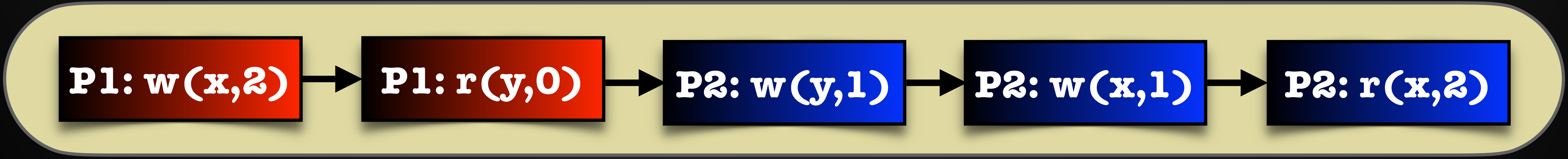
P1 ← y=0,other | x=2,self ← x=2

P2 ← y=1,self | x=1,self ← y=1

Dual TSO

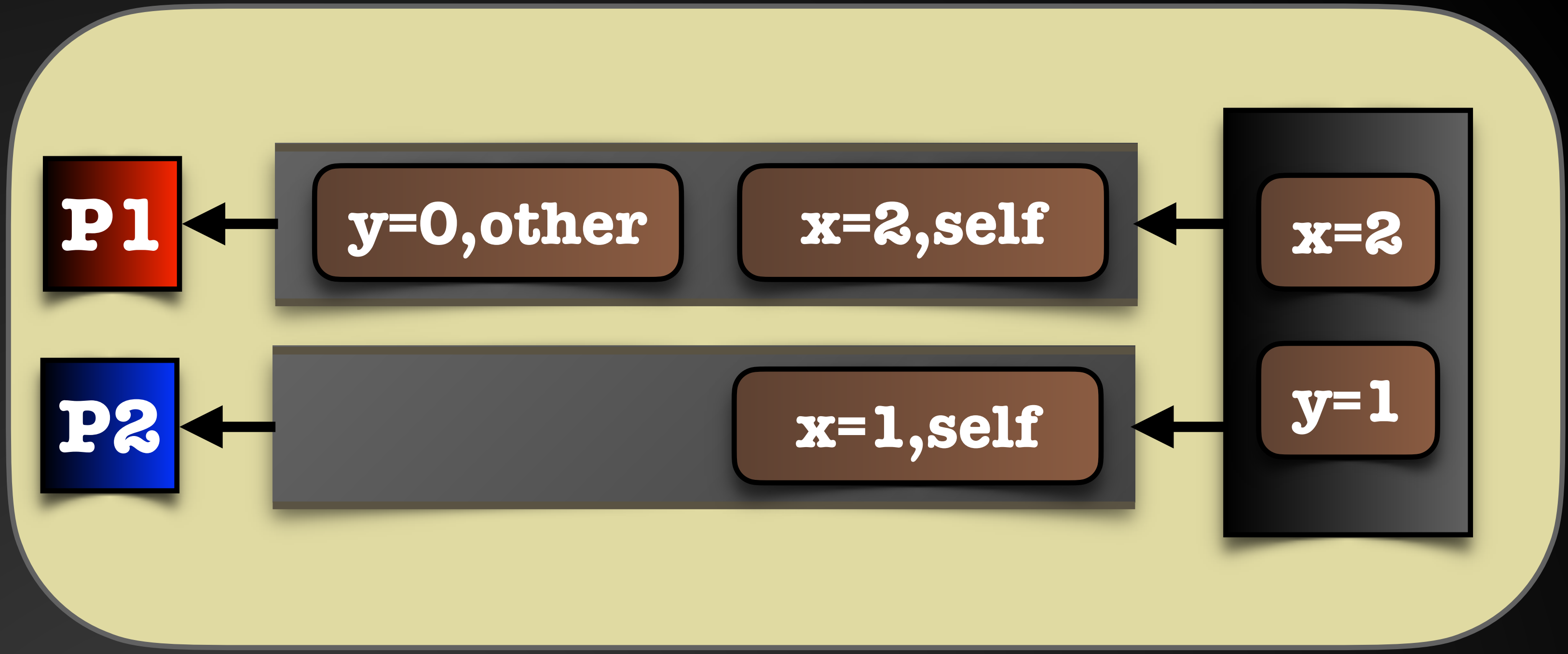


P2: w(y,1) → P2: w(x,1) → P1: w(x,2)



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

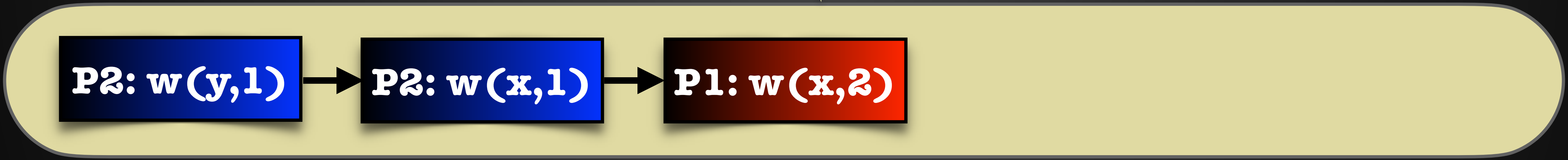
Classical TSO



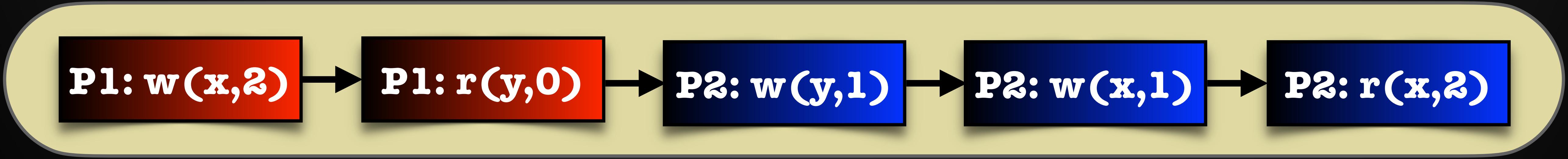
P1 ← y=0,other | x=2,self ← x=2

P2 ← x=1,self ← y=1

Dual TSO

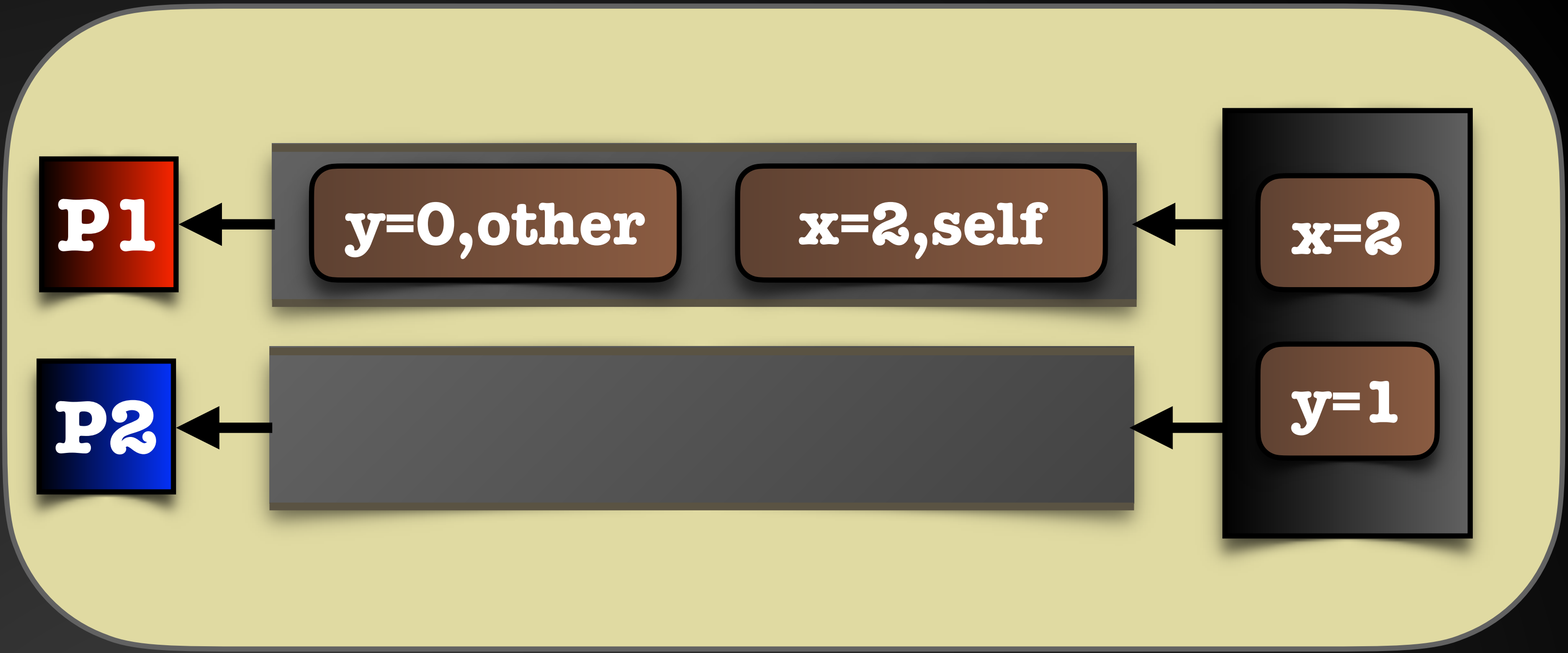


P2: w(y,1) → P2: w(x,1) → P1: w(x,2)



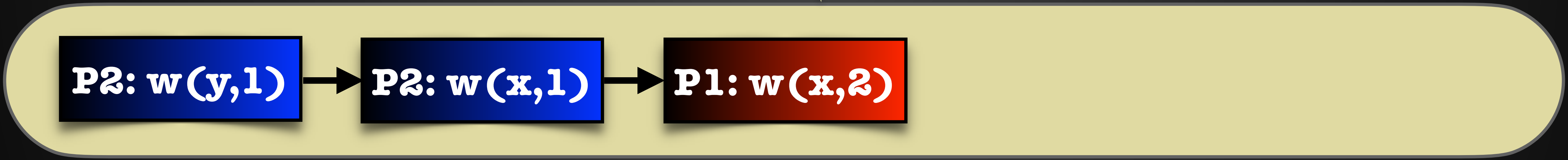
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

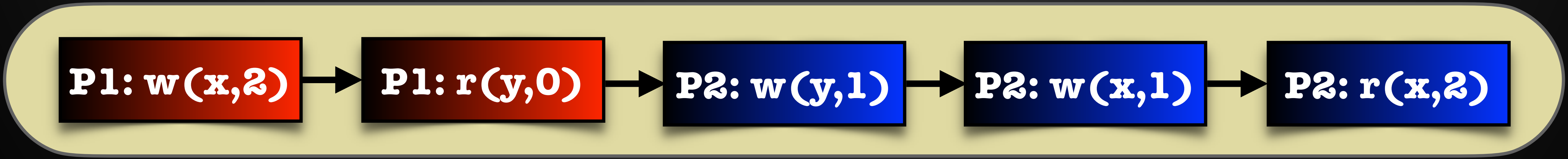
Classical TSO


P1 ← y=0,other | x=2,self ← x=2

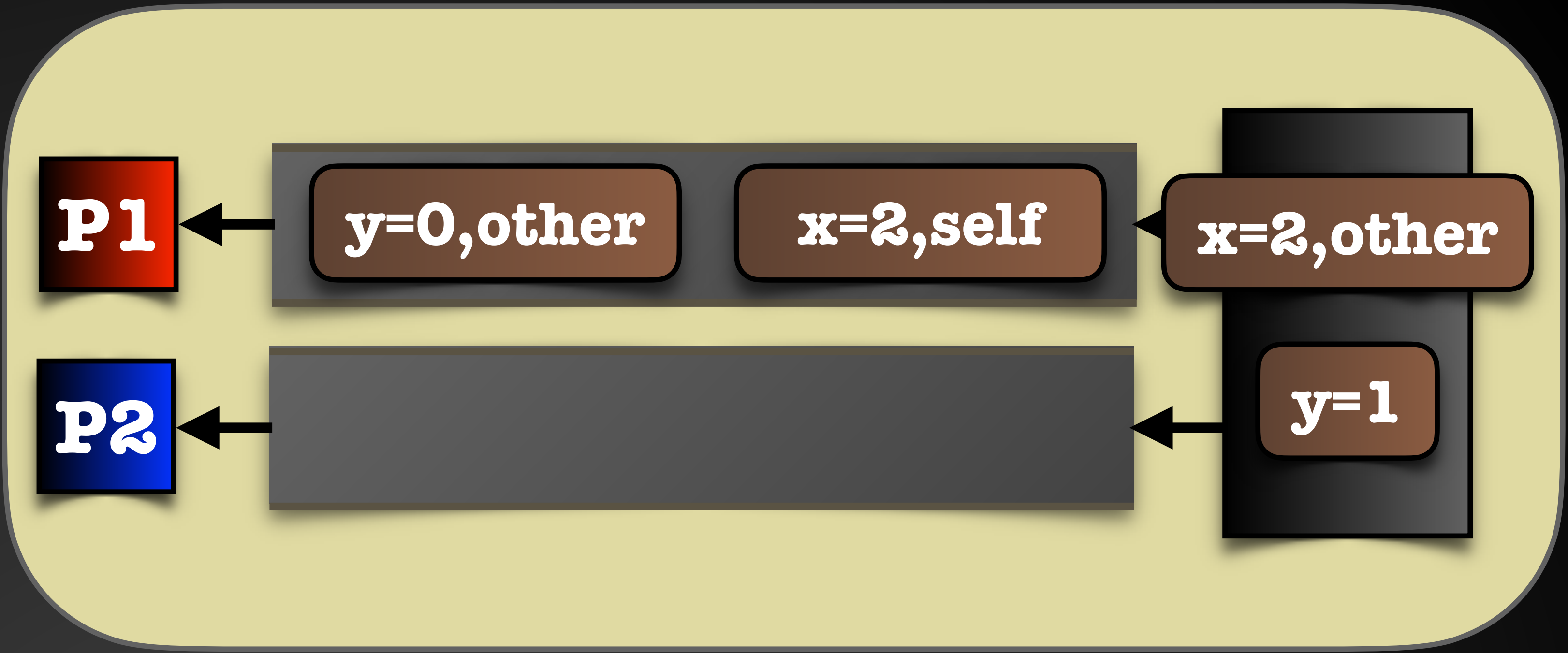
P2 ← ← y=1

Dual TSO


P2: w(y,1) → P2: w(x,1) → P1: w(x,2)


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

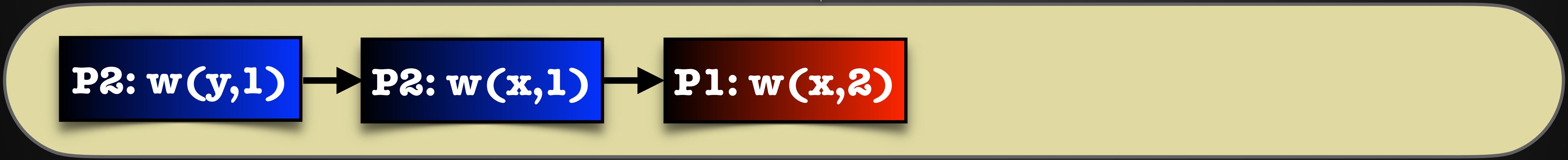
Classical TSO



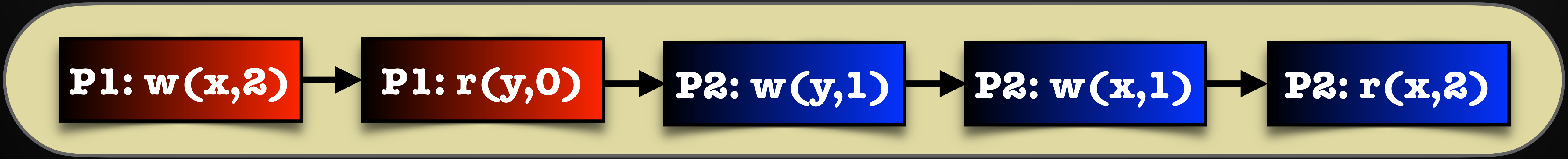
P1 ← y=0,other | x=2,self ← x=2,other

P2 ← ← y=1

Dual TSO

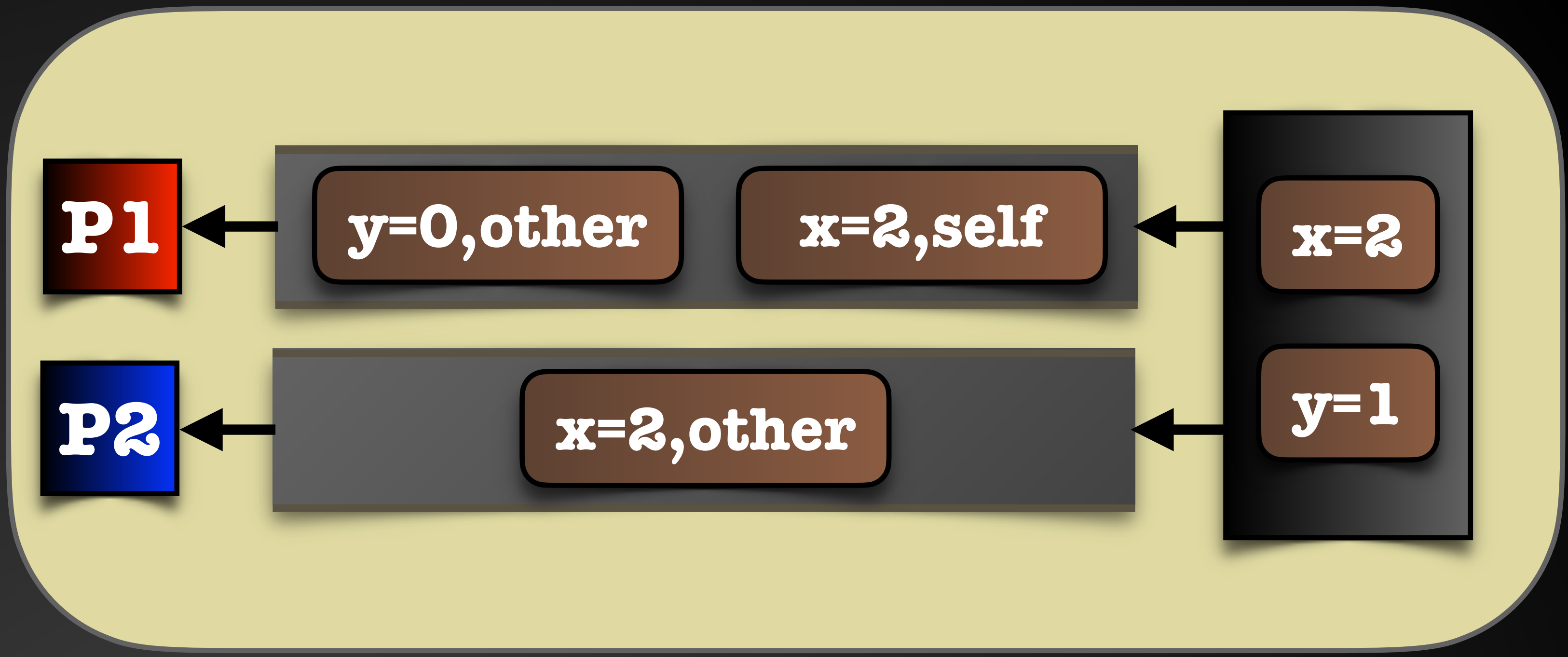


P2: w(y,1) → P2: w(x,1) → P1: w(x,2)



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

y=0,other

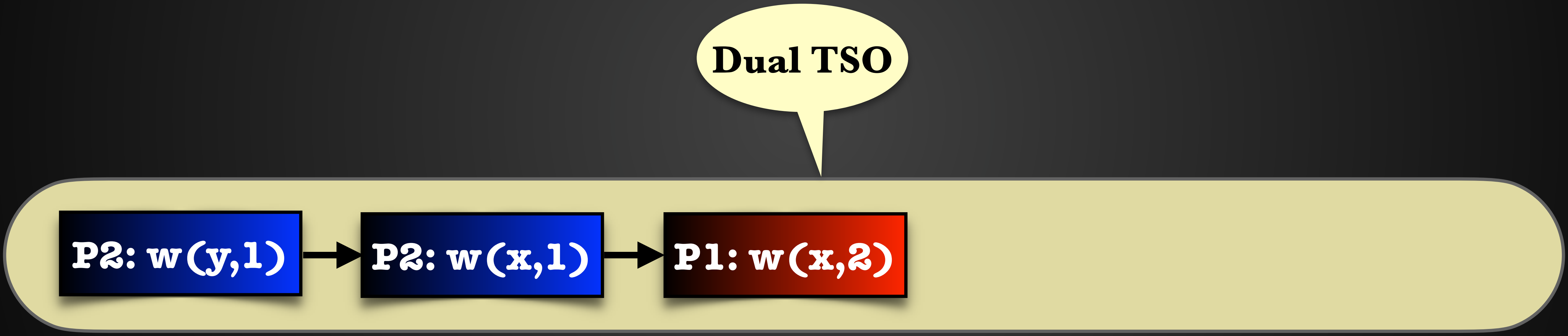
x=2,self

x=2

P2

x=2,other

y=1

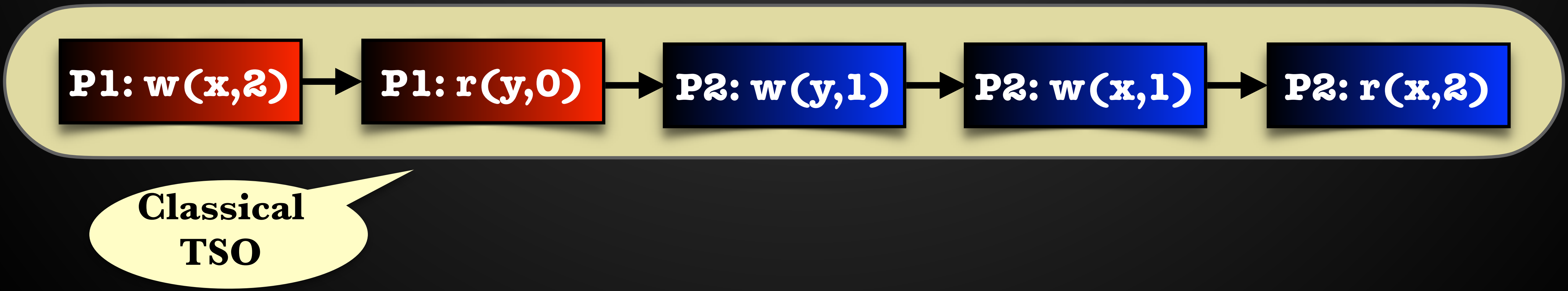


Dual TSO

P2: w(y,1)

P2: w(x,1)

P1: w(x,2)



P1: w(x,2)

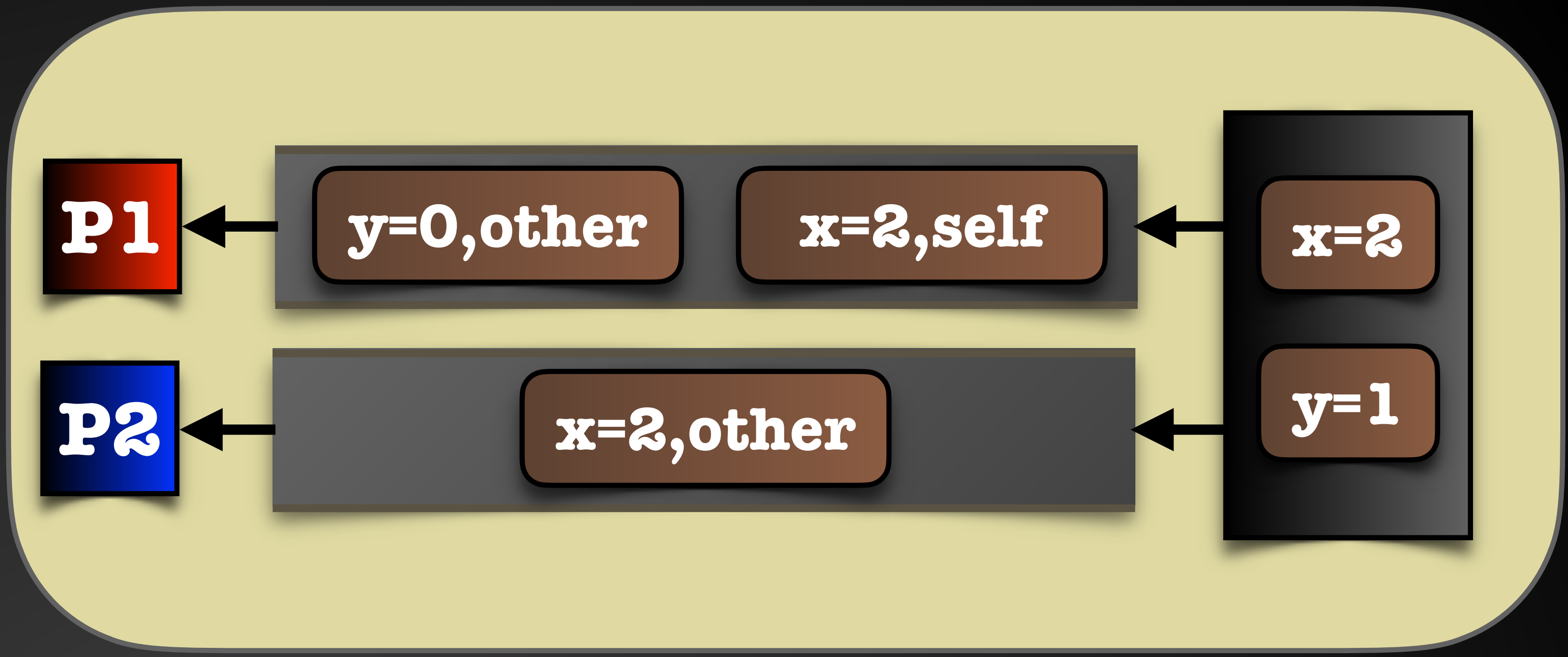
P1: r(y,0)

P2: w(y,1)

P2: w(x,1)

P2: r(x,2)

Classical TSO



P1

y=0,other

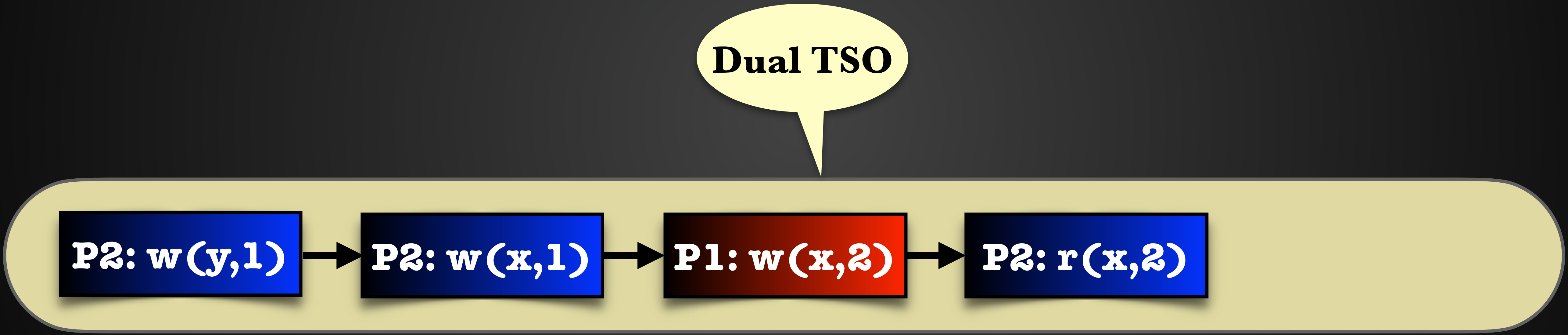
x=2,self

P2

x=2,other

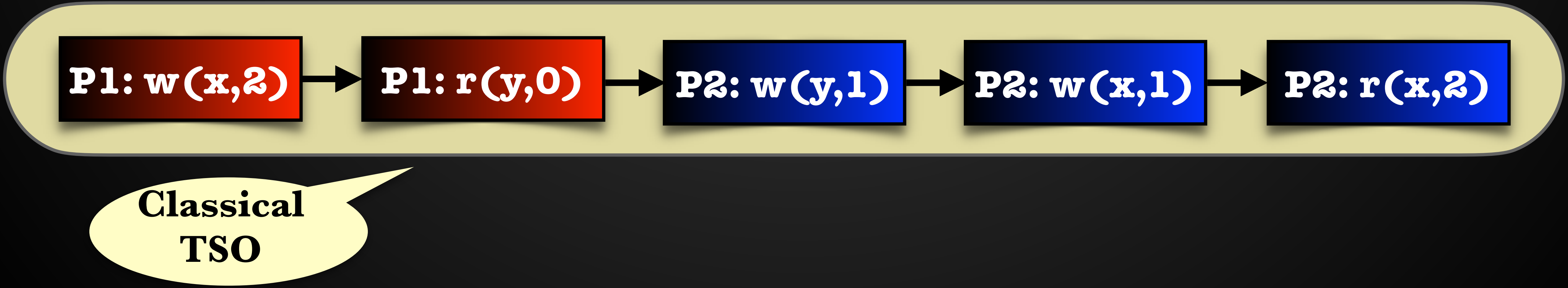
x=2

y=1



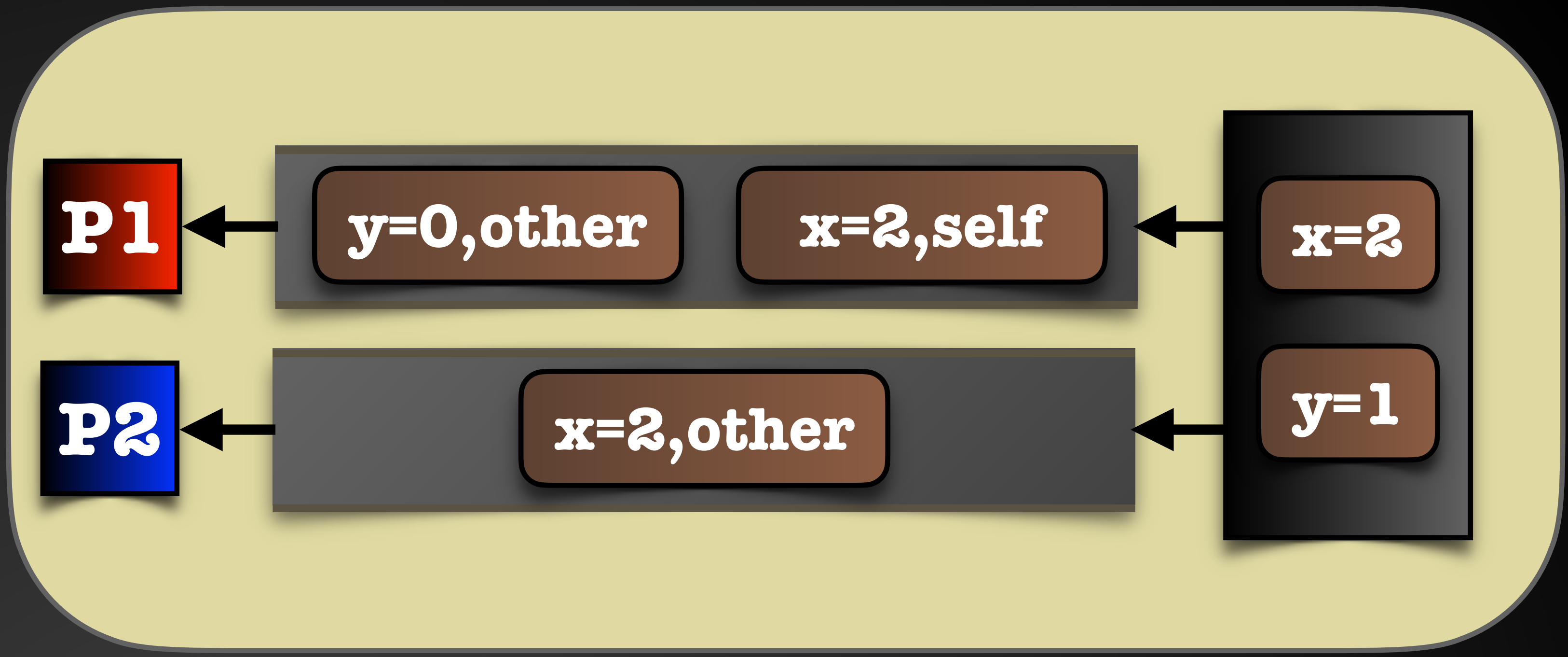
Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2)



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1 ← y=0,other | x=2,self ← x=2

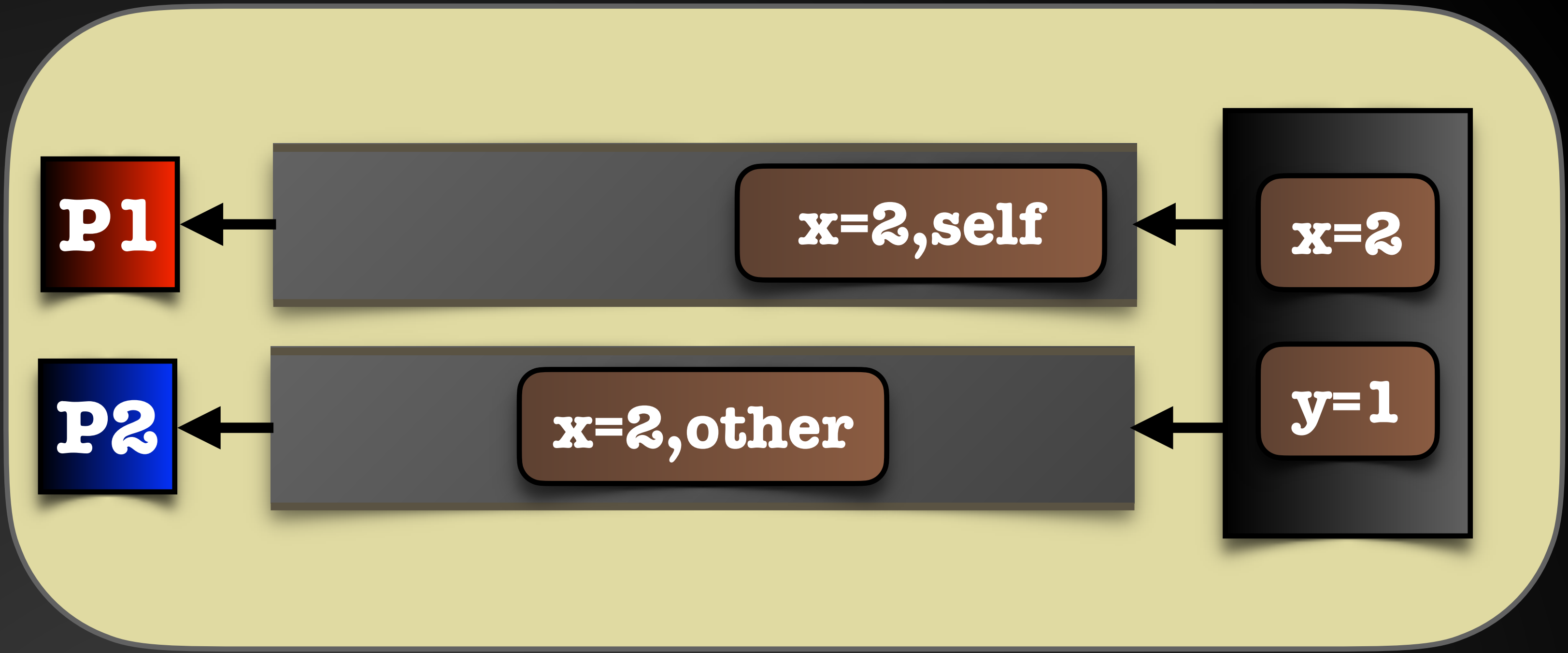
P2 ← x=2,other ← y=1

Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



P1

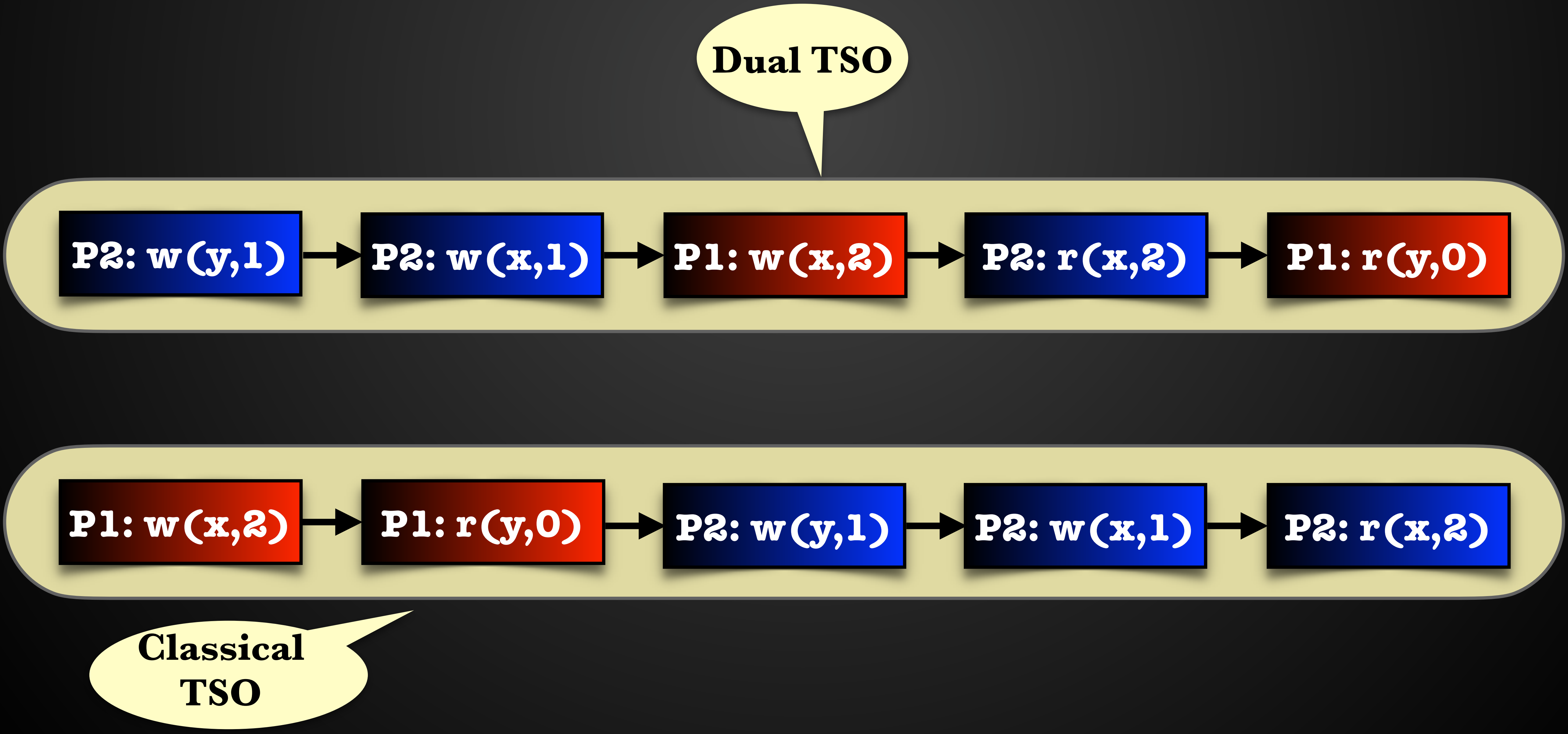
x=2,self

x=2

P2

x=2,other

y=1

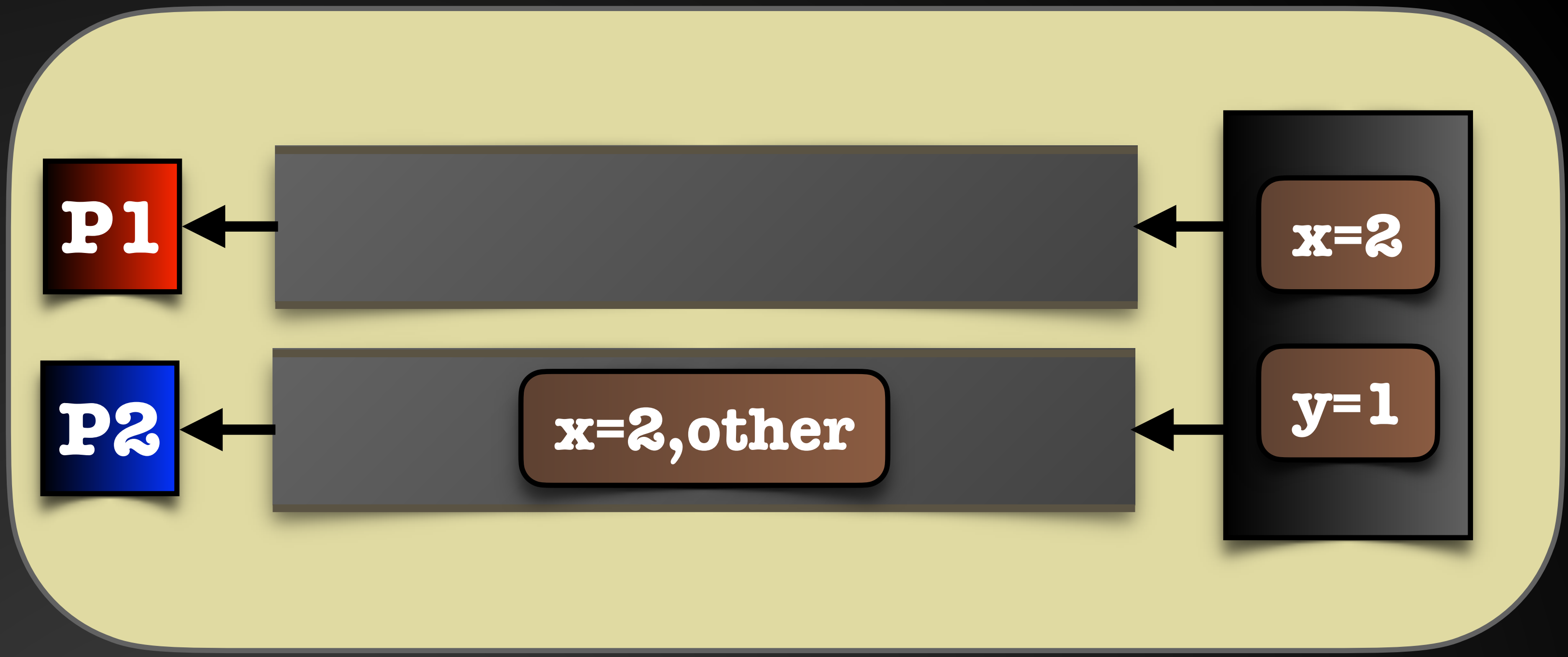


Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



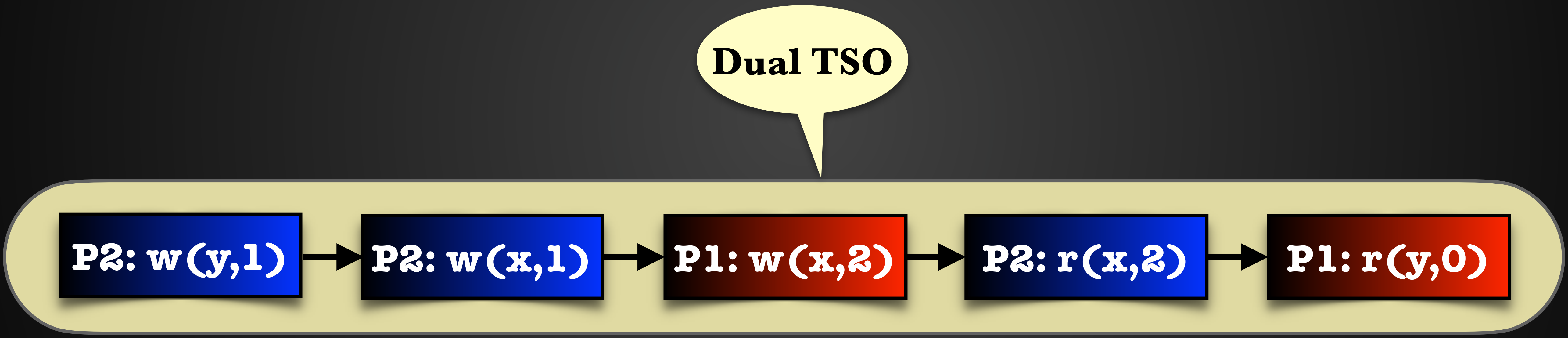
P1

P2

x=2,other

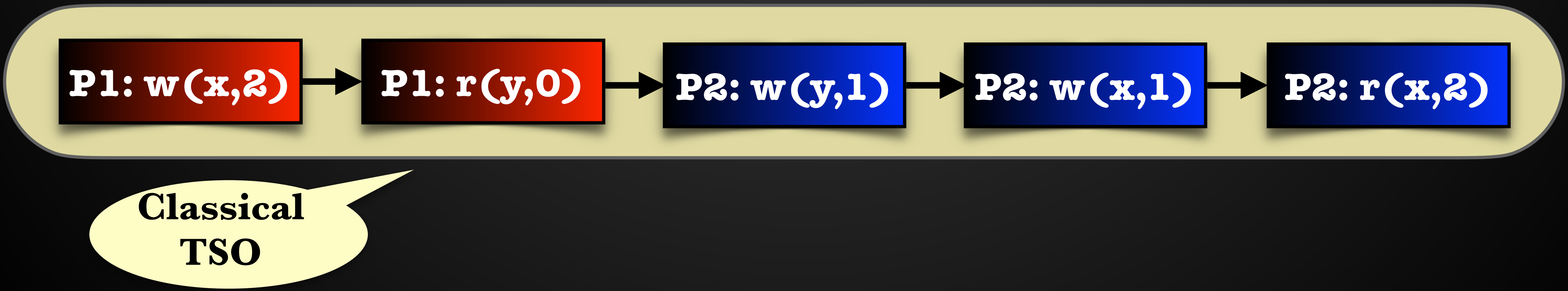
x=2

y=1



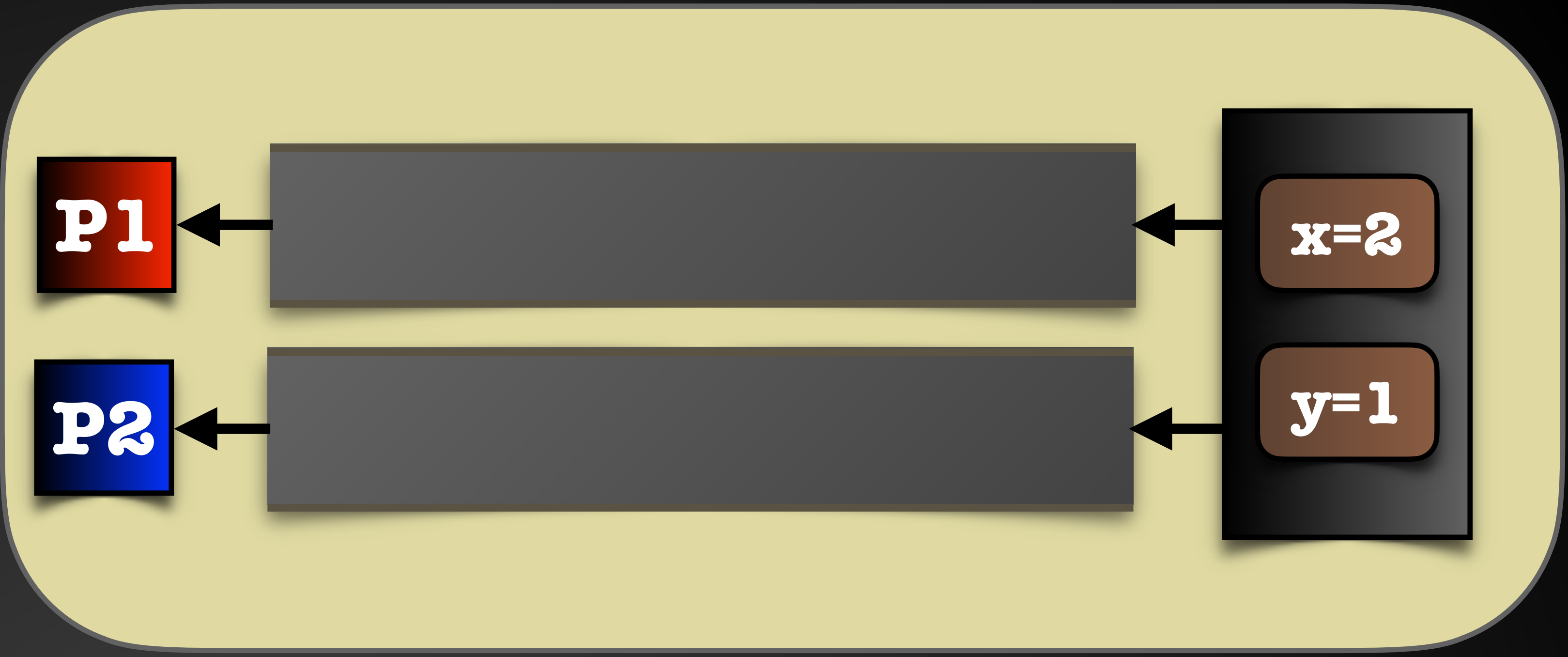
Dual TSO

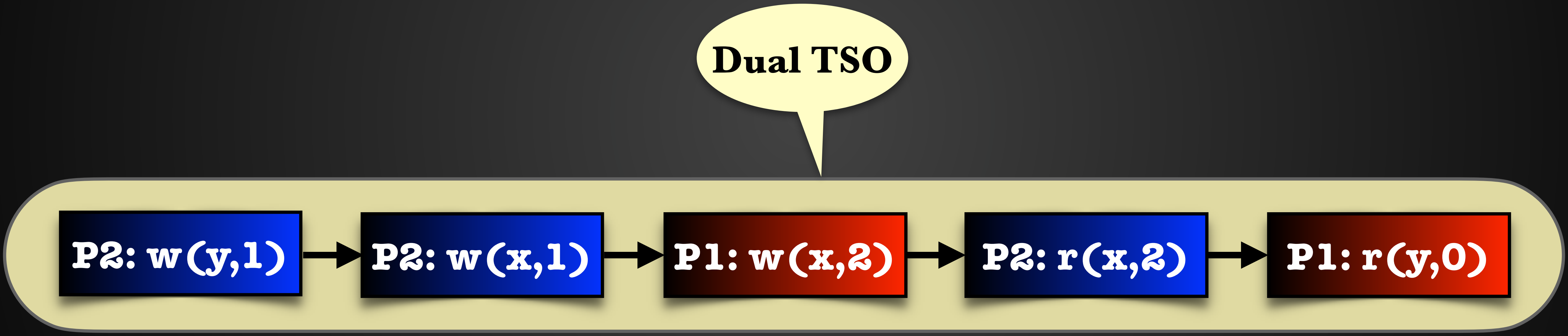
P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)



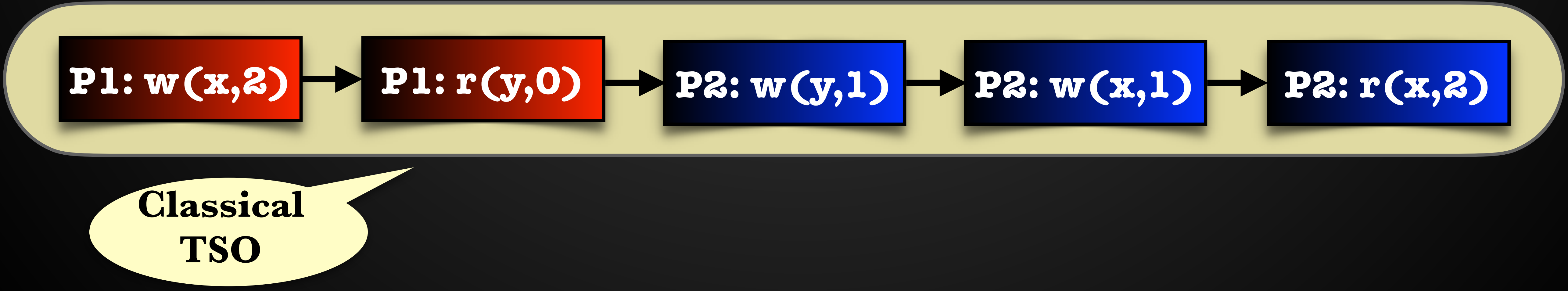
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

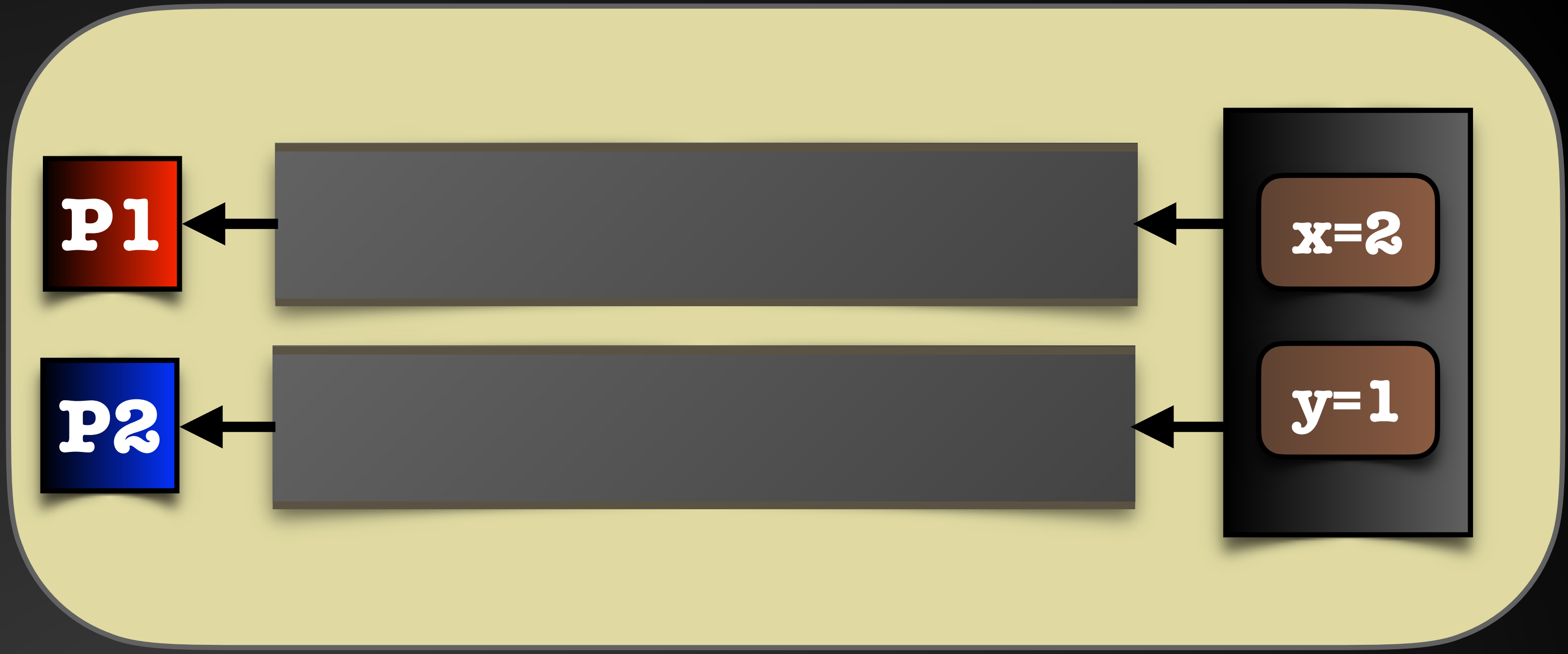



Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

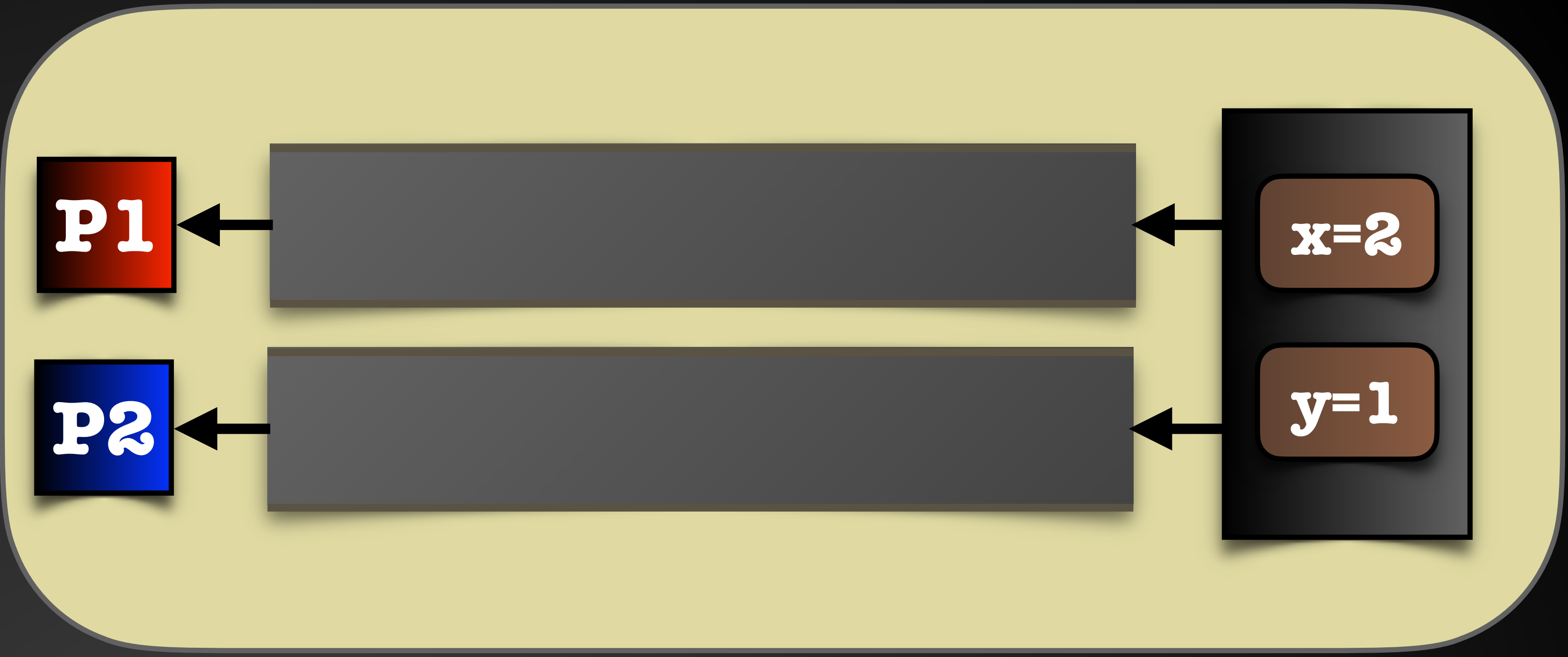


Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)

P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

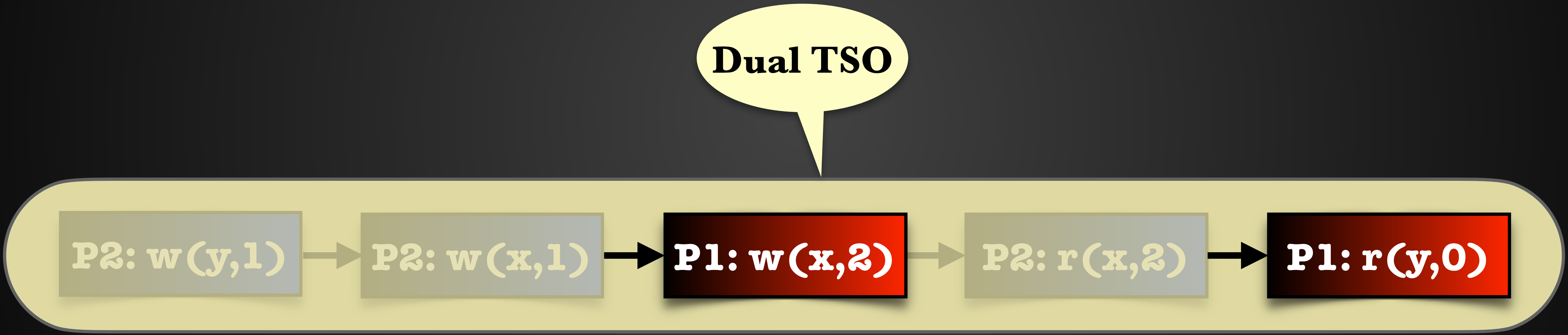


P1

P2

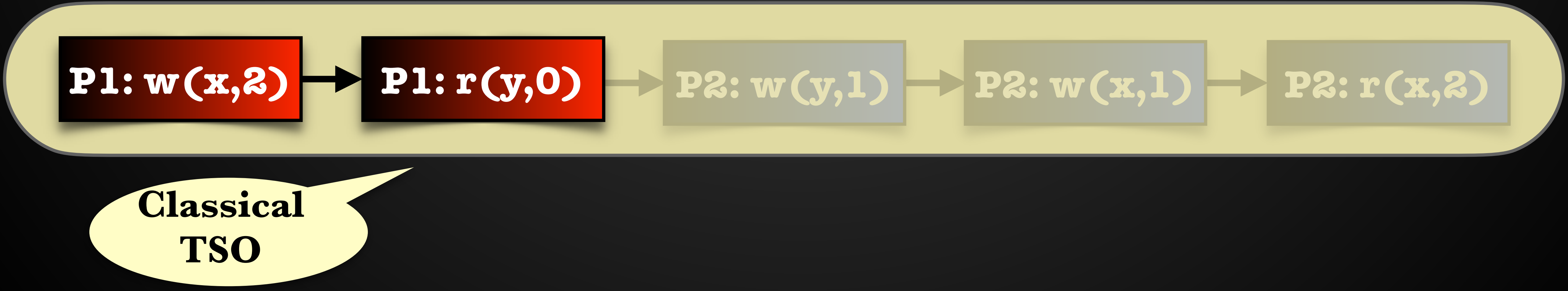
x=2

y=1



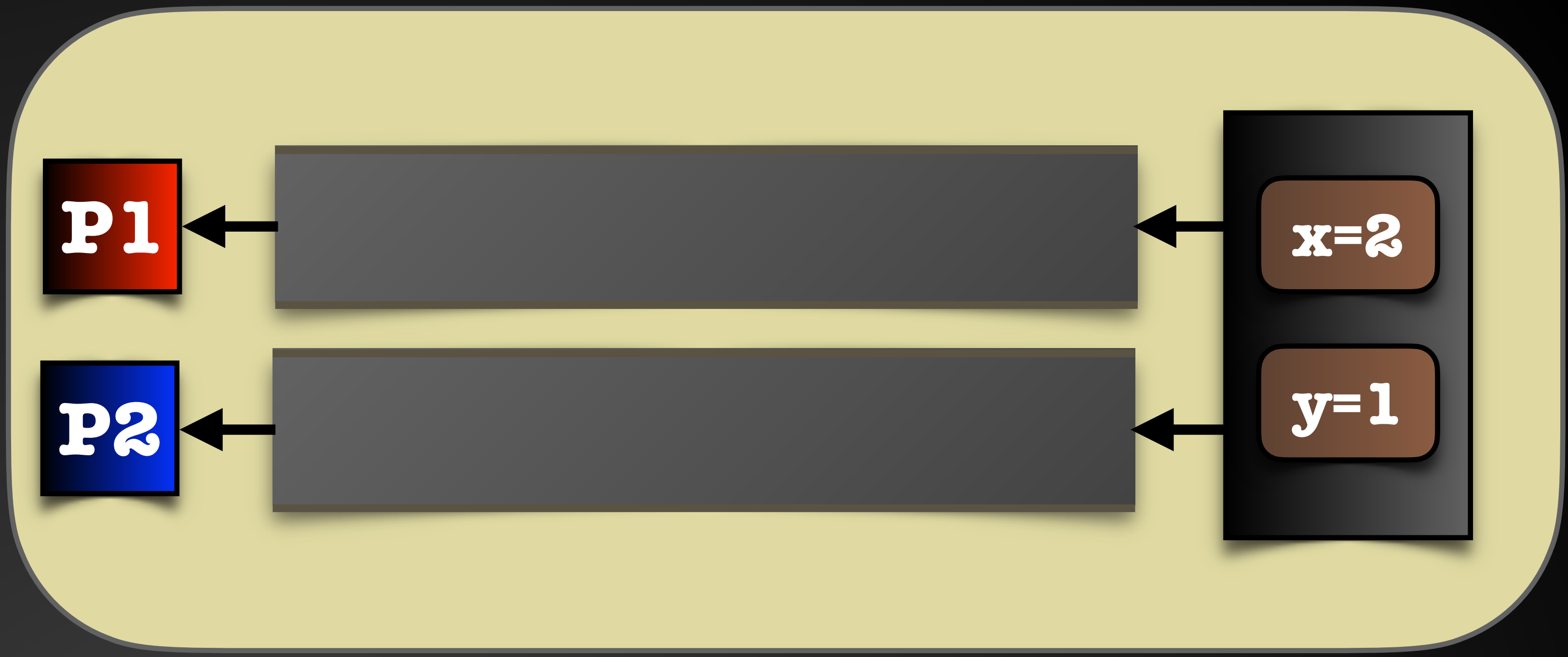
Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)

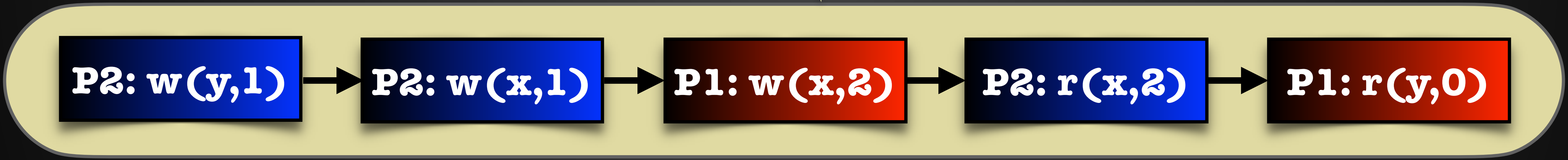


P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

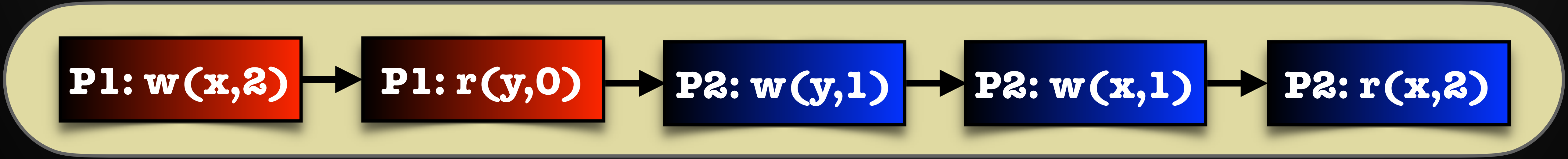
Classical TSO



Dual TSO

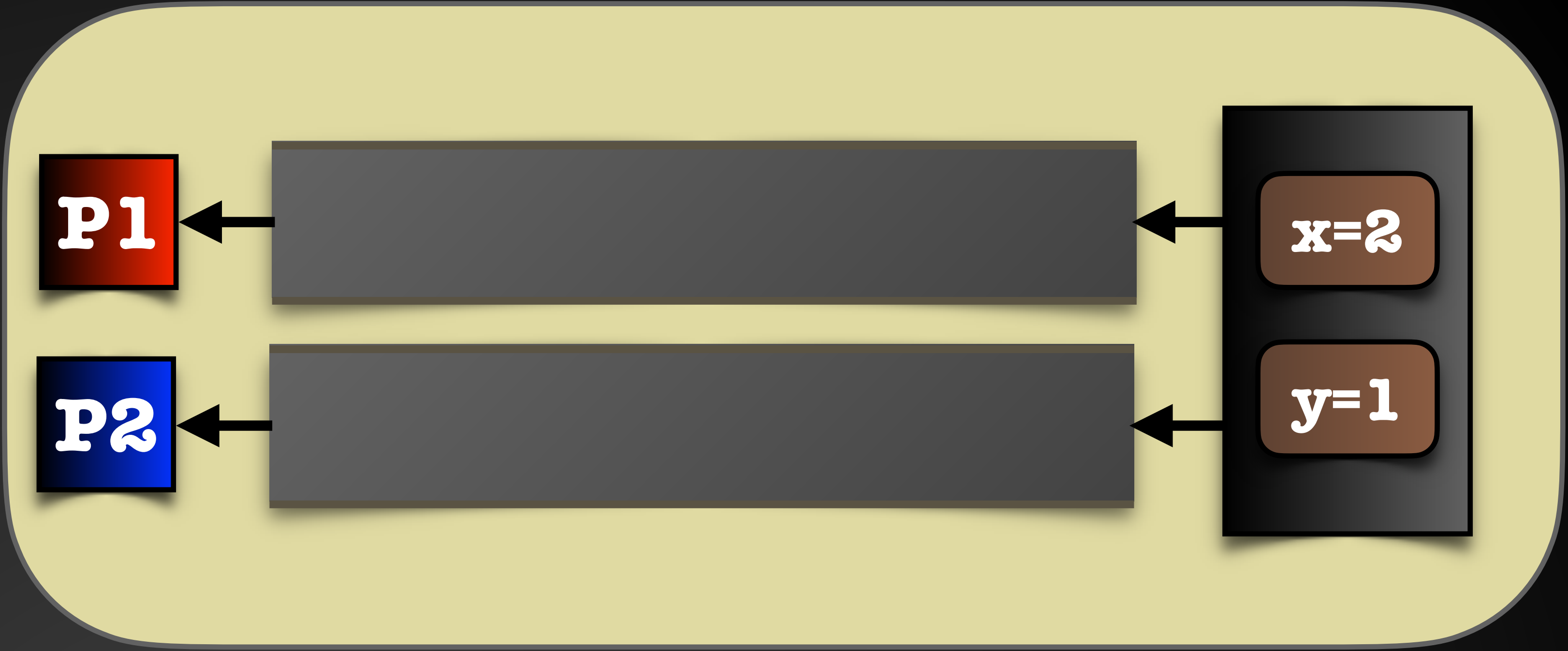


P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)



P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO



Dual TSO

P2: w(y,1) → P2: w(x,1) → P1: w(x,2) → P2: r(x,2) → P1: r(y,0)

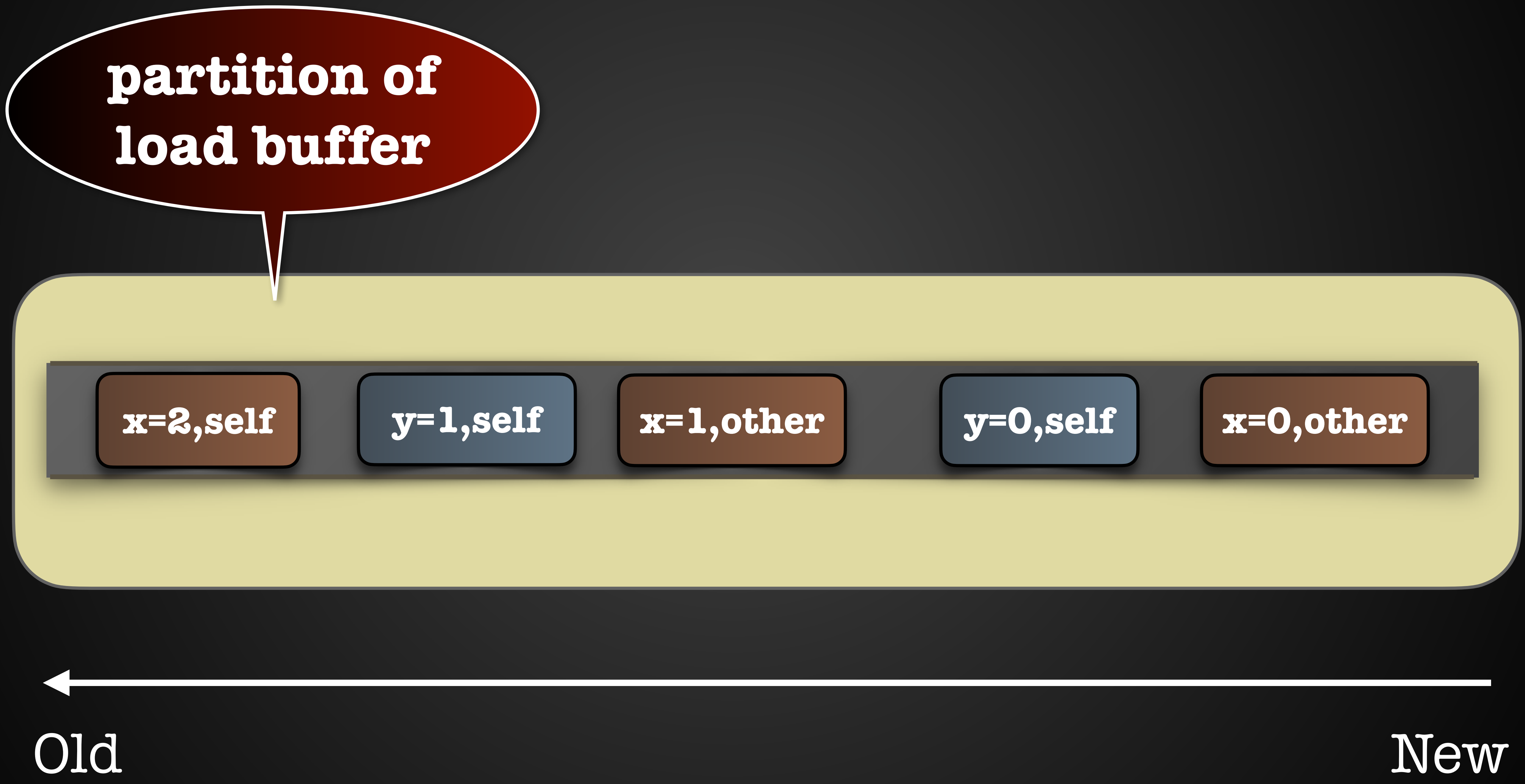
P1: w(x,2) → P1: r(y,0) → P2: w(y,1) → P2: w(x,1) → P2: r(x,2)

Classical TSO

Outline

- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- **Verification**
- Monitors
- Synthesis

Dual TSO - Monotonicity



partition of load buffer

x=2,self

y=1,self

x=1,other

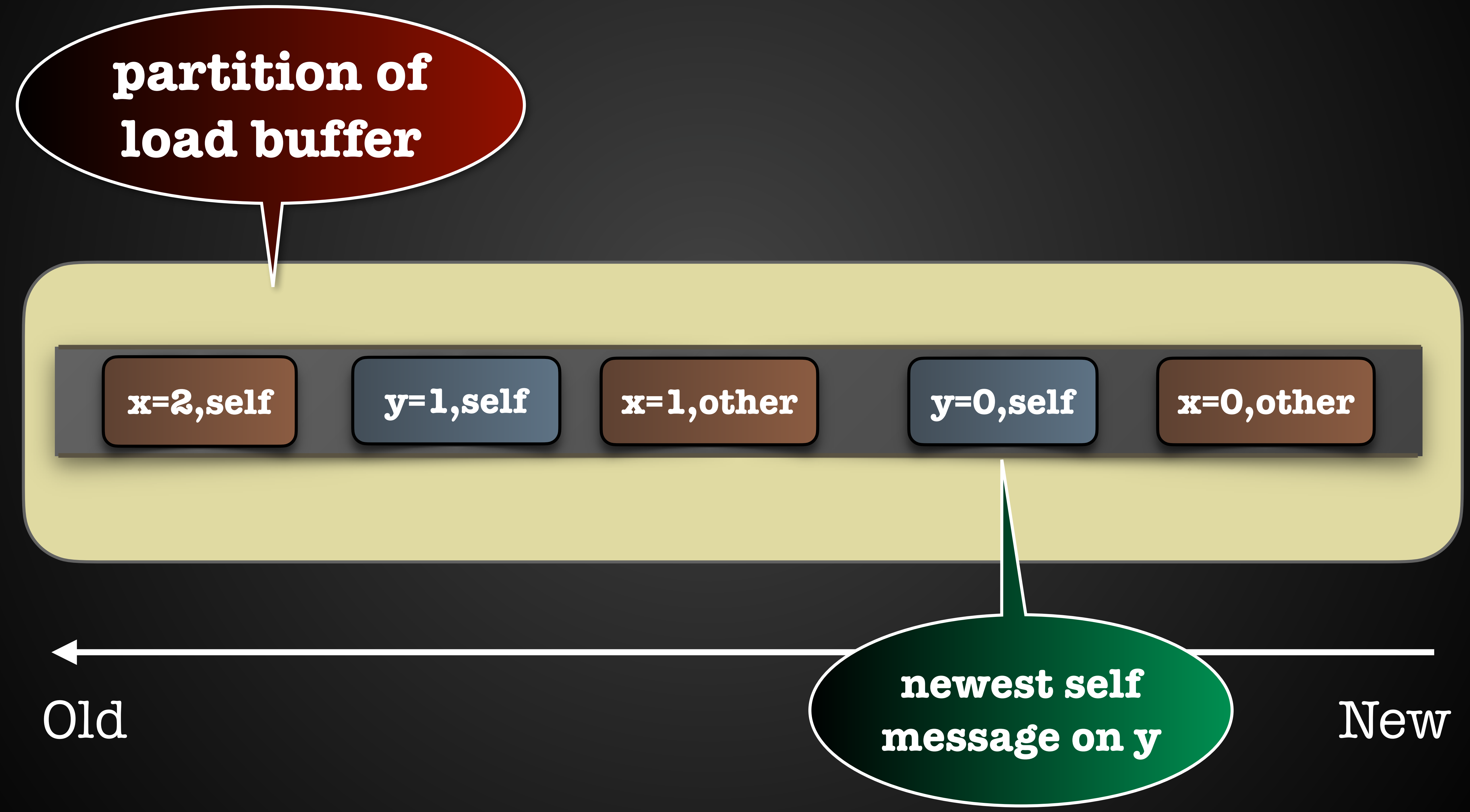
y=0,self

x=0,other

Old

New

Dual TSO - Monotonicity



partition of load buffer

x=2,self

y=1,self

x=1,other

y=0,self

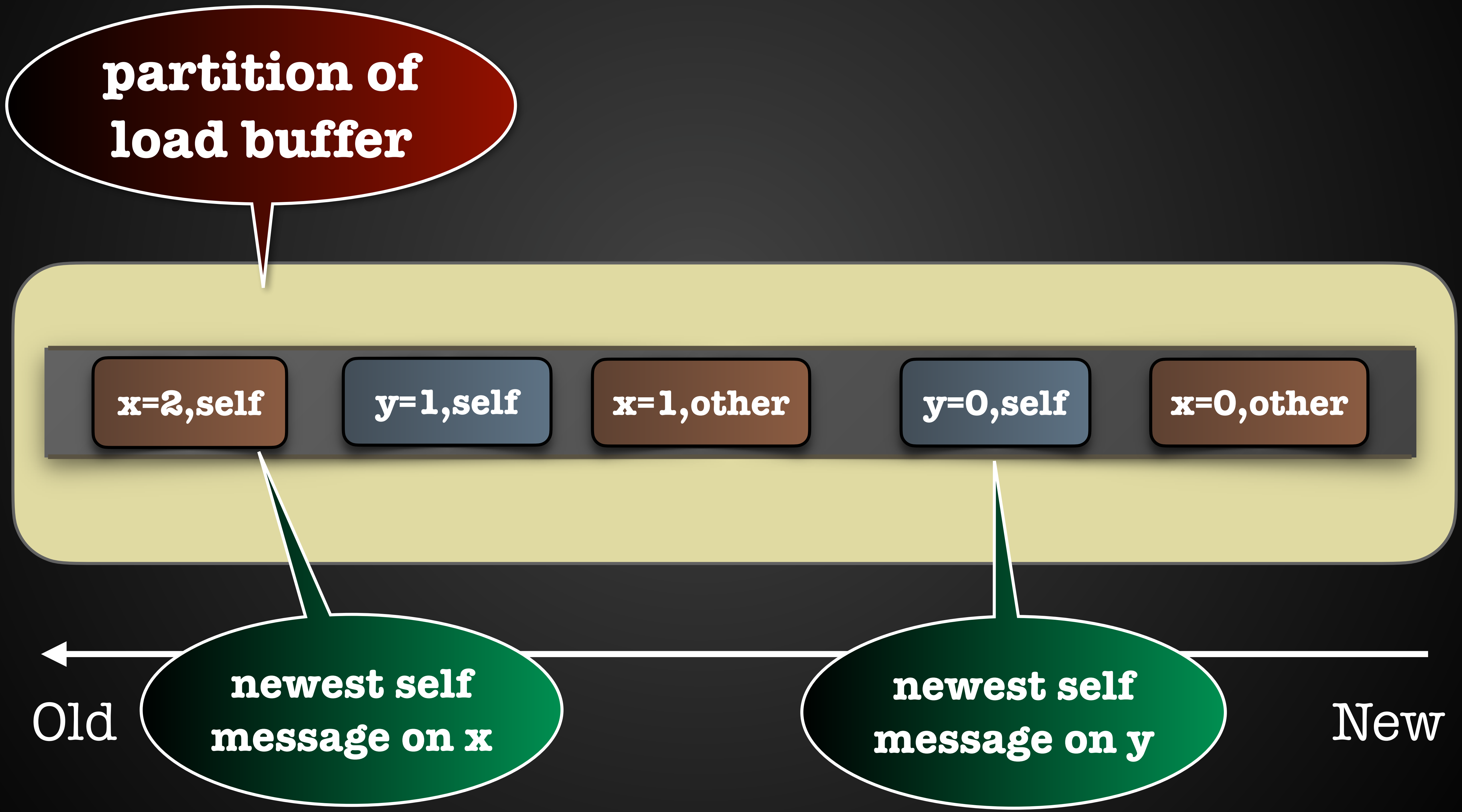
x=0,other

newest self message on y

Old

New

Dual TSO - Monotonicity



partition of load buffer

x=2,self

y=1,self

x=1,other

y=0,self

x=0,other

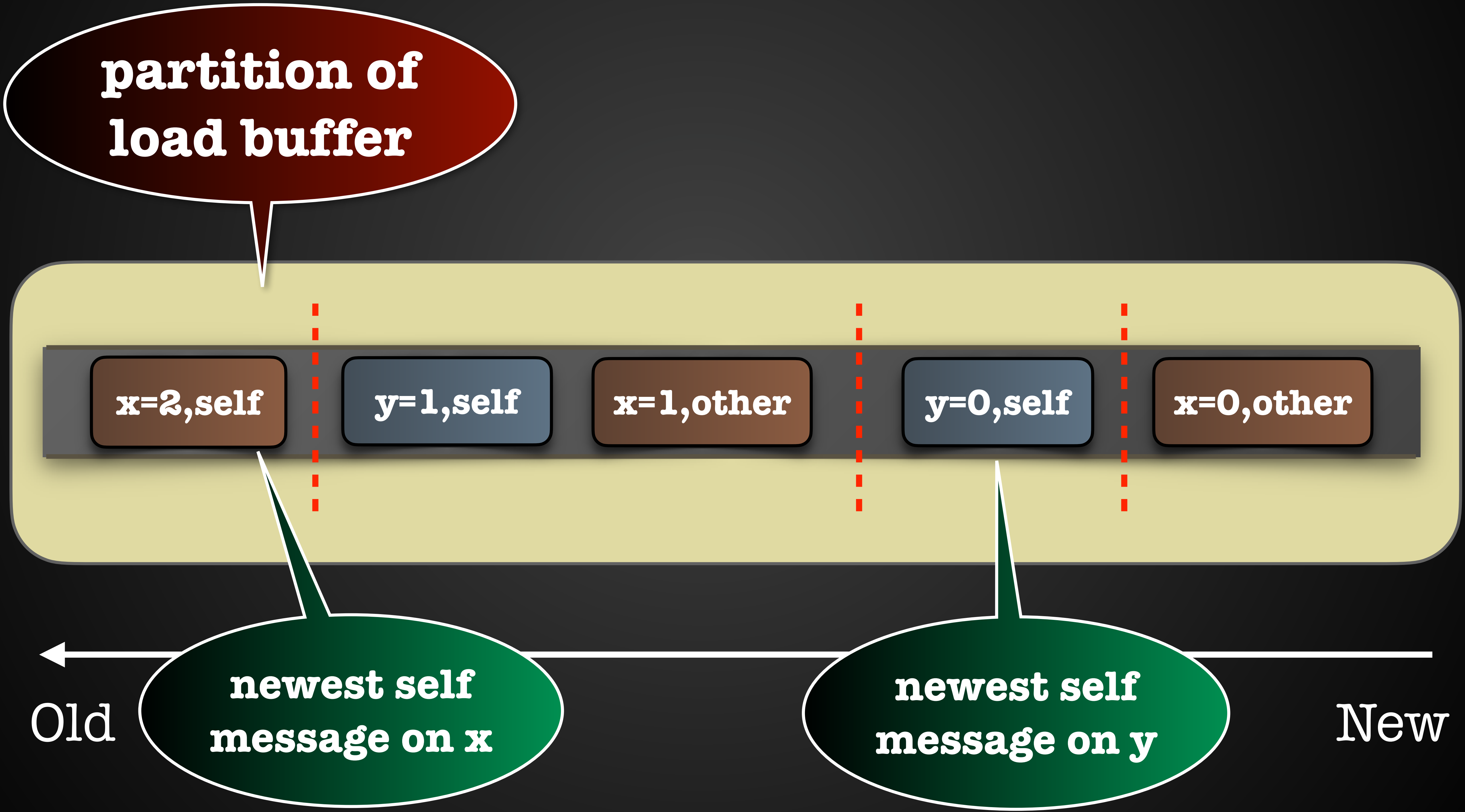
newest self message on x

newest self message on y

Old

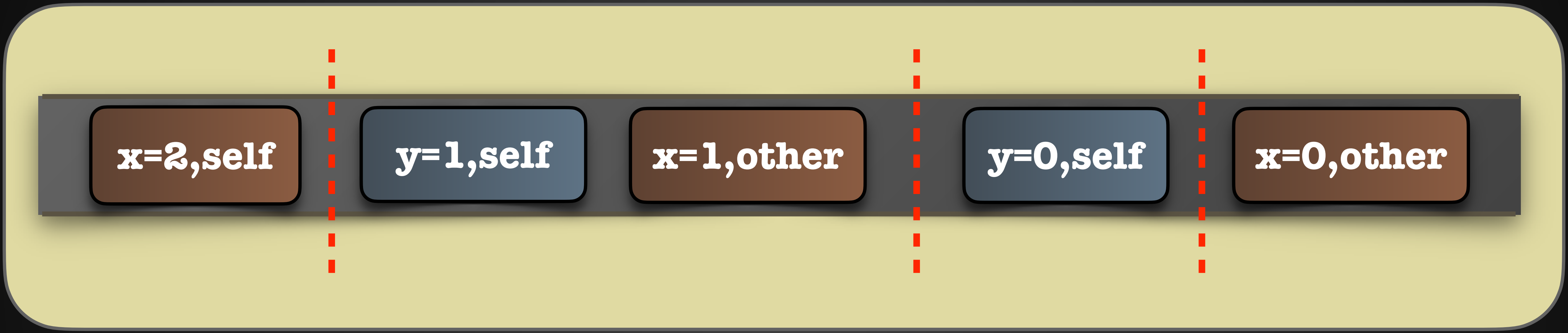
New

Dual TSO - Monotonicity



Dual TSO - Monotonicity

Ordering on Buffers



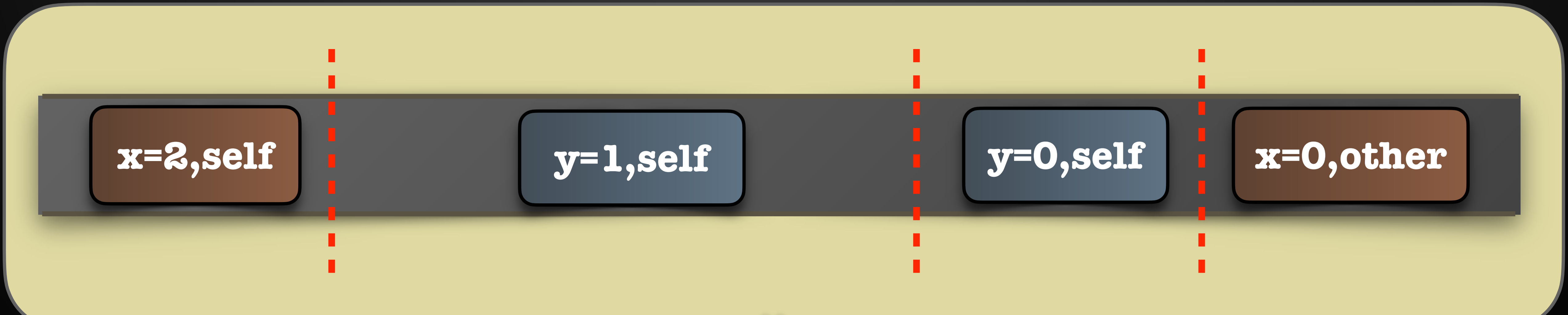
x=2,self

y=1,self

x=1,other

y=0,self

x=0,other



x=2,self

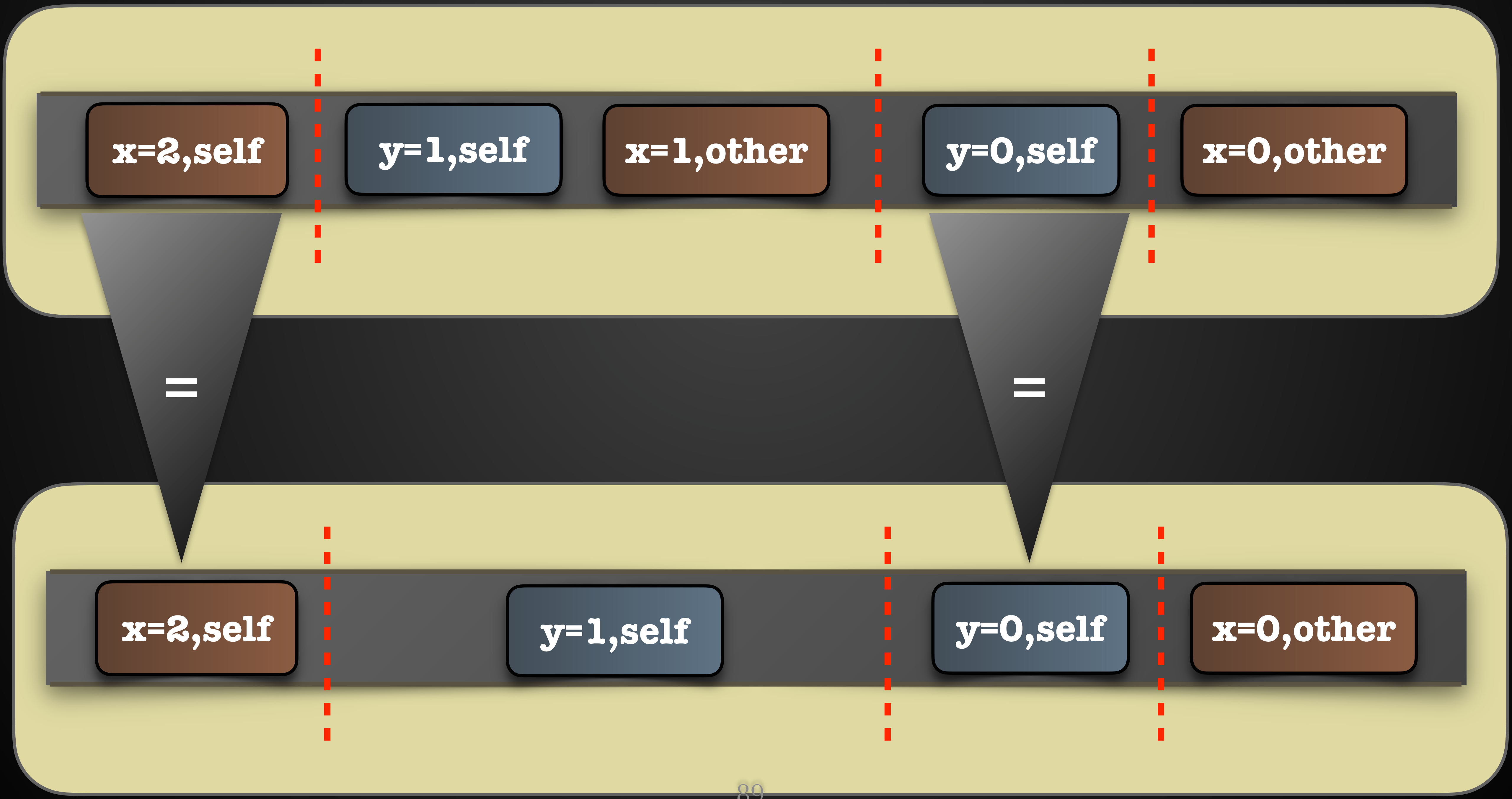
y=1,self

y=0,self

x=0,other

Dual TSO - Monotonicity

Ordering on Buffers



x=2,self y=1,self x=1,other y=0,self x=0,other

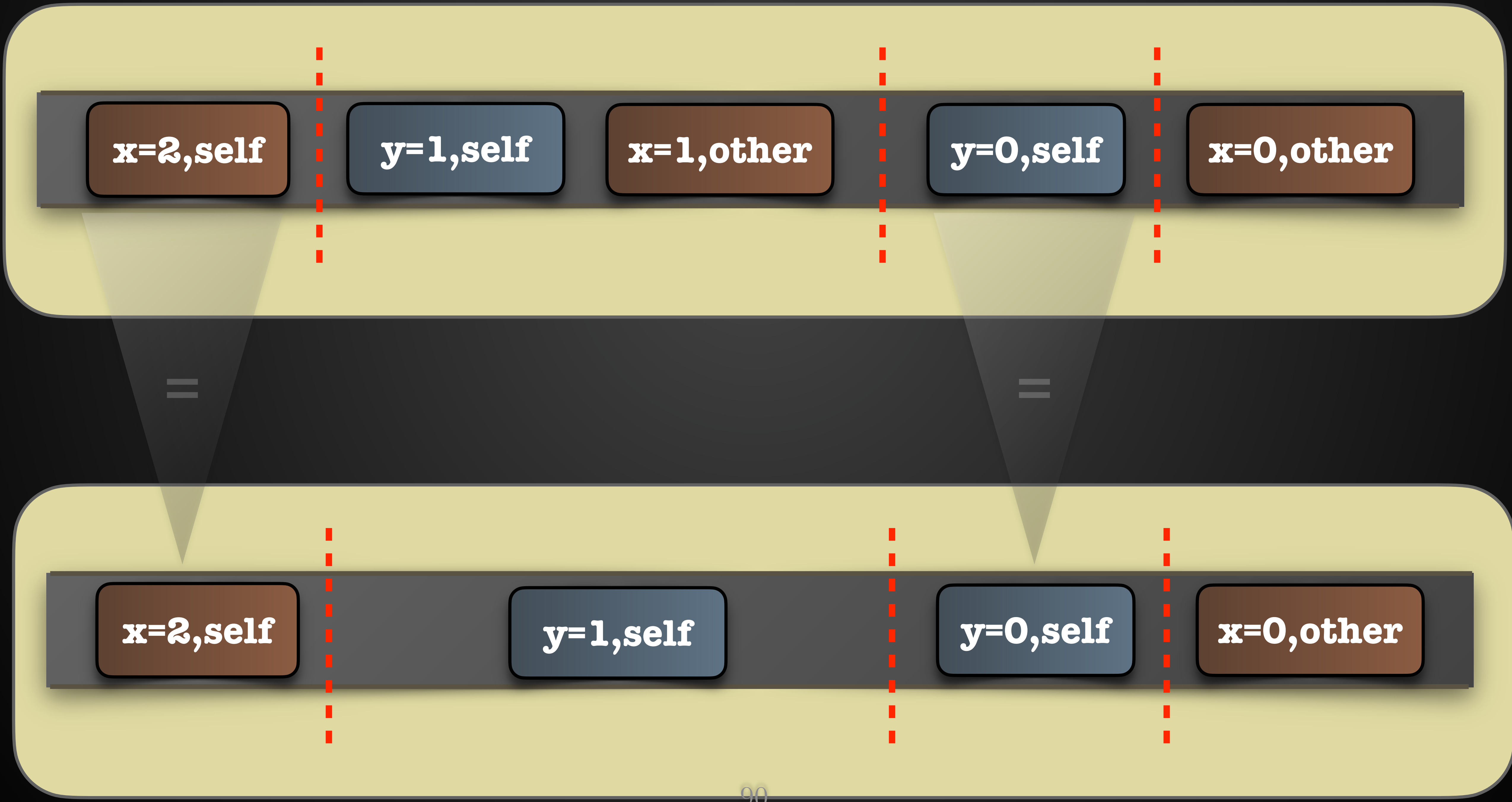
=

=

x=2,self y=1,self y=0,self x=0,other

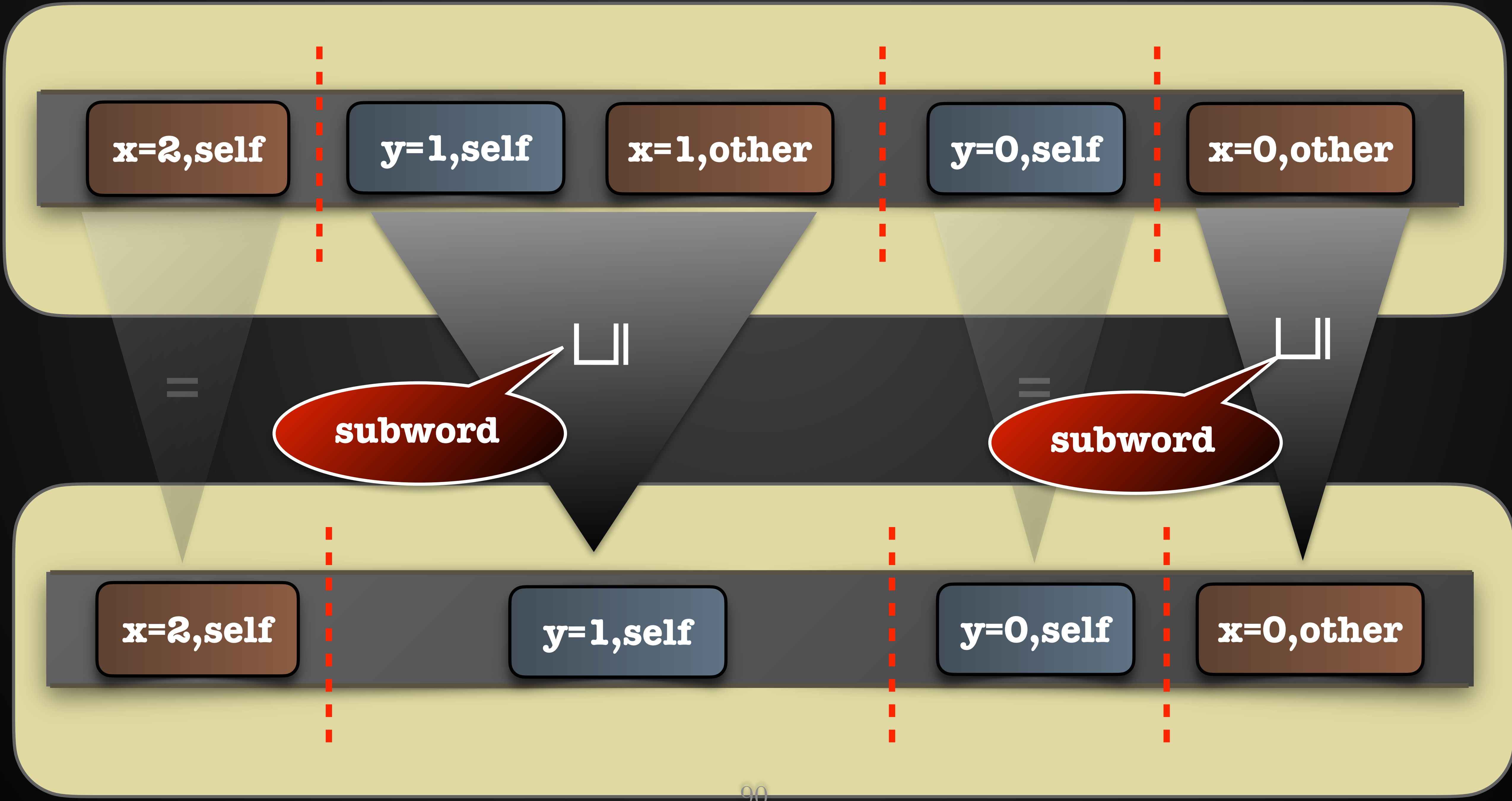
Dual TSO - Monotonicity

Ordering on Buffers



Dual TSO - Monotonicity

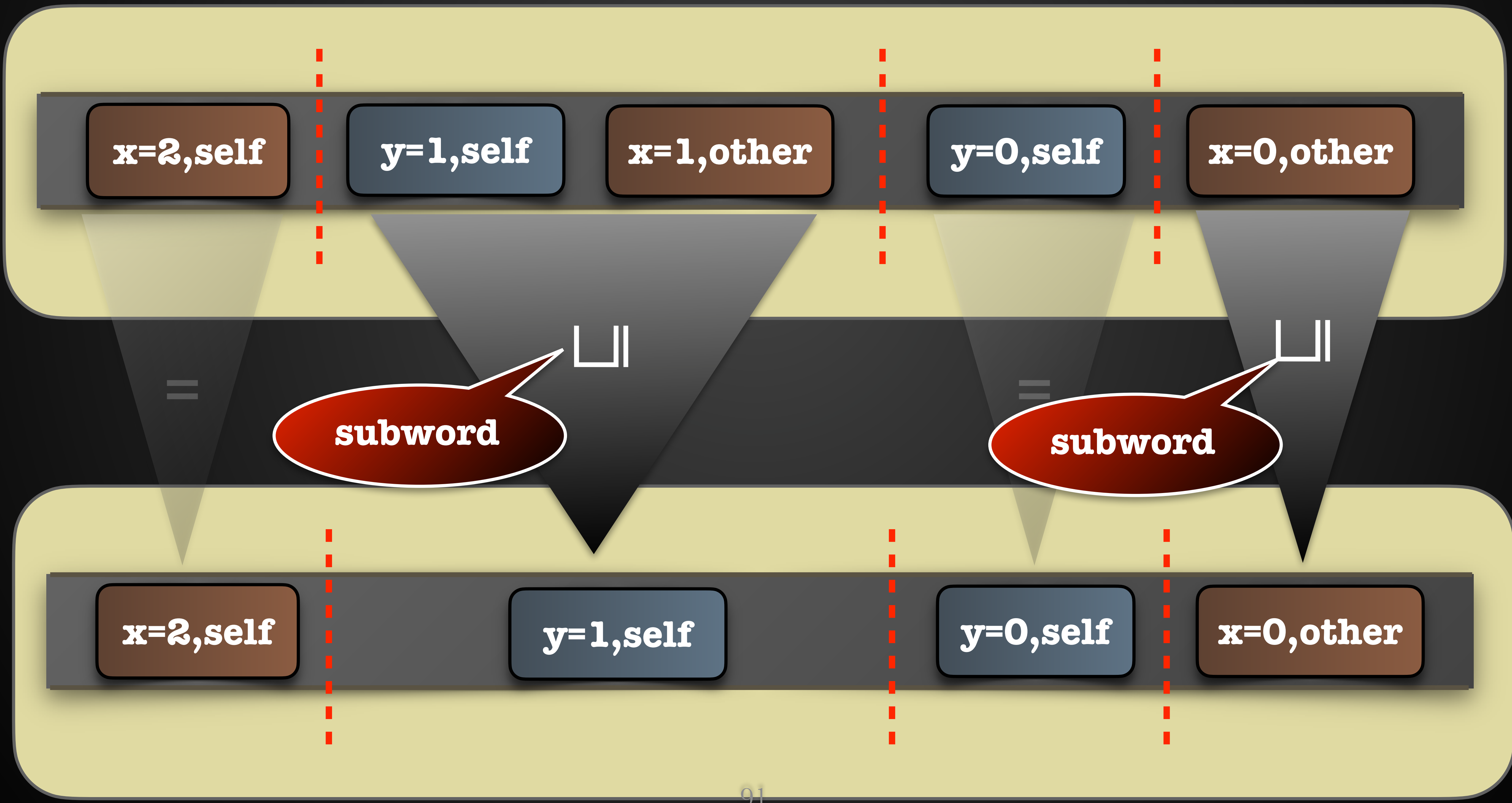
Ordering on Buffers



Dual TSO - Monotonicity

$ab \sqsubseteq xaybz$

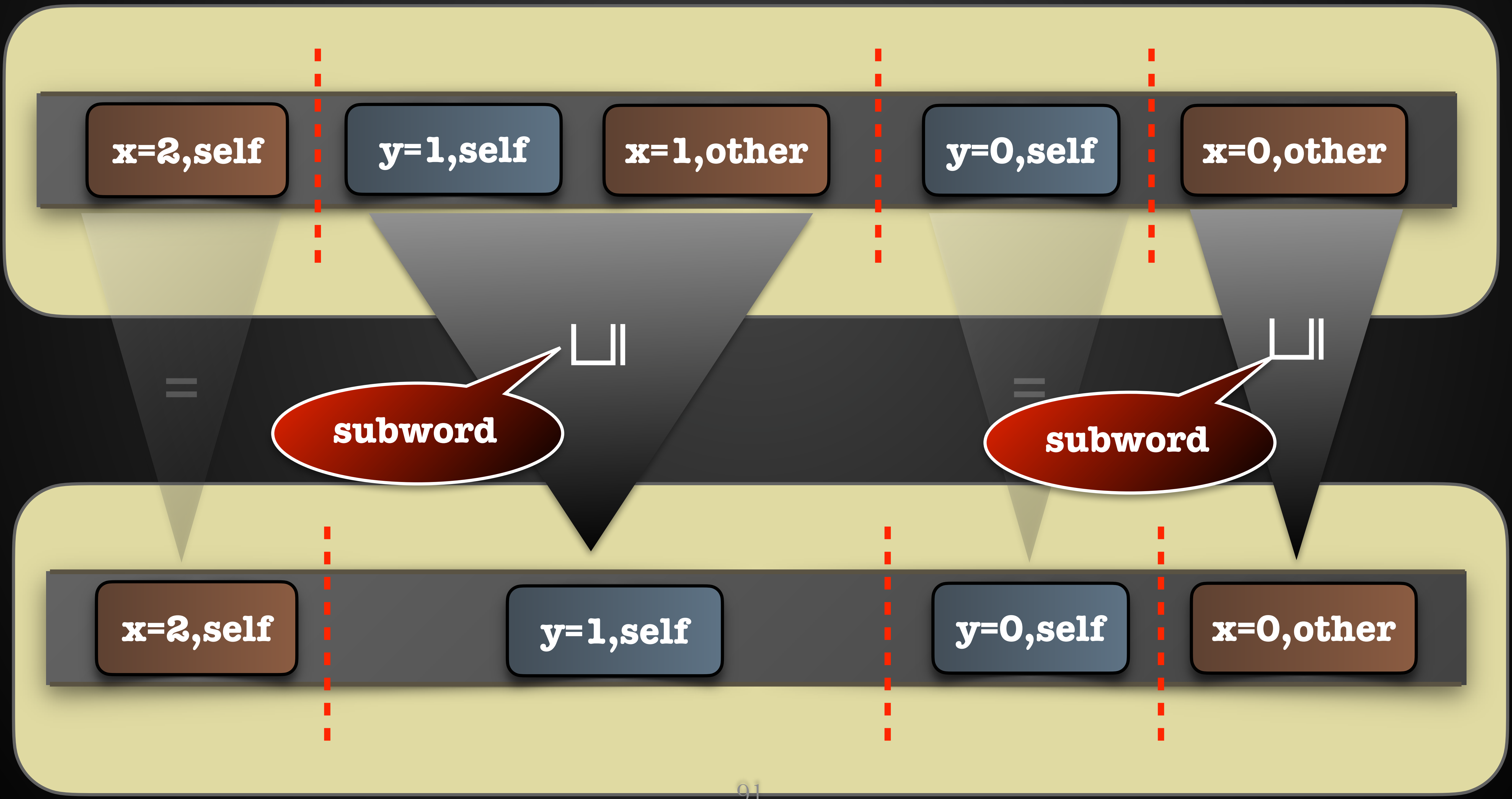
Ordering on Buffers



Dual TSO - Monotonicity

$ab \sqsubseteq xaybz$

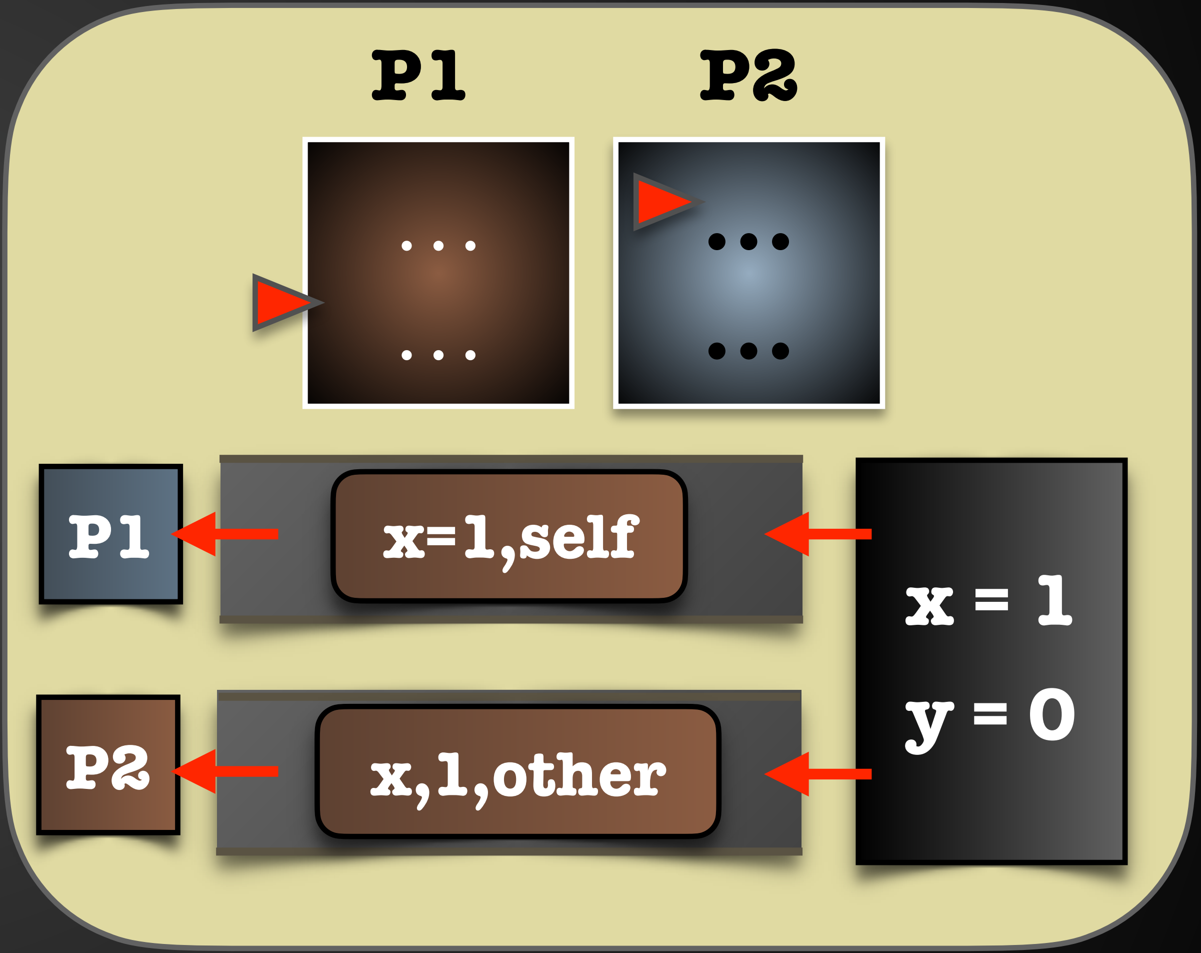
Ordering on Buffers



Dual TSO - Monotonicity

Ordering on Configurations

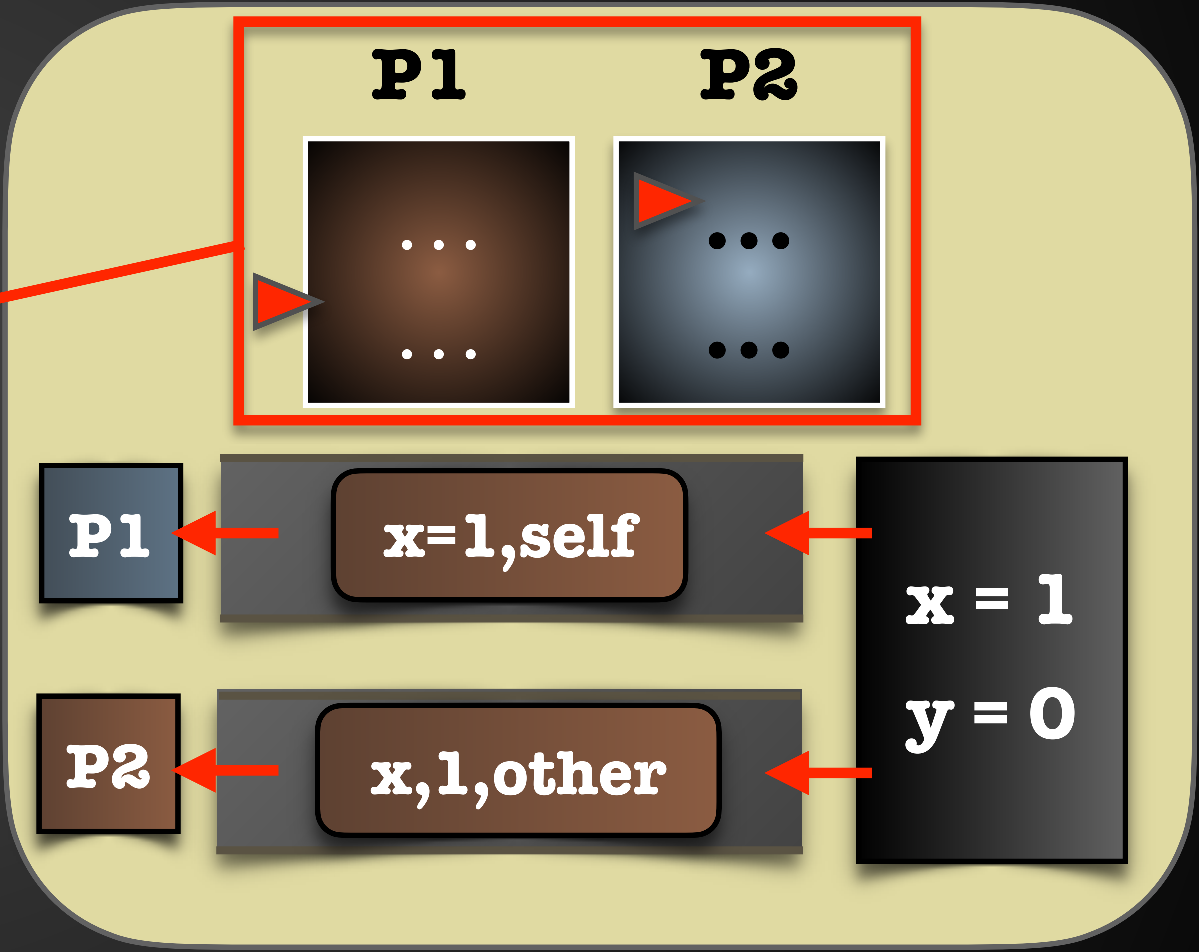
- identical process states
- identical memory state
- sub-word relation on buffers



Dual TSO - Monotonicity

Ordering on Configurations

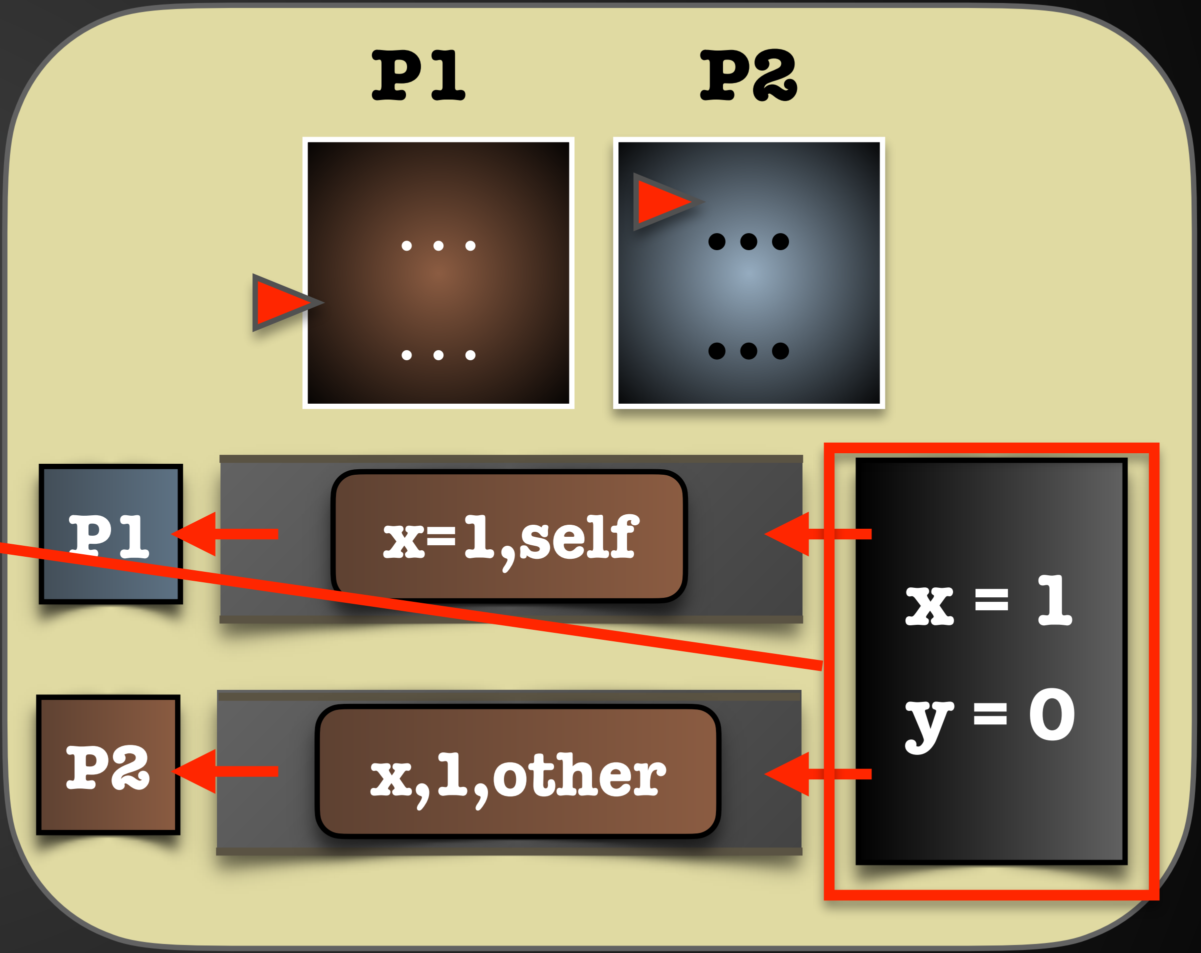
- identical process states
- identical memory state
- sub-word relation on buffers



Dual TSO - Monotonicity

Ordering on Configurations

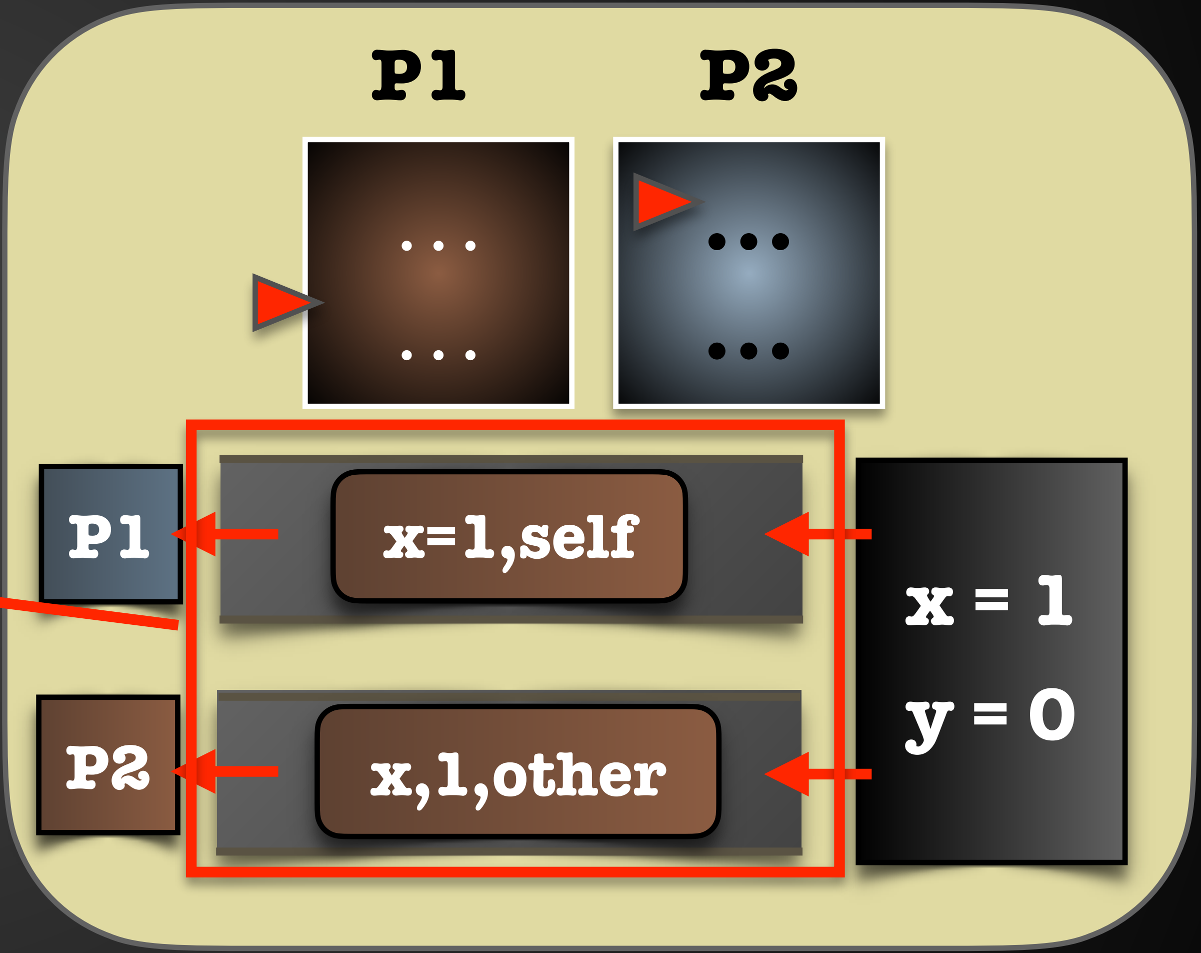
- identical process states
- identical memory state
- sub-word relation on buffers



Dual TSO - Monotonicity

Ordering on Configurations

- identical process states
- identical memory state
- sub-word relation on buffers



Dual TSO - Monotonicity

Ordering on Configurations

$$\begin{tikzcd}
\mathbf{c}_1 \arrow[r, "\text{Monotonicity}"'] \arrow[d, phantom, sloped, "\sqsubseteq"] & \mathbf{c}_2 \\
\mathbf{c}_3 &
\end{tikzcd}$$

Dual TSO - Monotonicity

Ordering on Configurations

$$\begin{tikzcd}
c_1 \arrow[r] \arrow[d, phantom, sloped, "\sqsubseteq"] \arrow[dr, phantom, "\text{Monotonicity}"] & c_2 \arrow[d, phantom, sloped, "\sqsubseteq"] \\
c_3 \arrow[r] & \exists\ c_4
\end{tikzcd}$$

Dual TSO - Monotonicity

- finite-state programs running on TSO:
  - reachability analysis terminates
  - reachability decidable

Experimental Results

Tool: Memorax

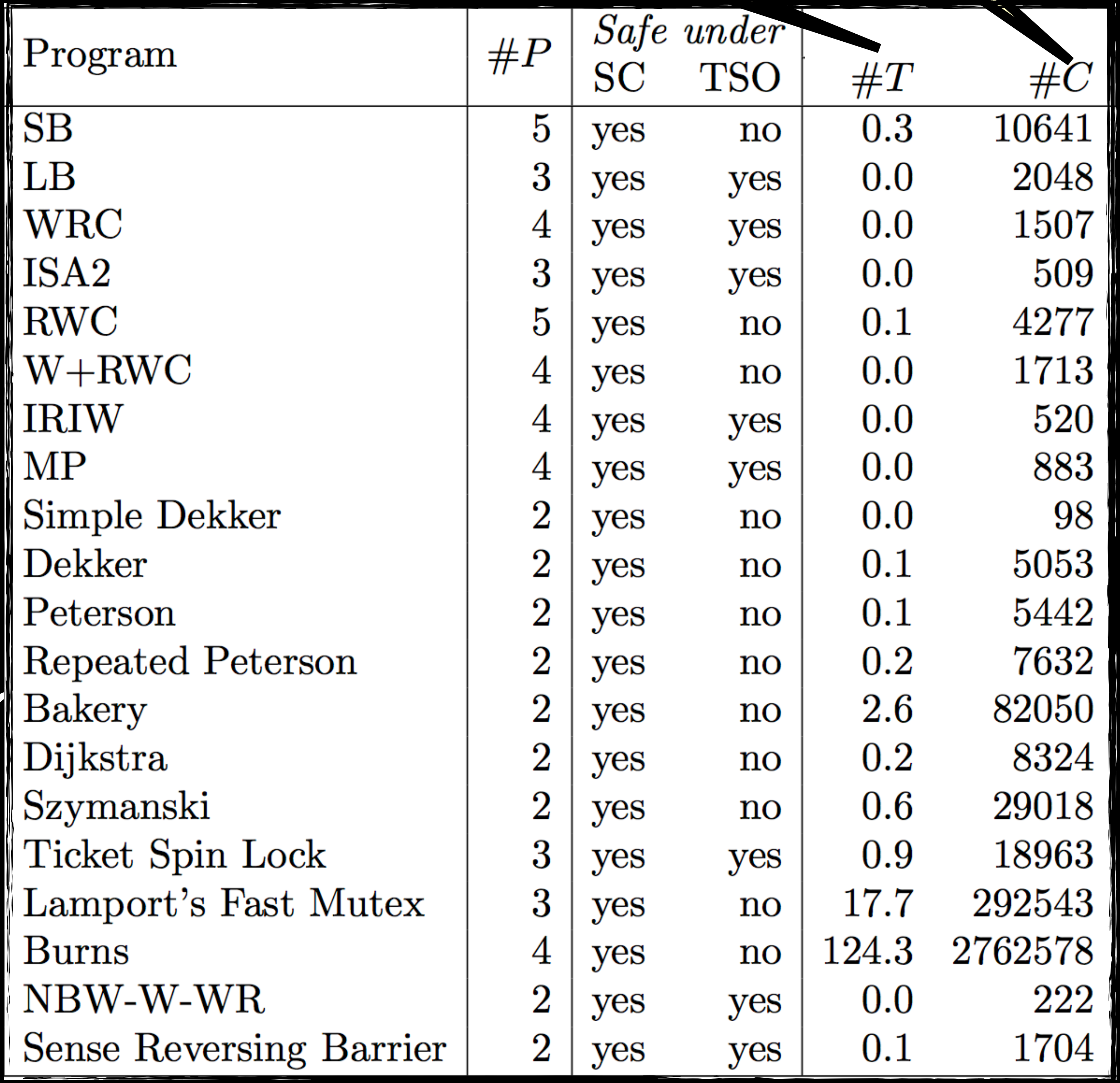
https://github.com/memorax/memorax

Experimental Results

time (secs)

# generated configurations

Tool: Memorax



Program	#P	*Safe under*		$\#T$	$\#C$
		SC	TSO		
SB	5	yes	no	0.3	10641
LB	3	yes	yes	0.0	2048
WRC	4	yes	yes	0.0	1507
ISA2	3	yes	yes	0.0	509
RWC	5	yes	no	0.1	4277
W+RWC	4	yes	no	0.0	1713
IRIW	4	yes	yes	0.0	520
MP	4	yes	yes	0.0	883
Simple Dekker	2	yes	no	0.0	98
Dekker	2	yes	no	0.1	5053
Peterson	2	yes	no	0.1	5442
Repeated Peterson	2	yes	no	0.2	7632
Bakery	2	yes	no	2.6	82050
Dijkstra	2	yes	no	0.2	8324
Szymanski	2	yes	no	0.6	29018
Ticket Spin Lock	3	yes	yes	0.9	18963
Lamport's Fast Mutex	3	yes	no	17.7	292543
Burns	4	yes	no	124.3	2762578
NBW-W-WR	2	yes	yes	0.0	222
Sense Reversing Barrier	2	yes	yes	0.1	1704

standard benchmarks: litmus tests and mutual exclusion

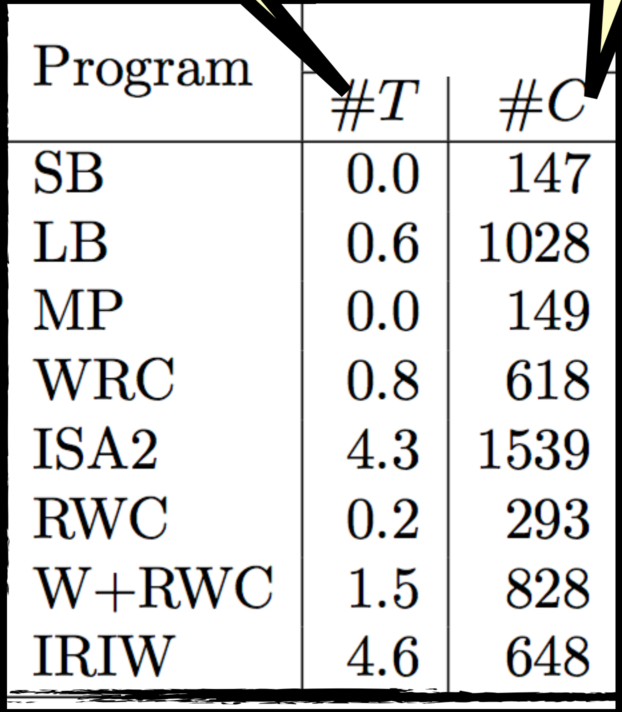
Experimental Results

Tool: Memorax

parameterized verification

time (secs)

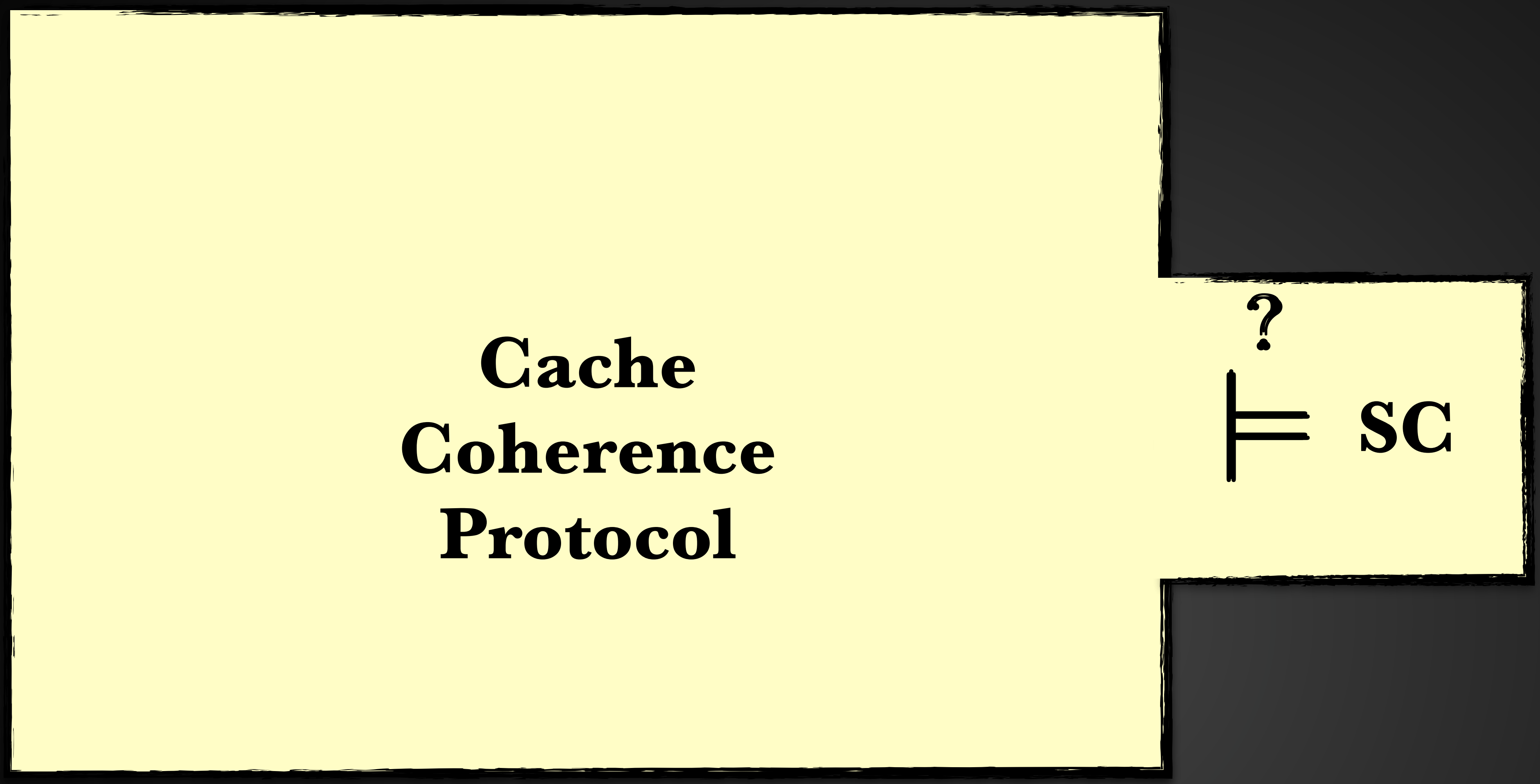
# generated configurations



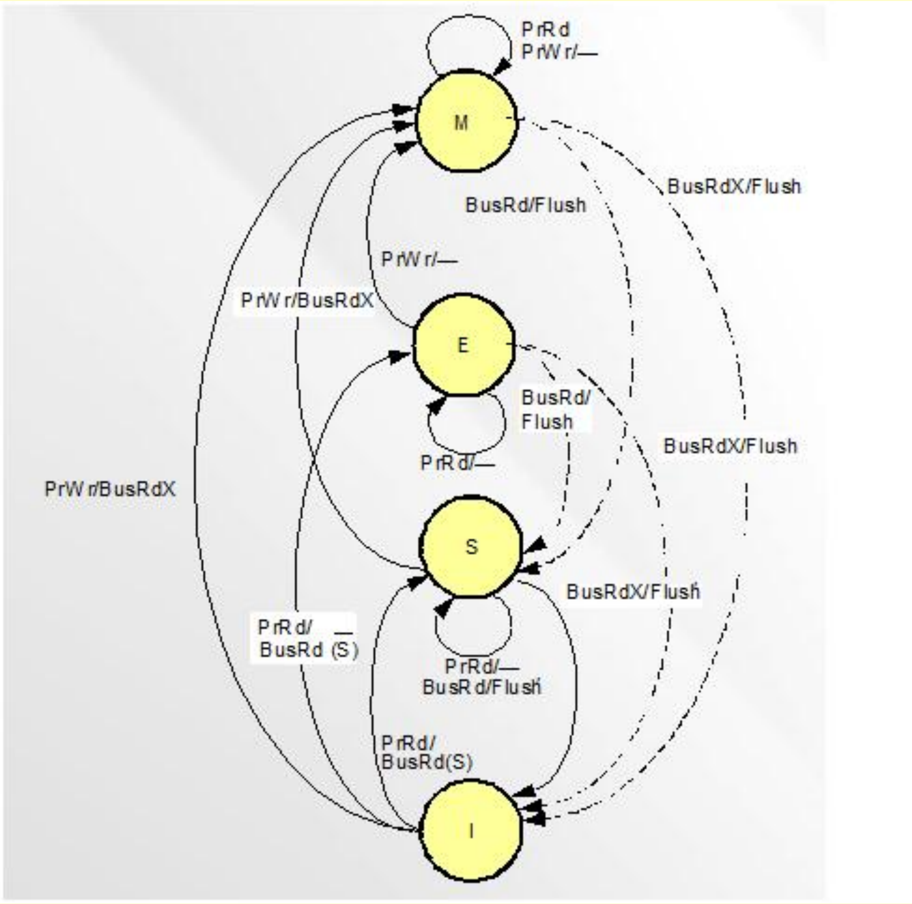
Program	#T	#C
SB	0.0	147
LB	0.6	1028
MP	0.0	149
WRC	0.8	618
ISA2	4.3	1539
RWC	0.2	293
W+RWC	1.5	828
IRIW	4.6	648

Outline

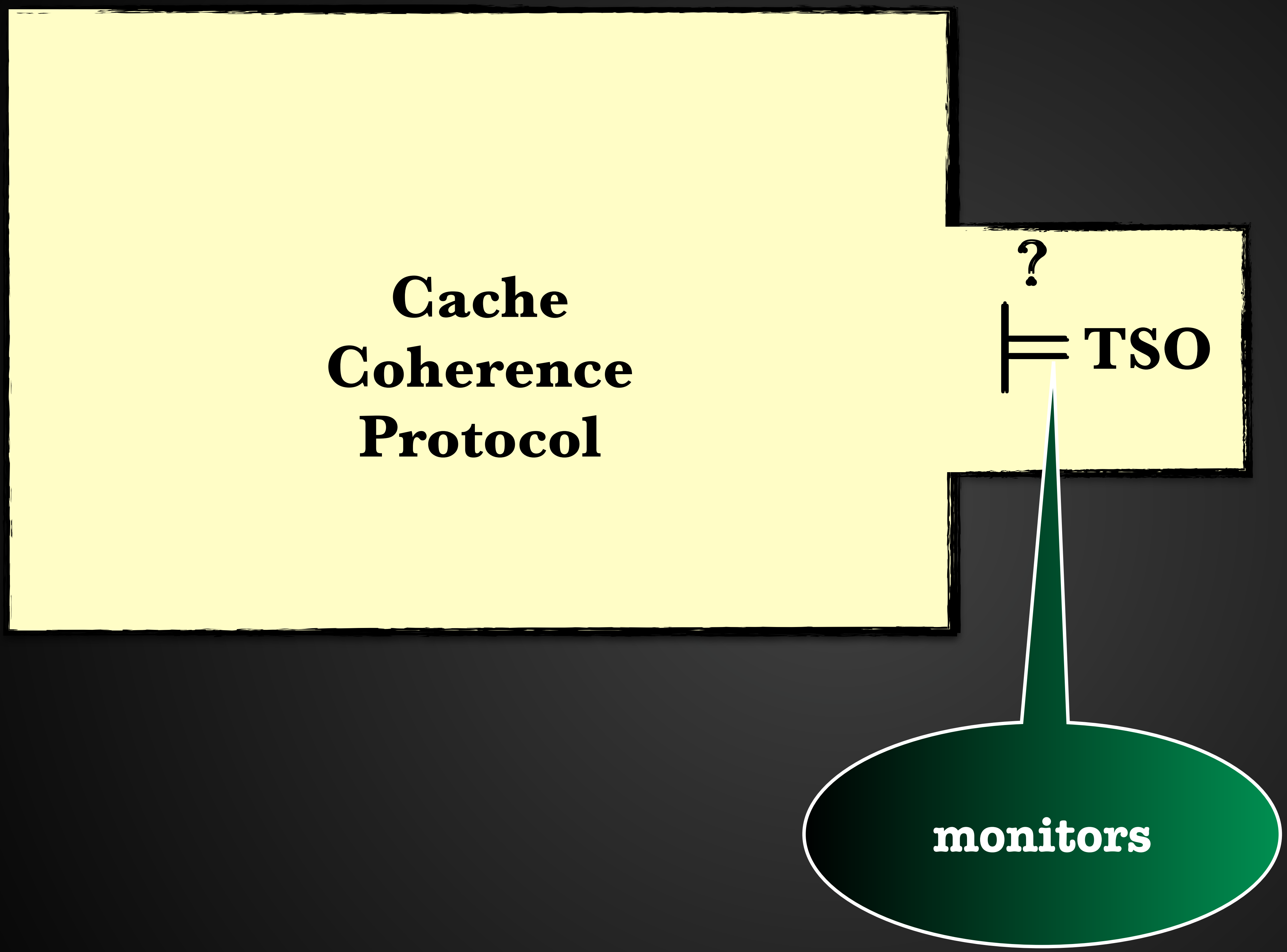
- Weak Consistency
- Total Store Order (TSO)
- Dual TSO
- Verification
- **Monitors**
- Synthesis



Cache Coherence Protocol $\overset{?}{\models}$ SC



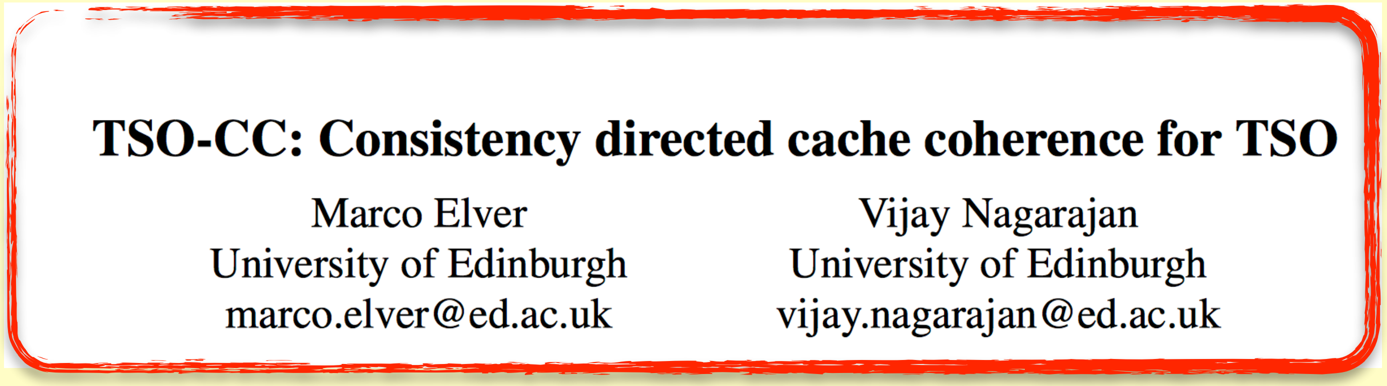
$\overset{?}{\models}$ **SC**



Cache Coherence Protocol

$\overset{?}{\models}$ TSO

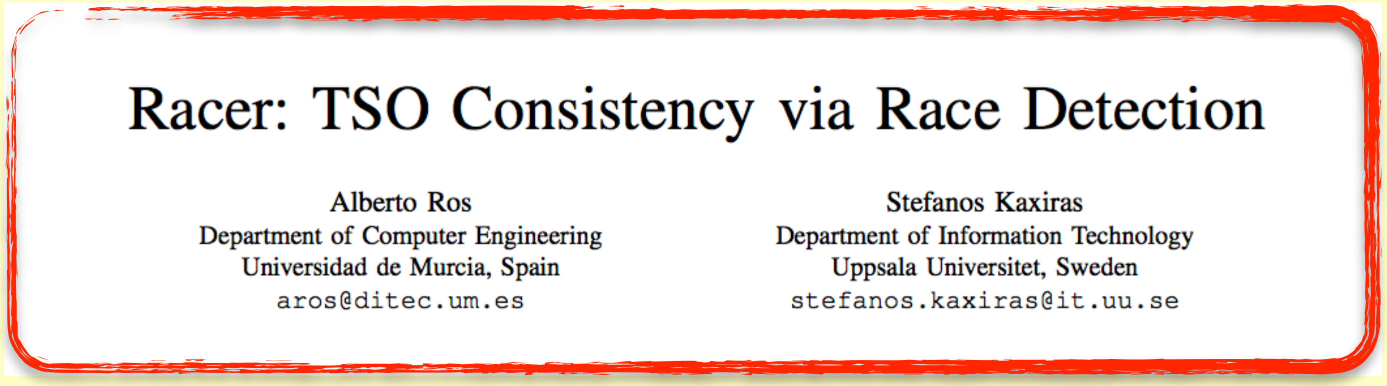
monitors



TSO-CC: Consistency directed cache coherence for TSO

Marco Elver
University of Edinburgh
marco.elver@ed.ac.uk

Vijay Nagarajan
University of Edinburgh
vijay.nagarajan@ed.ac.uk



Racer: TSO Consistency via Race Detection

Alberto Ros
Department of Computer Engineering
Universidad de Murcia, Spain
aros@ditec.um.es

Stefanos Kaxiras
Department of Information Technology
Uppsala Universitet, Sweden
stefanos.kaxiras@it.uu.se

$\overset{?}{\models}$ **TSO**

monitors

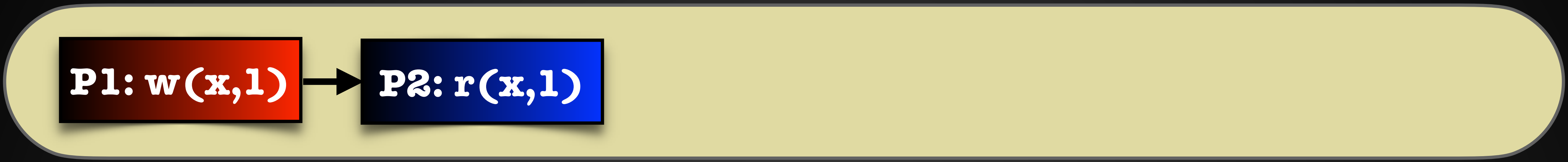
TSO-Counter-Examples

TSO-Counter-Examples

TSO-Counter-Examples

P1: w(x,1)

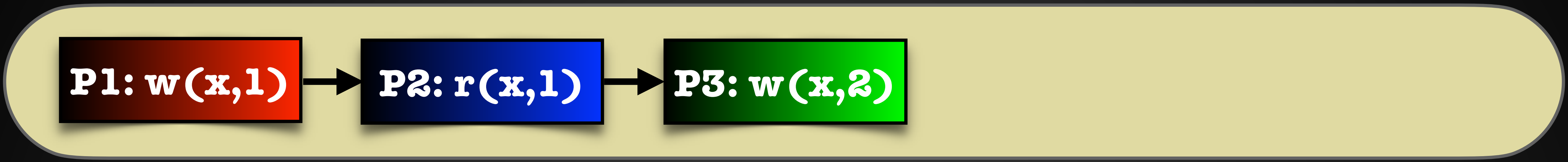
TSO-Counter-Examples



P1: w(x,1)

P2: r(x,1)

TSO-Counter-Examples

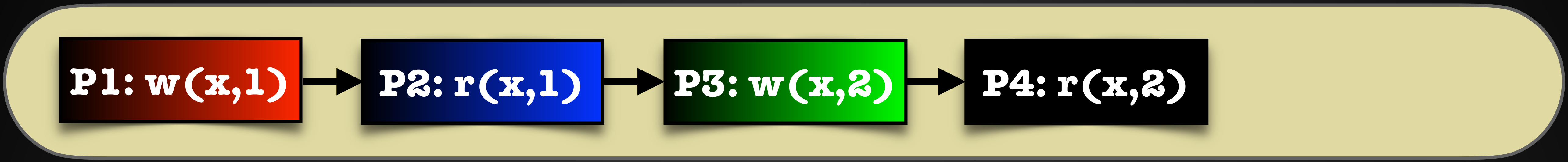


P1: w(x,1)

P2: r(x,1)

P3: w(x,2)

TSO-Counter-Examples

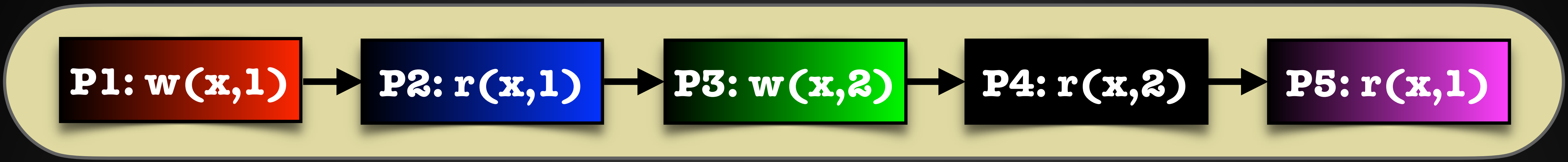


P1: w(x,1)

P2: r(x,1)

P3: w(x,2)

P4: r(x,2)



P1: w(x,1)

P2: r(x,1)

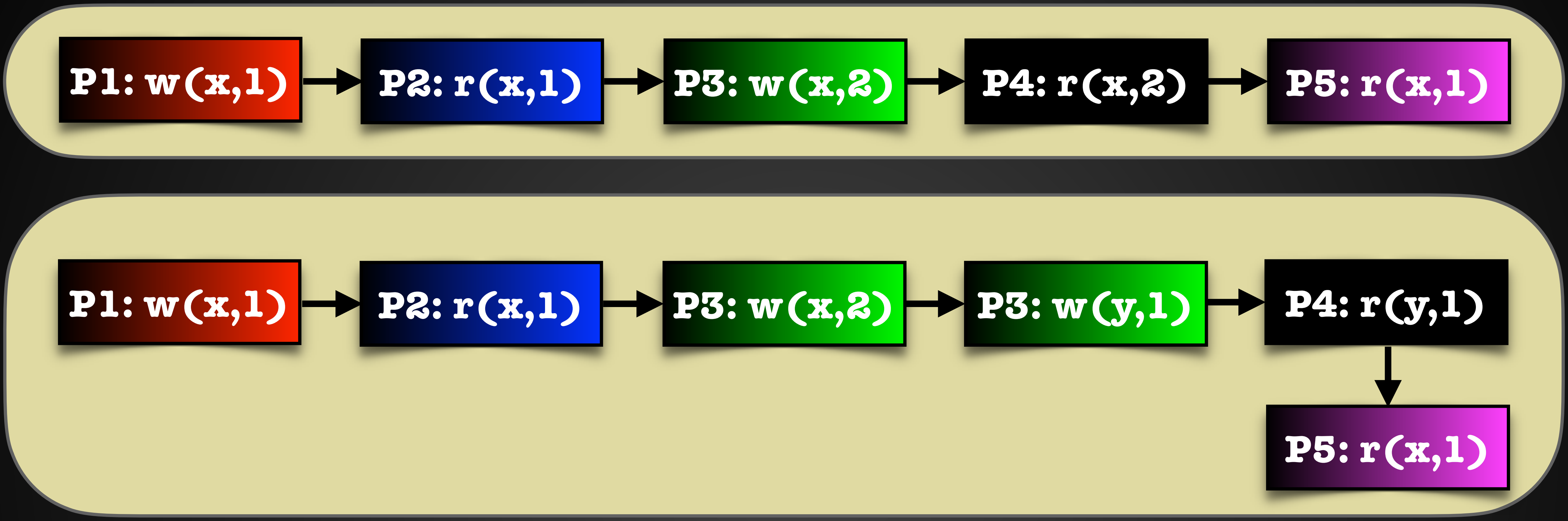
P3: w(x,2)

P4: r(x,2)

P5: r(x,1)

TSO-Counter-Examples

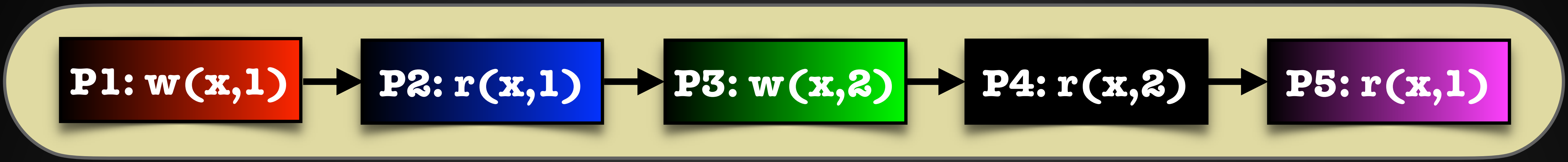
TSO-Counter-Examples

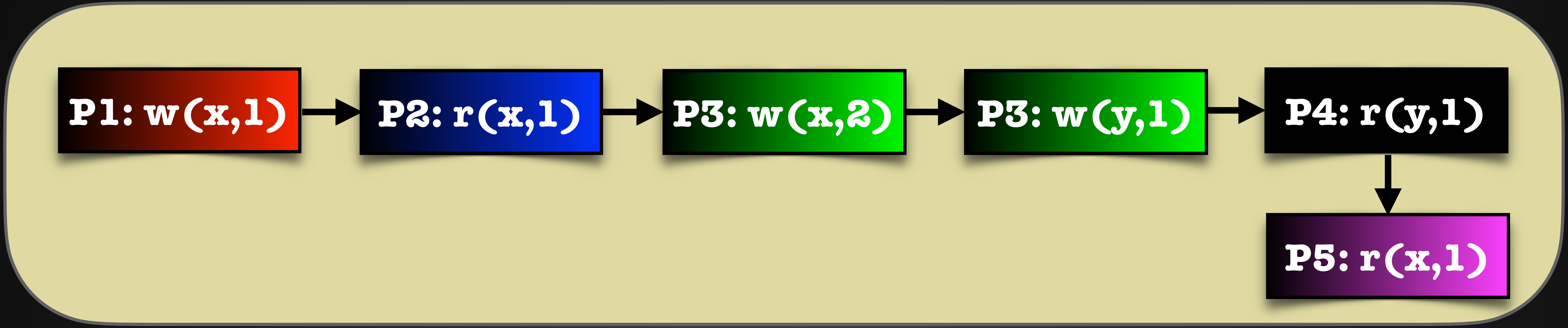


P1: w(x,1) → P2: r(x,1) → P3: w(x,2) → P4: r(x,2) → P5: r(x,1)

P1: w(x,1) → P2: r(x,1) → P3: w(x,2) → P3: w(y,1) → P4: r(y,1) → P5: r(x,1)

TSO-Counter-Examples





TSO $\equiv$ 12 counter-examples

Conclusion

- **Weak Consistency**
- **Total Store Order (TSO)**
- **Dual TSO**

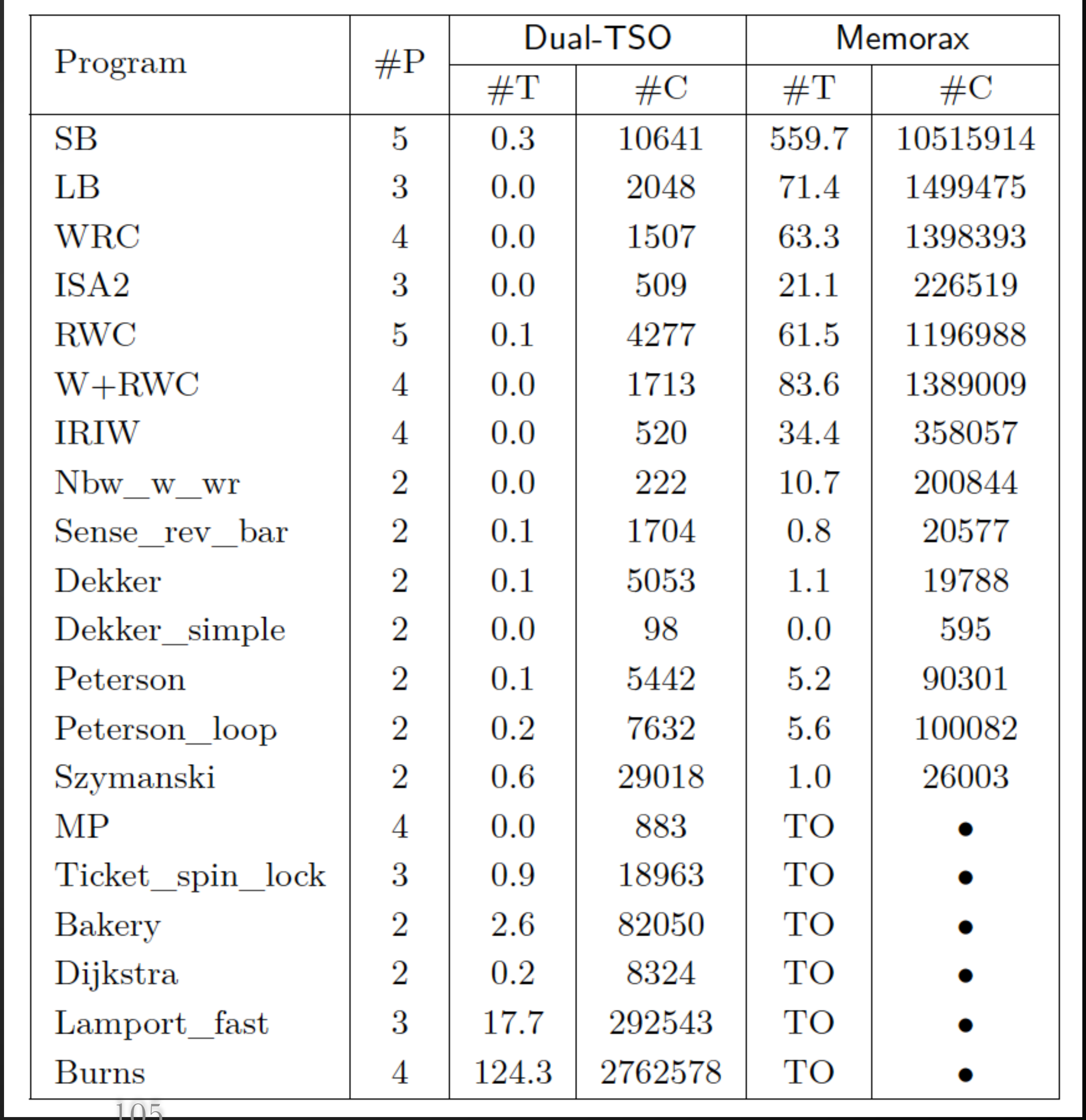
Current Work

- **Weak Cache Verification**
- **Other memory models, e.g., POWER, ARM, C11**
- **Stateless Model Checking**
- **Monitor Design**

Experimental Results

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**



Program	#P	Dual-TSO		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	•
Ticket_spin_lock	3	0.9	18963	TO	•
Bakery	2	2.6	82050	TO	•
Dijkstra	2	0.2	8324	TO	•
Lamport_fast	3	17.7	292543	TO	•
Burns	4	124.3	2762578	TO	•

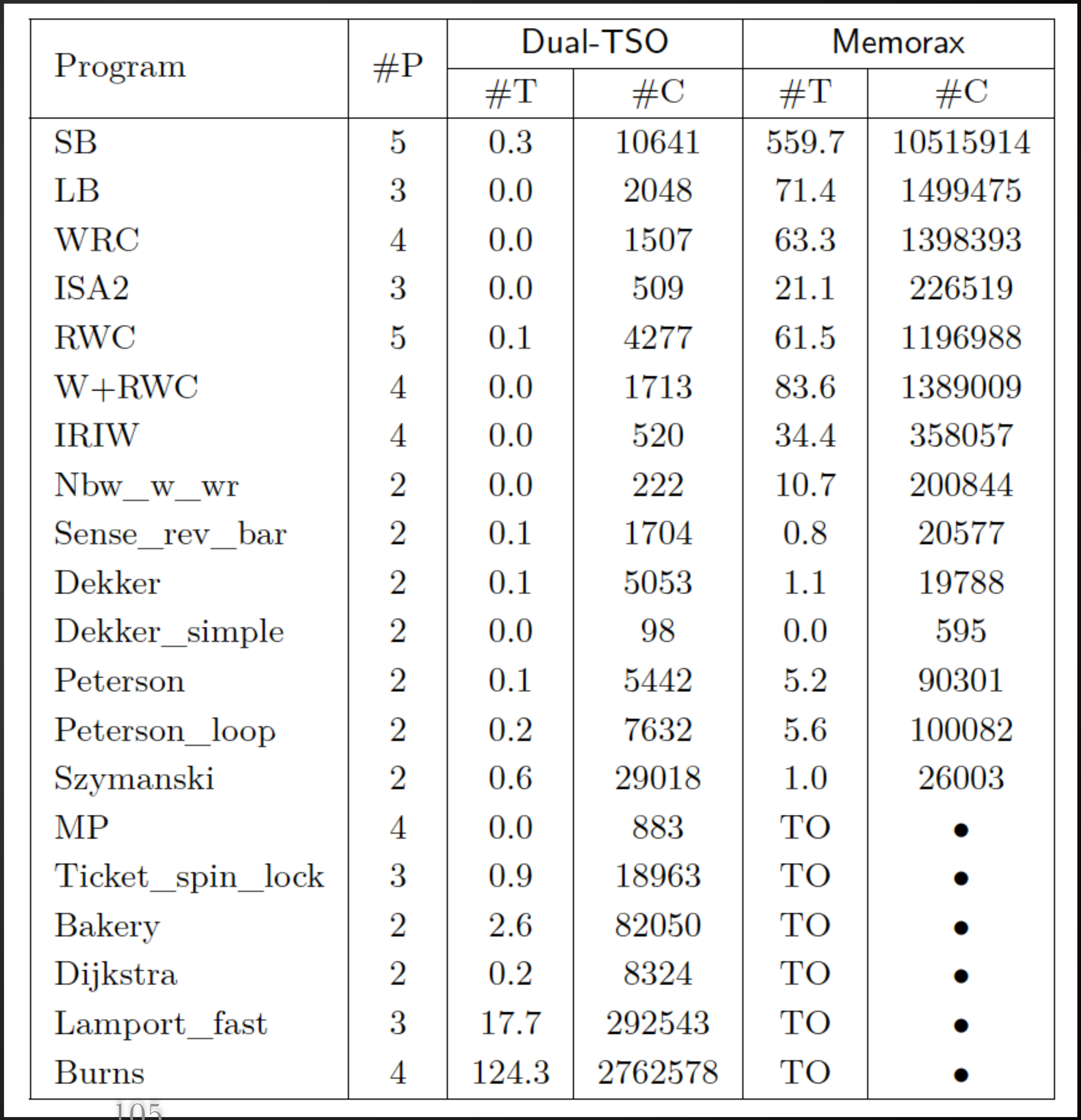
https://www.it.uu.se/katalog/tuang296/dual-tso

Experimental Results

Single buffer approach (exact method [TACAS12+13])

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**



Program	#P	Dual-TSO		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	•
Ticket_spin_lock	3	0.9	18963	TO	•
Bakery	2	2.6	82050	TO	•
Dijkstra	2	0.2	8324	TO	•
Lamport_fast	3	17.7	292543	TO	•
Burns	4	124.3	2762578	TO	•

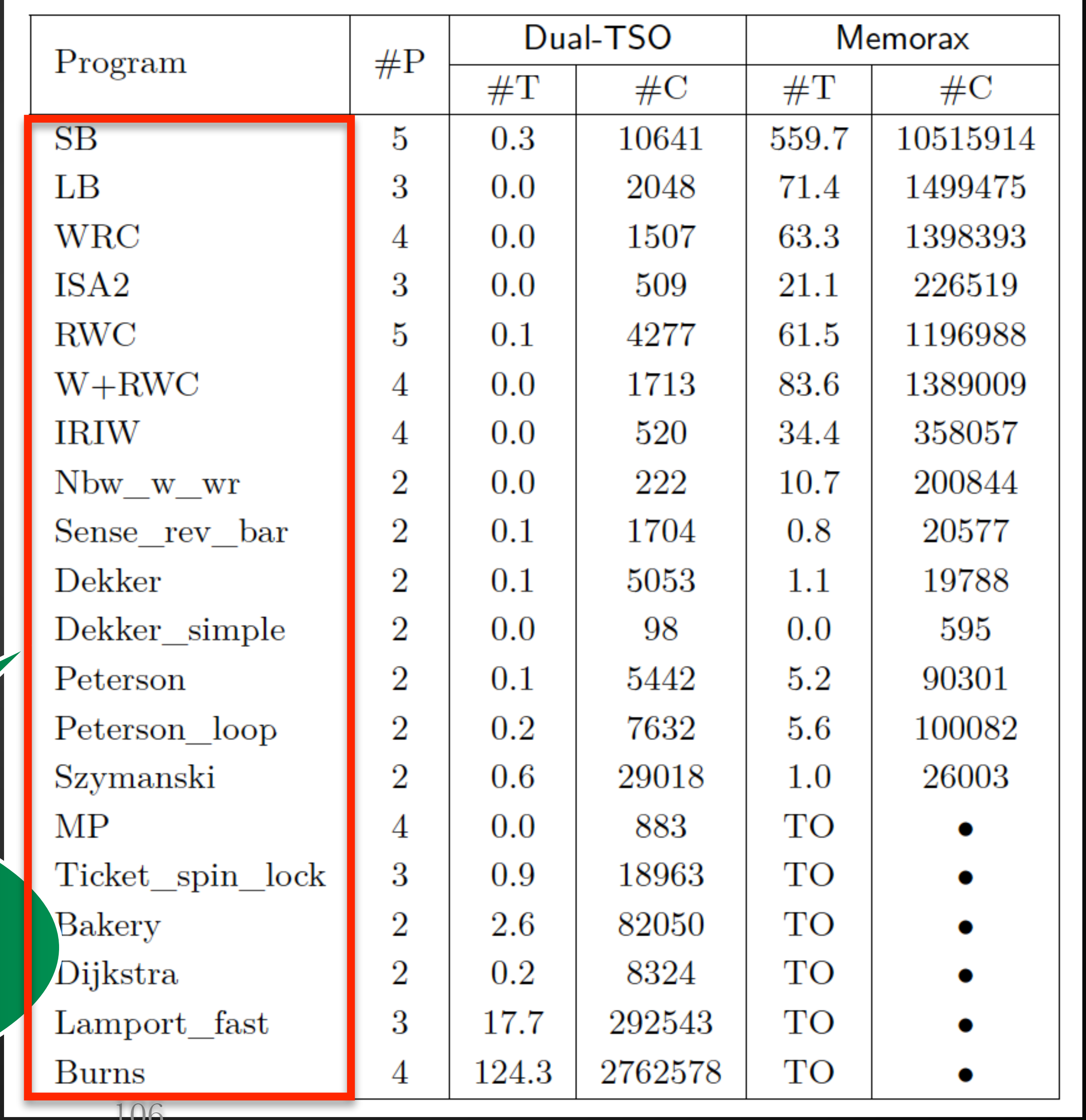
https://www.it.uu.se/katalog/tuang296/dual-tso

Experimental Results

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**

standard benchmarks: litmus tests and mutual exclusion algorithms



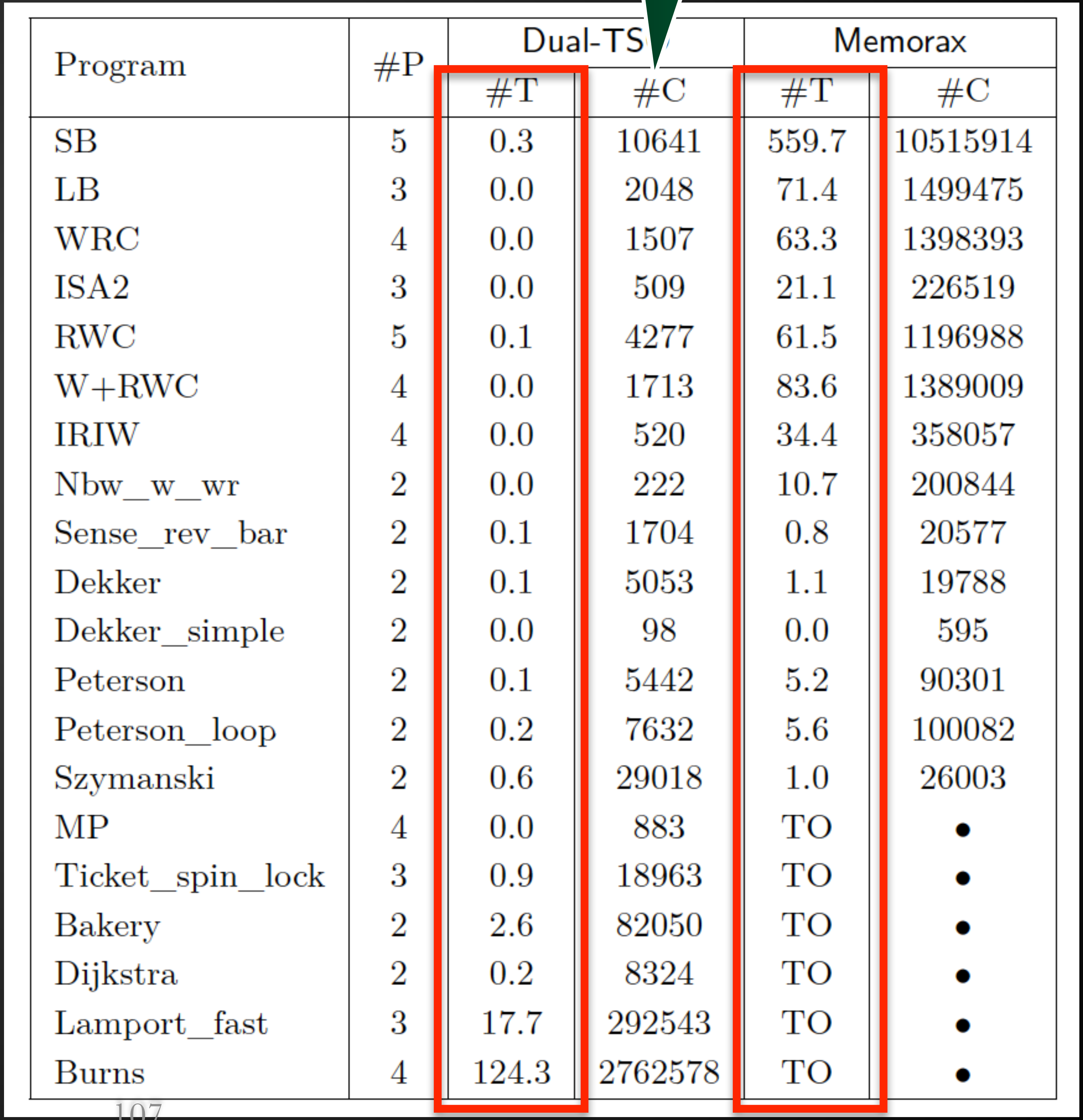
Program	#P	Dual-TSO		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	●
Ticket_spin_lock	3	0.9	18963	TO	●
Bakery	2	2.6	82050	TO	●
Dijkstra	2	0.2	8324	TO	●
Lamport_fast	3	17.7	292543	TO	●
Burns	4	124.3	2762578	TO	●

Experimental R

running time in seconds

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**



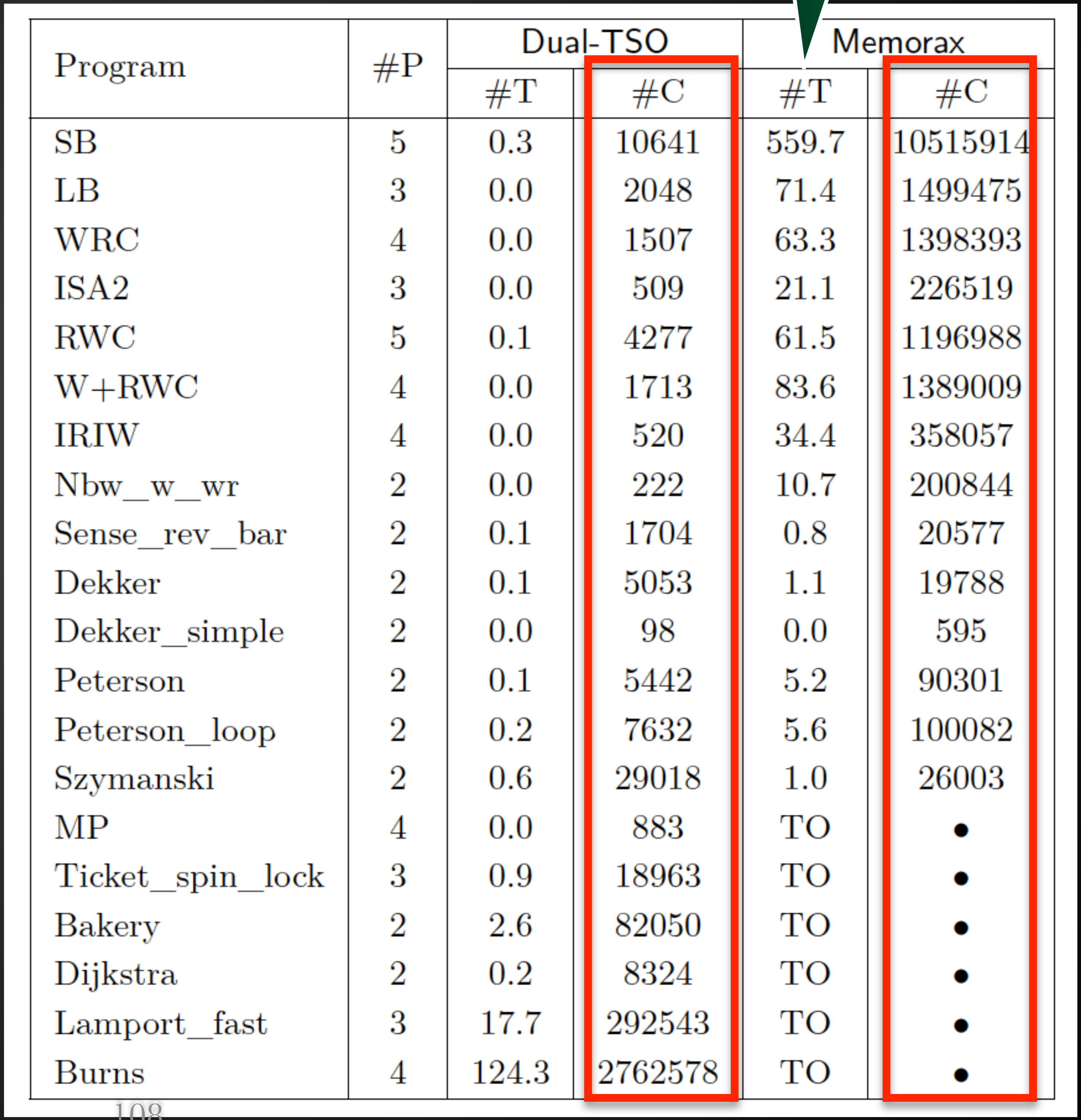
Program	#P	Dual-TS		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	•
Ticket_spin_lock	3	0.9	18963	TO	•
Bakery	2	2.6	82050	TO	•
Dijkstra	2	0.2	8324	TO	•
Lamport_fast	3	17.7	292543	TO	•
Burns	4	124.3	2762578	TO	•

Experimental Res

generated configurations

Dual-TSO vs Memorax

- **Running time**
- **Memory consumption**



Program	#P	Dual-TSO		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	•
Ticket_spin_lock	3	0.9	18963	TO	•
Bakery	2	2.6	82050	TO	•
Dijkstra	2	0.2	8324	TO	•
Lamport_fast	3	17.7	292543	TO	•
Burns	4	124.3	2762578	TO	•

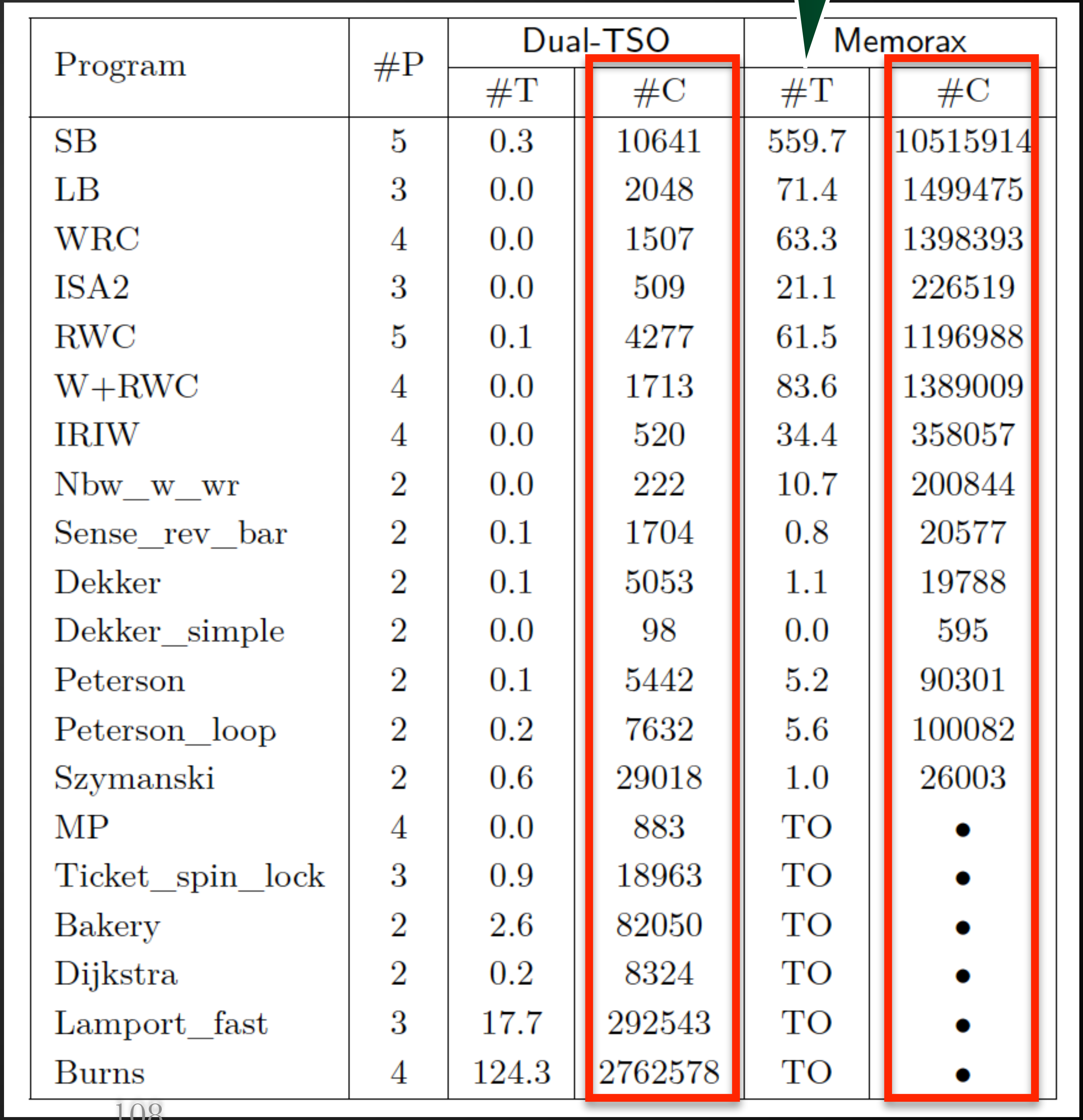
Experimental Res

generated configurations

Dual-TSO vs Memorax

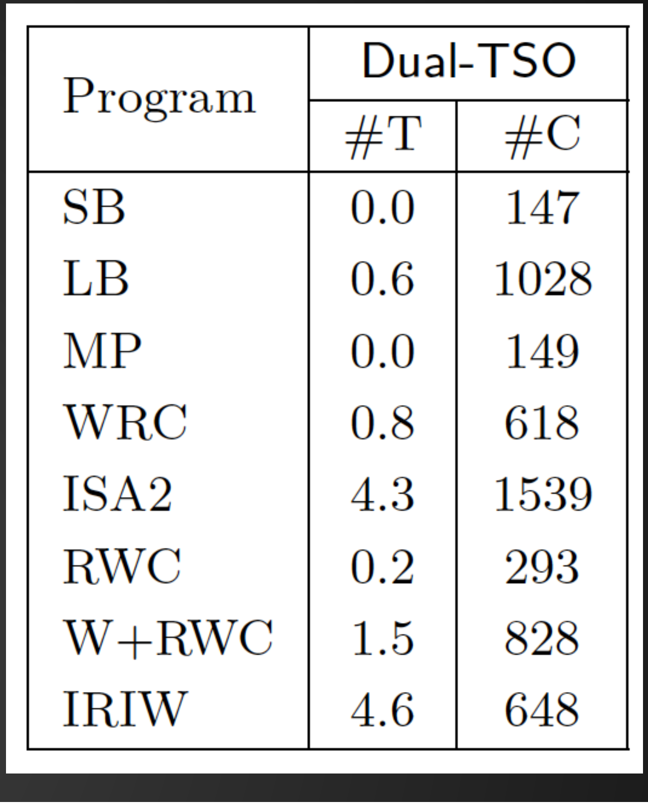
- **Running time**
- **Memory consumption**

Dual-TSO is faster and uses less memory in most of examples

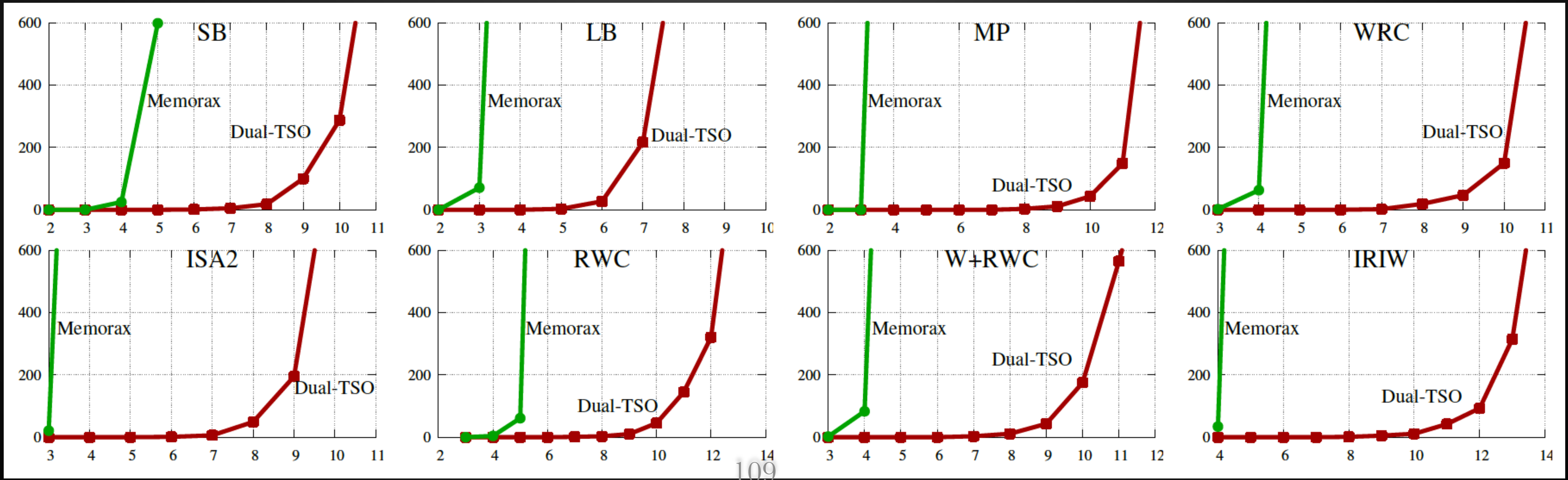


Program	#P	Dual-TSO		Memorax	
		#T	#C	#T	#C
SB	5	0.3	10641	559.7	10515914
LB	3	0.0	2048	71.4	1499475
WRC	4	0.0	1507	63.3	1398393
ISA2	3	0.0	509	21.1	226519
RWC	5	0.1	4277	61.5	1196988
W+RWC	4	0.0	1713	83.6	1389009
IRIW	4	0.0	520	34.4	358057
Nbw_w_wr	2	0.0	222	10.7	200844
Sense_rev_bar	2	0.1	1704	0.8	20577
Dekker	2	0.1	5053	1.1	19788
Dekker_simple	2	0.0	98	0.0	595
Peterson	2	0.1	5442	5.2	90301
Peterson_loop	2	0.2	7632	5.6	100082
Szymanski	2	0.6	29018	1.0	26003
MP	4	0.0	883	TO	•
Ticket_spin_lock	3	0.9	18963	TO	•
Bakery	2	2.6	82050	TO	•
Dijkstra	2	0.2	8324	TO	•
Lamport_fast	3	17.7	292543	TO	•
Burns	4	124.3	2762578	TO	•

Experimental Results
Parameterised Cases

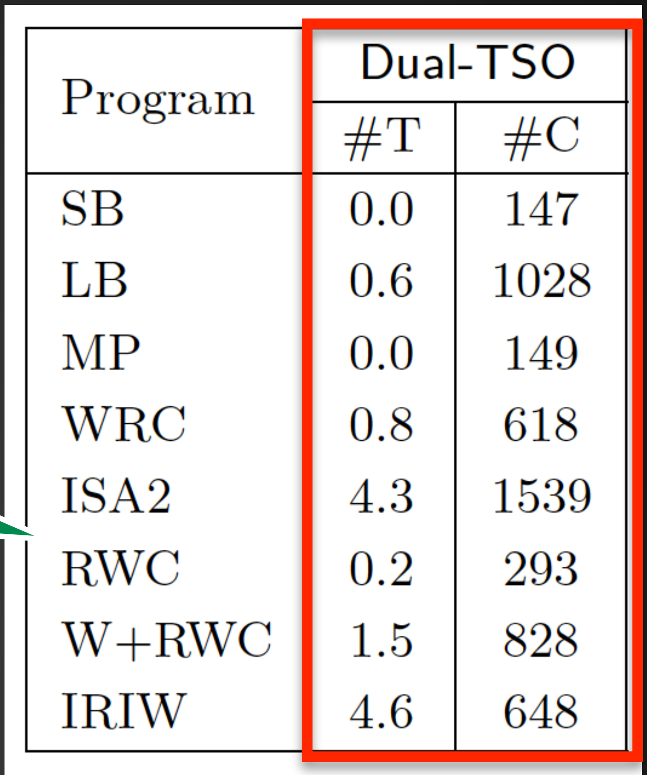


<table>
<tr><th rowspan="2">Program</th><th colspan="2">Dual-TSO</th></tr>
<tr><th>#T</th><th>#C</th></tr>
<tr><td>SB</td><td>0.0</td><td>147</td></tr>
<tr><td>LB</td><td>0.6</td><td>1028</td></tr>
<tr><td>MP</td><td>0.0</td><td>149</td></tr>
<tr><td>WRC</td><td>0.8</td><td>618</td></tr>
<tr><td>ISA2</td><td>4.3</td><td>1539</td></tr>
<tr><td>RWC</td><td>0.2</td><td>293</td></tr>
<tr><td>W+RWC</td><td>1.5</td><td>828</td></tr>
<tr><td>IRIW</td><td>4.6</td><td>648</td></tr>
</table>

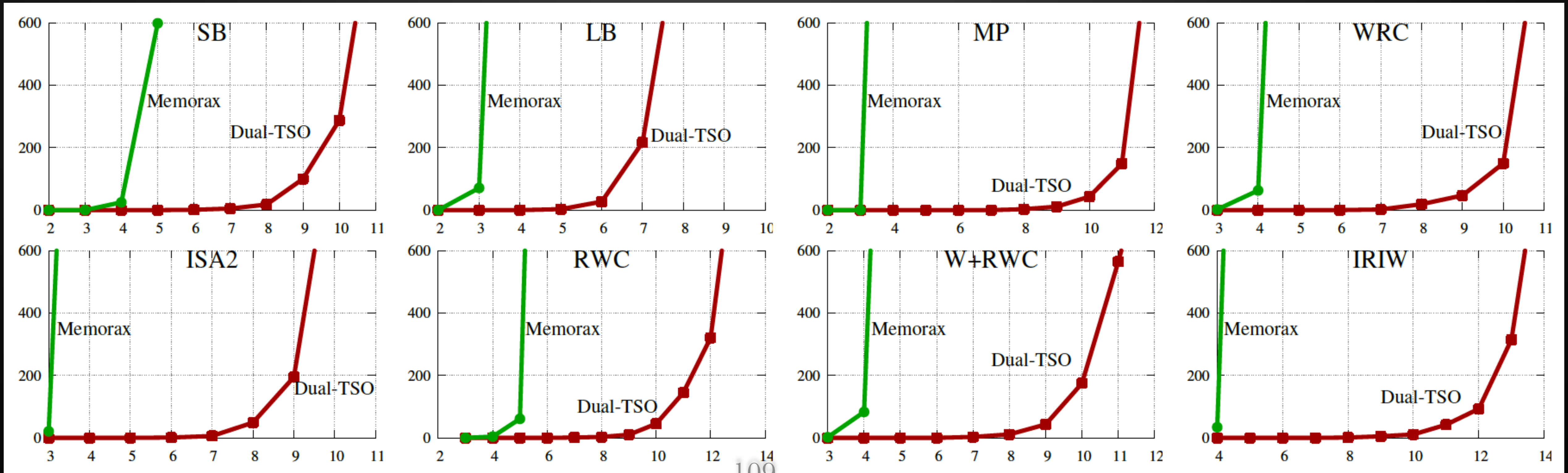


Experimental Results
Parameterised Cases

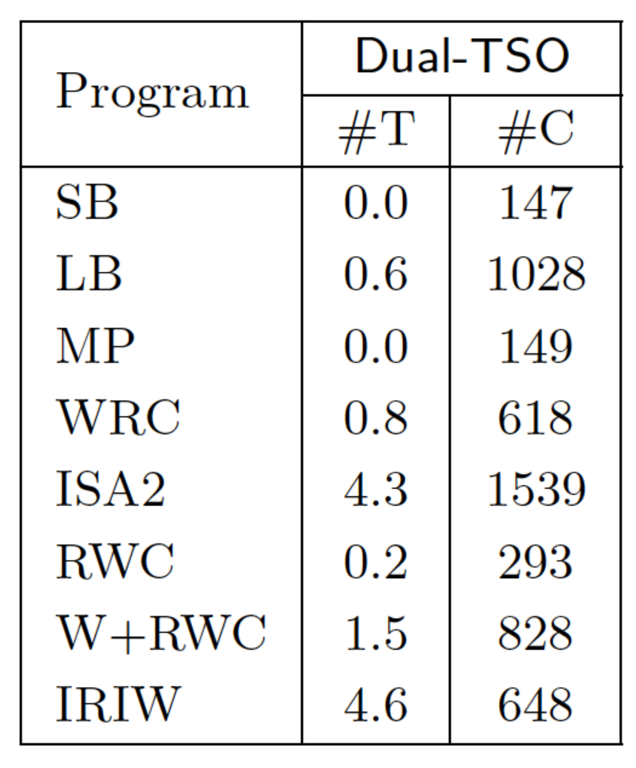
unbounded number of processes



Program	Dual-TSO	
	#T	#C
SB	0.0	147
LB	0.6	1028
MP	0.0	149
WRC	0.8	618
ISA2	4.3	1539
RWC	0.2	293
W+RWC	1.5	828
IRIW	4.6	648

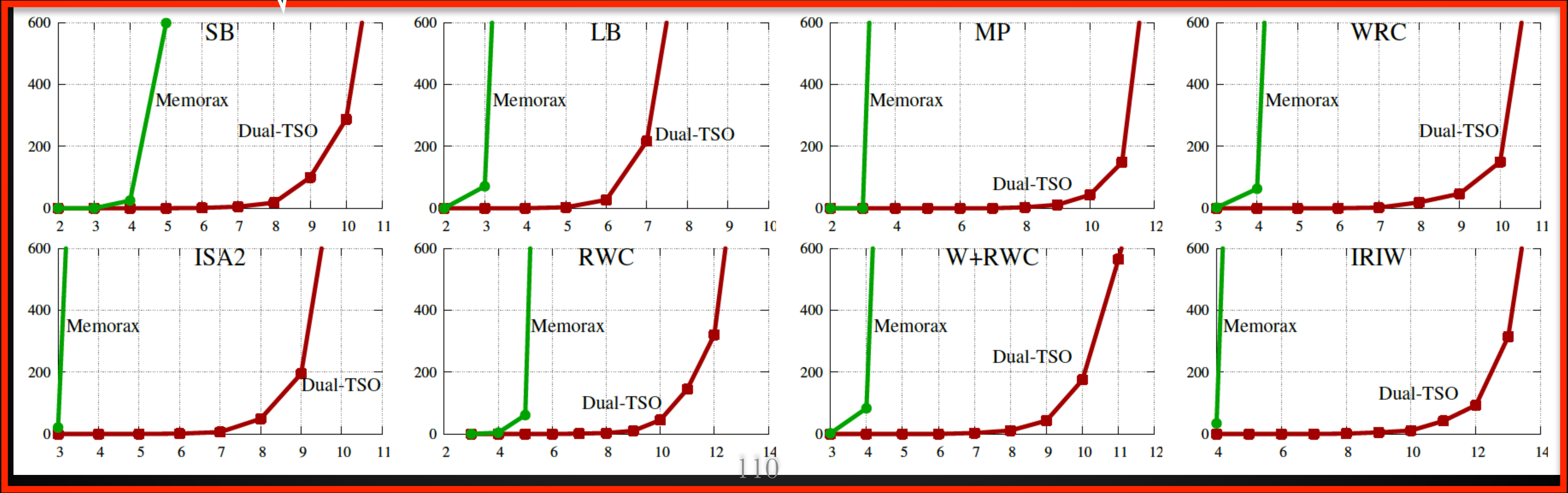


Experimental Results
Parameterised Cases



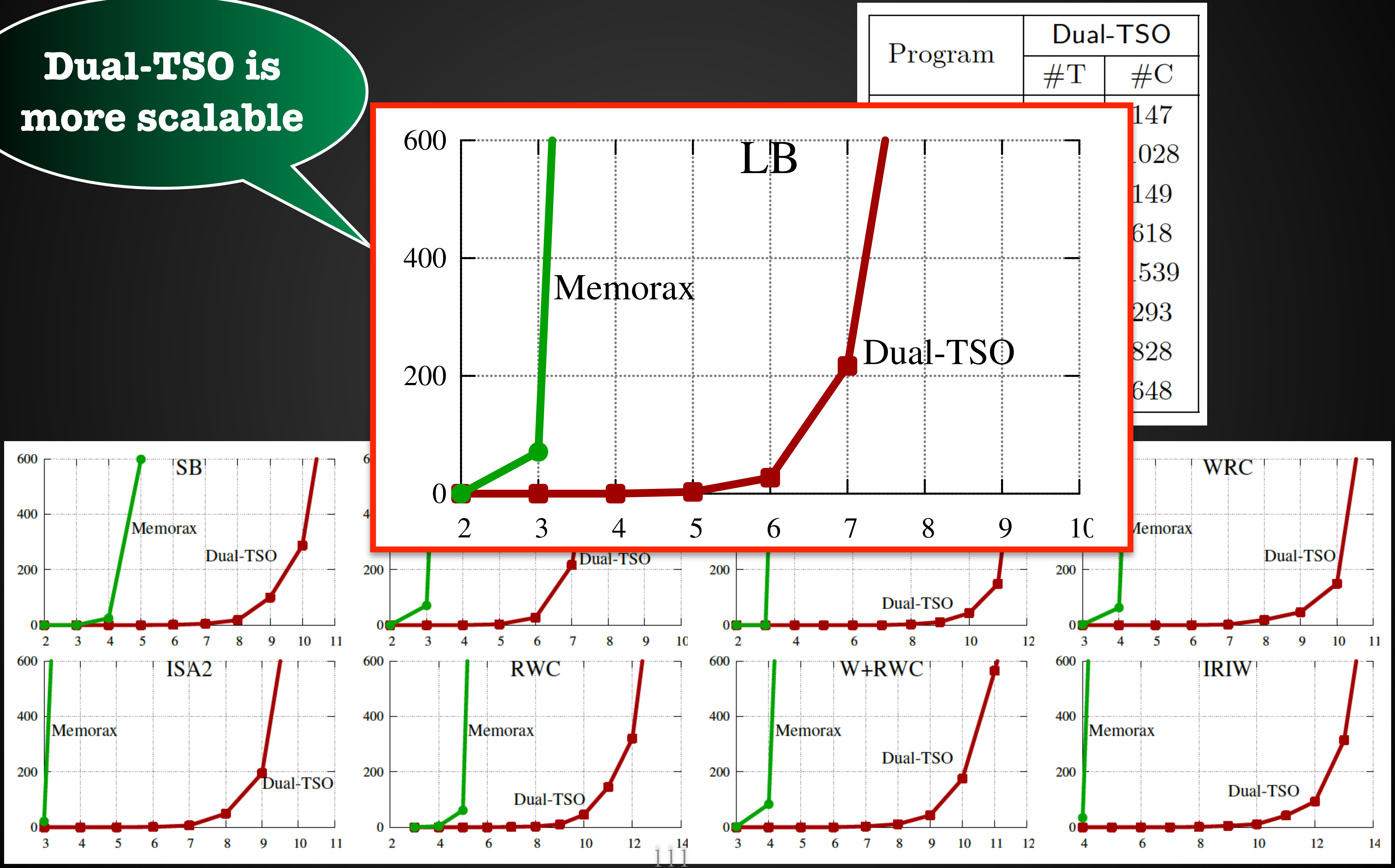
Program	Dual-TSO	
	#T	#C
SB	0.0	147
LB	0.6	1028
MP	0.0	149
WRC	0.8	618
ISA2	4.3	1539
RWC	0.2	293
W+RWC	1.5	828
IRIW	4.6	648

increasing the number of processes



Experimental Results
Parameterised Cases

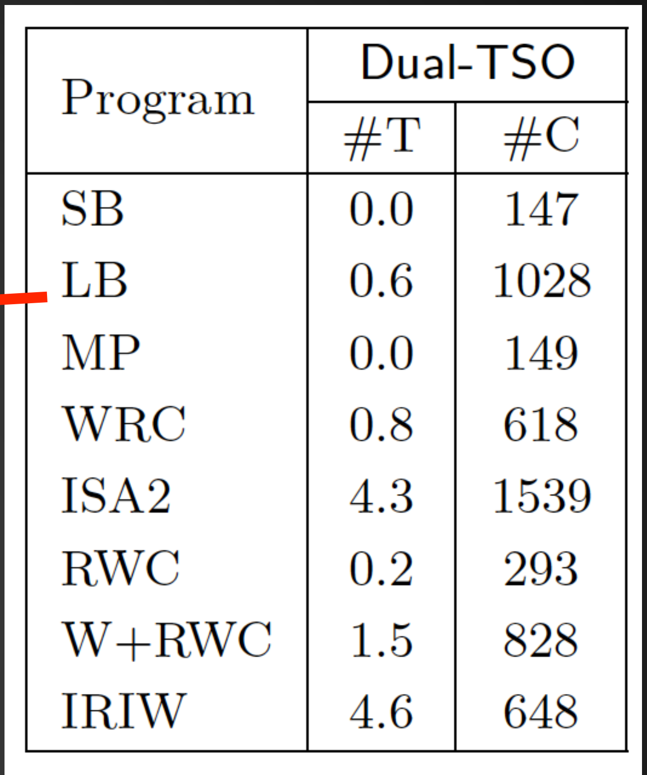
Dual-TSO is more scalable



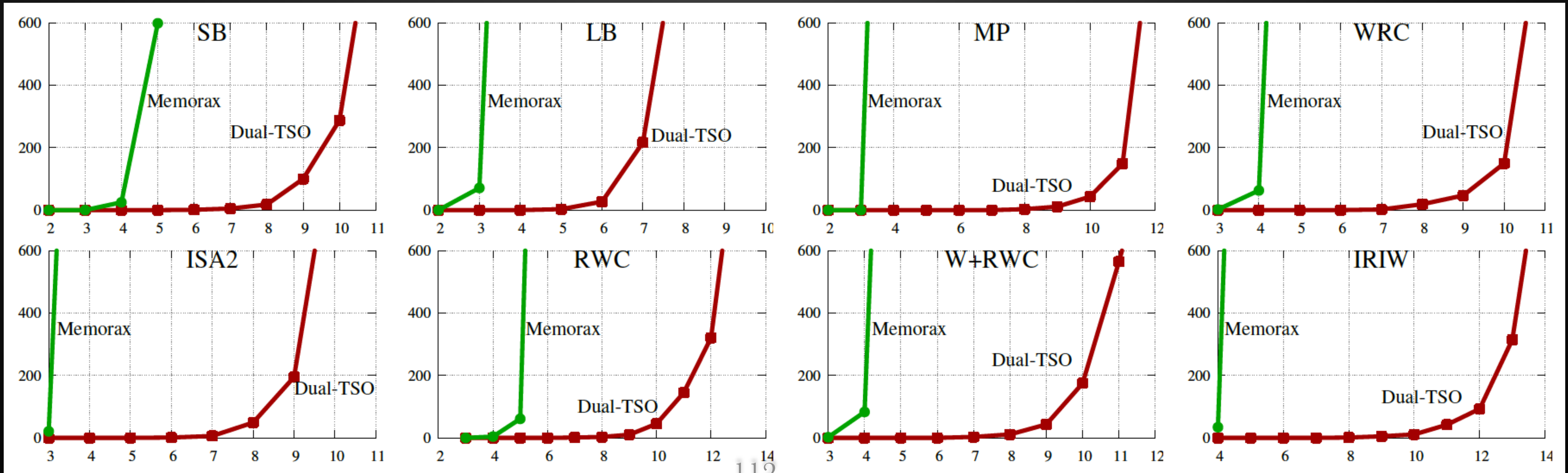
Program	Dual-TSO	
	#T	#C
		147
		028
		149
		618
		539
		293
		828
		648

Experimental Results
Parameterised Cases

Dual-TSO is more efficient and scalable



Program	Dual-TSO	
	#T	#C
SB	0.0	147
LB	0.6	1028
MP	0.0	149
WRC	0.8	618
ISA2	4.3	1539
RWC	0.2	293
W+RWC	1.5	828
IRIW	4.6	648


SB
LB
MP
WRC
ISA2
RWC
W+RWC
IRIW
Memorax
Dual-TSO