Blunting an Adversary Against Randomized Concurrent Programs with Linearizable Implementations

Hagit Attiya (Technion)
Constantin Enea (Ecole Polytechnique)
Jennifer Welch (Texas A&M University)

Using an Abstract Multi-Writer Register

Implemented from Single-Writer Registers

Write(v,X)

read TS_0,..., read TS_{n-1}

TS_i = max TS_j +1

write $\langle v, TS_i, i\rangle$ to R_i

Read(X)

read R_0,..., read R_{n-1}

return v_j with

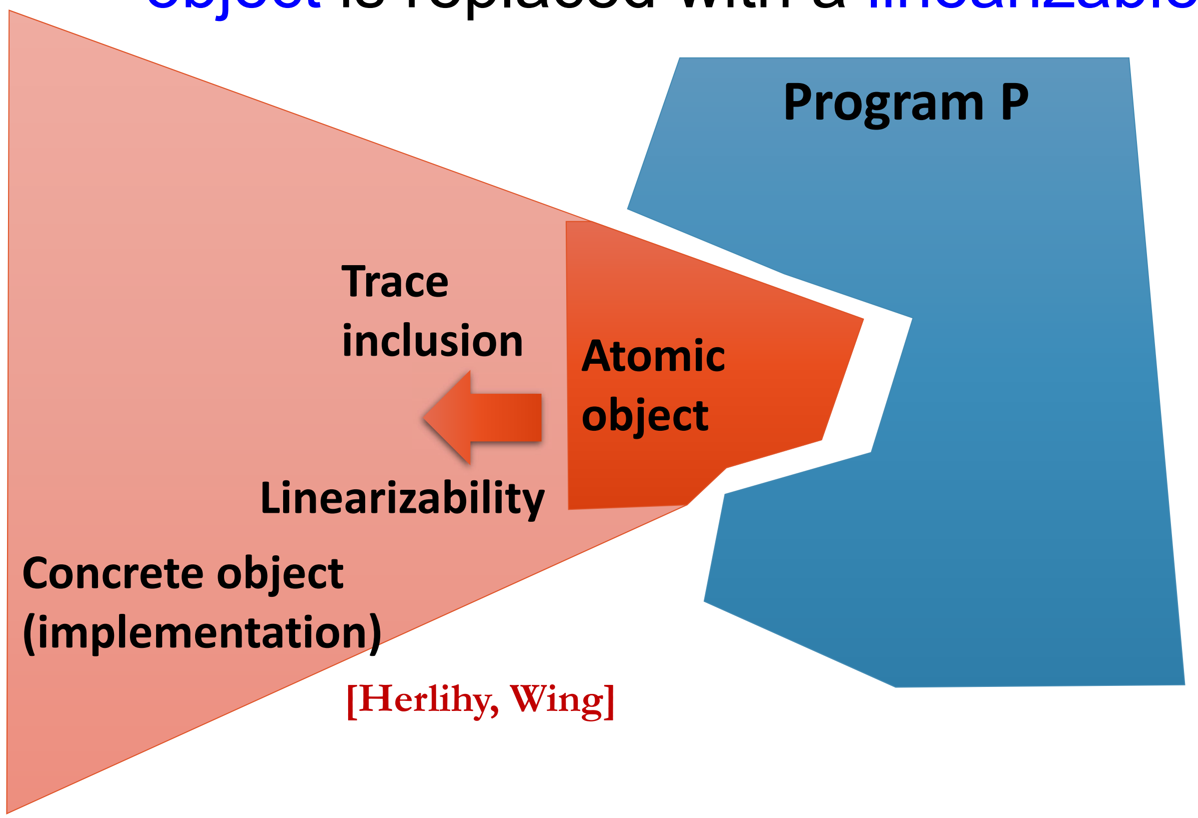
maximal $<TS_j, j>$

Program P

[Vitanyi, Awerbuch]

Linearizability Preserves Trace Properties

If φ is a property of a trace, it is preserved when atomic object is replaced with a linearizable implementation

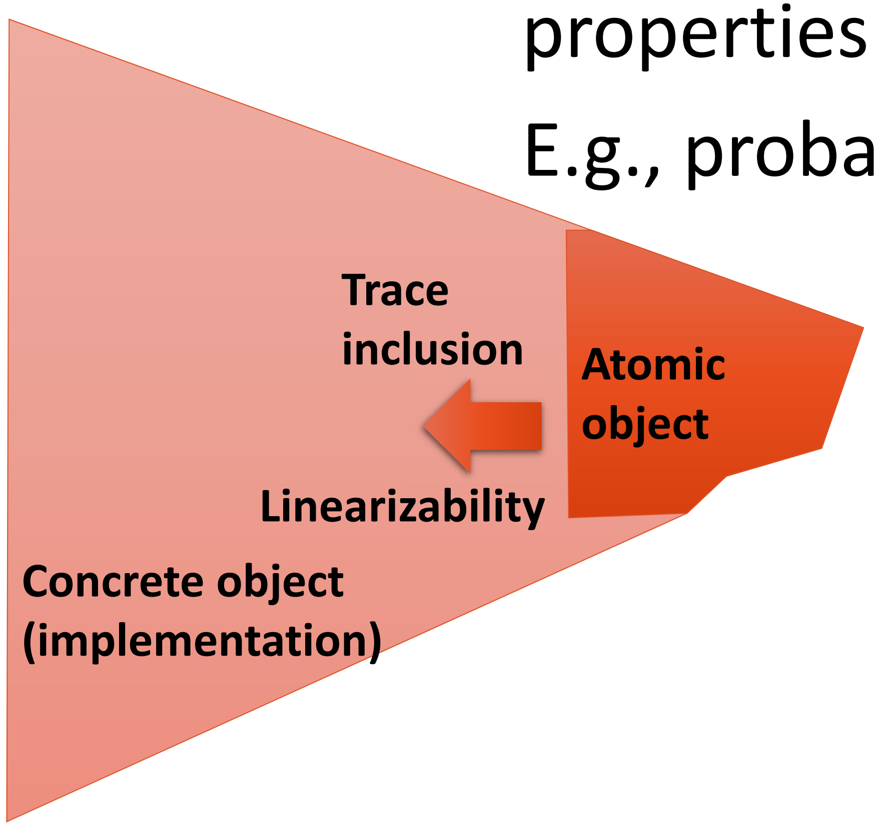


$\models \varphi$

But not Hyper-properties

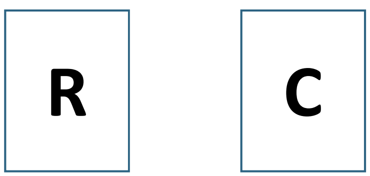
Linearizability does not preserve properties of sets of traces

E.g., probability distributions



[Golab, Higham, Woelfel, STOC 2011]

Example w/ MWSR Register



Initially $R = \perp$, $C = -1$

<u>**Code for** p_0**,** p_1</u>**:**

$R \leftarrow i$

if $i = 0$ **then** $C \leftarrow$ **flip 0 or 1**

<u>**Code for** p_2</u>**:**

$r \leftarrow R$; $c \leftarrow C$

if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

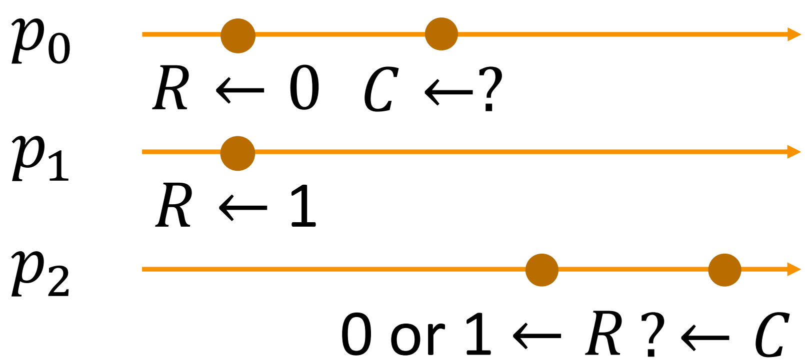
then loop forever

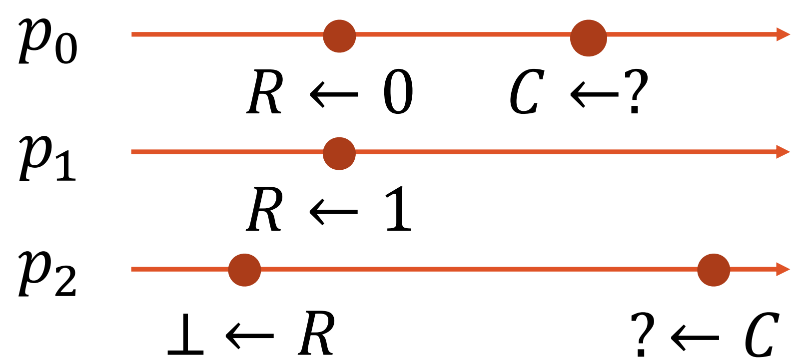
else terminate

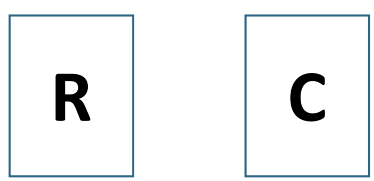
p_2 terminates with probability > ½ w/ atomic mwsr register

Example by Noa Schiller, based on [A, Enea, Welch] and [Hadzilacos, Hu, Toueg, PODC 2021]

Example w/ MWSR Register







Initially $R = \perp$, $C = -1$

<u>Code for p_0, p_1</u>:

$R \leftarrow i$

if $i = 0$ **then** $C \leftarrow$ **flip 0 or 1**

<u>Code for p_2</u>:

$r \leftarrow R$; $c \leftarrow C$

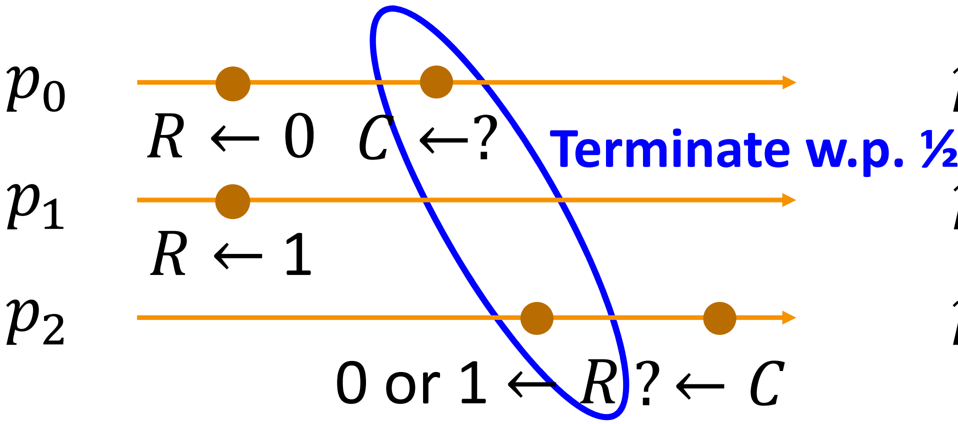
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

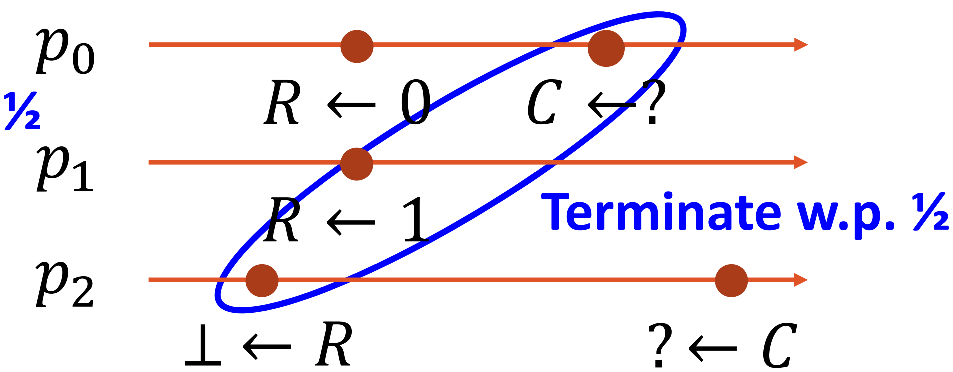
then loop forever

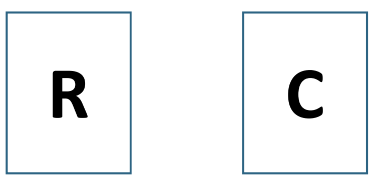
else terminate

p_2 terminates with probability > ½ w/ atomic mwsr register

Example w/ MWSR Register







Initially $R = \perp$, $C = -1$

<u>**Code for** p_0, p_1</u>**:**

$R \leftarrow i$

if $i = 0$ **then** $C \leftarrow$ **flip 0 or 1**

<u>**Code for** p_2</u>**:**

$r \leftarrow R$; $c \leftarrow C$

if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

then loop forever

else terminate

p_2 terminates with probability > ½ w/ atomic mwsr register

Using VA Implementation

```
Write(v,R)
read TS0,...,read TSn-1
TSi = max TSj +1
write ⟨v,TSi,i⟩ to Ri

Read(R)
read R0,...,read Rn-1
return vj with maximal <TSj,j>
```

Initially $R = \perp$, $C = -1$

Code for p_0, p_1:

$R \leftarrow i$

if $i = 0$ then $C \leftarrow$ flip 0 or 1

Code for p_2:

$r \leftarrow R$; $c \leftarrow C$

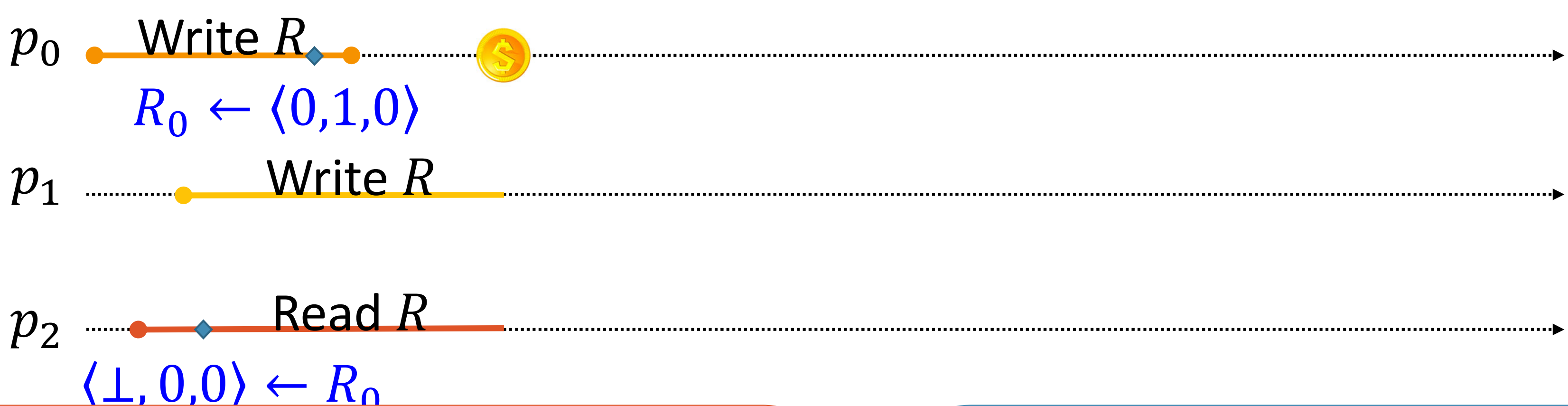
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

then loop forever

else terminate

A strong adversary can make p_2 always loop forever

Using VA Implementation



Write(v,R)

read TS_0,...,read TS_{n-1}

TS_i = max TS_j +1

write $\langle$v,TS_i,i$\rangle$ to R_i

Read(R)

read R_0,...,read R_{n-1}

return v_j with maximal <TS_j,j>

Initially $R = \perp$, $C = -1$

Code for p_0, p_1:

$R \leftarrow i$

if $i = 0$ then $C \leftarrow$ flip 0 or 1

Code for p_2:

$r \leftarrow R$; $c \leftarrow C$

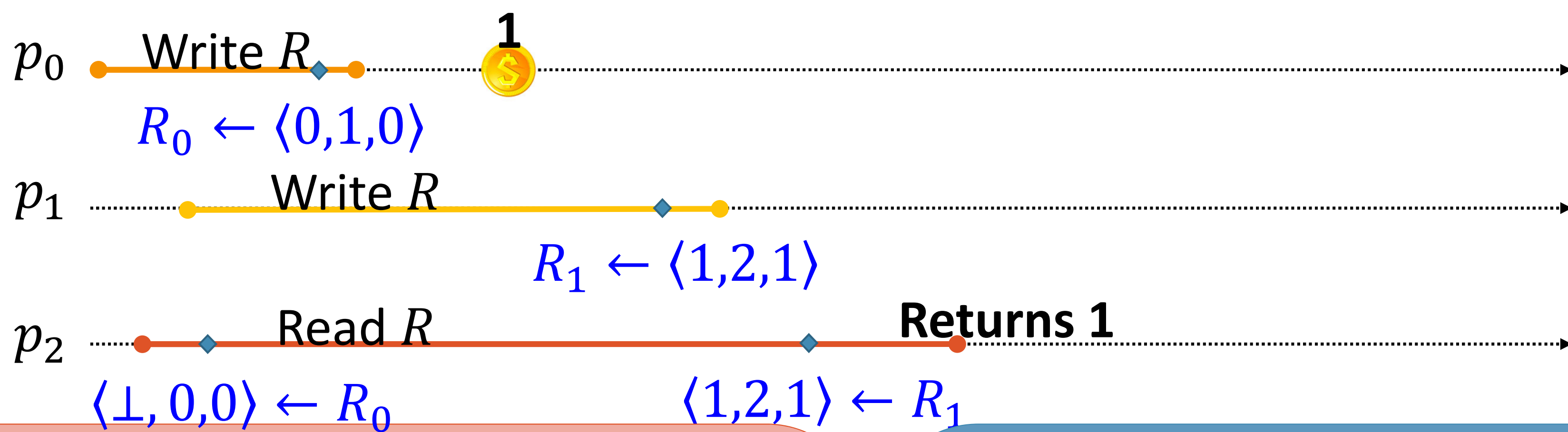
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

then loop forever

else terminate

A strong adversary can make p_2 always loop forever

Example w/ VA: Coin = 1




```
Write(v,R)
read TS_0,...,read TS_n-1
TS_i = max TS_j +1
write ⟨v,TS_i,i⟩ to R_i

Read(R)
read R_0,...,read R_n-1
return v_j with maximal <TS_j,j>
```

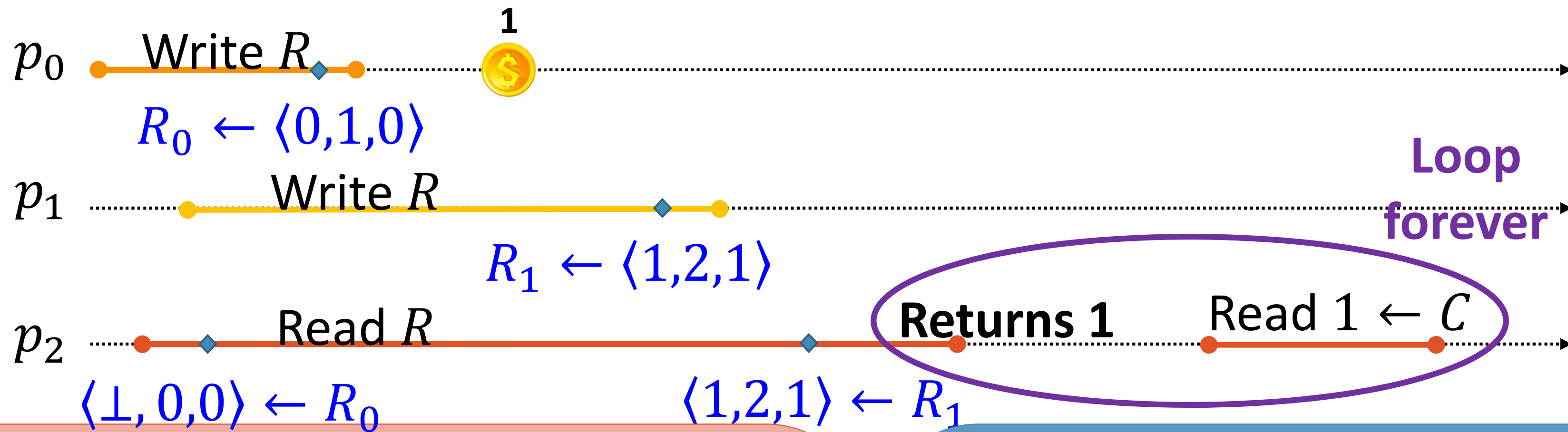
Initially $R = \perp$, $C = -1$

Code for p_0, p_1:
$R \leftarrow i$
if $i = 0$ then $C \leftarrow$ flip 0 or 1

Code for p_2:
$r \leftarrow R$; $c \leftarrow C$
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$
then loop forever
else terminate

A strong adversary can make p_2 always loop forever

Example w/ VA: Coin = 1

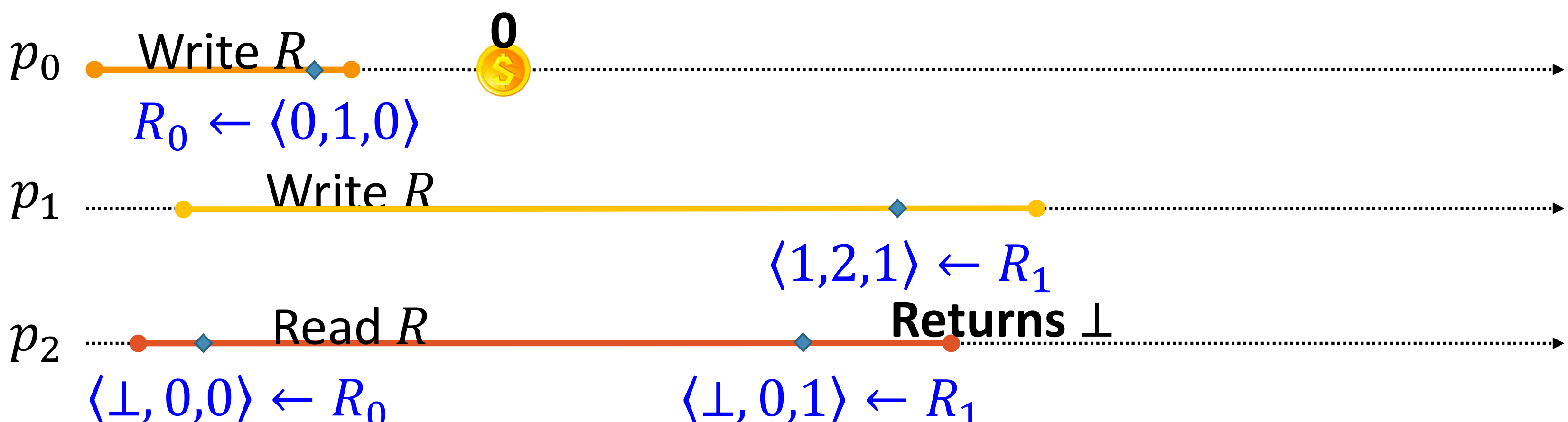



```
Write(v,R)
read TS_0,...,read TS_{n-1}
TS_i = max TS_j +1
write ⟨v,TS_i,i⟩ to R_i

Read(R)
read R_0,...,read R_{n-1}
return v_j with maximal <TS_j,j>
```

```
Initially R =⊥, C = −1
Code for p_0, p_1:
R ← i
if i = 0 then C ← flip 0 or 1

Code for p_2:
r ← R; c ← C
if (c = 0 ∧ r =⊥) ∨ (c = 1 ∧ r ≠⊥)
     then loop forever
else terminate
```

A strong adversary can make p_2 always loop forever

Example w/ VA: Coin = 0



Write(v,R)

read TS_0,...,read TS_{n-1}

TS_i = max TS_j +1

write $\langle v, TS_i, i \rangle$ to R_i

Read(R)

read R_0,...,read R_{n-1}

return v_j with maximal <TS_j,j>

Initially $R = \perp$, $C = -1$

Code for p_0, p_1:

$R \leftarrow i$

if $i = 0$ then $C \leftarrow$ flip 0 or 1

Code for p_2:

$r \leftarrow R$; $c \leftarrow C$

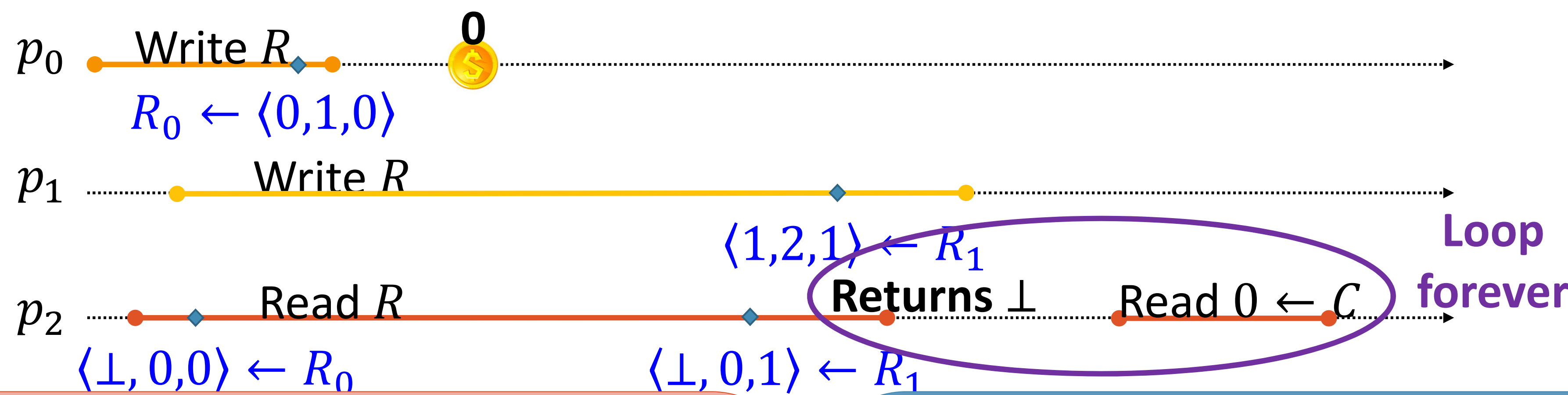
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

then loop forever

else terminate

A strong adversary can make p_2 always loop forever

Example w/ VA: Coin = 0



<u>Write(v,R)</u>
read TS_0,...,read TS_{n-1}
TS_i = max TS_j +1
write $\langle v, TS_i, i \rangle$ to R_i

<u>Read(R)</u>
read R_0,...,read R_{n-1}
return v_j with maximal $\langle TS_j, j \rangle$

Initially $R = \perp$, $C = -1$
<u>Code for p_0, p_1</u>:
$R \leftarrow i$
if $i = 0$ then $C \leftarrow$ flip 0 or 1

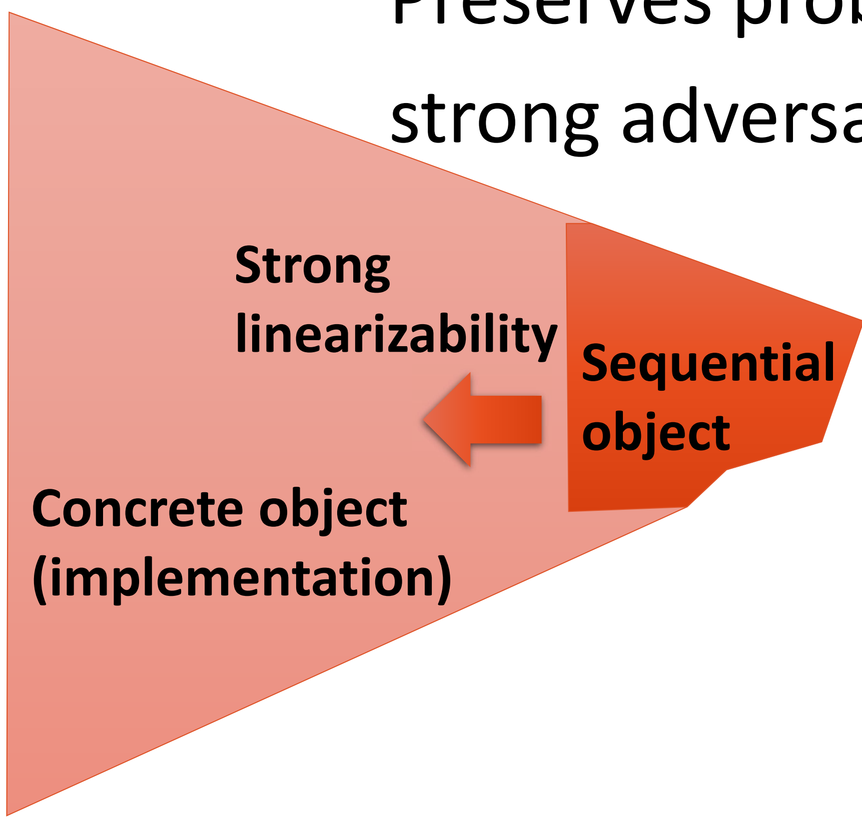
<u>Code for p_2</u>:
$r \leftarrow R$; $c \leftarrow C$
if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$
then loop forever
else terminate

A strong adversary can make p_2 always loop forever

Strong Linearizability

Linearization points are prefix preserving

Preserves probability distributions under strong adversaries



[Golab, Higham, Woelfel, STOC 2011]

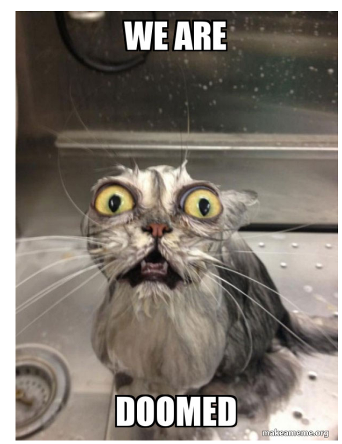
Many Objects Don't Have Strongly Linearizable Implementations

Counter-example ⇨ V&A MW register is not strongly linearizable

In fact, there is no wait-free strongly-linearizable MW register implementation from SW registers

Also, no wait-free strongly-linearizable snapshot implementation

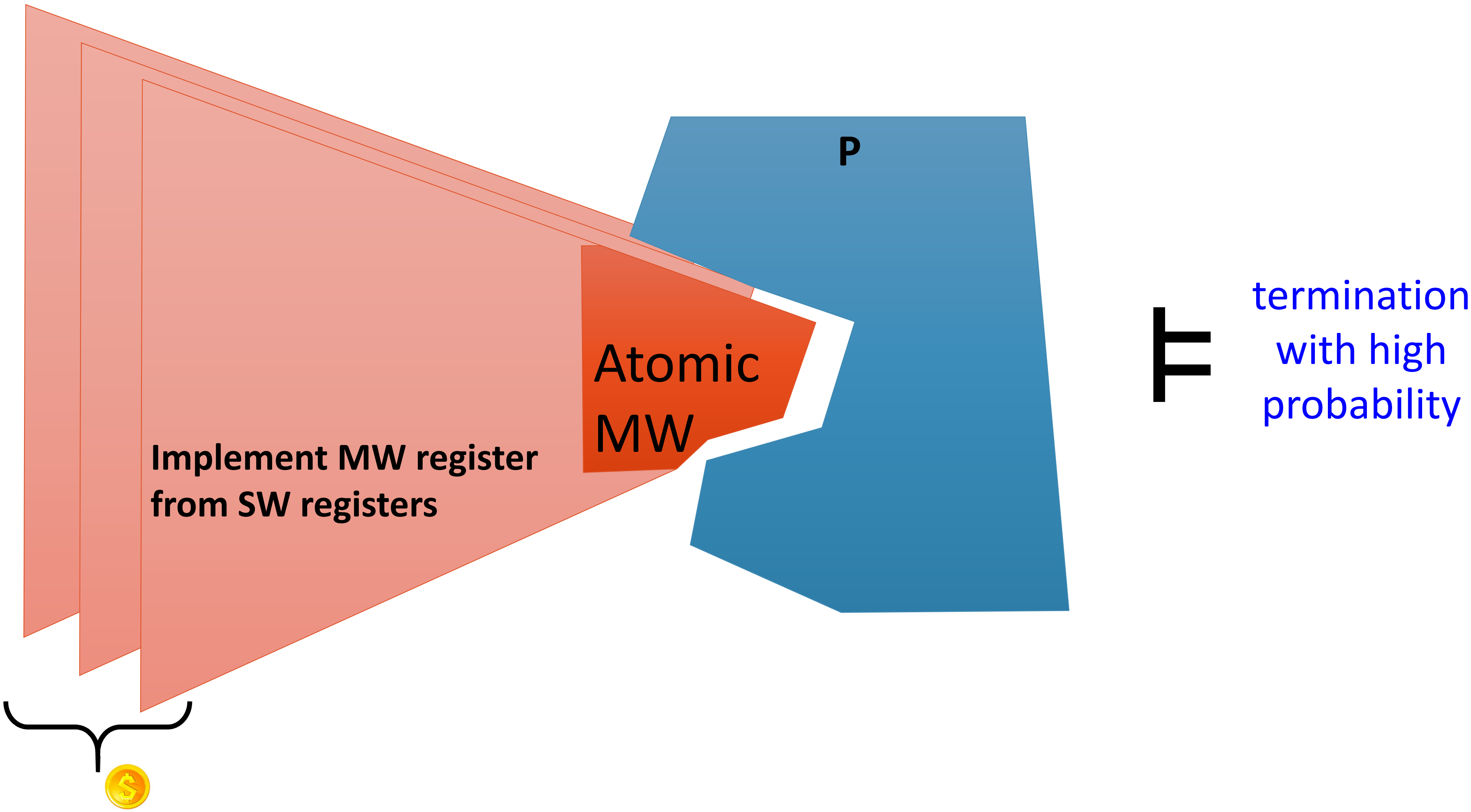
[Helmi, Higham, Woelfel, PODC 2012]



No message-passing simulation of a register

[A, Enea, Welch, DISC 2021] [Chan, Hadzilacos, Hu, Toueg]

Use Randomization to Blunt the Adversary



E.g., Blunting w/ One Coin Flip

p_2 terminates with constant probability with VA^2

```
Write(v,X)
read TS_0,...,read TS_{n-1}
TS_i = max TS_j +1
write ⟨v,TS_i,i⟩ to R_i
Read(X)
read R_0,...,read R_{n-1}
v1 = v_j with maximal <TS_j,j>
read R_0,...,read R_{n-1}
v2 = v_j with maximal <TS_j,j>
return v1 or v2 with prob. ½
```

Initially $R = \perp$, $C = -1$

Code for p_0, p_1:

$R \leftarrow i$

if $i = 0$ then $C \leftarrow$ flip 0 or 1

Code for p_2:

$r \leftarrow R$; $c \leftarrow C$

if $(c = 0 \wedge r = \perp) \vee (c = 1 \wedge r \neq \perp)$

then loop forever

else terminate

Tail Strong Linearizability

Identify a preamble of the operation, after which, it is mapped in a prefix-preserving manner ("strongly linearizable")

```
Write(v,X)
read TS_0,...,read TS_{n-1}
TS_i = max TS_j +1
write ⟨v,TS_i,i⟩ to R_i
Read(X)
read R_0,...,read R_{n-1}
v = v_j with maximal <TS_j,j>


return v
```

Effect-free preamble doesn't impact concurrently-running processes

Blunting

Tail strongly linearizable objects with an effect-free preamble

- Repeat the preamble k times
- Randomly pick one of the iterations to continue with

```
Write(v,X)
read TS_0,...,read TS_n-1
TS_i = max TS_j +1
write ⟨v,TS_i,i⟩ to R_i
Read(X)
read R_0,...,read R_n-1
v1 = v_j with maximal <TS_j,j>
read R_0,...,read R_n-1
v2 = v_j with maximal <TS_j,j>
return v1 or v2 with prob. ½
```

Blunting, Specifically

Tail strongly linearizable objects with a read-only preamble, e.g.,

- Multi-reader registers from single-reader registers [Israeli, Li 1993]
- Multi-writer registers from single-writer registers [Vitanyi, Awerbuch 1986]
- ABD, Snapshots [Afek et al.]

For an n-process program P with r coin flips, using tail-strongly-linearizable objects O with effect-free preambles & any $k \geq r$,

$$\Pr[\,O^k] \leq \Pr[O_a] + (\Pr[O] - \Pr[\,O_a])\left(1 - \left(\frac{k-r}{k}\right)^{n-1}\right)$$

probability of a **bad** outcome B when P uses k-preamble-iterated versions of objects in O

probability of B when P uses atomic versions of objects in O

probability of B when P uses objects in O

E.g., in our Example

p_2 terminates with probability $> {}^1/_8$ with VA^2

and $> {}^2/_9$ with VA^3

$$1-\left(\frac{3-1}{3}\right)^2 = {}^5/_9$$

Non-termination probability w/ VA^2

$${}^1/_2$$

$$1-{}^1/_2$$

$$1-\left(\frac{2-1}{2}\right)^2 = {}^3/_4$$

$$\Pr[O^k] \leq \Pr[O_a] + (\Pr[O] - \Pr[O_a])\left(1-\left(\frac{k-r}{k}\right)^{n-1}\right)$$

probability of a **bad** outcome B when P uses k-preamble-iterated versions of objects in O

probability of B when P uses atomic versions of objects in O

probability of B when P uses objects in O

Let X be the event that all random choices in O^k objects return an iteration of the preamble that does NOT overlap any random step of the program, then

$$\Pr[O^k] \;=\; \Pr[O^k|X]\cdot\Pr[X] \;+\; \Pr[O^k|\neg X]\cdot(1-\Pr[X])$$

$$\leq \Pr[O_a]\cdot\Pr[X]$$

when chosen preambles don't overlap any program random steps, O^k objects behave like atomic objects

when chosen preamble overlaps program random step, O^k objects are no worse than O objects

$$+\Pr[O]\cdot(1-\Pr[X])$$

Since $\Pr[X] \;\geq \left(\frac{k-r}{k}\right)^{n-1}$, rearrangement gives that

$$\Pr[\,O^k] \leq \Pr[O_a] + (\Pr[O] - \Pr[\,O_a])\left(1-\left(\frac{k-r}{k}\right)^{n-1}\right)$$

probability of a **bad** outcome B when P uses k-preamble-iterated versions of objects in O

probability of B when P uses atomic versions of objects in O

probability of B when P uses objects in O

Wrap-Up

Write strong linearizability [Hadzilacos, Hu, Toueg, PODC 2021]
does not help with our example

Tradeoff between # iterations and decreased prob. of bad outcome
Reduce # random steps considered in the analysis, based on program structure (e.g., communication-closed layers)

Object implementations
w/o effect-free preambles

Transactions &
Cryptographic protocols