

Higher-order Interference in Deny-Guarantee

Mike Dodds, Xinyu Feng,
Matthew Parkinson and Viktor Vafeiadis

Deny-guarantee

Approach:

- Extension of rely-guarantee reasoning
- Treat interference as a resource
- Split using separation logic

Result:

- Handles fork / join naturally

This talk

I'm *not* going to talk about fork / join.
(see *Deny-guarantee Reasoning*, ESOP '09)

Instead: heap algorithms with higher-order
interference

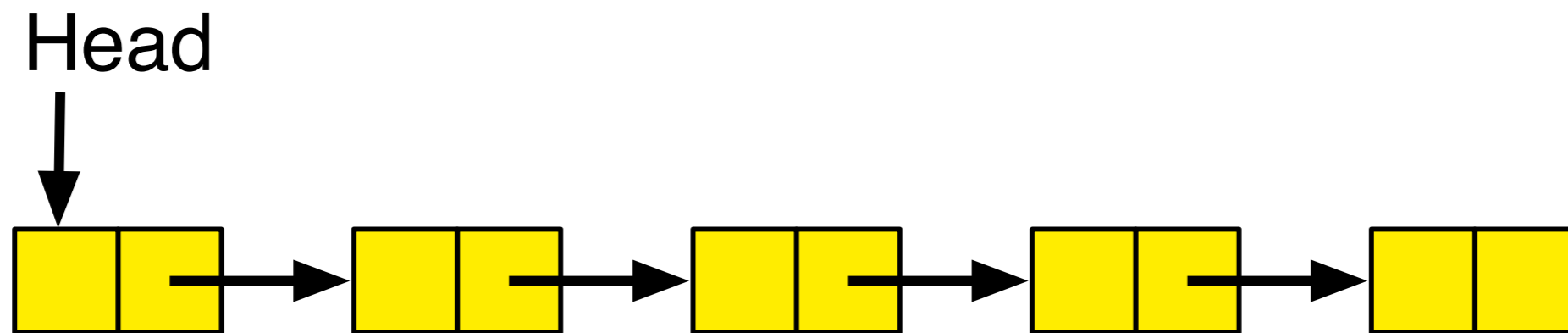
Sales pitch

Higher-order interference: ‘actions about actions’

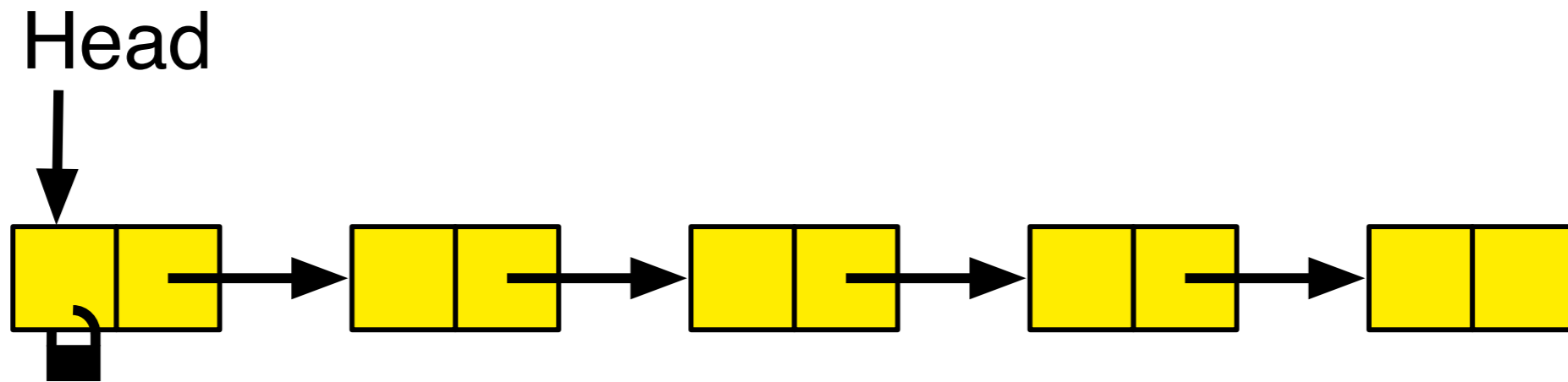
Higher-order interference allows us to

- Avoid auxiliary variables
- Generalise proofs
- Simplify proofs

Fine-grained locking

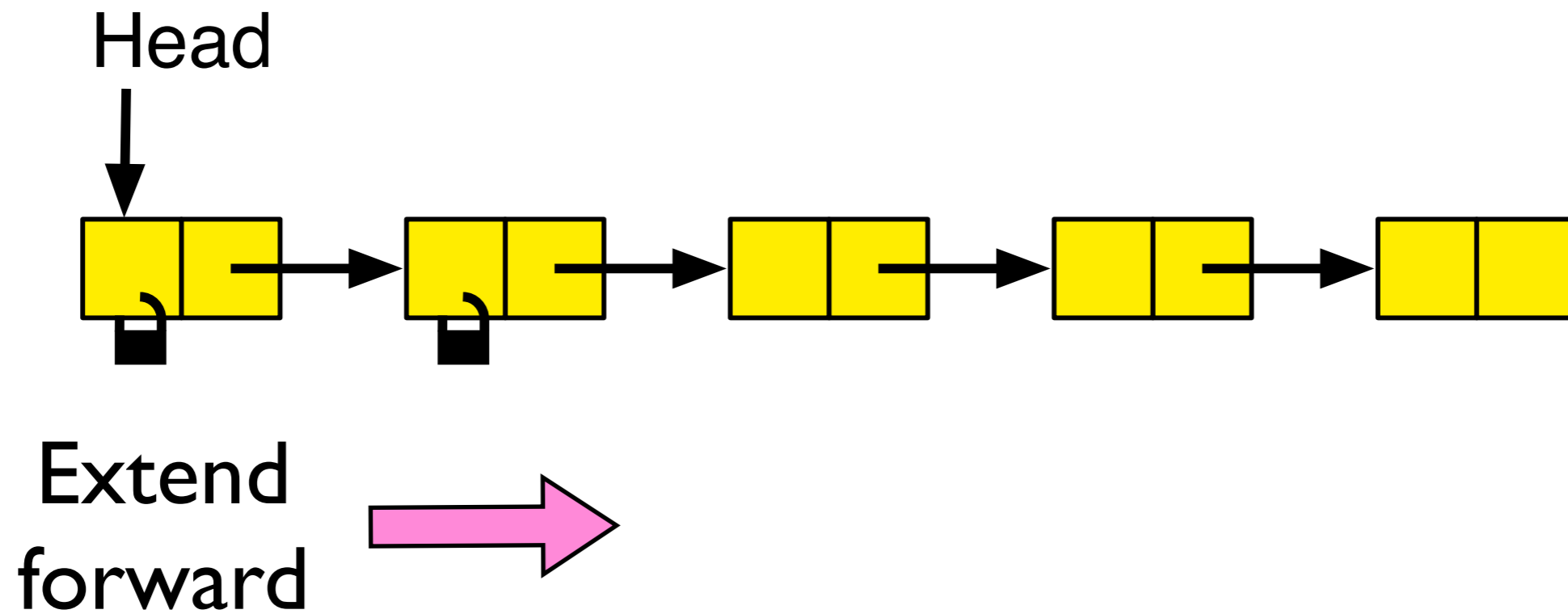


Fine-grained locking

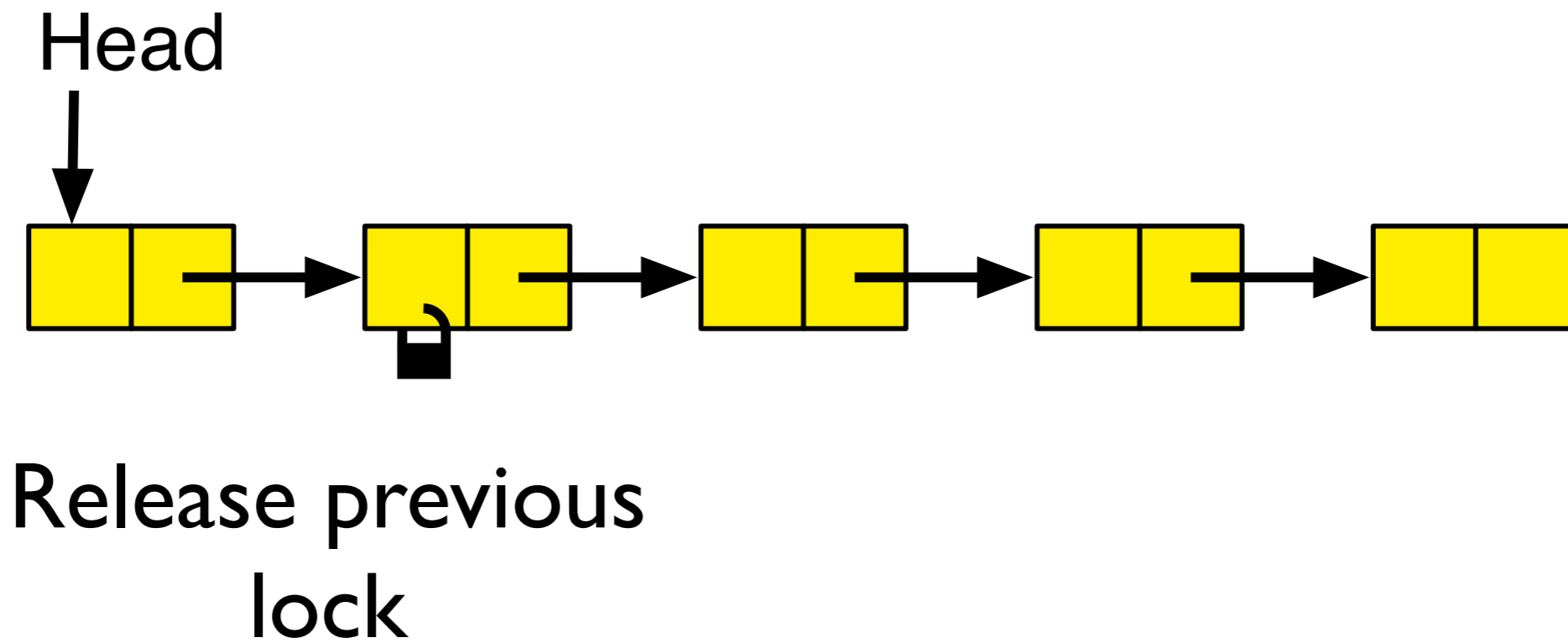


Lock the head

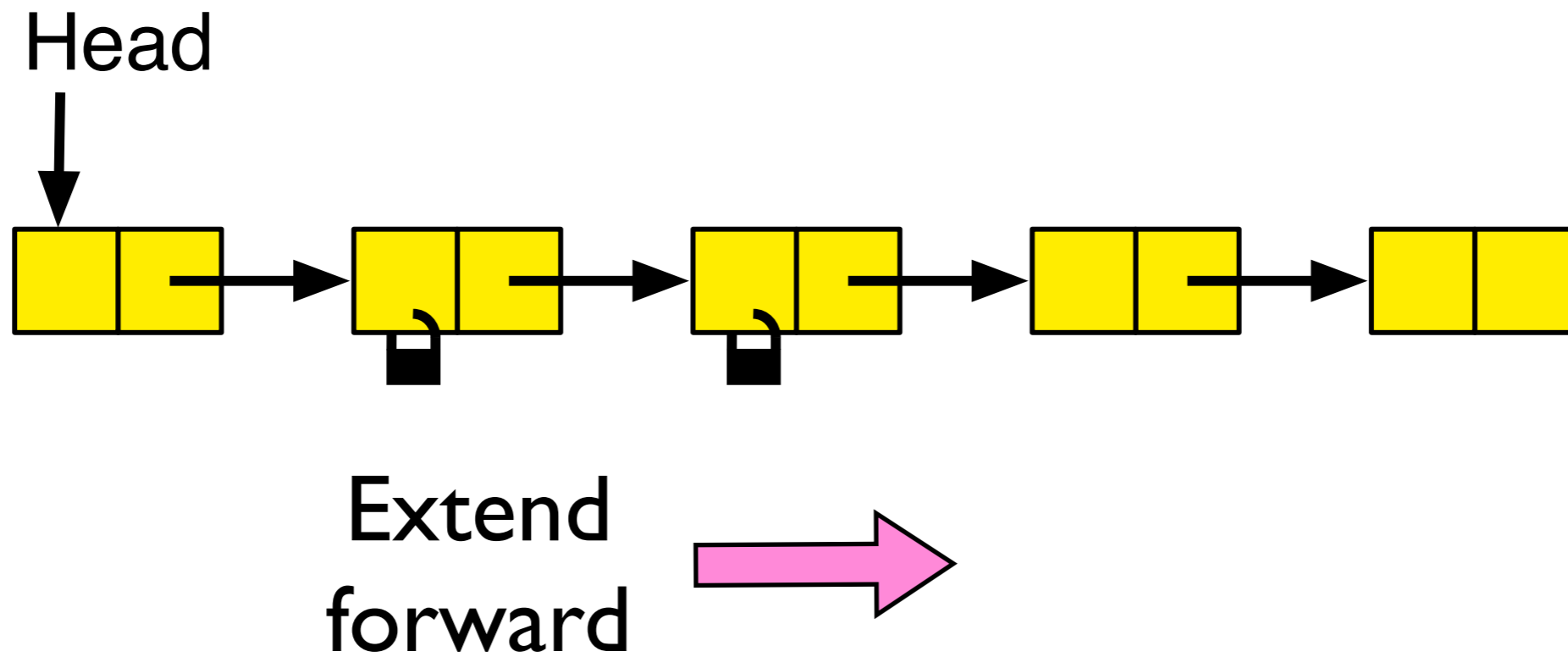
Fine-grained locking



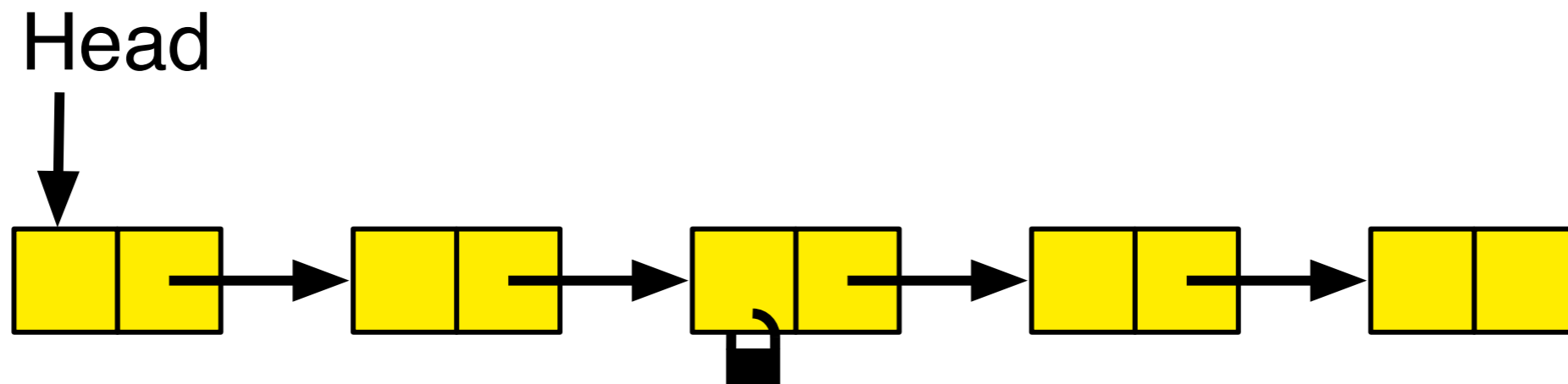
Fine-grained locking



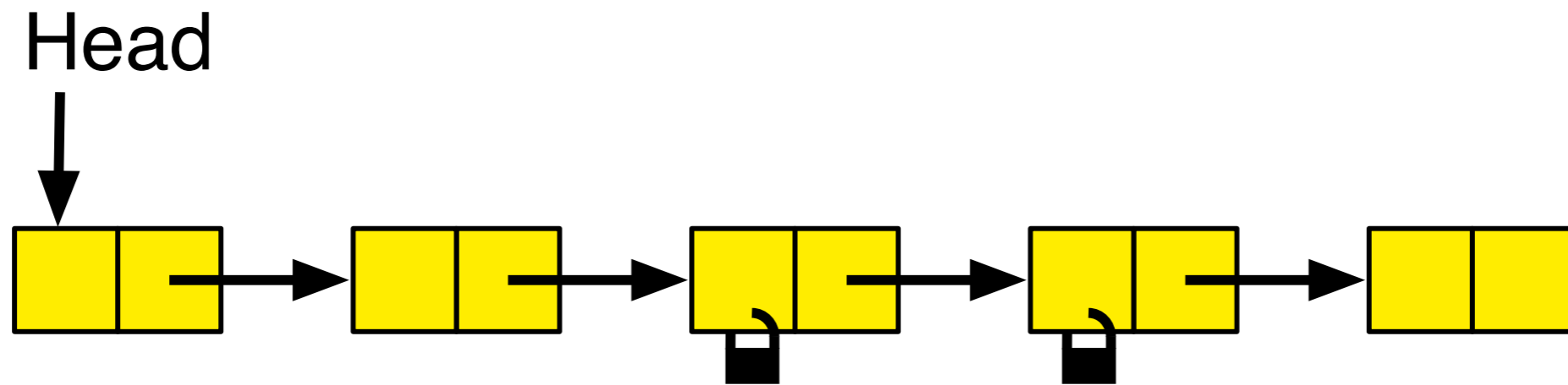
Fine-grained locking



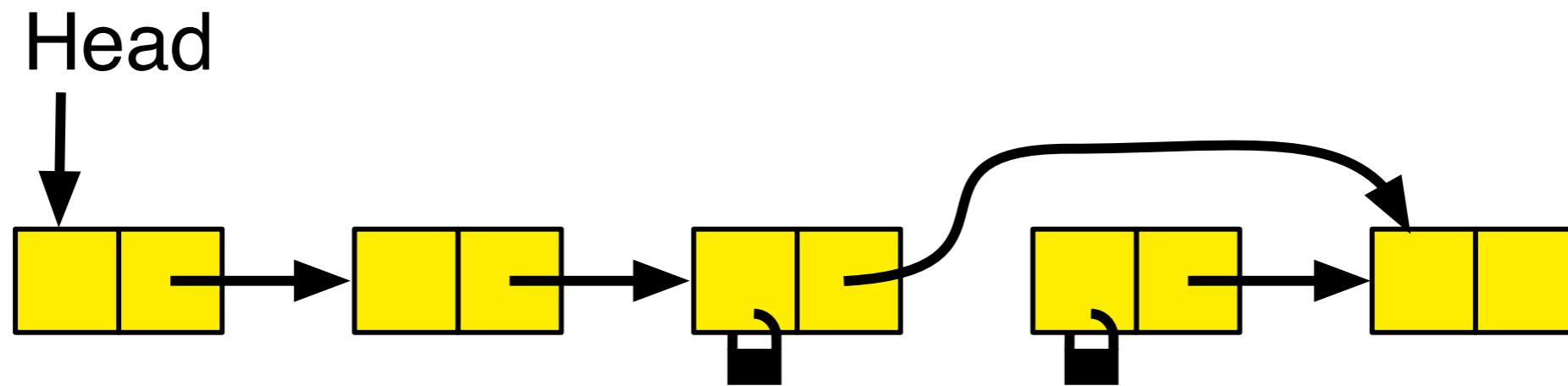
Fine-grained locking



Fine-grained locking

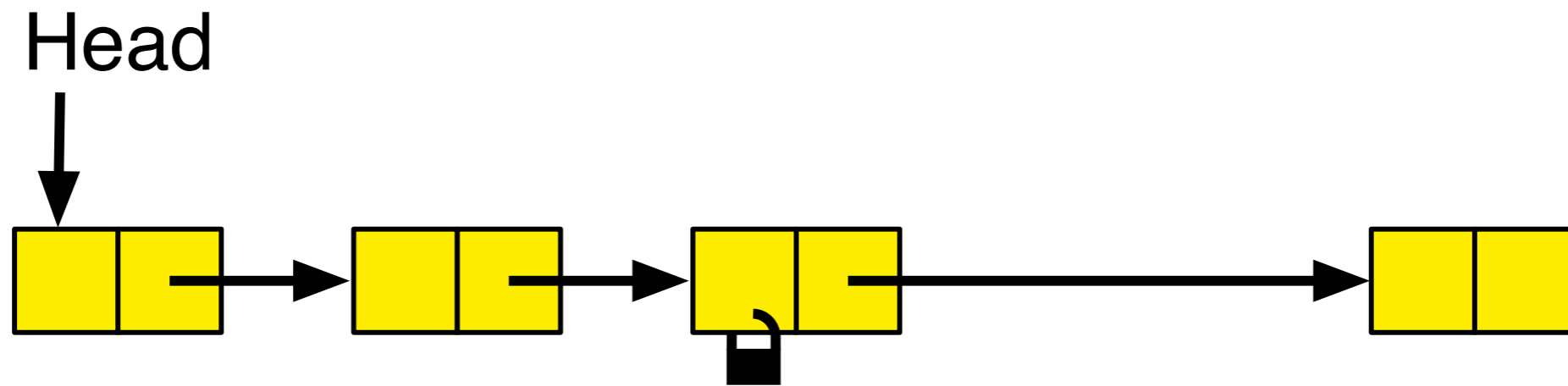


Fine-grained locking



Remove locked node
by a pointer swing

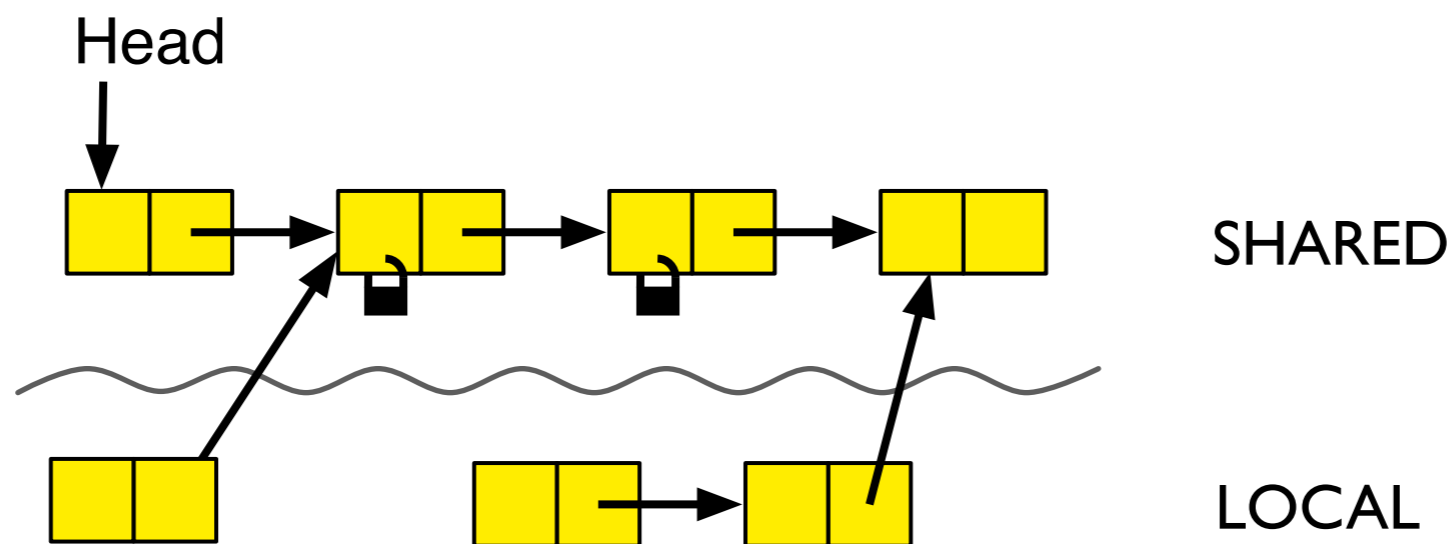
Fine-grained locking



Free the
redundant node

The RGSep approach

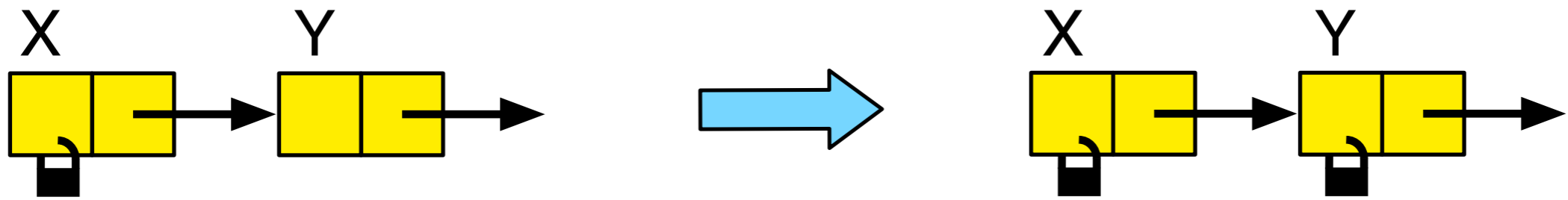
Divide the heap into shared and local state



Model interference over the shared state only

RGSep actions

Locked nodes stay in the shared state

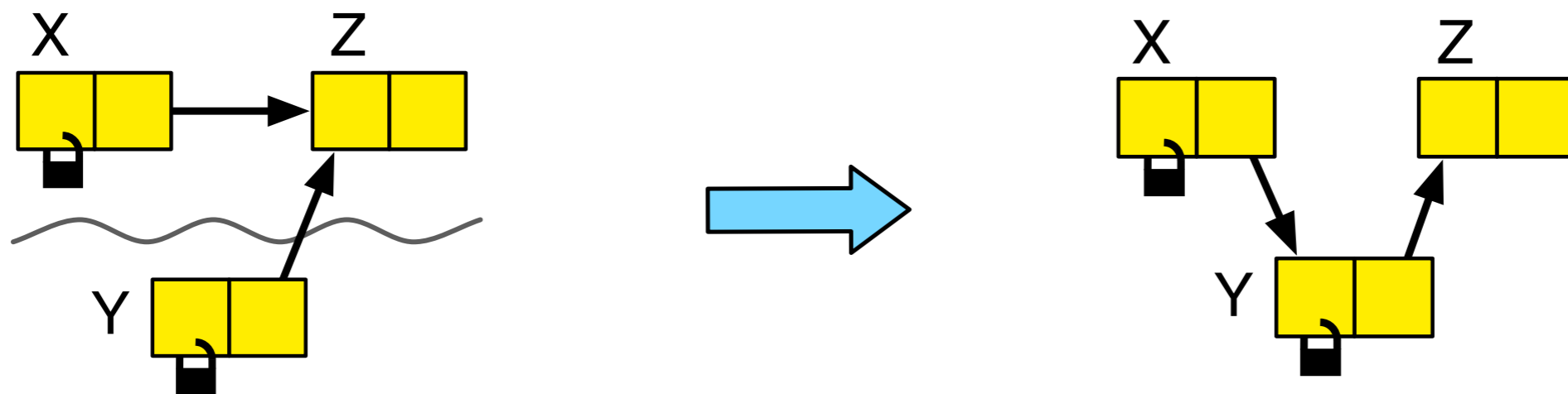


Consequently, blocks have to be manipulated explicitly by actions

RGSep actions

Nodes added and removed by explicit pointer swings

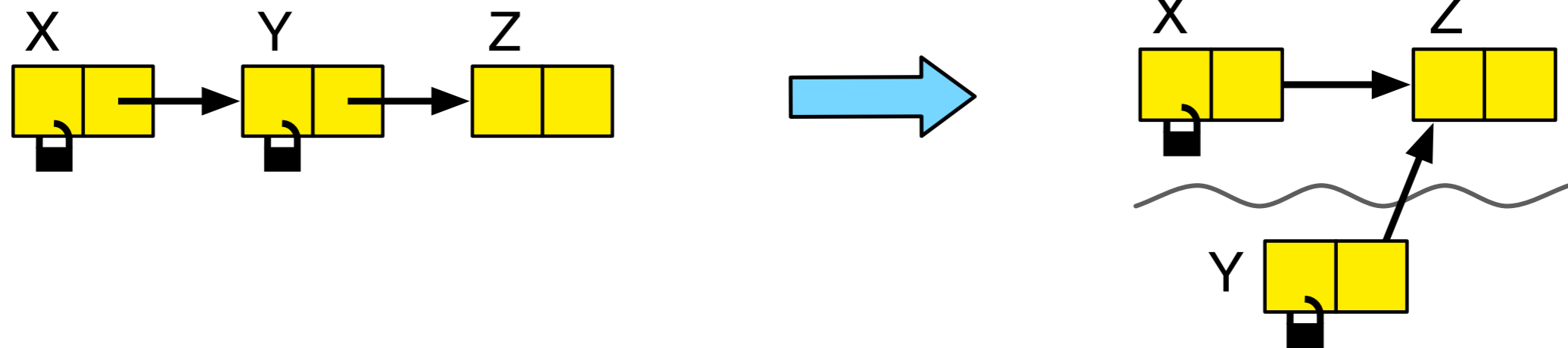
ADD:



RGSep actions

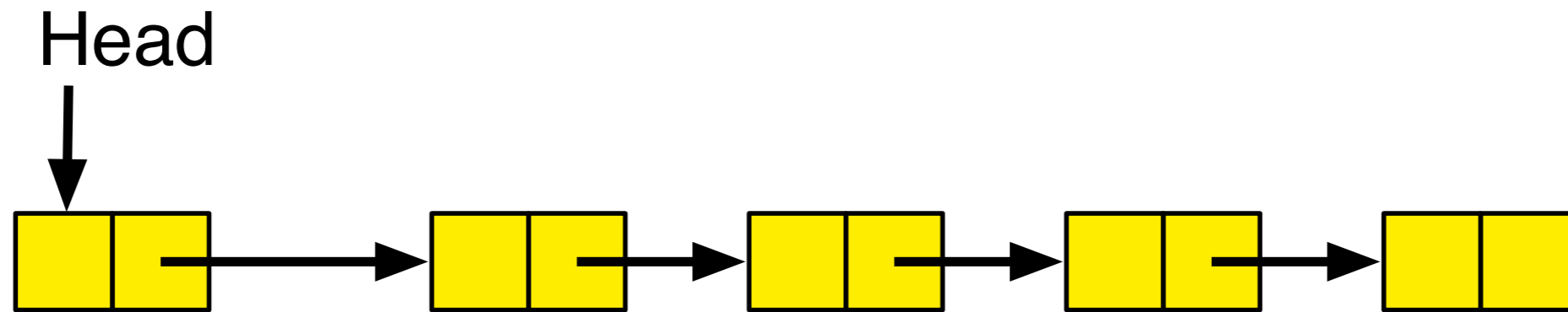
Nodes added and removed by explicit pointer swings

REMOVE:

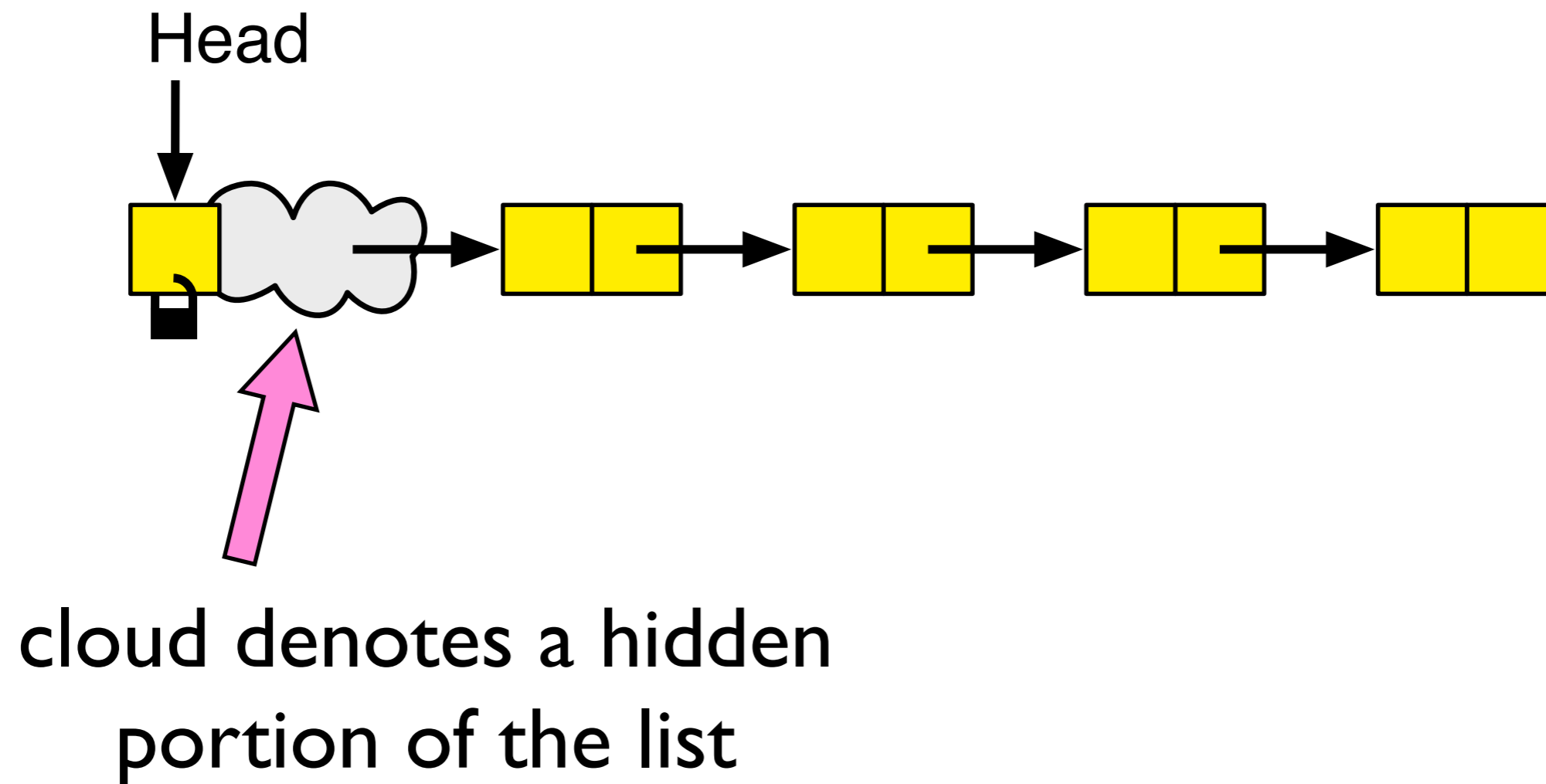


This is too restrictive

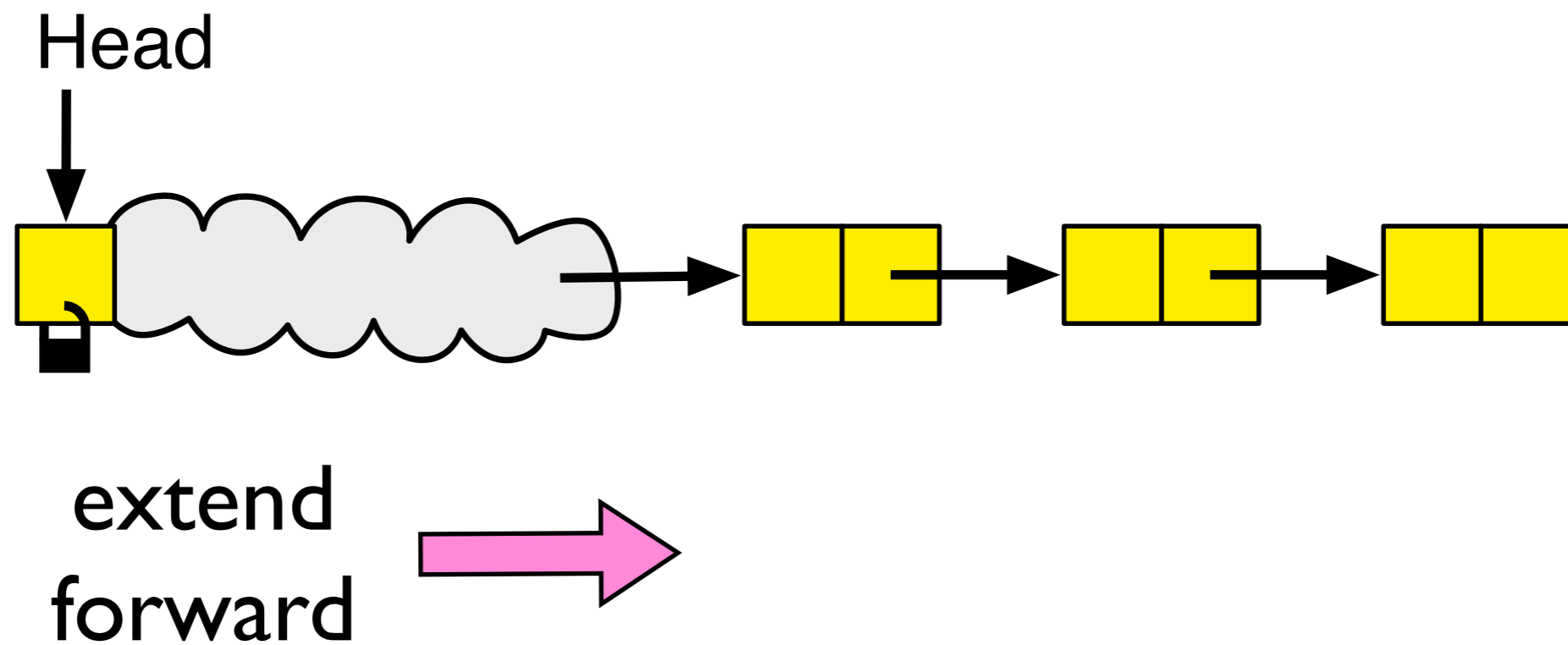
Conceptual view



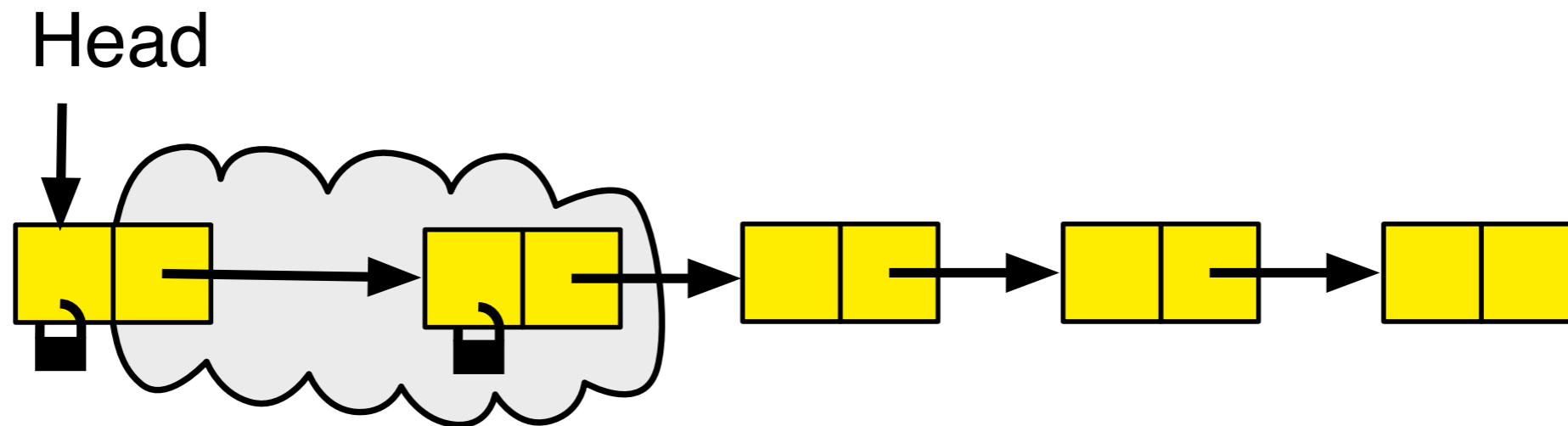
Conceptual view



Conceptual view

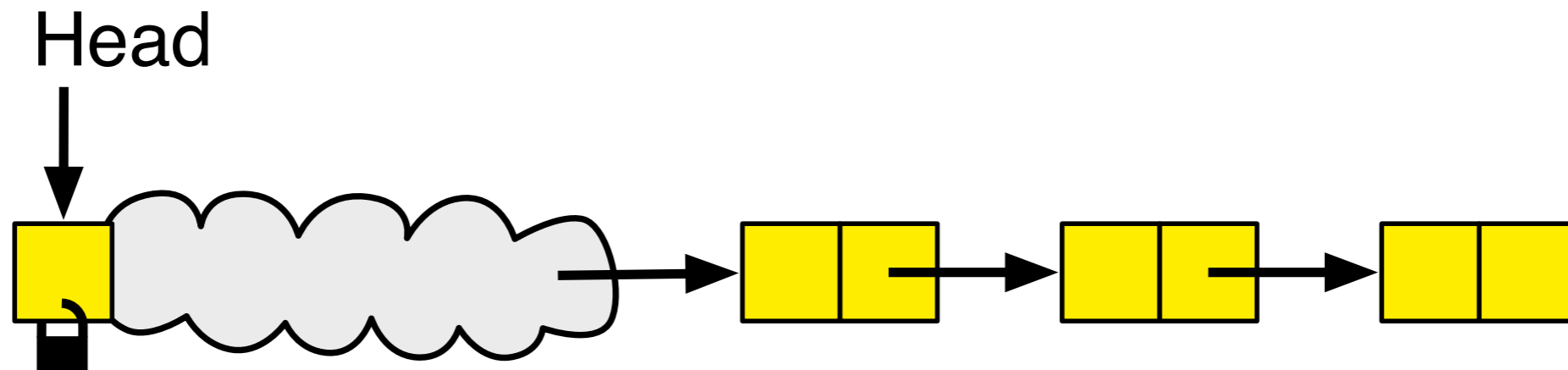


Conceptual view

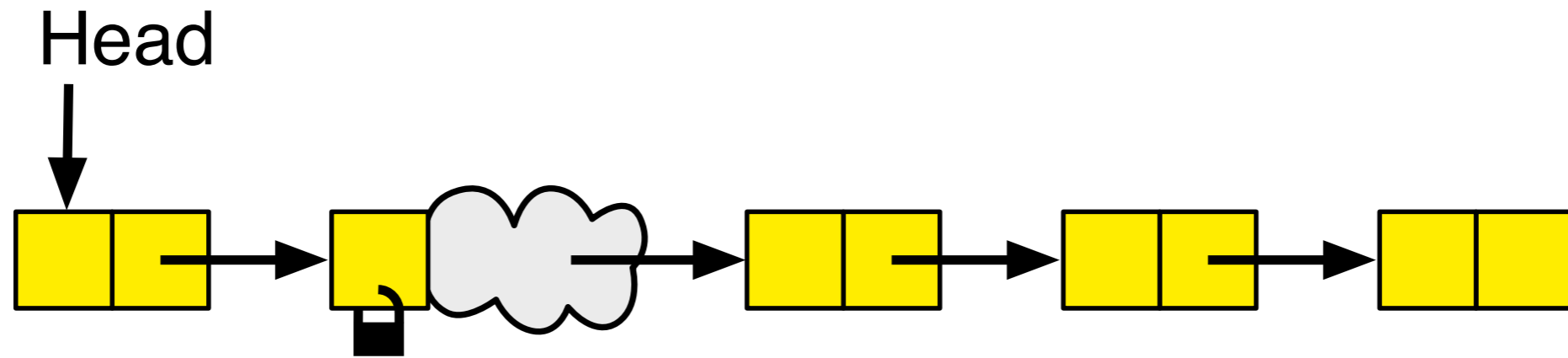


locked nodes are hidden
from the public state

Conceptual view

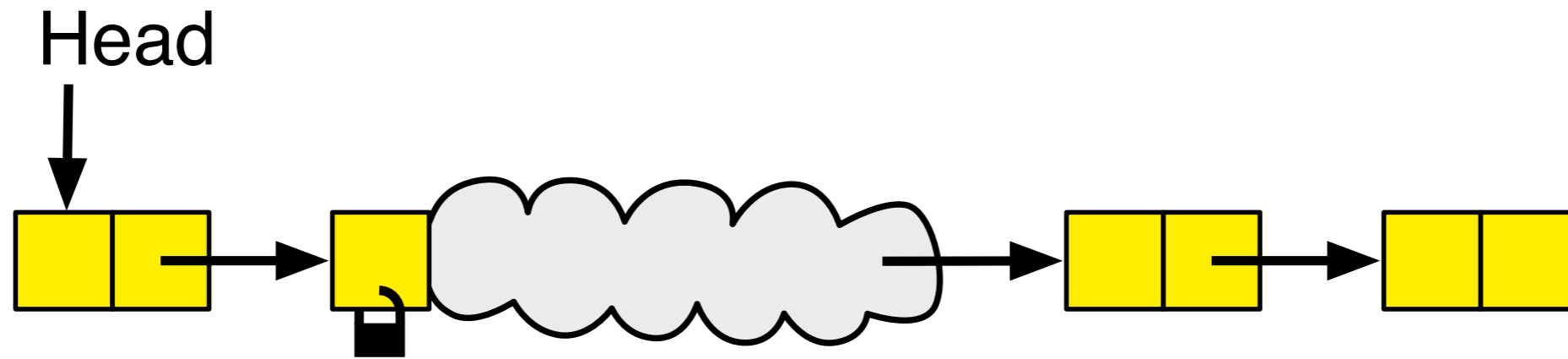


Conceptual view

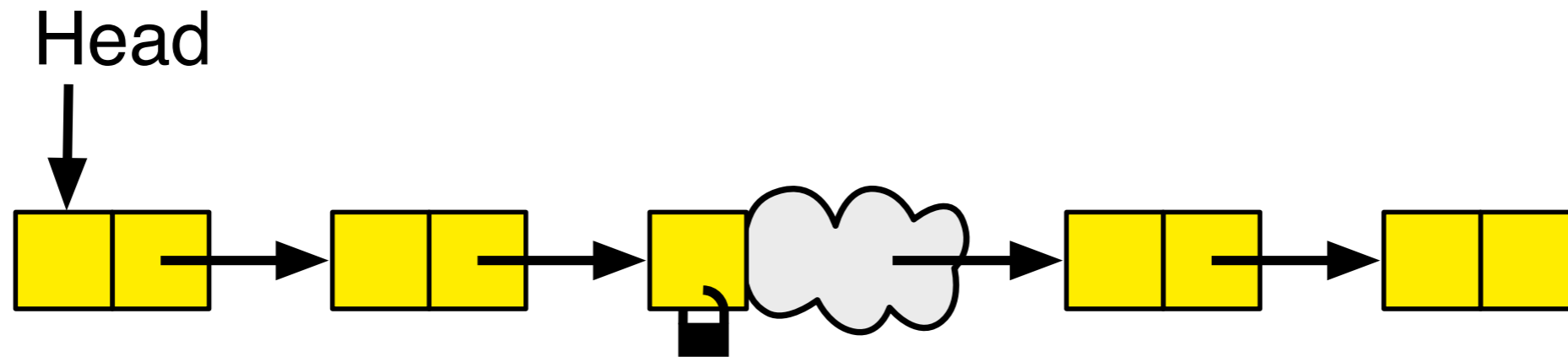


release some list from
the hidden state

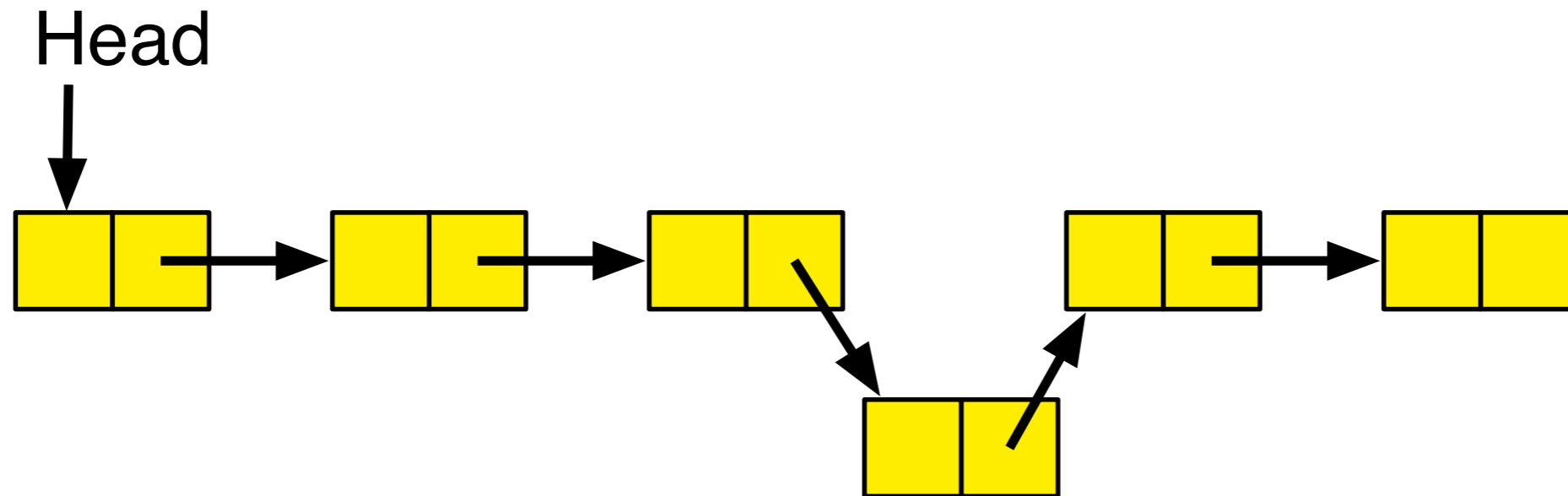
Conceptual view



Conceptual view

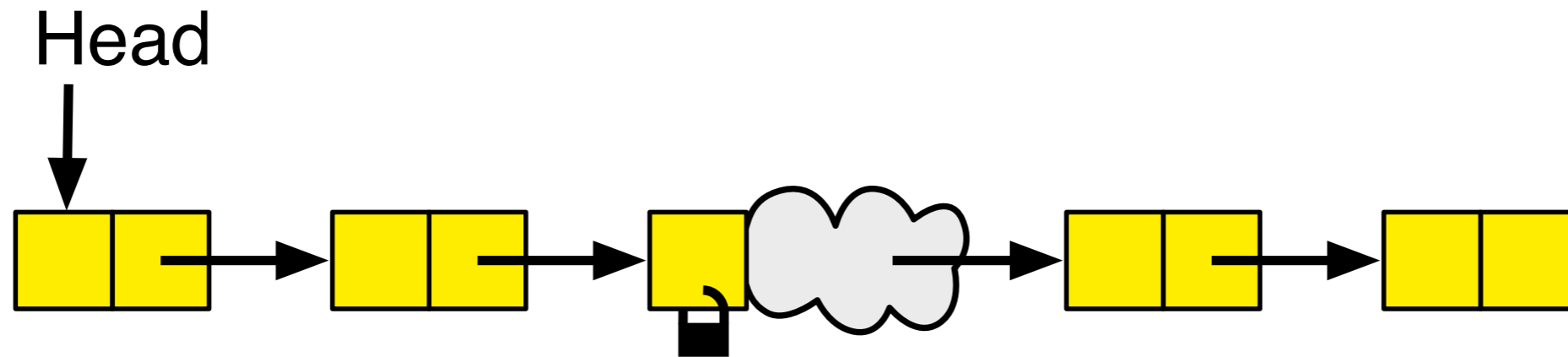


Conceptual view

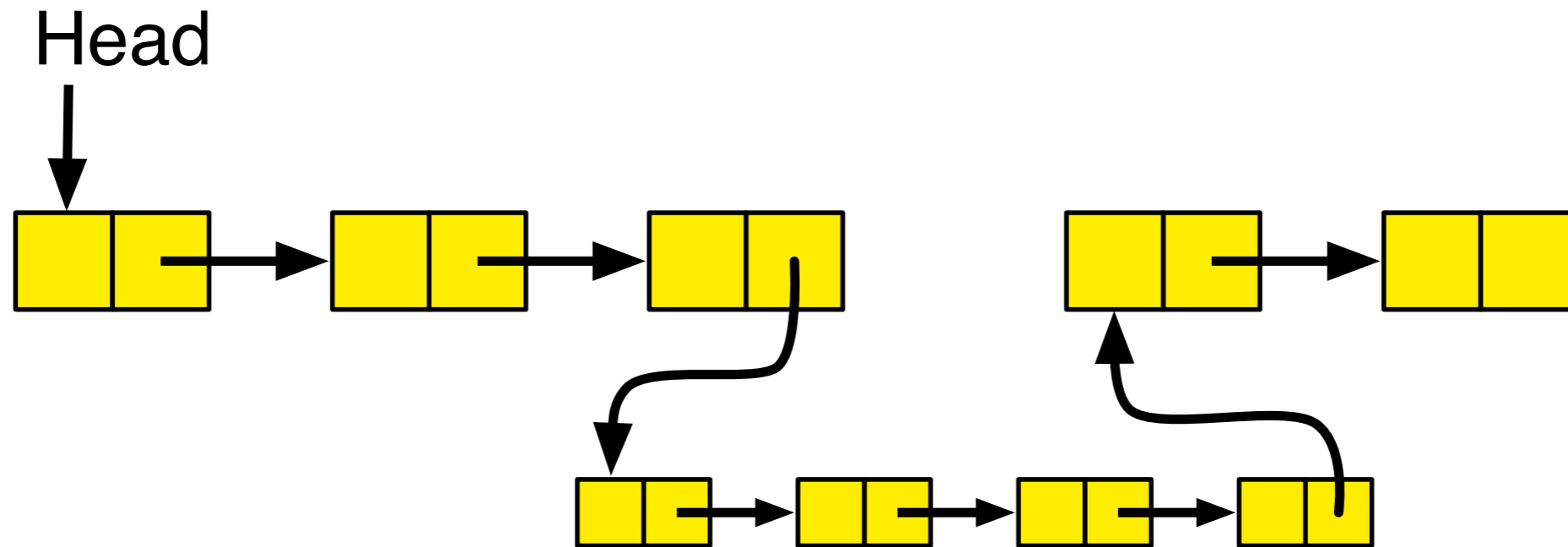


can remove the lock and release
a single node from hidden area

Conceptual view

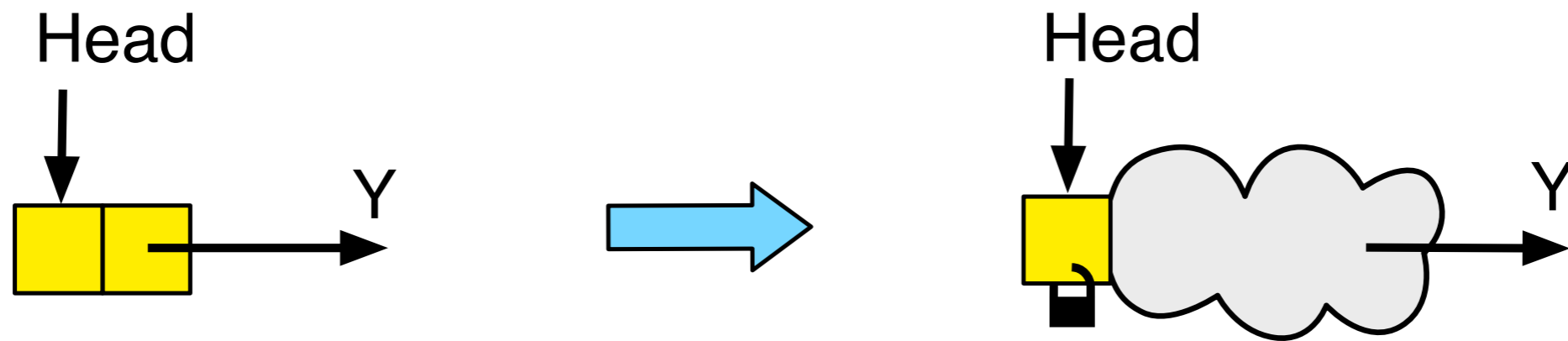


Conceptual view



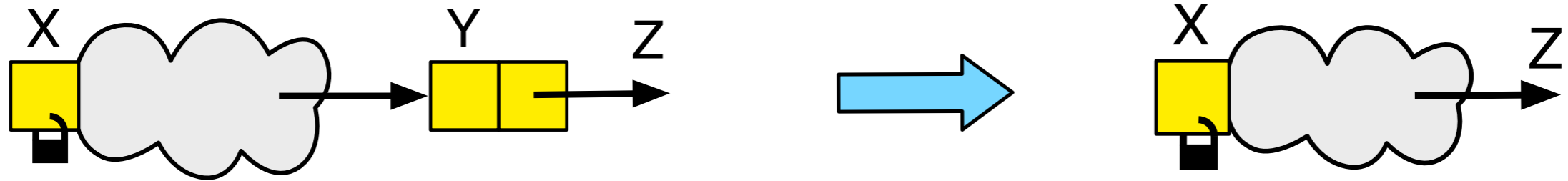
can replace hidden area
with arbitrary unlocked list

Intuitive actions



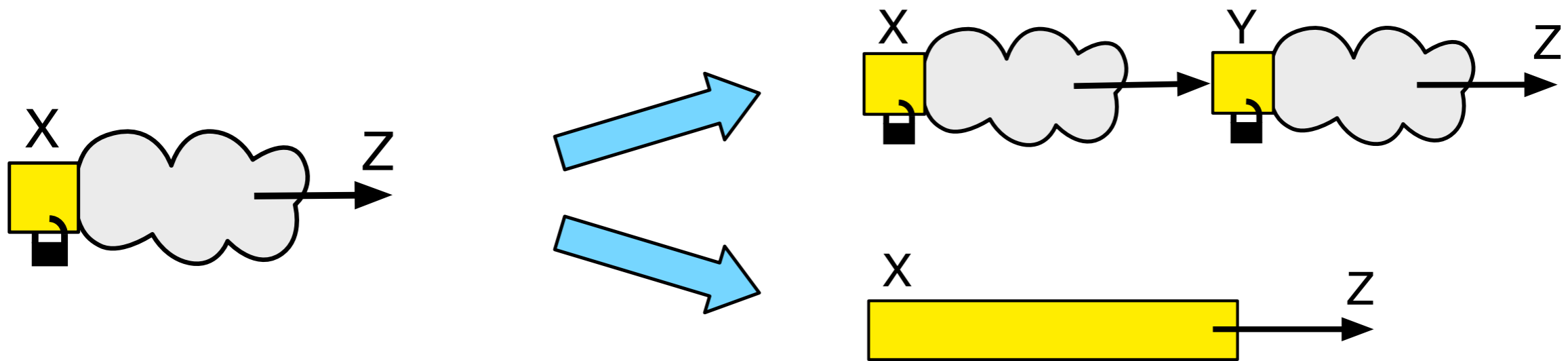
Lock the head and
add a hidden section

Intuitive actions



Extend the hidden section
(hide more of the list)

Intuitive actions



Split the hidden section,

or

return an unlocked list segment

How can we capture these actions?

Deny-guarantee Permissions

Associate each action with a member of PermDG:

$$pr \in \text{Perms} \stackrel{\text{def}}{=} \text{Actions} \rightarrow \text{PermDG}$$

Objects in PermDG capture the four ways an action can be permitted:

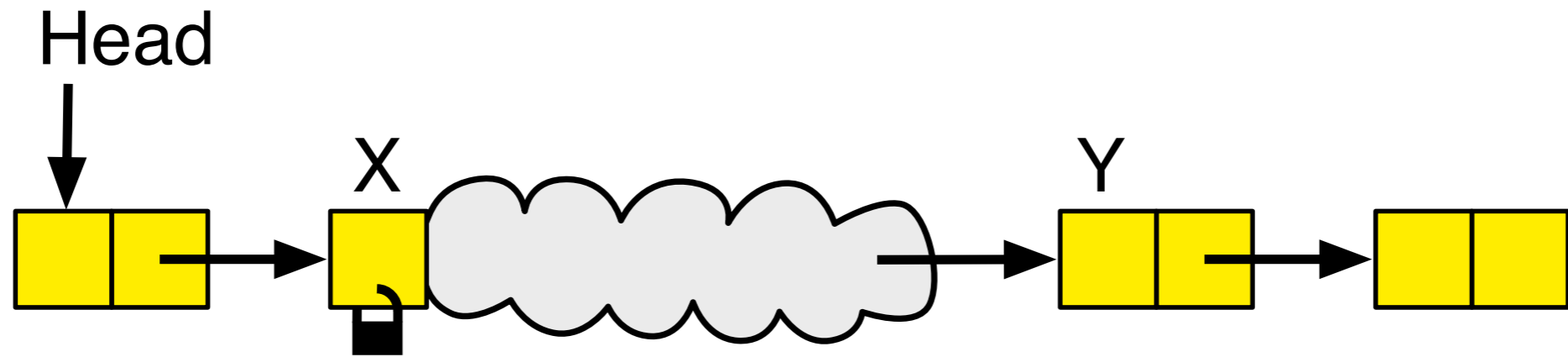
thread, environment	not thread, environment
thread, not environment	not thread, not environment

Local and shared state

An action is permitted to a thread if it's permitted by the local state.

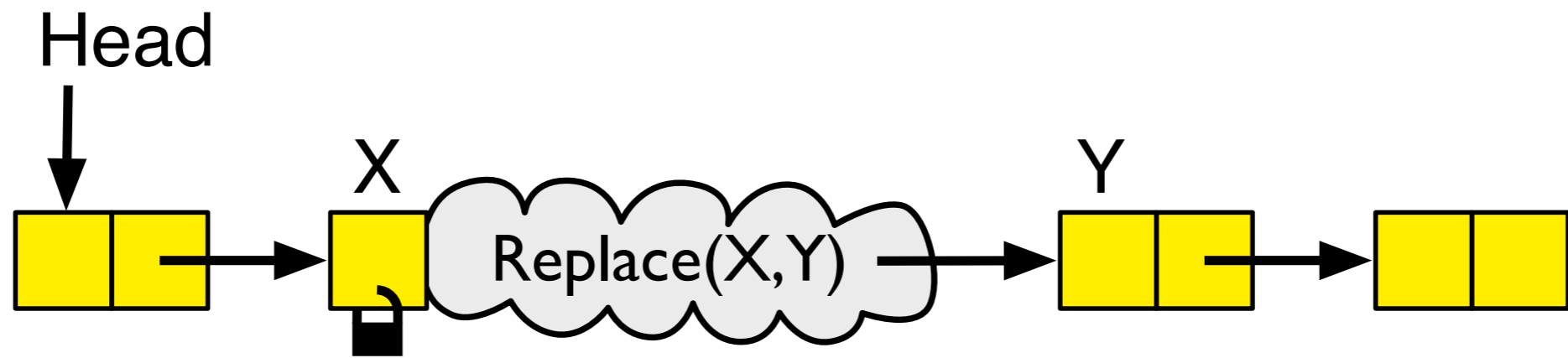
An action can occur as interference if it's possible according to the local and shared state

Gaps as permissions



Permission to insert a list
from $(X+1)$ to Y , or
extend the permission

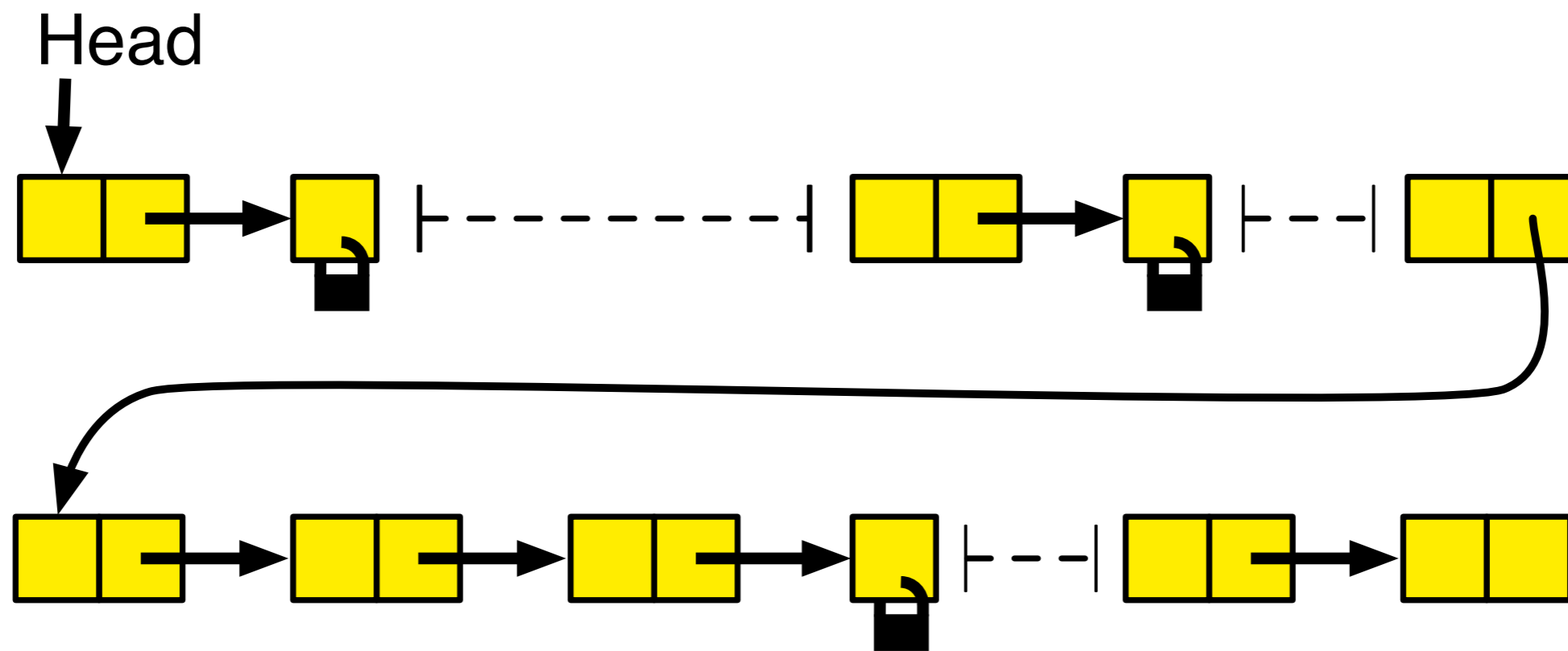
Gaps as permissions



Permission to insert a list
from $(X+1)$ to Y , or
extend the permission

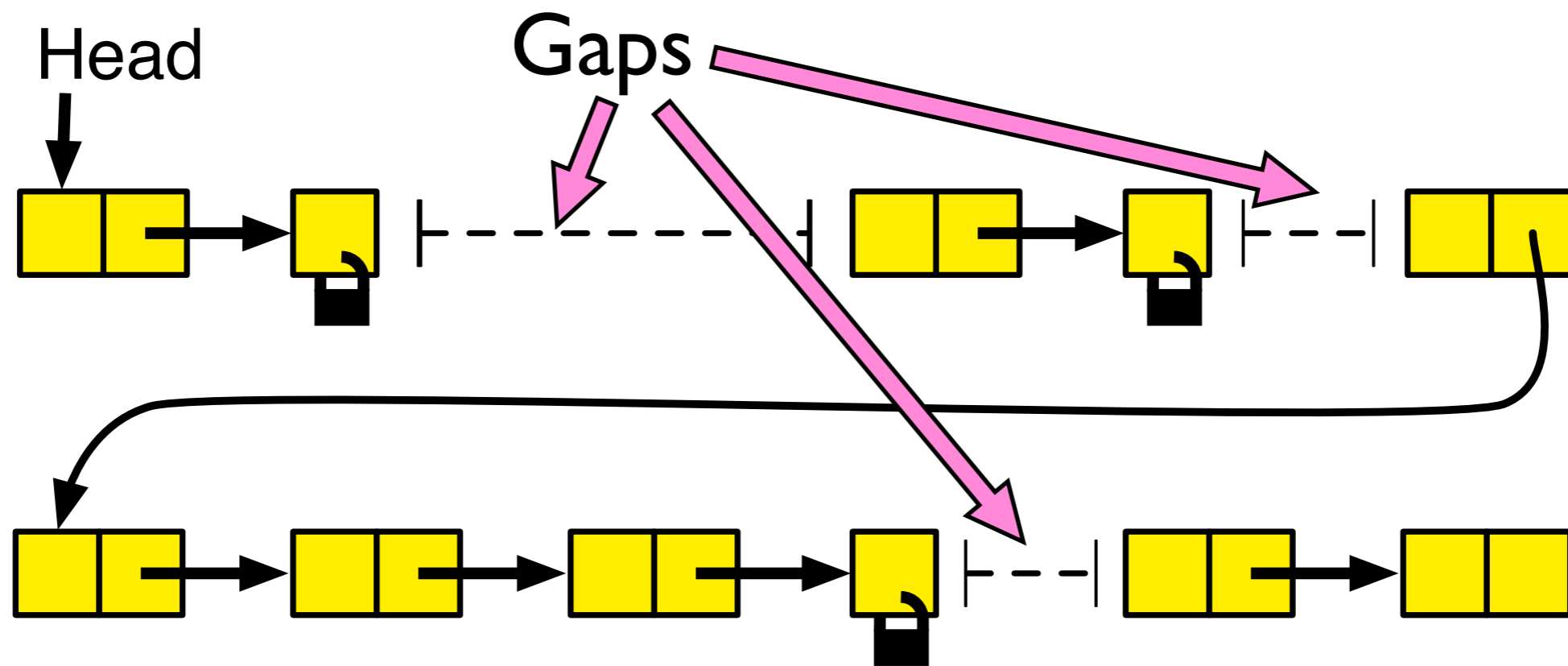
Shared state

Shared state consists of a list with gaps:



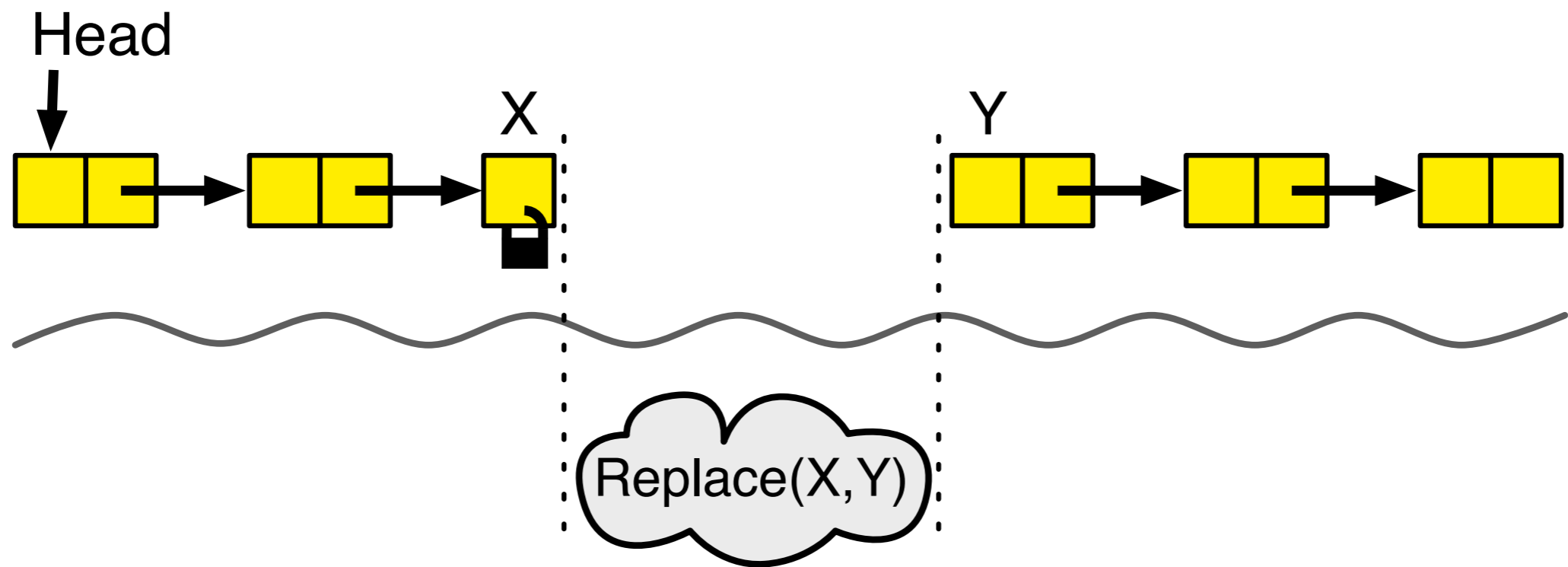
Shared state

Shared state consists of a list with gaps:



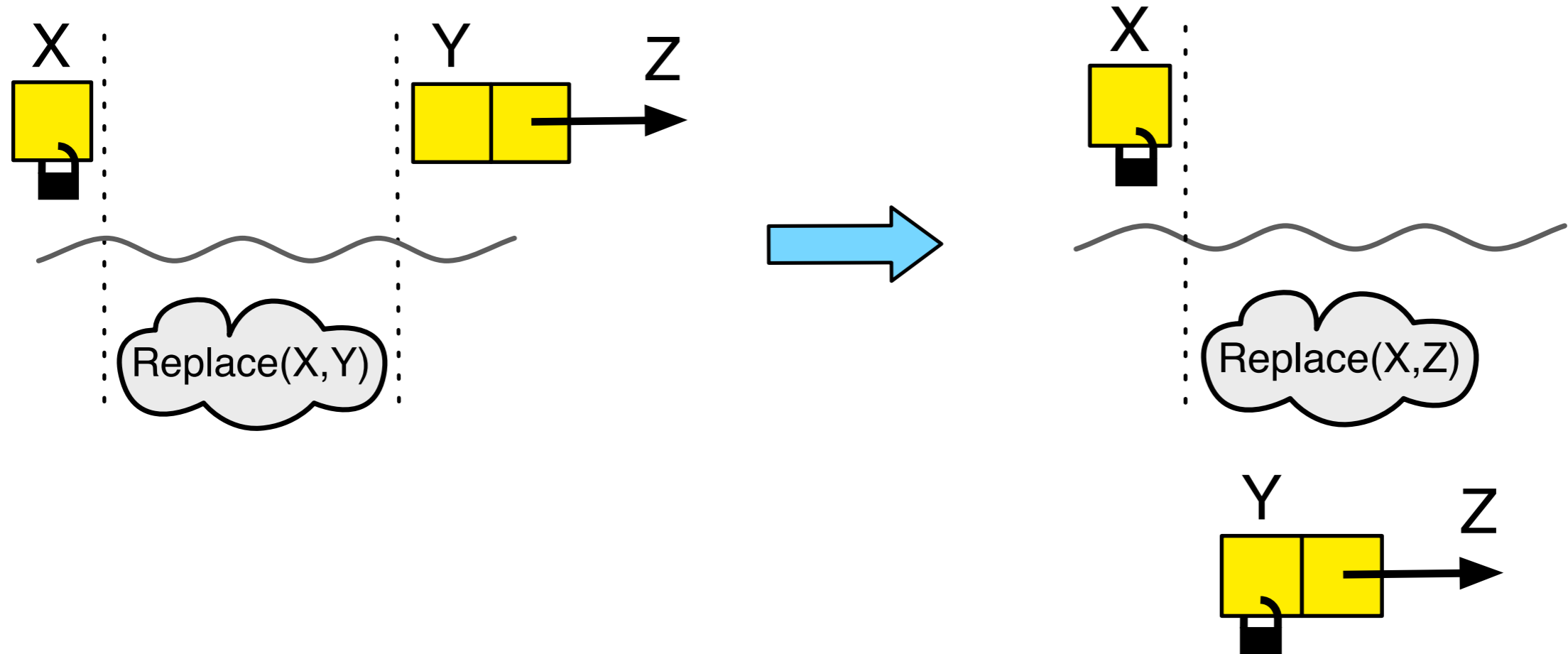
Local state

Permissions are held in local state

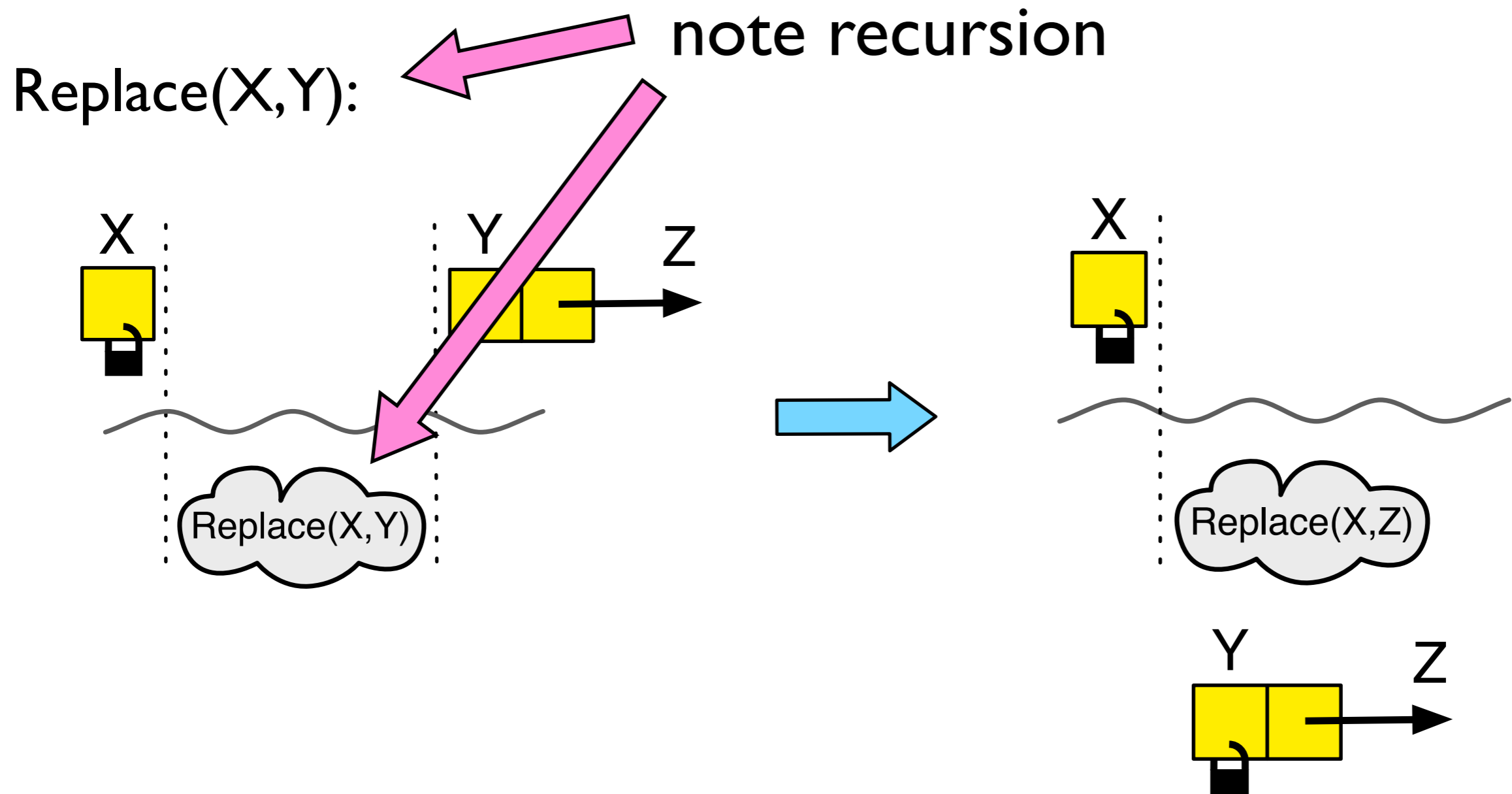


Higher-order actions

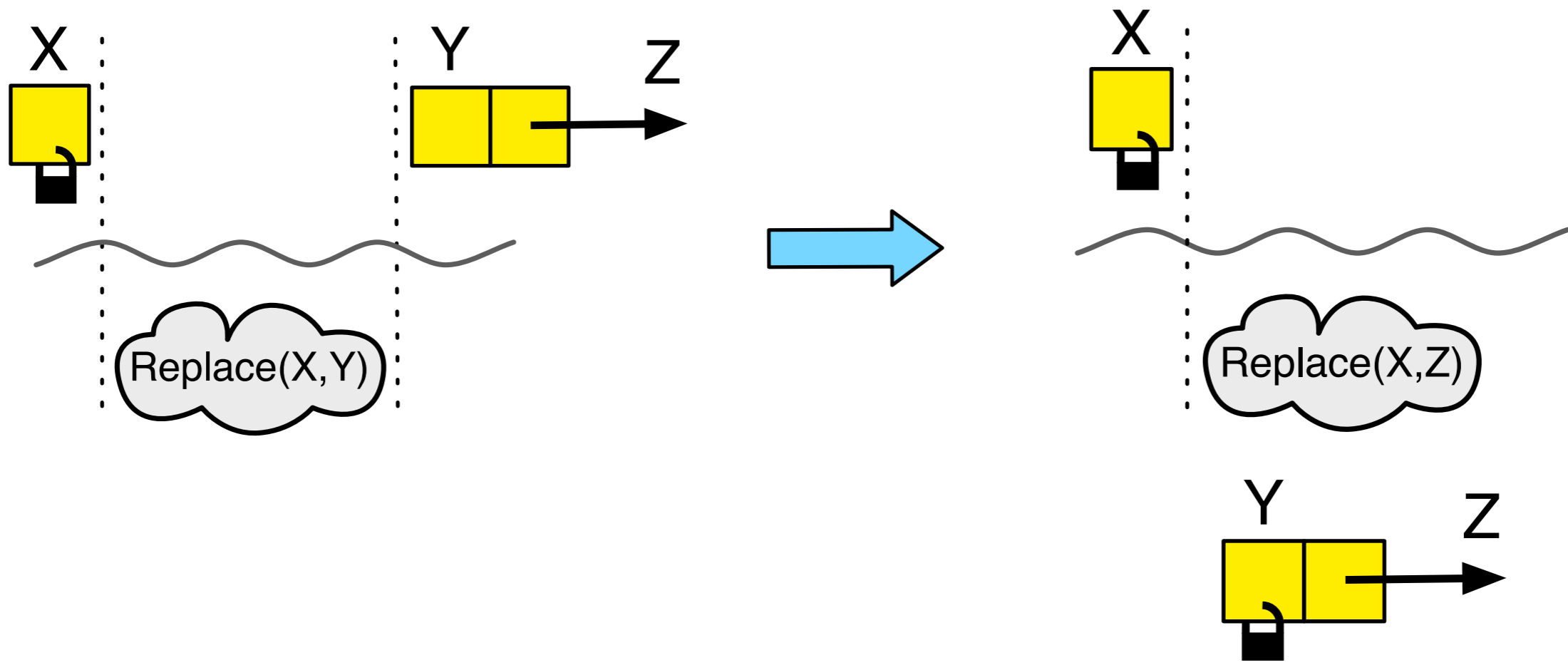
Replace(X,Y):



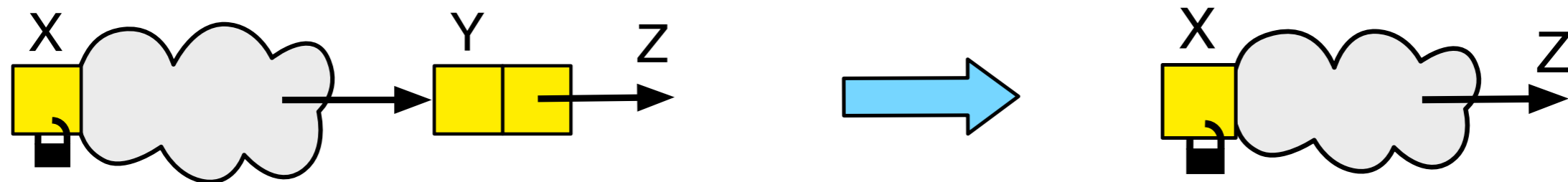
Higher-order actions



Replace(X,Y):

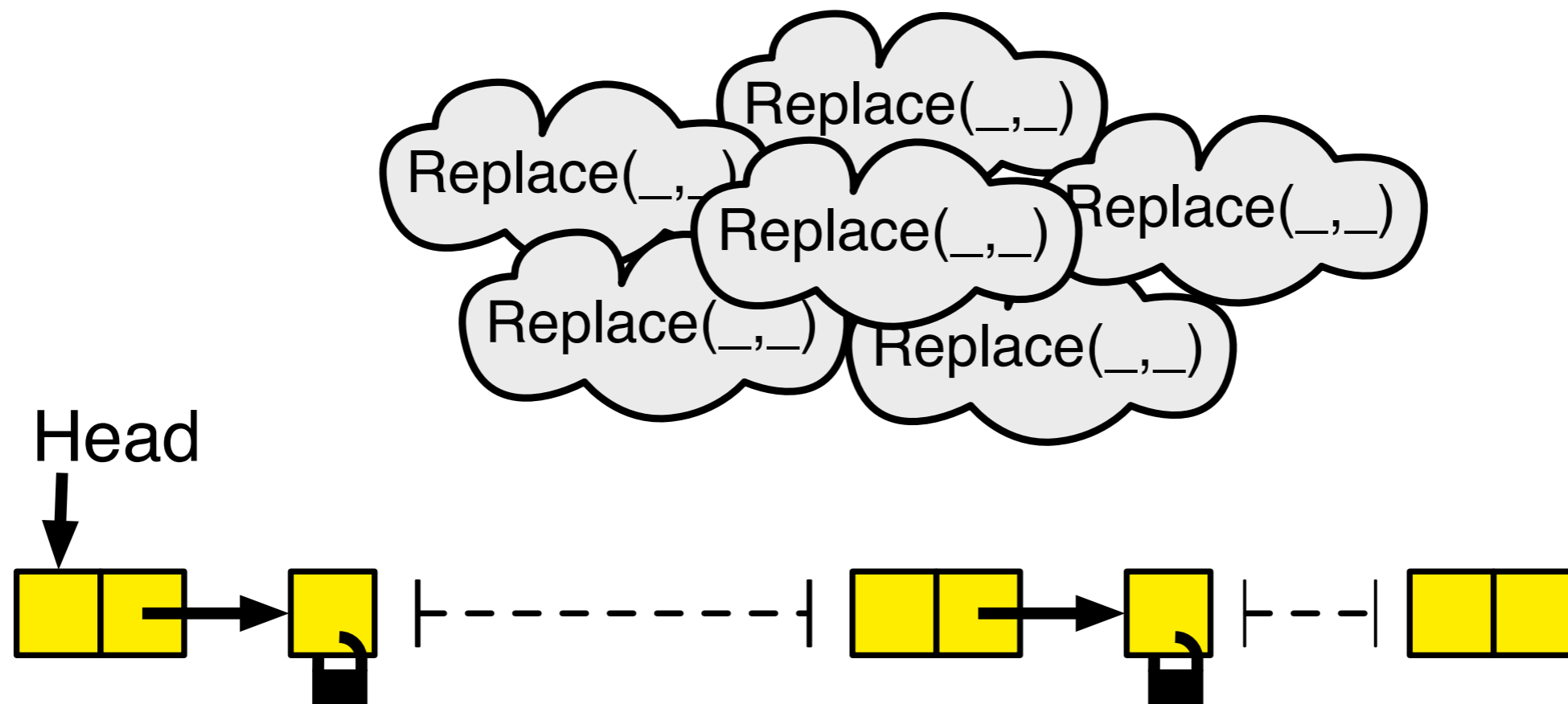


Corresponds to this intuitive action:



Unused actions

Unused permissions are held in the public state.



Unused actions

Shared state contains all possible unused actions

- This is a bug, not a feature!
- Makes invariants more complex

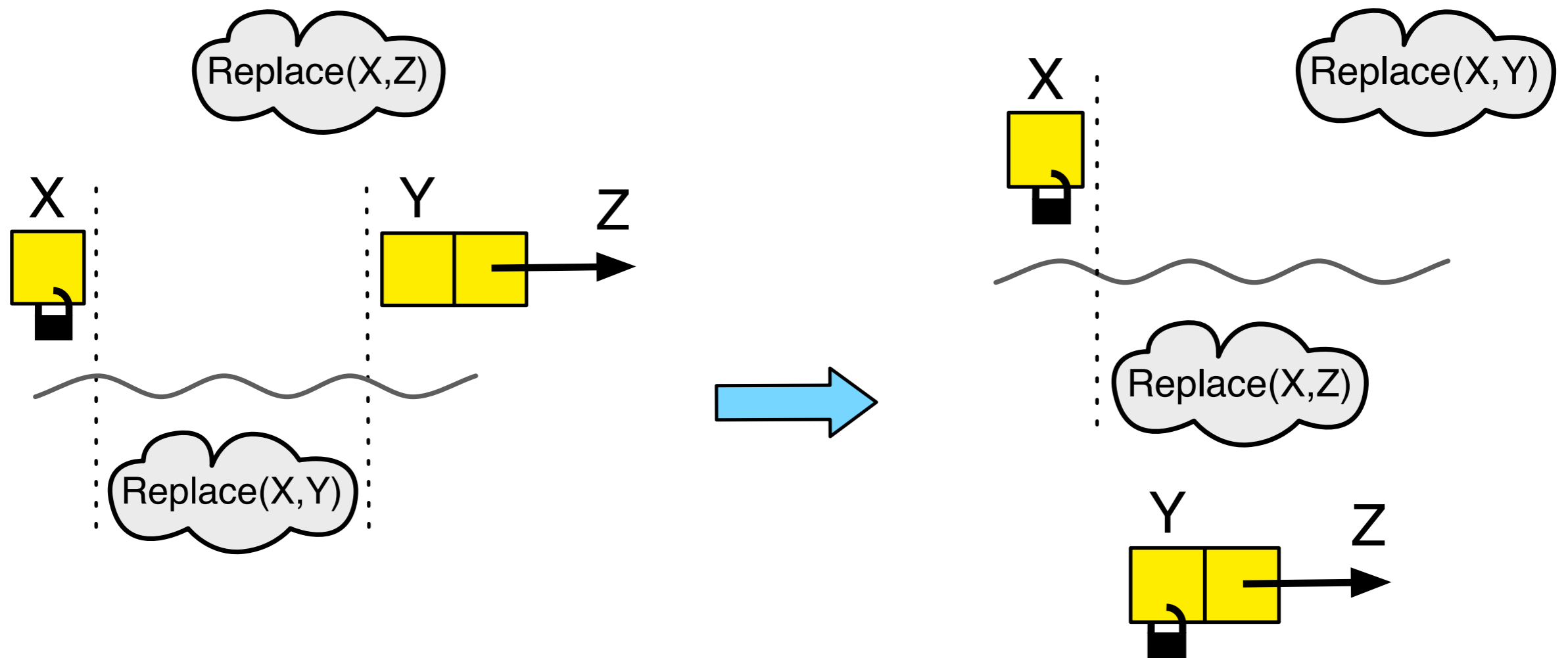
Source of the problem: actions are global

Solution: *local* deny-guarantee

(but we don't know how to do this, yet)

Accounting for actions

Replace(X,Y):



Higher-order actions

Permissions are defined over action *names*

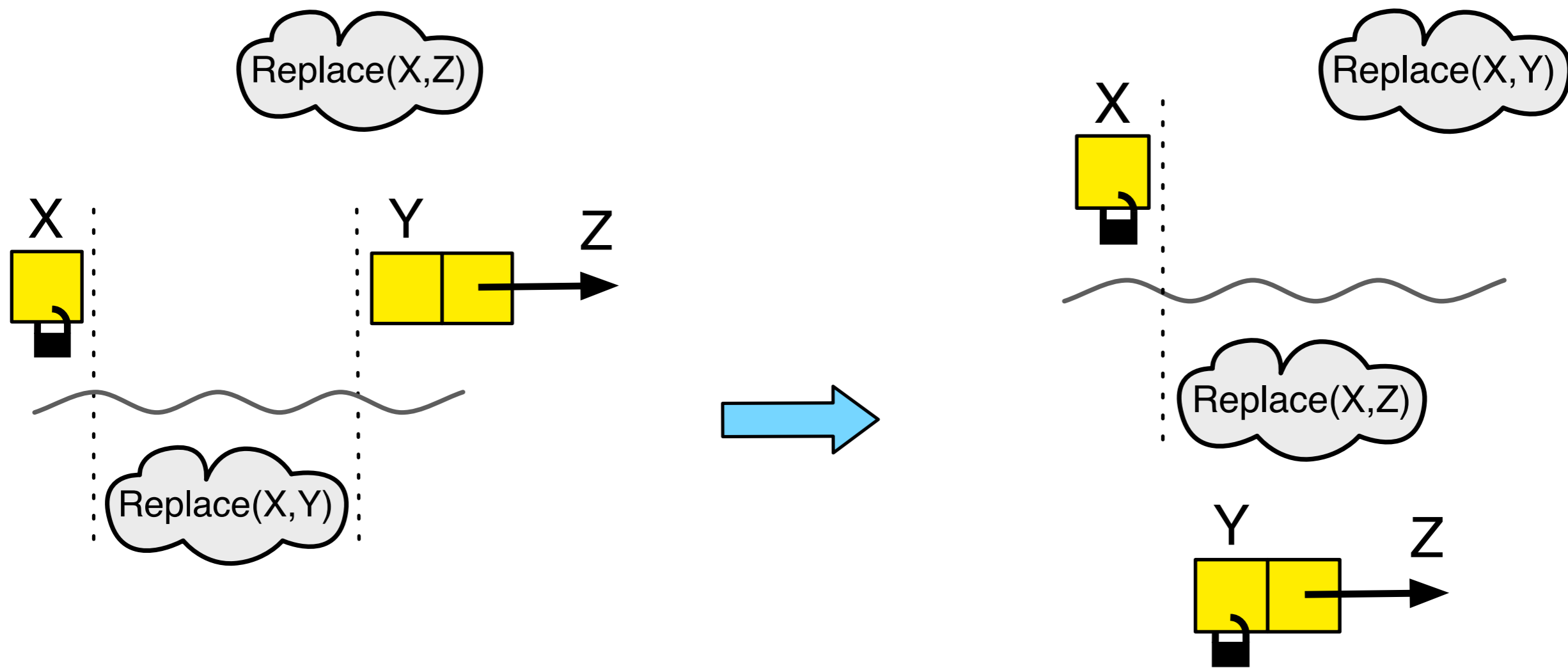
$$pr \in \text{Perms} \stackrel{\text{def}}{=} \text{Names} \times \text{Vals}^* \rightarrow \text{PermDG}$$

Semantics defined by an environment

$$\text{Worlds} \stackrel{\text{def}}{=} \text{Perms} \times \text{States}$$

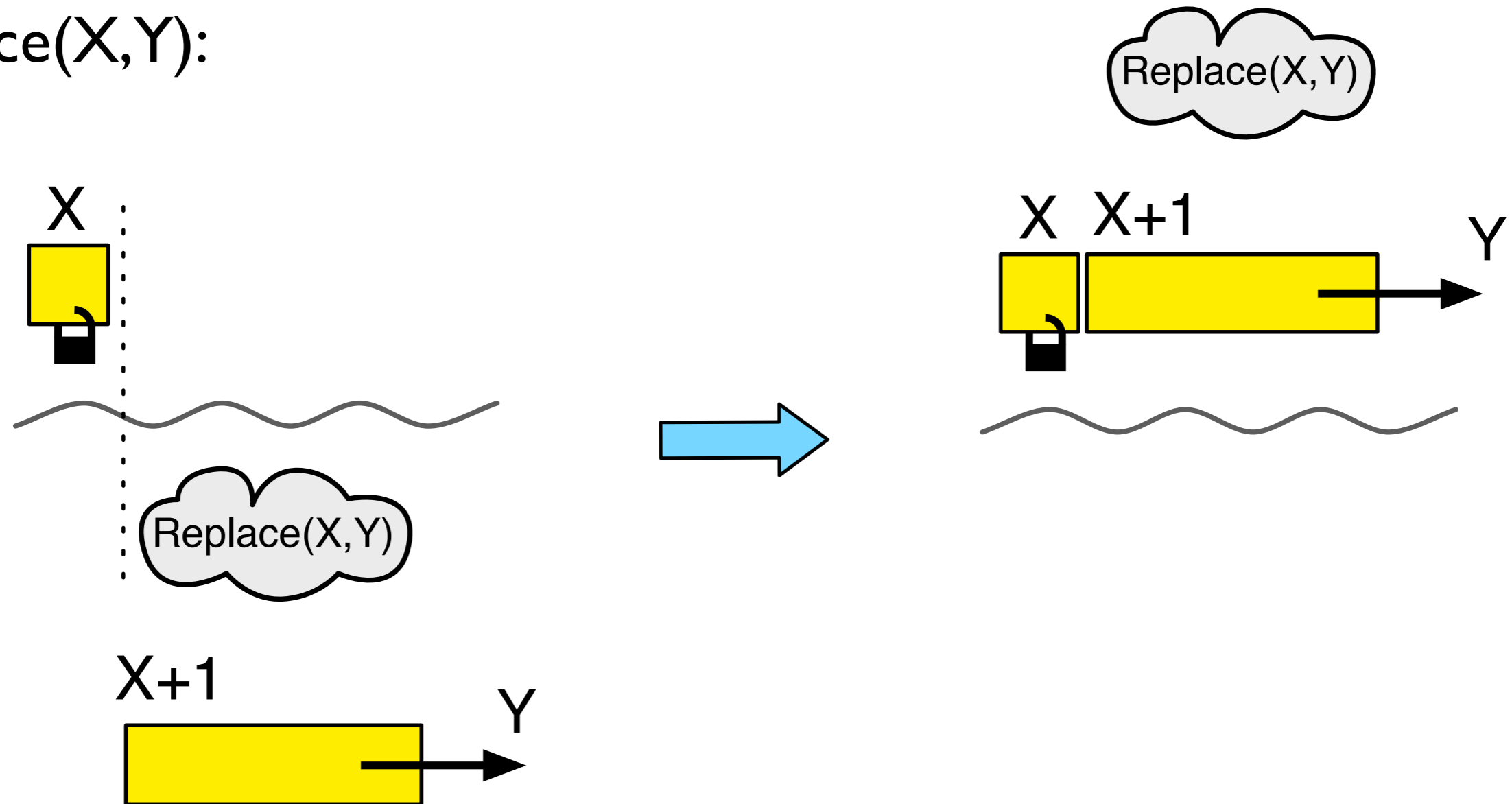
$$\eta \in \text{Envs} \stackrel{\text{def}}{=} \text{Names} \times \text{Vals}^* \rightarrow \mathcal{P}(\text{Worlds} \times \text{Worlds})$$

$$\text{REP}(x, y) : \frac{\text{L}(x) * [\text{REP}(x, z)]_1 * \text{Un}(y, v, z)}{[\text{REP}(x, y)]_1} \rightsquigarrow \frac{\text{L}(x) * [\text{REP}(x, y)]_1}{[\text{REP}(x, z)]_1 * \text{Un}(y, v, z)}$$



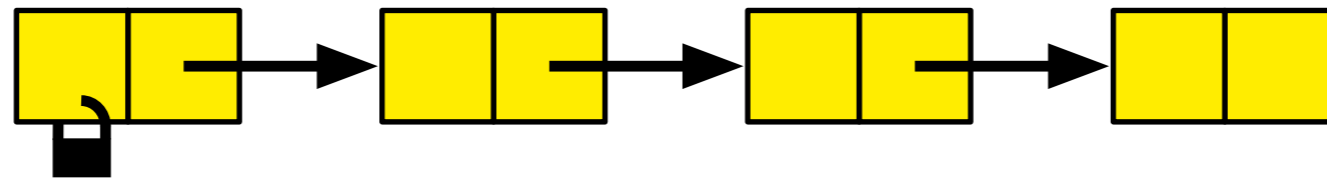
Replace is overloaded

Replace(X,Y):



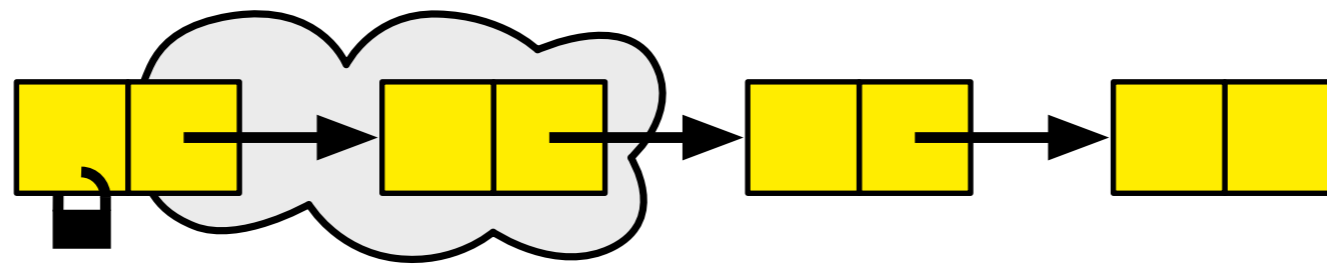
Algorithm insights

Lock extension just requires a test, not locking



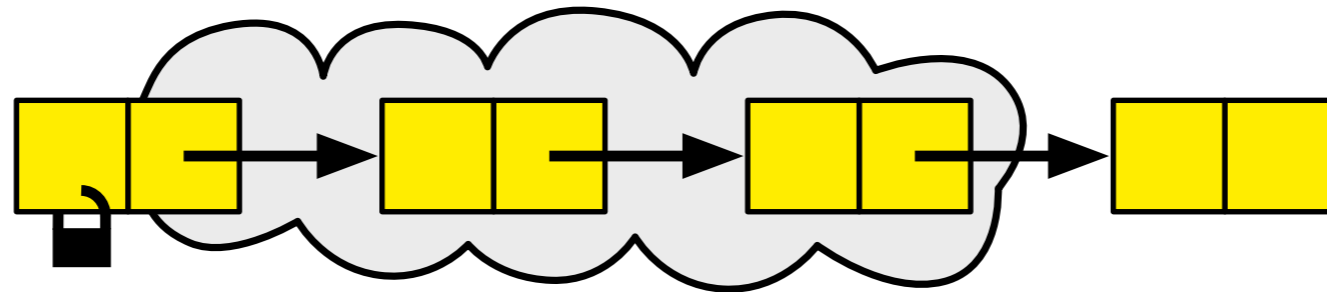
Algorithm insights

Lock extension just requires a test, not locking



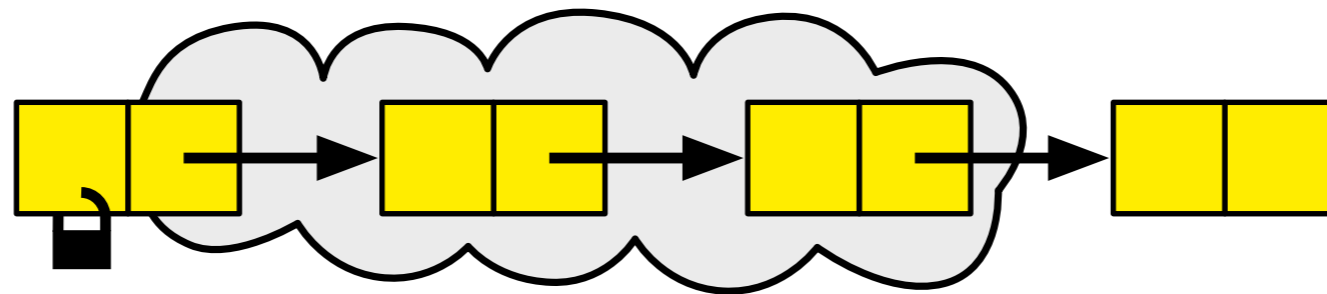
Algorithm insights

Lock extension just requires a test, not locking



Algorithm insights

Lock extension just requires a test, not locking



Consequently, we don't need a CAS to extend.
(we do need one at the list head)

Other applications

We can prove M. Michael's non-blocking stack without auxiliary variables

Updates to hazard pointers also manipulate permission on Push actions

Status and future work

Work is ongoing:

- Don't yet have a full semantics, proof of soundness etc.
- We're looking for more examples

Future objectives:

- Elimination of auxiliary variables?
- Local deny-guarantee?

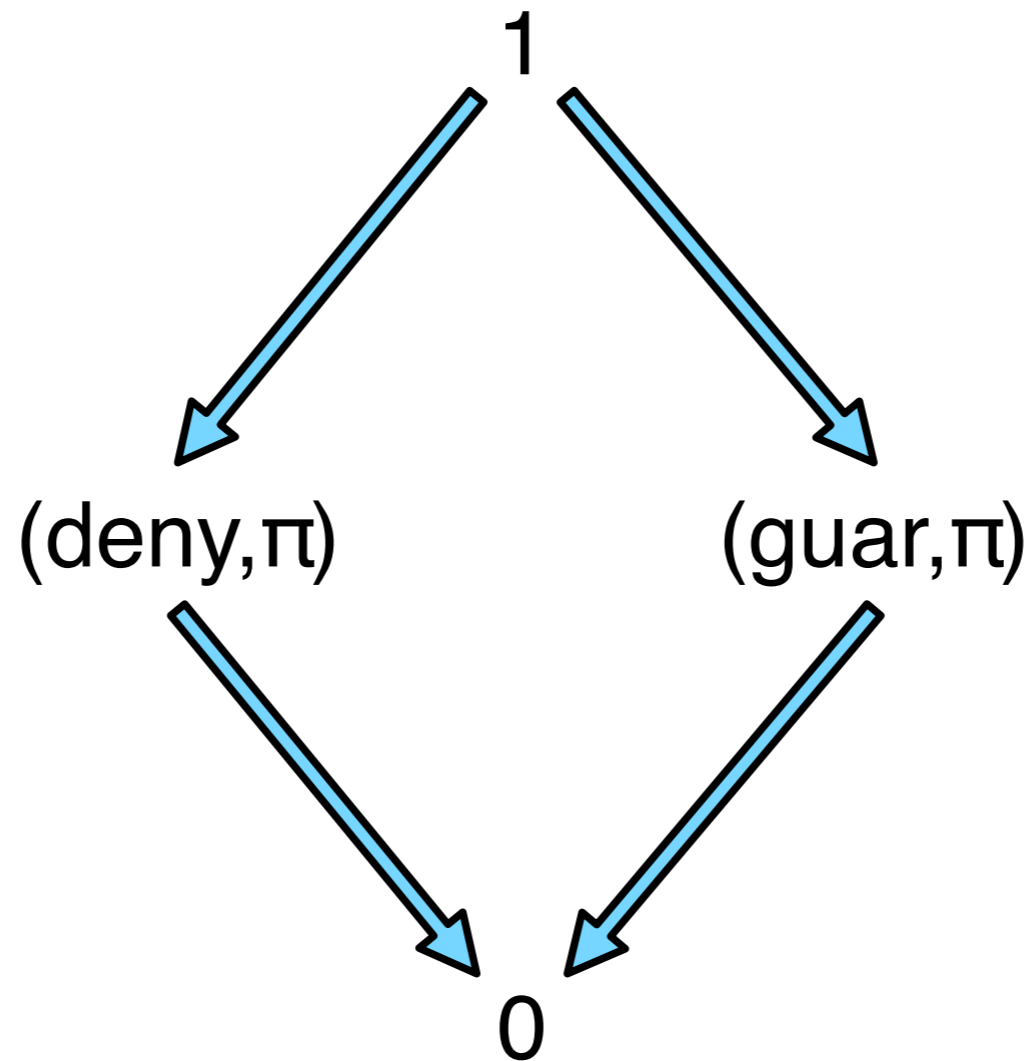
Any questions?

Deny-guarantee Semantics

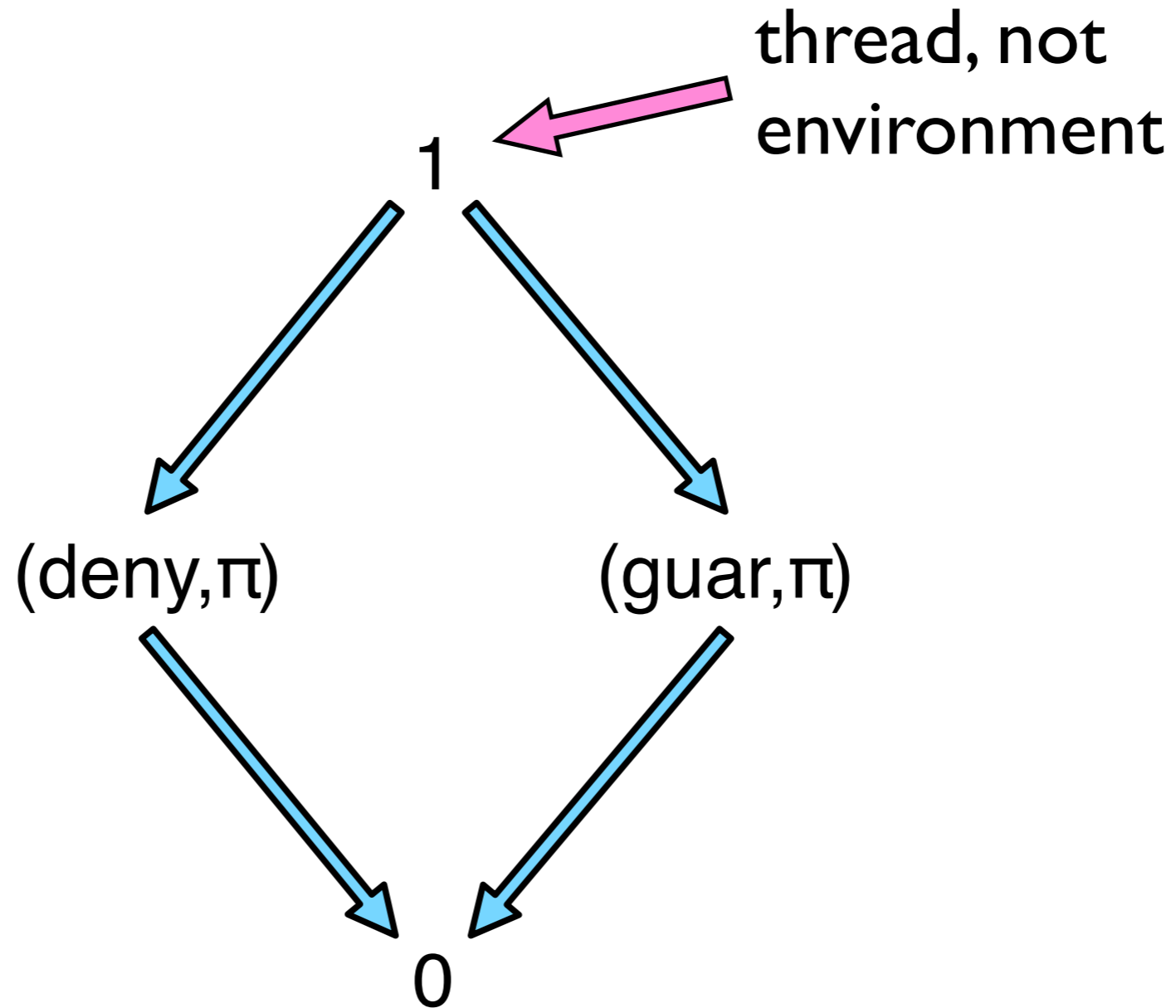
PermDG is defined as follows:

$$\begin{aligned} \text{PermDG} &\stackrel{\text{def}}{=} (\{\text{deny}\} \times (0..1)) \\ &\uplus (\{\text{guar}\} \times (0..1)) \\ &\uplus \{1\} \uplus \{0\} \end{aligned}$$

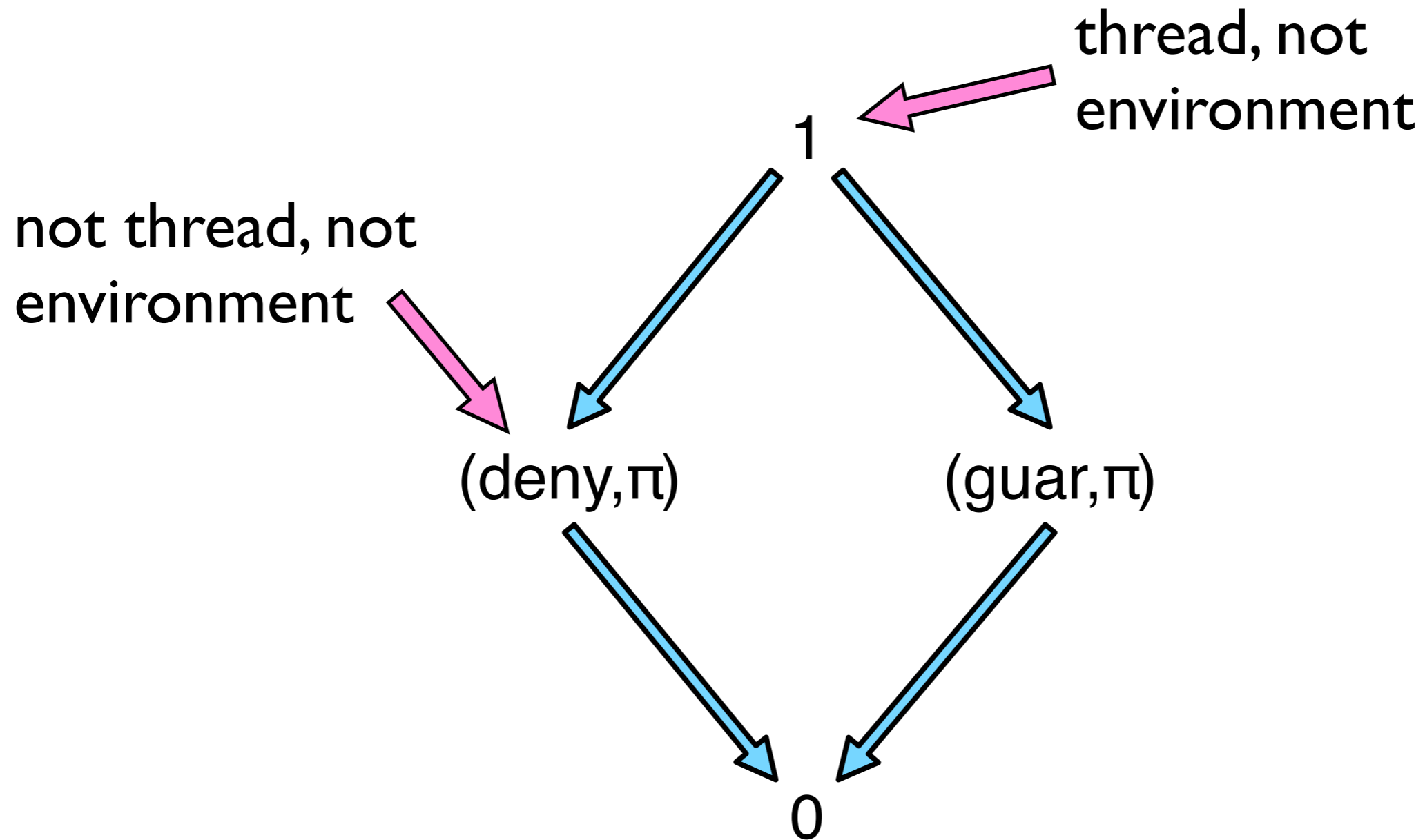
Meaning of permission



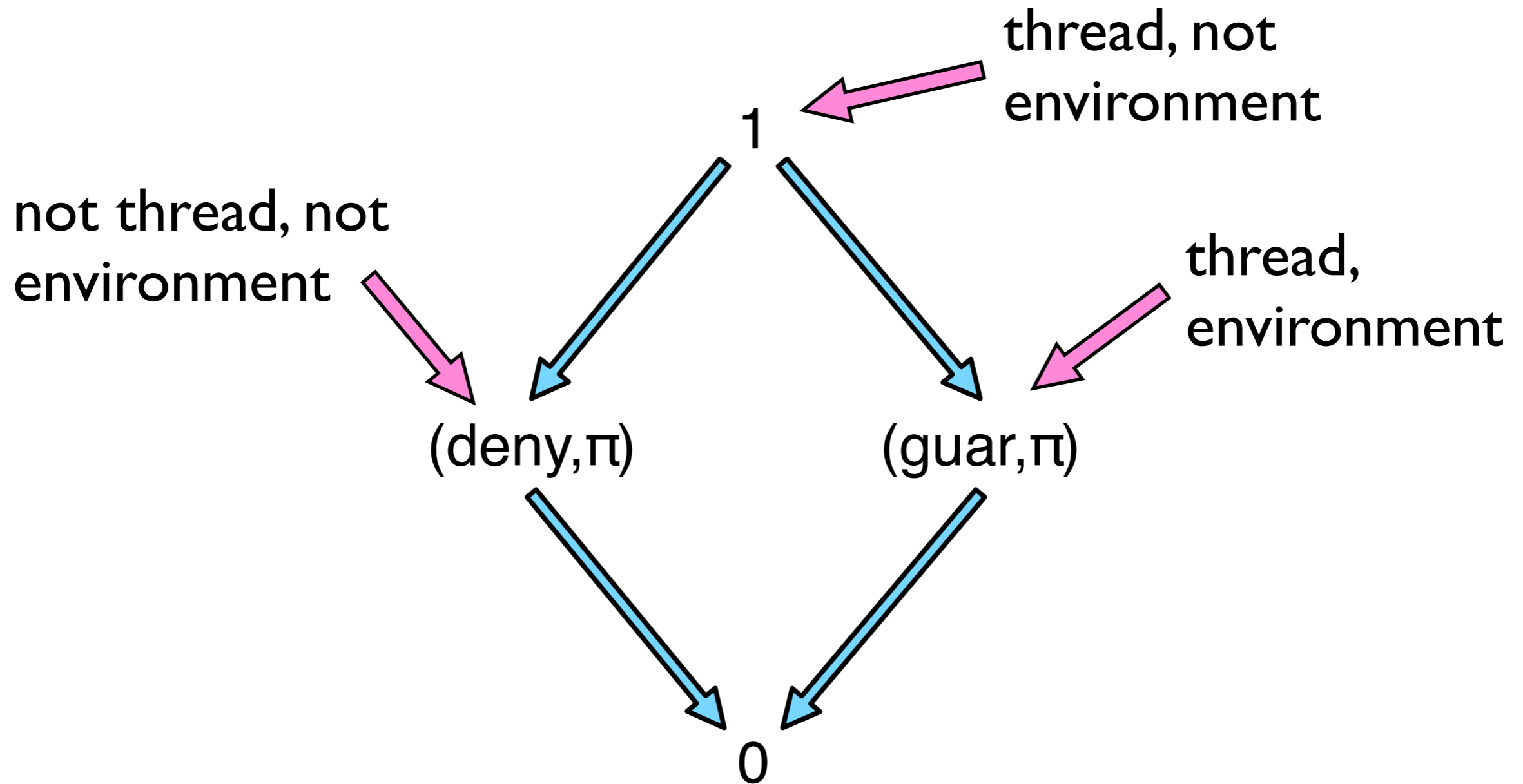
Meaning of permission



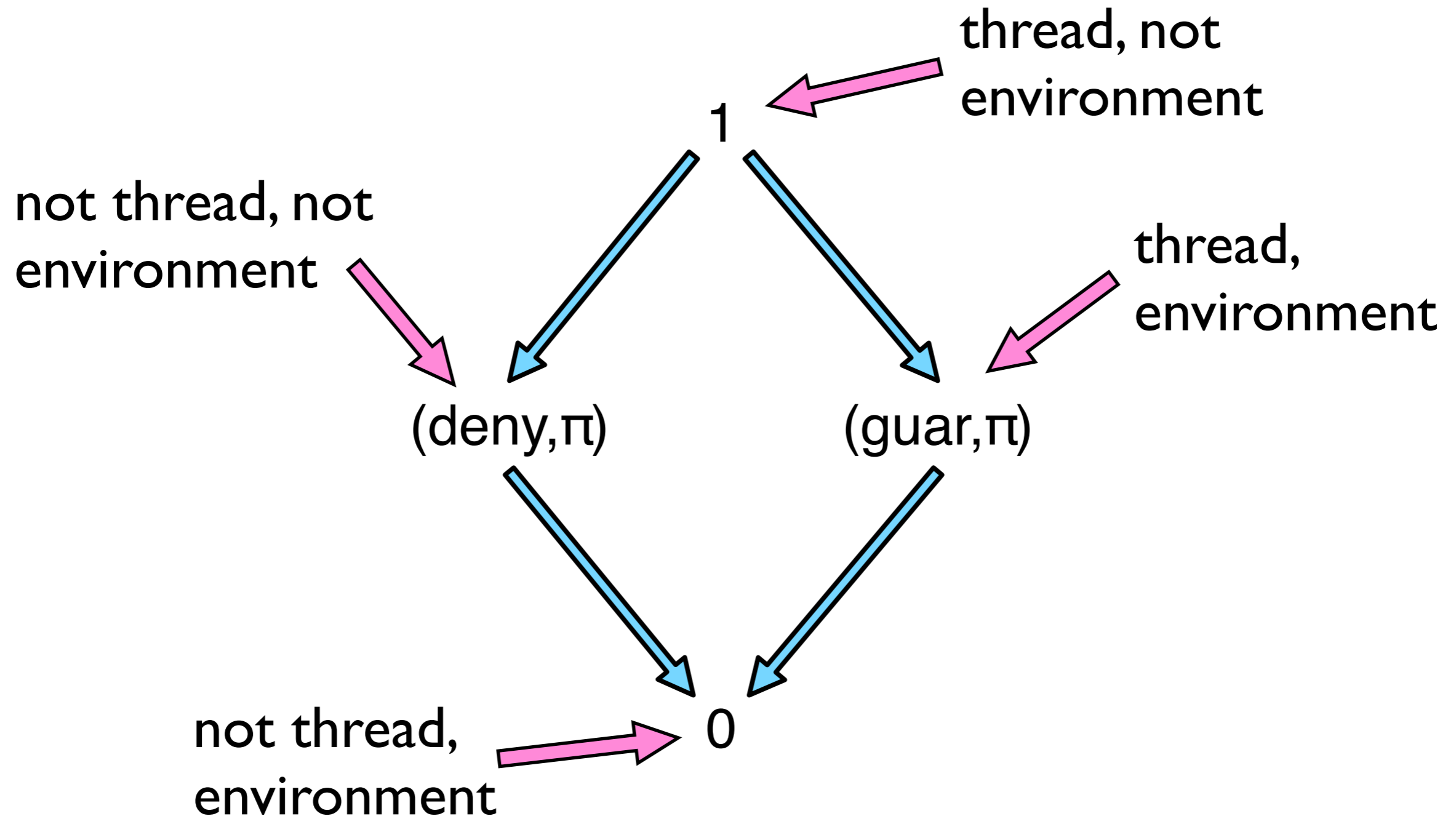
Meaning of permission



Meaning of permission



Meaning of permission



Combining permissions

Join permissions by addition

$$0 + p = p \qquad \qquad \qquad 1 + 0 = 1$$

$$(guar, k) + (guar, k') = (guar, k + k')$$

$$(deny, k) + (deny, k') = (deny, k + k')$$

Define the star for permissions based on this