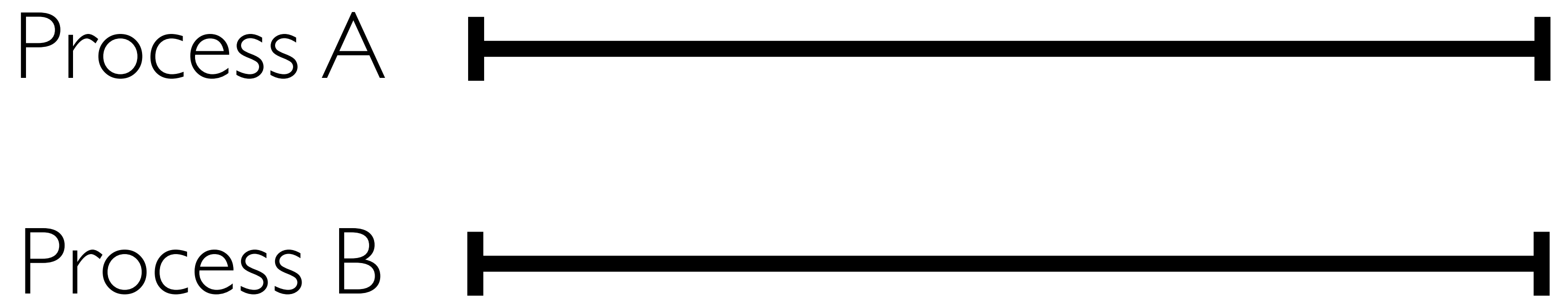
Logical relations for fine-grained concurrency

Aaron Turon

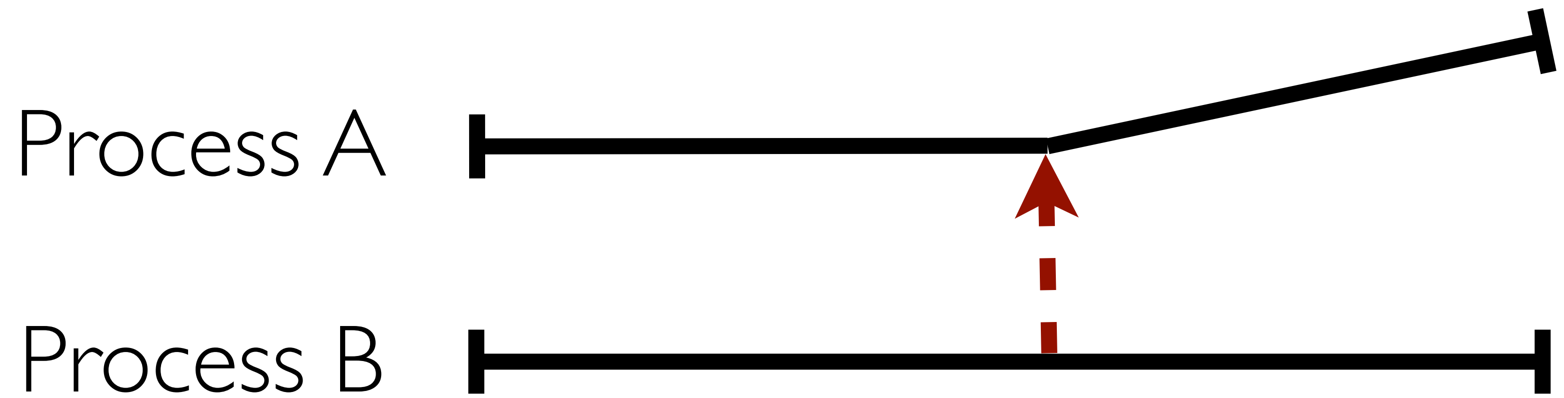
with

Jacob Thamsborg, Amal Ahmed,
Lars Birkedal, Derek Dreyer

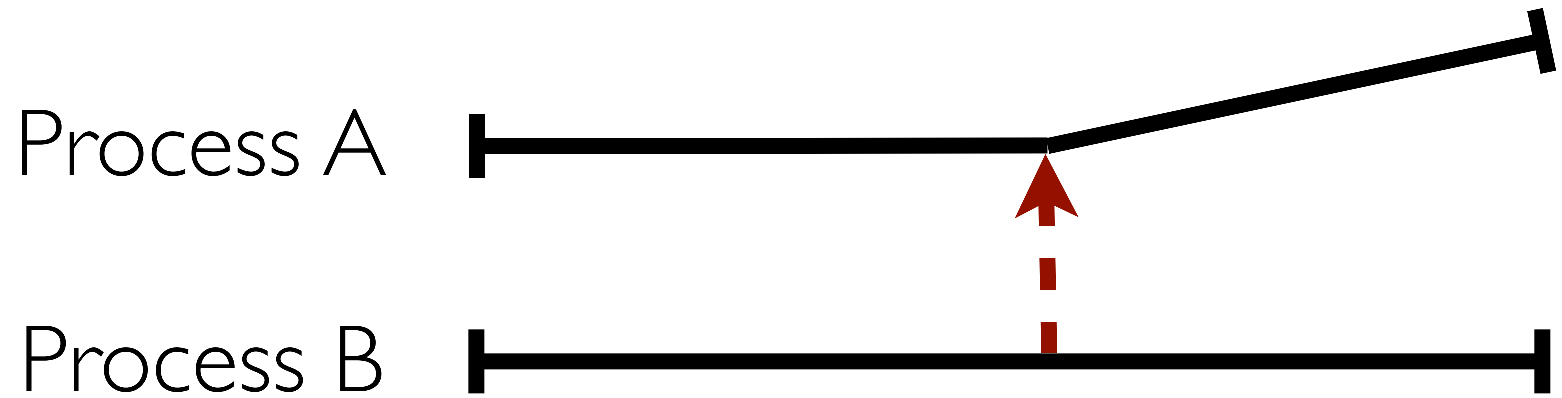
Concurrency is overlapped execution:



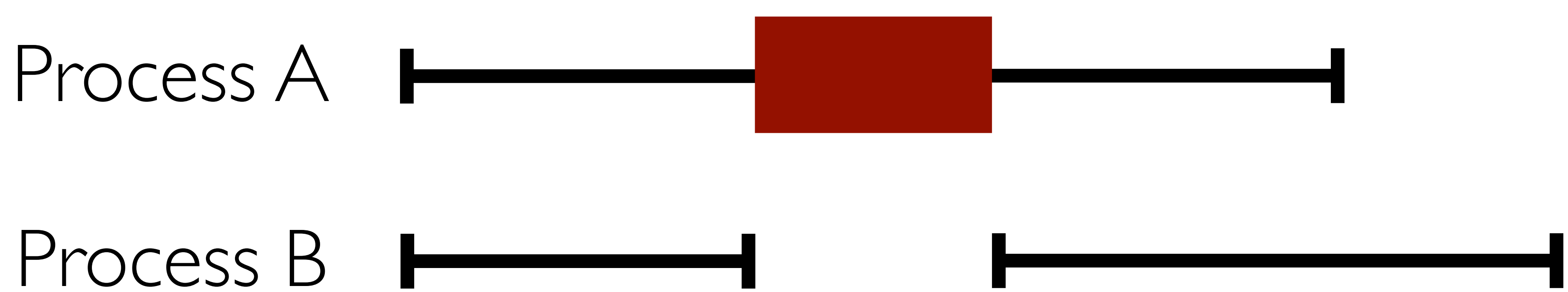
Concurrency is overlapped execution:



Concurrency is overlapped execution:

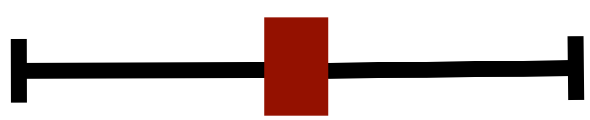


Atomicity is its antidote:



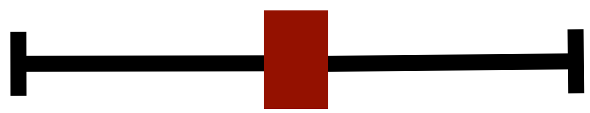
Granularity (in time & space)

Coarse-grained atomicity:
easier reasoning



Fine-grained atomicity:
lower latency
higher throughput (if parallel)

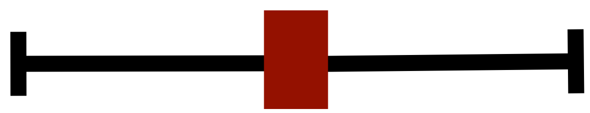
Atomicity abstraction



Fine-grained
implementation

Coarse-grained
specification

Atomicity abstraction



Fine-grained implementation

$\leq$

Coarse-grained specification

Refines, i.e., contextually approximates

Contributions

Direct, insightful refinement method
(no linearizability!)

Handles *higher-order, polymorphic* languages

Scales to *sophisticated* algorithms

Key Idea

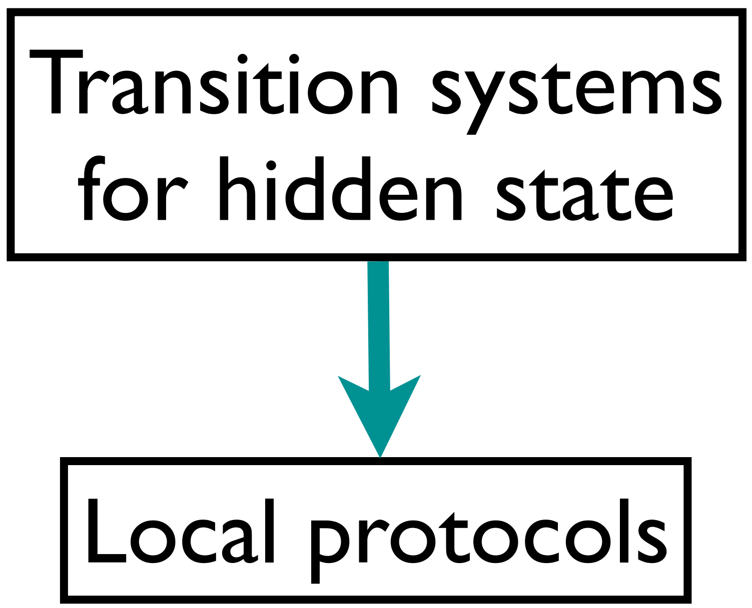
Atomicity abstraction, like data abstraction, relies on hiding

Approach

Transition systems for hidden state

State of the art: Kripke Logical Relations, ICFP10

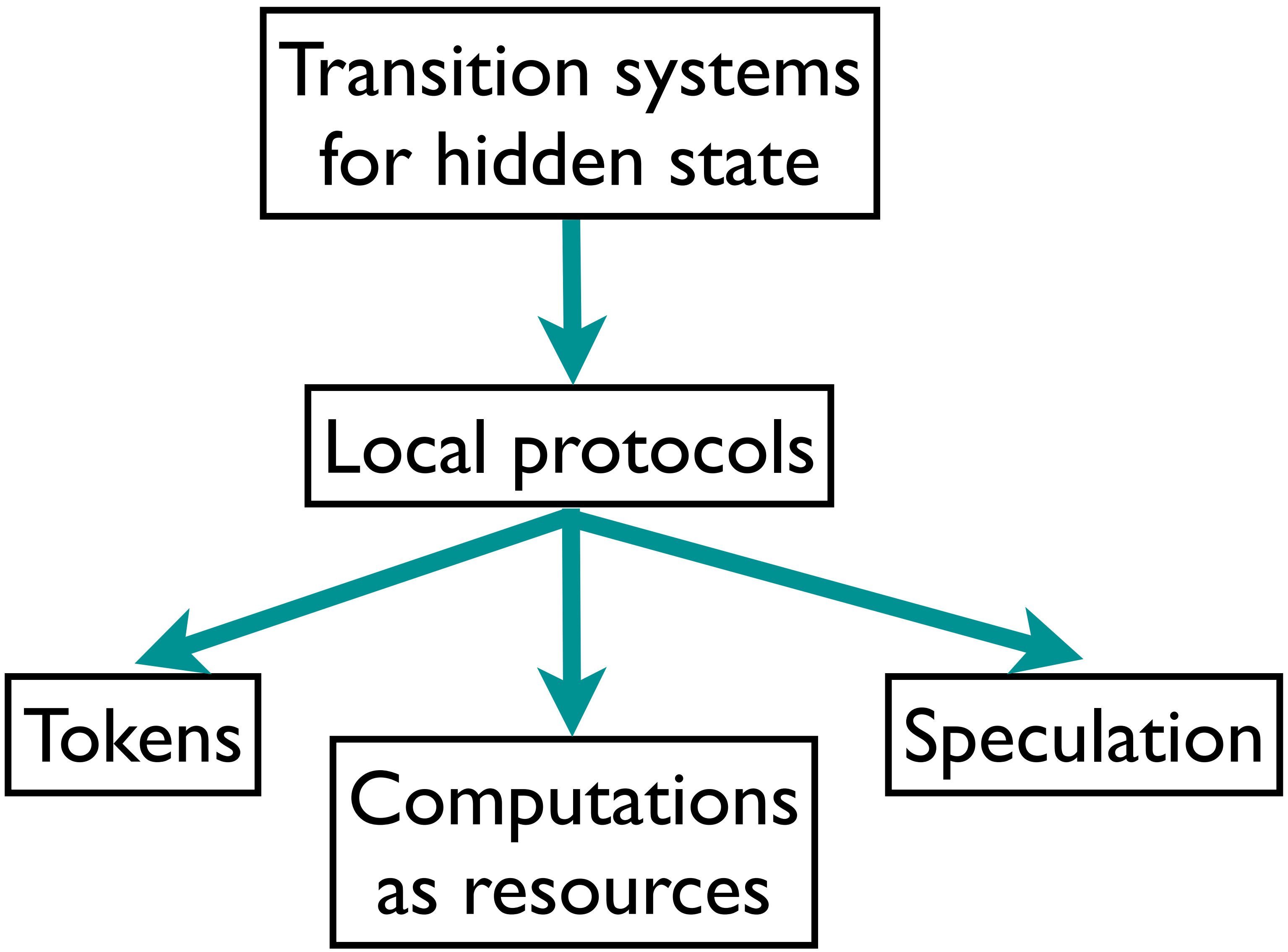
Approach



State of the art: Kripke Logical Relations, ICFP10

Fine granularity

Approach

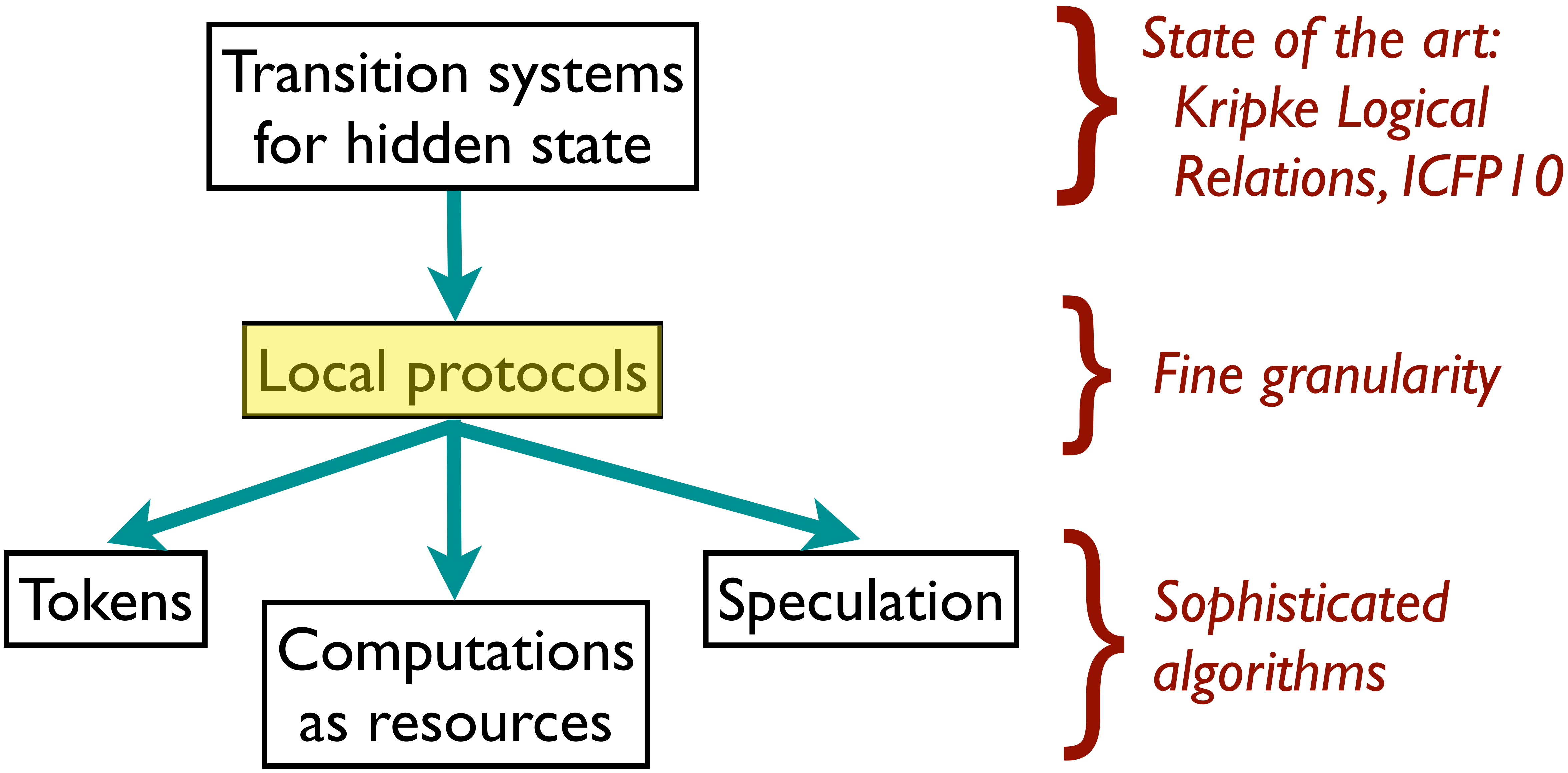


State of the art: Kripke Logical Relations, ICFP10

Fine granularity

Sophisticated algorithms

Approach

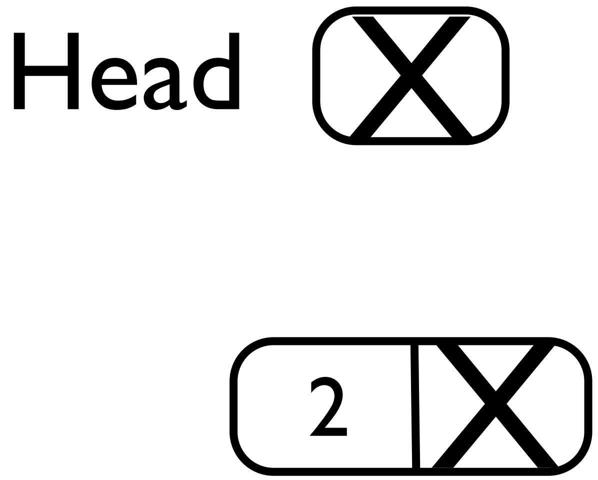

Transition systems for hidden state
Local protocols
Tokens
Computations as resources
Speculation
State of the art: Kripke Logical Relations, ICFP10
Fine granularity
Sophisticated algorithms

Compare and set

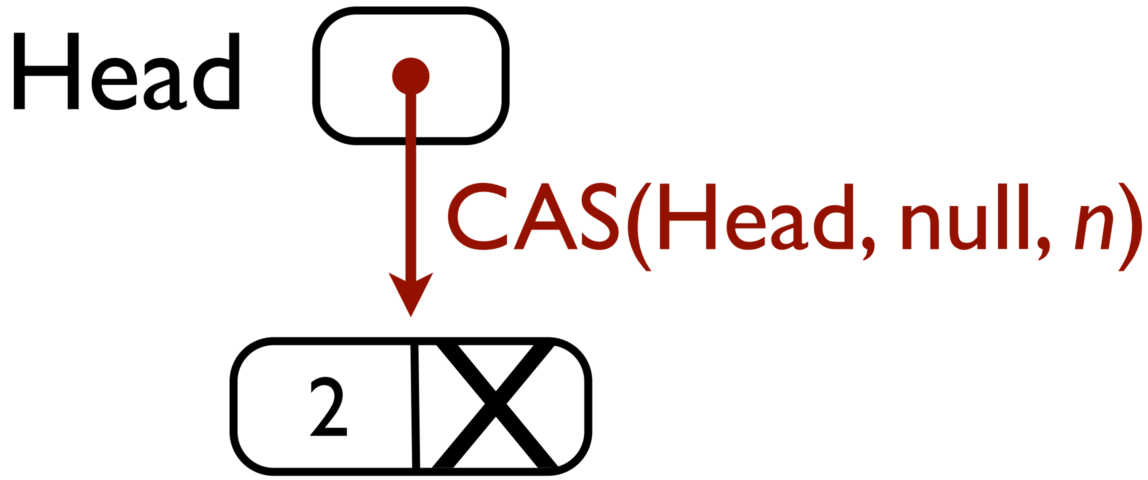
```
cas(cell, old, new) = atomic {
    if !cell := old
    then cell := new; true
    else false
}
```

A Lock-free "Queue"

A Lock-free “Queue”

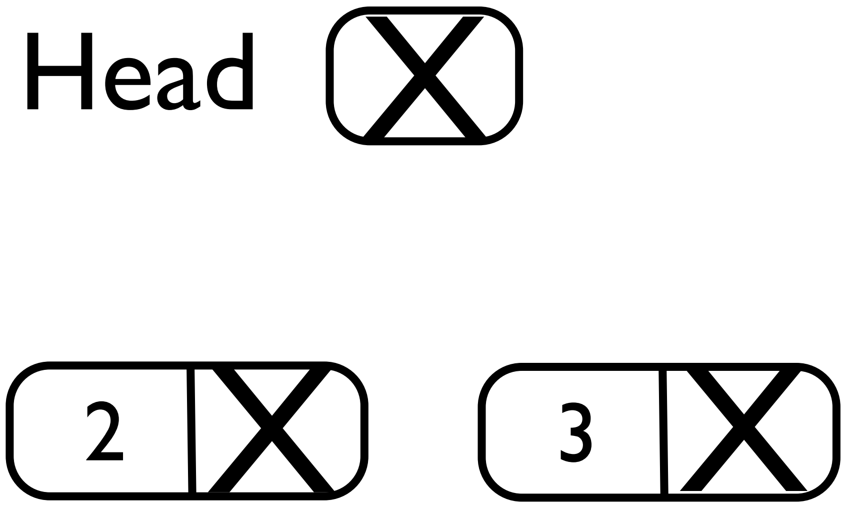


A Lock-free "Queue"

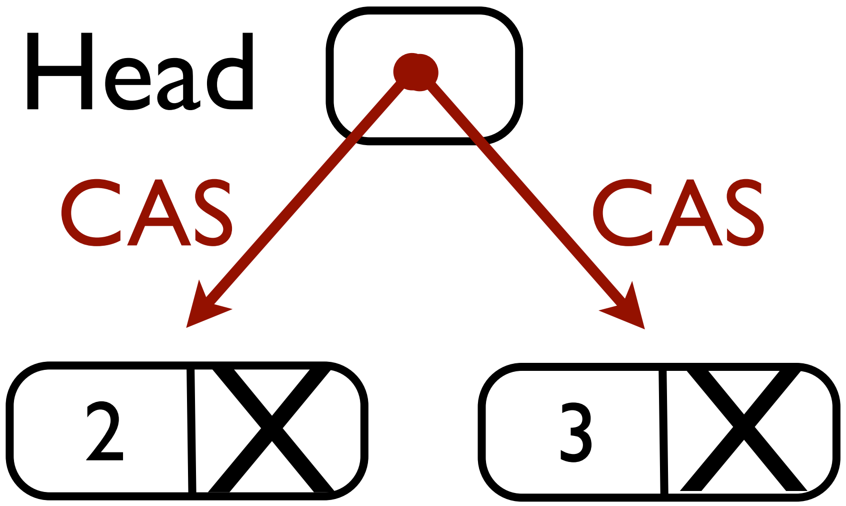


A Lock-free “Queue”

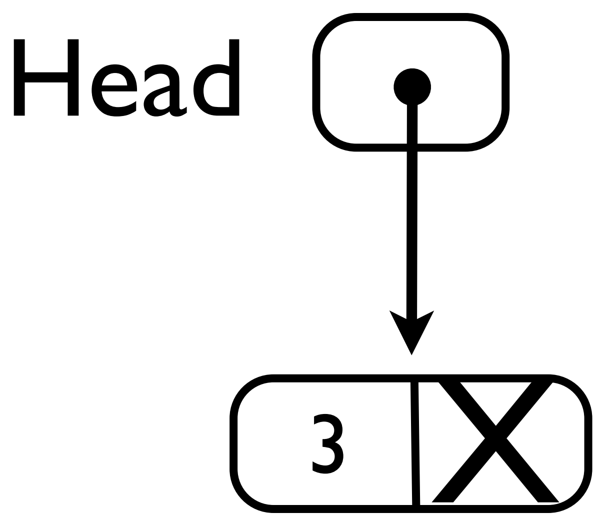
A Lock-free "Queue"



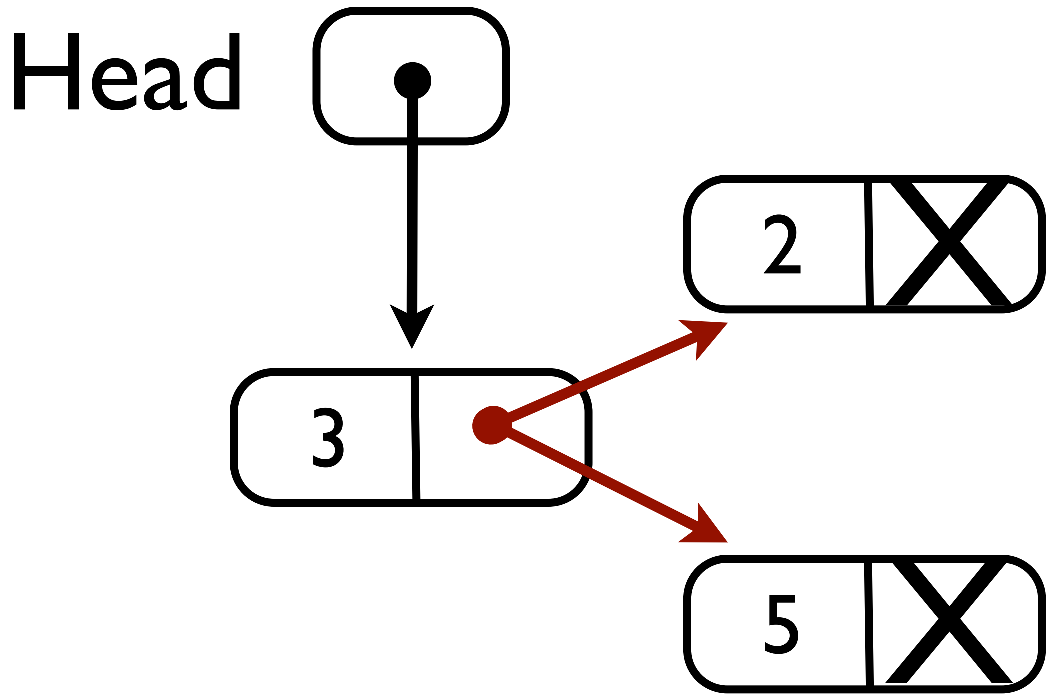
A Lock-free "Queue"



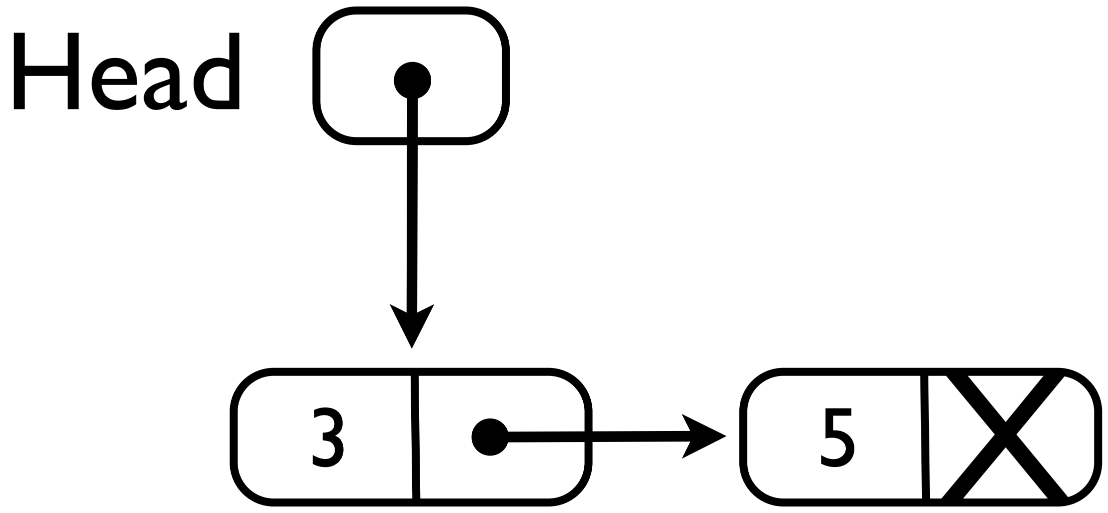
A Lock-free “Queue”



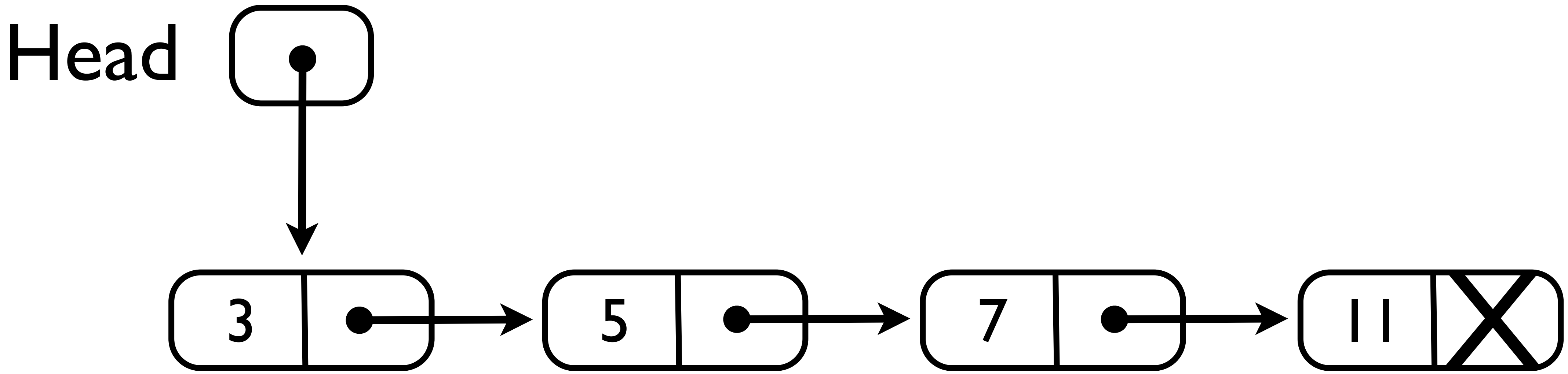
A Lock-free "Queue"



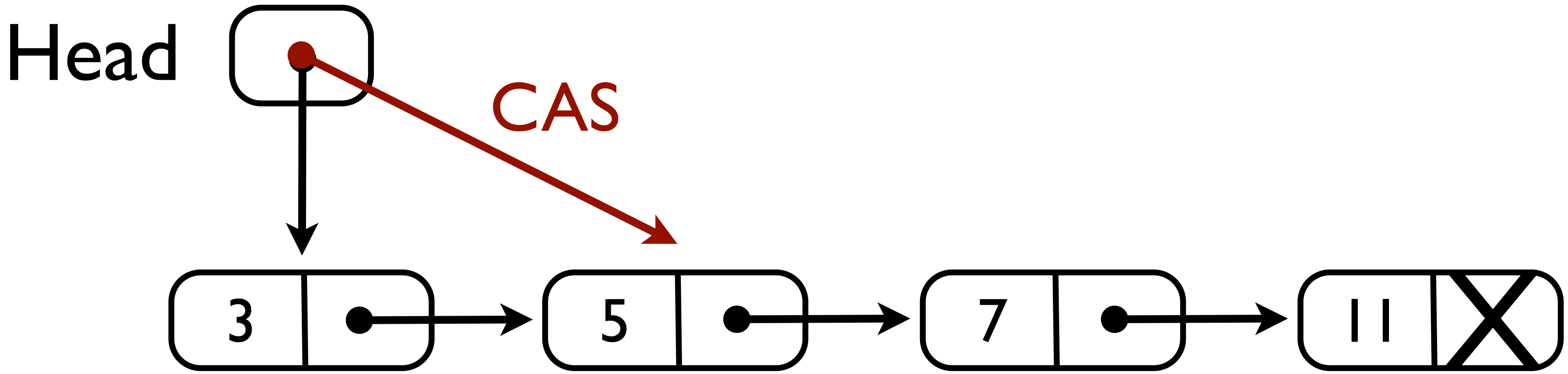
A Lock-free "Queue"



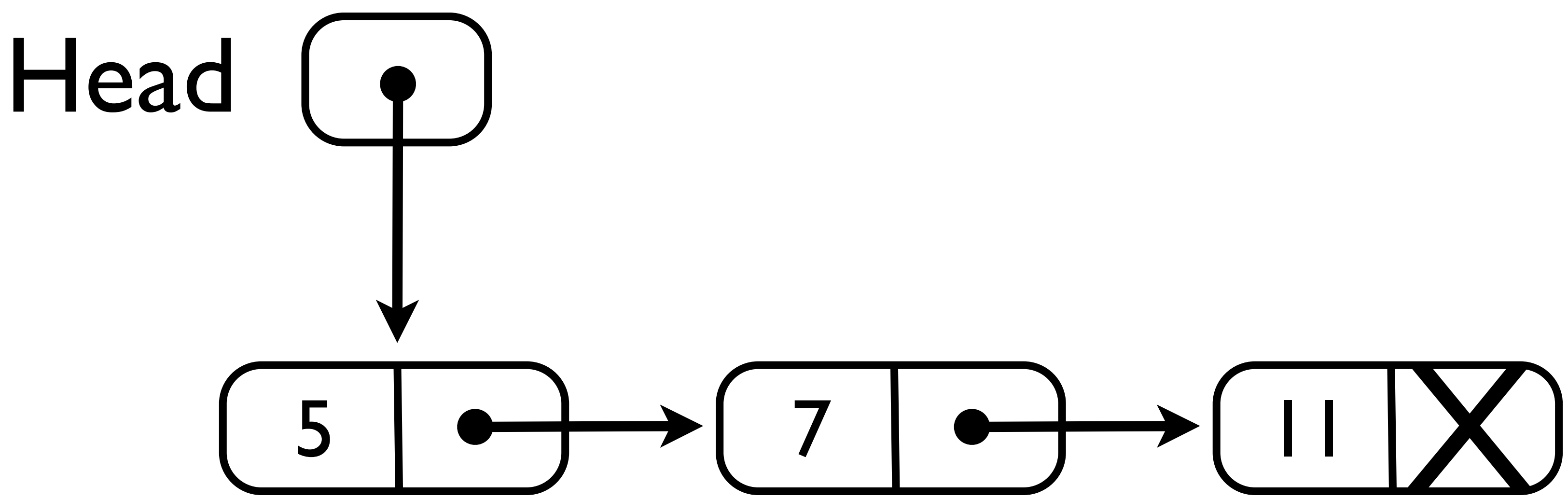
A Lock-free "Queue"



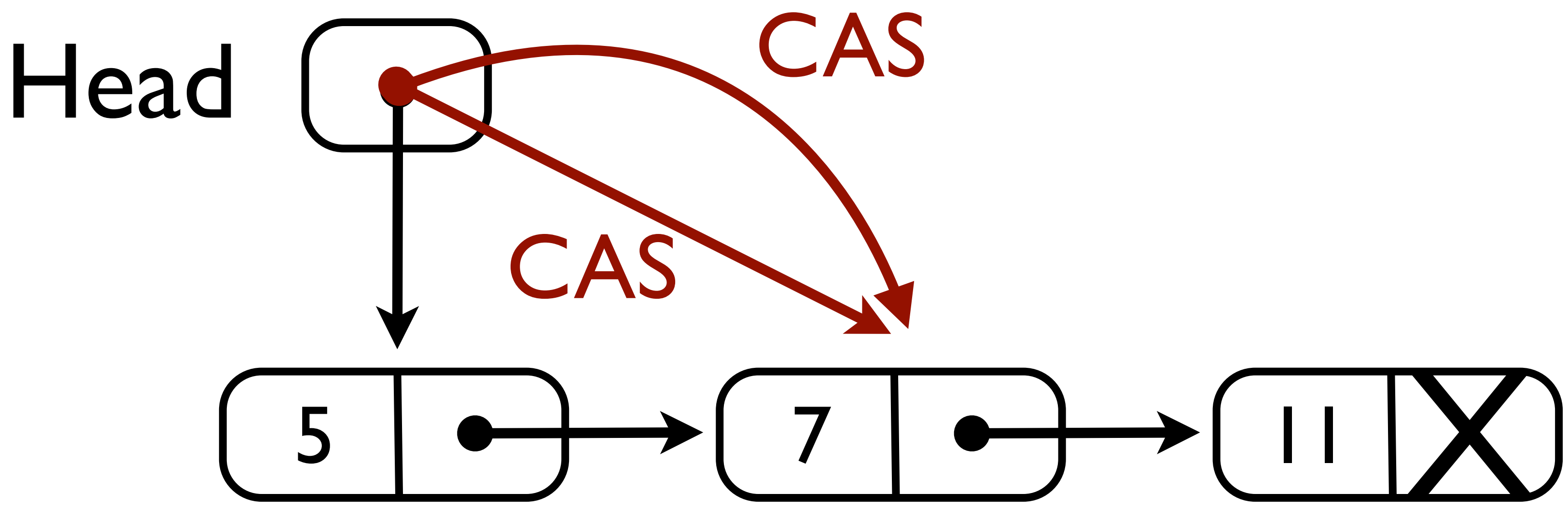
A Lock-free "Queue"



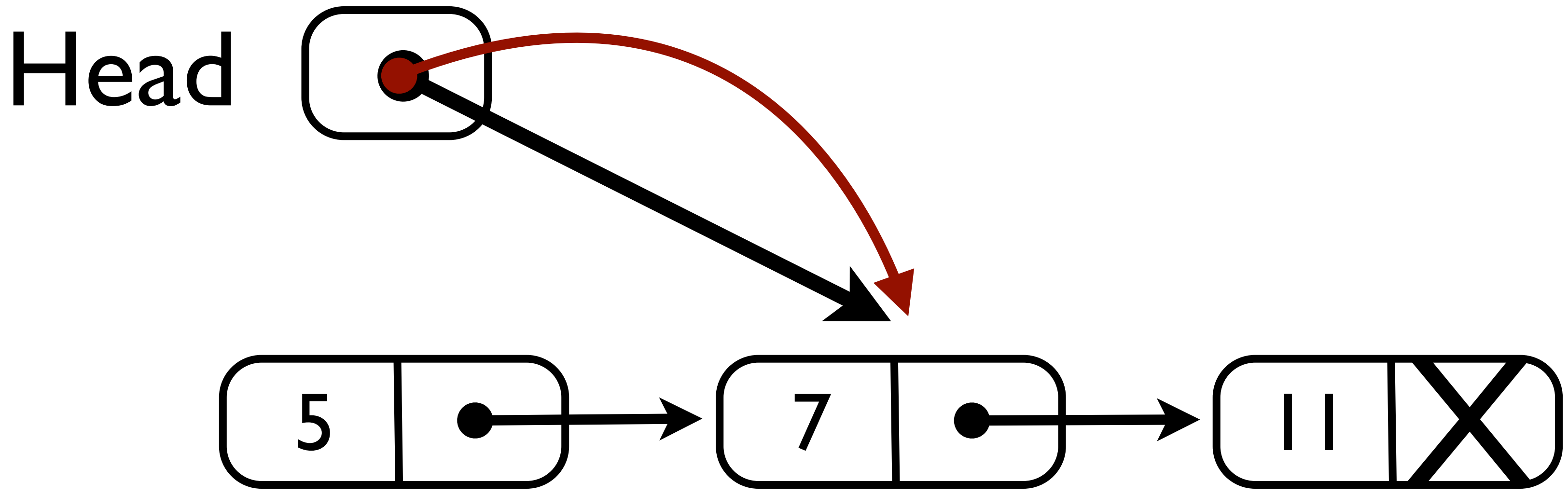
A Lock-free “Queue”



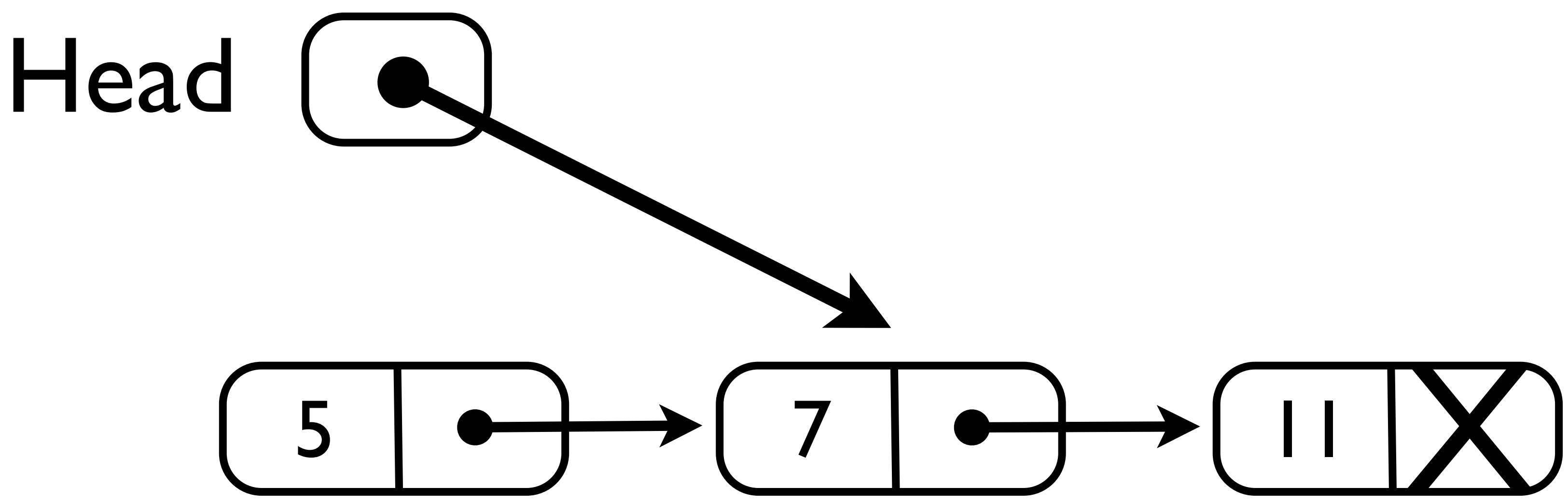
A Lock-free “Queue”



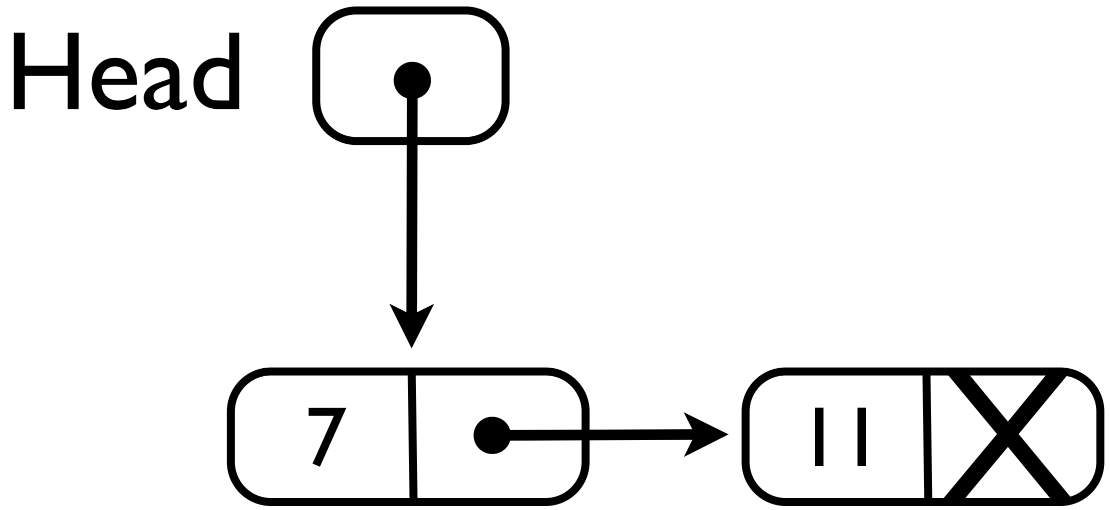
A Lock-free "Queue"



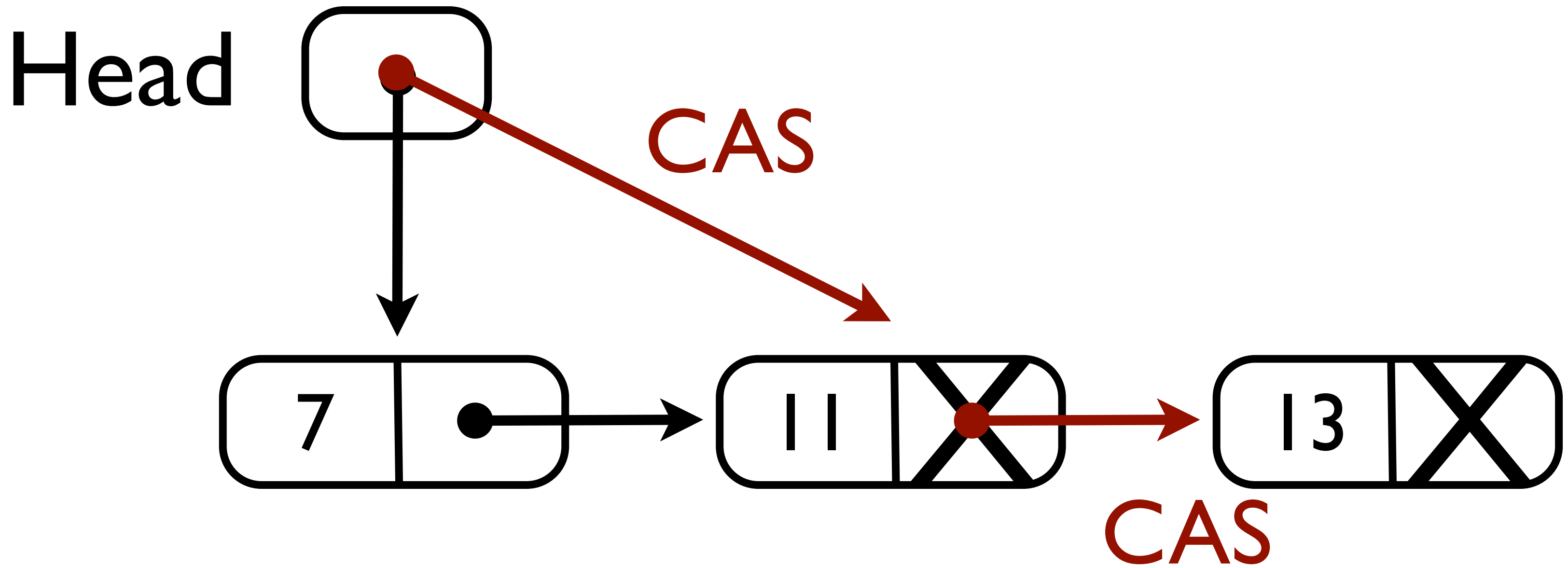
A Lock-free “Queue”



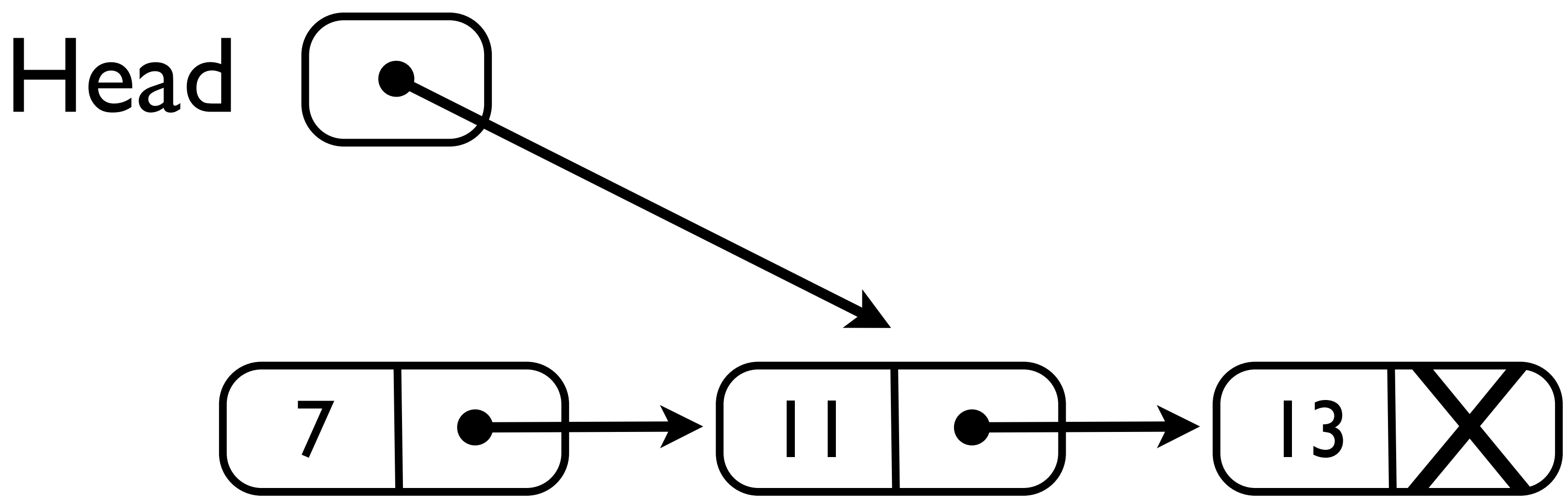
A Lock-free “Queue”



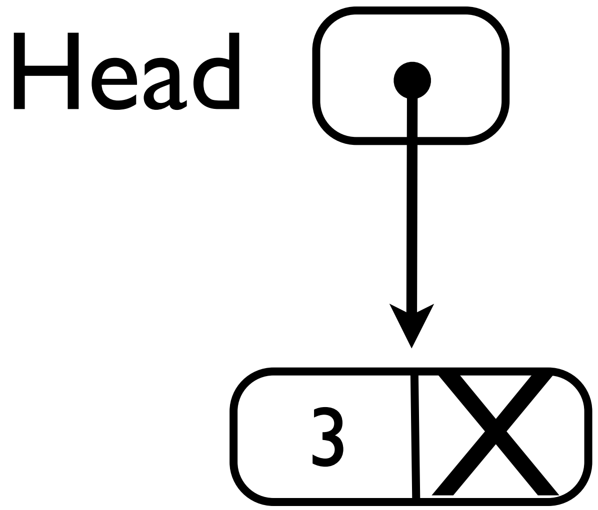
A Lock-free "Queue"



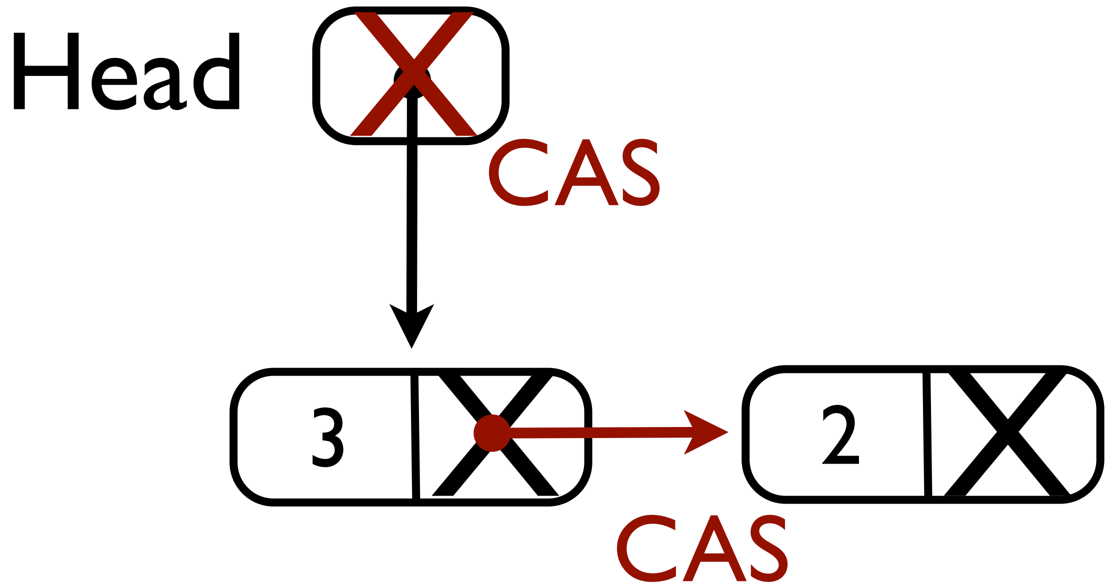
A Lock-free "Queue"



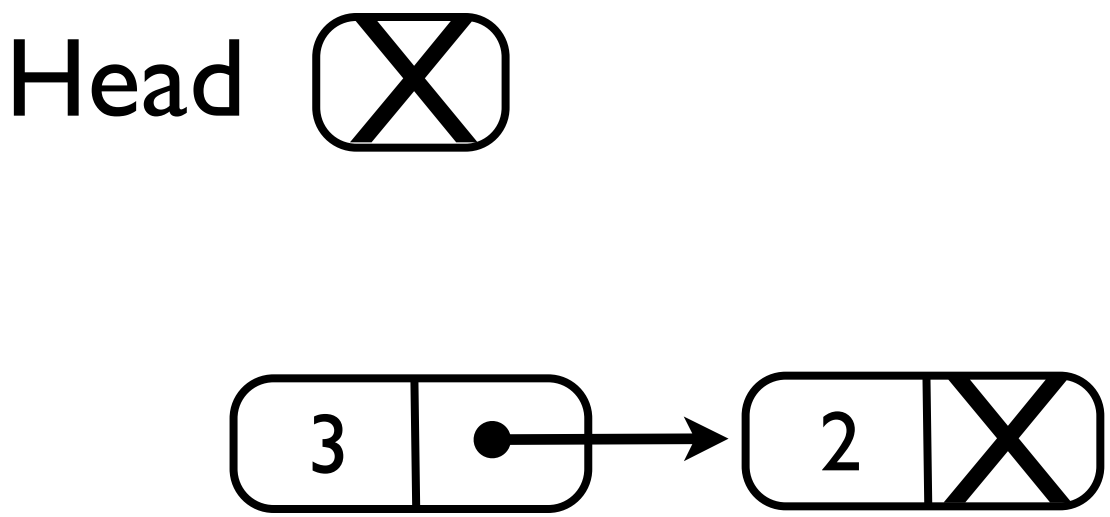
A Lock-free "Queue"



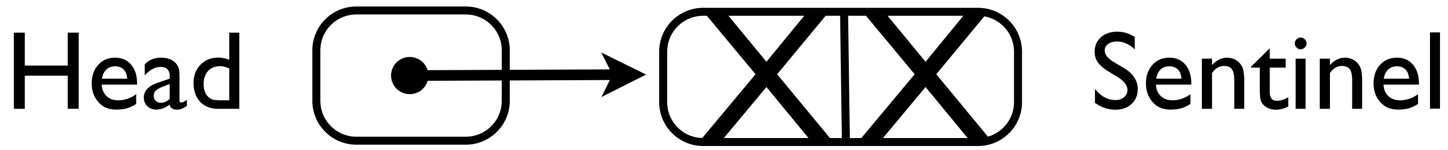
A Lock-free “Queue”



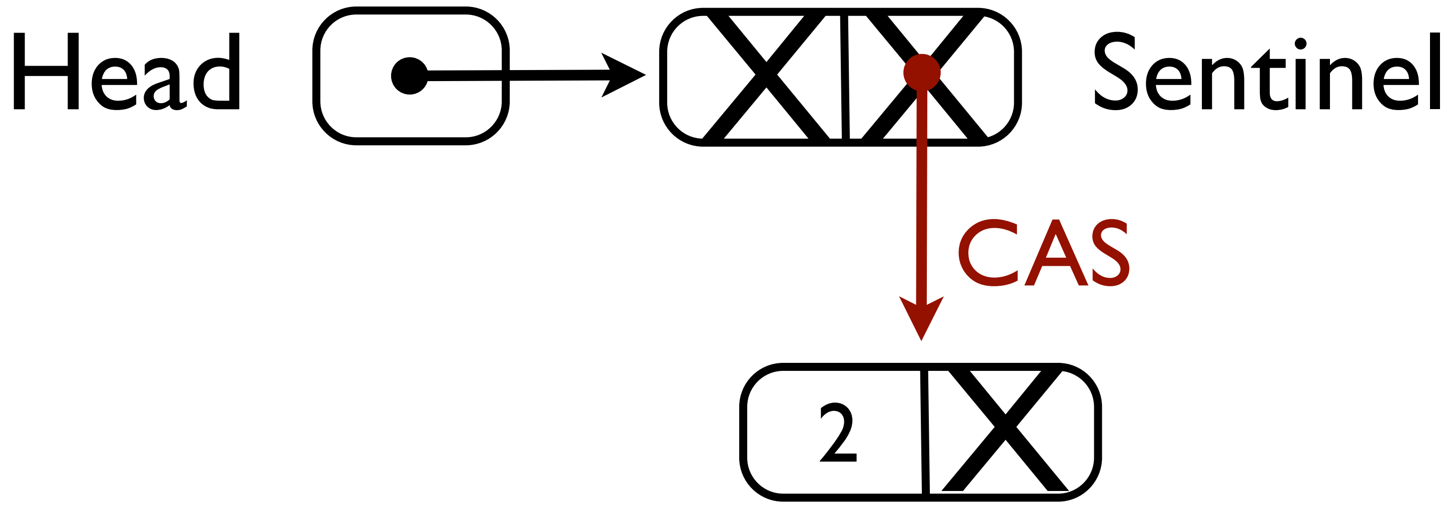
A Lock-free “Queue”



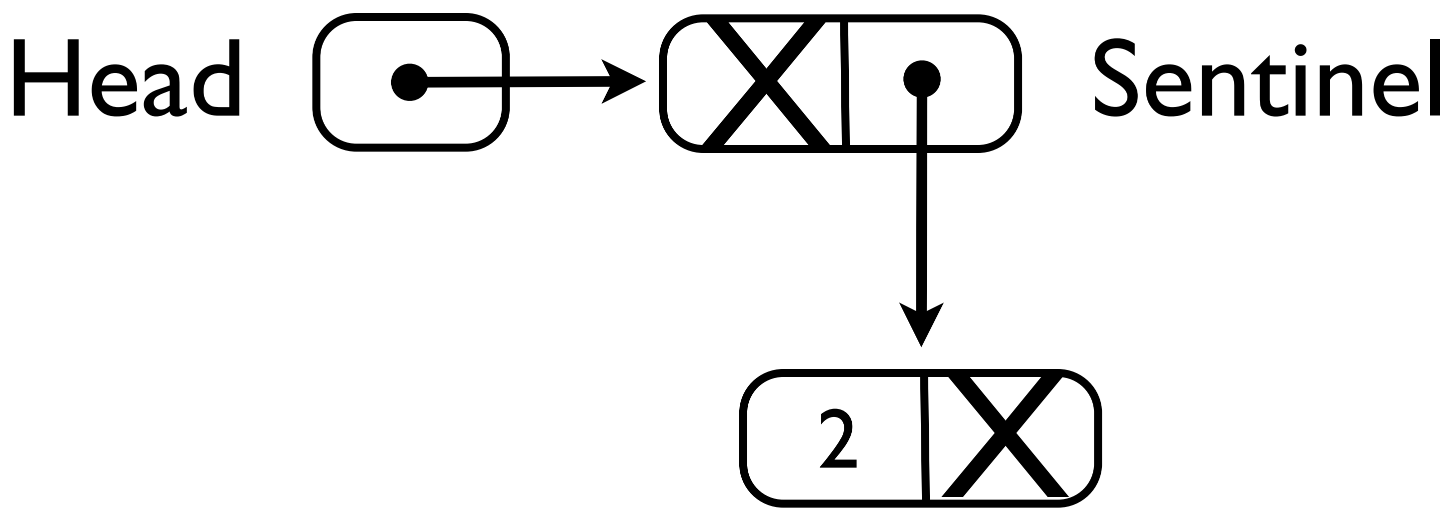
The Michael-Scott Queue



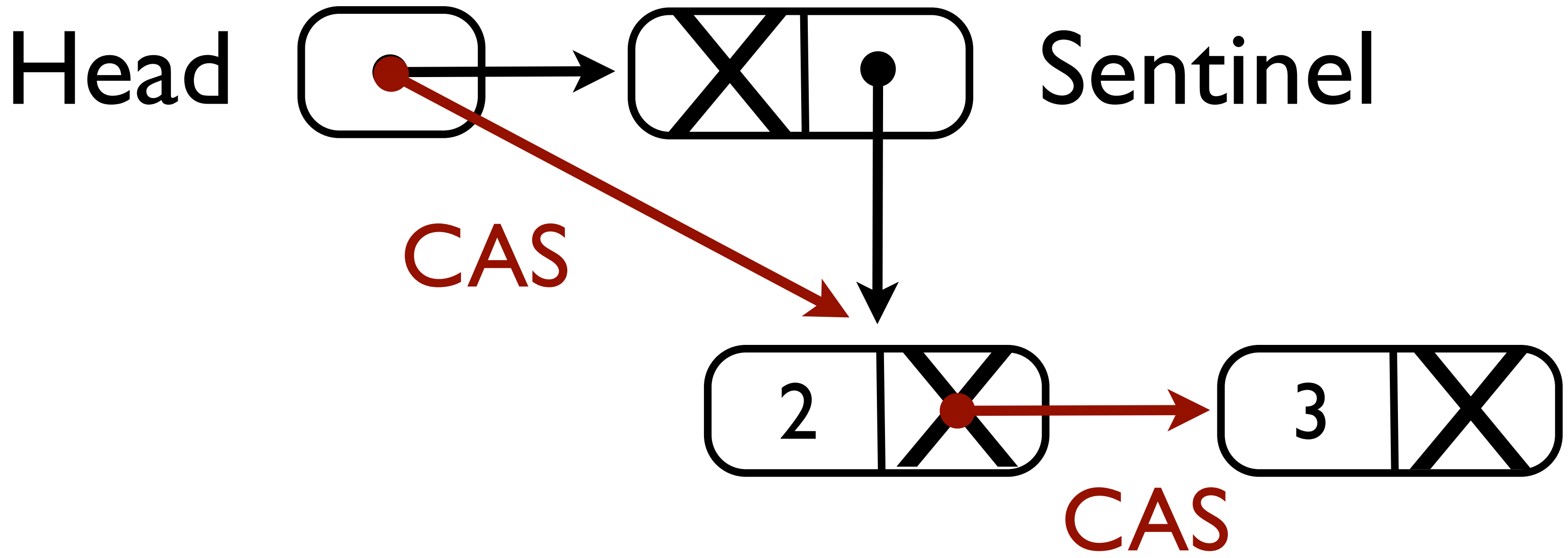
The Michael-Scott Queue



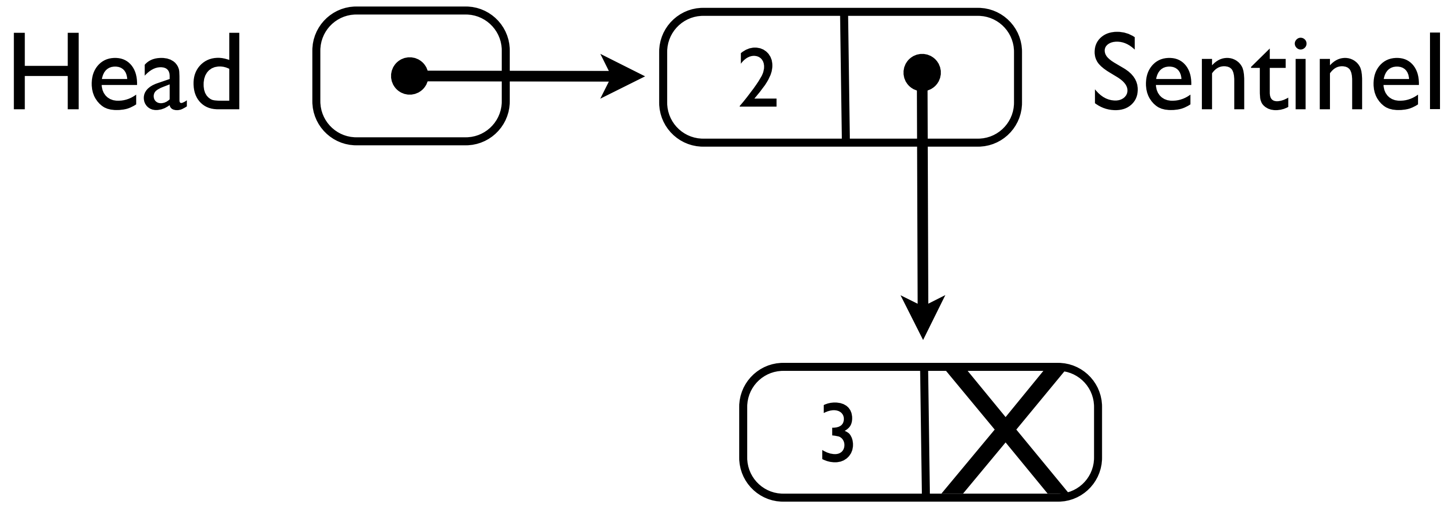
The Michael-Scott Queue



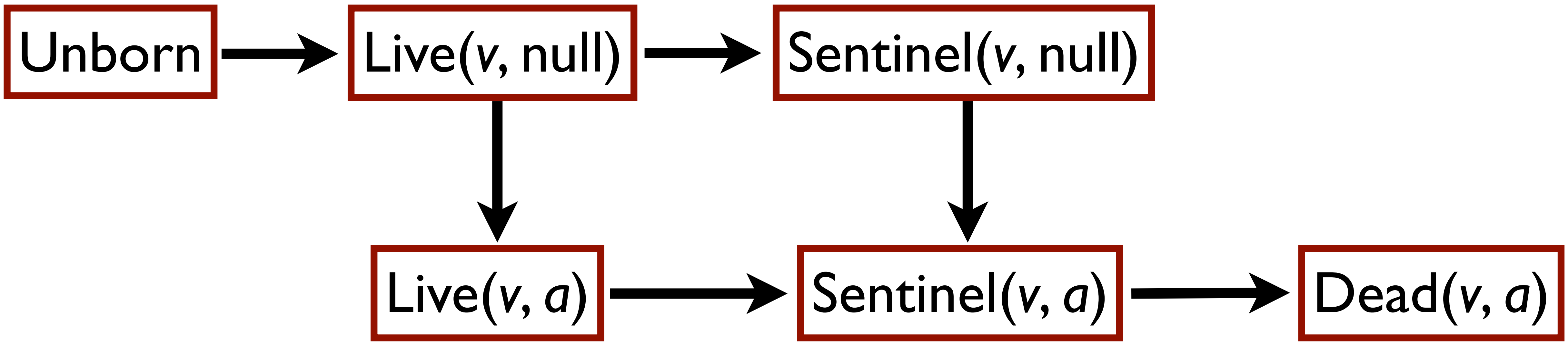
The Michael-Scott Queue



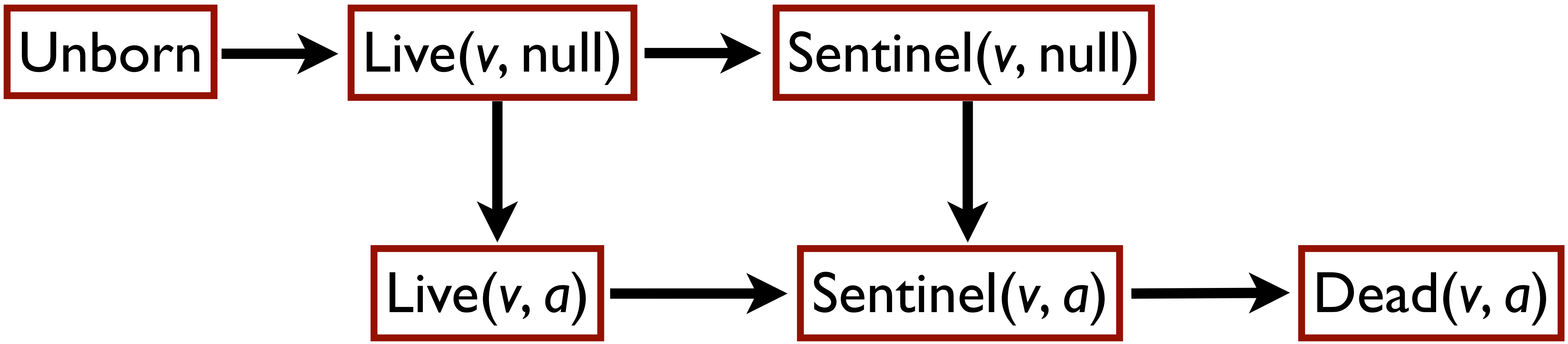
The Michael-Scott Queue



The life story of a node



The life story of a node



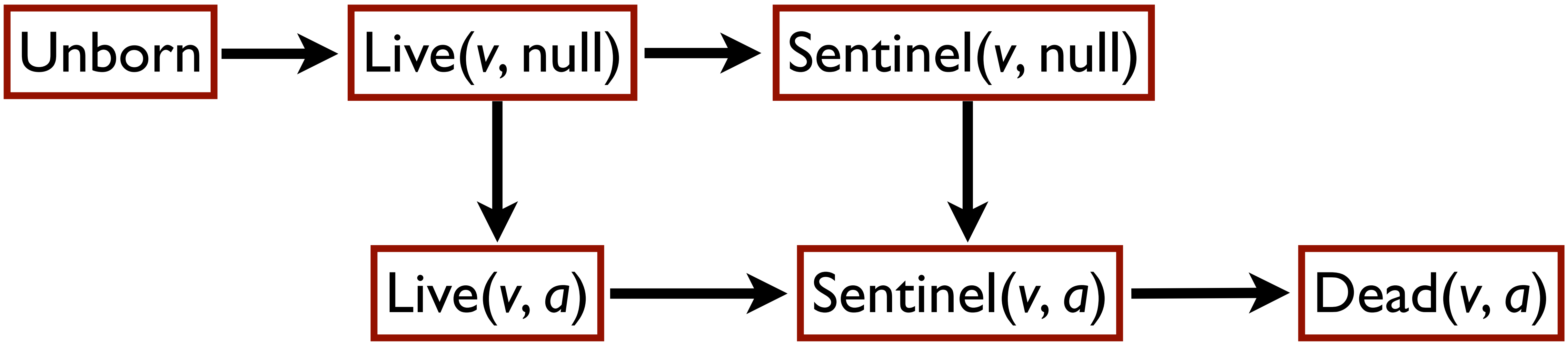
Global constraints:

Exactly one sentinel

Live iff reachable from sentinel

Nonnull next pointers are born

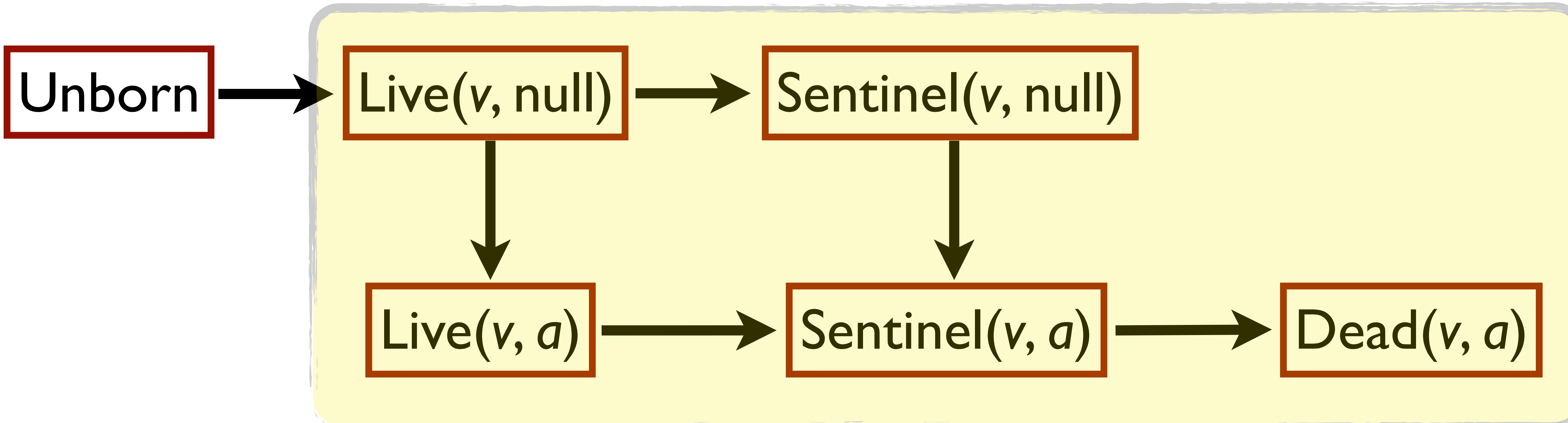
The life story of a node



Global constraints:
Exactly one sentinel
Live iff reachable from sentinel
Nonnull next pointers are born

```
linkNode(p, n) = case !(p.next)
  of null ⇒
        if cas(p.next, null, n)
        then () else linkNode(p, n)
  | p’ ⇒ linkNode(p’, n)
```

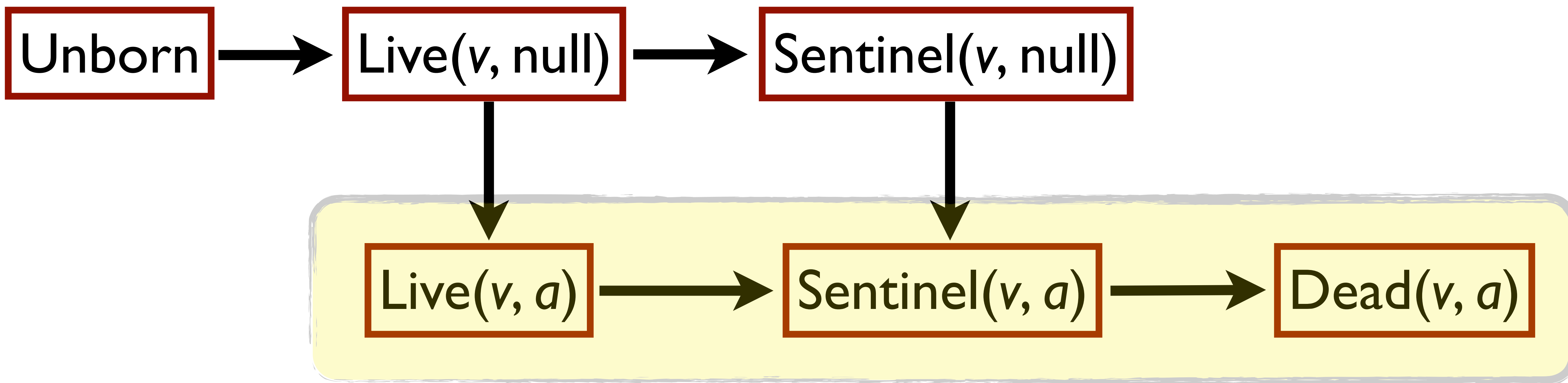
The life story of a node



Global constraints:
Exactly one sentinel
Live iff reachable from sentinel
Nonnull next pointers are born

```
linkNode(p, n) = case !(p.next)
  of null ⇒
      if cas(p.next, null, n)
      then () else linkNode(p, n)
  | p' ⇒ linkNode(p', n)
```

The life story of a node



Global constraints:

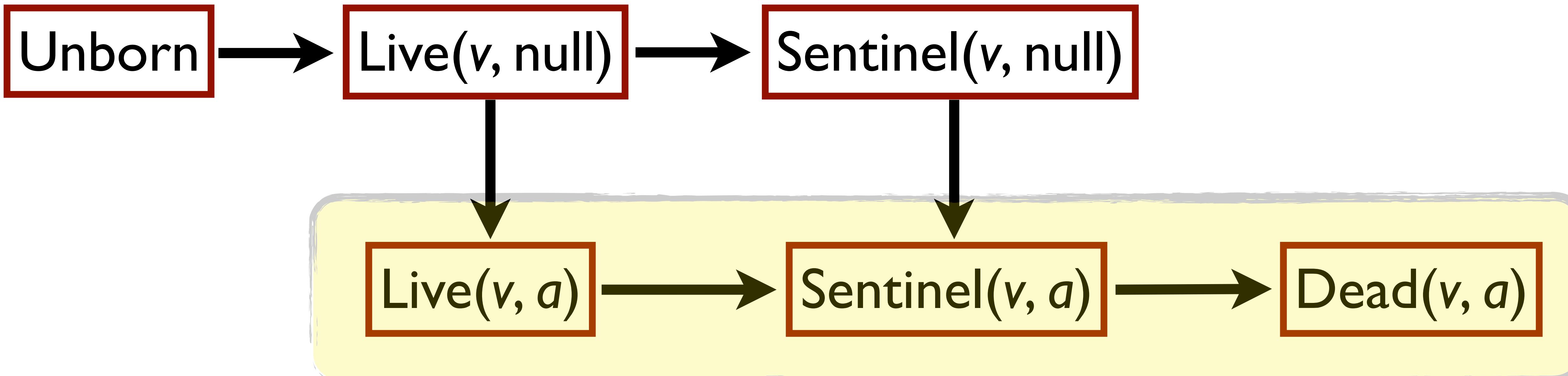
Exactly one sentinel

Live iff reachable from sentinel

Nonnull next pointers are born

```
linkNode(p, n) = case !(p.next)
  of null ⇒
        if cas(p.next, null, n)
        then () else linkNode(p, n)
  | p' ⇒ linkNode(p', n)
```

The life story of a node



Global constraints:

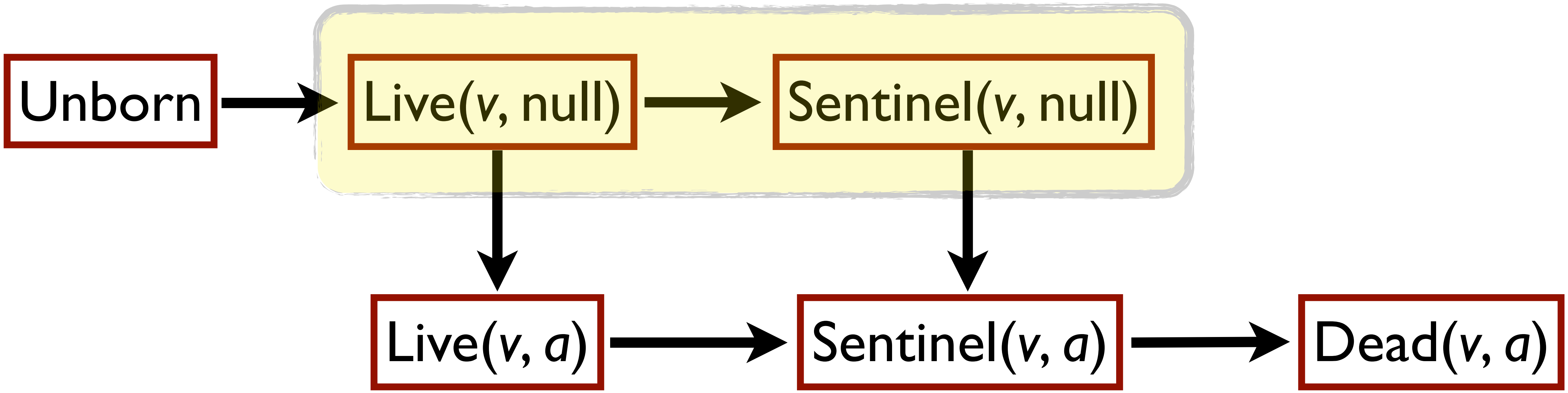
Exactly one sentinel

Live iff reachable from sentinel

Nonnull next pointers are born

```
linkNode(p, n) = case !(p.next)
  of null ⇒
       if cas(p.next, null, n)
       then () else linkNode(p, n)
   | p’ ⇒ linkNode(p’, n)
```

The life story of a node



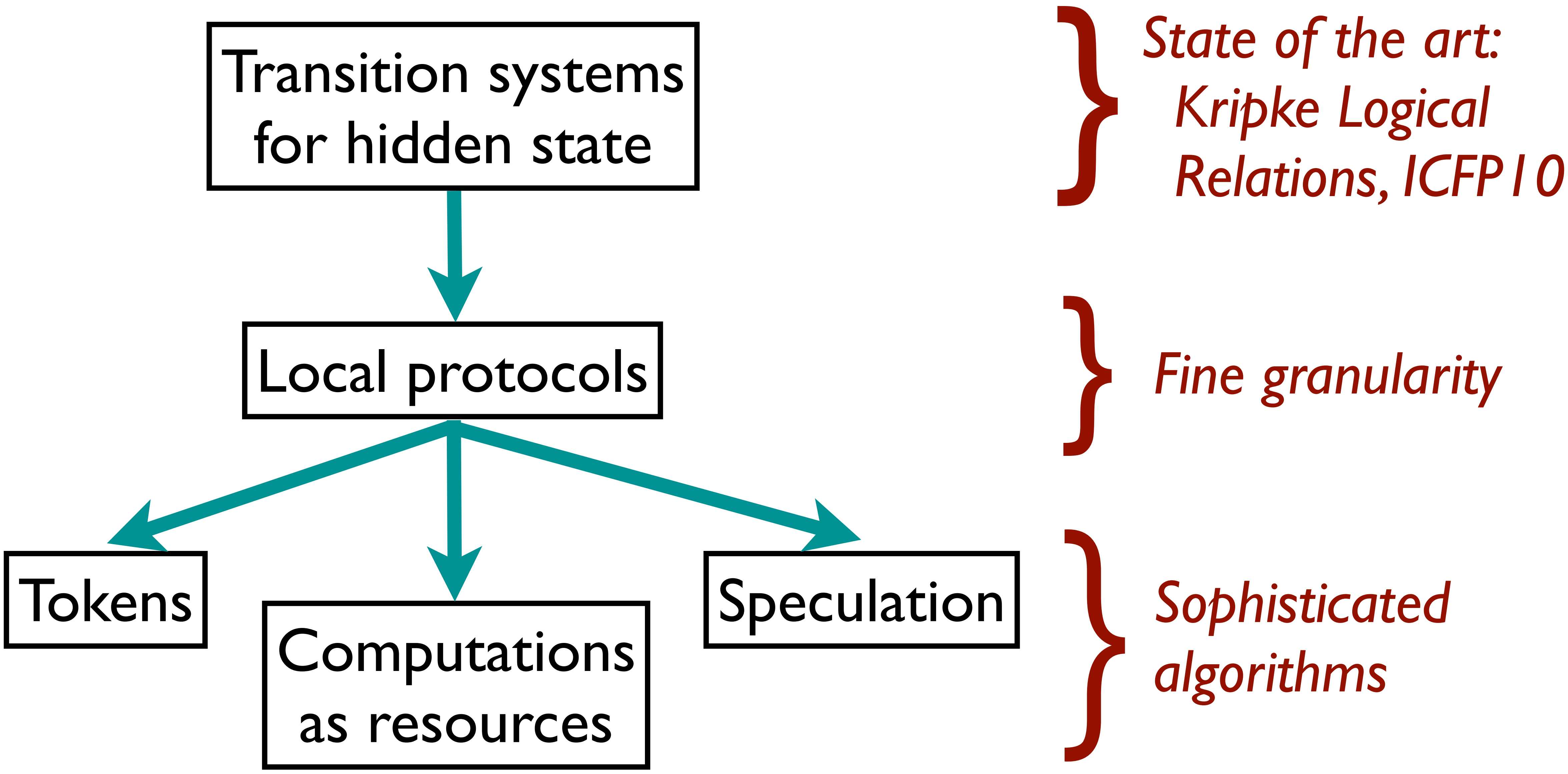
Global constraints:
Exactly one sentinel
Live iff reachable from sentinel
Nonnull next pointers are born

```
linkNode(p, n) = case !(p.next)
  of null ⇒
       if cas(p.next, null, n)
       then () else linkNode(p, n)
   | p' ⇒ linkNode(p', n)
```

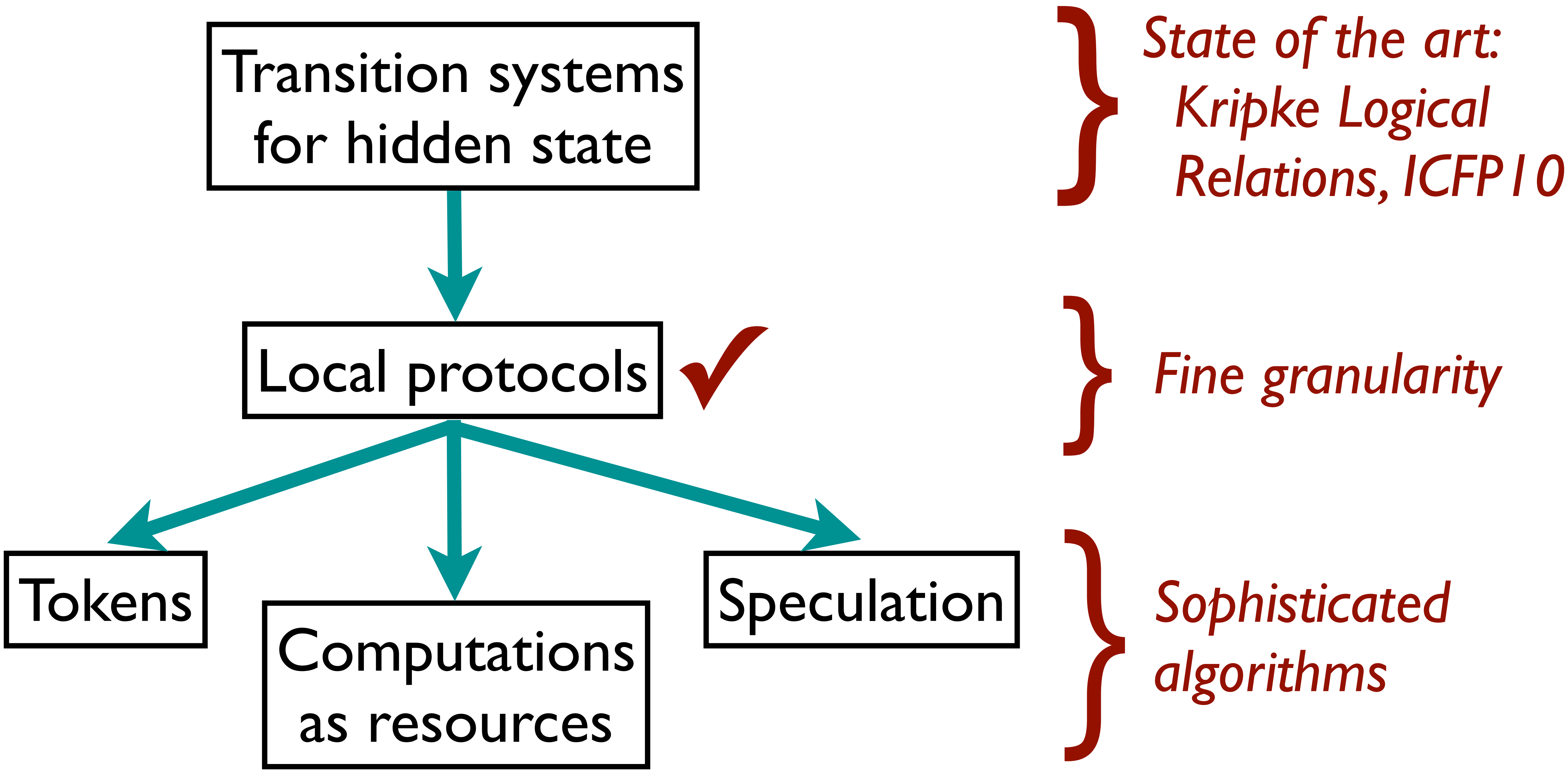
Key Idea

Local protocols capture abstract knowledge/interference at the same granularity as the algorithm operates

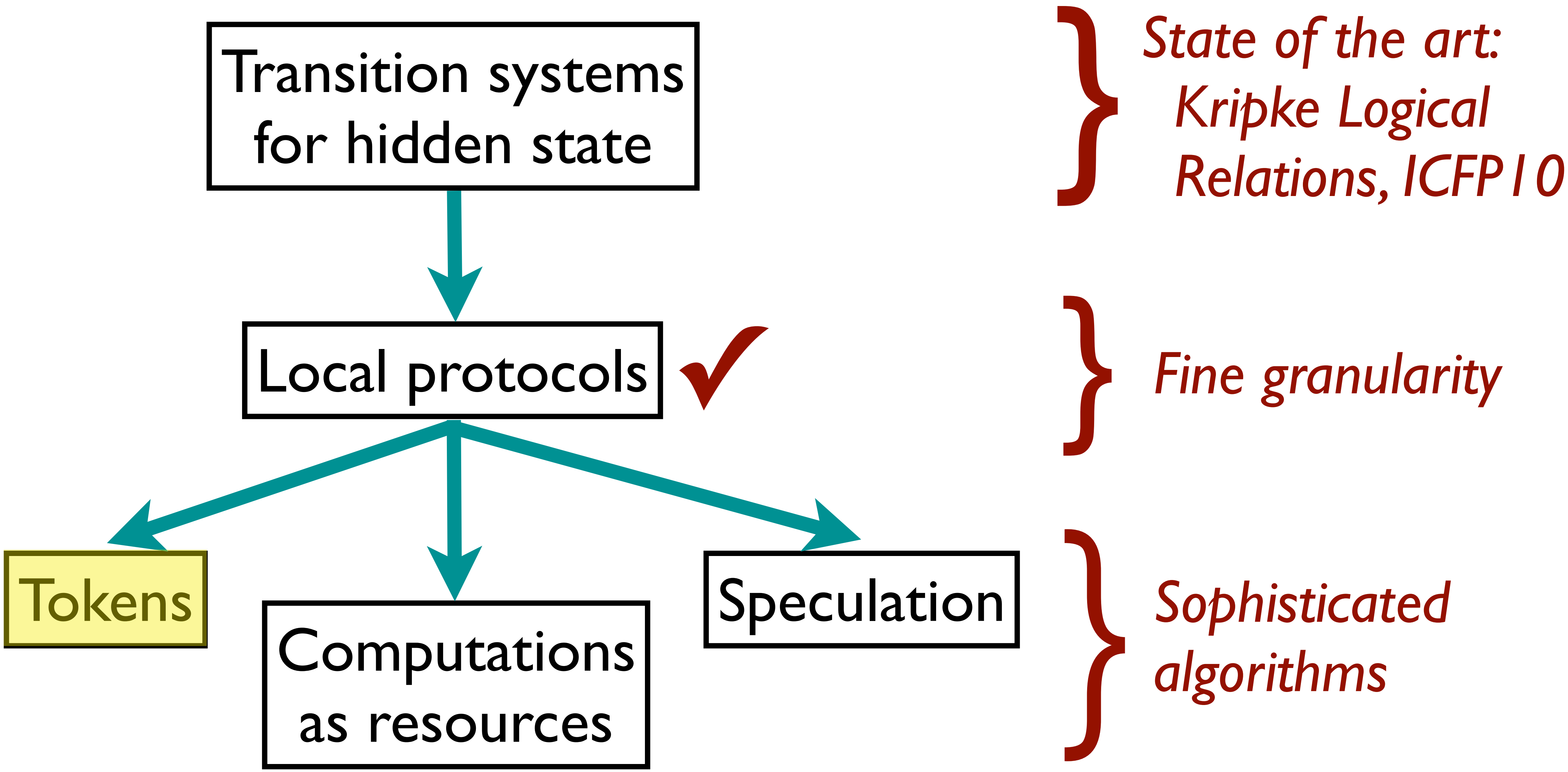
Approach

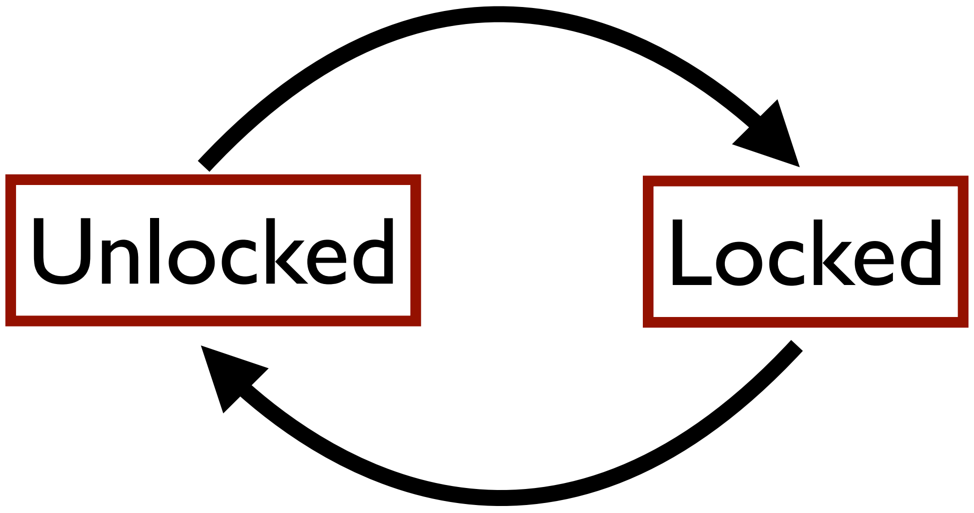

Transition systems for hidden state
Local protocols
Tokens
Computations as resources
Speculation
State of the art: Kripke Logical Relations, ICFP10
Fine granularity
Sophisticated algorithms

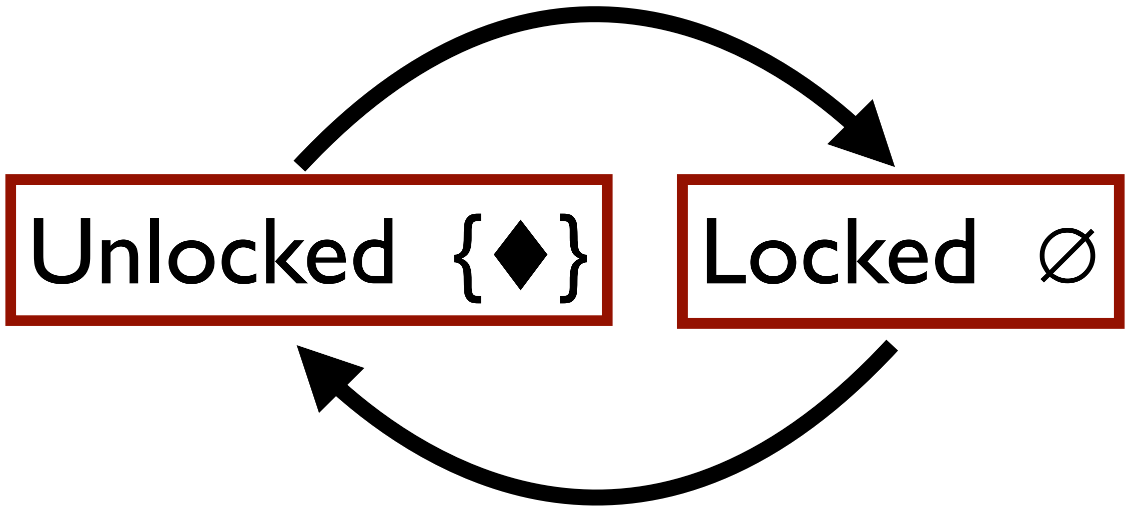
Approach


Transition systems for hidden state
Local protocols ✓
Tokens
Computations as resources
Speculation
State of the art: Kripke Logical Relations, ICFP10
Fine granularity
Sophisticated algorithms

Approach


Transition systems for hidden state
Local protocols ✓
Tokens
Computations as resources
Speculation
State of the art: Kripke Logical Relations, ICFP10
Fine granularity
Sophisticated algorithms


Unlocked
Locked


Unlocked {♦}
Locked ∅

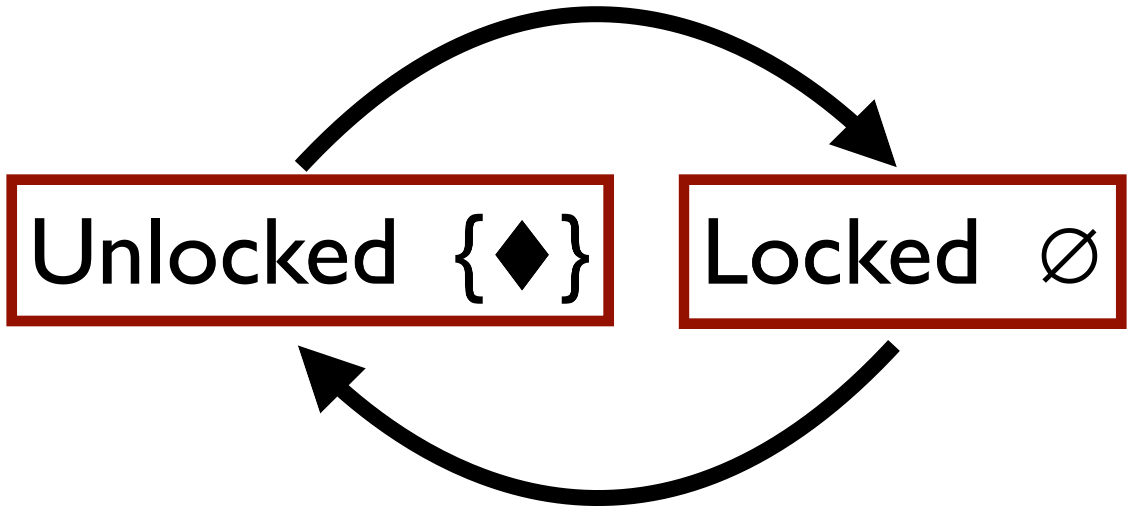
Conservation Law

Before *After*

$$T \uplus P = T' \uplus P'$$

Thread's tokens

Protocol's tokens



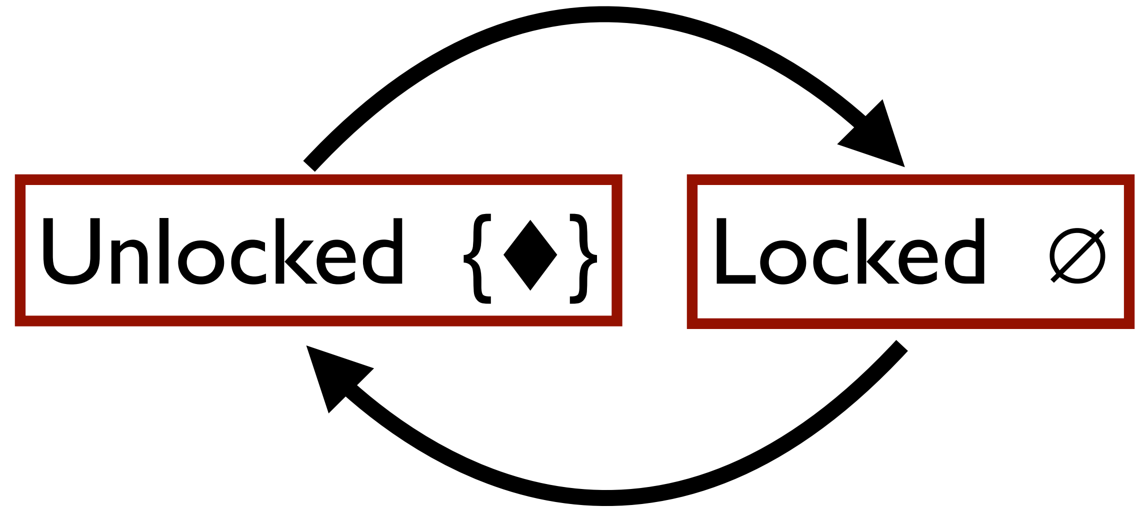
Conservation Law

Before *After*

$T \uplus P = T' \uplus P'$

Thread's tokens (→ T)

Protocol's tokens (→ P)



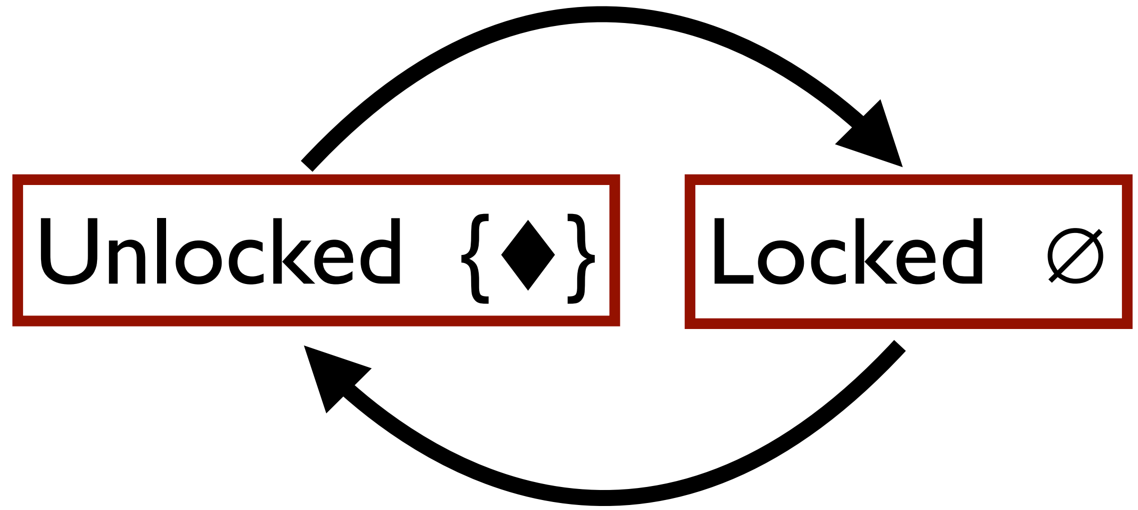
Unlocked → Locked: $\underbrace{\emptyset}_{T} \uplus \underbrace{\{\blacklozenge\}}_{P} = \underbrace{?}_{T'} \uplus \underbrace{\emptyset}_{P'}$

Conservation Law

Before *After*

$$T \uplus P = T' \uplus P'$$

Thread's tokens (T) — Protocol's tokens (P)



Unlocked → Locked: $\underbrace{\emptyset}_{T} \uplus \underbrace{\{\blacklozenge\}}_{P} = \underbrace{?}_{T'} \uplus \underbrace{\emptyset}_{P'}$

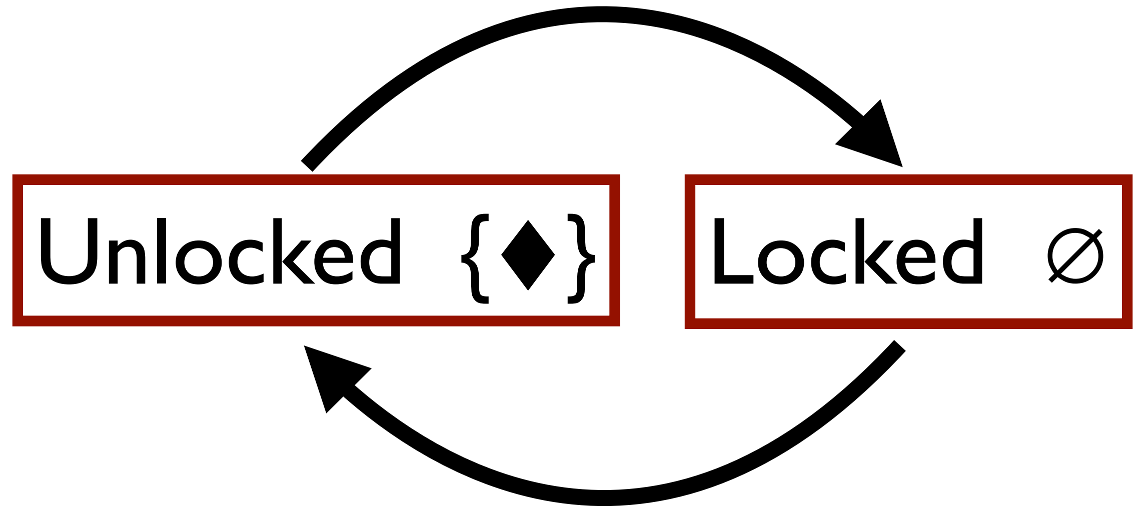
Conservation Law

Before *After*

$$T \uplus P = T' \uplus P'$$

Thread's tokens

Protocol's tokens



T P T' P'

Unlocked → Locked: $\emptyset \uplus \{\blacklozenge\} = \{\blacklozenge\} \uplus \emptyset$

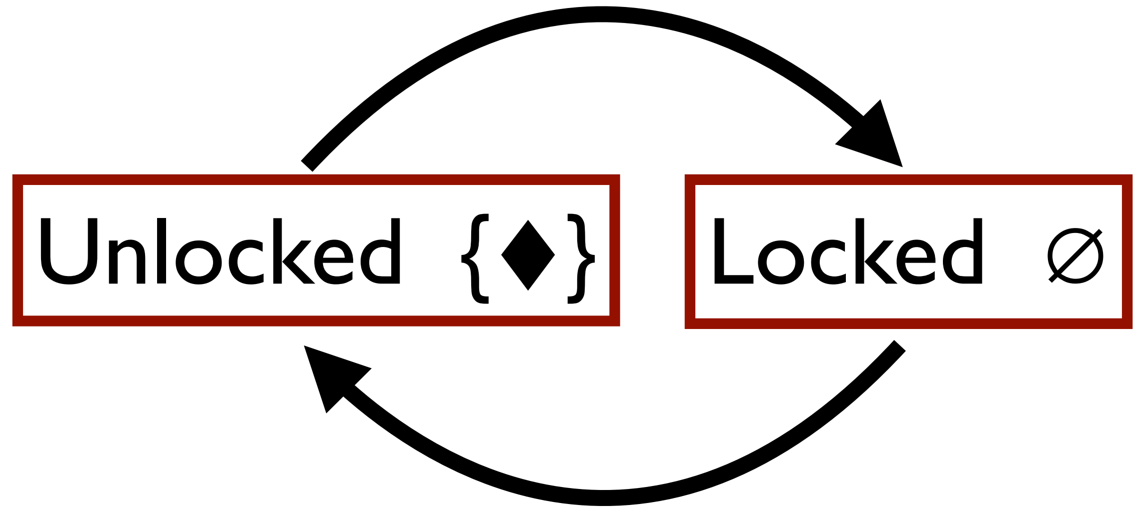
Conservation Law

Before *After*

$$T \uplus P = T' \uplus P'$$

T: Thread's tokens

P: Protocol's tokens



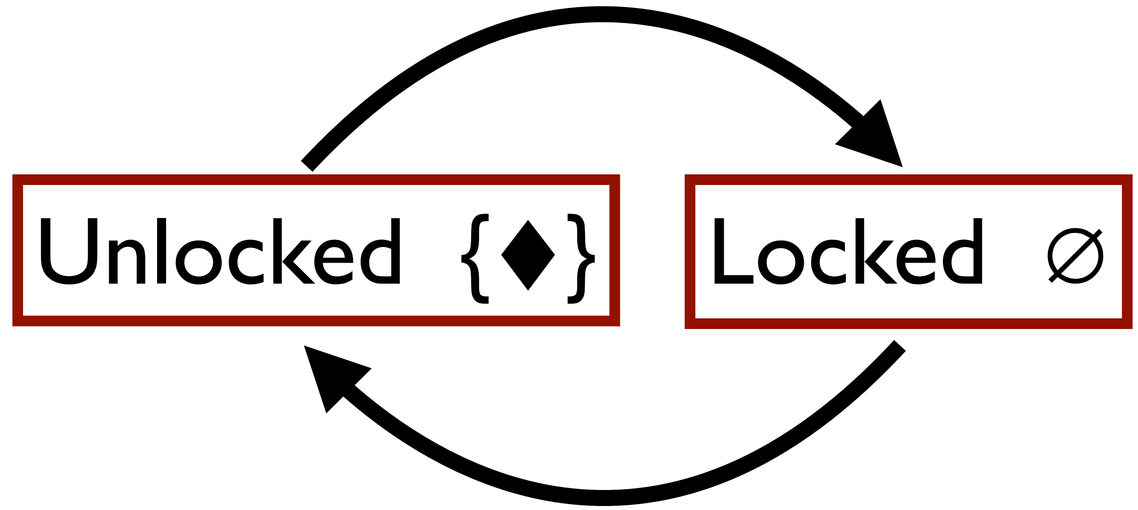
	T		P		T'		P'
Unlocked → Locked:	$\emptyset$	$\uplus$	$\{\blacklozenge\}$	$=$	$\{\blacklozenge\}$	$\uplus$	$\emptyset$
Locked → Unlocked:	$\emptyset$	$\uplus$	$\emptyset$	$=$	**?**	$\uplus$	$\{\blacklozenge\}$

Conservation Law

Before *After*

$$T \uplus P = T' \uplus P'$$

Thread's tokens ↗ (T)   Protocol's tokens ↖ (P)



T P T' P'

Unlocked → Locked: $\emptyset \uplus \{\blacklozenge\} = \{\blacklozenge\} \uplus \emptyset$

~~Locked → Unlocked: $\emptyset \uplus \emptyset = ? \uplus \{\blacklozenge\}$~~ *Impossible*

Key Idea

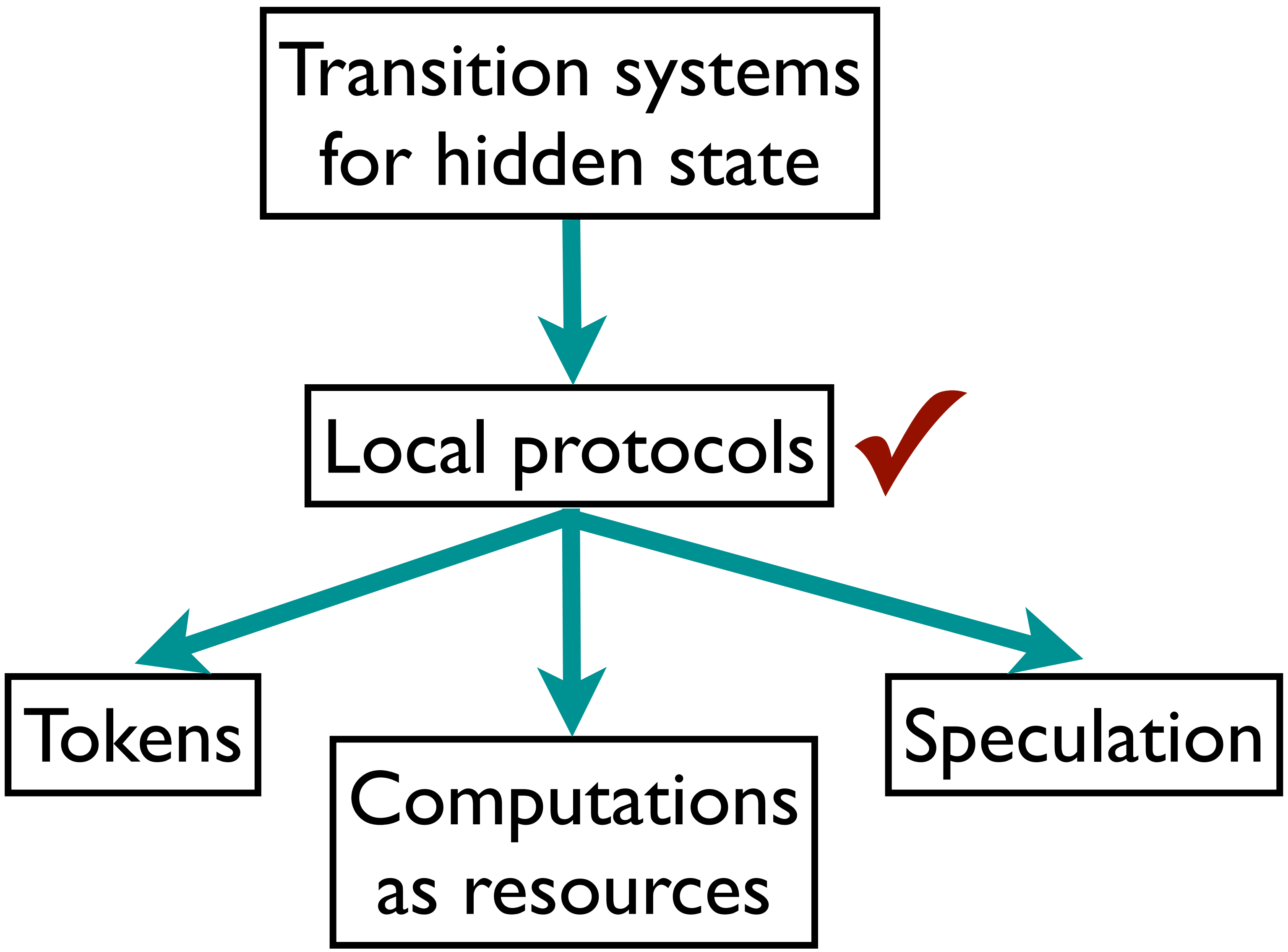
Tokens enable threads to gain and lose roles *dynamically*, via transitions

Key Idea

Tokens enable threads to gain and lose roles *dynamically*, via transitions

Rely: transitions with environment tokens
Guarantee: transitions with thread's tokens

Approach

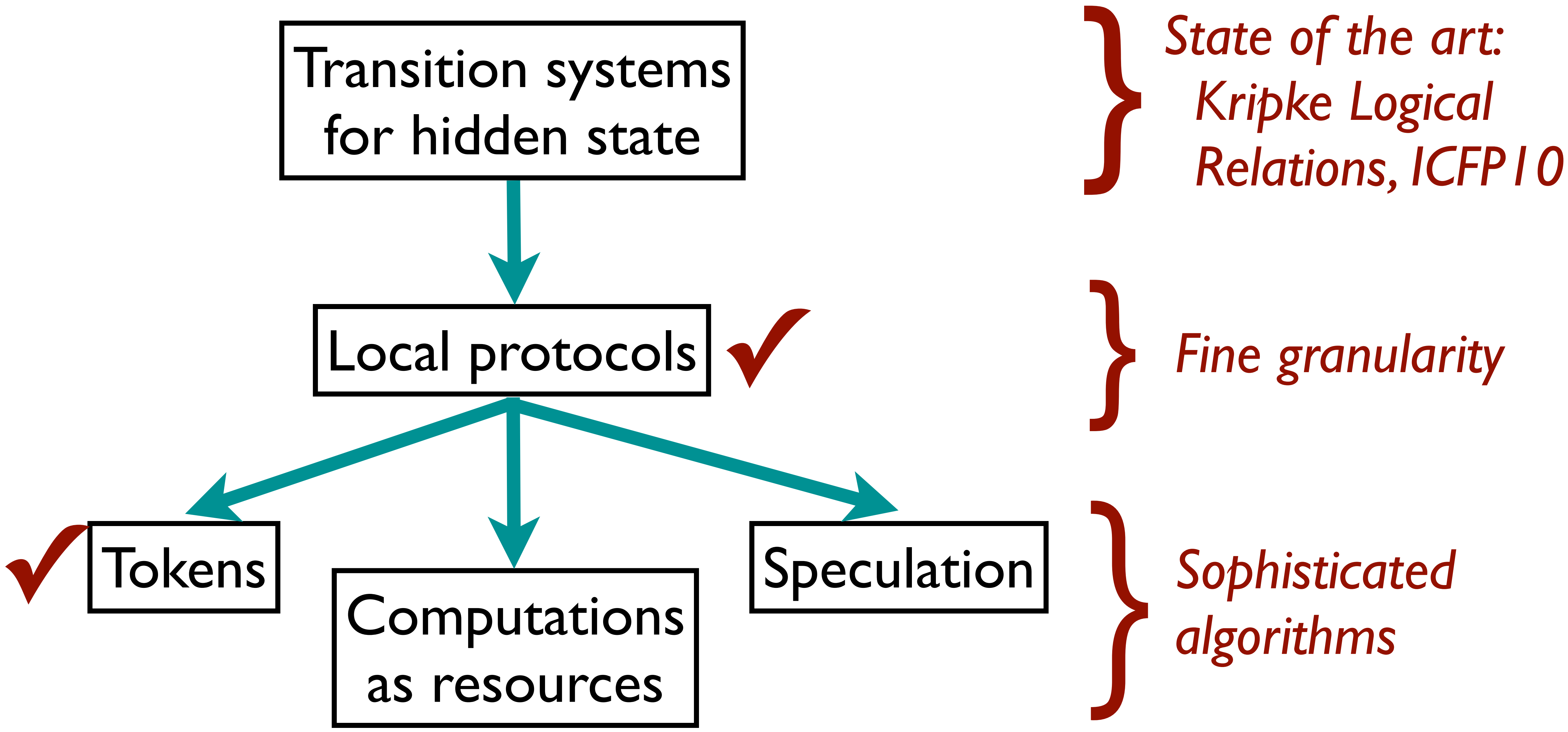


State of the art: Kripke Logical Relations, ICFP10

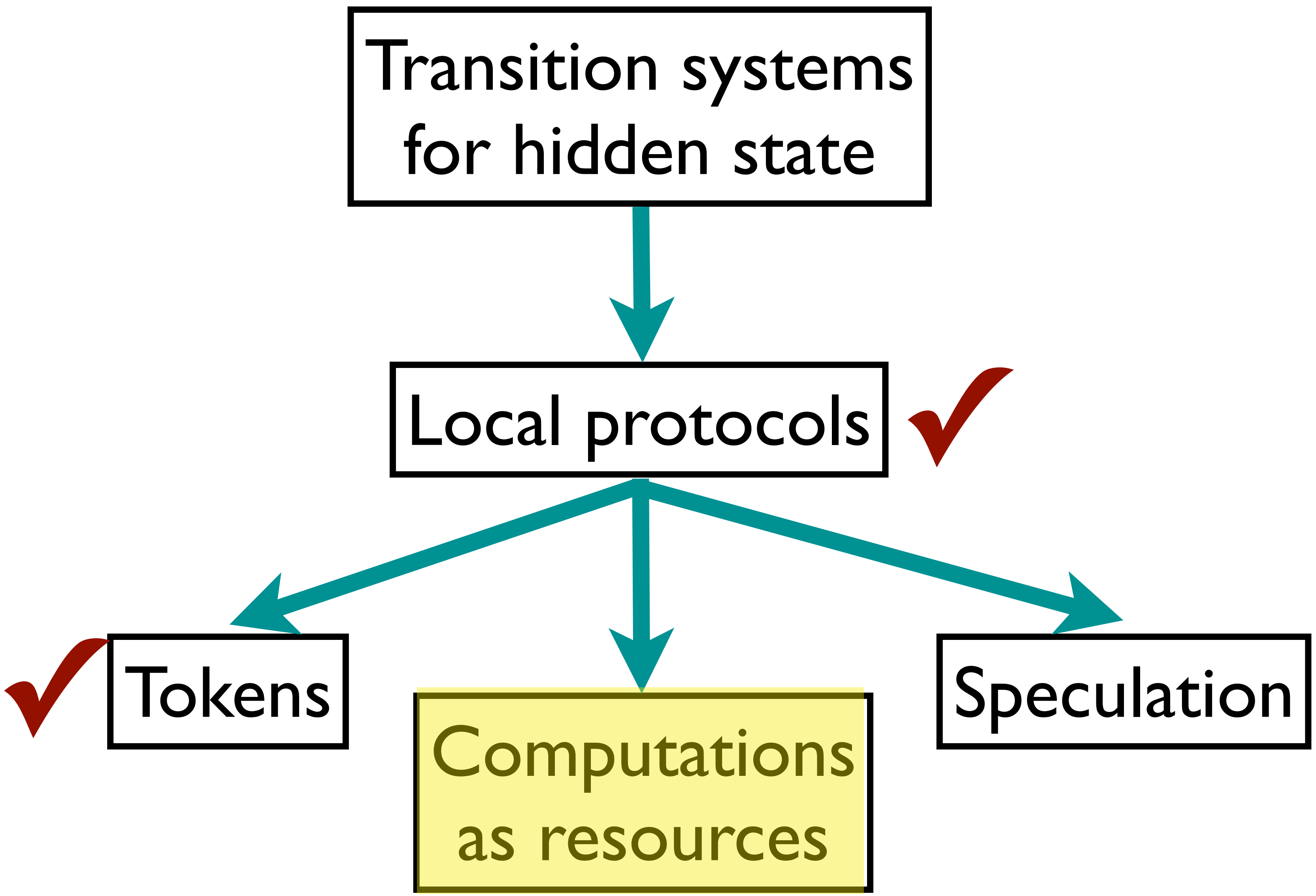
Fine granularity

Sophisticated algorithms

Approach


Transition systems for hidden state
Local protocols
Tokens
Computations as resources
Speculation
State of the art: Kripke Logical Relations, ICFP10
Fine granularity
Sophisticated algorithms

Approach



State of the art: Kripke Logical Relations, ICFP10

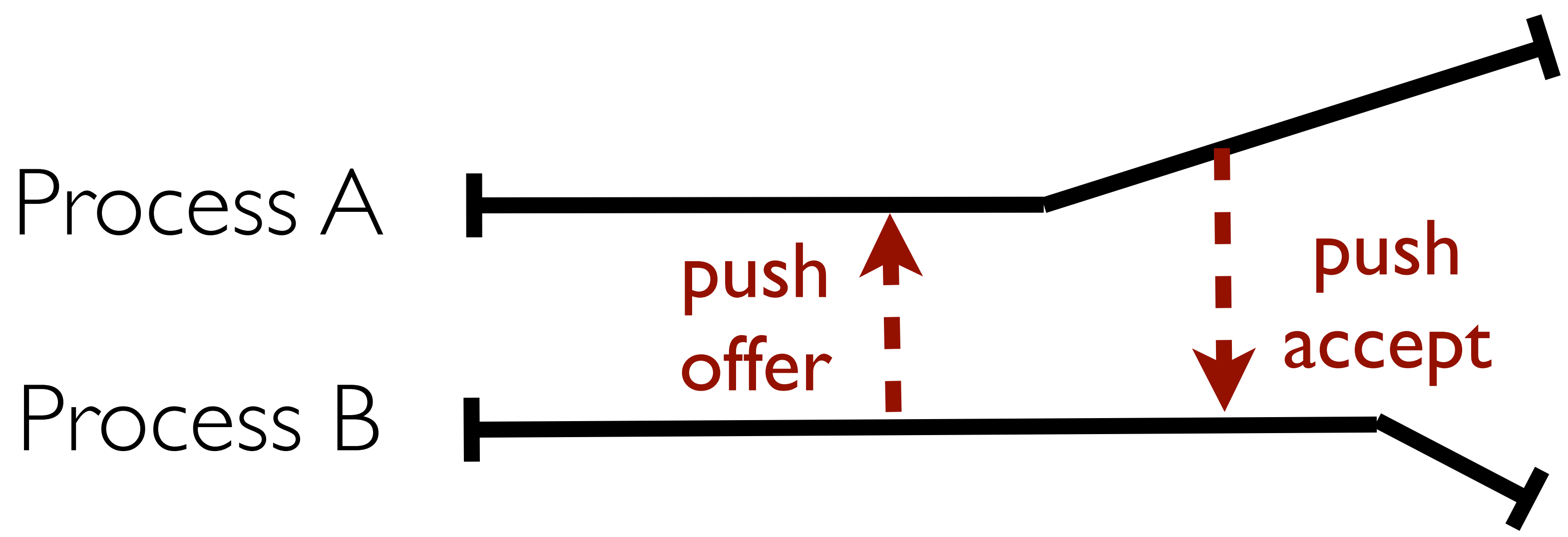
Fine granularity

Sophisticated algorithms

Problem:

Cooperation is inherently non-thread-local

e.g. elimination stacks



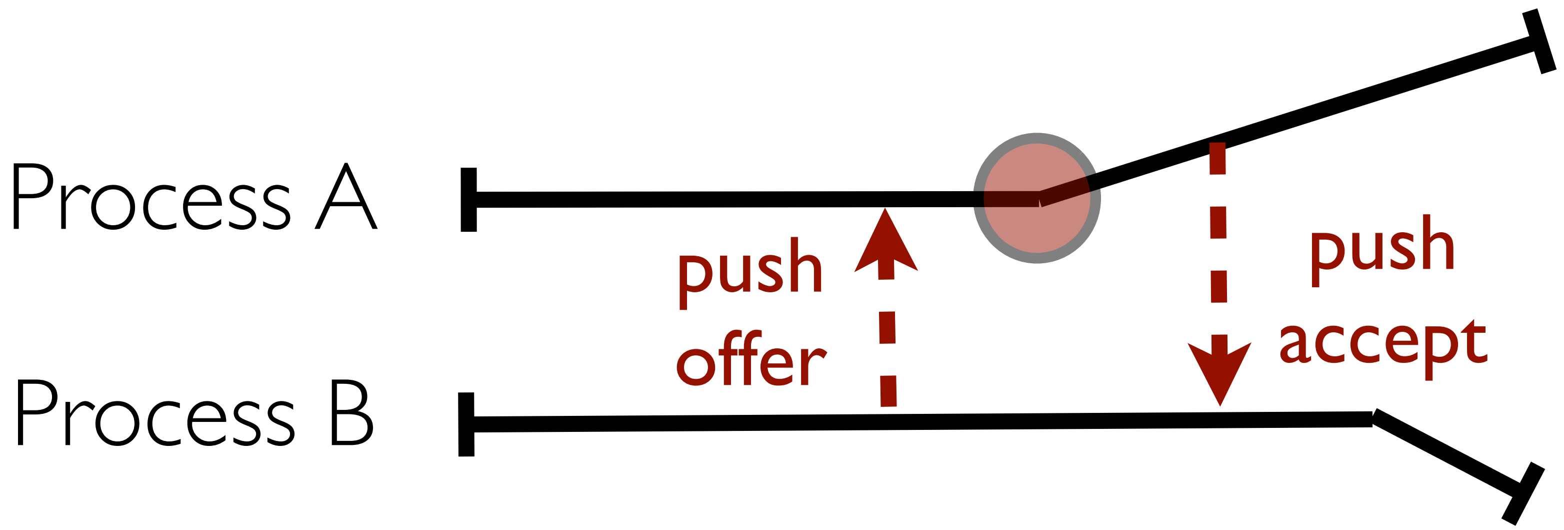
Solution:

Make thread's specification a *shareable resource*

Problem:

Cooperation is inherently non-thread-local

e.g. elimination stacks



Solution:

Make thread's specification a *shareable resource*

Algorithm	Local protocols	Tokens	Computational resources	Speculation	
Treiber's Stack	X				✓
Michael-Scott Queue	X				✓
Hand-over-hand set	X	X			✶
Elimination Flags	X	X	X		✓
Lazy set	X	X	X	X	✶
Conditional CAS	X	X	X	X	✓
K-CAS	X	X	X	X	✶

The Takeaway

Analyze fine-grained algorithms with fine-grained protocols!

To scale up, must address:
- varying roles
- cooperation
- nondeterminism

Our contributions:
- tokens
- computational resources
- speculation