Modeling Process

Rich Hickey

Which are more fundamental?

- Messages, classes, encapsulation, inheritance, dispatch...
- Time, value, identity, state, persistence, transience, place, perception, visibility, memory, process...

Coming to Terms

Value

- An *immutable* magnitude, quantity, number... or immutable composite thereof

Identity

- A putative entity we associate with a series of causally related values (states) over time

State

- Value of an identity at a moment in time

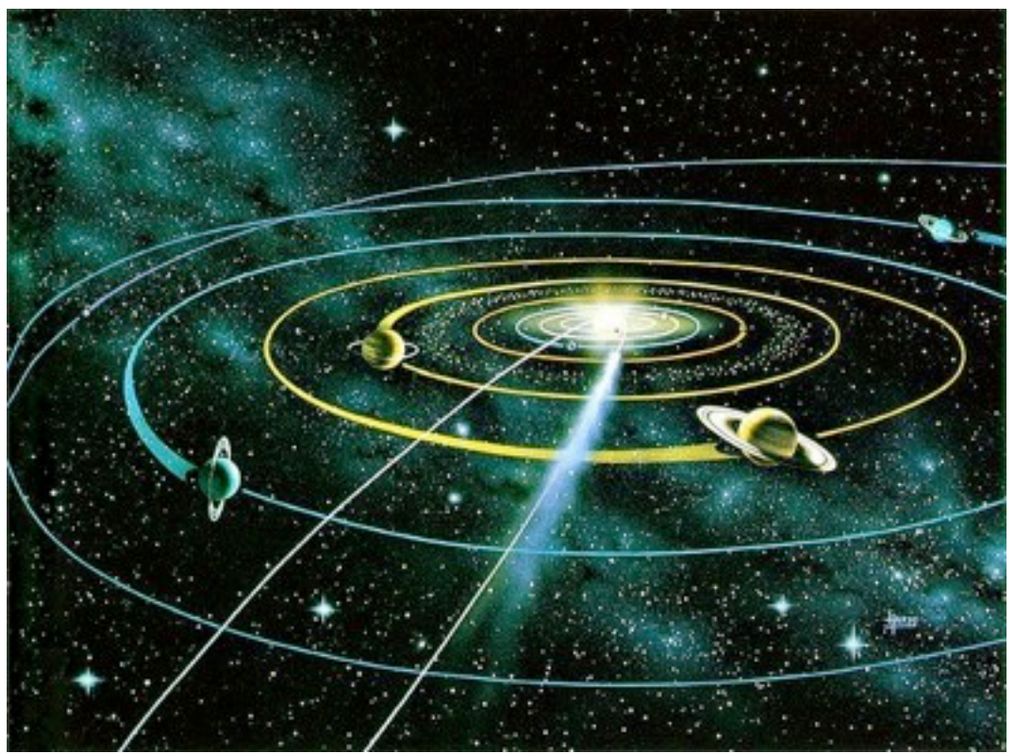
Time

- Relative before/after ordering of causal values

A Real Problem

Place

- "open space"
- Relative
- Include time coordinate, and process results happen in new places

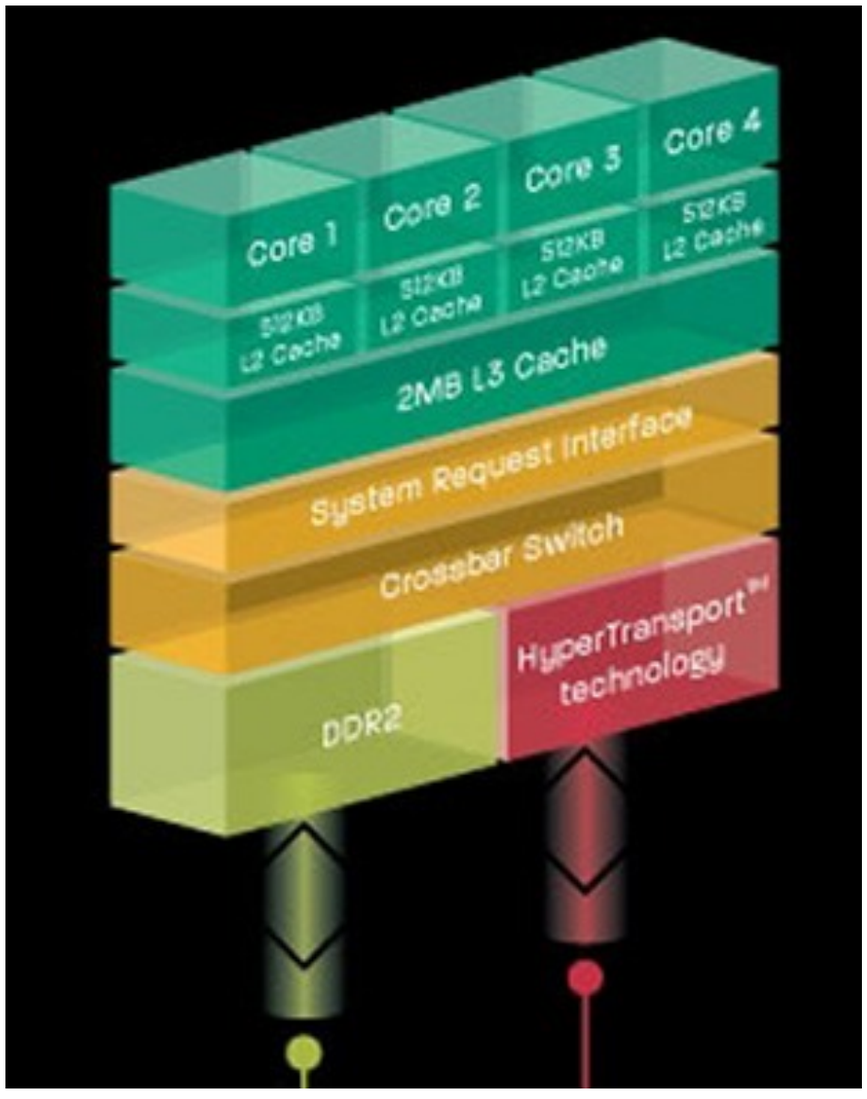


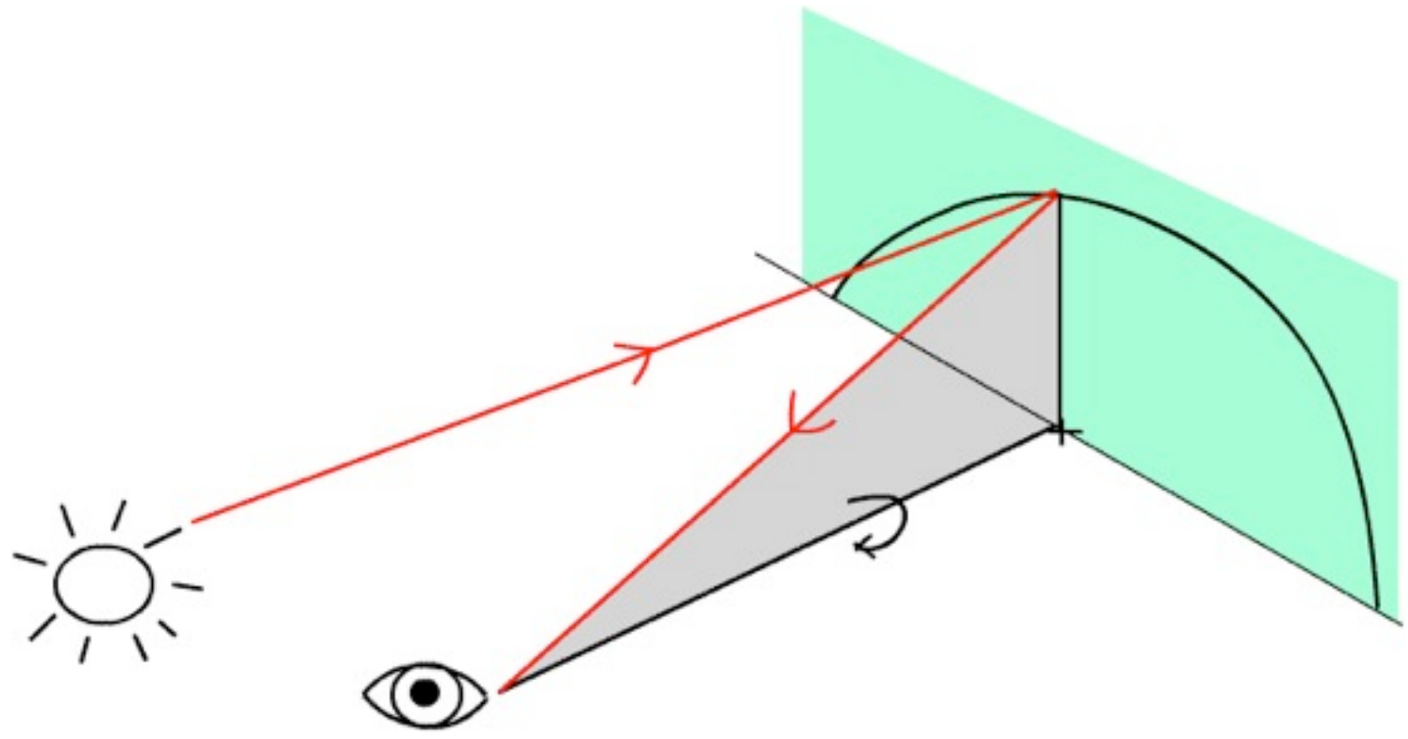
What Would a Program Do?

Not this!

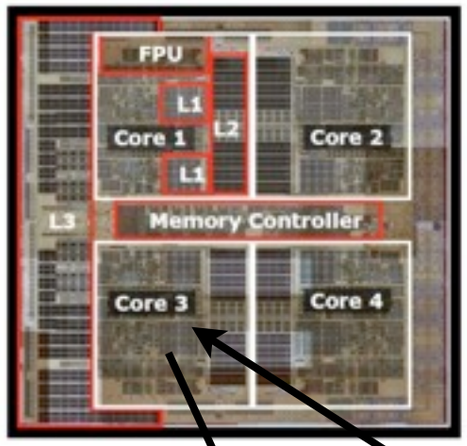
Are places in charge?

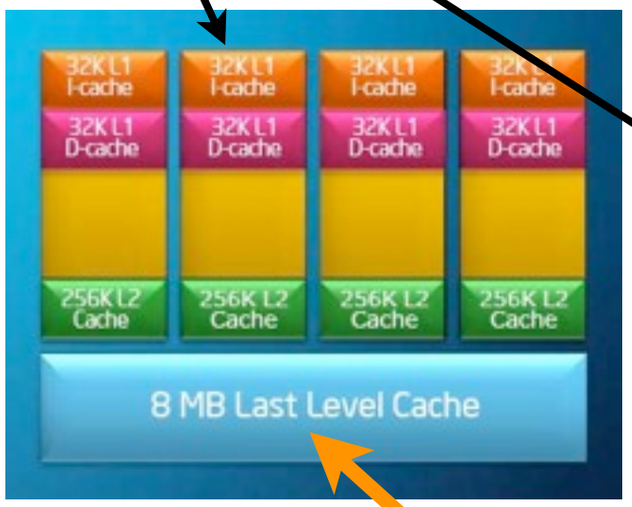
What do we see?





Our Problem



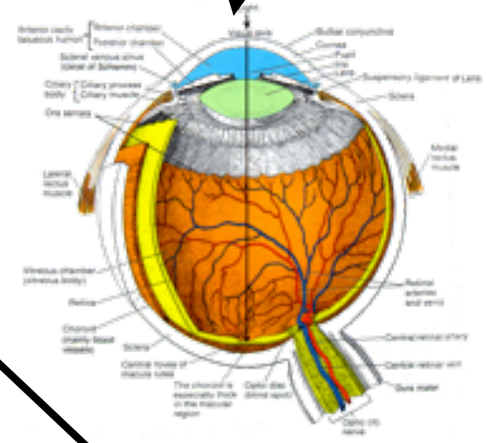


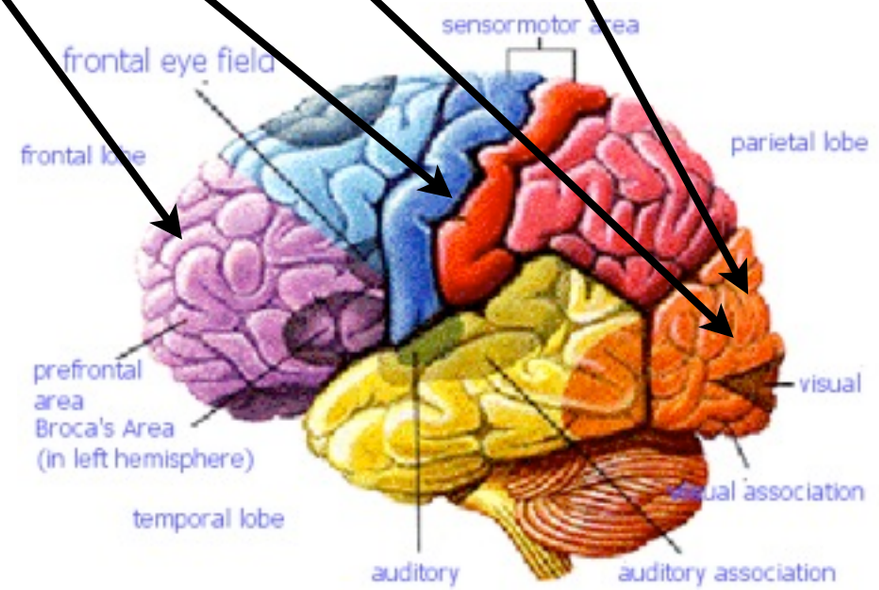
Reality

Perception

Memory

Logic





Perception

- Perceive - “take entirely”
- Sensory systems only ever perceive the past
- Discretizing, snapshots
- Most useful when coupled with memory
- Fidelity matters
- Visible == “can be perceived”
  - not merely ‘reachable via reference’

Memory

- "mindful, remembering"
- If our mental memory behaved the way we use computer memory, we'd be ill
  - In the mind we talk about *forming* memories
    - New memories about the same identities don't replace the old
- Fidelity matters
- Stability matters (persistence)

Program Memory

- Sometimes we use computer memory like brain memory
- Sometimes like perception
- Sometimes (commonly, most of the time) like places

Using the same memory for everything

- Destroys the past
- Corrupts remembering
- Interferes with perception
- We must use memory for all three things, but not necessarily *the same* memory

Process

Process

- "go forward, advance"
- They're not called "food calculators"
- Potentially richer than this
  - Manipulate contents of place(s)
  - May involve multiple forces

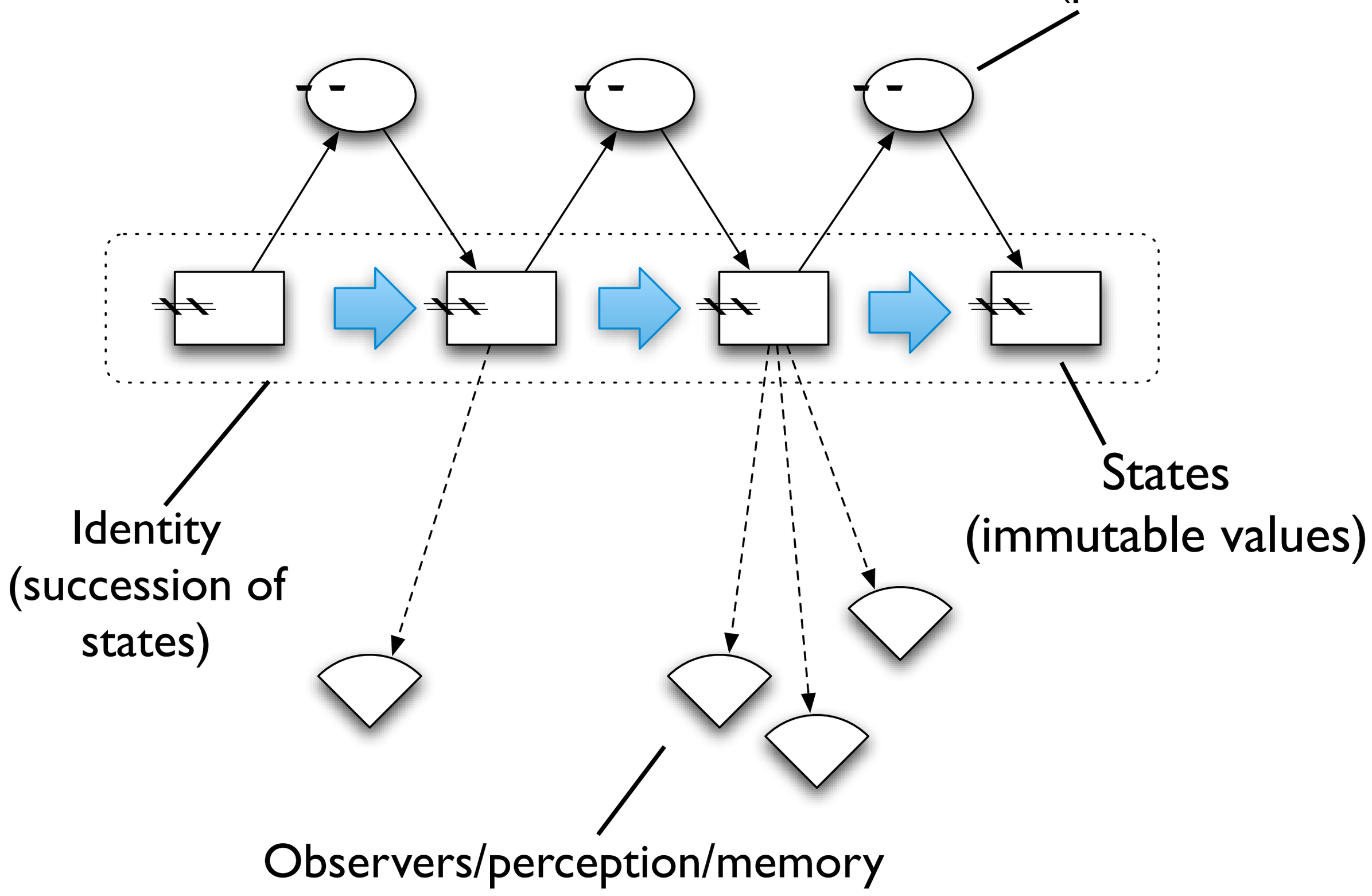
Process across multiple places

Process with multiple forces/participants

Philosophy

- Things don't change in place
  - Becomes obvious once you incorporate time as a dimension
  - Place includes time
  - The future is a (multi-force) function of the past
  - Co-located entities can observe each other without cooperation
  - Coordination is desirable in local context

Epochal Time Model



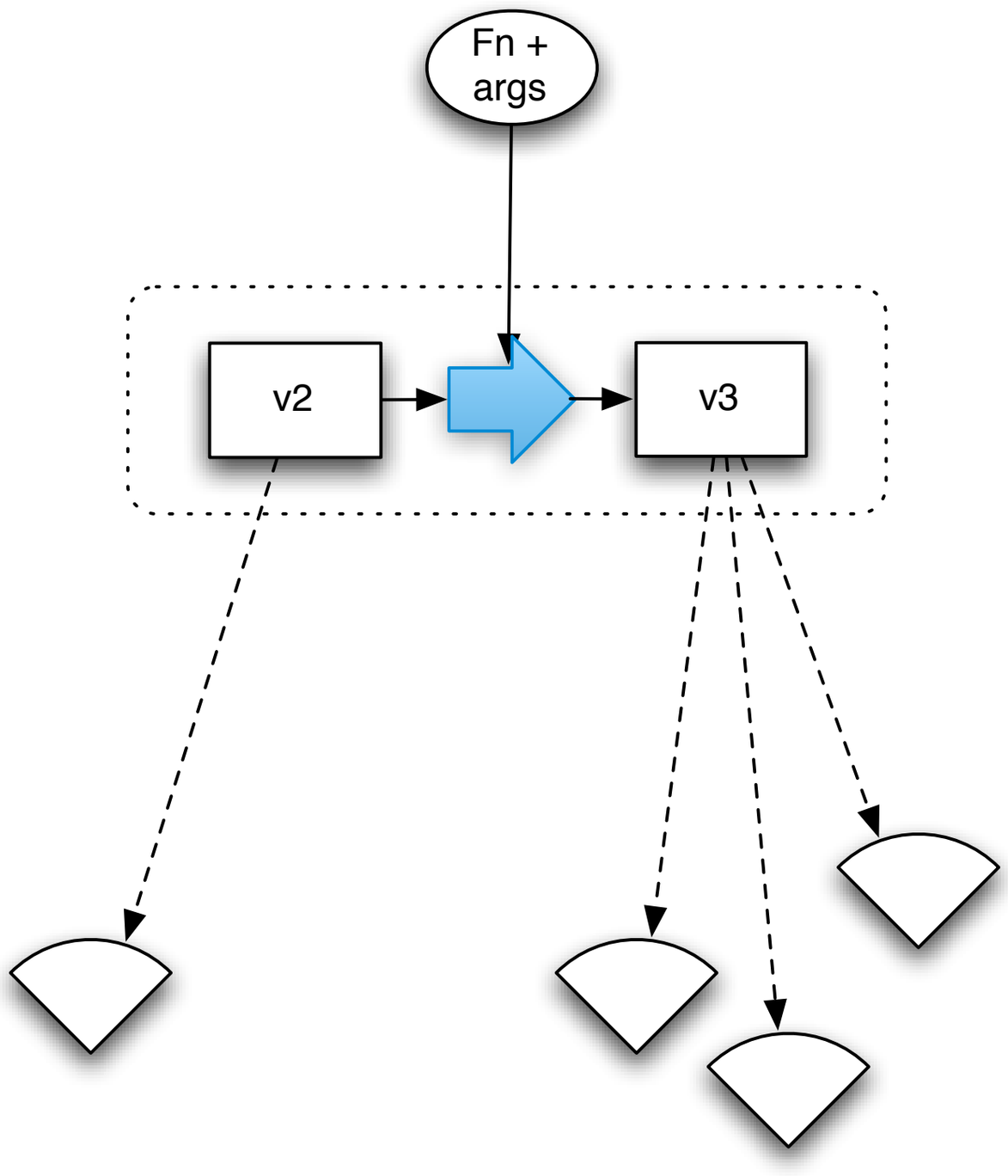
Persistent

- "lasting or enduring tenaciously"
  - Root: "to stand firm permanently"
- When applied to data structures
  - A) safe on the disk (not today's topic)
  - B) immutable++
- Great fit for perceptions and memories

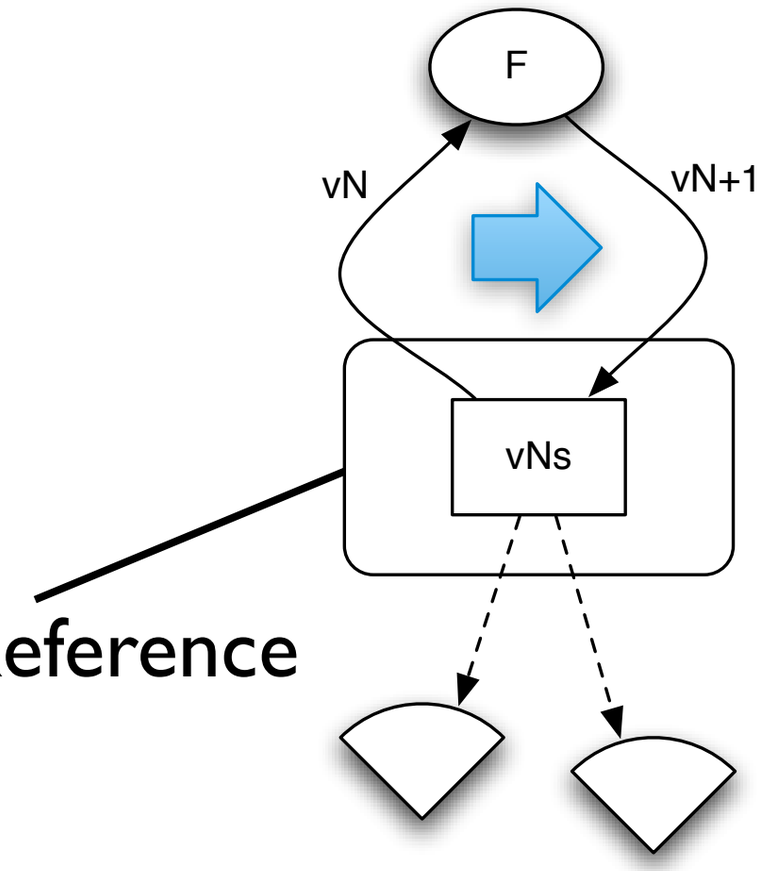
Identity Constructs as Gatekeepers of Time

- Responsible for coherent successive states
  - Multiple semantics possible
- And providing proper values to observers
- Support coordination (multiple places) and process functions supplied from multiple threads of control (multiple participants)

Functional Model



CAS as Time Construct

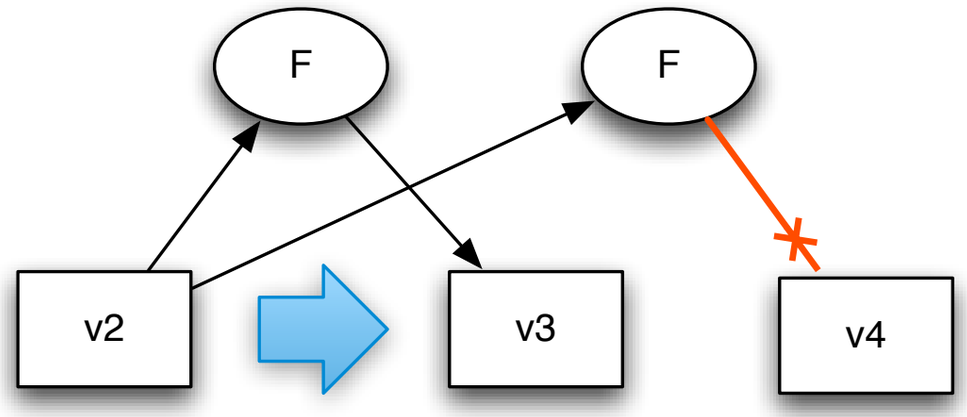


AtomicReference

```
(swap! an-atom f args)
```

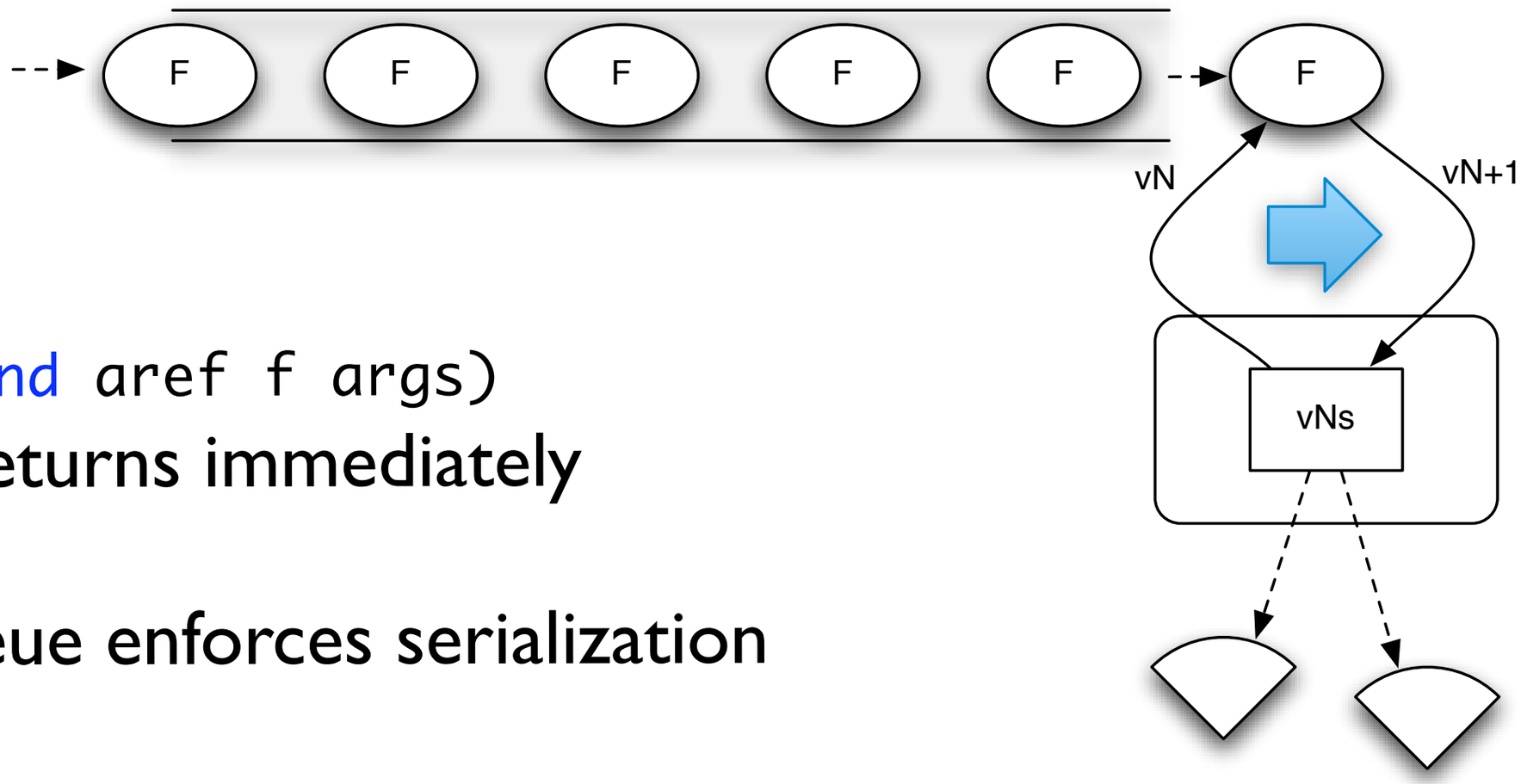
`(f vN args)` *becomes* vN+1

- can automate spin



- 1:1 timeline/identity
- Atomic state succession
- Point-in-time value perception

Agents as Time Construct




```
(send aref f args)
```
returns immediately

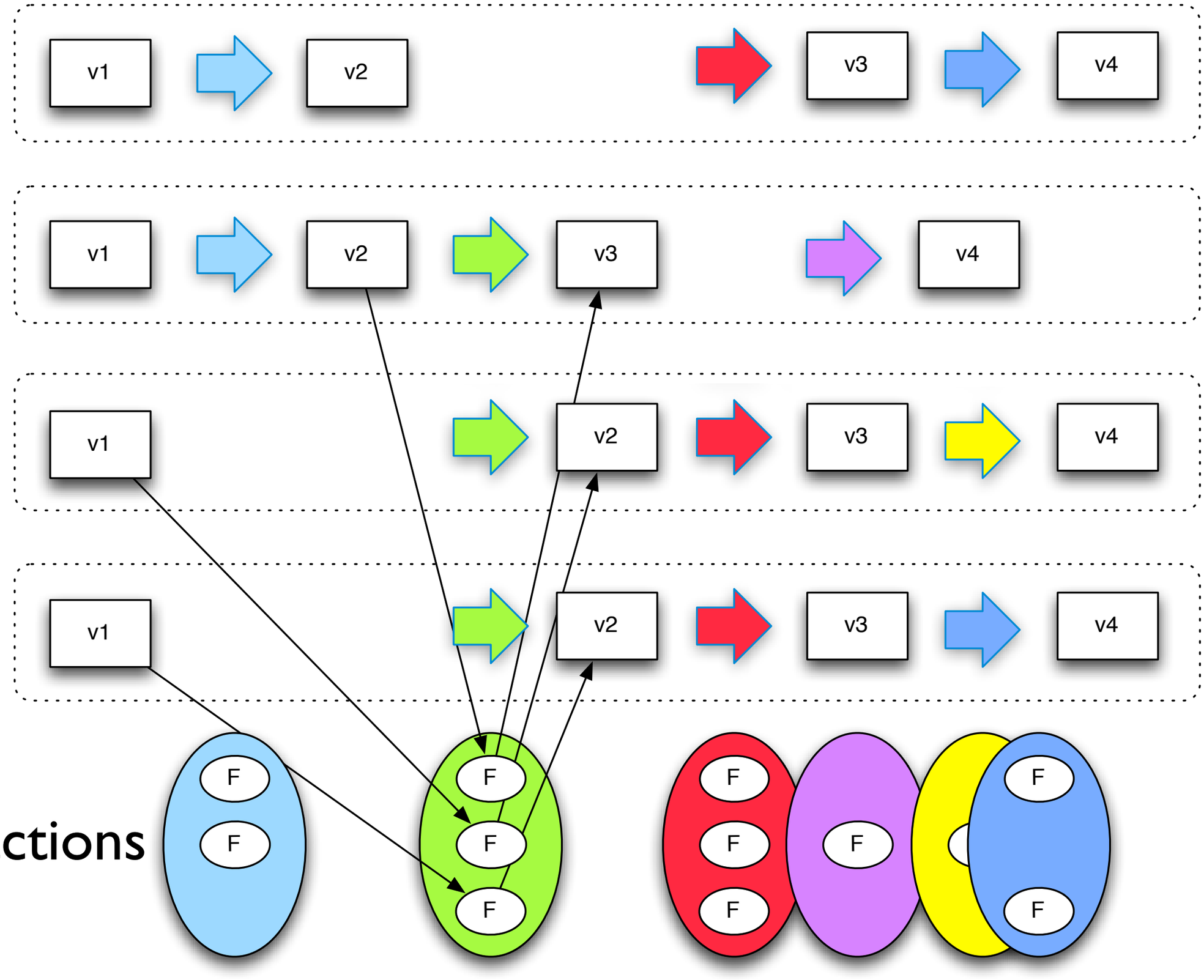
queue enforces serialization

`(f vN args)` becomes vN+1

happens asynchronously in thread pool thread

- 1:1 timeline/identity
- Atomic state succession
- Point-in-time value perception

STM as Time Construct



Transactions

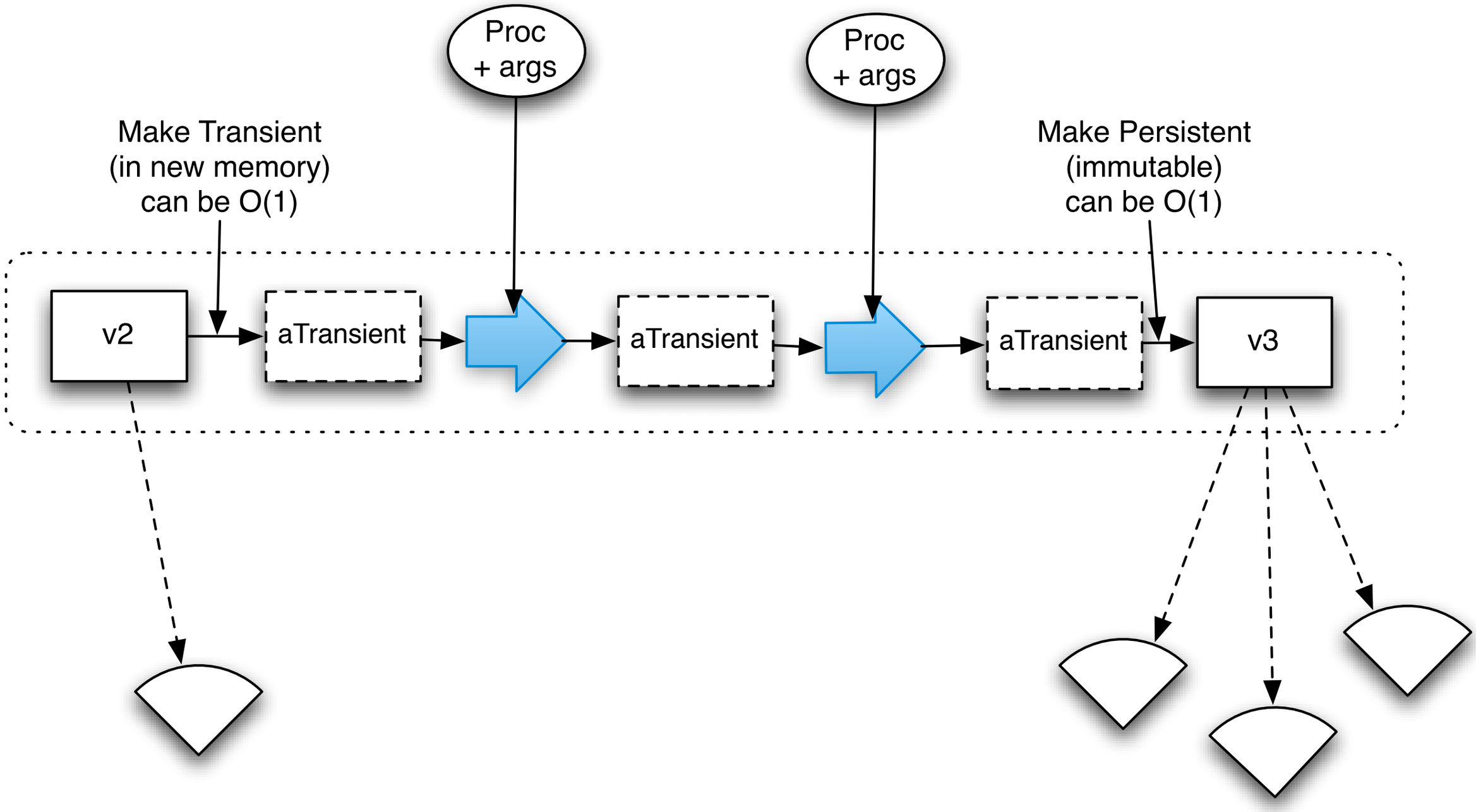
But...

- What if my logical unit of work involves a million steps?
- Creating a million interim values via pure function invocation is a waste
- “I’m going back to my cubbyholes!”

Transient

- "not lasting, enduring, or permanent; transitory"
  - Root: "go across"
- When applied to data structures:
  - Not persistent!
  - Each operation returns the *next* transient
  - Can't presume modify-in-place
    - Doesn't preclude it either
    - No identity

Transient-Based Model

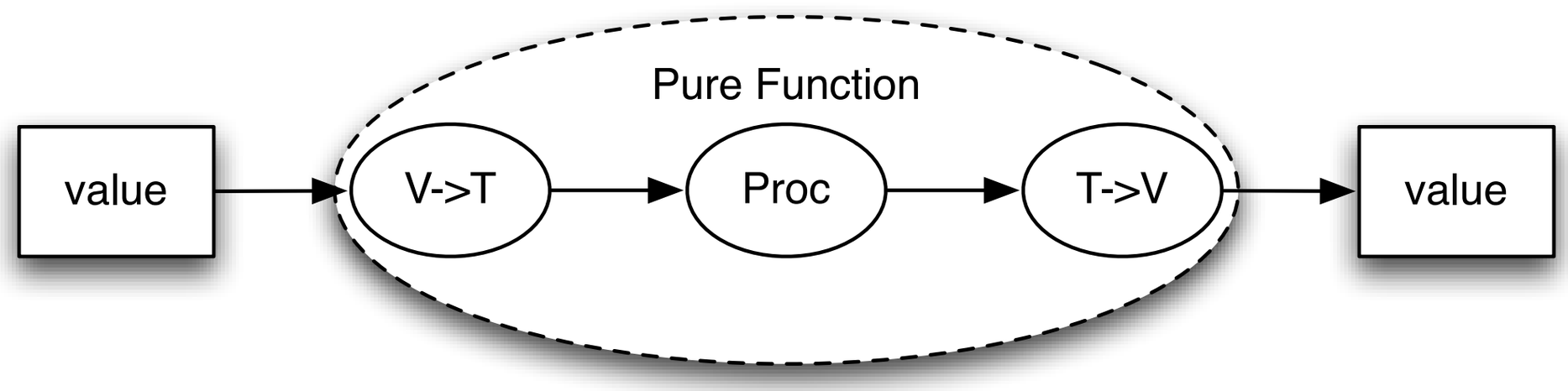


What about those Procs?

- Might modify their arguments
- Isn't this just icky mutable side-effecting coding again?
  - hard to test
  - difficult to reason about
  - No!

Proc

- Function of transient to transient
- Like pure function, can't effect the world nor be effected by it
- Only used in a context where transient cannot leak
- Can always be sandwiched in value->transient and transient->value functions and become 'pure'



The sweet, creamy, efficient middle of pure functions

Transient/Proc Model

- Prototype implementation - ‘pods’
- Can support multiple participants, in multiple threads
  - and coordination of multiple identities/places
  - even ad hoc grouping
- But not arbitrary composition/nesting
  - Same limitation as locks, but detectable
- Agents could support as well

Summary

- We need to talk about these things
- Better, more precise language and terms
- Language and library support
  - Examine high-level abstractions and constructs in terms of these fundamental issues

Thanks for listening!

http://clojure.org