Clojure
Lisp for the Real World

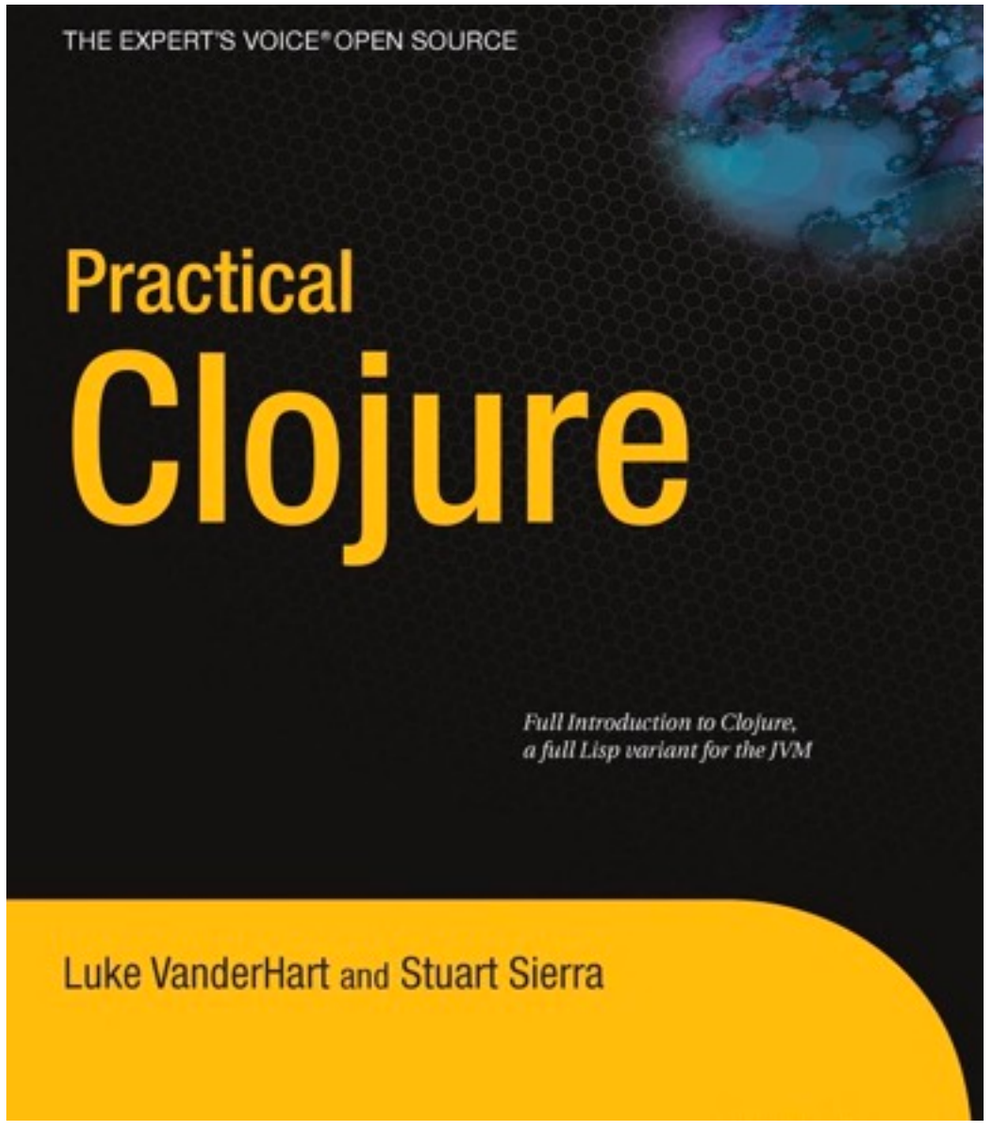
@stuartsierra
#clojure

Stuart Sierra

Relevance, Inc.

Clojure/core

Clojure contributor

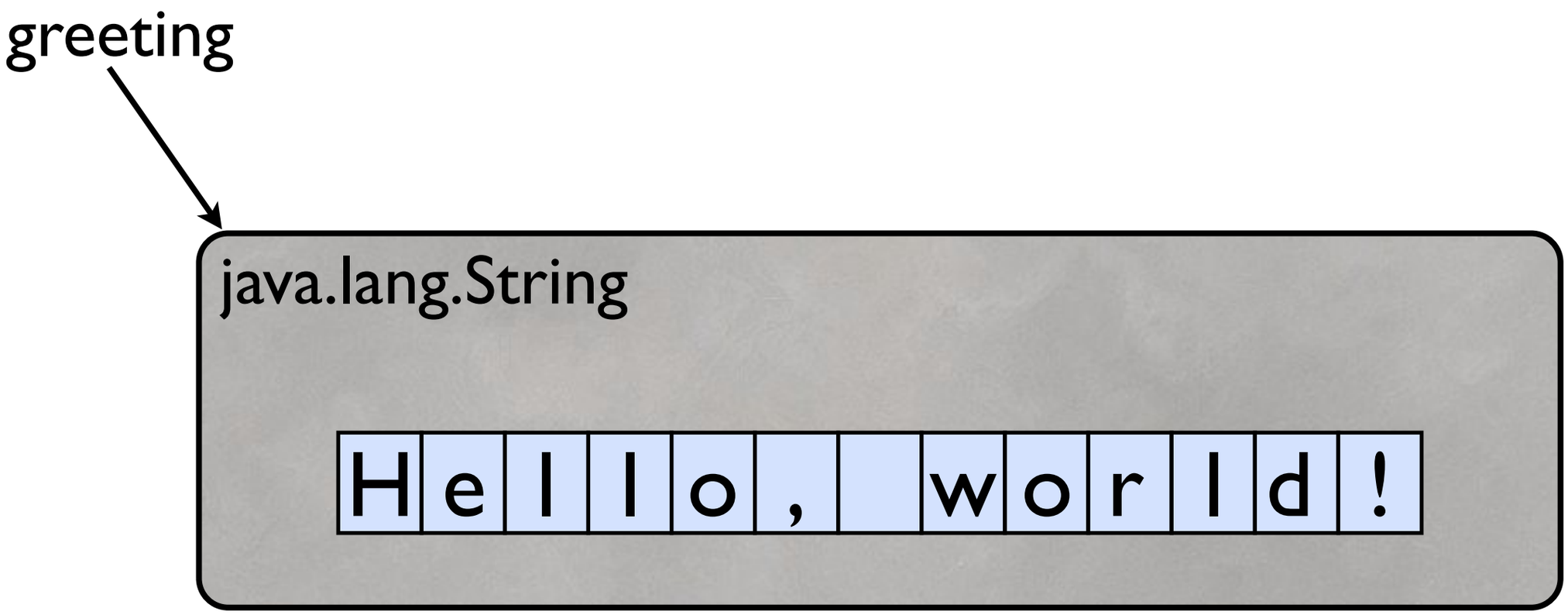


Values

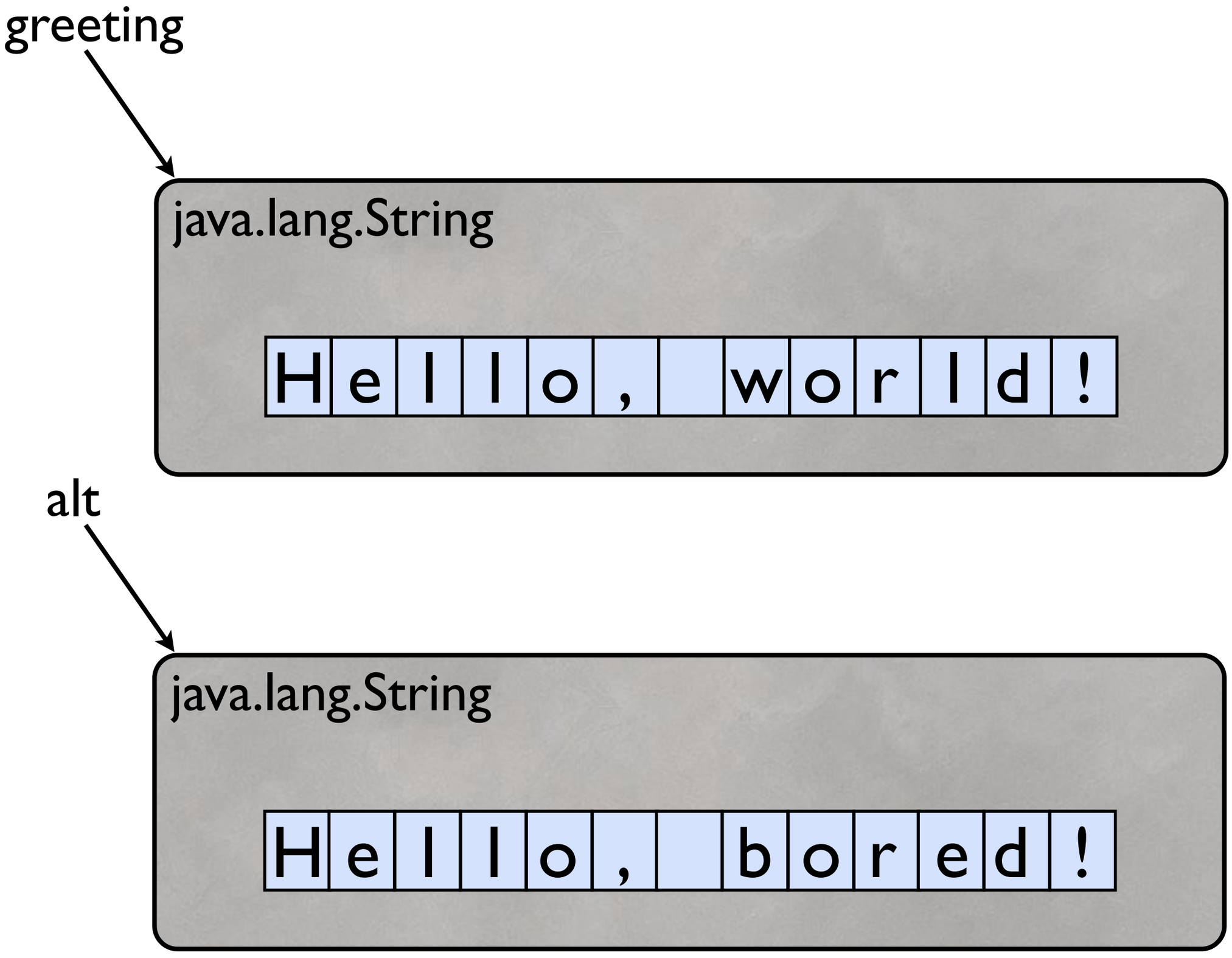
Values

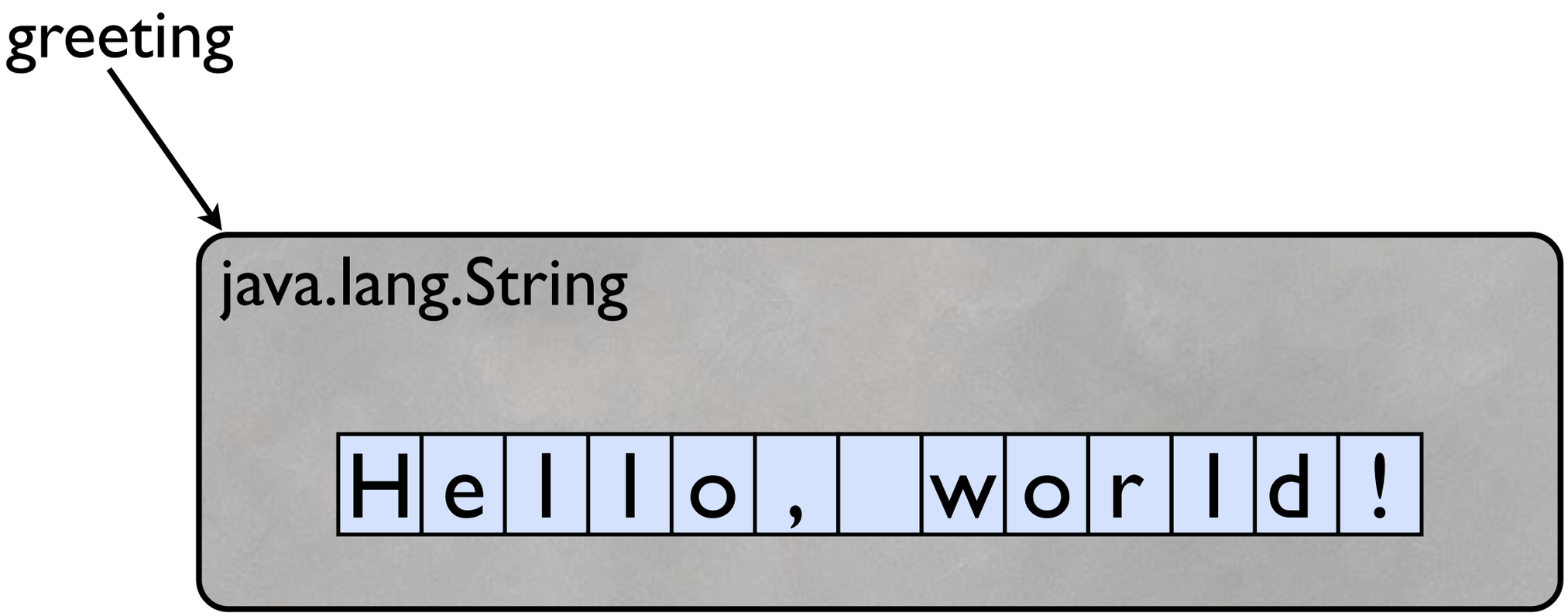
- Immutable

String greeting = new String("Hello, world!");



```
String alt = greeting.replace("world", "bored");
```





Immutable

Clojure Values

`3`	`Long`
`6.022e23`	`Double`
`"Hello, world!"`	`String`
`3.0M`	`BigDecimal`
`92233720368547758O8N`	`BigInt`
`2/3`	`Ratio`

Clojure Values

even?	Symbol
`:last-name`	Keyword
`(print 99 "bottles")`	List
`[3.14 :pi]`	Vector
`{:x 3 "y" 4}`	Map
`#{7 9 "foo"}`	Set

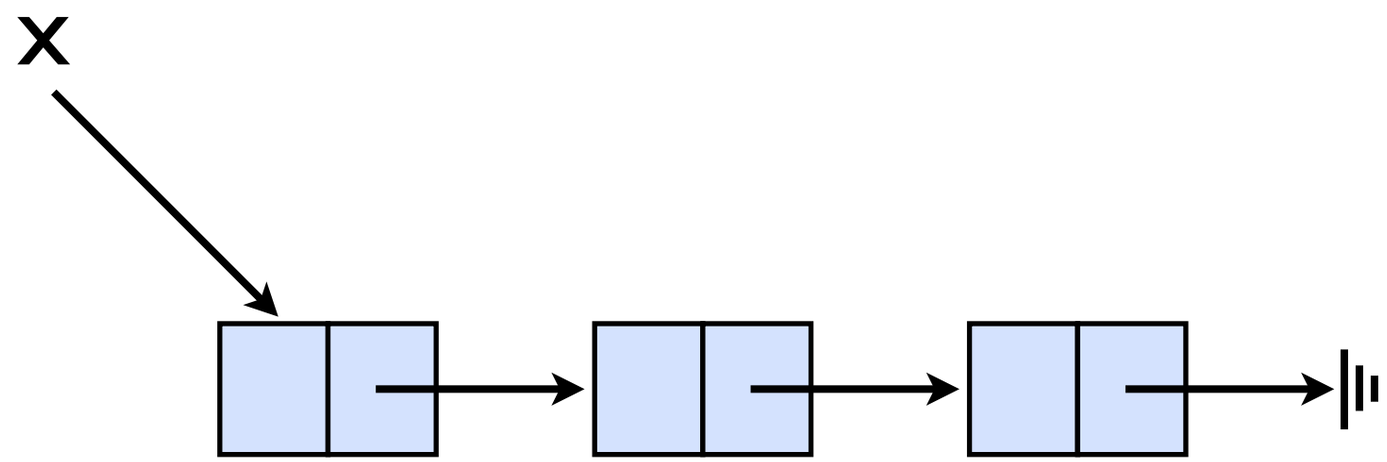
Clojure Values

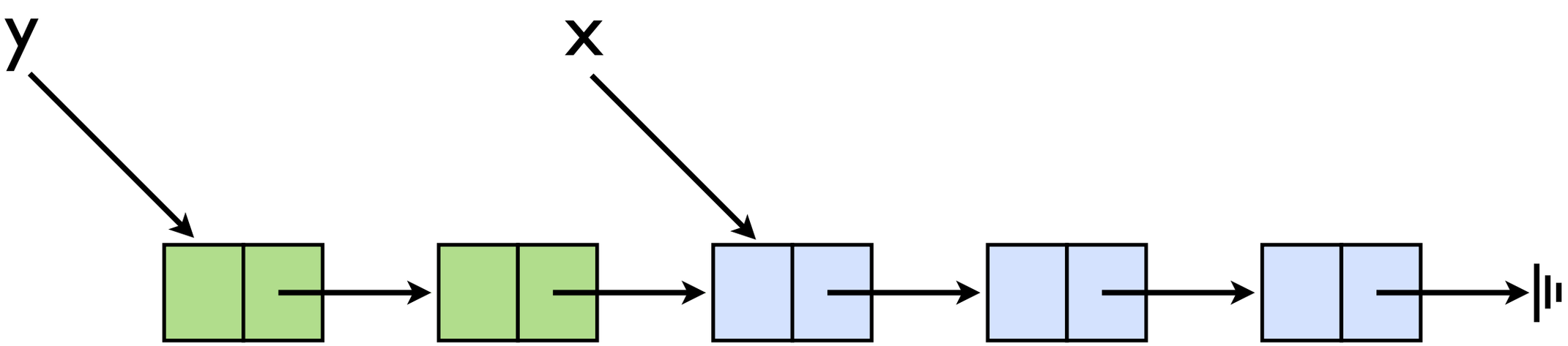
- Immutable

Clojure Values

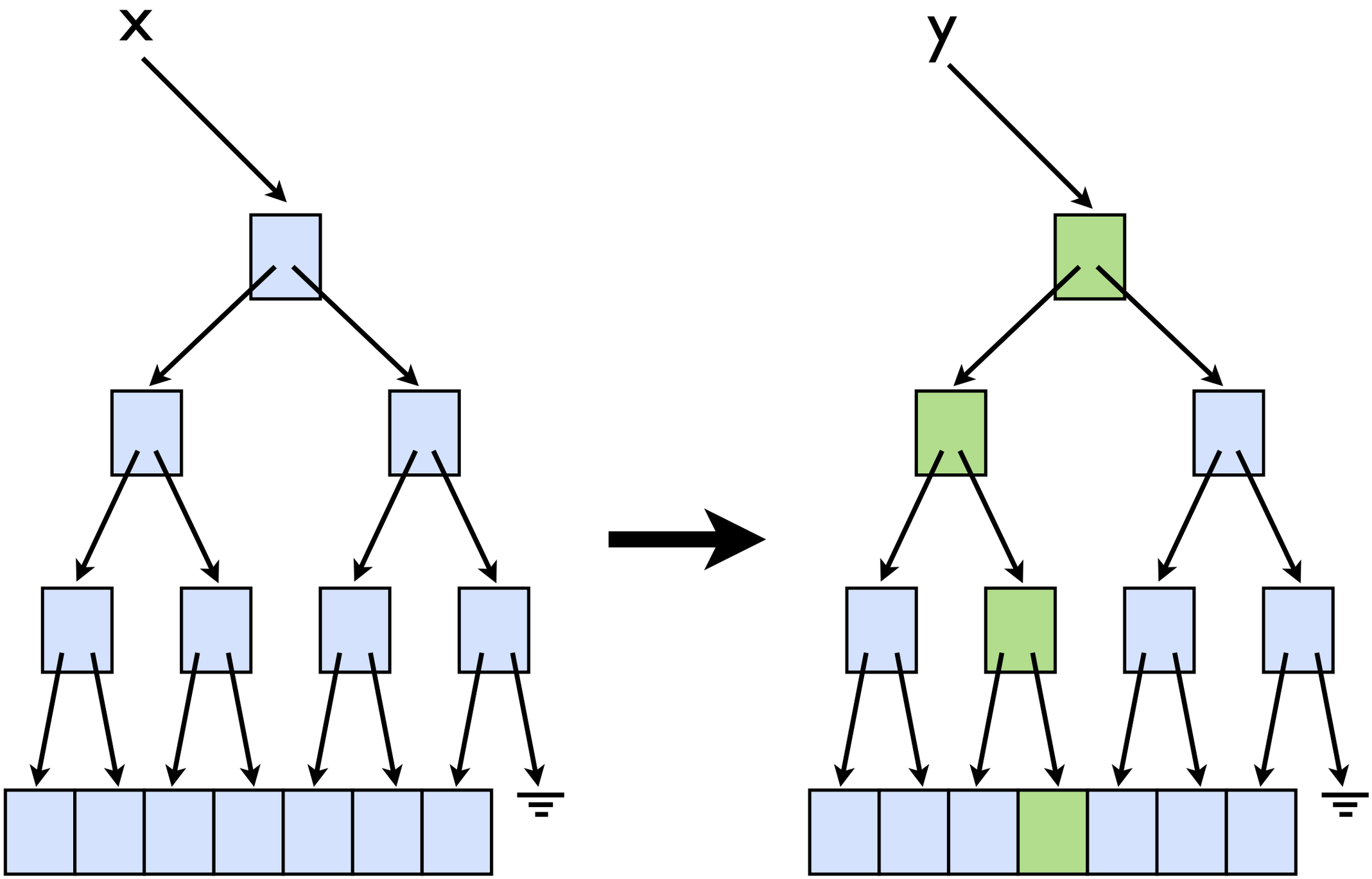
- Immutable
- Persistent

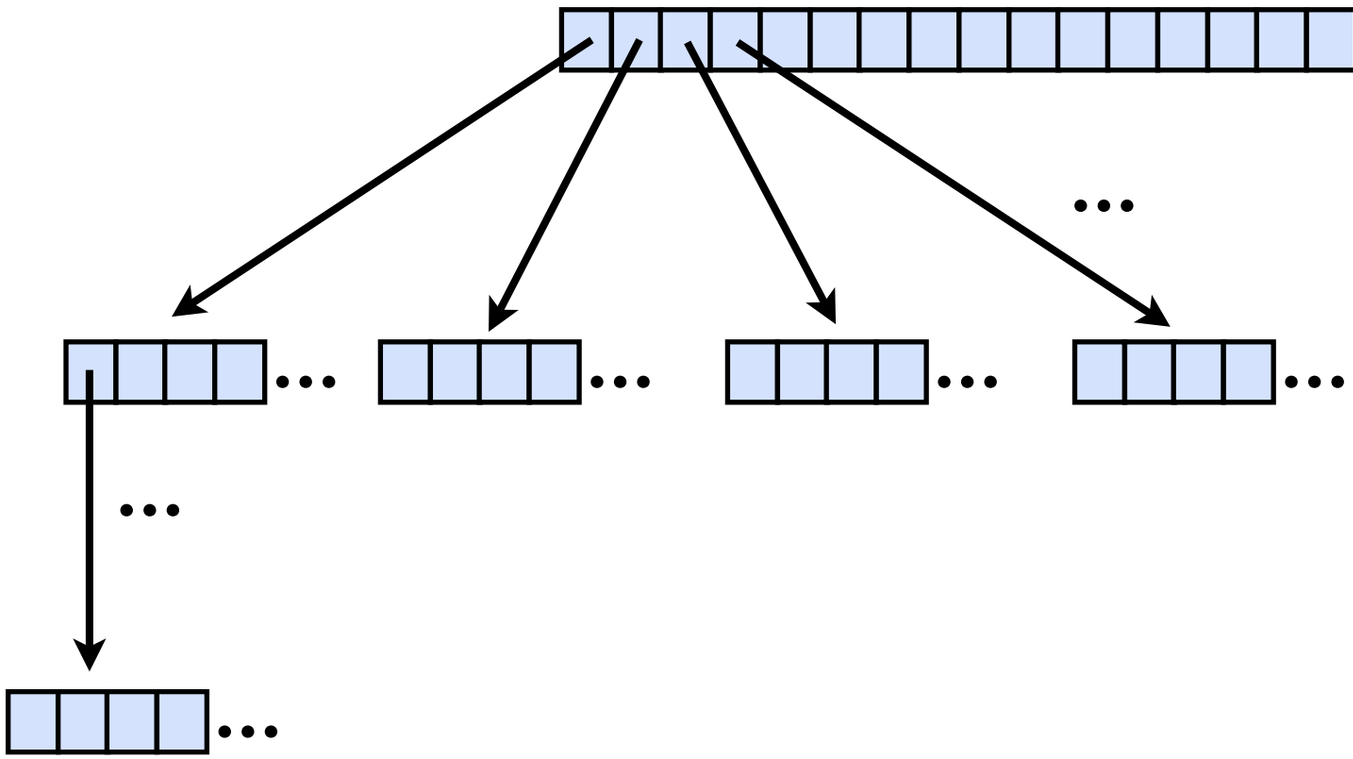
Shared Structure


X



Persistent


x
y



Clojure Values

List	O(1) at the head O(n) anywhere else
Vector	$O(\log_{32} n)$ access
Map	$O(\log_{32} n)$ access
Set	$O(\log_{32} n)$ contains?

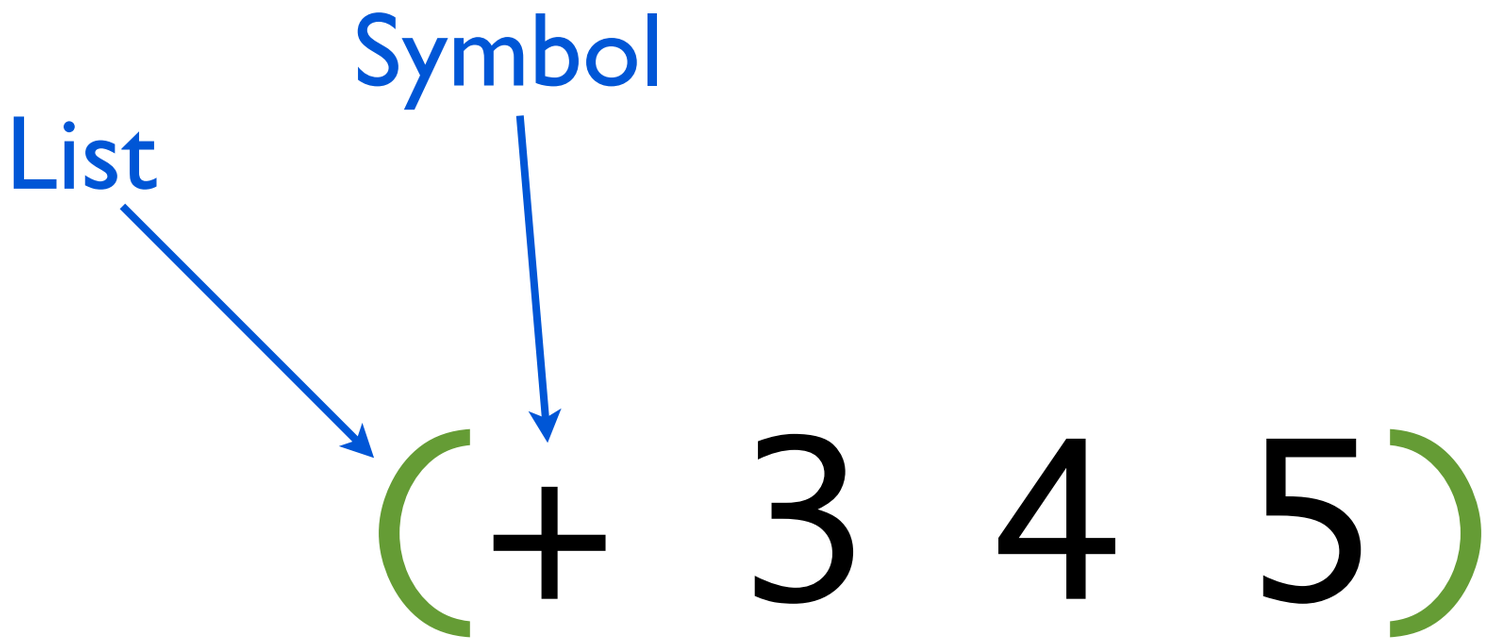
In the real world,

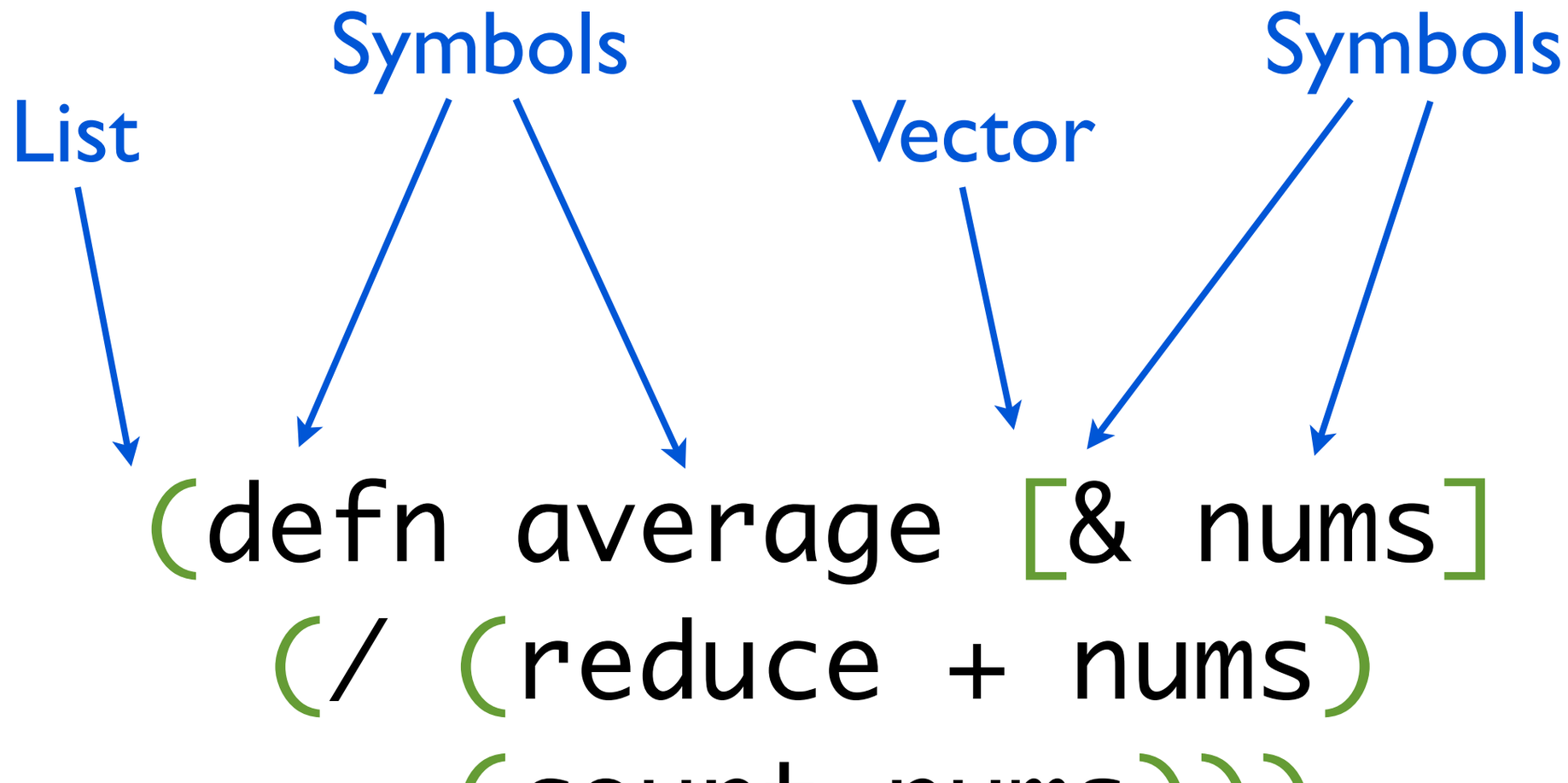
$\log_{32} n$ is fast enough.

Literal Data

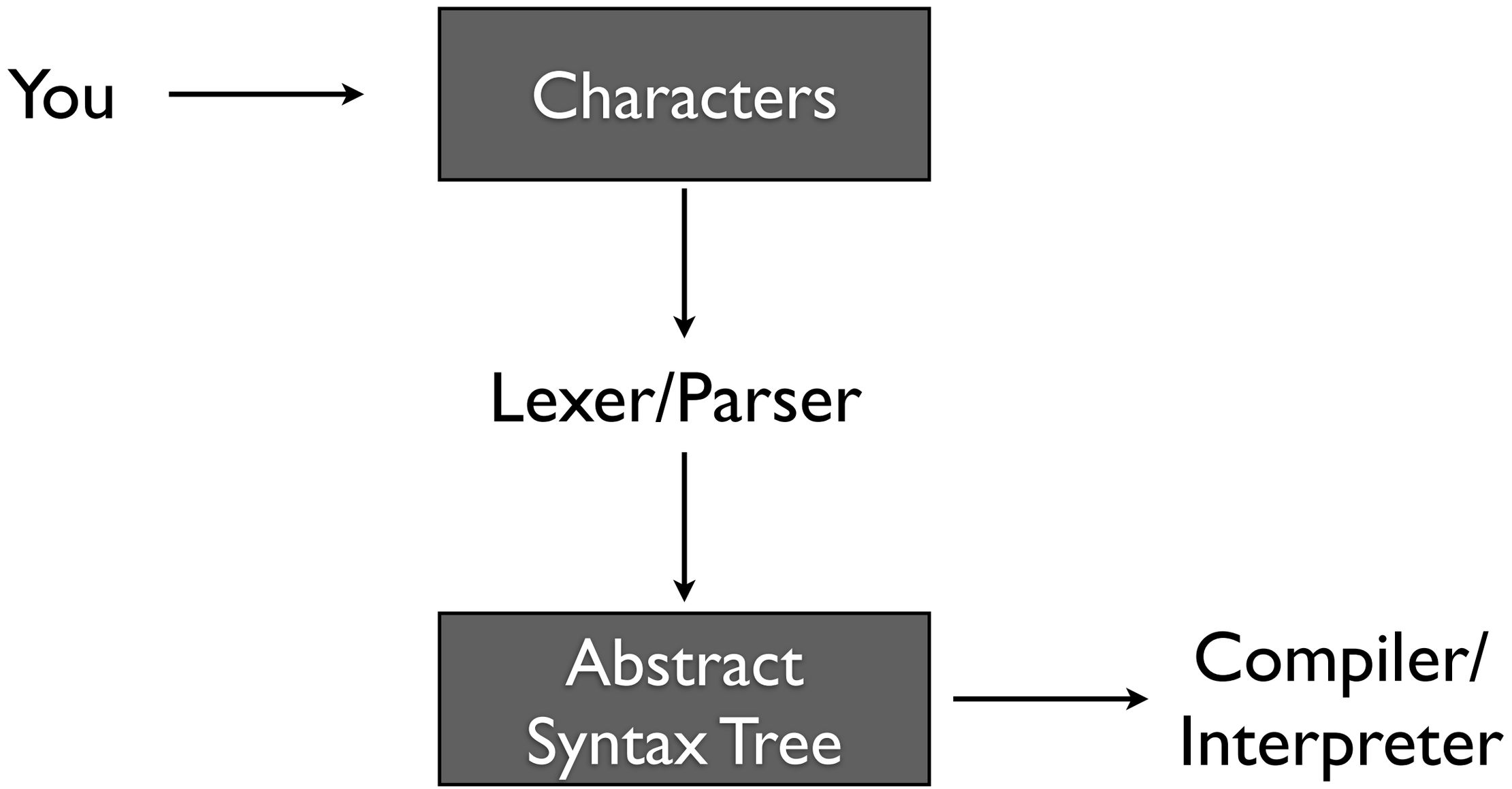
A Set of Maps

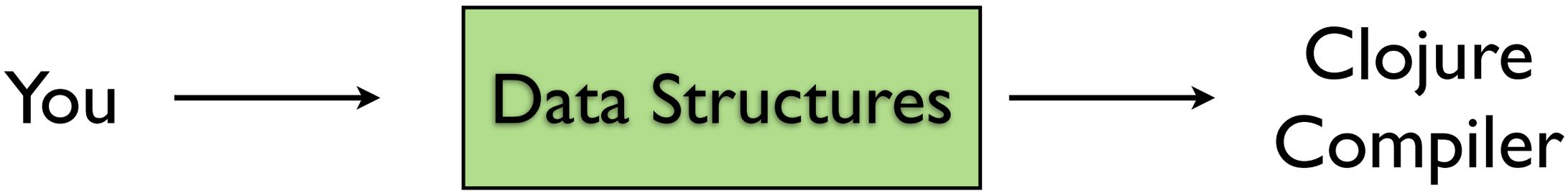
```
#{ {:first "Stuart" :last "Sierra"}
   {:first "Luke"   :last "VanderHart"} }
```


Symbol
List
(+ 3 4 5)


List
Symbols
Vector
Symbols
(defn average [& nums]
(/ (reduce + nums)
(count nums)))

```
(average 3 4 5)
```


You
Characters
Lexer/Parser
Abstract
Syntax Tree
Compiler/
Interpreter


You
Data Structures
Clojure
Compiler

```
(let [file ...initializer...]
  (try ... do something ...
    (finally
      (.close file))))
```

```clojure
(defmacro with-open [bindings & body]
  `(let ~bindings
     (try ~@body
       (finally
         (.close ~(first bindings))))))
```

```
(let [file ...initializer...]
  (try ... do something ...
    (finally
      (.close file))))
```

```
(with-open [file ...initializer...]
  ... do something ...)
```

Metaprogramming

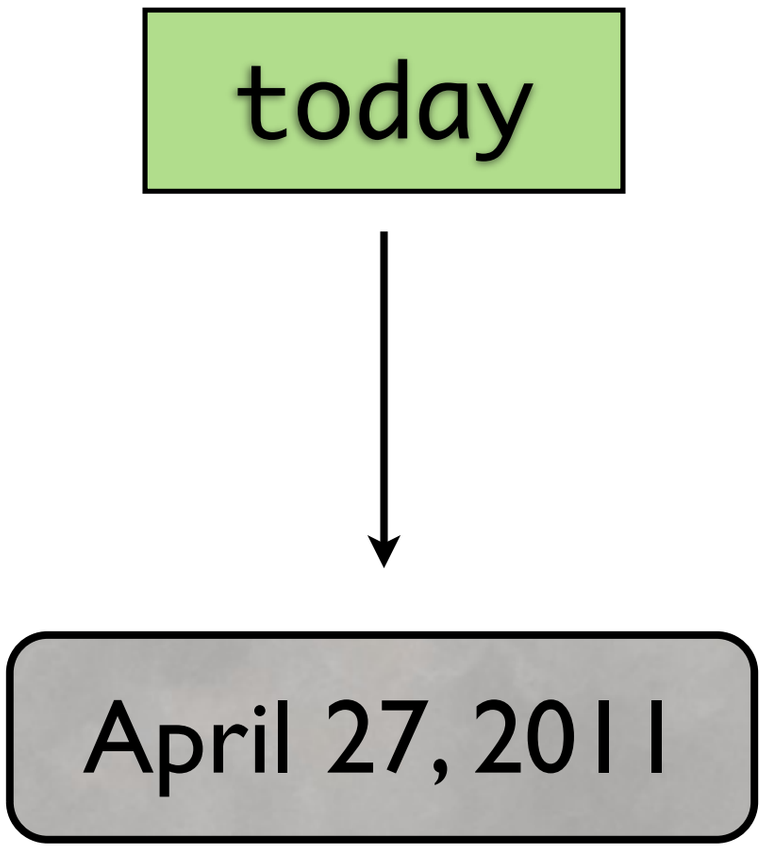
Values

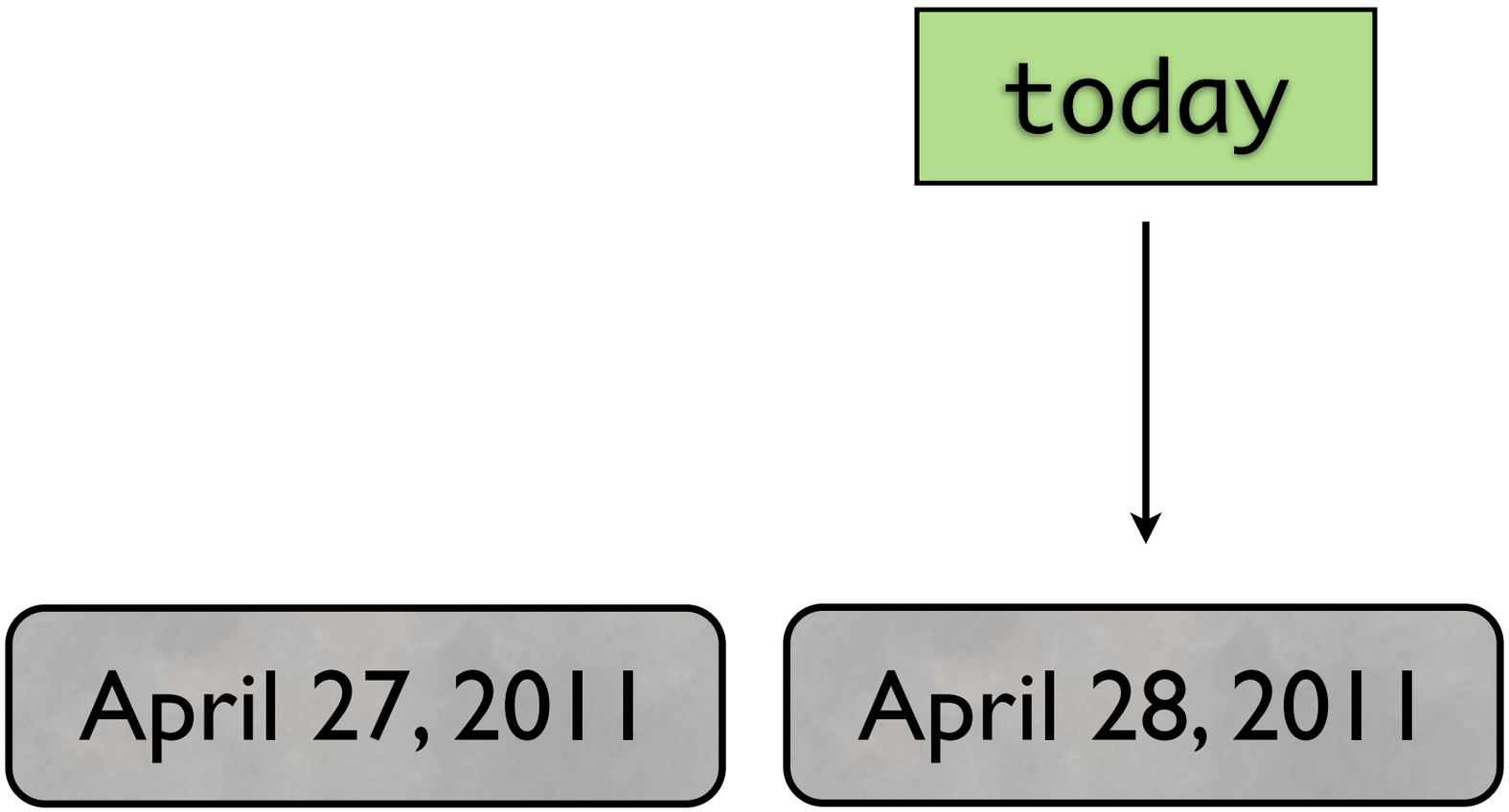
- Immutable

```
class Date {
  public void setDate(int date);
  public void setMonth(int month);
  public void setYear(int year);
}
```

Mutable!

Programming with immutable values means never having to say you're sorry.


today
April 27, 2011


today
April 27, 2011
April 28, 2011

Identity

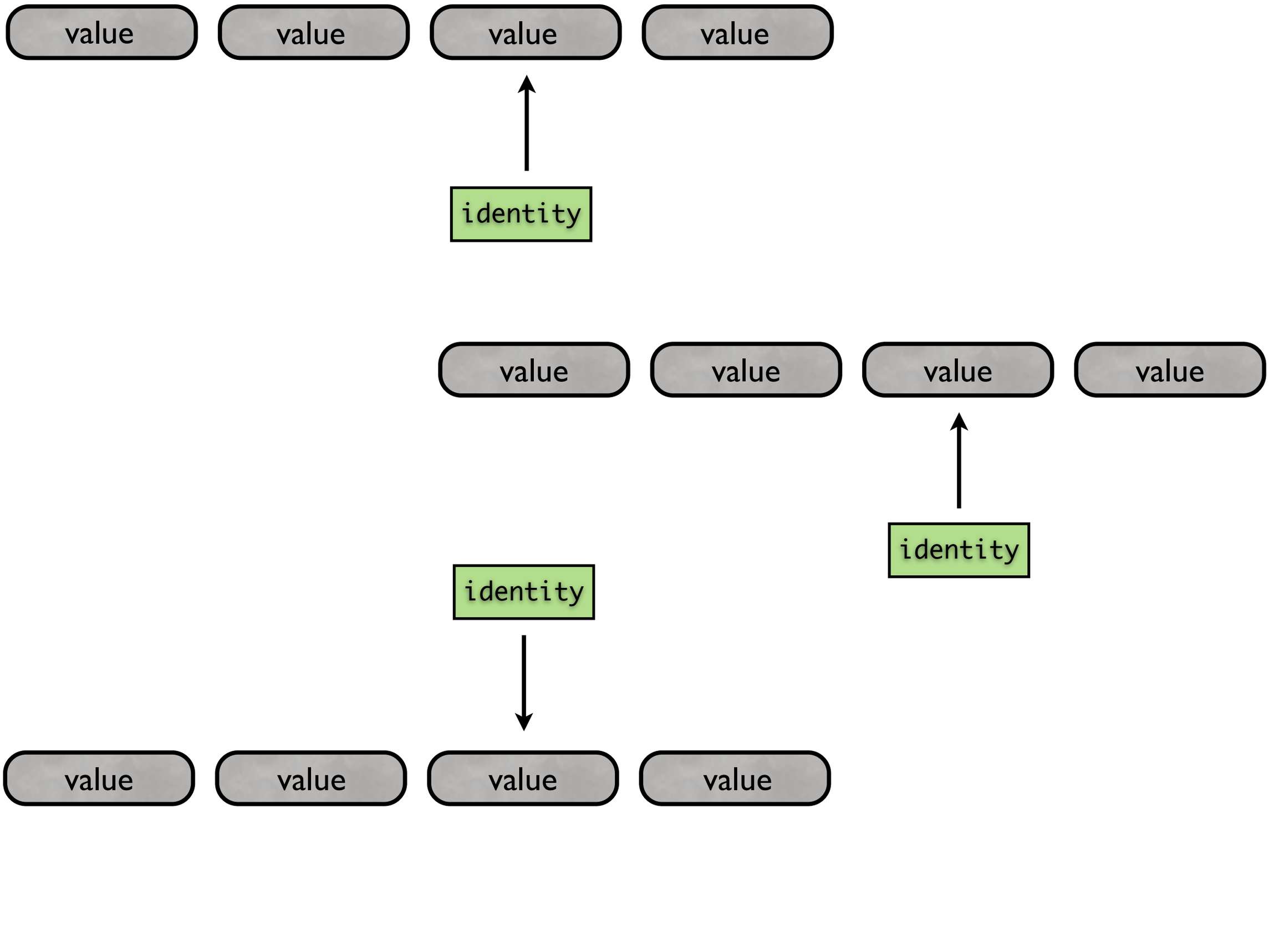
today

State

April 27, 2011

April 28, 2011

Values


value
value
value
value
identity
value
value
value
value
identity
identity
value
value
value
value

Database Transactions

Atomic
Consistent
Isolated
Durable

Clojure Transactions

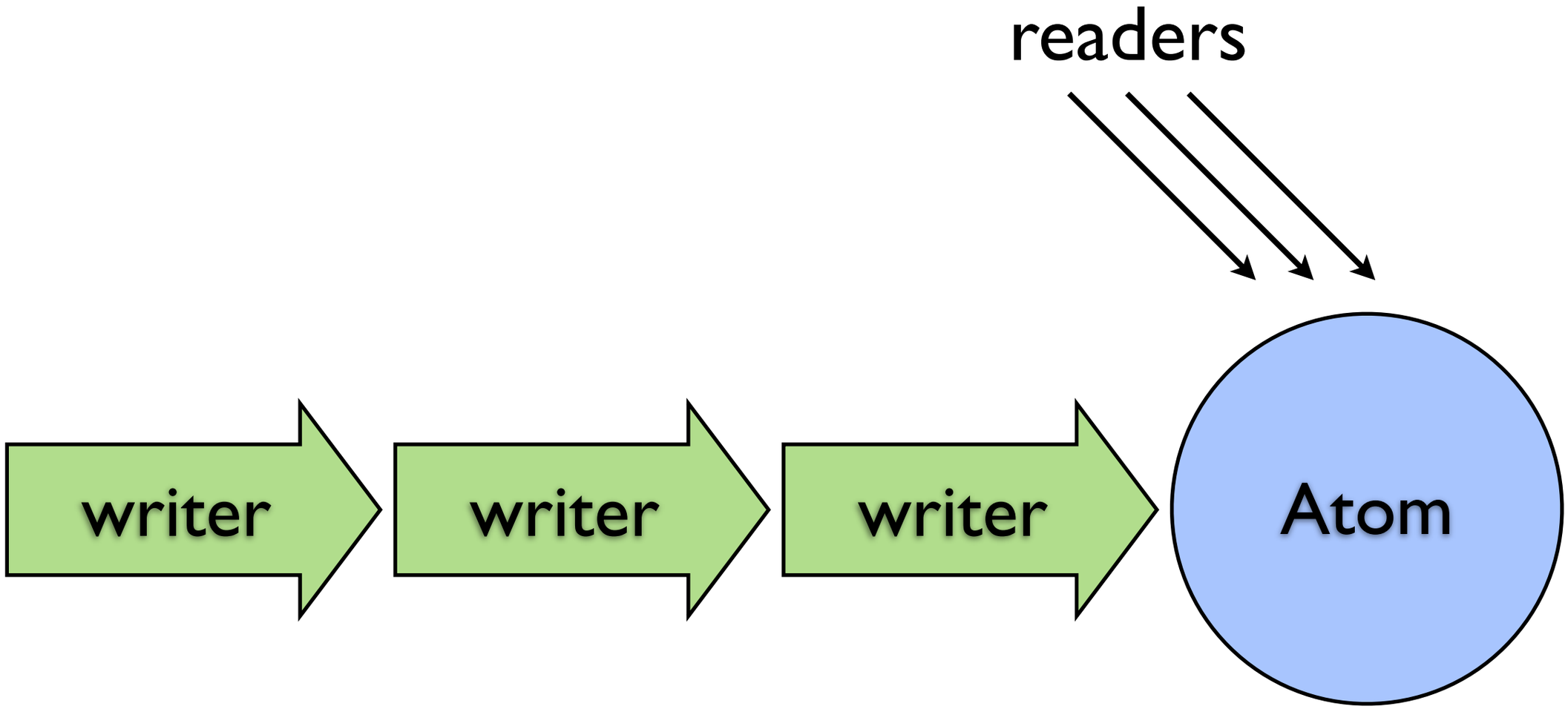
Atomic

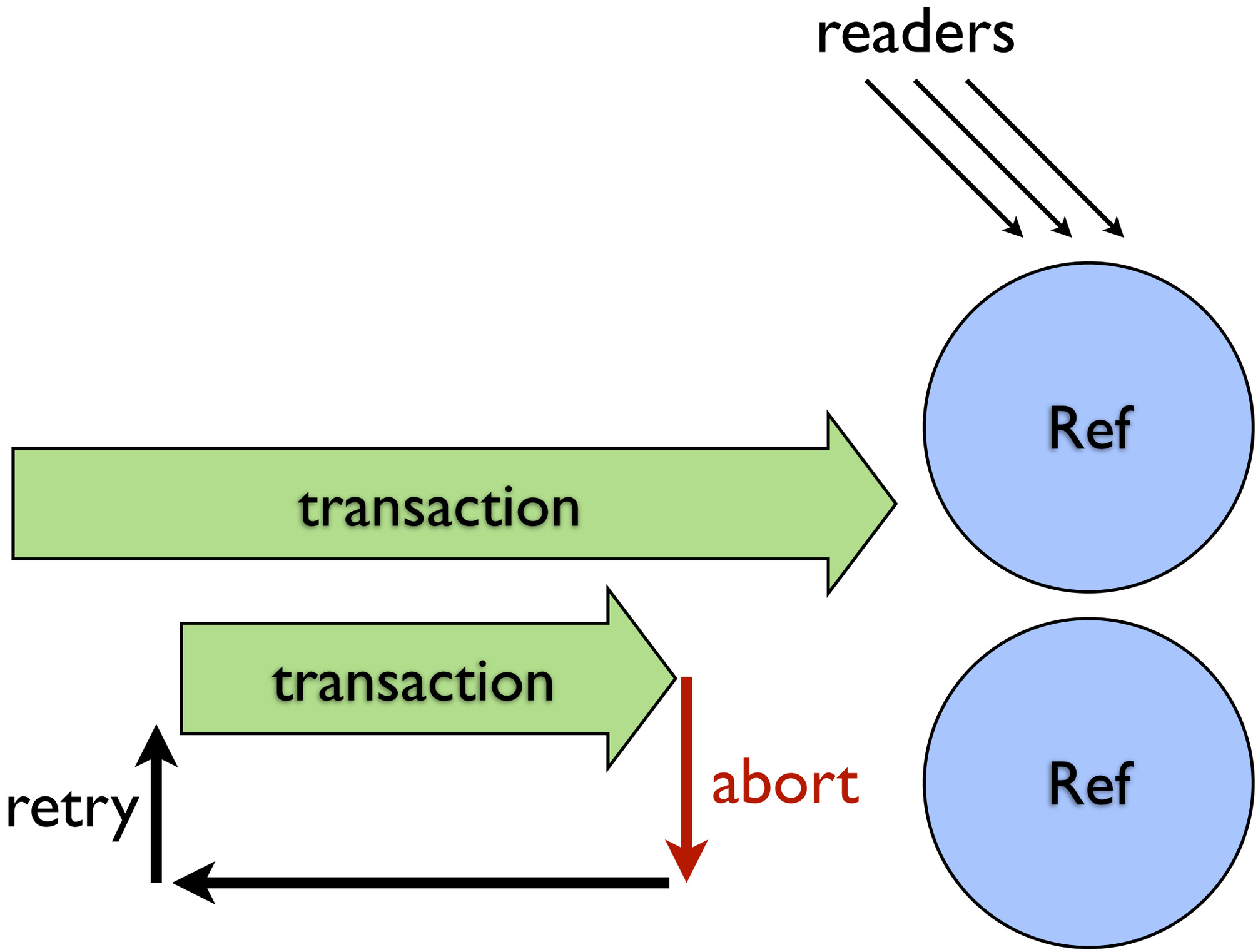
Consistent

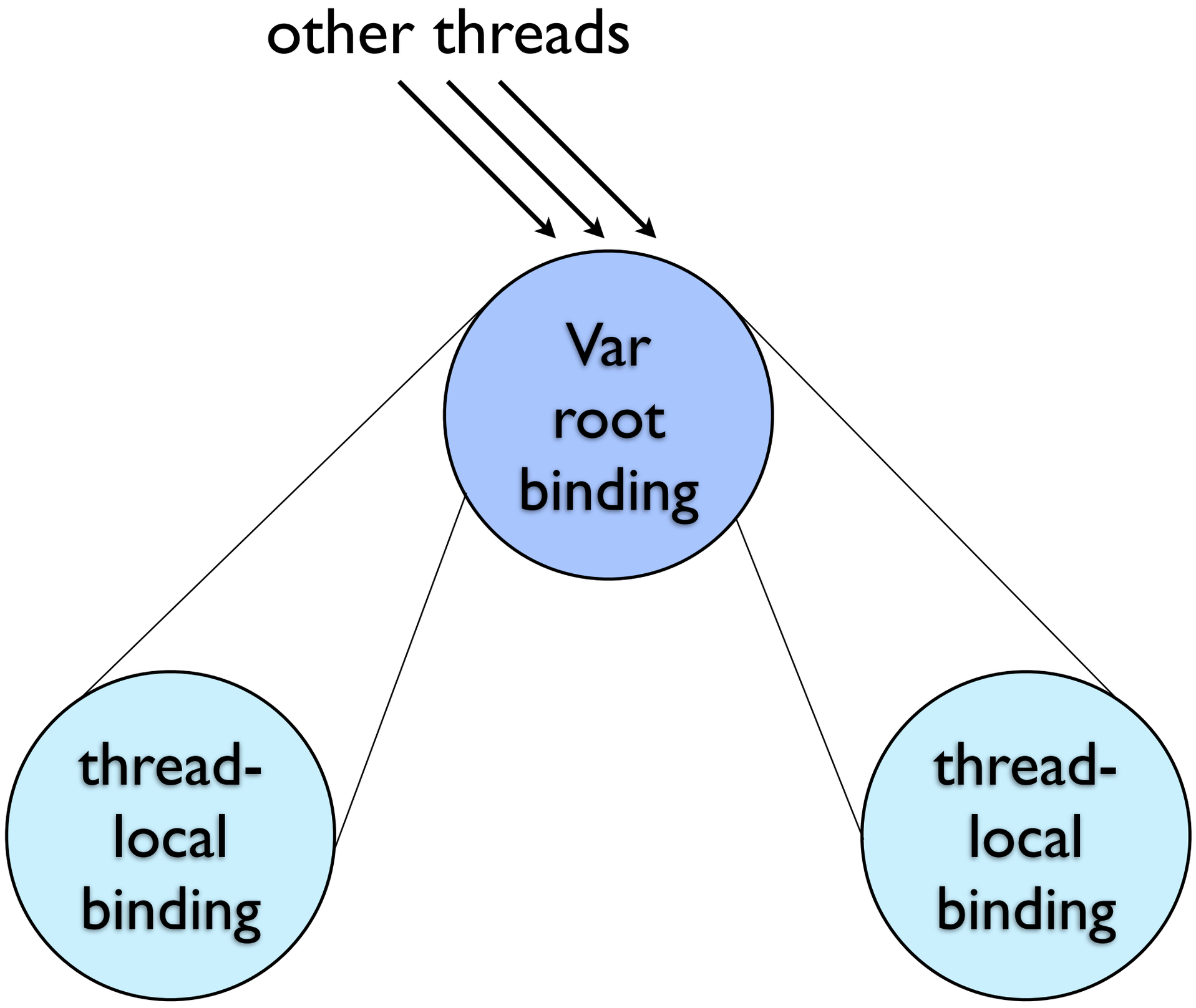
Isolated

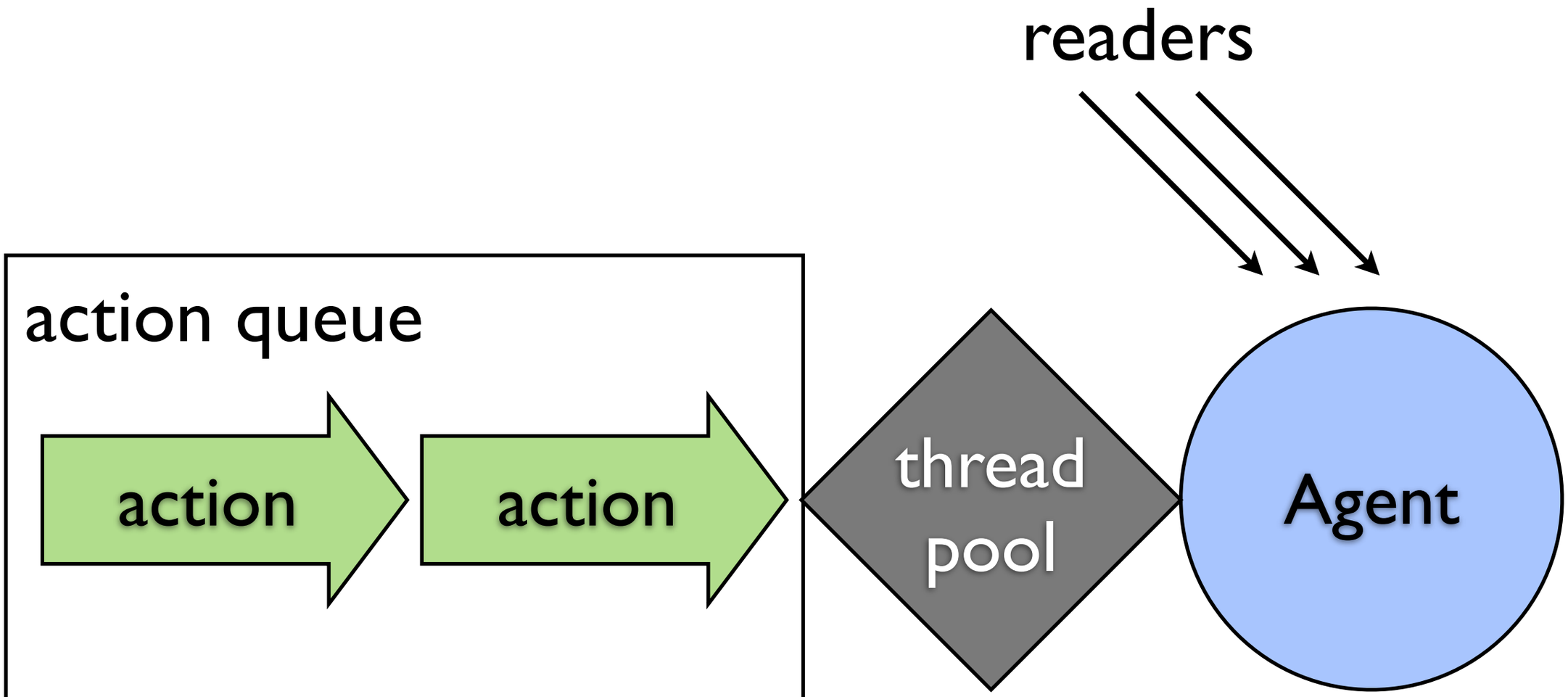
Mutable References

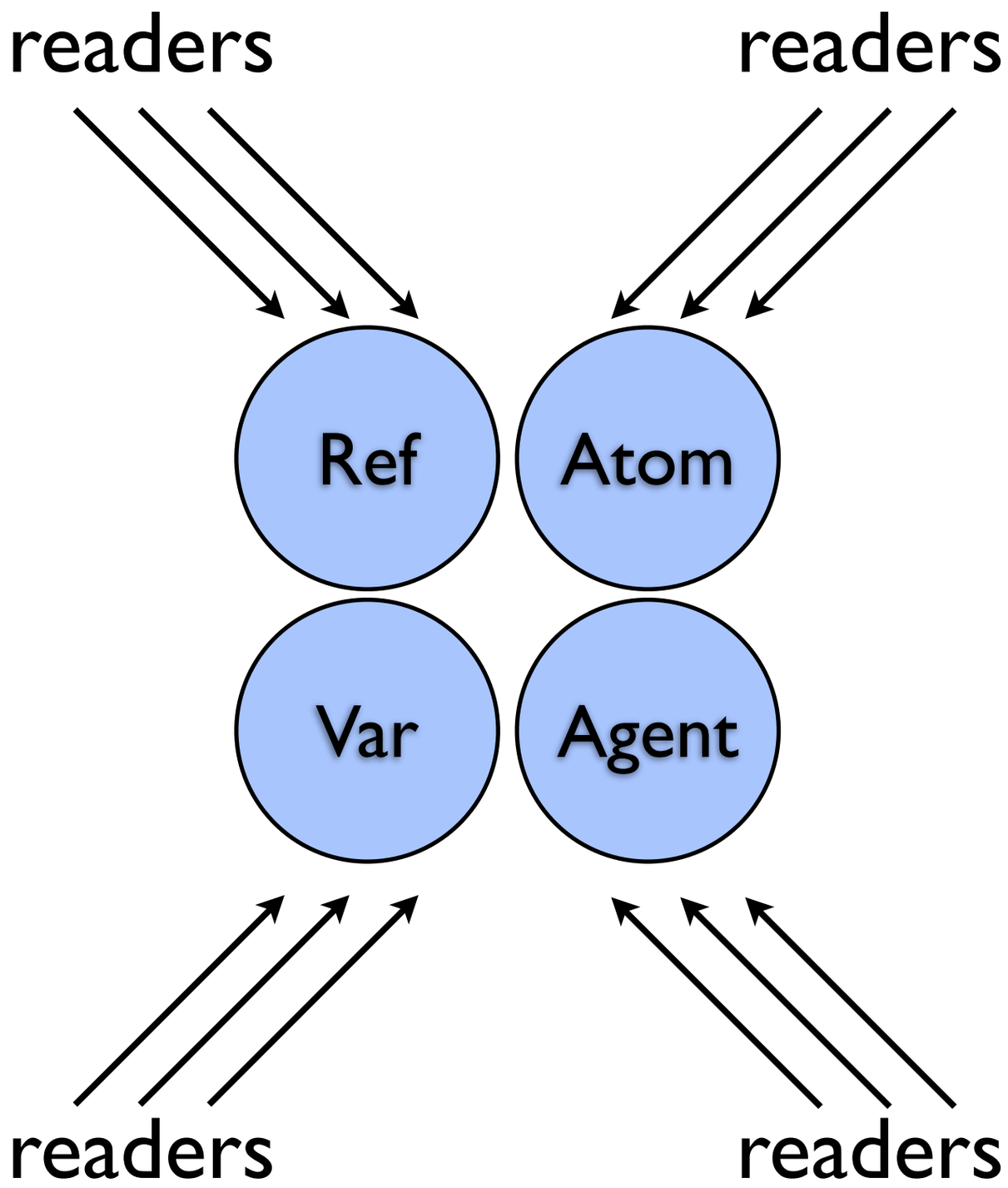
- Atom
- Ref
- Var
- Agent


readers
writer
writer
writer
Atom


readers
Ref
transaction
transaction
retry
abort
Ref

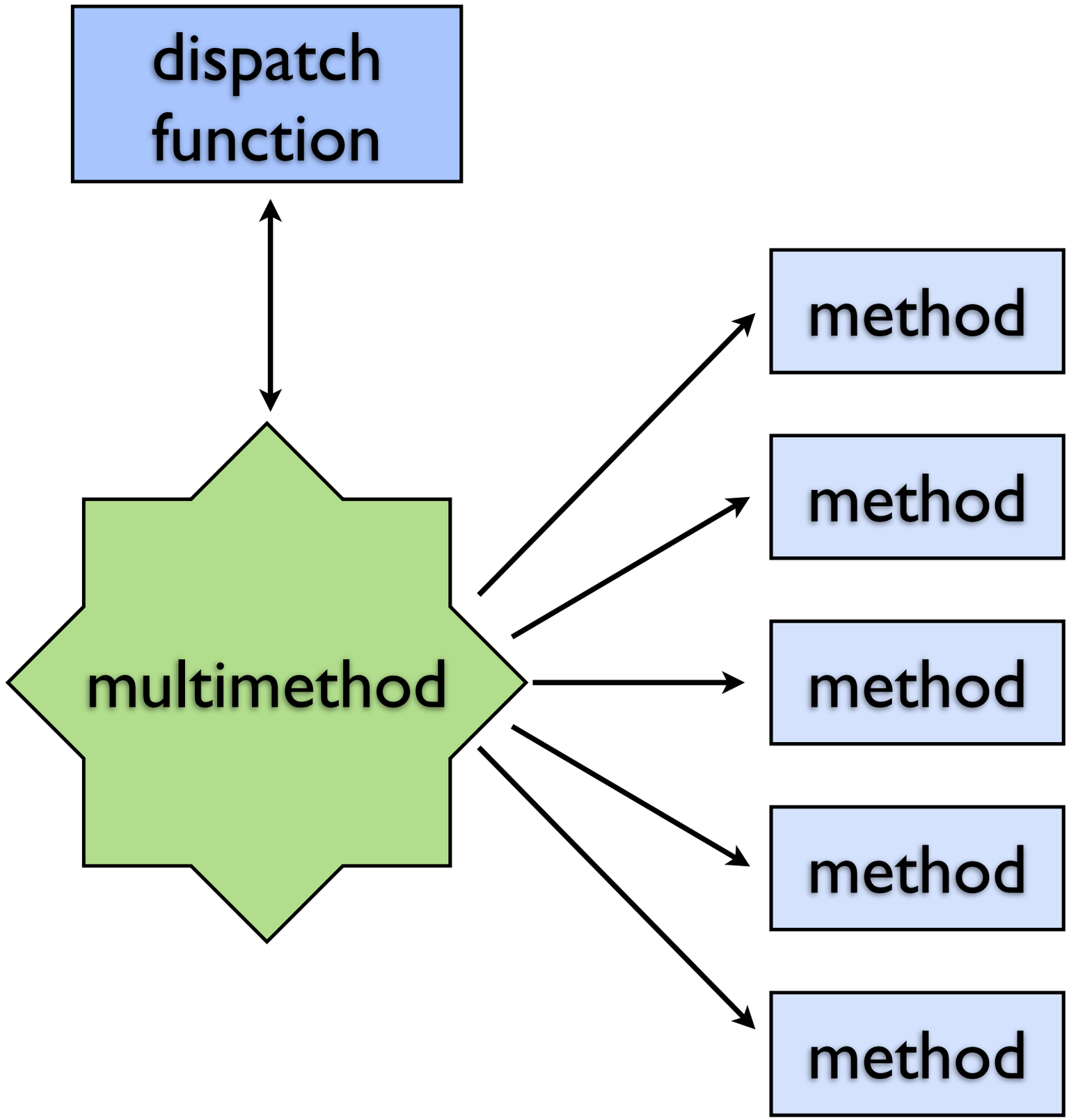

other threads
Var
root
binding
thread-
local
binding
thread-
local
binding


readers
action queue
action
action
thread
pool
Agent

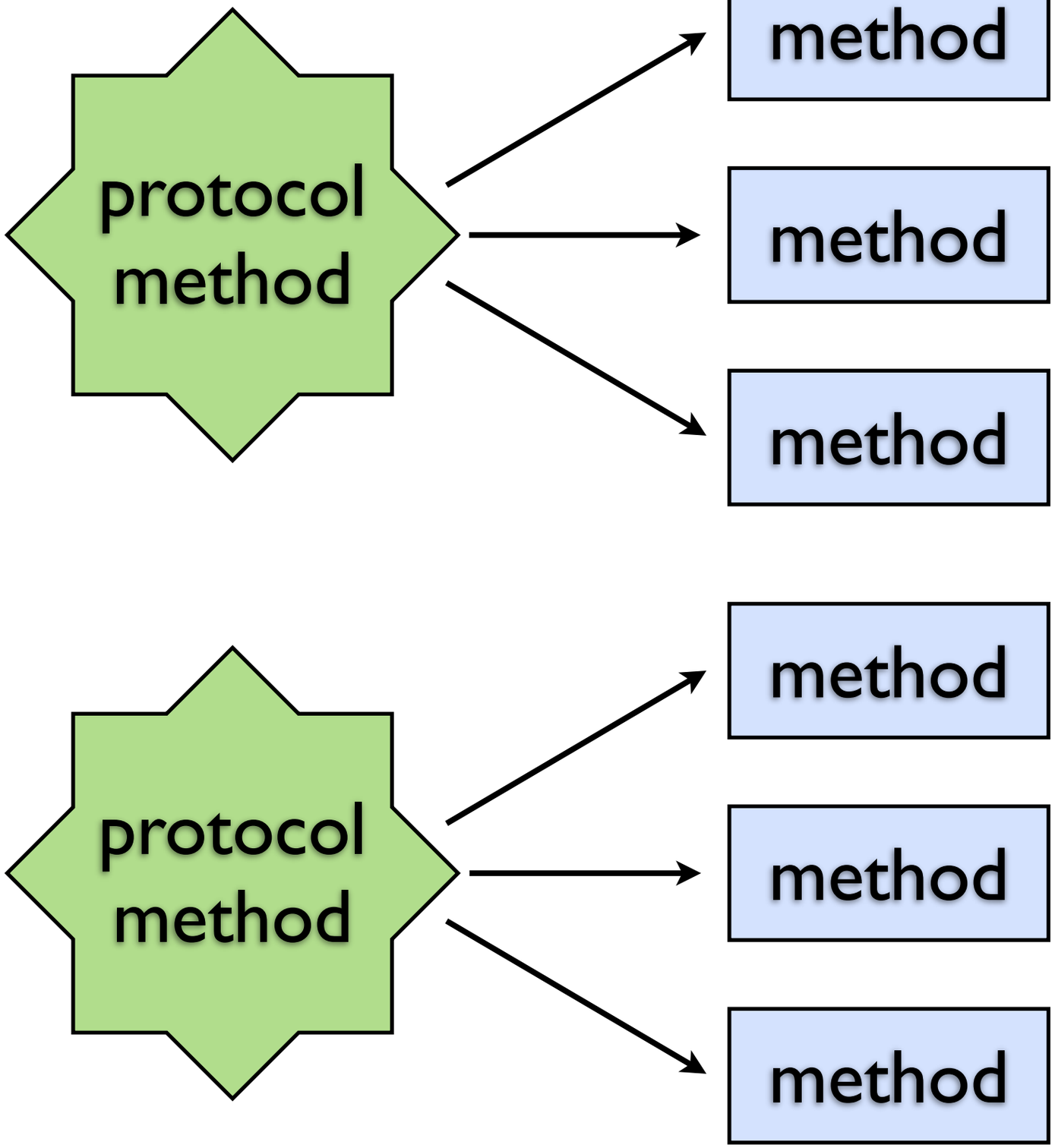

readers
readers
Ref
Atom
Var
Agent
readers
readers

Object-Oriented Programming (The Good Parts)

Polymorphism


dispatch function
multimethod
method
method
method
method
method

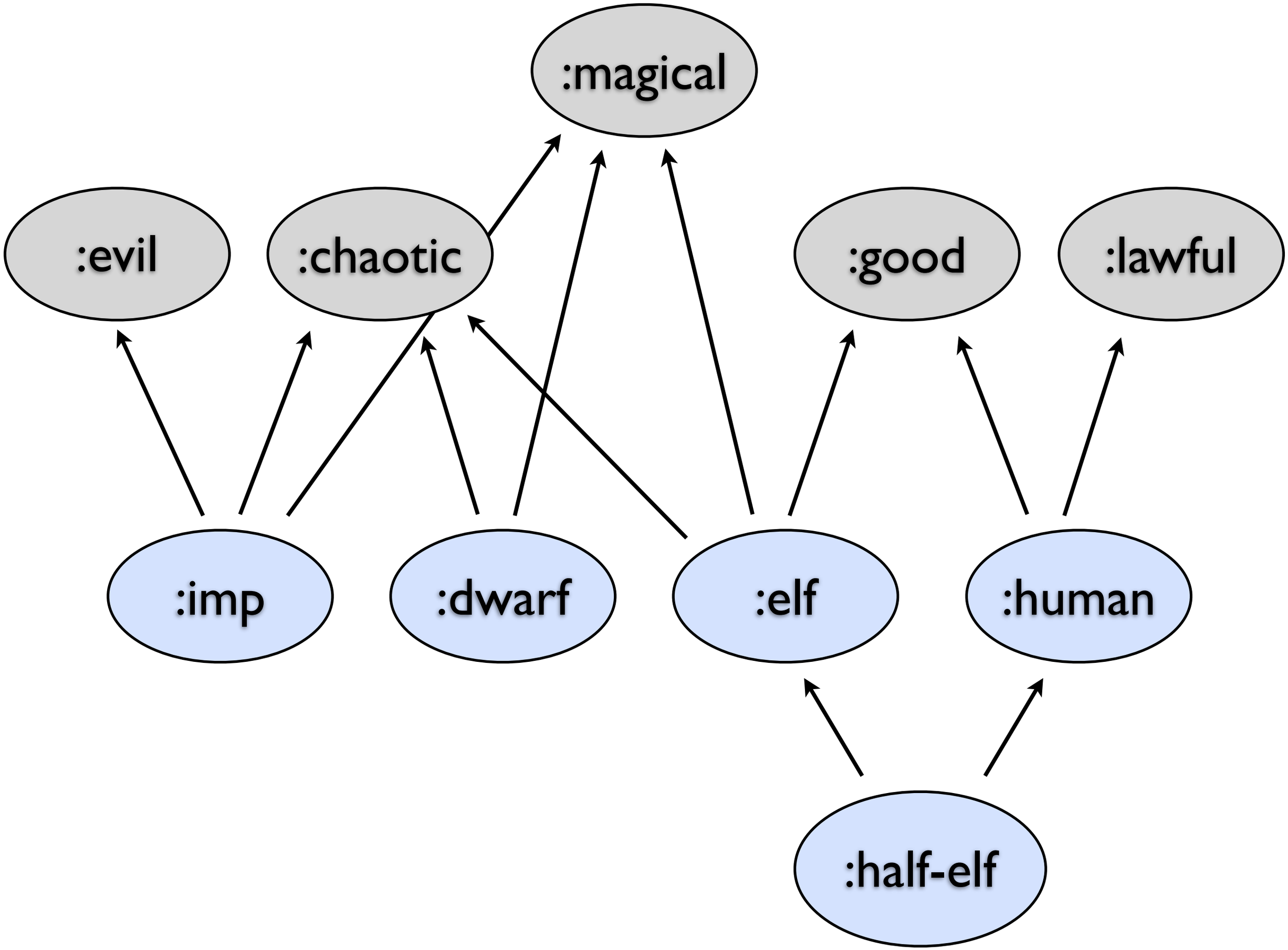
Protocol


protocol
method
method
method
method
protocol
method
method
method

	Existing Interfaces	Your new protocol here
Existing Types	Existing Method Implementations	
Your new type here		and here!

"Solving the Expression Problem with Clojure"

IBM developerWorks
December 2010


:magical
:evil
:chaotic
:good
:lawful
:imp
:dwarf
:elf
:human
:half-elf

Platform

Platform

constructor

```
(def m (ConcurrentHashMap. 100))
```

```
(.put m "key" "value")
```

method call

More

- Clojure: clojure.org
- Clojure/core: clojure.com
- Training: pragmaticstudio.com/clojure

Image Credits

- openclipart.org
- pdtextures.blogspot.com
- Clojure logo by Tom Hickey