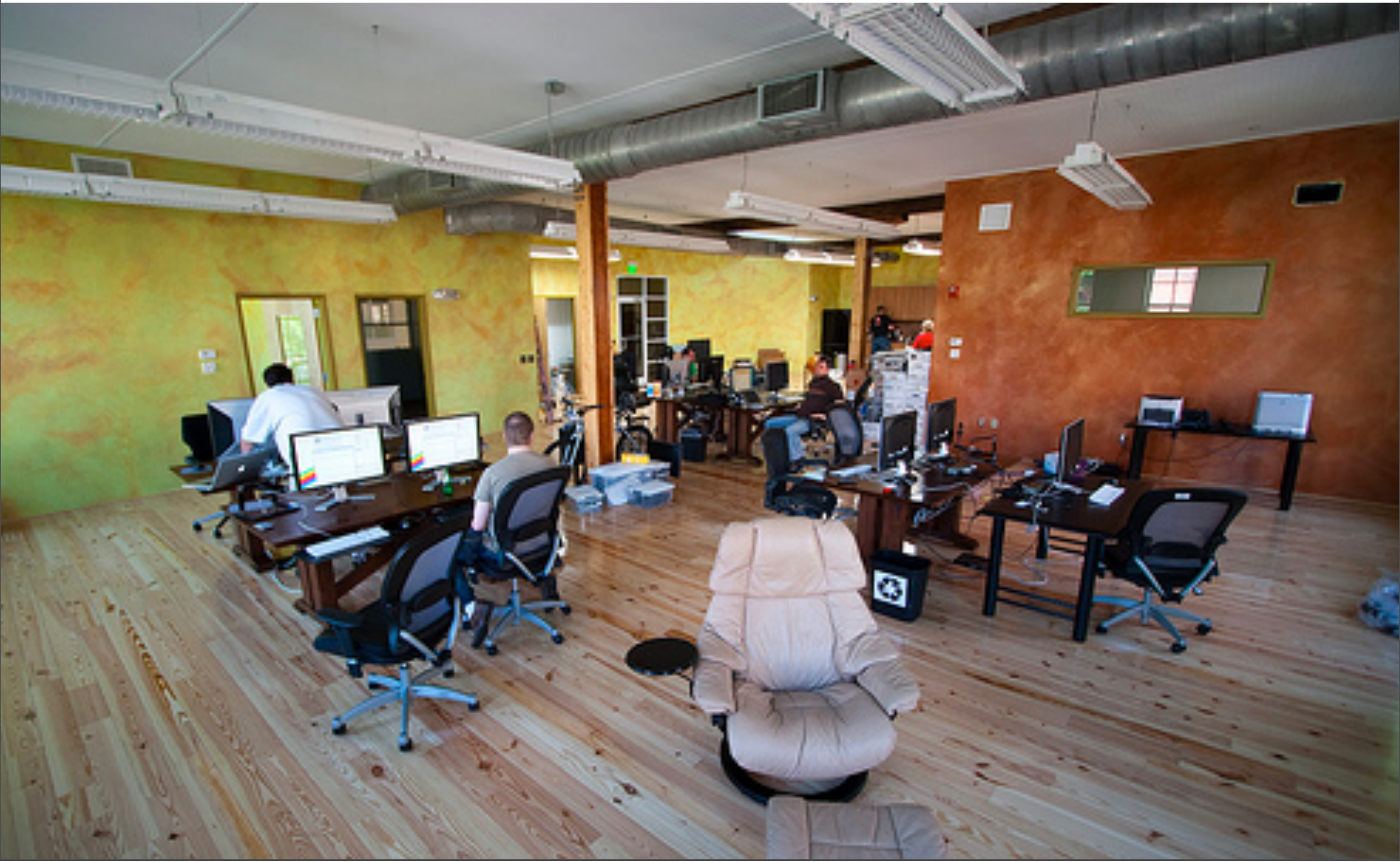
clojure in the field

http://github.com/stuarthalloway/clojure-presentations

stuart halloway
http://thinkrelevance.com

about us



what we are building

rule-based systems

social networking applications

scalable web services

near-real-time simulators

clojure's four elevators

1. lisp
2. java
3. functional
4. state

1.lisp

feature	industry norm	cool kids	lisp
conditionals	✔	✔	✔
variables	✔	✔	✔
garbage collection	✔	✔	✔
recursion	✔	✔	✔
function type		✔	✔
symbol type		✔	✔
whole language available		✔	✔
everything's an expression			✔
homoiconicity			✔

http://www.paulgraham.com/diff.html

2. java interop

java new

java	`new Widget("foo")`
clojure	`(new Widget "foo")`
clojure sugar	`(Widget. "red")`

access static members

java	`Math.PI`
clojure	`(. Math PI)`
clojure sugar	`Math.PI`

access instance members

java	`rnd.nextInt()`
clojure	`(. rnd nextInt)`
clojure sugar	`(.nextInt rnd)`

atomic data types

type	example	java equivalent
string	`"foo"`	String
character	`\f`	Character
regex	`#"fo*"`	Pattern
a. p. integer	`42`	Integer/Long/BigInteger
double	`3.14159`	Double
a.p. double	`3.14159M`	BigDecimal
boolean	`true`	Boolean
nil	`nil`	`null`
symbol	`foo, +`	N/A
keyword	`:foo, ::foo`	N/A

3. functional

imperative style

```java
public class StringUtils {
  public static boolean isBlank(String str) {
    int strLen;
    if (str == null || (strLen = str.length()) == 0) {
      return true;
    }
    for (int i = 0; i < strLen; i++) {
      if ((Character.isWhitespace(str.charAt(i)) == false)) {
        return false;
      }
    }
    return true;
  }
}
```

functional style

```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```

data literals

type	properties	example
list	singly-linked, insert at front	`(1 2 3)`
vector	indexed, insert at rear	`[1 2 3]`
map	key/value	`{:a 100 :b 90}`
set	key	`#{:a :b}`

persistent data structures

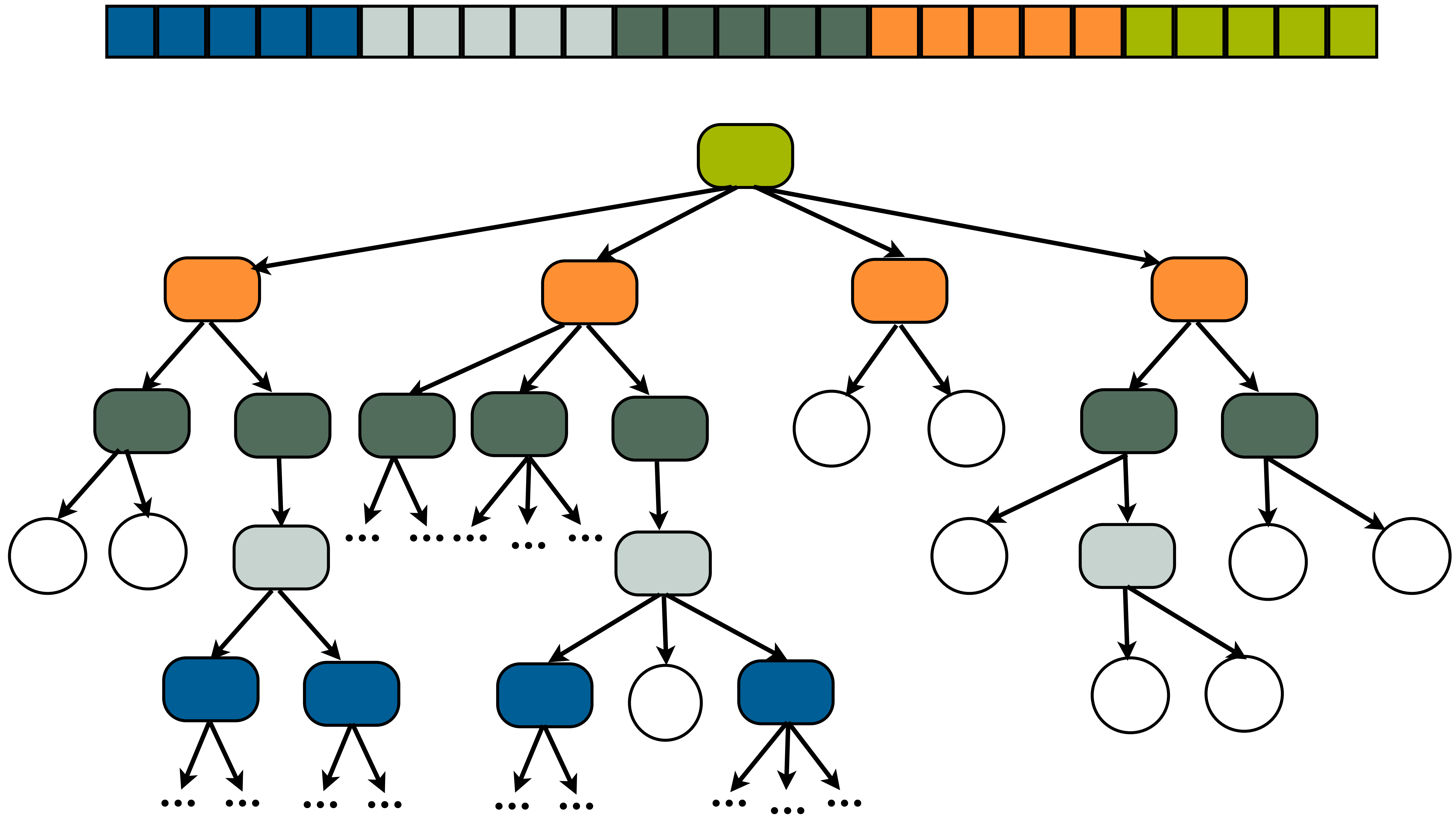
immutable

“change” by function application

maintain performance guarantees

full-fidelity old versions

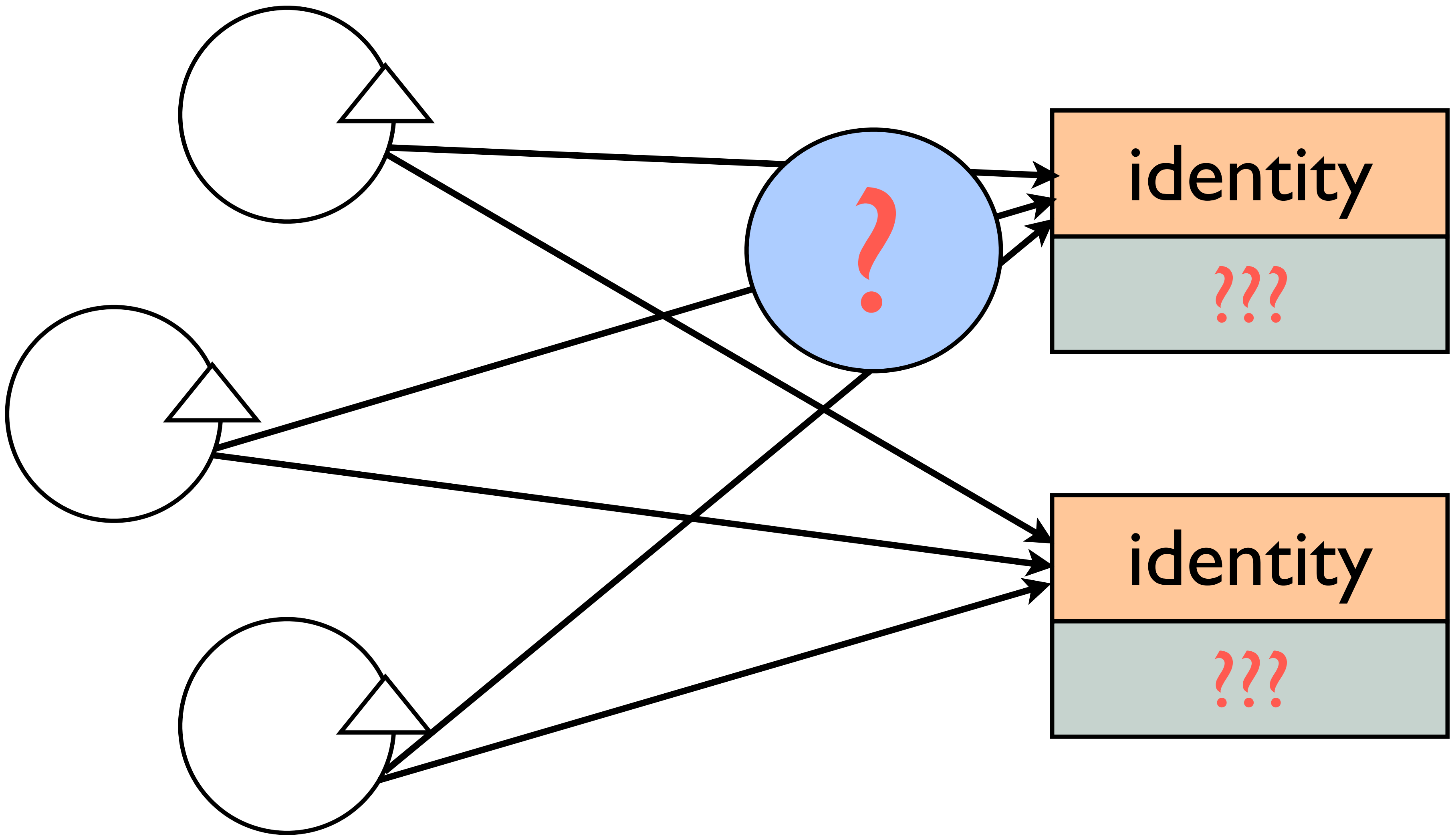
32-way tries



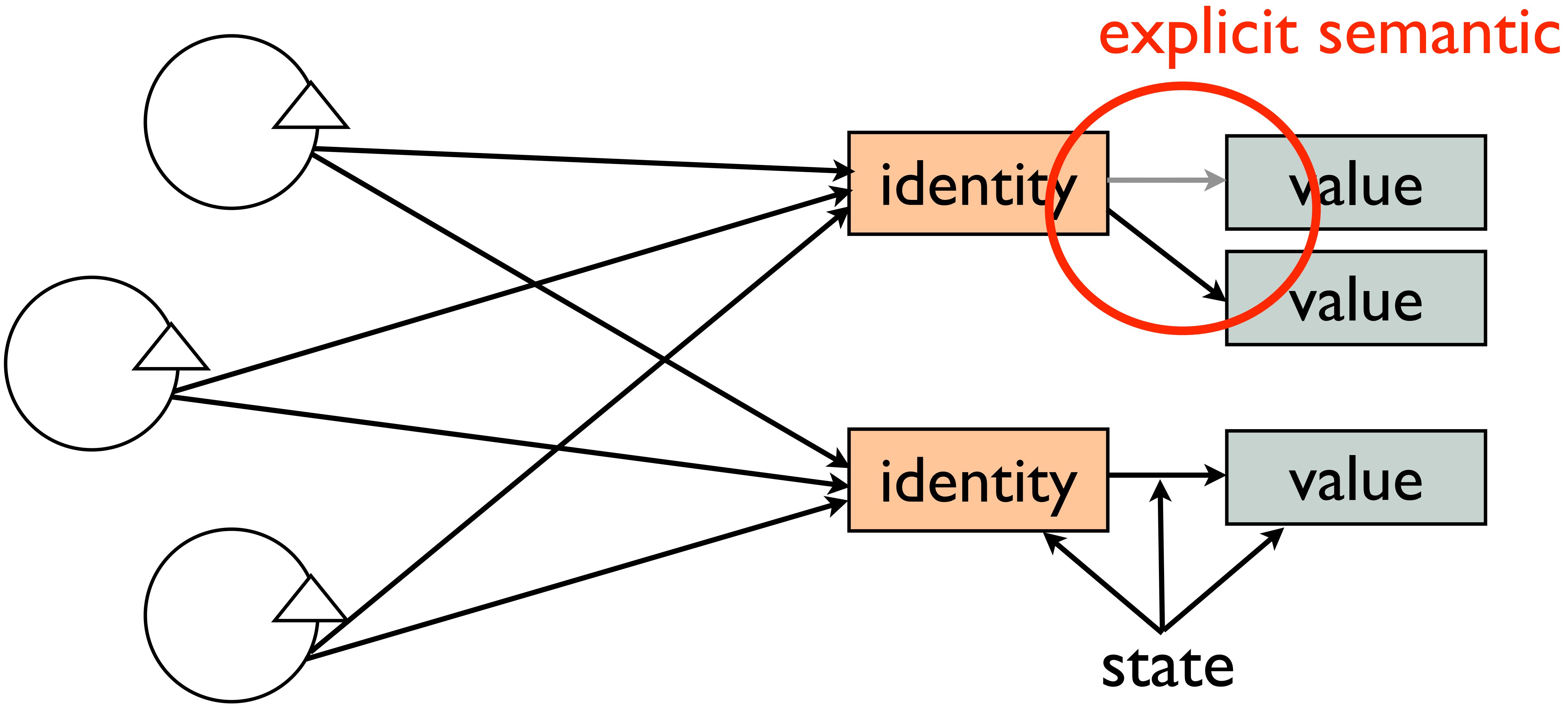
clojure: ‘cause log32 n is fast enough!

4. state ~~concurrency~~

mutable oo



clojure


explicit semantic
identity
value
value
identity
value
state

terms

1. **value:** immutable data in a persistent data structure

2. **identity:** series of causally related values over time

3. **state:** identity at a point in time

ref example: chat

identity

```
(def messages (ref ()))
```

initial value

updating

apply an...

```
(defn add-message [msg]
  (dosync (alter messages conj msg)))
```

scope a transaction

...update fn

bdd meets fp

clojure.test

```
(deftest test-rules
  (are [result boardstr]
    (= result
       (apply rules (str->board boardstr)))
       :dying  "...
                .O.
                ..."

       :off    "O.O
                ...
                O.O"

       :on     "|||
                O.O
                |||"))
```

to wrap or not to wrap

files or strings?

```
(java.io.File. "foo")
-> #<File foo>
```

```
(as-file "foo")
-> #<File foo>
```

the learning curve

mitigating learning curve

pairing

open source fridays

mailing list, irc

libraries

contribs you need!

contrib	usage
ns-utils	explore namespaces
pprint	human friendly data printing
repl-utils	javadoc, show, source
seq-utils	extend the sequence uberlibrary
shell-out	call the host OS
str-utils	strings *and regular expressions*

other libs

compojure

clojure.http.client

redis-clojure

incanter

clj-facebook

clj-mql

clj-record

java libs

jline

joda-time

stringtemplate

supercsv

cohesion

shipping it

deployment

- today
  - capistrano
  - chef
  - contegix, slicehost, ec2
- future
  - "clojure chef"?
  - contegix, zeus, slicehost, ec2

pain points

test automation

~~maven~~ java ecosystem

convention over configuration

error messages

living without objects

editor support

pleasure points

libraries

readability

destructuring

metadata

multimethods

macros

reference types

data conversion

repl

paredit

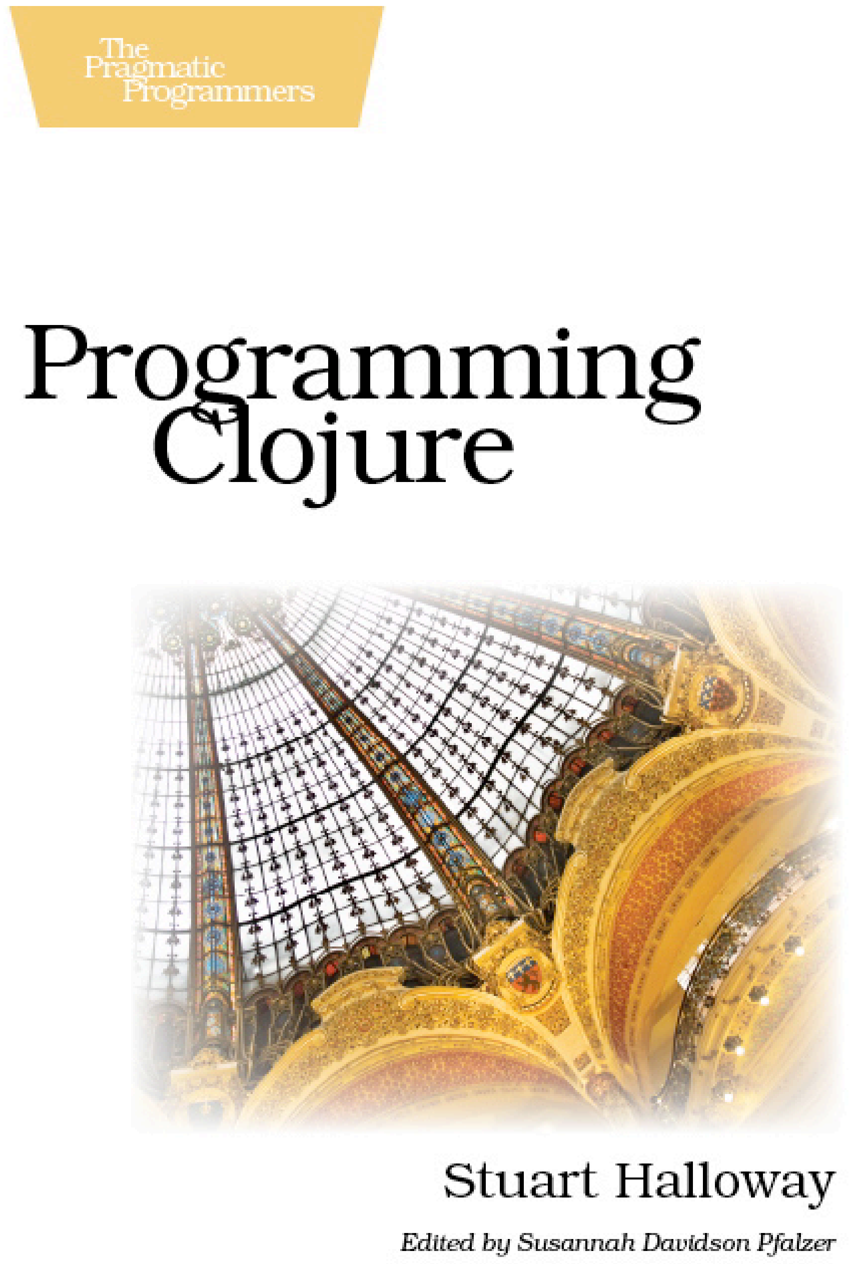
namespaces

composability

java interop

we have removed clojure from the risk checklist on new projects

Email:	stu@thinkrelevance.com
Office:	919-442-3030
Twitter:	twitter.com/stuarthalloway
Facebook:	stuart.halloway
Github:	stuarthalloway
This talk:	http://github.com/stuarthalloway/clojure-presentations
Talks:	http://blog.thinkrelevance.com/talks
Blog:	http://blog.thinkrelevance.com
Book:	http://tinyurl.com/clojure


The Pragmatic Programmers
Programming Clojure
Stuart Halloway
Edited by Susannah Davidson Pfalzer