Kilim

Isolation-typed actors for Java

Sriram Srinivasan

University of Cambridge, Opera Group
Sriram.Srinivasan@cl.cam.ac.uk

http://kilim.malhar.net

background

- processors increasingly distributed

  Multiple cores, processors, boxes, data centers

- isolation
- concurrency: how to get safety *and* speed?

  memory-models and consistency

  problems with shared memory concurrency

  incorporating external actions

- server applications are data-flow networks

communicating sequential procs

- message passing everywhere
  - shared nothing == failure isolation
  - easy to reason about, debug
  - unified view of concurrent and distributed
- lightweight threads
  - automatic stack management
  - maps to user-level concurrent tasks

objections to message passing

- async programming is hard
  - verbose, inversion of control
- heavyweight threads
- message passing is expensive
  - copying
  - context switching

Kilim

- Ultra-lightweight threads
- Message-passing
- Messages distinct from objects
  - No internal aliasing
  - Linear ownership transfer
- Safe, zero-copy message passing
- Run-time library (scheduler, typed mailboxes, timer)

Programming model

kilim tasks

```
class HttpConn  extends  Task {
    @pausable
    public void execute() {
        while (true) {
            HttpMsg m = readReq();
            processMsg(m);
        }}

        @pausable
        public HttpMsg readReq() {
            ....
        }
    }

new HttpConn(mbox).start();
```

mailboxes for messaging

```
class MyTask extends  Task {
    Mailbox<Msg> mb, outmb;
    public @pausable void execute() {
        while (true) {
            Msg m = mb.get();
            process(m);
            outmb.put(m);
        }
    }
}
```

weaving

- java kilim.tools.Weaver -d ./classes HttpConn
- -or-
- java kilim.tools.Weaver -d ./classes ./classes

Internals

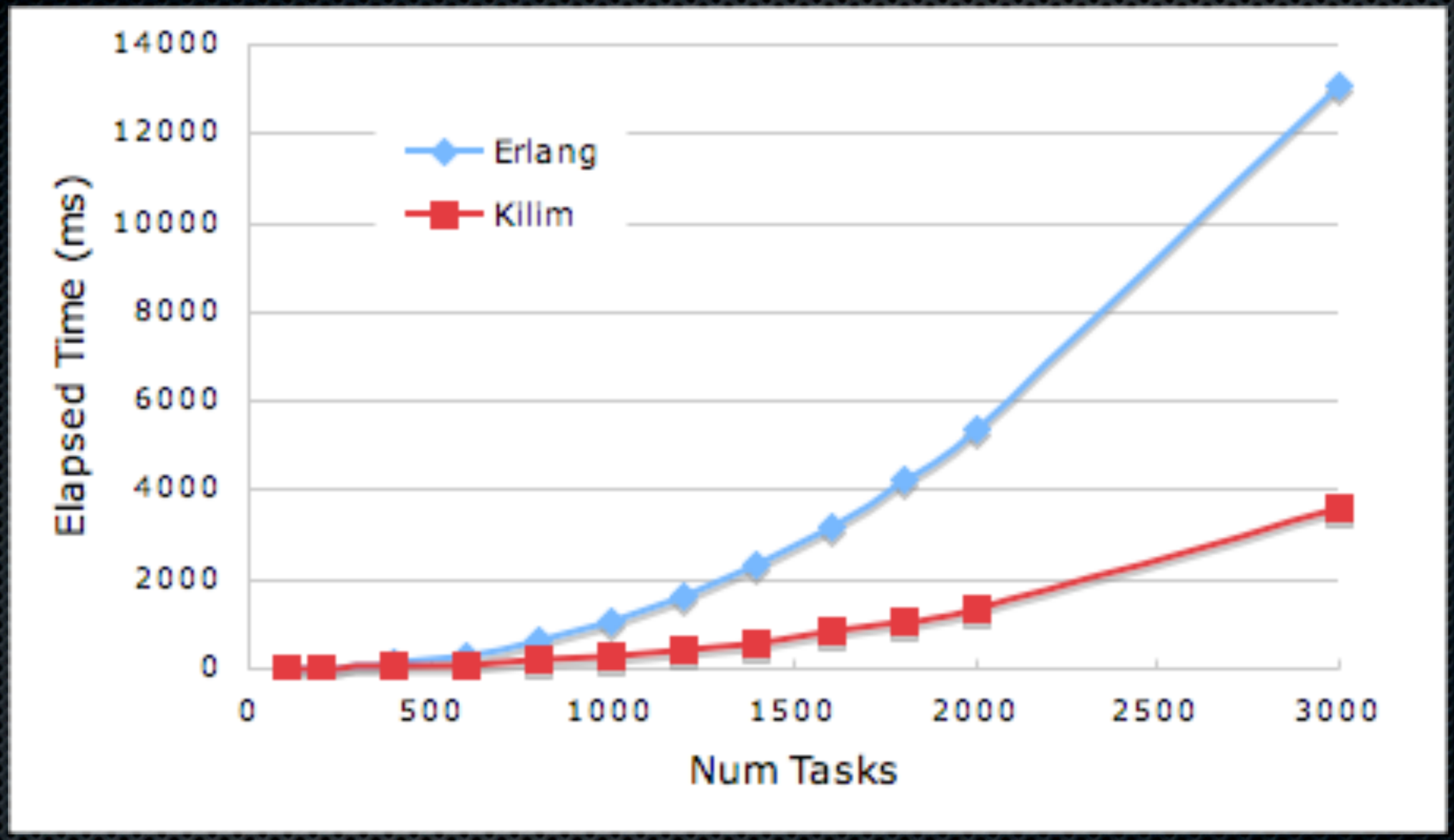
CPS transformation

```
public @pausable void foo(Object o ) {
    for (int i = .... ) {
        bar(o);
        print(i, o);
    }
}}
```

CPS transformation

```
public @pausable void foo(Object o, Fiber f) {
    goto f.pc;
    for (int i = ... ) {
        L1:
            bar(o, f);
            if f.isPausing
                f.store:   pc=L1, i, o
                return
            else if f.needsRestoring
                f.restore o, i
            print(i, o);
    }
}}
```

n tasks, n^2 messages



task creation

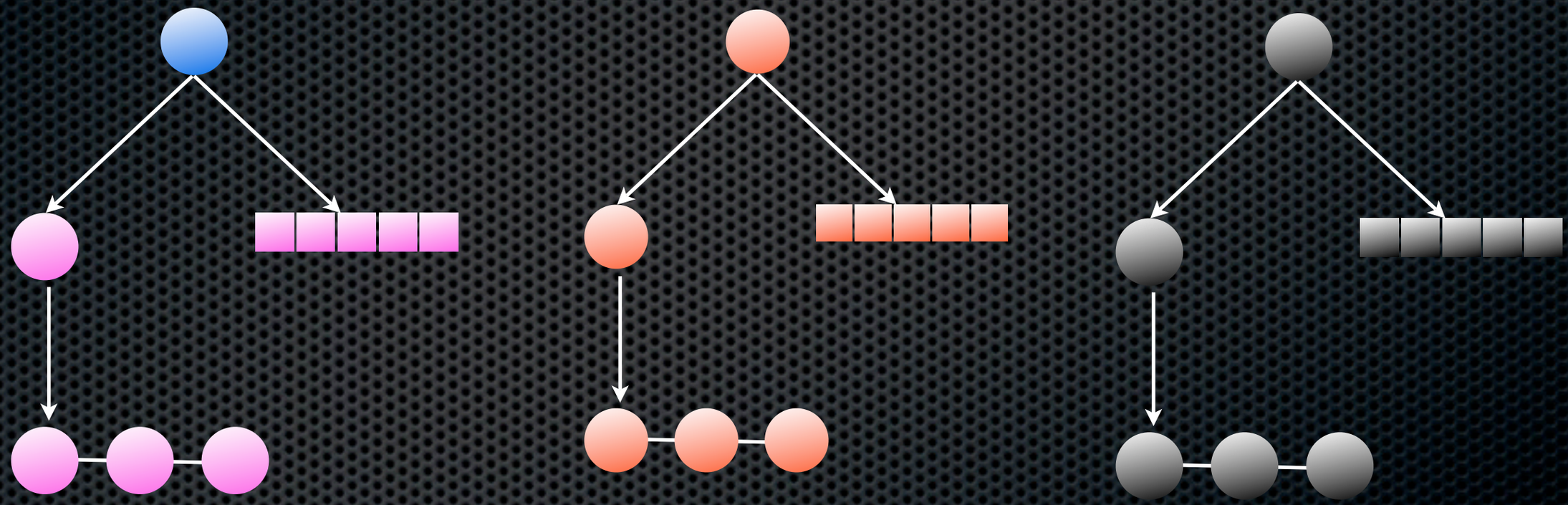
options for message safety

- Immutable messages
- copies
- locks (?)
- linear type systems
- ownership types

mutable messages in Kilim

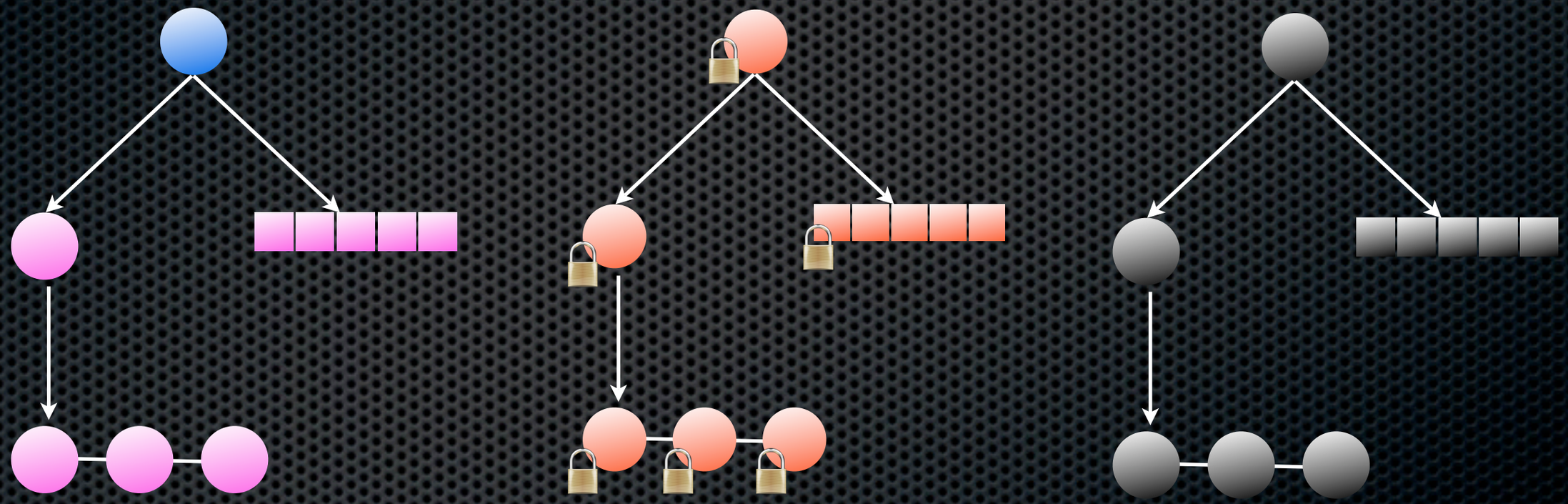
- philosophically different from objects
  - Implement **`Message`**
  - Primitives, refs to **`Message`**, arrays
- no internal aliasing allowed
- public structure encouraged
- ownership passed linearly

capabilities



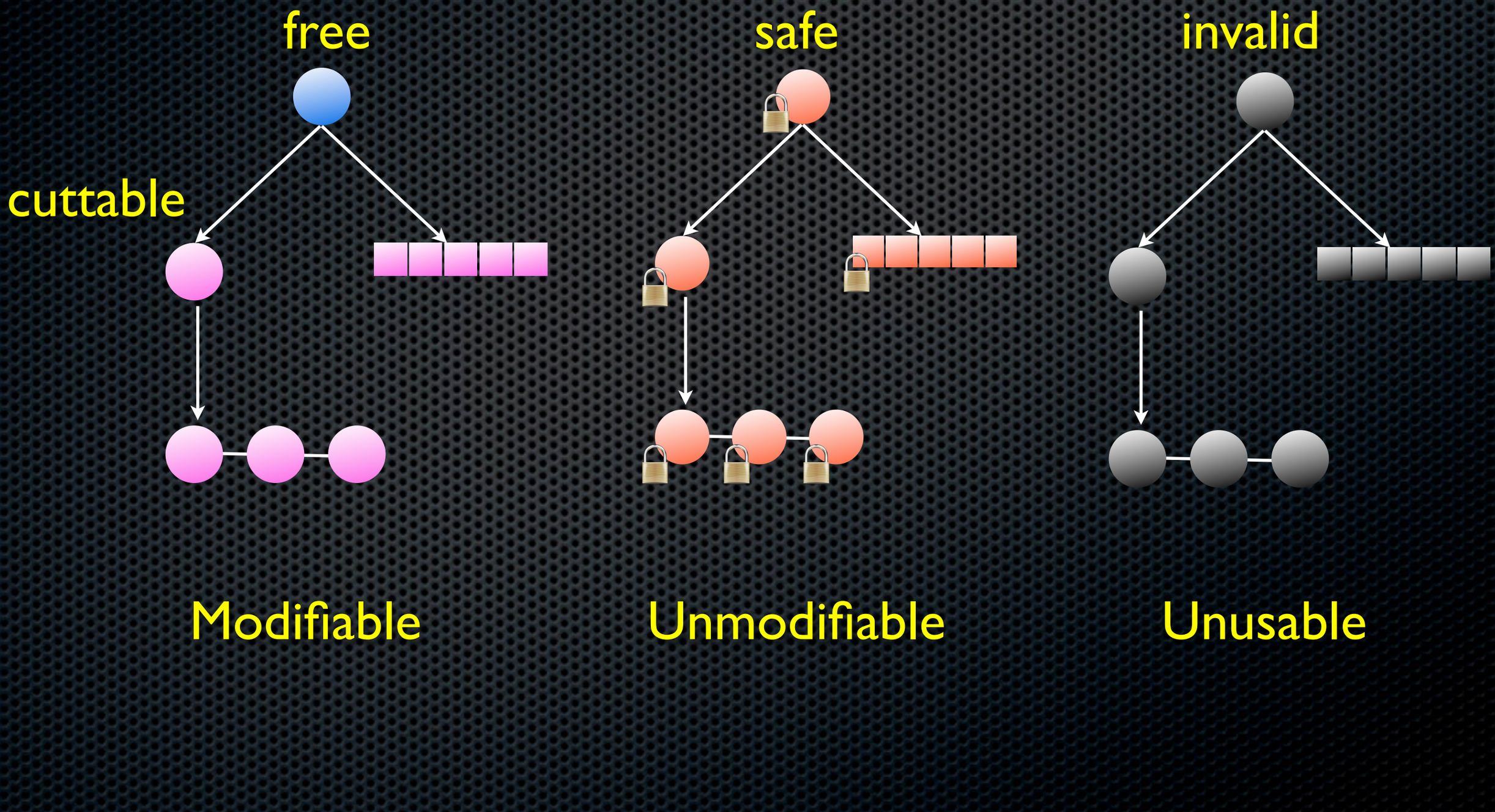
Modifiable Unmodifiable Unusable

capabilities



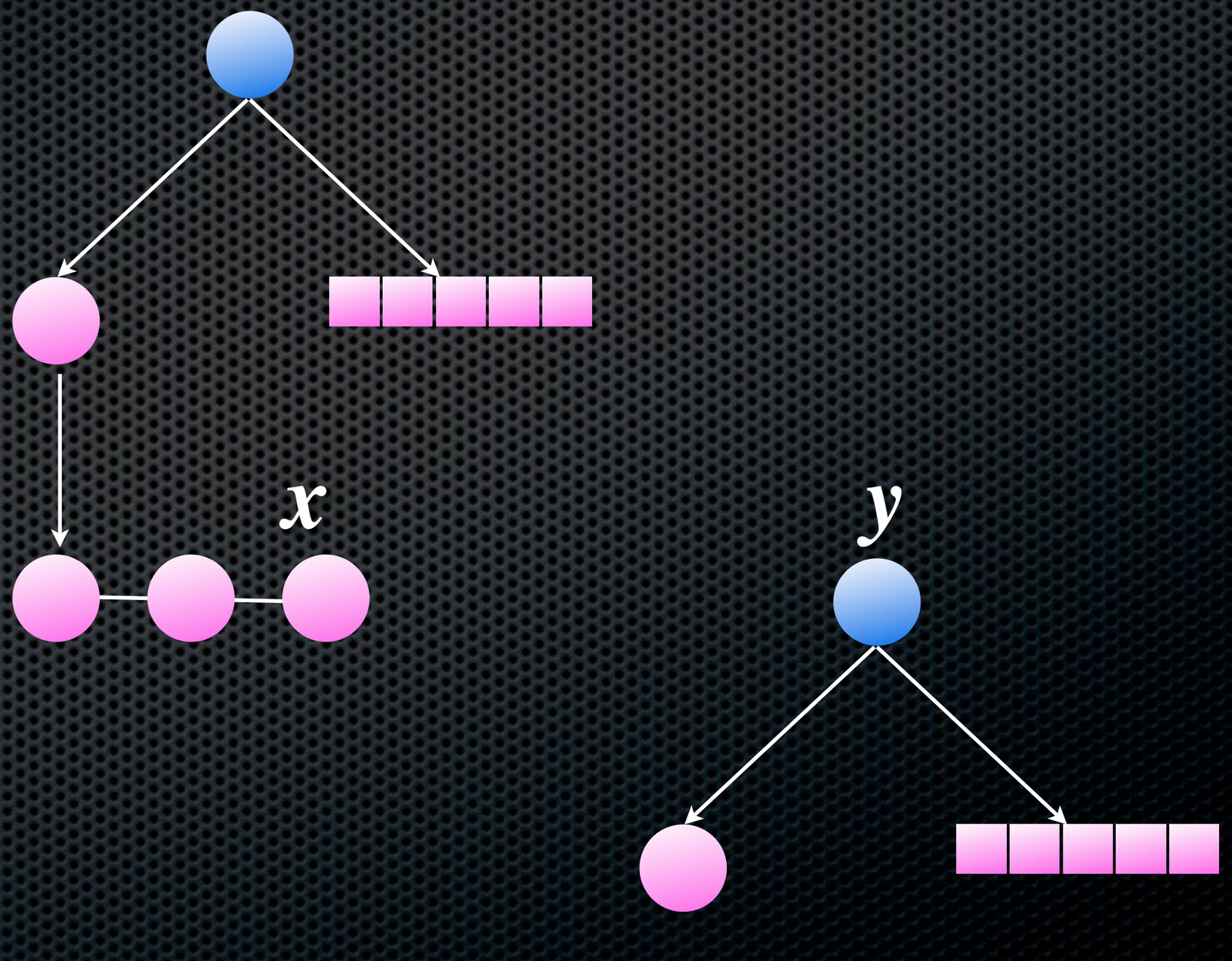
Modifiable Unmodifiable Unusable

capabilities



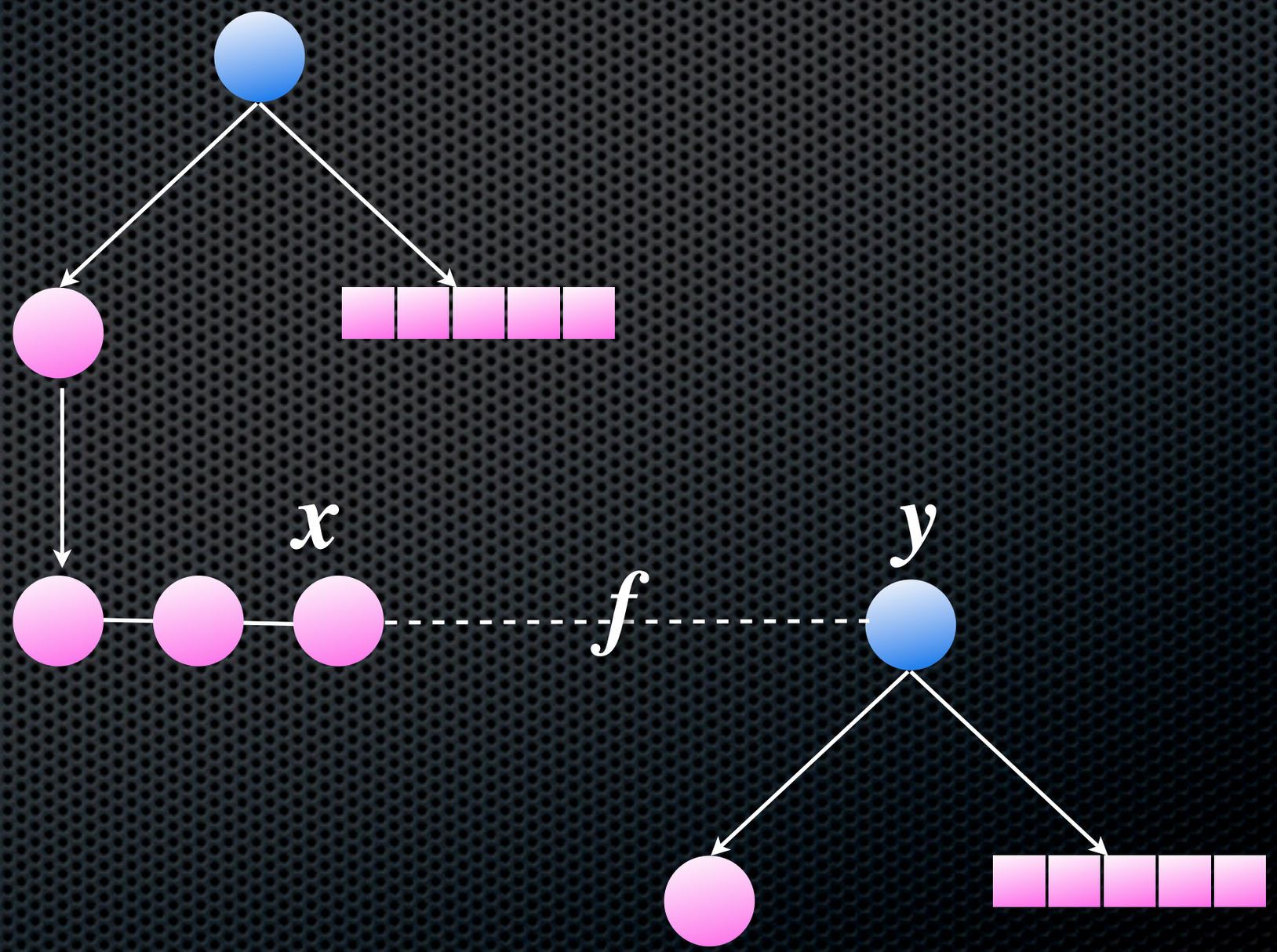
assignment

$$x.f = y$$



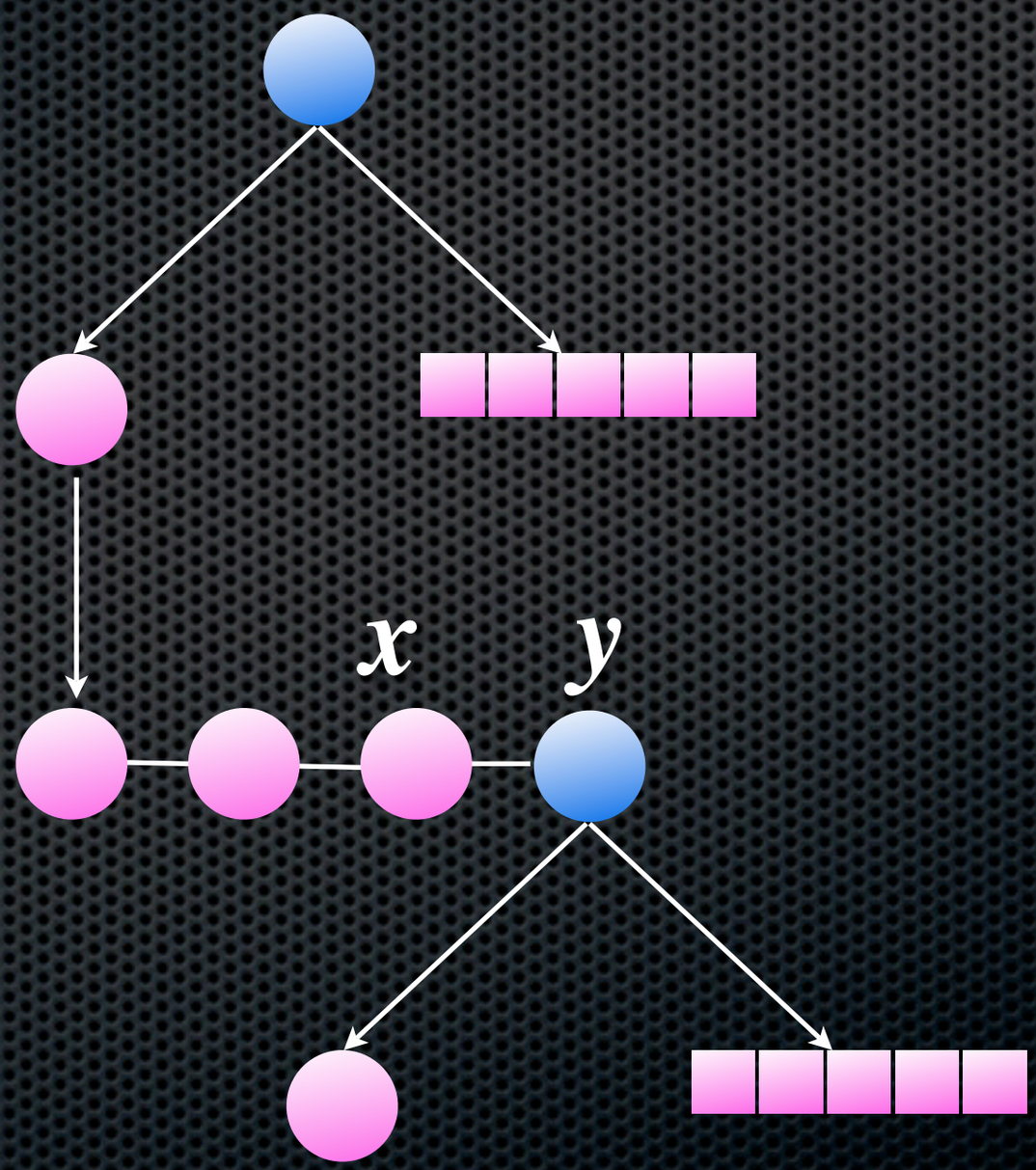
assignment

$$x.f = y$$



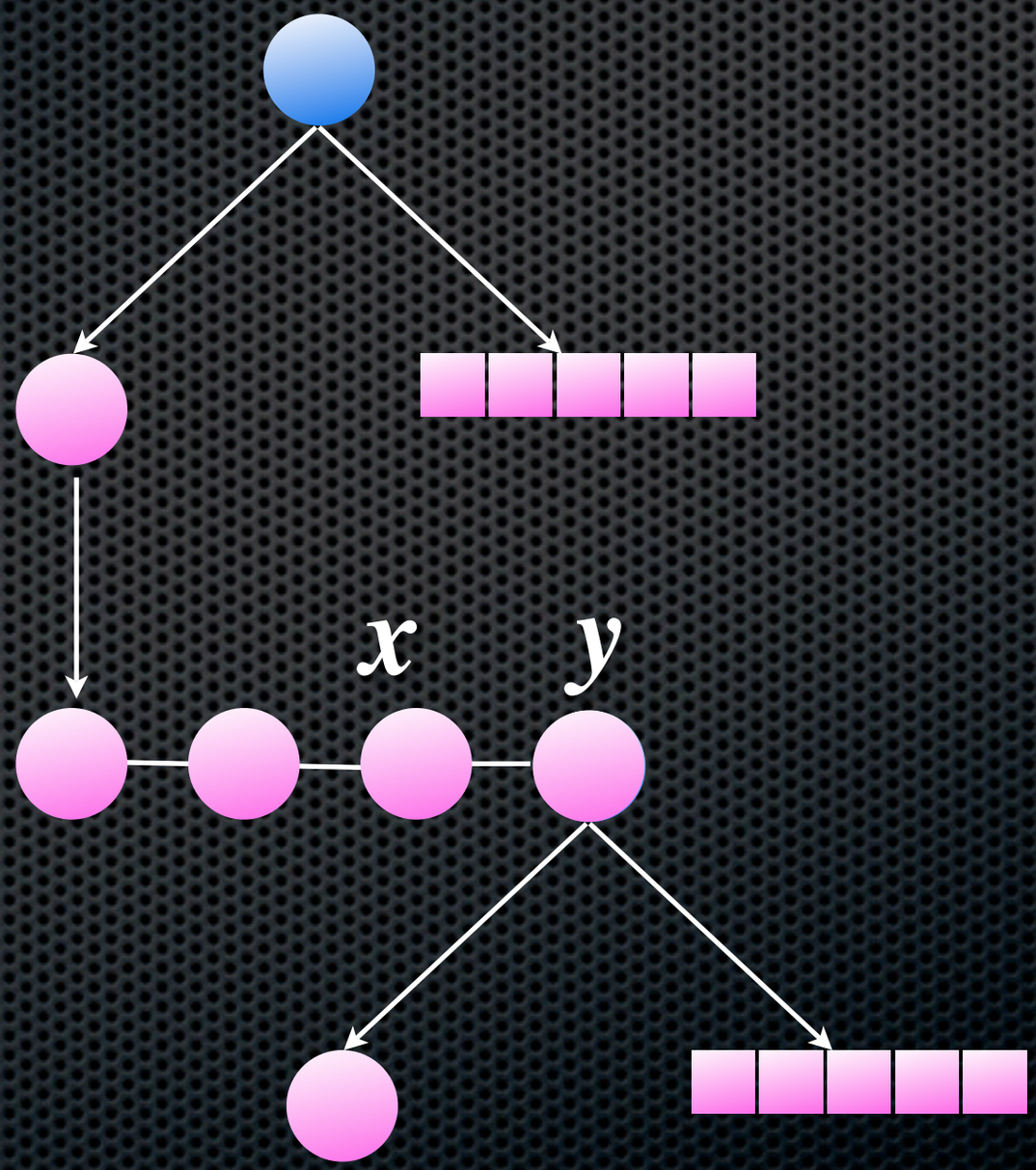
assignment

$$x.f = y$$



assignment

$$x.f = y$$



capability inference

```
class Event implements Message  {
    Event a;
    Event b [ ];
}
```

```
void foo(@free Event ev, @safe Event msg) {
   p  = new Event();
   msg.a  = p;
   ev.a  = p;
   ev.b[2] = p;
   ev.a  = msg;
}
```

capability inference

```
class Event implements Message  {
    Event a;
    Event b [ ];
}
```

```
void foo(@free Event ev, @safe Event msg) {
    p  = new Event();
    msg.a  = p;
    ev.a  = p;
    ev.b[2] = p;
    ev.a  = msg;
}
```

capability inference

```
class Event implements Message  {
    Event a;
    Event b [ ];
}
```

```
void foo(@free Event ev, @safe Event msg) {
  p  = new Event();
  msg.a  = p;          ✗  not modifiable
  ev.a  = p;
  ev.b[2] = p;
  ev.a  = msg;
}
```

capability inference

```
class Event implements Message  {
    Event a;
    Event b [ ];
}
```

```
void foo(@free Event ev, @safe Event msg) {
  p  = new Event();
  msg.a  = p;          ✗  not modifiable
  ev.a  = p;           ✓
  ev.b[2] = p;
  ev.a  = msg;
}
```

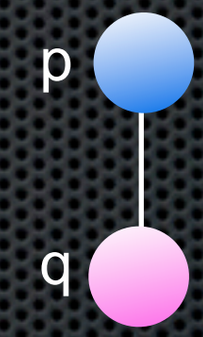
capability inference

```
class Event implements Message  {
    Event a;
    Event b [ ];
}
```

```
void foo(@free Event ev, @safe Event msg) {
  p  = new Event();
  msg.a  = p;          ✗ not modifiable
  ev.a  = p;           ✓
  ev.b[2]  = p;        ✗ p not free
  ev.a  = msg;         ✗ safe, cannot be assigned
}
```

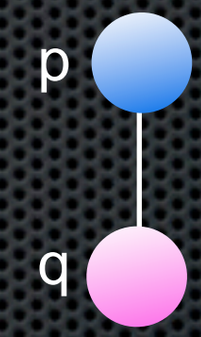
method calls

```
void foo (@free p) {
    q = p.f;
    print(q);          ✓
    mb.put(q);
    mb.put (p);
    print(p);
}
```



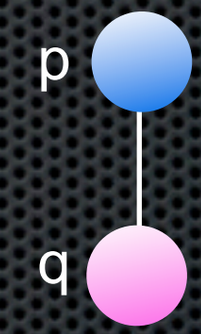
method calls

```
void foo (@free p) {
    q = p.f;
    print(q);          ✓
    mb.put(q);         ✗  q not root
►   mb.put (p);
    print(p);
}
```



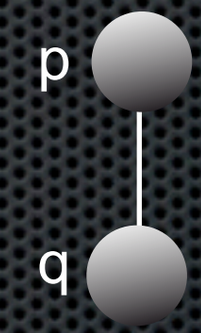
method calls

```
void foo (@free p) {
    q = p.f;
    print(q);          ✓
    mb.put(q);         ✗  q not root
    mb.put (p);        ✓
    print(p);
}
```



method calls

```
void foo (@free p) {
    q = p.f;
    print(q);          ✓
    mb.put(q);         ✗  q not root
    mb.put (p);        ✓
    print(p);          ✗  p, q invalid
}
```



cut operator

```
foo(@free  root, @cuttable mid) {
    if ( ...)
        r = root
    else
        r = cut(mid.f)
    mb.put(r)
}
```

static analysis

- Shape Analysis for heap abstraction
- Transfer of Ownership between tasks

== TOI between methods

Kilim summary

Tasks	lightweight automatic stack mgmt failure isolation state Isolation	cooperative tasking
Messaging	fast safe request reordering flow control open structures	tree-shaped structures (only when mutable)
Ease	uniform syntax for distr. & conc. prog monitorable use existing classes	fixed task granulariy

references

- Kilim: Isolation-typed Actors for Java.
  Sriram Srinivasan, Alan Mycroft. In ECOOP 2008
- A Thread of Ones Own.
  Sriram Srinivasan.
  New Horizons in Compilers Wkshp (2005)
- Making Reliable Distributed Systems in the Presence of Software Errors.
  Armstrong, J, PhD thesis, RIT Stockholm (2003)
- Alias Burying: Unique Variables without Destructive Reads.
  Boyland, J. In Soft. Pract. & Exper. (2001)
- Shape Analysis.
  Wilhelm, R., Sagiv, S., Reps, T.W. In ETAPS (2000)

http://kilim.malhar.net