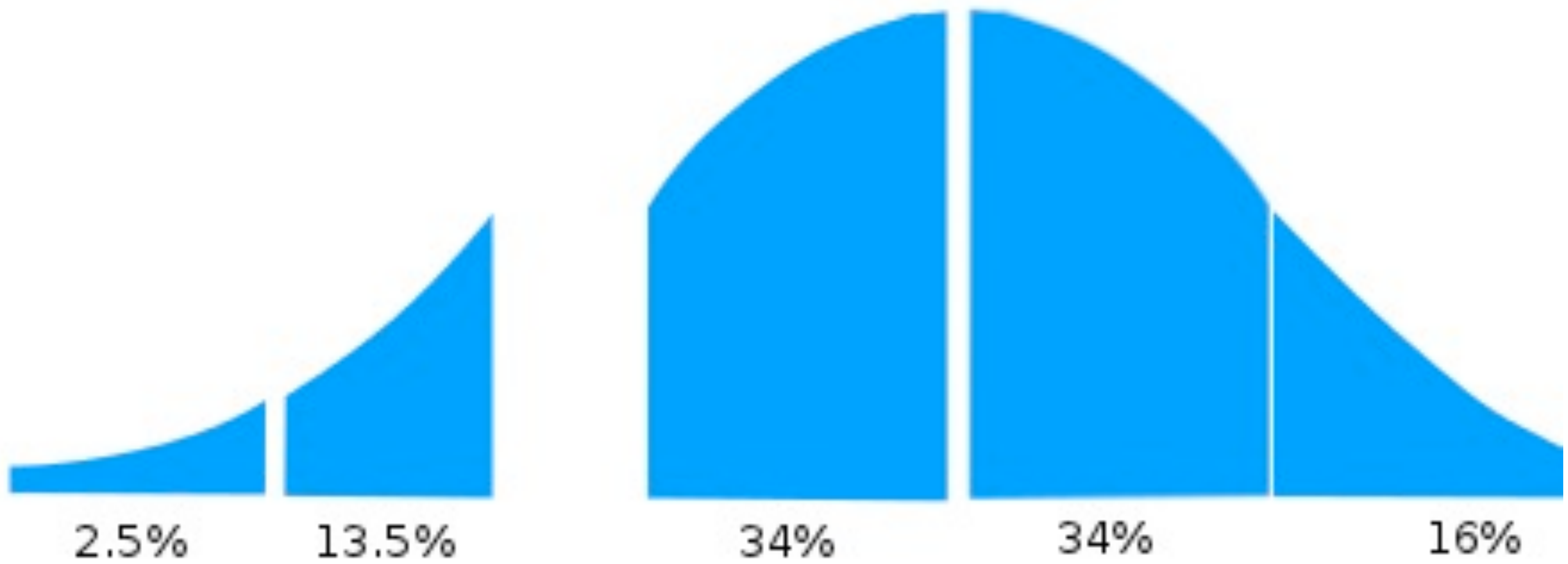
Game of Threads
You spawn or you die

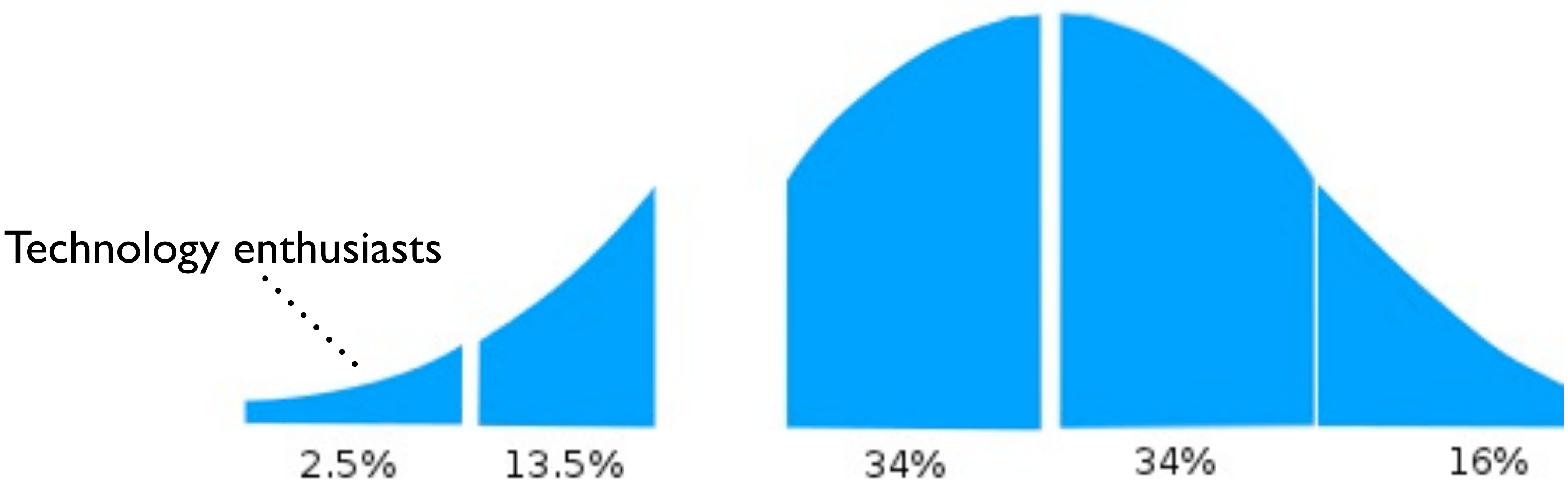
Torben Hoffmann
CTO, Erlang Solutions
torben.hoffmann@erlang-solutions.com
@LeHoff

Erlang SOLUTIONS

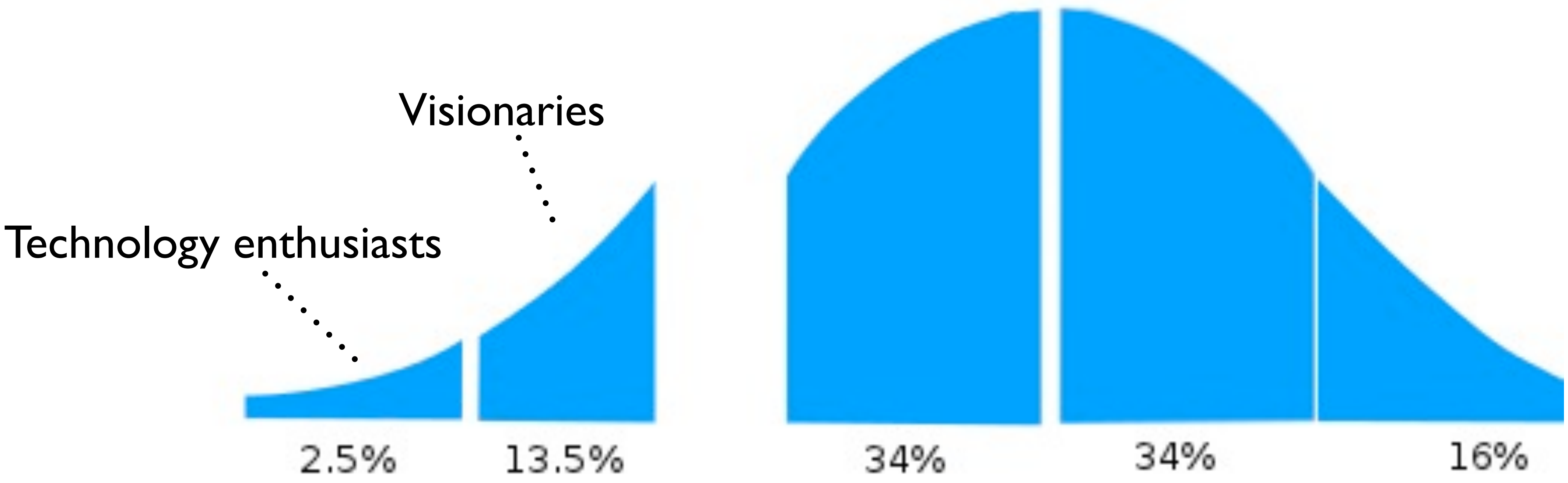
Technology Adoption Lifecyle



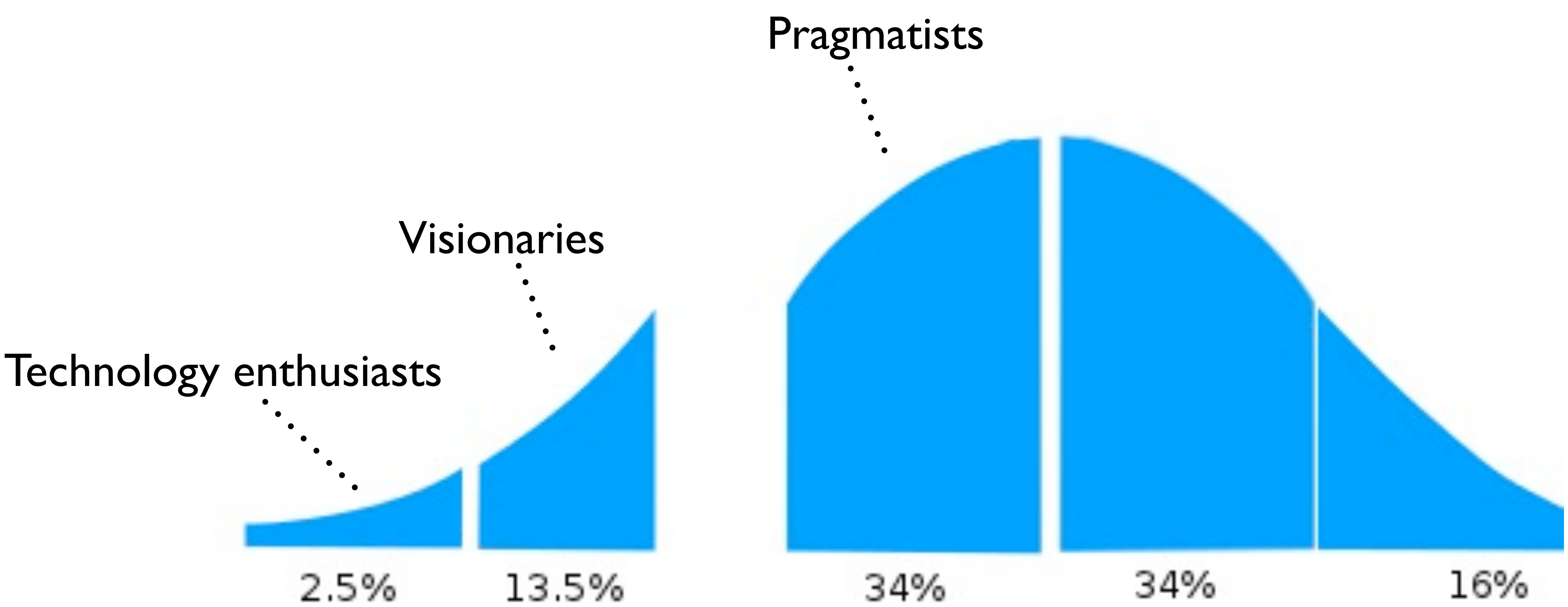
Technology Adoption Lifecyle



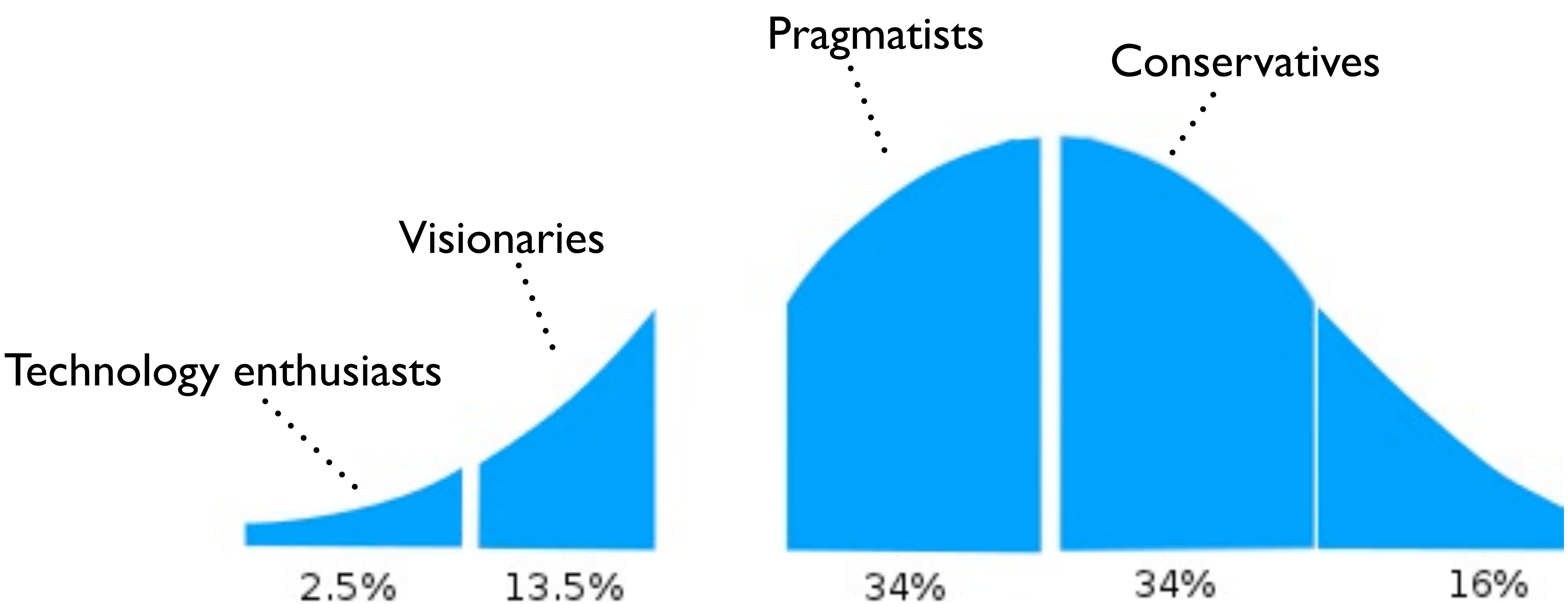
Technology Adoption Lifecyle



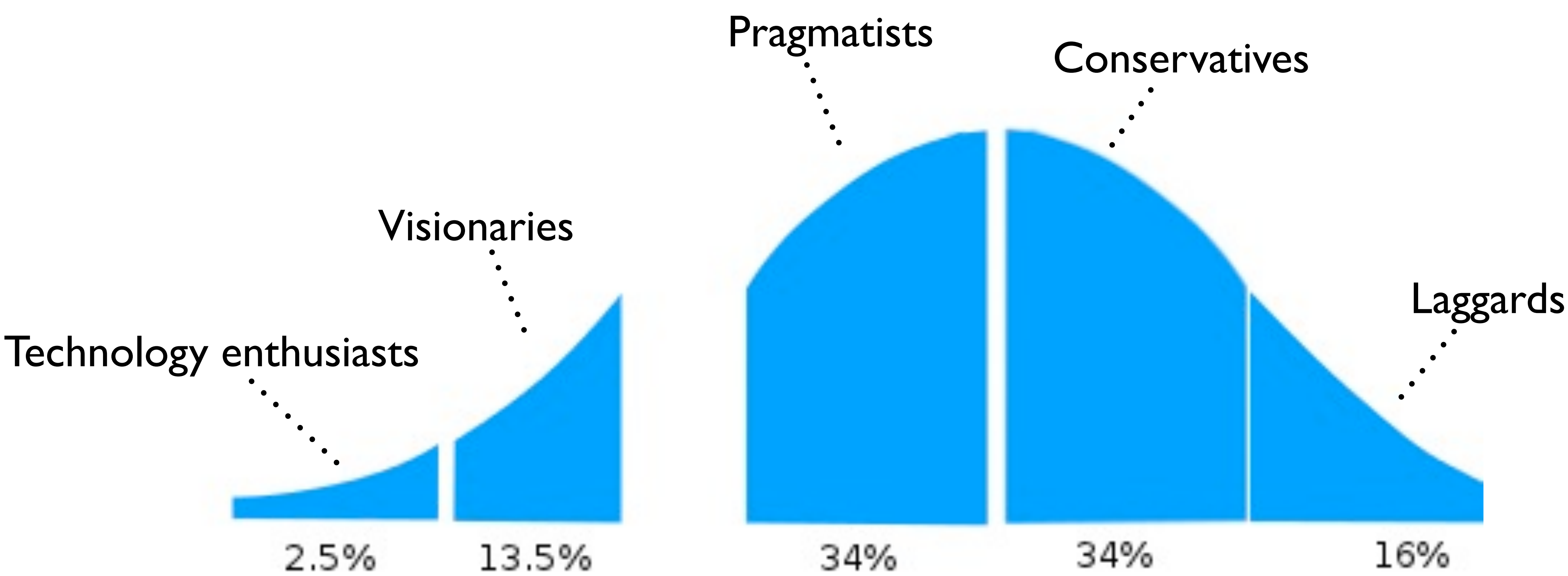
Technology Adoption Lifecyle



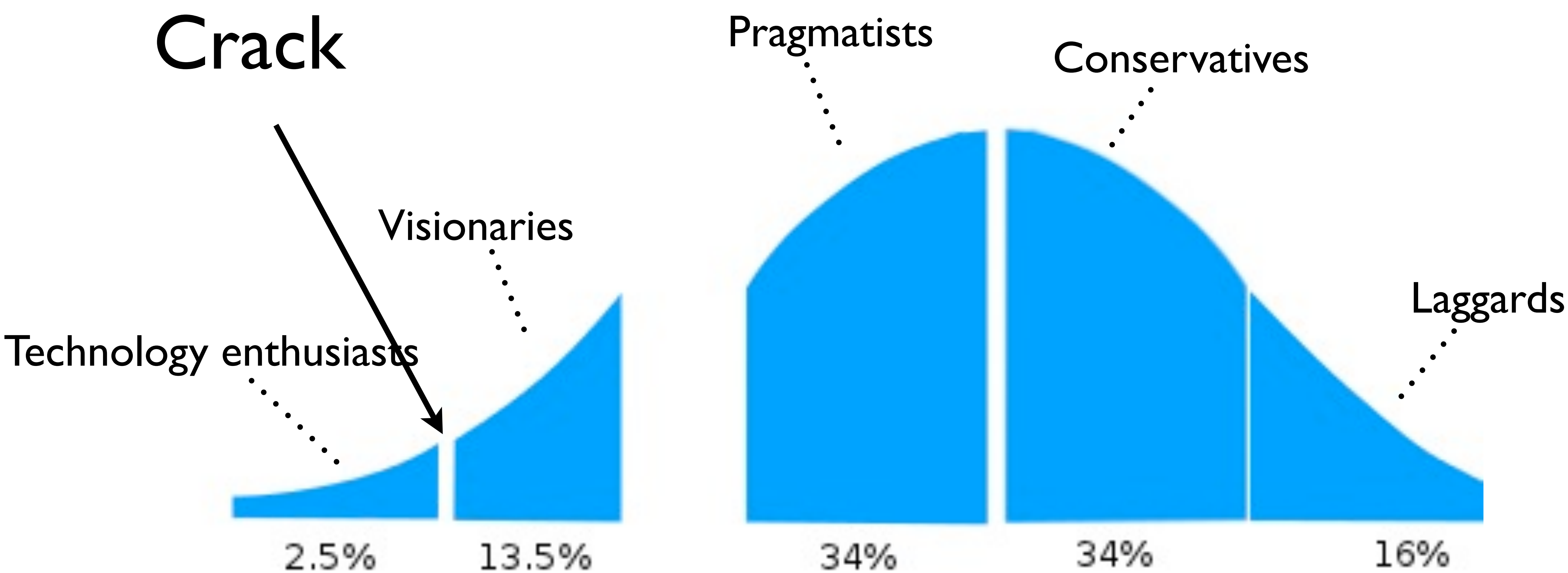
Technology Adoption Lifecyle



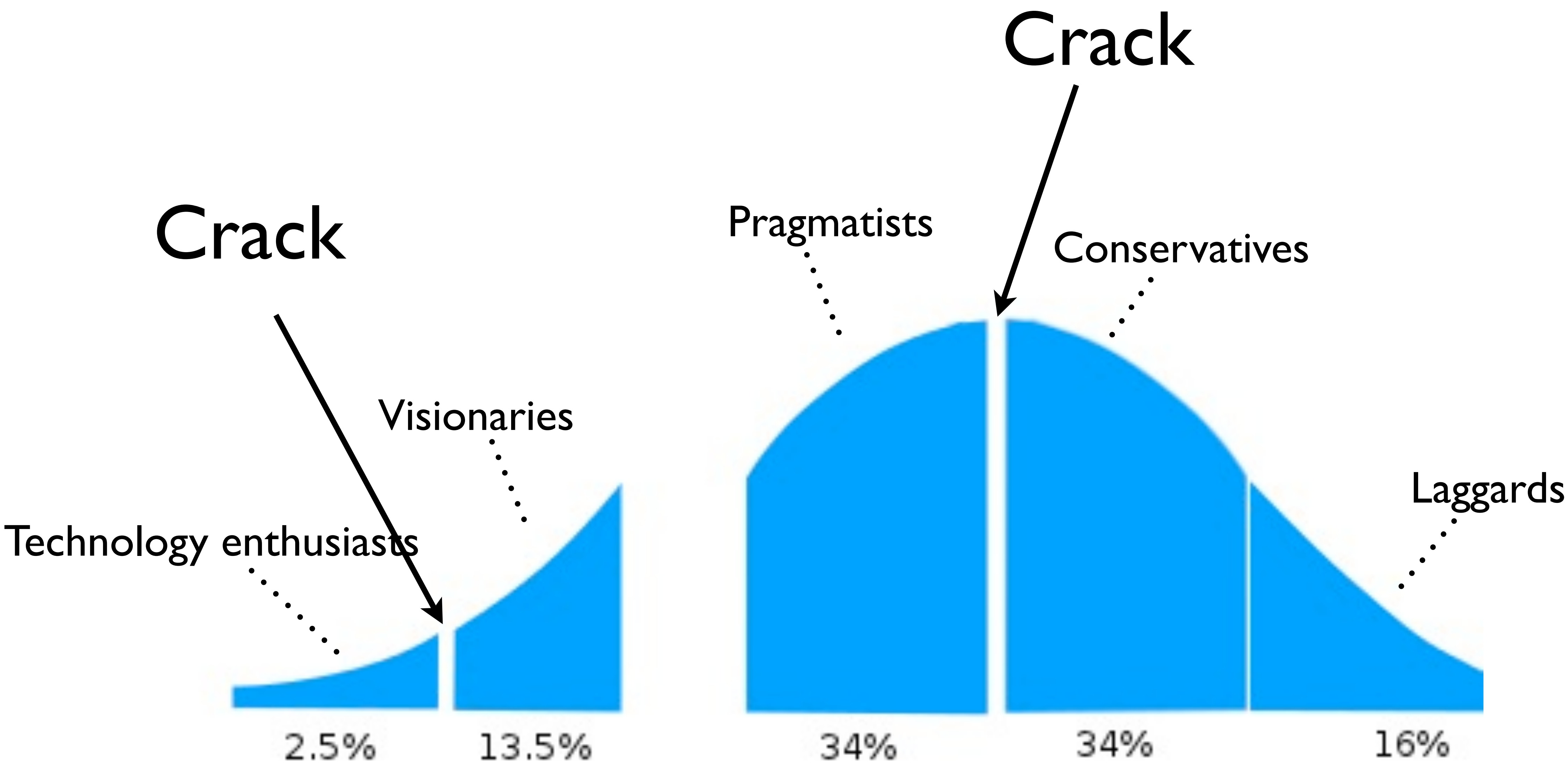
Technology Adoption Lifecyle



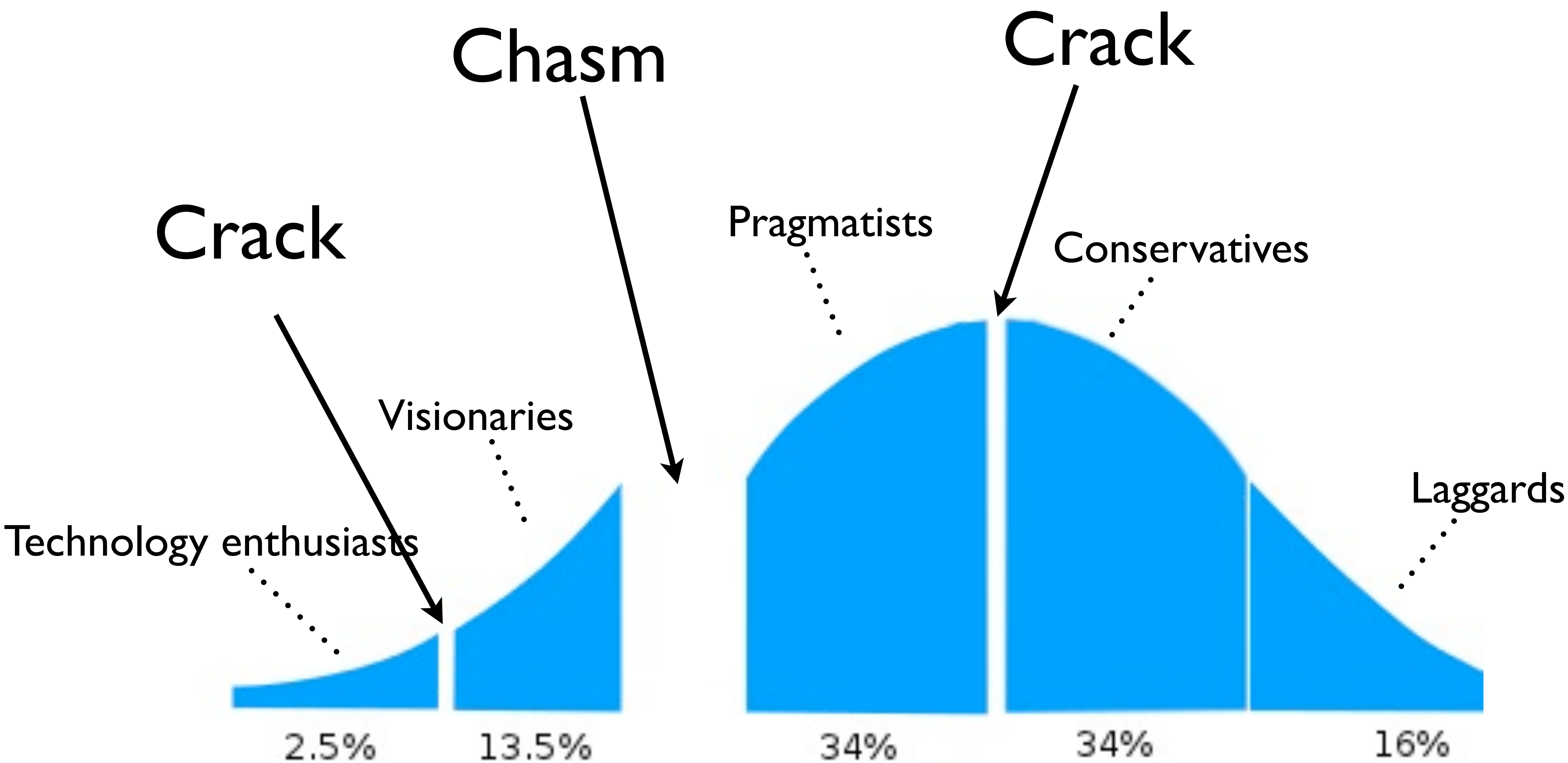
Technology Adoption Lifecyle



Technology Adoption Lifecyle



Technology Adoption Lifecyle



Free Lunch

Source: www.educationnews.org

Free Lunch

Source: www.educationnews.org

Free Lunch

Source: www.educationnews.org

We all love the concept

Free Lunch

Source: www.educationnews.org

We all love the concept

Experience rules it out

Free Lunch

Source: www.educationnews.org

We all love the concept

Experience rules it out

Paying for Lunch

Source: www.tidensnyheder.dk

Source: www.surreyartists.co.uk

Paying for Lunch

Source: www.tidensnyheder.dk

Source: www.surreyartists.co.uk

Unavoidable, but the price varies

After Lunch

After Lunch

Sometimes you have to pay an extra price

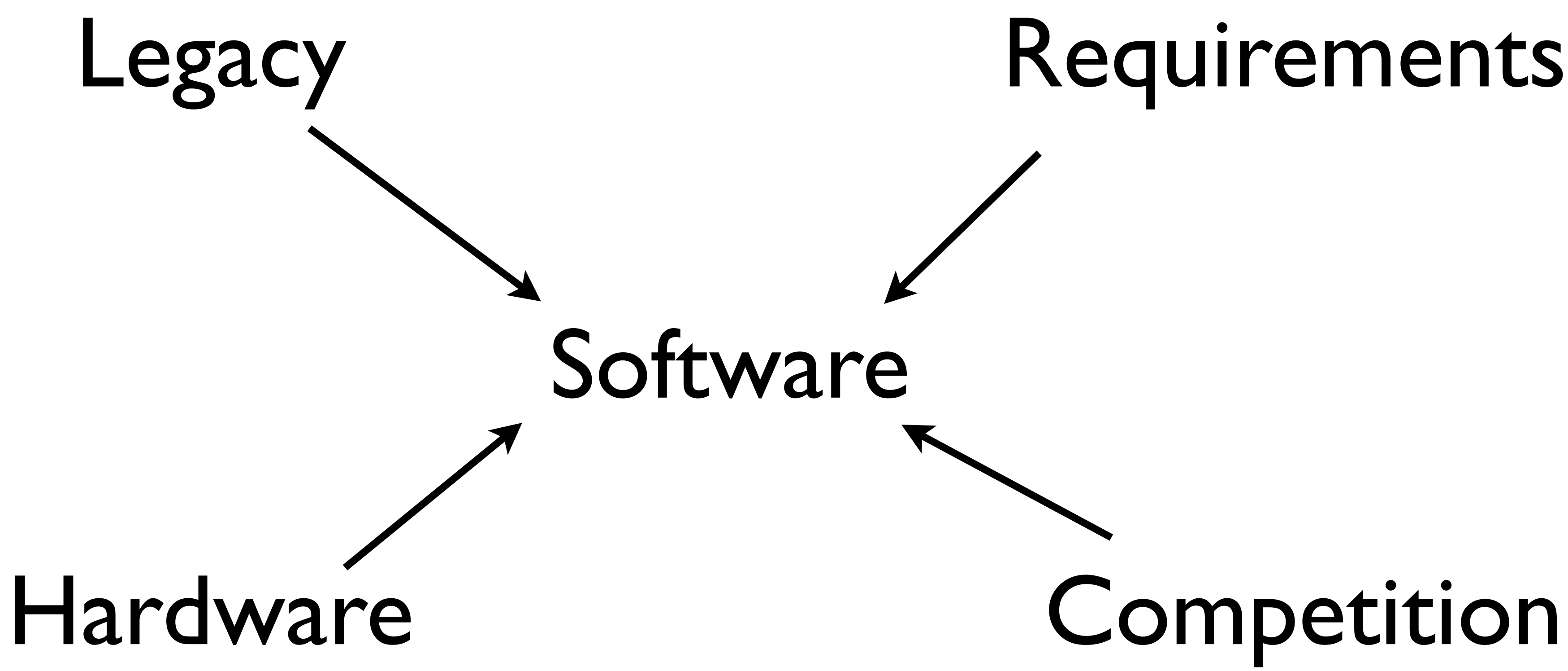
Software Lunch

Silver bullet (*n*)

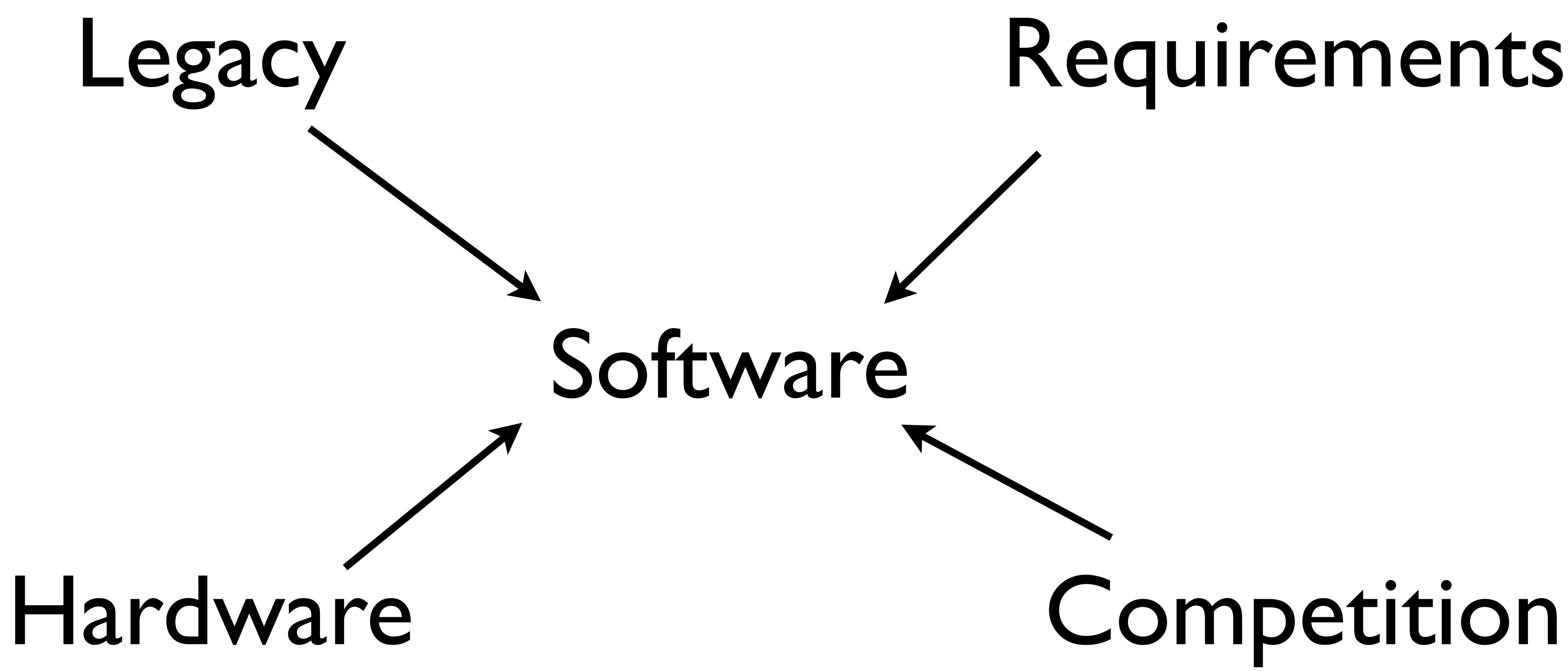
1. software slang for free lunch
2. used to kill vampires

Drives the creation of new languages

Context

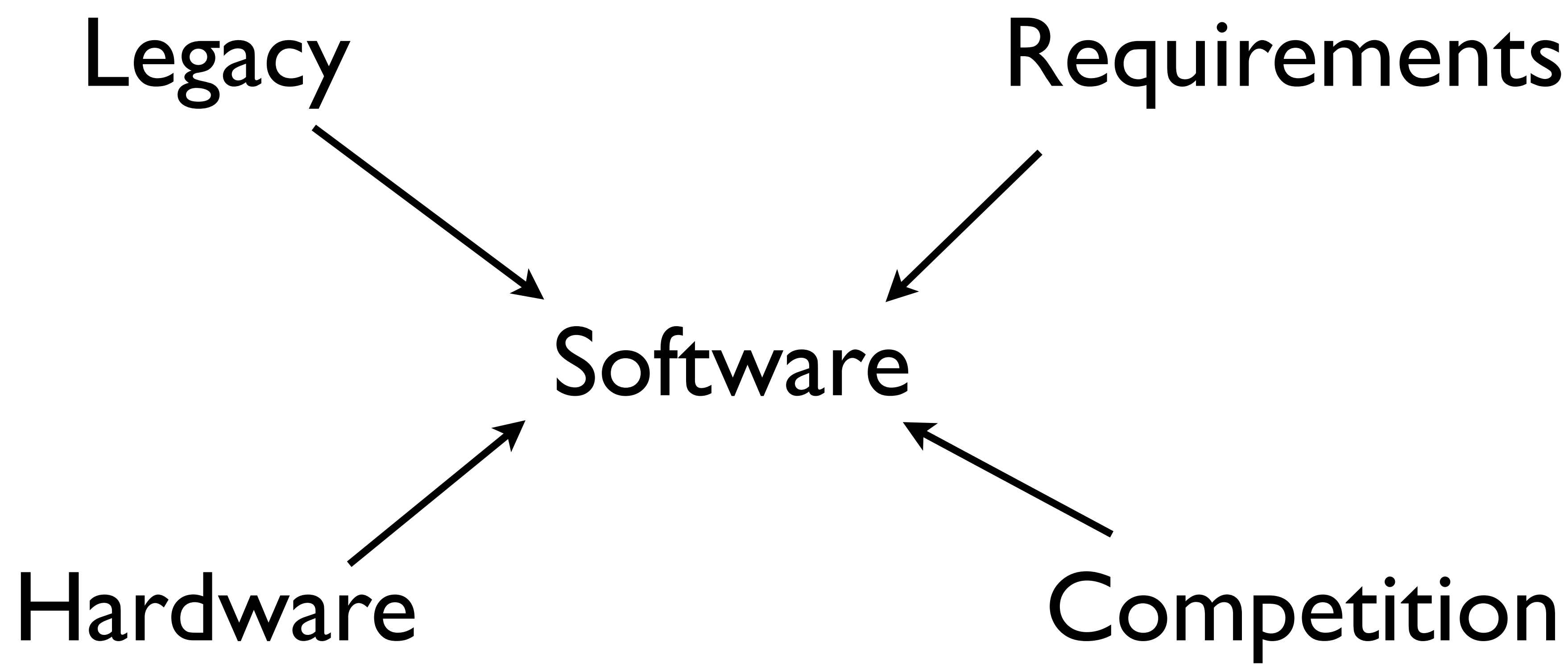


Context



Context changes all the time

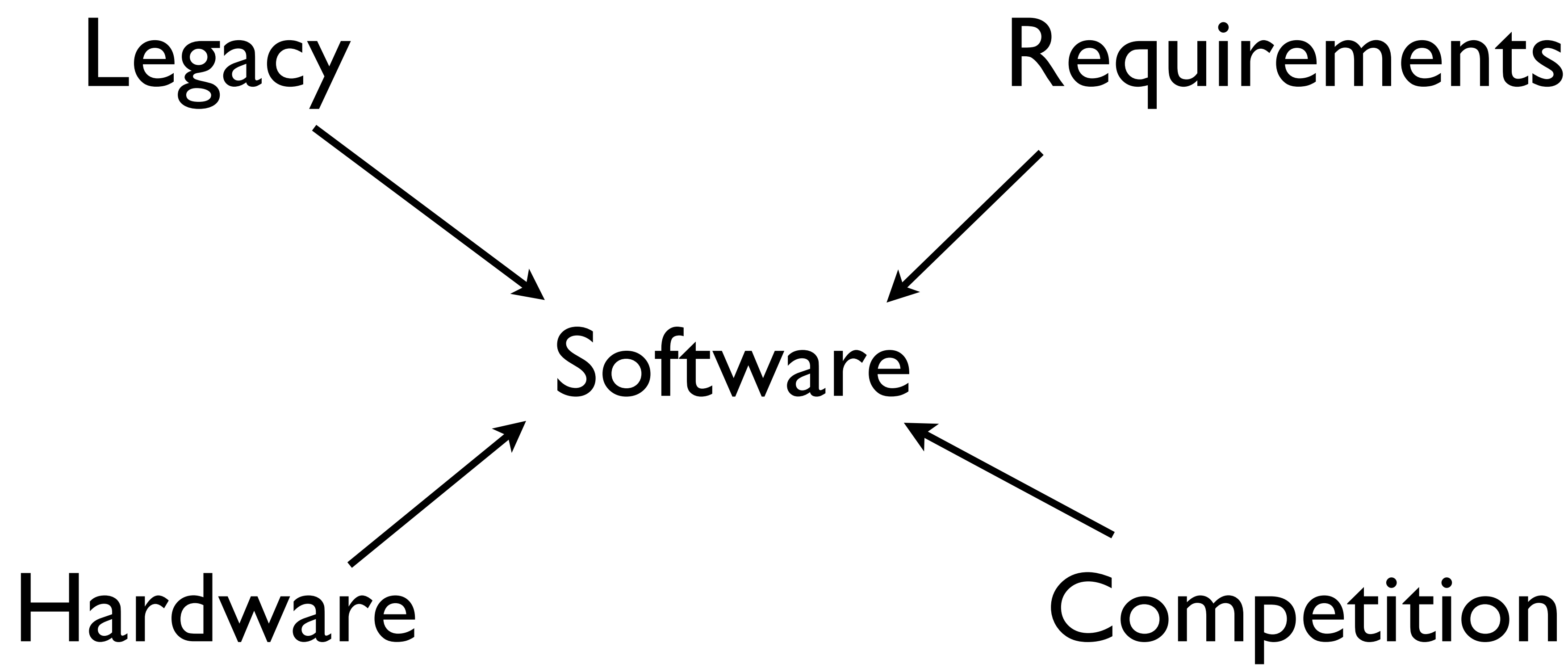
Context



Context changes all the time

Software has to follow suit

Context



Context changes all the time

Software has to follow suit

Each context is a market segment to conquer

Right tool for the job

Right tool for the job

Nice concept

Right tool for the job

Nice concept

How many tools can one use?

Right tool for the job

Nice concept

How many tools can one use?

Effectively?

Right tool for the job

Nice concept

How many tools can one use?

Effectively?

Learning cost

Right tool for the job

Nice concept

How many tools can one use?

Effectively?

Learning cost

Switching cost

Right tool for the job

Nice concept

How many tools can one use?

Effectively?

Learning cost

Switching cost

Pain = willingness to change

Right tool for the job

Nice concept

How many tools can one use?

Effectively?

Learning cost

Switching cost

Pain = willingness to change

Erlang’s domain

Erlang's domain

Concurrency

Erlang's domain

Concurrency

Low latency

Erlang's domain

Concurrency

Low latency

Resilience

Erlang's domain

Concurrency

Low latency

Resilience

When it fits:
High productivity
Short time-to-market

Erlang's domain

Concurrency

Low latency

Resilience

When it fits:
High productivity
Short time-to-market

But not perfect for everything :-(

Why ICE?

Why ICE?

Erlang VM designed for concurrency

Why ICE?

Erlang VM designed for concurrency

Parallelism an afterthought

Why ICE?

Erlang VM designed for concurrency

Parallelism an afterthought

No low-level optimisations

Why ICE?

Erlang VM designed for concurrency

Parallelism an afterthought

No low-level optimisations

How ICE?

How ICE?

Declarative programs with latent parallelism

How ICE?

Declarative programs with latent parallelism

Based on TransLucid

How ICE?

Declarative programs with latent parallelism

Based on TransLucid

Tweak data structures to get scalable performance

How ICE?

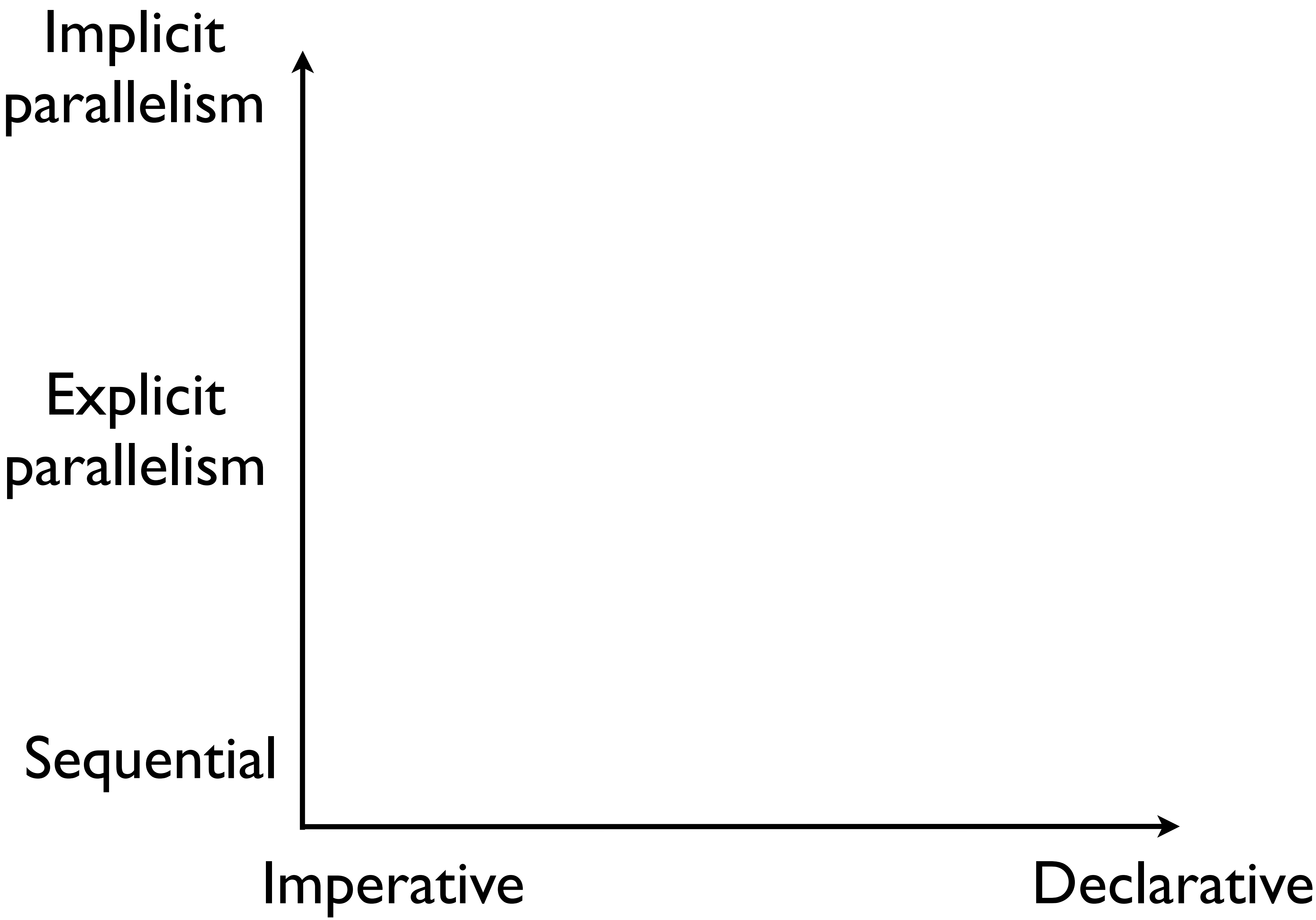
Declarative programs with latent parallelism

Based on TransLucid

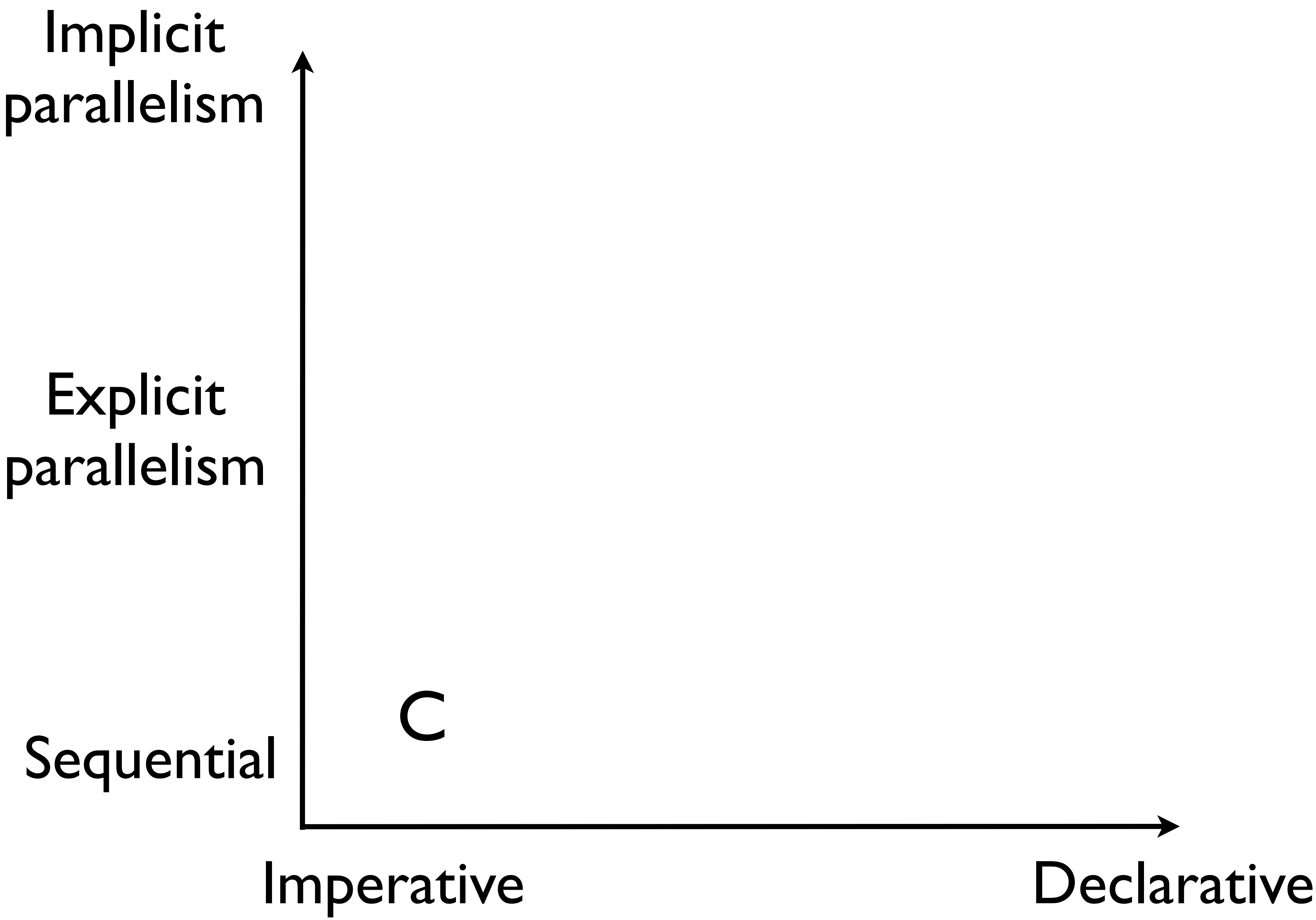
Tweak data structures to get scalable performance

Built on top of the Erlang VM

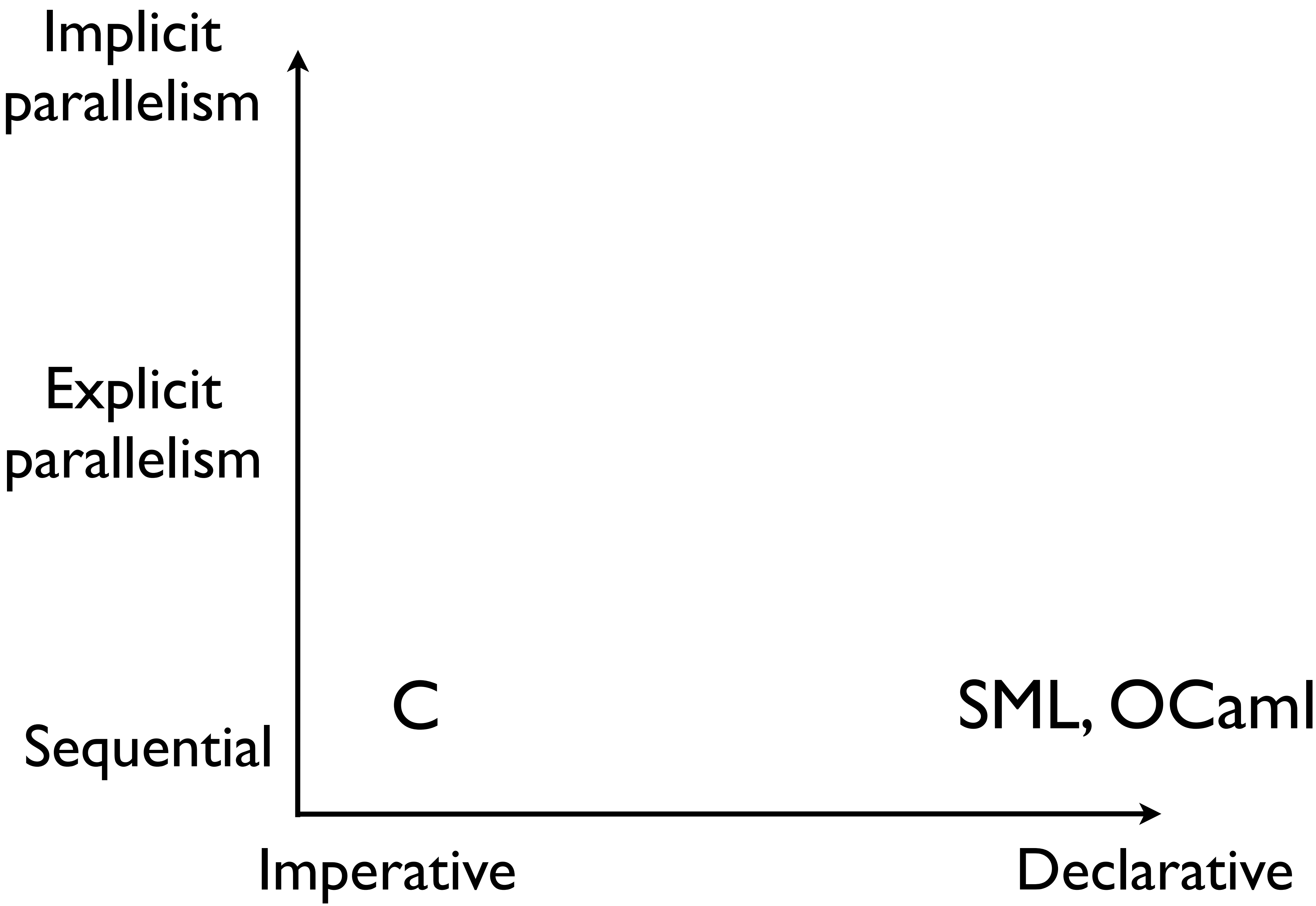
Programming



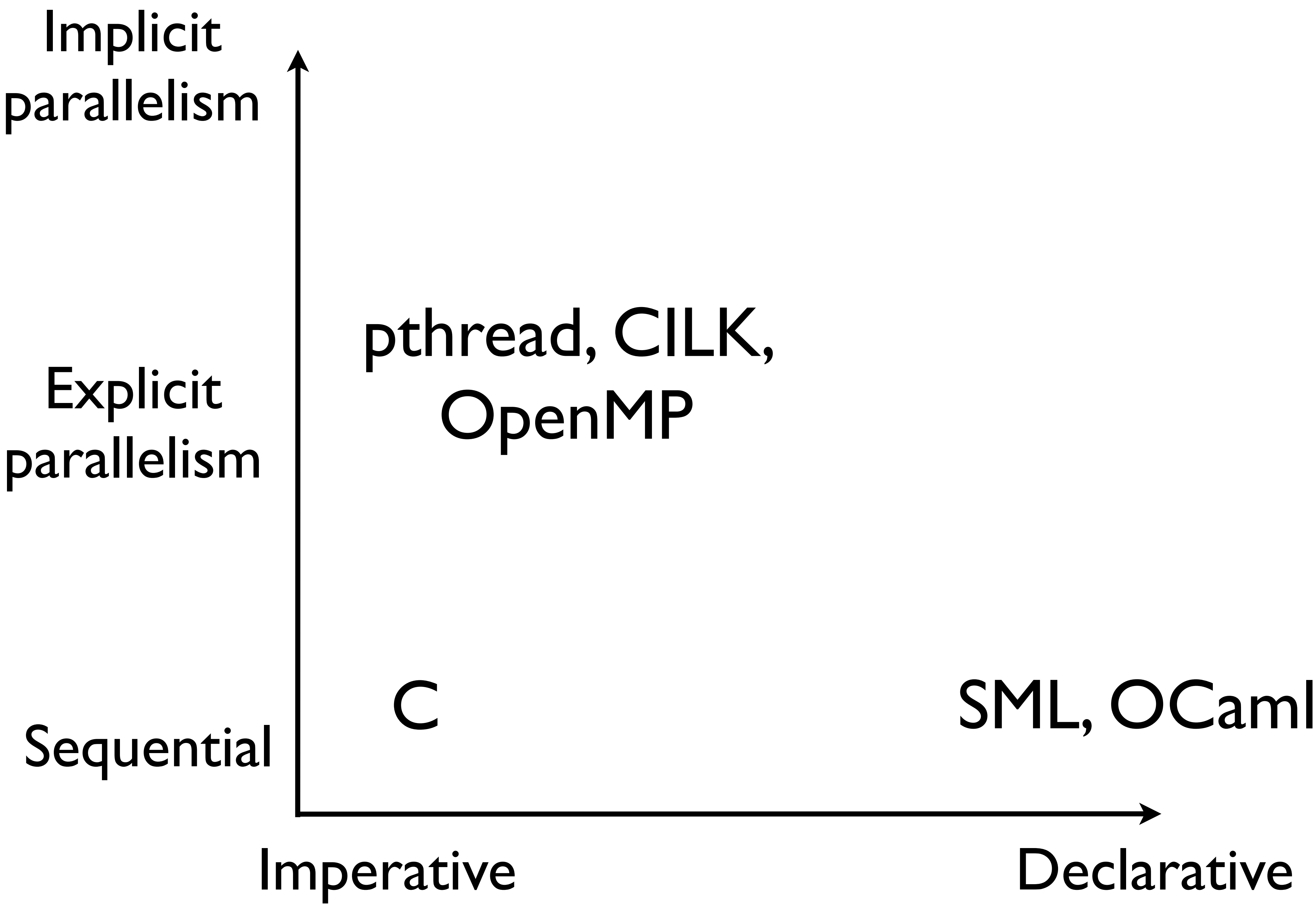
Programming



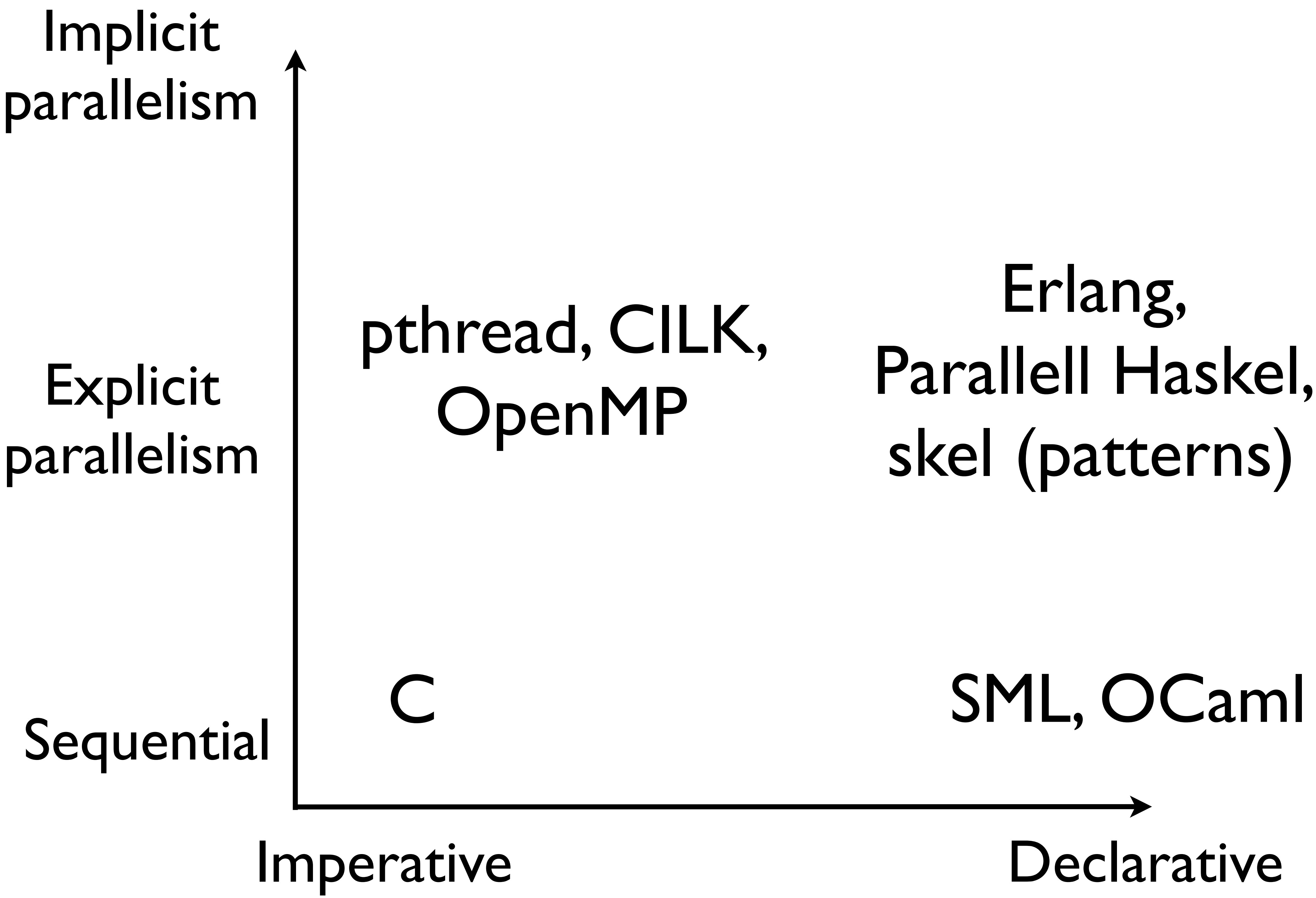
Programming



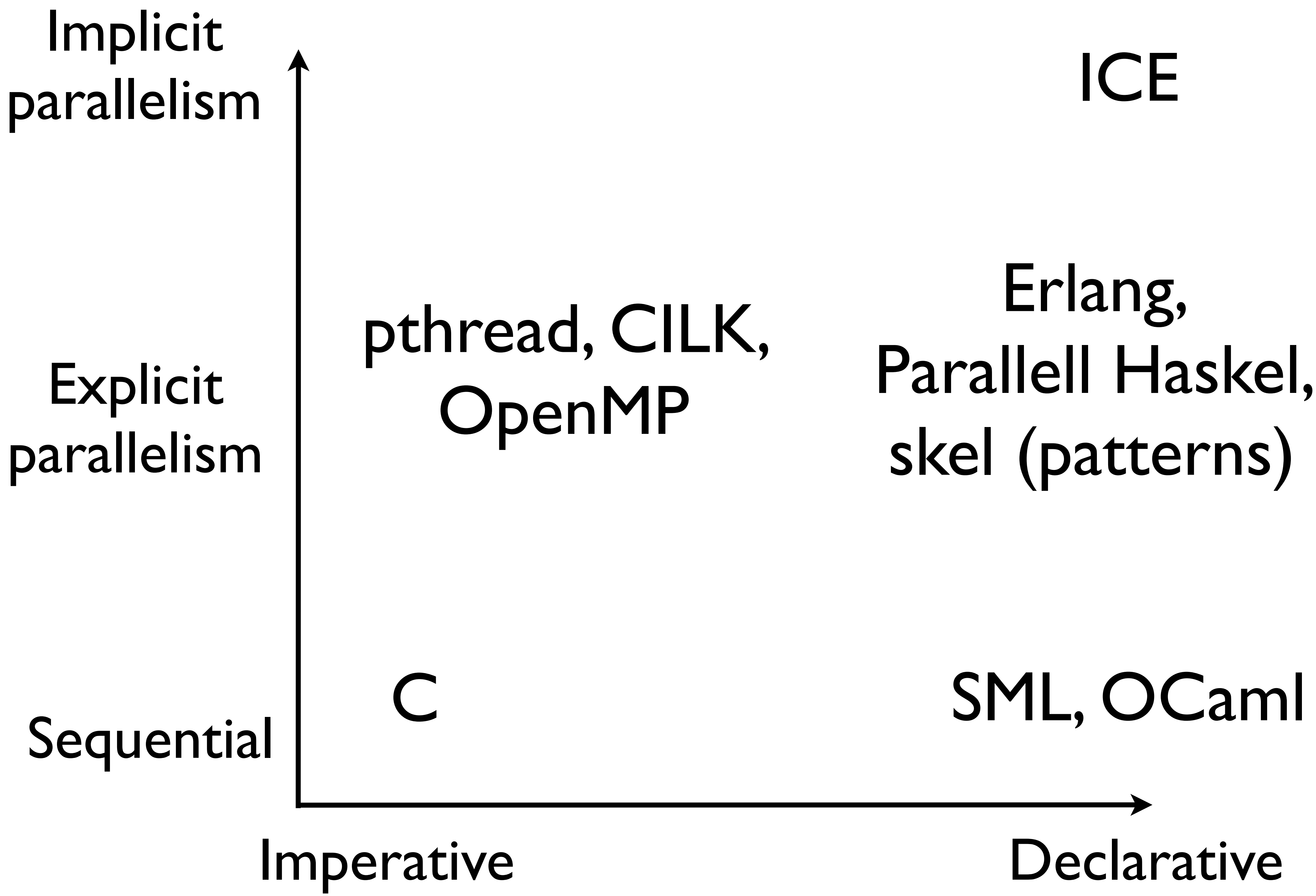
Programming



Programming



Programming




Erlang IS COMING
GAME OF THREADS
YOU SPAWN OR YOU DIE

ParaPhrase

ParaPhrase

EU funded FP7 project

ParaPhrase

EU funded FP7 project

Parallelism on heterogeneous platforms

ParaPhrase

EU funded FP7 project

Parallelism on heterogeneous platforms

Pattern based approach

ParaPhrase

EU funded FP7 project

Parallelism on heterogeneous platforms

Pattern based approach

Refactor the parallel patterns in

ParaPhrase Example

ParaPhrase Example

```
f(g(X))
```

ParaPhrase Example

`f(g(X))`

becomes

ParaPhrase Example

```
f(g(X))
```

becomes

```
skel:run(
  [{farm, [{seq, fun ?MODULE:g/1}], 24},
   {farm, [{seq, fun ?MODULE:f/1}], 24}],
   X])
```

ParaPhrase Example

```
f(g(X))
```

becomes

```
skel:run(
  [{farm, [{seq, fun ?MODULE:g/1}], 24},
   {farm, [{seq, fun ?MODULE:f/1}], 24}],
   X])
```

Productivity: hours instead of days

I want more.

I want more.

I know about wanting more.
I invented the concept.
The question is how much more.

Intensionality

Intensionality

Extreme version of declarative programming

Higher-level than functional programming

Focus on composition in a math like way

Extensional data needed to give the intensional program something concrete to work on

Elements

Elements

Intensional language (parser and evaluator)

Elements

Intensional language (parser and evaluator)

Extensional specification component

Elements

Intensional language (parser and evaluator)

Extensional specification component

Process abstraction & scheduling mechanism

Elements

Intensional language (parser and evaluator)

Extensional specification component

Process abstraction & scheduling mechanism

Core Idea

Core Idea

Demands spark off parallel computations

Glue

Intensional

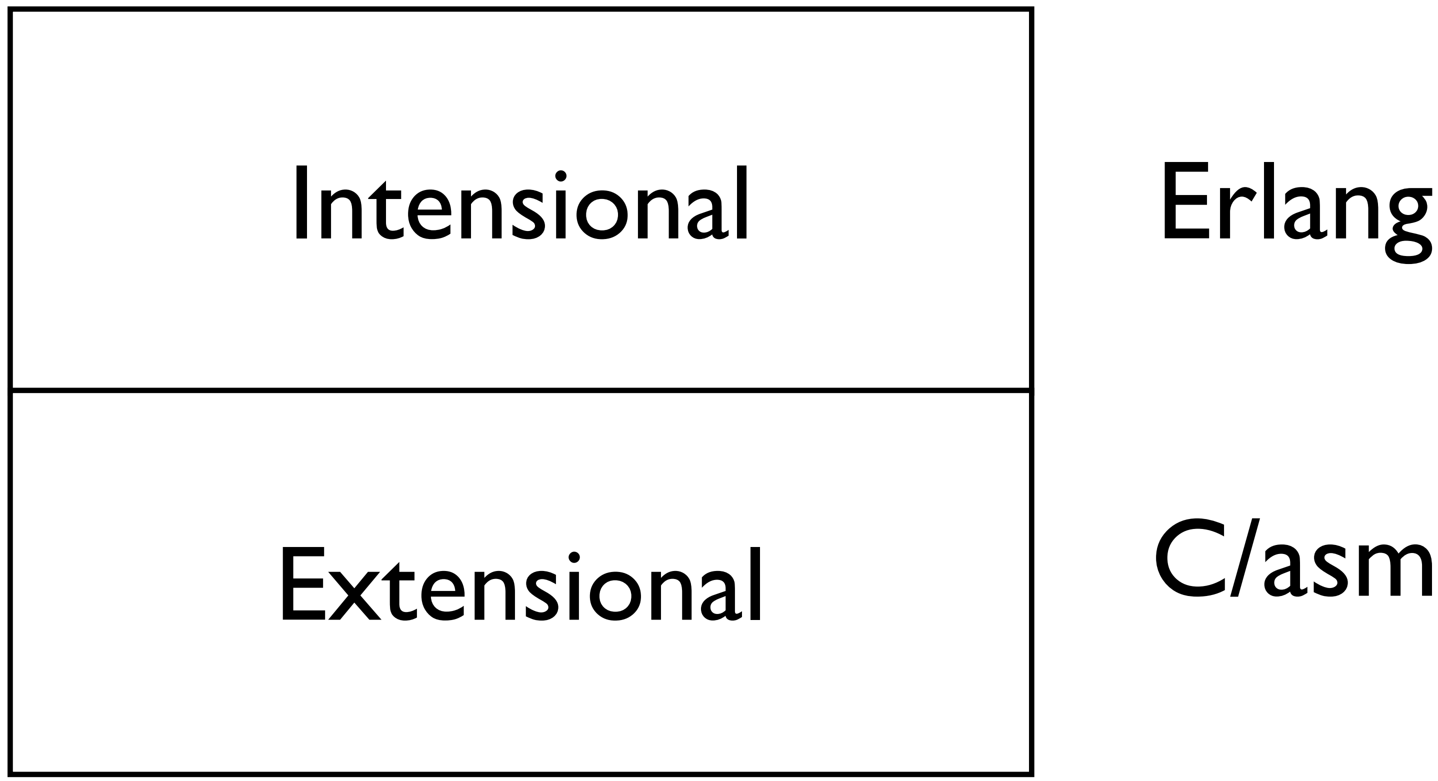
Extensional

Glue

Intensional
Extensional

Erlang

Glue



Glue

Intensional	Erlang	`var C = A + B`
Extensional	C/asm	

Glue

Intensional	Erlang	`var C = A + B`
Extensional	C/asm	`{A, float, 512, 64}`

Variables in ICE

```
var A = 42 + 2*#.x + #.y
```

Variables in ICE

```
var A = 42 + 2*#.x + #.y
```

Specifies this 2d thingy

Variables in ICE

```
var A = 42 + 2*#.x + #.y
```

Specifies this 2d thingy

'A'	0	1	2	3	$\overset{\#.x}{\rightarrow}$
0	42	44	46	48	$\cdots$
1	43	45	47	49	$\cdots$
2	44	46	48	50	$\cdots$
3	45	47	49	51	$\cdots$
$\#.y\downarrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Variables in ICE

```
var A = 42 + 2*#.x + #.y
```

Specifies this 2d thingy

'A'	0	1	2	3	$\overset{\#.x}{\rightarrow}$
0	42	44	46	48	$\cdots$
1	43	45	47	49	$\cdots$
2	44	46	48	50	$\cdots$
3	45	47	49	51	$\cdots$
$\#.y\downarrow$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Infinite table = extensional view of our intension

Demands and Context

```
A @ [ x <- 3, y <- 5]
```

"demand for the value of A
at the context x=3 and y=5"

Examples

Fibonacci

Equation:

```
var Fib = if #.n <= 1 then
        #.n
       else
        Fib @ [n<- #.n-1] +
        Fib @ [n<- #.n-2]
       fi
```

Fibonacci

Equation:

```
var Fib = if #.n <= 1 then
          #.n
         else
          Fib @ [n<- #.n-1] +
          Fib @ [n<- #.n-2]
         fi
```

Demand

Fibonacci

Demand:

Fibonacci

Demand:

```
fib @ [n <- 5]
```

Fibonacci

Demand:

```
            fib @ [n <- 5]
           ↙
fib @ [n <- 4]
```

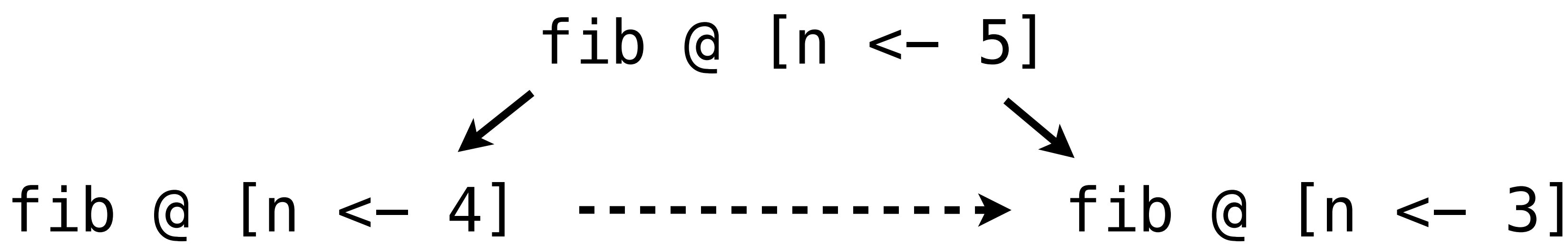
Fibonacci

Demand:

```
              fib @ [n <- 5]
             ↙              ↘
fib @ [n <- 4]               fib @ [n <- 3]
```

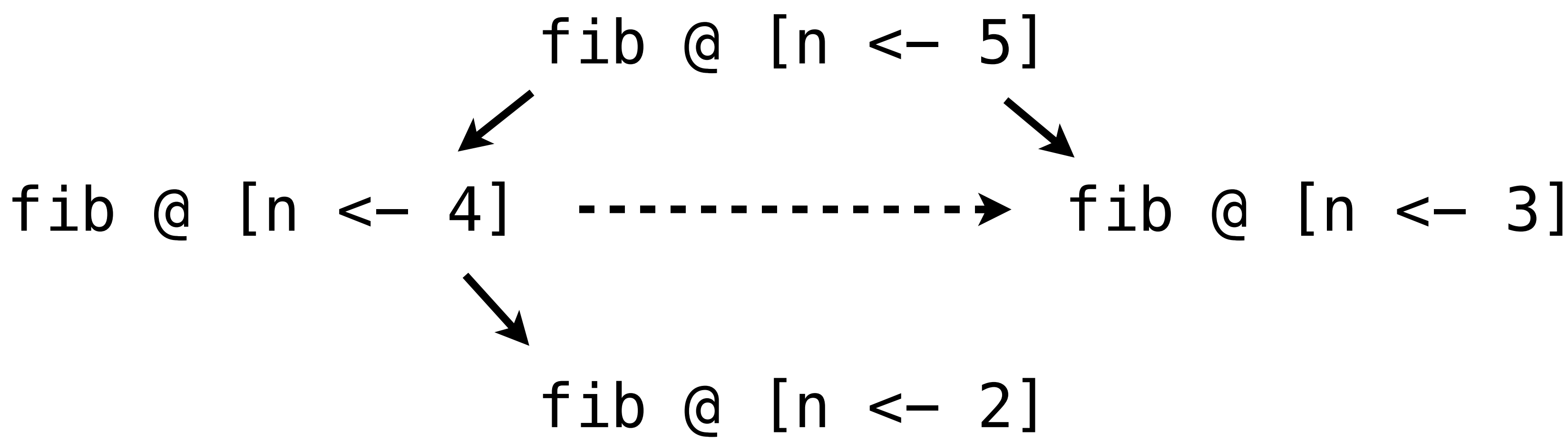
Fibonacci

Demand:



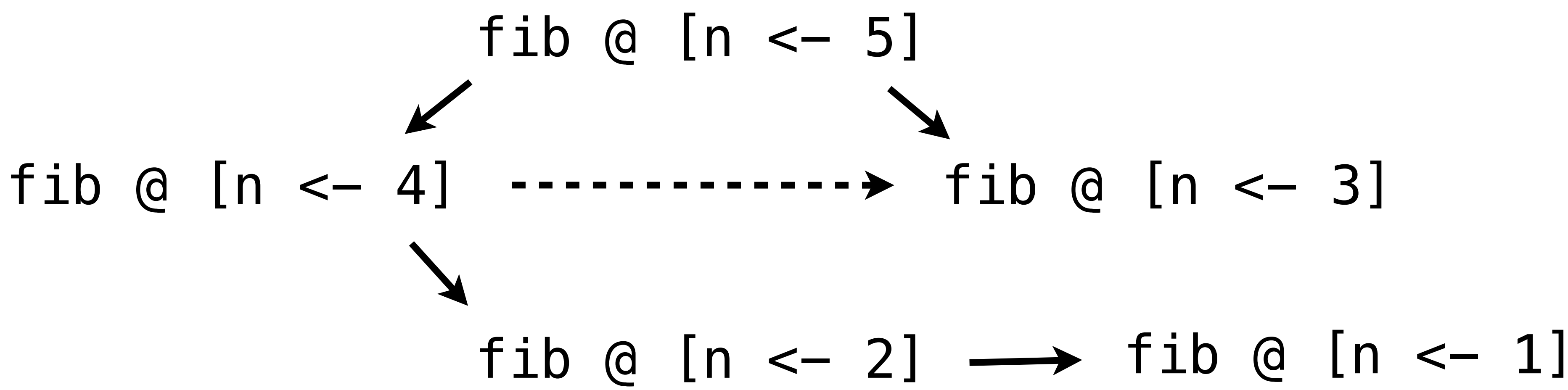
Fibonacci

Demand:



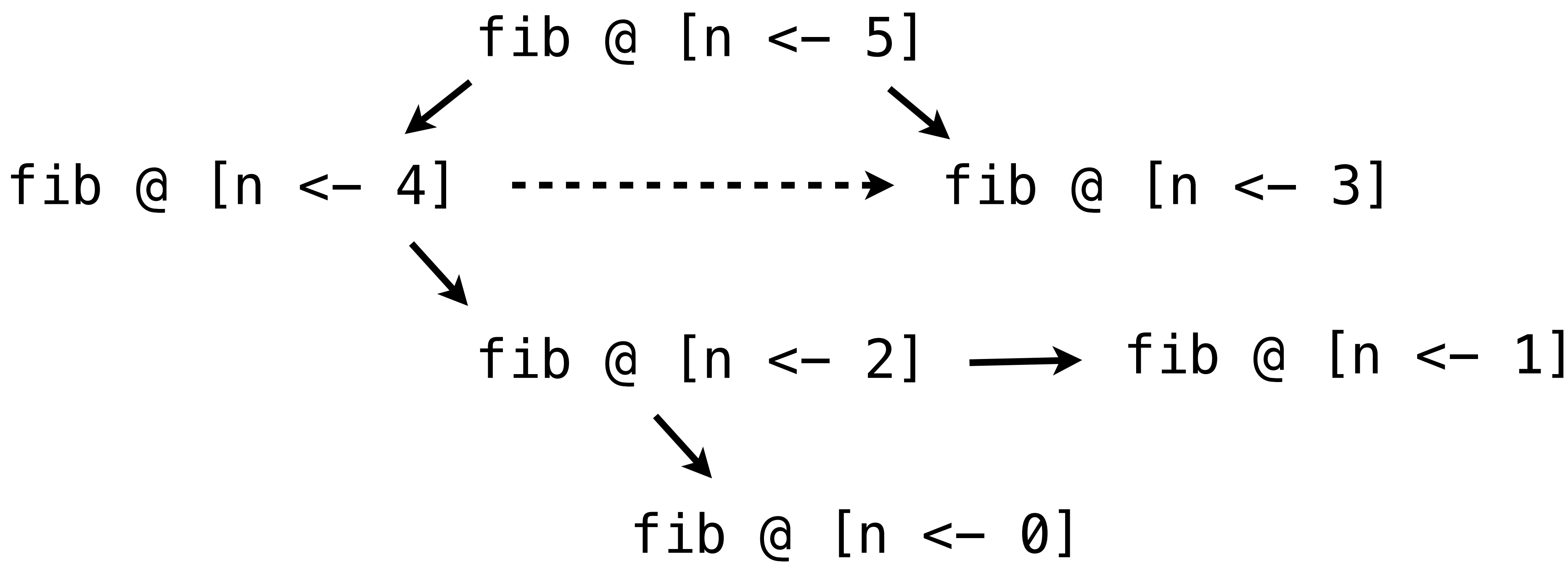
Fibonacci

Demand:



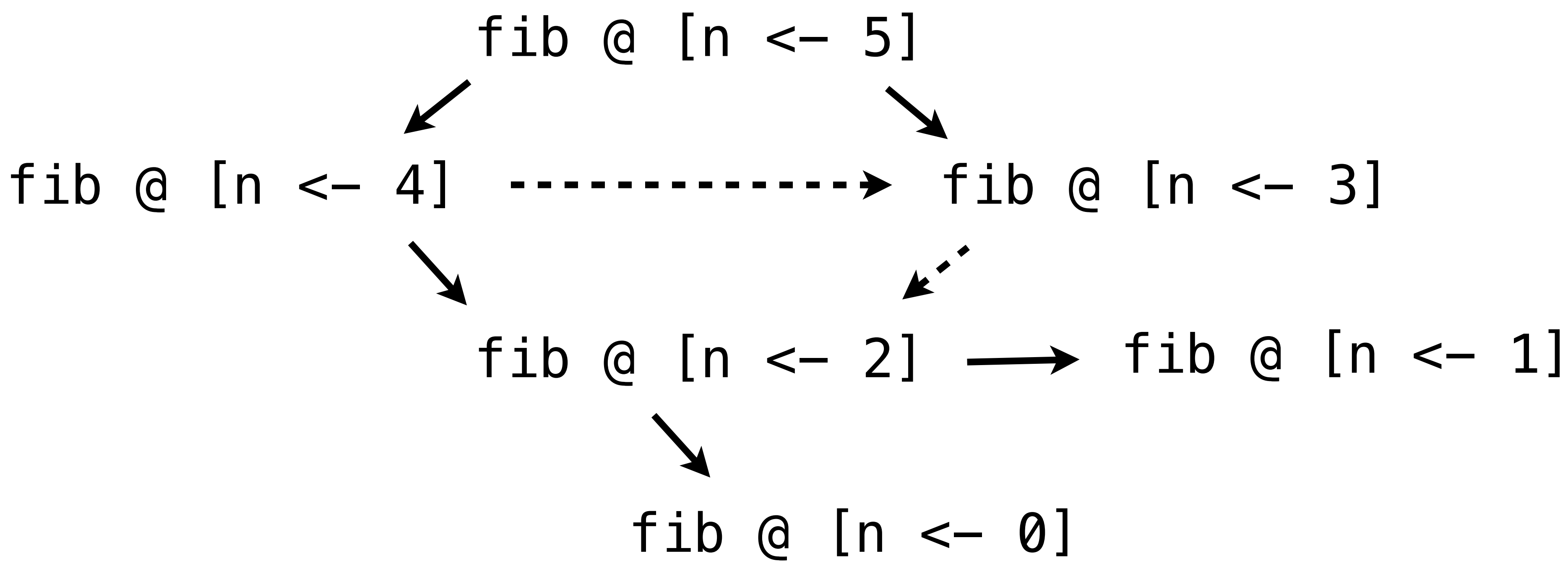
Fibonacci

Demand:



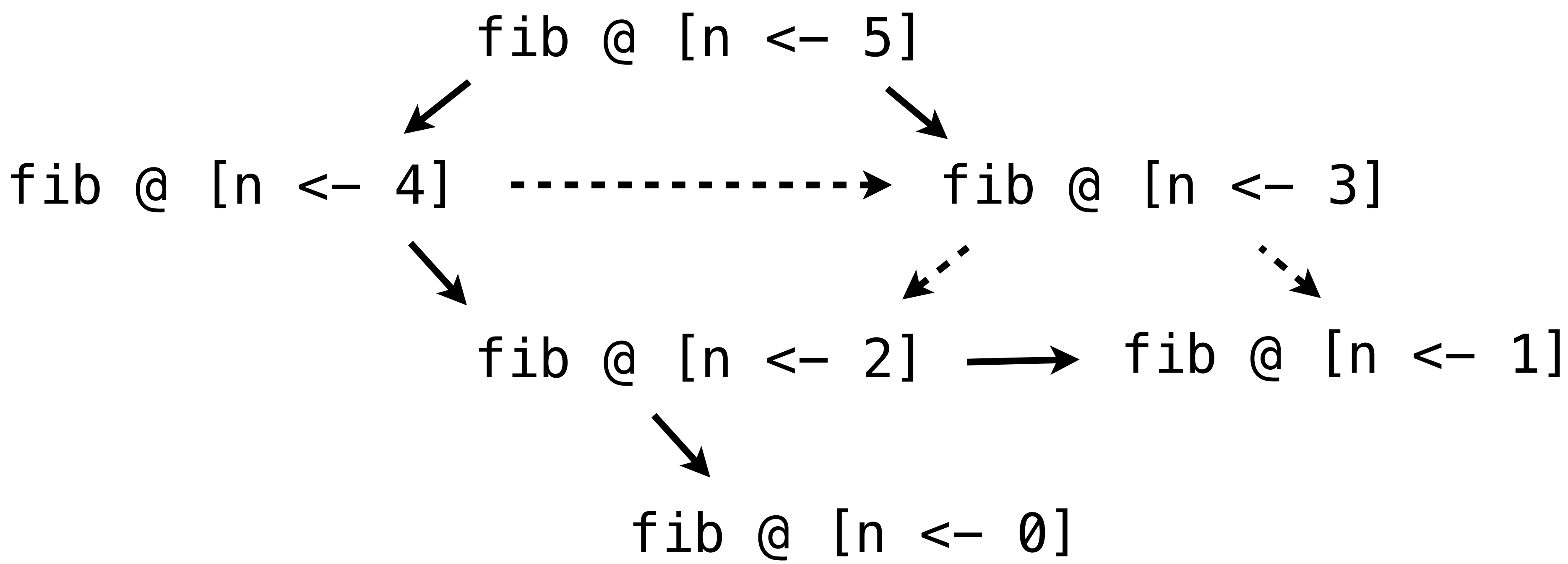
Fibonacci

Demand:



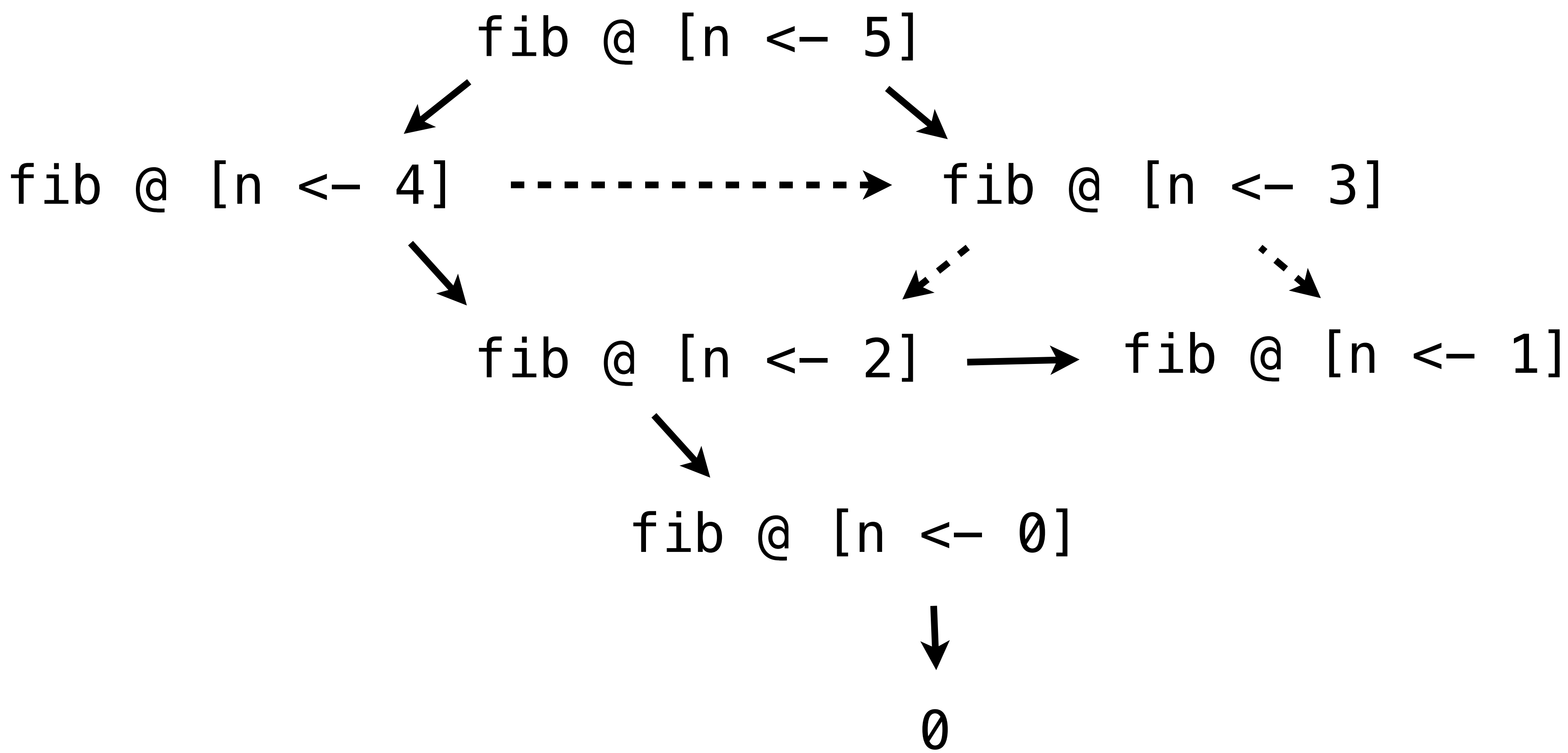
Fibonacci

Demand:



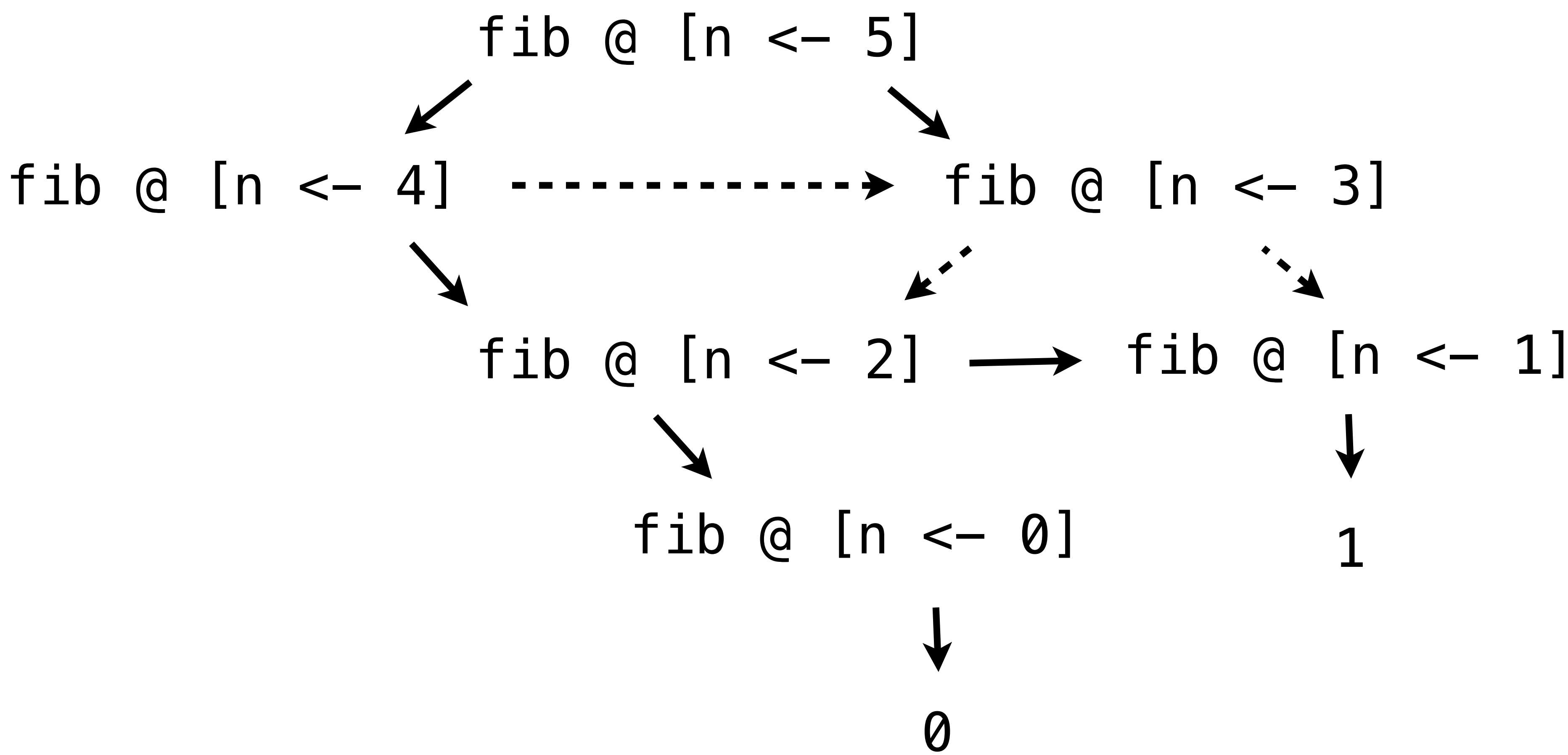
Fibonacci

Demand:



Fibonacci

Demand:



LaPlacian Relaxation

What is the field strength here?

LaPlacian Relaxation

Equation:

```
S where
  var S = if ELECTRODE then POTENTIAL
          else fby.t 0 (avg S)
          fi
  fun avg A = (prev.x A + next.x A +
               prev.y A + next.y A) / 4
```

LaPlacian Relaxation

Electrode at (3,4)
with potential 5

LaPlacian Relaxation

```
S@[x<-4, y<-4, t<-2]
```

Electrode at (3,4)
with potential 5

LaPlacian Relaxation

Electrode at (3,4) with potential 5

```
S@[x<-4, y<-4, t<-2]
        ↓
(avg S)@[x<-4, y<-4, t<-1]
```

LaPlacian Relaxation

Electrode at (3,4)
with potential 5

```
S@[x<-4, y<-4, t<-2]
```

↓

```
(avg S)@[x<-4, y<-4, t<-1]
```

↓

```
(S@[x<-3, y<-4, t<-1]+ S@[x<-5, y<-4, t<-1]+
S@[x<-4, y<-3, t<-1]+ S@[x<-4, y<-5, t<-1])/4
```

LaPlacian Relaxation

Electrode at (3,4)
with potential 5

```
S@[x<-4, y<-4, t<-2]
            ↓
(avg S)@[x<-4, y<-4, t<-1]
            ↓
(S@[x<-3, y<-4, t<-1]+ S@[x<-5, y<-4, t<-1]+
S@[x<-4, y<-3, t<-1]+ S@[x<-4, y<-5, t<-1])/4
            ↓
(5 + (avg S)@[x<-5, y<-4, t<-0]+
(avg S)@[x<-4, y<-3, t<-0]+(avg S)@[x<-4, y<-5, t<-0])/4
```

LaPlacian Relaxation

Electrode at (3,4)
with potential 5

```
S@[x<-4, y<-4, t<-2]
↓
(avg S)@[x<-4, y<-4, t<-1]
↓
(S@[x<-3, y<-4, t<-1]+ S@[x<-5, y<-4, t<-1]+
S@[x<-4, y<-3, t<-1]+ S@[x<-4, y<-5, t<-1])/4
↓
(5 + (avg S)@[x<-5, y<-4, t<-0]+
(avg S)@[x<-4, y<-3, t<-0]+(avg S)@[x<-4, y<-5, t<-0])/4
↓
(5 + (S@[x<-4, y<-4, t<-0] +
S@[x<-6,y<-4,t<-0]+S@[x<-5,y<-3,t<-0]+S@[x<-5,y<-5,t<-0])
)/4+...)/4
```

LaPlacian Relaxation

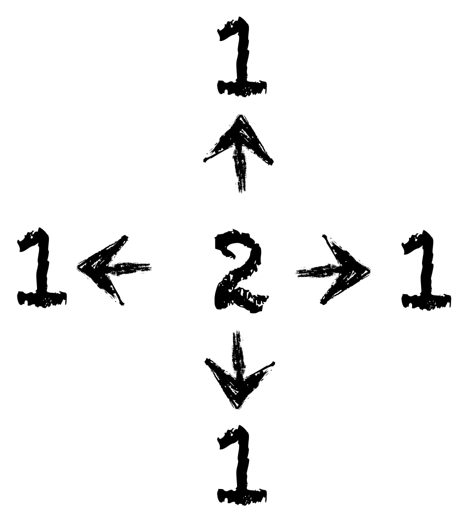
Electrode at (3,4)
with potential 5

```
S@[x<-4, y<-4, t<-2]
        ↓
(avg S)@[x<-4, y<-4, t<-1]
        ↓
(S@[x<-3, y<-4, t<-1]+ S@[x<-5, y<-4, t<-1]+
S@[x<-4, y<-3, t<-1]+ S@[x<-4, y<-5, t<-1])/4
        ↓
(5 + (avg S)@[x<-5, y<-4, t<-0]+
(avg S)@[x<-4, y<-3, t<-0]+(avg S)@[x<-4, y<-5, t<-0])/4
        ↓
(5 + (S@[x<-4, y<-4, t<-0] +
S@[x<-6,y<-4,t<-0]+S@[x<-5,y<-3,t<-0]+S@[x<-5,y<-5,t<-0])
)/4+...)/4
        ↓
(5 + (0+0+0+0)/4+...)/4
```

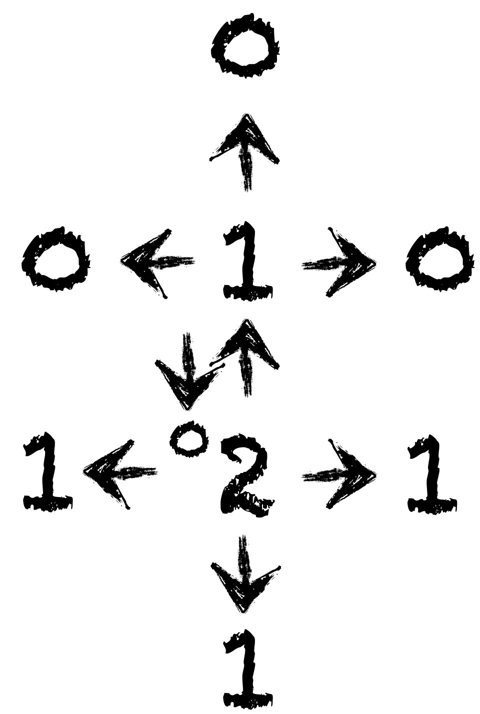
LaPlacian Relaxation

2

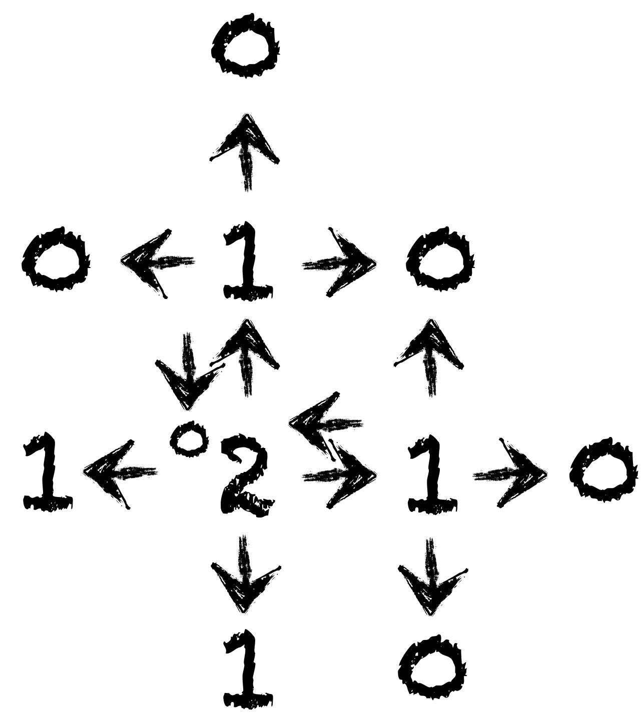
LaPlacian Relaxation



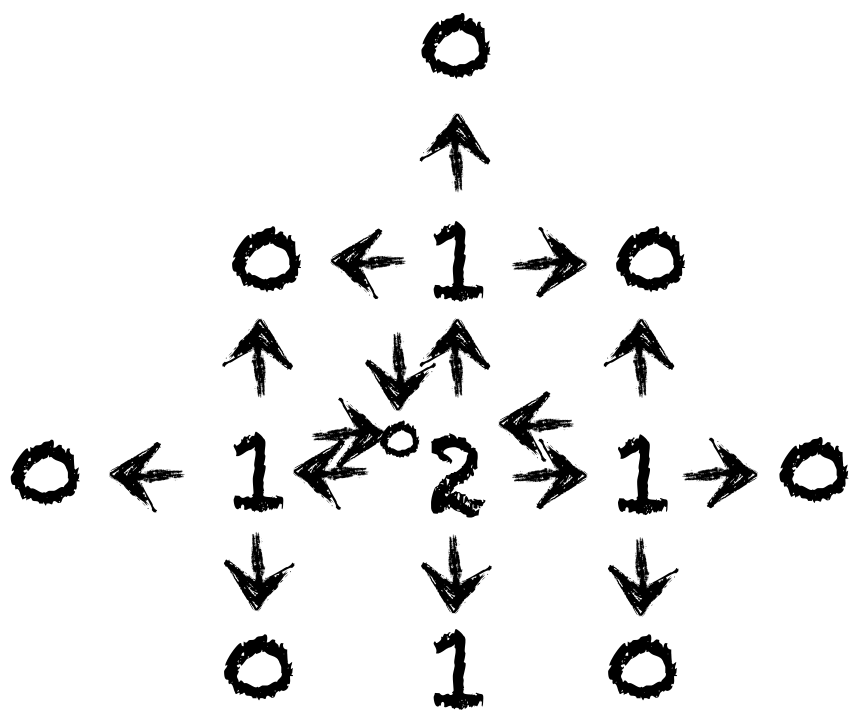
LaPlacian Relaxation



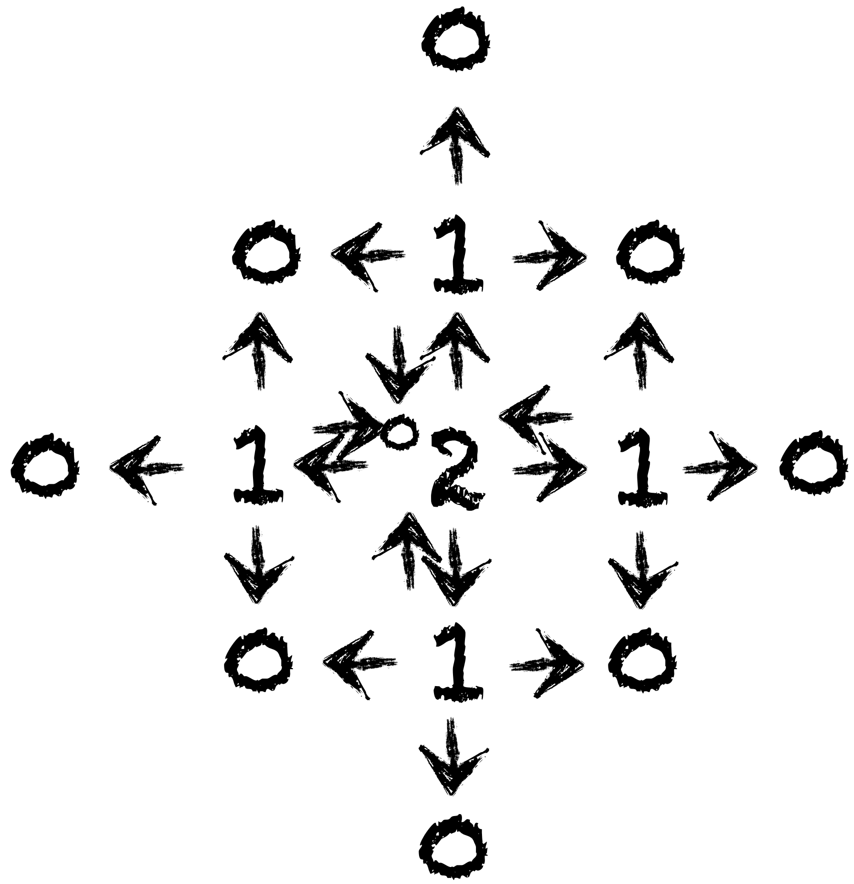
LaPlacian Relaxation



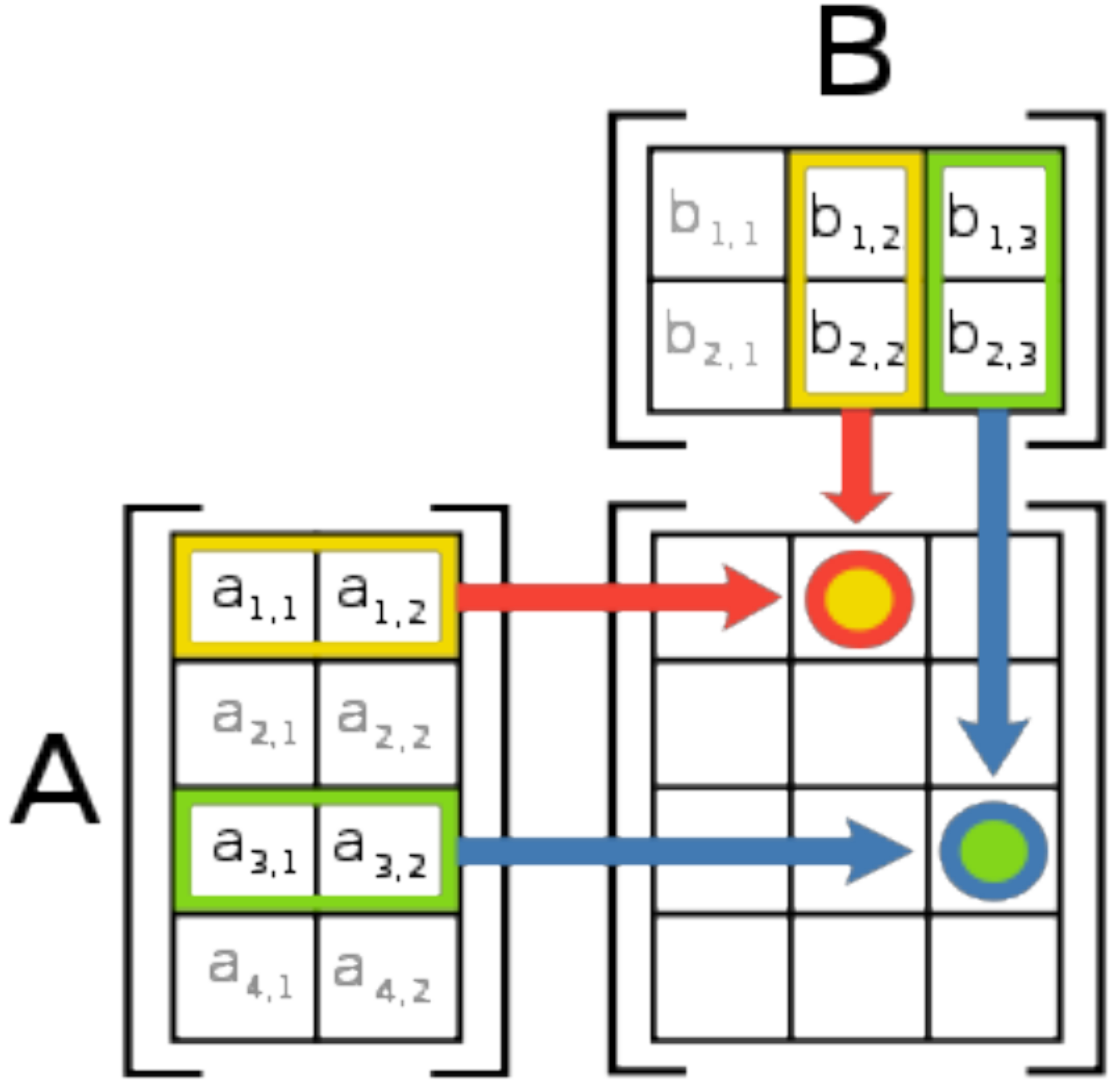
LaPlacian Relaxation



LaPlacian Relaxation



Matrix multiplication



Source: rosalind.info

Matrix mult in ICE

```
fun multiply.d_r.d_c.k X Y = W
      where
        dim d <- 0
        var Xd = rotate.d_c.d X
        var Yd = rotate.d_r.d Y
        var Z = Xd * Yd
        var W = sum.d.k Z
      end
```

Matrix mult in ICE

```
fun multiply.d_r.d_c.k X Y = W
      where
        dim d <- 0
        var Xd = rotate.d_c.d X
        var Yd = rotate.d_r.d Y
        var Z = Xd * Yd
        var W = sum.d.k Z
      end
```

```
fun sum.d_x.n X = Y @ [d_x <- n]
  where
    var Y = fby.d_x 0 (X + Y)
  end
```

Status

Status

Scanner and parser works

Status

Scanner and parser works

Evaluator near alpha level

Status

Scanner and parser works

Evaluator near alpha level

Cache at alpha level

Next Steps

Add I/O to ease big examples

Increase usability through use cases

- Options pricer and your cool case!

Efficient offloading to GPU et al

Parallelise evaluator using ParaPhrase tools

Release beta-version (target Feb 2014)