Developing Reasonable Programs

Matt Might

University of Utah

matt.might.net

The future of...

The future of...

...programs

The future of...

...programs

...languages

The future of...

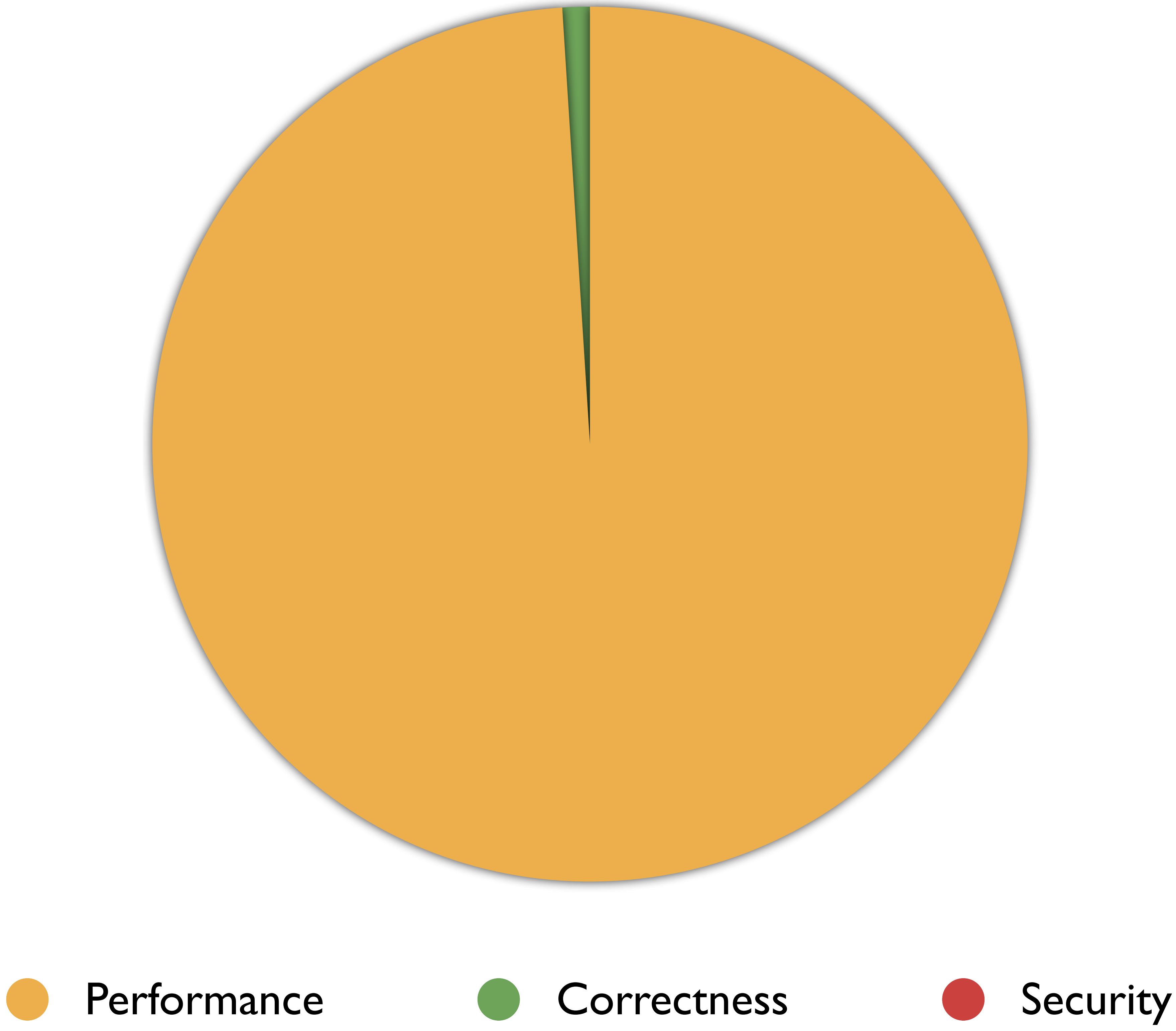
...programs

...languages

...compilers

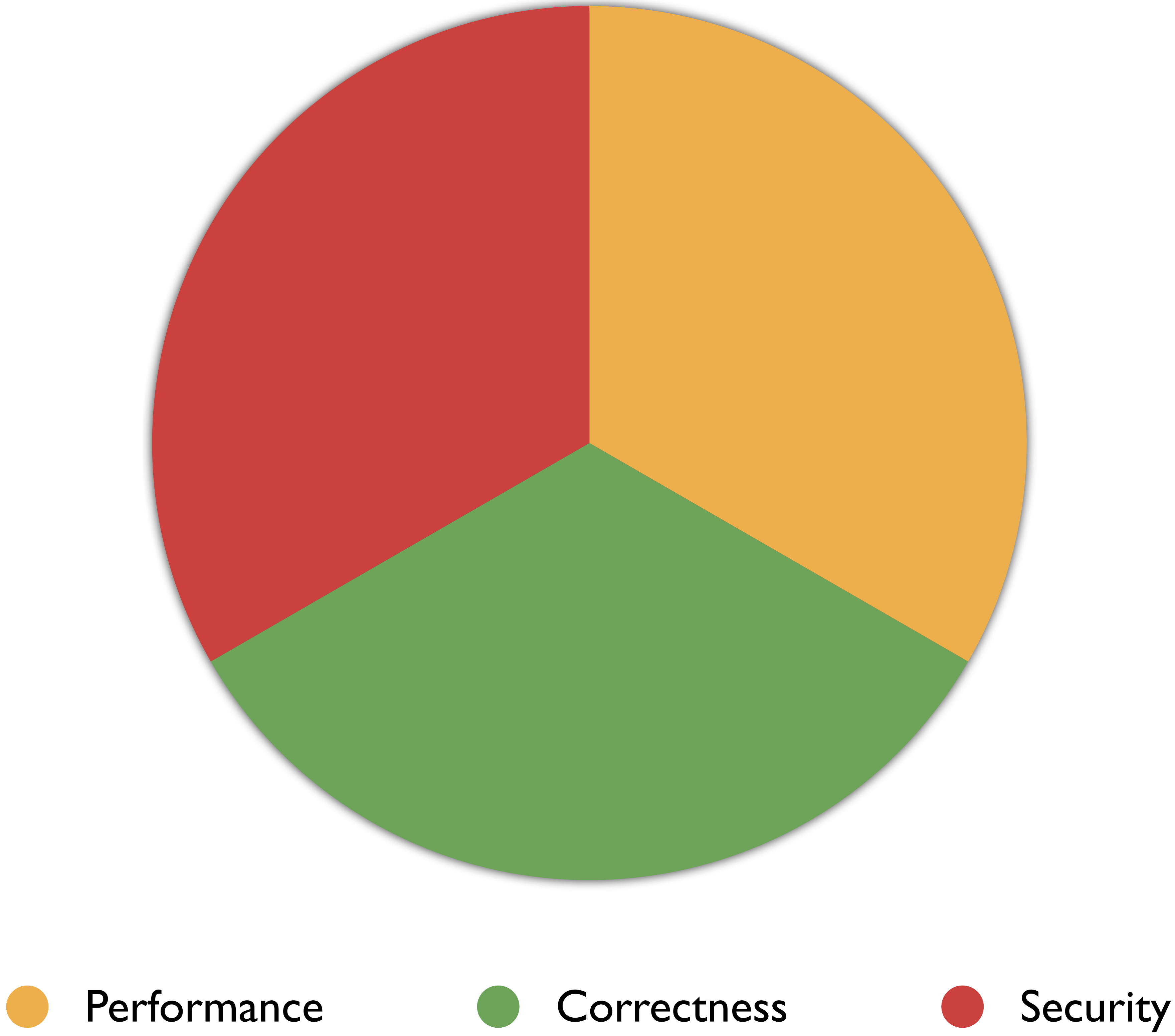
WARNING

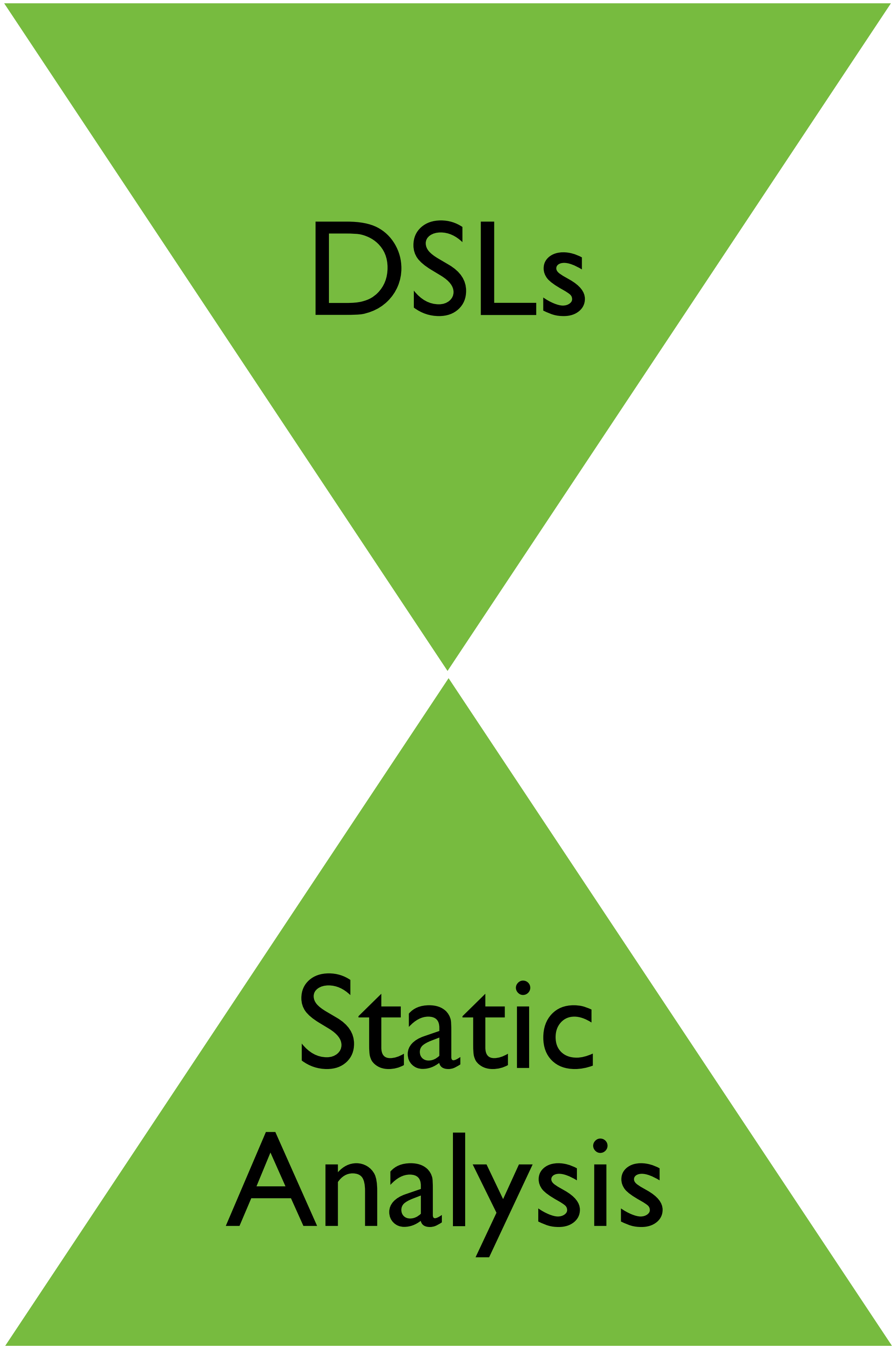
1946



201X

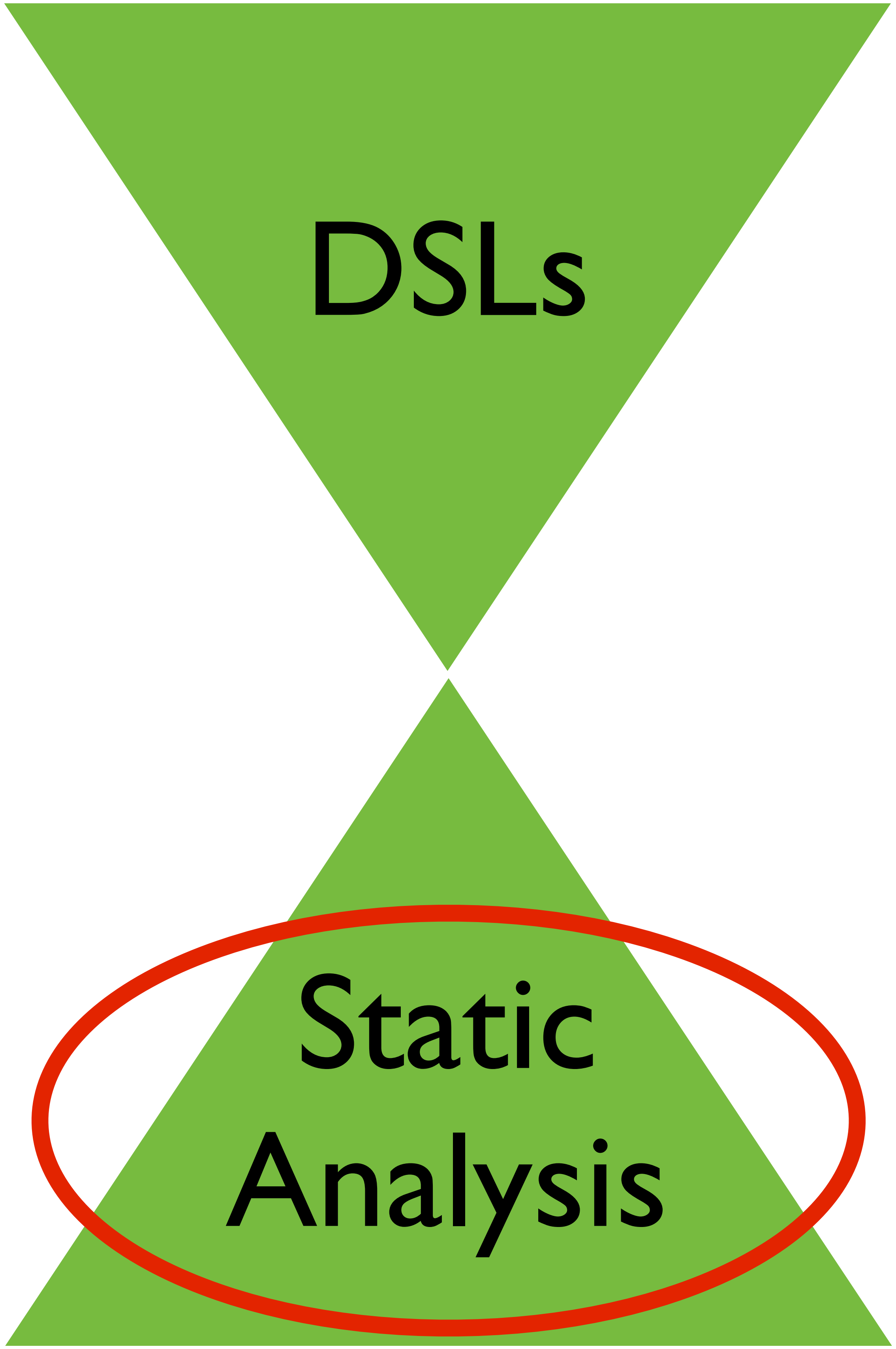
201X





DSLs

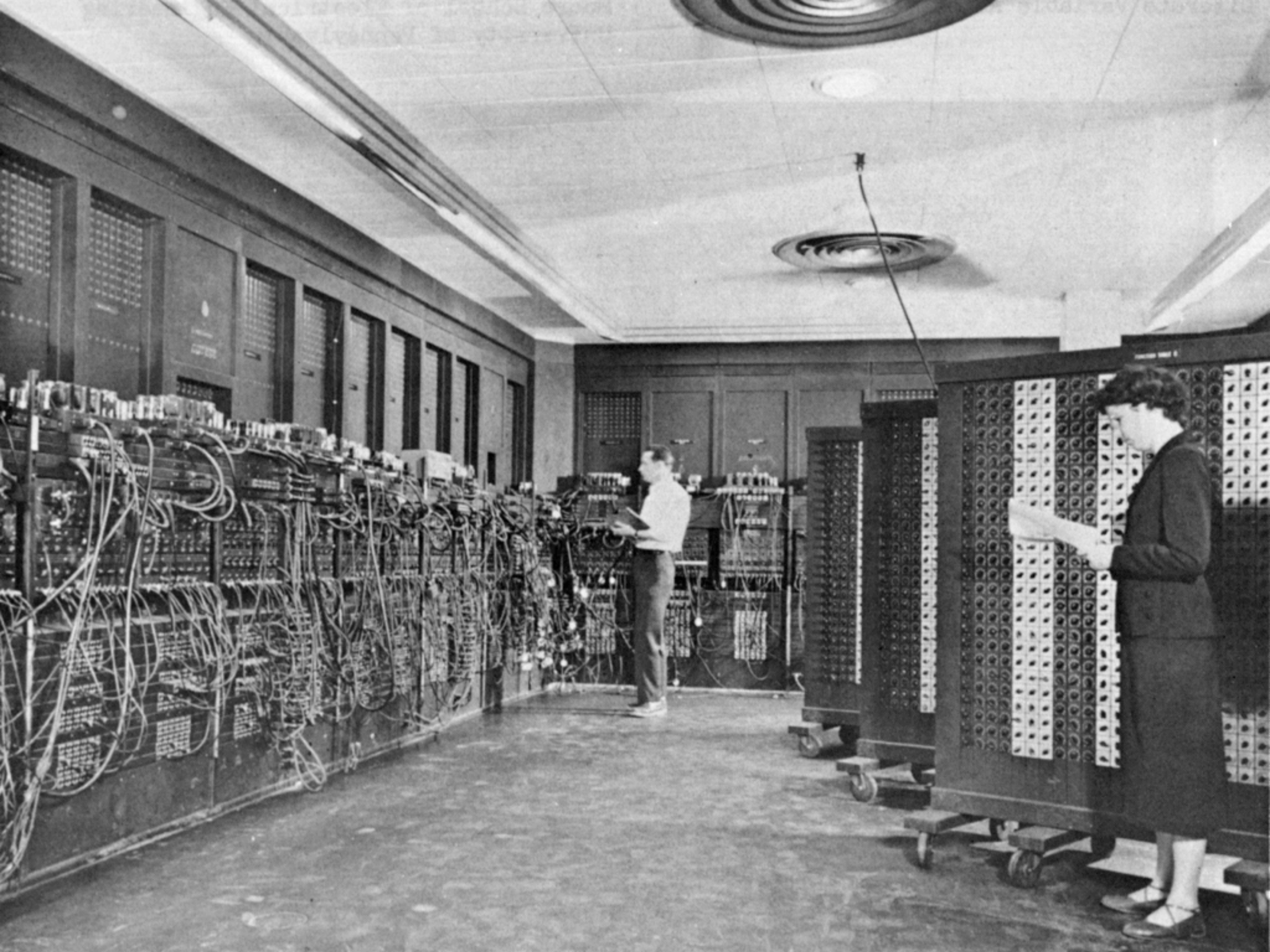
Static Analysis

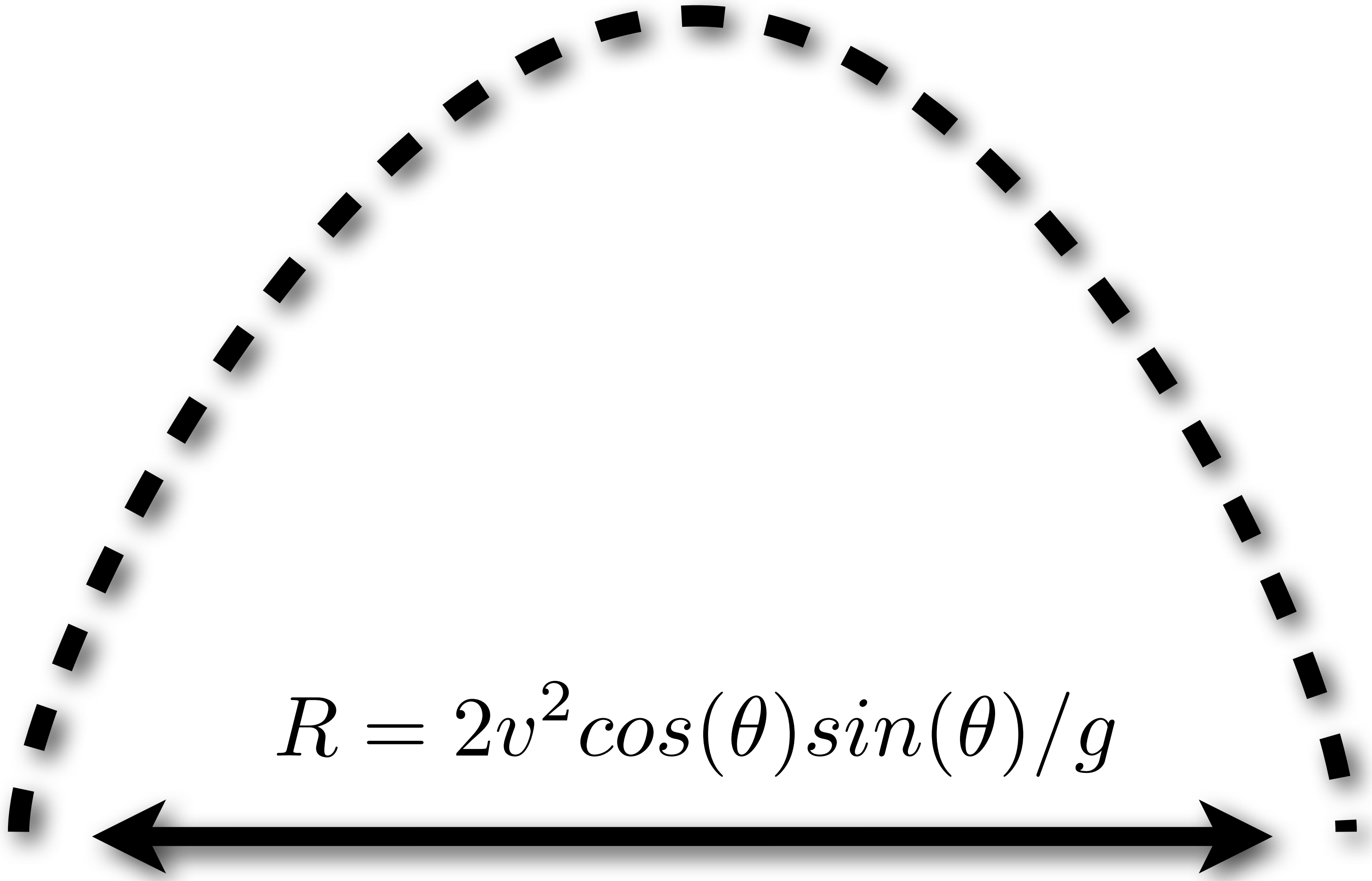


DSLs

Static Analysis

</shortversion>





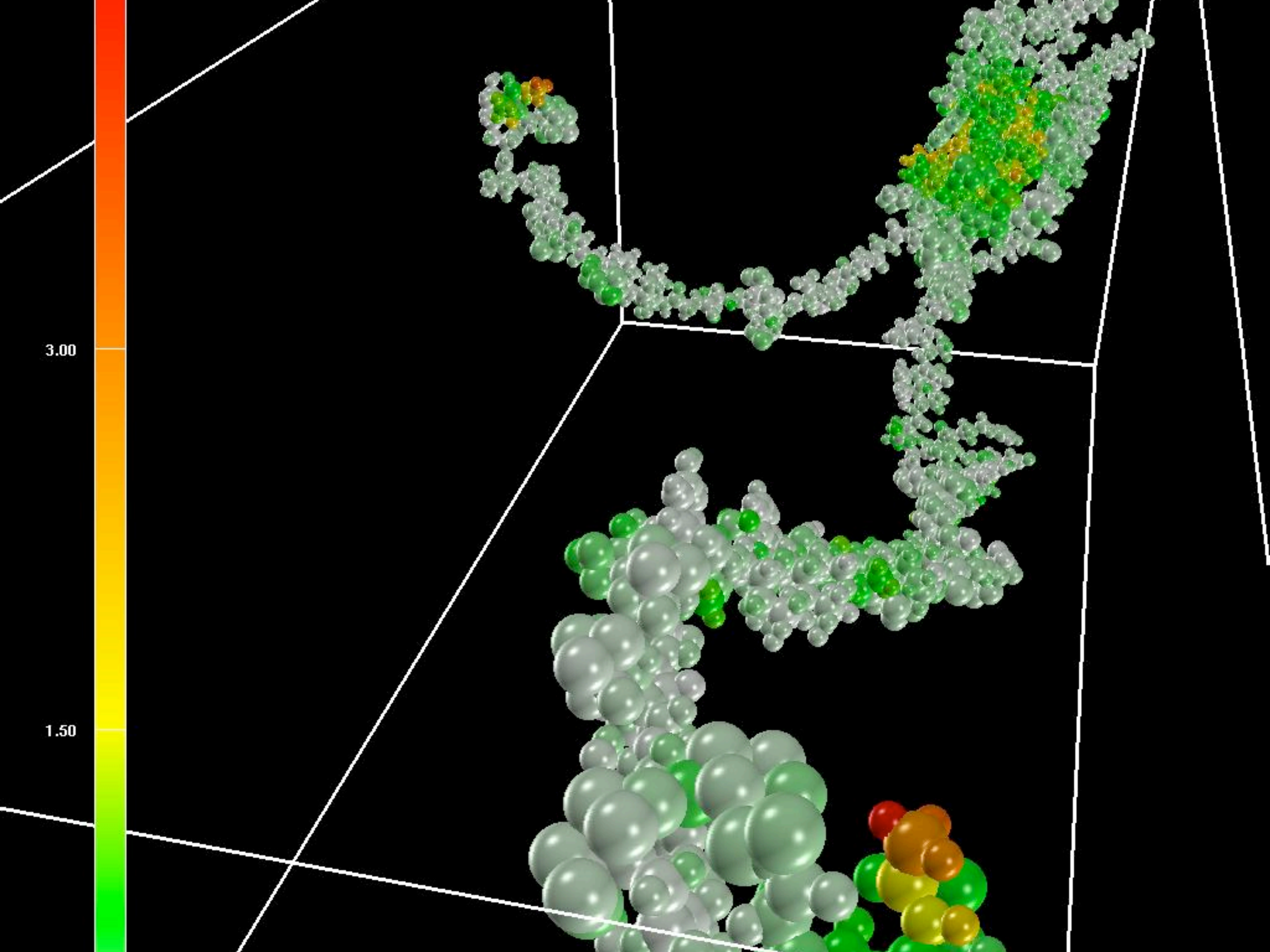
$$R = 2v^2 cos(\theta) sin(\theta)/g$$

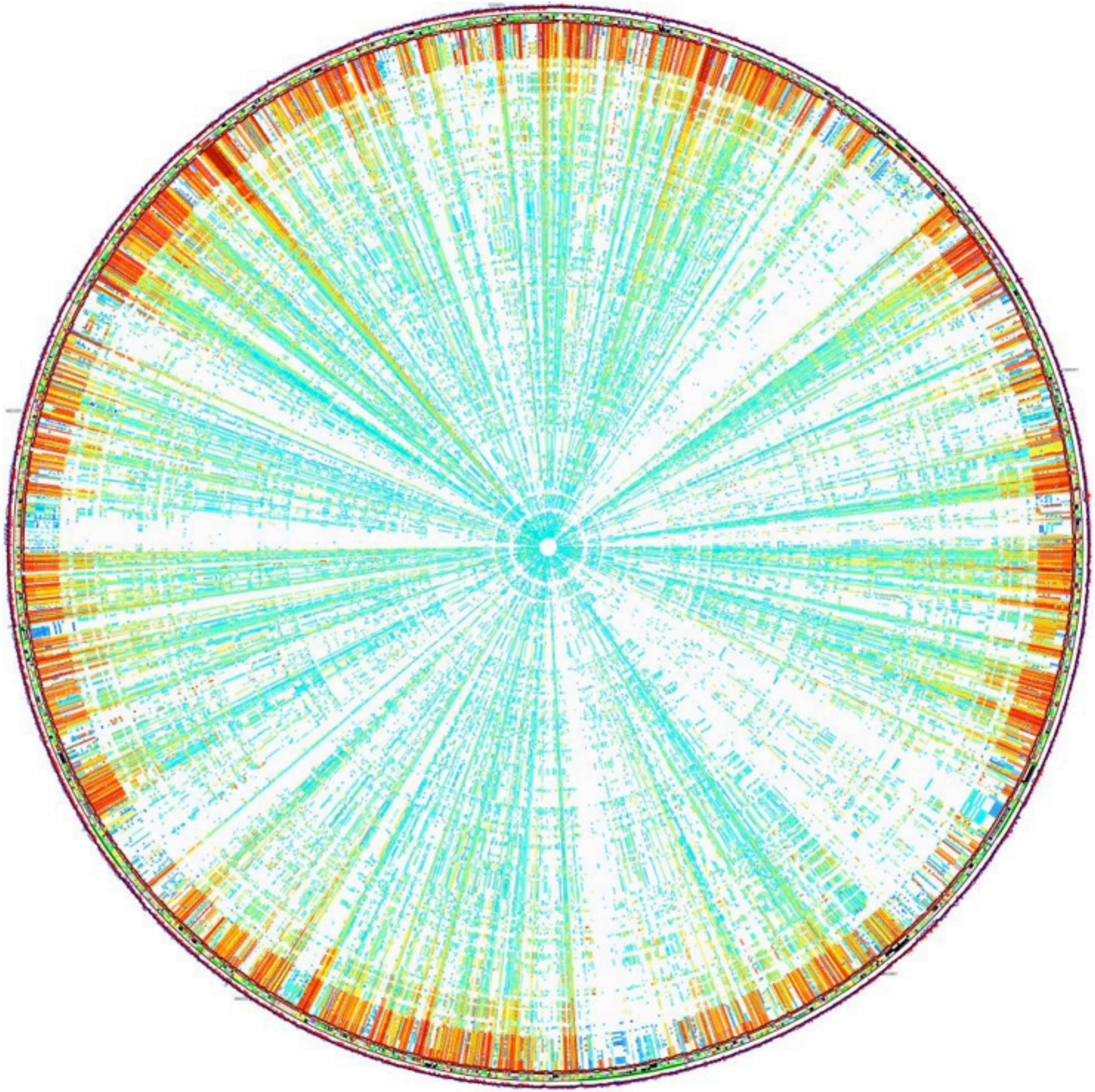
35 divisions per second.

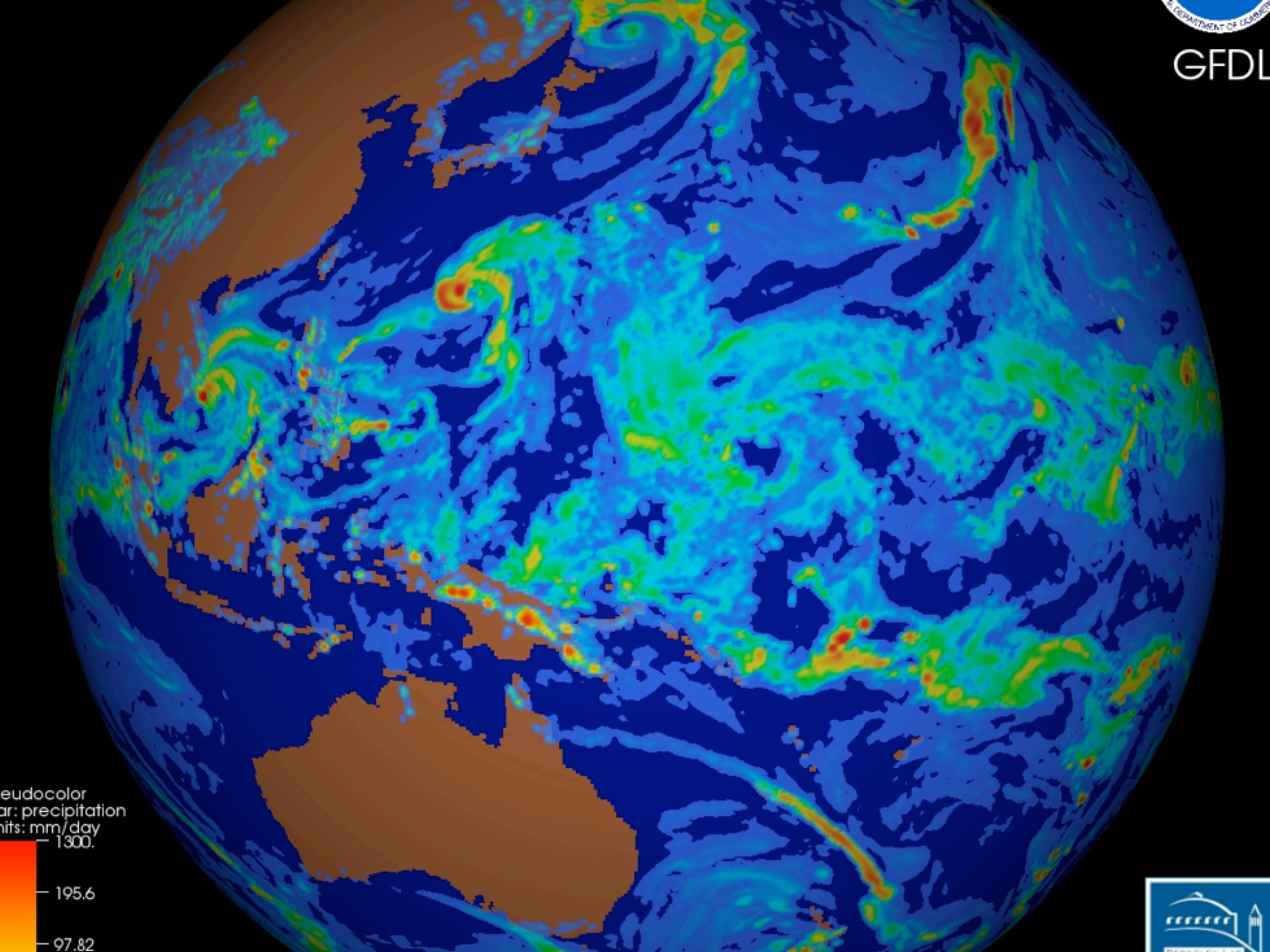
2.9 divisions per second.

Performance mattered.

IBM
System x3755

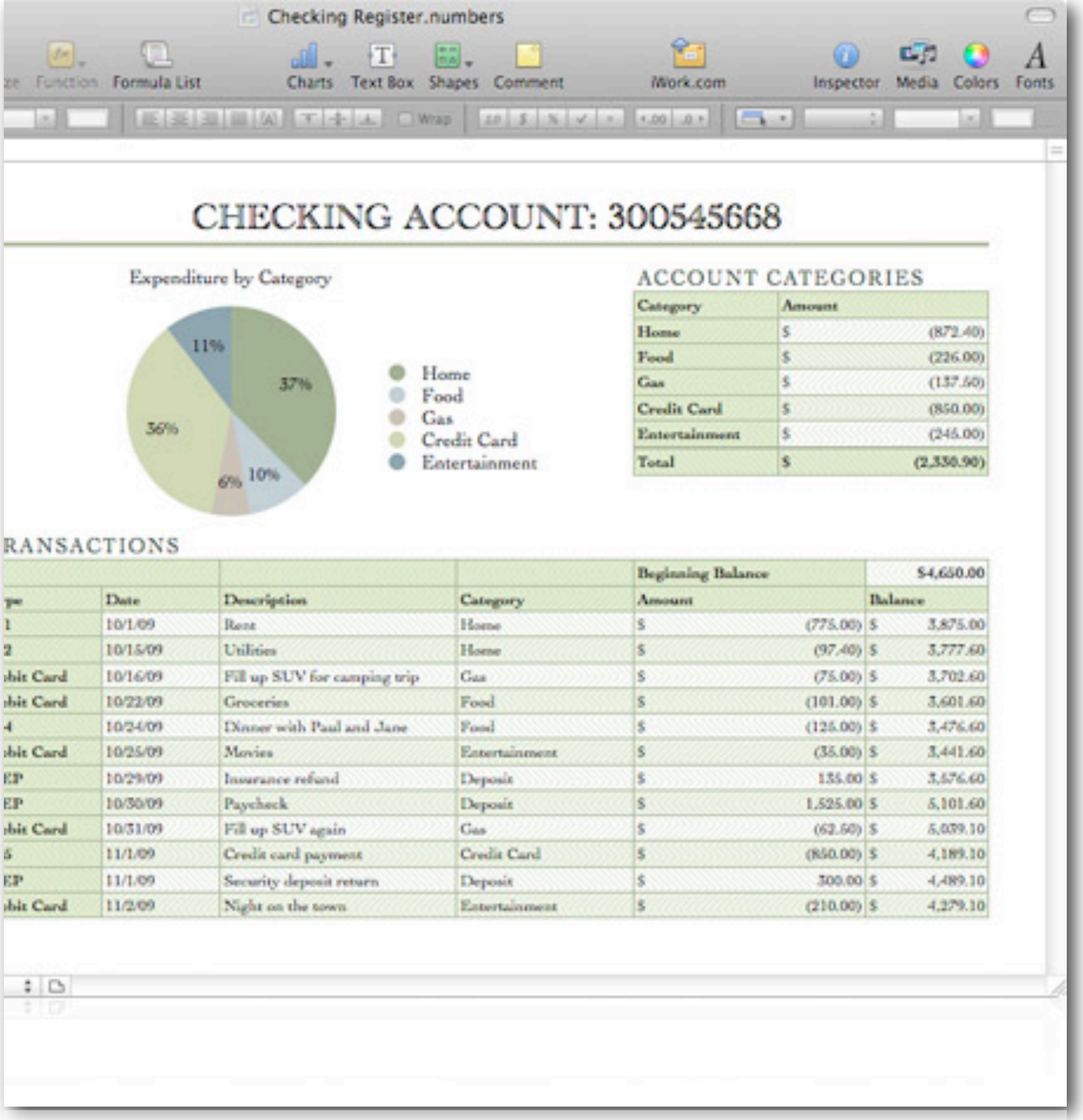

3.00
1.50




GFDL
eudocolor
ar: precipitation
nits: mm/day
1300.
195.6
97.82

Performance still matters.

DOOR OPEN
SELCAL
FIRE
TORQUE
RPM
TIT (°C)
FUEL FLOW (PPH)
OIL TEMP (°C)
GB PRES (PSI)
OIL PRES (PSI)
OIL QTY (GAL)
COOL FLAP
FMS1
FMS2
TERRAIN
BRAKES
AUXILIARY
UTILITY
ANTI SKID
LEFT
NOSE
RIGHT
LOCK RELEASE
HORN SILENCE
HANDLE LIGHT TEST
UP
DN
FLAPS
1
2
3
4



CHECKING ACCOUNT: 300545668

ACCOUNT CATEGORIES

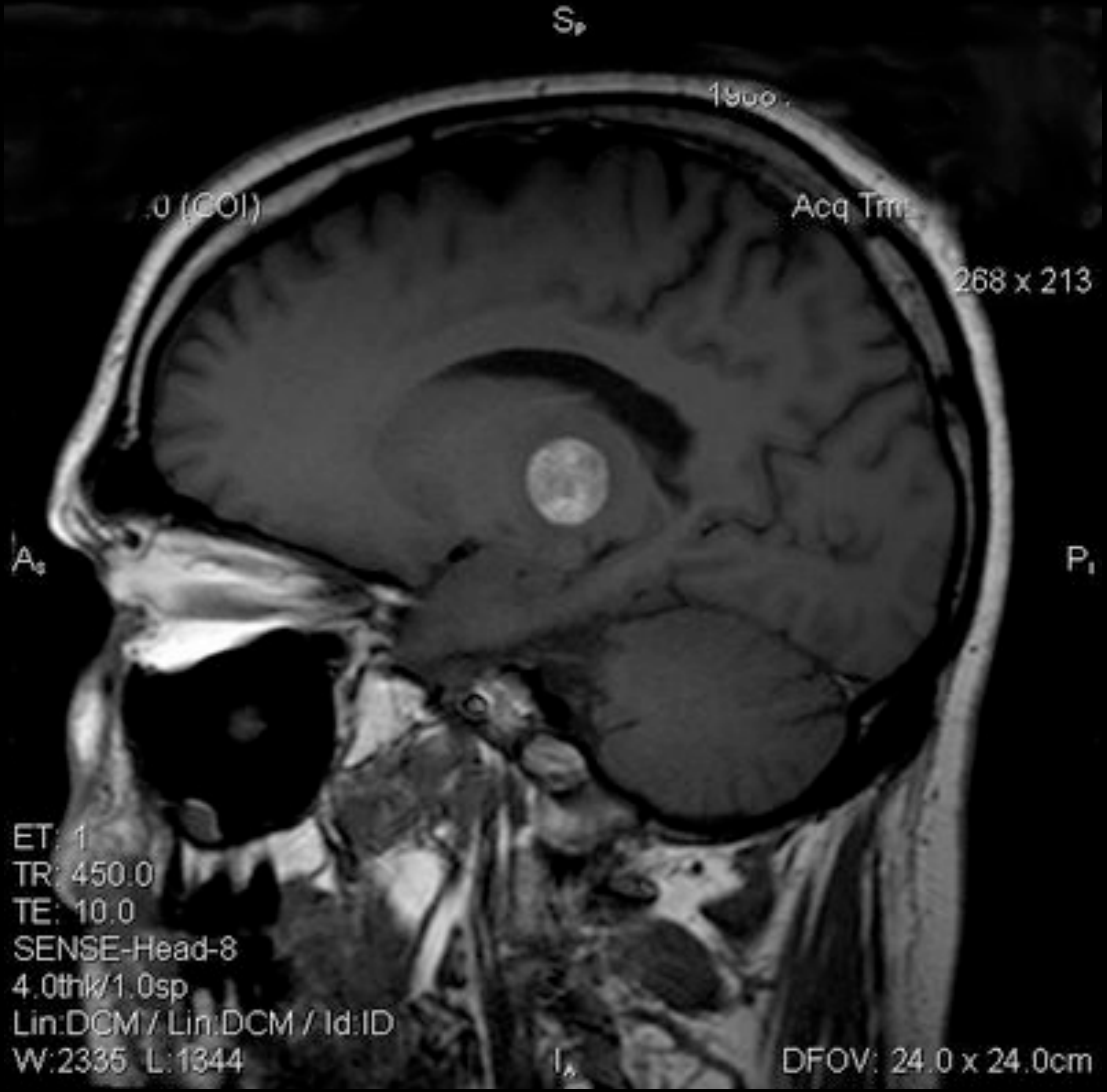
Category	Amount	
Home	$	(872.40)
Food	$	(226.00)
Gas	$	(137.50)
Credit Card	$	(850.00)
Entertainment	$	(245.00)
Total	$	(2,330.90)

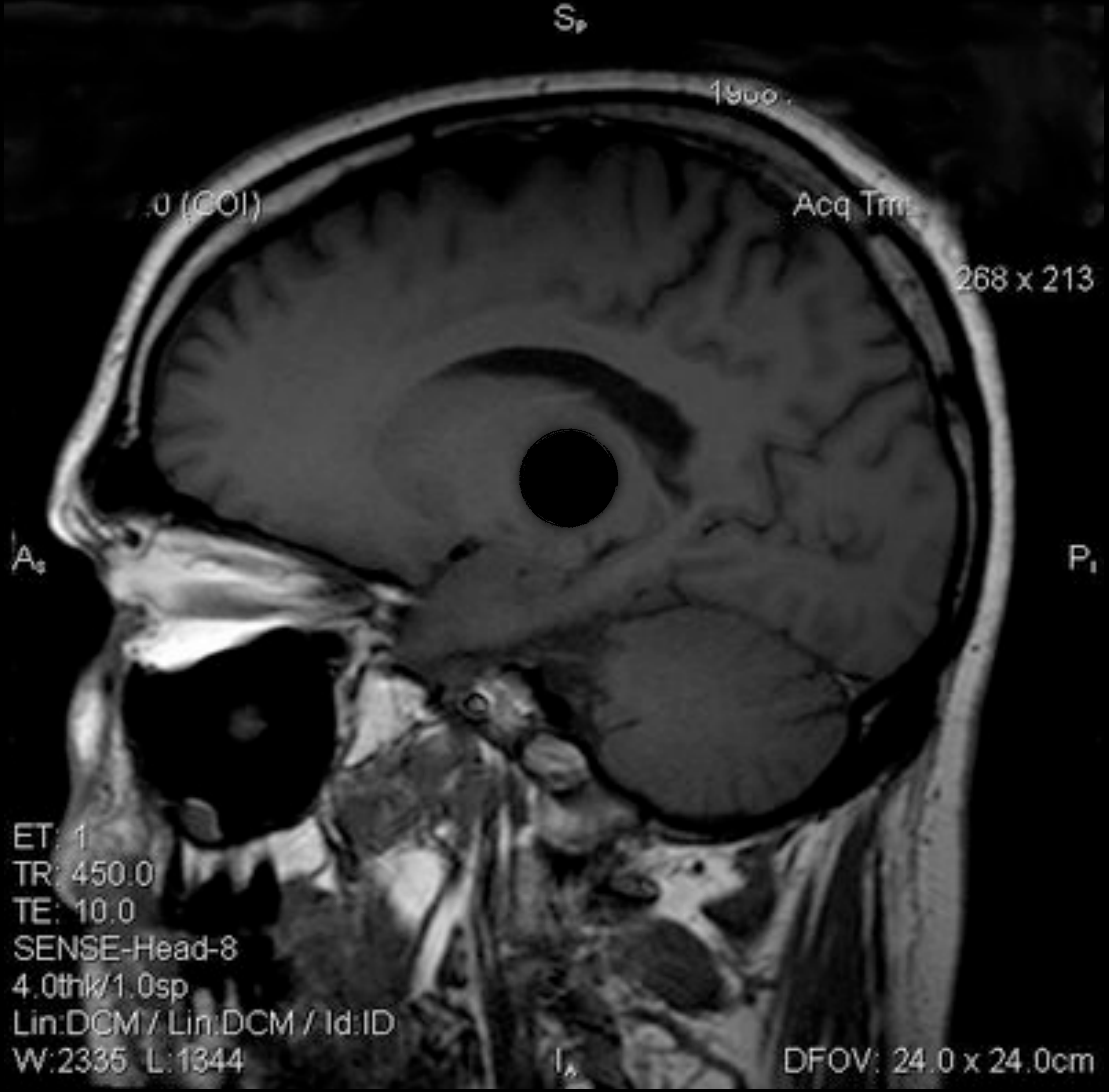
RANSACTIONS

				Beginning Balance			$4,650.00
pe	Date	Description	Category	Amount		Balance	
1	10/1/09	Rent	Home	$	(775.00)	$	3,875.00
2	10/15/09	Utilities	Home	$	(97.40)	$	3,777.60
bit Card	10/16/09	Fill up SUV for camping trip	Gas	$	(75.00)	$	3,702.60
bit Card	10/22/09	Groceries	Food	$	(101.00)	$	3,601.60
4	10/24/09	Dinner with Paul and Jane	Food	$	(125.00)	$	3,476.60
bit Card	10/25/09	Movies	Entertainment	$	(35.00)	$	3,441.60
EP	10/29/09	Insurance refund	Deposit	$	135.00	$	3,576.60
EP	10/30/09	Paycheck	Deposit	$	1,525.00	$	5,101.60
bit Card	10/31/09	Fill up SUV again	Gas	$	(62.50)	$	5,039.10
5	11/1/09	Credit card payment	Credit Card	$	(850.00)	$	4,189.10
EP	11/1/09	Security deposit return	Deposit	$	300.00	$	4,489.10
bit Card	11/2/09	Night on the town	Entertainment	$	(210.00)	$	4,279.10

Correctness matters.

SIEMENS

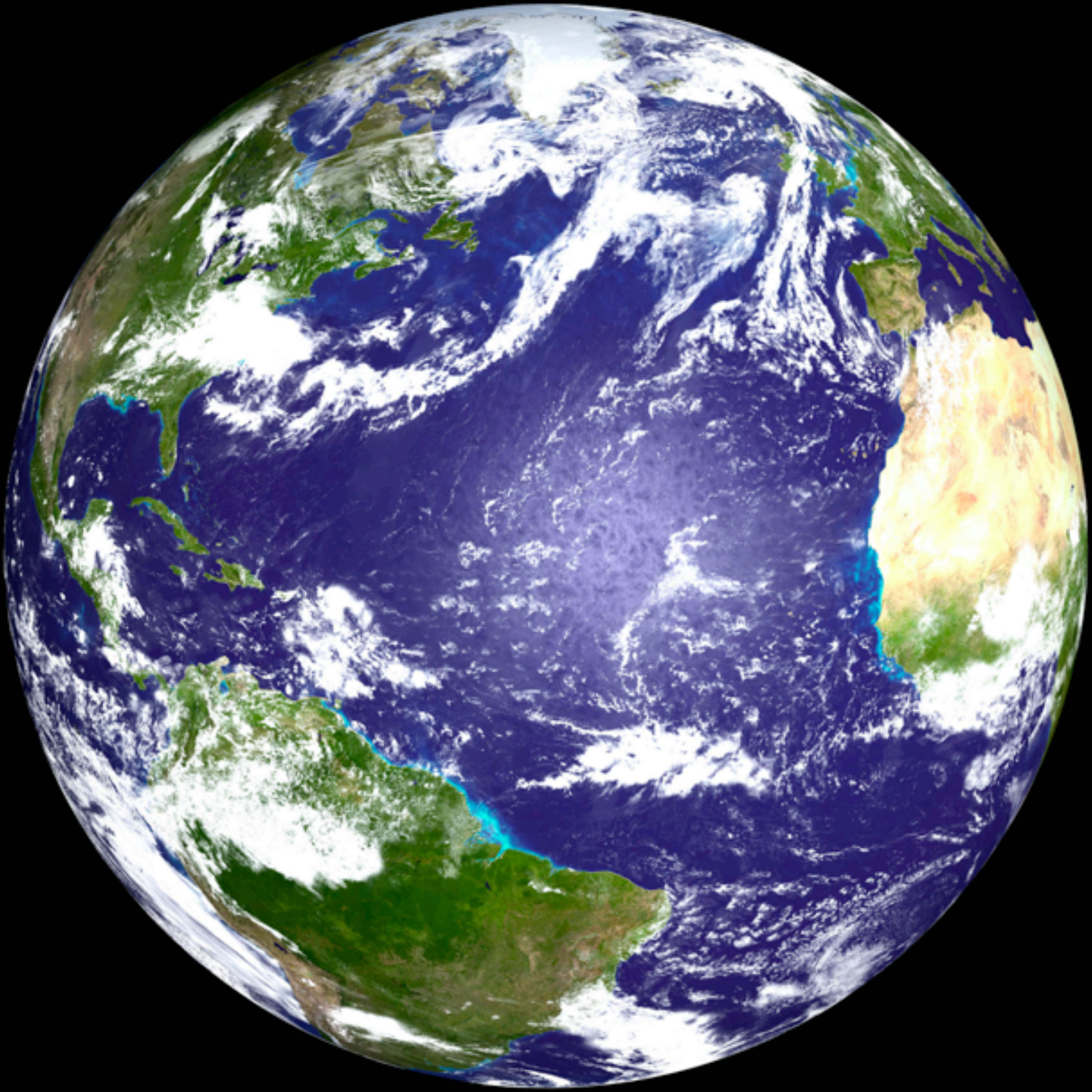

S
.0 (COI)
Acq Tm
268 x 213
A
P
ET: 1
TR: 450.0
TE: 10.0
SENSE-Head-8
4.0thk/1.0sp
Lin:DCM / Lin:DCM / Id:ID
W:2335 L:1344
I
DFOV: 24.0 x 24.0cm

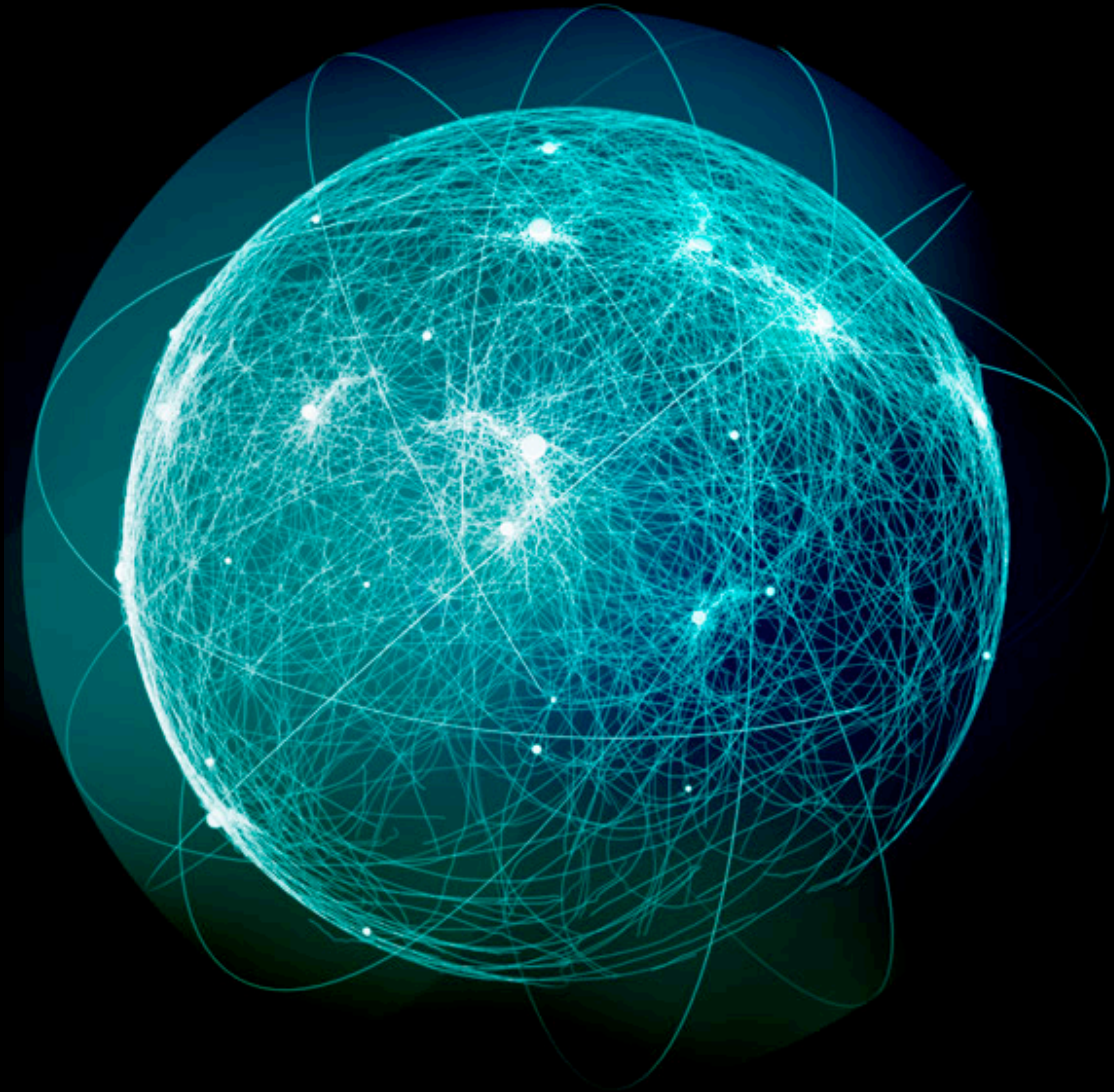

S
.0 (COI)
Acq Tm
268 x 213
A
P
ET: 1
TR: 450.0
TE: 10.0
SENSE-Head-8
4.0thk/1.0sp
Lin:DCM / Lin:DCM / Id:ID
W:2335 L:1344
I
DFOV: 24.0 x 24.0cm

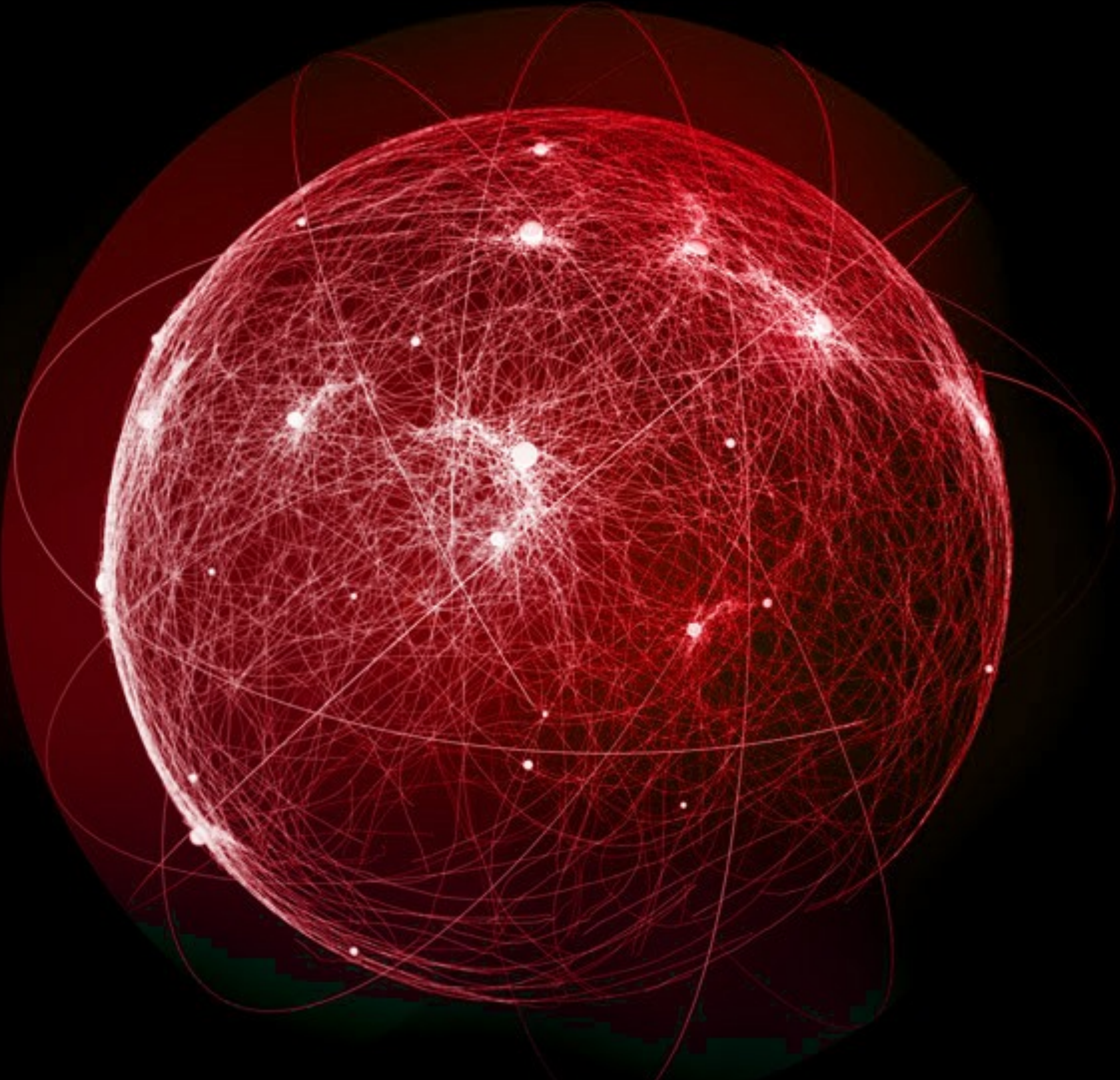
Correctness really matters.

JICO

ZULU TIME
AFNETOPS/CC Vision
Stripes Crew Assignments
AFNOC Mission
AFNetOps C2 Relationship







Security matters.

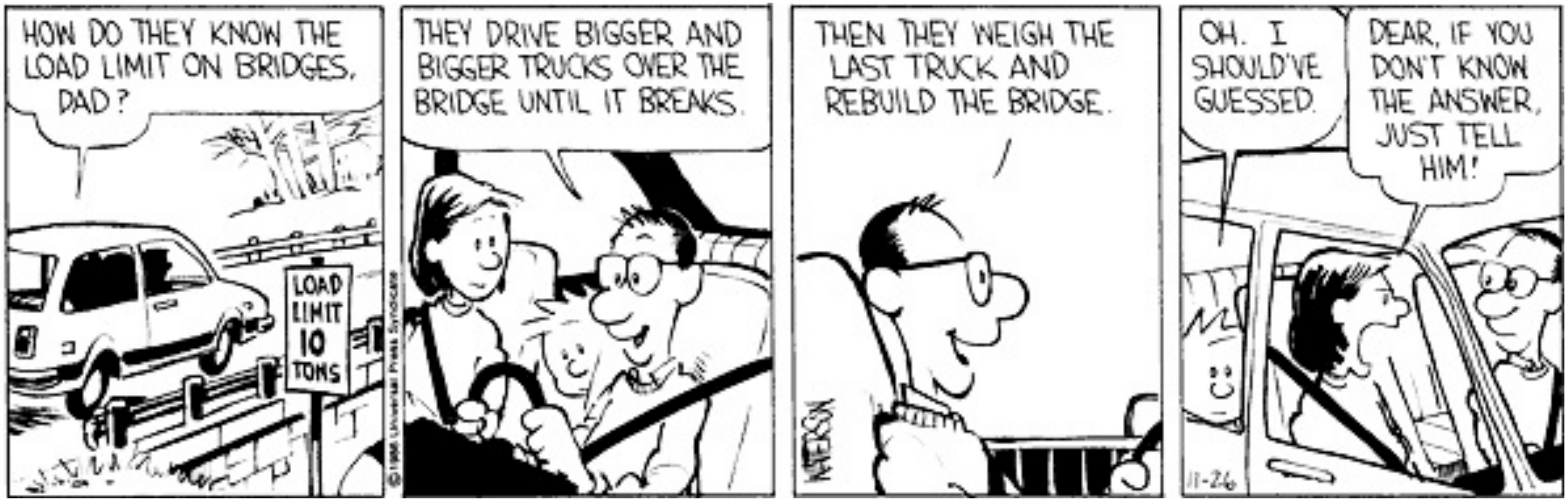
What makes software slow, buggy and insecure?

We can’t predict it.

We can’t reason.

We can’t engineer.

Software “engineering”

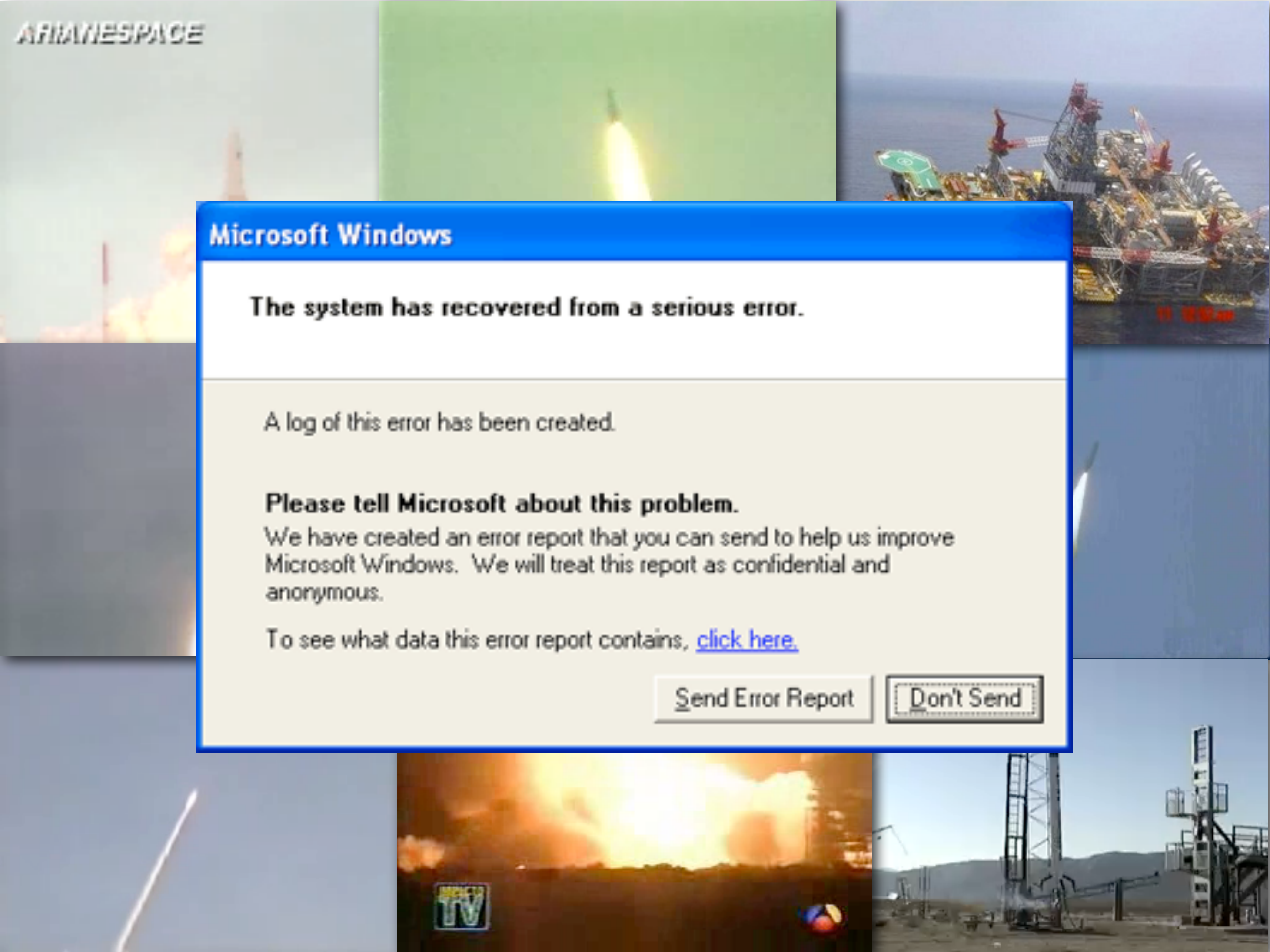

HOW DO THEY KNOW THE LOAD LIMIT ON BRIDGES, DAD?
LOAD LIMIT 10 TONS
THEY DRIVE BIGGER AND BIGGER TRUCKS OVER THE BRIDGE UNTIL IT BREAKS.
THEN THEY WEIGH THE LAST TRUCK AND REBUILD THE BRIDGE.
OH. I SHOULD'VE GUESSED.
DEAR, IF YOU DON'T KNOW THE ANSWER, JUST TELL HIM!
WATERSON
11-26

```
                         PowerPoint

A fatal exception 0E has occurred at 0137:BFFA21C9. The current
application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will
  lose any unsaved information in all applications.

                   Press any key to continue _
```



Microsoft Windows

The system has recovered from a serious error.

A log of this error has been created.

Please tell Microsoft about this problem.

We have created an error report that you can send to help us improve Microsoft Windows. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

We need engineering.

We need reasonable programs.

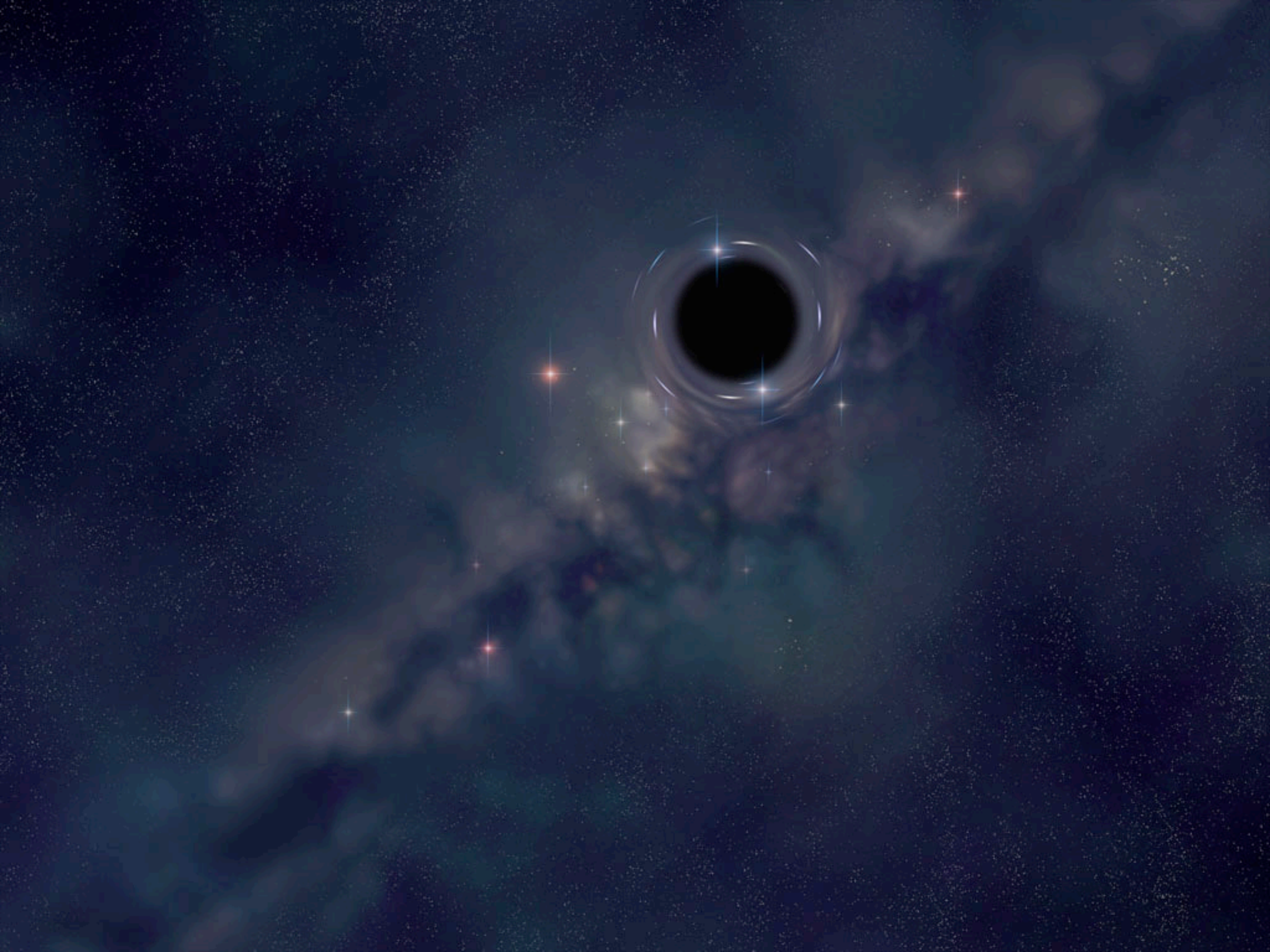
We need prediction.

So, why can’t we predict what software will do?

So, why can't we predict what software will do?

Because Alan Turing said we can't.

“Thou shalt not write a program which determines whether a program halts.”



while P(x)

Interesting question?

Interesting question?
Undecidable.

But,

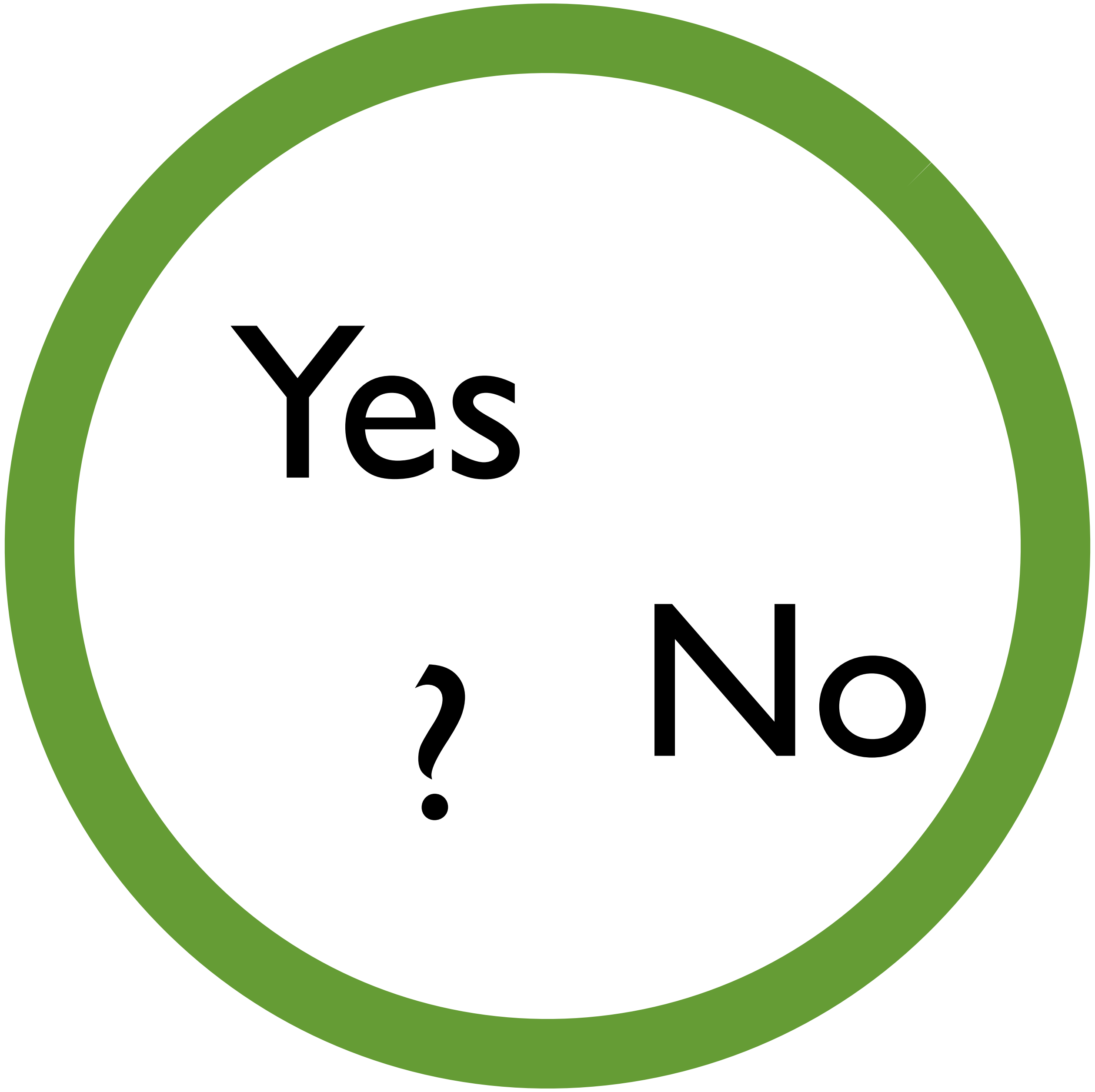
there's a loop hole...

there’s a loop hole...

...in the loop hole.

Yes

No



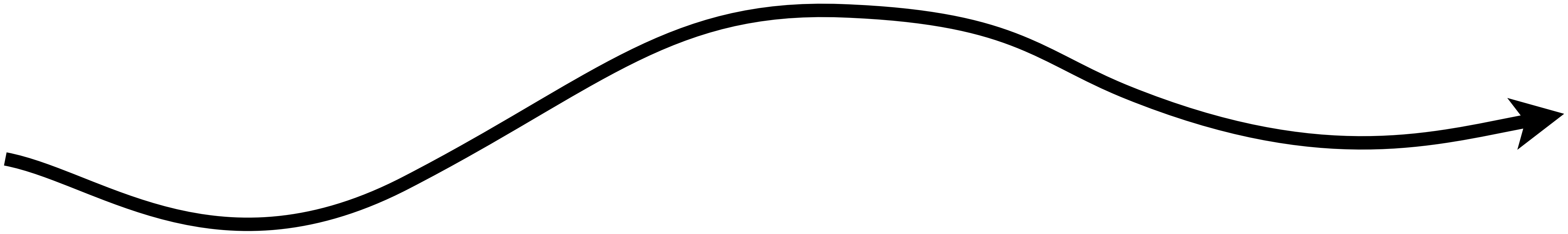
Yes

?

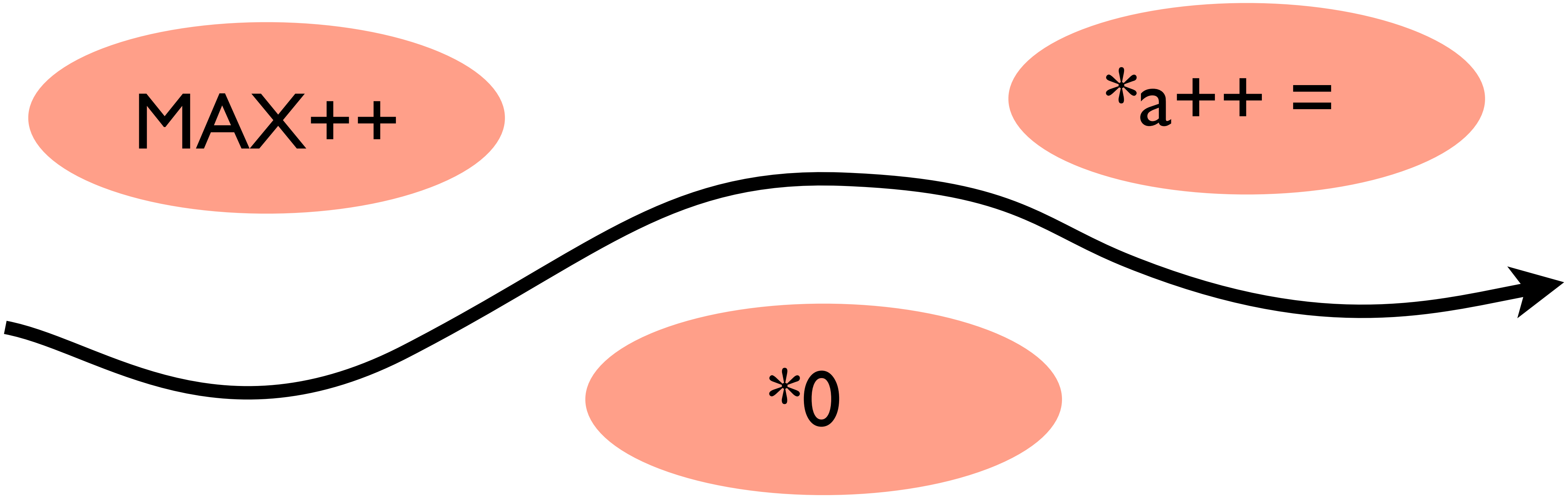
No

The static analysis game

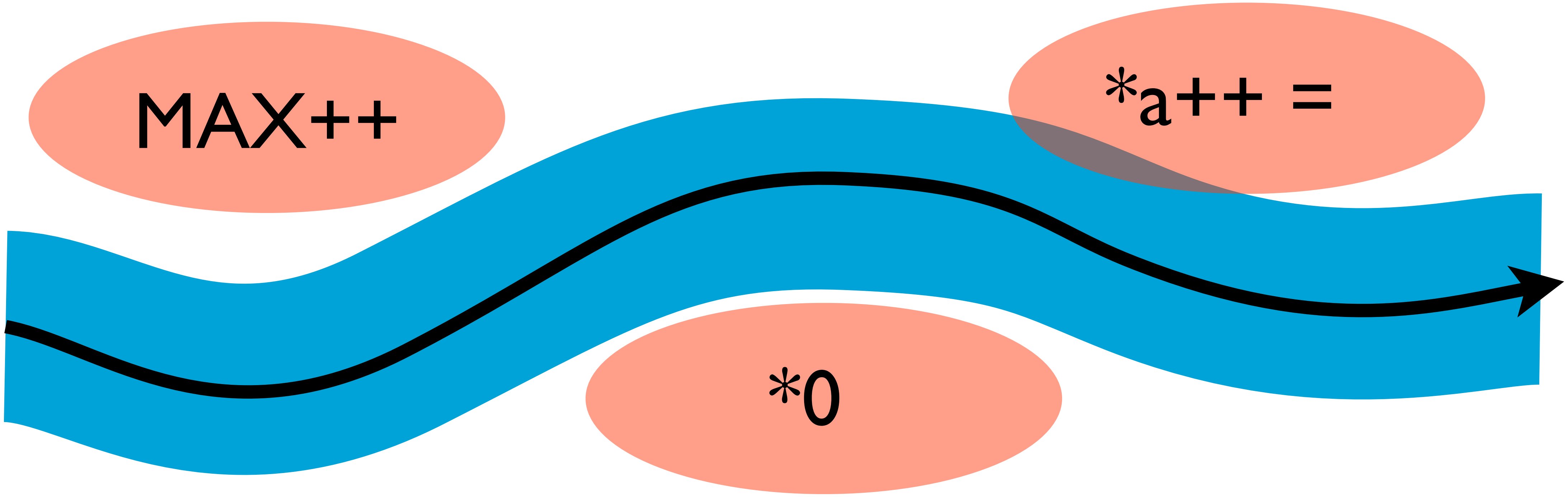
The static analysis game



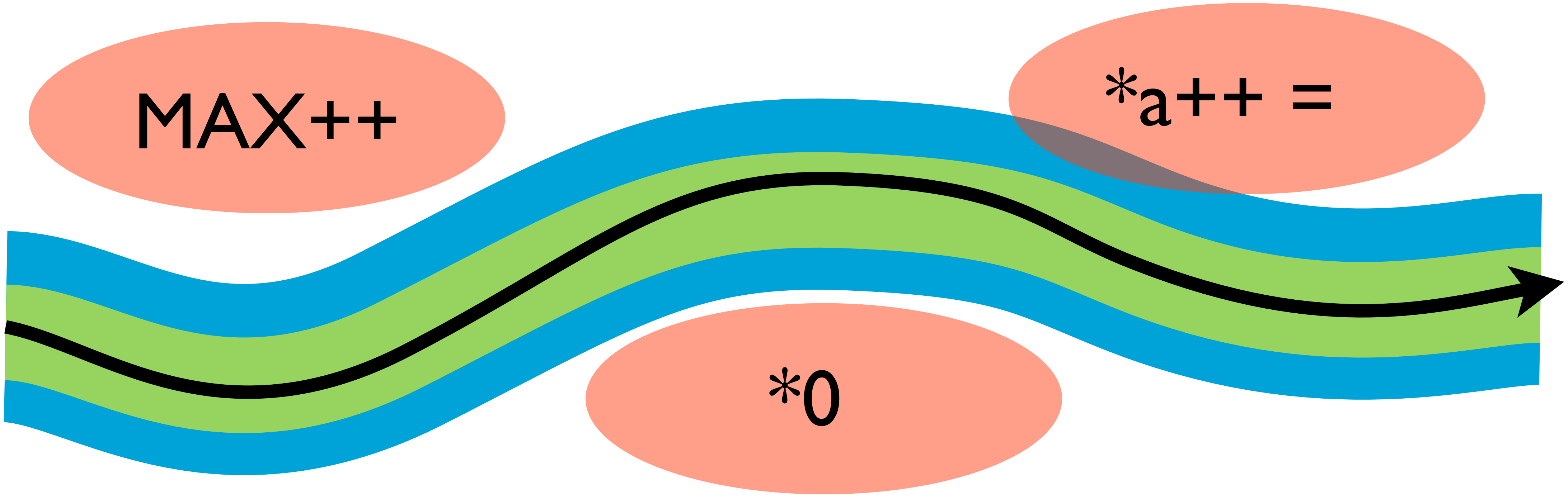
The static analysis game



The static analysis game



The static analysis game



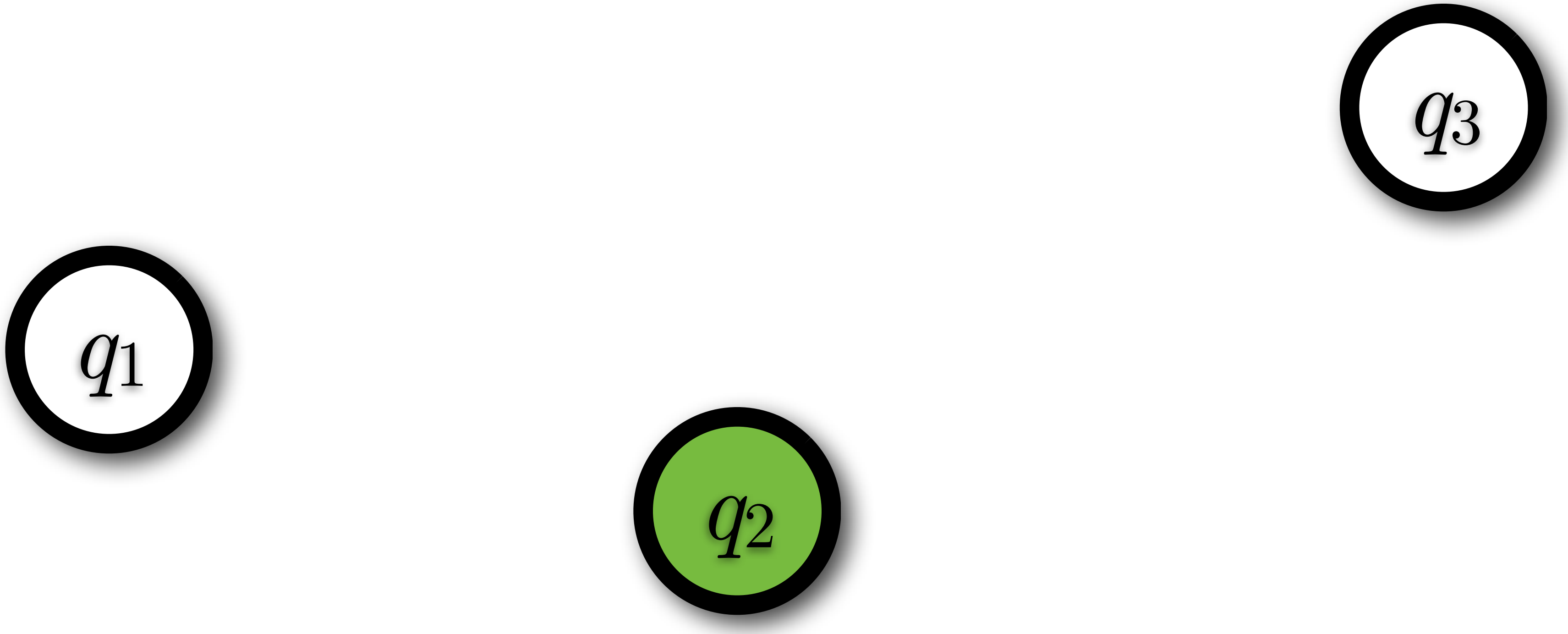
Another way?

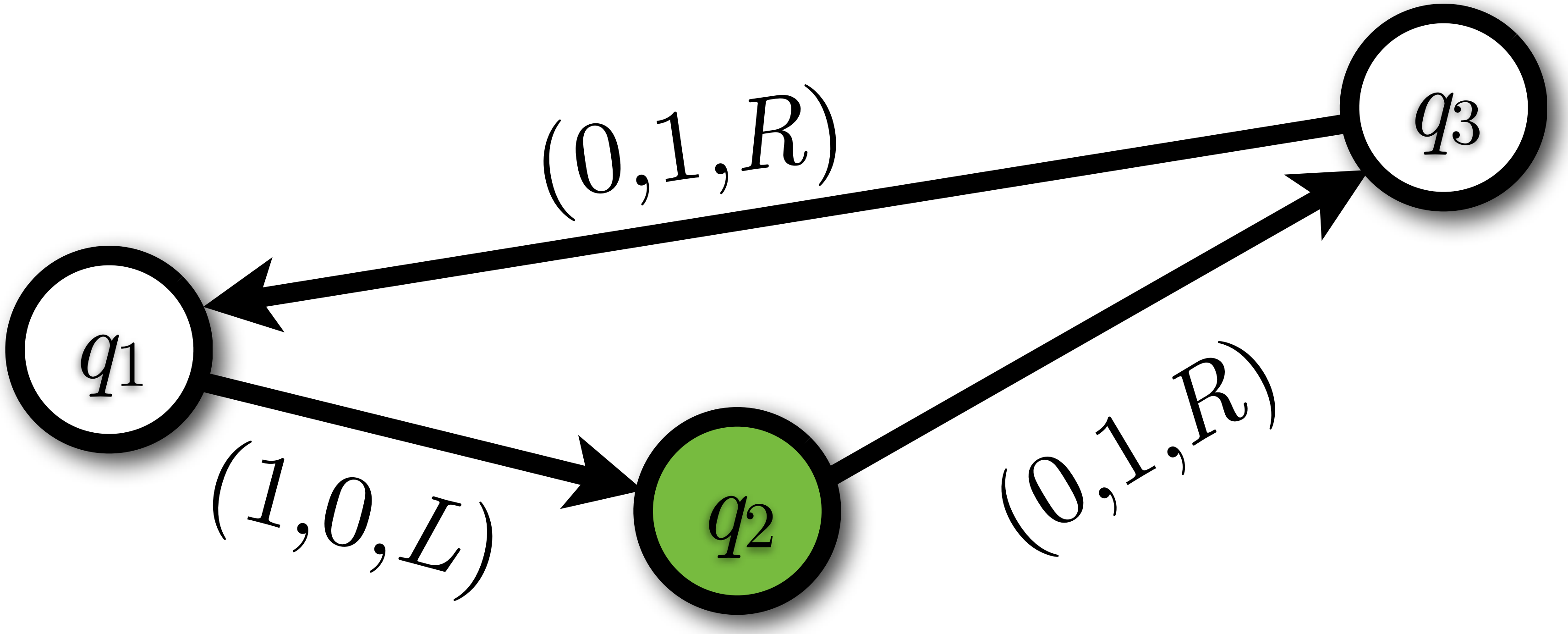
Don’t use Turing machines.

Static analysis

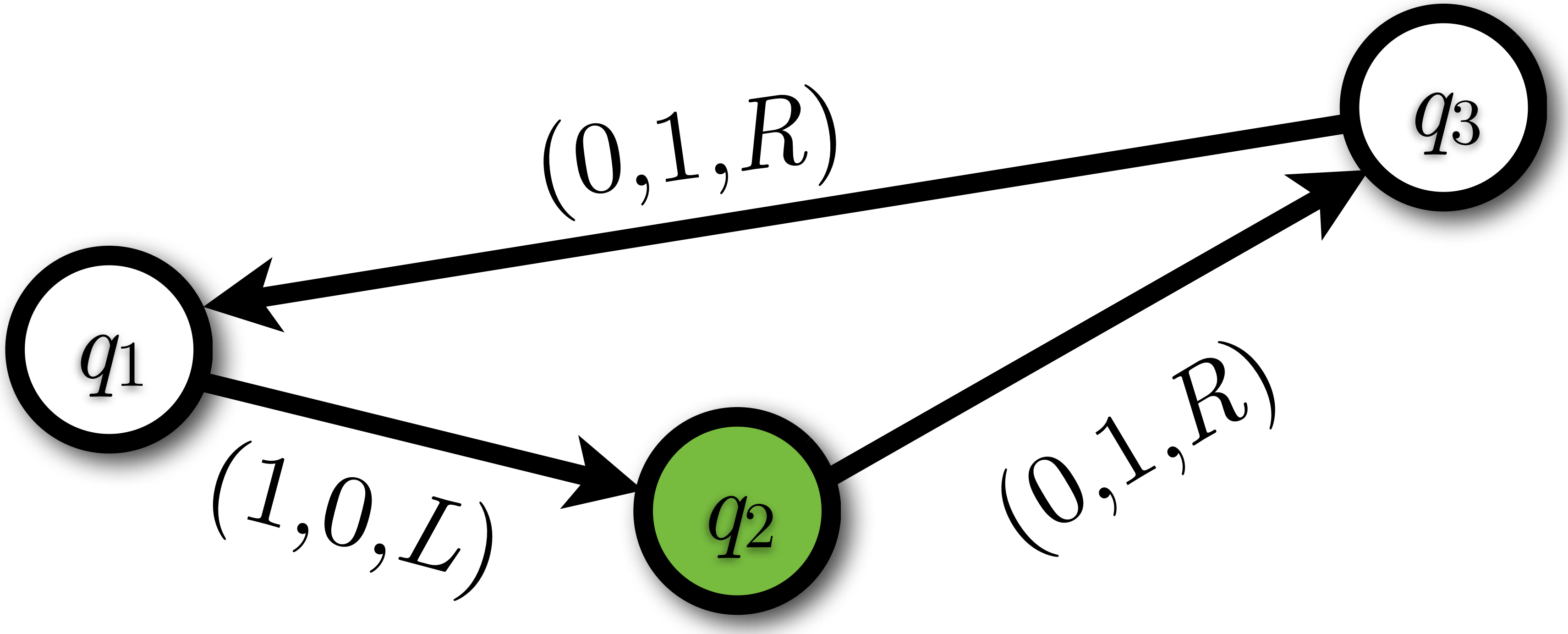
Sub-Turing languages

How do you play the static analysis game?

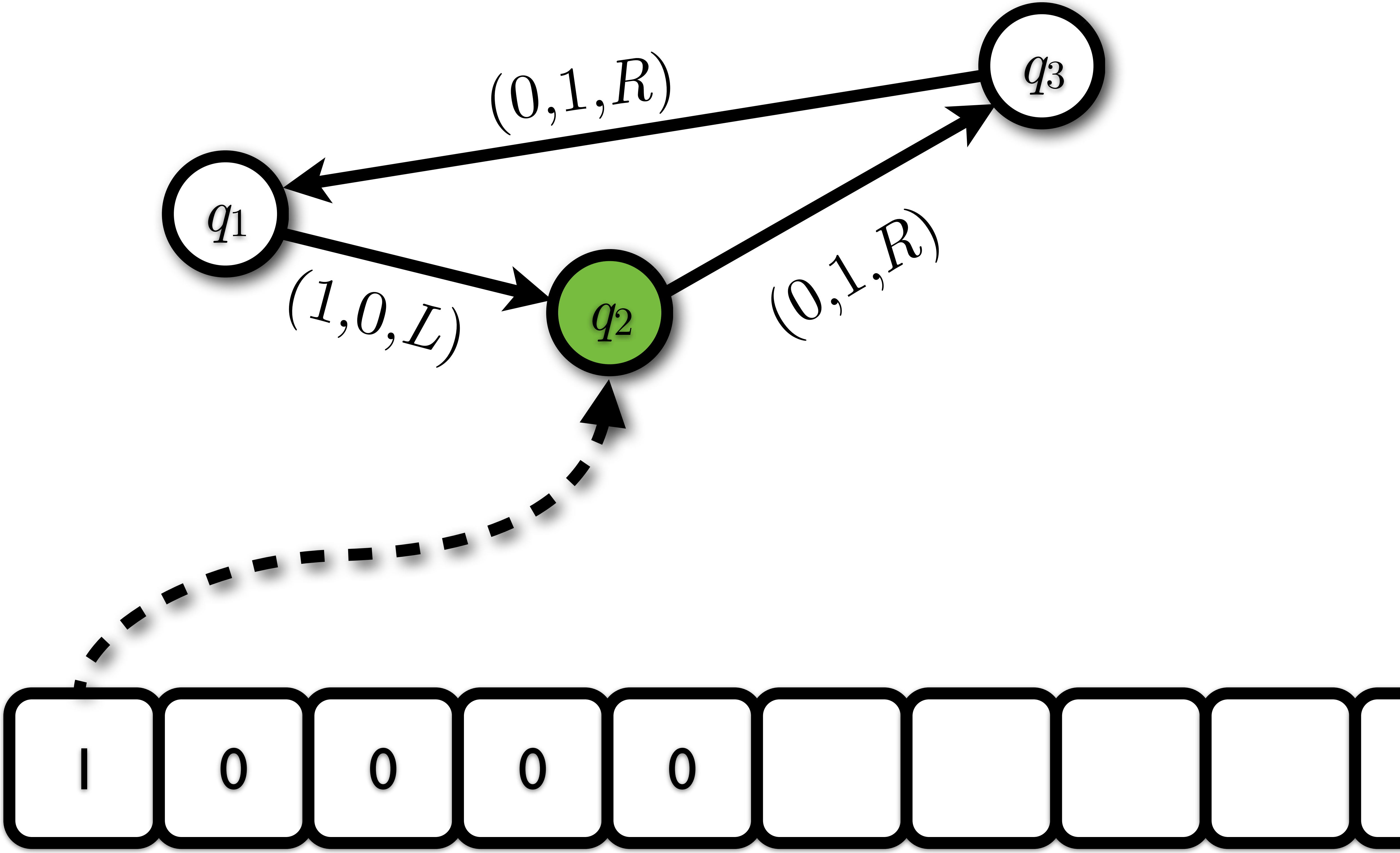

q_1
q_2
q_3



$(0,1,R)$
q_3
q_1
$(1,0,L)$
q_2
$(0,1,R)$

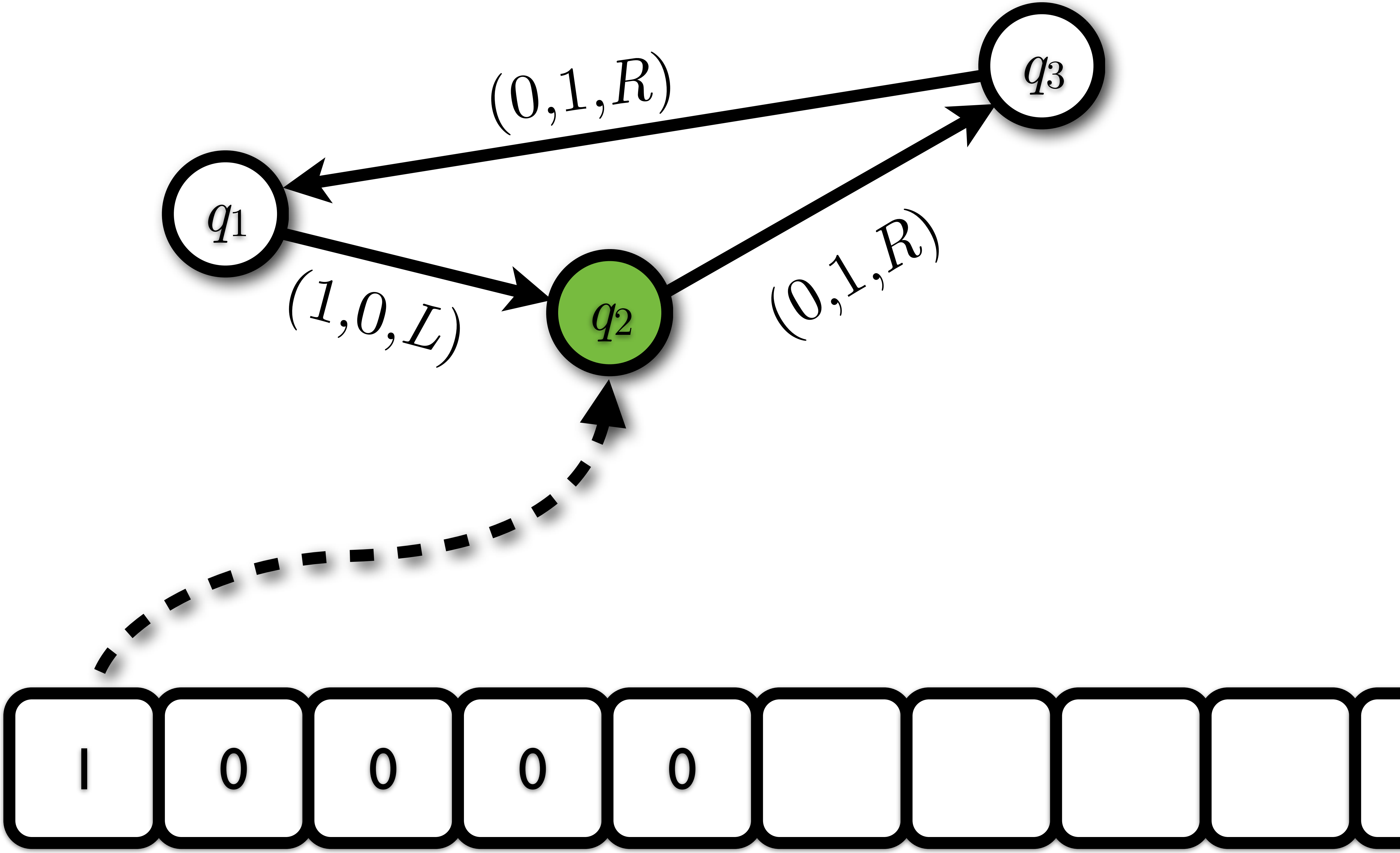


1	0	0	0	0					

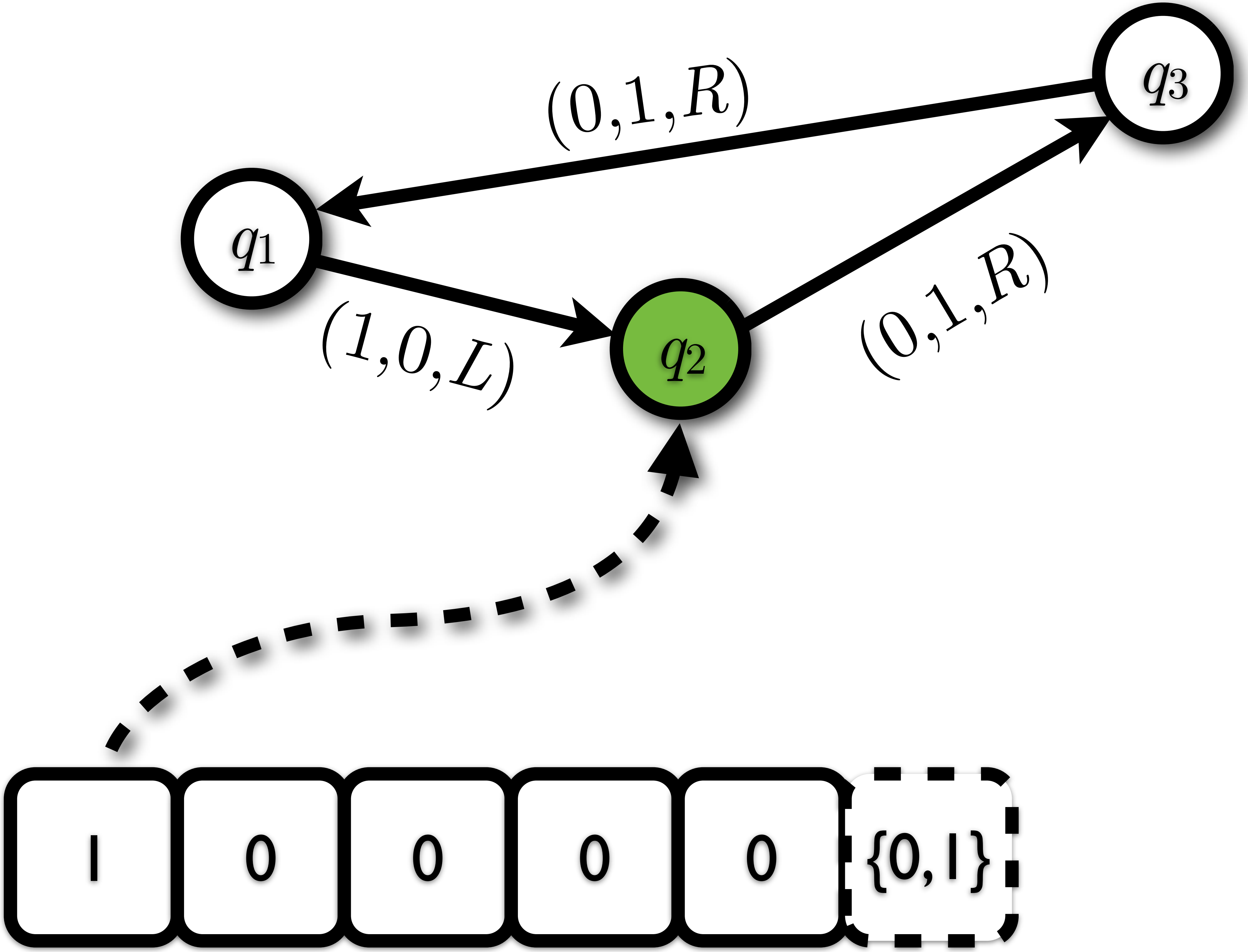

$(0,1,R)$
q_3
q_1
$(1,0,L)$
q_2
$(0,1,R)$
1
0
0
0
0

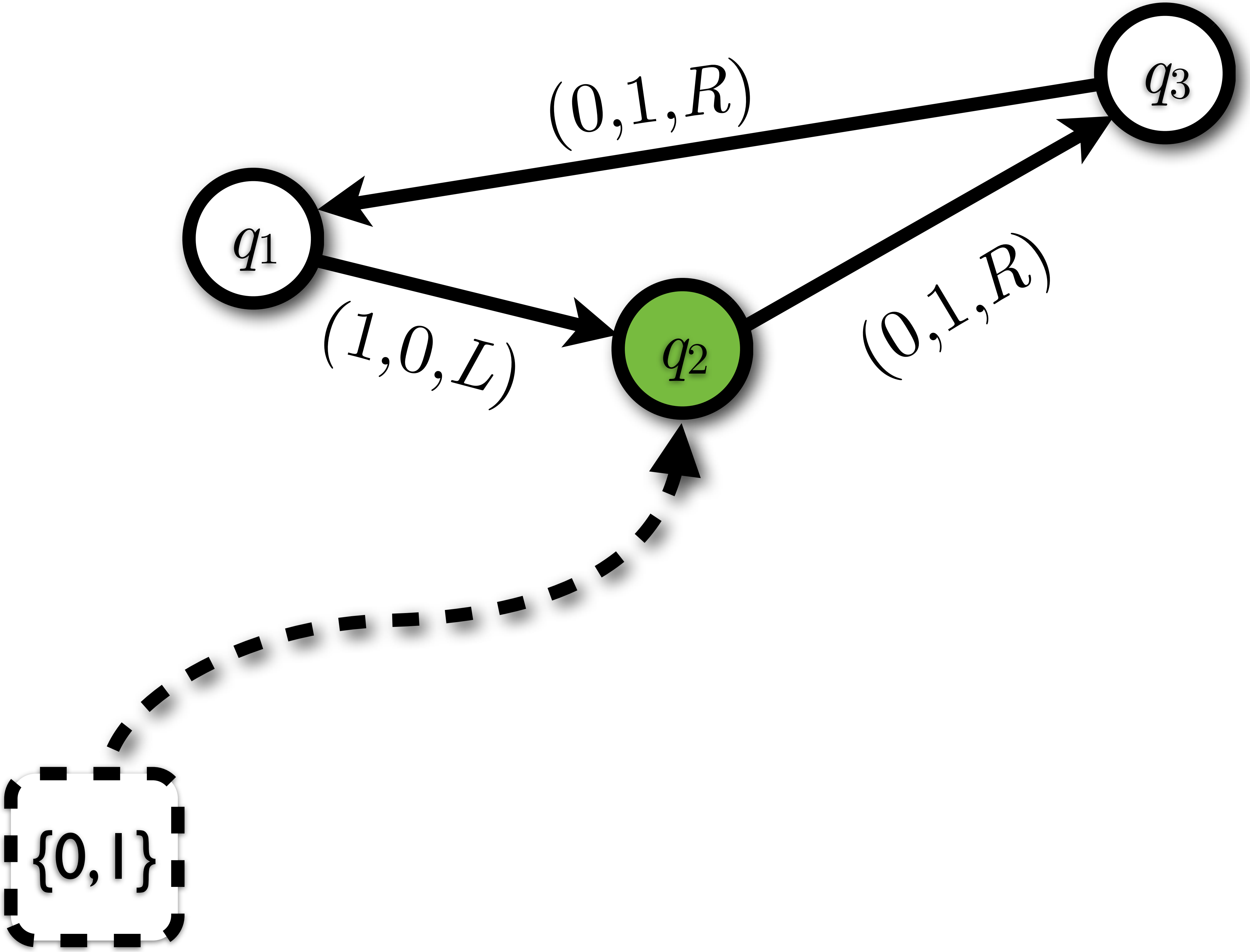
How to approximate?

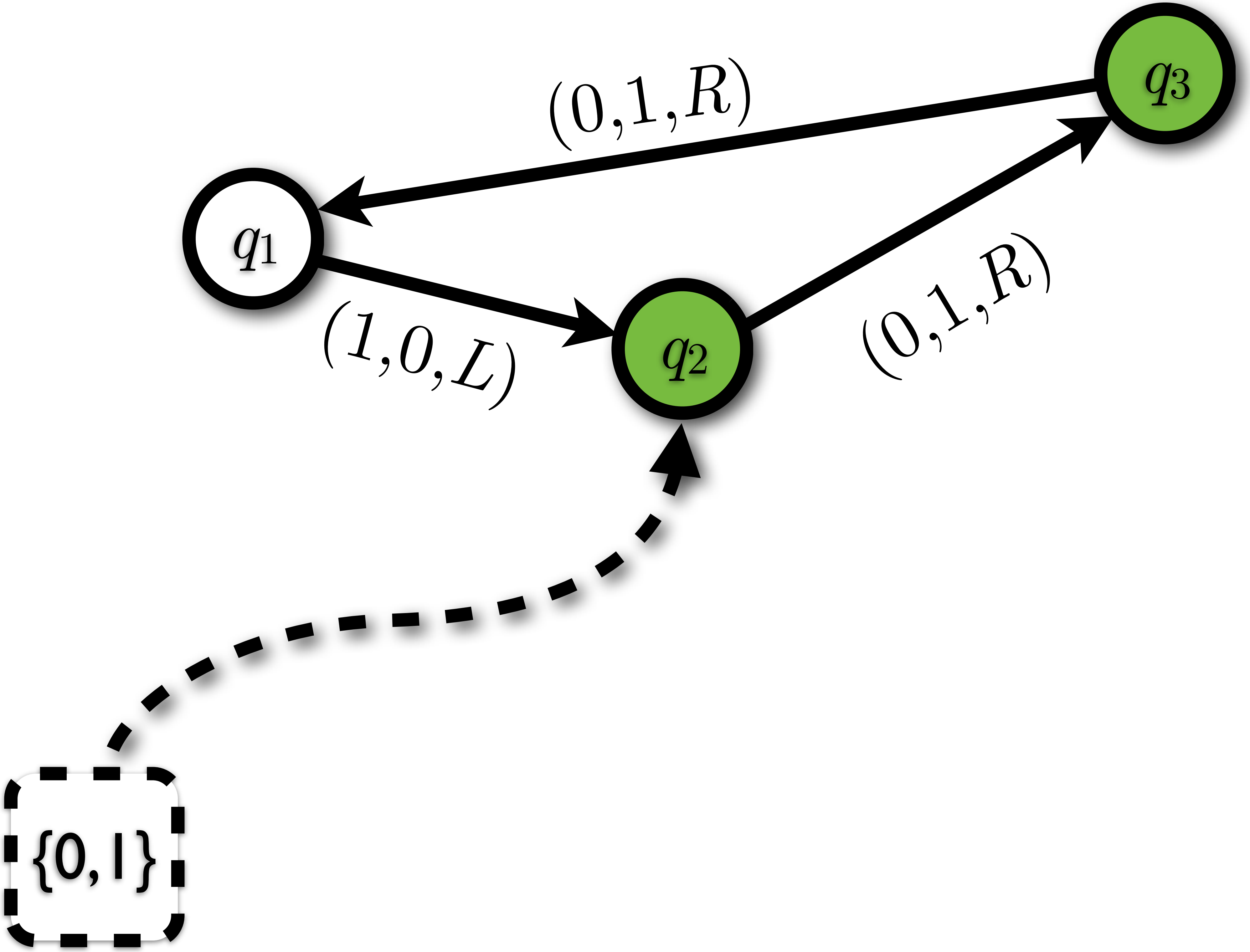
Make it finite!



$(0,1,R)$
q_3
q_1
$(1,0,L)$
q_2
$(0,1,R)$
1
0
0
0
0


(0,1,R)
q_3
q_1
(1,0,L)
q_2
(0,1,R)
1
0
0
0
0
{0,1}


$(0,1,R)$
q_3
q_1
$(1,0,L)$
q_2
$(0,1,R)$
{0,1}



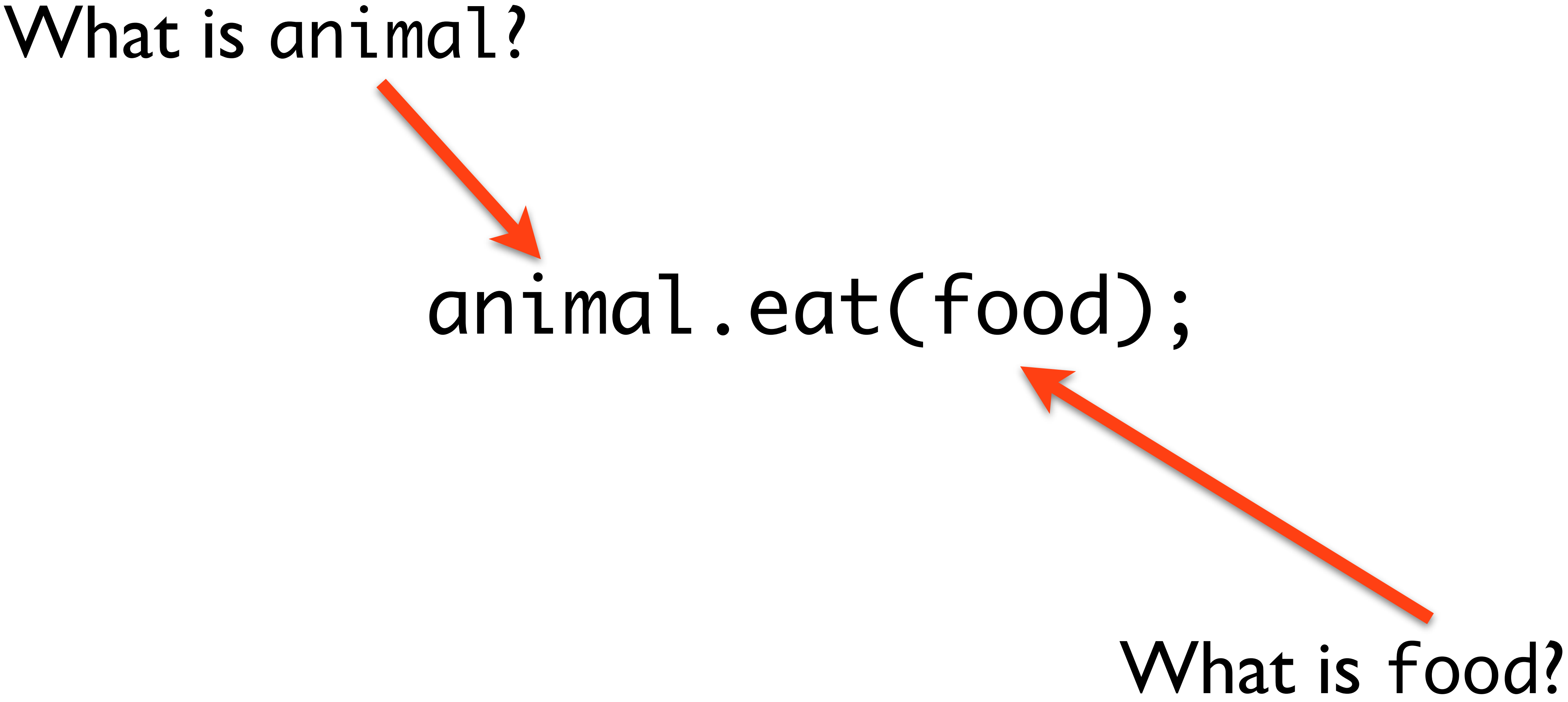
q_3
$(0,1,R)$
q_1
$(1,0,L)$
q_2
$(0,1,R)$
{0,1}

Why is static analysis hard?

What happens here?

```
animal.eat(food);
```

What happens here?



What is `animal`?

```
animal.eat(food);
```

What is `food`?

What happens here?

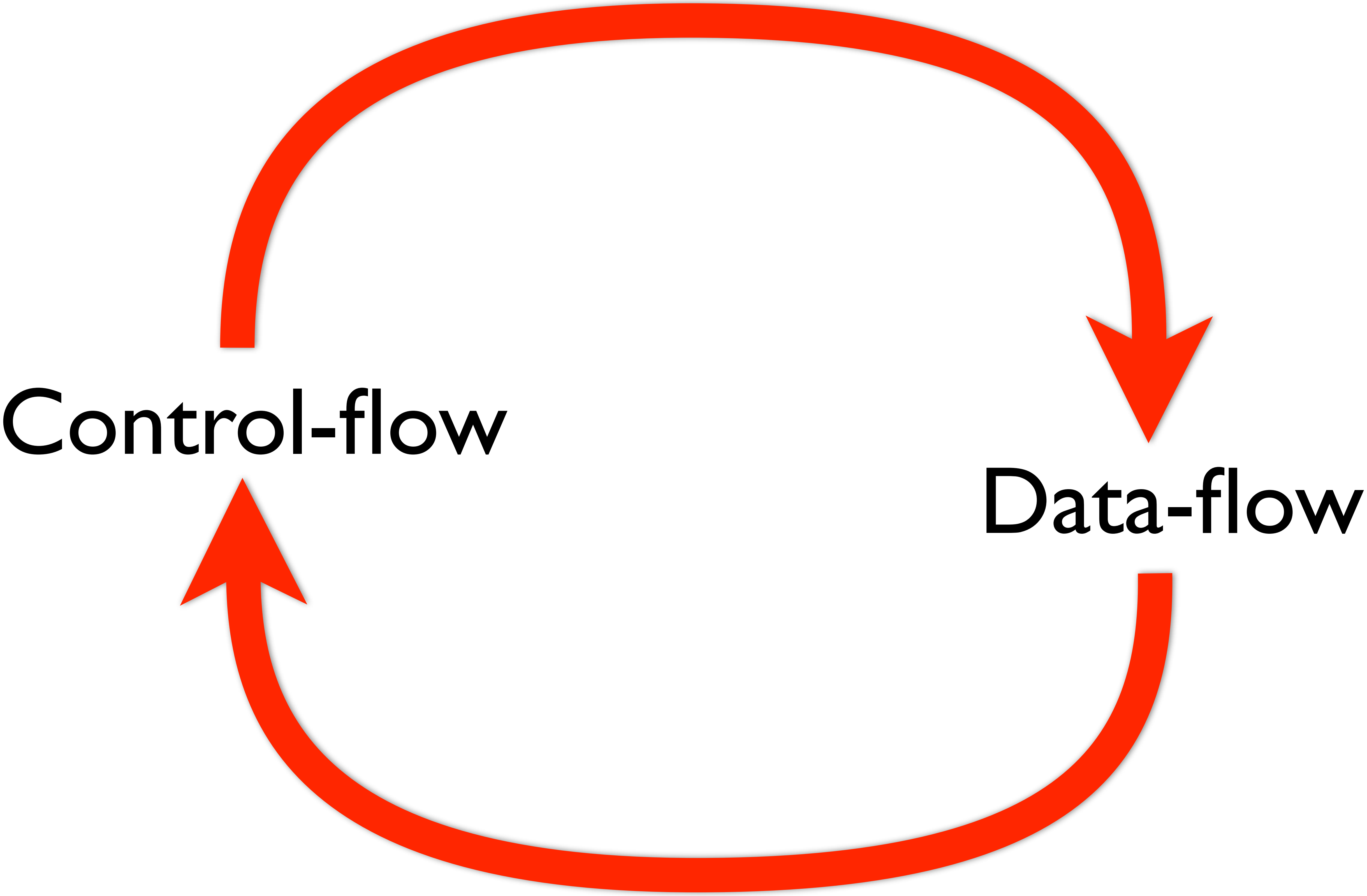
```
void process (Animal animal) {
   food = world.gather() ;
   animal.eat(food);
}
```

What happens here?

Who calls process?

```
void process (Animal animal) {
    food = world.gather() ;
    animal.eat(food);
}
```

What is world?



Control-flow

Data-flow

Why so entangled?

Value = Object

Value = Object

= Class + Record

$$\begin{aligned}\text{Value} &= \text{Object}\\ &= \text{Class} + \text{Record}\\ &\subseteq \text{Code} + \text{Data}\end{aligned}$$

Old idea:
Untie code & data.

(In ten minutes)

What language exemplifies code + data?

λ-calculus.

λ-calculus (Church, 1928)

λ-calculus (Church, 1928)

- Minimalist, universal language

Alonzo Church

λ-calculus (Church, 1928)

- Minimalist, universal language
- Three expression types:

v [variable]

Alonzo Church

λ-calculus (Church, 1928)

- Minimalist, universal language
- Three expression types:

v	[variable]
$e_1(e_2)$	[function application]

Alonzo Church

λ-calculus (Church, 1928)

- Minimalist, universal language
- Three expression types:

v	[variable]
$e_1(e_2)$	[function application]
$\lambda v. e$	[anonymous function]

Alonzo Church

$$(\lambda x.x^2)(3) = 9$$

Lisp and Scheme

- v ≡ `v`
- $f(e)$ ≡ `(f e)`
- $\lambda v.e$ ≡ `(lambda (v) e)`

Python

- v ≡ `v`
- $f(e)$ ≡ `f(e)`
- $\lambda v.e$ ≡ `lambda v: e`

Ruby

- v ≡ `v`
- $f(e)$ ≡ `f(e)`
- $\lambda v. e$ ≡ `lambda { |v| return e }`

JavaScript

- v ≡ `v`
- $f(e)$ ≡ `f(e)`
- $\lambda v.e$ ≡ `function (v) { return e ; }`

Java

- v ≡ `v`
- $f(e)$ ≡ `f.call(e)`
- $\lambda v.e$ ≡ `new Value () { public Value call (Value v) { return e ; } } ;`

λ-fortified

- Lisp
- SML
- Haskell
- Scala
- Java
- C#
- C++
- Python
- Ruby
- Smalltalk
- JavaScript
- PHP(!)

Value = Closure

Value = Closure

= Lambda + Env

$$\begin{aligned} \text{Value} &= \text{Closure} \\ &= \text{Lambda} + \text{Env} \\ &\subseteq \text{Code} + \text{Data} \end{aligned}$$

Assertion:
If we can do λ's,
we can do objects.

How to bound control?

Control-flow question

Given a call site `f(x)`, what could `f` be?

$f(x)$

```
let f = λz.z
     in f(x)
```

$\lambda f.f(x)$

Classical approach

The approximation

- Value = Code x Data
- Closure = Lambda x Env
- Object = Class x Record

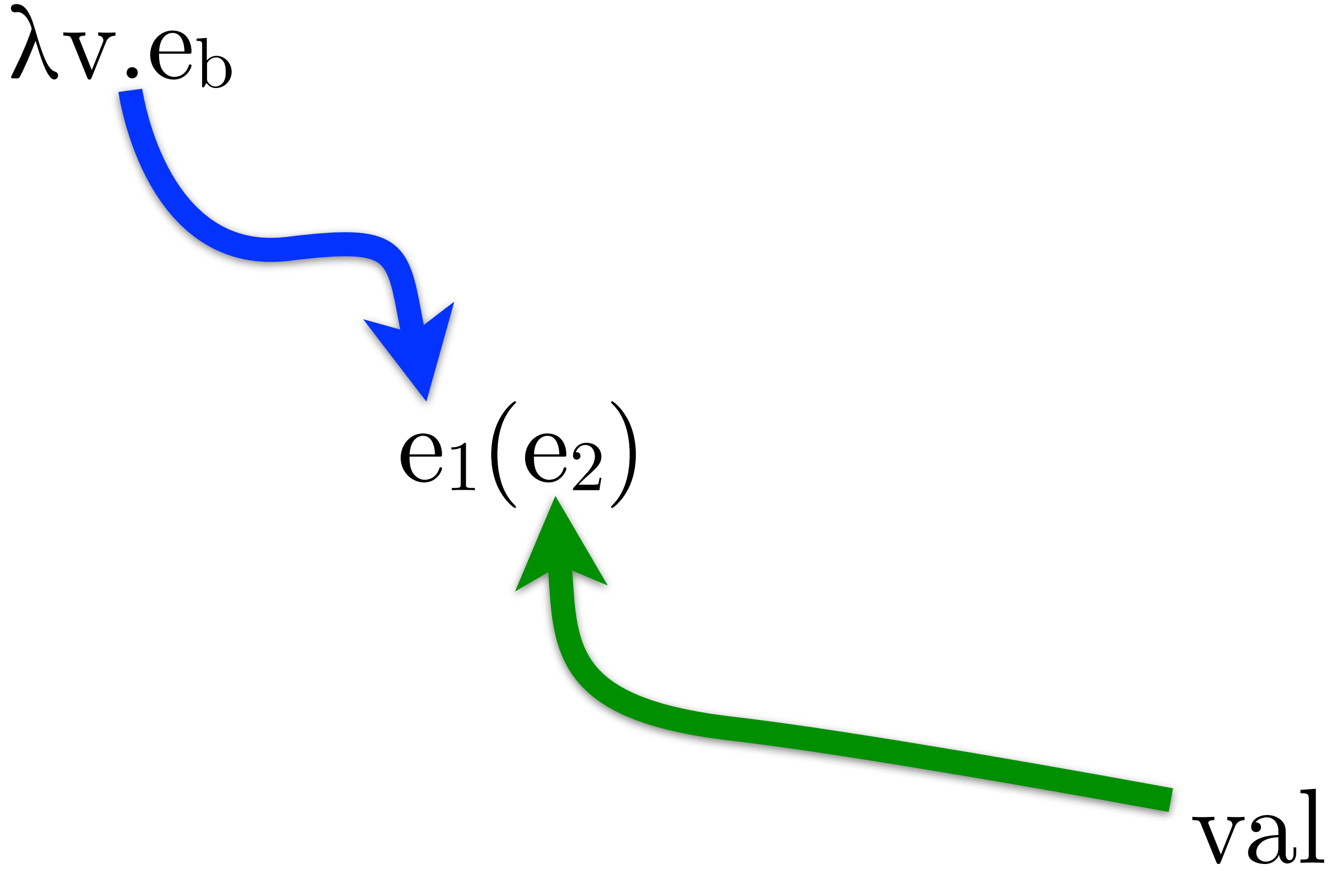
The approximation

- Value = Code
- Closure = Lambda
- Object = Class

How do λ’s flow?

$$e_1(e_2)$$

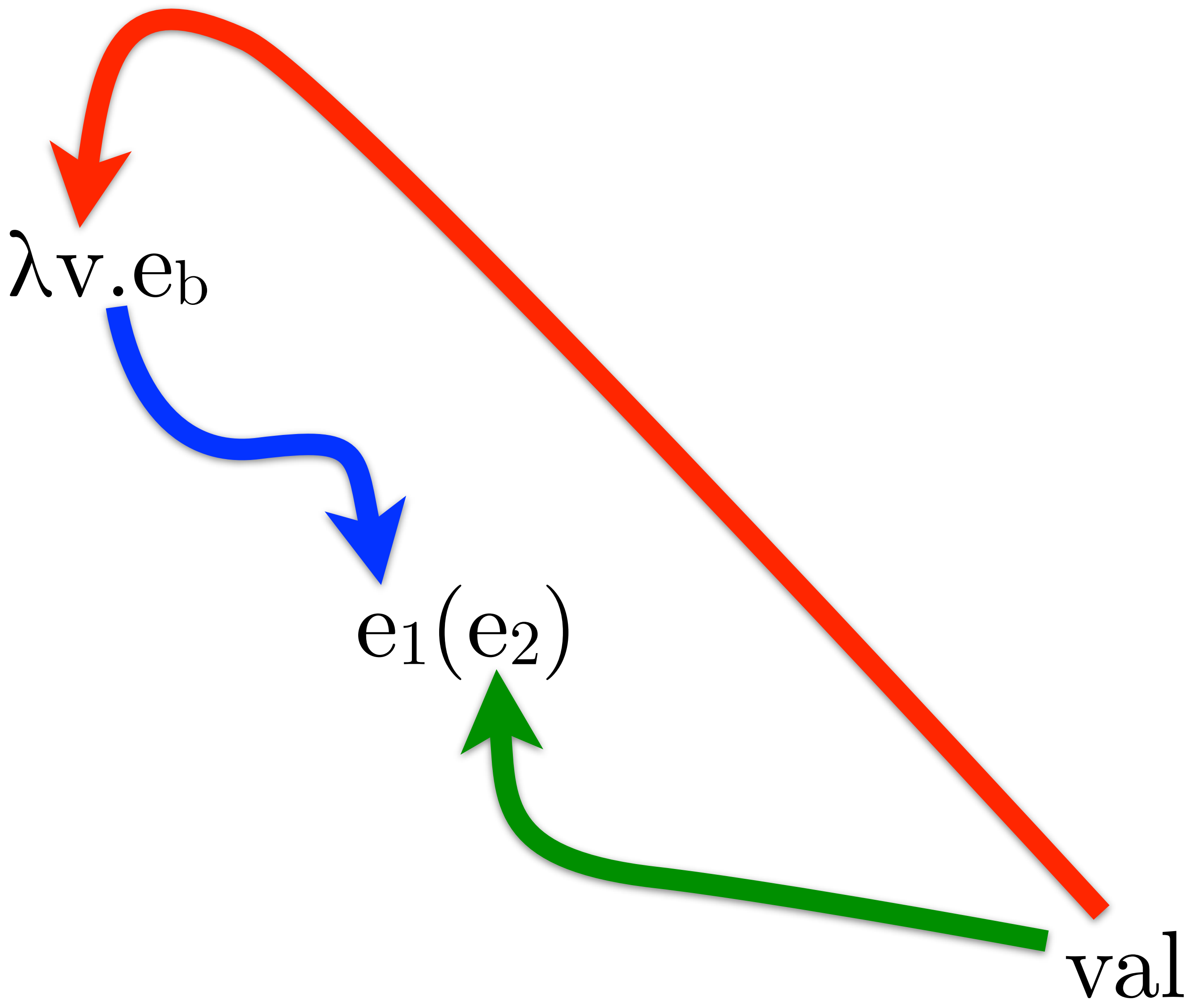
$$\begin{tikzcd} \lambda v.e_b \arrow[dr, rightsquigarrow] & \\ & e_1(e_2) \end{tikzcd}$$



$\lambda v.e_b$

$e_1(e_2)$

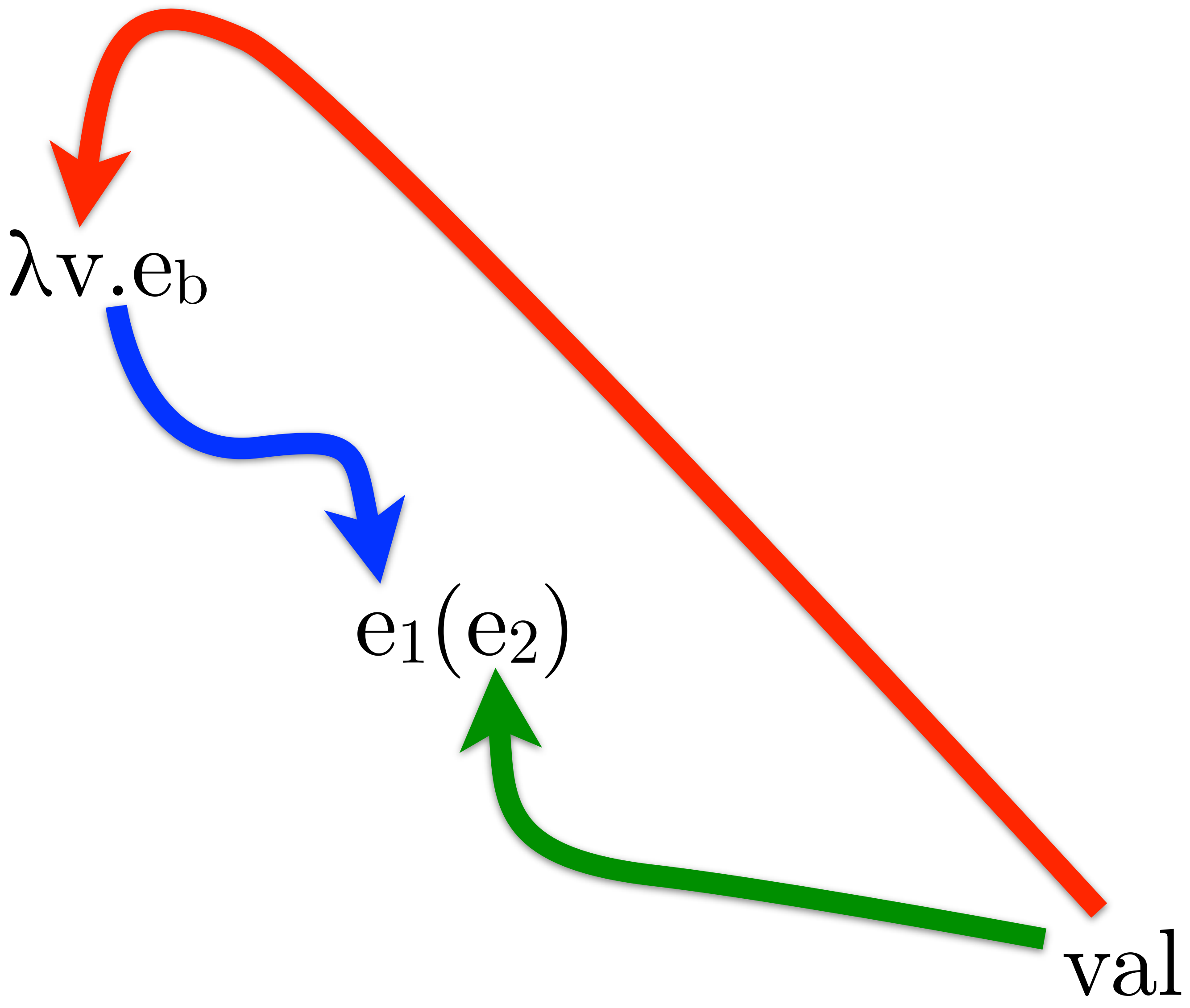
val



$\lambda v.e_b$

$e_1(e_2)$

val

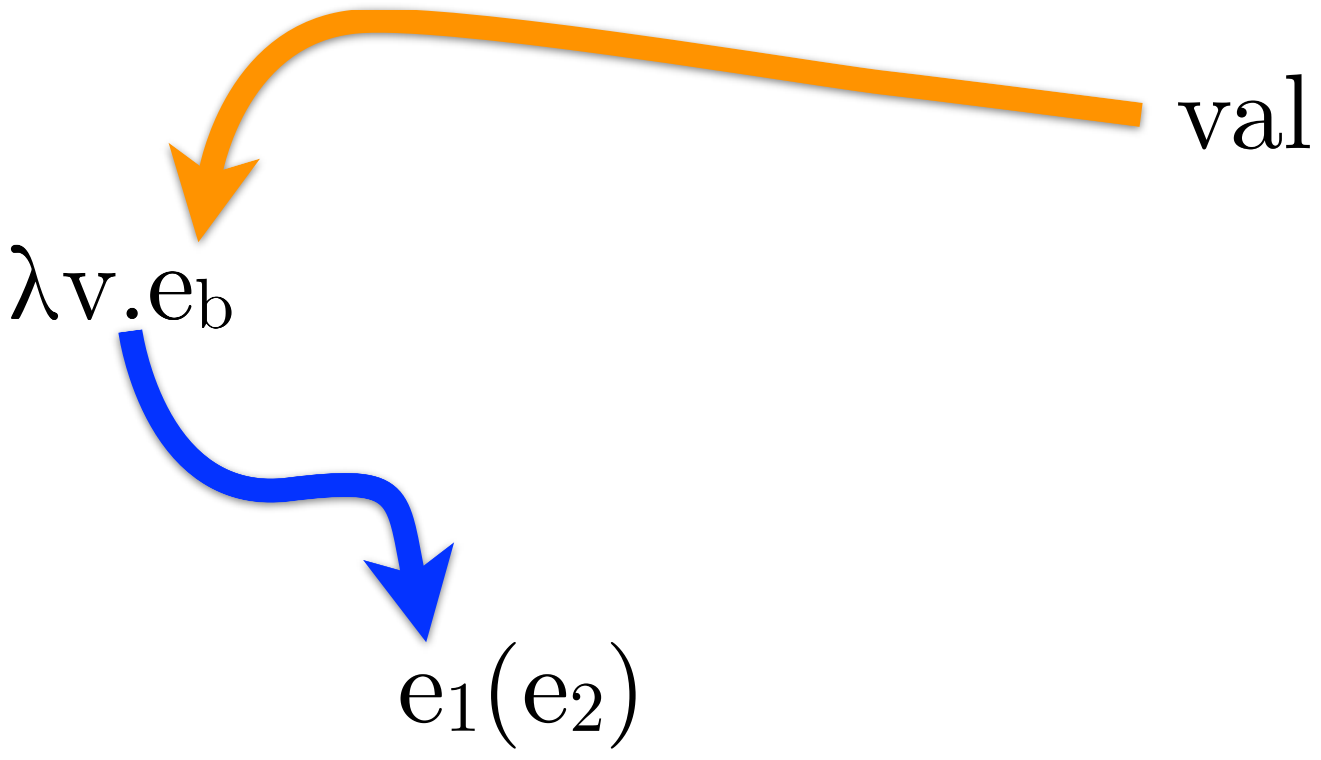


$\lambda v.e_b$

$e_1(e_2)$

val

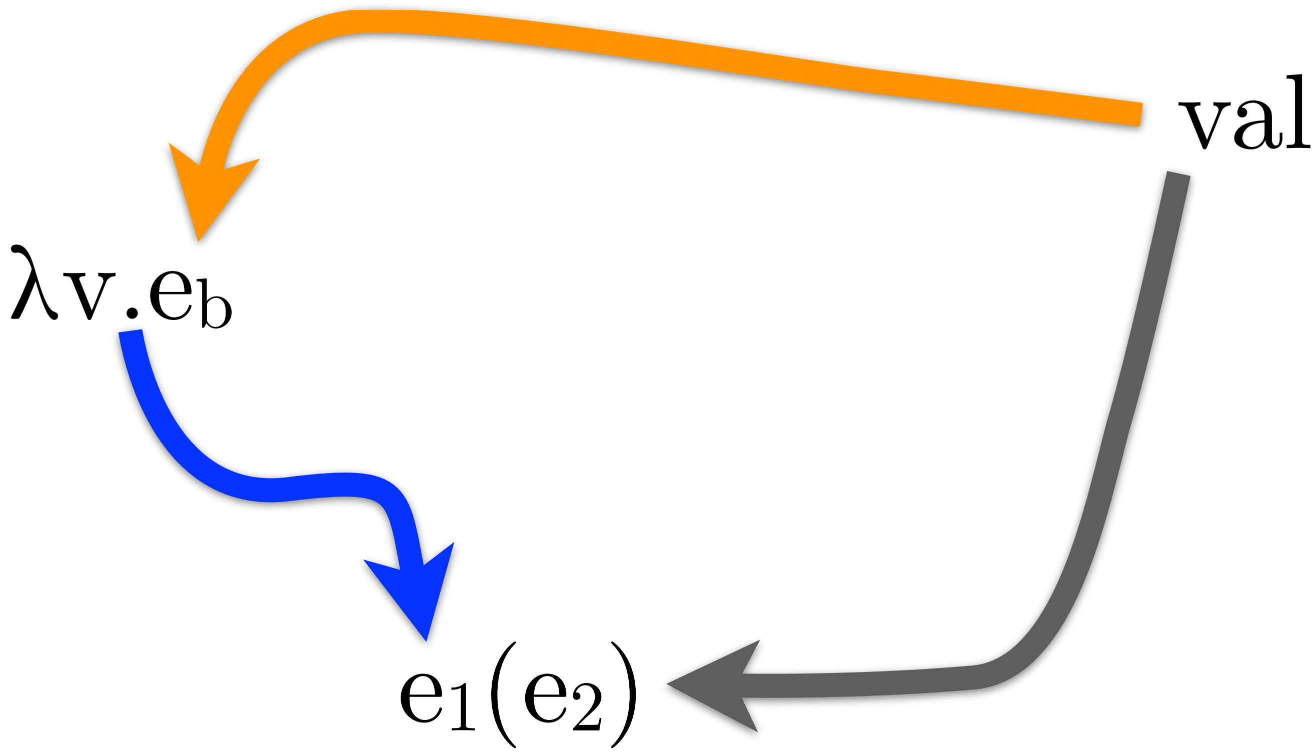
$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1] \quad \text{and} \quad val \in \text{FlowsTo}[e_2]}{val \in \text{FlowsTo}[v]}$$



val

$\lambda v.e_b$

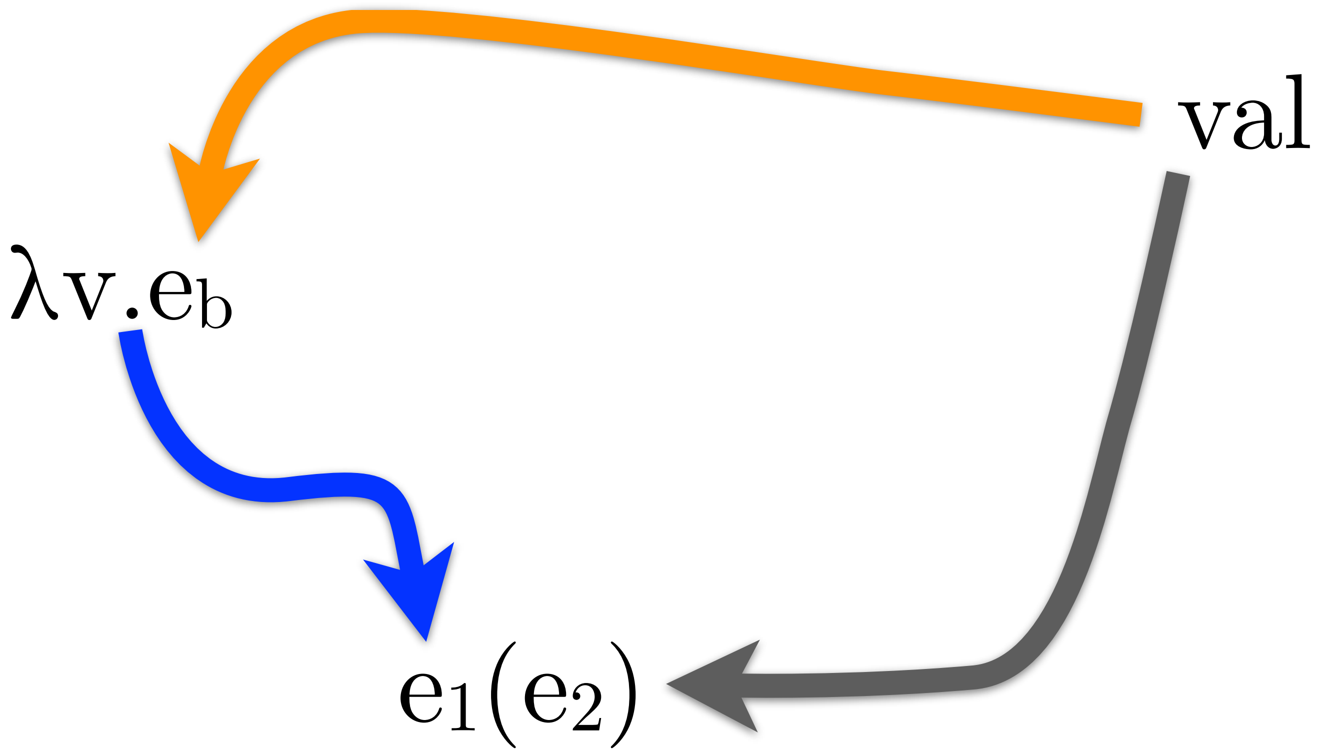
$e_1(e_2)$



val

$\lambda v.e_b$

$e_1(e_2)$



$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1] \quad \text{and} \quad \text{val} \in \text{FlowsTo}[e_b]}{\text{val} \in \text{FlowsTo}[e_1(e_2)]}$$

$$\frac{\lambda \mathrm{v}.\mathrm{e}_{\mathrm{b}} \in \mathrm{FlowsTo}[\mathrm{e}_1] \quad \text{and} \quad \mathrm{val} \in \mathrm{FlowsTo}[\mathrm{e}_{\mathrm{b}}]}{\mathrm{val} \in \underline{\mathrm{FlowsTo}}[\mathrm{e}_1(\mathrm{e}_2)]}$$

$$\lambda v.e_b \in \mathrm{FlowsTo}[\lambda v.e_b]$$

$$\frac{\lambda v.e_b \in \mathrm{FlowsTo}[e_1] \quad \text{and} \quad \mathrm{val} \in \mathrm{FlowsTo}[e_b]}{\mathrm{val} \in \mathrm{FlowsTo}[e_1(e_2)]}$$

$$\frac{\lambda v.e_b \in \mathrm{FlowsTo}[e_1] \quad \text{and} \quad \mathrm{val} \in \mathrm{FlowsTo}[e_2]}{\mathrm{val} \in \mathrm{FlowsTo}[v]}$$

0CFA (Shivers, 1988)

$$\{\lambda v.e_b\} \subseteq \mathrm{FlowsTo}[\lambda v.e_b]$$

$$\frac{\lambda v.e_b \in \mathrm{FlowsTo}[e_1]}{\mathrm{FlowsTo}[e_b] \subseteq \mathrm{FlowsTo}[e_1(e_2)]}$$

$$\frac{\lambda v.e_b \in \mathrm{FlowsTo}[e_1]}{\mathrm{FlowsTo}[e_2] \subseteq \mathrm{FlowsTo}[v]}$$

0CFA (Shivers, 1988)

$$\{\lambda v.e_b\} \subseteq \text{FlowsTo}[\lambda v.e_b]$$

$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1]}{\text{FlowsTo}[e_b] \subseteq \text{FlowsTo}[e_1(e_2)]}$$

$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1]}{\text{FlowsTo}[e_2] \subseteq \text{FlowsTo}[v]}$$

0CFA (Shivers, 1988)

$$\{\lambda v.e_b\} \subseteq \text{FlowsTo}[\lambda v.e_b]$$

$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1]}{\text{FlowsTo}[e_b] \subseteq \text{FlowsTo}[e_1(e_2)]}$$

$$\frac{\lambda v.e_b \in \text{FlowsTo}[e_1]}{\text{FlowsTo}[e_2] \subseteq \text{FlowsTo}[v]}$$

+ Constraint Solver

= Control-flow analysis

But…

It’s slow.

It’s weak.

It’s imprecise.

Problem: Cross-flow

```
map f list
```

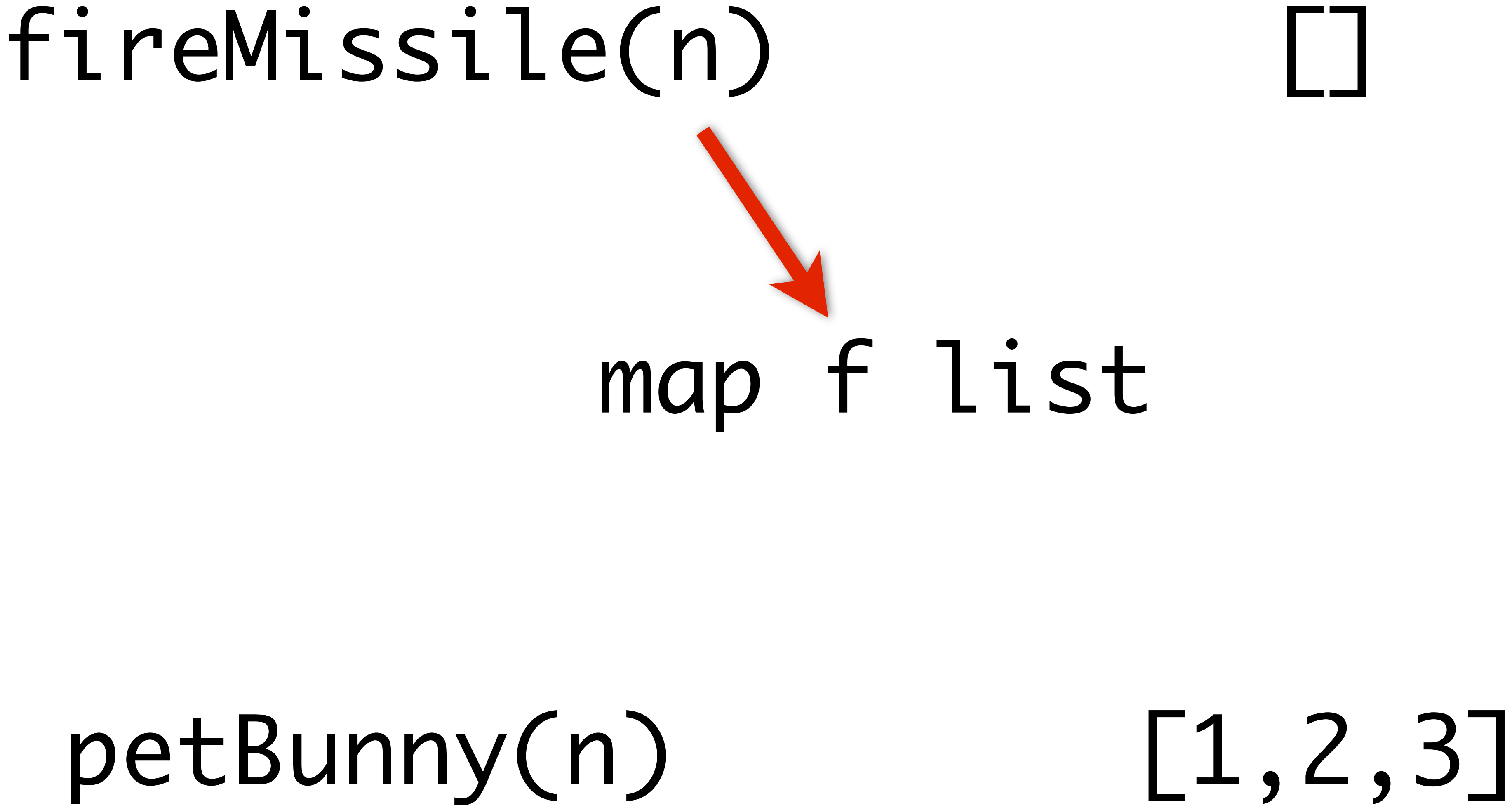
Problem: Cross-flow

```
fireMissile(n)            []

           map f list

  petBunny(n)          [1,2,3]
```

Problem: Cross-flow

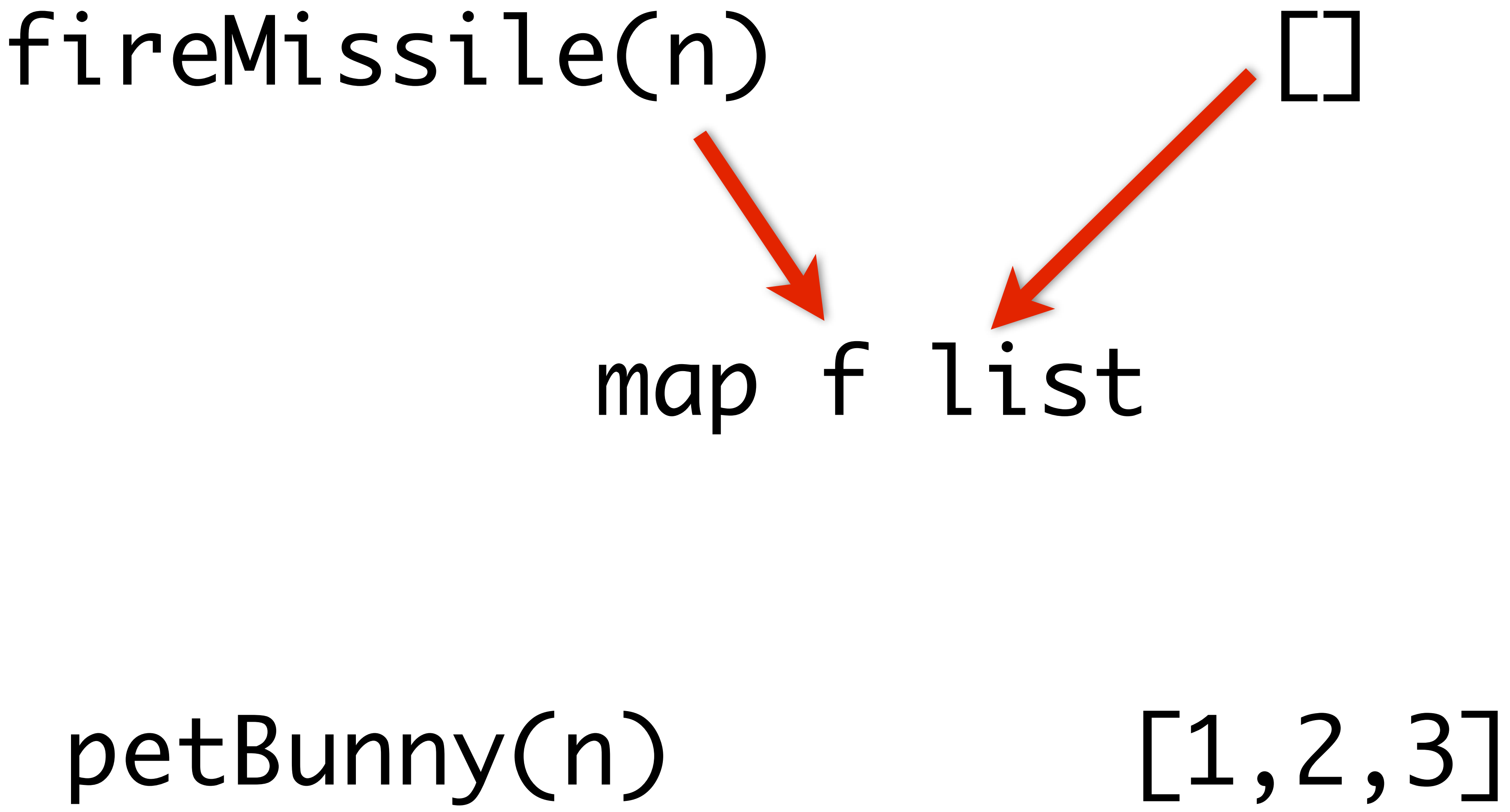


```
fireMissile(n)          []

        map f list

petBunny(n)             [1,2,3]
```

Problem: Cross-flow



fireMissile(n)

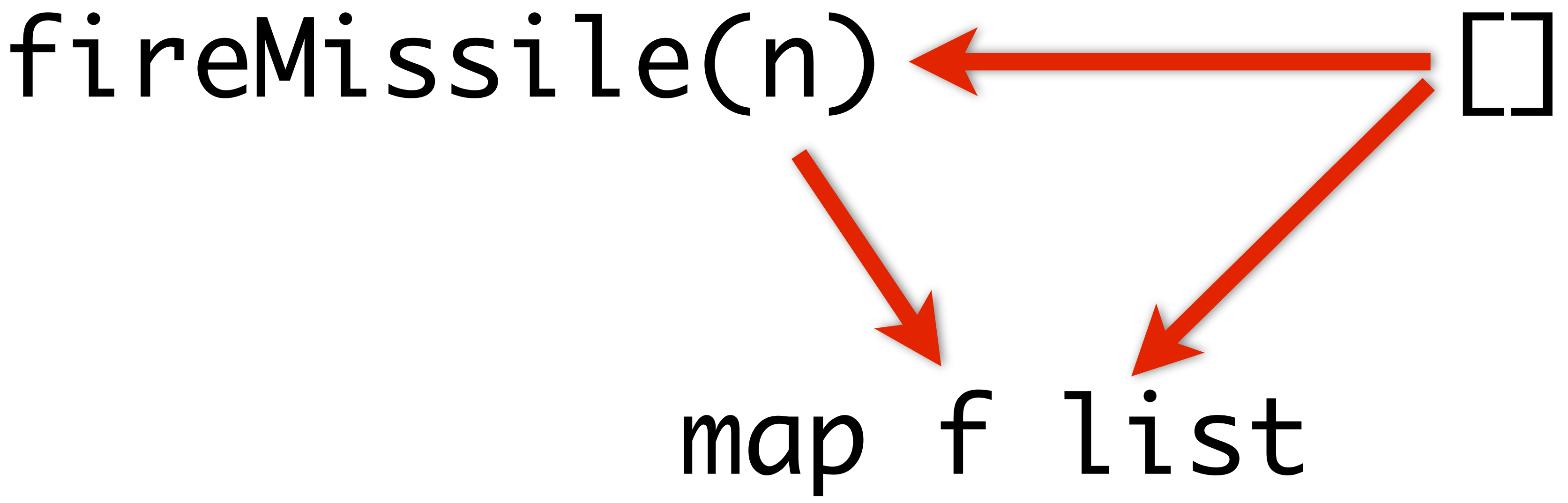
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



fireMissile(n)

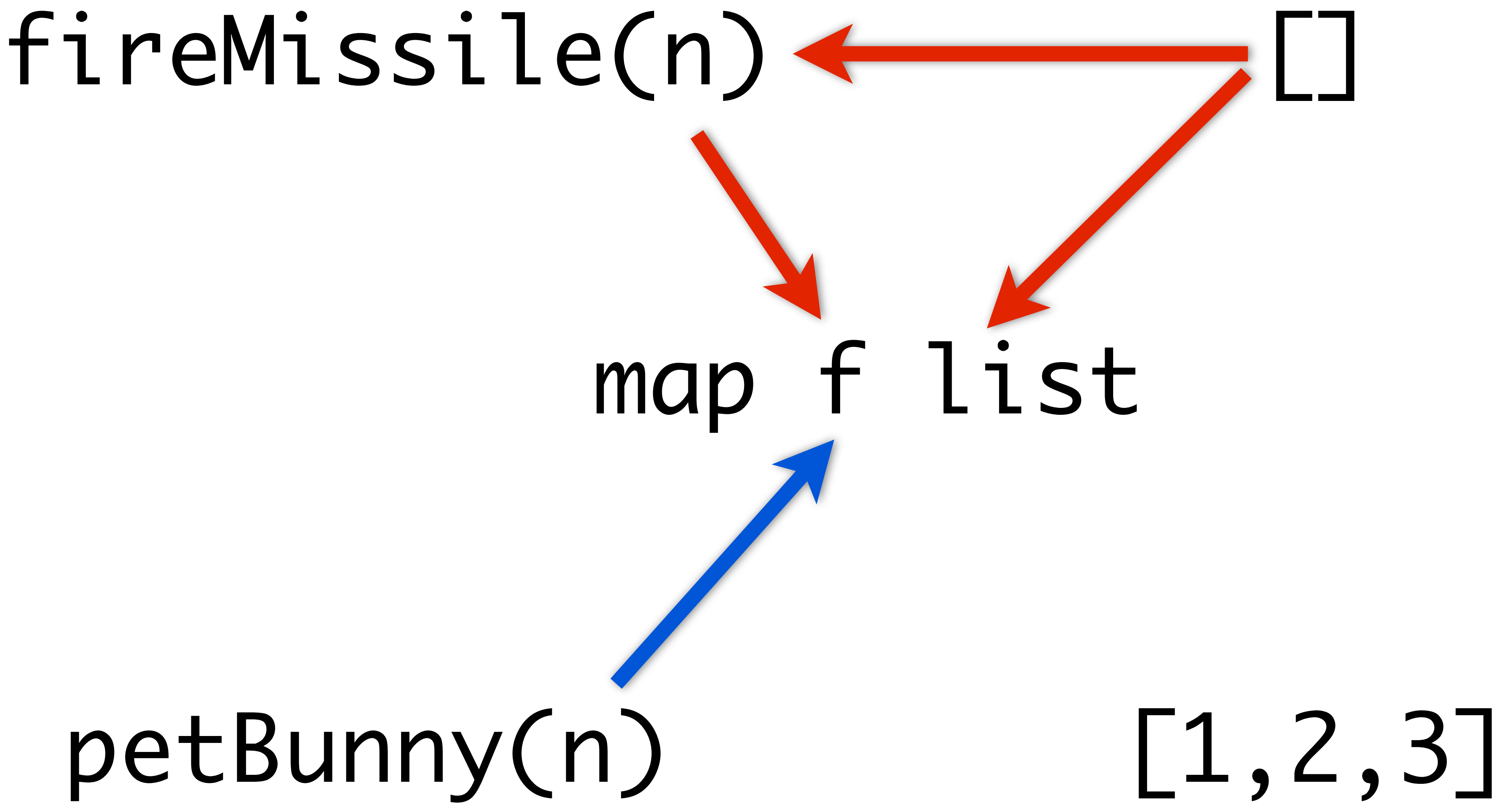
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



fireMissile(n)

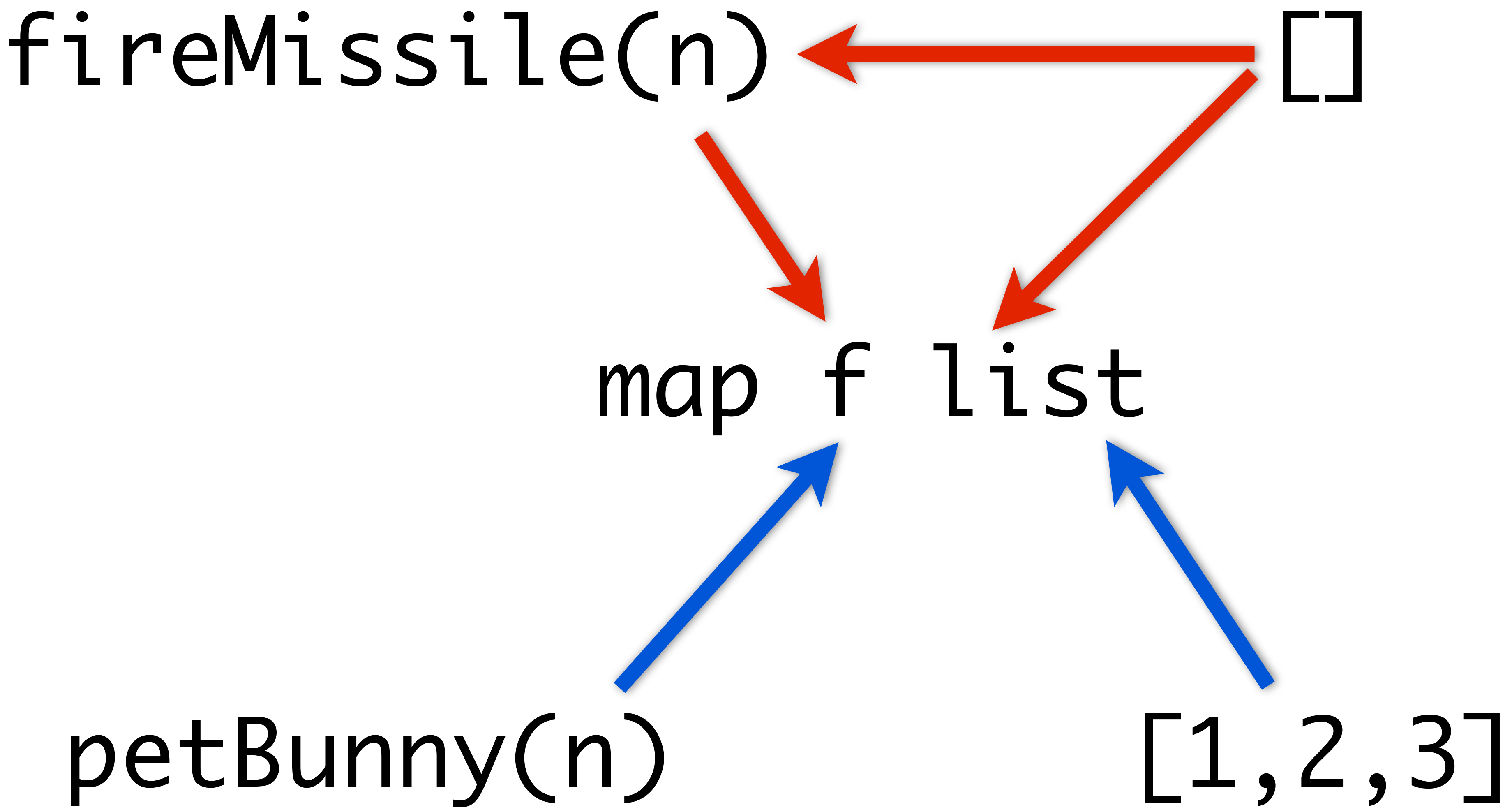
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



fireMissile(n)

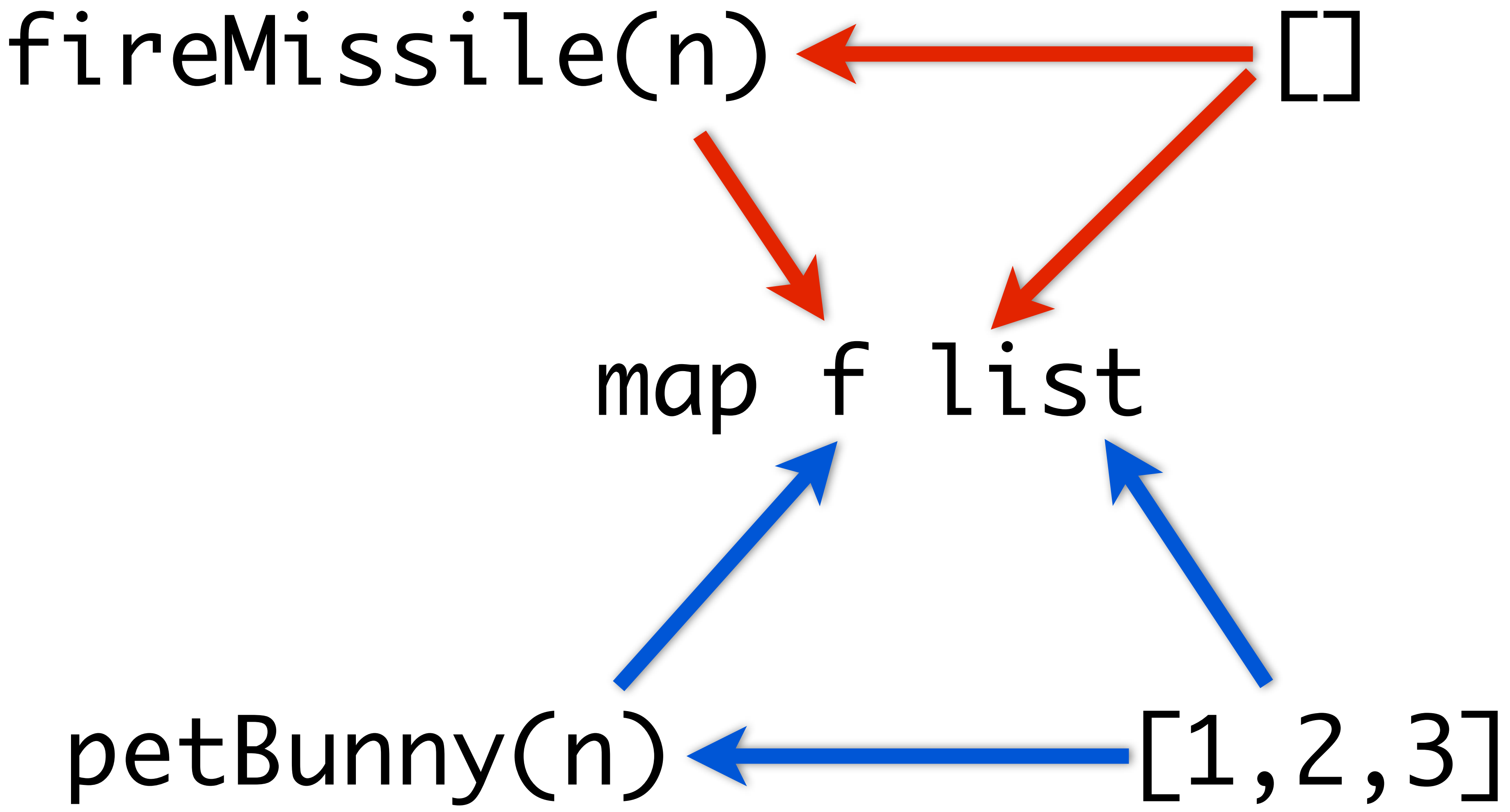
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



fireMissile(n)

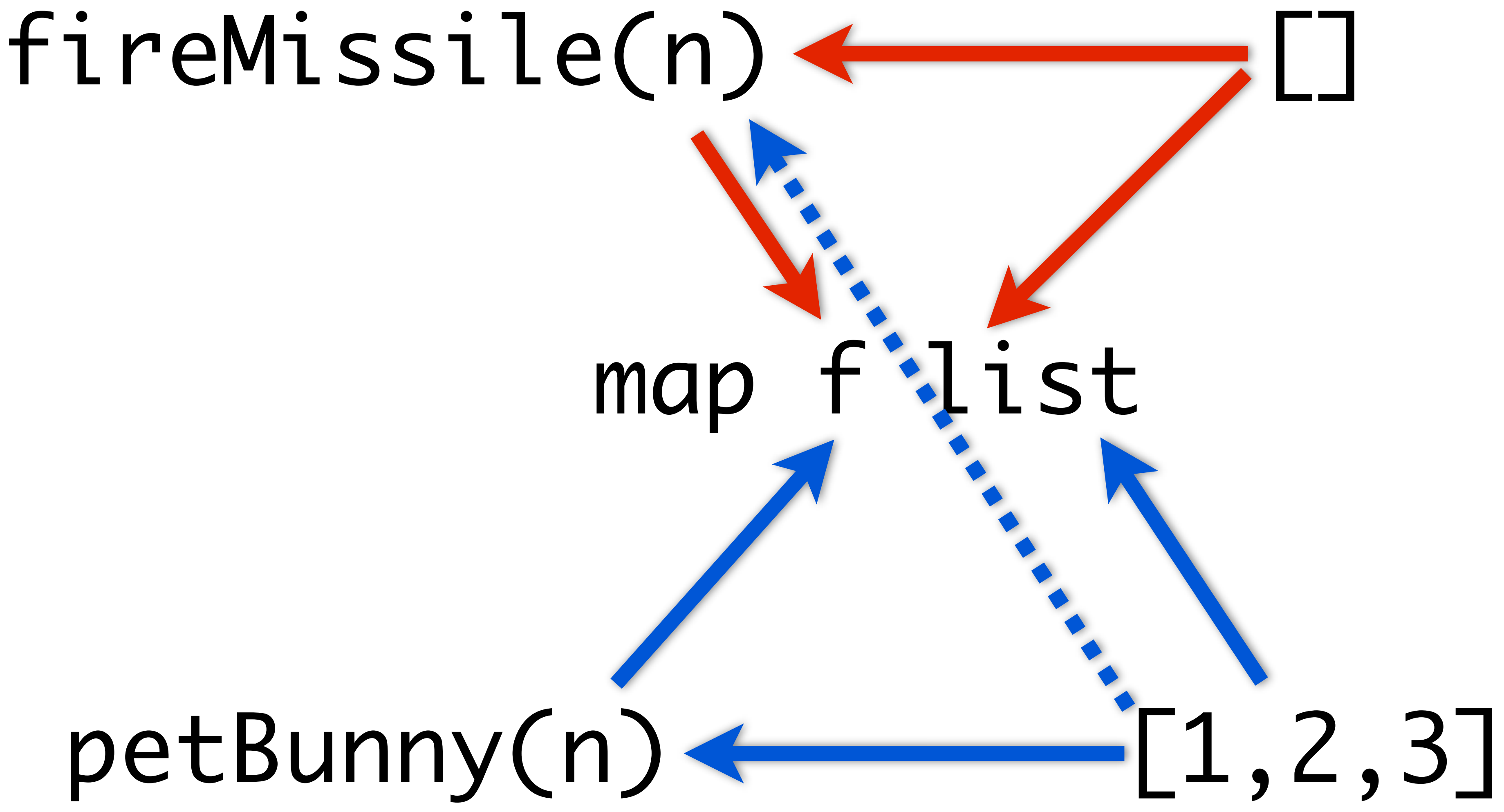
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



fireMissile(n)

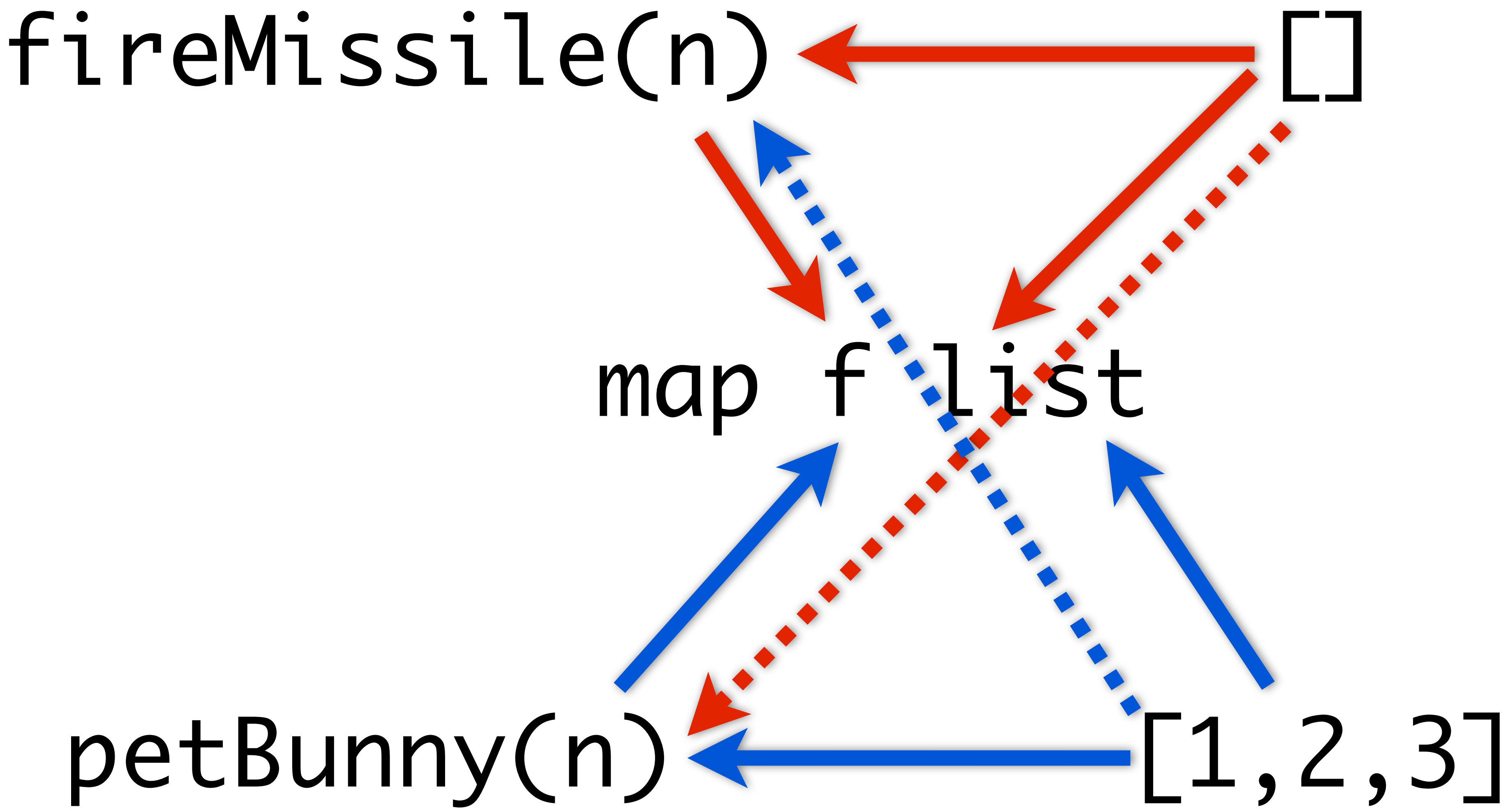
[]

map f list

petBunny(n)

[1,2,3]

Problem: Cross-flow



No attention to order.

Monotonic.

A different approach: Small-step analysis

(Joint work with David Van Horn)

Easier to understand.

Simpler to derive.

Faster to compute.

A program is an
infinite state machine.

An analysis is a
finite state machine.

Small-step machine

Small-step machine

- Convert program e into machine state s_0

Small-step machine

- Convert program e into machine state s_0
- Transition from state s_n to state $\mathrm{s}_{\mathrm{n}+1}$

$$\begin{tikzcd}
\mathrm{e} \arrow[d] & & & & & \\
\mathrm{s}_0 \arrow[r] & \mathrm{s}_1 \arrow[r] & \mathrm{s}_2 \arrow[r] & \mathrm{s}_3 \arrow[r] & \mathrm{s}_4 \arrow[r] & \ldots
\end{tikzcd}$$

Analysis machine

$$\begin{tikzcd}
e \arrow[d] & & & & & \\
s_0 \arrow[r] & s_1 \arrow[r] & s_2 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[r] & \ldots
\end{tikzcd}$$

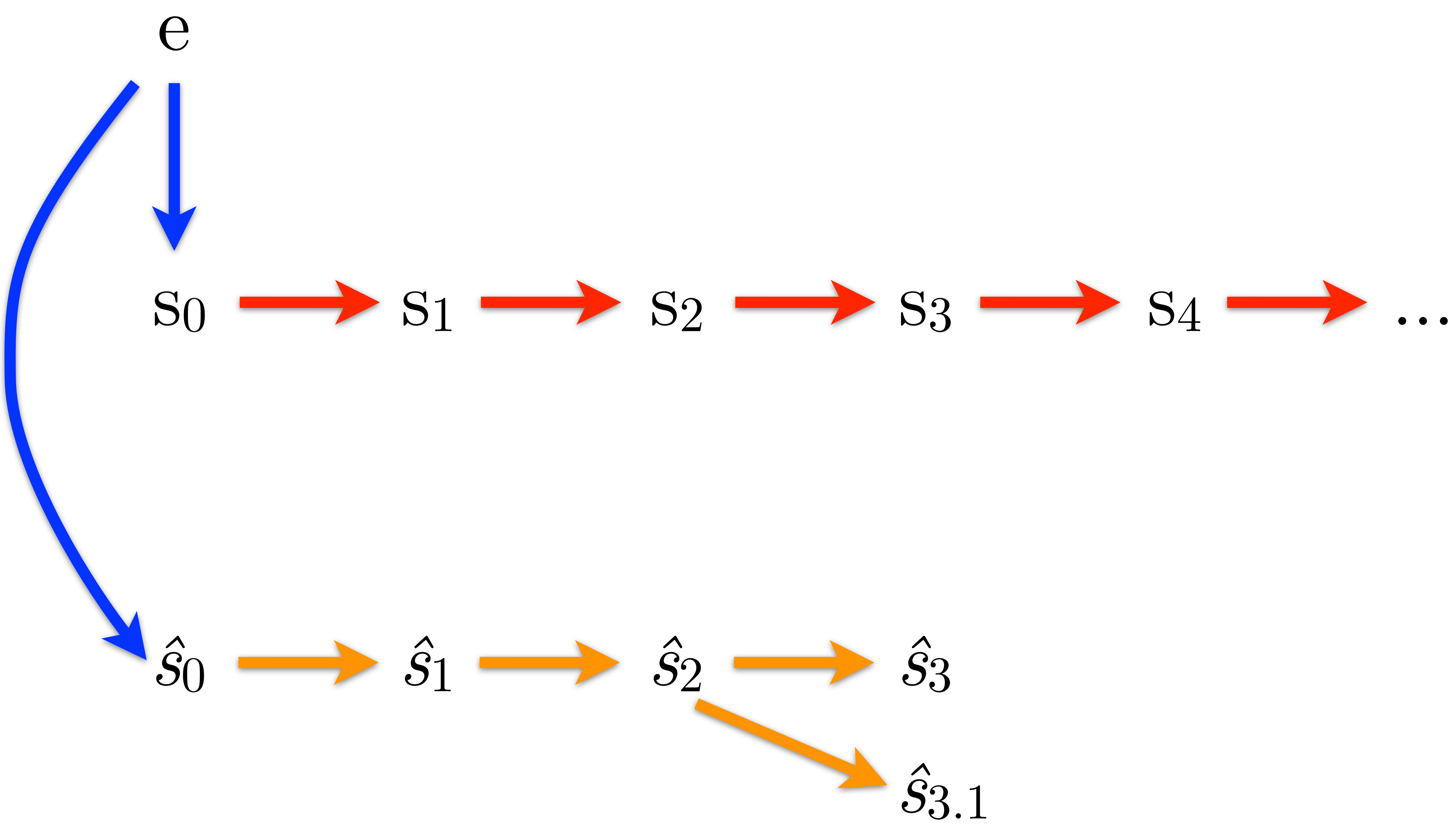
Analysis machine

$$\begin{tikzcd}
e \arrow[d] & & & & & \\
s_0 \arrow[r] & s_1 \arrow[r] & s_2 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[r] & \dots
\end{tikzcd}$$

Analysis machine

$$\begin{tikzcd}
e \arrow[d] \arrow[dd, bend right] & & & & & \\
s_0 \arrow[r] & s_1 \arrow[r] & s_2 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[r] & \dots \\
\hat{s}_0 & & & & &
\end{tikzcd}$$

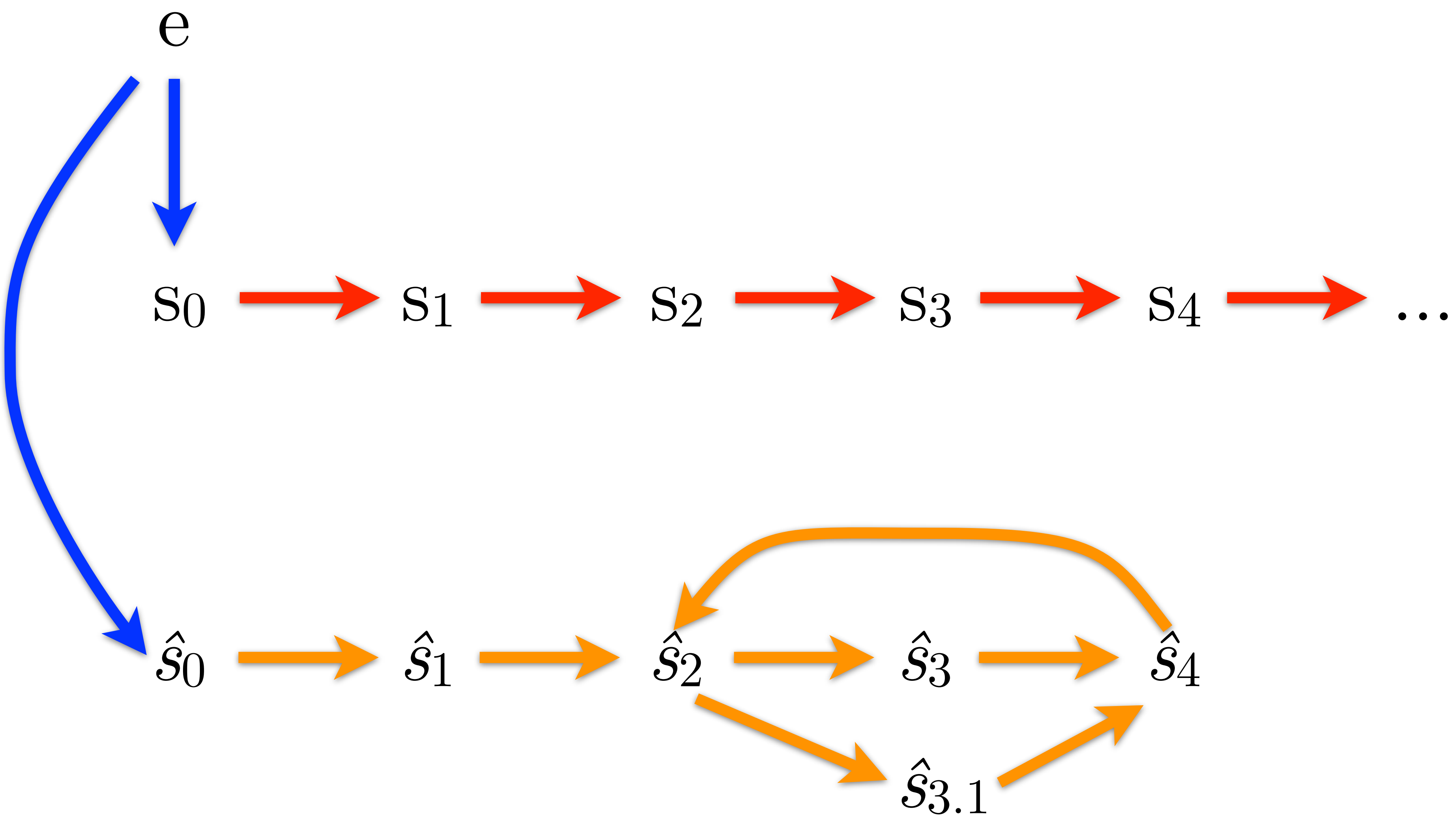
Analysis machine



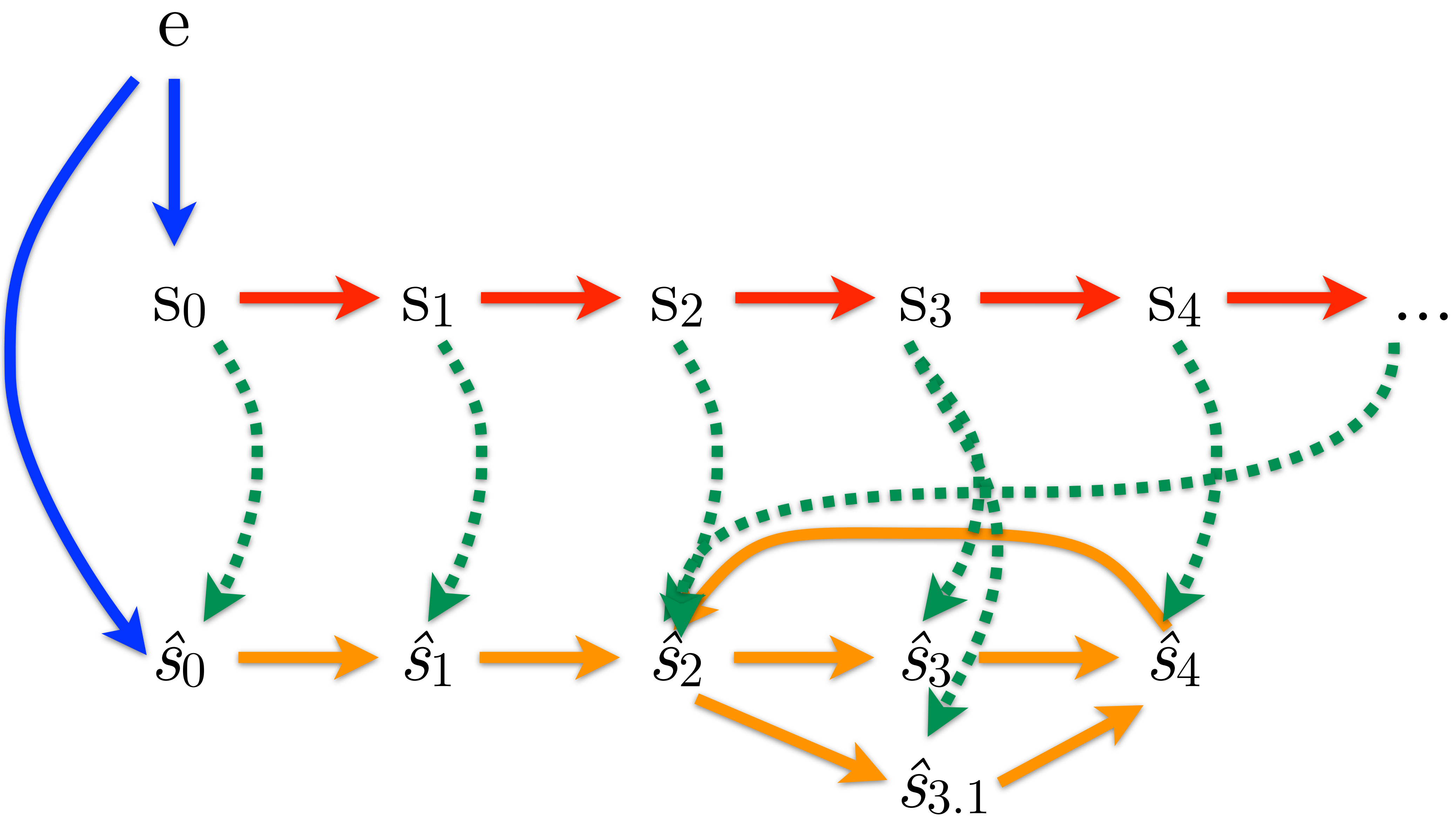
Analysis machine

$$\begin{tikzcd}
e \arrow[d] \arrow[dd, bend right] & & & & & \\
s_0 \arrow[r] & s_1 \arrow[r] & s_2 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[r] & \ldots \\
\hat{s}_0 \arrow[r] & \hat{s}_1 \arrow[r] & \hat{s}_2 \arrow[r] \arrow[dr] & \hat{s}_3 \arrow[r] & \hat{s}_4 & \\
 & & & \hat{s}_{3.1} \arrow[ur] & &
\end{tikzcd}$$

Analysis machine



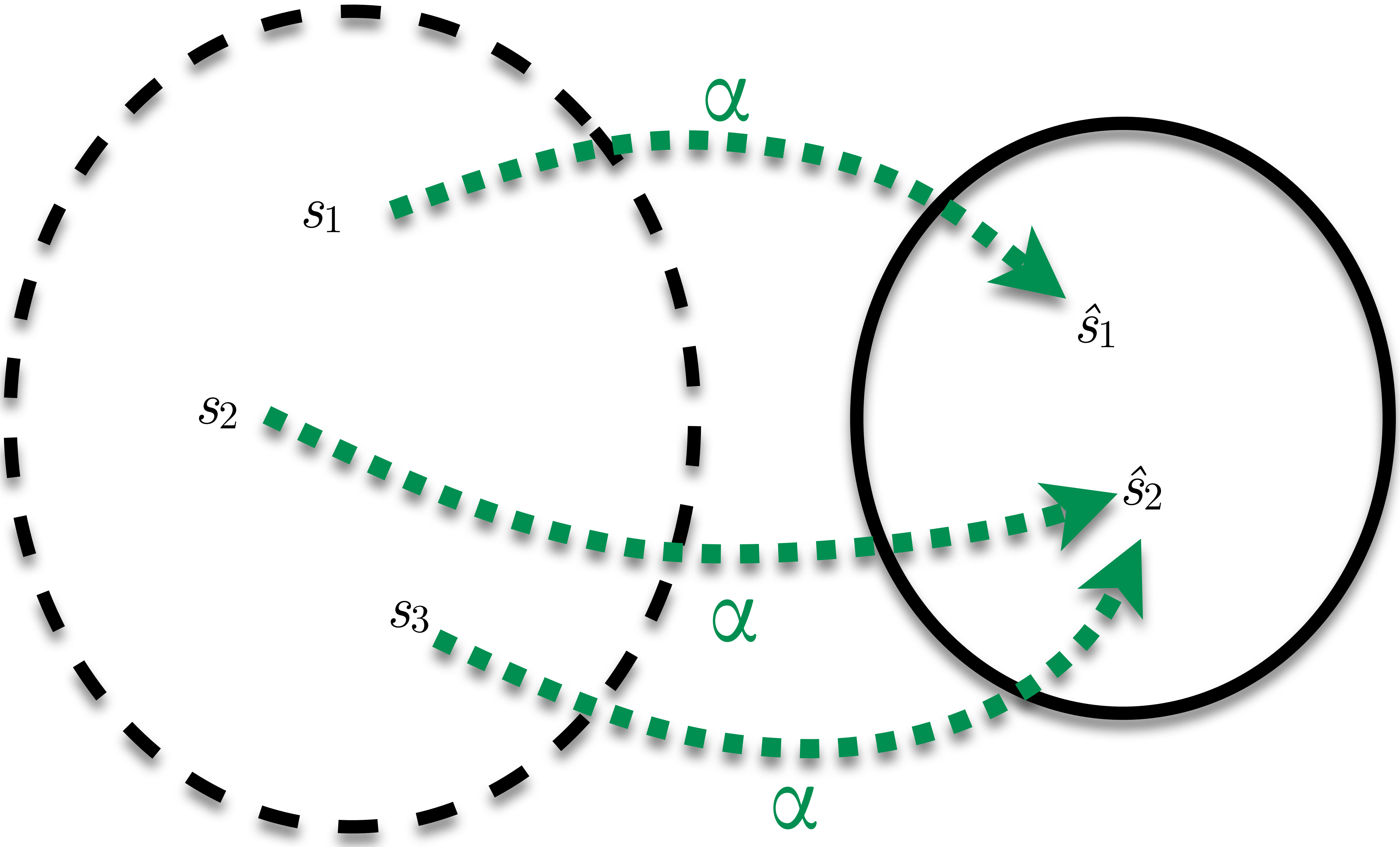
Analysis machine



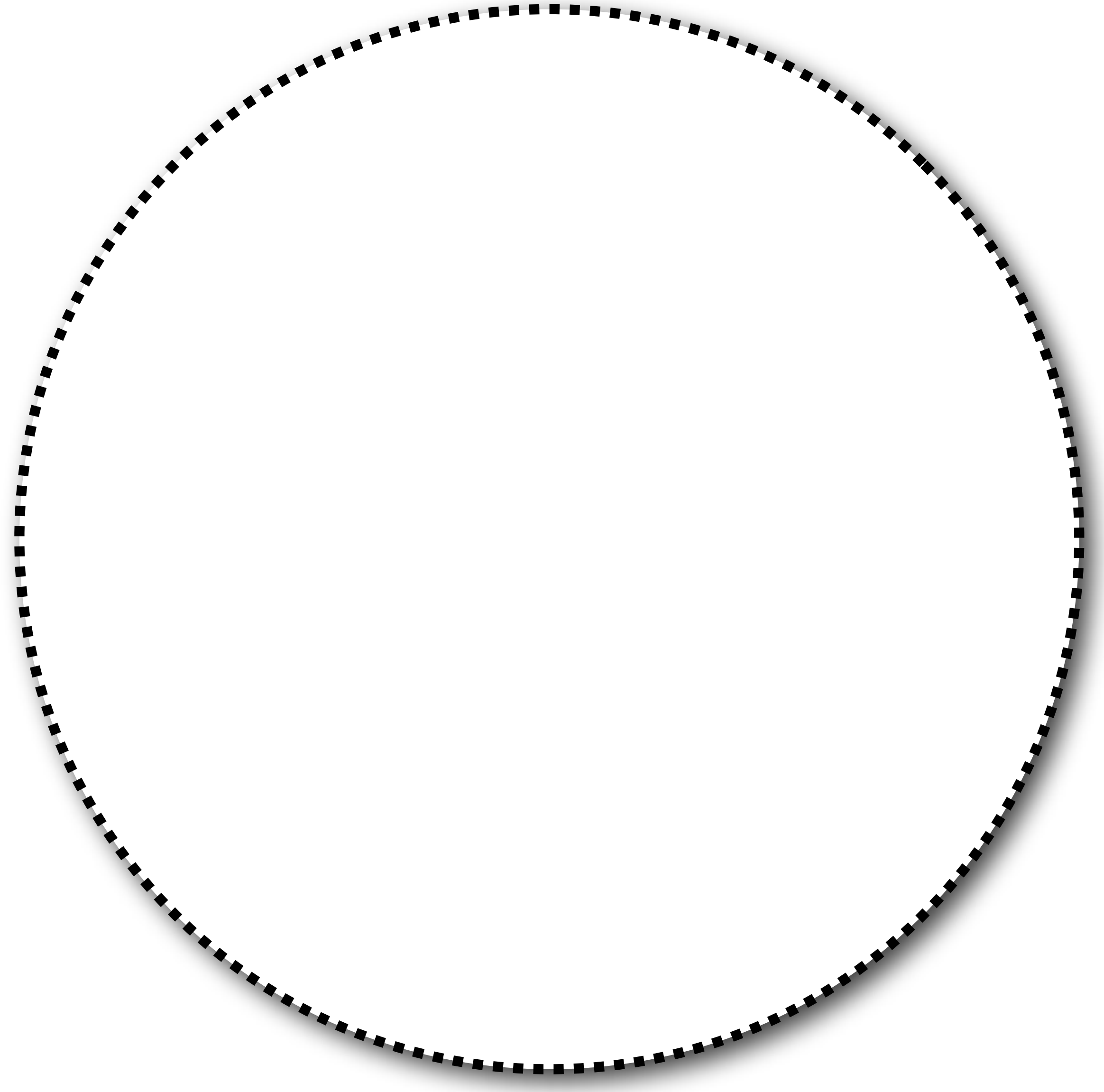
Theorem: The analysis simulates the machine.

“Concrete” “Abstract”

“Concrete” “Abstract”

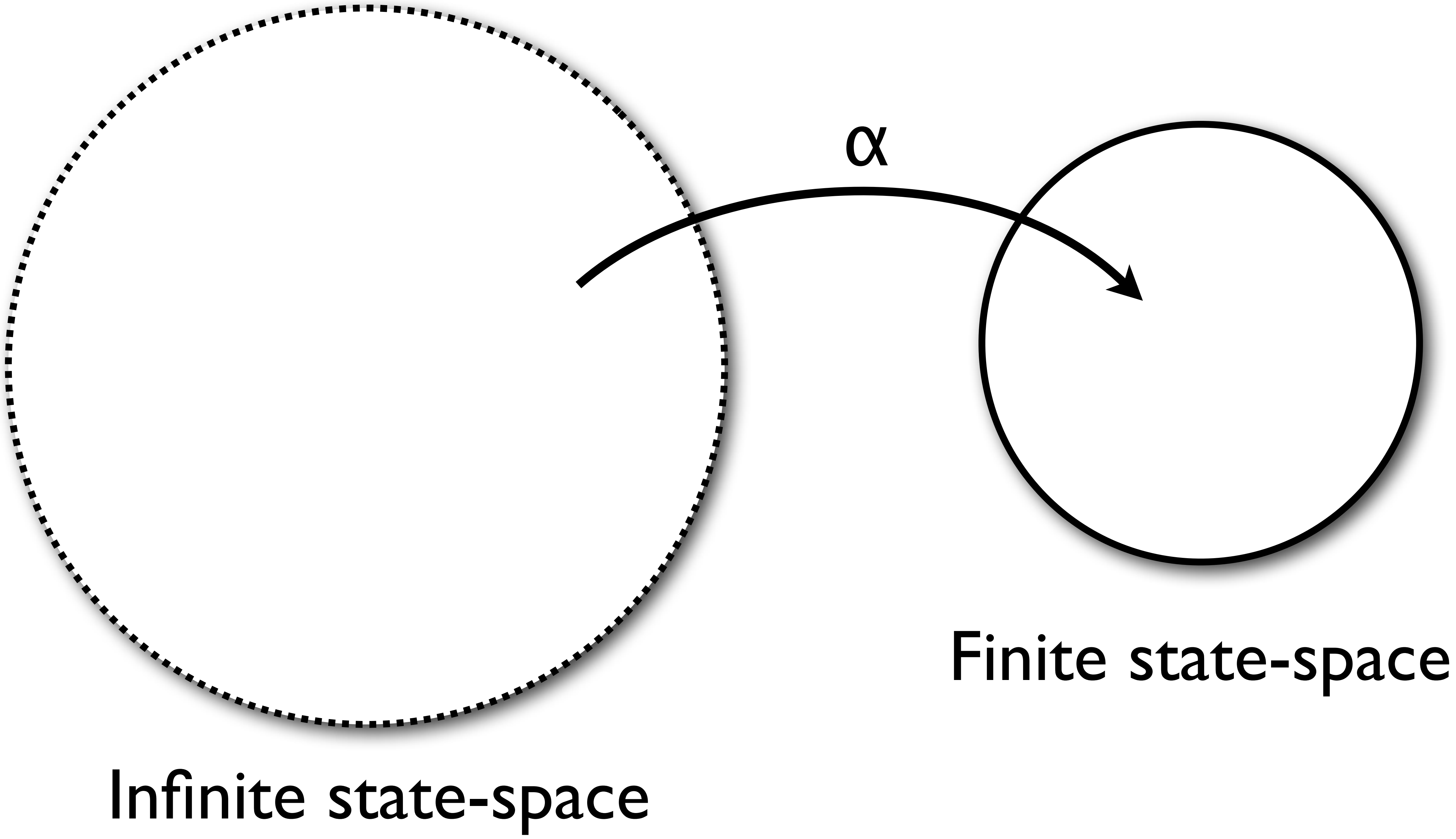


Small-step analysis...



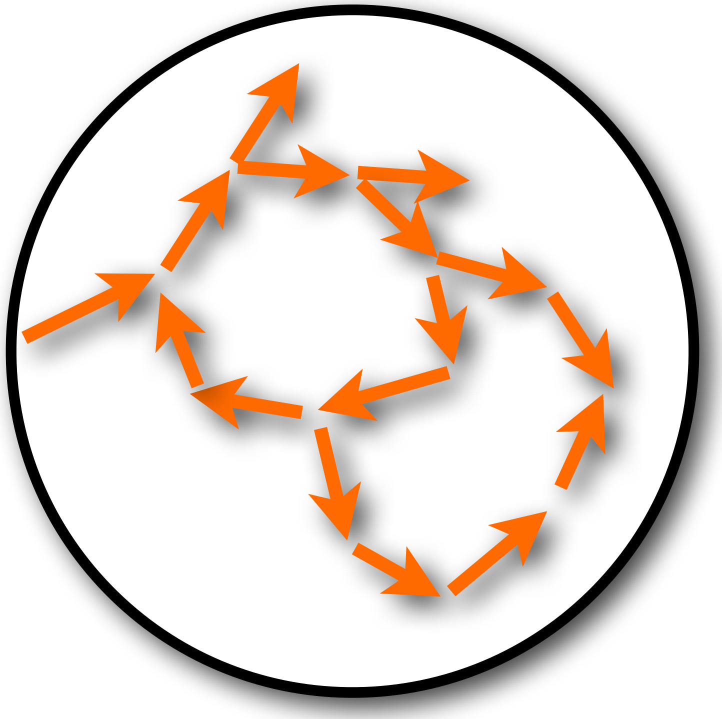
Infinite state-space

Small-step analysis...

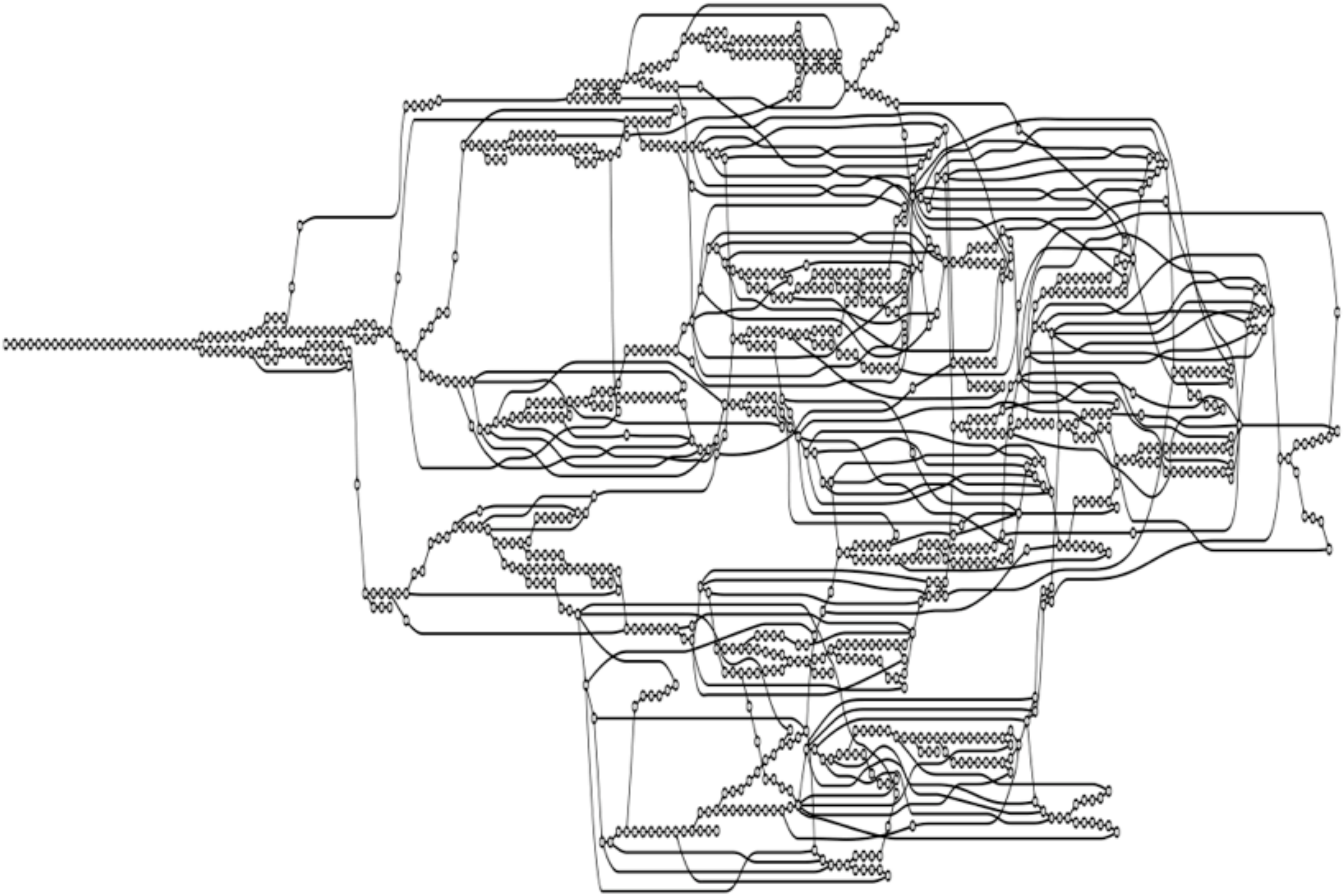


...is bounded graph search.

...is bounded graph search.



Finite state-space



Example:
Small steps for CPS

Continuation-passing style

$$f, e \in \mathsf{Exp} = \mathsf{Var} + \mathsf{Lam} + \mathsf{App}$$

Continuation-passing style

$$f, e \in \mathsf{Exp} = \mathsf{Var} + \mathsf{Lam}$$

Continuation-passing style

$$f, e \in \mathsf{Exp} = \mathsf{Var} + \mathsf{Lam}$$

$$lam \in \mathsf{Lam} ::= (\lambda\ (v_1 \ldots v_n)\ call)$$

Continuation-passing style

$$
\begin{aligned}
f, e &\in \mathsf{Exp} = \mathsf{Var} + \mathsf{Lam} \\
lam &\in \mathsf{Lam} ::= (\lambda\ (v_1 \ldots v_n)\ call) \\
call &\in \mathsf{Call} ::= (f\ e_1 \ldots e_n)
\end{aligned}
$$

No call returns

Callers pass callbacks

Still Turing-complete

Concrete state-space

$$\varsigma \in \Sigma = \textsf{Call} \times \mathit{Env}$$

Concrete state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times Env$$

$$\rho \in Env = \mathsf{Var} \rightharpoonup Clo$$

Concrete state-space

$$\begin{aligned} \varsigma \in \Sigma &= \mathsf{Call} \times \mathit{Env} \\ \rho \in \mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Clo} \\ \mathit{clo} \in \mathit{Clo} &= \mathsf{Lam} \times \mathit{Env} \end{aligned}$$

Concrete semantics

$$(\Rightarrow) \subseteq \Sigma \times \Sigma$$

Concrete semantics

$$\mathcal{E} : \mathsf{Exp} \times Env \rightharpoonup Clo$$

$$\mathcal{E}(\,lam, \rho) = (\,lam, \rho)$$

$$\mathcal{E}(v, \rho) = \rho(v)$$

$(\llbracket \texttt{(}f\ e_1 \ldots e_n\texttt{)} \rrbracket, \rho) \Rightarrow (call, \rho'')$, where

$$(\llbracket \texttt{(}f\ e_1 \ldots e_n\texttt{)} \rrbracket, \rho) \Rightarrow (call, \rho''), \text{ where}$$

$$(\llbracket \texttt{(λ (}v_1 \ldots v_n\texttt{)}\ call\texttt{)} \rrbracket, \rho') = \mathcal{E}(f, \rho)$$

$$\begin{aligned}(\llbracket(f\ e_1 \ldots e_n)\rrbracket, \rho) &\Rightarrow (call, \rho''), \text{ where}\\ (\llbracket(\lambda\ (v_1 \ldots v_n)\ \ call)\rrbracket, \rho') &= \mathcal{E}(f, \rho)\\ clo_i &= \mathcal{E}(e_i, \rho)\end{aligned}$$

$$\begin{aligned}
(\llbracket(f\ e_1 \ldots e_n)\rrbracket, \rho) &\Rightarrow (call, \rho''), \text{ where} \\
(\llbracket(\lambda\ (v_1 \ldots v_n)\ \ call)\rrbracket, \rho') &= \mathcal{E}(f, \rho) \\
clo_i &= \mathcal{E}(e_i, \rho) \\
\rho'' &= \rho'[v_i \mapsto clo_i]
\end{aligned}$$

To analyze?

Make it finite!

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \qquad \mathit{Clo}$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam} \times \mathit{Env}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \qquad \mathit{Clo}$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam} \times \mathit{Env}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \mathit{Clo}$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam} \times \mathit{Env}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \qquad \mathit{Clo}$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam} \times \mathit{Env}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \qquad \mathit{Clo}$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam} \times \mathit{Env}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times Env$$

$$\rho \in Env = \mathsf{Var} \rightharpoonup \qquad Clo$$

$$clo \in Clo = \mathsf{Lam}$$

Abstract state-space

$$\varsigma \in \Sigma = \mathsf{Call} \times \mathit{Env}$$

$$\rho \in \mathit{Env} = \mathsf{Var} \rightharpoonup \mathcal{P}\left(\mathit{Clo}\right)$$

$$\mathit{clo} \in \mathit{Clo} = \mathsf{Lam}$$

Abstract state-space

$$\hat{\varsigma} \in \hat{\Sigma} = \mathsf{Call} \times \widehat{Env}$$

$$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \rightharpoonup \mathcal{P}\left(\widehat{Clo}\right)$$

$$\widehat{clo} \in \widehat{Clo} = \mathsf{Lam}$$

$$\alpha(lam, \rho) = lam$$

$$\alpha(call, \rho) = (call, \alpha(\rho))$$

$$\alpha(\rho) = \lambda v. \left\{ \alpha(\rho'(v)) : \rho' \text{ is reachable in } \rho \right\}$$

Abstract semantics

$$(\leadsto) \subseteq \hat{\Sigma} \times \hat{\Sigma}$$

Abstract semantics

$$\mathcal{E} : \mathsf{Exp} \times Env \rightarrow \qquad Clo$$

Abstract semantics

$$\hat{\mathcal{E}} : \mathsf{Exp} \times \widehat{Env} \to \mathcal{P}\left(\widehat{Clo}\right)$$

$$\hat{\mathcal{E}}(lam, \hat{\rho}) = (lam, \hat{\rho})$$

$$\hat{\mathcal{E}}(v, \hat{\rho}) = \hat{\rho}(v)$$

$$\hat{\mathcal{E}}(lam, \hat{\rho}) = \{lam\}$$

$$\hat{\mathcal{E}}(v, \hat{\rho}) = \hat{\rho}(v)$$

$(\llbracket \texttt{(}f\ e_1 \ldots e_n\texttt{)} \rrbracket, \hat{\rho}) \leadsto (call, \hat{\rho}')$, where

$([\![(f\ e_1 \ldots e_n)]\!], \hat{\rho}) \rightsquigarrow (call, \hat{\rho}')$, where

$[\![(\lambda\ (v_1 \ldots v_n)\ call)]\!] \in \hat{\mathcal{E}}(f, \hat{\rho})$

$$(\llbracket (f\ e_1 \ldots e_n) \rrbracket, \hat{\rho}) \rightsquigarrow (call, \hat{\rho}'), \text{ where}$$

$$\llbracket (\lambda\ (v_1 \ldots v_n)\ \ call) \rrbracket \in \hat{\mathcal{E}}(f, \hat{\rho})$$

$$\hat{C}_i = \hat{\mathcal{E}}(e_i, \hat{\rho})$$

$$(\llbracket (f\ e_1 \ldots e_n) \rrbracket, \hat{\rho}) \leadsto (call, \hat{\rho}'), \text{ where}$$

$$\llbracket (\lambda\ (v_1 \ldots v_n)\ \ call) \rrbracket \in \hat{\mathcal{E}}(f, \hat{\rho})$$

$$\hat{C}_i = \hat{\mathcal{E}}(e_i, \hat{\rho})$$

$$\hat{\rho}' = \hat{\rho} \sqcup [v_i \mapsto \hat{C}_i]$$

$(\llbracket(f\ e_1 \ldots e_n)\rrbracket, \hat{\rho}) \rightsquigarrow (call, \hat{\rho}')$, where

$$\llbracket(\lambda\ (v_1 \ldots v_n)\ \ call)\rrbracket \in \hat{\mathcal{E}}(f, \hat{\rho})$$

$$\hat{C}_i = \hat{\mathcal{E}}(e_i, \hat{\rho})$$

$$\hat{\rho}' = \hat{\rho} \sqcup [v_i \mapsto \hat{C}_i]$$

$$
\begin{aligned}
(\llbracket \mathsf{(}f\ e_1 \ldots e_n\mathsf{)} \rrbracket, \hat{\rho}) &\leadsto (call, \hat{\rho}'), \text{ where} \\
\llbracket \mathsf{(\lambda\ (}v_1 \ldots v_n\mathsf{)}\ call\mathsf{)} \rrbracket &\in \hat{\mathcal{E}}(f, \hat{\rho}) \\
\hat{C}_i &= \hat{\mathcal{E}}(e_i, \hat{\rho}) \\
\hat{\rho}' &= \hat{\rho} \sqcup [v_i \mapsto \hat{C}_i]
\end{aligned}
$$

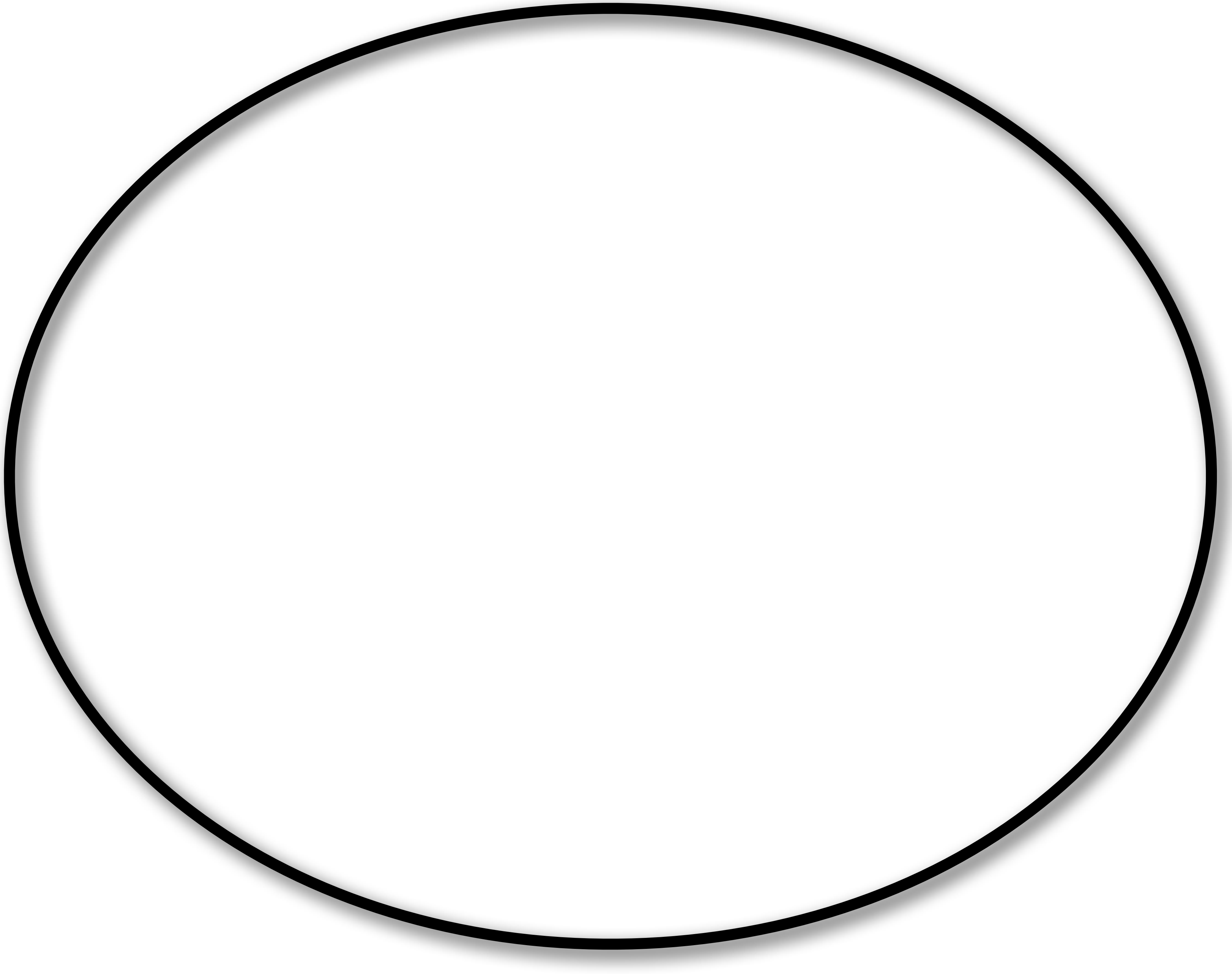
$$
\begin{aligned}
(\llbracket \mathsf{(}f\ e_1 \ldots e_n\mathsf{)} \rrbracket, \rho) &\Rightarrow (call, \rho''), \text{ where} \\
(\llbracket \mathsf{(\lambda\ (}v_1 \ldots v_n\mathsf{)}\ call\mathsf{)} \rrbracket, \rho') &= \mathcal{E}(f, \rho) \\
clo_i &= \mathcal{E}(e_i, \rho) \\
\rho'' &= \rho'[v_i \mapsto clo_i]
\end{aligned}
$$

Soundness

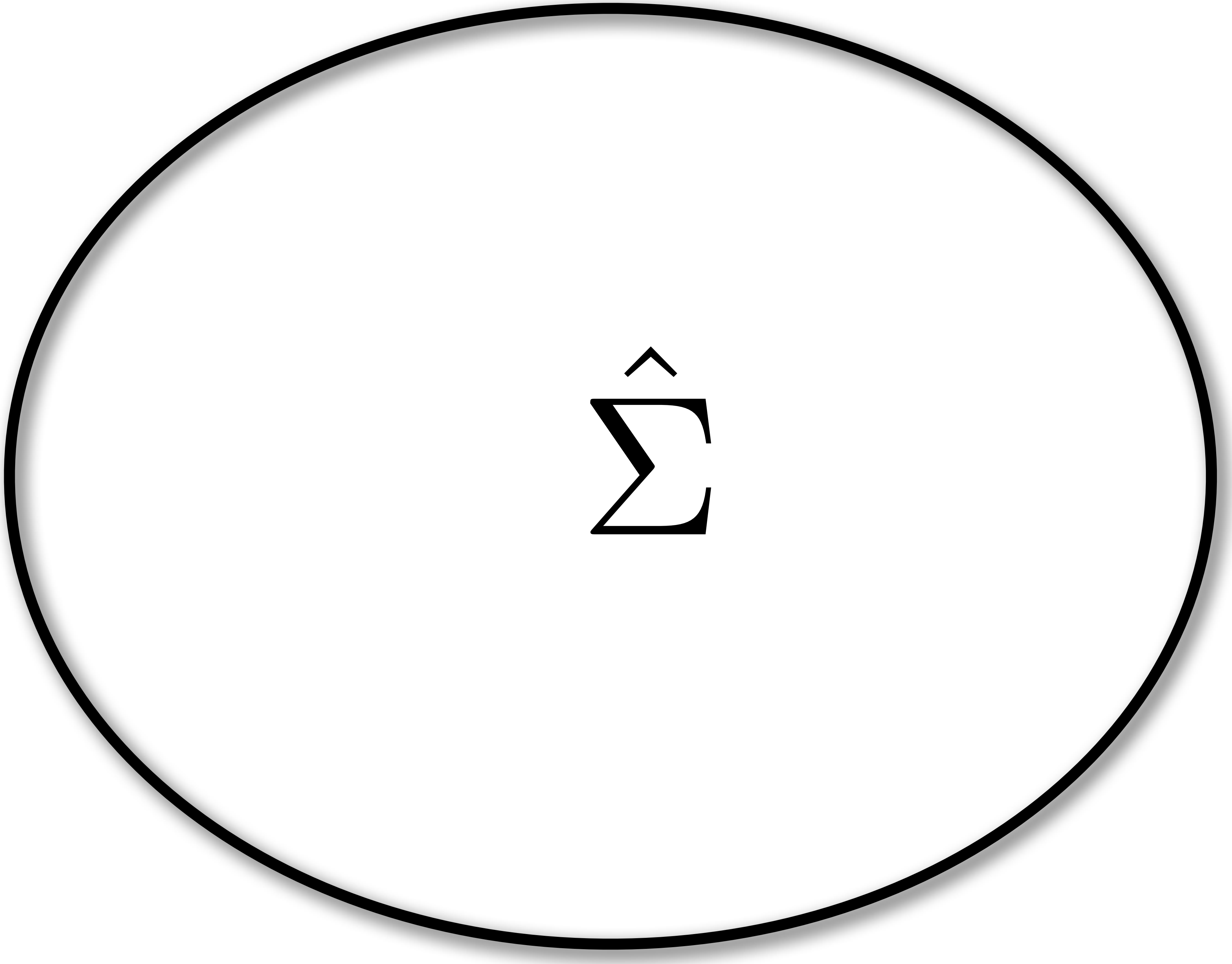
$$\begin{tikzcd}
\varsigma \arrow[r, "\Rightarrow"] \arrow[d, "\alpha"'] & \varsigma' \arrow[d, "\alpha"] \\
{} \arrow[d, "\sqsubseteq"'] & {} \arrow[d, "\sqsubseteq"] \\
\hat{\varsigma} \arrow[r, dotted, "\leadsto"'] & \hat{\varsigma}'
\end{tikzcd}$$

Theorem: If the concrete takes a step,
then the abstract can take a matching step.

Running 0CFA

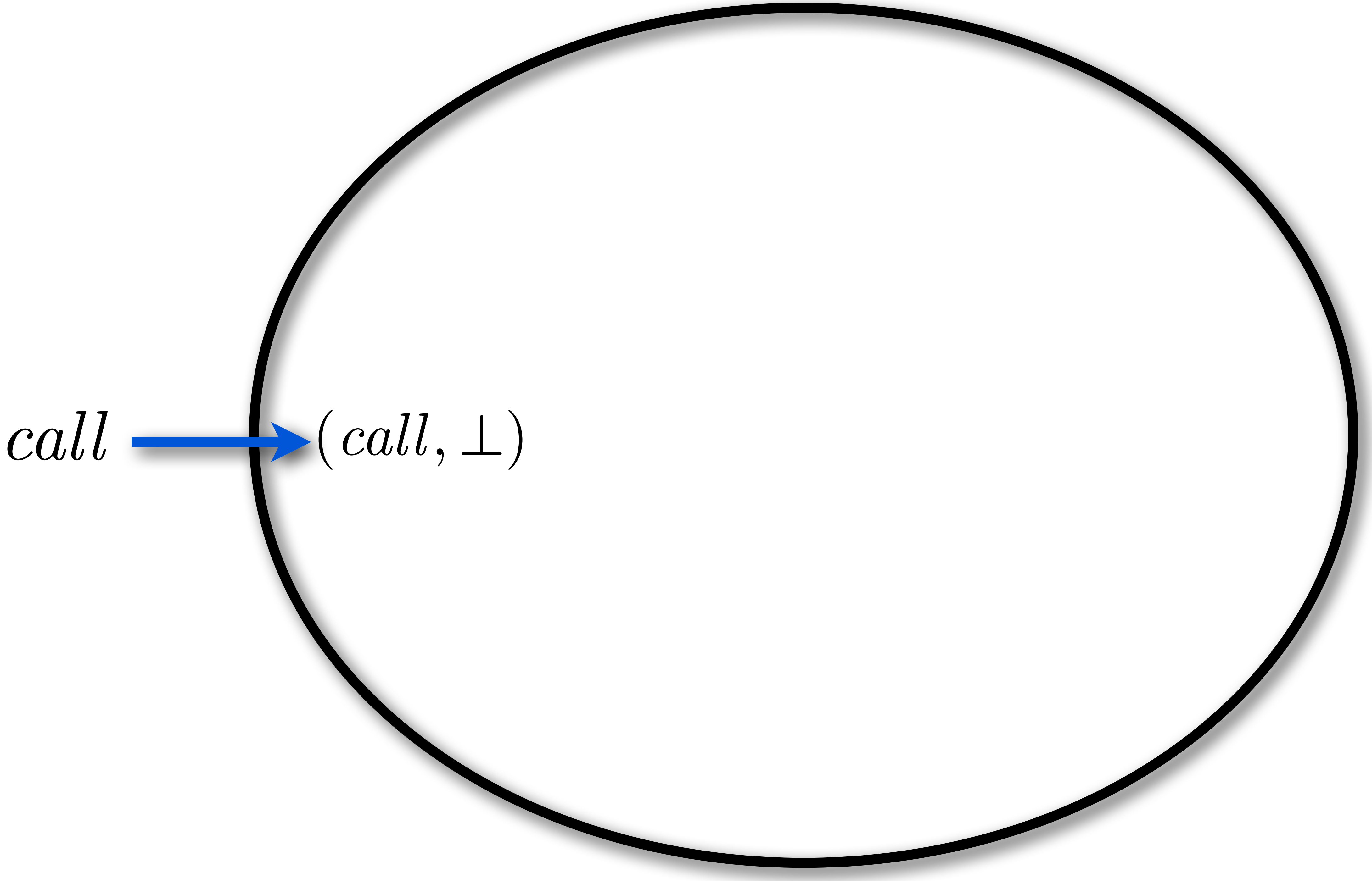


Running 0CFA

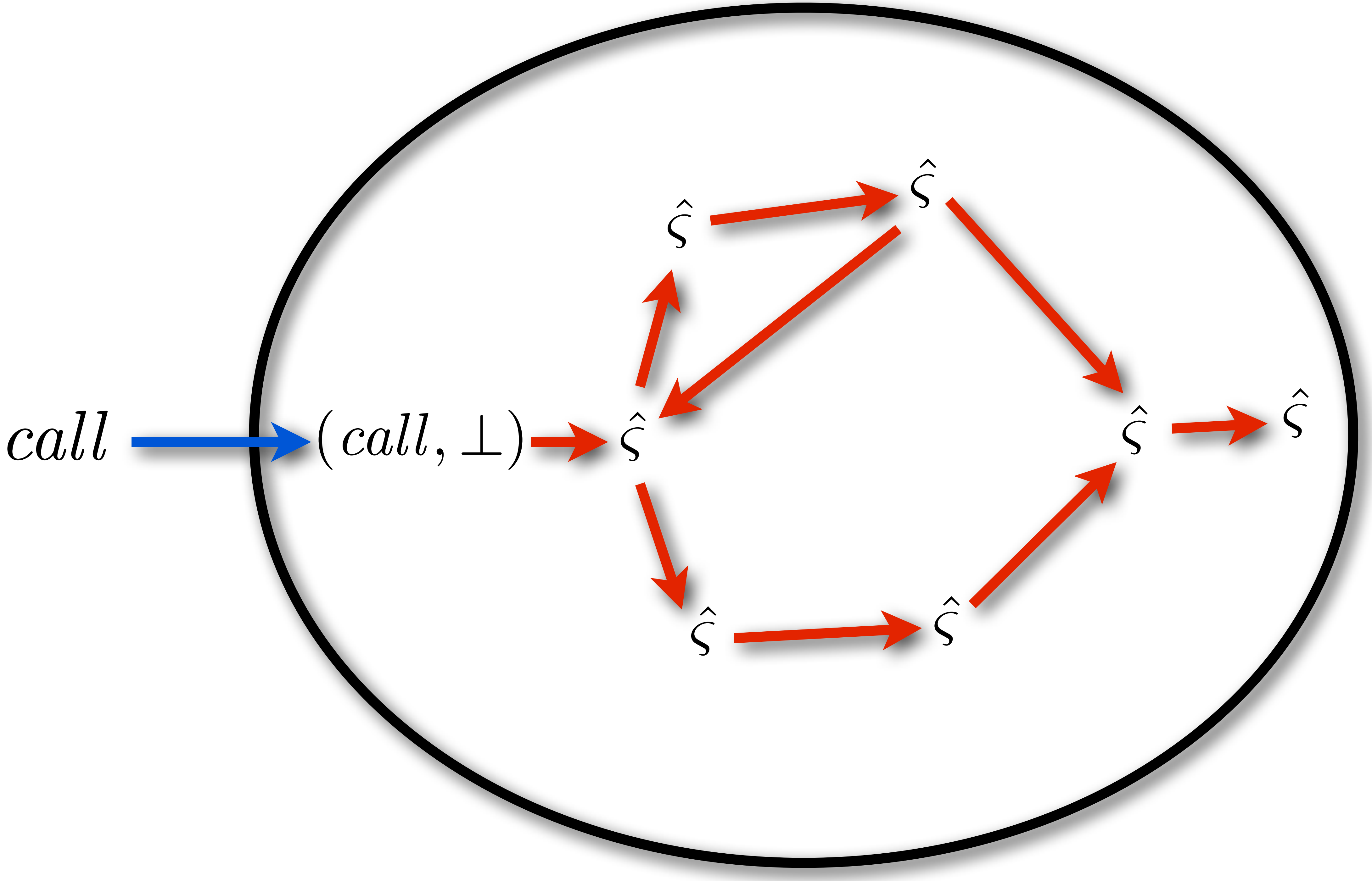


$\hat{\Sigma}$

Running 0CFA



Running 0CFA



Order between states is preserved.

Monotonic growth not required.

How about the next level?

ANF

$$
\begin{aligned}
f, \text{æ} \in \mathsf{AExp} &= \mathsf{Var} + \mathsf{Lam} \\
e \in \mathsf{Exp} &::= \texttt{(let ((}v\ call\texttt{))}\ e'\texttt{)} \\
&\mid \quad call \\
&\mid \quad \text{æ} \\
call \in \mathsf{Call} &::= \texttt{(}f\ \text{æ}_1 \ldots \text{æ}_n\texttt{)}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times \mathit{Kont} \\
\mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\mathit{Store} &= \mathit{Addr} \to \mathit{Clo} \\
\mathit{Kont} &= \mathsf{Var} \times \mathsf{Exp} \times \mathit{Env} \times \mathit{Kont} + \{\mathbf{halt}\}
\end{aligned}
$$

Addr is an infinite set of addresses

$$\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times \mathit{Kont} \\
\mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\mathit{Store} &= \mathit{Addr} \to \mathit{Clo} \\
\mathit{Kont} &= \mathsf{Var} \times \mathsf{Exp} \times \mathit{Env} \times \mathit{Kont} + \{\mathbf{halt}\}
\end{aligned}$$

Addr is an infinite set of addresses

$$
\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times \mathit{Kont} \\
\mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\mathit{Store} &= \mathit{Addr} \to \mathit{Clo} \\
\mathit{Kont} &= \mathsf{Var} \times \mathsf{Exp} \times \mathit{Env} \times \mathit{Kont} + \{\mathbf{halt}\}
\end{aligned}
$$

Addr is an infinite set of addresses

$$
\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times \mathit{Kont} \\
\mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\mathit{Store} &= \mathit{Addr} \to \quad \mathit{Clo} \\
\mathit{Kont} &= \mathsf{Var} \times \mathsf{Exp} \times \mathit{Env} \times \mathit{Kont} + \{\mathbf{halt}\}
\end{aligned}
$$

Addr is an infinite set of addresses

$$
\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times \mathit{Kont} \\
\mathit{Env} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\mathit{Store} &= \mathit{Addr} \to \quad \mathit{Clo} \\
\mathit{Kont} &= \mathsf{Var} \times \mathsf{Exp} \times \mathit{Env} \times \mathit{Addr} + \{\mathbf{halt}\}
\end{aligned}
$$

Addr is an infinite set of addresses

$$
\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times Kont \\
Env &= \mathsf{Var} \rightharpoonup Addr \\
Store &= Addr \to \quad Clo + Kont \\
Kont &= \mathsf{Var} \times \mathsf{Exp} \times Env \times Addr + \{\mathbf{halt}\}
\end{aligned}
$$

$Addr$ is an infinite set of addresses

$$\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times Kont \\
Env &= \mathsf{Var} \rightharpoonup Addr \\
Store &= Addr \to \quad Clo + Kont \\
Kont &= \mathsf{Var} \times \mathsf{Exp} \times Env \times Addr + \{\mathbf{halt}\}
\end{aligned}$$

$Addr$ is an finite set of addresses

$$\begin{aligned}
\Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Store} \times Kont \\
Env &= \mathsf{Var} \rightharpoonup Addr \\
Store &= Addr \to \mathcal{P}\left(Clo + Kont\right) \\
Kont &= \mathsf{Var} \times \mathsf{Exp} \times Env \times Addr + \{\mathbf{halt}\}
\end{aligned}$$

$Addr$ is an finite set of addresses

And, other machines?

CEK (F&F, 1986)

$$\varsigma \longmapsto_{CEK} \varsigma'$$

$\langle x, \rho, \kappa \rangle$	$\langle v, \rho', \kappa \rangle$ where $\rho(x) = (v, \rho')$
$\langle (e_0 e_1), \rho, \kappa \rangle$	$\langle e_0, \rho, \mathbf{ar}(e_1, \rho, \kappa) \rangle$
$\langle v, \rho, \mathbf{ar}(e, \rho', \kappa) \rangle$	$\langle e, \rho', \mathbf{fn}(v, \rho, \kappa) \rangle$
$\langle v, \rho, \mathbf{fn}((\lambda x.e), \rho', \kappa) \rangle$	$\langle e, \rho'[x \mapsto (v, \rho)], \kappa \rangle$

Krivine (ICFP 2010)

$$\begin{array}{rcl}
\varsigma \in \Sigma & = & \mathit{Exp} \times \mathit{Env} \times \mathit{Store} \times \mathit{Kont} \\
s \in \mathit{Storable} & ::= & \mathbf{d}(e, \rho) \mid \mathbf{c}(v, \rho) \\
\kappa \in \mathit{Kont} & ::= & \mathbf{mt} \mid \mathbf{c_1}(a, \kappa) \mid \mathbf{c_2}(a, \kappa)
\end{array}$$

CM (ICFP 2010)

$$\varsigma \longmapsto_{CM} \varsigma'$$

$$\begin{array}{l|r}
\langle \texttt{fail}, \rho, \sigma, \kappa \rangle & \langle \texttt{fail}, \rho, \sigma, \mathbf{mt}^{\emptyset} \rangle \\
\langle \texttt{(frame } R\ e\texttt{)}, \rho, \sigma, \kappa \rangle & \langle e, \rho, \sigma, \kappa[\overline{R} \mapsto \text{deny}] \rangle \\
\langle \texttt{(grant } R\ e\texttt{)}, \rho, \sigma, \kappa \rangle & \langle e, \rho, \sigma, \kappa[R \mapsto \text{grant}] \rangle \\
\langle \texttt{(test } R\ e_0\ e_1\texttt{)}, \rho, \sigma, \kappa \rangle & \begin{cases} \langle e_0, \rho, \sigma, \kappa \rangle & \text{if } \mathcal{OK}(R, \kappa), \\ \langle e_1, \rho, \sigma, \kappa \rangle & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{rcl}
\mathcal{OK}(\emptyset, \kappa) & & \\
\mathcal{OK}(R, \mathbf{mt}^m) & \iff & (R \cap m^{-1}(\text{deny}) = \emptyset) \\
\left. \begin{array}{r} \mathcal{OK}(R, \mathbf{fn}^m(v, \rho, \kappa)) \\ \mathcal{OK}(R, \mathbf{ar}^m(e, \rho, \kappa)) \end{array} \right\} & \iff & \begin{array}{l} (R \cap m^{-1}(\text{deny}) = \emptyset) \wedge \\ \quad \mathcal{OK}(R \setminus m^{-1}(\text{grant}), \kappa) \end{array}
\end{array}$$

Java (PLDI 2010)

$$
\begin{aligned}
\varsigma \in \Sigma &= \mathsf{Stmt} \times \mathit{BEnv} \times \mathit{Store} \times \mathit{KontPtr} \times \mathit{Time} \\
\beta \in \mathit{BEnv} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\sigma \in \mathit{Store} &= \mathit{Addr} \rightharpoonup D \\
d \in D &= \mathit{Val} \\
\mathit{val} \in \mathit{Val} &= \mathit{Obj} + \mathit{Kont} \\
o \in \mathit{Obj} &= \mathsf{ClassName} \times \mathit{BEnv} \\
\kappa \in \mathit{Kont} &= \mathsf{Var} \times \mathsf{Stmt} \times \mathit{BEnv} \times \mathit{KontPtr} \\
a \in \mathit{Addr} & \text{ is a set of addresses} \\
\overset{\kappa}{p} \in \mathit{KontPtr} &\subseteq \mathit{Addr} \\
t \in \mathit{Time} & \text{ is a set of time-stamps.}
\end{aligned}
$$

C/LLVM

$$
\begin{array}{rll}
\varsigma \in & \mathit{State} & = \mathit{Eval} + \mathit{Apply} + \mathit{AppCont} + \mathit{AppFun} \\
 & \mathit{Eval} & = \mathit{STMT}^{*} \times \mathit{FrmPtr} \times \mathit{Conf} \times \mathit{StkPtr} \\
 & \mathit{Apply} & = \mathit{LHS}^{*} \times D^{*} \times \mathit{Eval} \\
 & \mathit{AppFun} & = \mathit{FUN} \times D^{*} \times \mathit{FrmPtr} \times \mathit{Conf} \times \mathit{StkPtr} \\
 & \mathit{AppCont} & = \mathit{Cont} \times D \times \mathit{Conf} \\
 & & \\
d \in & D & = \mathit{Val} \\
\mathit{val} \in & \mathit{Val} & = \mathit{Cont} + \mathit{FUN} + \mathit{Loc} + \mathit{Bas} \\
\kappa \in & \mathit{Cont} & = \mathit{LHS} \times \mathit{STMT}^{*} \times \mathit{FrmPtr} \times \mathit{StkPtr} \\
\mathit{bas} \in & \mathit{Bas} & = \text{a set of basic values} \\
 & & \\
\mathit{loc} \in & \mathit{Loc} & = \mathit{Addr} + \mathit{StkPtr} + \mathit{Bind} \\
a \in & \mathit{Addr} & = \text{an infinite set of heap pointers} \\
\mathit{sp} \in & \mathit{StkPtr} & = \text{an infinite set of stack pointers} \\
\mathit{fp} \in & \mathit{FrmPtr} & = \mathit{StkPtr} \\
b \in & \mathit{Bind} & = \mathit{VAR} \times \mathit{FrmPtr} \\
 & & \\
c \in & \mathit{Conf} & = \mathit{Store} \times \mathit{Succ} \times \mathit{Pred} \\
\sigma \in & \mathit{Store} & = \mathit{Loc} \rightharpoonup D \\
\sigma_{+} \in & \mathit{Succ} & = \mathit{Loc} \rightharpoonup \mathit{Loc} \\
\sigma_{-} \in & \mathit{Pred} & = \mathit{Loc} \rightharpoonup \mathit{Loc}
\end{array}
$$

Up next

JavaScript

$$
\begin{array}{rcll}
\varsigma \in \Sigma & = & (\mathsf{Stmt} + \mathsf{Body}) \times \mathit{BEnv} \times \mathit{Store} \times \mathit{FPtr} & \text{[states]} \\
\beta \in \mathit{BEnv} & = & \mathsf{Var} \rightharpoonup \mathit{Addr} & \text{[binding environments]} \\
\sigma \in \mathit{Store} & = & \mathit{Addr} \rightharpoonup D & \text{[stores]} \\
d \in D & = & \mathit{Val} & \text{[denotable values]} \\
\mathit{val} \in \mathit{Val} & = & \mathit{Bas} + \mathit{Clo} + \mathit{Kont} + \mathit{Loc} & \text{[values]} \\
\mathit{bas} \in \mathit{Bas} & = & \mathit{String} + \mathit{Num} + \mathit{Boolean} & \text{[basic values]} \\
\mathit{clo} \in \mathit{Clo} & = & \mathsf{Fun} \times \mathit{BEnv} & \text{[closures]} \\
\kappa \in \mathit{Kont} & ::= & \mathbf{ret}(v, \beta, s, \mathit{fp}) & \text{[return continuations]} \\
 & \mid & \mathbf{ex}(v, \beta, s, \mathit{fp}, s') & \text{[exceptional continuations} \\
a \in \mathit{Addr} & = & \mathit{Bind} + \mathit{Field} + \mathit{FPtr} & \text{[addresses]} \\
b \in \mathit{Bind} & = & \mathsf{Var} \times \mathit{Contour} & \text{[bindings]} \\
\mathit{field} \in \mathit{Field} & = & \mathit{Loc} \times \mathit{String} & \text{[object fields]} \\
\mathit{fp} \in \mathit{FPtr} & = & \mathit{Contour} & \text{[frame pointers]} \\
\mathit{cn} \in \mathit{Contour} & \multicolumn{2}{l}{\text{is an } \textbf{infinite} \text{ set of contours}} & \\
\mathit{loc} \in \mathit{Loc} & \multicolumn{2}{l}{\text{is an } \textbf{infinite} \text{ set of locations}} &
\end{array}
$$

Bonus: Compositionality

Direct products

Direct products

$$\begin{tikzcd}
 & \Pi \arrow[r] & \Pi' \\
s \arrow[ur, dashed] \arrow[dr, dashed] & & \\
 & \hat{s} \arrow[r] & \hat{s}'
\end{tikzcd}$$

Direct products

$$\begin{tikzcd}
 & \Pi \arrow[r] \arrow[ddr] & \Pi' \\
s \arrow[ur, dashed] \arrow[dr, dashed] & & \\
 & \hat{s} \arrow[r] \arrow[uur] & \hat{s}'
\end{tikzcd}$$

Direct products

$$\begin{tikzcd}
 & \Pi \arrow[r] \arrow[ddr] & \Pi' \arrow[r] \arrow[ddr] & \Pi'' \arrow[r] \arrow[ddr] & \Pi''' \arrow[r] \arrow[ddr] & \cdots \\
s \arrow[ur, dashed] \arrow[dr, dashed] & & & & & \\
 & \hat{s} \arrow[r] \arrow[uur] & \hat{s}' \arrow[r] \arrow[uur] & \hat{s}'' \arrow[r] \arrow[uur] & \hat{s}''' \arrow[r] \arrow[uur] & \cdots
\end{tikzcd}$$

Application:
Array-bounds checks

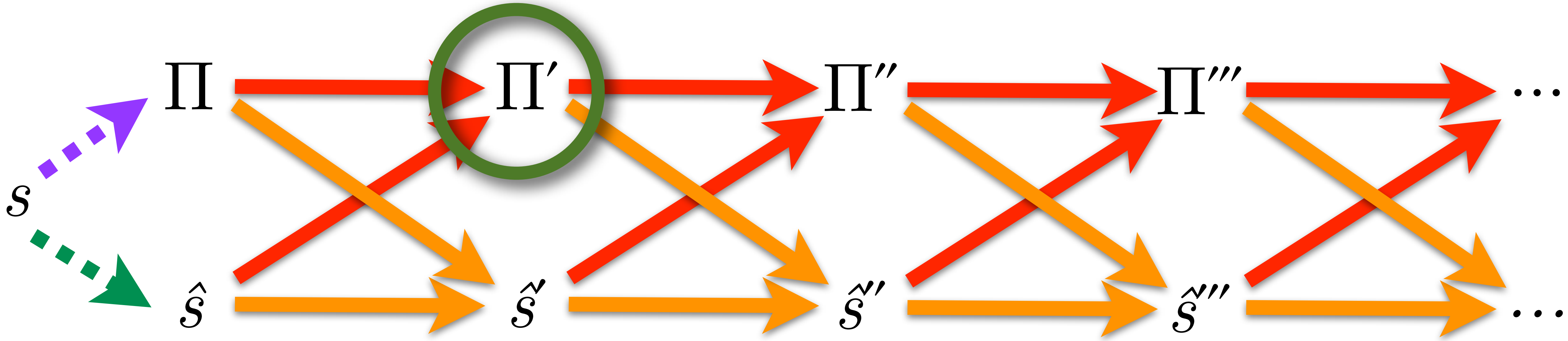
Logic-flow analysis

Logic-flow analysis

$$\begin{tikzcd}
 & \Pi \arrow[r] \arrow[ddr] & \Pi' \arrow[r] \arrow[ddr] & \Pi'' \arrow[r] \arrow[ddr] & \Pi''' \arrow[r] \arrow[ddr] & \cdots \\
s \arrow[ur, dashed] \arrow[dr, dashed] & & & & & \\
 & \hat{s} \arrow[r] \arrow[uur] & \hat{s}' \arrow[r] \arrow[uur] & \hat{s}'' \arrow[r] \arrow[uur] & \hat{s}''' \arrow[r] \arrow[uur] & \cdots
\end{tikzcd}$$

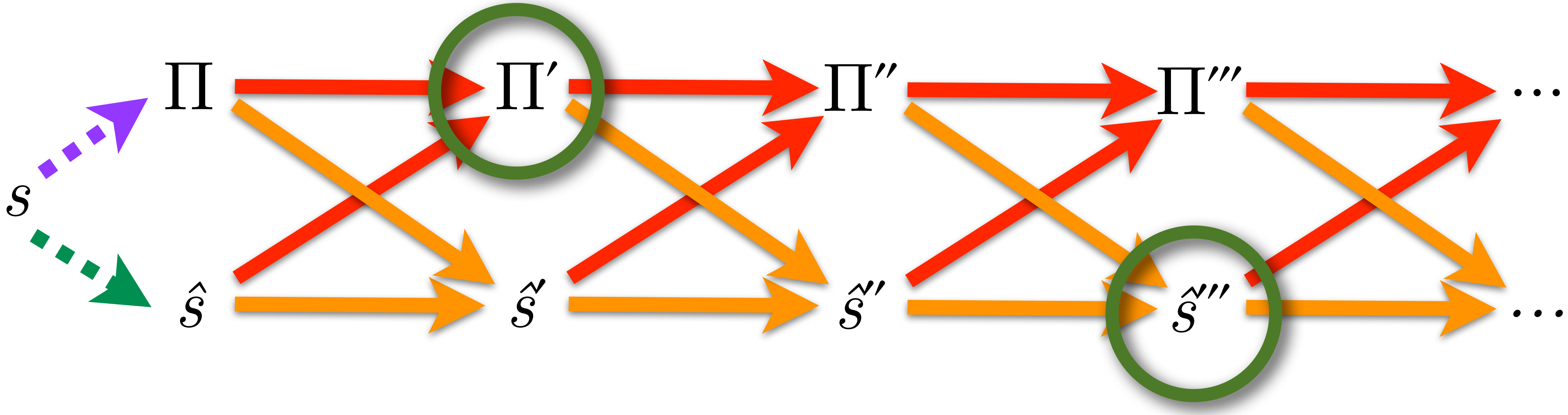
Logic-flow analysis

```
i < length(a)
```



Logic-flow analysis

`i < length(a)`



`a[i]`

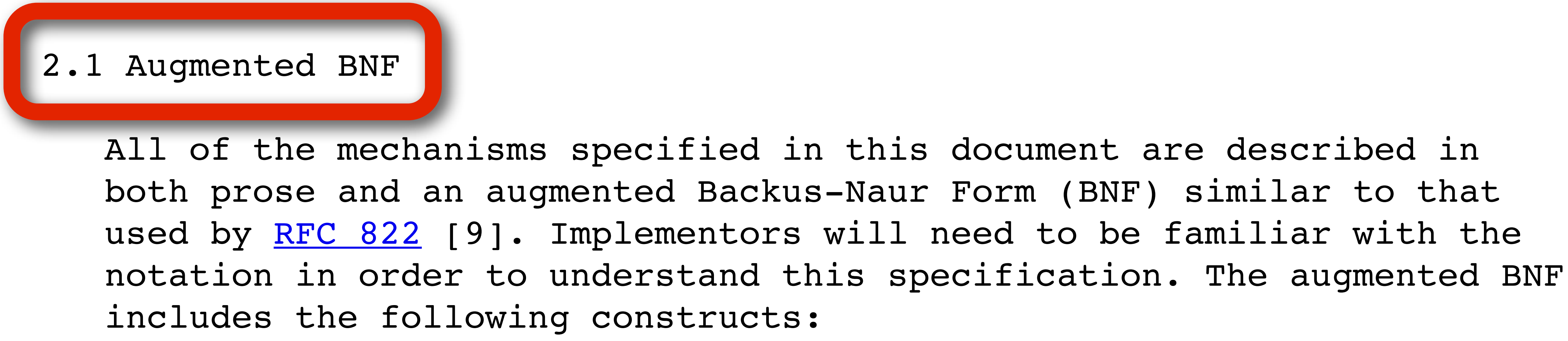
What about sub-Turing domain-specific languages?

Regex
Yacc
Datalog
SQL

Avoid halting problem.

WARNING

RFC 2616 (HTTP 1.1)



2.1 Augmented BNF

All of the mechanisms specified in this document are described in both prose and an augmented Backus-Naur Form (BNF) similar to that used by RFC 822 [9]. Implementors will need to be familiar with the notation in order to understand this specification. The augmented BNF includes the following constructs:

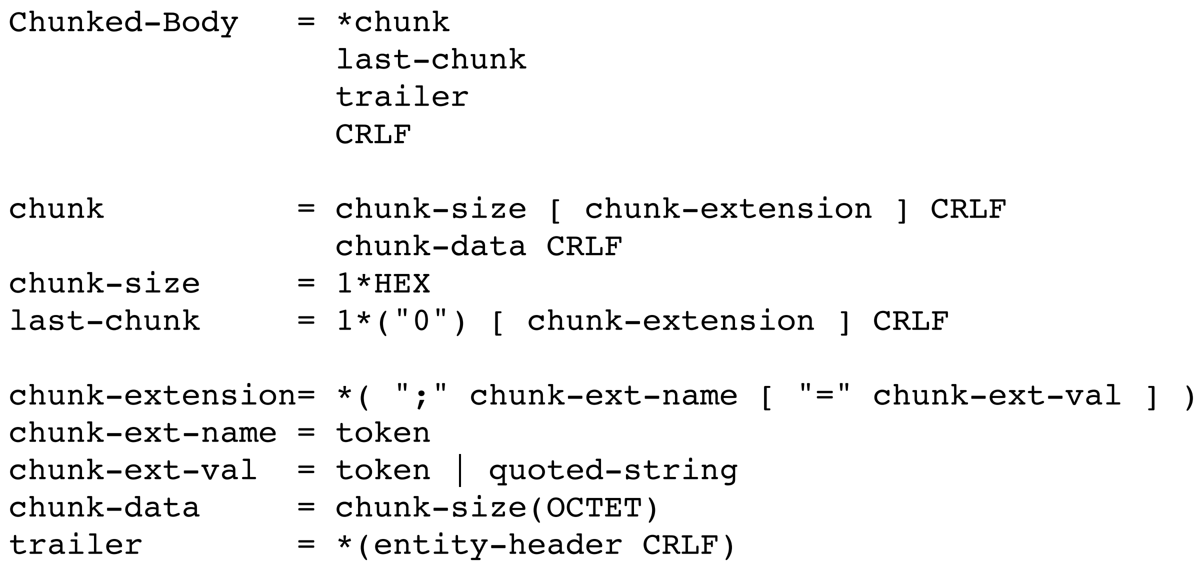
```
media-type   = type "/" subtype *( ";" parameter )
type         = token
subtype      = token
```

```
HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
```

```
LWS = [CRLF] 1*( SP | HT )
```

```
separators = "(" | ")" | "<" | ">" | "@"
           | "," | ";" | ":" | "\" | <">
           | "/" | "[" | "]" | "?" | "="
           | "{" | "}" | SP | HT
```

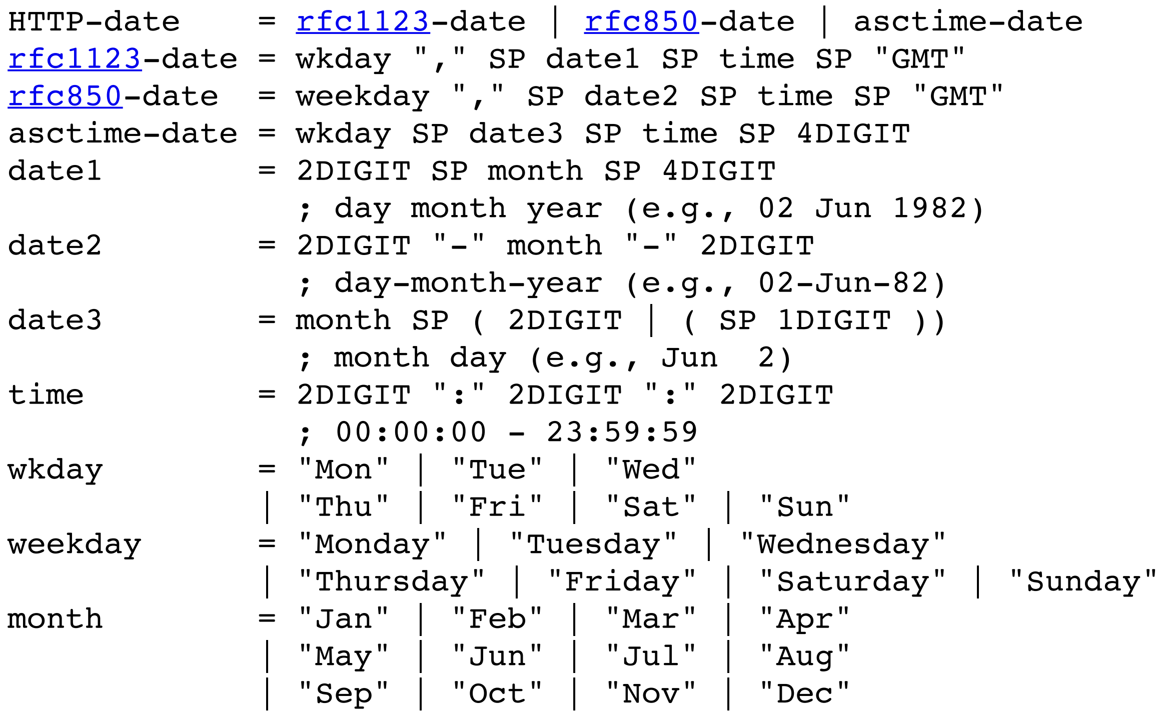
```
http_URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]
```



```
Chunked-Body   = *chunk
                 last-chunk
                 trailer
                 CRLF

chunk          = chunk-size [ chunk-extension ] CRLF
                 chunk-data CRLF
chunk-size     = 1*HEX
last-chunk     = 1*("0") [ chunk-extension ] CRLF

chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
chunk-ext-name = token
chunk-ext-val  = token | quoted-string
chunk-data     = chunk-size(OCTET)
trailer        = *(entity-header CRLF)
```



```
HTTP-date    = rfc1123-date | rfc850-date | asctime-date
rfc1123-date = wkday "," SP date1 SP time SP "GMT"
rfc850-date  = weekday "," SP date2 SP time SP "GMT"
asctime-date = wkday SP date3 SP time SP 4DIGIT
date1        = 2DIGIT SP month SP 4DIGIT
               ; day month year (e.g., 02 Jun 1982)
date2        = 2DIGIT "-" month "-" 2DIGIT
               ; day-month-year (e.g., 02-Jun-82)
date3        = month SP ( 2DIGIT | ( SP 1DIGIT ))
               ; month day (e.g., Jun  2)
time         = 2DIGIT ":" 2DIGIT ":" 2DIGIT
               ; 00:00:00 - 23:59:59
wkday        = "Mon" | "Tue" | "Wed"
             | "Thu" | "Fri" | "Sat" | "Sun"
weekday      = "Monday" | "Tuesday" | "Wednesday"
             | "Thursday" | "Friday" | "Saturday" | "Sunday"
month        = "Jan" | "Feb" | "Mar" | "Apr"
             | "May" | "Jun" | "Jul" | "Aug"
             | "Sep" | "Oct" | "Nov" | "Dec"
```

RFC 3501 (IMAPv4)

```
address          = "(" addr-name SP addr-adl SP addr-mailbox SP
                   addr-host ")"

addr-adl         = nstring
                     ; Holds route from [RFC-2822] route-addr if
                     ; non-NIL

addr-host        = nstring
                     ; NIL indicates [RFC-2822] group syntax.
                     ; Otherwise, holds [RFC-2822] domain name

addr-mailbox     = nstring
                     ; NIL indicates end of [RFC-2822] group; if
                     ; non-NIL and addr-host is NIL, holds
                     ; [RFC-2822] group name.
                     ; Otherwise, holds [RFC-2822] local-part
                     ; after removing [RFC-2822] quoting

addr-name        = nstring
                     ; If non-NIL, holds phrase from [RFC-2822]
                     ; mailbox after removing [RFC-2822] quoting

append           = "APPEND" SP mailbox [SP flag-list] [SP date-time] SP
                   literal

astring          = 1*ASTRING-CHAR / string

ASTRING-CHAR   = ATOM-CHAR / resp-specials

atom             = 1*ATOM-CHAR

ATOM-CHAR        = <any CHAR except atom-specials>

atom-specials    = "(" / ")" / "{" / SP / CTL / list-wildcards /
                   quoted-specials / resp-specials

authenticate     = "AUTHENTICATE" SP auth-type *(CRLF base64)

auth-type        = atom
                     ; Defined by [SASL]

base64           = *(4base64-char) [base64-terminal]

base64-char      = ALPHA / DIGIT / "+" / "/"
                     ; Case-sensitive

base64-terminal = (2base64-char "==") / (3base64-char "=")

body             = "(" (body-type-1part / body-type-mpart) ")"

body-extension  = nstring / number /
                    "(" body-extension *(SP body-extension) ")"
                     ; Future expansion.  Client implementations
                     ; MUST accept body-extension fields.  Server
                     ; implementations MUST NOT generate
                     ; body-extension fields except as defined by
                     ; future standard or standards-track
                     ; revisions of this specification.

body-ext-1part  = body-fld-md5 [SP body-fld-dsp [SP body-fld-lang
                  [SP body-fld-loc *(SP body-extension)]]]
                    ; MUST NOT be returned on non-extensible
                    ; "BODY" fetch

body-ext-mpart  = body-fld-param [SP body-fld-dsp [SP body-fld-lang
                  [SP body-fld-loc *(SP body-extension)]]]
                    ; MUST NOT be returned on non-extensible
                    ; "BODY" fetch

body-fields     = body-fld-param SP body-fld-id SP body-fld-desc SP
                  body-fld-enc SP body-fld-octets

body-fld-desc   = nstring

body-fld-dsp    = "(" string SP body-fld-param ")" / nil

body-fld-enc    = (DQUOTE ("7BIT" / "8BIT" / "BINARY" / "BASE64"/
                  "QUOTED-PRINTABLE") DQUOTE) / string

body-fld-id     = nstring

body-fld-lang   = nstring / "(" string *(SP string) ")"

body-fld-loc    = nstring

body-fld-lines  = number

body-fld-md5    = nstring

body-fld-octets = number

body-fld-param  = "(" string SP string *(SP string SP string) ")" / nil

body-type-1part = (body-type-basic / body-type-msg / body-type-text)
                  [SP body-ext-1part]

body-type-basic = media-basic SP body-fields
                    ; MESSAGE subtype MUST NOT be "RFC822"

body-type-mpart = 1*body SP media-subtype
                  [SP body-ext-mpart]

body-type-msg   = media-message SP body-fields SP envelope
                  SP body SP body-fld-lines

body-type-text  = media-text SP body-fields SP body-fld-lines

capability      = ("AUTH=" auth-type) / atom
                    ; New capabilities MUST begin with "X" or be
                    ; registered with IANA as standard or
                    ; standards-track

capability-data = "CAPABILITY" *(SP capability) SP "IMAP4rev1"
                  *(SP capability)
                    ; Servers MUST implement the STARTTLS, AUTH=PLAIN,
                    ; and LOGINDISABLED capabilities
                    ; Servers which offer RFC 1730 compatibility MUST
                    ; list "IMAP4" as the first capability.

CHAR8           = %x01-ff
                    ; any OCTET except NUL, %x00

command         = tag SP (command-any / command-auth / command-nonauth /
                  command-select) CRLF
                    ; Modal based on state

command-any     = "CAPABILITY" / "LOGOUT" / "NOOP" / x-command
                    ; Valid in all states

command-auth    = append / create / delete / examine / list / lsub /
                  rename / select / status / subscribe / unsubscribe
                    ; Valid only in Authenticated or Selected state

command-nonauth = login / authenticate / "STARTTLS"
                    ; Valid only when in Not Authenticated state

command-select  = "CHECK" / "CLOSE" / "EXPUNGE" / copy / fetch / store /
                  uid / search
                    ; Valid only when in Selected state

continue-req    = "+" SP (resp-text / base64) CRLF

copy            = "COPY" SP sequence-set SP mailbox

create          = "CREATE" SP mailbox
                    ; Use of INBOX gives a NO error

date            = date-text / DQUOTE date-text DQUOTE

date-day        = 1*2DIGIT
                    ; Day of month

date-day-fixed  = (SP DIGIT) / 2DIGIT
                    ; Fixed-format version of date-day

date-month      = "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" /
                  "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"

date-text       = date-day "-" date-month "-" date-year

date-year       = 4DIGIT

date-time       = DQUOTE date-day-fixed "-" date-month "-" date-year
                  SP time SP zone DQUOTE

delete          = "DELETE" SP mailbox
                    ; Use of INBOX gives a NO error

digit-nz        = %x31-39
                    ; 1-9

envelope        = "(" env-date SP env-subject SP env-from SP
                  env-sender SP env-reply-to SP env-to SP env-cc SP
                  env-bcc SP env-in-reply-to SP env-message-id ")"

env-bcc         = "(" 1*address ")" / nil

env-cc          = "(" 1*address ")" / nil

env-date        = nstring

env-from        = "(" 1*address ")" / nil

env-in-reply-to = nstring

env-message-id  = nstring

env-reply-to    = "(" 1*address ")" / nil

env-sender      = "(" 1*address ")" / nil

env-subject     = nstring

env-to          = "(" 1*address ")" / nil

examine         = "EXAMINE" SP mailbox

fetch           = "FETCH" SP sequence-set SP ("ALL" / "FULL" / "FAST" /
                  fetch-att / "(" fetch-att *(SP fetch-att) ")")

fetch-att       = "ENVELOPE" / "FLAGS" / "INTERNALDATE" /
                  "RFC822" [".HEADER" / ".SIZE" / ".TEXT"] /
                  "BODY" ["STRUCTURE"] / "UID" /
                  "BODY" section ["<" number "." nz-number ">"] /
                  "BODY.PEEK" section ["<" number "." nz-number ">"]

flag            = "\Answered" / "\Flagged" / "\Deleted" /
                  "\Seen" / "\Draft" / flag-keyword / flag-extension
                    ; Does not include "\Recent"

flag-extension  = "\" atom
                    ; Future expansion.  Client implementations
                    ; MUST accept flag-extension flags.  Server
                    ; implementations MUST NOT generate
                    ; flag-extension flags except as defined by
                    ; future standard or standards-track
                    ; revisions of this specification.

flag-fetch      = flag / "\Recent"

flag-keyword    = atom

flag-list       = "(" [flag *(SP flag)] ")"

flag-perm       = flag / "\*"

greeting        = "*" SP (resp-cond-auth / resp-cond-bye) CRLF

header-fld-name = astring

header-list     = "(" header-fld-name *(SP header-fld-name) ")"

list            = "LIST" SP mailbox SP list-mailbox

list-mailbox    = 1*list-char / string

list-char       = ATOM-CHAR / list-wildcards / resp-specials

list-wildcards  = "%" / "*"

literal         = "{" number "}" CRLF *CHAR8
                    ; Number represents the number of CHAR8s

login           = "LOGIN" SP userid SP password

lsub            = "LSUB" SP mailbox SP list-mailbox

mailbox         = "INBOX" / astring
                    ; INBOX is case-insensitive.  All case variants of
                    ; INBOX (e.g., "iNbOx") MUST be interpreted as INBOX
                    ; not as an astring.  An astring which consists of
                    ; the case-insensitive sequence "I" "N" "B" "O" "X"
                    ; is considered to be INBOX and not an astring.
                    ;  Refer to section 5.1 for further
                    ; semantic details of mailbox names.

mailbox-data    =  "FLAGS" SP flag-list / "LIST" SP mailbox-list /
                   "LSUB" SP mailbox-list / "SEARCH" *(SP nz-number) /
                   "STATUS" SP mailbox SP "(" [status-att-list] ")" /
                   number SP "EXISTS" / number SP "RECENT"

mailbox-list    = "(" [mbx-list-flags] ")" SP
                   (DQUOTE QUOTED-CHAR DQUOTE / nil) SP mailbox

mbx-list-flags  = *(mbx-list-oflag SP) mbx-list-sflag
                  *(SP mbx-list-oflag) /
                  mbx-list-oflag *(SP mbx-list-oflag)

mbx-list-oflag  = "\Noinferiors" / flag-extension
                    ; Other flags; multiple possible per LIST response

mbx-list-sflag  = "\Noselect" / "\Marked" / "\Unmarked"
                    ; Selectability flags; only one per LIST response

media-basic     = ((DQUOTE ("APPLICATION" / "AUDIO" / "IMAGE" /
                  "MESSAGE" / "VIDEO") DQUOTE) / string) SP
                  media-subtype
                    ; Defined in [MIME-IMT]

media-message   = DQUOTE "MESSAGE" DQUOTE SP DQUOTE "RFC822" DQUOTE
                    ; Defined in [MIME-IMT]

media-subtype   = string
                    ; Defined in [MIME-IMT]

                    ; The Universal Time zone is "+0000".

media-text      = DQUOTE "TEXT" DQUOTE SP media-subtype
                    ; Defined in [MIME-IMT]

message-data    = nz-number SP ("EXPUNGE" / ("FETCH" SP msg-att))

msg-att         = "(" (msg-att-dynamic / msg-att-static)
                   *(SP (msg-att-dynamic / msg-att-static)) ")"

msg-att-dynamic = "FLAGS" SP "(" [flag-fetch *(SP flag-fetch)] ")"
                    ; MAY change for a message

msg-att-static  = "ENVELOPE" SP envelope / "INTERNALDATE" SP date-time /
                  "RFC822" [".HEADER" / ".TEXT"] SP nstring /
                  "RFC822.SIZE" SP number /
                  "BODY" ["STRUCTURE"] SP body /
                  "BODY" section ["<" number ">"] SP nstring /
                  "UID" SP uniqueid
                    ; MUST NOT change for a message

nil             = "NIL"

nstring         = string / nil

number          = 1*DIGIT
                    ; Unsigned 32-bit integer
                    ; (0 <= n < 4,294,967,296)

nz-number       = digit-nz *DIGIT
                    ; Non-zero unsigned 32-bit integer
                    ; (0 < n < 4,294,967,296)

password        = astring

quoted          = DQUOTE *QUOTED-CHAR DQUOTE

QUOTED-CHAR     = <any TEXT-CHAR except quoted-specials> /
                  "\" quoted-specials

quoted-specials = DQUOTE / "\"

rename          = "RENAME" SP mailbox SP mailbox
                    ; Use of INBOX as a destination gives a NO error

response        = *(continue-req / response-data) response-done

response-data   = "*" SP (resp-cond-state / resp-cond-bye /
                  mailbox-data / message-data / capability-data) CRLF

response-done   = response-tagged / response-fatal

response-fatal  = "*" SP resp-cond-bye CRLF
                    ; Server closes connection immediately

response-tagged = tag SP resp-cond-state CRLF

resp-cond-auth  = ("OK" / "PREAUTH") SP resp-text
                    ; Authentication condition

resp-cond-bye   = "BYE" SP resp-text

resp-cond-state = ("OK" / "NO" / "BAD") SP resp-text
                    ; Status condition

resp-specials   = "]"

resp-text       = ["[" resp-text-code "]" SP] text

resp-text-code  = "ALERT" /
                  "BADCHARSET" [SP "(" astring *(SP astring) ")" ] /
                  capability-data / "PARSE" /
                  "PERMANENTFLAGS" SP "("
                  [flag-perm *(SP flag-perm)] ")" /
                  "READ-ONLY" / "READ-WRITE" / "TRYCREATE" /
                  "UIDNEXT" SP nz-number / "UIDVALIDITY" SP nz-number /
                  "UNSEEN" SP nz-number /
                  atom [SP 1*<any TEXT-CHAR except "]">]

search          = "SEARCH" [SP "CHARSET" SP astring] 1*(SP search-key)
                    ; CHARSET argument to MUST be registered with IANA

search-key      = "ALL" / "ANSWERED" / "BCC" SP astring /
                  "BEFORE" SP date / "BODY" SP astring /
                  "CC" SP astring / "DELETED" / "FLAGGED" /
                  "FROM" SP astring / "KEYWORD" SP flag-keyword /
                  "NEW" / "OLD" / "ON" SP date / "RECENT" / "SEEN" /
                  "SINCE" SP date / "SUBJECT" SP astring /
                  "TEXT" SP astring / "TO" SP astring /
                  "UNANSWERED" / "UNDELETED" / "UNFLAGGED" /
                  "UNKEYWORD" SP flag-keyword / "UNSEEN" /
                    ; Above this line were in [IMAP2]
                  "DRAFT" / "HEADER" SP header-fld-name SP astring /
                  "LARGER" SP number / "NOT" SP search-key /
                  "OR" SP search-key SP search-key /
                  "SENTBEFORE" SP date / "SENTON" SP date /
                  "SENTSINCE" SP date / "SMALLER" SP number /
                  "UID" SP sequence-set / "UNDRAFT" / sequence-set /
                  "(" search-key *(SP search-key) ")"

section         = "[" [section-spec] "]"

section-msgtext = "HEADER" / "HEADER.FIELDS" [".NOT"] SP header-list /
                  "TEXT"
                    ; top-level or MESSAGE/RFC822 part

section-part    = nz-number *("." nz-number)
                    ; body part nesting

section-spec    = section-msgtext / (section-part ["." section-text])

section-text    = section-msgtext / "MIME"
                    ; text other than actual body part (headers, etc.)

select          = "SELECT" SP mailbox

seq-number      = nz-number / "*"
                    ; message sequence number (COPY, FETCH, STORE
                    ; commands) or unique identifier (UID COPY,
                    ; UID FETCH, UID STORE commands).
                    ; * represents the largest number in use.  In
                    ; the case of message sequence numbers, it is
                    ; the number of messages in a non-empty mailbox.
                    ; In the case of unique identifiers, it is the
                    ; unique identifier of the last message in the
                    ; mailbox or, if the mailbox is empty, the
                    ; mailbox's current UIDNEXT value.
                    ; The server should respond with a tagged BAD
                    ; response to a command that uses a message
                    ; sequence number greater than the number of
                    ; messages in the selected mailbox.  This
                    ; includes "*" if the selected mailbox is empty.

seq-range       = seq-number ":" seq-number
                    ; two seq-number values and all values between
                    ; these two regardless of order.
                    ; Example: 2:4 and 4:2 are equivalent and indicate
                    ; values 2, 3, and 4.
                    ; Example: a unique identifier sequence range of
                    ; 3291:* includes the UID of the last message in
                    ; the mailbox, even if that value is less than 3291.

sequence-set    = (seq-number / seq-range) *("," sequence-set)
                    ; set of seq-number values, regardless of order.
                    ; Servers MAY coalesce overlaps and/or execute the
                    ; sequence in any order.
                    ; Example: a message sequence number set of
                    ; 2,4:7,9,12:* for a mailbox with 15 messages is
                    ; equivalent to 2,4,5,6,7,9,12,13,14,15
                    ; Example: a message sequence number set of *:4,5:7
                    ; for a mailbox with 10 messages is equivalent to
                    ; 10,9,8,7,6,5,4,5,6,7 and MAY be reordered and
                    ; overlap coalesced to be 4,5,6,7,8,9,10.

status          = "STATUS" SP mailbox SP
                  "(" status-att *(SP status-att) ")"

status-att      = "MESSAGES" / "RECENT" / "UIDNEXT" / "UIDVALIDITY" /
                  "UNSEEN"

status-att-list =  status-att SP number *(SP status-att SP number)

store           = "STORE" SP sequence-set SP store-att-flags

store-att-flags = (["+" / "-"] "FLAGS" [".SILENT"]) SP
                  (flag-list / (flag *(SP flag)))

string          = quoted / literal

subscribe       = "SUBSCRIBE" SP mailbox

tag             = 1*<any ASTRING-CHAR except "+">

text            = 1*TEXT-CHAR

TEXT-CHAR       = <any CHAR except CR and LF>

time            = 2DIGIT ":" 2DIGIT ":" 2DIGIT
                    ; Hours minutes seconds

uid             = "UID" SP (copy / fetch / search / store)
                    ; Unique identifiers used instead of message
                    ; sequence numbers

uniqueid        = nz-number
                    ; Strictly ascending

unsubscribe     = "UNSUBSCRIBE" SP mailbox

userid          = astring

x-command       = "X" atom <experimental command arguments>

zone            = ("+" / "-") 4DIGIT
                    ; Signed four-digit value of hhmm representing
                    ; hours and minutes east of Greenwich (that is,
                    ; the amount that the given time differs from
                    ; Universal Time).  Subtracting the timezone
                    ; from the given time will give the UT form.
```

RFC 2812 (IRC)

The Augmented BNF representation for this is:

```
message    =  [ ":" prefix SPACE ] command [ params ] crlf
prefix     =  servername / ( nickname [ [ "!" user ] "@" host ] )
command    =  1*letter / 3digit
params     =  *14( SPACE middle ) [ SPACE ":" trailing ]
           =/ 14( SPACE middle ) [ SPACE [ ":" ] trailing ]

nospcrlfcl =  %x01-09 / %x0B-0C / %x0E-1F / %x21-39 / %x3B-FF
                ; any octet except NUL, CR, LF, " " and ":"
middle     =  nospcrlfcl *( ":" / nospcrlfcl )
trailing   =  *( ":" / " " / nospcrlfcl )

SPACE      =  %x20        ; space character
crlf       =  %x0D %x0A   ; "carriage return" "linefeed"

target     =  nickname / server
msgtarget  =  msgto *( "," msgto )
msgto      =  channel / ( user [ "%" host ] "@" servername )
msgto      =/ ( user "%" host ) / targetmask
msgto      =/ nickname / ( nickname "!" user "@" host )
channel    =  ( "#" / "+" / ( "!" channelid ) / "&" ) chanstring
              [ ":" chanstring ]
servername =  hostname
host       =  hostname / hostaddr
hostname   =  shortname *( "." shortname )
shortname  =  ( letter / digit ) *( letter / digit / "-" )
              *( letter / digit )
                ; as specified in RFC 1123 [HNAME]
hostaddr   =  ip4addr / ip6addr
ip4addr    =  1*3digit "." 1*3digit "." 1*3digit "." 1*3digit
ip6addr    =  1*hexdigit 7( ":" 1*hexdigit )
ip6addr    =/ "0:0:0:0:0:" ( "0" / "FFFF" ) ":" ip4addr
nickname   =  ( letter / special ) *8( letter / digit / special / "-" )
targetmask =  ( "$" / "#" ) mask
                ; see details on allowed masks in section 3.3.1
chanstring =  %x01-07 / %x08-09 / %x0B-0C / %x0E-1F / %x21-2B
chanstring =/ %x2D-39 / %x3B-FF
                ; any octet except NUL, BELL, CR, LF, " ", "," and ":"
channelid  = 5( %x41-5A / digit )   ; 5( A-Z / 0-9 )


user       =  1*( %x01-09 / %x0B-0C / %x0E-1F / %x21-3F / %x41-FF )
                ; any octet except NUL, CR, LF, " " and "@"
key        =  1*23( %x01-05 / %x07-08 / %x0C / %x0E-1F / %x21-7F )
                ; any 7-bit US_ASCII character,
                ; except NUL, CR, LF, FF, h/v TABs, and " "
letter     =  %x41-5A / %x61-7A       ; A-Z / a-z
digit      =  %x30-39                 ; 0-9
hexdigit   =  digit / "A" / "B" / "C" / "D" / "E" / "F"
special    =  %x5B-60 / %x7B-7D
                 ; "[", "]", "\", "`", "_", "^", "{", "|", "}"
```

Efficient parsing techniques exist.

LALR(k)

Earley

LL(k)

Operator precedence

GLR

SLR

CYK

LR(k)

Combinators

PEG

packrat

Parsing tools abound.

Yacc

Parsec

ANTLR

NLTK

Happy

Flex

Bison

CUPS

Ragel

PLY

State of the art?

```
*buf++
```

Apache

2,179 lines of C

lighttpd

1,211 lines of C

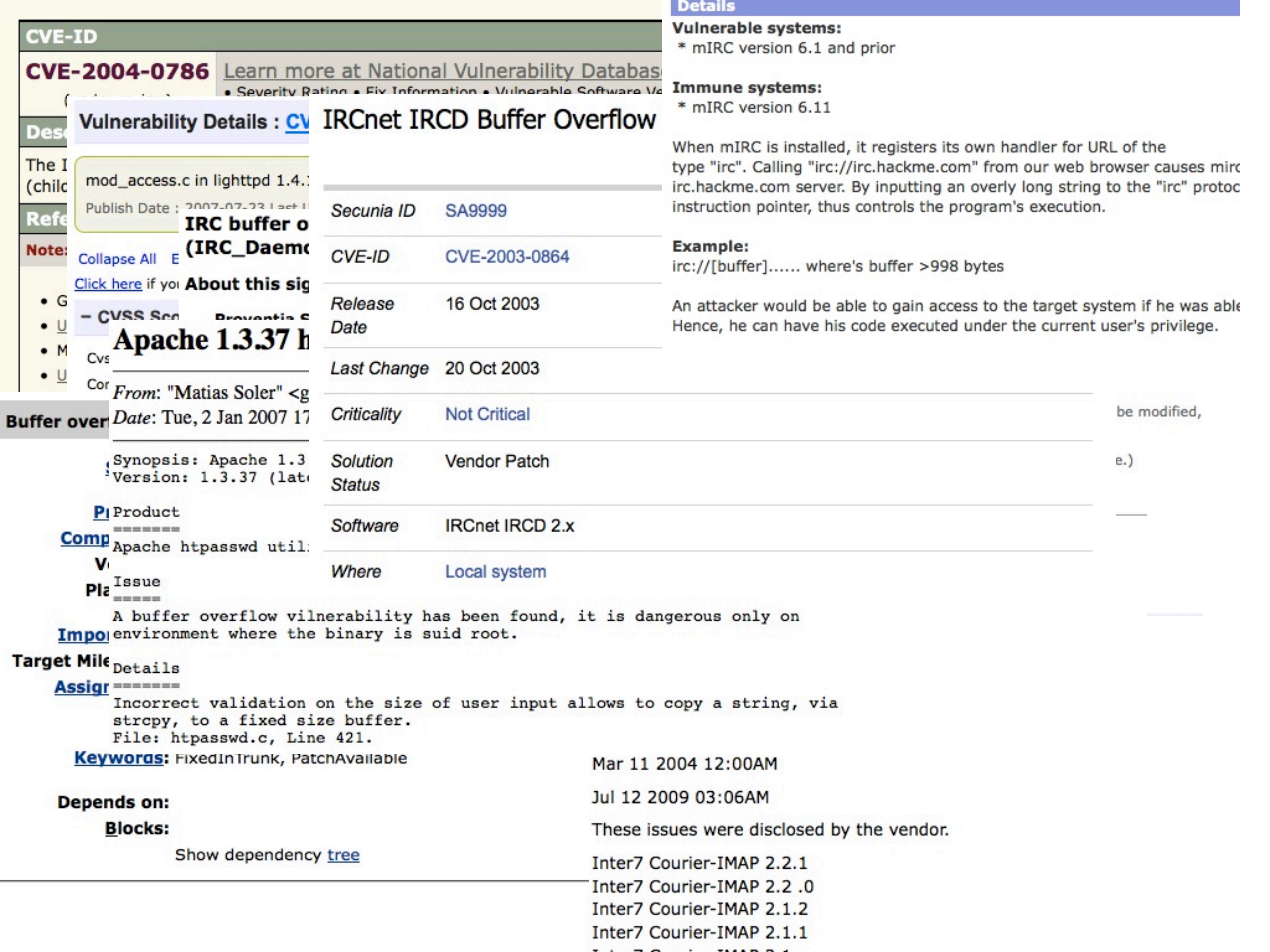
freenode IRCD

> 2000 lines of C

Courier IMAP

2,633 lines of C

Result?



CVE-ID

CVE-2004-0786 Learn more at National Vulnerability Databas

• Severity Rating • Fix Information • Vulnerable Software Ve

Des

The I
(chil

Refe

Note:

• G
• U
• M
• U

Vulnerability Details : CV

mod_access.c in lighttpd 1.4.

Publish Date : 2007-07-23 Last U

Collapse All E

Click here if yo

– CVSS Sco

Cvs

Co

IRC buffer o
(IRC_Daemo

About this sig

Proventia S

Apache 1.3.37 h

From: "Matias Soler" <g

Date: Tue, 2 Jan 2007 17

```
Synopsis: Apache 1.3
Version: 1.3.37 (lat

Product
=======
Apache htpasswd util

Issue
=====
A buffer overflow vilnerability has been found, it is dangerous only on
environment where the binary is suid root.

Details
=======
Incorrect validation on the size of user input allows to copy a string, via
strcpy, to a fixed size buffer.
File: htpasswd.c, Line 421.
```

Buffer over

Pr

Comp

V

Pla

Impo

Target Mile

Assig

Keywords: FixedInTrunk, PatchAvailable

Depends on:

Blocks:

Show dependency tree

IRCnet IRCD Buffer Overflow

Secunia ID	SA9999
CVE-ID	CVE-2003-0864
Release Date	16 Oct 2003
Last Change	20 Oct 2003
Criticality	Not Critical
Solution Status	Vendor Patch
Software	IRCnet IRCD 2.x
Where	Local system

Details

Vulnerable systems:
* mIRC version 6.1 and prior

Immune systems:
* mIRC version 6.11

When mIRC is installed, it registers its own handler for URL of the type "irc". Calling "irc://irc.hackme.com" from our web browser causes mirc irc.hackme.com server. By inputting an overly long string to the "irc" protoc instruction pointer, thus controls the program's execution.

Example:
irc://[buffer]...... where's buffer >998 bytes

An attacker would be able to gain access to the target system if he was able Hence, he can have his code executed under the current user's privilege.

be modified,

e.)

Mar 11 2004 12:00AM

Jul 12 2009 03:06AM

These issues were disclosed by the vendor.

Inter7 Courier-IMAP 2.2.1
Inter7 Courier-IMAP 2.2 .0
Inter7 Courier-IMAP 2.1.2
Inter7 Courier-IMAP 2.1.1
Inter7 Courier-IMAP 2.1

Why!?

Yacc blocks on `read()`.

Yacc needs continuations.

The continuation of a parser is its derivative.

For more, google:
“Yacc is Dead”

The future is...

The future is...

...safe, correct

The future is...

...safe, correct

...domain-specific

The future is...

...safe, correct

...domain-specific

...deep analysis

Thanks!

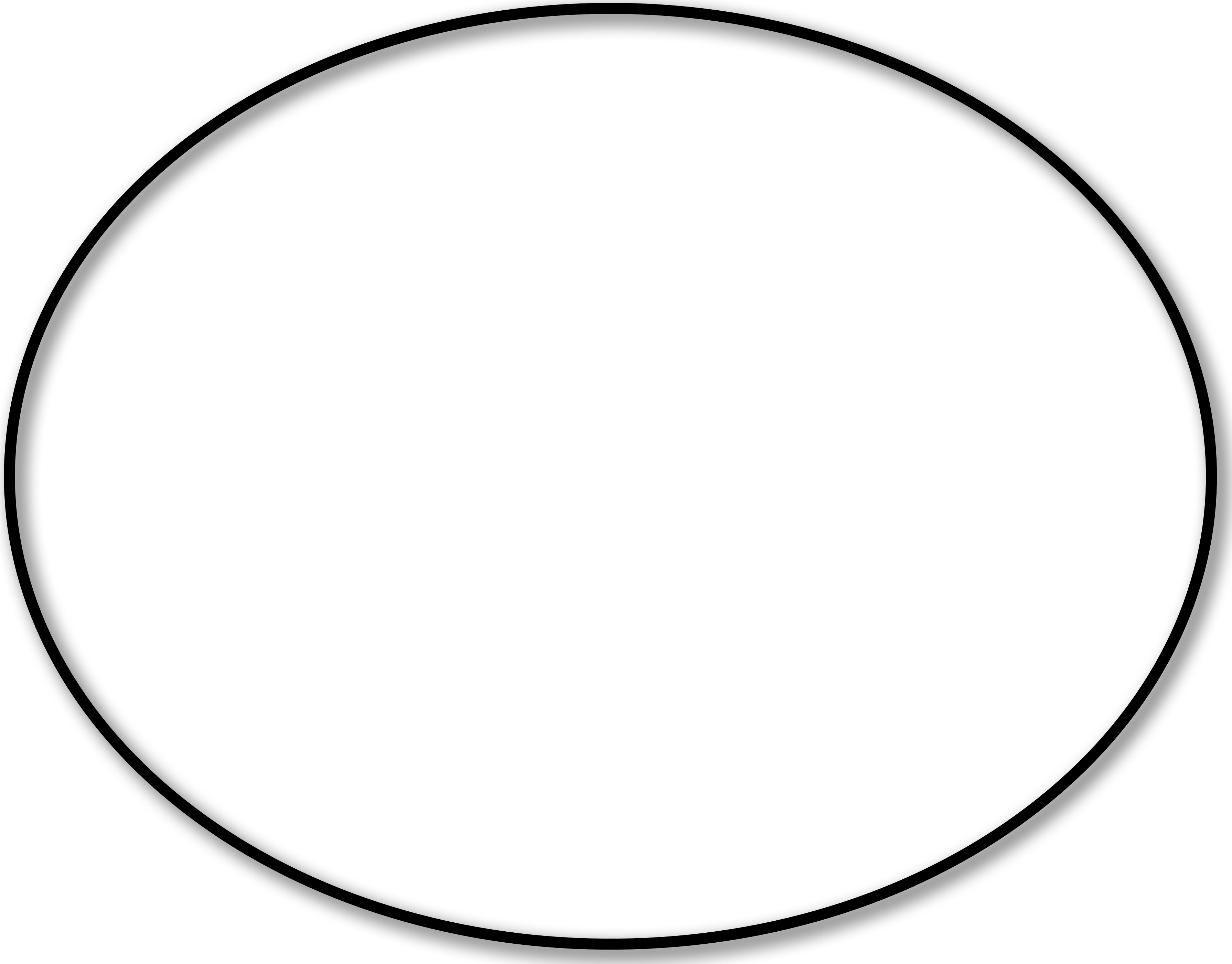
matt.might.net

@mattmight

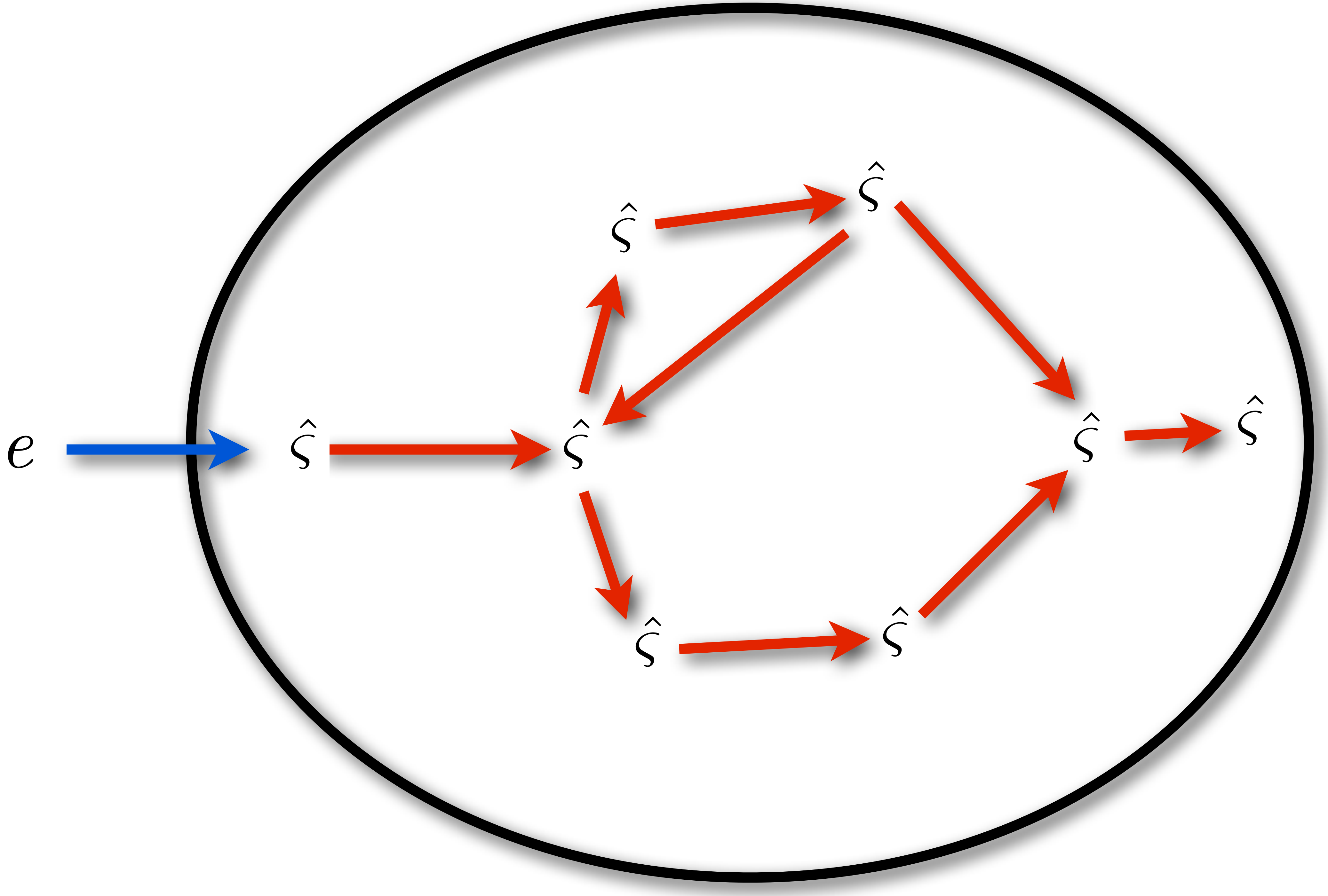
- POPL 2006: Analysis of environments & stacks
- ICFP 2006: Abstract garbage collection
- PLDI 2006: Enabling coroutine fusion
- POPL 2007: Logic-flow analysis (for arrays)
- PLDI 2010: Featherweight Java analysis
- ICFP 2010: Deriving small-step analyzers
- SFP 2010: Pushdown small-step analysis
- POPL 2011: Small-step analysis on the GPU

Application:
Dependence analysis

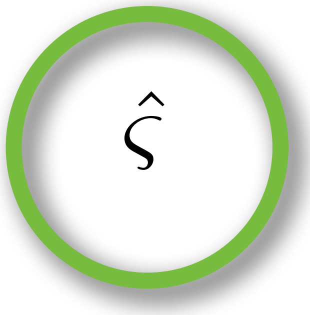
Dependence analysis



Dependence analysis

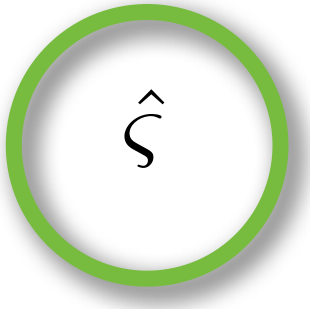


Dependence analysis



Dependence analysis

What resources are written?

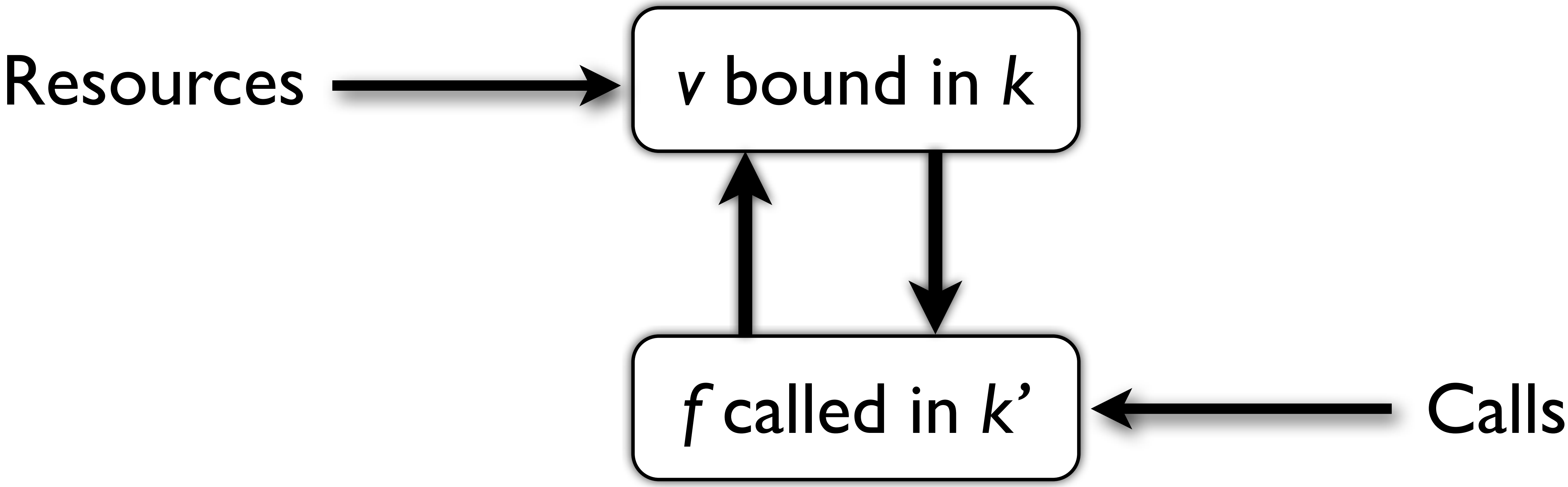


What resources are read?

Which calling contexts are live on stack?

Context-sensitive dependence graphs

Context-sensitive dependence graphs



```
f()

g()
```

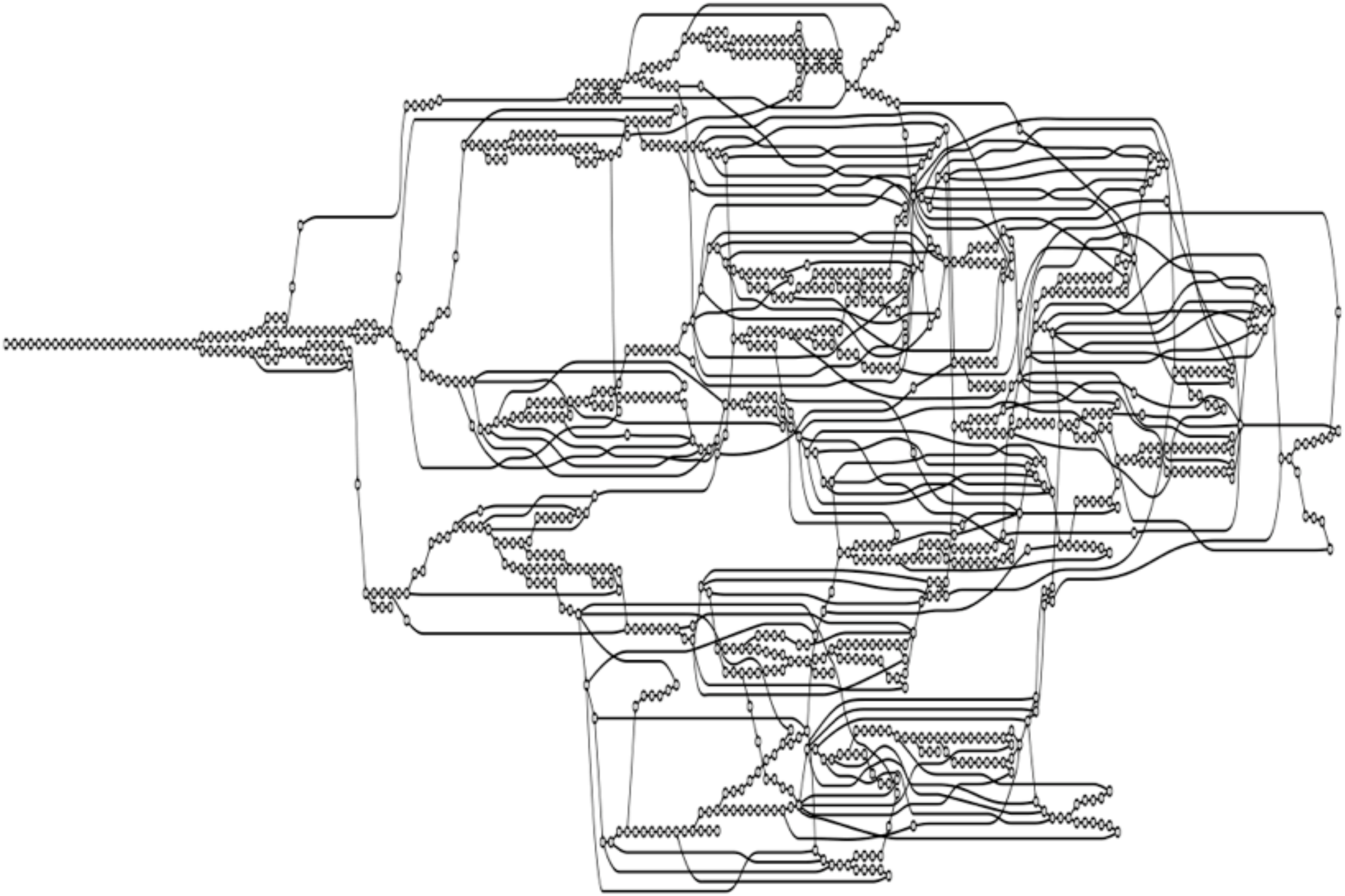
```
f()  ||  g()
```

Advanced technique: Abstract garbage collection

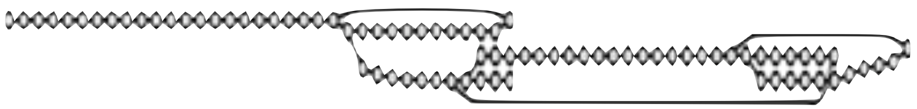
Abstract objects can die too.

Effects of abstract GC

Effects of abstract GC



Effects of abstract GC

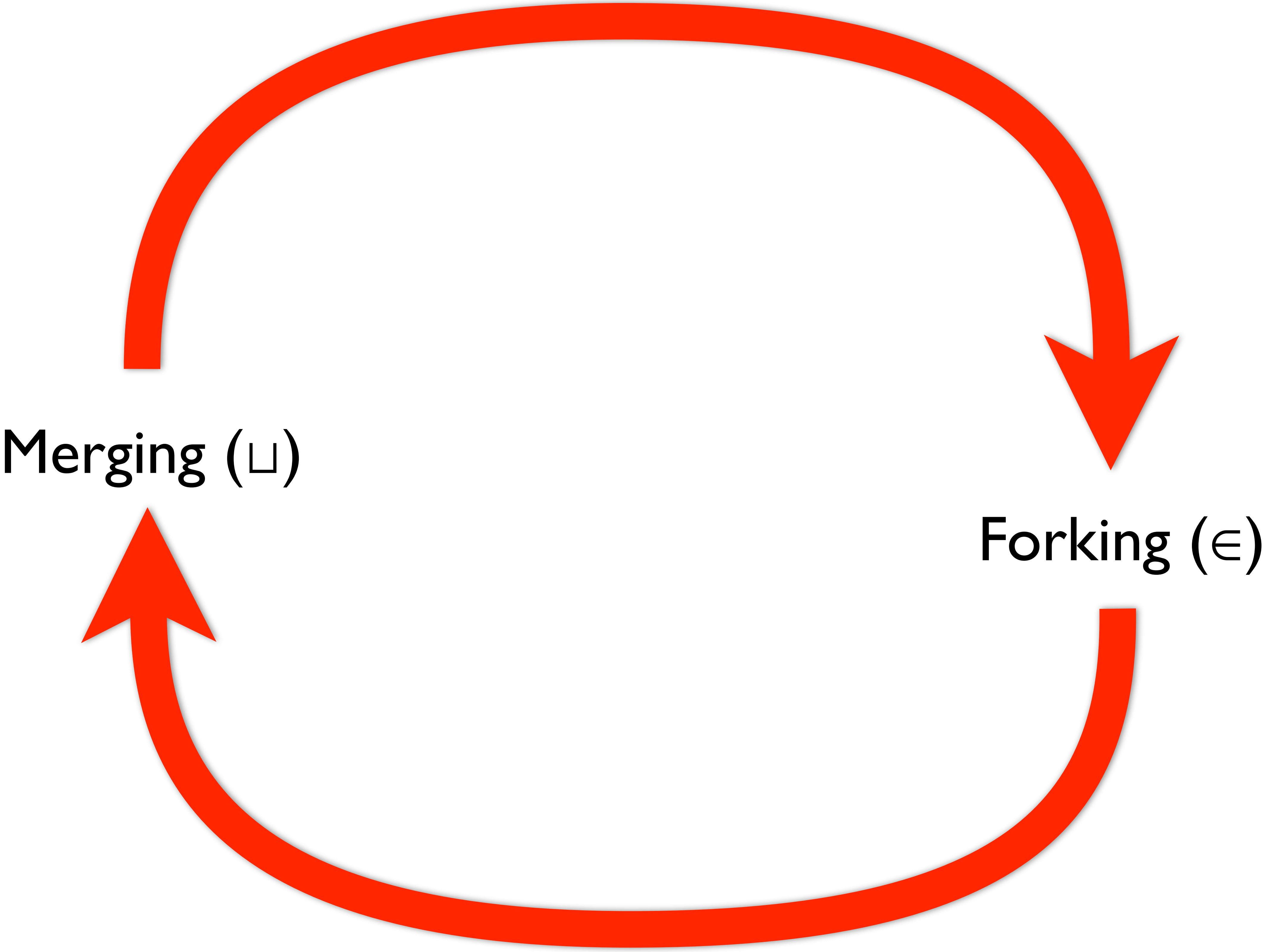


Vicious cyle

Merging ($\sqcup$)

Forking ($\in$)

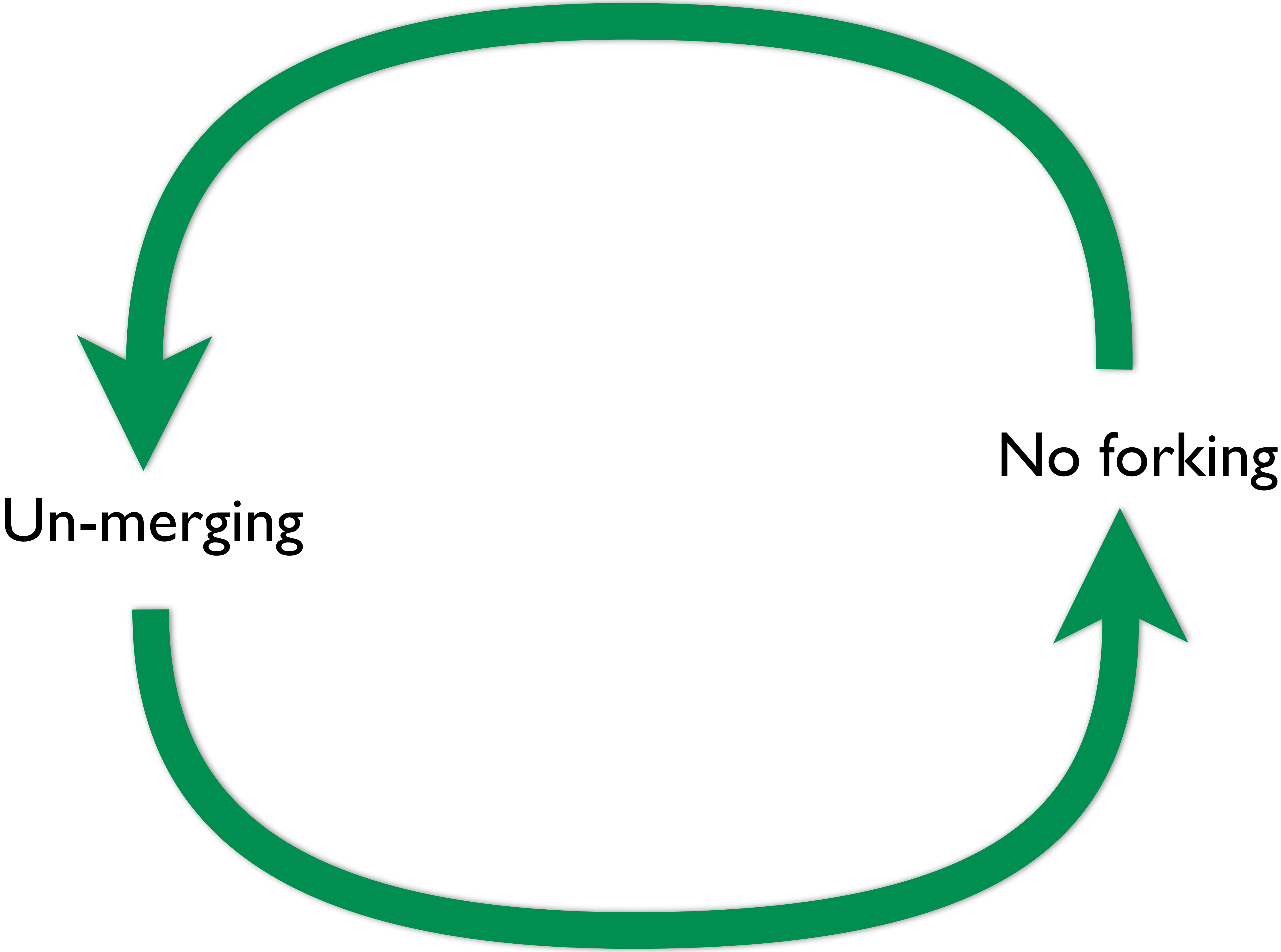
Vicious cyle



Merging ($\sqcup$)

Forking ($\in$)

Virtuous cycle



No forking

Un-merging

Orders of magnitude