Introspective Pushdown Analysis

Chris Earl
University of Utah
cwearl.com

Ilya Sergey
KU Leuven
@ilyasergey

Matt Might
University of Utah
matt.might.net
@mattmight

Dave Van Horn
Northeastern University
lambda-calcul.us
@lambda_calculus

“grad-school Vietnam”

“charred remains”

“would-be Ph.D.s”

-Olin Shivers

“academic War on Terror”

“unwinnable”

“never-ending”

-Me, to my grad students

What is flow analysis?

What is flow analysis?

What is wrong with it?

What is flow analysis?

What is wrong with it?

How do we fix it?

How do we fix the fixes?

We use fixed points.

What is flow analysis?

What is control-flow analysis?

```
(f x)
```

What is `f`?

Why not run the program?

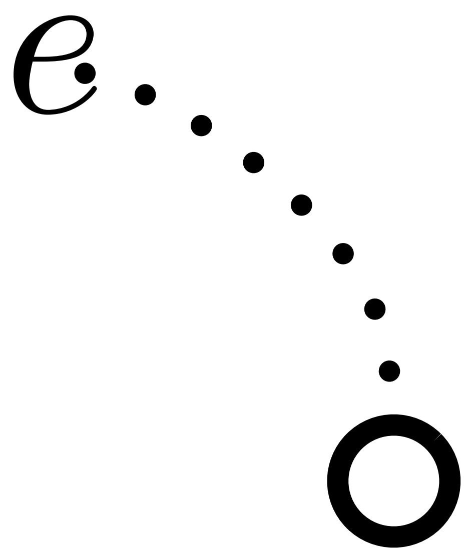
CEK

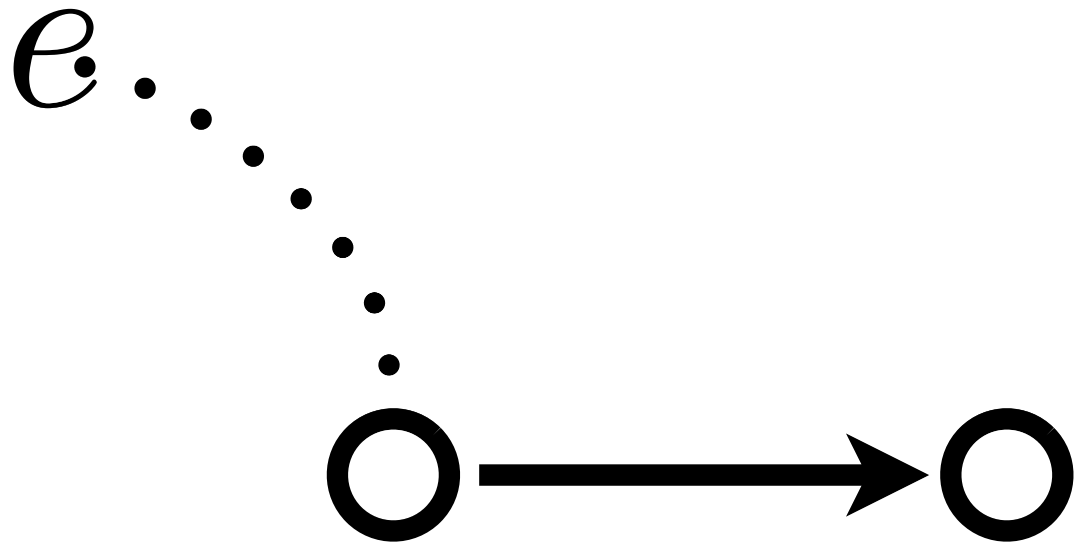
CEK

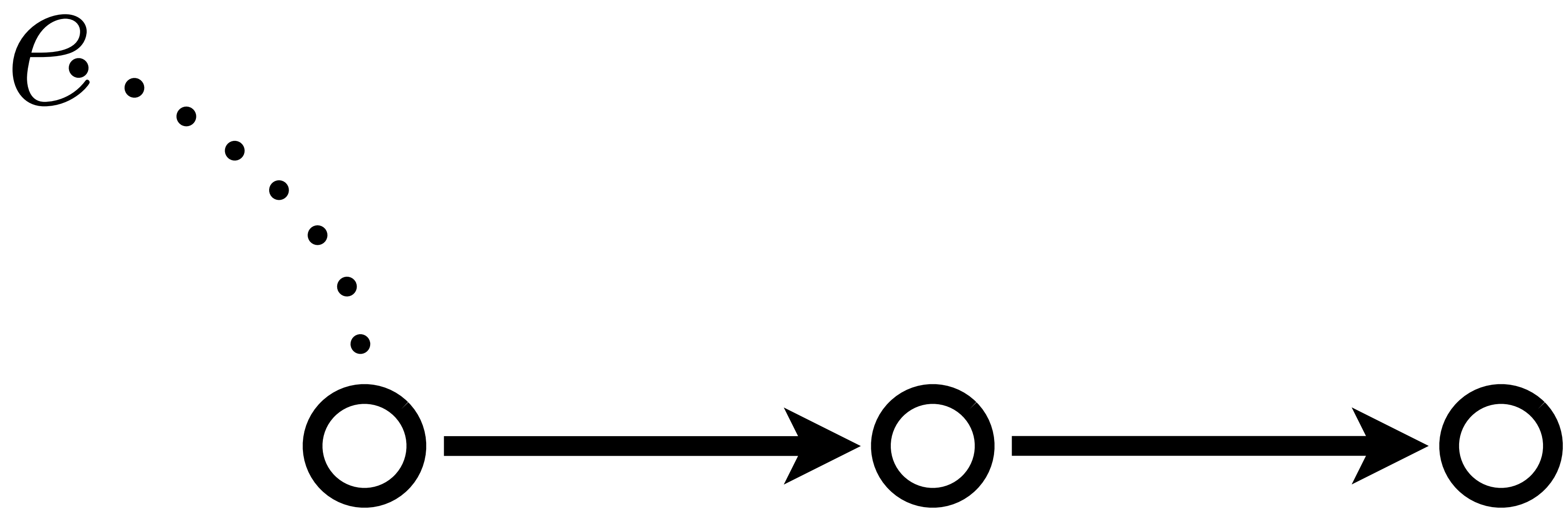
(Felleisen and Friedman, 1986)

e

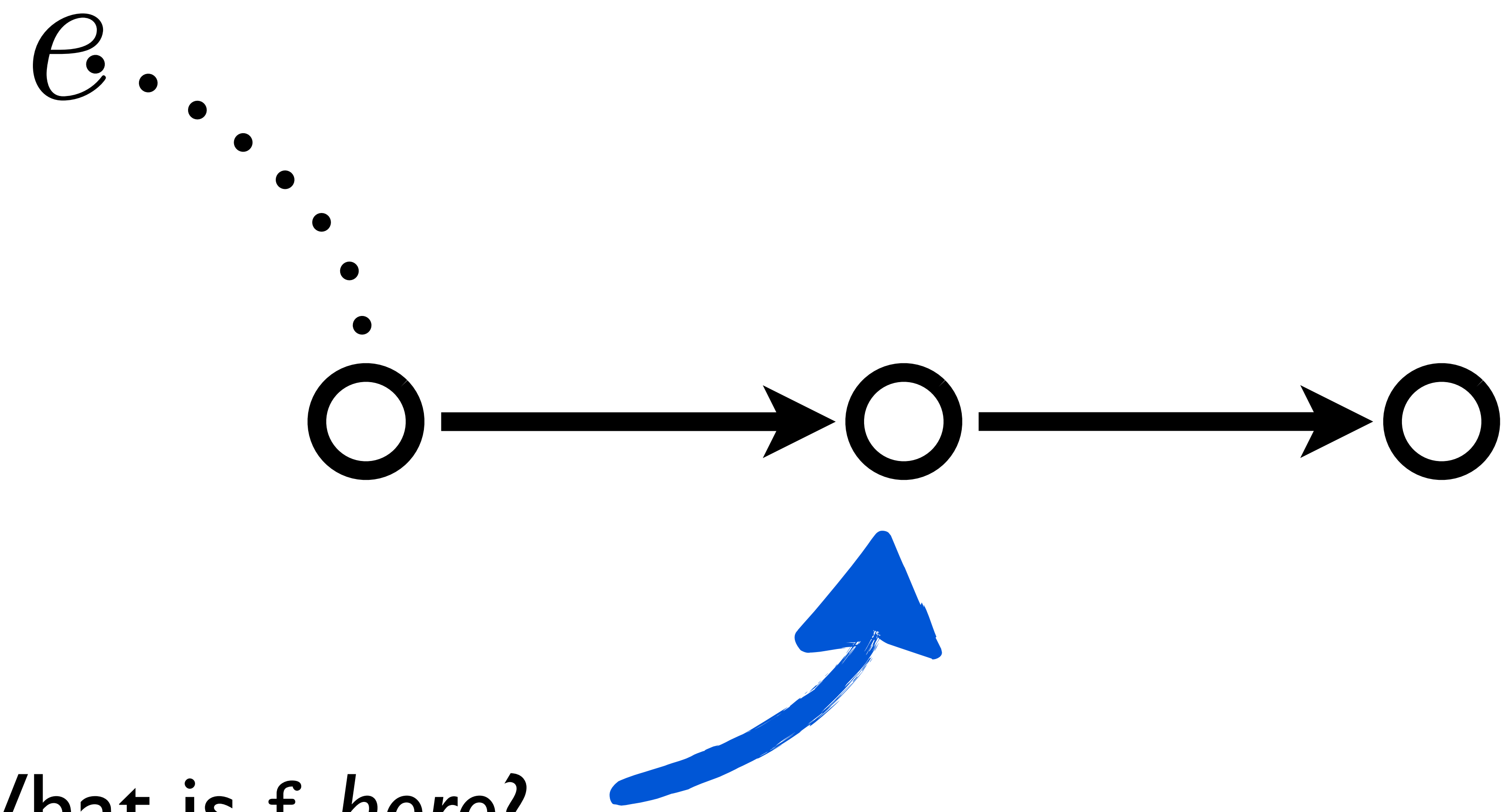
e


e

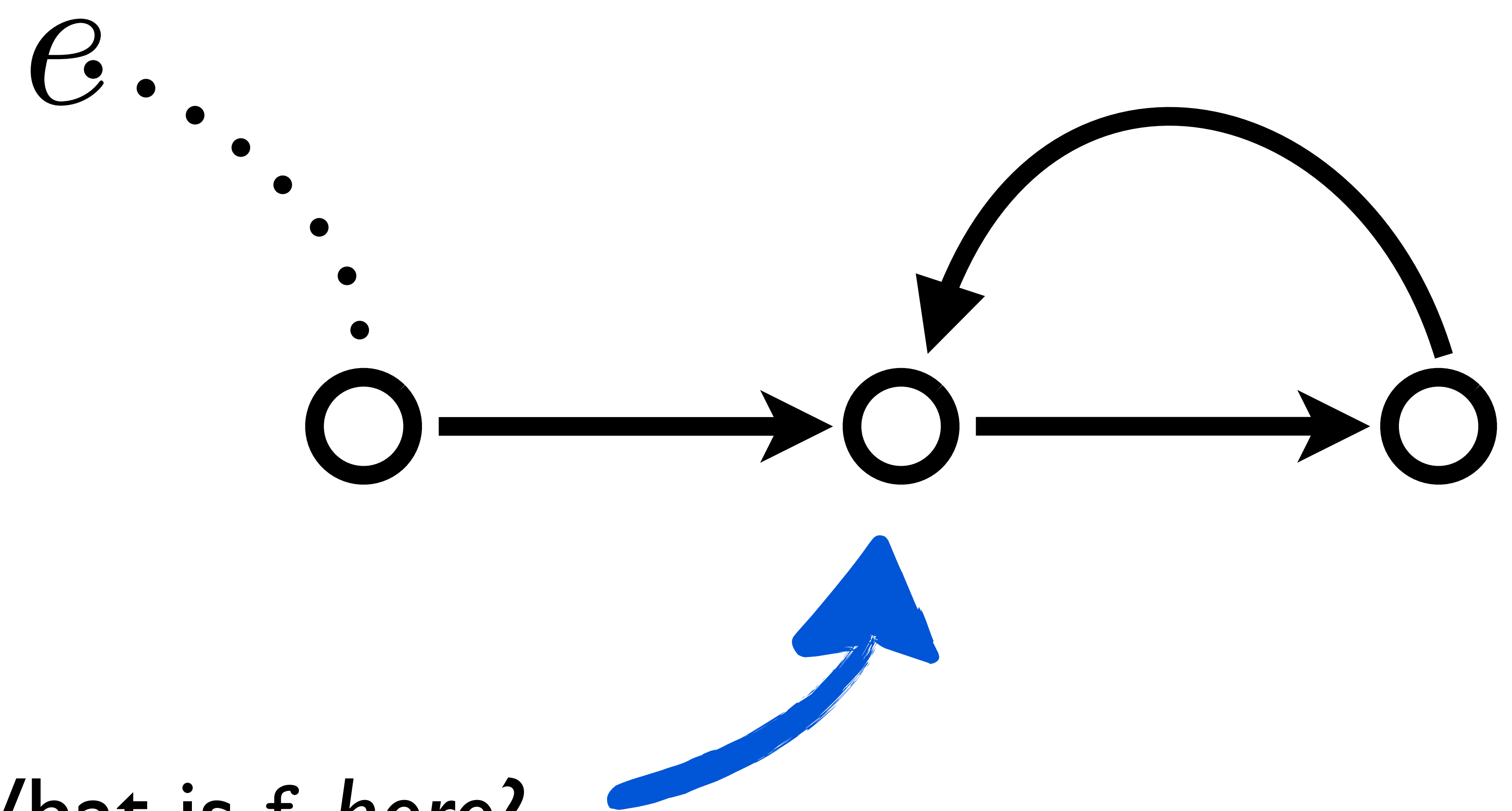

e



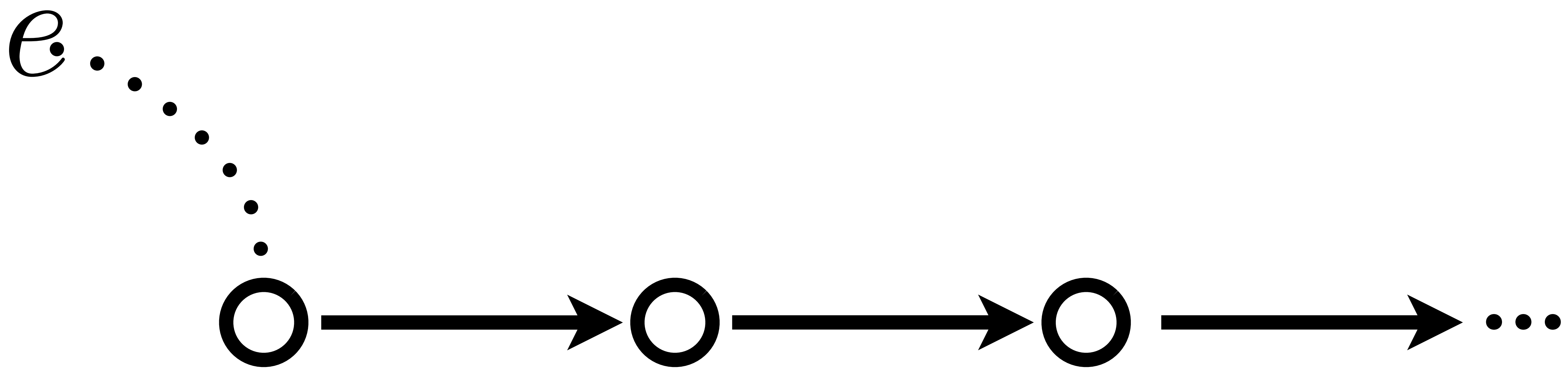
e



What is `f`, *here*?



What is `f`, *here*?


e

Make it terminate?

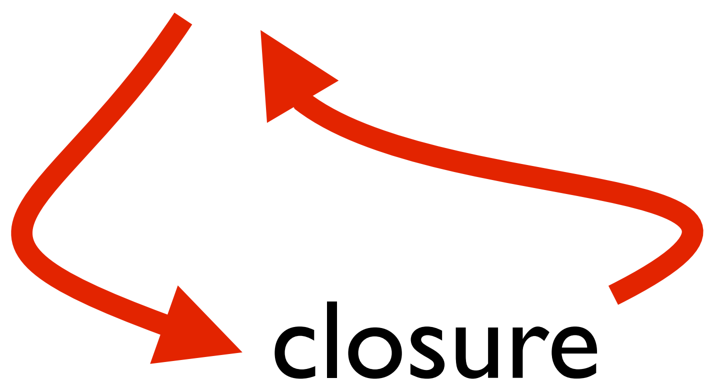
Make it finite.

Make it finite.

(Might, SAS 2010) (Van Horn and Might, ICFP 2010)

CEK

CEK



CEK

$$E = V \rightarrow \lambda\ x\ E$$

CEK

$$E = V \rightarrow \lambda x\ E$$

CEK

$$E = V \to A$$

CEK

CEK

CEK

CESK

CESK

(Felleisen and Friedman, 1987)

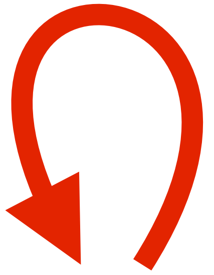
CESK

$$S = A \rightarrow \lambda \times E$$

CESK

$$S = A \to \lambda \times E$$

CESK



CESK

CESK*

CESK*

$$K^* \subset A$$

CESK*

$$S = A \rightarrow \lambda \times E + K$$

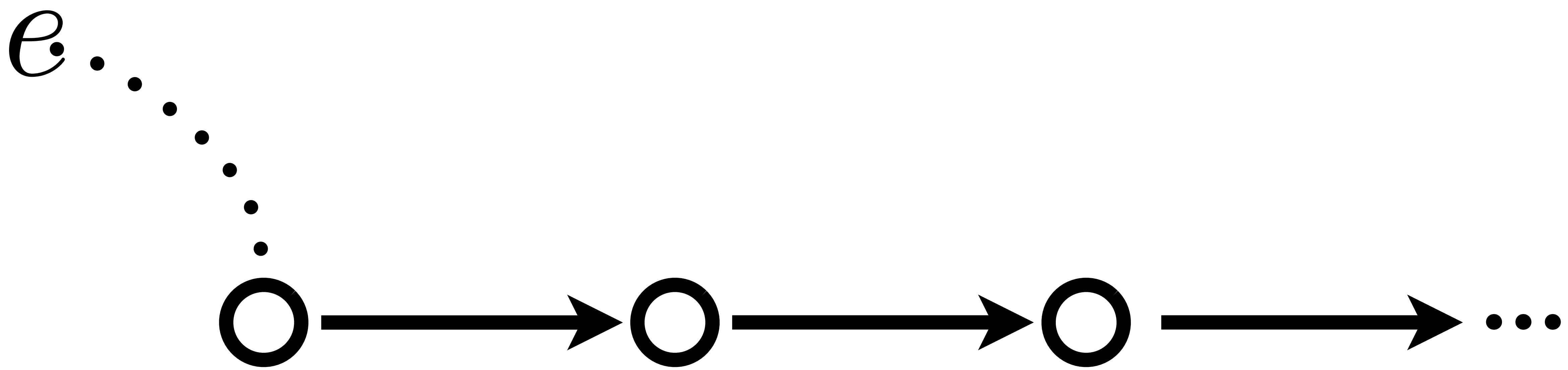
CESK*

CESK*

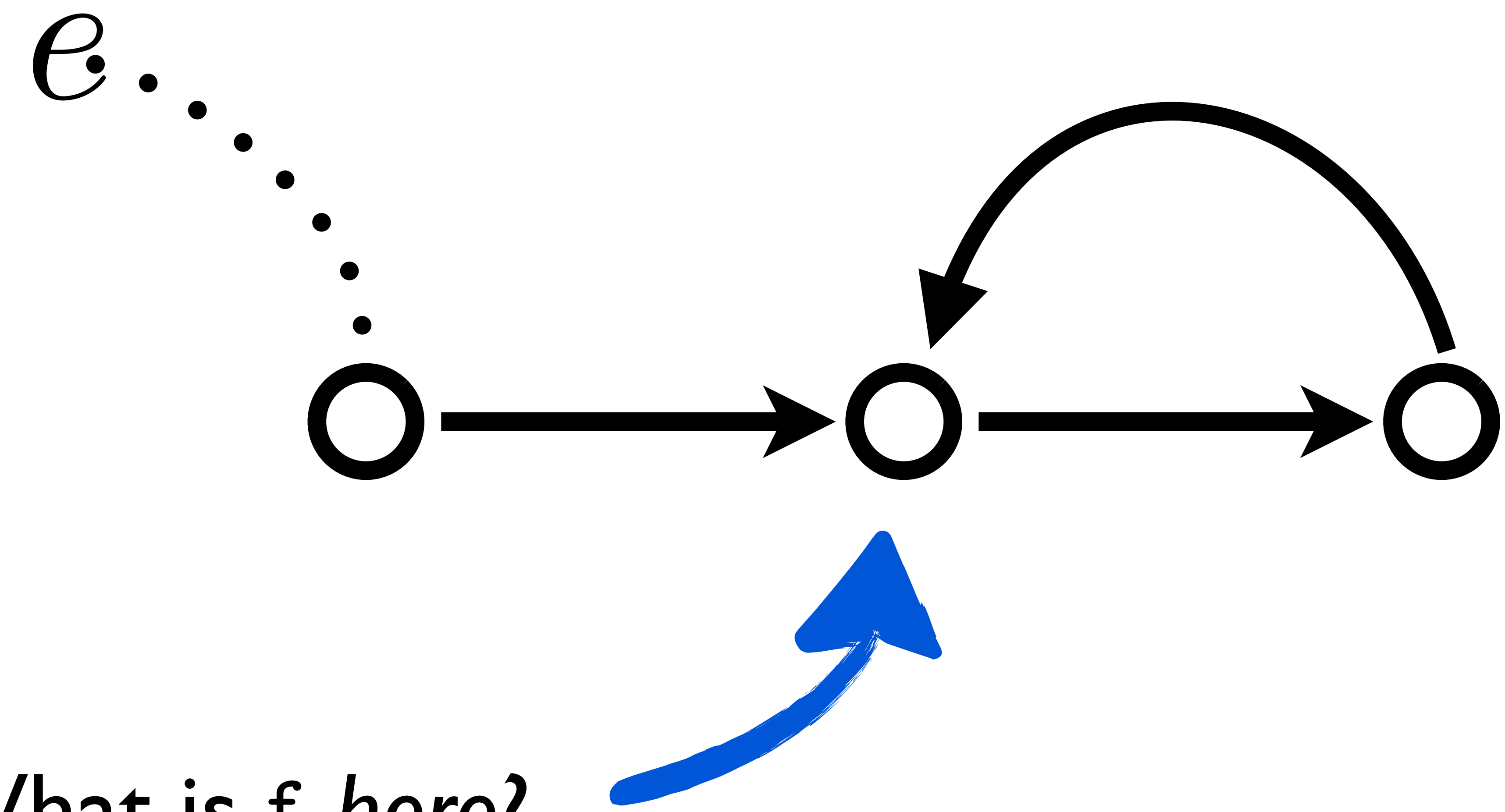
$C E \hat{S} K^*$

CESK*

e



e



What is $\mathtt{f}$, *here*?

What’s wrong with flow analysis?

Control-flow forks.

Data-flow merges.

Why?

$$S = A \rightarrow D$$

S = A → D

$$S = \hat{A} \longrightarrow D$$

$$S = \hat{A} \longrightarrow D$$

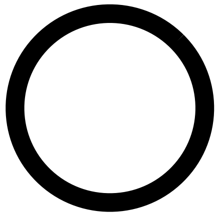
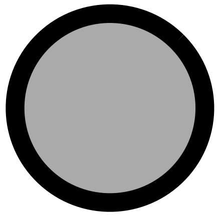
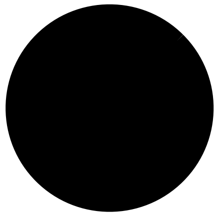
$$S = \hat{A} \rightarrow \hat{D}$$

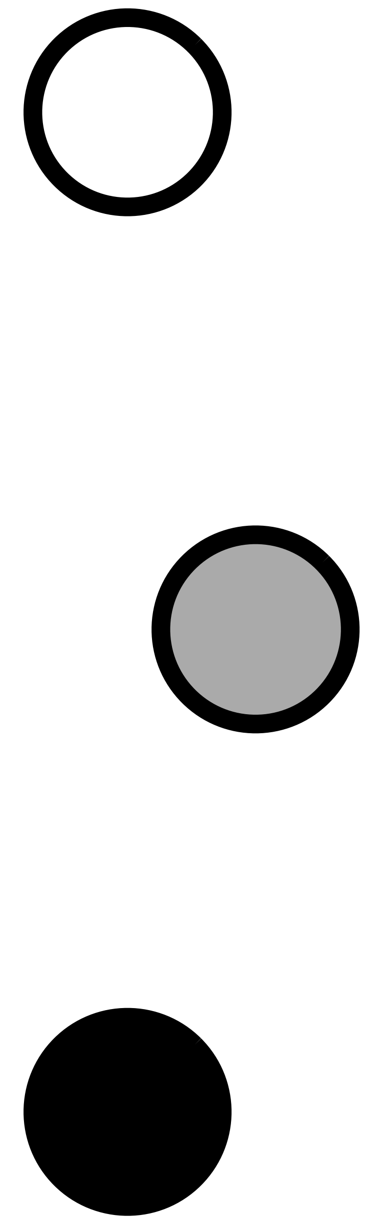
$$\widehat{S} = \widehat{A} \longrightarrow \mathcal{P}(\widehat{D})$$

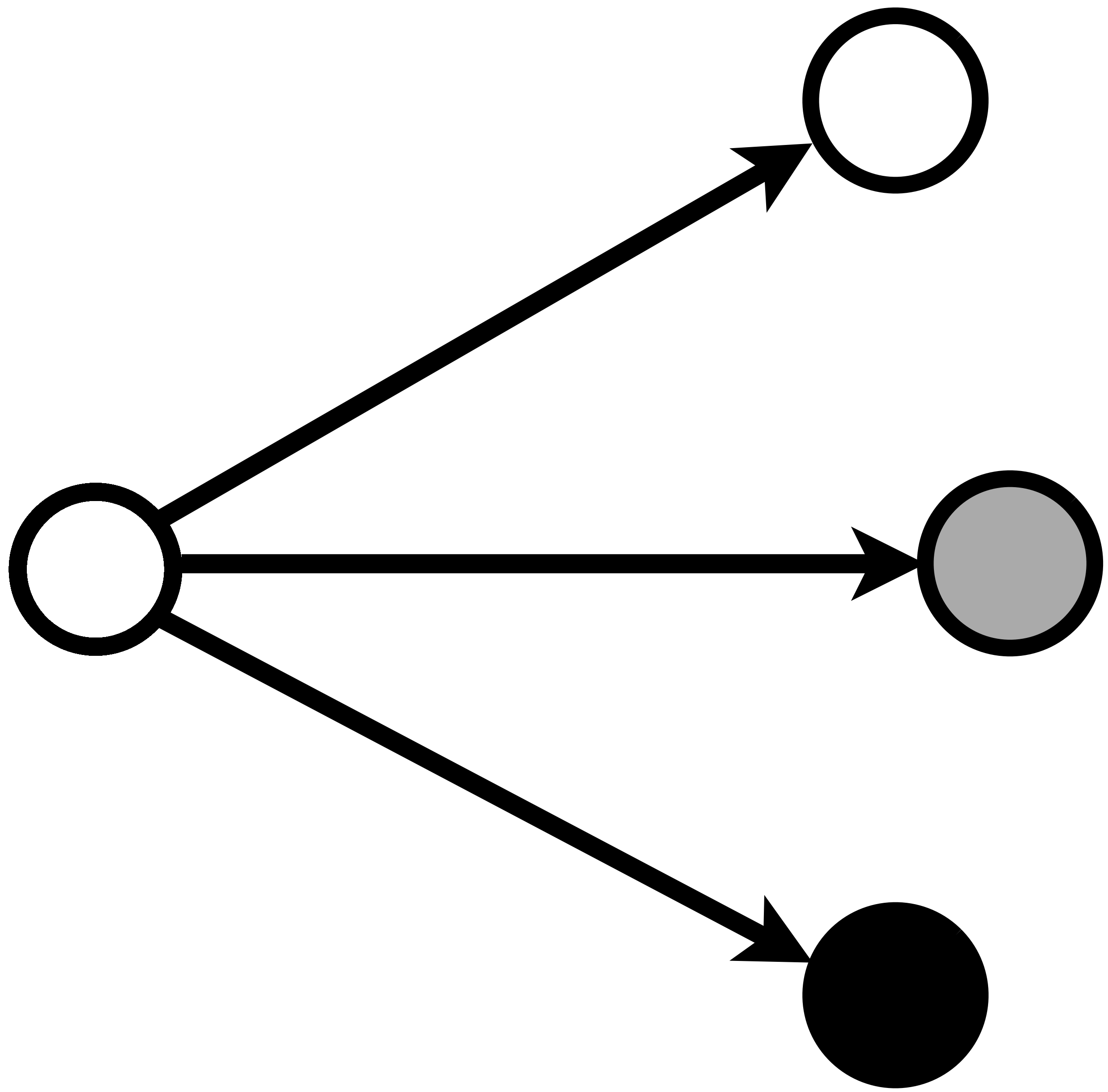
```
(f x)
```

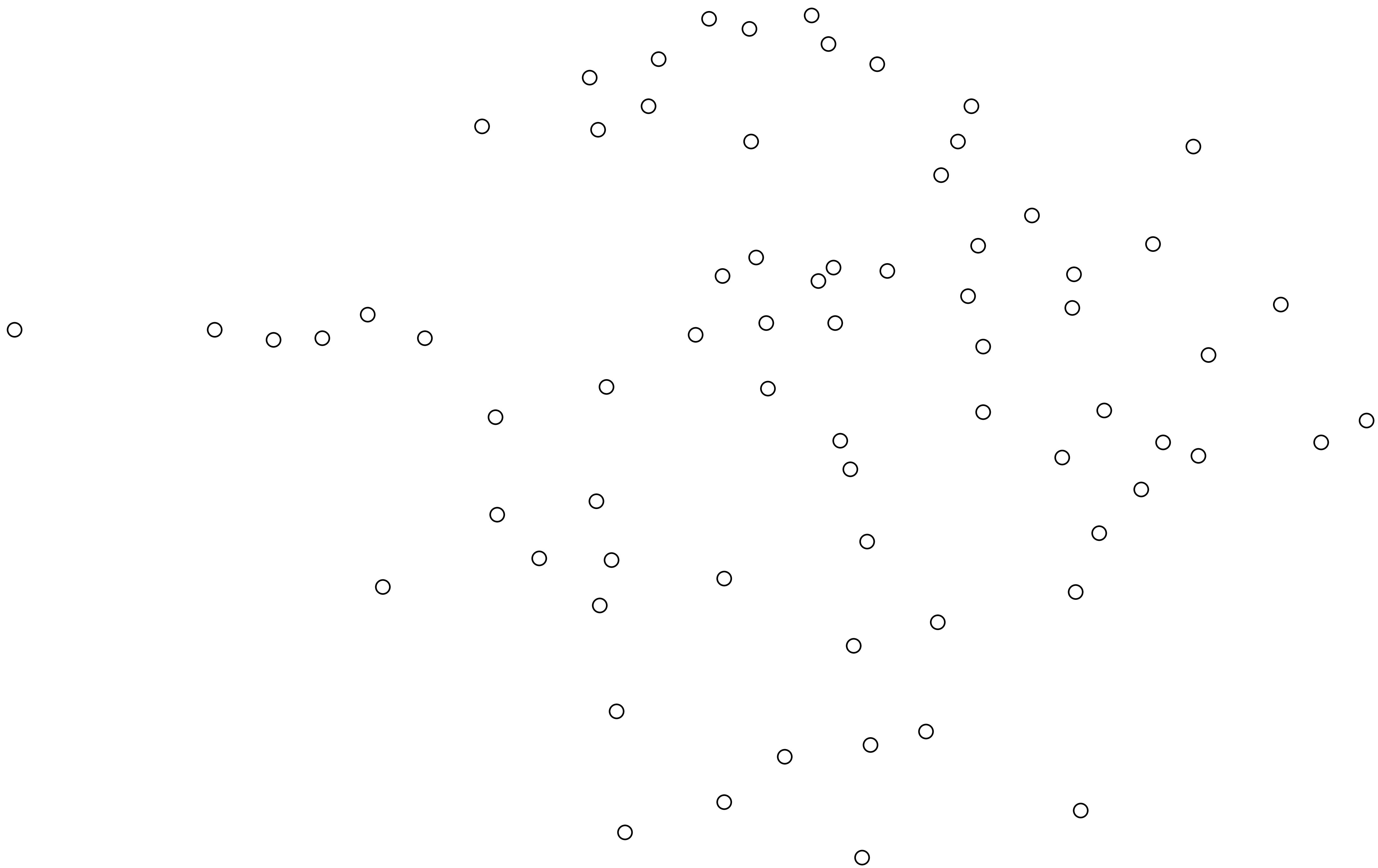
What is `f`?

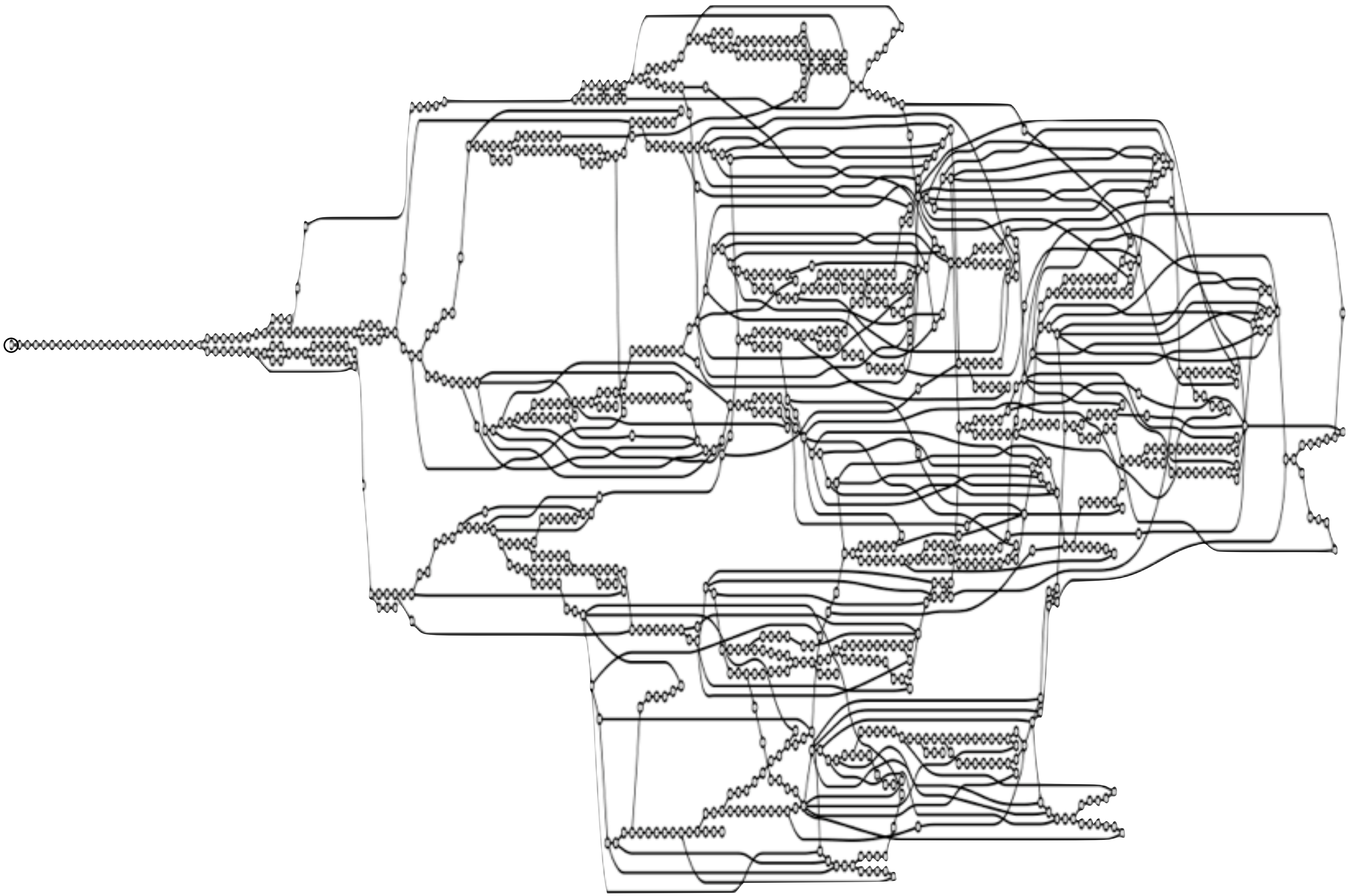
`f` is or or

`f` is  or  or 









Problem?

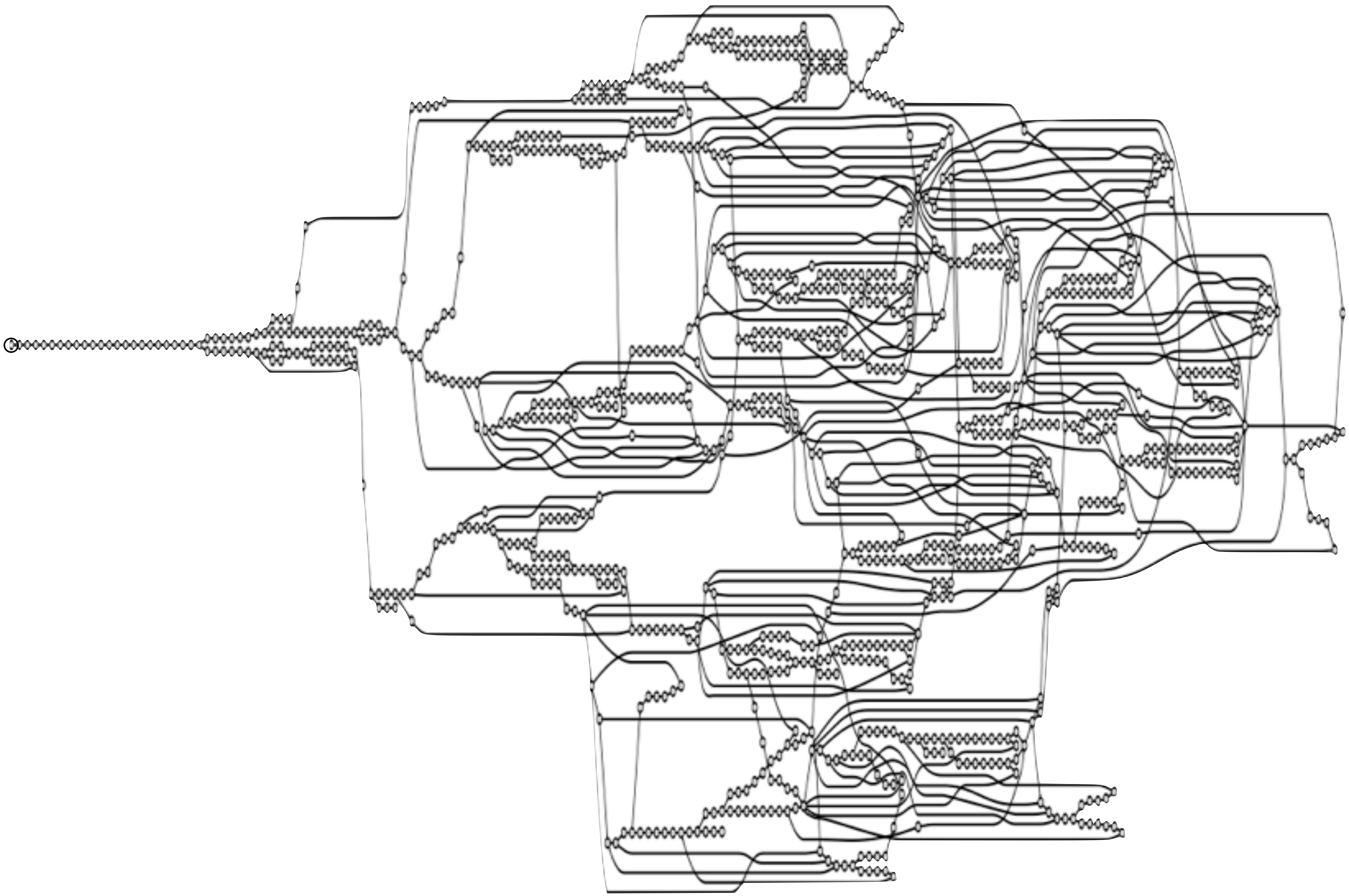
Finite store.

Solution?

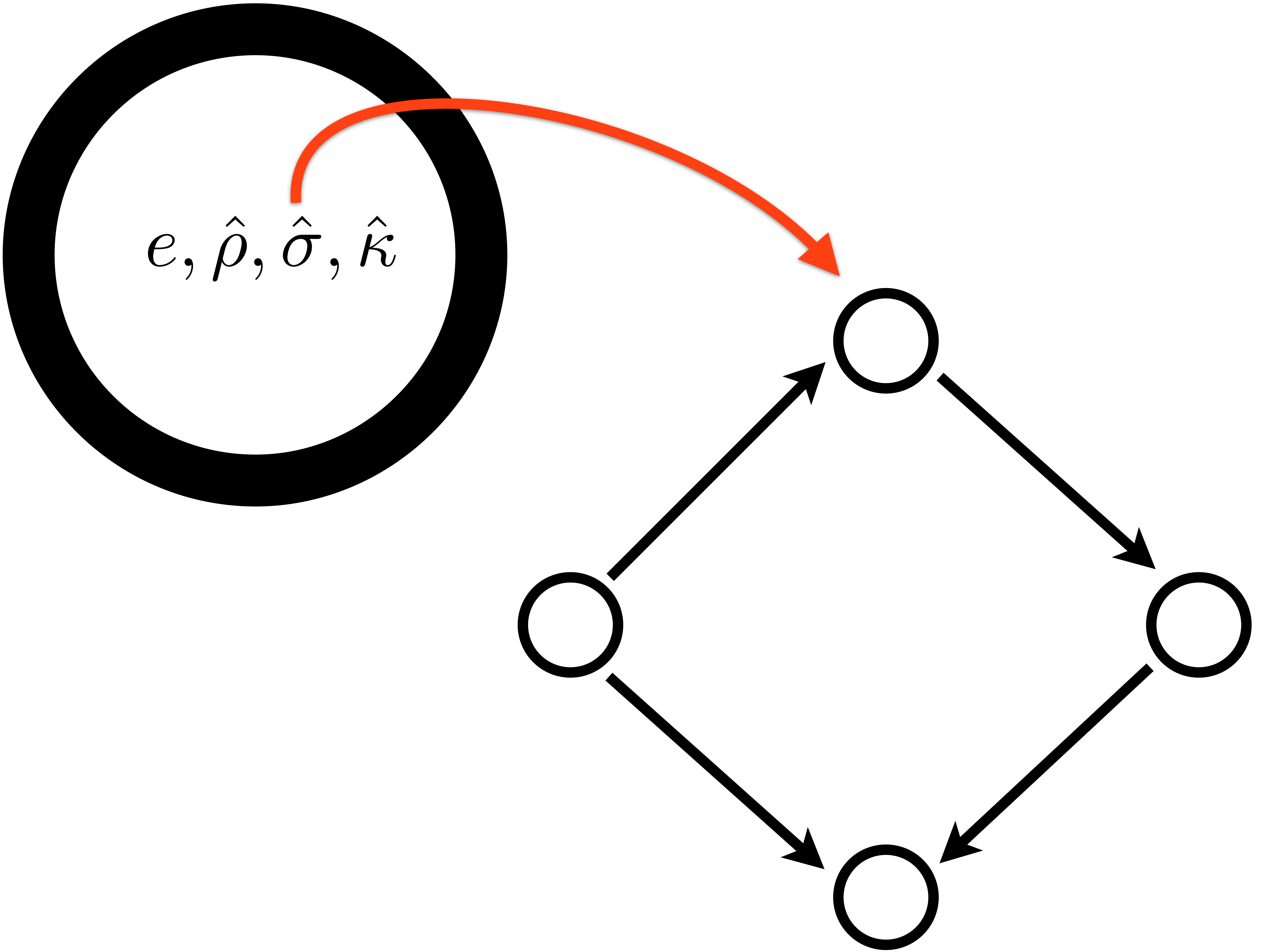
Toss garbage.

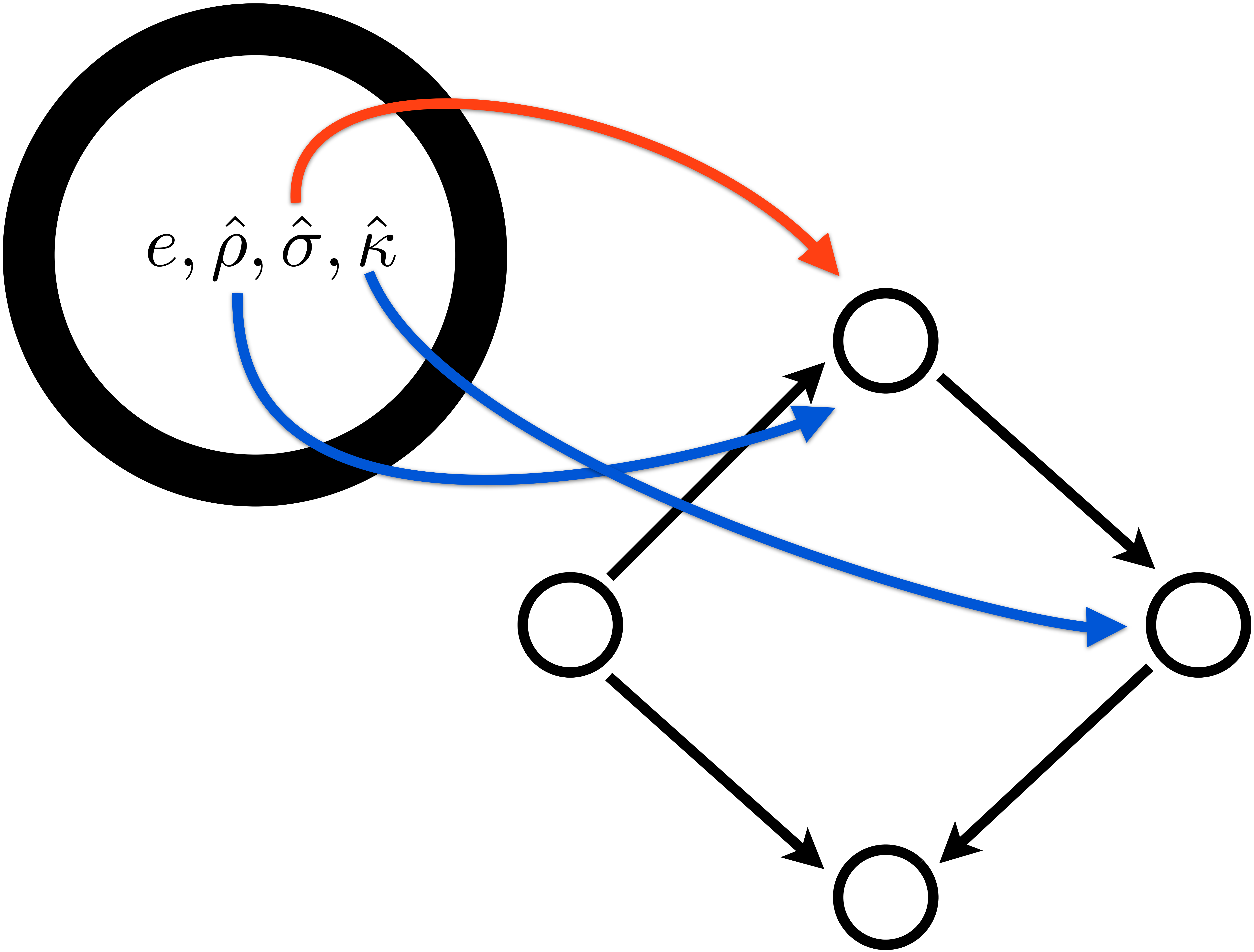
Toss garbage.

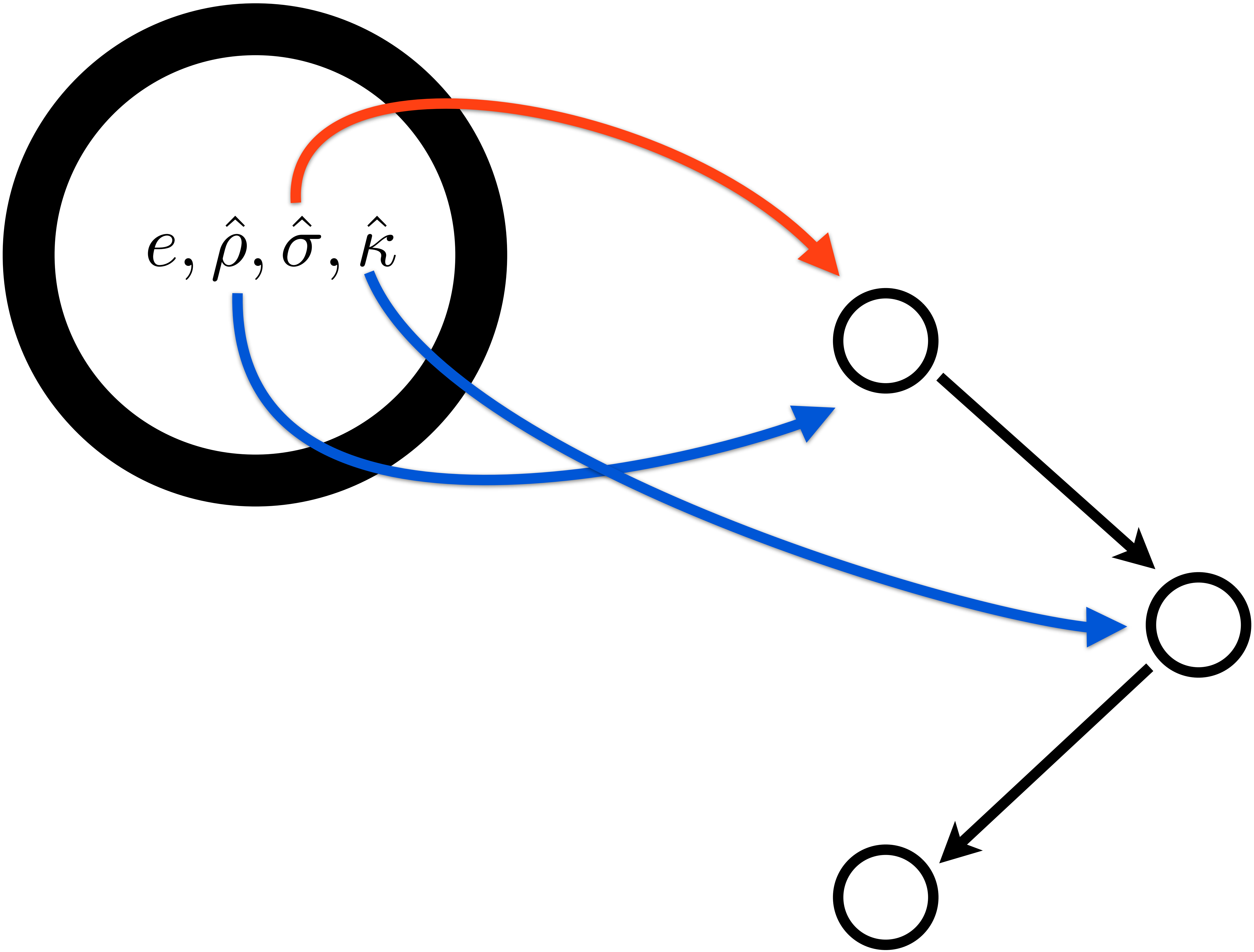
(Might & Shivers, ICFP 2006)



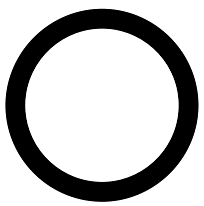

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

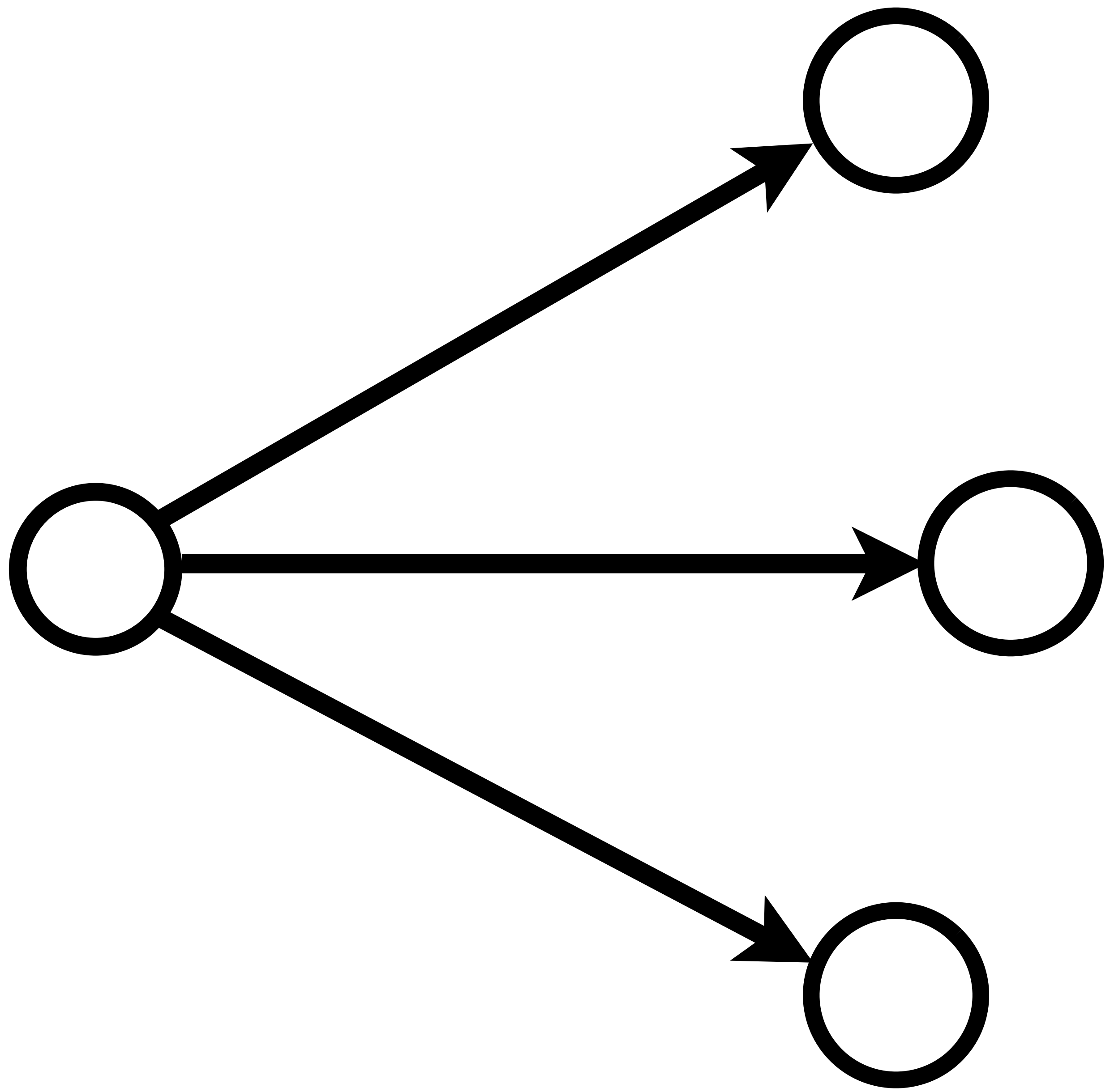

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

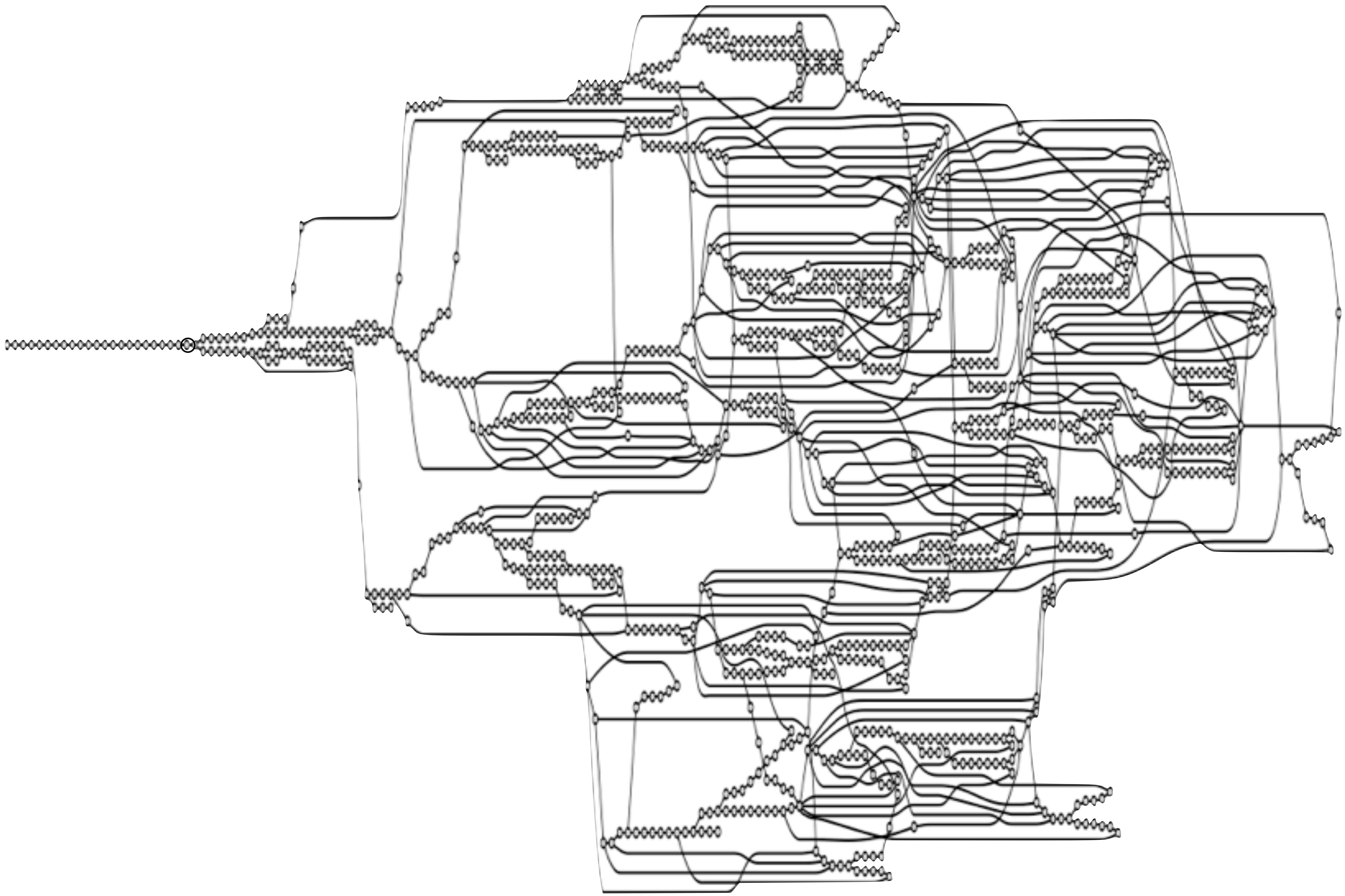

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

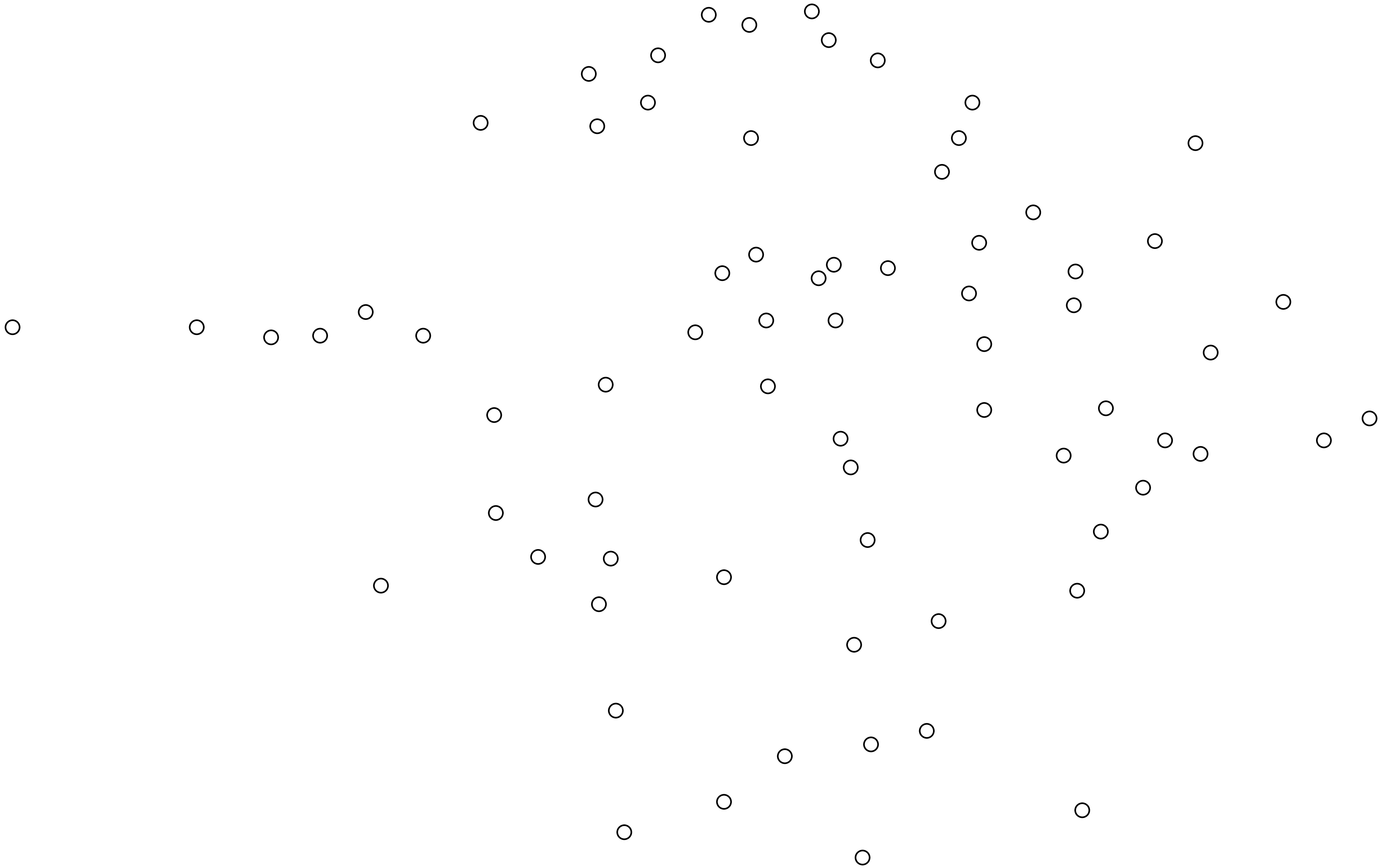

$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

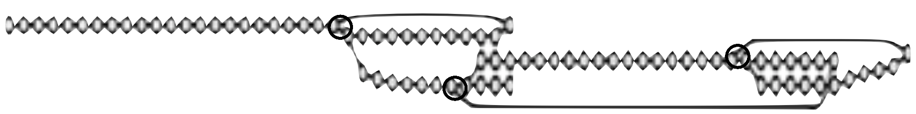

$e, \hat{\rho}, \hat{\sigma}', \hat{\kappa}$









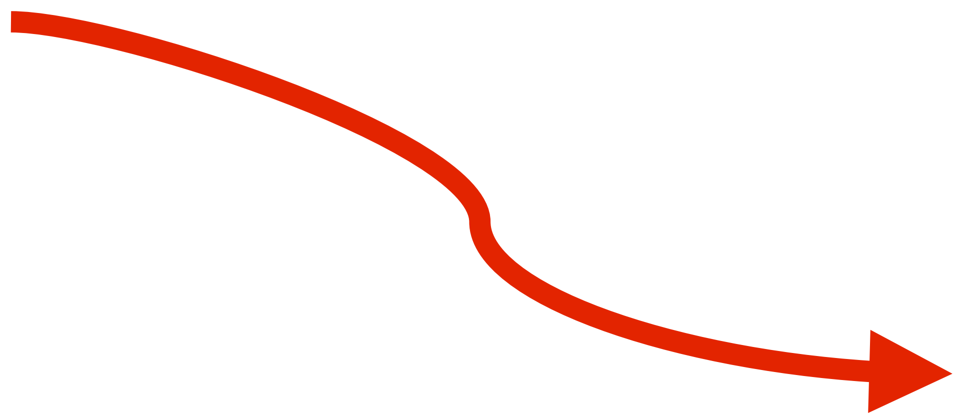


Control-flow forks.

Return-flow forks.

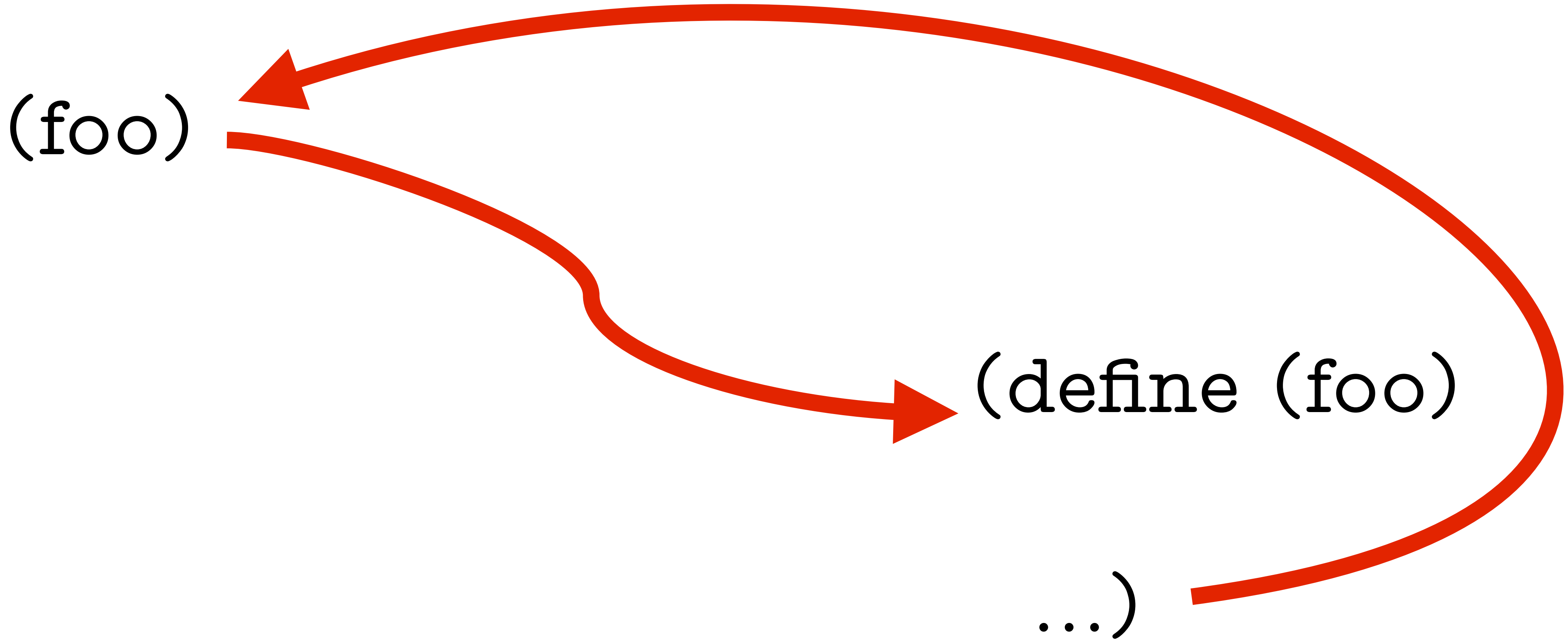
```
(foo)

                    (define (foo)
                      ...)

(foo)
```

```
(foo)
```

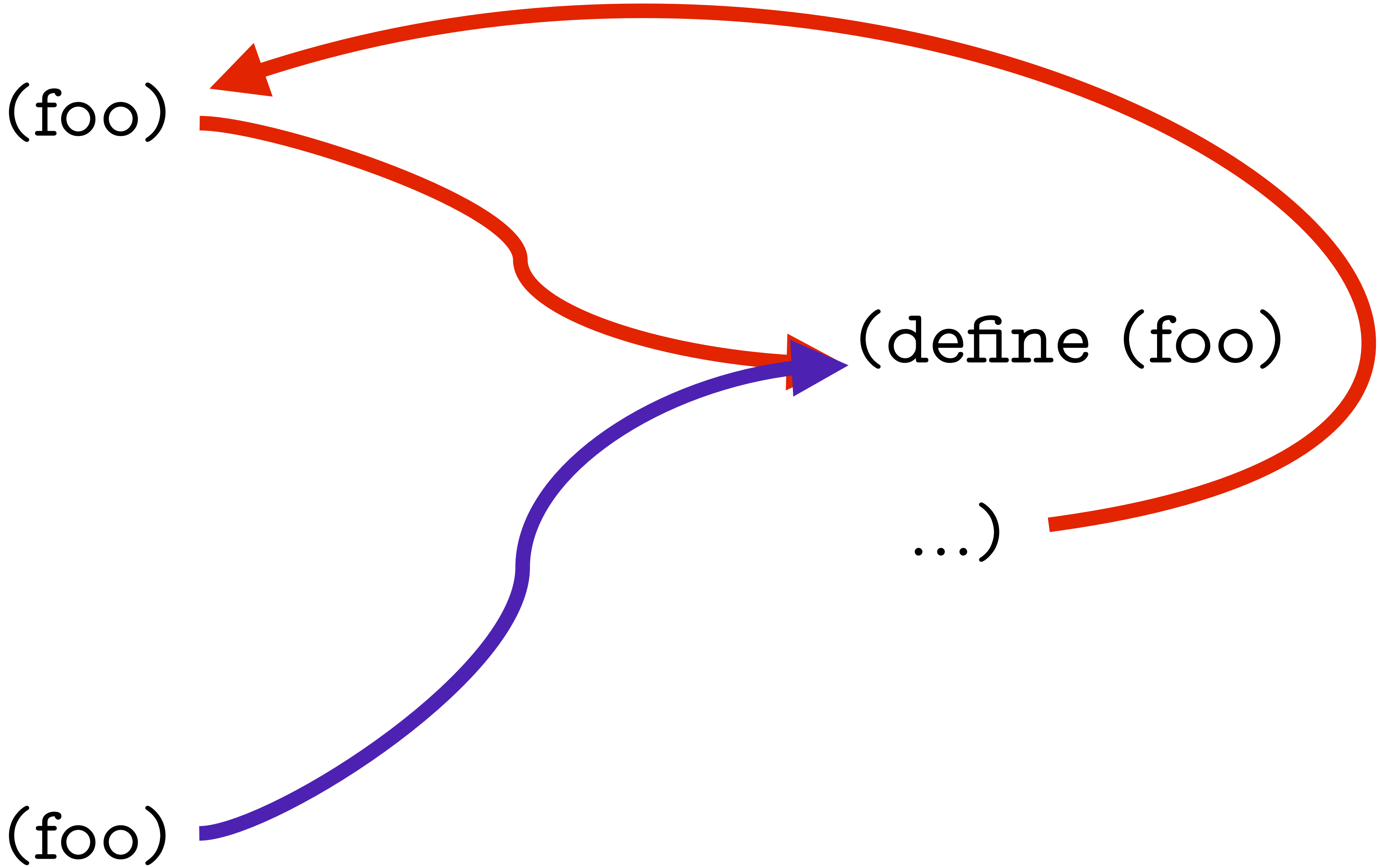


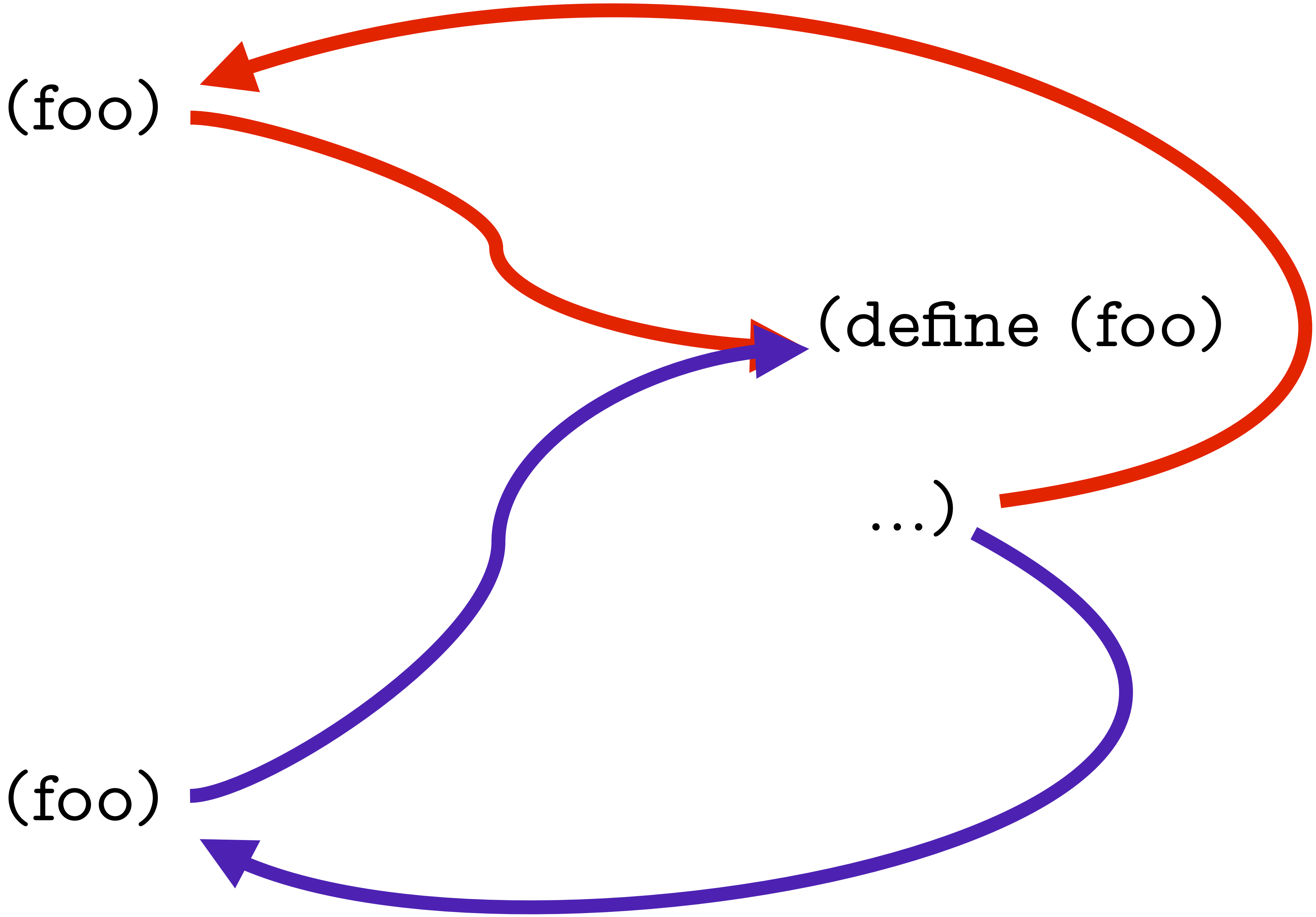
```
(define (foo)
  ...)
```

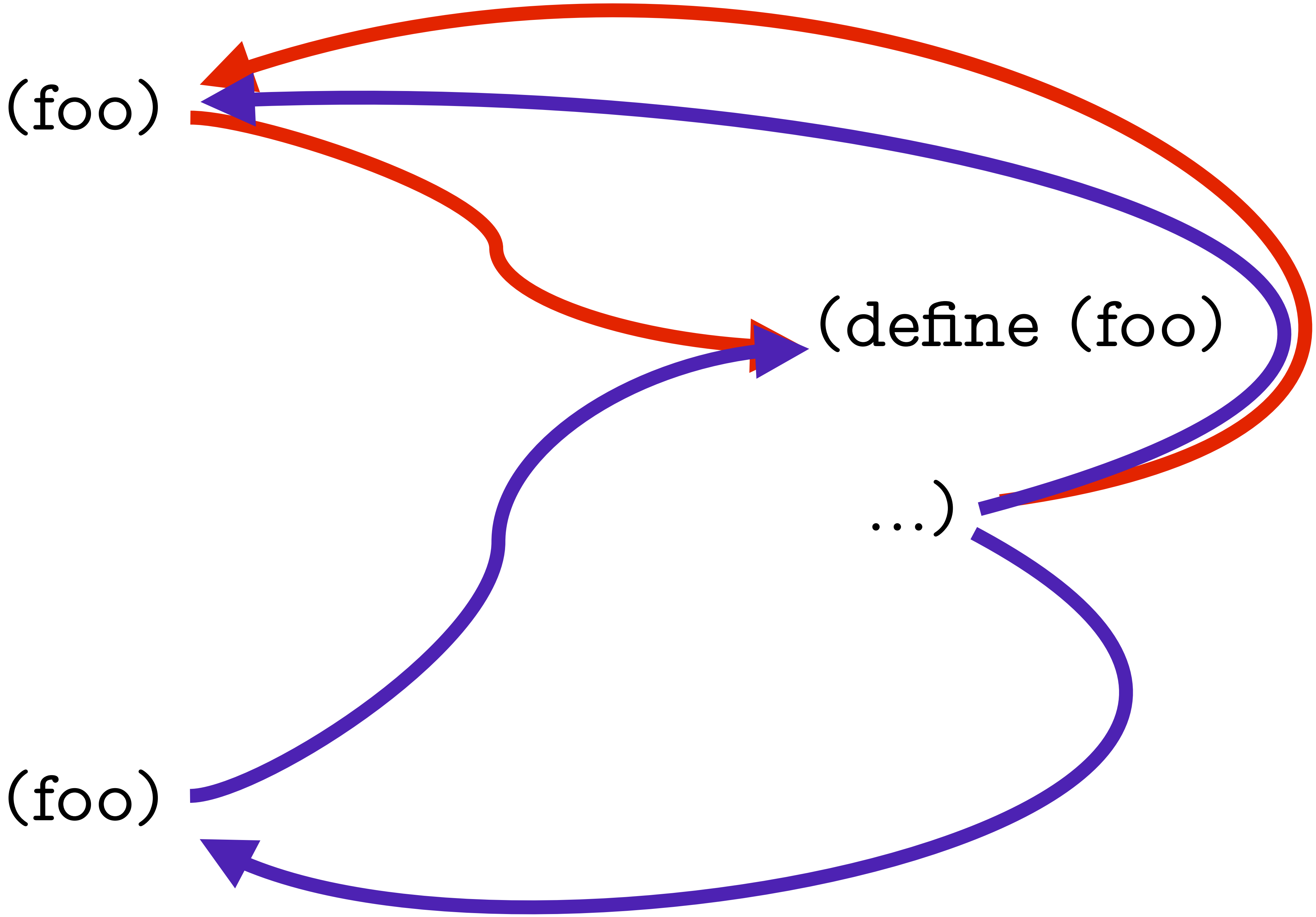
```
(foo)
```




```
(foo)
```


(foo)
(define (foo)
...)
(foo)


(foo)
(define (foo)
...)
(foo)


(foo)
(define (foo)
...)
(foo)

$$\widehat{S} = \widehat{A} \longrightarrow \mathcal{P}(\widehat{D})$$

$$\widehat{S} = \widehat{A} \longrightarrow \mathcal{P}(\lambda \times \widehat{E} + \widehat{K})$$

CESK

CESK*

CESK*

CESK

CES K

$\widehat{CES}$ $\hat{K}$

$\widehat{CES}\ \hat{K}$

control state

$\widehat{CES}\ \widehat{K}$

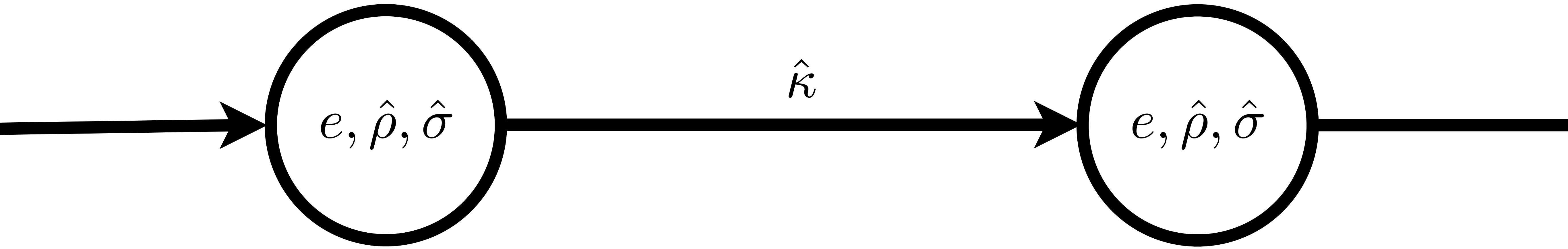
control state stack

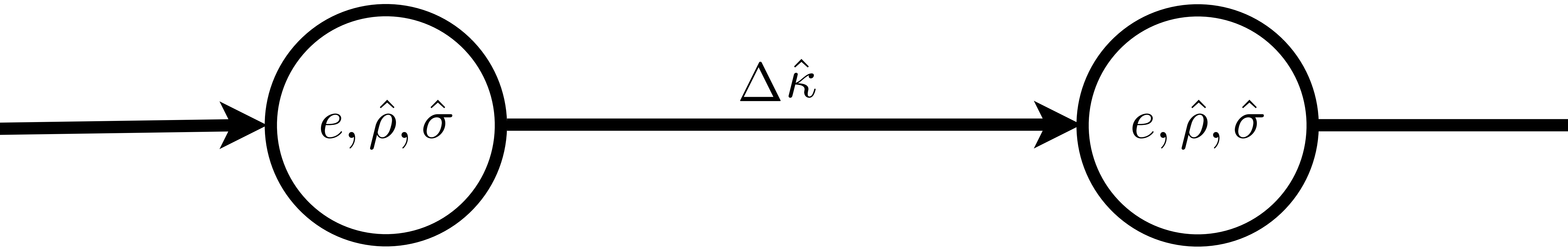
control state + stack = pushdown

control state + stack = pushdown

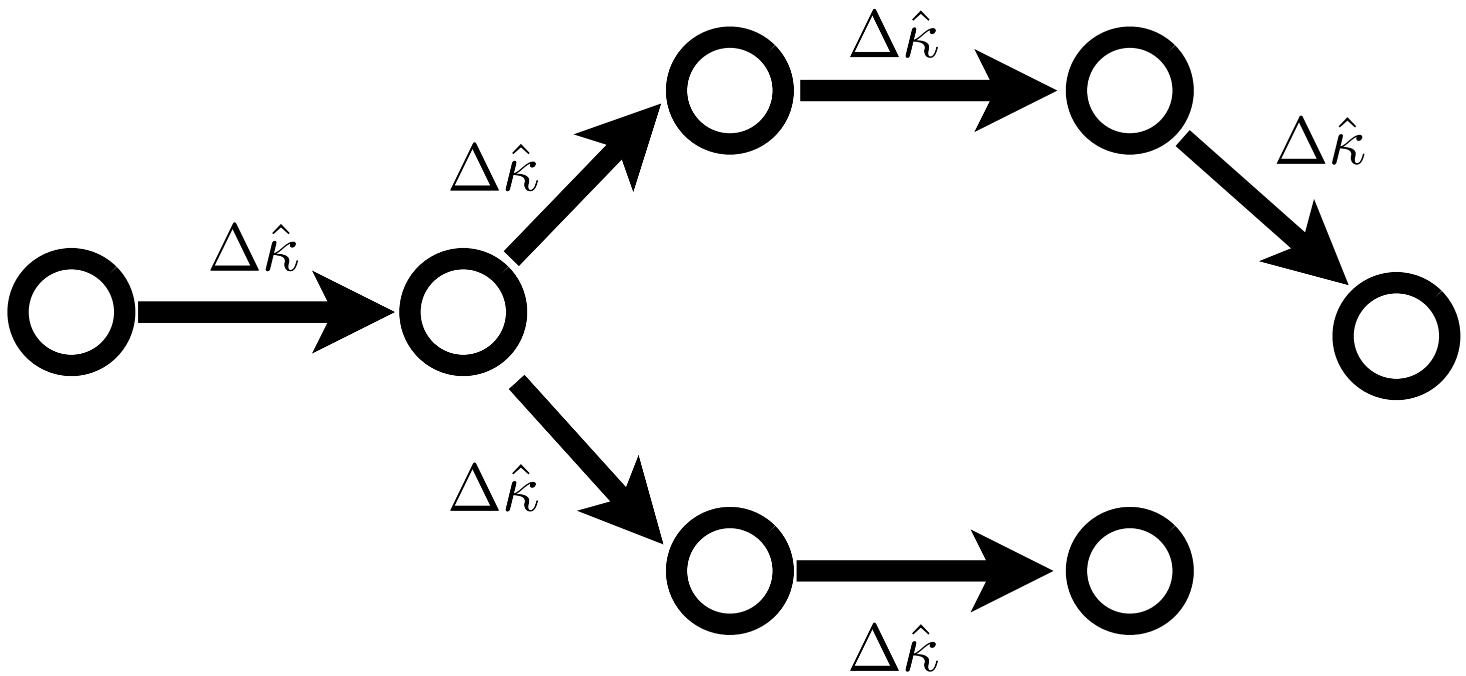
(Vardoulakis and Shivers, CFA2)


$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$
$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$

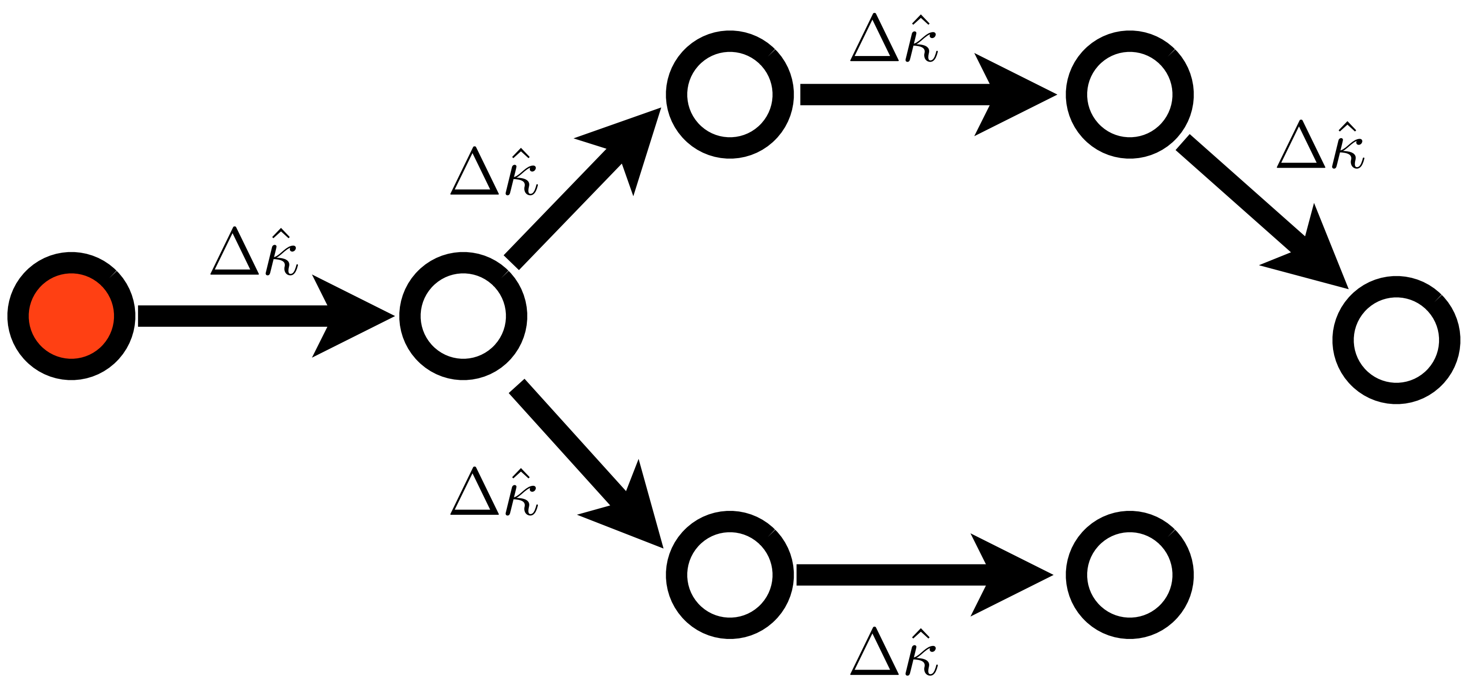

$e, \hat{\rho}, \hat{\sigma}$
$\hat{\kappa}$
$e, \hat{\rho}, \hat{\sigma}$



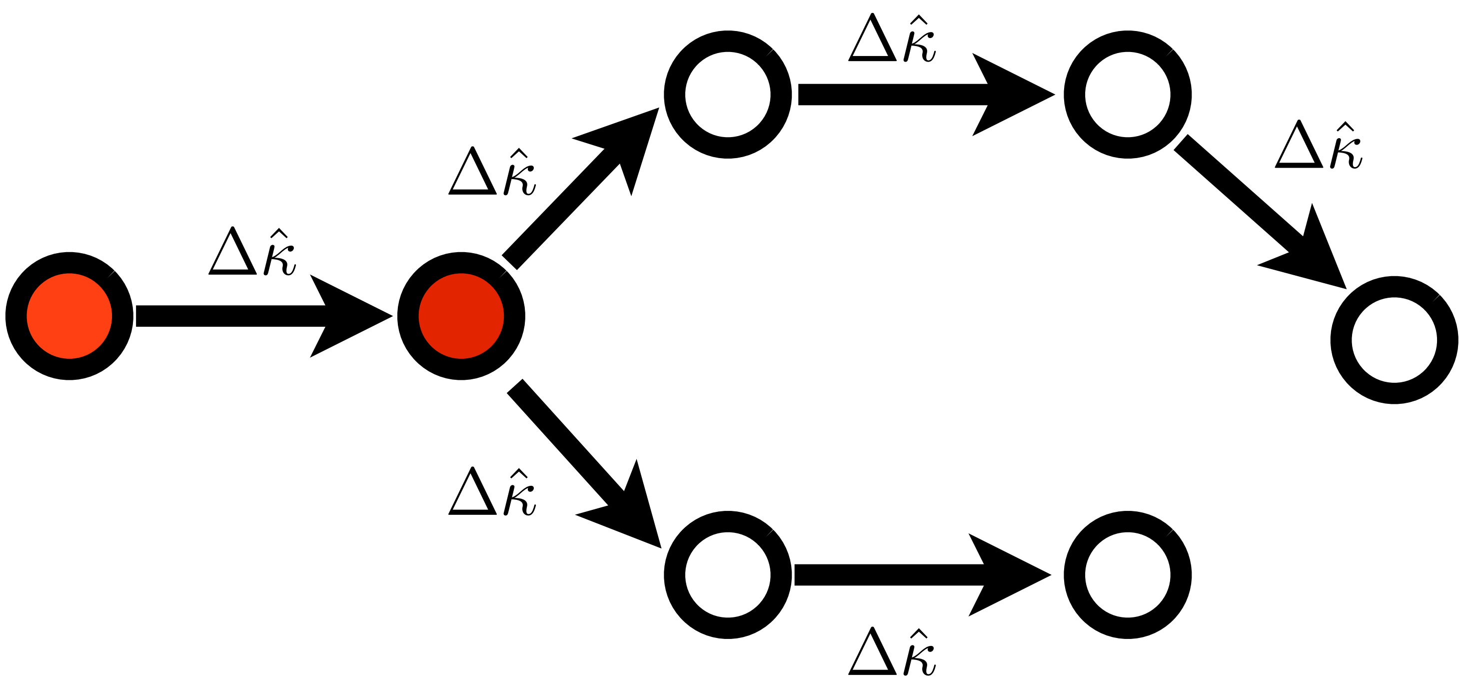
$e, \hat{\rho}, \hat{\sigma}$
$\Delta\hat{\kappa}$
$e, \hat{\rho}, \hat{\sigma}$



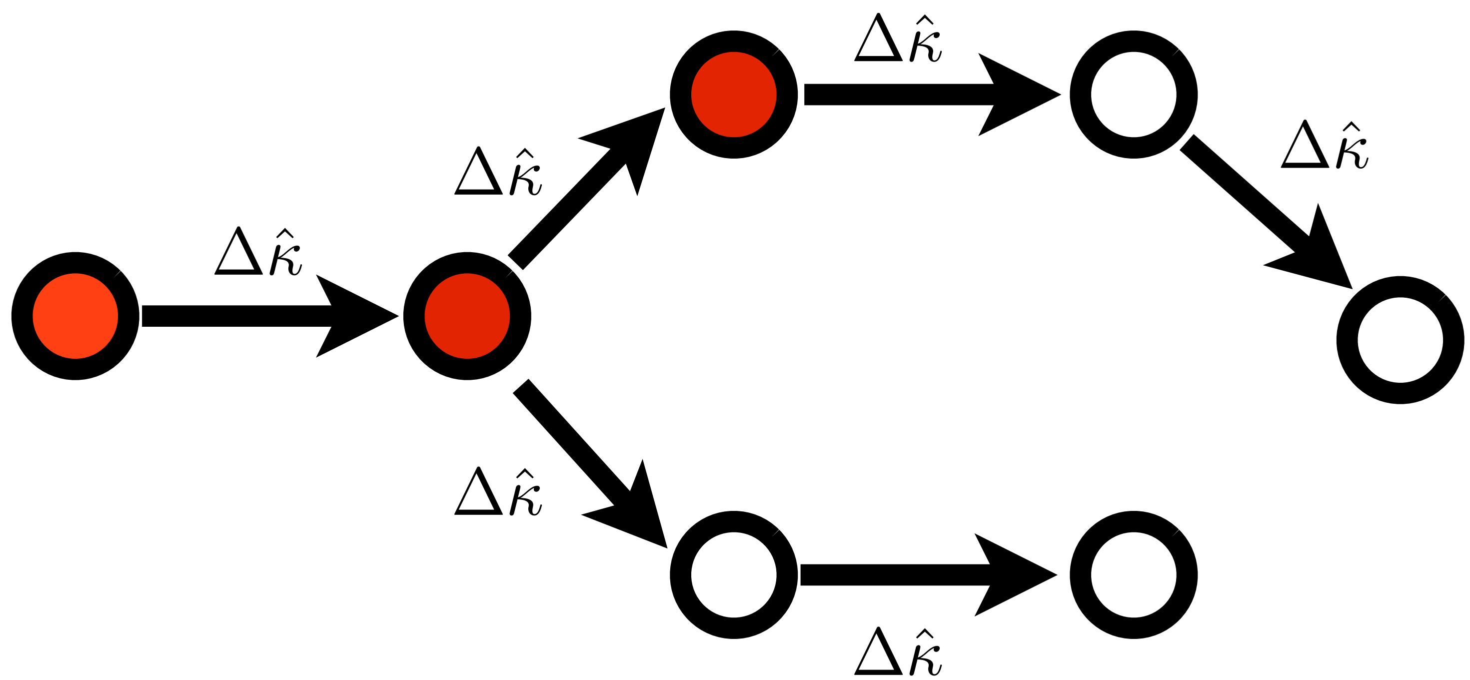
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$

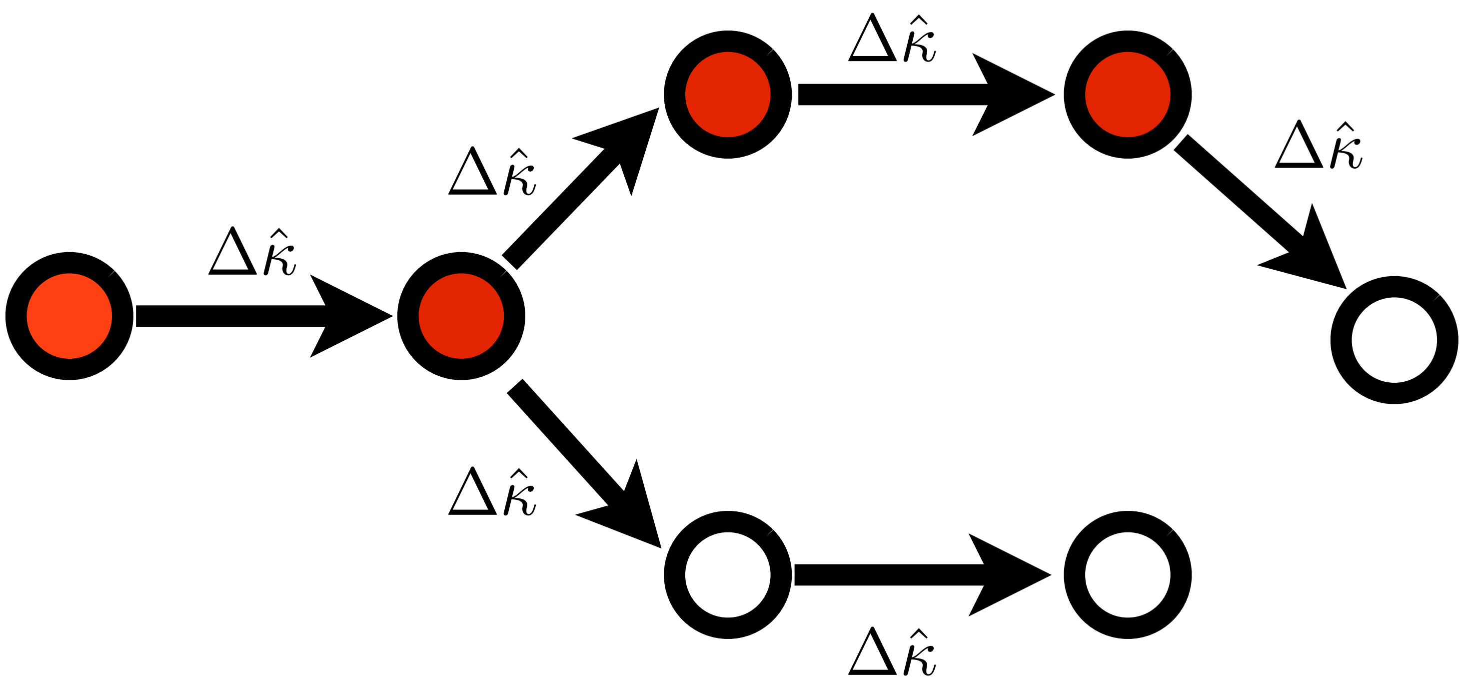


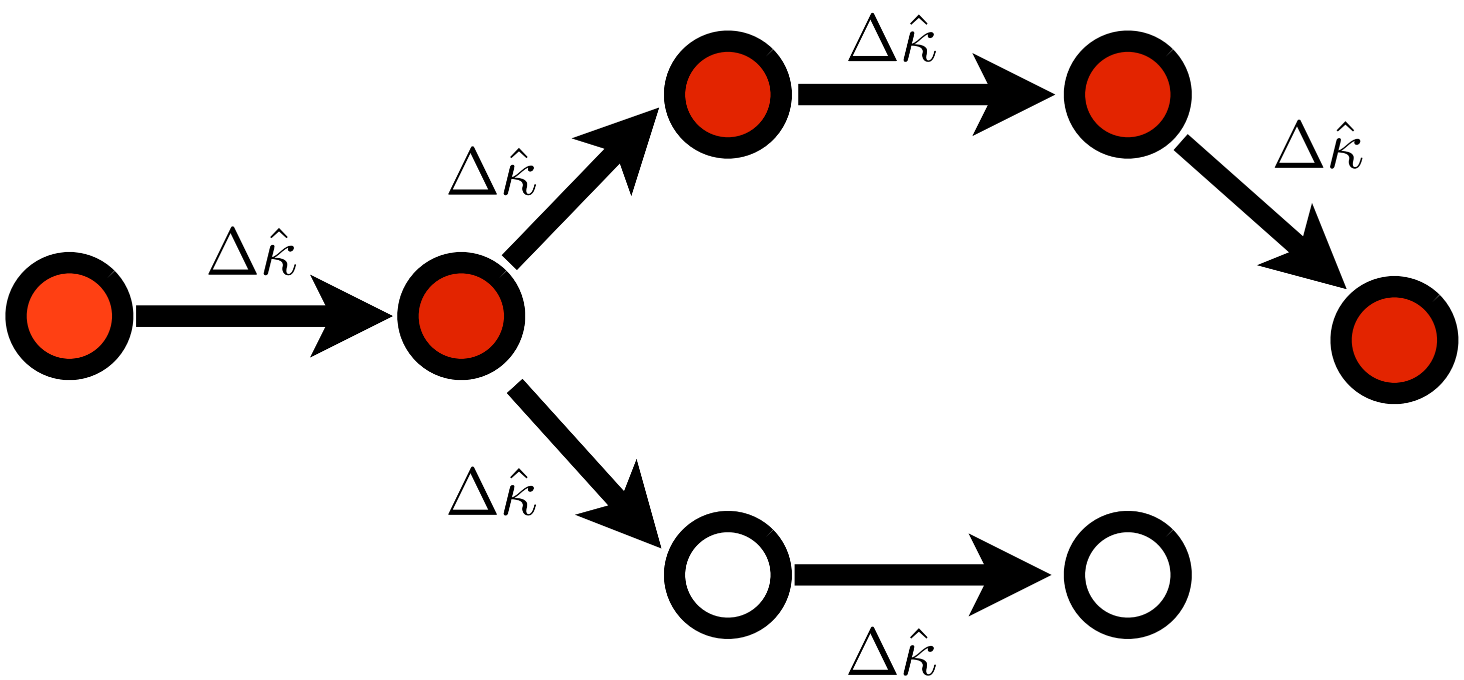
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$

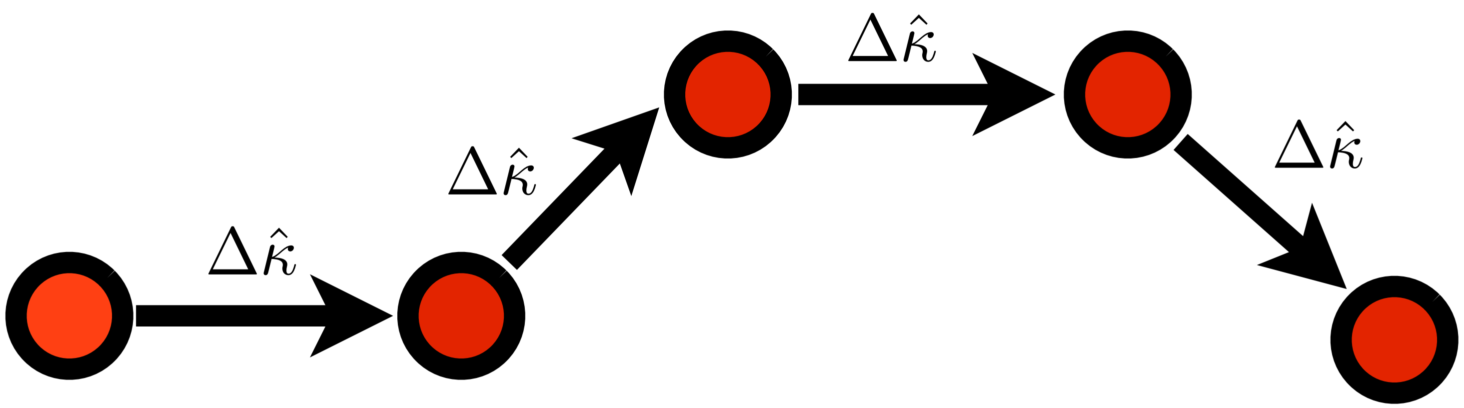


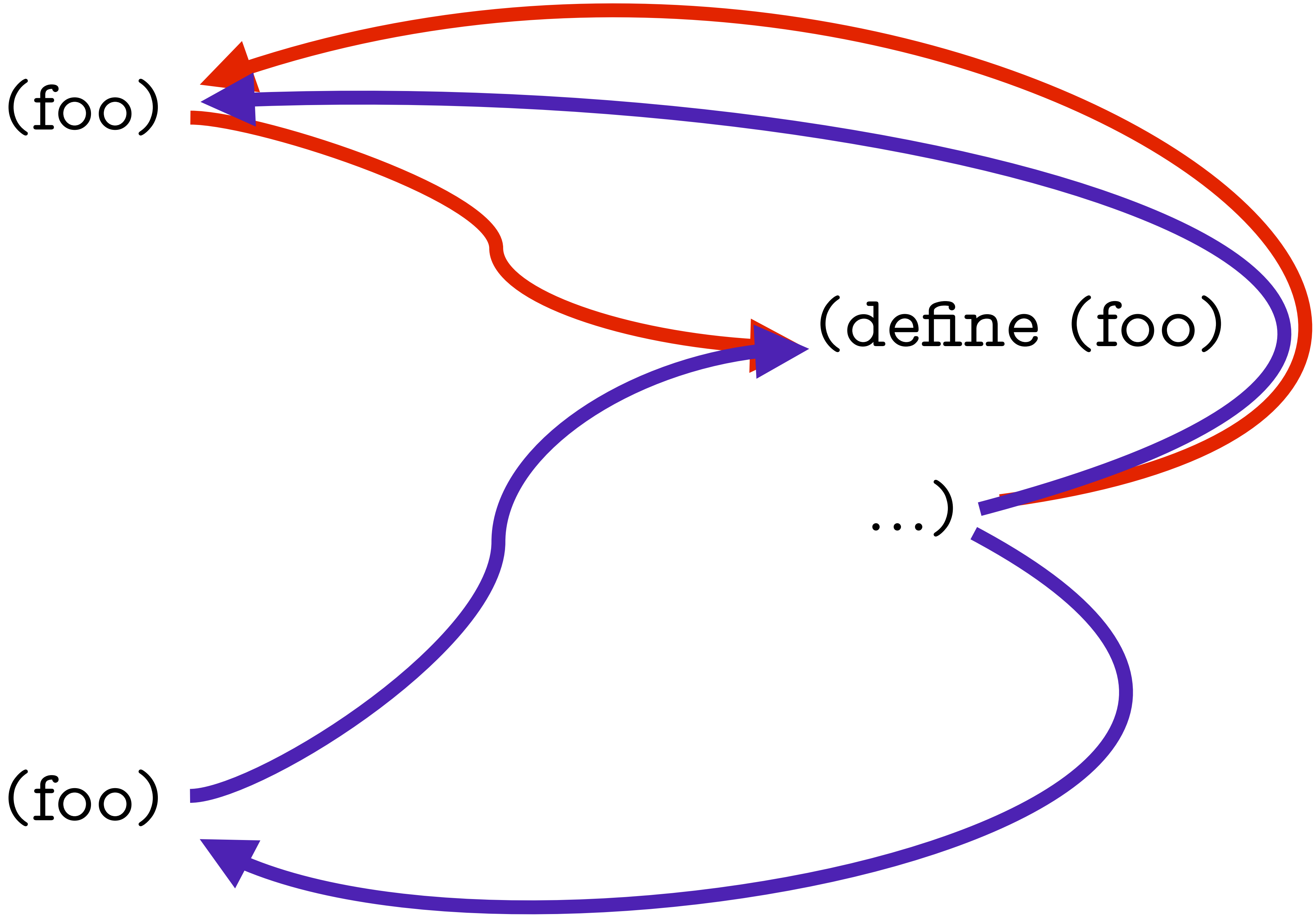
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$


$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$

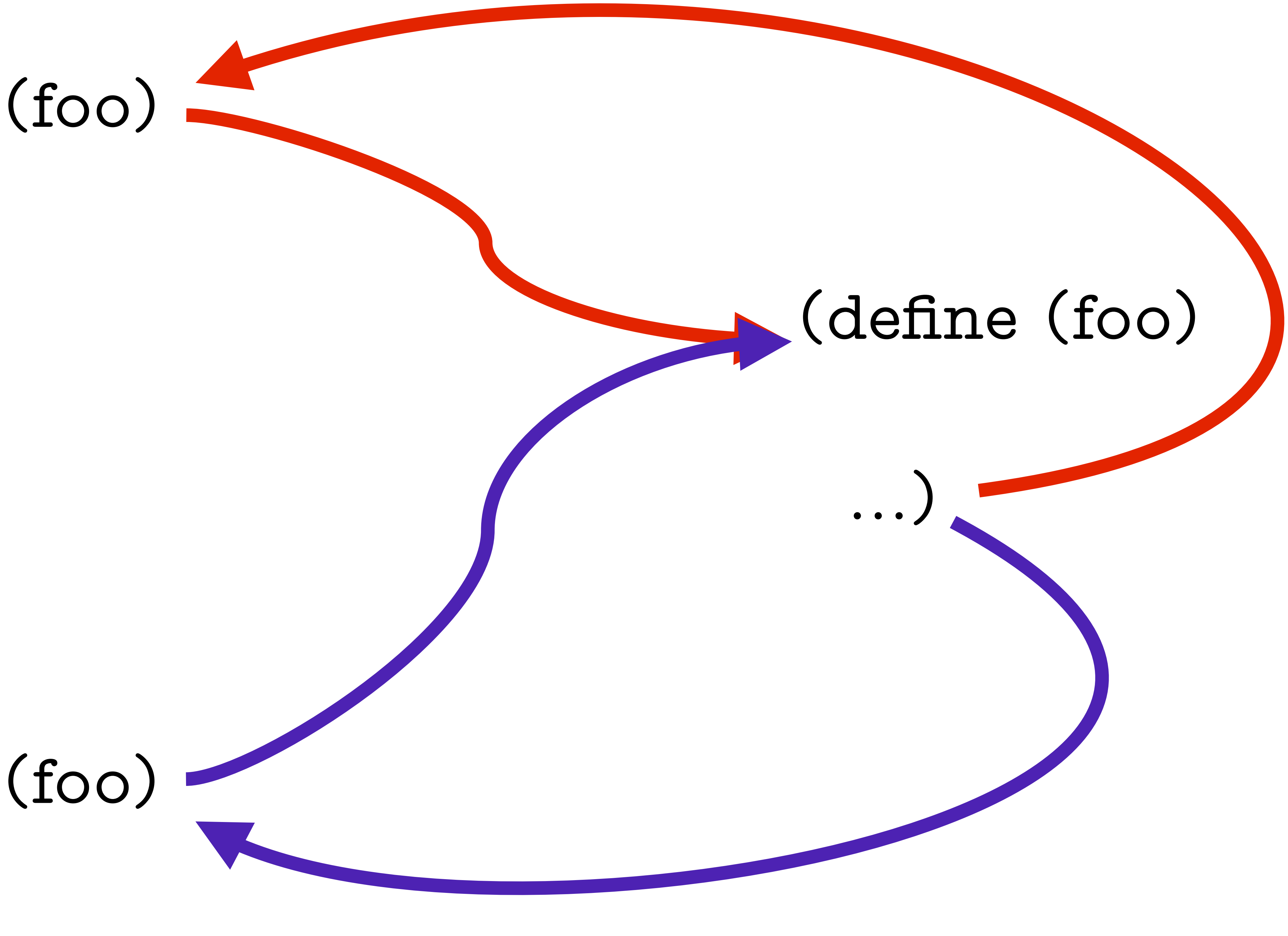

$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$


$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$


$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$



(foo)
(define (foo)
...)
(foo)


(foo)
(define (foo)
...)
(foo)

GC & pushdown?

Halt!

$$\delta : Q \times \Delta\Gamma \rightarrow \mathcal{P}(Q)$$

Control state

Next states

$$\delta : Q \times \Delta\Gamma \to \mathcal{P}(Q)$$

Stack change

$$\delta : Q \times \Delta\Gamma \times \Gamma^* \rightarrow \mathcal{P}(Q)$$

Entire stack

$$\delta : Q \times \Delta\Gamma \times \Gamma^* \rightarrow \mathcal{P}(Q)$$

$$\delta : Q \times \Delta\Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q)$$

Infinite stacks

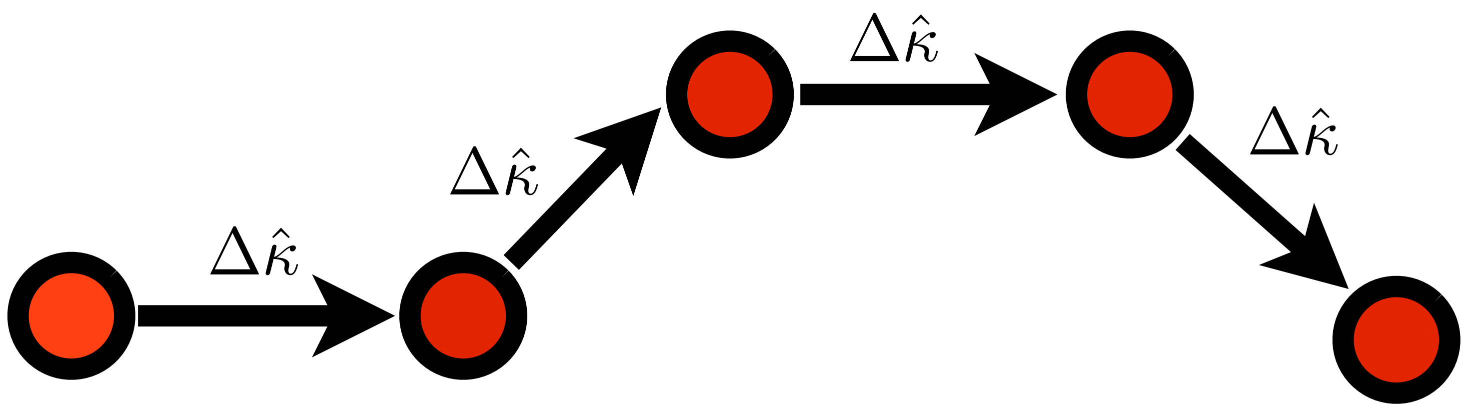
↓

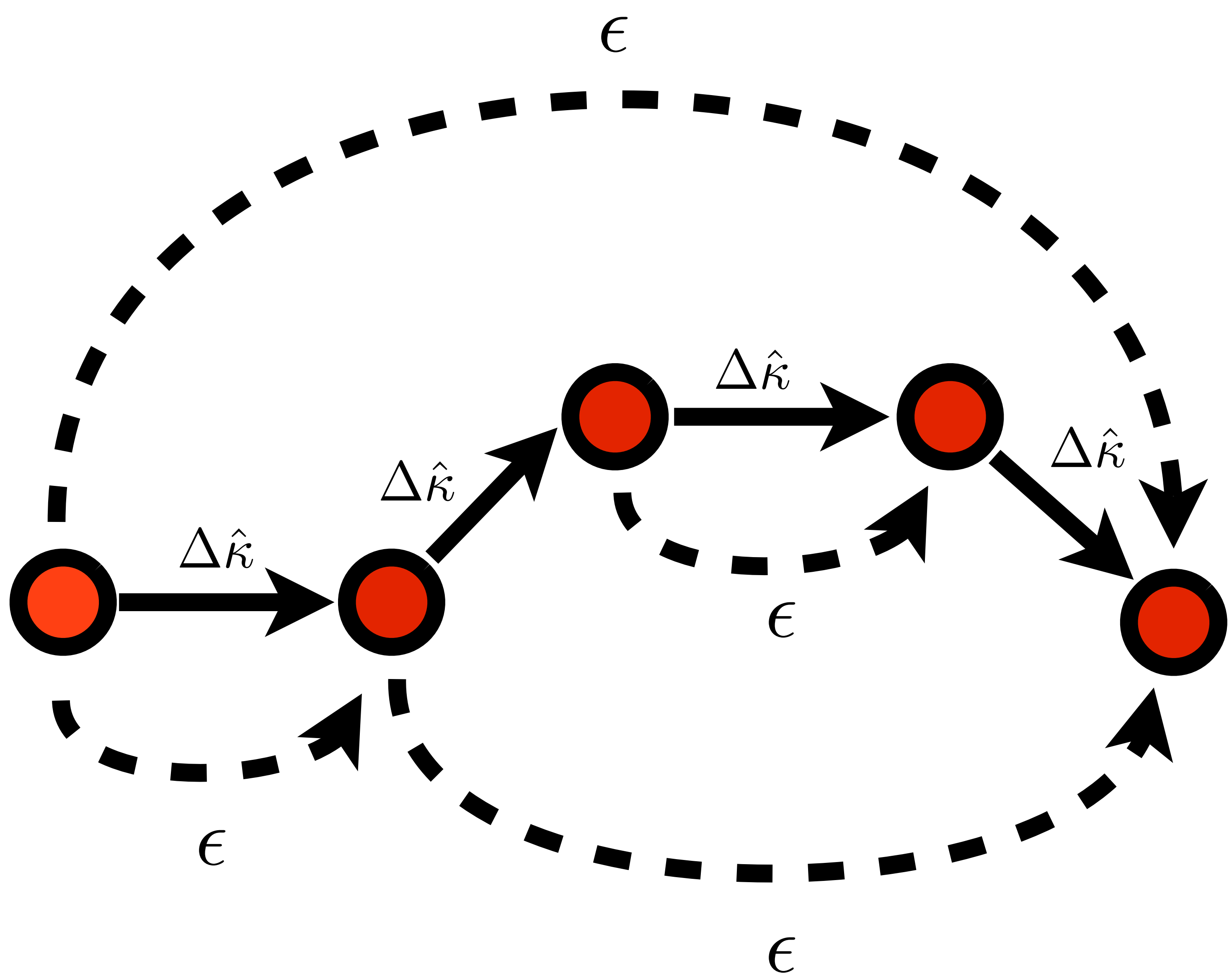
$$\delta : Q \times \Delta\Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q)$$

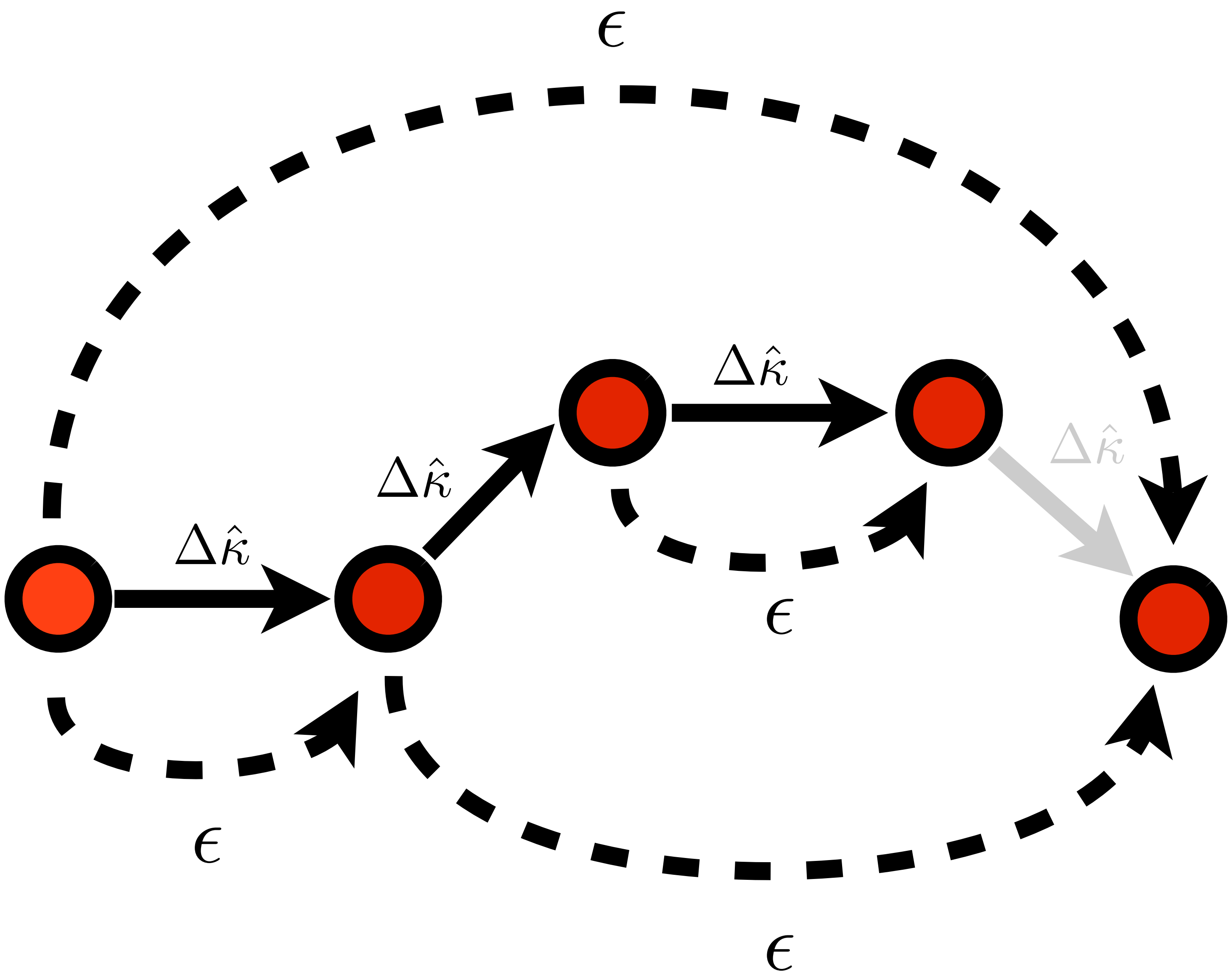
How do we fix the fixes?

$$\delta : Q \times \Delta\Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q)$$

$$\mathcal{P}(\Gamma^*)$$

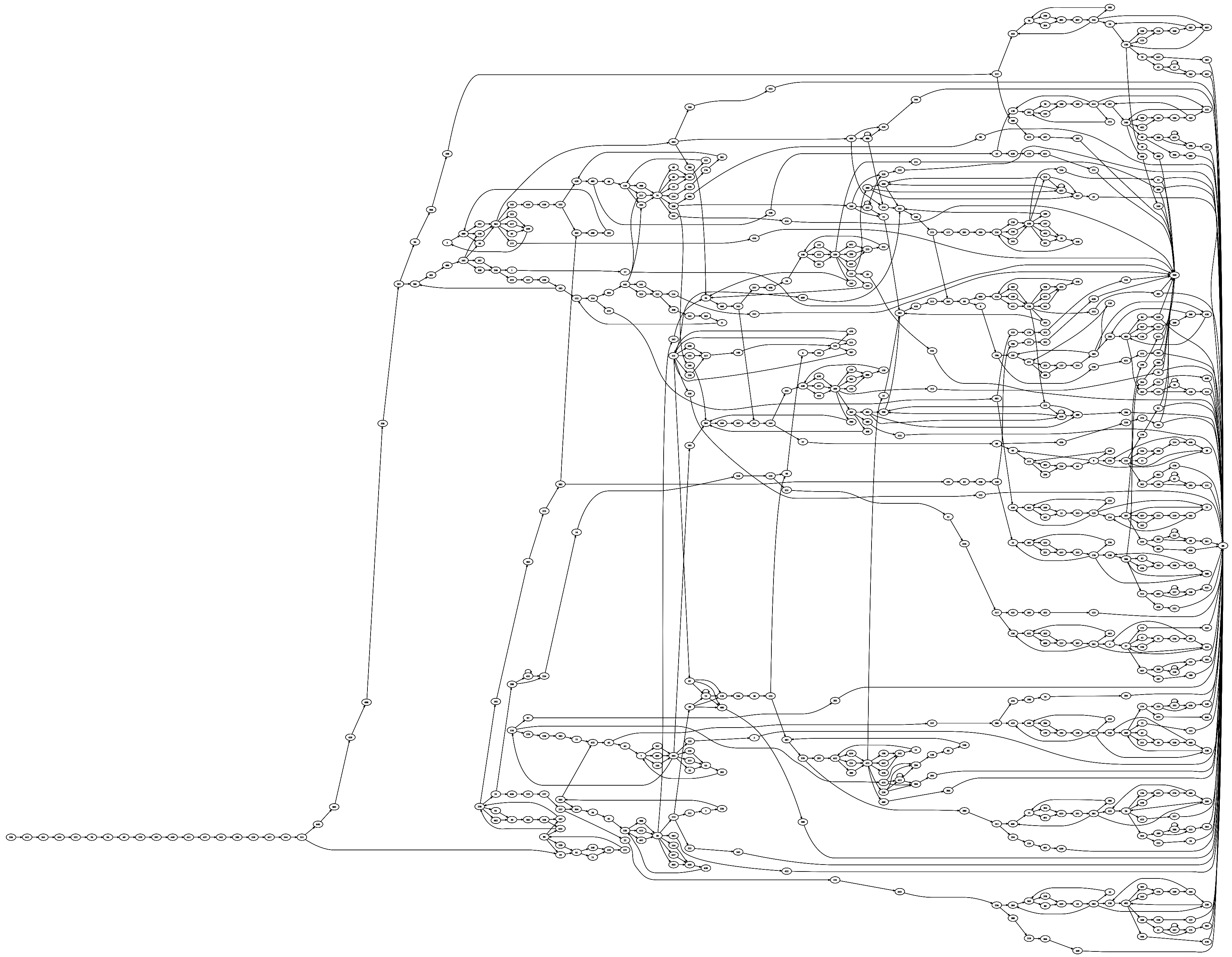

$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$

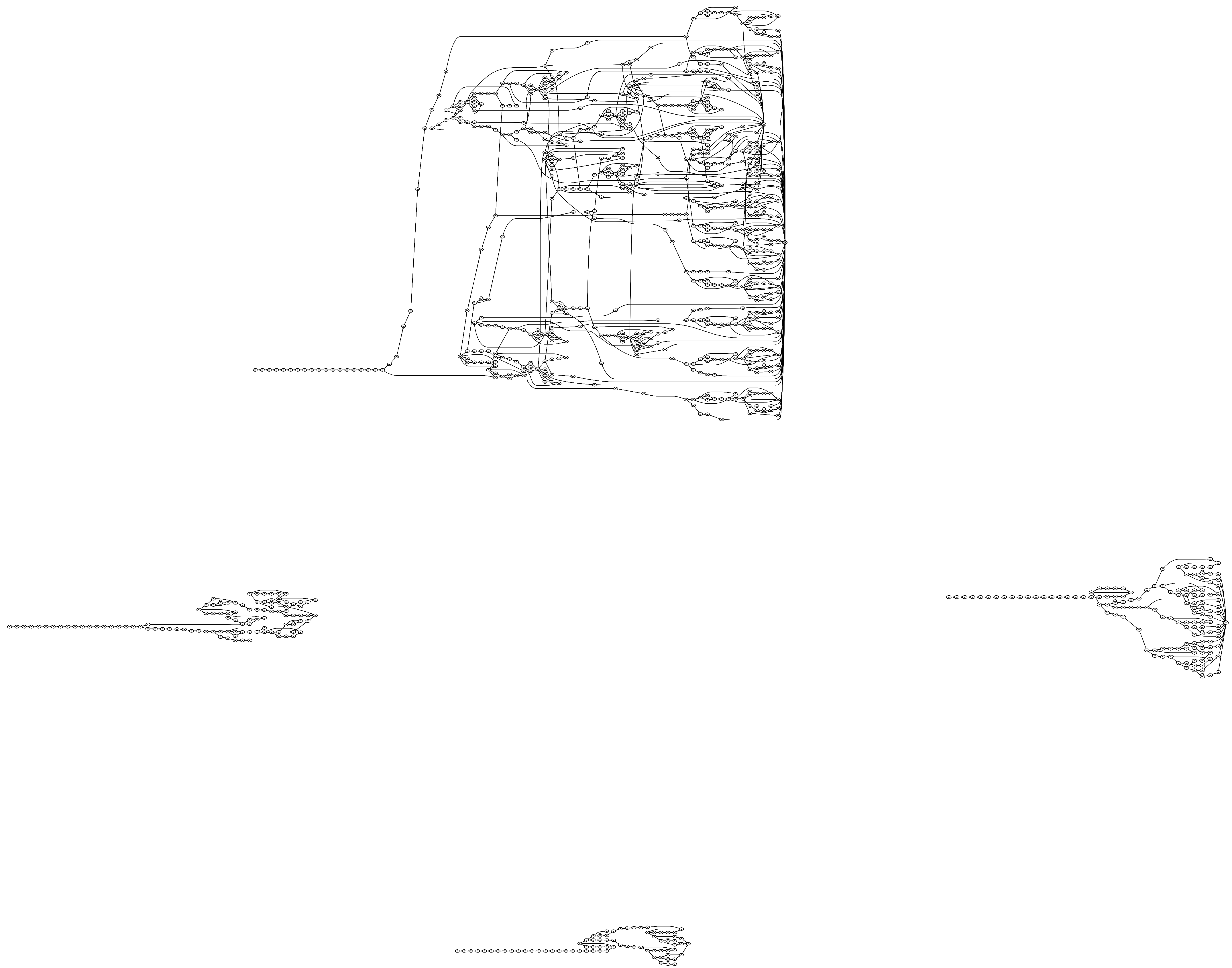

ϵ
Δκ̂
Δκ̂
Δκ̂
Δκ̂
ϵ
ϵ
ϵ

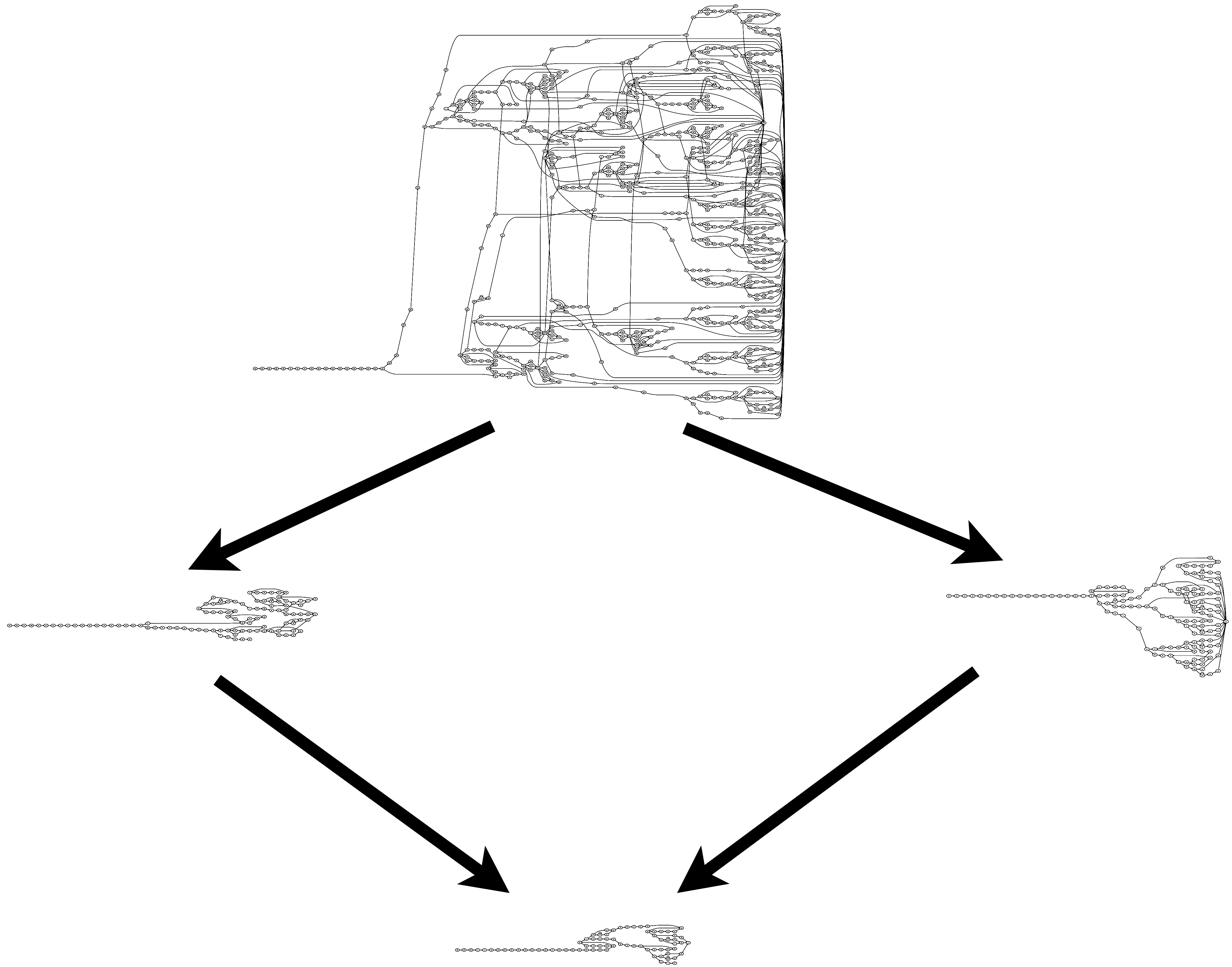

ϵ
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
$\Delta\hat{\kappa}$
ϵ
ϵ
ϵ

$$\mathcal{P}(\Gamma^*)$$

Result?







$$g \in \Gamma_{\pm} ::= \epsilon \mid \gamma_{+} \mid \gamma_{-}$$

$$(q, \vec{\gamma}) \underset{M}{\overset{\epsilon}{\longmapsto}} (q', \vec{\gamma}) \text{ iff } q \overset{\epsilon}{\rightarrowtail} q' \in \delta$$

$$(q, \gamma : \vec{\gamma}) \underset{M}{\overset{\gamma_{-}}{\longmapsto}} (q', \vec{\gamma}) \text{ iff } q \overset{\gamma_{-}}{\rightarrowtail} q' \in \delta$$

$$(q, \vec{\gamma}) \underset{M}{\overset{\gamma_{+}}{\longmapsto}} (q', \gamma : \vec{\gamma}) \text{ iff } q \overset{\gamma_{+}}{\rightarrowtail} q' \in \delta$$

$$\lfloor \vec{g}\, \gamma_{+} \gamma_{-}\, \vec{g}\,' \rfloor = \lfloor \vec{g}\, \vec{g}\,' \rfloor \qquad \lfloor \vec{g}\, \epsilon\, \vec{g}\,' \rfloor = \lfloor \vec{g}\, \vec{g}\,' \rfloor.$$

$$\lceil \gamma_{+} \gamma'_{+} \ldots \gamma^{(n)}_{+} \rceil = \langle \gamma^{(n)}, \ldots, \gamma', \gamma \rangle$$

$$c_0 \underset{M}{\overset{g_1 \ldots g_n}{\longmapsto\!\!\!\rightarrow}} c_n \text{ iff } c_0 \underset{M}{\overset{g_1}{\longmapsto}} c_1 \underset{M}{\overset{g_2}{\longmapsto}} \cdots \underset{M}{\overset{g_{n-1}}{\longmapsto}} c_{n-1} \underset{M}{\overset{g_n}{\longmapsto}} c_n$$

$$\left\{ q : q_0 \underset{M}{\longmapsto\!\!\!\rightarrow} q \right\}$$

$$c \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} c' \text{ iff } (q_0, \langle\rangle) \underset{M}{\overset{*}{\longmapsto}} c \text{ and } c \underset{M}{\overset{g}{\longmapsto}} c'.$$

$$q \underset{M}{\longmapsto\!\!\!\rightarrow} q' \text{ iff } q \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} q' \text{ for some action } g.$$

$$q \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} q' \text{ iff } (q, \vec{\gamma}) \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} (q', \vec{\gamma}\,') \text{ for some stacks } \vec{\gamma}, \vec{\gamma}\,'$$

$$\mathcal{A}(lam, \rho, \sigma) = (lam, \rho)$$
$$\mathcal{A}(v, \rho, \sigma) = \sigma(\rho(v))$$

$c \in Conf = \mathsf{Exp} \times Env \times Store \times Kont$
$\rho \in Env = \mathsf{Var} \rightharpoonup Addr$
$\sigma \in Store = Addr \to Clo$
$clo \in Clo = \mathsf{Lam} \times Env$
$\kappa \in Kont = Frame^{*}$
$\phi \in Frame = \mathsf{Var} \times \mathsf{Exp} \times Env$
$a \in Addr$ is an infinite set of addresses

$e \in \mathsf{Exp} ::= \texttt{(let ((}v\ call\texttt{))}\ e\texttt{)}$
$\quad\mid\ call$
$\quad\mid\ æ$
$f, æ \in \mathsf{Atom} ::= v \mid lam$
$lam \in \mathsf{Lam} ::= \texttt{(}\lambda\ \texttt{(}v\texttt{)}\ e\texttt{)}$
$call \in \mathsf{Call} ::= \texttt{(}f\ æ\texttt{)}$
$v \in \mathsf{Var}$ is a set of identifiers

$$\mathcal{E}(e) = \{c : \mathcal{I}(e) \Rightarrow^{*} c\}$$

$$\hat{c}_0 = \hat{\mathcal{I}}(e) = (e, [], [], \langle\rangle).$$

$$\overbrace{([\![\texttt{(let ((}v\ call\texttt{))}\ e\texttt{)}]\!], \rho, \sigma, \kappa)}^{c} \Rightarrow \overbrace{(call, \rho, \sigma, (v, e, \rho) : \kappa)}^{c'}.$$

$$Addr = \mathbb{N}$$
$$alloc(v, (e, \rho, \sigma, \kappa)) = 1 + \max(dom(\sigma))$$

$$\overbrace{([\![\texttt{(}f\ æ\texttt{)}]\!], \rho, \sigma, \kappa)}^{c} \Rightarrow \overbrace{(e, \rho'', \sigma', \kappa)}^{c'}, \text{ where}$$
$$([\![\texttt{(}\lambda\ \texttt{(}v\texttt{)}\ e\texttt{)}]\!], \rho') = \mathcal{A}(f, \rho, \sigma)$$
$$a = alloc(v, c)$$
$$\rho'' = \rho'[v \mapsto a]$$
$$\sigma' = \sigma[a \mapsto \mathcal{A}(æ, \rho, \sigma)].$$

$$\overbrace{(æ, \rho, \sigma, (v, e, \rho') : \kappa)}^{c} \Rightarrow \overbrace{(e, \rho'', \sigma', \kappa)}^{c'}, \text{ where}$$
$$a = alloc(v, c)$$
$$\rho'' = \rho'[v \mapsto a]$$
$$\sigma' = \sigma[a \mapsto \mathcal{A}(æ, \rho, \sigma)].$$

$$(\hat{\sigma} \sqcup \hat{\sigma}')(\hat{a}) = \hat{\sigma}(\hat{a}) \cup \hat{\sigma}'(\hat{a}).$$

$$\hat{\mathcal{E}}(e) = \left\{ \hat{c} : \hat{\mathcal{I}}(e) \leadsto^{*} \hat{c} \right\}$$

$$\hat{\mathcal{A}}(lam, \hat{\rho}, \hat{\sigma}) = \{(lam, \hat{\rho})\}$$
$$\hat{\mathcal{A}}(v, \hat{\rho}, \hat{\sigma}) = \hat{\sigma}(\hat{\rho}(v))$$

$$\overbrace{(æ, \hat{\rho}, \hat{\sigma}, (v, e, \hat{\rho}') : \hat{\kappa})}^{\hat{c}} \leadsto \overbrace{(e, \hat{\rho}'', \hat{\sigma}', \hat{\kappa})}^{\hat{c}'}, \text{ where}$$
$$\hat{a} = \widehat{alloc}(v, \hat{c})$$
$$\hat{\rho}'' = \hat{\rho}'[v \mapsto \hat{a}]$$
$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a} \mapsto \hat{\mathcal{A}}(æ, \hat{\rho}, \hat{\sigma})].$$

$\hat{c} \in \widehat{Conf} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Kont}$
$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \rightharpoonup \widehat{Addr}$
$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \to \mathcal{P}\left(\widehat{Clo}\right)$
$\widehat{clo} \in \widehat{Clo} = \mathsf{Lam} \times \widehat{Env}$
$\hat{\kappa} \in \widehat{Kont} = \widehat{Frame}^{*}$
$\hat{\phi} \in \widehat{Frame} = \mathsf{Var} \times \mathsf{Exp} \times \widehat{Env}$
$\hat{a} \in \widehat{Addr}$ is a *finite* set of addresses

$$\overbrace{([\![\texttt{(let ((}v\ call\texttt{))}\ e\texttt{)}]\!], \hat{\rho}, \hat{\sigma}, \hat{\kappa})}^{\hat{c}} \leadsto \overbrace{(call, \hat{\rho}, \hat{\sigma}, (v, e, \hat{\rho}) : \hat{\kappa})}^{\hat{c}'}.$$

$$\overbrace{([\![\texttt{(}f\ æ\texttt{)}]\!], \hat{\rho}, \hat{\sigma}, \hat{\kappa})}^{\hat{c}} \leadsto \overbrace{(e, \hat{\rho}'', \hat{\sigma}', \hat{\kappa})}^{\hat{c}'}, \text{ where}$$
$$([\![\texttt{(}\lambda\ \texttt{(}v\texttt{)}\ e\texttt{)}]\!], \hat{\rho}') \in \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$
$$\hat{a} = \widehat{alloc}(v, \hat{c})$$
$$\hat{\rho}'' = \hat{\rho}'[v \mapsto \hat{a}]$$
$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a} \mapsto \hat{\mathcal{A}}(æ, \hat{\rho}, \hat{\sigma})].$$

$$\widehat{Addr} = \mathsf{Var}$$
$$alloc(v, \hat{c}) = v.$$

$$\widehat{Addr} = \mathsf{Var} \times \mathsf{Exp}$$
$$alloc(v, (e, \hat{\rho}, \hat{\sigma}, \hat{\kappa})) = (v, e).$$

$$O((|\Gamma| m^2) \times (|\Gamma_{\pm}|^3 m^3)) = O(|\Gamma|^4 m^5).$$

$$\widehat{Addr} = \mathsf{Var} + \mathsf{Var} \times \mathsf{Exp}$$
$$alloc(v, ([\![\texttt{(}f\ æ\texttt{)}]\!], \hat{\rho}, \hat{\sigma}, \hat{\kappa})) = \begin{cases} (v, [\![\texttt{(}f\ æ\texttt{)}]\!]) & f \text{ is let-bound} \\ v & \text{otherwise.} \end{cases}$$

$$\widehat{Addr} = \mathsf{Var} \times \mathsf{Exp}^{k}$$
$$\widehat{alloc}(v, \langle (e_1, \hat{\rho}_1, \hat{\sigma}_1, \hat{\kappa}_1), \ldots \rangle) = (v, \langle e_1, \ldots, e_k \rangle).$$

$$\hat{\rho} \sqsubseteq \hat{\rho}' \text{ iff } \hat{\rho}(v) = \hat{\rho}'(v) \text{ for all } v \in dom(\hat{\rho});$$

$$\hat{\sigma} \sqsubseteq \hat{\sigma}' \text{ iff } \hat{\sigma}(\hat{a}) \sqsubseteq \hat{\sigma}'(\hat{a}) \text{ for all } \hat{a} \in dom(\hat{\sigma});$$

$$(lam, \hat{\rho}) \sqsubseteq (lam, \hat{\rho}') \text{ iff } \hat{\rho} \sqsubseteq \hat{\rho}'; \qquad (v, e, \hat{\rho}) \sqsubseteq (v, e, \hat{\rho}') \text{ iff } \hat{\rho} \sqsubseteq \hat{\rho}';$$

$$\langle \hat{\phi}_1, \ldots, \hat{\phi}_n \rangle \sqsubseteq \langle \hat{\phi}'_1, \ldots, \hat{\phi}'_n \rangle \text{ iff } \hat{\phi}_i \sqsubseteq \hat{\phi}'_i;$$

$$(e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}) \sqsubseteq (e, \hat{\rho}', \hat{\sigma}', \hat{\kappa}') \text{ iff } \hat{\rho} \sqsubseteq \hat{\rho}' \text{ and } \hat{\sigma} \sqsubseteq \hat{\sigma}' \text{ and } \hat{\kappa} \sqsubseteq \hat{\kappa}'.$$

$$\alpha(e, \rho, \sigma, \kappa) = (e, \alpha(\rho), \alpha(\sigma), \alpha(\kappa))$$
$$\alpha(\rho) = \lambda v. \alpha(\rho(v))$$
$$\alpha(\sigma) = \lambda \hat{a}. \bigsqcup_{\alpha(a) = \hat{a}} \{\alpha(\sigma(a))\}$$
$$\alpha \langle \phi_1, \ldots, \phi_n \rangle = \langle \alpha(\phi_1), \ldots, \alpha(\phi_n) \rangle$$
$$\alpha(v, e, \rho) = (v, e, \alpha(\rho))$$
$\alpha(a)$ is determined by the allocation functions.

Theorem 4.1. *If:*
$$\alpha(c) \sqsubseteq \hat{c} \text{ and } c \Rightarrow c',$$
then there must exist $\hat{c}' \in \widehat{Conf}$ *such that:*
$$\alpha(c') \sqsubseteq \hat{c}' \text{ and } \hat{c} \leadsto \hat{c}'.$$

$$\widehat{\mathcal{PDS}}(e) = (Q, \Gamma, \delta, q_0), \text{ where}$$
$$Q = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store}$$
$$\Gamma = \widehat{Frame}$$
$$(q, \epsilon, q') \in \delta \text{ iff } (q, \hat{\kappa}) \leadsto (q', \hat{\kappa}) \text{ for all } \hat{\kappa}$$
$$(q, \hat{\phi}_{-}, q') \in \delta \text{ iff } (q, \hat{\phi} : \hat{\kappa}) \leadsto (q', \hat{\kappa}) \text{ for all } \hat{\kappa}$$
$$(q, \hat{\phi}'_{+}, q') \in \delta \text{ iff } (q, \hat{\kappa}) \leadsto (q', \hat{\phi}' : \hat{\kappa}) \text{ for all } \hat{\kappa}$$
$$(q_0, \langle\rangle) = \hat{\mathcal{I}}(e).$$

$$\hat{G}(\overbrace{e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}}^{\hat{c}}) = (e, \hat{\rho}, \hat{\sigma} | Reachable(\hat{c}), \hat{\kappa}),$$

$$Reachable(\overbrace{e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}}^{\hat{c}}) = \left\{ \hat{a} : \hat{a}_0 \in Root(\hat{c}) \text{ and } \hat{a}_0 \underset{\hat{\sigma}}{\overset{*}{\multimap}} \hat{a} \right\}$$

$$Root(e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}) = range(\hat{\rho}) \cup StackRoot(\hat{\kappa}).$$

$$StackRoot \langle (v_1, e_1, \hat{\rho}_1), \ldots, (v_n, e_n, \hat{\rho}_n) \rangle = \bigcup_{i} range(\hat{\rho}_i),$$

$$\hat{a} \underset{\hat{\sigma}}{\multimap} \hat{a}' \text{ iff there exists } (lam, \hat{\rho}) \in \hat{\sigma}(\hat{a}) \text{ such that } \hat{a}' \in range(\hat{\rho}).$$

$$(\leadsto_{\mathrm{GC}}) = (\leadsto) \circ \hat{G}$$

$$q \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} q' \text{ iff there exists } \vec{g} \text{ such that } q_0 \underset{M}{\overset{\vec{g}}{\longmapsto\!\!\!\rightarrow}} q \text{ and } (q, [\vec{g}], g, q') \in \delta,$$
$$\text{where } q \underset{M}{\overset{\langle g_1, \ldots, g_n \rangle}{\longmapsto\!\!\!\rightarrow}} q' \text{ iff } q \underset{M}{\overset{g_1}{\longmapsto\!\!\!\rightarrow}} q_1 \underset{M}{\overset{g_2}{\longmapsto\!\!\!\rightarrow}} \cdots \underset{M}{\overset{g_n}{\longmapsto\!\!\!\rightarrow}} q'.$$

$$q_0 \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} q \text{ iff } (q_0, \langle\rangle, g, q) \in \delta.$$

$$S = \left\{ q : q_0 \underset{M}{\overset{\vec{g}}{\longmapsto\!\!\!\rightarrow}} q \text{ for some stack-action sequence } \vec{g} \right\}$$

$$E = \left\{ q \overset{g}{\rightarrowtail} q' : q \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} q' \right\}$$

$$\widehat{\mathcal{IPDS}}(e) = (Q, \Gamma, \delta, q_0)$$
$$Q = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store}$$
$$\Gamma = \widehat{Frame}$$
$$(q, \hat{\kappa}, \epsilon, q') \in \delta \text{ iff } \hat{G}(q, \hat{\kappa}) \leadsto (q', \hat{\kappa})$$
$$(q, \hat{\phi} : \hat{\kappa}, \hat{\phi}_{-}, q') \in \delta \text{ iff } \hat{G}(q, \hat{\phi} : \hat{\kappa}) \leadsto (q', \hat{\kappa})$$
$$(q, \hat{\kappa}, \hat{\phi}_{+}, q') \in \delta \text{ iff } \hat{G}(q, \hat{\kappa}) \leadsto (q', \hat{\phi} : \hat{\kappa})$$
$$(q_0, \langle\rangle) = \hat{\mathcal{I}}(e).$$

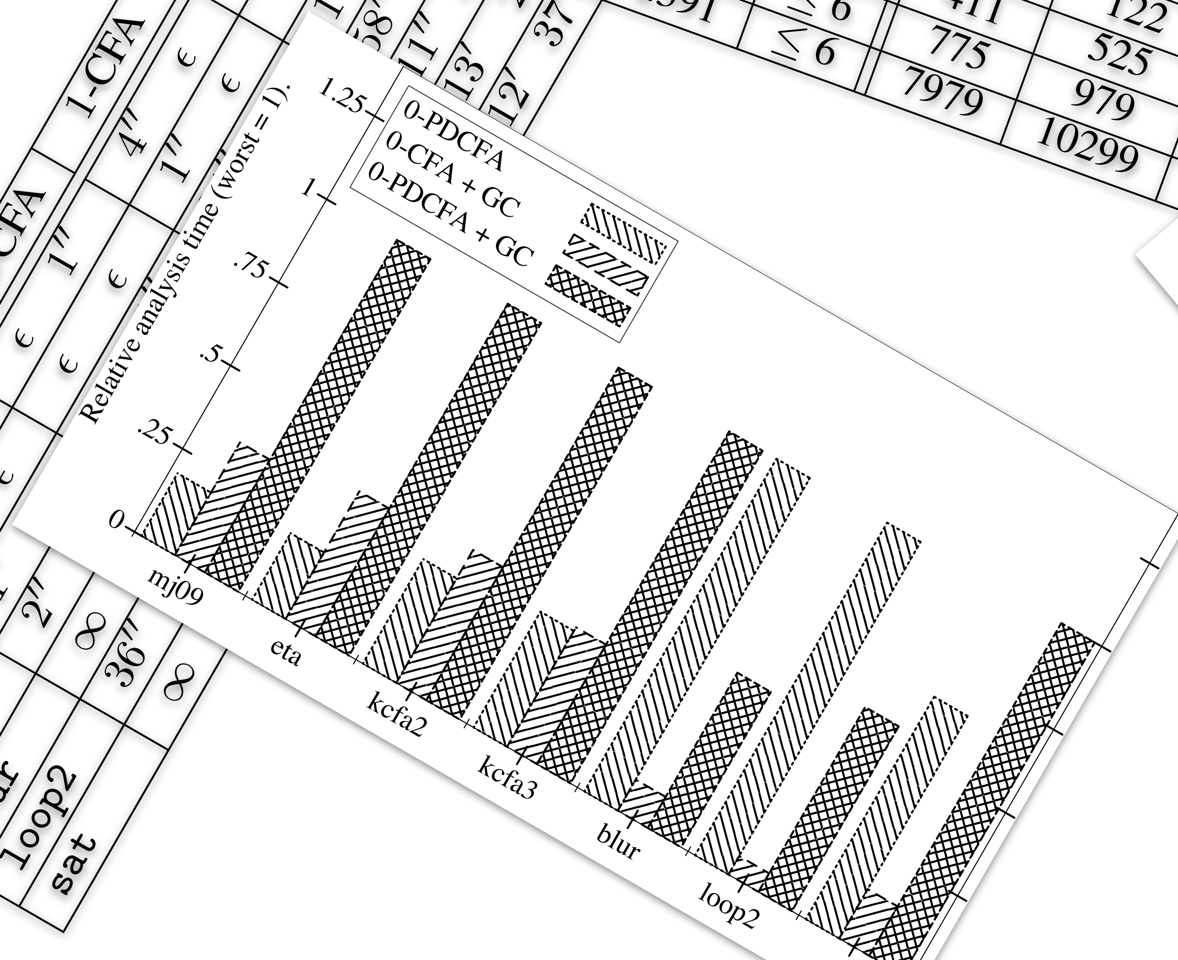
$$Stacks(\overbrace{S, \Gamma, E, s_0}^{M})(s) = (S, \Gamma, \delta, s_0, \{s\}), \text{ where}$$
$$(s', \gamma, s'') \in \delta \text{ if } (s', \gamma_{+}, s'') \in E$$
$$(s', \epsilon, s'') \in \delta \text{ if } s' \underset{M}{\overset{\vec{g}}{\longmapsto\!\!\!\rightarrow}} s'' \text{ and } [\vec{g}] = \epsilon.$$

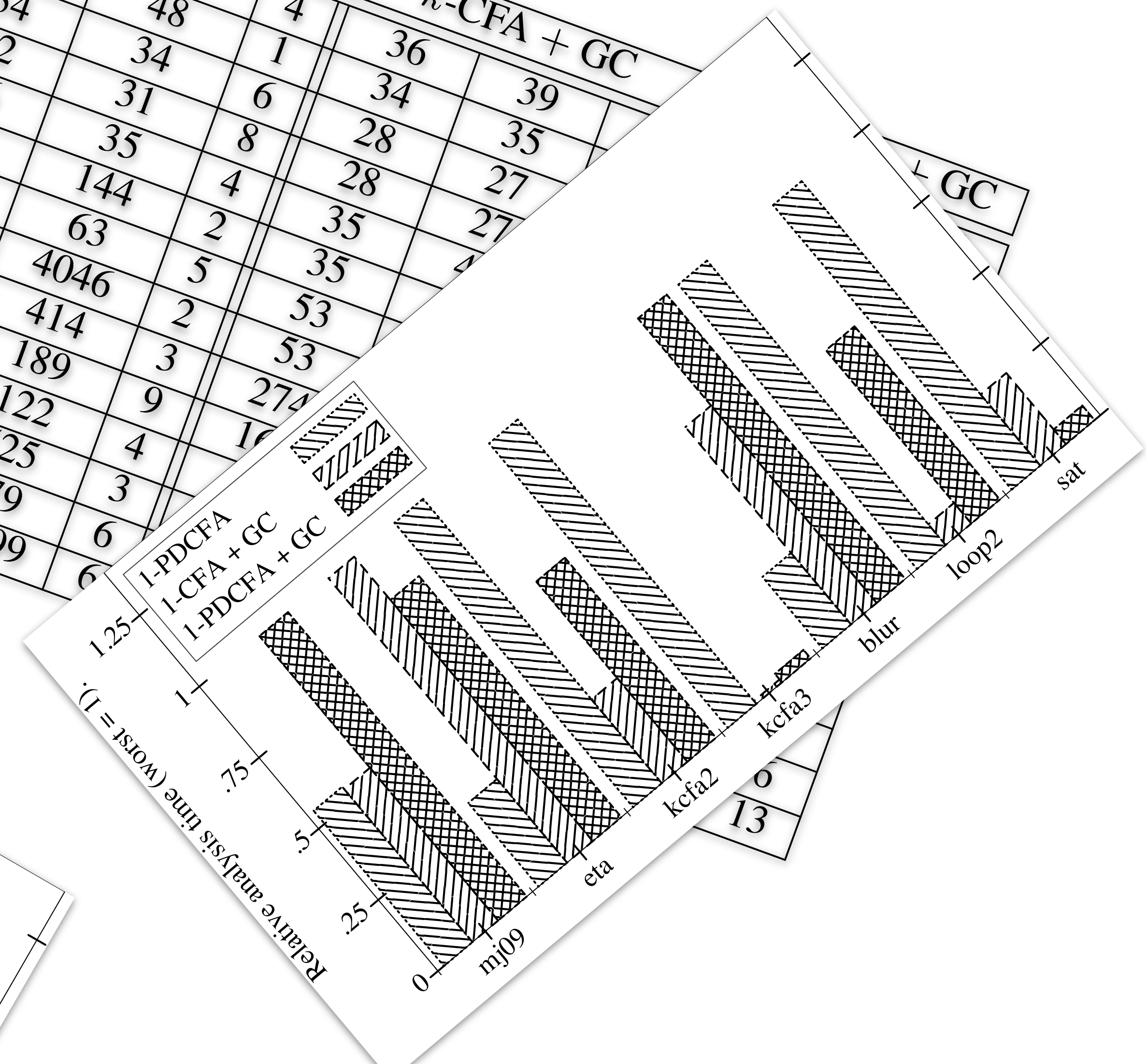
$$\mathcal{F}(M) = f, \text{ where}$$
$$M = (Q, \Gamma, \delta, q_0)$$
$$f(S, \Gamma, E, s_0) = (S', \Gamma, E', s_0), \text{ where}$$
$$S' = S \cup \left\{ s' : s \in S \text{ and } s \underset{M}{\longmapsto\!\!\!\rightarrow} s' \right\} \cup \{s_0\}$$
$$E' = E \cup \left\{ s \overset{g}{\rightarrowtail} s' : s \in S \text{ and } s \underset{M}{\overset{g}{\longmapsto\!\!\!\rightarrow}} s' \right\}.$$

Theorem 7.1. $\mathcal{DSG}(M) = \mathrm{lfp}(\mathcal{F}(M)).$

Program	Exp	Var	k	k-CFA			k-PDCFA			k-CFA + GC			k-PDCFA + GC		
mj09	19	8	0	83	107	4	38	38	4	36	39	[illegible]	[illegible]	[illegible]	[illegible]
			1	454	812	1	44	48	1	34	35	[illegible]	[illegible]	[illegible]	[illegible]
eta	21	13	0	63	74	4	34	34	6	28	35	[illegible]	[illegible]	[illegible]	[illegible]
			1	33	33	8	32	31	8	28	27	[illegible]	[illegible]	[illegible]	[illegible]
kcfa2	20	10	0	194	236	3	36	35	4	35	27	[illegible]	[illegible]	[illegible]	[illegible]
			1	970	1935	1	87	144	2	35	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
kcfa3	25	13	0	272	327	4	58	63	5	53	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
			1	> 7119	> 14201	≤ 1	1761	4046	2	53	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
blur	40	20	0	> 1419	> 2435	≤ 3	280	414	3	274	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
			1	261	340	9	177	189	9	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
loop2	41	14	0	228	252	4	113	122	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
			1	> 10867	> 16040	≤ 3	411	525	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
sat	63	31	0	> 5362	> 7610	≤ 6	775	979	6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6
			1	> 8395	> 12391	≤ 6	7979	10299	6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	13

Program	0-CFA		0-PDCFA		1-CFA		1-PDCFA	
mj09	1″	ϵ	ϵ	1″	4″	ϵ	ϵ	ϵ
eta	ϵ	ϵ	ϵ	ϵ	1″	ϵ	ϵ	ϵ
kcfa2	1″	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1″	ϵ
kcfa3	2″	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	58″	2″
blur	∞	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	11″	55″
loop2	36″	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	13′	2′
sat	∞	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12′	37″





http://github.com/ilyasergey/reachability http://github.com/ilyasergey/reachability

CRAPL

Finitization is double-edged.

Progress attacks finitization.

We can skirt inside decidability.

Tak.

Complexity?

- Monovariant, global store? Polynomial.
- Polyvariant? Exponential.
- Flat environments, global store? Polynomial.

Alternative?

$$\delta : Q \times \Delta\Gamma \times \mathcal{P}(\Gamma^*) \rightarrow \mathcal{P}(Q)$$

$$\delta : Q \times \Delta\Gamma \times \mathcal{P}(\Delta\Gamma) \rightarrow \mathcal{P}(Q)$$

$$\delta : Q \times \mathcal{P}(\Delta\Gamma) \times \Delta\Gamma \to \mathcal{P}(Q)$$

$$\delta : \overbrace{Q \times \mathcal{P}(\Delta\Gamma)}^{\text{control states}} \times \Delta\Gamma \rightarrow \mathcal{P}(Q)$$