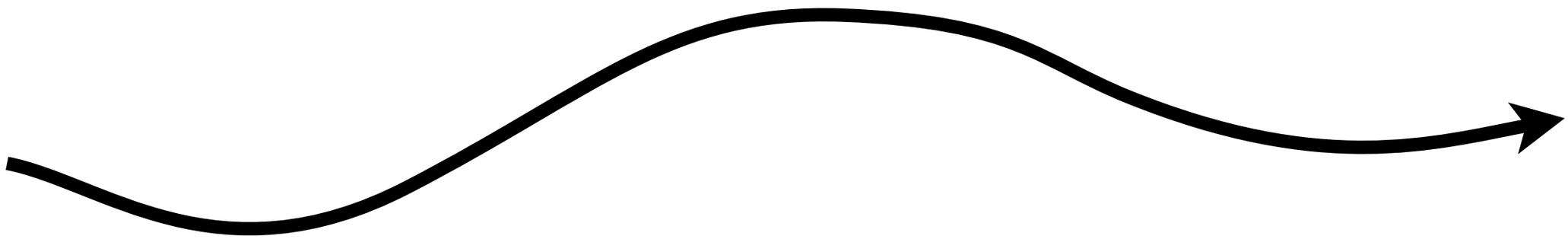
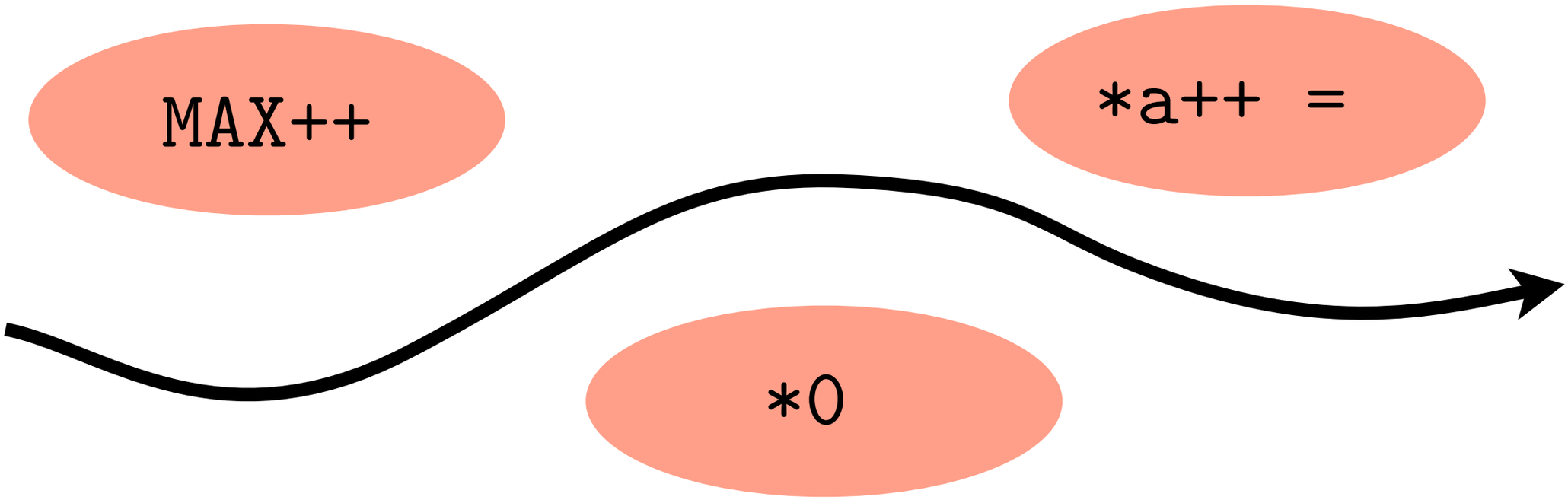
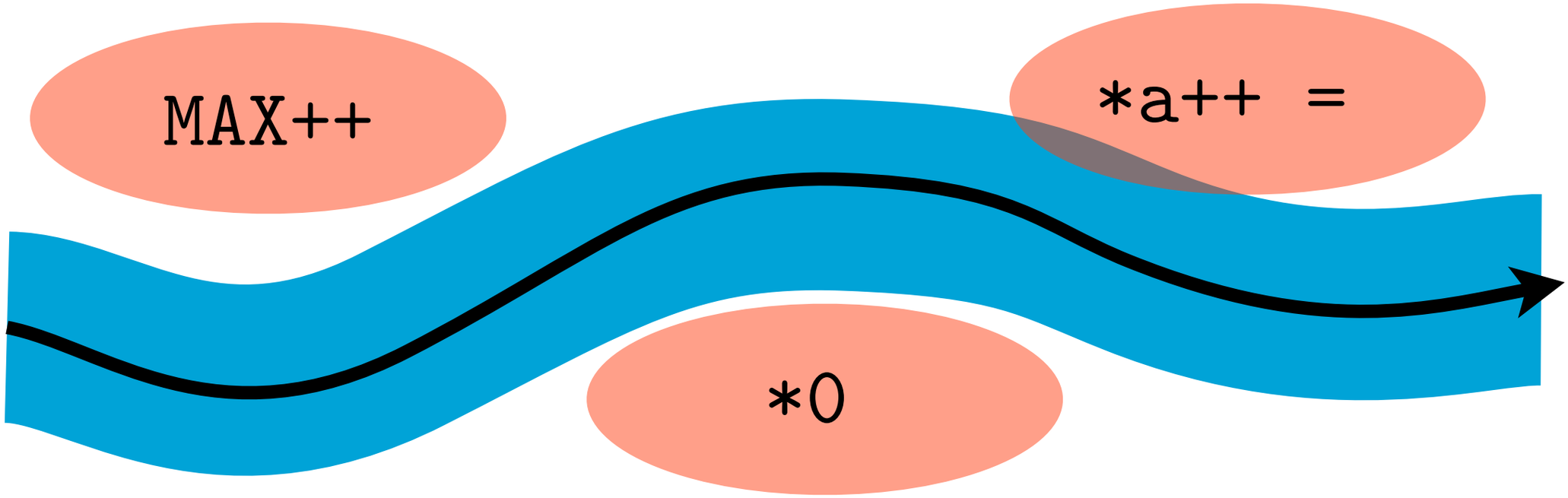
Deriving Abstract Interpreters

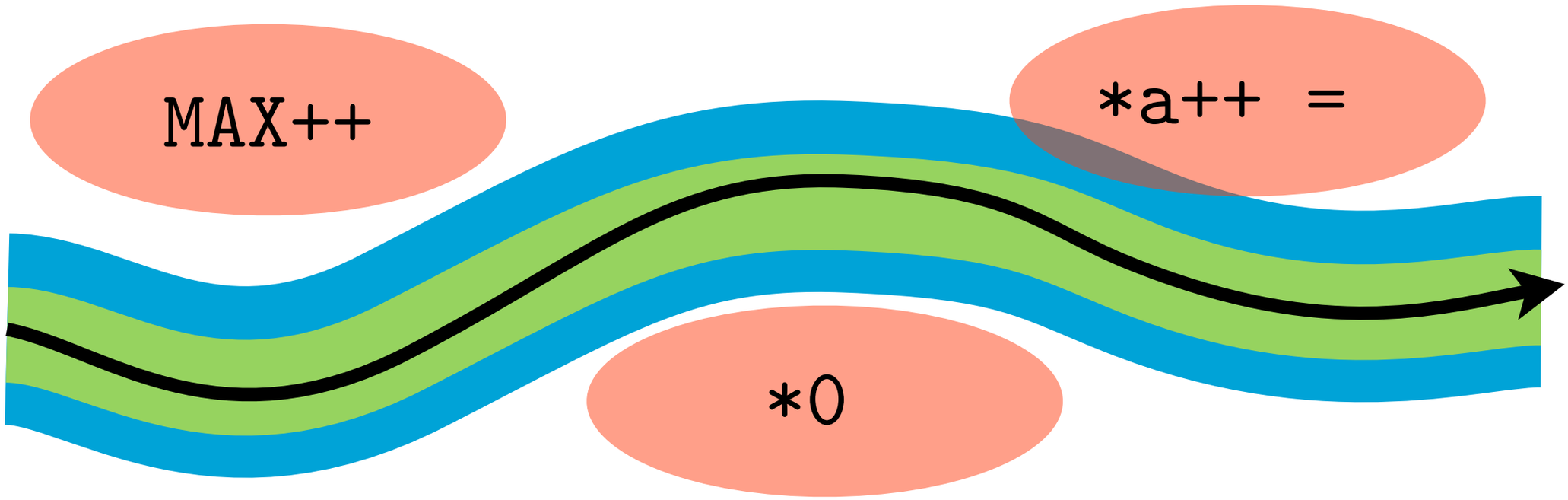
Matt Might
University of Utah
matt.might.net
@mattmight

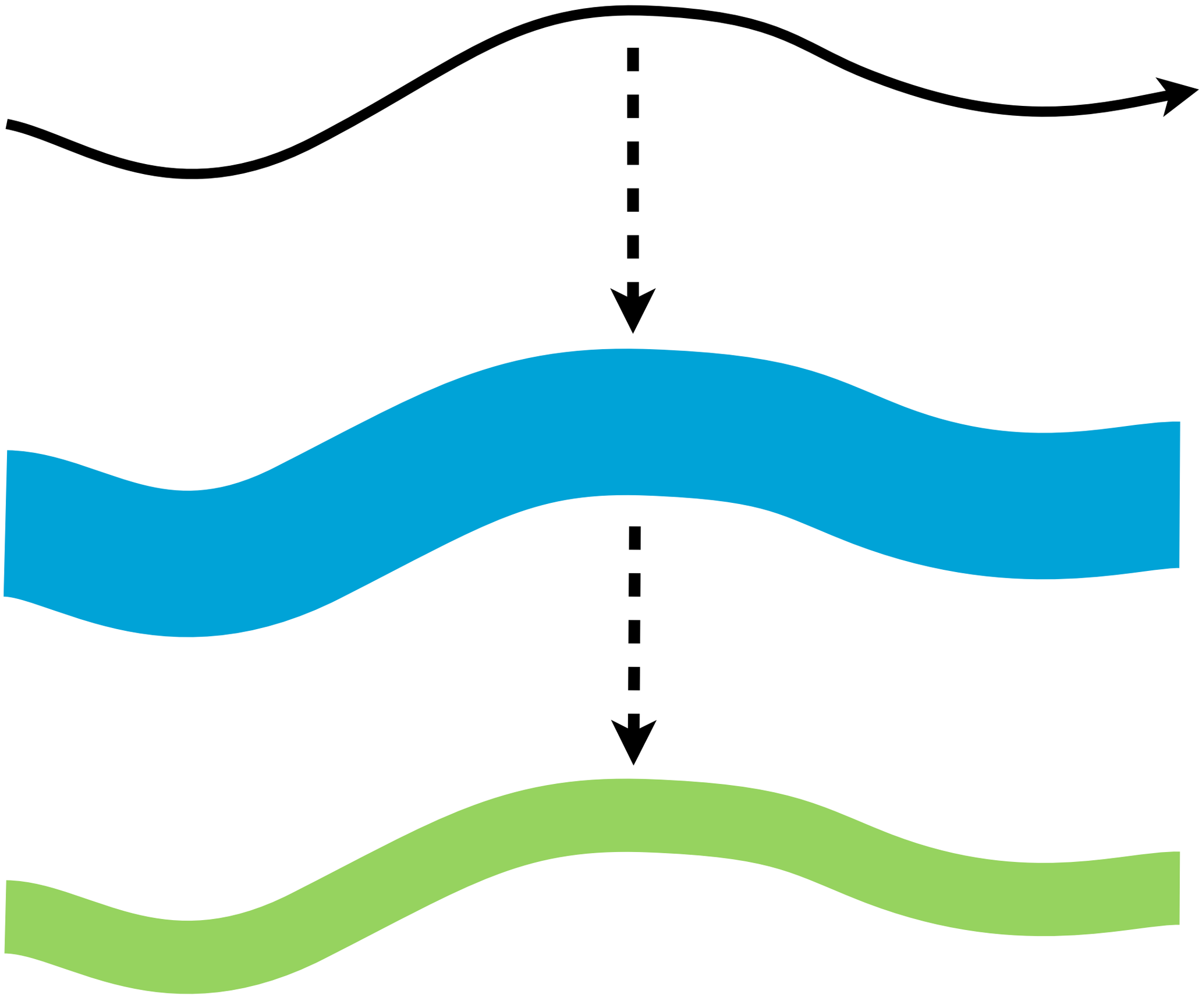
static analysis




MAX++
*a++ =
*0


MAX++
*a++ =
*0


MAX++
*a++ =
*0



How do you teach this?

f

$\hat{f}$

g

$\hat{g}$

Years later...

h

h

$\hat{h}$

Madness

to this

Method

Method

to this

Madness

Outline

- Small steps
- CPS, 0CFA
- ANF, PDFA

Small-step analysis

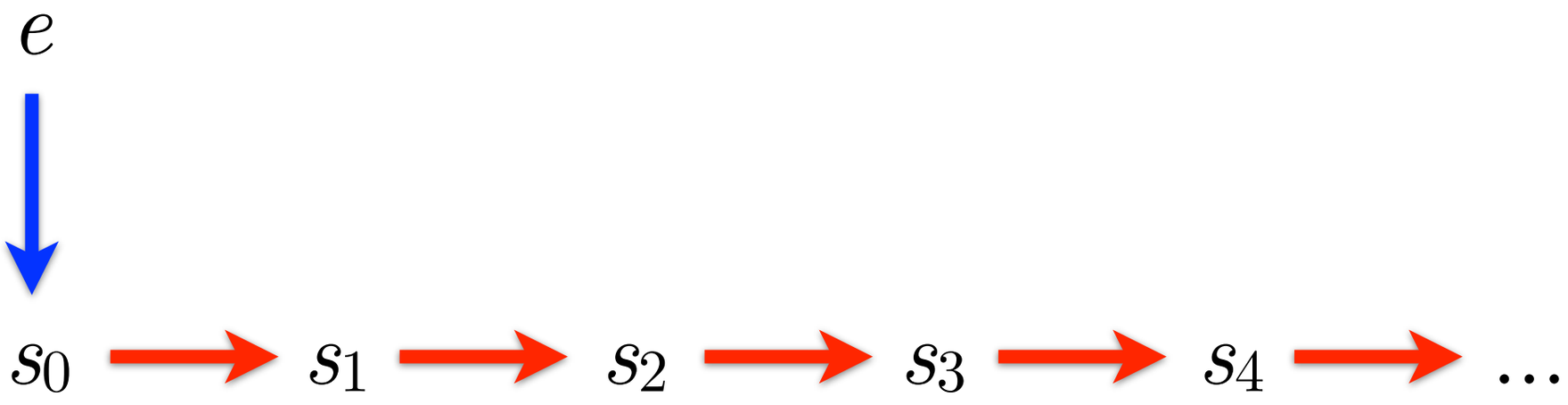
Concrete semantics

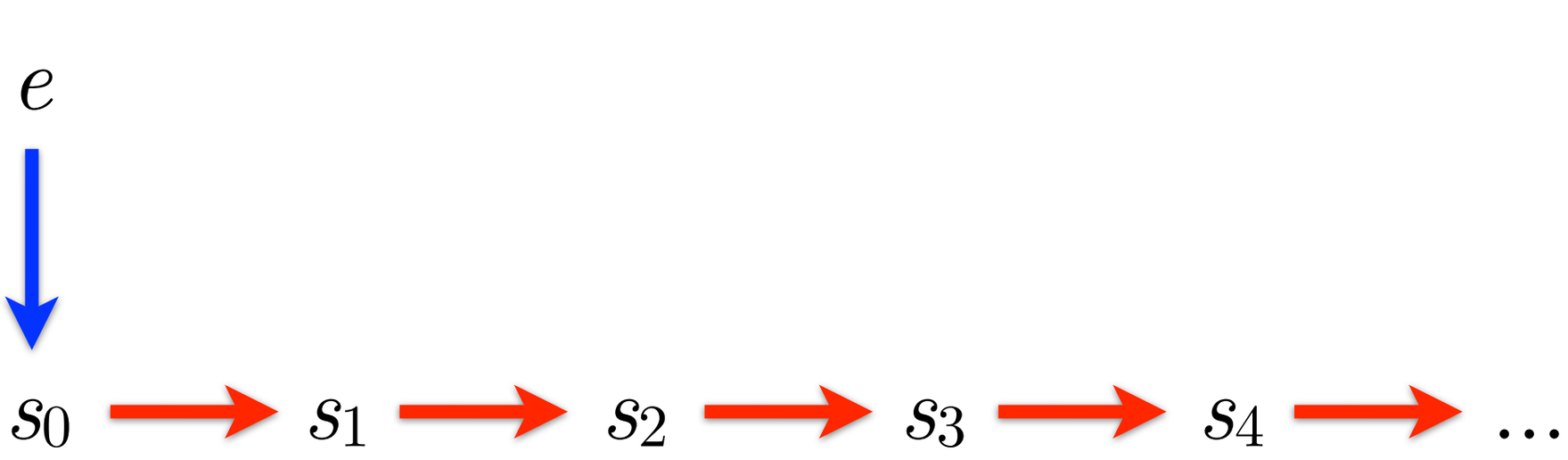
Concrete semantics

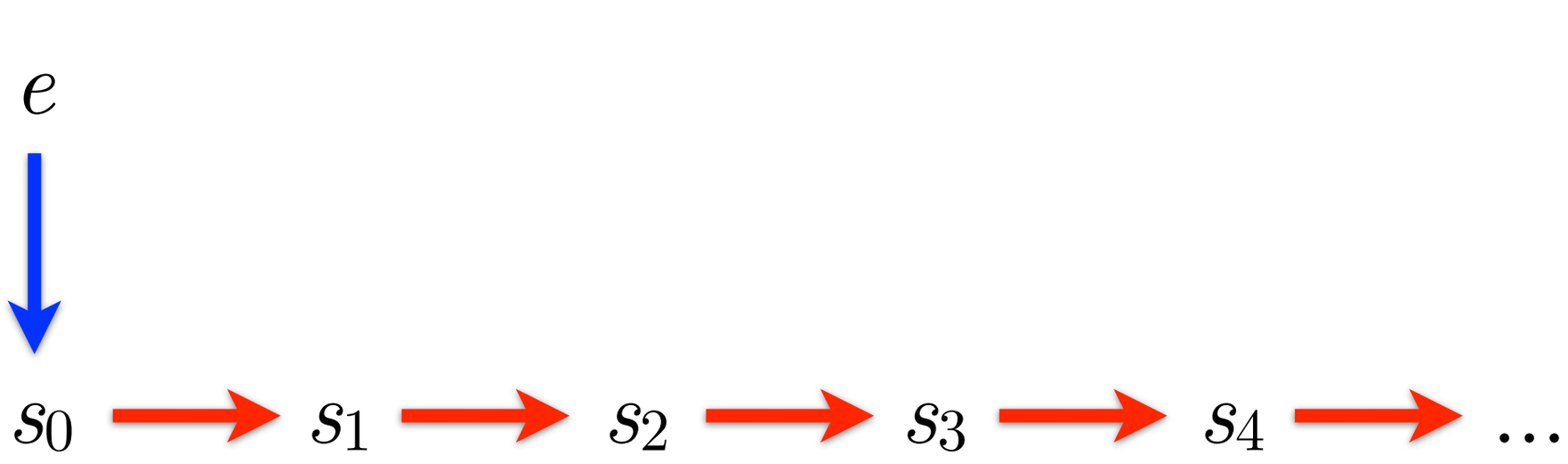
- Convert program e into machine state s_0

Concrete semantics

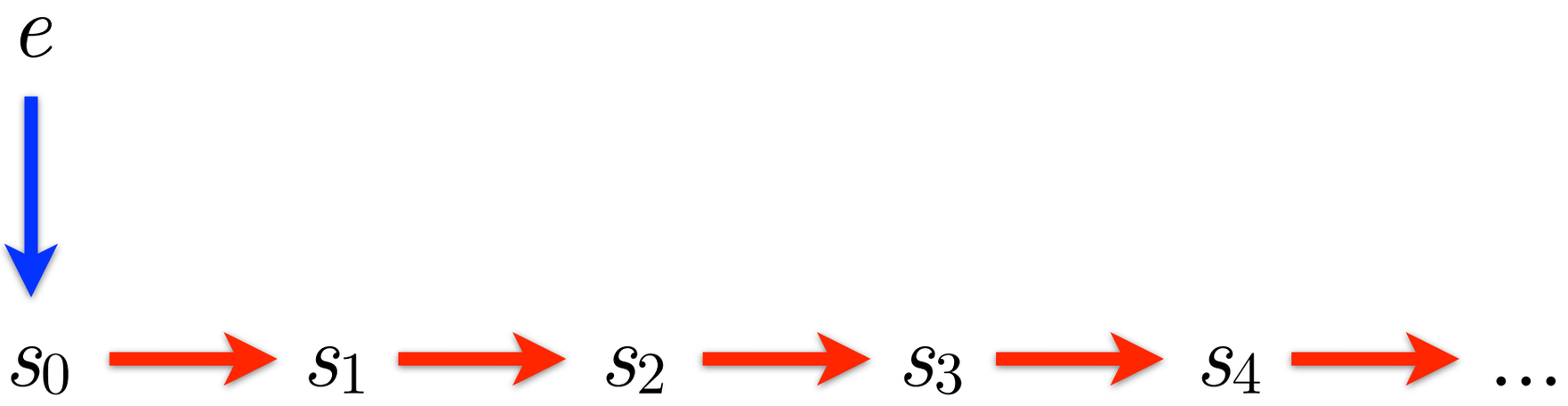
- Convert program e into machine state s_0
- Transition from state s_n to state s_{n+1}



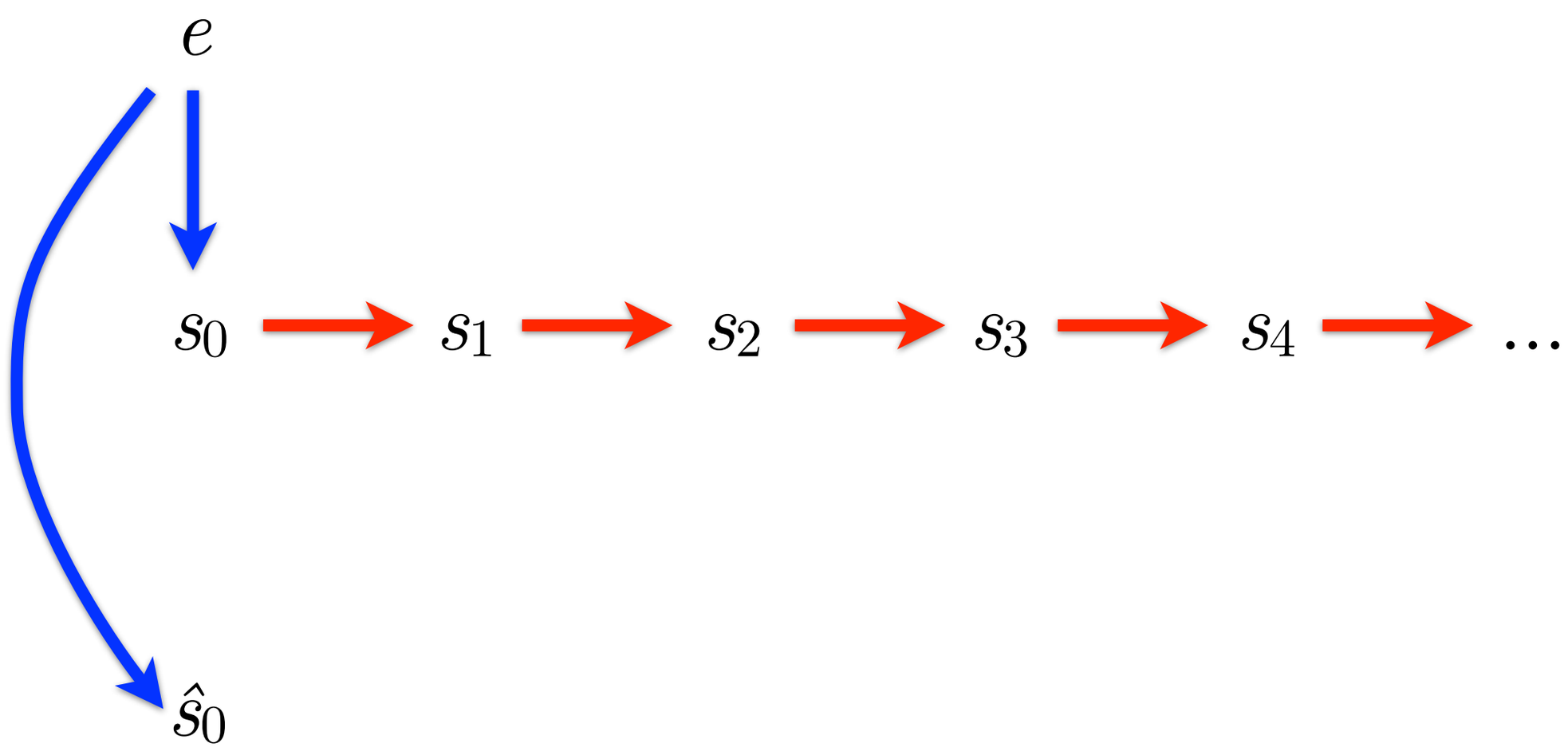

e
s_0
s_1
s_2
s_3
s_4
...


e
s_0
s_1
s_2
s_3
s_4
$\ldots$

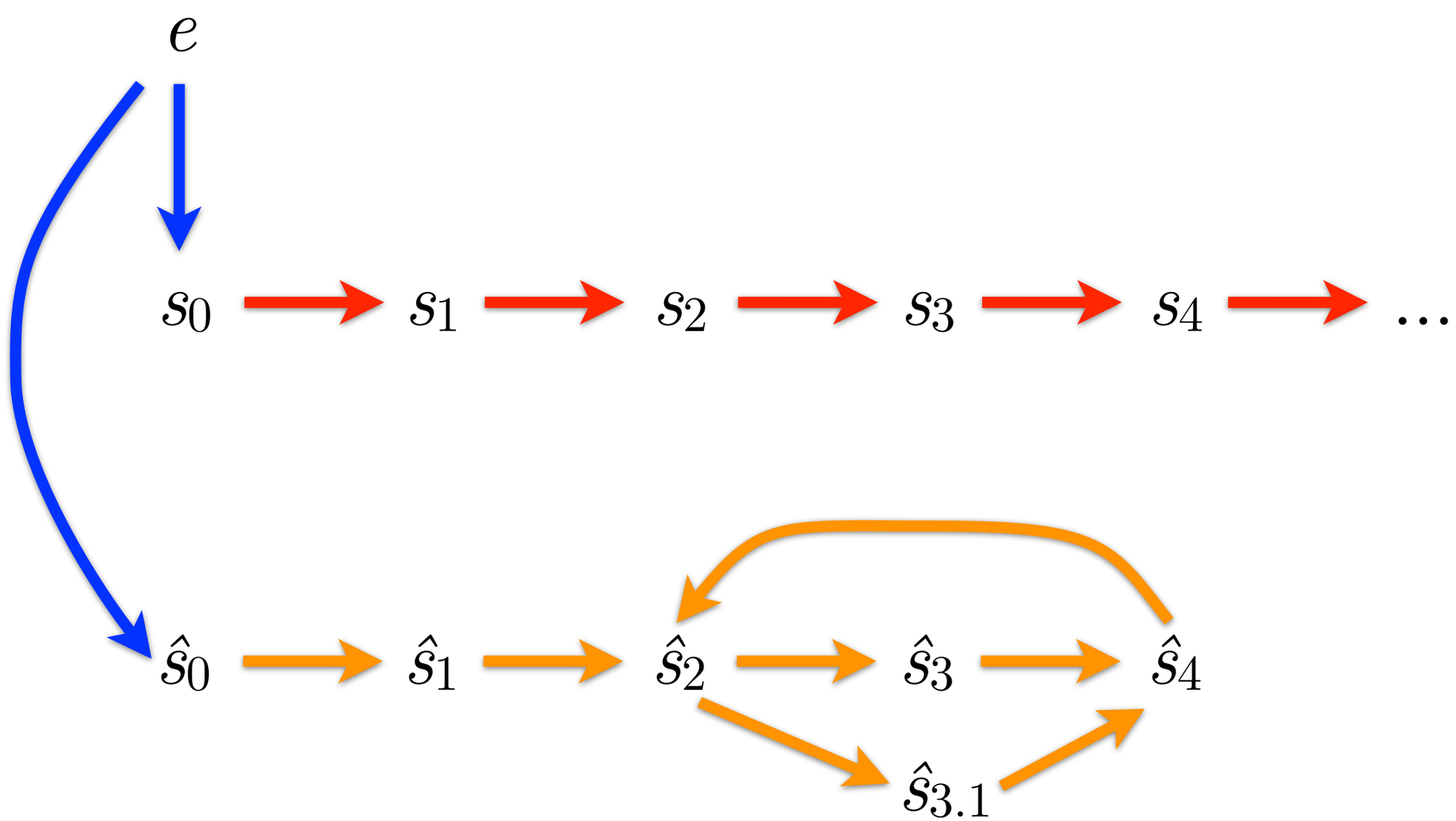
Abstract semantics



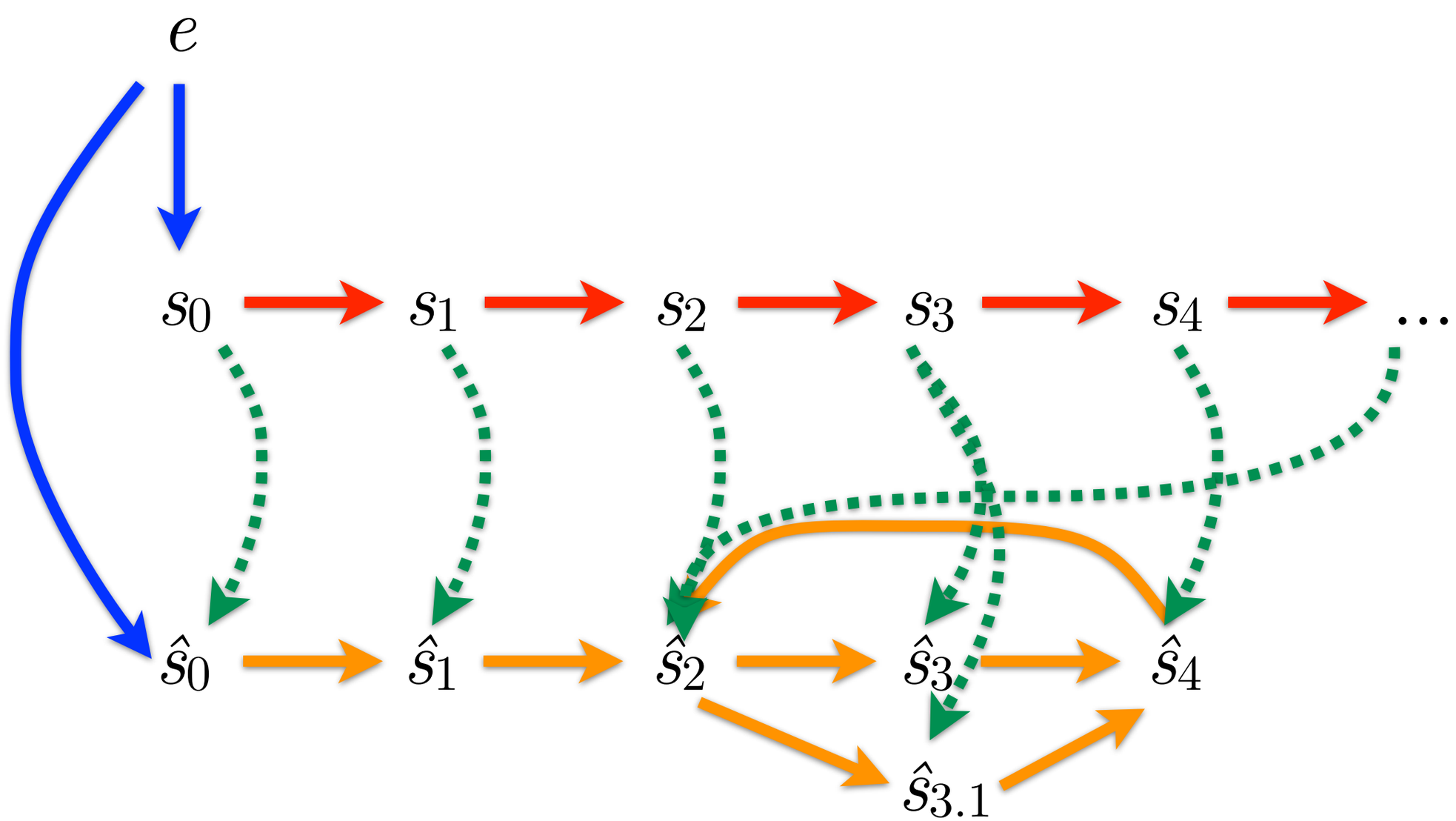
Abstract semantics



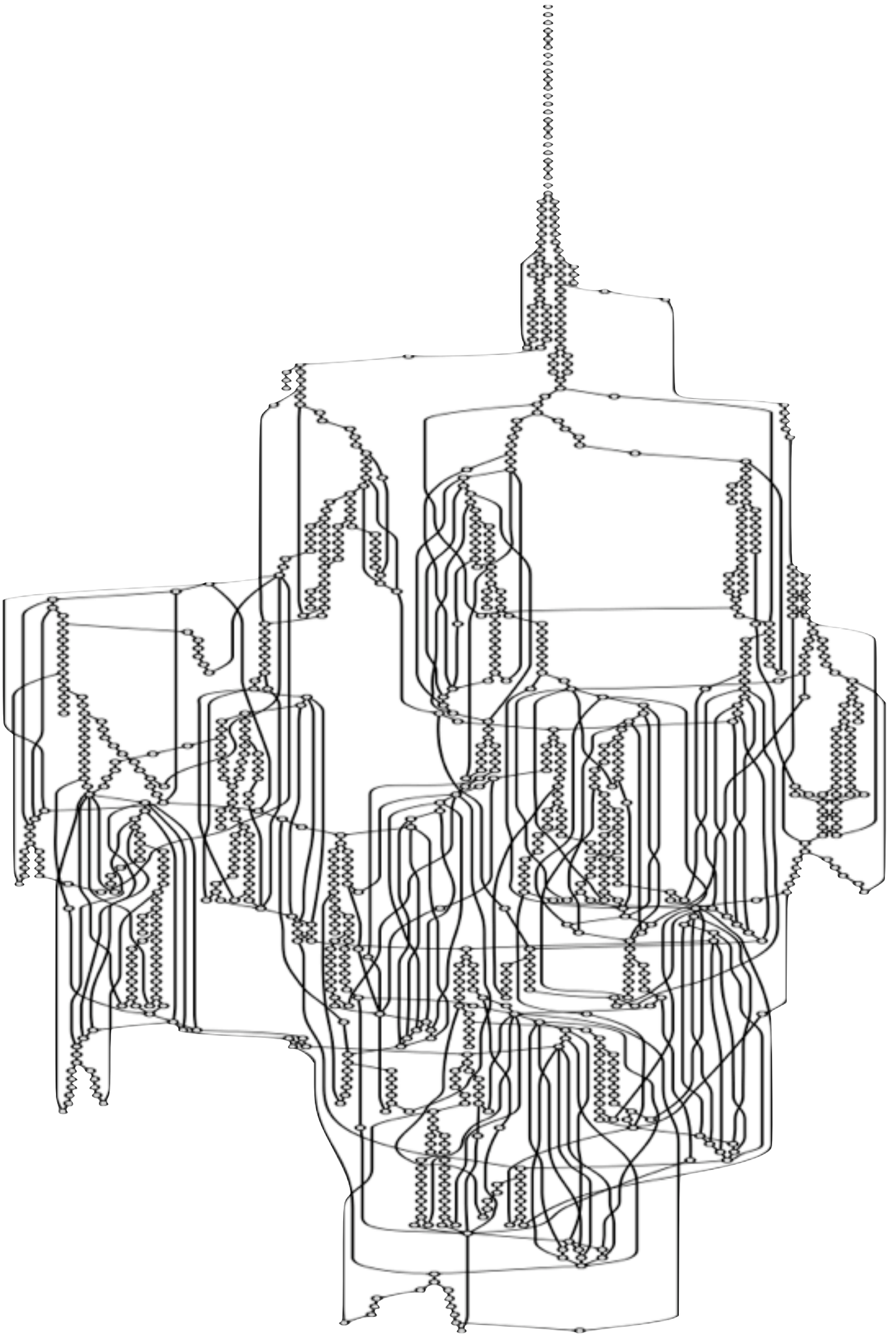
Abstract semantics



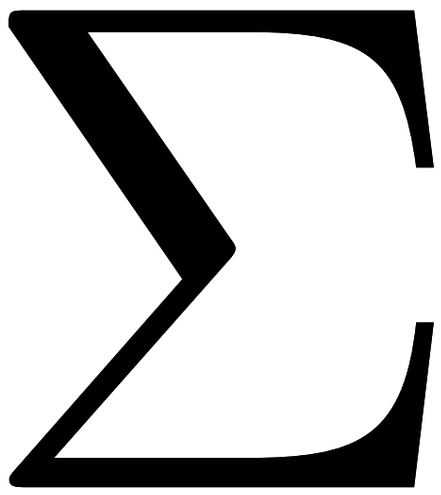
Abstract semantics



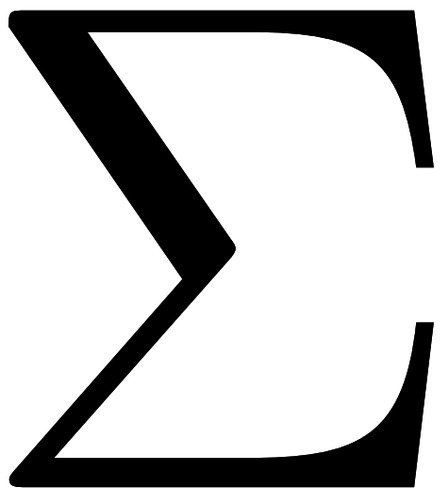
Theorem: The abstract simulates the concrete.



Components of small-step



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



Σ

$$\alpha : \sum \to \hat{\sum}$$

$\hat{\Sigma}$

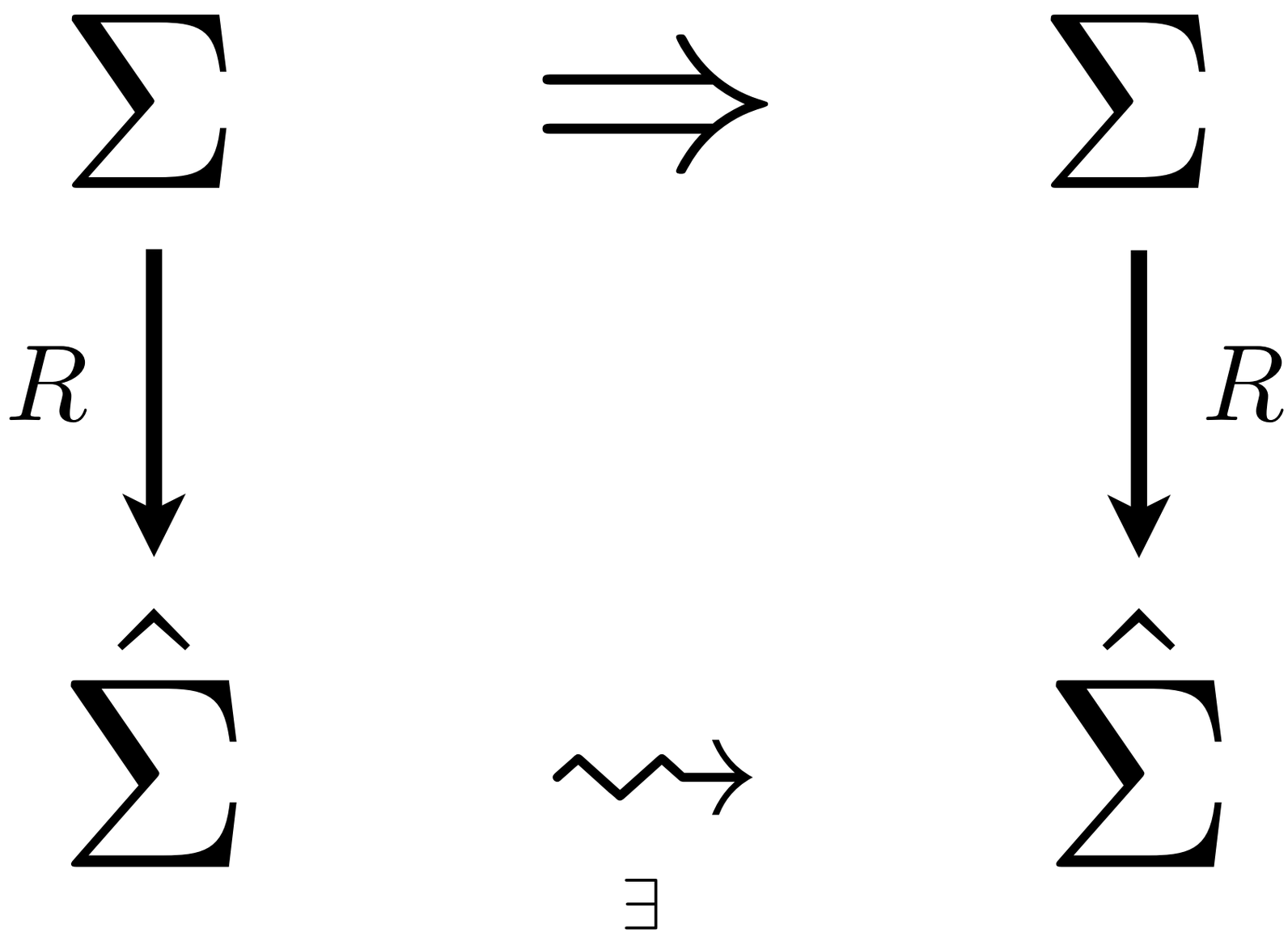
$\hat{\Sigma}$

$$(\rightsquigarrow) \subset \hat{\Sigma} \times \hat{\Sigma}$$

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

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

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


Σ
$\Rightarrow$
Σ
R
R
$\hat{\Sigma}$
$\rightsquigarrow$
$\exists$
$\hat{\Sigma}$

$$R(\varsigma, \hat{\varsigma}) \text{ iff } \alpha(\varsigma) \sqsubseteq \hat{\varsigma}$$

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

$$\hat{L} = \mathcal{P}(\hat{\boldsymbol{\Sigma}})$$

Classical abstract interpretation?

$$\hat{L} = \mathcal{P}(\hat{\Sigma})$$

$$\hat{L} = \mathcal{P}(\hat{\boldsymbol{\Sigma}})$$

$$f : L \to L$$

$$\hat{f}(\hat{S}) = \{\hat{\varsigma}_0\} \cup \{\hat{\varsigma}' \mid \hat{\varsigma} \rightsquigarrow \hat{\varsigma}' \text{ and } \hat{\varsigma} \in \hat{S}\}$$

$$\mathrm{lfp}(\hat{f})$$

$$\mathrm{lfp}(\hat{f}) = \{\hat{\varsigma}' \mid \hat{\varsigma}_0 \rightsquigarrow^* \hat{\varsigma}'\}$$

$$\hat{V} = \{\hat{\varsigma}' \mid \hat{\varsigma}_0 \rightsquigarrow^* \hat{\varsigma}'\}$$

Case study: CPS and 0CFA

$$v \in \mathsf{Var} \text{ is a set of variables}$$

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

$$f, æ \in \mathsf{AExp} ::= v \mid lam$$

$$call \in \mathsf{Call} ::= (f\ æ_1 \ldots æ_n)$$

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

$$Env = \mathsf{Var} \to Clo$$

$$Clo = \mathsf{Lam} \times Env$$

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

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

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

$$\mathcal{A} : \mathsf{AExp} \times Env \to Clo$$

$$\mathcal{A} : \mathsf{AExp} \times Env \rightarrow Clo$$

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

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

$$\mathcal{I} : \mathsf{Call} \to \Sigma$$

$$\mathcal{I} : \mathsf{Call} \to \Sigma$$

$$\mathcal{I}(call) = (call, [])$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (\quad\quad, \quad\quad), \text{ where}$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (\quad\quad, \quad\quad), \text{ where}$$

$$f$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (\quad\quad , \quad\quad), \text{ where}$$
$$= \mathcal{A}(f, \quad)$$

$$([\![(f\ æ_1 \ldots æ_n)]\!], \rho) \Rightarrow (\quad\quad , \quad\quad \ddots), \text{ where}$$
$$= \mathcal{A}(f, \rho)$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (\quad\quad , \quad\quad), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ \ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$([\![(f\ æ_1 \ldots æ_n)]\!], \rho) \Rightarrow (call, \quad \ddots), \text{ where}$$

$$([\![(\lambda\ (v_1 \ldots v_n)\ call)]\!], \rho') = \mathcal{A}(f, \rho)$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (call, \quad), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (call, \quad \ddots), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (call, \quad \ddots), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (call, \rho'')\text{, where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

Structural abstraction?

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

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

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

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

$$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \to \widehat{Clo}$$

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

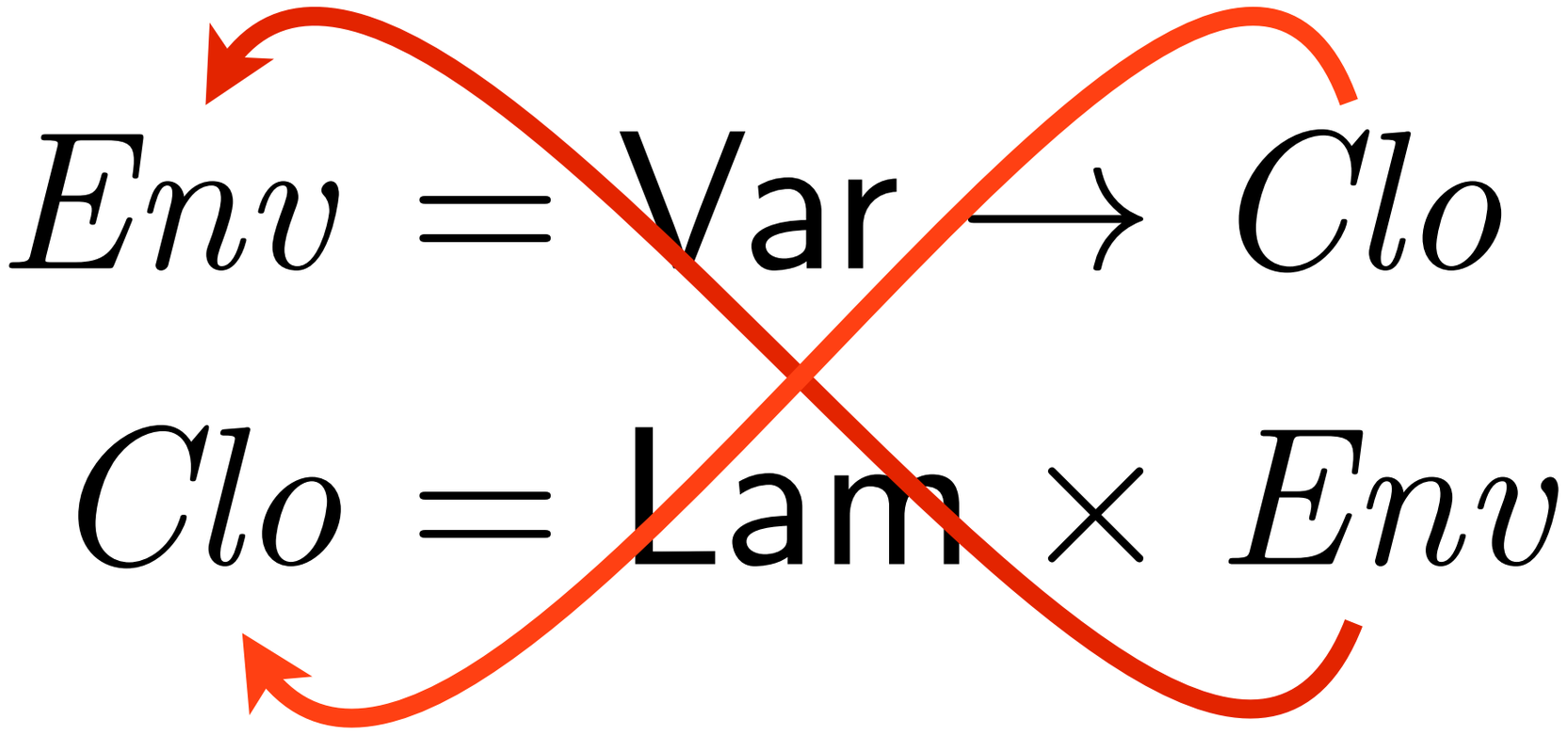
$$\varsigma \in \Sigma = \mathsf{Call} \times Env$$

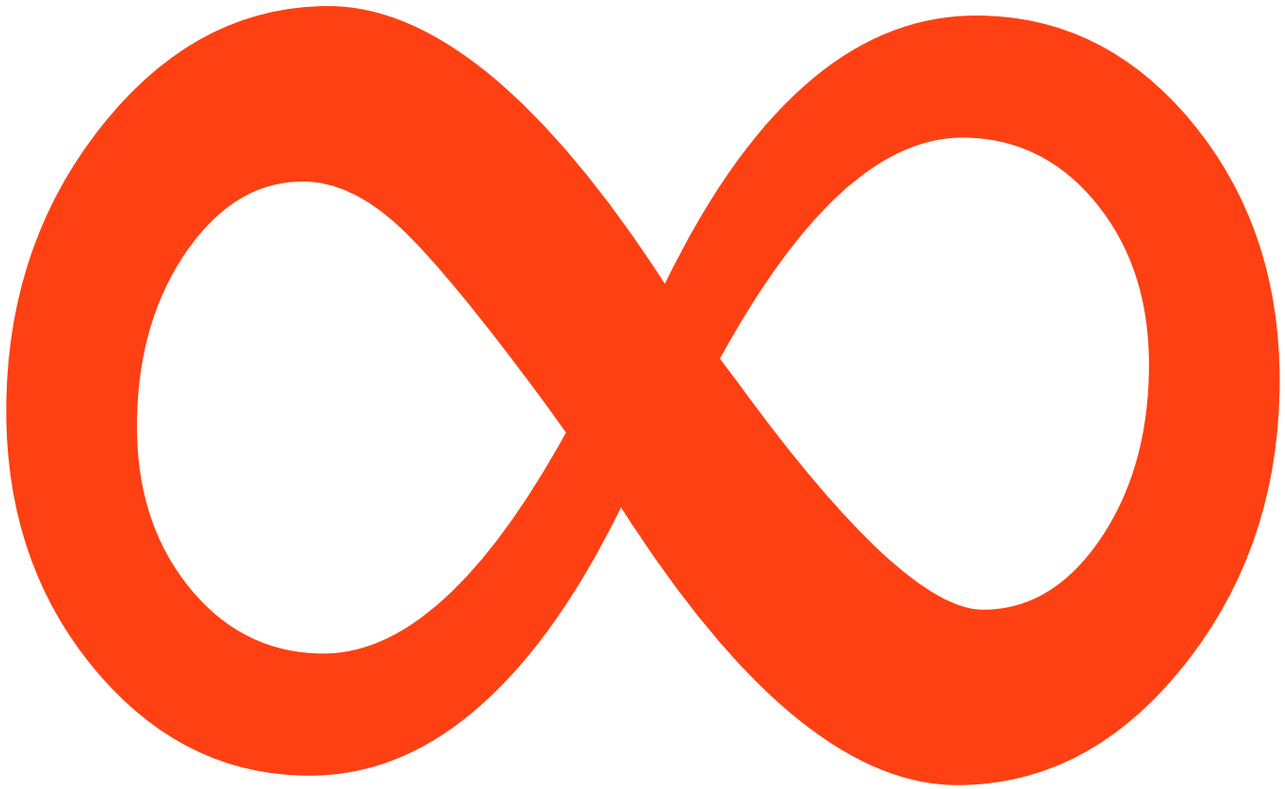
$$\rho \in Env = \mathsf{Var} \to Clo$$

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

$$Env = \mathsf{Var} \to Clo$$

$$Clo = \mathsf{Lam} \times Env$$


$Env = \mathsf{Var} \to Clo$
$Clo = \mathsf{Lam} \times Env$



How to cut the knot?

Scott & Strachey, 1966

Store-passing transform.

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

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

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

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

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

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

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

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

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

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

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

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

$$\sigma \in Store = Addr \to$$

$a \in Addr$ is an infinite set.

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

$$\rho \in Env = \mathsf{Var} \to$$

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

$$\sigma \in Store = Addr \to Clo$$

$a \in Addr$ is an infinite set.

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

$$\rho \in Env = \mathsf{Var} \to Addr$$

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

$$\sigma \in Store = Addr \to Clo$$

$$a \in Addr \text{ is an infinite set.}$$

$$\mathcal{A} : \mathsf{AExp} \times Env \rightarrow Clo$$

$$\mathcal{A} : \mathsf{AExp} \times \mathit{Env} \times \mathit{Store} \to \mathit{Clo}$$

$$\mathcal{A} : \mathsf{AExp} \times \mathit{Env} \times \mathit{Store} \to \mathit{Clo}$$

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

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

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho) \Rightarrow (call, \rho''), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

$$(\llbracket(f\ æ_1 \ldots æ_n)\rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho''\ ), \text{ where}$$

$$(\llbracket(\lambda\ (v_1 \ldots v_n)\ call)\rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho\ )]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto \mathcal{A}(æ_i, \rho\ , \sigma)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto a_i]$$

$$\mathcal{A}(æ_i, \rho\ , \sigma)]$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto a_i]$$

$$\mathcal{A}(æ_i, \rho\ , \sigma)]$$

$$a_i = alloc(v_i, \sigma)$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \rho\ , \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto a_i]$$

$$\sigma' = \sigma[a_i \mapsto \mathcal{A}(æ_i, \rho\ , \sigma)]$$

$$a_i = alloc(v_i, \sigma)$$

Abstraction

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

$$\rho \in Env = \mathsf{Var} \to Addr$$

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

$$\sigma \in Store = Addr \to Clo$$

$a \in Addr$ is an infinite set.

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

$$\rho \in Env = \mathsf{Var} \to Addr$$

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

$$\sigma \in Store = Addr \to Clo$$

$a \in Addr$ is an infinite set.

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

$$\rho \in Env = \mathsf{Var} \to Addr$$

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

$$\sigma \in Store = Addr \to Clo$$

$a \in Addr$ is a finite set.

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

$$\rho \in Env = \mathsf{Var} \to Addr$$

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

$$\sigma \in Store = Addr \to \mathcal{P}(\ Clo\ )$$

$$a \in Addr \text{ is a finite set.}$$

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

$$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \to \widehat{Addr}$$

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

$$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \to \mathcal{P}(\ \widehat{Clo}\ )$$

$$\hat{a} \in \widehat{Addr} \text{ is a finite set.}$$

$$\widehat{Addr} : Addr \to Addr$$

$$\alpha : \sum \to \sum$$

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

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

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

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

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

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

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

$$\alpha(\rho) = \lambda v.\alpha(\rho(v))$$

$$\alpha(\sigma) = \lambda \hat{a}. \bigsqcup_{\alpha(a)=\hat{a}} \{\alpha(\sigma(a))\}$$

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

$$\alpha(a) \text{ is to be continued...}$$

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

$$\alpha(\rho) = \lambda v.\, \alpha(\rho(v))$$

$$\alpha(\sigma) = \lambda \hat{a}. \bigsqcup \{\alpha(\sigma(a))\}$$
$$\alpha(a)=\hat{a}$$

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

$\alpha(a)$ is to be continued...

$$(\llbracket(f\ æ_1 \ldots æ_n)\rrbracket, \rho, \sigma) \Rightarrow (call, \rho'', \sigma'), \text{ where}$$

$$(\llbracket(\lambda\ (v_1 \ldots v_n)\ call)\rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto a_i]$$

$$\sigma' = \sigma\left[a_i \mapsto \mathcal{A}(æ_i, \rho, \sigma)\right]$$

$$a_i = alloc(v_i, \sigma)$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \hat{\rho}, \hat{\sigma}) \Rightarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \hat{\rho}') = \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$

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

$$\hat{\sigma}' = \hat{\sigma}\,[\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \hat{\rho}, \hat{\sigma}) \rightsquigarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \hat{\rho}') = \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$

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

$$\hat{\sigma}' = \hat{\sigma}\,[\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \hat{\rho}, \hat{\sigma}) \rightsquigarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \hat{\rho}') = \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$

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

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

$$(\llbracket (f\ æ_1 \ldots æ_n) \rrbracket, \hat{\rho}, \hat{\sigma}) \rightsquigarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

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

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

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

Allocation

$$(\llbracket(f\ æ_1 \ldots æ_n)\rrbracket, \hat{\rho}, \hat{\sigma}) \Rightarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

$$(\llbracket(\lambda\ (v_1 \ldots v_n)\ call)\rrbracket, \hat{\rho}') = \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$

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

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

$$(\llbracket(f\ æ_1 \ldots æ_n)\rrbracket, \hat{\rho}, \hat{\sigma}) \Rightarrow (call, \hat{\rho}'', \hat{\sigma}'), \text{ where}$$

$$(\llbracket(\lambda\ (v_1 \ldots v_n)\ call)\rrbracket, \hat{\rho}') = \hat{\mathcal{A}}(f, \hat{\rho}, \hat{\sigma})$$

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

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{\mathcal{A}}(æ_i, \hat{\rho}, \hat{\sigma})]$$

$$\hat{a}_i = \widehat{alloc}(v_i, \hat{\sigma})$$

Determines polyvariance

0CFA

$$Addr = \mathsf{Var}$$

$$\widehat{alloc}(v, \hat{\sigma}) =$$

$$\widehat{alloc}(v, \hat{\sigma}) = v$$

Lessons?

Step 1: Cut recursion.

Step 2: Structurally abstract.

Case study: ANF

$$
\begin{aligned}
e \in \mathsf{Exp} &::= \texttt{(let ((}v\ call\texttt{))}\ e\texttt{)} \\
&\mid\ call \\
&\mid\ æ \\
f, æ \in \mathsf{AExp} &::= v \mid lam \\
lam \in \mathsf{Lam} &::= (\lambda\ (v)\ e) \\
call \in \mathsf{Call} &::= (f\ æ) \\
v \in \mathsf{Var} &\text{ is a set of identifiers}
\end{aligned}
$$

ČESK

CESK*

CESK*

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Kont$$

C E S K

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Kont$$

state-space

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Kont$$

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Kont$$

$$Store = Addr \to Clo$$

$$Env = \mathsf{Var} \to Addr$$

$$Clo = \mathsf{Lam} \times Env$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$

$$\mid \ \mathbf{halt}$$

$$\hat{\Sigma} = \mathsf{Exp} \times \hat{Env} \times \hat{Store} \times \hat{Kont}$$

$$\hat{Store} = \hat{Addr} \to \hat{Clo}$$

$$Env = \mathsf{Var} \to Addr$$

$$\hat{Clo} = \mathsf{Lam} \times \hat{Env}$$

$$\hat{\kappa} \in \hat{Kont} ::= \mathbf{letk}(v, e, \hat{\rho}, \hat{\kappa}) \mid \mathbf{halt}$$

$$\hat{\Sigma} = \mathsf{Exp} \times \hat{Env} \times \hat{Store} \times \hat{Kont}$$

$$\hat{Store} = \hat{Addr} \to \hat{Clo}$$

$$Env = \mathsf{Var} \to Addr$$

$$\hat{Clo} = \mathsf{Lam} \times \hat{Env}$$

$$\hat{\kappa} \in \hat{Kont} ::= \mathbf{letk}(v, e, \hat{\rho}, \hat{\kappa}) \mid \mathbf{halt}$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$

$$\hat{\kappa} \in \widehat{Kont} ::= \mathbf{letk}(v, e, \hat{\rho}, \hat{\kappa})$$

Store-allocate $Kont$

$$Store = Addr \to Clo$$

$$Store = Addr \to Clo + Kont$$

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Kont$$

$$\Sigma = \mathsf{Exp} \times Env \times Store \times Addr$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, a)$$

$$(\llbracket (f\ æ) \rrbracket, \rho, \sigma, a_\kappa) \Rightarrow (e, \rho', \sigma', a_\kappa), \text{ where}$$

$$(\llbracket (\lambda\ (v_1 \ldots v_n)\ call) \rrbracket, \rho') = \mathcal{A}(f, \rho, \sigma)$$

$$\rho'' = \rho'[v_i \mapsto a_i]$$

$$\sigma' = \sigma[a_i \mapsto \mathcal{A}(æ_i, \rho, \sigma)]$$

$$a_i = alloc(v_i, \ldots)$$

$$(æ, \rho, \sigma, a_\kappa) \Rightarrow (e, \rho'', \sigma', a'_\kappa), \text{ where}$$

$$\mathbf{letk}(v, e, \rho', a'_\kappa) = \sigma(a_k)$$

$$\rho'' = \rho'[v \mapsto a]$$

$$\sigma' = \sigma[a \mapsto \mathcal{A}(æ, \rho, \sigma)]$$

$$a = alloc(v, \ldots)$$

$$([\![\texttt{(let ((}v\ call\texttt{))}\ e\texttt{)}]\!], \rho, \sigma, a_\kappa) \Rightarrow (call, \rho, \sigma', a'_\kappa), \text{ where}$$

$$\sigma' = \sigma[a'_\kappa \mapsto \mathbf{letk}(v, e, \rho, a_\kappa)]$$

$$a'_\kappa = alloc(\ldots)$$

Structural abstraction

$$\varsigma \in \Sigma = \mathsf{Exp} \times Env \times Store \times Addr$$

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

$$\sigma \in Store = Addr \rightharpoonup Clo + Kont$$

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

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, a) \mid \mathbf{halt}$$

$$a \in Addr \text{ is an infinite set}$$

$$\varsigma \in \Sigma = \mathsf{Exp} \times Env \times Store \times Addr$$

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

$$\sigma \in Store = Addr \rightharpoonup Clo + Kont$$

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

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, a) \mid \mathbf{halt}$$

$$a \in Addr \text{ is a finite set}$$

$$\varsigma \in \Sigma = \mathsf{Exp} \times Env \times Store \times Addr$$

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

$$\sigma \in Store = Addr \rightharpoonup \mathcal{P}(Clo + Kont)$$

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

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, a)$$
$$\mid \quad \mathbf{halt}$$

$a \in Addr$ is a finite set

$$\widehat{\varsigma} \in \widehat{\Sigma} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Addr}$$

$$\widehat{\rho} \in \widehat{Env} = \mathsf{Var} \rightharpoonup \widehat{Addr}$$

$$\widehat{\sigma} \in \widehat{Store} = \widehat{Addr} \rightharpoonup \mathcal{P}(\widehat{Clo} + \widehat{Kont})$$

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

$$\widehat{\kappa} \in \widehat{Kont} ::= \mathbf{letk}(v, e, \widehat{\rho}, \widehat{a}) \mid \mathbf{halt}$$

$$\widehat{a} \in \widehat{Addr} \text{ is a finite set}$$

Pushdown?

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$
$$\mid \quad \mathbf{halt}$$

$$\kappa \in Kont ::= \mathbf{letk}(v, e, \rho, \kappa)$$
$$\mid \quad \mathbf{halt}$$
$$::\cong \langle (v_1, e_1, \rho_1), \ldots, (v_n, e_n, \rho_n) \rangle$$

$$\kappa \in Kont = Frame^*$$

$$\phi \in Frame = \mathsf{Var} \times \mathsf{Exp} \times Env$$

$$c \in Conf = \mathsf{Exp} \times Env \times Store \times Kont$$

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

$$\sigma \in Store = Addr \to Clo$$

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

$$\kappa \in Kont = Frame^*$$

$$\phi \in Frame = \mathsf{Var} \times \mathsf{Exp} \times Env$$

$a \in Addr$ is an infinite set of addresses

$$\hat{c} \in \widehat{Conf} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Kont}$$

$$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \rightharpoonup \widehat{Addr}$$

$$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \to \mathcal{P}\left(\widehat{Clo}\right)$$

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

$$\hat{\kappa} \in \widehat{Kont} = \widehat{Frame}^*$$

$$\hat{\phi} \in \widehat{Frame} = \mathsf{Var} \times \mathsf{Exp} \times \widehat{Env}$$

$\hat{a} \in \widehat{Addr}$ is a *finite* set of addresses

$$\hat{c} \in \widehat{Conf} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Kont}$$

$$\hat{\rho} \in \widehat{Env} = \mathsf{Var} \rightharpoonup \widehat{Addr}$$

$$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \to \mathcal{P}\left(\widehat{Clo}\right)$$

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

$$\hat{\kappa} \in \widehat{Kont} = \widehat{Frame}^*$$

$$\hat{\phi} \in \widehat{Frame} = \mathsf{Var} \times \mathsf{Exp} \times \widehat{Env}$$

$\hat{a} \in \widehat{Addr}$ is a *finite* set of addresses

It’s pushdown!

$$\hat{c} \in \widehat{Conf} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Kont}$$

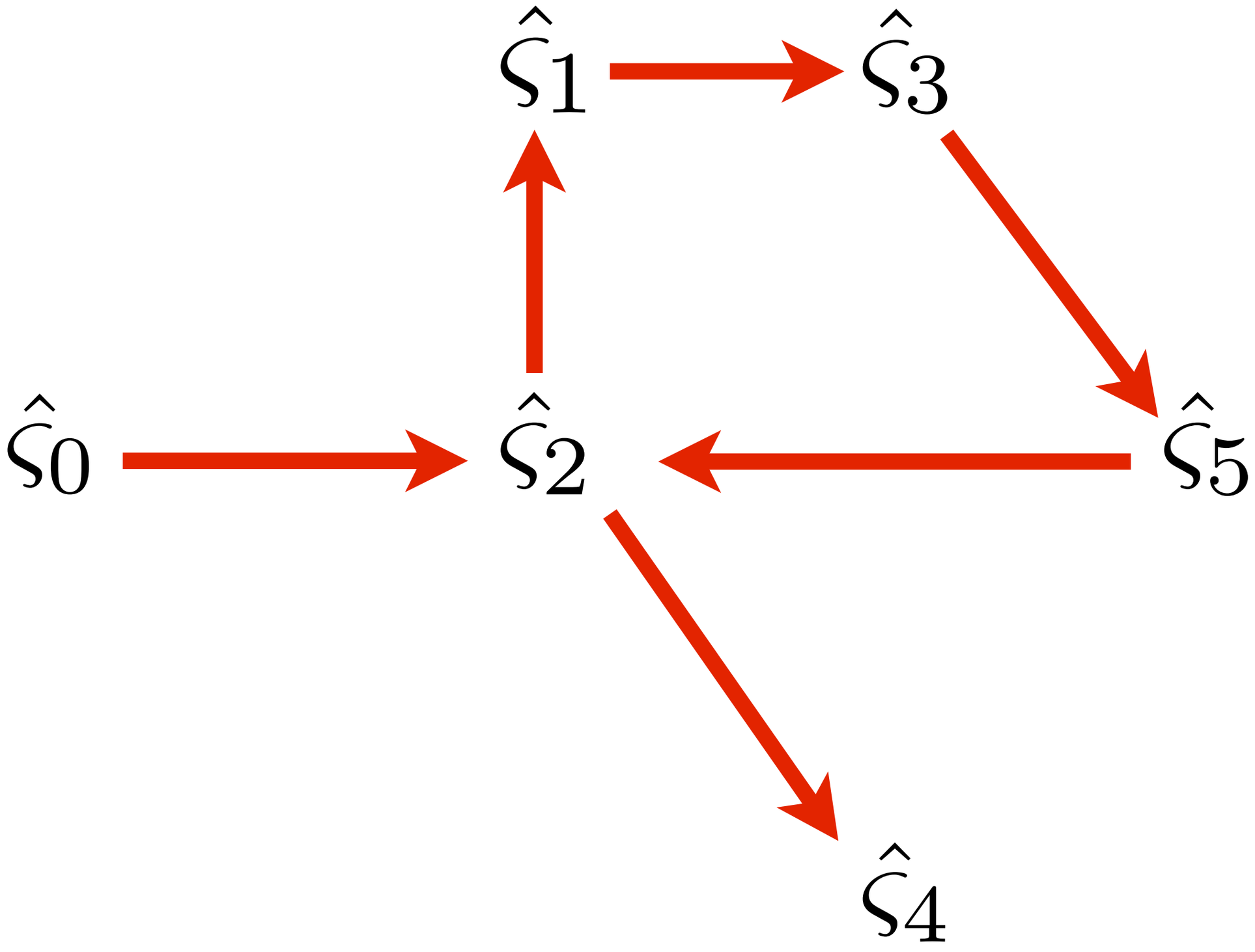
control states

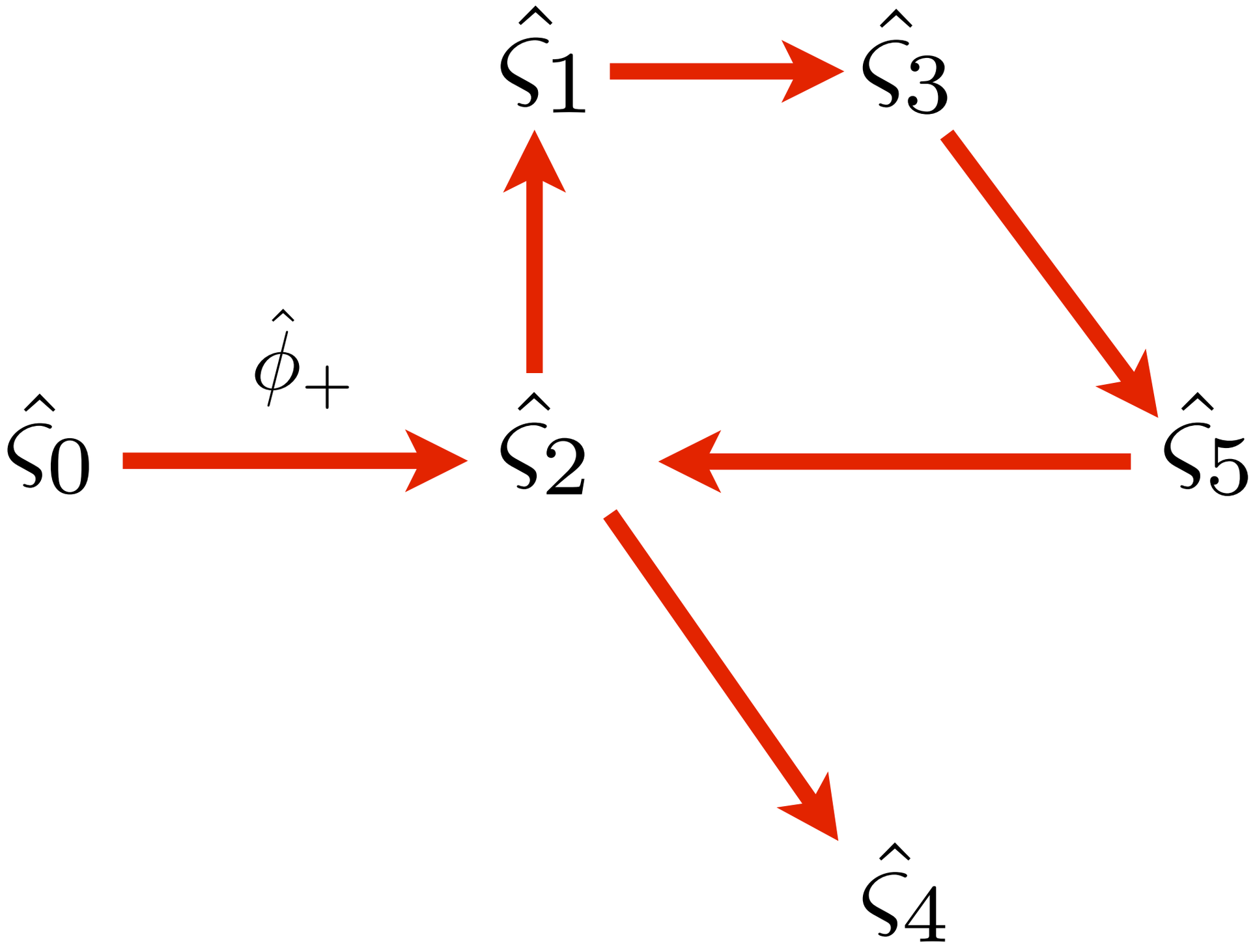
stack

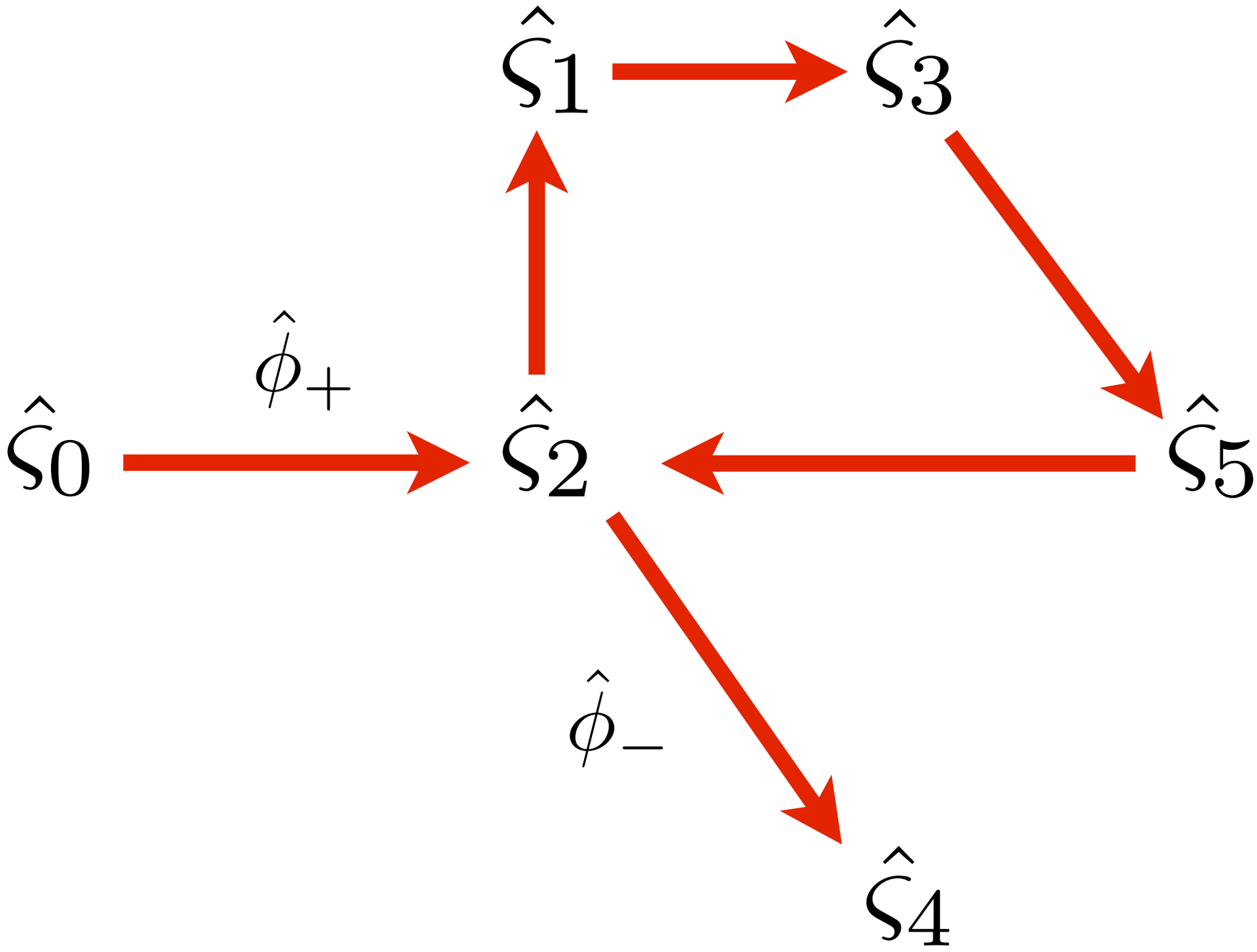
$$\hat{c} \in \widehat{Conf} = \mathsf{Exp} \times \widehat{Env} \times \widehat{Store} \times \widehat{Kont}$$

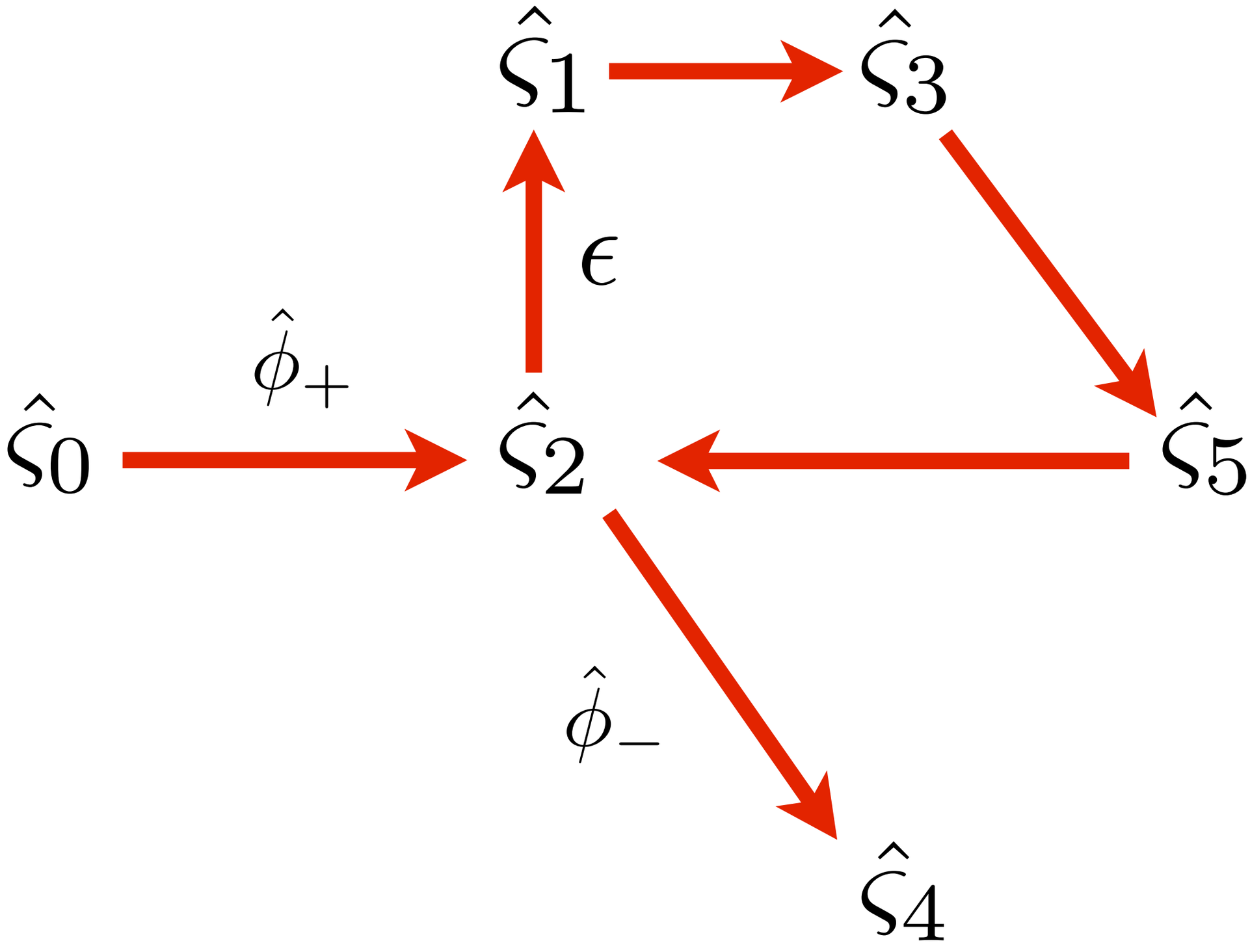
Control-state reachability

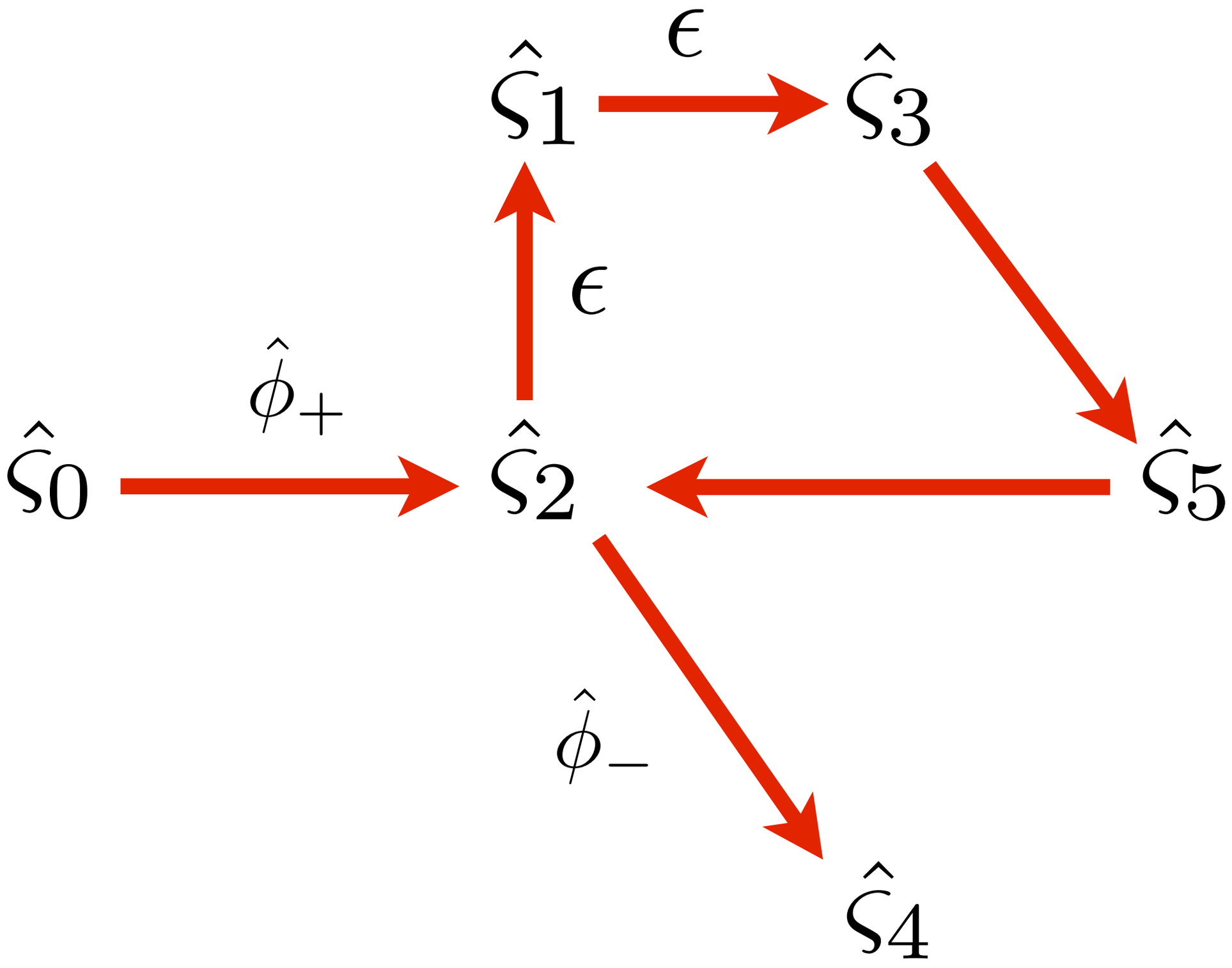
$\hat{\varsigma}_0$

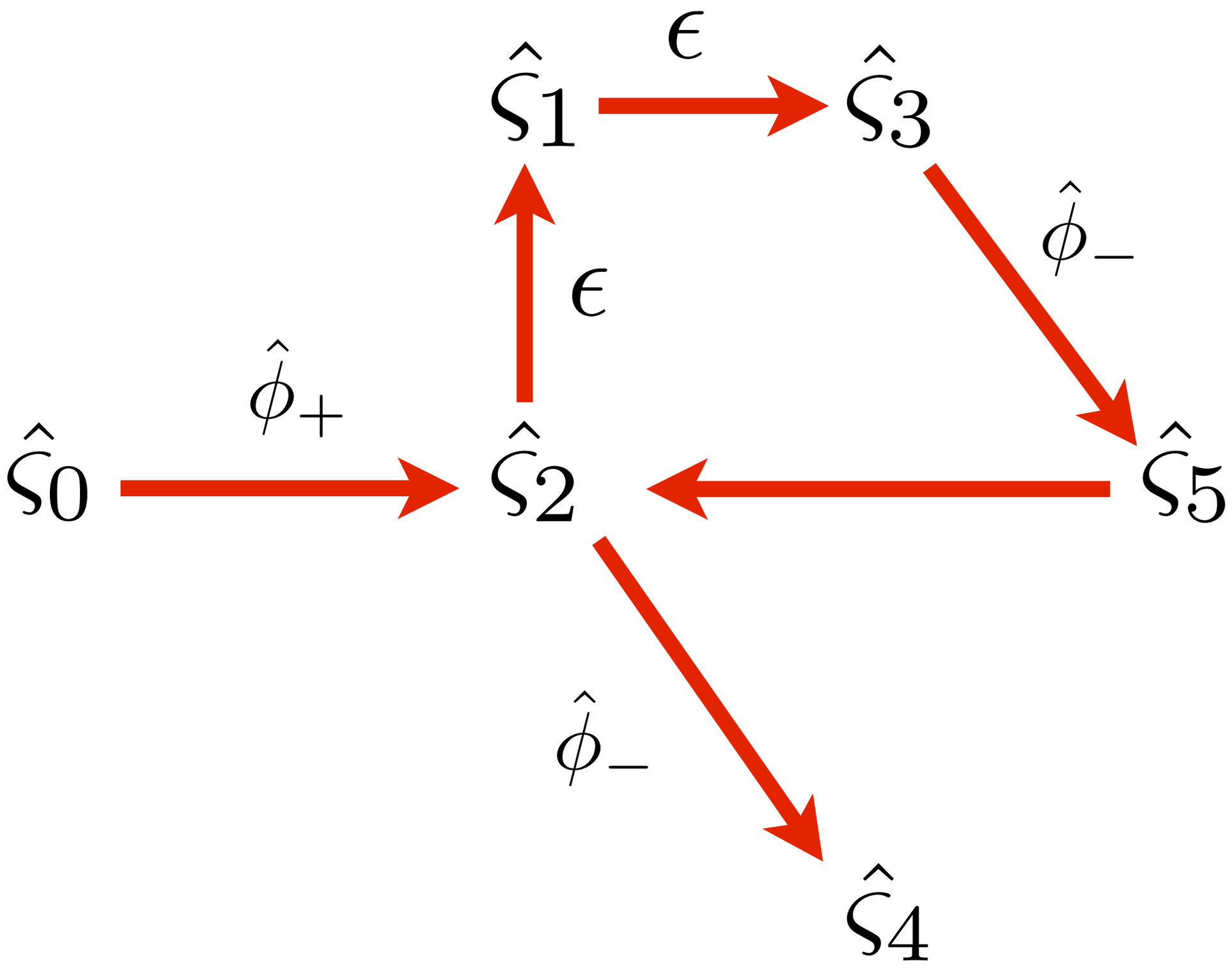

$\hat{\varsigma}_1$
$\hat{\varsigma}_3$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\varsigma}_4$

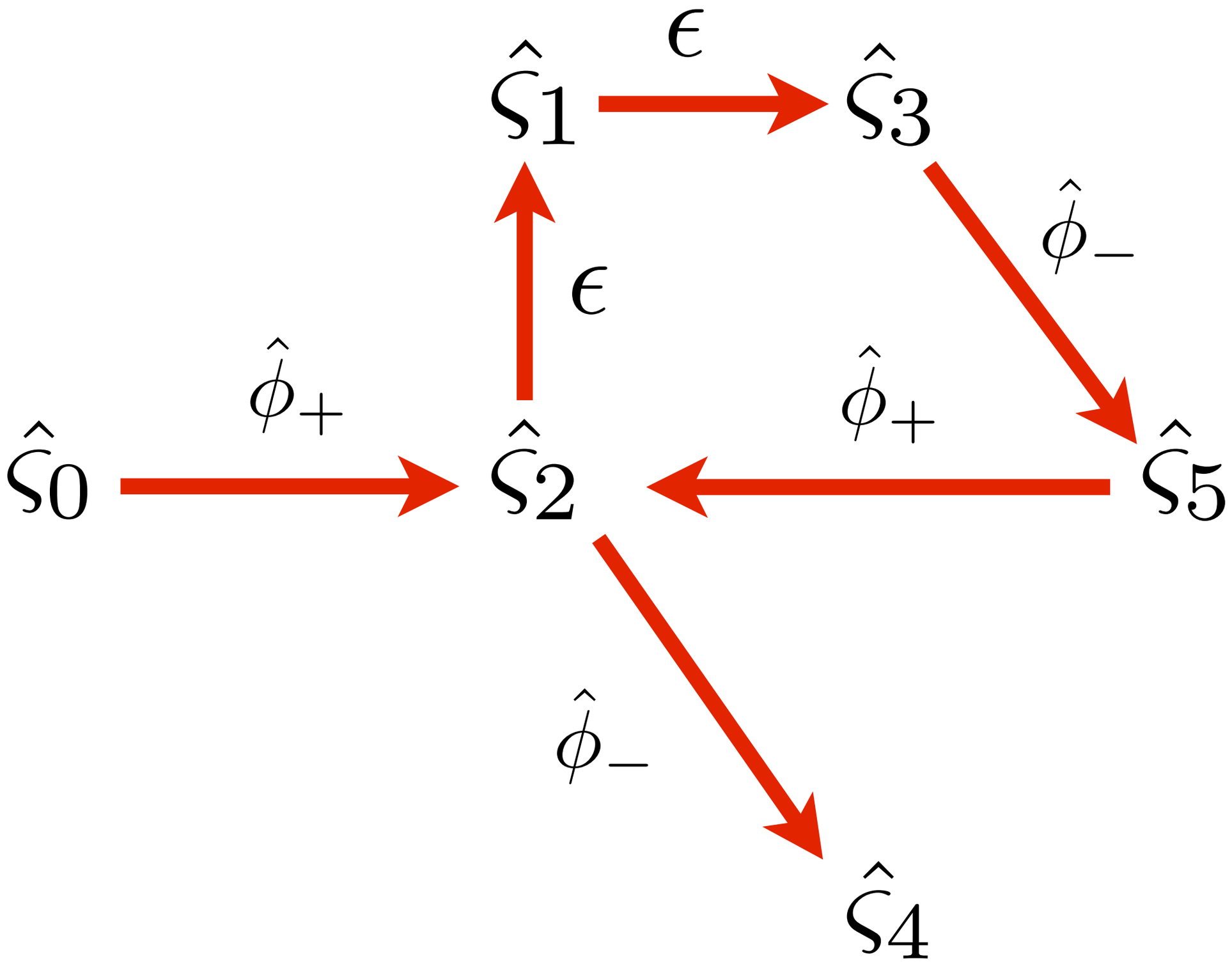

$\hat{\varsigma}_1$
$\hat{\varsigma}_3$
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\varsigma}_4$


$\hat{\varsigma}_1$
$\hat{\varsigma}_3$
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\phi}_-$
$\hat{\varsigma}_4$


$\hat{\varsigma}_1$
$\hat{\varsigma}_3$
ϵ
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\phi}_-$
$\hat{\varsigma}_4$


$\hat{\varsigma}_1$
ϵ
$\hat{\varsigma}_3$
ϵ
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\phi}_-$
$\hat{\varsigma}_4$


$\hat{\varsigma}_1$
ϵ
$\hat{\varsigma}_3$
$\hat{\phi}_-$
ϵ
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\phi}_-$
$\hat{\varsigma}_4$


$\hat{\varsigma}_1$
ϵ
$\hat{\varsigma}_3$
$\hat{\phi}_-$
ϵ
$\hat{\phi}_+$
$\hat{\phi}_+$
$\hat{\varsigma}_0$
$\hat{\varsigma}_2$
$\hat{\varsigma}_5$
$\hat{\phi}_-$
$\hat{\varsigma}_4$

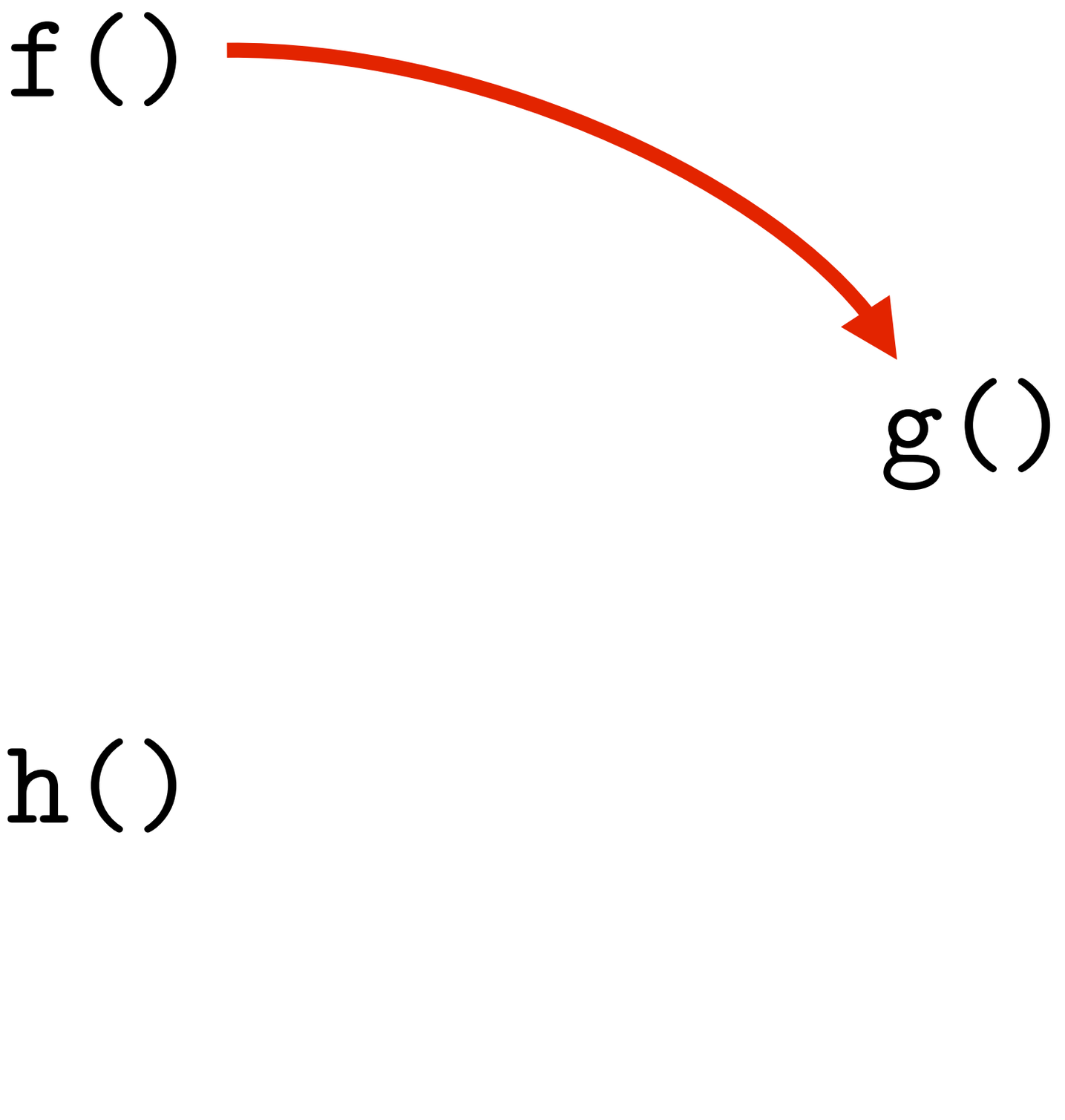
Advantage?

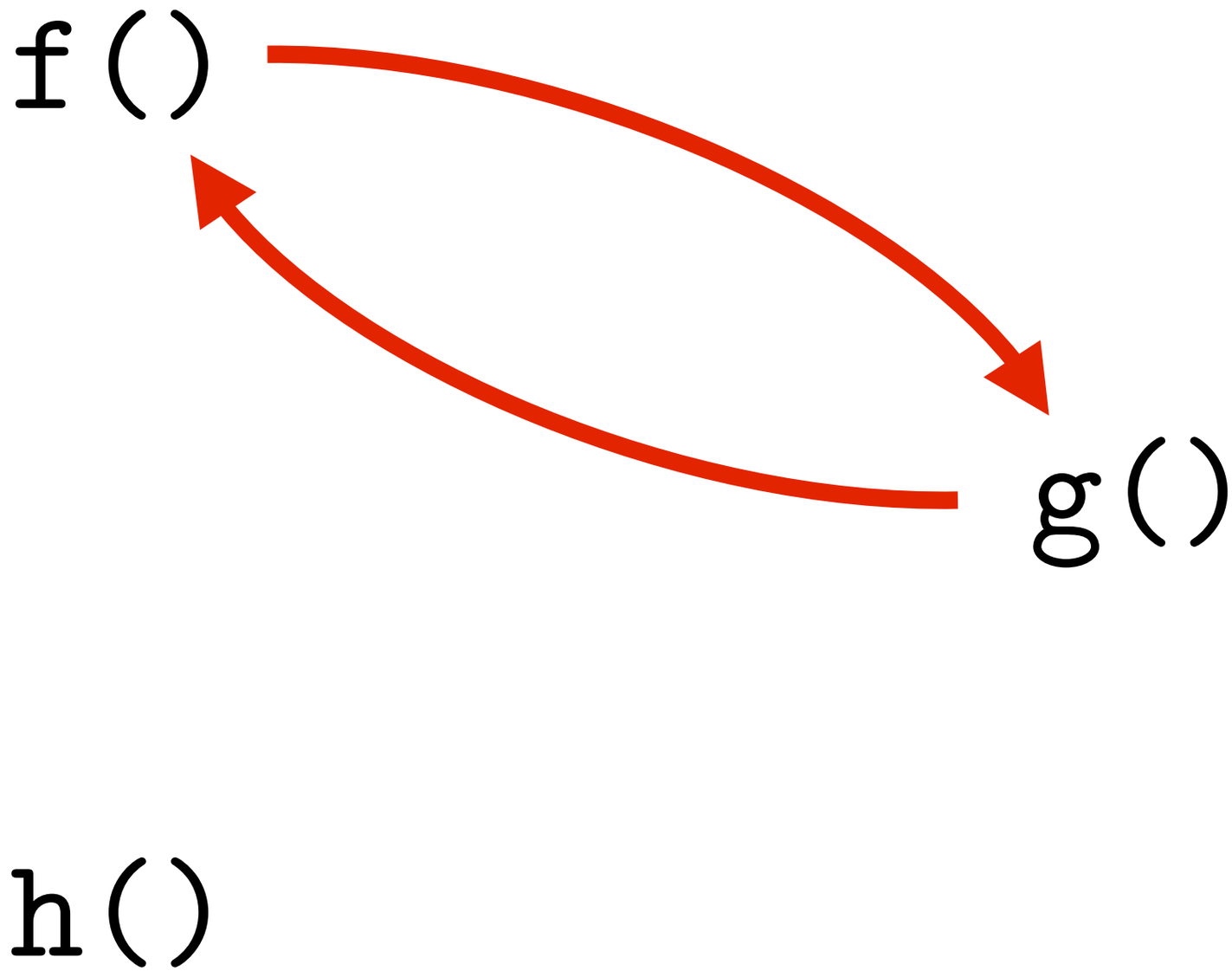
Matches returns to calls.

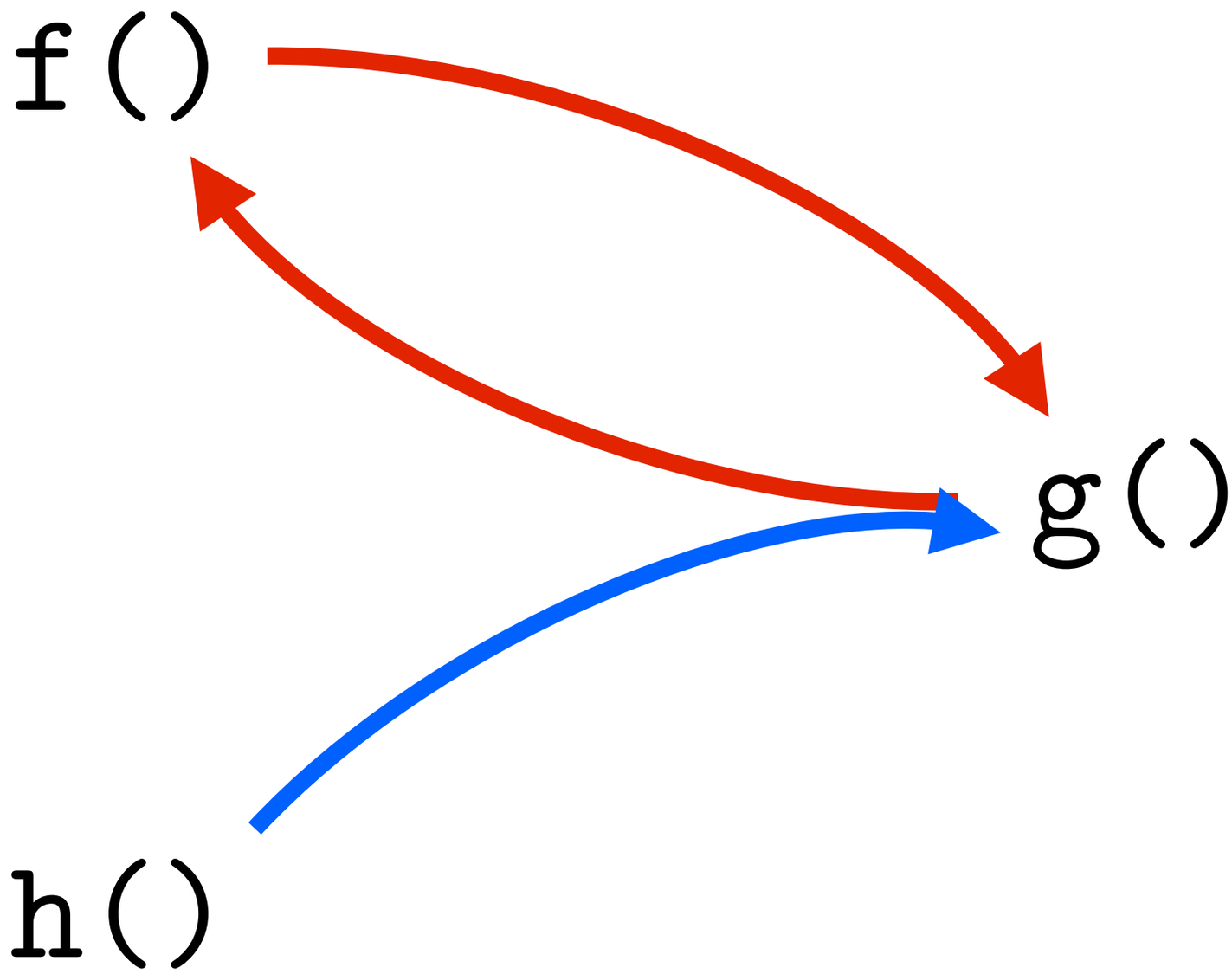
f()

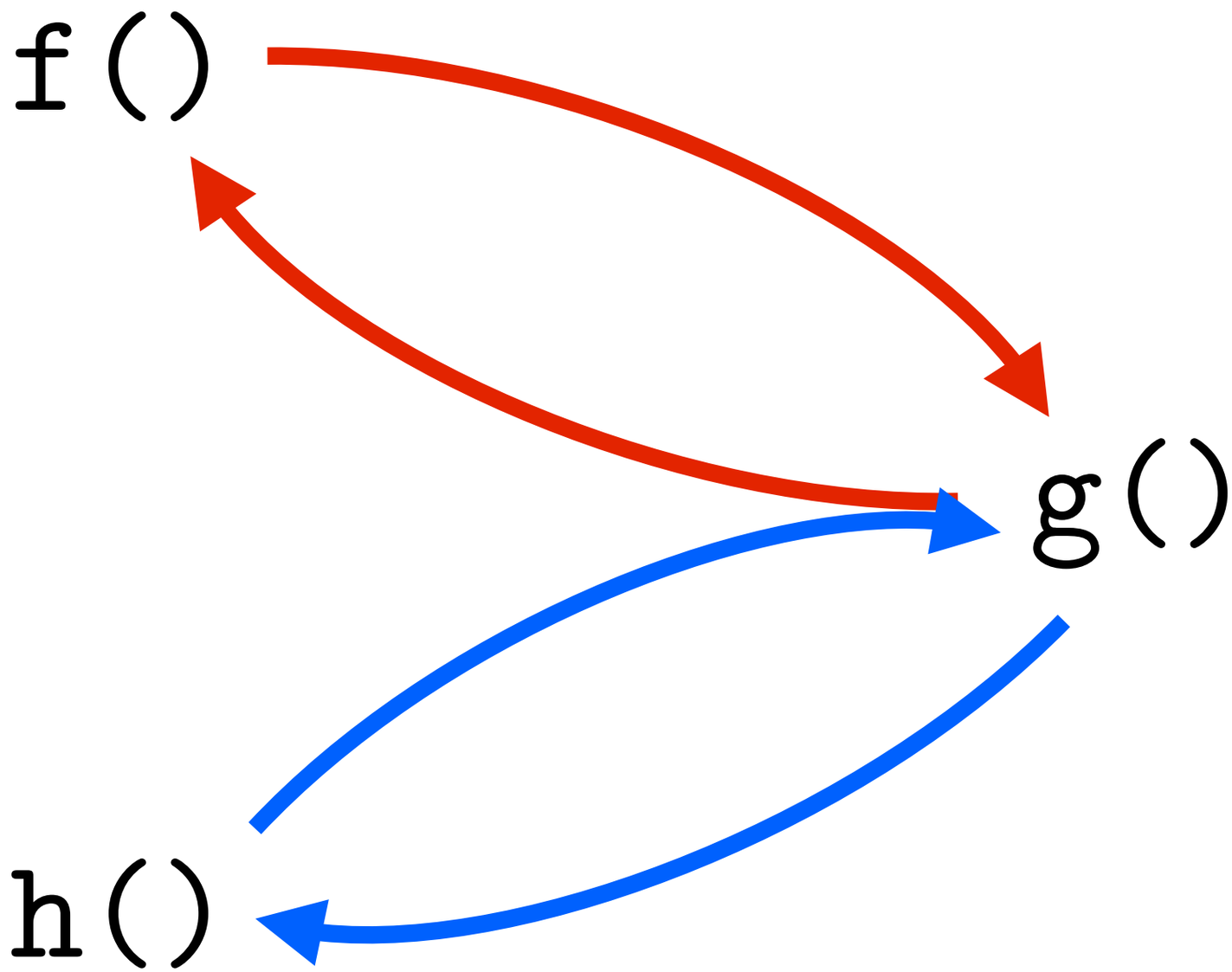
g()

h()


f()
g()
h()


f()
g()
h()

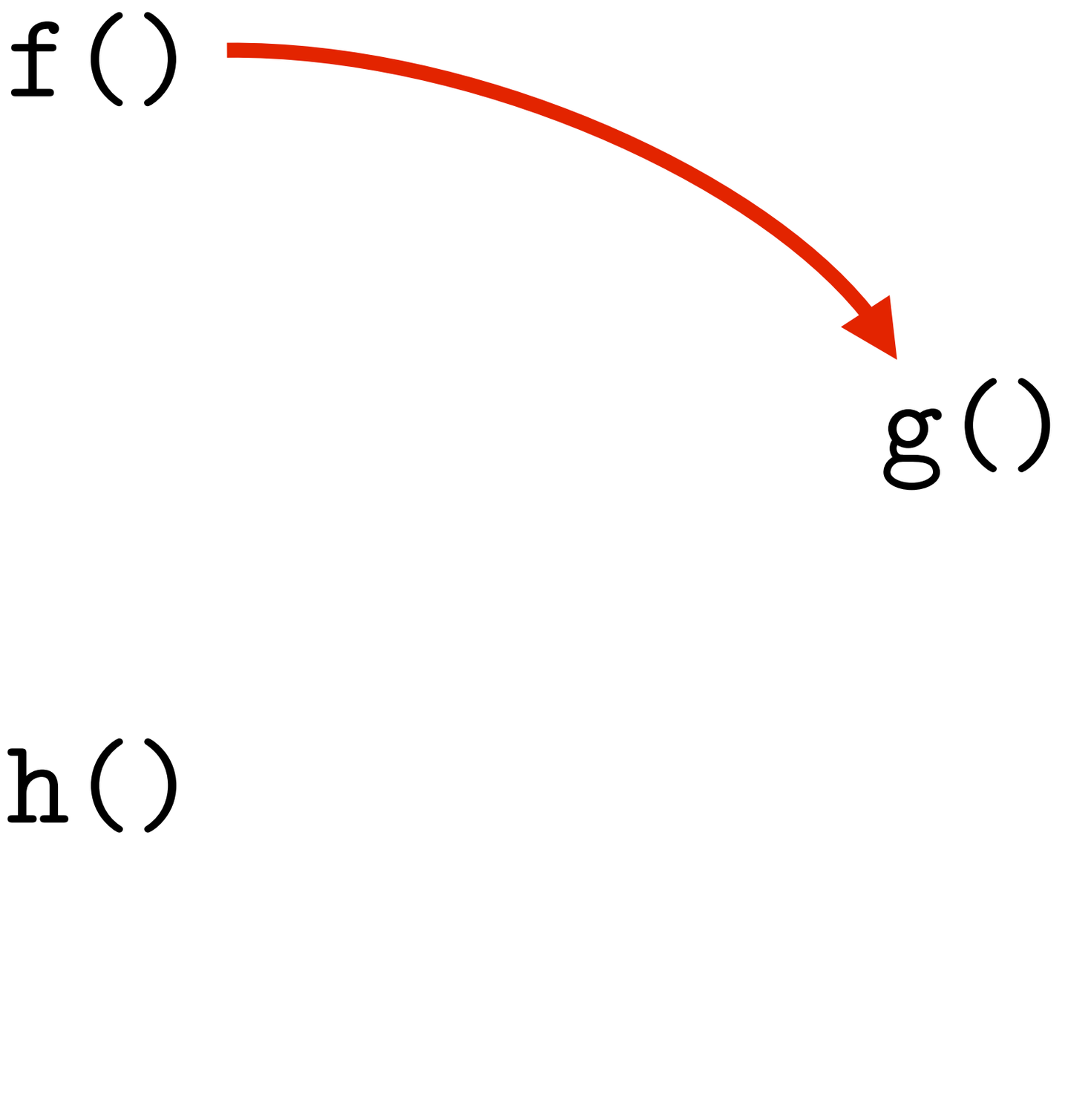

f()
g()
h()

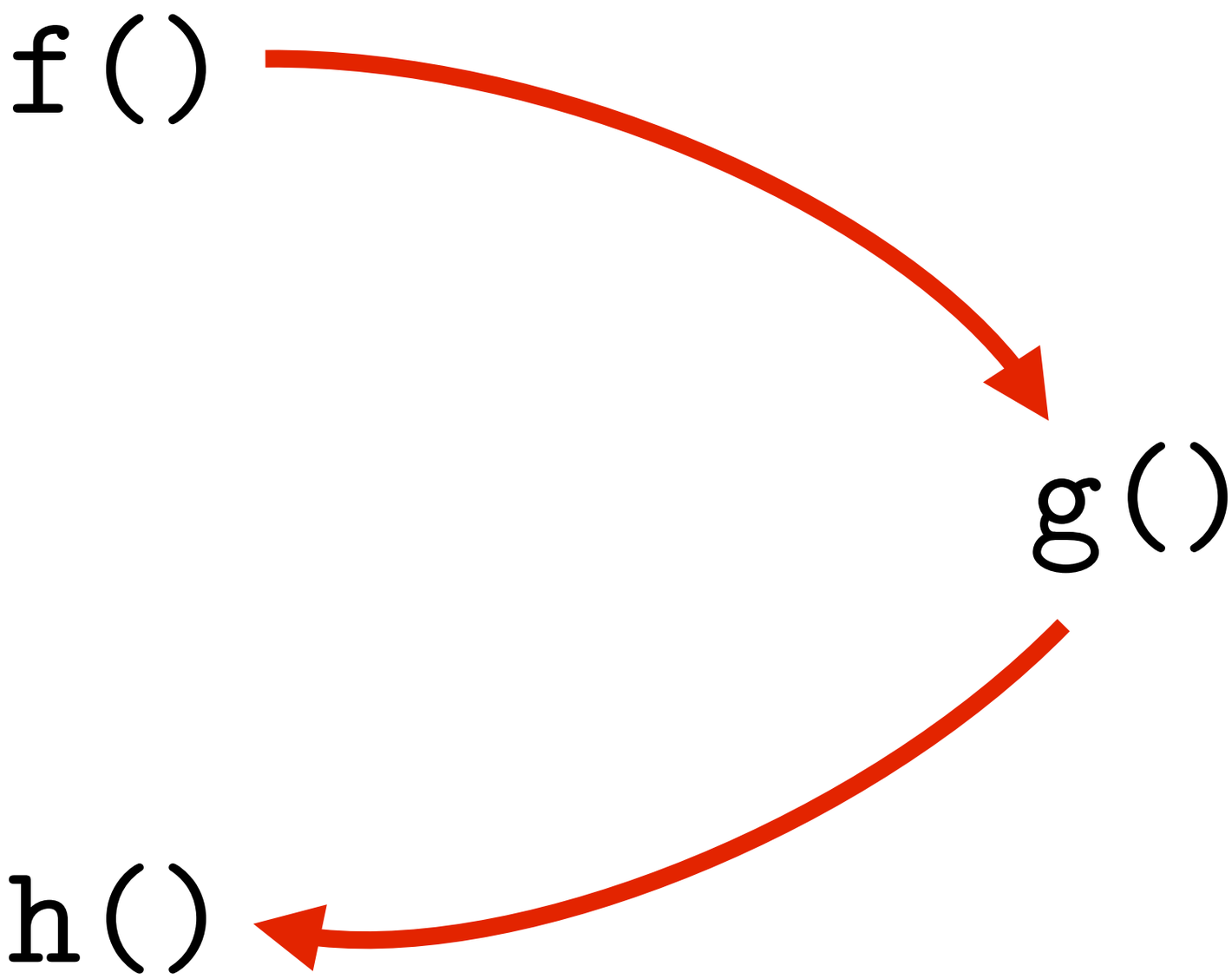

f()
g()
h()

f()

g()

h()


f()
g()
h()

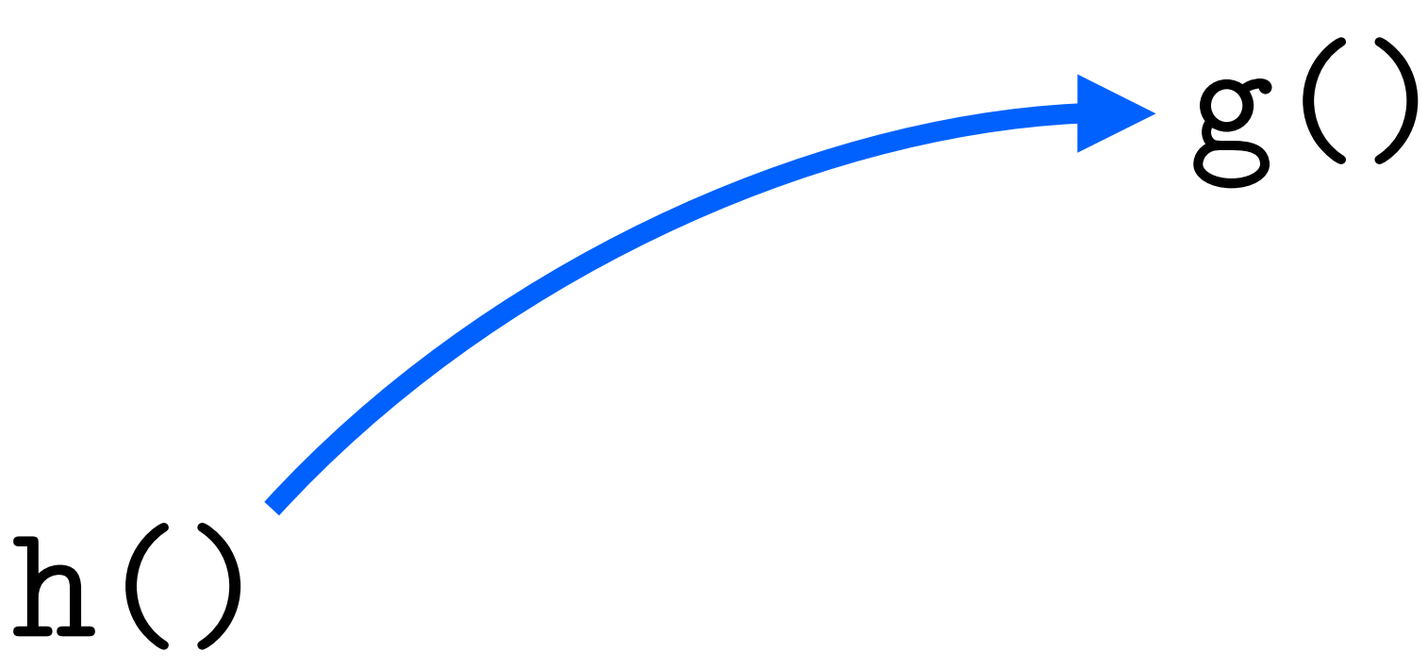

f()
g()
h()

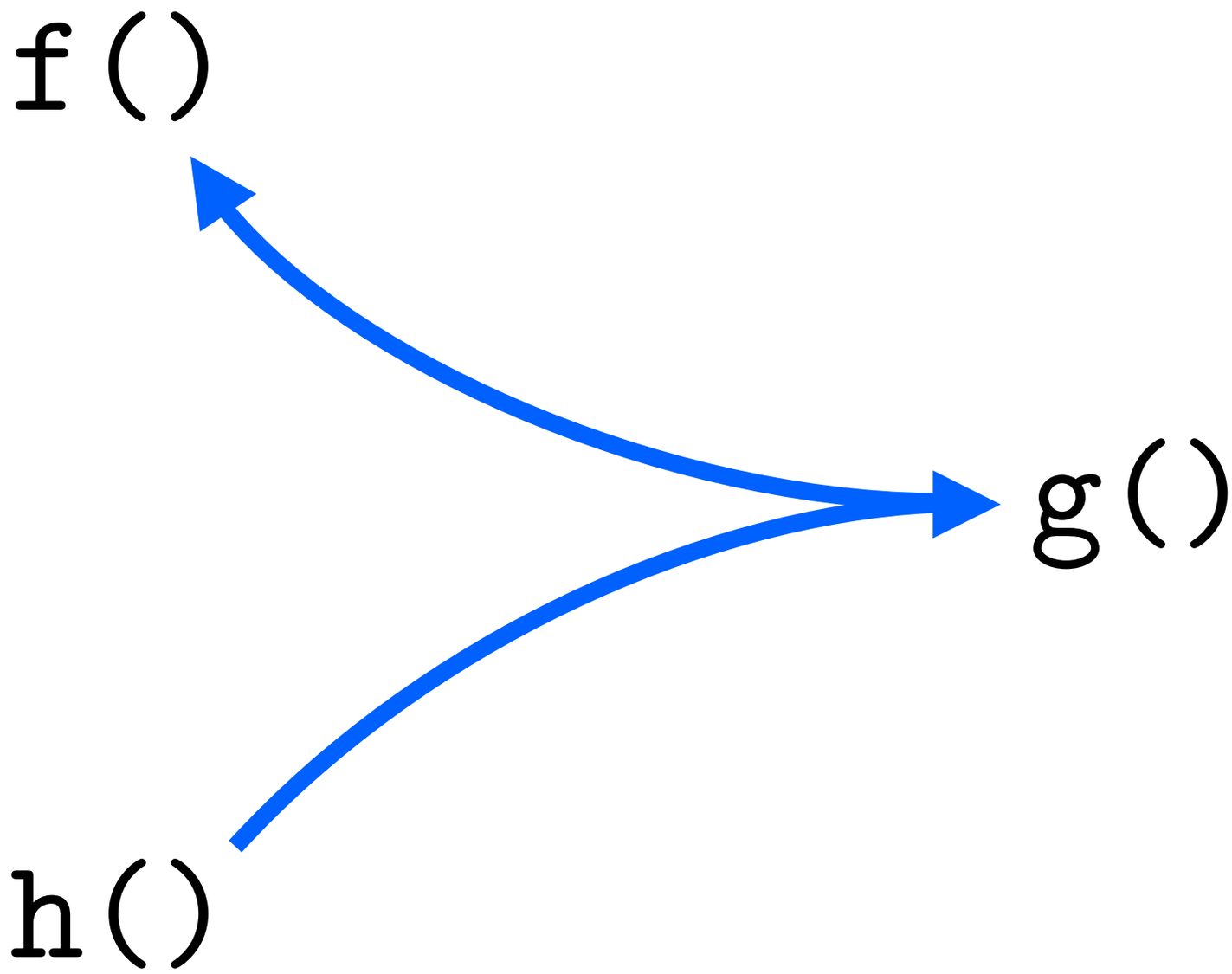
f()

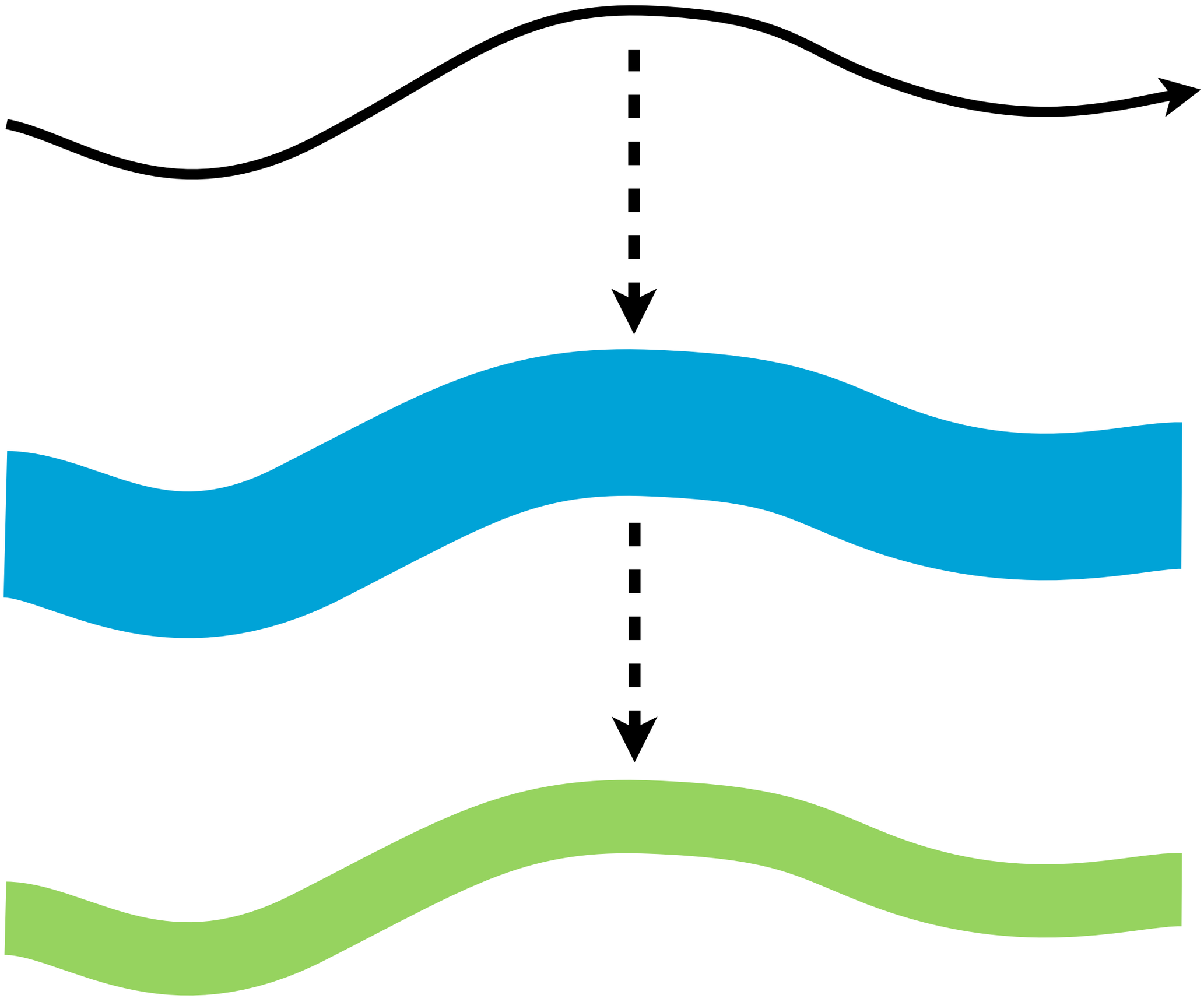
g()

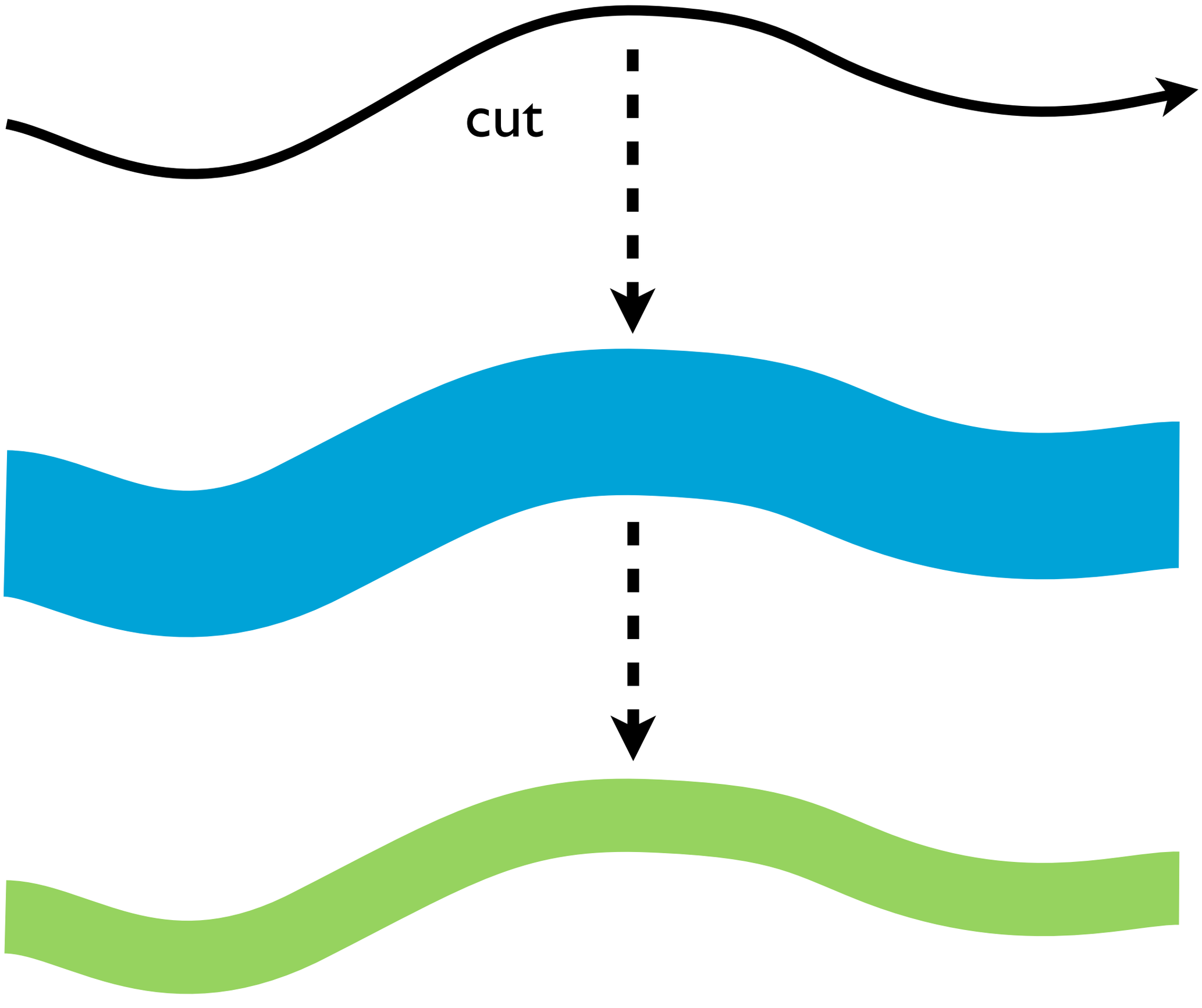
h()

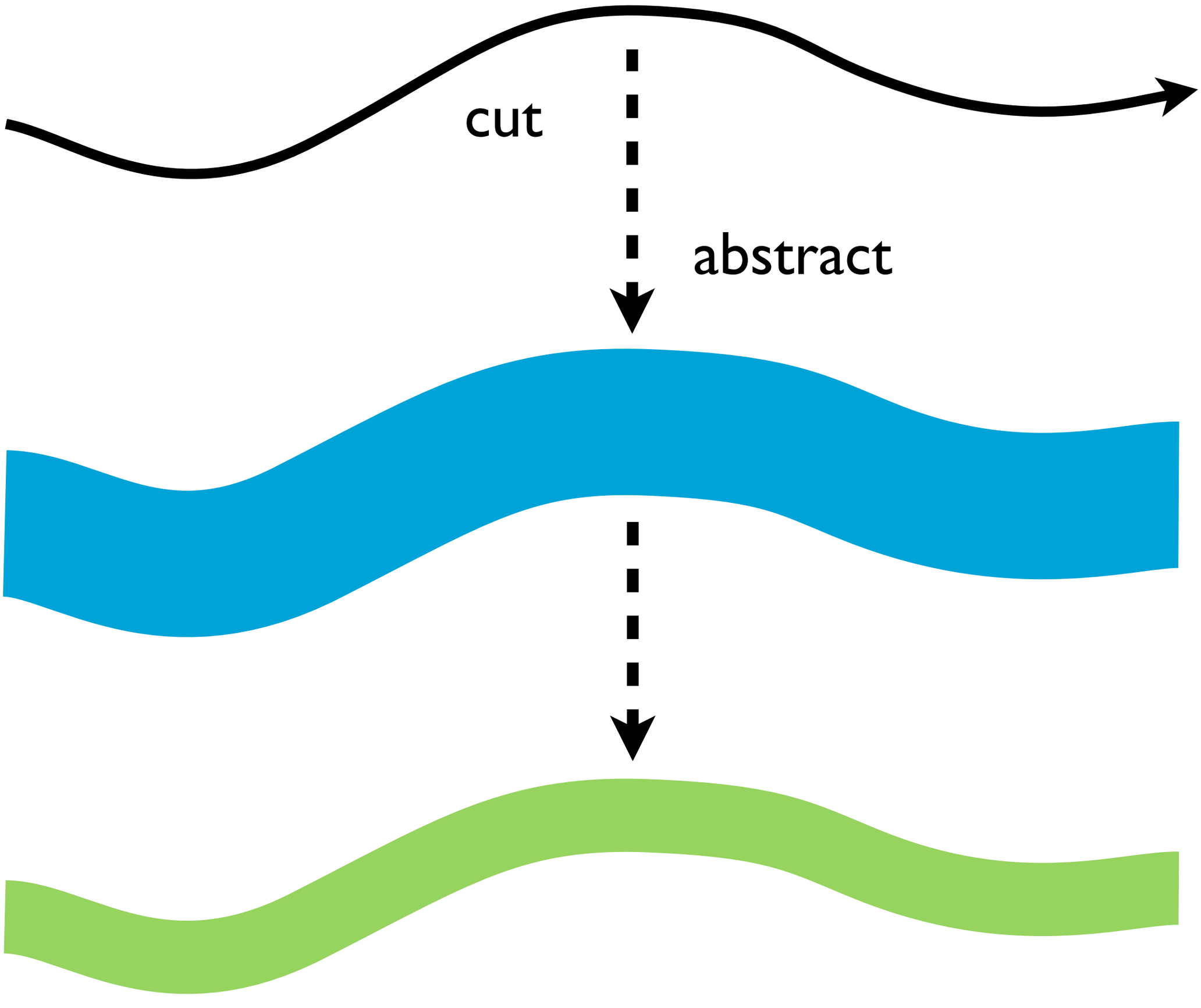
f()

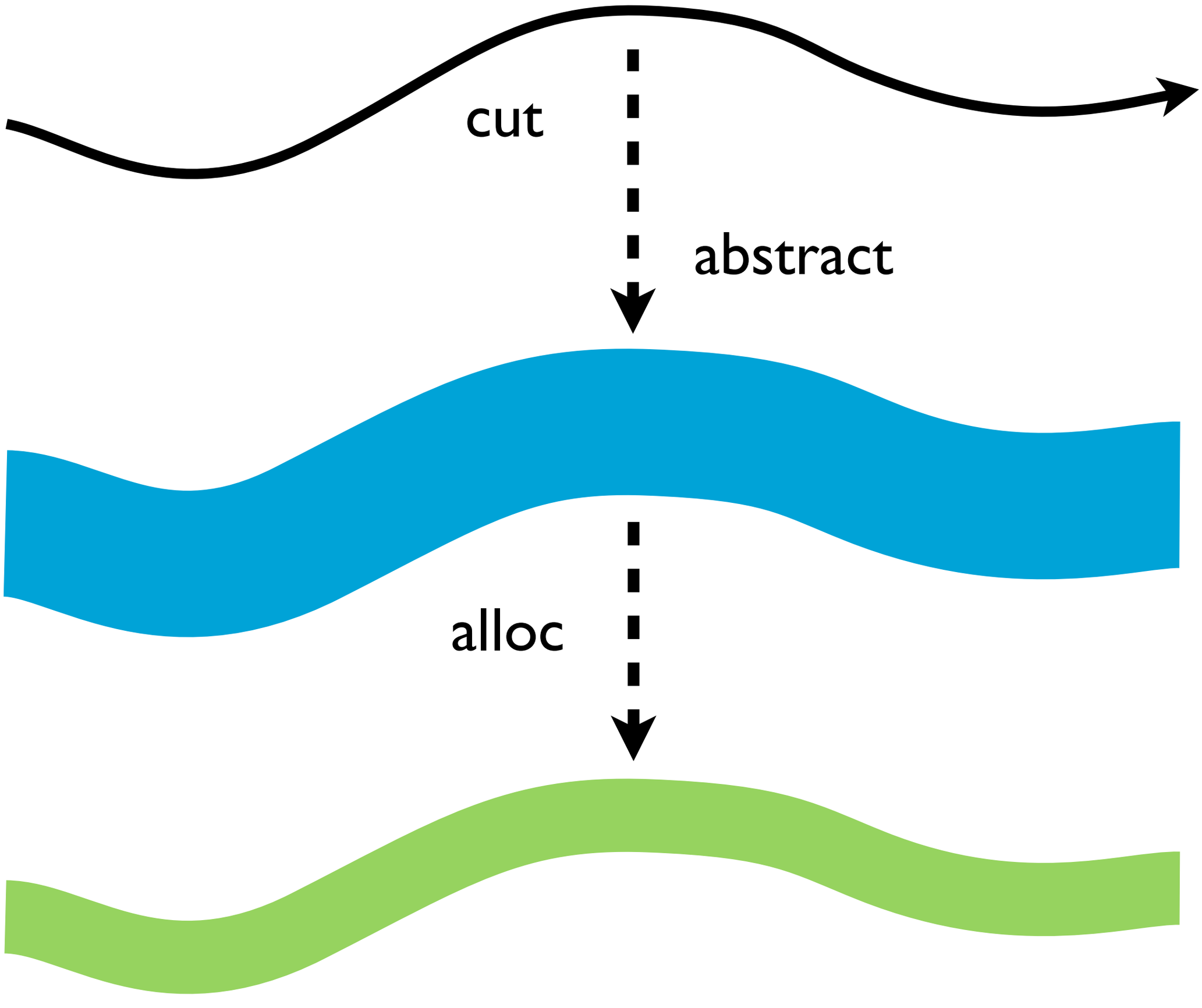


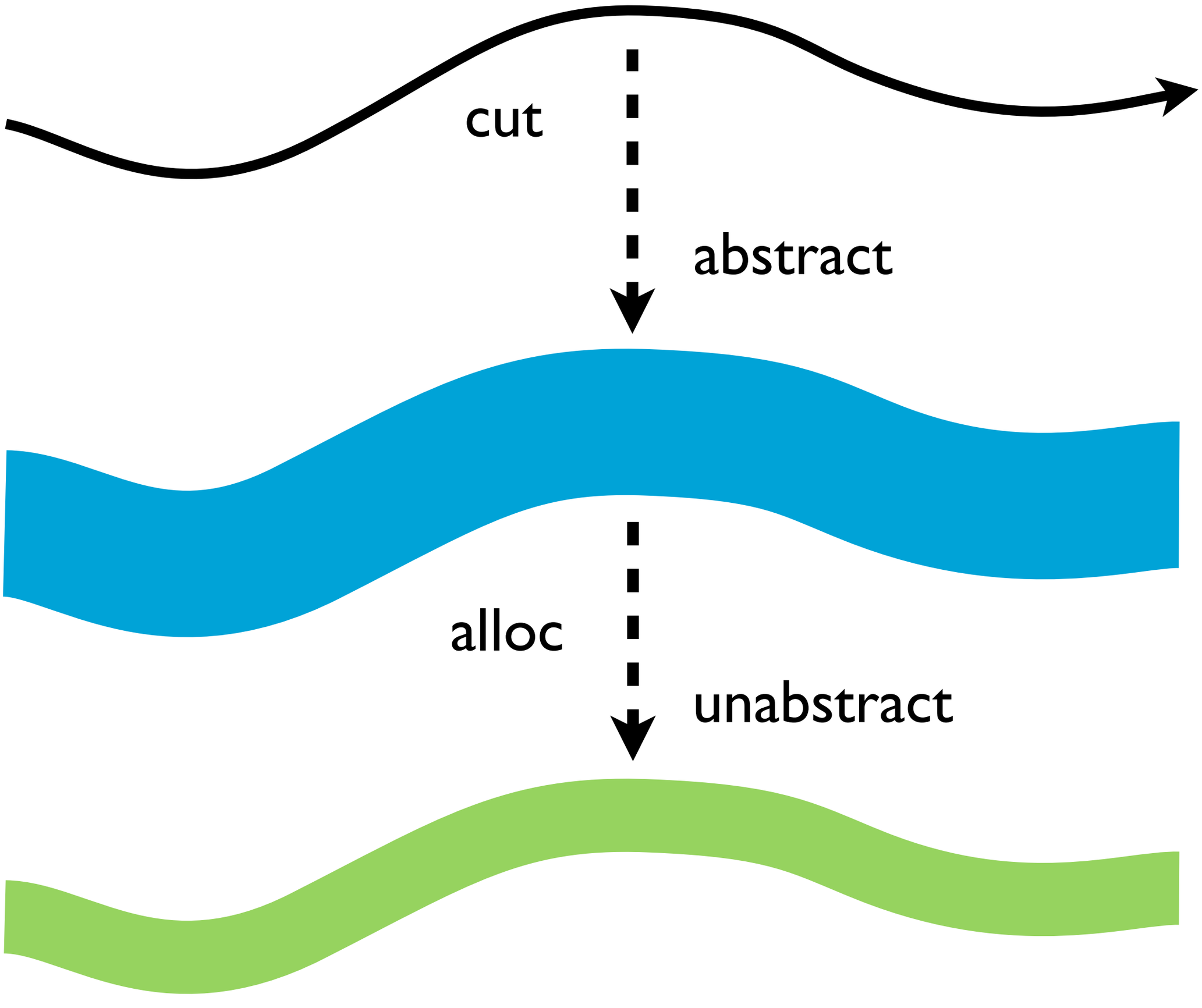

f()
g()
h()




cut


cut
abstract


cut
abstract
alloc


cut
abstract
alloc
unabstract

Talk!

Talk!

More in ICFP 2010, SAS 2010, Scheme 2010, SAS 2011