Control-flow analysis of higher-order programs: Part II

Matt Might
University of Utah
matt.might.net

Correction

- (Chaudhuri, 2008) implies subcubic 0CFA

- (Heintze, McAllaster, 1997) proved 0CFA 2NPDA-hard
- (Rytter, 1985) showed 2NPDA to be $O(n^3/\log n)$
- (Midtgaard & Van Horn, 2009) explicitly for 0CFA

Clarifications

- An analysis is **partially correct** if it is sound for terminating programs.
- An analysis is **totally correct** if it is sound for all programs.

Useful words, dangerous words

When two people use the same word, the word becomes useful.

Danvy's Law

Useful words, dangerous words

When two people use the same word, the word becomes useful.

Danvy's Law

When two people use the same word, but disagree on its meaning, the word becomes dangerous.

Might's Conjecture

High-level objective

To understand:

- Polyvariant
- Monovariant
- Context-sensitive
- Context-insensitive

How do you get better precision than 0CFA?

k-CFA (Shivers, 1991)

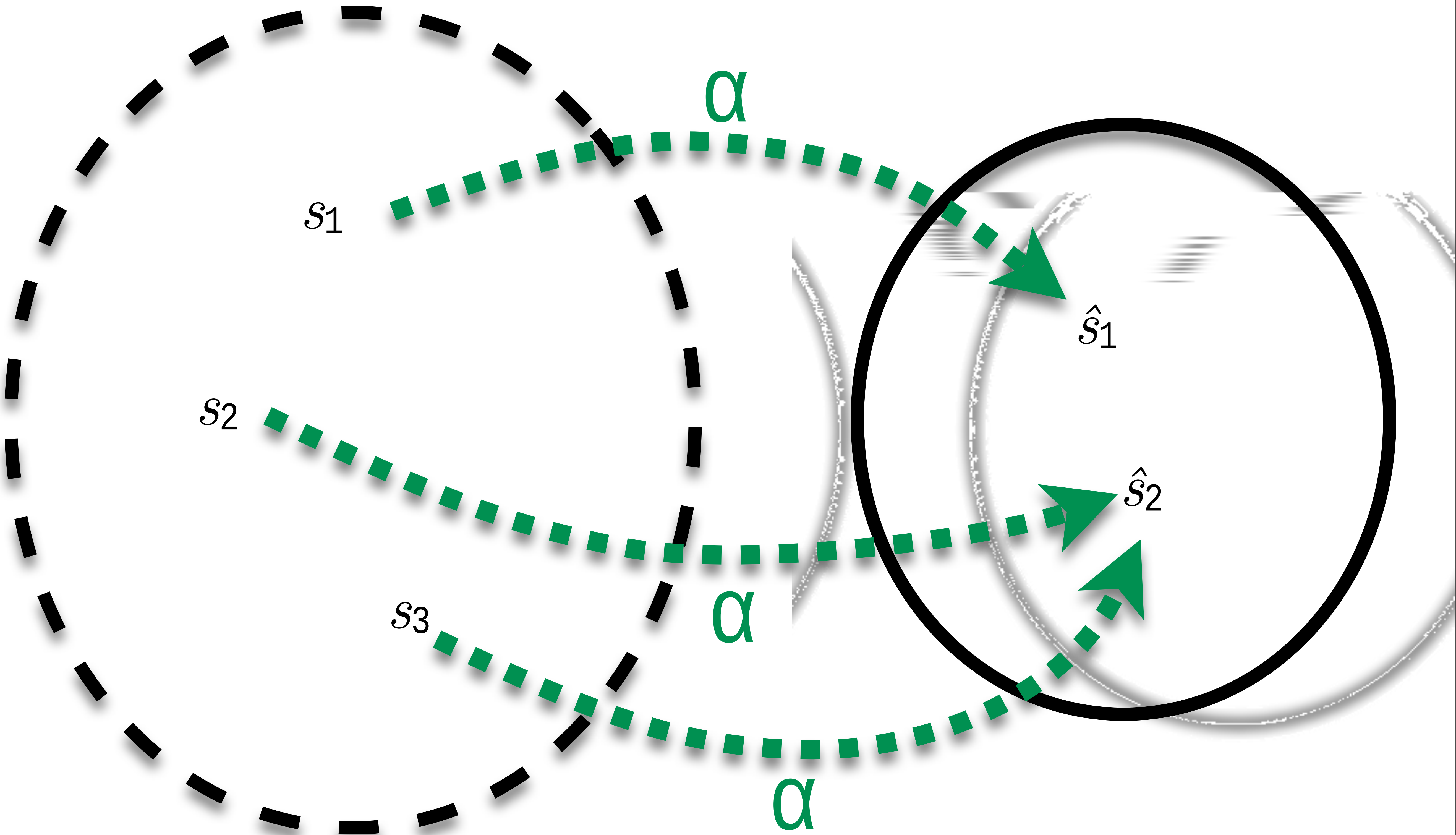
- A hierarchy of analyses, $k = 0, \ldots, \infty$
- $k + 1$ is always more precise than k
- $k = \infty$ is the concrete semantics

Observation

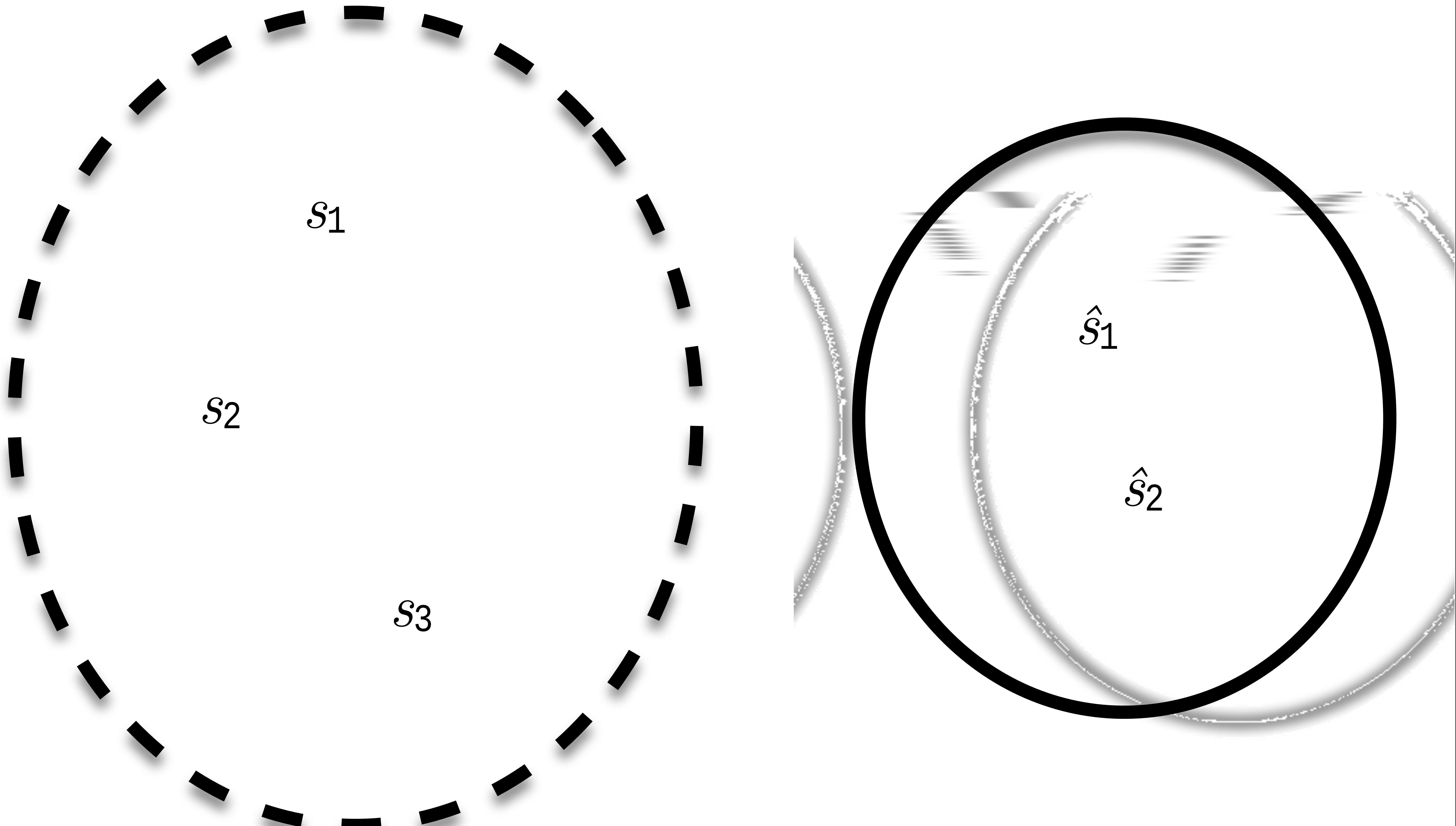
The structure of the abstraction influences precision.

Abstraction

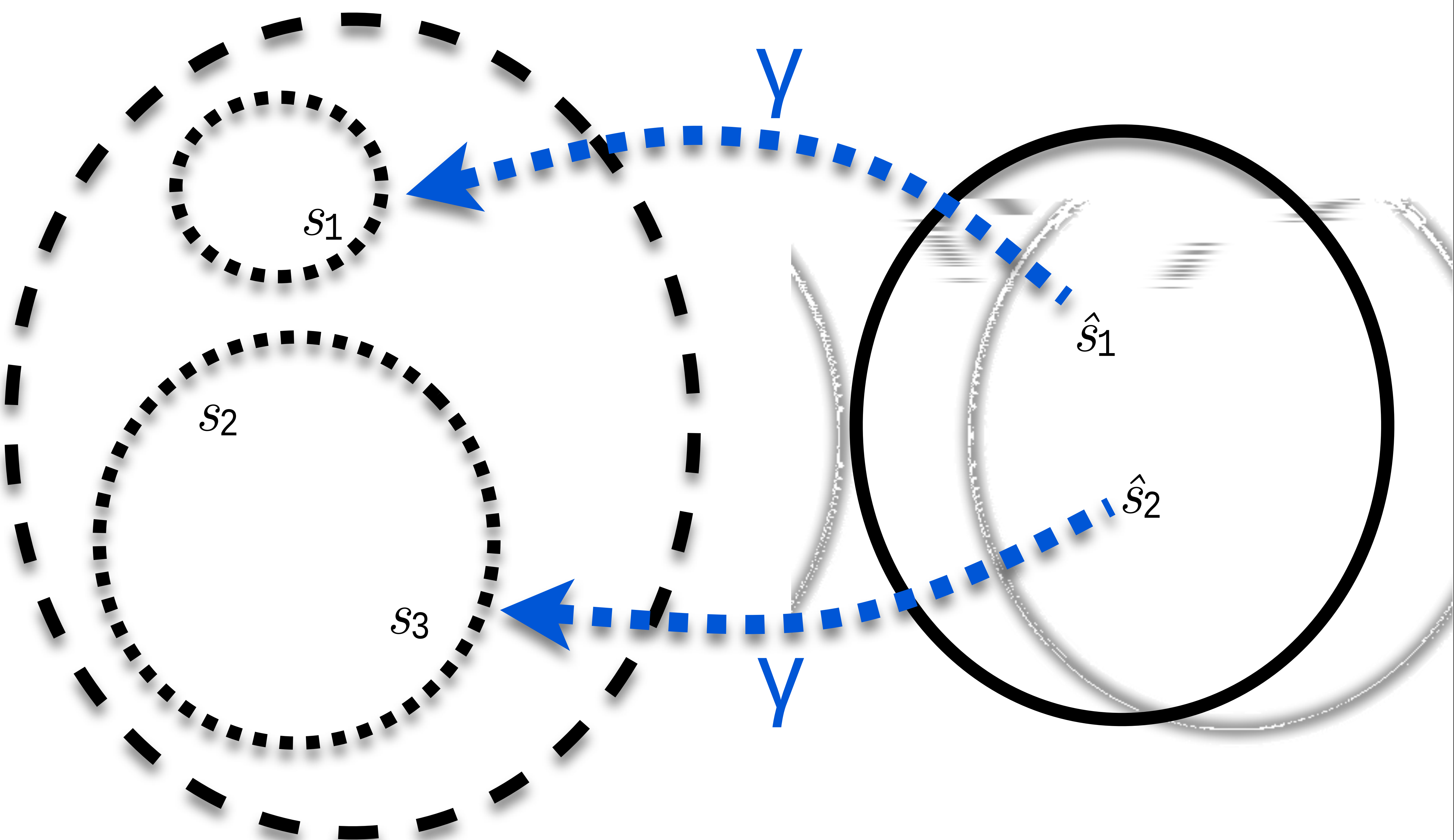
Abstraction



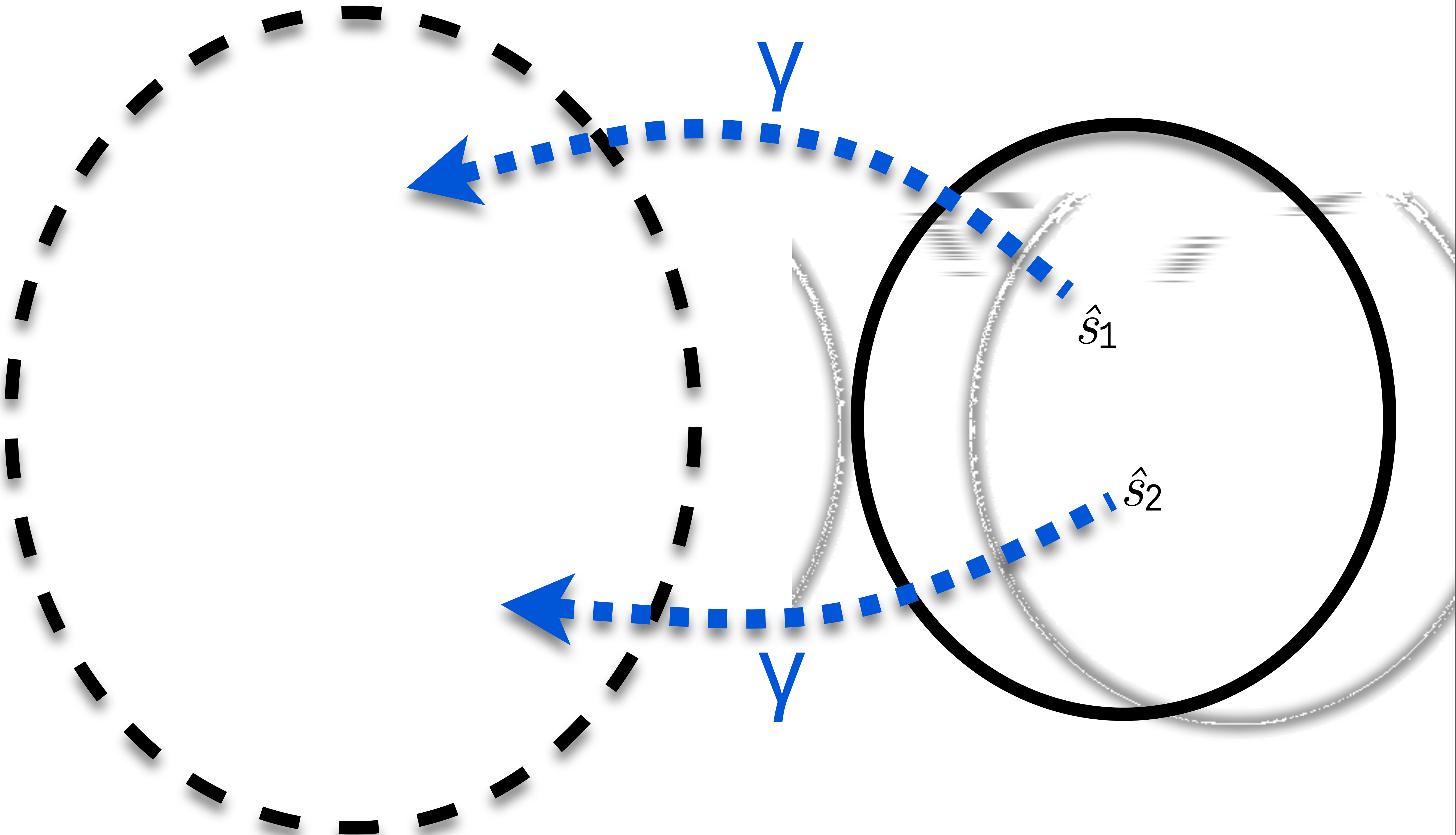
Concretization



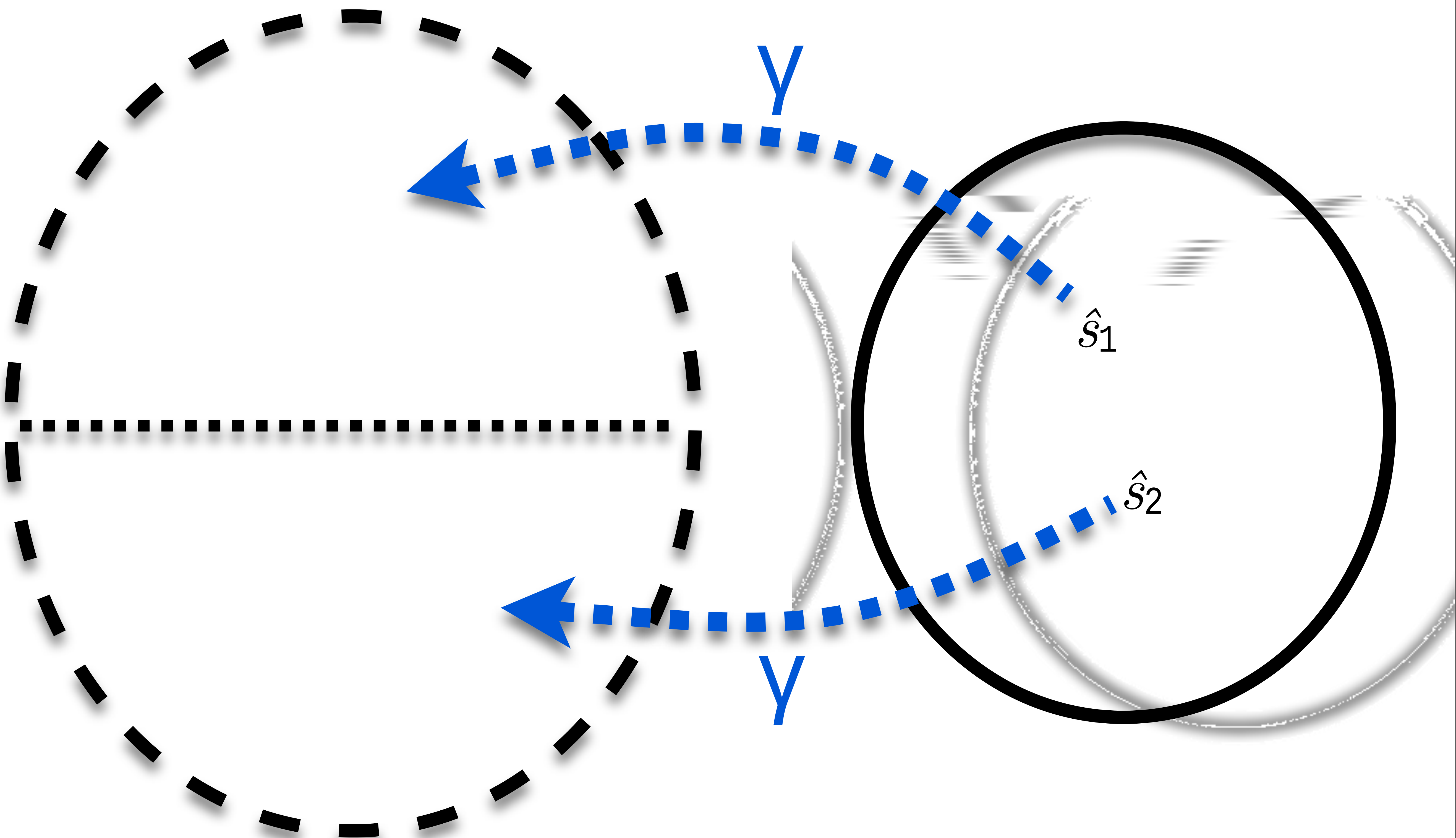
Concretization



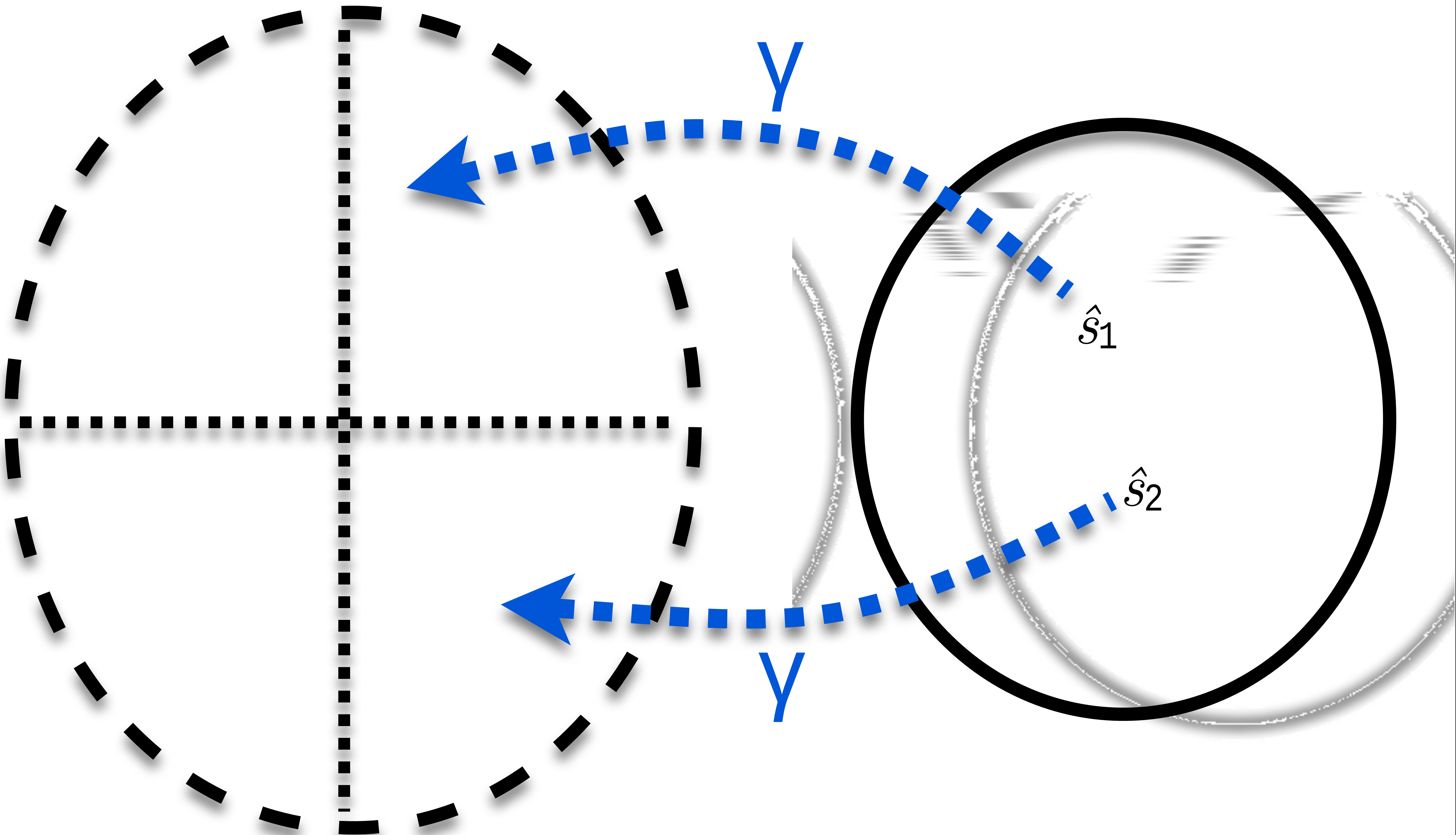
Partition



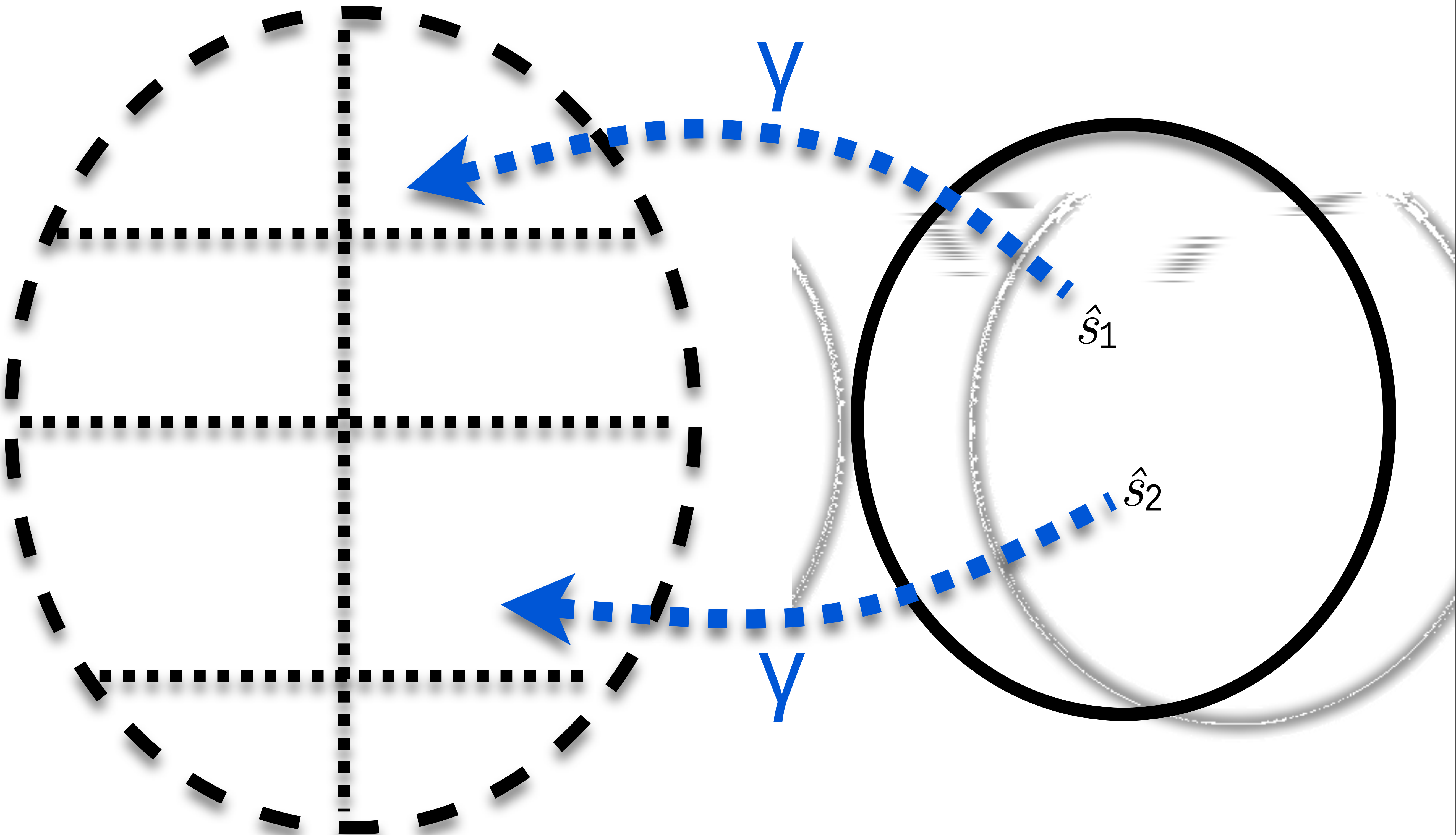
Partition



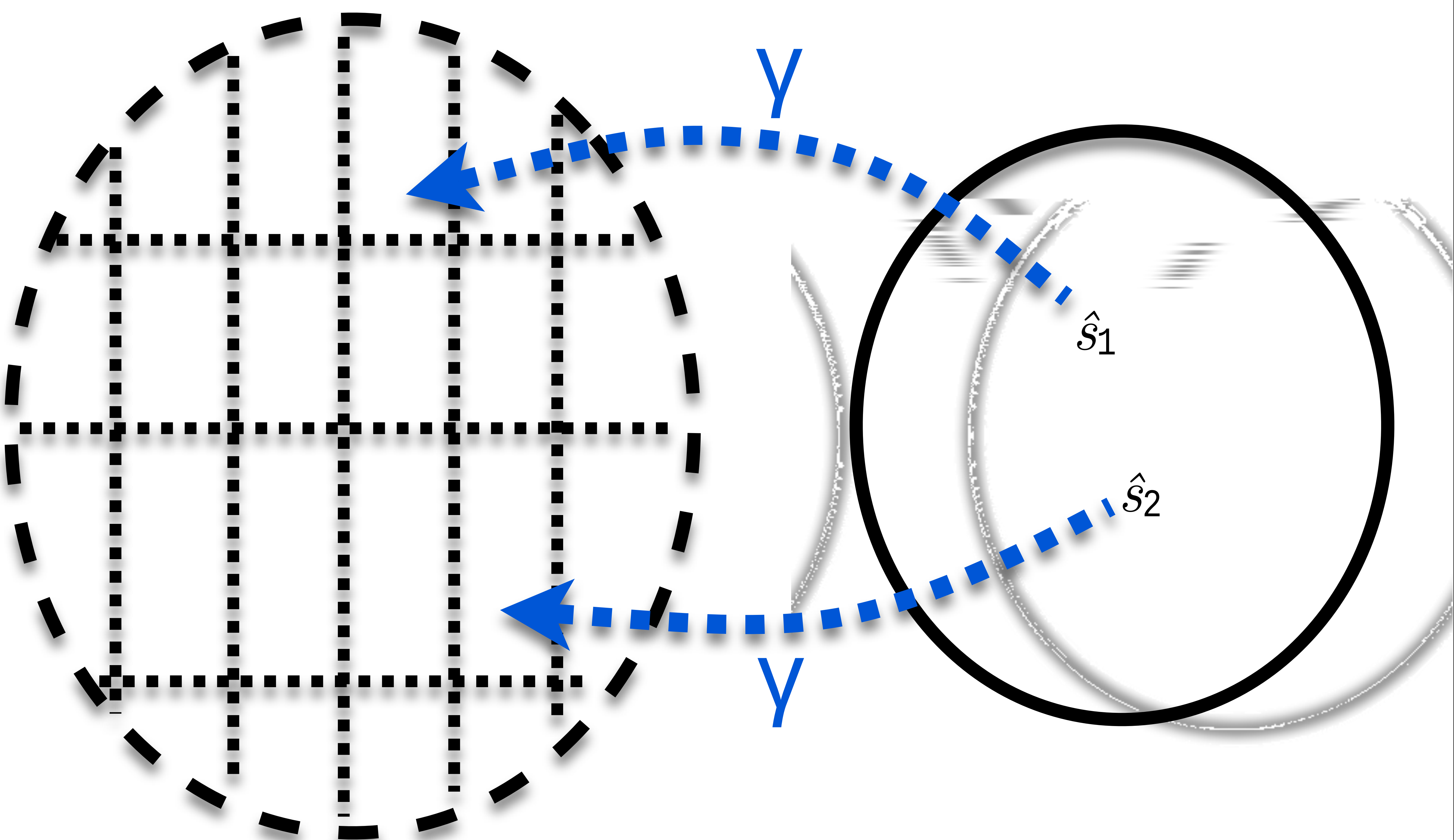
Partition



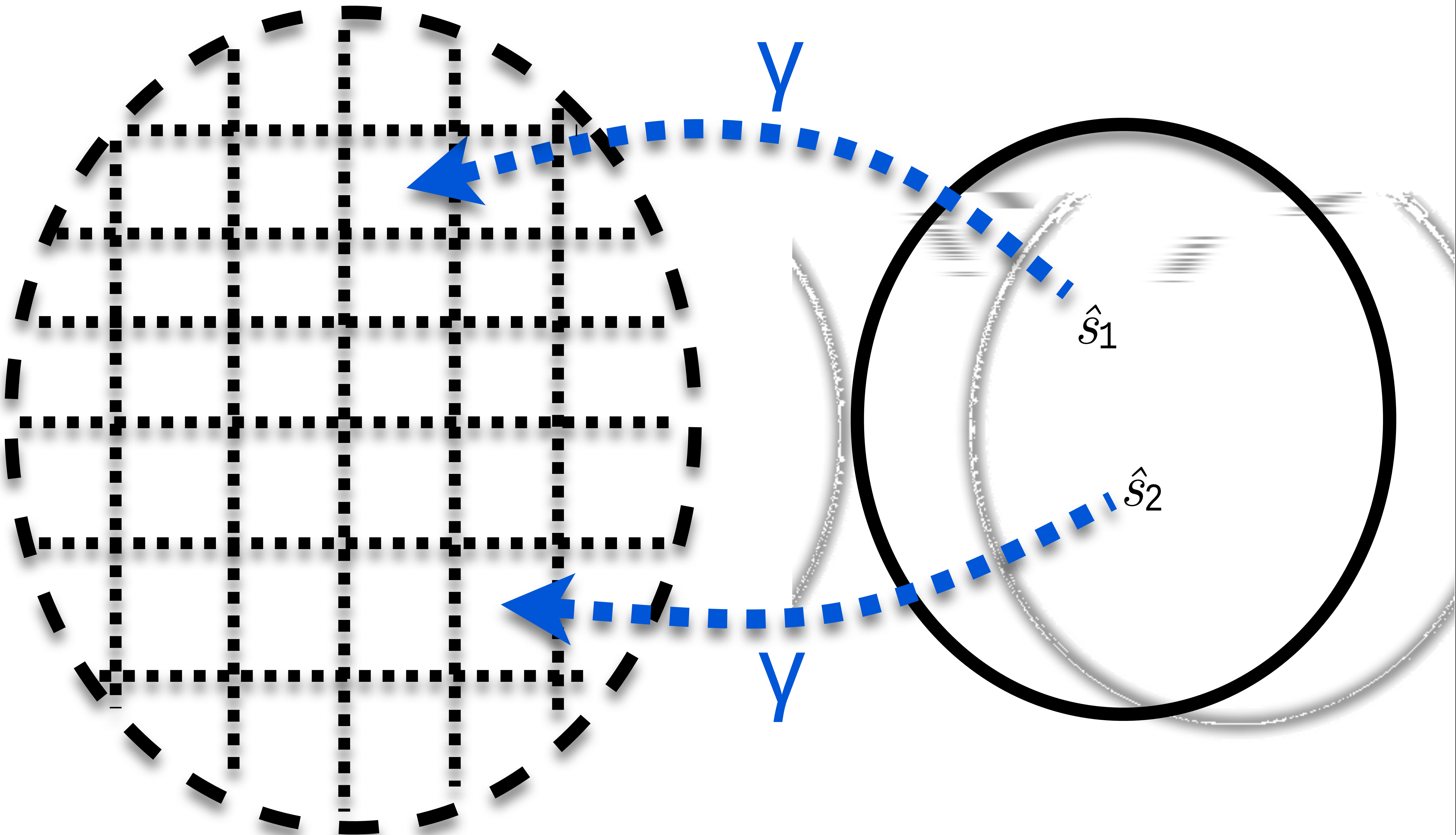
Partition



Partition



Partition



Problem

Concrete state-space unfriendly to abstraction:

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

$$\rho \quad Env = \mathsf{Var} \quad Clo$$

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

Standard solution

Reformulate the concrete semantics.

Standard solution

Reformulate the concrete semantics.

But, how?

Stumbling toward *k*-CFA

- Add `letrec` construct to CPS; or
- Add side effects construct to CPS; or
- Make state-space non-recursive

Stumbling toward *k*-CFA

- Add `letrec` construct to CPS; or
- Add side effects construct to CPS; or
- Make state-space non-recursive

↑

Reynolds' rule

CPS with `letrec`

$v \in$ Var

$f, e \in$ Exp = Var + Lam

$lam \in$ Lam ::= (λ ($v_1 \ldots v_n$) $call$)

$call \in$ Call ::= (f $e_1 \ldots e_n$)

CPS with `letrec`

$$v \in \text{Var}$$

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

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

$$call \in \text{Call} ::= \texttt{(}f\ e_1 \ldots e_n\texttt{)}$$

$$\mid\ \texttt{(letrec ((}v\ lam\texttt{))}\ call\texttt{)}$$

Options for `letrec`

Options for `letrec`

- Use **fix** to create infinite closures
- Desugar using the Y Combinator
- Binding-factor the environment

Factored environment

$$\rho \in Env = \mathsf{Var} \rightharpoonup \mathsf{Lam} \times Env$$

$$\beta \in BEnv = \mathsf{Var} \rightharpoonup Addr$$

$$\sigma \in Store = Addr \to \mathsf{Lam} \times BEnv$$

$$\rho \cong \sigma \circ \beta$$

Factored environment

$$\rho \in Env = \mathsf{Var} \rightharpoonup \mathsf{Lam} \times Env$$

$$\beta \in BEnv = \mathsf{Var} \rightharpoonup Addr$$

$$\sigma \in Store = Addr \rightarrow \mathsf{Lam} \times BEnv$$

$$\rho \cong \sigma \circ \beta$$

Factored environment

$$\rho \in Env = \mathsf{Var} \rightharpoonup \mathsf{Lam} \times Env$$

$$\beta \in BEnv = \mathsf{Var} \rightharpoonup Addr$$

$$\sigma \in Store = Addr \rightarrow \mathsf{Lam} \times BEnv$$

$$\rho \cong \sigma \circ \beta$$

Concrete state-space

$$
\begin{aligned}
\varsigma \quad \Sigma &= \mathsf{Call} \times \mathit{BEnv} \times \mathit{Store} \\
\beta \quad \mathit{BEnv} &= \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\sigma \quad \mathit{Store} &= \mathit{Addr} \rightharpoonup \mathit{Clo} \\
\mathit{clo} \quad \mathit{Clo} &= \mathsf{Lam} \times \mathit{BEnv}
\end{aligned}
$$

a $\quad$ Addr is a set of addresses

Environment look-up

- Look up address of variable: $\beta(v)$
- Look up value of address: σ

Factored evaluator

$$\mathcal{E}(v, \beta, \sigma) = \sigma(\beta(v))$$

$$\mathcal{E}(\qquad, \beta, \sigma) = (\qquad, \beta)$$

Environment addition

- Allocate fresh address: a'
- Extend binding environment: $\beta[v \mapsto a']$
- Extend store: $\sigma[a' \mapsto clo']$

But, where are we going to get a fresh address?

Options for freshness

1. Nondeterministically choose an address; or
2. Construct a total order on addresses; or
3. Thread time-stamp through the semantics

Constraints on time-stamps?

- Infinite in number.
- Monotonically increasable.

Concrete state-space (with time-stamps)

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

$$\beta \in BEnv = \mathsf{Var} \rightharpoonup Addr$$

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

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

$\mathsf{a} \in Addr$ is a set of addresses

$\mathsf{t} \in Time$ is a set of time-stamps

Concrete semantics

When $call = [\![(f\ e_1 \ldots e_n)]\!]$:

$$
\begin{aligned}
(call, \beta, \sigma, t) &\Rightarrow (call', \beta'', \sigma', t'), \text{ where} \\
(lam, \beta') &= \mathcal{E}(f, \beta, \sigma) \\
clo_i &= \mathcal{E}(e_i, \beta, \sigma) \\
lam &= [\![(\lambda\ (v_1 \ldots v_n)\ call')]\!] \\
t' &= tick(call, t) \\
a_i &= alloc(v_i, t') \\
\beta'' &= \beta'[v_i \mapsto a_i] \\
\sigma' &= \sigma[a_i \mapsto clo_i]
\end{aligned}
$$

Concrete semantics

When $call = [\![\texttt{(letrec ((}v\ lam\texttt{))}\ call'\texttt{)}]\!]$:

$$
\begin{aligned}
(call, \beta, \sigma, t) \Rightarrow (call', \beta', \sigma', t'), \text{ where}\\
t' &= tick(call, t)\\
a &= alloc(v, t')\\
\beta' &= \beta[v \mapsto a]\\
clo &= \mathcal{E}(lam, \beta', \sigma)\\
\sigma' &= \sigma[a \mapsto clo]
\end{aligned}
$$

Concrete parameters

$$tick : \text{Call} \times Time \rightarrow Time$$

$$alloc : \text{Var} \times Time \rightarrow Addr$$

Constraints, formalized

$t < tick(call, t)$ [monotonicity]

If $v \neq v'$, then $alloc(v, t) \neq alloc(v', t)$. [local uniqueness]

If $t \neq t'$, then $alloc(v, t) \neq alloc(v', t')$. [global uniqueness]

The easy solution

$$Time = \mathbb{N}$$

$$Addr = \mathsf{Var} \times Time$$

$$tick(\_, t) = t + 1$$

$$alloc(v, t) = (v, t)$$

Abstracting to *k*-CFA

k-CFA state-space

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

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

$\hat{t} \in \widehat{Time}$ is a **finite** set of time-stamps

k-CFA state-space

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

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

$\hat{t} \in \widehat{Time}$ is a **finite** set of time-stamps

Is this state-space finite?

Non-recursive

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

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

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \to \widehat{Addr}$$

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

$\hat{t} \in \widehat{Time}$ is a **finite** set of time-stamps

Non-recursive

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

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

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \to \widehat{Addr}$$

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

$\hat{t} \in \widehat{Time}$ is a **finite** set of time-stamps

Abstraction maps

$$\alpha(call, \beta, \sigma, t) = (call, \alpha(\beta), \alpha(\sigma), \alpha(t))$$

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

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

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

$\alpha(a)$ is set by parameter

$\alpha(t)$ is set by parameter

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

k-CFA components

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

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

$$\hat{\mathcal{E}} : \mathsf{Exp} \times \widehat{BEnv} \times \widehat{Store} \rightarrow \mathcal{P}\left(\widehat{Clo}\right)$$

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

$$\hat{\mathcal{E}}(lam, \hat{\beta}, \hat{\sigma}) = \left\{(lam, \hat{\beta})\right\}$$

Semantics for k-CFA

When $call = [\![(f\ e_1 \ldots e_n)]\!]$:

$$
\begin{aligned}
(call, \hat{\beta}, \hat{\sigma}, \hat{t}) &\rightsquigarrow (call', \hat{\beta}'', \hat{\sigma}', \hat{t}'), \text{ where} \\
(lam, \hat{\beta}') &\in \hat{\mathcal{E}}(f, \hat{\beta}, \hat{\sigma}) \\
\hat{C}_i &= \hat{\mathcal{E}}(e_i, \hat{\beta}, \hat{\sigma}) \\
lam &= [\![(\lambda\ (v_1 \ldots v_n)\ call')]\!] \\
\hat{t}' &= \widehat{tick}(call, \hat{t}) \\
\hat{a}_i &= \widehat{alloc}(v_i, \hat{t}') \\
\hat{\beta}'' &= \hat{\beta}'[v_i \mapsto \hat{a}_i] \\
\hat{\sigma}' &= \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{C}_i]
\end{aligned}
$$

Semantics for *k*-CFA

When $call$ = $[\![$`(letrec ((`v lam`))` $call'$`)`$]\!]$:

$$(call, \hat{\beta}, \hat{\sigma}, \hat{t}) \leadsto (call', \hat{\beta}', \hat{\sigma}', \hat{t}'), \text{ where}$$

$$\hat{t}' = \widehat{tick}(call, \hat{t})$$

$$\hat{a} = \widehat{alloc}(v, \hat{t}')$$

$$\hat{\beta}' = \hat{\beta}[v \mapsto \hat{a}]$$

$$\hat{C} = \hat{\mathcal{E}}(lam, \hat{\beta}', \hat{\sigma})$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a} \mapsto \hat{C}]$$

Parameters for *k*-CFA

$$\widehat{tick} : \text{Call} \times \widehat{Time} \rightarrow \widehat{Time}$$

$$\widehat{alloc} : \text{Var} \times \widehat{Time} \rightarrow \widehat{Addr}$$

Exercise: Add `let`

When $call = [\![$`(let ((`v e`))` $call'$`)`$]\!]$:

$$(call, \beta, \sigma, t) \Rightarrow (call', \beta', \sigma', t'), \text{ where}$$

Exercise: Add `let`

When $call = [\![(\texttt{let}\ ((v\ e))\ call')]\!]$:

$$(call, \beta, \sigma, t) \Rightarrow (call', \beta', \sigma', t'), \text{ where}$$

When $call = [\![(f\ e_1 \ldots e_n)]\!]$:

$$(call, \beta, \sigma, t) \Rightarrow (call', \beta'', \sigma', t'), \text{ where}$$

$$(lam, \beta') = \mathcal{E}(f, \beta, \sigma)$$

$$clo_i = \mathcal{E}(e_i, \beta, \sigma)$$

$$lam = [\![(\lambda\ (v_1 \ldots v_n)\ call')]\!]$$

$$t' = tick(call, t)$$

$$a_i = alloc(v_i, t')$$

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

$$\sigma' = \sigma[a_i \mapsto clo_i]$$

Exercise: Add `let`

When $call = [\![(\texttt{let ((}v\ e\texttt{))}\ call'\texttt{)}]\!]$:

$$(call, \beta, \sigma, t) \Rightarrow (call', \beta', \sigma', t'), \text{ where}$$

$$t' = tick(call, t)$$

$$a = alloc(v, t')$$

$$clo = \mathcal{E}(e, \beta, \sigma)$$

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

$$\sigma' = \sigma[a \mapsto clo]$$

Exercise: Add `let`

When $call = [\![$`(let ((`v e`))` $call'$`)`$]\!]$:

$$(call, \hat{\beta}, \hat{\sigma}, \hat{t}) \leadsto (call', \hat{\beta}', \hat{\sigma}', \hat{t}'), \text{ where}$$

Exercise: Add `let`

When $call = [\![\texttt{(let ((}v\ e\texttt{))}\ call'\texttt{)}]\!]$:

$$(call, \hat{\beta}, \hat{\sigma}, \hat{t}) \leadsto (call', \hat{\beta}', \hat{\sigma}', \hat{t}'), \text{ where}$$

$$\hat{t}' = \widehat{tick}(call, \hat{t})$$

$$\hat{a} = \widehat{alloc}(v, \hat{t}')$$

$$\hat{C} = \hat{\mathcal{E}}(e, \hat{\beta}, \hat{\sigma})$$

$$\hat{\beta}' = \hat{\beta}[v \mapsto \hat{a}]$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a} \mapsto \hat{C}]$$

Constraints on parameters

If $\alpha(t) \sqsubseteq \hat{t}$, then $\alpha(tick(call, t)) \sqsubseteq \widehat{tick}(call, \hat{t})$.

If $\alpha(t) \sqsubseteq \hat{t}$, then $\alpha(alloc(v, t)) \sqsubseteq \widehat{alloc}(v, \hat{t})$.

The *k*-CFA solution

$$Time = \mathsf{Call}^*$$

$$Addr = \mathsf{Var} \times Time$$

$$\widehat{Time} = \mathsf{Call}^k$$

$$\widehat{Addr} = \mathsf{Var} \times \widehat{Time}$$

$$tick(call, t) = call : t$$

$$alloc(v, t) = (v, t)$$

$$\widehat{tick}(call, \hat{t}) = \mathit{first}_k(call : \hat{t})$$

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

Abstraction maps

$$\alpha(t) = \mathit{first}_k(t)$$
$$\alpha(v, t) = (v, \alpha(t))$$

Context-sensitivity and polyvariance

Context-sensitivity

- Times determine **context-sensitivity**
- Higher *k* retains more context/history
- Bigger *k* means bigger abstract state-space

State-space and *k*

$$
\begin{aligned}
|\hat{\Sigma}| = {} & |\mathsf{Call}| \\
& \times (|\mathsf{Var}| \times | \mathsf{Call}|^k)^{|\mathsf{Var}|} && (|\widehat{BEnv}|) \\
& \times \left(2^{|\mathsf{Lam}| \times (|\mathsf{Var}| \times |\mathsf{Call}|^k)\ \mathsf{Var}}\right)^{|\mathsf{Var}| \times |\mathsf{Call}|^k} && (|\widehat{Store}|) \\
& \times |\mathsf{Call}|^k && (|\widehat{Time}|)
\end{aligned}
$$

Polyvariance

Polyvariance = max abstract addresses per variable.

In *k*-CFA, polyvariance is $|Call|^k$.

k = 0 is monovariant.

$$k = 0, k = 1$$

Solving for 0CFA

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

$$\hat{a} \in \widehat{Addr} = \mathsf{Var} \times \widehat{Time}$$

$$\hat{t} \in \widehat{Time} = \{\langle\rangle\}$$

Solving for 0CFA

$\hat{\varsigma} \in \hat{\Sigma} = \mathsf{Call}$ ~~$\times \widehat{BEnv}$~~ $\times \widehat{Store}$ ~~$\times \widehat{Time}$~~

~~$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightharpoonup \widehat{Addr}$~~

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

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

$\hat{a} \in \widehat{Addr} = \mathsf{Var}$ ~~$\times \widehat{Time}$~~

~~$\hat{t} \in \widehat{Time} = \{()\}$~~

Solving for 0CFA

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

$$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \rightarrow \mathcal{P}\left(\mathsf{Lam}\right)$$

Solving for 0CFA

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

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

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

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

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

Solving for 0CFA

$$([\![(f\ e_1 \ldots e_n)]\!], \hat{\sigma}) \leadsto (call', \hat{\sigma}'), \text{ where}$$

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

$$L_i = \hat{\mathcal{E}}(e_i, \hat{\sigma})$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [v_i \mapsto L_i]$$

Solving for 0CFA

$$(\llbracket \texttt{(letrec ((}v\ lam\texttt{))}\ call'\texttt{)} \rrbracket, \hat{\sigma}) \leadsto (call', \hat{\sigma}'), \text{ where}$$

$$\hat{\sigma}' = \hat{\sigma} \quad [v \to \{lam\}]$$

Solving for 1CFA

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

$$\hat{\mathsf{a}} \in \widehat{Addr} = \mathsf{Var} \times \widehat{Time}$$

$$\hat{\mathsf{t}} \in \widehat{Time} \cong \mathsf{Call}$$

Solving for 1CFA

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

$$\hat{a} \in \widehat{Addr} = \mathsf{Var} \times \widehat{Time}$$

$$\hat{t} \in \widehat{Time} \simeq \mathsf{Call}$$

Solving for 1CFA

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

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightarrow \widehat{Addr}$$

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

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

$$\hat{\mathsf{a}} \in \widehat{Addr} = \mathsf{Var} \times \mathsf{Call}$$

Solving for 1CFA

When $call = [\![(f\ e_1 \ldots e_n)]\!]$:

$$
\begin{aligned}
(call, \hat{\beta}, \hat{\sigma}) &\leadsto (call', \hat{\beta}'', \hat{\sigma}'), \text{ where} \\
(lam, \hat{\beta}') &\in \hat{\mathcal{E}}(f, \hat{\beta}, \hat{\sigma}) \\
\hat{C}_i &= \hat{\mathcal{E}}(e_i, \hat{\beta}, \hat{\sigma}) \\
lam &= [\![(\lambda\ (v_1 \ldots v_n)\ call')]\!] \\
\hat{\beta}'' &= \hat{\beta}'[v_i \mapsto (v_i, call)] \\
\hat{\sigma}' &= \hat{\sigma} \sqcup [(v_i, call) \rightarrow \hat{C}_i]
\end{aligned}
$$

Exercise

$$call_1 = \texttt{(let ((id (λ (x q) } \overbrace{\texttt{(q x)}}^{call_{\texttt{q}}} \texttt{)))}$$

$$call_2 = \texttt{(id 3 (λ (v1)}$$

$$call_3 = \texttt{(id 4 (λ (v2)}$$

$$call_4 = \texttt{(halt v2))))}$$

Exercise

When $call = [\![\texttt{(}f\ e_1 \ldots e_n\texttt{)}]\!]$:

$$call_1 = \texttt{(let ((id (λ (x q)}\ \overbrace{\texttt{(q x)}}^{call_{\texttt{q}}}\ \texttt{)))}$$

$$call_2 = \quad \texttt{(id 3 (λ (v1)}$$

$$call_3 = \qquad \texttt{(id 4 (λ (v2)}$$

$$call_4 = \qquad\quad \texttt{(halt v2))))}$$

$(call, \hat{\beta}, \hat{\sigma}) \rightsquigarrow (call', \hat{\beta}'', \hat{\sigma}')$, where

$$(lam, \hat{\beta}') \in \hat{\mathcal{E}}(f, \hat{\beta}, \hat{\sigma})$$

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

$$lam = [\![\texttt{(λ (}v_1 \ldots v_n\texttt{)}\ call'\texttt{)}]\!]$$

$$\hat{\beta}'' = \hat{\beta}'[v_i \mapsto (v_i, call)]$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [(v_i, call) \rightarrow \hat{C}_i]$$

$$(call_1, \bot)$$

$$(call_2, [\texttt{id} \mapsto (\texttt{id}, call_1)],$$
$$[(\texttt{id}, call_1) \mapsto \{(\lambda_1, \bot)\}])$$

Answer

$$call_1 = \texttt{(let ((id (λ (x q) } \overbrace{\texttt{(q x)}}^{call_q} \texttt{)))}$$
$$call_2 = \texttt{(id 3 (λ (v1)}$$
$$call_3 = \texttt{(id 4 (λ (v2)}$$
$$call_4 = \texttt{(halt v2))))}$$

$$(call_1, \bot)$$

$$(call_2, [\texttt{id} \mapsto (\texttt{id}, call_1)],$$
$$[(\texttt{id}, call_1) \mapsto \{(\lambda_1, \bot)\}])$$

$$(call_\texttt{q}, [\texttt{q} \mapsto (\texttt{q}, call_2), \texttt{x} \mapsto (\texttt{x}, call_2)]$$
$$[(\texttt{q}, call_2) \mapsto \{(\lambda_2, [\texttt{id} \mapsto (\texttt{id}, call_1)])\}$$
$$(\texttt{x}, call_2) \mapsto \{3\}, \ldots])$$

$$(call_3, [\texttt{v1} \mapsto (\texttt{v1}, call_\texttt{q}), \ldots], [(\texttt{v1}, call_\texttt{q}) \mapsto \{3\}, \ldots])$$

$$(call_\texttt{q}, [\texttt{q} \mapsto (\texttt{q}, call_3), \texttt{x} \mapsto (\texttt{x}, call_3)],$$
$$[(\texttt{q}, call_3) \mapsto \left\{(\lambda_3, [\texttt{v1} \mapsto (\texttt{v1}, call_\texttt{q}), \ldots])\right\}$$
$$(\texttt{x}, call_3) \mapsto \{3\}, \ldots])$$

$$(call_4 [\texttt{v2} \mapsto (\texttt{v2}, call_\texttt{q}), \ldots],$$
$$[(\texttt{v2}, call_\texttt{q}) \mapsto \{4\}, \ldots])$$

0CFA *v.* **1CFA**

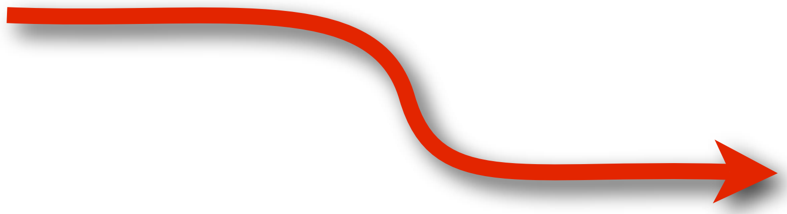
```
(f x)

                    (λ (q) ...)


(f a)
```

0CFA *v.* **1CFA**

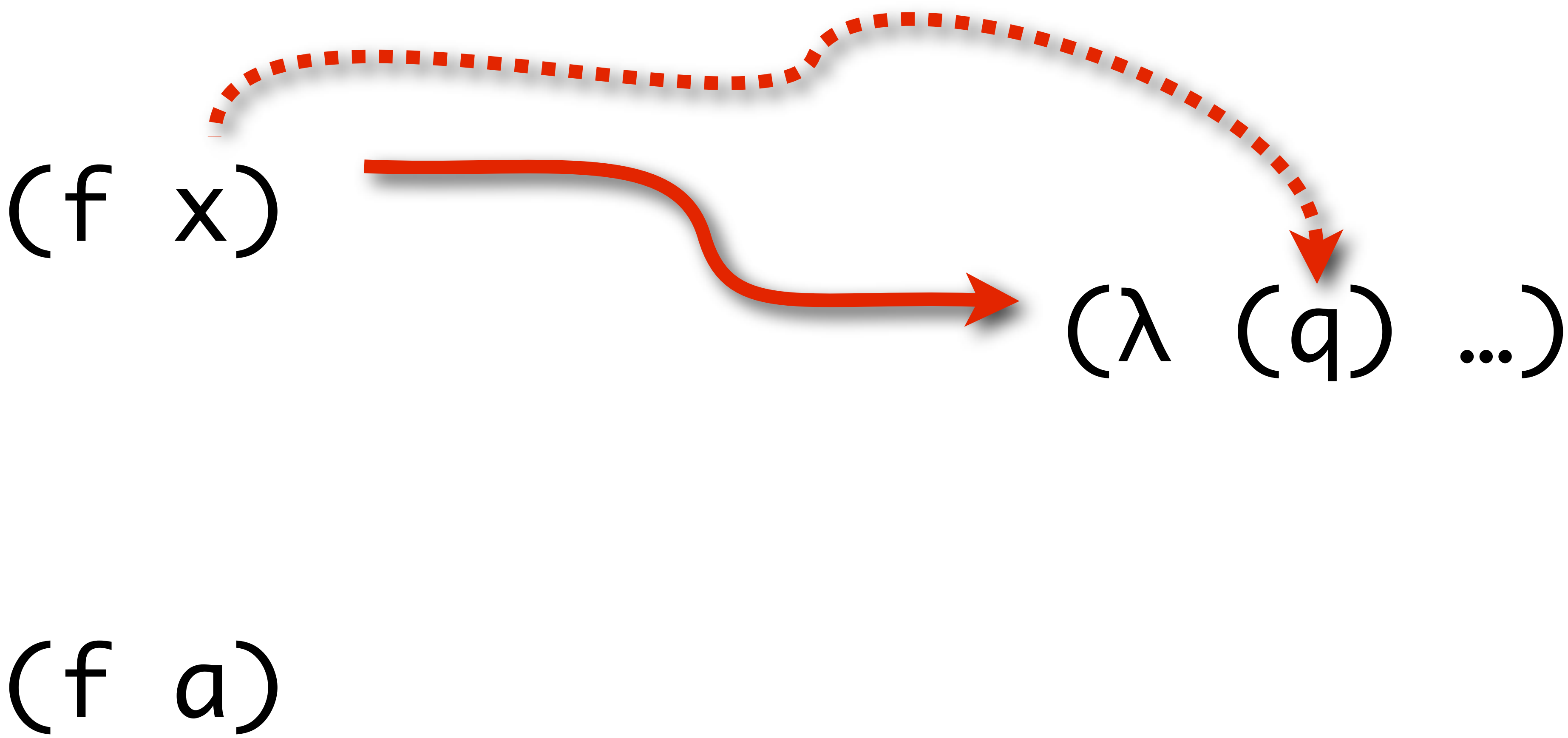
```
(f x)
```



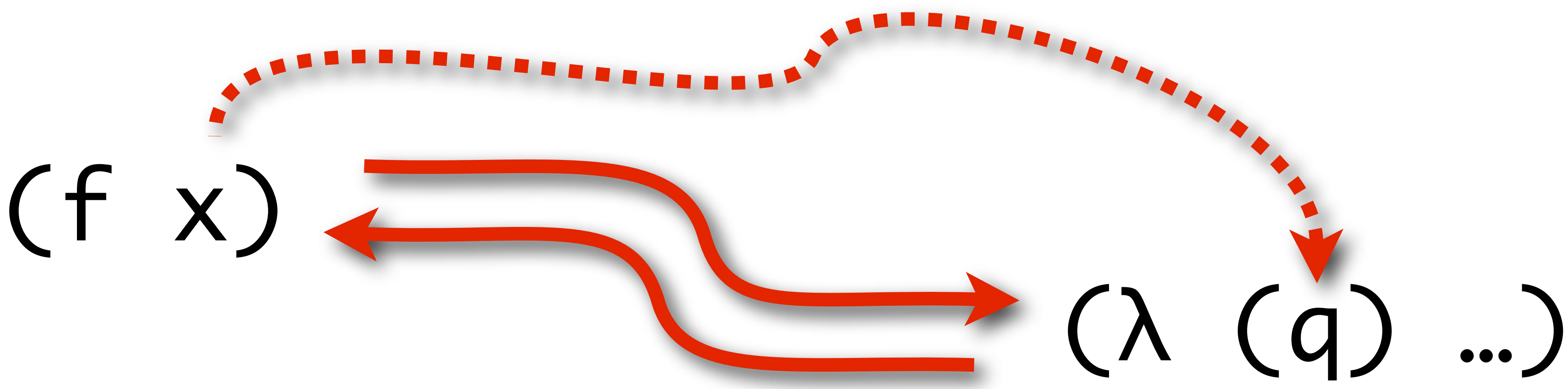
```
(λ (q) ...)
```

```
(f a)
```

0CFA *v.* **1CFA**

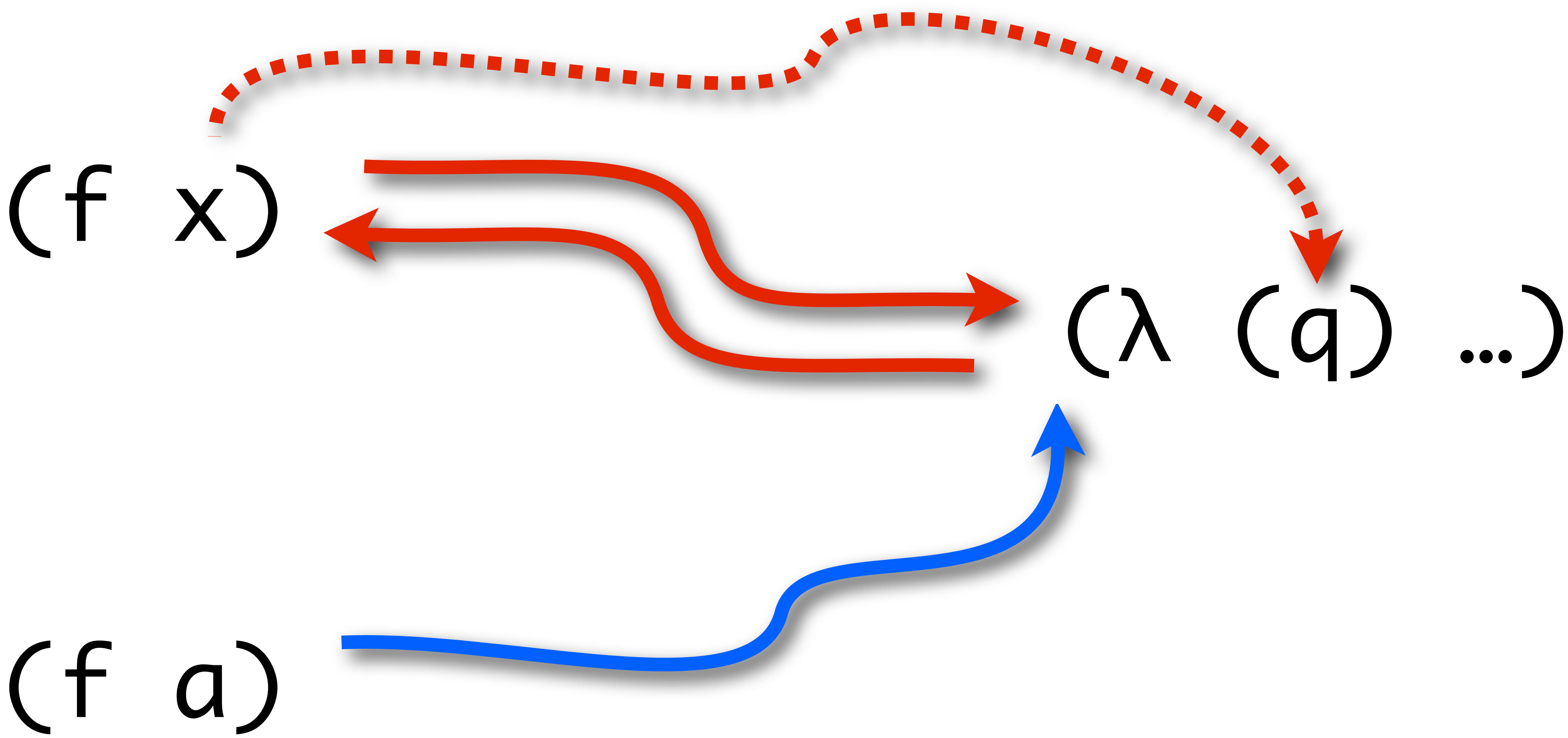


0CFA *v.* **1CFA**

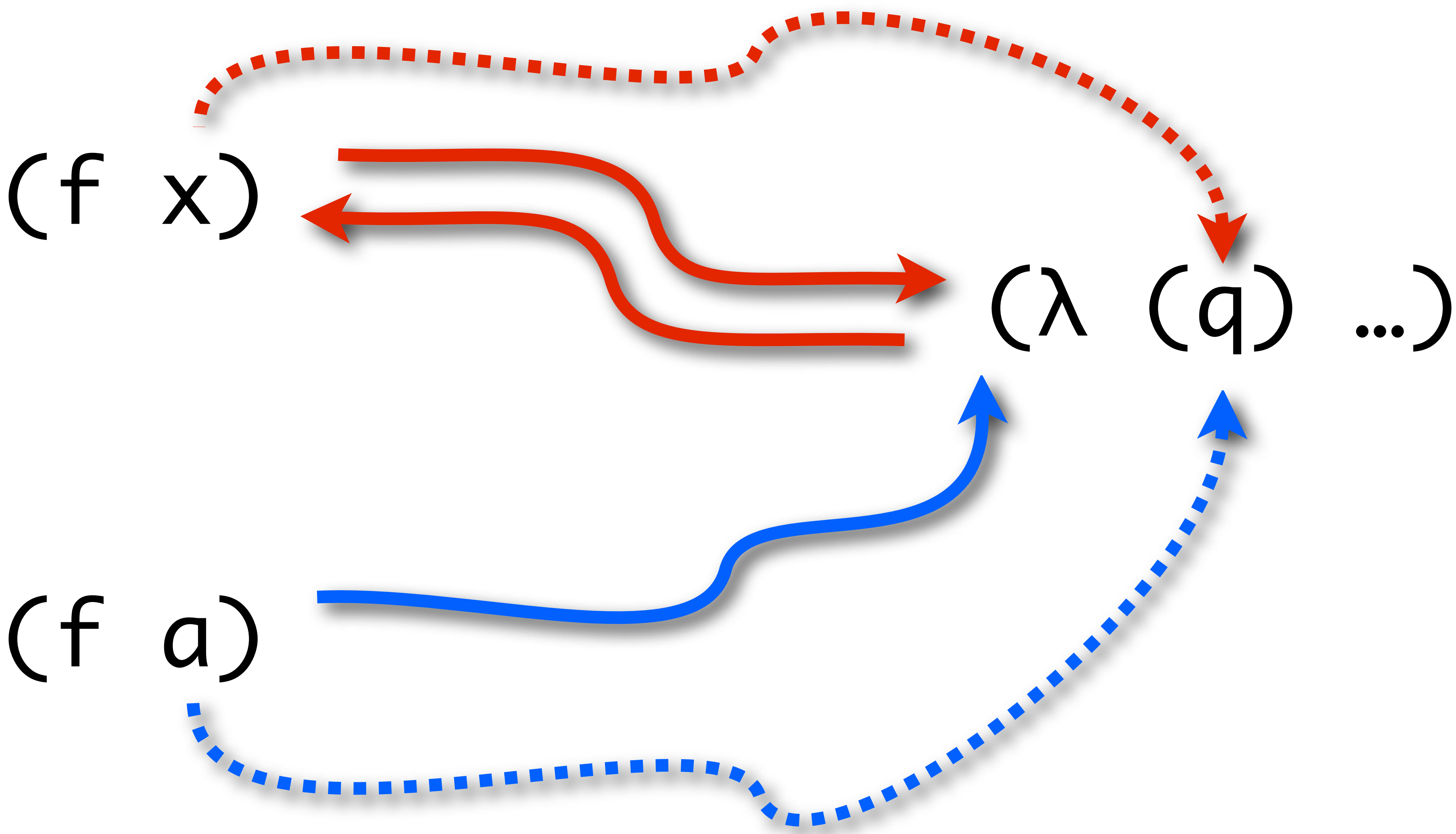


(f a)

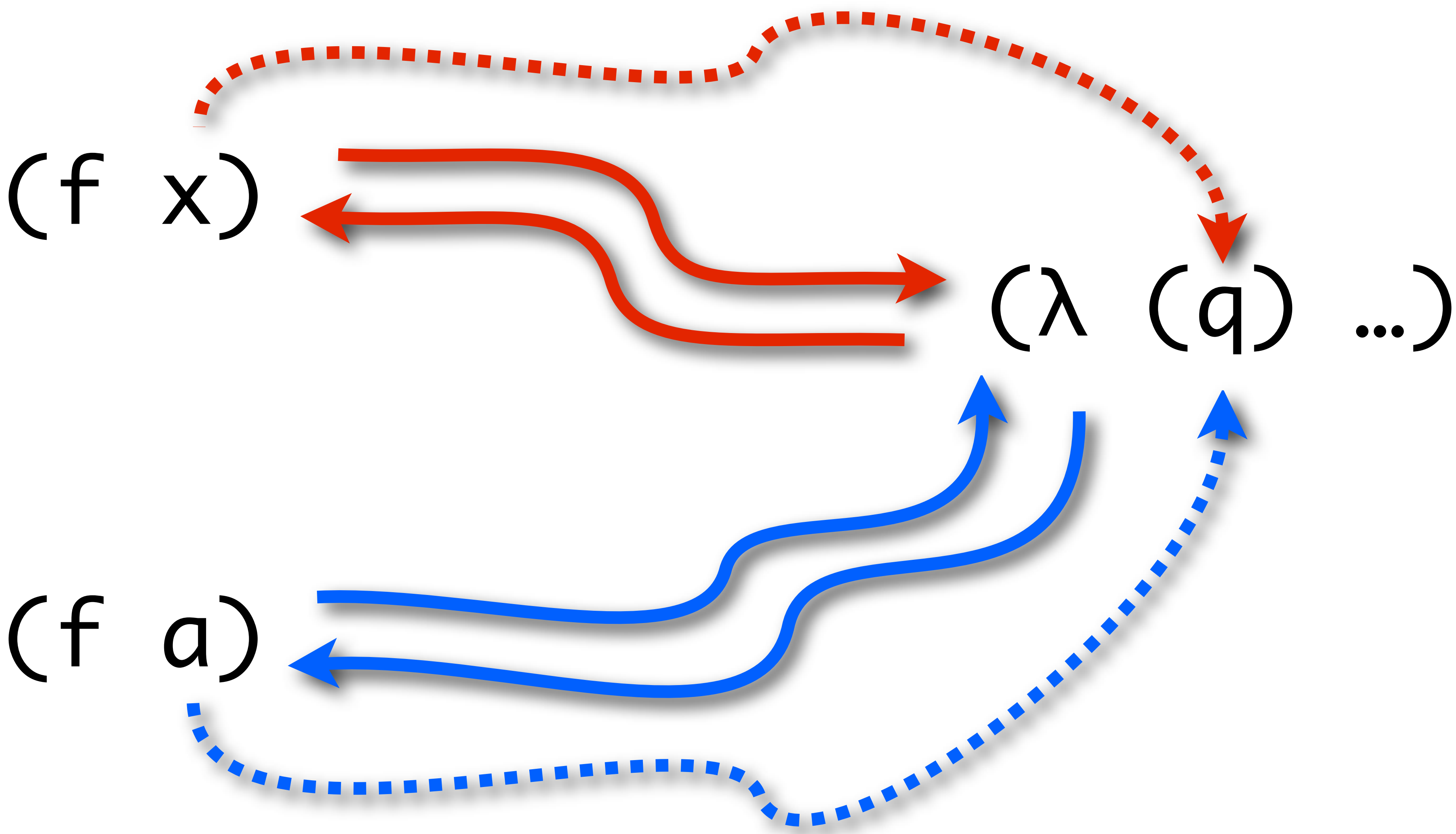
0CFA *v.* **1CFA**



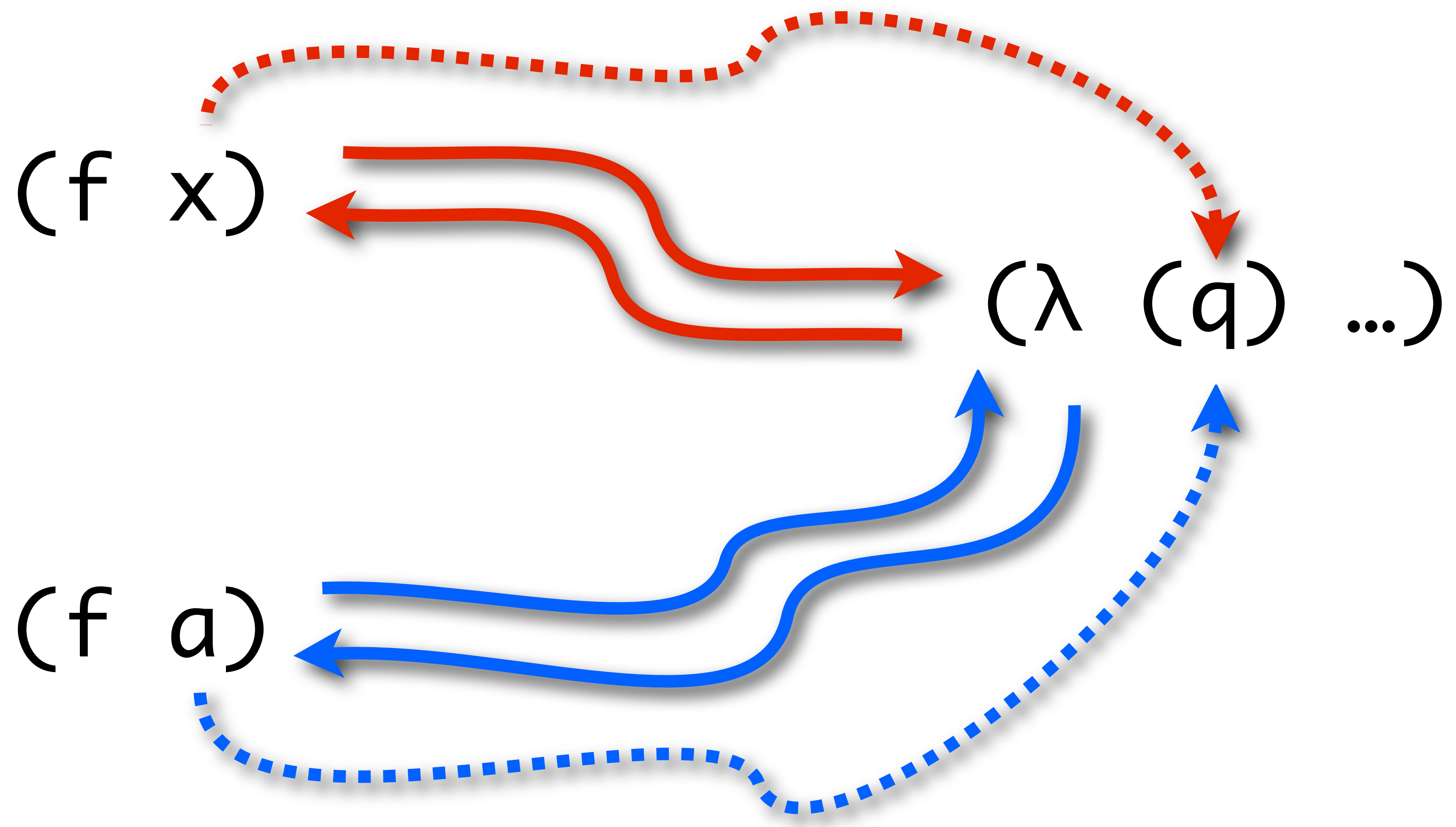
0CFA *v.* **1CFA**



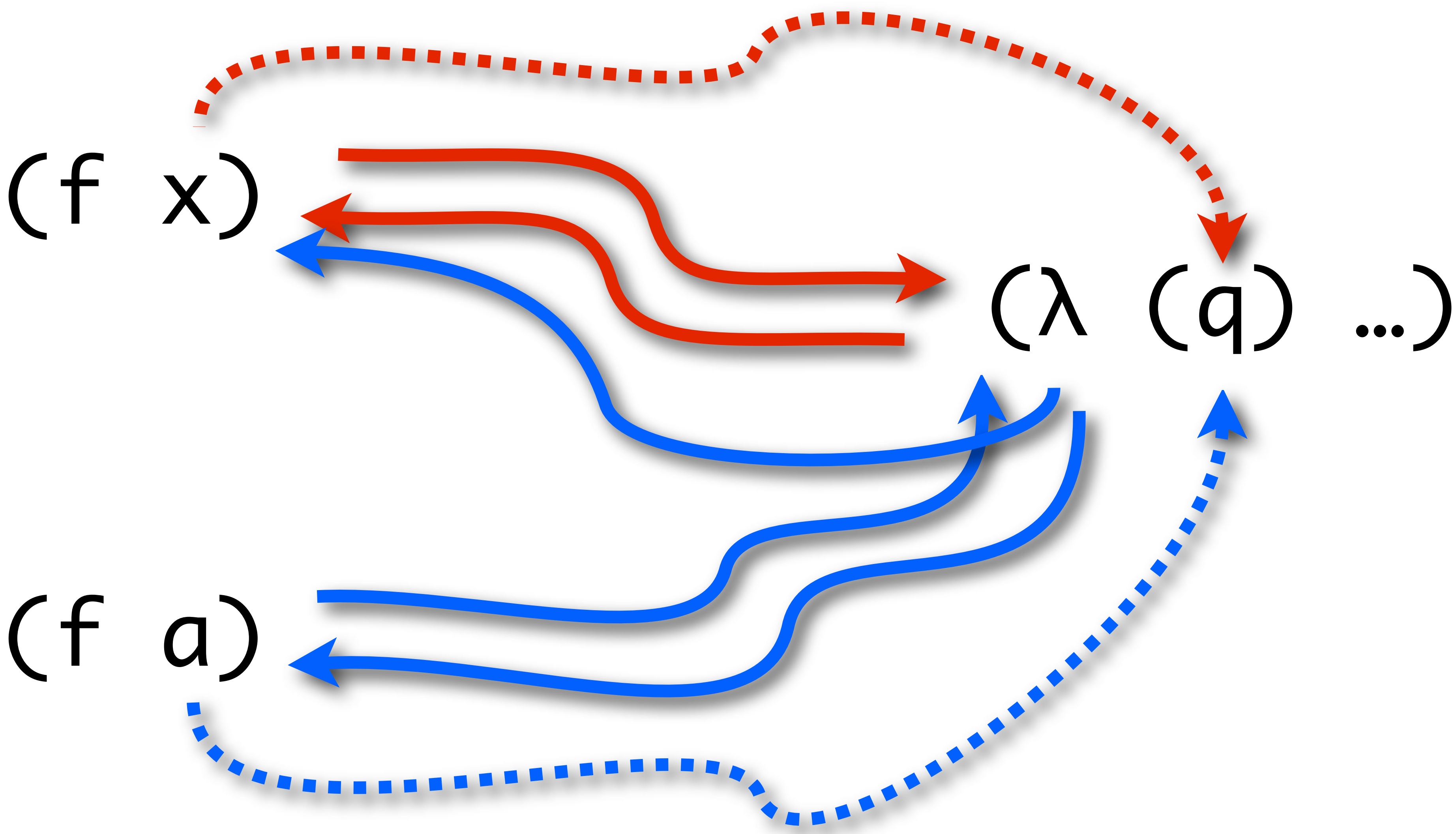
0CFA *v.* **1CFA**



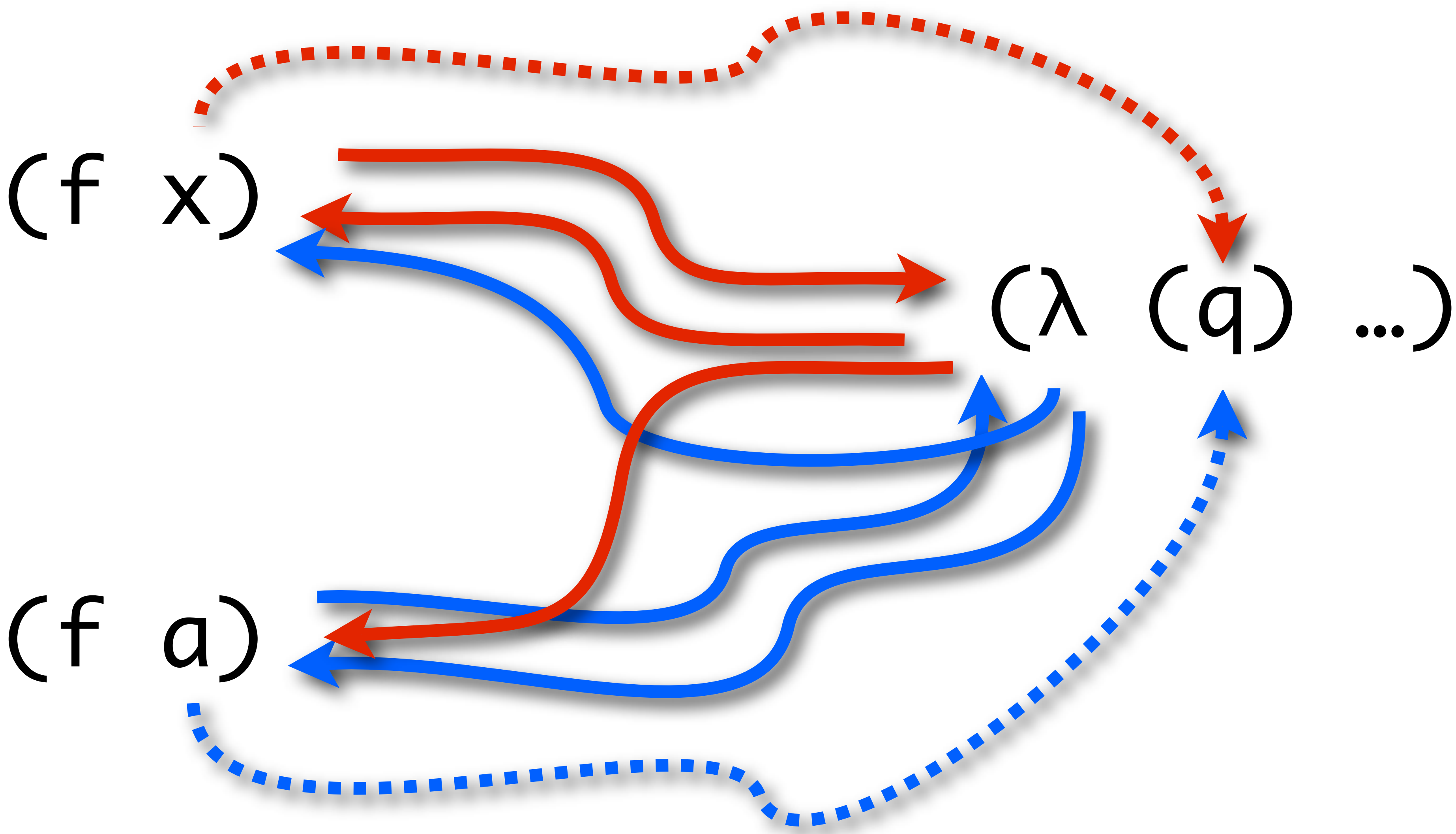
0CFA *v.* 1CFA



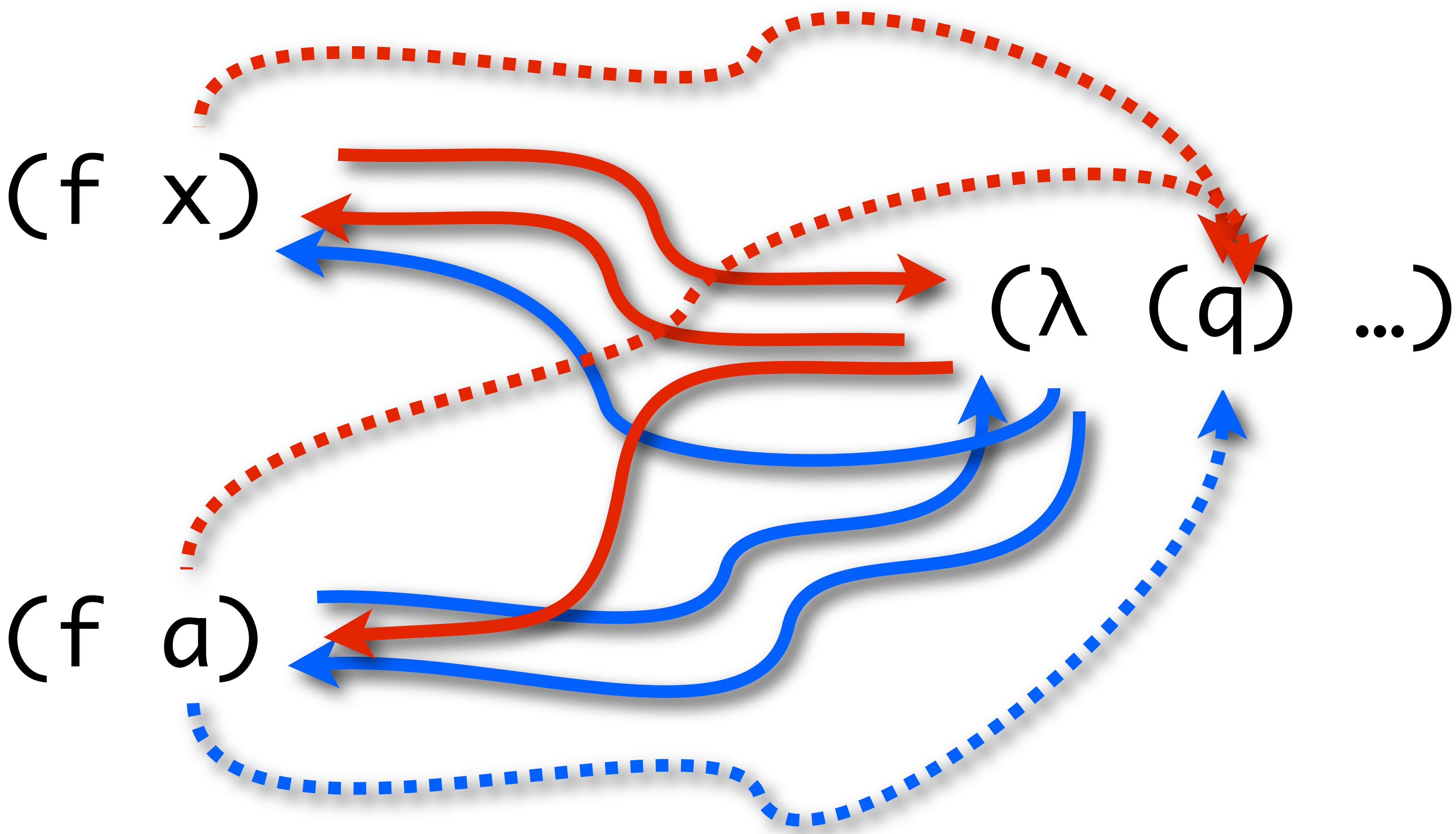
0CFA *v.* 1CFA



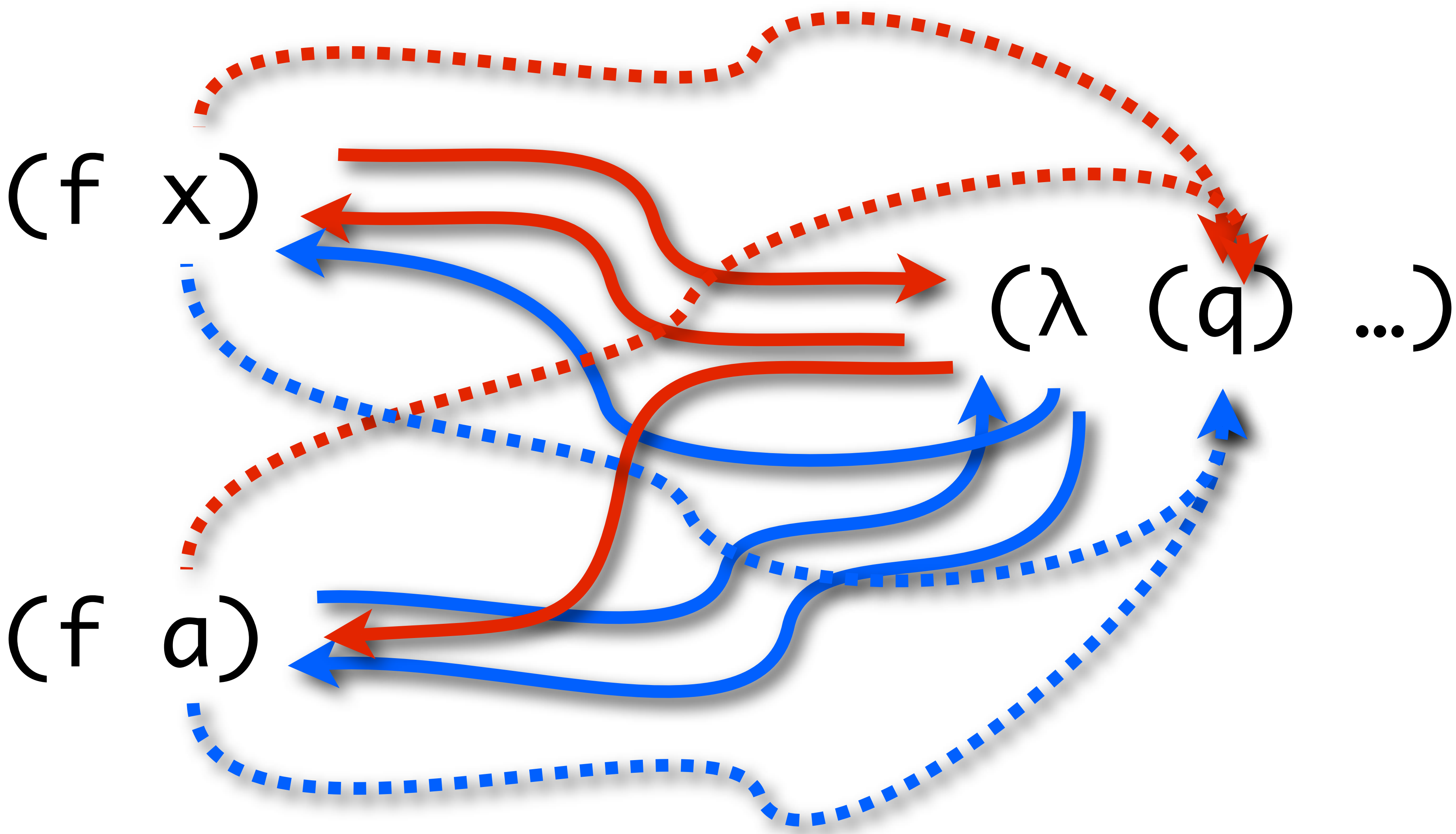
0CFA *v.* 1CFA



0CFA *v.* 1CFA



0CFA *v.* 1CFA



Small-step *k*-CFA for ANF

Why A-Normal Form?

- CPS is global transform; ANF is local
- Simpler than ordinary direct-style
- The essence of run-time stack handling

ANF: A-Normal Form

$$
\begin{aligned}
v &\in \textsf{Var} \\
f, e &\in \textsf{Exp} = \textsf{Var} + \textsf{Lam} \\
lam &\in \textsf{Lam} ::= (\lambda\ (v_1 \ldots v_n)\ body) \\
body &\in \textsf{Body} ::= (f\ e_1 \ldots e_n) \\
&\quad \mid\ (\texttt{let}\ ((v\ body_0))\ body_1) \\
&\quad \mid\ e
\end{aligned}
$$

CESK-like machine

$$
\begin{aligned}
\varsigma \in \Sigma &= \text{Body} \times \mathit{BEnv} \times \mathit{Store} \times \mathit{Kont} \times \mathit{Time} \\
\beta \in \mathit{BEnv} &= \text{Var} \rightharpoonup \mathit{Addr} \\
\sigma \in \mathit{Store} &= \mathit{Addr} \to D \\
d \in D &= \mathit{Val} \\
\mathit{val} \in \mathit{Val} &= \mathit{Clo} \\
\mathit{clo} \in \mathit{Clo} &= \text{Lam} \times \mathit{BEnv} \\
\kappa \in \mathit{Kont} &= \text{Var} \times \text{Body} \times \mathit{Env} \times \mathit{Kont} \\
&\quad + \{\mathbf{halt}\} \\
a \in \mathit{Addr} &\text{ is a set of addresses} \\
t \in \mathit{Time} &\text{ is a set of time-stamps}
\end{aligned}
$$

CESK-like machine

$$
\begin{aligned}
\varsigma \in \Sigma &= \text{Body} \times \mathit{BEnv} \times \mathit{Store} \times \mathit{Kont} \times \mathit{Time} \\
\beta \in \mathit{BEnv} &= \text{Var} \rightharpoonup \mathit{Addr} \\
\sigma \in \mathit{Store} &= \mathit{Addr} \to D \\
d \in D &= \mathit{Val} \\
\mathit{val} \in \mathit{Val} &= \mathit{Clo} \\
\mathit{clo} \in \mathit{Clo} &= \text{Lam} \times \mathit{BEnv} \\
\kappa \in \mathit{Kont} &= \text{Var} \times \text{Body} \times \mathit{Env} \times \mathit{Kont} \\
&+ \{\mathbf{halt}\} \\
a \in \mathit{Addr} & \text{ is a set of addresses} \\
t \in \mathit{Time} & \text{ is a set of time-stamps}
\end{aligned}
$$

But, there's a problem.

Concrete state-space

$$\varsigma \in \Sigma = \mathsf{Body} \times \mathit{BEnv} \times \mathit{Store} \times \mathit{KontPtr} \times \mathit{Time}$$

$$\beta \in \mathit{BEnv} = \mathsf{Var} \rightharpoonup \mathit{Addr}$$

$$\sigma \in \mathit{Store} = \mathit{Addr} \to D$$

$$d \in D = \mathit{Val}$$

$$\mathit{val} \in \mathit{Val} = \mathit{Clo} + \mathit{Kont}$$

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

$$\kappa \in \mathit{Kont} = \mathsf{Var} \times \mathsf{Body} \times \mathit{Env} \times \mathit{KontPtr}$$

$a \in \mathit{Addr}$ is a set of addresses

$t \in \mathit{Time}$ is a set of time-stamps

$$\overset{\kappa}{p} \in \mathit{KontPtr} \subseteq \mathit{Addr}$$

Concrete state-space

$$\varsigma \in \Sigma = \mathsf{Body} \times \mathit{BEnv} \times \mathit{Store} \times \mathit{KontPtr} \times \mathit{Time}$$

$$\beta \in \mathit{BEnv} = \mathsf{Var} \rightharpoonup \mathit{Addr}$$

$$\sigma \in \mathit{Store} = \mathit{Addr} \to D$$

$$d \in D = \mathit{Val}$$

$$\mathit{val} \in \mathit{Val} = \mathit{Clo} + \mathit{Kont}$$

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

$$\kappa \in \mathit{Kont} = \mathsf{Var} \times \mathsf{Body} \times \mathit{Env} \times \mathit{KontPtr}$$

$a \in \mathit{Addr}$ is a set of addresses

$t \in \mathit{Time}$ is a set of time-stamps

$$\overset{\kappa}{p} \in \mathit{KontPtr} \subseteq \mathit{Addr}$$

Concrete semantics

When $body = [\![(f\ e_1 \ldots e_n)]\!]$:

$$
\begin{aligned}
(body, \beta, \sigma, \overset{\kappa}{p}, t) &\Rightarrow (body', \beta'', \sigma', \overset{\kappa}{p}, t'), \text{ where} \\
(lam, \beta') &= \mathcal{E}(f, \beta, \sigma) \\
clo_i &= \mathcal{E}(e_i, \beta, \sigma) \\
lam &= [\![(\lambda\ (v_1 \ldots v_n)\ body')]\!] \\
t' &= tick(body, t) \\
a_i &= alloc(v_i, t') \\
\beta'' &= \beta'[v_i \mapsto a_i] \\
\sigma' &= \sigma[a_i \mapsto clo_i]
\end{aligned}
$$

Concrete semantics

When *body* $= e$:

$$
\begin{aligned}
(\textit{body}, \beta, \sigma, \overset{\kappa}{p}, t) &\Rightarrow (\textit{body}', \beta'', \sigma', \overset{\kappa}{p}', t'), \text{ where} \\
(v, \textit{body}', \beta', \overset{\kappa}{p}') &= \sigma(\overset{\kappa}{p}) \\
t' &= \textit{tick}(\textit{body}, t) \\
a &= \textit{alloc}(v, t') \\
d &= \square(e, \beta, \sigma) \\
\beta'' &= \beta'[v \mapsto a] \\
\sigma' &= \sigma[a \mapsto d]
\end{aligned}
$$

Concrete semantics

When body = $[\![$`(let ((`v body_0`))` body_1`)`$]\!]$:

$$
\begin{aligned}
(\mathit{body}, \beta, \sigma, \overset{\kappa}{p}, t) \Rightarrow (\mathit{body}_0, \beta, \sigma', \overset{\kappa}{p}', t'), \text{ where} \\
\kappa &= (v, \mathit{body}_1, \beta, \overset{\kappa}{p}) \\
t' &= \mathit{tick}(\mathit{body}, t) \\
\overset{\kappa}{p}' &= \mathit{alloc}_\kappa(\mathit{body}_0, t') \\
\sigma' &= \sigma[\overset{\kappa}{p}' \mapsto \kappa]
\end{aligned}
$$

Abstract state-space

$$\hat{\varsigma} \in \hat{\Sigma} = \mathsf{Body} \times \widehat{BEnv} \times \widehat{Store} \times \widehat{KontPtr} \times \widehat{Time}$$

$$\hat{\beta} \in \widehat{BEnv} = \mathsf{Var} \rightharpoonup \widehat{Addr}$$

$$\hat{\sigma} \in \widehat{Store} = \widehat{Addr} \to \hat{D}$$

$$\hat{d} \in \hat{D} = \mathcal{P}(\widehat{Val})$$

$$\widehat{val} \in \widehat{Val} = \widehat{Clo} + \widehat{Kont}$$

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

$$\hat{\kappa} \in \widehat{Kont} = \mathsf{Var} \times \mathsf{Body} \times \widehat{Env} \times \widehat{KontPtr}$$

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

$\hat{t} \in \widehat{Time}$ is a **finite** set of time-stamps

$$\overset{\hat{\kappa}}{p} \in \widehat{KontPtr} \subseteq \widehat{Addr}$$

k-CFA for ANF

When $body = [\![(f\ e_1 \dots e_n)]\!]$:

$$
\begin{aligned}
(body, \hat{\beta}, \hat{\sigma}, \hat{p}^{\hat{\kappa}}, \hat{t}) \rightsquigarrow (body', \hat{\beta}'', \hat{\sigma}', \hat{p}^{\hat{\kappa}}, \hat{t}'), \text{ where} \\
(lam, \hat{\beta}') &\in \hat{\mathcal{E}}(f, \hat{\beta}, \hat{\sigma}) \\
\hat{d}_i &= \hat{\mathcal{E}}(e_i, \hat{\beta}, \hat{\sigma}) \\
lam &= [\![(\lambda\ (v_1 \dots v_n)\ body')]\!] \\
\hat{t}' &= \widehat{tick}(body, \hat{t}) \\
\hat{a}_i &= \widehat{alloc}(v_i, \hat{t}') \\
\hat{\beta}'' &= \hat{\beta}'[v_i \mapsto \hat{a}_i] \\
\hat{\sigma}' &= \hat{\sigma} \sqcup [\hat{a}_i \mapsto \hat{d}_i]
\end{aligned}
$$

k-CFA for ANF

When $body = e$:

$$(body, \hat{\beta}, \hat{\sigma}, \overset{\hat{\kappa}}{p}, \hat{t}) \leadsto (body', \hat{\beta}'', \hat{\sigma}', \overset{\hat{\kappa}}{p}', \hat{t}')\text{, where}$$

$$(v, body', \hat{\beta}', \overset{\hat{\kappa}}{p}') \in \hat{\sigma}(\overset{\hat{\kappa}}{p})$$

$$\hat{t}' = \widehat{tick}(body, \hat{t})$$

$$\hat{a} = \widehat{alloc}(v, \hat{t}')$$

$$\hat{d} = \hat{\square}(e, \hat{\beta}, \hat{\sigma})$$

$$\hat{\beta}'' = \hat{\beta}'[v \mapsto \hat{a}]$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a} \mapsto \hat{d}]$$

k-CFA for ANF

When $body = [\![\texttt{(let ((}v\ body_{\texttt{0}}\texttt{))}\ body_{\texttt{1}}\texttt{)}]\!]$:

$$(body, \hat{\beta}, \hat{\sigma}, \hat{p}^{\hat{\kappa}}, \hat{t}) \leadsto (body_{\texttt{0}}, \hat{\beta}, \hat{\sigma}', \hat{p}^{\hat{\kappa}\prime}, \hat{t}'), \text{ where}$$

$$\hat{\kappa} = (v, body_{\texttt{1}}, \hat{\beta}, \hat{p}^{\hat{\kappa}})$$

$$\hat{t}' = \widehat{tick}(body, \hat{t})$$

$$\hat{p}^{\hat{\kappa}\prime} = \widehat{alloc}_{\kappa}(body_{\texttt{0}}, \hat{t}')$$

$$\hat{\beta}'' = \hat{\beta}'[v \mapsto \hat{a}]$$

$$\hat{\sigma}' = \hat{\sigma} \sqcup [\hat{p}^{\hat{\kappa}\prime} \mapsto \{\hat{\kappa}\}]$$

Questions?