semantics, binding environments map variables to bindings. A binding b is a commemorative token minted for each instance of a variable receiving a value; for example, in k-CFA, a binding is a variable name paired with the time-stamp at which it was bound. The value environment ve tracks the denotable values (D) associated with every binding. A denotable value d is a closure.

In CFAs, bindings—the atomic components of environments—play the role that addresses do in pointer analysis. Our ultimate goal is to infer relationships between the concrete values behind abstract bindings. For example, we want to be able to show that bindings to the variable `v` at some set of times are equal, under the value environment, to the bindings to the variable `x` at some other set of times. (In the pure λ-calculus, the only obvious relationships between bindings are equality and inequality.)

In CFA theory, time-stamps also go by the less-intuitive name of *contours*. Both the concrete and the abstract state-spaces leave the exact structure of time-stamps and bindings undefined. The choices for bindings determine the polyvariance of the analysis. Time-stamps encode the history of execution in some fashion, so that under abstraction, their structure determines the context in context-sensitivity.

The concrete and abstract state-spaces are linked by a parameterized second-order abstraction map, $\alpha^\eta : \Sigma \to \hat{\Sigma}$, where the parameter $\eta : (Addr \to \widehat{Addr}) \cup (Time \to \widehat{Time})$ abstracts both bindings and times:

$$\begin{aligned}
\alpha^\eta(call, \beta, ve, t) &= (\alpha^\eta(V), \alpha^\eta(\beta), \alpha^\eta(ve), \eta(t)) \\
\alpha^\eta_{BEnv}(\beta) &= \lambda v.\eta(\beta(v)) \\
\alpha^\eta_{VEnv}(ve) &= \lambda \hat{b}. \bigsqcup_{\eta(b)=\hat{b}} \alpha^\eta(ve(b)) \\
\alpha^\eta_{D}(d) &= \{\alpha^\eta_{Val}(d)\} \\
\alpha^\eta_{Val}(lam, \beta) &= (lam, \alpha^\eta(\beta)).
\end{aligned}$$

2.2 Transition rules

With state-spaces defined, we can specify the concrete transition relation for CPS, $(\Rightarrow) \subseteq \Sigma \times \Sigma$; then we can define its corresponding abstraction under the map α^η, $(\rightsquigarrow) \subseteq \hat{\Sigma} \times \hat{\Sigma}$. With the help of an argument-expression evaluator, $\mathcal{E} : \mathsf{Exp} \times BEnv \times VEnv \rightharpoonup D$:

$$\begin{aligned}
\mathcal{E}\ (v, \beta, ve)\ &= ve(\beta(v)) \\
\mathcal{E}\ (lam, \beta, ve)\ &= (lam, \beta),
\end{aligned}$$