1.3 Insight: Environments as data structures; bindings as addresses

Under the hood, environments are *dynamically allocated data structures* that determine the value of a λ-term's free variables, and as a consequence, the meaning of the function represented by a closure. When we adapt and extend the principles of shape analysis (specifically, singleton abstractions [2, 4] and shape predicates [23]) to these environments, we can reason about the meaning of and relationships between higher-order functions. As we adapt, we find that, in a higher-order control-flow analysis, bindings are the proper analog of addresses. More importantly, we will be able to solve the aforementioned problems beyond the reach of traditional CFA.

1.4 Contributions

We define the generalized environment problem. We define higher-order rematerialization as a novel client of the generalized environment problem, and we note that super-β inlining and globalization—both known to be beyond the reach CFA—are also clients of the generalized environment problem. We find the philosophical analog of shape analysis for higher-order programs; specifically, we find that we can view binding environments as data structures, bindings as addresses and value environments as heaps. Under this correspondence, we discover *anodization*, a means for achieving both singleton abstraction and focusing; and we discover *binding invariants* as an analog of shape predicates. We use this analysis to solve the generalized environment problem.

2 Platform: Small-step semantics, concrete and abstract

For our investigation into higher-order shape analysis, our platform is a small-step framework for the multi-argument continuation-passing-style λ-calculus:

$$f, e \in \mathsf{Exp} = \mathsf{Var} + \mathsf{Lam} \qquad v \in \mathsf{Var} ::= id^{\ell}$$

$$\ell \in \mathsf{Lab} \text{ is a set of labels} \qquad lam \in \mathsf{Lam} ::= (\lambda^{\ell}\ (v_1 \dots v_n)\ call)$$

$$call \in \mathsf{Call} ::= (f\ e_1 \dots e_n)^{\ell}.$$

2.1 State-spaces

The concrete state-space (Σ in Figure 1) for the small-step machine has four components: (1) a call site *call*, (2) a binding environment β to determine the bindings of free variables, (3) a value environment *ve* to determine the value of bindings, and (4) a time-stamp t to encode the current context/history.

The abstract state-space ($\hat{\Sigma}$ in Figure 1) parallels the structure of the concrete state-spaces. For these domains, we assume the natural partial orders; for example, $\widehat{ve} \sqcup \widehat{ve}' = \lambda\hat{b}.\widehat{ve}(\hat{b}) \cup \widehat{ve}'(\hat{b})$.

Binding environments (*BEnv*), as a component of both machine states and closures, are the environments to which the environment problem refers. In our