5 Application: Higher-order rematerialization

Now that we have a generalized environment analysis, we can precisely state the condition under which higher-order rematerialization is safe. Might's work on the correctness of super-β inlining formally defined *safe* to mean that the transformed program and the untransformed program maintain a bisimulation in their concrete executions [16].

Theorem 4. *It is safe to rematerialize the expression e' in place of the expression e in the call site call iff for every reachable compound abstract state of the form $((call, \hat{\beta}'', \widehat{ve}, \hat{t}), \equiv)$, it is the case that $\hat{\mathcal{E}}(e', \hat{\beta}'', \widehat{ve}) = (lam', \hat{\beta}')$ and $\hat{\mathcal{E}}(e, \hat{\beta}'', \widehat{ve}) = (lam, \hat{\beta})$ and the relation $\sigma \subseteq \mathsf{Var} \times \mathsf{Var}$ is a substitution that unifies the free variables of lam' with lam and for each $(v', v) \in \sigma$, $\hat{\beta}'(v') \equiv \hat{\beta}(v)$.*

Proof. The proof of bisimulation has a structure identical to that of the proof correctness for super-β inlining in [16].

6 Related work

Clearly, this work draws on the Cousots' abstract interpretation [5, 6]. Binding invariants succeed the Cousots' work as a relational abstraction of higher-order programs [7, 8], with the distinction that binding invariants range over abstract bindings instead of formal parameters. Binding invariants were also inspired by Gulwani *et al.*'s quantified abstract domains [9]; there is an implicit universal quantification ranging over concrete constituents in the definition of the abstraction map $\alpha^{\eta}_{\equiv}$. This work also falls within and retains the advantages of Schmidt's small-step abstract interpretive framework [24]. As a generalization of control-flow analysis, the platform of Section 2 is a small-step reformulation of Shivers's denotational CFA [27], which itself was a extension of Jones's original CFA [13]. Like the Nielsons' unifying work on CFA [22], this work is an implicit argument in favor of the inherent flexibility of abstract interpretation for the static analysis of higher-order programs. In contrast with constraint-based, type-based and model-checking CFAs, small-step abstract interpretive CFAs are easy to extend via direct products and parameterization.

From shape analysis, anodized bindings draw on singleton abstraction while binding invariants are inspired by both predicate-based abstractions [3] and three-valued logic analysis [23]. Chase *et. al* had early work on counting-based singleton abstractions [4], while Hudak's work on analysis of first-order functional programs employed a precursor to counting-based singleton abstraction [10]. Anodization, using factored sets of singleton and non-singleton bindings, is most closely related to the Balakrishnan and Reps's recency abstraction [2], except that anodization works on bindings instead of addresses, and anodization is not restricted to a most-recent allocation policy. Superficially, one might also term Jones and Bohr's work on termination analysis of the untyped λ-calculus via size-change as another kind of shape analysis for higher-order programs [14].