The paper [1] formalizes the dual notions of subject and context blame soundness for languages with higher-order contracts. It also formalizes the contract analogues of union and intersection types and proves two sound monitoring strategies (assuming pure, terminating contracts). The authors explain monitoring for unions & intersections in terms of a constraint graph data structure.

Strengths

- The context soundness theorem is a wonderful contribution. I think if I have some free time this winter I should try to reproduce it for more of the related work.

  Also I appreciated their emphasis that contracts are meant to detect violations, not guarantee correctness. Really, the whole formalism is very well done.

- The examples were few, and always insightful.

Weaknesses

- Why mention TreatJS without a brief evaluation or link to the ECOOP paper?

- The introductory sections are pretty technical. If I wasn't already familiar with contracts from Racket & the literature I would have a hard time understanding.

- The paper was mostly very good about citing related work—except for effects! Their theorems clearly do not hold for stateful or non-terminating contracts because of the backtracking needed to check intersection & union contracts. The authors should have mentioned this, and I would have liked to hear their ideas.

  Their ECOOP paper on TreatJS [2] uses sandboxing to limit some effects of contracts. That's good, but what happens if a value satisfies the right clause of a disjunction but the contract system checks the left clause first and in doing so enters an infinite loop?

  Sandboxes also don't address issues with state internal to contracts. Suppose we had the affine contract $((A \to^1 B) \cap (A' \to^1 B))$ and a program that exercised both alternatives exactly once. If I understand the authors' contract scheme, the first function application will trigger both affine flags. Then the second call will fail.

References

[1] Matthias Keil and Peter Thiemann. Blame assignment for higher-order contracts with intersection and union. In *ICFP*, 2015.

[2] Matthias Keil and Peter Thiemann. Treatjs: Higher-order contracts for javascript. In *ECOOP*, 2015.