Affine type systems are useful, but unconventional. The paper [2] demonstrates the usefulness of affine types by encoding a socket API. It then shows how to integrate an affine type system with a conventional (System F) type system by giving a multi-language semantics based on the lump embedding [1]. The key innovation is using a 1-bit reference to implement affine contracts.

The paper's main technical result is that well-typed affine programs which reduce to a value will preserve their type. These affine programs may contain subterms type-checked by System F. The paper also claims that all double-uses of affine values at runtime are due to System F code and are detected by the contract system.

Strengths

- The multi-language system is a thorough argument that affine types could be added to any strongly typed language without affecting current and future type rules.
- This paper is a very pretty composition of prior work on contracts, multi-language semantics, and substructural types. It was nice to see the papers we've read over the past few weeks come together.

Weaknesses

- Jesse's website has a few implementations of this work—a Haskell prototype, redex model, and Racket package. If these existed when the paper was published, then they should have been mentioned in a brief "evaluation" section. What I would really like to know is whether the implementation compiles and runs efficiently with the inter-language translations. Even though most translations were the identity or opaque, I imagine that repeatedly walking terms to do these "simple" conversions could get very expensive.
- Okay, as cool as the multi-language system is, I didn't think it was relevant to the main point of "stateful contracts". Instead of 10 pages on a correct multi-language system, I would have preferred to see more extensions and examples of contracts with internal state. Are there any challenges keeping state other than a 1-bit monotonic reference? (I don't think there would be.) Can we see any more examples of system protocols, or maybe correct mutex semantics? Could this work replace the cited uses of session types or typestate? I think any of these would have been a better and more motivating use of the space.

References

[1] Jacob Matthews and Robert Bruce Findler. Operational semantics for multi-language programs. In *POPL*, 2007.

[2] Jesse A. Tov and Ricardo Pucella. Stateful contracts for affine types. In *ESOP*, 2010.