The paper defines *manifest contracts* as contracts whose predicates are reflected in the refinement types of variables during typechecking, and *latent contracts* as contracts that play no role in typechecking [2]. For languages with higher-order and dependent contracts, we learn how to translate between manifest and latent contract system and what blame guarantees (if any) are lost in translations. Translation losses may happen only in the dependent case, and only when translating from a *lax* contract system to a dependent or *picky* one.

Strengths

- Clarifies a lot of related work and vocabulary.
- Lax contracts seem like a bug to me, and the authors apparently think that picky contracts are "arguably more correct", but I'm glad the paper gave lax and picky contracts equal attention.
- The correspondence theorems attribute blame, rather than raising generic errors.

Weaknesses

- The main result is not very exciting. I'm not sure when I would want to use these translations, except to settle an argument. Also, polymorphism and recursion are left unformalized.

HOPE notes [1]

- Maybe the sandboxing from TreatJS [3] and Matthew Flatt's scope sets [**?**] could help manage stateful contracts. Sandboxing would make contract checks idempotent (so we don't need to worry that lax vs. picky semantics drastically change our observations) and scope sets might be the answer to the "unbound" variable mentioned in Section 2.1. (I badly need to read the scope-sets paper.)
- I don't quite get the issue with circularity. Is there anything new here that hasn't come up dealing with recursive types or state in logical relations? Seems like "stateful manifest contracts in Hoare Type Theory" would be a great place to start this work.
- This paper seems to assume that all manifest contracts affect the typechecker. If so, I disagree. I think it would be fine for the type checker to prove as much as it could statically, but then "give up" and leave nastier predicates outside the type system. This would only affect the number of

dynamic checks, and a real implementation could issue warnings about why the contract is too complicated to typecheck.

- Why does he reference Racket 6.1 in 2015?
- The slides were much more fun to read than the abstract.

References

[1] Michael Greenberg. Combining manifest contracts with state. In *HOPE*, 2015.

[2] Michael Greenberg, Benjamin C. Pierce, and Stephanie Weirich. Contracts made manifest. In *POPL*, 2010.

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