This paper re-evaluates the meaning of dependent contracts and proposes *complete monitoring* as a notion of correctness [1]. A complete monitor guarantees that evaluation never gets stuck and blame errors are never ambiguous (blame always falls on precisely one party). The paper argues by example that complete monitors are useful for tracking mutable variables and implementing gradual typing.

Strengths

- I've been convinced that complete monitors are always better than picky and sometimes better than lax. (Before I started reading, I preferred lax because (1) my contract is within my trust boundary and (2) getting to the codomain contract should always imply the domain contract held. This was a good argument that I shouldn't need to trust my contracts.)
- The main result is clear and well-motivated, though I'm not sure if single-ownership is brilliant & obvious or missing some subtle use-cases.[1]

Weaknesses

- *indy*. It is a very boring name, and doesn't immediately communicate the (intended?) mnemonic: contracts are independent parties. I think *careful* or *paranoid* would make a better companion to lax and picky.
- I can't believe they talk about references without bringing up stateful contracts. This must be why Racket keeps both lax & indy contracts—there are some cases where invoking the contract twice changes the program's semantics.
- Complete monitoring is a very nice statement, but if the core goal is to make debugging easy, I think the best option is to just print a stack trace. Having one faulty party sounds nice, but may not help diagnose the core issue (see footnote 1). Knowing the flow of values that led to the fault is definitely better.

An aside: the idea of spot-checking function arguments reminds me of unit tests. I know they're not the same, because these values appear at runtime, but I'd like to hear the author's opinion on how contracts and tests fit together, just as I've heard how contracts and types fit together.

[1] Maybe I'm misunderstanding, but it seems like when a new party claims ownership of a value, it hides the value's original source.

References

[1] Christos Dimoulas, Sam Tobin-Hochstadt, and Matthias Felleisen. Complete monitors for behavioral contracts. In *ESOP*, 2012.