Mind the Gap:
Artifacts vs Insights in PL Theory

Nada Amin[*] and Tiark Rompf [*‡]

[*]EPFL: {first.last}@epfl.ch
[‡]Oracle Labs: {first.last}@oracle.com

A common template for papers at PL conferences is to present a calculus and prove some properties such as soundness. How can we assess the value of such papers? In this talk, we suggest that the most valuable formalization efforts are those which explore the design space. Soundness results rarely compose (think let-polymorphism and mutable references), yet real systems need to combine features and make compromises. Hence, we need to mind the gap: give implementors of practical systems not just core calculi – polished but of questionable applicability – but real insights into the trade-offs of the landscape. A good example is the work of Olivier Danvy and his collaborators on inter-derivation of semantic artifacts, which exposes deep connections between a large class of individually proposed evaluators and abstract machines.

In more general terms, we argue that we should have the same standard for theoretical and practical artifacts. While PL researchers are very well aware that only limited conclusions can be drawn from individual implemented systems, it is less widely appreciated that the same holds for individual calculi. Instead of focusing on the artifacts themselves, we should ask: What insights did we gain from building them?

We draw from our experience formalizing Dependent Object Types (DOT) – a core calculus for path-dependent types, which play an essential role in unifying object and module systems in a language like Scala. At first, we worked top-down from a polished design, minding the gap to soundness: When we would get stuck in the proofs, we would look for counterexamples, patch the calculus and reiterate. Eventually, we switched to a bottom-up exploration, minding the gap to expressivity: At each iteration, we would design a calculus by small tweaks to the previous ones, and mechanically prove it sound. Through this process, we have gained key insights into which combinations of features of our original calculus pose challenges and identified key trade-off between desirable traits such as nominality or subtyping transitivity.

In this talk, we argue that papers should expose the sausage factory of designing calculi, and the mine fields in the landscape. We advocate that formalization efforts should learn from tried and true systems engineering practices: design top down, implement and test bottom-up, and we argue for tools, such as Twelf, which enable both mechanizing the metatheory and assessing the expressivity, using the same model – not only for convenience, but especially for confidence.

POPL OBT 2014 – Off the Beaten Track