Type refinement as a unifying principle

Noam Zeilberger

Ecole Polytechnique

Frank Pfenning special session
LFMTP @ FLOC 2022
Haifa (from Paris)
1 August 2022

Origins of type refinement

Research program started by Frank in the late 1980s.

According to him (circa 2015), « the goals have drifted somewhat over time, but can be summarized as:

(a) capture more precise properties of programs that we already know to be well-typed in a simpler discipline in order to catch more errors;

(b) retain the good theoretical and practical properties of the simpler disciplines such as effective type checking; and

(c) all the other good things like usability, modularity, elegance, etc., just as for any system of types. »

Frank's role in developing the program

Co-authored / authored two germinal papers:

- Tim Freeman and F.Pf., *Refinement Types for ML*, PLDI 1991
- Pf., *Refinement Types for Logical Frameworks*, Workshop on Types for Proofs and Programs, May 1993

Supervised / co-supervised at least six PhD theses on the topic.

Co-author / author of many (≥ 11) additional papers on or around type refinement through the 1990s-2020s, including several that had a deep impact on my research...

Refinement Types for ML (1991)

Excerpt:

> *[...] In this paper we summarize the design of a system of subtypes for ML which preserves the desirable properties listed above, while at the same time providing for specification and inference of significantly more precise type information. We call the resulting types* ***refinement types****, as they can be thought of as refining user-defined data types of ML. [...] Thus, using refinement types, the programmer is encouraged to make explicit the distinctions which currently must remain implicit or informal in code comments.*

Refinement Types for Logical Frameworks (1993)

Excerpt:

> *Over the past two years we have carried out extensive experiments in the application of the LF Logical Framework to represent and implement deductive systems and their metatheory. [...] Despite its expressive power, certain weaknesses of LF emerged during these experiments. One of these is the absence of any direct form of subtyping. Clearly, this is not a theoretical problem: what is informally presented as subtyping can be encoded either via explicit coercions or via auxiliary judgments as we will illustrate below.* ***In practice, however, this becomes a significant burden, and encodings are further removed from informal mathematical practice than desirable.***

Type refinement and computational effects (2000s)

ICFP'00 paper with Rowan Davies showed that a *value restriction* is needed on $\wedge$-introduction, and that the classic subtyping law $(A \to B) \wedge (A \to C) \leq A \to (B \wedge C)$ is unsound for CBV in the presence of effects. (That paper also introduced an elegant *bidirectional* type checking algorithm for verifying refinements.)

POPL'04 paper with Jana Dunfield observed that a dual restriction is needed on $\vee$-elimination, and implicitly that the subtyping law $(A \to C) \wedge (B \to C) \leq (A \vee B) \to C$ is unsound for effectful CBN.

Both of these papers were a big influence on my thesis research on polarities and focusing. I found it fascinating how type refinements could be used to observe semantic properties like evaluation order. In a sense, type refinement systems unify syntax with semantics!

From deductive systems to type refinement systems

When I started collaborating with Paul-André Melliès (ca. 2010), one of our goals was to better understand the categorical semantics of focusing proofs. An idea was to think of a formula as denoting a presheaf of proofs of that formula in a given context, which is only slightly more sophisticated than thinking of a formula as denoting the subset of contexts in which it is true.

After many twists and turns, we had a leap of understanding while considering the seemingly unrelated question: what is a type refinement system, categorically? Although such systems admit many different *models* (e.g., where refinements denote subsets of values), in a more basic sense a type refinement system *is* just a functor from a category of derivations to a category of terms.

Functors as type refinement systems

This insight, inspired by Frank et al's work, led us to consider general functors $p : \mathbb{D} \to \mathbb{T}$ as type refinement systems, and to try to analyze different deductive systems by starting from this most general setup before imposing any other assumptions upon p.

For example, in our POPL 2015 paper, we discussed how Hoare Logic may be naturally considered as a type refinement system, where predicates refine the type of the global state W. The "rule of composition" reduces to functoriality of p, and the "rule of consequence" to functoriality combined with the category axioms.

The separating conjunction and implication of Separation Logic can then be nicely modelled by imposing the additional assumption that W has the structure of a monoid in $\mathbb{T}$ and that p is a monoidal closed bifibration.

From typing to parsing

In recent work, Paul-André and I explained how this perspective may be applied to better understand derivability in context-free grammars. Intuitively, every non-terminal of an ordinary CFG may be seen as refining the type of strings, and it turns out to be natural and useful to define a more general notion of *CFG over a category*, where each non-terminal refines some type of arrows $A \to B$. For example, we use this to construct the intersection of a context-free language with a regular language, by building an intermediate CFG that generates a language of runs of an NDFA.

See: PAM & NZ, *Parsing as a lifting problem and the Chomsky-Schützenberger Representation Theorem*, MFPS 2022

Type refinement as a unifying principle

The functorial view of type refinement seems to have a lot of explanatory power for analyzing different kinds of deductive, computational, and linguistic models:

- λ-calculi with Curry-style typing and subtyping
- Hoare logic and separation logic
- Sequent calculi for substructural logics
- Finite-state automata
- Context-free grammars
- ...

Much work to be done, and still on the quest for a unified theory!

Lesson learned from Frank: mathematical elegance cannot be separated from practical considerations, and they often go hand-in-hand!