Type in Type and Schematic Affine Recursion

Aaron Stump and Victor Taelin

August 17, 2025

1 Terminating Computation First

Constructive type theories based on the Curry-Howard isomorphism enforce logical soundness by ensuring that all programs are uniformly terminating. Proofs are identified with programs, and diverging programs would constitute proofs of any proposition. So these must be ruled out statically. The approach adopted in systems like Coq, Agda, and Lean is to enforce termination through a combination of typing and syntactic checks for structural decrease at recursive calls.

This paper proposes an alternative, where termination is enforced through syntactic checks alone, prior to typing. The approach has the drawback that programs must be significantly restricted in order to guarantee termination without any reference to types. There is a notable benefit, however: the type system is now no longer required to enforce termination. This greatly increases the options for typing, which now needs only to satisfy type safety, as required in Programming Languages.

We present a language SAR, which combines an untyped affine lambda calculus with a form of structural recursion. With no further restriction, this language allows diverging terms. So we impose what Alves et al. call the "closed-by-construction" restriction on structural recursion, found also in [1]. This requires that the functions to be iterated when recursing are closed. But this rules out most of the usual higher-order functions like `map` on lists, where the function to iterate calls a function f that is given as a variable bound outside the recursion. We address this problem by proposing a language of schematic terms, where such variables f are not λ-bound, but treated schematically. This allows generic definition of functions like `map`, without losing termination.

With termination of SAR established prior to typing, we are free to adopt a more exotic type system than possible in other constructive type theories, where the burden of termination falls on typing. To demonstrate this, we consider a dependent type system called Typed SAR, with the "*Type* : *Type*" principle.

2 Previous work on termination with affine recursion

3 Schematic affine recursion

4 Typing without termination

References

[1] Ugo Dal Lago. The geometry of linear higher-order recursion. *ACM Trans. Comput. Log.*, 10(2):8:1–8:38, 2009.