The Next 700 Type Systems

Carlo Angiuli

March 31, 2017

Type systems classify programs in a way that enables compositional reasoning about their behavior. As a result, type systems have found a place as one of the major organizing principles of modern programming languages; much programming language research focuses on new ways of classifying programs in order to capture more sophisticated invariants, including dependent, gradual, refinement, linear, intersection, and existential types.

However, I feel that modern type systems focus too narrowly on *classification*, which is but one of the twenty-one definitions of the noun *type* in the Oxford English Dictionary. This myopic view of types has impeded the vast majority of the possible subdisciplines of type theory, as depicted in Figure 1.

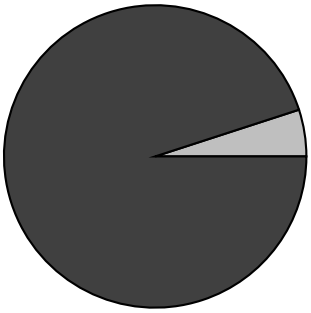

Figure 1: 95.2% of the definitions of *type, n.*, have not been explored in the context of type systems.

In the remainder of this paper, we describe a family of unimplemented type systems that is intended to span differences of meaning by a group of disparate frameworks (Landin, 1966).

Symbol, emblem. This sense of type theory is also known as *symbology*, a lesser-known field whose most famous researcher is Robert Langdon (Brown, 2000). Although Langdon has found success focusing his efforts toward the Illuminati and Catholic Church, it appears that other, less dangerous applications of symbology remain underexplored.

A pattern stamped onto the face of a coin. In this sense, type systems are methods of stamping patterns on coins. The earliest type systems required manually hammering a piece of metal between two dies. Improvements in metal-working and industrial technologies has enabled extensive automation in modern type systems. A recent theoretical advance in type systems occurred during the 2013 United States debt-ceiling crisis, in which some economists suggested that the United States Treasury could mint a $1 trillion platinum coin in order to keep the government afloat without raising the debt ceiling (Matthews, 2013).

Tip. The theory of tips was famously studied by medieval theologians, who sought to compute the number of angels that could simultaneously dance on the head of a pin. (Although this has not been experimentally validated, Aquinas (1274) suggested that two angels cannot be in the same place.) While there are no known applications of tip theory, advances could lead to new transistor technologies. Unfortunately, funding sources are unlikely to surface.

A small block bearing a raised character, for use in printing. Modern type systems were invented by Johannes Gutenberg in 1440, and within decades, made an enormous impact on European society. I suggest that programming language researchers take credit for this early technological breakthrough in type systems.

The sort of person to whom one is attracted (*one's type*). It is well-known that types are (perhaps most) useful as tools for specifying interfaces at abstraction boundaries. Types also, apparently, apply at attraction boundaries; further research is warranted.

Printed characters (*in type*). TEX is the most popular type system among traditional type theorists. When coupled with its large ecosystem of packages, TEX's type system is both expressive and aesthetically-pleasing, contrary to the popular belief that it has no types, and in fact lacks any facilities for abstraction.

An imperial edict released by Emperor Constans II in AD 648 prohibiting discussion of monothelitism. The Type of Constans was a ban on the debate between monothelitism and dyothelitism—whether Jesus Christ, having both divine and human nature, possessed a single will, or two wills (divine and human).

In fact, traditional type systems are already ideal for restricting inquiry into the nature of things; recall, for instance, the fable of Professors Descartes and Bessel in Reynolds (1983). To a programming language researcher, it is clear that the Type of Constans is parametrically polymorphic in the will of Christ, and the prohibition of discussion is simply a free theorem (Wadler, 1989).

The specimen originally used to name a species (*type specimen*). This is clearly just a mode of use of singleton types; given a specimen s, its species is the denotation of the type $S(s)$ of all specimens equal to s.

***type, v.* To write with a keyboard.** Although debate rages on over which type system is best—QWERTY, Dvorak, Colemak, et cetera—most evidence remains anecdotal. This debate has not impacted the success of QWERTY and its close relatives (Noyes, 1983), but the field is nevertheless in dire need of rigorous theoretical study. Given the relevance of keyboards to both proving and programming, I suggest submitting research on this subject to the annual TYPES International Conference on Types for Proofs and Programs.

References

[1] Thomas Aquinas. *Summa Theologiæ*. 1274.

[2] Dan Brown. *Angels & Demons*. New York: Simon and Schuster, 2000. ISBN: 978-0-7434-1239-1.

[3] P. J. Landin. "The Next 700 Programming Languages". In: *Commun. ACM* 9.3 (Mar. 1966), pp. 157–166. ISSN: 0001-0782. DOI: 10.1145/365230.365257. URL: http://doi.acm.org/10.1145/365230.365257.

[4] Dylan Matthews. "Michael Castle: Unsuspecting godfather of the $1 trillion coin solution". In: *The Washington Post* (Jan. 2013). URL: https://www.washingtonpost.com/news/wonk/wp/2013/01/04/michael-castle-unsuspecting-godfather-of-the-1-trillion-coin-solution.

[5] Jan Noyes. "The QWERTY keyboard: a review". In: *International Journal of Man-Machine Studies* 18.3 (1983), pp. 265–281. ISSN: 0020-7373. DOI: http://dx.doi.org/10.1016/S0020-7373(83)80010-8. URL: http://www.sciencedirect.com/science/article/pii/S0020737383800108.

[6] John C. Reynolds. "Types, abstraction, and parametric polymorphism". In: *Information Processing* (1983), pp. 513–523.

[7] Philip Wadler. "Theorems for Free!" In: *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture*. FPCA '89. Imperial College, London, United Kingdom: ACM, 1989, pp. 347–359. ISBN: 0-89791-328-0. DOI: 10.1145/99370.99404. URL: http://doi.acm.org/10.1145/99370.99404.