Type theory

Canonical name	TypeTheory
Date of creation	2013-11-12 21:26:45
Last modified on	2013-11-12 21:26:45
Owner	PMBookProject (1000683)
Last modified by	PMBookProject (1000683)
Numerical id	1
Author	PMBookProject (1000683)
Entry type	Feature
Classification	msc 03B15

Type theory was originally invented by Bertrand Russell [?], as a device for blocking the paradoxes in the logical foundations of mathematics that were under investigation at the time. It was later developed as a rigorous formal system in its own right (under the name "λ-calculus") by Alonzo Church [?],[?],[?]. Although it is not generally regarded as the foundation for classical mathematics, set theory being more customary, type theory still has numerous applications, especially in computer science and the theory of programming languages [?]. Per Martin-Löf [?],[?],[?],[?], among others, developed a "predicative" modification of Church's type system, which is now usually called dependent, constructive, intuitionistic, or simply *Martin--Löf type theory*. This is the basis of the system that we consider here; it was originally intended as a rigorous framework for the formalization of constructive mathematics. In what follows, we will often use "type theory" to refer specifically to this system and similar ones, although type theory as a subject is much broader (see [?],[?] for the history of type theory).

In type theory, unlike set theory, objects are classified using a primitive notion of *type*, similar to the data-types used in programming languages. These elaborately structured types can be used to express detailed specifications of the objects classified, giving rise to principles of reasoning about these objects. To take a very simple example, the objects of a product type $A \times B$ are known to be of the form (a, b), and so one automatically knows how to construct them and how to decompose them. Similarly, an object of function type $A \to B$ can be acquired from an object of type B parametrized by objects of type A, and can be evaluated at an argument of type A. This rigidly predictable behavior of all objects (as opposed to set theory's more liberal formation principles, allowing inhomogeneous sets) is one aspect of type theory that has led to its extensive use in verifying the correctness of computer programs. The clear reasoning principles associated with the construction of types also form the basis of modern *computer proof assistants*, which are used for formalizing mathematics and verifying the correctness of formalized proofs. We return to this aspect of type theory below.

One problem in understanding type theory from a mathematical point of view, however, has always been that the basic concept of *type* is unlike that of *set* in ways that have been hard to make precise. We believe that the new idea of regarding types, not as strange sets (perhaps constructed without using classical logic), but as spaces, viewed from the perspective of homotopy theory, is a significant step forward. In particular, it solves the problem of understanding how the notion of equality of elements of a type differs from that of elements of a set.

In homotopy theory one is concerned with spaces and continuous mappings between them, up to homotopy. A *homotopy* between a pair of continuous maps $f : X \to Y$ and $g : X \to Y$ is a continuous map $H : X \times [0,1] \to Y$ satisfying $H(x,0) = f(x)$ and $H(x,1) = g(x)$. The homotopy H may be thought of as a "continuous deformation" of f into g. The spaces X and Y are said to be *homotopy equivalent*, $X \simeq Y$, if there are continuous maps going back and forth, the composites of which are homotopical to the respective identity mappings, i.e., if they are isomorphic "up to homotopy". Homotopy equivalent spaces have the same algebraic invariants (e.g., homology, or the fundamental group), and are said to have the same *homotopy type*.

References

[1] Alonzo Church. A set of postulates for the foundation of logic 2. *Annals of Mathematics*, 34:839–864 1933

[2] Alonzo Church. A formulation of of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68 1940

[3] Alonzo Church, *The Calculi of Lambda Conversation* Princeton University Press,1941

[4] Fairouz Kamareddine and Twan Laan and Rob Nederpelt, *A Modern Perspective on Type Theory: From its Origins until Today* Kluwer,2004

[5] Martin-Löf,Per. An intuitionistic theory of types. In *Twenty-five years of constructive type theory (Venice,1995)*,volume 36 of Oxford Logic Guides, pages 127–172. Oxford University Press, 1998.

[6] Martin-Löf,Per. An intuitionistic theory of types: predicative part. In *Logic Colloquium '73,Proceedings of the Logic Colloquium*,volume 80 of Studies in Logic and the Foundations of Mathematics, pages 73–118. North-Holland, 1975.

[7] Martin-Löf,Per. Constructive mathematics and computer programming. In *Logic,Methodology and Philosophy of Science VI,Proceedings of the Sixth International Congress of Logic,Methodology and Philosophy of Science,Hannover 1979*,volume 104 of Studies in Logic and the Foundations of Mathematics, pages 153–175. North-Holland, 1982.

[8] Martin-Löf,Per, *Intuitionistic type theory* Bibliopolis,1984

[9] Benjamin C. Pierce, *Types and Programming Languages* MIT Press,2002

[10] Bertand Russell. Mathematical logic based on the theory of types. *American Journal of Mathematics*, 30:222–262 1908

[11] Giovanni Sommaruga, *History and Philosophy of Constructive Type Theory* Kluwer,2010