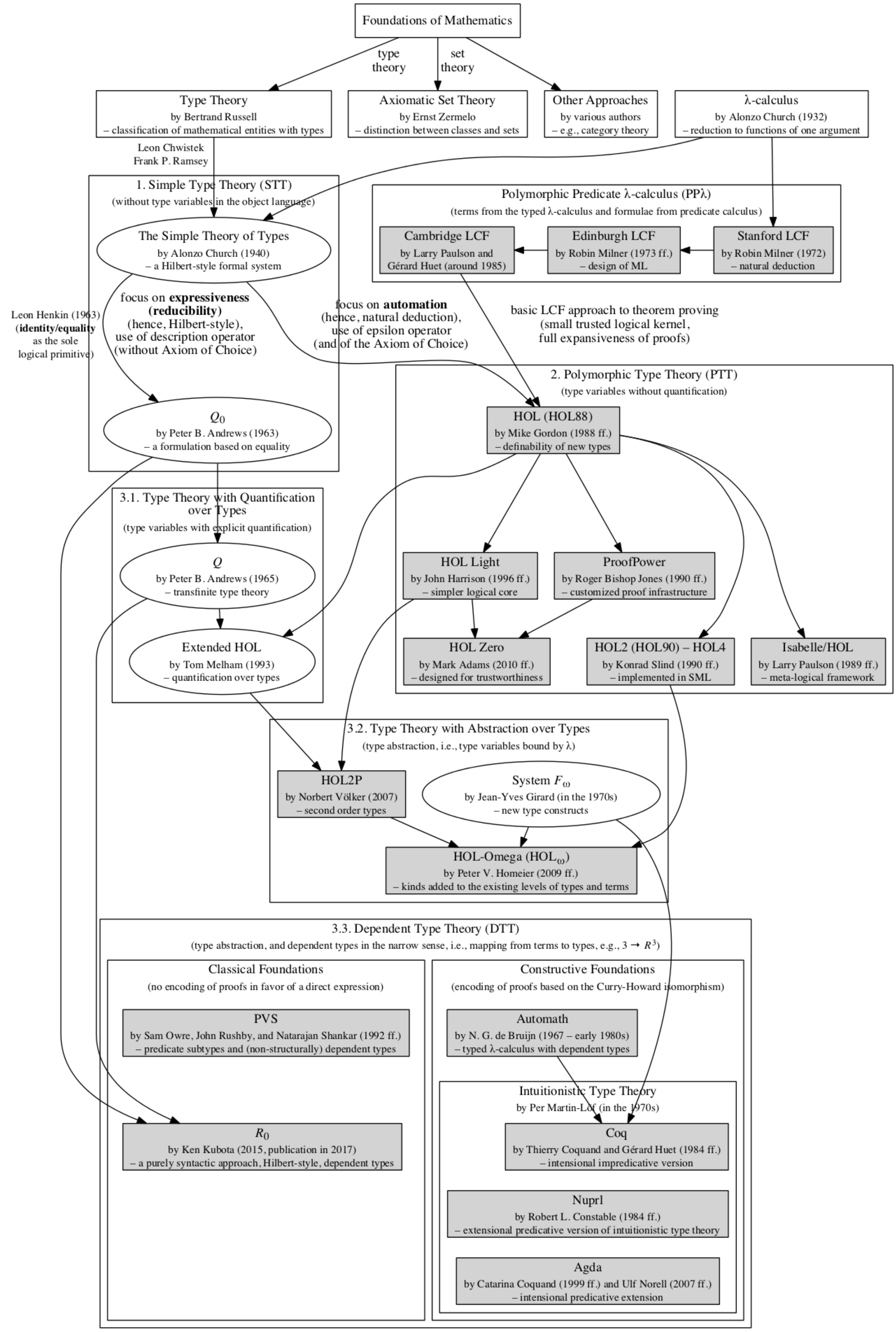
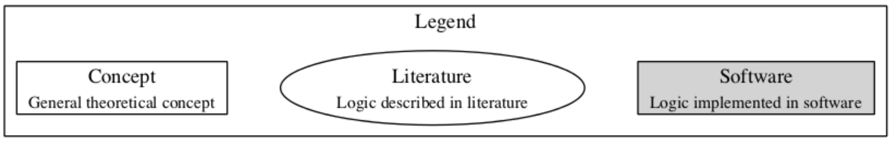

Foundations of Mathematics
type theory
set theory
Type Theory
by Bertrand Russell
– classification of mathematical entities with types
Axiomatic Set Theory
by Ernst Zermelo
– distinction between classes and sets
Other Approaches
by various authors
– e.g., category theory
λ-calculus
by Alonzo Church (1932)
– reduction to functions of one argument
Leon Chwistek
Frank P. Ramsey
1. Simple Type Theory (STT)
(without type variables in the object language)
The Simple Theory of Types
by Alonzo Church (1940)
– a Hilbert-style formal system
Polymorphic Predicate λ-calculus (PPλ)
(terms from the typed λ-calculus and formulae from predicate calculus)
Cambridge LCF
by Larry Paulson and Gérard Huet (around 1985)
Edinburgh LCF
by Robin Milner (1973 ff.)
– design of ML
Stanford LCF
by Robin Milner (1972)
– natural deduction
focus on **expressiveness** **(reducibility)** (hence, Hilbert-style), use of description operator (without Axiom of Choice)
focus on **automation** (hence, natural deduction), use of epsilon operator (and of the Axiom of Choice)
basic LCF approach to theorem proving (small trusted logical kernel, full expansiveness of proofs)
Leon Henkin (1963) (**identity/equality** as the sole logical primitive)
2. Polymorphic Type Theory (PTT)
(type variables without quantification)
Q_0
by Peter B. Andrews (1963)
– a formulation based on equality
HOL (HOL88)
by Mike Gordon (1988 ff.)
– definability of new types
3.1. Type Theory with Quantification over Types
(type variables with explicit quantification)
Q
by Peter B. Andrews (1965)
– transfinite type theory
Extended HOL
by Tom Melham (1993)
– quantification over types
HOL Light
by John Harrison (1996 ff.)
– simpler logical core
ProofPower
by Roger Bishop Jones (1990 ff.)
– customized proof infrastructure
HOL Zero
by Mark Adams (2010 ff.)
– designed for trustworthiness
HOL2 (HOL90) – HOL4
by Konrad Slind (1990 ff.)
– implemented in SML
Isabelle/HOL
by Larry Paulson (1989 ff.)
– meta-logical framework
3.2. Type Theory with Abstraction over Types
(type abstraction, i.e., type variables bound by λ)
HOL2P
by Norbert Völker (2007)
– second order types
System F_ω
by Jean-Yves Girard (in the 1970s)
– new type constructs
HOL-Omega (HOL_ω)
by Peter V. Homeier (2009 ff.)
– kinds added to the existing levels of types and terms
3.3. Dependent Type Theory (DTT)
(type abstraction, and dependent types in the narrow sense, i.e., mapping from terms to types, e.g., $3 \to R^3$)
Classical Foundations
(no encoding of proofs in favor of a direct expression)
PVS
by Sam Owre, John Rushby, and Natarajan Shankar (1992 ff.)
– predicate subtypes and (non-structurally) dependent types
R_0
by Ken Kubota (2015, publication in 2017)
– a purely syntactic approach, Hilbert-style, dependent types
Constructive Foundations
(encoding of proofs based on the Curry-Howard isomorphism)
Automath
by N. G. de Bruijn (1967 – early 1980s)
– typed λ-calculus with dependent types
Intuitionistic Type Theory
by Per Martin-Löf (in the 1970s)
Coq
by Thierry Coquand and Gérard Huet (1984 ff.)
– intensional impredicative version
Nuprl
by Robert L. Constable (1984 ff.)
– extensional predicative version of intuitionistic type theory
Agda
by Catarina Coquand (1999 ff.) and Ulf Norell (2007 ff.)
– intensional predicative extension



Contents

1 Literature and Links

- The Simple Theory of Types: [Church, 1940]

- $\mathcal{Q}_0$: [Andrews, 2002, pp. 210–215; first edition Andrews, 1986, pp. 161–165; very first publication in Andrews, 1963, pp. 345 f., 350]. Elementary logic is developed without any appeal to the Axiom of Choice, which may be assumed later [Andrews, 2002, p. 236]. Adding an Axiom of Infinity (as Axiom 6) for number theory yields $\mathcal{Q}_0^\infty$ [Andrews, 2002, p. 259]. A short description [Andrews, 2014a] is available online at:
http://plato.stanford.edu/entries/type-theory-church/#ForBasEqu

- HOL: [Gordon, 2001] (revised version of the 1985 paper), [Gordon and Melham, 1993].[1] The part on the logic written by Andrew Pitts[2] is available online (with minor modifications) at:
http://freefr.dl.sourceforge.net/project/hol/hol/kananaskis-10/kananaskis-10-logic.pdf
For a historical account on the HOL family, see [Gordon, 2000].

- HOL4: [Slind and Norrish, 2008], http://hol-theorem-prover.org

- HOL Light: [Harrison, 2009],[3] http://www.cl.cam.ac.uk/~jrh13/hol-light/

- HOL Zero: [Adams, 2010], http://www.proof-technologies.com/holzero/

- Isabelle/HOL: [Paulson, 1989], http://www.cl.cam.ac.uk/research/hvg/Isabelle/

- ProofPower: [Arthan and Jones, 2005], http://www.lemma-one.com/ProofPower/index/

- $\mathcal{Q}$: [Andrews, 1965]

- Extended HOL: [Melham, 1993b]

- F_ω: [Girard, 1986][4]

[1] Further definitional principles are proposed and discussed in [Kumar et al., 2014], [Arthan, 2014], and [Arthan, 2016].

[2] Part III: The HOL Logic [Gordon and Melham, 1993, pp. 191–232]. "Part III presents the logic supported by HOL ('The HOL Logic'). It consists of two chapters: an informal introduction and a formal set-theoretic semantics (written by Andrew Pitts)." [Gordon and Melham, 1993, p. xv] In an email to the author, Andrew Pitts confirmed that originally he wrote the material in the file available online, which contains both (!) chapters.

[3] Further motivation for HOL Light is explained in [Harrison, 1995].

[4] For an introduction, see also chapter 30 of [Pierce, 2002, pp. 449 ff.].

- HOL2P: [Völker, 2007], http://cswww.essex.ac.uk/staff/norbert/hol2p/
- HOL-Omega (HOL_ω): [Homeier, 2009], http://www.trustworthytools.com/id17.html
- Automath: [Daalen, 1994],[5] http://www.win.tue.nl/automath/
  A modern reimplementation: [Wiedijk, 2002], http://www.cs.ru.nl/~freek/aut/
- PVS: [Owre, Rushby, and Shankar, 1992], http://pvs.csl.sri.com
- $\mathcal{R}_0$: [Kubota, 2015], http://doi.org/10.4444/100.10
- Coq: [Coquand and Huet, 1985], [Coquand and Huet, 1988], [Coquand and Paulin, 1990], http://coq.inria.fr
- Nuprl: [Constable, 1986], http://www.nuprl.org
- Agda: [Norell, 2007],[6] http://wiki.portal.chalmers.se/agda/[7]

Please note that logics and their implementation below the expressiveness of higher-order logic (e.g., first-order logic) are not considered.

Neither the graph nor the above list are comprehensive. Further logics are William M. Farmer's $\mathcal{Q}_0^u$ and $\mathcal{Q}_0^{uqe}$ (extensions of Andrews' $\mathcal{Q}_0$ with undefinedness). Further type-theoretic systems are TPS by Peter B. Andrews et al., IMPS (with partial functions and subtypes) by William M. Farmer et al., and *Prooftoys* (a natural deduction variant of Andrews' $\mathcal{Q}_0$) by Cris Perdue, and a well-known set-theoretic system is *Mizar* by Andrzej Trybulec. Logical frameworks such as *Isabelle* by Larry Paulson and *Metamath* by Norman Megill can implement both type-theoretic and set-theoretic logics. Historically, "[t]he original LCF team at Stanford consisted of Robin Milner, assisted by Whitfield Diffie" [Gordon, 2000, p. 169] – now known for the Diffie-Hellman (DH) key exchange in cryptography. Both Rob Arthan and Roger Bishop Jones generously credit each other for having the main rôle in the creation of ProofPower.

- $\mathcal{Q}_0^u$: [Farmer, 2014]
- $\mathcal{Q}_0^{uqe}$: [Farmer, 2015]
- TPS: http://gtps.math.cmu.edu/tps.html
- IMPS: http://imps.mcmaster.ca
- Prooftoys (and Mathtoys): http://prooftoys.org, http://mathtoys.org
- Mizar: http://mizar.org
- Metamath: http://metamath.org

[5] "The standard reference about Automath is [Nederpelt, Geuvers, and Vrijer, 1994]. This is a compilation of almost all important Automath publications. It contains as (A.3) a good introduction to Automath [Daalen, 1994]. The main important Automath paper that is missing from this collection is the paper [de Bruijn, 1991] about telescopes." [Wiedijk, 2002, p. 365]

[6] "The particular choice of type theory is not crucial and the theory we choose is roughly Luo's UTT [Luo, 1994] extended with Σ-types and η-laws." [Norell, 2007, p. 14]

[7] The current version is Agda 2. For Agda version 1, see: http://ocvs.cfv.jp/Agda/.

A more comprehensive list with descriptions is available at: http://www.nuprl.org/Intro/others.html.

For an excellent discussion, and a comparison with set-theoretic approaches such as Mizar, see [Wiedijk, 2003] and [Wiedijk, 2007].

For a survey on interactive theorem proving, see [Harrison, Urban, and Wiedijk, 2014], which includes a description of the recently completed formalization of the proof of the *Kepler conjecture*. Despite the currently limited expressiveness of the formal language (lacking explicit quantification over type variables and dependent types[8]), in 2014, theorem provers verified this proof at the front of mathematical research established by Thomas Hales together with Ferguson in 1998 and published in full in 2006.[9]

2 Design Decisions

- Alonzo Church on the *reduction to functions of one argument*: "Adopting a device due to Schönfinkel, we treat a function of two variables as a function of one variable whose values are functions of one variable, and a function of three or more variables similarly." [Church, 1932, p. 352]

- Alonzo Church *introducing λ-notation*: "If $\mathbf{M}$ is any formula containing the variable $\mathbf{x}$, then $\lambda\mathbf{x}[\mathbf{M}]$ is a symbol for the function whose values are those given by the formula." [Church, 1932, p. 352]

- Peter B. Andrews on the *description operator* (and Axiom 5 of $\mathcal{Q}_0$): "Henkin remarks at the end of [Henkin, 1963] that when one passes from the theory of propositional types to the full theory of finite types, it becomes necessary to add a constant $\iota_{1(01)}$ to denote a descriptor function, and an appropriate axiom involving this constant. We note that for this axiom it suffices to take the simple formula $\iota_{1(01)}(\lambda x_1(x_1 \equiv y_1)) \equiv y_1$ [...]." [Andrews, 1963, p. 350]

- Peter B. Andrews on *finding as simple, natural, and expressive a formulation of type theory as possible*: "Chapter 5 introduces a system $\mathcal{Q}_0$ of typed λ-calculus which is essentially the system introduced by Alonzo Church in [Church, 1940], except that [...] equality relations rather than quantifiers and propositional connectives are primitive. [...] The discussion turns to finding as simple, natural, and expressive a formulation of type theory as possible. [...] This leads to an exposition of the basic ideas underlying $\mathcal{Q}_0$." [Andrews, 2002, p. xvi]

- Peter B. Andrews on *expressiveness* as the main criterion for the design of $\mathcal{Q}_0$: "Therefore we shall turn our attention to finding a formulation of type theory which is as expressive as possible, allowing mathematical ideas to be expressed precisely with a minimum of circumlocutions, and which is as simple and economical as is possible without sacrificing expressiveness. The reader will observe that the formal language we find arises very naturally from a few fundamental design decisions." [Andrews, 2002, pp. 205 f.]

[8] "The theorem prover HOL Light [...] is based on a logic without dependent types, but we can still encode the index N as a type (roughly, an arbitrary indexing type of size N)." [Hales and Harrison, 2010, p. 6]

[9] The computer aided verification became a desire, as the traditional peer review of the proof of about 300 pages and 40,000 lines of additional computer code had to be given up after four years of team efforts. Initiated by Hales in 2006 under the project name *Flyspeck*, the verification was completed successfully in 2014, proving the *Kepler conjecture* from 1611 (forming a part of Hilbert's 18th problem) which "states that no arrangement of identical balls in ordinary 3-dimensional space has a higher packing density than the obvious cannonball arrangement" [Harrison, Urban, and Wiedijk, 2014, p. 192] with a density of $\pi/\sqrt{18}$ ($\approx 74\,\%$). "The formal verifications have been carried out in the HOL Light and Isabelle proof assistants, with a second verification of the main statement in HOL Zero." [Hales, 2015, p. 2]

- Peter B. Andrews on *equality as the basic primitive notion*: "A formulation of type theory based on these ideas was introduced by Church in [Church, 1940], and proved complete by Henkin in [Henkin, 1950]. In this system equality can be defined using connectives and quantifiers. However, it is also possible to define connectives and quantifiers in terms of equality. Equality is a very basic and simple notion, so instead of using Church's formulation of type theory, we shall use a slight variant of it (first introduced in [Henkin, 1963] and simplified in [Andrews, 1963]) in which equality is taken as the basic primitive notion." [Andrews, 2002, p. 208]

- Peter B. Andrews introducing *ordered pairs* (*without extending the formal language*): "The expression $[\lambda g_{ooo} \blacksquare g_{ooo} x_o y_o]$ can be used to represent the ordered pair $\langle x_{[o]}, y_o \rangle$". [Andrews, 2002, p. 208]

- Peter B. Andrews introducing *induction* as part of the definition of natural numbers (*without extending the formal language*): "The Induction Principle simply limits the size of the set." [Andrews, 2002, p. 259] "We next define the natural numbers. These are equivalence classes of sets of individuals, and so have type $(o(o\imath))$.
  Definitions. Let σ be the type symbol $(o(o\imath))$.
  [...]
  $\mathbb{N}_{o\sigma}$ stands for $[\lambda n_\sigma \forall p_{o\sigma} \blacksquare [p_{o\sigma} 0_\sigma \ \wedge \ \forall x_\sigma \blacksquare p_{o\sigma} x_\sigma \supset p_{o\sigma} \blacksquare S_{\sigma\sigma} x_\sigma] \supset p_{o\sigma} n_\sigma]$
  [...]
  [...] The wff S represents the successor function. [...] $\mathbb{N}_{o\sigma}$ represents the set of natural numbers, i.e., the intersection of all sets which contain 0 and are closed under S." [Andrews, 2002, p. 260]

- Peter B. Andrews introducing *recursion* by defining a recursion operator (*without extending the formal language*): "Indeed, since f is uniquely determined by h and g, we can define a recursion operator R such that $f = Rhg$." [Andrews, 2002, p. 281]
  "Definition. $R_{\sigma\sigma\sigma(\sigma\sigma\sigma)}$ stands for $[\lambda h_{\sigma\sigma\sigma} \lambda g_\sigma \lambda n_\sigma \,\imath\, m_\sigma \forall w_{o\sigma\sigma} \blacksquare [w_{o\sigma\sigma} 0_\sigma g_\sigma \ \wedge \ \forall x_\sigma \forall y_\sigma \blacksquare w_{o\sigma\sigma} x_\sigma y_\sigma \supset w_{o\sigma\sigma} [S_{\sigma\sigma} x_\sigma] h_{\sigma\sigma\sigma} x_\sigma y_\sigma] \supset w_{o\sigma\sigma} n_\sigma m_\sigma]$.
  6400. $\vdash [R_{\sigma\sigma\sigma(\sigma\sigma\sigma)} h_{\sigma\sigma\sigma} g_\sigma] 0_\sigma = g_\sigma \ \wedge \ \dot{\forall} n_\sigma \blacksquare [R h_{\sigma\sigma\sigma} g_\sigma][S_{\sigma\sigma} n] = h_{\sigma\sigma\sigma} n_\sigma \blacksquare [R h_{\sigma\sigma\sigma} g_\sigma] n_\sigma$"
  [Andrews, 2002, p. 282, boldface as in the original] ("We [...] let $[\imath\, \mathbf{z}_\gamma \mathbf{A}]$ stand for $\iota_{\gamma(o\gamma)}[\lambda \mathbf{z}_\gamma \mathbf{A}]$." [Andrews, 2002, p. 234, boldface as in the original]. For the definition of the universal quantifier on numbers $\dot{\forall}$, see [Andrews, 2002, p. 260].)
  "To illustrate the usefulness of R, define $+_{\sigma\sigma\sigma}$ as $R_{\sigma\sigma\sigma(\sigma\sigma\sigma)}[\lambda x_\sigma S_{\sigma\sigma}]$, and $\mathbf{A}_\sigma + \mathbf{B}_\sigma$ as $+_{\sigma\sigma\sigma} \mathbf{A}_\sigma \mathbf{B}_\sigma$. [...] Similarly, $\times_{\sigma\sigma\sigma}$ can be defined as $[\lambda m_\sigma \blacksquare R_{\sigma\sigma\sigma(\sigma\sigma\sigma)}[\lambda x_\sigma \blacksquare +_{\sigma\sigma\sigma} m_\sigma] 0_\sigma]$, and $\mathbf{A}_\sigma \times \mathbf{B}_\sigma$ as $\times_{\sigma\sigma\sigma} \mathbf{A}_\sigma \mathbf{B}_\sigma$." [Andrews, 2002, p. 284, boldface as in the original]

- Andrew M. Pitts on the *natural deduction* system HOL in comparison to the *axiomatic (Hilbert-style) deductive system* $\mathcal{Q}_0$: "From a logical point of view, it would be better to have a simpler substitution primitive, such as 'Rule R' of Andrews' logic $\mathcal{Q}_0$, and then to derive more complex rules from it." [Gordon and Melham, 1993, p. 213]

- Mike Gordon characterizing the *HOL logic* (without mentioning the definability of new types): "The HOL logic is a version of Church's simple theory of types [Church, 1940] modified by:
  − allowing types to contain variables (i.e. be *polymorphic*) and
  − simplifying the axioms and rules of inference." [Gordon, 2001, p. 4]

- Mike Gordon and Peter B. Andrews defining the conditional using *the epsilon operator vs. the description operator*.
  Mike Gordon (HOL): "Many things that are normally primitive can be defined using the ε-operator. For example, the conditional term $\mathsf{Cond}\ t\ t_1\ t_2$ (meaning 'if t then t_1 else t_2') can be defined by:
  $\mathsf{Cond}\ t\ t_1\ t_2 \ = \ \varepsilon x.((t = \mathsf{T}) \Rightarrow (x = t_1)) \ \wedge \ ((t = \mathsf{F}) \Rightarrow (x = t_2))$" [Gordon, 2001, p. 24]

Peter B. Andrews ($\mathcal{Q}_0$): “DEFINITION: Let $C_{\gamma o\gamma\gamma}$ be
$[\lambda x_\gamma \lambda y_\gamma \lambda p_o \,\imath\, q_\gamma \,\blacksquare\, [p_o \wedge \,\blacksquare\, x_\gamma = q_\gamma] \vee [\sim p_o \wedge \,\blacksquare\, y_\gamma = q_\gamma]]$
$C_{\gamma o\gamma\gamma} x_\gamma y_\gamma p_o$ can be read ‘if p_o then x_γ, else y_γ’.
5313. $\vdash [C_{\gamma o\gamma\gamma} x_\gamma y_\gamma T_o = x_\gamma] \wedge \,\blacksquare\, C_{\gamma o\gamma\gamma} x_\gamma y_\gamma F_o = y_\gamma$” [Andrews, 2002, p. 235, boldface as in the original]
Formal verification of theorem **5313** in the $\mathcal{R}_0$ implementation (with type abstraction):
“$:= \quad CHOOS \quad \ldots$
$\ldots [\lambda t_\tau.[\lambda x_t.[\lambda y_t.[\lambda p_o.(\iota_{t(ot)}[\lambda q_t.(\vee_{ooo}(\wedge_{ooo} p_o(=_{ott} x_t q_t))(\wedge_{ooo}(\sim_{oo} p_o)(=_{ott} y_t q_t)))_o])_t]_{(to)}]_{(tot)}]_{(tott)}]$
$\#$ wff 2080 : $[\lambda t.[\lambda x.[\lambda y.[\lambda p.(\iota\,[\lambda q.(\vee\,(\wedge\, p\,(=x\,q))\,(\wedge\,(\sim p)\,(=y\,q)))])]]]]_{\backslash 4o\backslash 3\backslash 2\tau}$ [...]
[...]
$\#$ $\wedge\,(=(CHOOS\,t\,x\,y\,T)\,x)\,(=(CHOOS\,t\,x\,y\,F)\,y) \quad := \quad A5313$
$\#$ $\wedge_{ooo}(=_{ott}(CHOOS_{\backslash 4o\backslash 3\backslash 2\tau} t_\tau x_t y_t T_o) x_t)(=_{ott}(CHOOS_{\backslash 4o\backslash 3\backslash 2\tau} t_\tau x_t y_t F_o) y_t)$
$:= \quad A5313$” [Kubota, 2016, pp. 165, 174]

- Mike Gordon on *explicit type variable quantification* in HOL: “In future versions of HOL it is expected that there will be explicit type variable quantification [Melham, 1993b], i.e., terms of the form $\forall\alpha.t$ (where α is a type variable). The right hand side of definitions will be required to be closed with respect to both term and type variables. Melham has shown that this will make defining mechanisms much cleaner and also permit an elegant treatment of type specifications.” [Gordon, 2000, p. 175, fn. 7]

- Tom Melham on *quantification over type variables at the level of terms (not types)*: “Note that we are not proposing an extension to the type language of HOL – the quantifications $\forall\alpha$.P and $\exists\alpha$.P are new term constructs, and not type constructs of the kind found (for example) in Girard’s system F [...]. The extended logic proposed here resembles system $\mathcal{Q}$, a transfinite type theory due to Andrews [Andrews, 1965]. It is, however, still much weaker than Andrews’ system.” [Melham, 1993b, pp. 8 f., emphasis as in the original]

- Tom Melham on *the relation of Melham’s Extended HOL to Andrews’ system $\mathcal{Q}$*: “I have found Andrews’ book [Andrews, 1965] invaluable in working out many of the technical details of the extension to HOL proposed here.” [Melham, 1993b, p. 23]

- Mike Gordon suggesting *type abstraction* for HOL: “The syntax and semantics of type variables are currently being studied by several logicians. A closely related area is the theory of ‘second order’ λ-terms like $\lambda\alpha.\,\lambda x{:}\alpha.\,x$, perhaps such terms should be included in the HOL logic.” [Gordon, 2001, p. 22]

- John Harrison on the *motivation for HOL Light*: “Part of my objective in HOL Light was to arrive at a simpler and more satisfying logical kernel that could nevertheless be the foundation of a practical tool. In particular HOL Light’s foundations address a couple of the issues [mentioned in] some self-criticism [in] the HOL documentation: the complicated primitive substitution rule and [...] the way the epsilon operator is plumbed deep into the foundations. [...]
[...]
In HOL Light the basic logical notions including the quantifiers are all defined independent of the epsilon operator (in contrast to its use even to define ‘there exists’ in the original HOL). The choice operator is eventually introduced but it is quite late. [...]
[...]
Ultimately, all the quantifiers and connectives in HOL Light are defined in terms of equality alone. In this respect the foundations are reminiscent of one of Andrews’[...] systems, but with the distinction that the definitions are intuitionistically admissible. [...]
[...]

I think there is something quite satisfying about the derivation of higher-order logic from these simple foundations based just on equality in a simple world of functions. [...]" [Harrison, 2016]

- Larry Paulson on the relevance of *natural deduction* for *automation*: "Church's axiom system is now antiquated, largely dating back to *Principia Mathematica*. There are improved formulations but most use the Hilbert style. Natural deduction is far superior for automated proof." [Paulson, 1989, p. 3, emphasis as in the original]

- Sam Owre and Natarajan Shankar on *(non-structurally) dependent types* in PVS: "One very important consequence of the above extension of the universe is that all type dependencies must be bounded [...]. This property is easily proved by induction on the structure of a PVS type [...]. In particular, there is no way to define a type constructor T^n in PVS [...]. If unbounded type dependencies were allowed in PVS, one can construct a dependent type such as $[n : nat \to T^n]$ whose representation is not in U as defined above." [Owre and Shankar, 1999, p. 35, emphasis as in the original] "PVS admits only a very restricted form of type dependency. In a dependent type $T(n)$, the parameter n can occur only within subtype predicates in $T(n)$. This means that the structure of $T(n)$ is invariant with respect to n. All possible ways of introducing type dependencies in PVS preserve this invariant. It follows that there is no way of defining a type $T(n)$, where $T(n)$ is A^n, i.e., the n-tuple over the type A." [Shankar and Owre, 2000, p. 45, emphasis as in the original]

- Robert L. Constable et al. on choosing a *predicative type structure* for Nuprl: "The type structure hierarchy of Nuprl resembles that of *Principia Mathematica*, the ancestor of all type theories. The hierarchy manifests itself in an unbounded cumulative hierarchy of universes, U_1, U_2, ..., where by cumulative hierarchy we mean that U_i is in U_{i+1} and that every element of U_i is also an element of U_{i+1}. Universes are themselves types, and every type occurs in a universe. In fact, A is a type if and only if it belongs to a universe. Conversely all the elements of a universe are types." [Constable, 1986, p. 5, emphasis as in the original] "The concept of a universe in this role, to organize the hierarchy of types, is suggested in [Artin, Grothendieck, & Verdier 72] and was used by Martin-Löf [Martin-Löf 73]. This is a means of achieving a *predicative* type structure as opposed to an *impredicative* one as in Girard [Girard 71] or Reynolds [Reynolds 83]." [Constable, 1986, p. 5, fn. 2, emphasis as in the original]

- Gérard Huet on the *development of* Coq: "The logical language used by Coq is a variety of type theory, called the *Calculus of Inductive Constructions*. [...] The λ-calculus notation, originally used for expressing functionality, could also be used as an encoding of natural deduction proofs. This Curry-Howard isomorphism was used by N. de Bruijn in the *Automath* project, the first full-scale attempt to develop and mechanically verify mathematical proofs. [...] Exploiting this Curry-Howard isomorphism, notable achievements in proof theory saw the emergence of two type-theoretic frameworks; the first one, Martin-Löf's *Intuitionistic Theory of Types*, attempts a new foundation of mathematics on constructive principles. The second one, Girard's polymorphic λ-calculus F_ω, is a very strong functional system in which we may represent higher-order logic proof structures. Combining both systems in a higher-order extension of the Automath languages, T. Coquand presented in 1985 the first version of the *Calculus of Constructions*, CoC. [...] The formalism was extended in 1989 by T. Coquand and C. Paulin with primitive inductive definitions, leading to the current *Calculus of Inductive Constructions*. [...]
[...]
A first implementation of CoC was started in 1984 by G. Huet and T. Coquand. [...]" [Huet, 1995, emphasis as in the original]

- Diederik van Daalen on *coding* logic using the *propositions-as-types* notion: "Now there are different ways of coding some logic into the objects-and-types framework. Here we only mention a so-called *functional interpretation* of logic, which gives rise to the *propositions-as-types* notion. This idea of interpreting logic was developed independently by de Bruijn and certain others, of whom we mention Howard [Howard, 1980], Prawitz [Prawitz 71], Girard [Girard 72] and Martin-Löf [Martin-Löf 75a]." [Daalen, 1994, p. 111, emphasis as in the original]

- Ana Bove and Peter Dybjer on *variants of Martin-Löf's intuitionistic type theory*: "The Curry-Howard interpretation was the basis for Martin-Löf's intuitionistic type theory [19,20,21]. In this theory propositions and types are actually identified. Although Martin-Löf's theory was primarily intended to be a foundational system for constructive mathematics, it can also be used as a programming language [20]. From the early 1980's and onwards, a number of computer systems implementing variants of Martin-Löf type theory were built. The most well-known are the NuPRL [Constable, 1986] system from Cornell implementing an extensional version of the theory, and the Coq [32] system from INRIA in France implementing an intensional impredicative version. The Agda system implements an intensional predicative extension of Martin-Löf type theory. It is the latest in a sequence of systems developed in Göteborg." [Bove and Dybjer, 2009, p. 59]

- Ulf Norell about using *encoding properties of values as types*: "Since dependent types allows types to talk about values, we can encode properties of values as types whose elements are proofs that the property is true. This means that a dependently typed programming language can be used as a logic." [Norell, 2009, p. 230]

- Freek Wiedijk introducing the notion of *Pollack-consistency*: "[W]e introduce the notion of *Pollack-consistency*. This property is related to a system being able to correctly parse formulas that it printed itself. In current systems it happens regularly that this fails." [Wiedijk, 2012, p. 85, emphasis as in the original]

- Mark Adams introducing the notion of *faithfulness*: "Also note that none of these notions fully address the issue of *faithfulness*, where internal representation and concrete syntax correctly correspond. A printer that printed `false` as `true` and `true` as `false` might be Pollack-consistent but would not be faithful." [Adams, 2016, p. 21, emphasis as in the original]

- Mark Adams on the *Pollack-consistency* of HOL Zero (not taking into account unpublished $\mathcal{R}_0$): "We believe HOL Zero does not suffer from any incompleteness or ambiguity in its parsers or printers, and printed output can always be parsed back in to give the same internal representation. This would make HOL Zero's parsers and printers well-behaved and Pollack-consistent. As far as we know, this would be a first amongst not only HOL systems, but also various other theorem proving systems that support concrete syntax, such as Coq and Mizar." [Adams, 2016, p. 34]

- Norman Megill on using the *minimum possible framework* with Metamath (not taking into account unpublished $\mathcal{R}_0$): "Unlike most other systems, Metamath attempts to use the minimum possible framework needed to express mathematics and its proofs. Other systems do not consider that aspect necessarily important, and their underlying computer programs can be large and complex in order to perform mathematical reasoning at a higher level. Metamath's proofs are often quite long compared to those of other systems, but they are completely transparent with nothing hidden from the user. All reasoning is done directly in the proof itself rather than by algorithms embedded in the verification program. Metamath is unique in this sense, offering an alternative approach for those attracted to its philosophy of simplicity." [Megill, 2017a]

- Norman Megill on *reverse engineering* in mathematical history: “As humans, we observe interesting patterns in these ‘meaningless’ symbol strings as they evolve from the axioms, and we attach meaning to them. One result is the set of natural numbers, whose properties match those we observe when we count everyday objects, and their extensions to rational and real numbers. Of course, numbers were discovered centuries before set theory, and historically they were ‘reversed engineered’ back to the axioms of set theory. The proof of $2 + 2 = 4$ shows what was involved in that reverse engineering, representing the work of many mathematicians from Dedekind to von Neumann. At the other extreme of abstraction is the theory of infinite sets or transfinite cardinal numbers. Some of the world’s most brilliant mathematicians have given us deep insight into this mysterious and wondrous universe, which is sometimes called ‘Cantor’s paradise.’ ” [Megill, 2017b]

3 Criticism

- Mike Gordon on the use of the *epsilon operator* in HOL: “It must be admitted that the ε-operator looks rather suspicious.” [Gordon, 2001, p. 24] “The inclusion of ε-terms into HOL ‘builds in’ the Axiom of Choice [...].” [Gordon, 2001, p. 24]

- Freek Wiedijk on the *(current) HOL family*: “There is one important reason why the HOLs are not yet attractive enough to be taken to be the QED system:

  – The HOL type system is too poor. As we already argued in the previous section, it is too weak to properly do abstract algebra.
    But it is worse than that. In the HOL type system there are no *dependent types*, nor is there any form of *subtyping*. (Mizar and Coq both have dependent types and some form of subtyping. In Mizar the subtyping is built into the type system. In Coq a similar effect is accomplished through the use of automatic *coercions*.)
    For formal mathematics it is essential to both have dependent types and some form of subtyping.” [Wiedijk, 2007, p. 130, emphasis as in the original]

- Freek Wiedijk on the *(current) Coq system*: “There are two important reasons why Coq is not yet attractive enough to be taken to be the QED system:

  – The foundations of Coq are too complicated. They are baroque to say the least. Maybe they are even *beyond* baroque. They might even be called *rococo*.
    There is no paper in which the foundations of Coq are spelled out in full mathematical precision. [...]
    [...]
    Also, the foundations of Coq are sufficiently complicated that they are tinkered with, and therefore change between versions of the system.

  – As already noted in the previous section, Coq is not designed for classical mathematics, which means that doing classical & extensional mathematics in it is not as easy as one would like it to be.” [Wiedijk, 2007, p. 130, emphasis as in the original]

- John Harrison, Josef Urban, and Freek Wiedijk on the *(current) HOL family*: “Another limitation of the simple HOL type system is that there is no explicit quantifier over polymorphic type variables, which can make many standard results like completeness theorems and universal properties awkward to express, though there are extensions with varying degrees of generality that fix this issue [Melham, 1993a; Völker, 2007; Homeier, 2009]. Inflexibilities of these kinds certainly arise in simple type theories, and it is not even clear that more flexible *dependent* type theories (where types can be parametrized by terms) are immune. For example, in one of the most impressive formalization efforts to date [Gonthier *et al.*, 2013] the entire group theory framework is developed in terms of subsets of a single universe group, apparently to avoid the complications from groups with general and possibly heterogeneous types." [Harrison, Urban, and Wiedijk, 2014, pp. 170 f., emphasis as in the original]

- John Harrison, Josef Urban, and Freek Wiedijk on the *(current) Coq system*: "Indeed, there does not seem to be any precise written specification of Coq's current foundations, other than the actual code." [Harrison, Urban, and Wiedijk, 2014, p. 188, fn. 29]

- Peter B. Andrews on *a false soundness assertion in Henkin's article on completeness*: "As shown in [Andrews, 1972b], there is a nonstandard general model for $\mathscr{C}$ in which the Axiom of Extensionality $\forall x_\beta(f_{\alpha\beta}x_\beta = g_{\alpha\beta}x_\beta) \supset (f_{\alpha\beta} = g_{\alpha\beta})$ is not valid, since the sets in this model are so sparse that the denotation of the defined equality formula $\mathscr{Q}_{o\alpha\alpha}$ is not the actual equality relation. Thus, Theorem 2 of [Henkin, 1950] (which asserts the completeness and soundness of $\mathscr{C}$) is technically incorrect. The apparently trivial soundness assertion is false." [Andrews, 2014b, p. 70] (Note that [Andrews, 1972b] bases on the preceding [Andrews, 1972a].)

4 Historical Notes

- Alonzo Church *first proposing a third alternative to Russell's type theory and Zermelo's axiomatic set theory* in 1932: "Rather than adopt the method of Russell for avoiding the familiar paradoxes of mathematical logic, or that of Zermelo, both of which appear somewhat artificial, we introduce for this purpose, as we have said, a certain restriction on the law of excluded middle." [Church, 1932, p. 347]

- Felice Cardone and J. Roger Hindley on the *invention of λ-calculus and the origin of the λ-notation*: "The λ-calculus was invented in about 1928 by Alonzo Church, and was first published in [Church, 1932]. Church was born in 1903 in Washington D.C. and studied at Princeton University. He made his career at Princeton until 1967, though in 1928–29 he visited Göttingen and Amsterdam.
  Around 1928 he began to build a formal system with the aim of providing a foundation for logic which would be more natural than Russell's type theory or Zermelo's set theory, and would not contain free variables (for reasons he explained in [Church, 1932, pp. 346–347]). He chose to base it on the concept of function rather than set, and his primitives included abstraction $\lambda x[M]$ and application $\{F\}(X)$, which we shall call here '$\lambda x.M$' and '(FX)'.
  (By the way, why did Church choose the notation 'λ'? In [Church, 1964, §2] he stated clearly that it came from the notation '$\hat{x}$' used for class-abstraction by Whitehead and Russell, by first modifying '$\hat{x}$' to '$\wedge x$' to distinguish function-abstraction from class-abstraction, and then changing '$\wedge$' to 'λ' for ease of printing. This origin was also reported in [Rosser, 1984, p.338]. On the other hand, in his later years Church told two enquirers that the choice was more accidental: a symbol was needed and 'λ' just happened to be chosen.)
  As mentioned earlier, Church was not the first to introduce an explicit notation for function-abstraction. But he was the first to state explicit formal conversion rules for the notation, and to analyse their consequences in depth." [Cardone and Hindley, 2009, pp. 730 f.]

- Felice Cardone and J. Roger Hindley on the *evolution of Church's simple type theory*: "Church's simple type theory was a function-based system, stemming from ideas of Frank Ramsey and Leon

Chwistek in the 1920s, for simplifying the type theory of [Russell and Whitehead, 1913]. Church lectured on his system in Princeton in 1937–38 before publishing it in [Church, 1940], and his lectures were attended by Turing, who later made some technical contributions. [...]
Church's system was analysed and extended in a series of Princeton Ph.D. theses from the 1940s onward, of which perhaps the best known are Leon Henkin's in 1947, published in [Henkin, 1950], and Peter Andrews', published in [Andrews, 1965]. Henkin gave two definitions of model of typed λ (*standard* and *general* models), and proved the completeness of simple type theory with respect to general models. Andrews extended Church's system to make a smooth theory of tran[s]finite types." [Cardone and Hindley, 2009, p. 737, emphasis as in the original]

- Peter B. Andrews on *Henkin's rôle for the development of* $\mathcal{Q}_0$: "Quine described how to make these definitions in the short final section of [16], but Henkin developed this topic much further in [Henkin, 1963], introducing an axiomatic system and establishing its soundness and completeness. [...]
  [...]
  Henkin's work played a decisive role in my life. [...] Henkin's work developing a formulation of Church's type theory with equality (identity) as the sole logical primitive was particularly important for me. I used such a formulation of full type theory, called $\mathcal{Q}_0$, in my Ph.D. thesis [Andrews, 1965] and the textbook [Andrews, 2002].
  [...]
  It is important to realize the significance of Henkin's contribution in developing a formulation of type theory based on equality. It is a real improvement of the system $\mathscr{C}$ discussed in [Henkin, 1950], which has primitive constants for propositional connectives and quantifiers [...]" [Andrews, 2014b, pp. 68 f.]

- Andrew M. Pitts on *type variables* in HOL: "In Church's original formulation of simple type theory, type variables are part of the meta-language and are used to range over object language types. Proofs that contain type variables were understood as proof schemes (i.e. families of proofs). To support such proof schemes *within* the HOL logic, type variables have been added to the object language type system." [Gordon and Melham, 1993, p. 195, emphasis as in the original]

- Andrew M. Pitts on the *set of rules and axioms* for HOL: "The particular set of rules and axioms chosen to axiomatize the HOL logic is rather arbitrary. It is partly based on the rules that were used in the LCF logic PPλ, since HOL was implemented by modifying the LCF system. In particular, the substitution rule `SUBST` is exactly the same as the corresponding rule in LCF; the code implementing this was written by Robin Milner and is highly optimized. Because substitution is such a pervasive activity in proof, it was felt to be important that the system primitive be as fast as possible. From a logical point of view, it would be better to have a simpler substitution primitive, such as 'Rule R' of Andrews' logic $\mathcal{Q}_0$, and then to derive more complex rules from it." [Gordon and Melham, 1993, p. 213]

- Mike Gordon on the *genesis of HOL*: "[...] [T]he terms [...] could be encoded [...] in such a way that the LSM expansion-law just becomes a derived rule [...]. This approach is both more elegant and rests on a firmer logical foundation, so I switched to it and HOL was born. [...]
  The logic supported by Cambridge LCF has the usual formula structure of predicate calculus, and the term structure of the typed λ-calculus. The type system, due to Milner, is essentially Church's original one [Church, 1940], but with type variables moved from the meta-language to the object language (in Church's system, a term with type variables is actually a meta-notation – a term-schema – denoting a family of terms, whereas in LCF it is a single polymorphic term). [...]

[...] HOL employs the LCF substitution because I wanted to use the existing efficient code. As a result the HOL logic ended up with a rather *ad hoc* formal basis. Another inheritance from LCF is the use of a natural deduction logic (Church used a Hilbert-style formal system). [...]" [Gordon, 2000, p. 173, emphasis as in the original]

- Mike Gordon on the *development of HOL*: "The design of HOL was largely taken 'off the shelf,' the theory being classical higher order logic and the implementation being LCF. The development of the system was, at first, primarily driven by hardware verification case studies." [Gordon, 2000, p. 174]
- Mike Gordon on the *validation of HOL's definitional principles*: "[...] Dr Andrew Pitts was commissioned to validate HOL's definitional principles. He produced informal proofs that they could not introduce inconsistency [Gordon and Melham, 1993, Chapter 16]." [Gordon, 2000, p. 175]
- Ondřej Kunčar and Andrei Popescu *classifying the HOL logic*: "Polymorphic HOL, more precisely, Classic[al] Higher-Order Logic with Infinity, Hilbert Choice and Rank-1 Polymorphism, endowed with a mechanism for constant and type definitions, was proposed in the [eigh]ties as a logic for interactive theorem provers by Mike Gordon, who also implemented the seminal HOL theorem prover [Gordon and Melham, 1993]." [Kunčar and Popescu, 2015, p. 234]

For a detailed treatment of epsilon terms, see [Slater, 2016].

References

Adams, Mark (2010). "Introducing HOL Zero (Extended Abstract)". In: *Mathematical Software – ICMS 2010: Third International Congress on Mathematical Software. Kobe, Japan, September 13-17, 2010. Proceedings*. Ed. by Komei Fukuda et al. Vol. 6327. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, pp. 142–143. ISBN: 978-3-642-15581-9. DOI: 10.1007/978-3-642-15582-6_25.

— (2016). "HOL Zero's Solutions for Pollack-Inconsistency". In: *Interactive Theorem Proving: 7th International Conference, ITP 2016. Nancy, France, August 22–25, 2016. Proceedings*. Ed. by Jasmin Christian Blanchette and Stephan Merz. Vol. 9807. Lecture Notes in Computer Science. Switzerland: Springer International Publishing, pp. 20–35. ISBN: 978-3-319-43143-7. DOI: 10.1007/978-3-319-43144-4_2. URL: http://www.proof-technologies.com/holzero/hzsyntax_itp2016.pdf.

Andrews, Peter B. (1963). "A Reduction of the Axioms for the Theory of Propositional Types". In: *Fundamenta Mathematicae* 52, pp. 345–350. URL: http://matwbn.icm.edu.pl/ksiazki/fm/fm52/fm52124.pdf.

— (1965). *A Transfinite Type Theory with Type Variables*. Studies in Logic and the Foundations of Mathematics. Amsterdam: North-Holland.

— (1972a). "General Models, Descriptions, and Choice in Type Theory". In: *Journal of Symbolic Logic* 37.2, pp. 385–394. DOI: 10.2307/2272981. URL: http://www.jstor.org/stable/2272981.

— (1972b). "General Models and Extensionality". In: *Journal of Symbolic Logic* 37.2, pp. 395–397. DOI: 10.2307/2272982. URL: http://www.jstor.org/stable/2272982.

— (1986). *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Orlando/Florida et al.: Academic Press. ISBN: 0-12-058535-9 (Hardcover) / 0-12-058536-7 (Paperback).

— (2002). *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Second edition. Dordrecht, Boston, London: Kluwer Academic Publishers. ISBN: 1-4020-0763-9. DOI: 10.1007/978-94-015-9934-4.

Andrews, Peter B. (2014a). “Church’s Type Theory”. In: *Stanford Encyclopedia of Philosophy (Spring 2014 Edition)*. Ed. by Edward N. Zalta. URL: http://plato.stanford.edu/archives/spr2014/entries/type-theory-church/.

— (2014b). “A Bit of History Related to Logic Based on Equality”. In: *The Life and Work of Leon Henkin: Essays on His Contributions*. Ed. by María Manzano, Ildikó Sain, and Enrique Alonso. Studies in Universal Logic. Cham/Switzerland, Heidelberg, New York et al.: Springer International Publishing, pp. 67–71. ISBN: 978-3-319-09718-3. DOI: 10.1007/978-3-319-09719-0_8.

Arthan, Rob (2014). “HOL Constant Definition Done Right”. In: *Interactive Theorem Proving: 5th International Conference, ITP 2014. Held as Part of the Vienna Summer of Logic, VSL 2014. Vienna, Austria, July 14–17, 2014. Proceedings*. Ed. by Gerwin Klein and Ruben Gamboa. Vol. 8558. Lecture Notes in Computer Science. Cham/Switzerland, Heidelberg, New York et al.: Springer International Publishing, pp. 531–536. ISBN: 978-3-319-08969-0. DOI: 10.1007/978-3-319-08970-6_34. URL: http://www.lemma-one.com/papers/hcddr.pdf.

— (2016). “On Definitions of Constants and Types in HOL”. In: *Journal of Automated Reasoning* 56.3, pp. 205–219. ISSN: 1573-0670. DOI: 10.1007/s10817-016-9366-4. URL: http://link.springer.com/article/10.1007/s10817-016-9366-4.

Arthan, Rob and Roger Bishop Jones (2005). “Z in HOL in ProofPower”. In: *FACS FACTS* 2005-1, pp. 39–54. URL: http://www.bcs.org/upload/pdf/facts200503.pdf.

Bove, Ana and Peter Dybjer (2009). “Dependent Types at Work”. In: *Language Engineering and Rigorous Software Development: International LerNet ALFA Summer School 2008. Piriapolis, Uruguay, February 24 – March 1, 2008. Revised Tutorial Lectures*. Ed. by Ana Bove et al. Vol. 5520. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer-Verlag, pp. 57–99. ISBN: 978-3-642-03152-6. DOI: 10.1007/978-3-642-03153-3_2. URL: http://www.cse.chalmers.se/~peterd/papers/DependentTypesAtWork.pdf.

Cardone, Felice and J. Roger Hindley (2009). “Lambda-Calculus and Combinators in the 20th Century”. In: *Logic from Russell to Church*. Ed. by Dov M. Gabbay and John Woods. Vol. 5. Handbook of the History of Logic. Amsterdam, Boston et al.: North-Holland (Elsevier), pp. 723–817. ISBN: 978-0-444-51620-6. DOI: 10.1016/S1874-5857(09)70018-4. URL: http://www.users.waitrose.com/~hindley/SomePapers_PDFs/2006CarHin,HistlamRp.pdf.

Church, Alonzo (1932). “A Set of Postulates for the Foundation of Logic”. In: *Annals of Mathematics* 33.2, pp. 346–366. DOI: 10.2307/1968337. URL: http://www.jstor.org/stable/1968337.

— (1940). “A Formulation of the Simple Theory of Types”. In: *Journal of Symbolic Logic* 5.2, pp. 56–68. DOI: 10.2307/2266170. URL: http://www.jstor.org/stable/2266170.

Constable, Robert L. et al. (1986). *Implementing Mathematics with the Nuprl Proof Development System*. Englewood Cliffs/New Jersey: Prentice-Hall. ISBN: 0-13-451832-2.

Coquand, Thierry and Gérard Huet (1985). “Constructions: A Higher Order Proof System for Mechanizing Mathematics”. In: *EUROCAL ’85: European Conference on Computer Algebra. Linz, Austria, April 1–3 1985. Proceedings Vol. 1: Invited Lectures*. Ed. by Bruno Buchberger. Vol. 203. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer, pp. 151–184. ISBN: 3-540-15983-5. DOI: 10.1007/3-540-15983-5_13. URL: https://hal.inria.fr/inria-00076155/document.

— (1988). “The Calculus of Constructions”. In: *Information and Computation* 76.2, pp. 95–120. ISSN: 0890-5401. DOI: 10.1016/0890-5401(88)90005-3. URL: http://www.sciencedirect.com/science/article/pii/0890540188900053/pdf?md5=ccd76829e62449e4aac711bc9891d380&pid=1-s2.0-0890540188900053-main.pdf.

Coquand, Thierry and Christine Paulin (1990). “Inductively defined types”. In: *COLOG-88: International Conference on Computer Logic. Tallinn, USSR, December 12–16, 1988. Proceedings*. Ed. by Per Martin-Löf and Grigori Mints. Vol. 417. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer, pp. 50–66. ISBN: 3-540-52335-9. DOI: 10.1007/3-540-52335-9_47.

Daalen, Diederik T. van (1994). “A Description of Automath and Some Aspects of its Language Theory”. In: *Selected Papers on Automath*. Ed. by R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. Vol. 133. Studies in Logic and the Foundations of Mathematics. Article first published in 1973. Amsterdam, Lausanne et al.: Elsevier, pp. 101–126. ISBN: 0-444-89822-0. DOI: 10.1016/S0049-237X(08)70201-5.

de Bruijn, N. G. (1991). “Telescopic Mappings in Typed Lambda Calculus”. In: *Information and Computation* 91.2, pp. 189–204. ISSN: 0890-5401. DOI: 10.1016/0890-5401(91)90066-B. URL: http://www.sciencedirect.com/science/article/pii/089054019190066B/pdf?md5=fabb26ffae6939be7d2840a767838fb2&pid=1-s2.0-089054019190066B-main.pdf.

Farmer, William M. (2014). *Andrews’ Type Theory with Undefinedness*. Revised on June 28, 2014. URL: http://imps.mcmaster.ca/doc/andrews.pdf (visited on 10/23/2016).

— (2015). “Simple Type Theory with Undefinedness, Quotation, and Evaluation”. In: arXiv: 1406.6706 [math.LO].

Girard, Jean-Yves (1986). “The System F of Variable Types, Fifteen Years Later”. In: *Theoretical Computer Science* 45, pp. 159–192. ISSN: 0304-3975. DOI: 10.1016/0304-3975(86)90044-7. URL: http://www.sciencedirect.com/science/article/pii/0304397586900447/pdf?md5=3f995af59dff05620ae4f4744b05aae6&pid=1-s2.0-0304397586900447-main.pdf.

Gordon, Michael J. C. (2000). “From LCF to HOL: A Short History”. In: *Proof, Language, and Interaction*. Ed. by Gordon Plotkin, Colin Stirling, and Mads Tofte. Cambridge/Massachusetts et al.: MIT Press, pp. 169–185. ISBN: 0-262-16188-5. URL: http://www.cl.cam.ac.uk/~mjcg/papers/HolHistory.pdf.

— (2001). *HOL: A Machine Oriented Formulation of Higher Order Logic*. Revised version of the original technical report UCAM-CL-TR-68 from 1985. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.6103&rep=rep1&type=pdf.

Gordon, Michael J. C. and Thomas F. Melham, eds. (1993). *Introduction to HOL: A theorem proving environment for higher order logic*. The part on the logic written by Andrew Pitts is available online (with minor modifications) at: http://freefr.dl.sourceforge.net/project/hol/hol/kananaskis-10/kananaskis-10-logic.pdf. Cambridge: Cambridge University Press. ISBN: 0-521-44189-7.

Hales, Thomas C. et al. (2015). “A Formal Proof of the Kepler Conjecture”. In: arXiv: 1501.02155 [math.MG].

Hales, Thomas C., John Harrison, et al. (2010). “A Revision of the Proof of the Kepler Conjecture”. In: *Discrete & Computational Geometry* 44.1, pp. 1–34. ISSN: 1432-0444. DOI: 10.1007/s00454-009-9148-4. URL: http://www.cl.cam.ac.uk/~jrh13/papers/revkepler.pdf.

Harrison, John (1995). *HOL Done Right (draft from August 21, 1995)*. URL: http://www.cl.cam.ac.uk/~jrh13/papers/holright.pdf (visited on 11/12/2016).

— (2009). “HOL Light: An Overview”. In: *Theorem Proving in Higher Order Logics*. Ed. by Stefan Berghofer et al. Vol. 5674. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 60–66. ISBN: 978-3-642-03358-2. DOI: 10.1007/978-3-642-03359-9_4. URL: http://www.cl.cam.ac.uk/~jrh13/papers/hollight.pdf.

— (2016). *E-Mail to Ken Kubota and the Hol-info mailing list, October 23, 2016*. URL: https://sourceforge.net/p/hol/mailman/message/35444612/.

Harrison, John, Josef Urban, and Freek Wiedijk (2014). “History of Interactive Theorem Proving”. In: *Computational Logic*. Ed. by Jörg H. Siekmann. Vol. 9. Handbook of the History of Logic. Amsterdam, Boston et al.: North-Holland (Elsevier), pp. 135–214. ISBN: 978-0-444-51624-4. DOI: 10.1016/B978-0-444-51624-4.50004-6. URL: http://www.cl.cam.ac.uk/~jrh13/papers/joerg.pdf.

Henkin, Leon (1950). “Completeness in the Theory of Types”. In: *Journal of Symbolic Logic* 15.2, pp. 81–91. DOI: 10.2307/2266967. URL: http://www.jstor.org/stable/2266967.

— (1963). “A Theory of Propositional Types”. In: *Fundamenta Mathematicae* 52, pp. 323–344. URL: http://matwbn.icm.edu.pl/ksiazki/fm/fm52/fm52123.pdf. Errata ibid., vol. 53 (1964), p. 119, http://matwbn.icm.edu.pl/ksiazki/fm/fm53/fm53125.pdf.

Homeier, Peter V. (2009). "The HOL-Omega Logic". In: *Theorem Proving in Higher Order Logics: 22nd International Conference, TPHOLs 2009. Munich, Germany, August 17-20, 2009. Proceedings*. Ed. by Stefan Berghofer et al. Vol. 5674. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer, pp. 244–259. ISBN: 978-3-642-03358-2. DOI: 10.1007/978-3-642-03359-9_18. URL: http://www.trustworthytools.com/holw/holw-lncs5674.pdf.

Howard, William A. (1980). "The Formulae-as-Types Notion of Construction". In: *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Ed. by Jonathan P. Seldin and J. Roger Hindley. Original manuscript from 1969. London, New York et al.: Academic Press, pp. 479–490. ISBN: 0-12-349050-2.

Huet, Gérard (1995). *Credits (Homepage of The Coq Proof Assistant)*. URL: https://coq.inria.fr/doc/Reference-Manual002.html (visited on 10/16/2016).

Kubota, Ken (2015). *On the Theory of Mathematical Forms (draft from May 18, 2015)*. Unpublished manuscript. SHA-512: a0dfe205eb1a2cb29efaa579d68fa2e5 45af74d8cd6c270cf4c95ed1ba6f7944 fdcffaef2e761c8215945a9dcd535a50 011d8303fd59f2c8a4e6f64125867dc4. DOI: 10.4444/100.10.

— (2016). *Excerpt from [Kubota, 2015] (p. 1, pp. 165-174, and pp. 350-352)*. URL: http://www.kenkubota.de/files/R0_draft_excerpt_with_axioms_and_proof_of_C_function.pdf (visited on 08/13/2016).

Kumar, Ramana et al. (2014). "HOL with Definitions: Semantics, Soundness, and a Verified Implementation". In: *Interactive Theorem Proving: 5th International Conference, ITP 2014. Held as Part of the Vienna Summer of Logic, VSL 2014. Vienna, Austria, July 14–17, 2014. Proceedings*. Ed. by Gerwin Klein and Ruben Gamboa. Vol. 8558. Lecture Notes in Computer Science. Cham/Switzerland, Heidelberg, New York et al.: Springer International Publishing, pp. 308–324. ISBN: 978-3-319-08969-0. DOI: 10.1007/978-3-319-08970-6_20. URL: https://cakeml.org/itp14.pdf.

Kunčar, Ondřej and Andrei Popescu (2015). "A Consistent Foundation for Isabelle/HOL". In: *Interactive Theorem Proving: 6th International Conference, ITP 2015. Nanjing, China, August 24–27, 2015. Proceedings*. Ed. by Christian Urban and Xingyuan Zhang. Vol. 9236. Lecture Notes in Computer Science. Cham/Switzerland, Heidelberg, New York et al.: Springer International Publishing, pp. 234–252. ISBN: 978-3-319-22101-4. DOI: 10.1007/978-3-319-22102-1_16. URL: http://andreipopescu.uk/pdf/ITP2015.pdf.

Luo, Zhaohui (1994). *Computation and Reasoning: A Type Theory for Computer Science*. Oxford: Clarendon Press (Oxford University Press). ISBN: 0-19-853835-9.

Megill, Norman (2017a). *Metamath Home Page*. URL: http://us.metamath.org (visited on 04/10/2017).

— (2017b). *Metamath Proof Explorer Home Page*. URL: http://us.metamath.org/mpegif/mmset.html (visited on 04/10/2017).

Melham, Thomas F. (1993a). "The HOL Logic Extended with Quantification over Type Variables". In: *Higher Order Logic Theorem Proving and its Applications: Proceedings of the IFIP TC10/WG10.2 International Workshop on Higher Order Logic Theorem Proving and its Applications – HOL '92. Leuven, Belgium, 21–24 September 1992*. Ed. by Luc J. M. Claesen and Michael J. C. Gordon. Vol. A-20. IFIP Transactions A: Computer Science and Technology. Amsterdam et al.: North-Holland, pp. 3–17. ISBN: 0-444-89880-8. DOI: 10.1016/B978-0-444-89880-7.50007-3.

— (1993b). "The HOL Logic Extended with Quantification over Type Variables". In: *Formal Methods in System Design* 3.1–2, pp. 7–24. DOI: 10.1007/BF01383982. URL: http://www.cs.ox.ac.uk/tom.melham/pub/Melham-1994-HLE.pdf. Slightly extended version of [Melham, 1993a].

Nederpelt, R. P., J. H. Geuvers, and R. C. de Vrijer, eds. (1994). *Selected Papers on Automath*. Vol. 133. Studies in Logic and the Foundations of Mathematics. Amsterdam, Lausanne et al.: Elsevier. ISBN: 0-444-89822-0.

Norell, Ulf (2007). "Towards a practical programming language based on dependent type theory". PhD thesis. Department of Computer Science and Engineering, Chalmers University of Technology. URL: http://www.cse.chalmers.se/~ulfn/papers/thesis.pdf.

Norell, Ulf (2009). "Dependently Typed Programming in Agda". In: *Advanced Functional Programming: 6th International School, AFP 2008. Heijen, The Netherlands, May 19–24, 2008. Revised Lectures*. Ed. by Pieter Koopman, Rinus Plasmeijer, and Doaitse Swierstra. Vol. 5832. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer, pp. 230–266. ISBN: 978-3-642-04651-3. DOI: 10.1007/978-3-642-04652-0_5. URL: http://www.cse.chalmers.se/~ulfn/papers/afp08/tutorial.pdf.

Owre, Sam, John M. Rushby, and Natarajan Shankar (1992). "PVS: A Prototype Verification System". In: *Automated Deduction – CADE-11: 11th International Conference on Automated Deduction. Saratoga Springs, NY, USA, June 15–18, 1992. Proceedings*. Ed. by Deepak Kapur. Vol. 607. Lecture Notes in Artificial Intelligence. Subseries of Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer-Verlag, pp. 748–752. ISBN: 3-540-55602-8. DOI: 10.1007/3-540-55602-8_217. URL: http://www.csl.sri.com/papers/cade92-pvs/cade92-pvs.pdf.

Owre, Sam and Natarajan Shankar (1999). *The Formal Semantics of PVS*. Tech. rep. CSL-97-2R. August 1997, Revised March 1999. SRI International, Computer Science Laboratory. URL: http://pvs.csl.sri.com/papers/csl-97-2/csl-97-2.pdf.

Paulson, Lawrence C. (1989). *A Formulation of the Simple Theory of Types (for Isabelle)*. Tech. rep. UCAM-CL-TR-175. University of Cambridge, Computer Laboratory. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-175.pdf.

Pierce, Benjamin C. (2002). *Types and Programming Languages*. Cambridge/Massachusetts et al.: MIT Press. ISBN: 0-262-16209-1.

Shankar, Natarajan and Sam Owre (2000). "Principles and Pragmatics of Subtyping in PVS". In: *Recent Trends in Algebraic Development Techniques: 14th International Workshop, WADT'99. Château de Bonas, September 15–18, 1999. Selected Papers*. Ed. by Didier Bert, Christine Choppy, and Peter Mosses. Vol. 1827. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer-Verlag, pp. 37–52. ISBN: 3-540-67898-0. DOI: 10.1007/978-3-540-44616-3_3. URL: http://www.csl.sri.com/papers/wadt99/wadt99.pdf.

Slater, Barry Hartley (2016). "Epsilon Calculi". In: *The Internet Encyclopedia of Philosophy*. ISSN: 2161-0002. URL: http://www.iep.utm.edu/ep-calc/ (visited on 11/14/2016).

Slind, Konrad and Michael Norrish (2008). "A Brief Overview of HOL4". In: *Theorem Proving in Higher Order Logics: 21st International Conference, TPHOLs 2008. Montreal, Canada, August 18-21, 2008. Proceedings*. Ed. by Otmane Ait Mohamed, César Muñoz, and Sofiène Tahar. Vol. 5170. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, pp. 28–32. ISBN: 978-3-540-71065-3. DOI: 10.1007/978-3-540-71067-7_6. URL: https://ts.data61.csiro.au/publications/nicta_publication_full_1482.pdf.

Völker, Norbert (2007). "HOL2P – A System of Classical Higher Order Logic with Second Order Polymorphism". In: *Theorem Proving in Higher Order Logics: 20th International Conference, TPHOLs 2007. Kaiserslautern, Germany, September 10–13, 2007. Proceedings*. Ed. by Klaus Schneider and Jens Brandt. Vol. 4732. Lecture Notes in Computer Science. Berlin, Heidelberg et al.: Springer, pp. 334–351. ISBN: 978-3-540-74590-7. DOI: 10.1007/978-3-540-74591-4_25. URL: http://dces.essex.ac.uk/staff/norbert/papers/tphols07.pdf.

Wiedijk, Freek (2002). "A New Implementation of Automath". In: *Journal of Automated Reasoning* 29.3, pp. 365–387. ISSN: 1573-0670. DOI: 10.1023/A:1021983302516. URL: http://www.cs.ru.nl/~freek/aut/aut.pdf.

— (2003). "Comparing Mathematical Provers". In: *Mathematical Knowledge Management: Second International Conference, MKM 2003. Bertinoro, Italy, February 16-18, 2003. Proceedings*. Ed. by Andrea Asperti, Bruno Buchberger, and James Harold Davenport. Vol. 2594. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, pp. 188–202. ISBN: 3-540-00568-4. DOI: 10.1007/3-540-36469-2_15. URL: http://www.cs.ru.nl/F.Wiedijk/comparison/diffs.pdf.

— (2007). "The QED Manifesto Revisited". In: *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*. Ed. by Roman Matuszewski and Anna Zalewska. Vol. 10 (23). Studies in Logic, Grammar and Rhetoric. Białystok: University of Białystok, pp. 121–133. ISBN: 978-83-7431-128-1. URL: http://logika.uwb.edu.pl/studies/download.php?volid=23&artid=fw&format=PDF.

Wiedijk, Freek (2012). “Pollack-inconsistency”. In: *Electronic Notes in Theoretical Computer Science* 285, pp. 85–100. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2012.06.008. URL: http://www.sciencedirect.com/science/article/pii/S157106611200028X/pdf?md5=04ceb92a245b5bde8c4eca0610032293&pid=1-s2.0-S157106611200028X-main.pdf.