Lambda calculus with types*

Adrian Rezuş (Nijmegen)

December 16, 2014

§1 Types have been first invented about a century ago in order to avoid contradictions in logic and so-called foundations of mathematics. In the context of λ-calculus, the founders of modern logic dreamed of a "type-free" mathematical realm instead. The λ-calculus – also known as "calculus of λ-conversion" – has a rather intricated history.[1] The "pure" part of the calculus – with no constants and extensionality assumptions (also called $\lambda\beta$-calculus, nowadays) – was first incorporated explicitly in a "system of logic" designed by Alonzo Church in the late twenties (two papers in print, 1932–1933). As a peculiarity, Church's calculus was "strict" in the sense it allowed only strict abstractions of the form λx.e[x], where the free variable x occurs actually in the functional "body", e[x]. Church's full system of logic contained also logical constants (implication, ingredients allowing to express quantifiers, and definite descriptions) and additional logical axioms, but no type-distinctions. Church found a contradiction in his first formulation, which he repaired in the second paper. Even so, the type-free construction was shown to be inconsistent (the Kleene-Rosser "paradox" [Kleene & Rosser 1936]).[2] This pioneering work was not a total fail-

*A review of Henk Barendregt, Will Dekkers, Richard Statman *et al.*, *Lambda Calculus With Types*, Cambridge UP & ASL [July 31] 2013 [*Perspectives in Logic*], xxii + 833 pp. [ISBN: 978-0-521-766-142 (hbk)] [USD 90.00, resp. GBP 60.00]. Most historical references left undocumented below can be retrieved from A. Rezuş, *A Bibliography of Lambda-Calculi, Combinatory Logics and Related Topics*, Mathematisch Centrum [CWI] Amsterdam 1982. The readers are supposed to be familiar with the main author's treatise *The Lambda Calculus, Its Syntax and Semantics*, North Holland 1984 [SLFM 103]. **NB** This is a preprint. For the final version see *Studia Logica 103* (6), 2015, pp. 1319–1326. DOI: 10.1007/s11225-015-9634-z.

[1]We mention briefly only the landmarks. For details, see, e.g., Felice Cardone and J. Roger Hindley, *Lambda-calculus and combinators in the 20th century*, in D. M. Gabbay & J. Woods (eds.) *Hanbook of the History of Logic 5*, North Holland / Elsevier 2009, pp. 723–817.

[2]In order to distinguish the "strict" version of the calculus from the unrestricted one, the former is also referred to as "$\lambda(\beta)$I-calculus", in the literature, while the unrestricted version is called "$\lambda(\beta)$K-calculus". — As a matter of fact, the pure $\lambda(\beta\eta$K)-calculus was a also implicit in Frege's logic (actually arithmetic) inconsistent system of the *Grundgesetze der Arithmetik* [volume 1, issued in 1893], GA, for short. In GA, functional abstraction (not necessarily strict) was a primitive notion, but unlike in [Church 1932–1933], functional application appeared to be definable explicitly, using other logical notions, taken as primitives in the system. Although Frege's GA contained – at least implicitly – some loose type-distinctions (mostly hidden in his heavy, baroque use of meta-letters), Ernst Zermelo and, later, Bertrand Russell were able to show that GA allowed proving things like $0 = 1$, so that Frege abandoned subsequently his lifetime foundational (logico-mathematical) project. Worth mentioning is also the fact that, by the turn of the previous century (cca 1903–1905), the young Russell extracted the

ure, however, since, in the same year (1936), Church and one of his PhD students (J. Barkely Rosser) extracted the "pure" part of the calculus ($\lambda\beta$) and showed its "freedom from contradiction", by analyzing "conversion" in terms of a concept of "reduction" (intuitively: term-rewriting, formalizing "computation" so to speak). The Church-Rosser theorem, published in [Church & Rosser 1936] – also known as "confluence theorem" for the corresponding notion of reduction – says, roughly, that results of λ-computations do not depend on the order the computation-steps are performed; it implies consistency and gives thereby the birth-date certificate of the calculus: 1936.[3] A different formalism, intended to express the properties of functional evaluation (known nowadays as "combinatory logic") was formulated already a decade before Church's system, during the early twenties, by Moses Schönfinkel. Schönfinkel lectured occasionally on his findings at "Augusta", in Göttingen, by the end of 1920, and a detailed record of the lecture has been preserved and published (by Heinrich Behmann, of Göttingen), a few years later (1924). Although printed in a leading German mathematical journal, the Schönfinkel paper had no echo among logicians and / or mathematicians. So, ultimately, the "Schönfinkel combinators" have been discovered independently – once more – by Haskell B. Curry, by the end of the twenties. Curry showed also the equivalence of his version of the Schönfinkel "combinatory logic" (which he formulated as an equational theory) with the pure $\lambda\beta\eta$(K)-calculus.[4] On a different line of thought, Church, assisted by one of his former PhD students, Stephen C. Kleene, was able to connect, during the mid-thirties, the pure type-free λ-calculus (the "strict" version) with the concept of effective arithmetical computation, via an appropriate notion of λ-definability for arithmetical functions (cf. Kleene's PhD Diss. [Kleene 1935]); Kleene showed that λ-definability is equivalent to general recursiveness (à la Gödel-Herbrand). A bit later, Kleene extended the concept of λ-definability to constructive ordinals [Kleene 1938]. Around 1936–1937, Alan Turing invented a different concept of computability (nowadays: "Turing [-machine] computability") and proved the equivalence of his formal concept with the Church-Kleene λ-definability, thus general recursiveness [Turing 1937]. The further identification general recursiveness = λ-definability = Turing computability (formal concepts) with the (informal) idea of human [Church-Kleene] resp. machine [Turing] computability is known as the "Church-Turing Thesis", and makes up, in a sense, the conceptual basis of the modern applications of λ-calculus in computing science. On the logical side, Church pursued his foundational endeavors around "a type free system of logic and arithmetic" in lectures held in Princeton

"pure" λ-calculus (viz., the "extensional" $\lambda\beta\eta$K-calculus") from Frege's GA. (Cf. Russell's "Collected Works", volume 4 [ed. Alasdair Urquhart], 1994.)

[3]The Church-Rosser theorem implies the existence of rather trivial models – so-called "term-models" – via a popular technique, well-known from pioneering work (Lindenbaum, Łukasiewicz,Tarski, Wajsberg) on propositional logic. Extensionality assumptions – i.e. equivalents of the so called η-rule, leading to $\lambda\beta\eta$-calculus –, have been considered explicitly by H. B. Curry. Term-models have been studied by Barendregt, in his PhD Diss., Utrecht 1971. Genuinely mathematical models emerged in 1969-1971 (Dana S. Scott and Gordon Plotkin).

[4]Results published in Curry's Göttingen *Inauguraldissertation* [1930]. The combinatory counterpart of Church's "strict" λ-calculus has been studied by Rosser in his PhD Diss. [1935].

[1935–1936][5], an idea highly appreciated by Kurt Gödel, later on [1944].

§2 Already implicit in Frege's GA, type distinctions emerged first explicitly – in connection with λ-calculus and combinatory logic – on at least two distinct routes: the so-called "functionality theory" [FT] of Curry, in print since 1934-1936 (part of a larger foundational enterprise, called "illative logic"; roughly: logic based on combinators [Curry *et al.* 1958, 1972]), and the so-called "simple theory of types" of Church [1940]. The latter is a refinement of Russell's work incorporated in "Principia Mathematica".[6] Putting aside Curry and some of his PhD students, the corresponding basic type-theories – where the only type-constructor is a binary primitive, $\rightarrow$, yielding complex types, A$\rightarrow$B, from atoms (roughly: implications, in logic, or function spaces, in ordinary mathematical practice) – have been extracted and studied in depth separately only several decades later, since the early seventies, more or less. This allowed making conceptual distinctions, more accurate comparisons and, above all, prompted the study of extensions in several directions. The book under review covers the "basic" theories, under different typing "styles" (Part I) – as briefly described below –, and two categories of extensions, including recursive types, in Part II, and so-called "intersection types", in Part III. Practically, the book consists of three distinct monographs. Roughly speaking, only Part I (cca 375 pp. in print, about half of the book, and rather loosely organized) has a direct bearing to logic as such. Insisting mainly on the "basic" theories, the authors have omitted a vast amount of material that has emerged during the last decades under the general heading "typed λ-calculus".[7] The most important omission is the so-called "Curry-Howard isomorphism" [CH], a subject of main concern in modern proof theory, dealing with typed λ-calculi where types are viewed as formulas / propositions, and terms are interpreted as proofs or "witnesses" for provable formulas / propositions.[8] Also omitted are the "higher-order", "de-

[5]See A. Church, *Mathematical Logic*, Princeton Lectures, October 1935 – January 1936 [Lecture Notes by F. A. Ficken, H. G. Landau, H. Ruja, R. R. Singleton, N. E. Steenrod, J. H. Sweer, and F. J. Weyl, typescript, 113 pp., copy in Princeton University Library, at Firestone]. The lecture notes have been revised, drastically condensed, and published eventually as *Calculi of Lambda-conversion*, Princeton UP 1941. [The original lectures remained unpublished thus far.] The corresponding formalism, based on an extension of the "strict" $\lambda\beta$-calculus – called "[strict] $\lambda\beta\delta$-calculus" – is consistent and exemplifies the idea of "ordinal logic", a concept distilled later by Turing [PhD Diss. 1939]. Although the resulting, type-free logic had a classical flavor at the propositional level, it exploited the full power of λ-definability [in strict $\lambda\beta\delta$] in order to *formalize* a transfinite sequence of universal quantifiers – and formal implications – in "ordinal progression" below the first non-constructive ordinal, in the sense of Church and Kleene, where *formalization* is viewed as a transfinite process. Later, Church abandoned the idea of building logic on a type-free basis and recomended his simple type theory [1940] or a formalized version of **ZF**, instead. A full record of Church's type-free ordinal logic can be found in the reviewer's PhD Diss. *Lambda-conversion and Logic*, Utrecht 1981.

[6]Curry has also proposed, already in the early forties, a "theory of generalized functionality" [GFT], intended to accommodate quantifiers in an "illative" setting (this is an anticipation of the so-called "dependent" typing studied by Nicolaas G. de Bruijn and Per Martin-Löf during the late sixties resp. the early seventies). In fact, Curry adapted, in his GFT, some of Church's ideas (of the Princeton Lectures 1935–1936) to his own foundational project.

[7]For some of the issues left out, see H. Barendregt, *Lambda calculi with types*, S. Abramsky *et al.* (eds.), *Handbook of Logic in Computer Science 2*, Oxford UP 1992, pp. 117–309.

[8]The first application of CH – for the "basic" theory – to a practical logic problem (axiom-

pendent", and "inductive" types, as well as closely related category theoretic developments[9]. The distinction between a Curry *vs* a Church typing "discipline", yields, essentially, two "basic" type theories: one with a "loose" typing, à la Curry, and a second one, with a "rigid" typing, so to speak, à la Church. The main difference consists of the fact that, under a Church typing, variables (either bound or free) retain their "rigid" types, while this is not the case under a Curry typing, where the bound variables loose their type-decorations.[10] Even though the "rigid" typing (Church) is conceptually sound and convenient, the Curry loose typing "style" is, in certain respects, more flexible, since it allows also considering type-systems equipped with an effectively generated notion of type-isomorhism, including, e.g., definitions by recursion (on types) and, possibly, additional type constructors as, e.g., "intersection" (as distinct from the usual product-construction which makes sense in logic and category theory).[11]

§3 The "basic" theories contain only "simple" types built up inductively from atoms by a single type-forming operation, written, for convenience, as an infix arrow $\to$ (so that complex types are only of the form $A \to B$) – rely on a type-assignment given by introduction and elimination rules for $\to$, similar (in fact, formally isomorphic) to the Jaśkowski-Genzten "natural deduction" [ND] rules for minimal implication. As long as we do not have additional – e.g., equational – constraints on types, the game to play looks rather trivial, and the pure λ-calculus serves as a kind of "witnessing" mechanism for provability in the pure implicational fragment of Johansson's *Minimalkalkül* [1936] (a subsystem of intuitionistic logic).[12] The difference between typing "styles" (Church *vs* Curry) appears in the syntactic structure of the terms: under a Curry typing we have less information as long as the terms contain λ-abstractors and we loose

atizability) is due, somewhat *avant la lettre*, to Alfred Tarski (cca 1925 [Rezuş 1982]). For a comprehensive survey of CH to date, see Morten H. Sørensen, and Paweł Urzyczyn, *Lectures on the Curry-Howard Isomorphism*, Elsevier 2006 [SLFM 149] (xiv + 442 pp.).

[9] Cf., e.g., Bart Jacobs, *Categorical Logic and Type Theory*, Elsevier 1999 [SLFM 141] (xviii + 760 pp.), and Vladimir Voevodsky *et al.*, *Homotopy Type Theory: Univalent Foundations of Mathematics*, The Univalent Foundations Program, IAS Princeton 2013 (x + 470 pp.).

[10] Characteristically, the Curry-style typed combinator (closed term) $\mathbf{I} := \lambda x.x$ can have any type in the class $A \to A$, where A ranges over type(expression)s, while, under the Church-style, we have a family of **I**-combinators $\mathbf{I}[A] := \lambda x^A.x^A$ where the type of $\mathbf{I}[A]$ is $A \to A$. A third – slightly different – "style" of "rigid" typing occurred, since the late sixities, in the work of N. G. de Bruijn and his collaborators at the Eindhoven University of Technology (The Netherlands) on Automath (short for "automated mathematics"), a family of formal systems meant to formalize and verify actual mathematical proofs automatically (on a computer). For all practical purposes, the de Bruijn typing "style" is just a typographical variation on the Curch typing "style", as adapted to "dependent" typing. With the example above, in the de Bruijn / Automath "style", we could write, more conveniently, $\lambda x{:}A.x$ for $\mathbf{I}[A]$. The difference (Church *vs* de Bruijn) appears only in the way of referring to open terms, i.e., in the way one intends to manipulate so-called [Automath] "contexts". A description of Automath (syntax and model theory) can be found in the reviewer's *Abstract Automath*, Mathematisch Centrum [CWI] Amsterdam 1983, and in H. Barendregt, and A. Rezuş, *Semantics for Classical Automath and related systems* [*Information and Control 59*, 1983, pp. 127–147].

[11] Of course, with such identifications, the proof-theoretic interpretation of the resulting λ-formalisms is ruined and it is unlikely proof-theorists would want to look into such extensions.

[12] There are, in fact, many other interpretations of the resulting formalism: we have plenty of models around, indeed, and there is hardly anything specific to intuitionism here.

the "unicity of typing", insured automatically under a Church typing régime. Formally, a type-assignment is taken to be relative to a context Γ (i.e., a sequence of pairwise distinct typed variables; assumption set or Curry "basis"), so that the associated ND-system generates statements $\Gamma \vdash a[\vec{x}] : A$, where $\vec{x}$ is a sequence of free variables occurring in Γ.[13] The first Part of the book is concerned with various properties of the typed terms under different typing "styles". Characteristic problems addressed in each case concern (strong) normalization, type-checking, typability, type-finding ("type-reconstruction" in the text), finding – and counting – inhabitants of a specific form, decidability questions, etc. Other topics, as e.g., λ-definability, are imported from the type-free calculus.[14] Model theory (essentially term-models) is touched upon in section **I.3**, dedicated to "Tools". Among the extensions obtained by adding specific (term-) constants to the "basic" theory worth mentioning are [**1**] a system with discriminators (a $\lambda\delta$-calculus, meant to handle a "higher order logic" [HOL] based on classical logic), [**2**] a system with projections and (surjective) pairing, called λ_{SP}[15], and its connection to cartesian closed monoids [CCMs, à la Joachim Lambek and Dana Scott, cca 1980], [**3**] Gödel's [1959] system **T** with higher order primitive recursion, [**4**] Spector's [1962] extension of **T** with bar recursion, and [**5**] Platek's [1966] extension of **T** with fixedpoint recursion.[16] Proof theory proper (here, more or less, Curry-Howard for minimal implication) is briefly discussed in section **I.6.3** (contributed by Silvia Ghilezan), under "Applications". Summing up, the first Part of the book makes up a large cocktail of otherwise useful, interesting data and mathematical results, reflecting the specific research interests of the authors (including some contributors), rather than a systematic conceptual treatement in a well-structured mathematical discipline.

[13]As regards the FT, Curry was aware, already during the thirties, of the possibility of interpreting his "functional characters" (our types) as logic implicational formulas, whereupon the fact that a term a has type A relative to a context Γ, $\Gamma \vdash a : A$, would mean "a is a proof of A [relative to Γ] " (or else, better: "a is a witness for A"; the alternative reading "a inhabits A" supposed to be neutral, suggests the fact that types might look like containers). The proof-theoretical interpretation has been extended to (propositional and first-order) intuitionistic logic, by Herbert Läuchli (1965, 1970) and William Howard (1969, in print 1980), to intuitionistic logic with propositional and so-called second-order quantifiers by Dag Prawitz (1965, 1971), Jean-Yves Girard (1971), and John Reynolds (1974), as well as to classical logic with propositional, first- and second-order quantifiers, and to some "intensional" logics (relevant, modal [à la Lewis], etc.) by the reviewer (so-called $\lambda\gamma$-calculi), during the mid-eigthies, building upon earlier work of Kolmogorov (1925), Glivenko (1928–1929) and Prawitz (1965). Similar extensions, in the classical direction, have been obtained, later, by Matthias Felleisen, Timothy Griffin, Chetan Murthy, Michel Parigot, Morten Sørensen, Jakob Rehof, etc.

[14]The corresponding concept is much weaker in a type-theoretical setting; this motivates some extensions, as those studied in the sections **I.5.3**–**I.5.5**, for instance.

[15]No direct relation to the "extended" λ-calculus, known as "λ-calculus with surjective pairing", $\lambda\pi$, in the literature (although some subtle connections might eventually emerge). Putting aside the fact that λ_{SP} has a single atomic type, the system is not compatible with Curry-Howard. On the other hand, like in the case of the $\lambda\pi$-calculus (typed "intuitionistically"), the notion of reduction associated to λ_{SP} is confluent and strongly normalizable. Somewhat surprisingly, however, the "surjective" theory λ_{SP} is Post-complete, **I.5.2**, pp. 272–273. A minor typo occurs in Proposition **5.2.10**, page 250: read "notion" for "notation".

[16]This section [**I.5**] of the book – which logicians would likely appreciate – is contributed by Marc Bezem, and makes up a booklet on its own (it occupies about 80 pp. in print).

In contrast, the Part II and, above all, the Part III are focussed on more specific topics, and, consequently, better organized. Part II is concerned with "basic" theories where [**1**] the types are equipped with specific isomorphisms (i.e., they are taken modulo a set of equations) or [**2**] they are generated by using an adddi-tional μ-abstraction operator (on types), as, e.g., in $\mu\alpha.[\alpha\to A]$, used to express solutions of recursive type-equations (whence the name "recursive typing"). Following a suggestion from Dana Scott [cca 1975], both issues are described in the same setting, by using type-agebras. Part III concerns a rather specific subject, invented around 1980 by Mariangiola Dezani [-Ciancaglini], Mario Coppo and Patrick Sallé, as an extension of Curry's FT, in view of finding a "typing discipline" for (strongly) normalizing terms (in type-free λ-calculus). Similar work in this direction has been done by Garrel Pottinger [1980]. The main additional ingredient is a new type-constructor $\cap$ – called type-intersection –, subjected to rules of introduction and elimination similar to the usual ND-rules for AND-introduction resp. AND-elimination in (minimal, intuitionistic, and classical) logic. The analogy with Curry-Howard stops here, because one does not have the underlying, richer terms-structure at hand (no primitive pairing – i.e., pairs and projections – constructs, for terms, like in the "extended" λ-calculus [= surjective $\lambda\pi$-calculus], say[17]). The type-structure is equipped, instead, with a preorder and, oft, also with a top-element U. Colloquially, the int-elim rules for $\cap$ state that if a term a inhabits both A and B then a also inhabits their intersection $A\cap B$ ($\cap$-introduction) and conversely ($\cap$-elimination). So, among other things, a term might get plenty of types – including the "universal" type U –, and every term is tyable. Furthermore, one can have additional atomic types – as, e.g., 0 and 1 –, and additional postulates (axioms and axiom-schemes resp. rule-schemes) governing the type-preorder, the type-equality, and the specific atoms, including the top-element, so that we end up with about a dozen of distinct "intersection type theories"[18]. As one might guess, a would-be attempt to associate such type-theories to a Curry-Howard interpretation is problematic, even if the "universal" type U is left aside. The interest in such extensions is, however, elsewhere, viz. in [**a**] a characterization of [strongly] normalizable terms and [**b**] a syntactic (type-theoretic) analysis of λ-calculus models.[19] The extant bibliography of this specific subject [Part III] is rather vast at the time of writing, and still vividly growing. Nearly each section of the book is copiously augmented by Exercises concerning issues not detailed in the main text, and, oft, even current research. Altogether, the book contains a wealth of useful mathematical information – presented in an elegant, clear style –, and one might expect that most graduate students and researchers in theoretical computer science, as well as many logicians interested in the applications of λ-calculus would find it worth keeping around and perusing, as an inspiring research tool.

[17] The would-be definable pairings in pure λ-calculus do not get the intended typing under the Curry type-assignment, anyway.

[18] Actually 13, labelled historically in the book by the names of their proponents. Notably, the presence of the universal type is not mandatory (three systems are top-less).

[19] The second concern stems from work of Henk Barendregt, Mario Coppo, and Mariangola Dezani [1983 and later] on so-called "filter models".