§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. The “pure” part of the calculus – with no constants and extensionality assumptions (also called λβ-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]). This pioneering work was not a total fail-
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 \(\lambda\)-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 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 \(\lambda\)-calculus (the “strict” version) with the concept of effective arithmetical computation, via an appropriate notion of \(\lambda\)-definability for arithmetical functions (cf. Kleene’s PhD Diss. [Kleene 1935]); Kleene showed that \(\lambda\)-definability is equivalent to general recursiveness (à la Gödel-Herbrand). A bit later, Kleene extended the concept of \(\lambda\)-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 \(\lambda\)-definability, thus general recursiveness [Turing 1937]. The further identification general recursiveness = \(\lambda\)-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 \(\lambda\)-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.

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 \(\eta\)-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” \(\lambda\)-calculus has been studied by Rosser in his PhD Diss. [1935].
§2 Already implicit in Frege’s GA, type distinctions emerged first explicitly – in connection with \(\lambda\)-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 \(\lambda\)-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 \(\lambda\)-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-
dependent”, and “inductive” types, as well as closely related category theoretic developments. 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 lose their type-decorations. 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).

§ 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 → (so that complex types are only of the form A → B) – rely on a type-assignment given by introduction and elimination rules for →, 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).

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
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 Γ ⊢ a[⃗x] : A, where ⃗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 λδ-calculus, meant to handle a “higher order logic” [HOL] based on classical logic), [2] a system with projections and (surjective) pairing, called λSP\(^{15}\), 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 treatment 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, Γ ⊢ 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 λγ-calculi), during the mid-eighties, 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”, λπ, in the literature (although some subtle connections might eventually emerge). Putting aside the fact that λSP has a single atomic type, the system is not compatible with Curry-Howard. On the other hand, like in the case of the λπ-calculus (typed “intuitionistically”), the notion of reduction associated to λSP is confluent and strongly normalizable. Somewhat surprisingly, however, the “surjective” theory λSP is Post-complete, I.5.2, pp. 272–273. A minor typo occurs in Proposition 5.2.10, page 250: read “notation” 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 additional \( \mu \)-abstraction operator (on types), as, e.g., in \( \mu \alpha . (\alpha \rightarrow A) \), 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 \( \lambda \)-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” \( \lambda \)-calculus [= surjective \( \lambda \pi \)-calculus], say17). 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 inherits both A and B then a also inherits 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 \( \lambda \)-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 \( \lambda \)-calculus would find it worth keeping around and perusing, as an inspiring research tool.

---

17The would-be definable pairings in pure \( \lambda \)-calculus do not get the intended typing under the Curry type-assignment, anyway.

18Actually 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).

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

---