YACC: Yet Another Church Calculus (Abstract)

Franco Barbanera[1], Mariangiola Dezani-Ciancaglini[2],
Ugo de' Liguoro[2], and Betti Venneri[3]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
[2] Dipartimento di Informatica, Università di Torino, Italy
[3] Dipartimento di Statistica, Informatica, Applicazioni, Università di Firenze, Italy

There are essentially two approaches to type theory for λ-calculus, introduced by Curry [9] and Church [7], respectively; Barendregt [5] dubbed them *à la Curry* and *à la Church*. The characteristic of type systems à la Church is that if a term is well-typed, then the type derivation that justifies the typing can be read out of the term itself; in particular, types decorate the binding of variables, like in $\lambda x : T.x$, where we also deduce that the type of the term is $T \to T$, which is also its unique (simple) type. In the case of type assignment systems à la Curry, types are assigned to untyped terms, and a term either has no type, or it has infinitely many types, all instances of the same type scheme, e.g. $\lambda x.x$ has $T \to T$, for all the (infinitely many) choices of T. Still, typable terms encode the structure of the typing derivation (if any), and because of this, typability of terms is decidable.

The intersection type system in [8] is an extension of Curry's one where there is the new type constructor $\cap$ and the universal type ω plus the rules [4]

$$\frac{\Gamma \vdash_\cap P : \mu \quad \Gamma \vdash_\cap P : \nu}{\Gamma \vdash_\cap P : \mu \cap \nu}\,[\cap I] \qquad \frac{}{\vdash_\cap P : \omega}\,[\omega]$$

which destroys the correspondence of the subject P in the conclusion with the structure of the typing derivation. Aiming at restoring such correspondence, as e.g. in [12], one may manage to extend the term syntax to keep track especially of the usages of Rule $[\cap I]$ in a typing derivation of the term, while requiring that the subjects in the premises of the rule are (essentially) the same. Indeed, if we understand that the subject of a typing statement encodes a derivation, then this rule is *proof-functional*, having a conclusion depending not only on the premises but also on their derivations, as already noticed in [13] and further investigated in [1,3].

The TIC calculus and type system. In [2], of which this short note is an abstract, we follow a different approach by encoding the information on type derivations in the types of terms, and in particular, the types decorating the occurrences of free and bound variables in a new calculus à la Church, which we dub *Typed Intersection λ-calculus*, shortly TIC. We distinguish among three possible usages of the intersection, corresponding to three distinct type constructors. The first one is when in the typing derivation all occurrences of the same variable have the same type and there is no use of Rule $[\cap I]$, e.g. in

[4] Γ is a typing environment associating term variables to types: $\Gamma ::= \emptyset \mid \Gamma, x : \mu$.

$$\vdash_\cap \lambda x.\lambda y.xy : (\phi \cap \psi \to \phi) \to \phi \cap \psi \to \phi$$

To this term it corresponds the TIC term

$$\lambda x : \phi \wedge \psi \to \phi.\lambda y : \phi \wedge \psi.x^{\phi\wedge\psi\to\phi} y^{\phi\wedge\psi}$$

which has type $(\phi \wedge \psi \to \phi) \to \phi \wedge \psi \to \phi$ in TIC, where $\cap$ has been replaced by $\wedge$.

The second one is when the same occurrence of a variable in the subject has different types in the sub-derivations that have been merged by Rule $[\cap I]$, e.g.

$$\frac{\vdash_\cap \lambda x.\lambda y.xy : (\phi \to \phi) \to \phi \to \phi \qquad \vdash_\cap \lambda x.\lambda y.xy : ((\psi \to \psi) \to \psi \to \psi}{\vdash_\cap \lambda x.\lambda y.xy : (\phi \to \phi) \to \phi \to \phi) \cap ((\psi \to \psi) \to \psi \to \psi)}\,[\cap I]$$

which is represented by the TIC term

$$\lambda x : (\phi \to \phi) \sqcap (\psi \to \psi).\lambda y : \phi \sqcap \psi.x^{(\phi\to\phi)\wedge(\psi\to\psi)} y^{\phi\wedge\psi}$$

which has type $((\phi \to \phi) \to \phi \to \phi) \wedge ((\psi \to \psi) \to \psi \to \psi)$. Notice that in the decoration of the variables in λx and λy the distinct types of these variables in the derivation are connected by $\sqcap$, while they have the intersections of these types in the body using $\wedge$.

The third case, which is characteristic of intersection-type systems, is when distinct occurrences of the same variable have different types in the derivation, like in

$$\frac{\dfrac{x : (\phi \to \psi) \cap \phi \vdash_\cap x : \phi \to \psi \qquad x : (\phi \to \psi) \cap \phi \vdash_\cap x : \phi}{x : (\phi \to \psi) \cap \phi \vdash_\cap xx : \psi}\,[\to E]}{\vdash_\cap \lambda x.xx : (\phi \to \psi) \cap \phi \;\to\; \psi}\,[\to I]$$

corresponding to the TIC term

$$\lambda x : (\phi \to \psi)\,\&\,\phi.x^{\phi\to\psi} x^{\phi}$$

of type $(\phi \to \psi)\,\&\,\phi \;\to\; \psi$ where the third (and last) variant of the intersection type constructor is $\&$.

Pseudo-terms of TIC are defined by the grammar

$$M ::= x^\sigma \mid \lambda x : \kappa.M \mid MM \mid \Omega$$

where the types σ and κ belong to "main types" and "conjunctive types" kinds below:

(base types)	$\alpha ::= \varphi \mid \theta \to \alpha$	(main types)	$\sigma, \tau ::= \alpha \mid \sigma \wedge \sigma$
(relevant left types)	$\vartheta ::= \sigma \mid \vartheta\,\&\,\vartheta$	(left types)	$\theta ::= \omega \mid \vartheta$
	(conjunctive types)	$\kappa ::= \theta \mid \kappa \sqcap \kappa$	

A TIC term is a pseudo-term that is the subject of a typing judgment derivable in the system in Figure 1. The rules in the TIC type-system make use of some auxiliary notions which we briefly explain (see [2] for detailed definitions). In Rule $[\to I]$ the mapping $\iota(M, x)$ collects from left to right all the types of the free occurrences of x in M holding the $\&$ conjunction of such types, where $\&$ is neither idempotent nor commutative, but just associative. We define $\iota(M, x) = \omega$

$$\frac{}{\vdash x^{\alpha} : \alpha}\,[\textsc{Var}] \qquad \frac{}{\vdash \Omega : \omega}\,[\omega] \qquad \frac{\vdash M : \alpha \quad \iota(M,x) = \theta}{\vdash \lambda x : \theta.M : \theta \to \alpha}\,[\to I] \qquad \frac{\vdash M : \omega \to \alpha}{\vdash M\Omega : \alpha}\,[\to E\omega]$$

$$\frac{\vdash M : \vartheta \to \alpha \quad \vdash N : \sigma \quad \vartheta \ltimes \sigma}{\vdash MN : \alpha}\,[\to E] \qquad \frac{\vdash M : \sigma \quad \vdash N : \tau}{\vdash M \bigwedge N : \sigma \wedge \tau}\,[\wedge I]$$

Fig. 1: TIC typing rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash x^{\beta} : \beta \quad \vdash x^{\alpha} : \alpha}{\vdash x^{\beta}x^{\alpha} : \omega \to \alpha} \qquad \vdash \Omega : \omega}{\vdash x^{\beta}x^{\alpha}\Omega : \alpha}}{\vdash M : \beta \& \alpha \to \alpha} \qquad \dfrac{\dfrac{\vdash y^{\alpha} : \alpha}{\vdash \lambda z : \omega.y^{\alpha} : \omega \to \alpha} \qquad \dfrac{\vdash y^{\phi} : \phi}{\vdash \lambda z : \omega.y^{\phi} : \omega \to \phi}}{\vdash N_1 : \beta \qquad \vdash N_2 : \alpha}}{\vdash N : \beta \wedge \alpha}}{\vdash MN : \alpha}$$

where

$$\begin{array}{ll} \alpha = \phi \to \omega \to \phi & \beta = \alpha \to \omega \to \alpha \\ N_1 = \lambda y : \alpha.\lambda z : \omega.y^{\alpha} & N_2 = \lambda y : \phi.\lambda z : \omega.y^{\phi} \\ M = \lambda x : \beta \,\&\, \alpha.x^{\beta}x^{\alpha}\Omega & N = \lambda y : \alpha \sqcap \phi.\lambda z : \omega \sqcap \omega.y^{\alpha \wedge \phi} \end{array}$$

Fig. 2: Example of typing.

if x does not occur in M. The relation $\vartheta \ltimes \sigma$ holds roughly when σ can be obtained from ϑ by replacing top level $\&$ with $\wedge$.

Finally, and more importantly, the pre-term $M \bigwedge N$ is well defined if the erasures of M and N coincide (up to renaming of bound variables). The more interesting clauses defining $\bigwedge$ are:

$$x^{\sigma} \bigwedge x^{\tau} = x^{\sigma \wedge \tau} \qquad (\lambda x : \kappa.M) \bigwedge (\lambda x : \kappa'.N) = \lambda x : \kappa \sqcap \kappa'.M \bigwedge N$$

that is the meta-operator $\bigwedge$ has the effect of introducing the $\wedge$-conjunction of possibly distinct types in the decoration of variable occurrences, while the types decorating variables in λ-bindings are the $\sqcap$-conjunction of their (possibly distinct) types in the respective terms. An example of TIC derivation is in Figure 2.

Reduction and main results. As a matter of fact, the meta-operator $\bigwedge$ is some sort of partial pairing, such that we can define its projections (see [2, Lemma 2(2)]):

If $\vdash M : \tau$ *and* $\tau \backsimeq \tau_1 \wedge \ldots \wedge \tau_n$ *with* $n \geq 2$,
then there is $\widetilde{\Pi_i^{\tau}}$ *such that* $\vdash \widetilde{\Pi_i^{\tau}}(M) : \tau_i$ *for each* i $(1 \leq i \leq n)$

where $\backsimeq$ is the equivalence relation induced by considering $\wedge$ associative. With this we can define reduction as the compatible closure of the rules:

$[\beta\&]$ $\quad (\lambda x : \&_{i \in I}\sigma_i.M)N \longrightarrow M[x^{\sigma_i} := N_i]_{i \in I}$
where $\sigma = type(N) \backsimeq \wedge_{i \in I}\sigma_i$ and $N_i = \widetilde{\Pi_i^{\sigma}}(N)$ for each $i \in I$

$[\beta\omega]$ $\quad (\lambda x : \omega.M)\Omega \longrightarrow M$

$$[\beta\sqcap] \qquad \frac{(\lambda x : \kappa_j.M_j)N_j \longrightarrow M'_j \quad j = 1,2}{(\lambda x : \kappa_1 \sqcap \kappa_2.M_1 \bigwedge M_2)(N_1 \bigwedge N_2) \longrightarrow M'_1 \bigwedge M'_2}$$

For example, by taking the TIC term of Figure 2,

$$\begin{aligned}
&(\lambda x : (\alpha \to \omega \to \alpha)\,\&\,\alpha).x^{\alpha\to\omega\to\alpha}x^{\alpha}\Omega)(\lambda y : \alpha \sqcap \phi.\lambda z : \omega \sqcap \omega.y^{\alpha\wedge\phi})\\
\longrightarrow_{\beta\&}\ &(\lambda y : \alpha.\lambda z : \omega.y^{\alpha})(\lambda y : \phi.\lambda z : \omega.y^{\phi})\Omega\\
\longrightarrow_{\beta\&}\ &(\lambda z : \omega.\lambda y : \phi.\lambda z : \omega.y^{\phi})\Omega\\
\longrightarrow_{\beta\omega}\ &\lambda y : \phi.\lambda z : \omega.y^{\phi}
\end{aligned}$$

where $\alpha = \phi \to \omega \to \phi$ and the arrows are decorated with the applied rule. The following is, instead, a simple application of Rule $[\beta\sqcap]$

$$\frac{(\lambda x : \phi.x^{\phi})y^{\phi} \longrightarrow y^{\phi} \quad (\lambda x : \psi.x^{\psi})y^{\psi} \longrightarrow y^{\psi}}{(\lambda x : \phi \sqcap \psi.x^{\phi\wedge\psi})y^{\phi\wedge\psi} \longrightarrow y^{\phi\wedge\psi}}\,[\beta\sqcap]$$

We list below the main results from [2].

Theorem 1. *If $\vdash M : \tau$ and $M \longrightarrow N$, then $\vdash N : \tau$.*

Define $\|M\|$ as the pure λ-term obtained from M by erasing all types and replacing Ω with $(\lambda x.xx)(\lambda x.xx)$.

Theorem 2. *If $M \longrightarrow N$, then $\|M\| \longrightarrow_{\beta} \|N\|$.*

Theorem 3. *If $\|M\| \longrightarrow_{\beta} P$, then there is a TIC term N such that $\|N\| = P$ and either $M \longrightarrow N$ or $N = M$.*

Theorem 4. *A TIC term M has a head normal form iff type$(M) \neq \omega$.*

Related works. In the literature there are many proposals for typed λ-calculi à la Church with intersection types. We only recall here some of the most significant ones.

The calculus of [15] has branching types and types with quantification over type selection parameters. There $\vdash_{\cap} \lambda x.x : (\phi \to \phi) \cap (\psi \to \psi)$ is represented by $\Lambda(\mathsf{join}\{i = \star, j = \star\}).\lambda x^{\{i=\phi,j=\psi\}}.x^{\{i=\phi,j=\psi\}}$, where $\mathsf{join}\{i = \star, j = \star\}$ is a branching type. Branching types avoid duplication, since they "squash together" the premises of the intersection introduction typing rule.

In the calculus of [12] typing depends on an "imperative-like" formulation of context, assigning types to term-variables at a given mark/location, and on a new notion of store, that remembers, trough modalities, the associations between marks and types. For example, $\vdash_{\cap} \lambda x.x : (\phi \to \phi) \cap (\psi \to \psi)$ can be written as the term $(\lambda x : 0.x)@(\lambda 0 : \phi.0) \cap (\lambda 0 : \psi.0)$ where 0 is a mark, and the modality for the subterm $\lambda x : 0.x$ is $(\lambda 0 : \phi.0) \cap (\lambda 0 : \psi.0)$.

A parallel term constructor $|$ representing the intersection is instead introduced in [6]. This allows to obtain, for any type derivation in the system $\vdash_{\cap}$, a corresponding type decorated term. For example, the term corresponding to $\vdash_{\cap} \lambda x.x : (\phi \to \phi) \cap (\psi \to \psi)$ is $\lambda x^{\phi}.x^{\phi}|\lambda y^{\psi}.y^{\psi}$.

A new and interesting solution is represented by *dimensional intersection type calculi* [10,11]. The typing judgements are of the shape $\Gamma \vdash \mathtt{P} : \mu$, where $\mathtt{P}$ is an *elaboration*, i.e., a λ-term where each sub-term is decorated with the set of types assigned to it. These decorations are enclosed between angle brackets. For

example, the judgement in dimensional intersection type calculi corresponding to $\vdash_{\cap} \lambda x.x : (\phi \to \phi) \cap (\psi \to \psi)$ is

$$\vdash_{<>} (\lambda x.x\langle\phi, \psi\rangle)\langle\phi \to \phi, \psi \to \psi\rangle : (\phi \to \phi) \cap (\psi \to \psi)$$

Our proposal is worthy of inclusion in the above scenario. Differently from the mentioned calculi, TIC has more type constructors and a less permissive type syntax than the type assignment system in [4]. Besides, it lacks the subsumption rule, as well as weakening and contraction. This implies that we do not have the isomorphism between typing à la Curry and à la Church, which is a feature of the calculi in [15,12,6,10,11]. However, due to the presence of the universal type ω, we characterise the λ-terms having head normal forms, while the calculi in [15,12,6,10,11] characterise strongly normalising λ-terms.

We split the intersection into distinct constructors taking inspiration from [14], where two distinct conjunction constructors reflect two possible shapes of derivations. The synchronous conjunction can be used only among equivalent deductions, while the asynchronous conjunction can be used among arbitrary derivations. This is the basis for building a logical system which is a proof theoretical justification of intersection type assignment systems.

The use of different constructors for abstracted variables and terms in our type system is reminiscent of the different linear logic conjunction operators used both on the left and on the right hand side of the linear logic entailment. This could be the starting point of an investigation on the notion of strong conjunction for linear logic and hence, via realisability, on a notion of "intersection linear types".

References

1. Alessi, F., Barbanera, F.: Strong conjunction and intersection types. In: Tarlecki, A. (ed.) MFCS. LNCS, vol. 520, pp. 64–73. Springer (1991). https://doi.org/10.1007/3-540-54345-7_49
2. Barbanera, F., Dezani-Ciancaglini, M., de'Liguoro, U., Venneri, B.: Yacc: Yet another church calculus. In: Capretta, V., Krebbers, R., Wiedijk, F. (eds.) Logics and Type Systems in Theory and Practice. LNCS, vol. 14560, pp. 17–35. Springer (2024)
3. Barbanera, F., Martini, S.: Proof-functional connectives and realizability. Archive of Mathematical Logic **33**(3), 189–211 (1994). https://doi.org/10.1007/BF01203032
4. Barendregt, H., Coppo, M., Dezani-Ciancaglini, M.: A filter lambda model and the completeness of type assignment. The Journal of Symbolic Logic **48**(4), 931–940 (1983). https://doi.org/10.2307/2273659
5. Barendregt, H.: Lambda calculi with types. In: Abramsky, S., Gabbay, D.M., Maibaum, T. (eds.) Handbook of Logic in Computer Science, Volume 2: Background: Computational Structures, pp. 117–309. Oxford University Press (1992). https://doi.org/10.1093/oso/9780198537618.003.0002
6. Bono, V., Venneri, B., Bettini, L.: A typed lambda calculus with intersection types. Theoretical Computer Science **398**(1-3), 95–113 (2008). https://doi.org/10.1016/j.tcs.2008.01.046

7. Church, A.: A formulation of the simple theory of types. The Journal of Symbolic Logic **5**(2), 56–68 (1940). https://doi.org/10.2307/2266170
8. Coppo, M., Dezani-Ciancaglini, M.: An extension of the basic functionality theory for the λ-calculus. Notre Dame Journal of Formal Logic **21**(4), 685–693 (1980). https://doi.org/10.1305/ndjfl/1093883253
9. Curry, H.B.: Functionality in combinatory logic. Proceedings of the National Academy of Science of the USA **20**, 584–590 (1934). https://doi.org/10.1073/pnas.20.11.584
10. Dudenhefner, A., Rehof, J.: Intersection type calculi of bounded dimension. In: Castagna, G., Gordon, A.D. (eds.) POPL. pp. 653–665. ACM (2017). https://doi.org/10.1145/3009837.3009862
11. Dudenhefner, A., Rehof, J.: Typability in bounded dimension. In: Ouaknine, J. (ed.) LICS. pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005127
12. Liquori, L., Ronchi Della Rocca, S.: Intersection-types à la Church. Information and Computation **205**(9), 1371–1386 (2007). https://doi.org/10.1016/j.ic.2007.03.005
13. Lopez-Escobar, E.G.K.: Proof functional connectives. In: Di Prisco, C.A. (ed.) Methods in Mathematical Logic. pp. 208–221. Springer (1985). https://doi.org/10.1007/BFb0075313
14. Pimentel, E., Ronchi Della Rocca, S., Roversi, L.: Intersection types from a proof-theoretic perspective. Fundamenta Informaticae **121**(1-4), 253–274 (2012). https://doi.org/10.3233/FI-2012-778
15. Wells, J.B., Haack, C.: Branching types. In: Métayer, D.L. (ed.) ESOP. LNCS, vol. 2305, pp. 115–132. Springer (2002). https://doi.org/10.1007/3-540-45927-8_9