Second order lambda calculus for meaning assembly: on the logical syntax of plurals

, Christian Retoré

SECOND ORDER LAMBDA CALCULUS FOR MEANING ASSEMBLY: ON THE LOGICAL SYNTAX OF PLURALS Richard Moot, Christian Retoré Université de Bordeaux, Labri-CNRS, INRIA

Overview In order to model a number of phenomena of lexical pragmatics in a compositional framework, several contributions developed in our team [1, 4, 6, 5] have used the system F of Jean-Yves Girard (1971) [2, 3] to compose logical formulae expressing the meaning, while standard Montague semantics uses the simply typed λ-calculus. Thereafter we discovered that it is also relevant to model classical questions of semantics like generalised quantification [7], and in this paper for *plurals*. We should insist that we deal here with the *syntax of semantics*, that is we do no speak of reference or interpretation in models: an issue which is quite problematic for sophisticated semantic phenomena.

What is system F ? Instead of the simply typed λ-calculus we use the second order λ-calculus, namely Girard system F (1971), see e.g. [3]. Base types are constant types (the usual ones of TY_n, $\mathbf{t}$, $\mathbf{e}_i$, lots of entity types), or variable types, α, β, ... When T_1 and T_2 are types, so is $T_1 \to T_2$ and when T is a type and α is a type variable, $\Pi\alpha.\ T$ is a type as well – α usually appears in T but not necessarily.

As opposed to other type theories e.g. (I)TT, the system is conceptually and formally extremely simple, quite powerful,... and paradox free. Term building operations include the ones of simply typed λ-calculus:

$\boxed{vc}$ Constants (resp. variables) of a given type T are terms: $c : T$ (resp. $x : T$).

$\boxed{\text{e}\lambda}$ If u is a term of type $T_1 \to T_2$ and v is a term of type T_1, then $u(v)$ is a term of type T_2.

$\boxed{\text{i}\lambda}$ If u is a term of type T_2 and x a variable of type T_1, then $\lambda x.\ u$ is a term of type $T_1 \to T_2$.

These usual operations are completed by quite similar operations handling quantification over all types:

$\boxed{e\Lambda}$ If u is a term of type $\Pi\alpha.\ U$ and T is a type, then $u\{T\}$ is a (specialized) term of type $U[\alpha := T]$

$\boxed{i\Lambda}$ If u is a term of type T and if there is no occurrence of the type variable α in the type of any free variable (u works uniformly for every type α), then $\Lambda\alpha.u$ is a term of type $\Pi\alpha.\ T$ (that's the universal view of u).

Remember usual beta-reduction is $(\lambda x^T.\ u)t^T \rightsquigarrow u[x := t]$ Here, beta-reduction for types and Λ works just the same: $(\Lambda\alpha.\ u)\{T\} \rightsquigarrow u[\alpha := T]$.

Universal quantification over types let you define internally lots of type constructors, including the common ones, product, lists of α, existential quantification over types, lists of objects of type α, booleans, and here we mainly use:

$$\text{integers } \mathbb{N} = \Pi X.(X \to X) \to (X \to X)$$

F as a glue logic The system F (intuitionistic *second* order propositional logic) is used as a glue language (a.k.a. meta logic), but it glues whichever formulae one likes e.g. first order formulae, or higher order formulae. A result in the PhD of Bruno Mery [4] garantees that when λ-calculus constants are the ones of order n order logic (with $n \in \mathbb{N}$ or $n = \omega$), the normal λ-terms of F of type t are formulae of n order logic. This is close to usual Montague semantics where simply typed λ-calculus (intuitionistic propositional logic) with two base types e and t is used to express and to glue formulae of a priori ω order logic although one often only considers first order logic, via reification. Here because we want to con sider sets, we use second order logic, i.e. we can have properties of properties as in *the committee met*.

The type system we use consists in t, e and type variables — here we only consider one type of individual for simplicity but we usually have several base types, $e_1, \ldots, e_n$ like in Ty_n.

The logical syntax of plurals within F Using second order, one can:

- consider a cardinal function who maps properties to natural numbers: it is a constant $\|.\|$ (written as a circumfix) with type $(e \to t) \to \mathbb{N}$ where $\mathbb{N}$ is defined as above.
- assert that a property is a set so to speak, $\lambda P^{e\to t}.\mathit{house\_inhabitants}(P)$
- say that a property is a set containing several individuals $\lambda P^{e\to t}.\|P\| > 1 \wedge \mathit{committee}(P)$[1].

We can thus remain quite close to categorial syntax and Montague semantics, as shown in the lexicon in Figure 1. This enables us to automatically compute the logical formula (second order), for sentences like:

(1) The committee met.

(2) The students wrote a paper.

The second-order type for "the" allows us to treat both "the students" (which will produce a term of type $e \to t$) and "the student" (a term of type e, thereby naturally excluding "*the student met")

[1] To model that a committee is not necessarily identical to its members, it would be more appropriate to use an intensional type $e \to s \to t$ as a standard in Montague grammar. However, we will abstract away from this complication

word/phrase	syntactic type	lambda-term
student	n	$\lambda x^e.student(x)$
committee	n	$\lambda P^{e\to t}.\|P\| > 1 \wedge committee(P)$
-s	$n\backslash n$	$\Lambda\alpha\lambda P^{\alpha\to t}\lambda Q^{\alpha\to t}.\|Q\| > 1 \wedge \forall x^\alpha.Q(x) \Rightarrow P(x)$
q	np/np	$\Lambda\alpha\lambda x^\alpha\lambda y^\alpha y = x$
and	$(np\backslash np)/np$	$\Lambda\alpha\lambda P^{\alpha\to t}\lambda Q^{\alpha\to t}\lambda x^\alpha.P(x) \vee Q(x)$
the	np/n	$\Lambda\alpha.\iota^{(\alpha\to t)\to\alpha}$
each	$(s/(np\backslash s))/n$	$\Lambda\alpha\lambda P^{\alpha\to t}\lambda Q^{\alpha\to t}\forall x^\alpha P(x) \Rightarrow Q(x)$
all	$(s/(np\backslash s))/n$	$\Lambda\alpha\lambda R^{(\alpha\to t)\to t}\lambda S^{(\alpha\to t)\to t}\forall P^{\alpha\to t}R(P) \Rightarrow S(P)$
met	$np\backslash s$	$\lambda P^{e\to t}.\|P\| > 1 \wedge meet(P)$
sneezed	$np\backslash s$	$\lambda x^e.sneeze(x)$
wrote_a_paper	$np\backslash s$	$\lambda P^{e\to t}.write\_a\_paper(P)$
$*$		$\Lambda\alpha\lambda P^{\alpha\to t}\lambda Q^{\alpha\to t}\forall x^\alpha.Q(x) \Rightarrow P(x)$
$\#$		$\Lambda\alpha\lambda R^{(\alpha\to t)\to t}\lambda S^{(\alpha\to t)\to t}\forall P^{\alpha\to t}.S(P) \Rightarrow R(P)$
c		$\Lambda\alpha\lambda R^{(\alpha\to t)\to t}\lambda P^{\alpha\to t}\forall x^\alpha.P(x) \Rightarrow \exists Q^{\alpha\to t}Q(x) \wedge Q \subseteq P \wedge R(Q)$
$met^\#$	$np\backslash s$	$\lambda R^{(e\to t)\to t}\forall P^{e\to t}.R(P) \Rightarrow \|P\| > 1 \wedge meet(P)$
$sneezed^*$	$np\backslash s$	$\lambda P^{e\to t}.\forall x^e.P(x) \Rightarrow sneeze(x)$
$wrote\_a\_paper^c$	$np\backslash s$	$\lambda P^{e\to t}.\forall x^e.P(x) \Rightarrow \exists Q^{e\to t}Q(x) \wedge Q \subseteq P \wedge write\_a\_paper(Q)$

Figure 1: A small lexicon for plurals.

and a similar generalisation holds for the plural suffix "-s" which lifts a set to a set of sets. Standard notions, like Quine's "lifting" of individuals to singleton sets (q) and operations like distributivity (*), coverings (c) are easily treated as well. A restricted distributivity (#) from sets of sets to its constituent subsets allows us to naturally give two readings for "the committees met" (one total meeting and one meeting for each of the committees).

The parallelism between the type coercions (*) and (#) and the lexical entries for "each" and "all" is remarkable (and could be exploited to give more compact lexical entries).

In short, a simple second-order lexicon allows us to give a transparent treatment of at least the basic facts about plurals without any other ontological commitments, such a "plural indivuals" or "group entities".

References

[1] C. Bassac, B. Mery, and C. Retoré. Towards a Type-Theoretical Account of Lexical Semantics. *Journal of Logic Language and Information*, 19(2):229–245, April 2010.

[2] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application: l'élimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 63–92, Amsterdam, 1971. North Holland.

[3] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[4] B. Mery. *Modélisation de la sémantique lexicale dans le cadre de la théorie des types*. PhD thesis, Université de Bordeaux 1, July 2011.

[5] R. Moot, L. Prévot, and C. Retoré. A discursive analysis of itineraries in an historical and regional corpus of travels. In *Constraints in discourse*, page http://passage.inria.fr/cid2011/doku.php, Ayay-roches-rouges, France, Sept. 2011. Projet ITIPY de la région Aquitaine.

[6] R. Moot, L. Prévot, and C. Retoré. Un calcul de termes typés pour la pragmatique lexicale. In *Traitement Automatique du Langage Naturel, TALN 2011*, pages 161–166, Montpellier, France, June 2011. Projet ITIPY de la région Aquitaine.

[7] C. Retoré. Specimens: "most of" generic nps in a contextually flexible type theory. In *Genius III*, 2011.