Types in Natural Language

Fairouz Kamareddine

Logic, due to the paradoxes, is absent from the type free λ-calculus. This makes such a calculus an unsuitable device for Natural Language (NL) Semantics. Moreover, the problems that arise from mixing the type free λ-calculus with logic lead to type theory and hence formalisations of NL were carried out in a strictly typed framework. It was shown however, that strict typing cannot capture the self-referential nature of language [6, 1, 3] and hence other approaches were needed. For example, [6] creates a notion of floating types which can be instantiated to particular instances of types whereas [1, 3] use a type free framework. In this paper, we will embed the typing system of [6] into a version of [3] giving an interpretation of Parsons' system in a type free theory where logic is present. We take the standpoint that type freeness is needed yet types are indispensable. On this ground, by constructing types in the type free theory, we obtain a framework which can be seen as a formalisation of Parsons' claim that Natural Language needs type freeness in order to accommodate self referentiality yet many sentences should be understood as implicitly typed. We improve a lot in the expressivity of Parsons' system by allowing him to talk about sentences that he could not talk about previously. Even more, with our flexible typing scheme, we can allow any sentence and type check it as long as its type is not *circular* (i.e. paradoxical). If the type is circular, we change the final type of the sentence so that a paradox is impossible to derive. This approach is certainly flexible.

We argue that NL cannot be rigidly typed and that if we start from the type free λ-calulus, we can flexibly type NL terms. Types are polymorphic in the sense that we allow variable types which can be instantiated to anything. For example, the identity function has type $\beta_0 \to \beta_0$, and the identity function applied to of type e will result in elements of type e. The polymorphic power of the system comes from the ability to typecheck all polymorphic functions even those which are problematic in other systems. For example the fixed point operator, $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ is typechecked to $(\beta_2 \to \beta_2) \to \beta_2$ and even YY is typechecked to β_2. $\omega = \lambda x.xx$ is typechecked to $(\beta_1 \to \beta_1) \to \beta_1$ and ω applied to itself is typechecked to β_1 . These types can be instantiated so that YI where I is $\lambda x : e.x$, is typechecked to e naturally. We believe this system is one of the first which can typecheck all the above while remaining very expressive and simple. Another nice characteristic of the system is its ability to combine logic and the type free λ-calculus while remaining consistent. So even though the Russell sentence $\lambda x.\neg(xx)$ is a well formed sentence of the system, its type cannot be found. In fact, the system returns an error message explaining that this sentence has a circular type. The same thing applies to Curry's sentence $(\lambda x.xx \to \bot)$. Finally, the typing scheme that we present has a wide range of applications (see [3, 2, 5, 4]). The reason being that even though types are very informative either in programming language (PL) or in NL, type freeness and the non-restricted typing schemes are a necessity in interpreting many NL and PL constructs. We believe it necessary not to be too scared of the paradoxes to the point of using too restricted languages.

References

[1] Chierchia, and Turner, R., Semantics and property theory, *Linguistics and Philosophy 11*, 261-302, '88.

[2] Kamareddine, F., A system at the cross roads of logic and functional programming, *Science of Computer Programming 19*, 239-279, '92.

[3] Kamareddine, F., and Klein, E., Polymorphism, Type containment and Nominalisation, *Logic, Language and Information 2*, 171-215, '93.

[4] Kamareddine, F., A type free theory and collective/distributive predication, *Logic, Language and Information 4 (2)*, 85-109, '95.

[5] Kamareddine, F., Important Issues in Foundational Formalisms, *Interest Group of Pure and applied Logic 3 (2,3)*, 291-317, '95.

[6] Parsons, T., Type Theory and Natural Language, *Linguistics, Philosophy and Montague grammar*, S Davis and M Mithum (eds), University of Texas press, 127-151, '79.