Introducing a Type-theoretical Approach to Universal Grammar

Erkki Luuk
Institute of Computer Science, University of Tartu, Estonia
erkkil@gmail.com

Abstract

The idea of Universal Grammar (UG) as the hypothetical linguistic structure shared by all human languages harkens back at least to the 13th century. The best known modern elaborations of the idea are due to Chomsky. Following a devastating critique from theoretical, typological and field linguistics, these elaborations, the idea of UG itself and the more general idea of language universals stand untenable and are largely abandoned.

The presentation shows how to tackle the hypothetical structure of UG in a framework very different from the Chomskyan ones, using dependent and polymorphic type theory. The expected outcome of this work in progress is a versatile logic for expressing natural language morphosyntax and compositional semantics.

1 Universal Grammar

Type theory, in its modern form (i.e. as dependent and/or polymorphic type theory), is the most expressive logical system (as compared to the nonlogical ones such as set and category theory). As [6, 2, 3, 1] have shown, complex type theories outshine simpler ones in accounting for natural language (NL) phenomena like anaphora, selectional restrictions, etc. Second, as the notion of type is inherently semantic (type := class of semantic values), it is by definition suited for analyzing universal phenomena in NL, the semantics of which is largely universal (as witnessed by the possibility of translation from any human language to another).

The talk introduces a notationally simple system (L) matching NL morphosyntax as closely as possible. The purpose of L is to express the universal core of NL morphosyntax in a formally and linguistically precise, yet parsimonious way. L is furnished with elementary and complex morphosyntactic types, some of which are universal, others nonuniversal or possibly universal. The system is based on dependent and polymorphic type theory, with Martin-Löf’s type theory (MLTT; [5]) as the main type-theoretical reference point. L is a relational logic in prefix notation, with formulas like RUN(THE(Y(man))), (Y(LOVE))(mary,john), etc. (0th order relations (1st order arguments) in small letters, all other relations in capitals; Y a tense/person/number (etc.) marker). Elementary relation symbols represent constant types inhabited by language-particular (proof) objects (e.g. the type 'man' has objects man, homme, etc.). Both elementary and complex formulas of L are morphosyntactic types that have semantics (by the definition of type and by their alignment with morphosyntactic categories that have semantics (also by definition)). By a wff of L we mean one that is wf both syntactically and semantically, i.e. wf and well-typed. In L’s universe U, all formulas are morphosyntactic types and all types are morphosyntactic formulas (by Curry-Howard isomorphism).

L is capable of analyzing anaphoric devices (pronouns, agreement, complementizers), linguistic quantifiers (straightforwardly, since quantifiers are higher-order relations), selectional restrictions (in par with [2, 1], but more directly, with formulas like READ₁,₃(x, y), where capital subscript letters impose type restrictions on relations argument(s) (READ₁,₃(x, y) means that x is restricted to type I and y to type S)), etc. The centerpiece of L are typing rules,
some simple examples of which are o(N, PN, PRO, X/*, GER) : X; this, that, those, these... : DET; a, the, other... : DET; o(DET, DEM) : D (o(x) := all terms of type(s) x (x a sequence of types)). Largely because of L’s metalanguage operators |, * |, /, etc., only 30-50 rules are required for a (possibly) complete specification of UG.

2 Type theory

Two main kinds of polymorphism in NL are term underspecification (“data type polymorphism” in computer science) and argument ambiguity (e.g., “subtype polymorphism” in computer science). The latter is handled with function o (Sec. 1), since a judgement o(A1, ..., An) : C is a polymorphic binary relation of type (X ∈ A1,..., An, C). MLTT handles polymorphism with universes, so in MLTT one would fix a universe U = A1,..., An and the relation’s type would be written (∑ X ∈ U) C or ∑ (U, C).

We give an example of casting L’s formulas to MLTT. Posit universes Rel and Arg (for morphosyntactic relation and argument, resp.). Write an L formula A(B) (A : Rel, B : Arg).

Since arguments depend on relations, we need dependent types, i.e. a modern type theory such as MLTT. Per MLTT’s notation, A(B) gets the preliminary form (∏ A, B). The main dependent type constructors in MLTT are ∑ and ∏. With (∏ A, B), ∏ would reverse the natural order by taking Rel for the domain and Arg for codomain, so we will use ∑ instead. Thus the formula map from L to MLTT is A(B) → ∑ (A, B).

Term underspecification is resolved by /-types (cf. [4]). An underspecified term b will be typed with a /-separated string of symbols of all types over which b’s type is polymorphic. A /-type is defined wrt. at least two other types C1,..., Cn between which no subtyping relation holds.

The description of UG (and more generally, NL grammar) with dependent and polymorphic types is very different from the traditional Chomskyan approximation of UG with rewrite rules and/or syntax trees. The main differences from existing type-theoretical approaches (e.g. [6, 2, 3, 1]) are a focus on UG, an integrated treatment of morphosyntax and sentential/phrasal semantics, and an employment of polymorphic and dependent types.

Acknowledgement The author thanks Erik Palmgren and Roussanka Loukanova. This work was supported by IUT20-56 “Computational models for Estonian”, the Swedish Institute, and the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

References