Type theory and Universal Grammar*

Erkki Luuk

Institute of Computer Science, University of Tartu, J. Liivi 2, Tartu 50409, Estonia
erkkil@gmail.com
http://ut.ee/~el/a

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 proposal tackles the hypothetical contents of UG using dependent and polymorphic type theory in a framework very different from the Chomskyan one(s). Linguistic-typologically, the key novelty is introducing universal supercategories (categories of categories) for natural language modeling. Type-theoretically, we introduce a typed logic for a precise, universal and parsimonious representation of natural language morphosyntax and compositional semantics. The implementation of the logic in the Coq proof assistant handles grammatical ambiguity (with polymorphic types), selectional restrictions, quantifiers, adjectives and many other categories with a partly universal set of types.

Keywords: natural language · Universal Grammar · type theory · Coq.

1 Introduction

Although the idea of Universal Grammar (UG) harkens back to at least Roger Bacon (cf. [37]), the modern version of the hypothesis is usually credited to Chomsky [9,10,11]. In modern times, the notion of UG has taken several forms: a substantive [10], diluted [23] and implicational [16] one. However, a logical (derivational) path from implicational universals (structural traits implying other structural traits) to functional dependencies to substantive universals has been overlooked. The present paper tries to unveil this possibility in the form of a direct type-theoretical account of substantive universals as types or typed formulas.

From the present viewpoint, type theory is essential for a logical specification of UG. First, in its complex form (i.e. as dependent and/or polymorphic type theory), it is the most expressive logical system (contrasted with the nonlogical ones such as set and category theory). In a logical approach (i.e. in one with

* I thank Erik Palmgren and the anonymous reviewers of CIFMA. This work has been supported by IUT20-56 and European Regional Development Fund through CEES.

simpler alternatives such as ZOL, FOL, SOL and HOL), complex type theories outshine simpler ones in accounting for phenomena like anaphora, selectional restrictions, etc. [1,28,27,35]. Second, as the notion of type is inherently semantic:

(0) type := a category of semantic value,

it is by definition suited for analyzing universal phenomena in natural language (NL), as NL semantics in largely universal (as witnessed by the possibility of translation from any human language to another). Thus, if we could build a fundamentally semantic description of grammar (e.g. one on top of and integrated with a semantically universal description of NL), it would at least stand a chance of being universal.

2 Preliminaries

Returning to modern notions of UG, there are, then, three to consider (some of which are not mutually exclusive): one of an innate language acquisition device (LAD) [23], the second of implicational universals [16], and the third of the hypothetical linguistic structure shared by all human languages [10,11]. The details of implicational universals imply a (larger than singleton) set of universal grammars, so the second notion is irrelevant if one insists (as we do) on a single UG. Thus we have two alternatives, the LAD which can be termed "weak" and the substantive universals that amount to a "strong" UG (an even stronger version can be constructed by the condition "the linguistic structure shared by all possible human languages"; however, we only indicate this possibility here and speak of a "strong UG" in the more lax sense henceforth). While the very fact that normal human infants acquire NL without an explicit supervision is an evidence for a LAD, an evidence for substantive universals is much more contentious. More recently, the strong version of the hypothesis, having suffered heavy blows from the sides of both linguistic theory (e.g. [23]) and comparative typological evidence (e.g. [13]), has been severely discredited and seemingly largely demolished and abandoned. In this paper, we use type theory to set up a credible case for a strong UG, resulting in a framework very different from Chomskyan ones. In a sense, our approach will be more formal; secondly, the usual (although frequently implicit, and perhaps even inessential) Chomskyan notion of syntax-as-grammar will be supplanted by morphosyntax-as-grammar, where, moreover, "morphosyntax" will be fundamentally semantical in nature (on account of being typed — cf. (0)). But let us start by introducing some key concepts.

In a nutshell, the picture is as follows: mathematically speaking, there are functions with arities and arguments; some functions, arguments and the resulting formulas correspond to morphosyntactic constituents. By a morphosyntactic constituent we mean a well-formed formula (abbr. wff) of a NL expression. For example, in

(i) `D man`,

(ii) `run (D (Y man))`,
(iii) `Y love (1st, 2nd)`,
(iv) `man D`,
(v) `m (D an)`,

where `Y` is tense/person/number (etc.) marker (we will make a precise sense of this term later), and `D` a demonstrative or determiner, (i)-(iii) are wff-s of e.g. *the man* and *the men run* and *john loves mary*, respectively, while (iv)-(v) are ill-formed ((v) is already a notational gibberish). (i)-(iii) make use of the following conventions:

(1) Complex formulas are written in prefix notation, $a\,b$ or $a(b)$, with a standing for a function and b for argument(s),
(2) Left-associativity, i.e. left to right valuation (or derivation),

To keep the representation in close correspondence with natural language (NL), we avoid extralinguistic and theory-specific features, such as the model-theoretic quantification, as in $\exists x.\,\mathtt{man}(x)$ or $\exists x.\,\mathtt{D}(\mathtt{man}(x))$. In many cases, such features can be added later if a specific extralinguistic interpretation is desired. By (2), we get *john loves mary* rather than vice versa for (iii), if *john* is the 1st argument. By (2), the person-number relation marker *-s* will apply to `love` rather than to the non-wff `love (john, mary)`.

3 The specification language

This seemingly rather rudimentary representation bears some remarkable features. First, assuming the universality of `D`, `Y`, and expressions for *man*, *love* and *run*, (i)-(iii) are completely universal. We will elaborate on this point below (in this section and in sections 4 and 7). Second, it can capture certain aspects of syntax, morphology as well as semantics; moreover, it manages to do so without a cumbersome notation (e.g. trees, phrase structure rules or attribute value matrices). All words and morphemes (even those fused with stems like the plural marker in men) have meaning (i.e. semantics); hence the representation is compositional in both morphosyntactic and semantic respects (the exact nature of the compositionality will be made more precise later). For example, `Y` is a morphological (although perhaps also a syntactic) category in English. Obviously, we could make it more precise by substituting `Y` with `3SG` (third person singular) in (iii); the main reason for us not doing so is a tradeoff between such precision and universality.

Let us agree on some terminology. First, call the formulas (i)-(iii) formulas of a specification language (SL for short). Then, we **specify** SL's formulas from NL expressions and **derive** NL expressions from SL's formulas. We also assume that

(3) Arguments must be either specified or derived before the relation expressions in which they appear,

(4) A NL expression is well-formed both syntactically and semantically, i.e. well-formed and well-typed.

(3) is a self-evident axiom applying to both SL and NL expressions[1]; (4) precludes agrammaticality and type mismatches from NL. In section 6, we introduce a mechanism for extending well-typedness to accommodate many otherwise simplistically ill typed expressions.

For parsimony, we take all elementary arguments in SL (such as `man`) to be nullary functions. Thus, SL's vocabulary consists of function symbols, commas, spaces, and (matching) parentheses (in section 4 we will see that commas are optional). Next, we assume that

(5) For a particular language, the symbols are type constants; in UG they are type variables (e.g. `man` valuates to *man* in English and *homme* in French),
(6) A wff is well-formed both syntactically and semantically, i.e. well-formed and well-typed.

Notice that (6) is the SL counterpart of (4).

Now we are in the position to say something about the desired **derivation algorithm**. For the algorithm to consistently derive only (and preferably, all) NL expressions, it should adhere to the principles of

(7) Deriving a NL expression at all stages of derivation,
(8) Serial and incremental derivation of NL expressions (deriving one expression per stage using all previously derived expressions),
(9) Deriving as many complex subexpressions of the endexpression as possible.

(7)-(8) follow from (3)-(4); in addition, (7) follows from (8). Intuitively, (3) and (7)-(8) can be seen as following from Frege's principle of compositionality (this aside, they are stipulated for simplicity). (9) is a metric of the quality of the algorithm. By a 'stage of derivation' we mean a derivation of a term (i.e. of a NL expression). For example, the derivation from `D man` is 2-staged, with the first deriving e.g. *man* and second *the man*; we can write the derivation "man > the man" or "m > tm" in the abbreviated form.

4 Specifying formulas

The derivation algorithm can be used for specifying the SL formula of a NL expressions. However, it is better not to do this by hand, since 1) it is easy to blunder, and 2) the principles (1)-(4) are those of the lambda-calculus [2], i.e. we can use a suitable programming language. For example, in the Coq proof assistant we could specify the formula of *i know the man who was ill* as[2]

(vi) `Y know i (who (Y COP ill (the man)))`,

[1] A relation cannot be applied to its arguments before there are some.
[2] COP is copula.

which we could write either like this or as

(vii) `Y know i (who (Y COP (ill, the man)))`

in our SL (because (vi) is curried, which makes it equivalent to (vii)). In particular,

(viii) `Y know i (who (Y COP (ill, D man)))` : `S` : $\mathcal{U}$,

where `S` is sentence, $\mathcal{U}$ the top-level universe in SL, and $x : y :=$ "x has type y". While `Y` could be preferred for universality, we will also use more precise notations in Coq, e.g. `PRES know i (who (PAST COP ill (the man)))`[3]. The notation in (viii) is mid-way between a particular language (which is not necessarily English) and UG. It is not UG because `who` and `COP` are likely non-universal, and it is not necessarily English because many (or even most) other languages correspond to it (after the variables `PRES`, `know`, etc. have been instantiated with language-particular values). If we wanted an English-particular notation, we might have written

(ix) `know i (who (was (ill, the man)))` : `S` : $\mathcal{U}$,

5 Quantifiers

Since quantifiers are higher-order relations [40], i.e. an nth order quantifier is an $(n+1)$th order relation, they are straightforwardly implemented as such in the formulas of SL. By 'quantifiers' I mean those in the usual linguistic sense of the word, i.e. expressions like *much, many, all, every, few, no, some*, etc., so we will have considerably less quantifiers than in the theory of generalized quantifiers, where even a proper name like *john* would be a (type <1>) quantifier [25]. Here are some well typed Coq examples involving quantifiers:

```
Check PAST leave (few (PL man)). (*few men left*)
Check PRES know (several (PL man)) (a (few (PL man))).
(*several men know a few men*)
```

6 Selectional restrictions

The next (optional) step is to add selectional restrictions to our formalism (cf. [1,27]). Selectional restrictions are (onto)logical restrictions on the types of arguments of NL relations. For example, an adjective like *red* imposes the restriction that its argument be of type physical entity, while a verb like *know* imposes a restriction that its subject be a sentient and object an informational entity. Provisionally, we can write this

[3] The full formalization is at `https://gitlab.com/jaam00/nlc/blob/master/cop.v`

(x) [red]: $X_{\text{Phy}} \to \texttt{P}$
(xi) [know]: $X_{\text{Sen}} \to Y_{\text{Inf}} \to \texttt{S}$,

where $[x]$ is the interpretation of x, P is phrase, S sentence and X, Y are type variables indexed by selectional restrictions. The indexing can be done in several ways but, in general, X, Y must be compound types (types that are syntactic compounds of multiple types or their terms). For example, if X is a record type then selectional restriction could be its field, if X is an application type then selectional restriction could be its argument, etc. Other examples include Σ-, Π- and Cartesian product types but, depending on the programming language, there could be more or considerably less. As Coq has a very (the most?) complex type system, it has many possibilities for this kind of indexing[4].

Selectional restrictions are followed by default but not always; e.g. one may say *red ideas* in the metonymical sense of *communist ideas*, etc. For such contexts we stipulate the rule of metaphor or metonymy elimination:

$$\frac{u_j : Z_j \qquad (x_h^e : X_h^e) \mapsto (y_j^e : Y_j^e)}{x \dots (u_j)^e \dots : W} \text{ MM-Elim,}$$

where x_h^e is a function x, eth argument of which is restricted to h; X_h^e a function type X, eth argument type of which is indexed by h; u_j a (possibly empty) argument u, restricted to j; $x \mapsto y :=$ "x is interpreted as y"; and $X_h^e, Y_j^e, Z_j, W : \mathcal{U}$, where $\mathcal{U}$ the top-level universe in SL. Notice that metaphor and metonymy have to be introduced manually, there is no rule for this.

By MM-Elim, whenever we have a metaphor/metonymy (x_h^e is interpreted as y_j^e) and possibly u_j, $x \dots (u_j)^e \dots$ is well typed in SL (and NL). For example, $\{\text{idea}_{\text{Inf}}, (\text{red}_{\text{Phy}^1} \mapsto \text{communist}_{\text{Inf}^1})\} \vdash \text{red idea}_{\text{Inf}} : W$. As we take all elementary arguments to be nullary relations, we also have $\{\text{red}_{\text{Phy}^0} \mapsto \text{communist}_{\text{Inf}^0}\} \vdash \text{red}_{\text{Inf}} : W$ for argumental uses of the words.

Below are some Coq examples, well or ill typed depending on whether they conform to selectional restrictions (TAM is a tense-aspect-mood-voice marker):

```
Check TAM COP ill (a man): S. (*a man is ill*)
Check TAM COP red (the hut): S. (*the hut is red*)
Fail Check TAM COP ill (a hut). (*the hut is ill*)
Fail Check ill hut.
```

7 Universals

To live up to the promise of a strong Universal Grammar, some claims as to the universality of certain categories should be made. The task is difficult, as the received view among those who know better, i.e. typological linguists, is along

[4] For an implementation illustrating some of them, see https://gitlab.com/jaam00/nlc/blob/master/compound.v

the lines that nothing in NL is universal [20,13]. In fact, this opinion or rather dogma, superficially informed by the wealth of data on NL diversity, seems to be as bulletproof as that of its arch-nemesis, the Chomsky's theory of Universal Grammar. While blunt statements that "nothing in NL is universal" can be as bluntly refuted by general counterexamples like sign, form, meaning, word, sentence, morpheme, phrase, etc., a more subtle, even if a tentative, refutation is not easily concocted.

However, I claim that the main difficulty is conceptual rather than factual, being due to the virtual non-existence of universally shared definitions. Indeed, if the very terminology is non-universal, how could anyone find anything universal to align with it? Being mired from the very start, the quest for linguistic universals over a non-universal terminological landscape looks utterly hopeless, even ridiculous by definition.

To convince the reader in this, imagine a linguist classifying some linguistic phenomena x, y, z as belonging to a category X, and claiming the universality of X on the basis of universality of x, y or z (the latter being too marginal, elementary or specific for their universality being of any interest by itself). In all likelihood, there is also a definition of X he adheres to. Enter the next linguist who defines X in some other way, with the result that at least one of x, y, z does not belong to it anymore, thus likely also dropping the universality of X. The pattern gets repeated over and over, until no-one with any common sense and experience is inclined to further the issue of the universality of X anymore. The final verdict (or what appears so) then seems to be that "X is not universal", while in fact, it was not even the universality but the identity of X that the issue was all about.

Without further ado, we will porpose some universal linguistic categories, defined by their function (which in some cases, however, may be not so easy to ascertain). The first is the category of proper name (`PN`), the proposed universality of which is unlikely to raise any objections. The second is connective (`CON`), exemplified in English by words like *and, but, or, not, neither, if, then*, etc. While we cannot prove the universality of connectives in all ca. 6000-7000 languages of the world, a language without them seems too deficient to seriously consider the possibility. It is entirely possible, however, that in some languages they appear not as separate words but as clitics or affixes. In addition, a connective can be also specified prosodically and/or by juxtaposing constituents[5]. The third is `XP`[6], the universality of which follows from that of proper names. Fourth, the universality of sentence (`S`) is beyond any reasonable doubt. Fifth, since all languages should allow for questions and answers, we must posit the

[5] In most written languages, the comma is a good example. Also, there is an anecdote of Bertrand Russell giving a flight attendant a lesson on the inclusiveness of *or* by answering "Yes" to her query on whether he wants tea or coffee. Seuren comments that if the story were true, Russell must have ignored the question's intonation that marks for an exclusive *or* [39].

[6] Frequently also referred to as NP or DP.

universality of interrogative sentence (IS). Sixth, the universality of connective compositions (CONC — *x* and *y*, *x* or *y*...) follows from that of CON.

Unfortunately, this is about as far as conventional grammatical categories get us. Beyond this, the universality of the categories we are interested in is dubious. For example, determiners (such as the English *a, the*) are likely non-universal [33]. The universality of adjectives, nouns, verbs, and most other sufficiently general grammatical categories is unclear [22,29,24,12,31]. Even the universality of dependent clauses is under doubt [14].

However, since our approach to syntax and morphology (i.e. grammar) is fundamentally semantical, we can proceed by taking a functionalist perspective. For example, while there are languages without cases, the function of case of "marking dependents for the relationship they bear to their heads" [4] should be universal across all languages with sufficiently complex head-dependent relations (where such book-keeping is necessary), i.e. in all full-blown languages — i.e. in all languages except pidgins. It is well known that the main difference between cases and adpositions is formal (the former being affixes or clitics and the latter words); semantically and pragmatically they are largely co-extensive [4,5]. From the functionalist perspective, it makes perfect sense to form a supercategory CA (case or adposition) for all cases and adpositions and posit its universality. Indeed, as cases exist in an overwhelming majority of languages as do adpositions, the probability of a full-blown language without both is negligible already statistically, and approaches 0 when combined with the semantic consideration. The universality of CAP (case/adposition phrase, e.g. *john, him, to the house* in English[7]) follows from the universality of CA.

The universality of the supercategory Q of numerals (e.g. *one, two*...) and quantifiers (*all, no, some, few*, etc.) seems semantically inevitable. However, it is unlikely that all quantifiers in all languages are syntactically equivalent to the English ones[8]. With roughly the same logic, we can form supercategories D (determiner or demonstrative[9]), TAM (tense, aspect, mood, voice), POS (genitive, possessive)[10], ADL (adverbs or other adverbial phrases[11]), and propose their universality.

This set of universal supercategories is still missing the most important ones. However, before we get to them, we should also posit the existence of polymorphic categories, exemplified by words like *sleep* and *run* in English. In linguistic typology, the general category is called flexible [29], and positing it is preferable

[7] *john* is in nominative or accusative, i.e. a CAP as well as XP.

[8] "Warlpiri and Gun-djeyhmi, for example, make use of verbal affixes to express various kinds of quantificational meaning. And Asurini quantifiers such as *all, many, two* do not form a syntactic constituent with the noun, because they do not belong to the category of determiners. They are instead members of other categories such as adverb, verb and noun" [33].

[9] In English, words like *this, that, those*, etc.

[10] The universality of possessives has been posited by [21].

[11] E.g. *quickly* and *in a hurry*, respectively, in English. It is a moot point whether adverbial participle should be also included in the category or analyzed somehow differently.

to positing two distinct *sleeps*, the noun and the verb. The latter is especially true for formalizations, where the *sleeps* would have to be already formally distinct (e.g. `sleep` and `sleep0`), thus contributing to the formalization's unnaturality. Here are some tests with a Coq implementation of the flexible that is polymorphic between function and argument:

```
Check sleep: gs _ _ _ _ _. (*typed as argument*)
Check sleep: NF → _ → S. (*typed as function*)
Fail Check PAST sleep man. (*fails since "man" is not an XP*)
Check PAST sleep (a man). (*a man slept*)
Check PAST sleep (few (PL man)). (*few men slept*)
Check PAST sleep (a (few (PL man))). (*a few men slept*)
Check a sleep.
Fail Check PAST sleep (a hut). (*a hut slept*)
Fail Check PAST sleep (a sleep). (*a sleep slept*)
```

The last two checks fail as they violate selectional restrictions.

Now we can posit the universality of the supercategories of core relation (`R` — verb, copula, infinitival relation, auxiliary verb or flexible-over-relation) and core argument (`X`[12] — noun, proper name, pronoun, gerund or flexible-over-argument)[13]. Since *sleep* "flexes" between relation and argument, it is both a flexible-over-relation and flexible-over-argument. There are many categories of flexibles, e.g. *have* may be a flexible between auxiliary verb (`AUX`) and infinitival relation (`IR`), a type which we write `AUX/IR`. So *sleep* has type `X/R`. Of course, in many languages you would translate *sleep* into different words depending on whether its type is `X` or `R` in the context, i.e. a semantic (near-)equivalent of *sleep* does not have type `X/R` in all languages. Notice that, differently from `X` and `R`, we do not posit the universality of type `X/R` (although this, too, is possible [29]). In fact, there are several possibly universal categories (e.g. nouns, verbs, adjectives, numerals, etc.) that we have not proposed as universal, as we want to give a conservative estimate.

Some of the types in the Coq implementation (e.g. *a, few...*) are clearly non-universal, while others (e.g. *sleep, man...*) are as clearly universal. Importantly, in Coq we can also define universal notations, e.g.

```
Parameter D: ∀ {x y z u w}, gs x y z u w → gp cp_x y z u w.
(*universal "D" declared as a variable*)
Notation D' := (_: gs _ _ _ _ _ → gp cp_x _ _ _ _).
(*universal "D'" defined as a notation*)
```

The universality of `D` and `D'` comes from `x`, `y`, `z`, `u`, `w` and `_` standing for any admissible term or type, whence the applicability of `D` and `D'` whenever one of

```
a: ∀ {x y z w}, gs x y z SG w → gp cp_x y z SG w
```

[12] From `XP`.

[13] Examples: an infinitival relation is *like* in *i like to run*, an auxiliary verb is *must* in *i must run* and a gerund is *running* in *running is healthy*.

```
the: ∀ {x y z u w}, gs x y z u w → gp cp_x y z u w
this: ∀ {x d w}, gs cs_s x d SG w → gp cp_x x d SG w
these: ∀ {x d b w}, gs cs_p x d b w → gp cp_p x d PLR w
```

applies:

```
Check PRES know (the man) (a (few (PL man))). (*the man knows a few men*)
Check TAM know (D' man) (D (Q (PL man))). (*e.g. "the man knows a few men"*)
```

The applications `TAM know (D man) (D (Q (PL man)))` , `TAM sleep (D (Q (PL man)))` , `D (Q (PL man))` , etc., could be universal, assuming the universality of plural (`PL`). The latter is a moot point in e.g. Japanese and Vietnamese, where in many contexts we must use "counter words" or numeral classifiers instead of generic plurals [15]. It is not clear whether (all) numeral classifiers could be viewed as specific kinds of plurals.

On a different note, it is also not entirely clear whether the composition `D (Q X)` is universal or whether it might be `Q (D X)` in some languages instead. While the logic of the universal determiner phrase (`XP`) suggests a `D` as its head (i.e. the highest-order function), the logic itself may be patterned more on English than ontology. The general problem behind this is that *the ontology may be always contaminated by the particular language in which it is expressed*, i.e. there is no correct and independent language for expressing the kind of ontology we are interested in. For such reasons, the universality of SL formulas exclusively composed of universal elementary types is, in general, an open problem.

8 Related work

This work should fit into the existing tradition of applying complex type theories on NL (e.g. [1,27,35]), while being sufficiently distinct from it. Our approach integrates three levels of linguistic description — morphology, syntax and compositional semantics — in the notion of compositionality, analyzed with functions, and connects it with UG. As such, it is quite unique. For comparison, it is convenient to review which of the four external ingredients (morphology, syntax, semantics, UG) is missing in other approaches.

It is noteworthy that while most modern type-theoretical approaches tackle only semantics [1,27,3] or semantics and syntax [6,8,7], and at least one syntax and morphology [36], there seems to be only one that (with certain qualifications) considers all three [36]. We will review this approach below. Likewise, as far as I know, there is only one paper, besides (and eponymous to) the present one, considering a type-theoretical approach to UG [37]. This approach is based on Grammatical Framework (GF) [36]. GF is a task-specific high-level programming language, and the task is that of formalizing grammars (not only NL grammars, although GF is geared towards writing formal NL grammars). Of the three levels we are interested in, only syntax and morphology constitute a grammar; correspondingly, GF lacks a native support for semantics, although Abstract Meaning Representation and FrameNet libraries have been partly implemented

for it [18,19]. GF is based on type theory, and organized into two levels: type checking and evaluation is performed on the level of abstract syntax, while the "pretty-printed" (linearized) level of concrete syntax is what the (object, e.g. natural) language users see and understand. While the abstract syntax consists of typed categories and functions, the concrete syntax has linearization types and functions. In particular, GF's abstract syntax is a logical framework [32], while the concrete syntaxes are e.g. English, Latin, C, Udmurt, etc. As such, GF's abstract syntax must be universal, while its universal applicability to all NLs is another matter. The main claims of [37] are that the latter universality has two dimensions — universality across languages and universality across subject matters, — and that it is possible to build cross-linguistically universal grammars on limited domains only. It is not clear whether the second claim is more general or applies to GF only.

Since GF is a full-blown programming language, it makes no sense to compare it with the present paper, so we will focus only on its connection to UG, as laid out in [37]. The main difference between the present approach and [37] is that we propose the universality of many specific categories or types, while the latter is, in comparison, a very general analysis of what can and has be(en) done in GF in terms of UG (cf. the main claims above). In particular, [37] does not propose the universality of a single category or construction, i.e. makes no attempt at a linguistic-typologically informed approach. This may reflect the fact that GF's resource grammar libraries have been partly implemented for ca. 40 languages [38], which is a tiny fraction of the ca. 6000 languages out there. Thus, if someone wants to know what is universal in NL, they must surely look beyond GF, while [37] is concerned only with GF, and the kind of universality (of abstract syntax over limited domains) it offers. The present approach to UG has no domain restrictions of the sort, and the idea motivating [37], the notion of a "language game" on a limited domain, and its universality (as opposed to the universality of a core component of NL), is alien to it. "One of the important points in Wittgenstein's late philosophy is that there is no such thing as language, but just a collection of language games (Wittgenstein 1953)" [37]. The Wittgenstein's viewpoint is hardly one that a linguist (typological or not) would subscribe to. "What we call a cross-linguistic language game corresponds to an area of multilingual activity and a tradition of translation, e.g. among scientists within one discipline, among employees within a multinational corporation, and among sportsmen practicing the same sport" [37]. Clearly, scientists, employees as well as sportsmen must get their linguistic meaning through, which entails compositional (syntactic, morphological and semantic) well-typedness. The latter, as we claim, is largely universal, if properly abstracted from the surface forms.

As for simply-typed [30,17,26] or non-type-theoretic [34] formalisms, I am unaware of any of them addressing UG. In general, UG has surprisingly little currency outside its Chomskyan use and those derived from it (criticism, reformulation, review, etc.). In this sense, [37] is an interesting exception.

9 Implementation

The NL fragment I have implemented in Coq[14] comprises stems, nouns, verbs, flexibles, proper names, pronouns, XPs, adjectives, determiners, demonstratives, quantifiers, tense-aspect-mood, gender, number and nonfinite markers, cases, sentences (both simple and complex), complementizers, copulas and selectional restrictions (for physical, informational, limbed, animate, biological and sentient entities). In addition, I have implemented universal (super)categories `PN`, `X`, `XP`, `CA`, `Q`, `D`, `TAM`.

Another implemented fragment[15] adds to Supplement A sentential, adjectival and generic adverbs, adpositions, connectives and connective phrases (for substantives, adjectives, adverbs and sentences). Other differences between the Supplements are that they use different encodings of selectional restrictions and that only Supplement A encodes universal supercategories. The linguistic categories not formalized in Supplements A or B are gerunds, participles, auxiliary verbs, interrogatives, numerals, negation, mass/count distinction and unspecified selectional restrictions (and possibly other categories). These are omitted not because of a special difficulty formalizing them would pose but remain a future work.

The goal of the implementation was to model not a quantitatively extensive but a structurally rich set of NL expressions. Thus, the implementation has few representatives of each category. A quantitative extension of this set should be generated semi-automatically, engaging tagged corpora or dictionaries with machine learning or standalone scripts.

The implementation makes heavy use of Coq's complex type system (dependent and polymorphic types), as well as its specialized features like custom notations, Coq's official tactic language Ltac, type classes (mostly for type inference), and canonical records (canonical instances of a record type, for notation overloading and type inference). It is probably feasible to implement the formalism in another language with dependent and polymorphic types (e.g. Haskell, OCaml or Agda), and maybe even in a typewise simpler statically typed language (or even in a dynamically typed language, e.g. by combining custom classes with set-theoretic operations). I encourage the interested readers to experiment along these lines in a language of their choice.

10 Conclusion

We have shown how to build an extensive and robust substantive UG with type theory using supercategories (categories of categories). As such, the main contributions of this paper fall under two rubrics: UG and type-theoretical modeling.

Starting from the first, we have proposed the universality of categories `PN`, `CON`, `XP`, `S`, `IS`, `CONC` and of supercategories `CA`, `CAP`, `Q`, `D`, `TAM`, `POS`, `ADL`, `R`, `X`. This is a conservative estimate, i.e. there should be more (rather than less) universal

[14] Supplement A: `https://gitlab.com/jaam00/nlc/blob/master/cop.v`
[15] Supplement B: `https://gitlab.com/jaam00/nlc/blob/master/frag.v`

categories in these lists, but this will also depend on your typological theory (namely on how it partitions language into categories). The lists are preliminary and open for methodological reasons already (cf. section 7) but should make a good first approximation of universal morphosyntactic categories in NL.

An advantage of using type theory is that it lends itself well to formalization. To account for systematic violations of selectional restrictions by metaphor and metonymy, we have shown how to model them type-theoretically. I have also implemented a fragment of NL and UG in the Coq proof assistant (ver. 8.9). Compositional semantics is incorporated to the formalism in the form of selectional restrictions, while syntax and morphology are represented by their respective categories. A novel and parsimonious feature of the Coq formalization is that it annotates syntactic, morphological, and compositional semantic information on a single level of type. This is in stark contrast with all other typed NL formalisms I know. For example, a modern formalism like AACG [26] requires (besides recruiting Categorial Grammar and category theory in addition to type theory) simply-typed lambda-calculus, two separately typed syntaxes (abstract and concrete), typed semantics, syntax-semantics and abstract-concrete syntax interfaces, and does not even cover compositional semantics, i.e. selectional restrictions. The advantages of (A)ACG [26,17], HPSG [34], GF [36] and many other formalisms are that they support language-specific constituent orders; in addition, ACG and AACG have truth-functional semantics. The first could be implemented with a language-dependent function from formulas to strings, the second has been implemented (with e.g. an Ltac tactic, which is only one possibility for this) in Supplement B.

Technically, the implementation shows how to model many (super)categories of NL, some of them universal, in a purely functional type system (i.e. one comprising only functions and their types) with dependent and polymorphic types. It seems likely that the underlying formalism could be also encoded in a simpler type system, which, along with implementing the missing categories mentioned in section 9, remains a future work.

References

1. Asher, N.: Selectional restrictions, types and categories. Journal of Applied Logic **12**(1), 75–87 (2014). https://doi.org/10.1016/j.jal.2013.08.002
2. Barendregt, H.: Lambda calculi with types. In: Handbook of Logic in Computer Science. pp. 117–309. Oxford University Press (1992)
3. Bekki, D., Asher, N.: Logical polysemy and subtyping. In: Motomura, Y., Butler, A., Bekki, D. (eds.) New Frontiers in Artificial Intelligence, pp. 17–24. Springer, Berlin, Heidelberg (2013)
4. Blake, B.J.: Case: Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge [England]; New York, NY, USA (2001)
5. Butt, M.: Theories of Case: Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge [England]; New York, NY, USA (2006)
6. Chatzikyriakidis, S., Luo, Z.: Natural language inference in Coq. Journal of Logic, Language and Information **23**(4), 441–480 (Dec 2014). https://doi.org/10.1007/s10849-014-9208-x

7. Chatzikyriakidis, S., Luo, Z.: Natural language reasoning using proof-assistant technology: Rich typing and beyond. In: Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS). pp. 37–45. Association for Computational Linguistics, Gothenburg, Sweden (April 2014), http://www.aclweb.org/anthology/W14-1405
8. Chatzikyriakidis, S., Luo, Z.: Proof assistants for natural language semantics. In: Amblard, M., de Groote, P., Pogodalla, S., Retoré, C. (eds.) Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016). pp. 85–98. Springer, Berlin, Heidelberg (2016), http://www.cs.rhul.ac.uk/~zhaohui/LACL16PA.pdf
9. Chomsky, N.: Remarks on nominalization. In: Jacobs, R., Rosenbaum, P. (eds.) Readings in English Transformational Grammar, pp. 184–221. Ginn, Waltham (1970)
10. Chomsky, N.: Lectures on Government and Binding. Foris, Dordrecht (1981)
11. Chomsky, N.: The Minimalist Program. Current studies in linguistics series 28, MIT Press, Cambridge, MA (1995)
12. Czaykowska-Higgins, E., Kinkade, M.D.: Salish languages and linguistics. In: Czaykowska-Higgins, E., Kinkade, M.D. (eds.) Salish Languages and Linguistics: Theoretical and Descriptive Perspectives, pp. 1–68. Mouton, The Hague (1998)
13. Evans, N., Levinson, S.C.: The myth of language universals: Language diversity and its importance for cognitive science. Behavioral and Brain Sciences **32**, 429–492 (2009)
14. Futrell, R., Stearns, L., Everett, D.L., Piantadosi, S.T., Gibson, E.: A corpus investigation of syntactic embedding in Pirahã. PloS one **11**(3), e0145289 (2016). https://doi.org/10.1371/journal.pone.0145289
15. Gil, D.: Numeral classifiers. In: Dryer, M.S., Haspelmath, M. (eds.) The World Atlas of Language Structures Online. Max Planck Institute for Evolutionary Anthropology, Leipzig (2013), https://wals.info/chapter/55
16. Greenberg, J.H.: Some universals of grammar with particular reference to the order of meaningful elements. In: Greenberg, J.H. (ed.) Universals of Grammar, pp. 73–113. MIT Press, Cambridge, MA (1966)
17. de Groote, P.: Towards abstract categorial grammars. In: Proceedings of 39th Annual Meeting of the Association for Computational Linguistics. pp. 252–259. Association for Computational Linguistics, Toulouse, France (Jul 2001). https://doi.org/10.3115/1073012.1073045, https://www.aclweb.org/anthology/P01-1033
18. Gruzitis, N., Dannélls, D.: A multilingual FrameNet-based grammar and lexicon for controlled natural language. Lang Resources & Evaluation **51**(1), 37–66 (2017). https://doi.org/10.1007/s10579-015-9321-8
19. Gruzitis, N.: Abstract meaning representation (amr) (2019), https://github.com/GrammaticalFramework/gf-contrib/tree/master/AMR
20. Haspelmath, M.: Pre-established categories don't exist: Consequences for language description and typology. Linguistic Typology **11**(1), 119–132 (2007). https://doi.org/10.1515/LINGTY.2007.011
21. Heine, B.: Cognitive Foundations of Grammar. Oxford University Press, Oxford (1997)
22. Himmelmann, N.P.: Lexical categories and voice in Tagalog. In: Austin, P., Musgrave, S. (eds.) Voice and Grammatical Functions in Austronesian Languages. CSLI, Stanford (2007)
23. Jackendoff, R.: Foundations of language: brain, meaning, grammar, evolution. Oxford University Press, Oxford, New York (2002)

24. Jacobsen, W.H.: Noun and verb in Nootkan. In: Efrat, B. (ed.) The Victorian Conference on Northwestern Languages, pp. 83–153. British Columbia Provincial Museum, Victoria (1979)
25. Keenan, E.L., Westerstahl, D.: Generalized quantifiers in linguistics and logic. In: Handbook of Logic and Language. pp. 859–910. Elsevier, Burlington, MA (2011)
26. Kiselyov, O.: Applicative abstract categorial grammar. In: Kanazawa, M., Moss, L.S., de Paiva, V. (eds.) NLCS'15. Third Workshop on Natural Language and Computer Science. EPiC Series in Computing, vol. 32, pp. 29–38. EasyChair (2015). https://doi.org/10.29007/s2m4, `https://easychair.org/publications/paper/RPN`
27. Luo, Z.: Type-theoretical semantics with coercive subtyping. In: Semantics and Linguistic Theory. vol. 20, pp. 38–56. Vancouver (2010)
28. Luo, Z.: Formal semantics in modern type theories: is it model-theoretic, proof-theoretic, or both? In: Asher, N., Soloviev, S. (eds.) Logical Aspects of Computational Linguistics 2014 (LACL 2014), pp. 177–188. No. 8535 in LNCS, Springer, Berlin, Heidelberg (2014)
29. Luuk, E.: Nouns, verbs and flexibles: implications for typologies of word classes. Language Sciences **32**(3), 349–365 (2010). https://doi.org/10.1016/j.langsci.2009.02.001
30. Montague, R.: The proper treatment of quantification in ordinary English. In: Portner, P., Partee, B.H. (eds.) Formal Semantics: The Essential Readings, pp. 17–34. Blackwell, Oxford (2002)
31. Peterson, J.: There's a grain of truth in every "myth", or, Why the discussion of lexical classes in Mundari isn't quite over yet. Linguistic Typology **9**(3), 391–405 (2005)
32. Pfenning, F.: Logical frameworks — a brief introduction. In: Schwichtenberg, H., Steinbrggen, R. (eds.) Proof and system-reliability. Springer, Berlin (2002), `http://www.cs.cmu.edu/~fp/papers/mdorf01.pdf`
33. Plank, F.: Determiner universal. In: The Universals Archive (2006), `https://typo.uni-konstanz.de/archive/nav/browse.php?number=1201`
34. Pollard, C., Sag, I.A.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Chicago (1994)
35. Ranta, A.: Type-theoretical grammar. Clarendon Press, Oxford; New York (1994)
36. Ranta, A.: Grammatical Framework: a type-theoretical grammar formalism. The Journal of Functional Programming **14**(2), 145–189 (2004). https://doi.org/10.1017/S0956796803004738
37. Ranta, A.: Type Theory and Universal Grammar. Philosophia Scienti. Travaux d'histoire et de philosophie des sciences (6), 115–131 (2006)
38. Ranta, A.: The status of the GF resource grammar library (2017), `http://www.grammaticalframework.org/lib/doc/status.html`
39. Seuren, P.A.M.: Language from within: Vol. 2. The logic of language. Oxford University Press, Oxford (2010)
40. Westerstahl, D.: Generalized quantifiers. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2016 edn. (2016), `https://plato.stanford.edu/archives/win2016/entries/generalized-quantifiers`