

# Type Theory and Universal Grammar

Aarne Ranta

Department of Computer Science and Engineering  
Chalmers University of Technology and Göteborg University

**Abstract.** The paper takes a look at the history of the idea of universal grammar and compares it to multilingual grammars, as formalized in the Grammatical Framework, GF.

**Résumé.** L'article jette un coup d'oeil sur l'histoire de l'idée d'une grammaire universelle et fait ensuite une comparaison avec les grammaires multilingues, telles que formalisées dans GF, "Grammatical Framework".

## 1 Universal grammar

### 1.1 Mediaeval ideas

Universal grammar is an idea often attributed to mediaeval philosophy. There are two famous quotes, appearing in Gilson (1922), and later requoted in Lyons (1968):

Grammar is substantially the same in all languages, even though it may undergo in them accidental variations.  
(Roger Bacon, 13th century)

He who knows grammar in one language, also knows it in another as far as the essentials are concerned. The fact that he cannot, however, speak another language, or understand those who speak it, arises from the difference of words and their formations, which is accidental to grammar.  
(Anonymous, 12th century)

Universal grammar was severely criticized in the Renaissance time by scholars such as Alexander Hegius and Erasmus. For many contemporary linguists, it is a notion that only an "armchair linguist" can maintain.

In the anonymous 12th century quote, languages are said to differ only as for “words and their formations”. Even a quick translation experiment would show a sense in which this cannot be true: an English four-word utterance is translation into a Finnish one-word utterance,

*also in my hoúse = talossanikin*

Moreover, the stress on the word *house* is important, since with another stress, the Finnish translation has two words:

*also in mý house = minunkin talossani.*

Nonetheless, we will see later that the idea that languages only differ as for “words and their formations” does make some sense, after all.

## 1.2 Universal language in Descartes and Leibniz

In a letter to Mersenne in 1629, Descartes commented on a universal grammar and dictionary that someone had been marketing commercially, with the promise that “anyone who learns this (universal) language, would also know all the others as dialects of it” (Descartes 1629). Descartes found the idea naïve, and raised several arguments against it. However, he also gave his own suggestion of a universal language, such that it would

establish an order among all thoughts that can enter in the human spirit, in the same way as there is a natural order among numbers, and as one can learn in one day the names of all numbers up to infinity and write them in an unknown language, even though they are an infinity of different words the invention of this language depends on the true philosophy; for it is impossible otherwise to denumerate all thoughts of men and order them, or even distinguish them into clear and simple ones

if anyone had well explained which are the simple ideas that are in the imagination of men, of which all that they think is composed ... then I would dare to hope for a universal language easy to learn, pronounce, and write and ... which would help judgement, representing all things to it so distinctly that error would be almost impossible

(Descartes 1629).

Descartes's suggestion is not widely known: the modern idea of universal language is usually traced back to Leibniz's *characteristica universalis*, a symbolic language permitting mechanized reasoning by means of a *calculus ratiocinator*. This proposal is from 1732, and advocates, like Descartes's, a mathematical notation such that the elements of the notation correspond to the elements of things and facts. The emergence of this idea in both Descartes and Leibniz is natural, given that they both made major contributions to mathematics with notational innovations as an important ingredient: analytic geometry in Descartes's case, and differential and integral calculus in Leibniz's.

Unlike Descartes, it is not sure whether Leibniz had explicit thoughts of his universal language as a bridge between different languages. The main aspect for both of them was that the notation would admit of a *calculus* to replace creative reasoning. A contemporary variant of the calculus idea is, of course, that a universal notation can be manipulated by a *computer program*, which can decide the correctness of judgements and—if the notation serves as bridge between languages—translate between languages.

### 1.3 Two dimensions of universality

Speaking of a universal grammar, or of a universal language, we have to distinct between two senses of universality:

*Horizontal universality*: generality across languages,

*Vertical universality*: generality across subject matters.

We chose “vertical” and “horizontal” mostly because we did not find better names, but they do suggest a major point we want to make, namely that these aspects are *orthogonal*. Therefore, to assess any proposal of and any argument against a “universal grammar”, we have to find out which sense is meant.

What Roger Bacon wrote about was explicitly about horizontal universality. Leibniz was explicit that he meant vertical universality. Descartes was concerned about both dimensions.

The *Sapir Whorf hypothesis* is a famous 20th century challenge of universality, and it is clearly about horizontal universality:

No two languages are ever sufficiently similar to be considered as representing the same social reality. The worlds in

which different societies live are distinct worlds, not merely  
the same world with different labels attached  
(Sapir 1929)

This point raised by Sapir is often discussed as a problem of *translation* between languages: not only is it difficult to *find* a translation from one language to another, but it may even happen that no translation *exists*, since the content expressed by the source language utterance has no counterpart in the target language.

Also vertical universality has important 20th century challenges. The *incompleteness* proof of Gödel (1931) implies that there cannot be a formal system that is complete for all mathematics—let alone for all subject matters, of which mathematics is but a fraction. And 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). Nor is there such a thing as the meaning of a word (*simpliciter*), but only its use in a language game. Now, an individual language game is a unit that has a set of rules that can possibly be formalized into a formal system; but the totality of language games cannot be formalized.

## 1.4 Cross-linguistic language games

The main thesis of this paper is that

we can achieve horizontal universality but not vertical universality.

In other words, we can build cross-linguistic grammars on limited domains, or, “we can translate language games”.

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. Of course, we have to leave it open whether horizontal universality covers *all* languages in the world: given a language game, we can only claim universality over those languages in which the game can be played.

Cross-linguistic equivalence based on language games is clearly different from the genetic relatedness of languages. There are several examples showing that shared cultural activities may be more important than genetic relatedness: Swedish (Germanic) and Finnish (Fenno-Ugrian) are

largely intertranslatable—there is even a bilinguality legislation in Finland saying that all official documents must exist in both languages; the two Germanies after the Second World War were reported to be drifting apart linguistically; as a non-sailor, I cannot speak about sailing with my brother, dedicated sailor, in our native language Finnish. One criterion, or test, of horizontal equivalence is that

it is possible to *translate* from one language to another.

This may of course fail in practice for some technical reason; hence we can replace this criterion by “translatability in principle”:

it is possible to *express the same things* in the two languages.

In a sense, translatability even in principle fails very soon: for instance, a pun based on ambiguity in one languages does not generally translate to others as the same pun. Lots of famous examples can be found from Bible translation. For instance, there are so-called alphabetical Psalms, where subsequent verses begin with subsequent letters of the Hebrew alphabet. Translators have had different ambition levels in reproducing this feature in other languages. Normally, however, we speak of translation on some *level of abstraction*. A typical level is semantical: the translation of expressions *as expressions for certain things*.

## 1.5 Defining a level of abstraction

A natural way to define the semantic level of abstraction is to take as the starting point the things expressed—not the languages! In other words, one can start with a formalized, mathematical model of those things and see how it is reflected in languages—instead of starting with one of the languages and trying to find a model for that language.

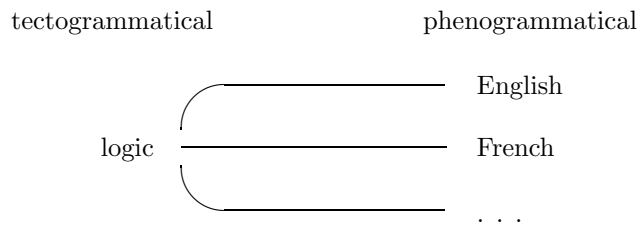
This approach is opposite to the idea of formalization exercises that are customary in elementary logic teaching. Such an exercise consists of translating natural language sentences into logical formulae, with a painful awareness that something is getting lost. For instance, the meaning of *love* is felt to be lost when formalizing *John loves Mary* as *love(John, Mary)*. It is much less painful to start from the formula and ask how to express it in natural language! The logic textbook of Kalish and Montague (1964) is one of the few that contains exercises in this direction: in retrospect, it can well be seen as a precursor of Montague grammars (see next section).

## 2 Multilingual grammars

### 2.1 Curry's model of multilingual grammar

The logician Curry published in 1963 a linguistic paper in which he distinguished between the *tectogrammatical structure* and the *phenogrammatical structure* of a language. The tectogrammatical structure has to do with how expressions are divided into meaningful parts; the phenogrammatical structure has to do with how expressions look like. It is the tectogrammatical structure to which, for instance, semantics applies; that subsequent verses of a Psalm begin with subsequent letters is rather a phenogrammatical fact.

Together with the distinction, Curry formulated a program for multilingual grammars: such a grammar would have one tectogrammatical structure and many phenogrammatical structures. For Curry, the tectogrammatical structure was to be described by combinatory logic. This gives the following picture:



The semantic aspect of Curry's model was developed in great detail by Montague (1974; the background of Montague's work in Curry's was pointed out by Dowty (1982)). Neither Curry nor Montague pursued the multilingual aspect, but it is present in the works of Shaumyan (1965) and Desclés (1994), both building on Curry, and in a later translation project based on Montague grammar, Rosetta (1995).

It is not clear if Curry and Montague also supported vertical universality; Montague, at least, seems to have thought that higher-order predicate calculus with modal operators was sufficient for everything; more accurately, that the semantics of English consists of giving truth conditions to all English sentences in a model of this calculus.



care of operations of “book-keeping” character, such as inference rule application, substitution, and variable binding. The gain achieved for implementing logic on computers can be considerable.

The logical framework idea works for both constructive and classical mathematics. de Bruijn’s AUTOMATH is an early example (1967) for classical mathematics. That many recent applications focus on constructive mathematics is mainly motivated by the possibility of extracting executable programs from proofs (Nordström & al. 1990).

## 2.4 GF judgements and grammars

As for abstract syntax, GF is just another logical framework (in fact, very close to ALF (Magnusson & Nordström 1994). The abstract syntax part of a grammar consists of judgements of the following two forms:

$$\begin{array}{ll} \mathbf{cat} C & \text{--- } C \text{ is a category} \\ \mathbf{fun} f : A & \text{--- } f \text{ is a function of type } A \end{array}$$

(Yet another form is definitions of a functions, but we won’t need it.)

What is not present in logical frameworks, are judgements for defining concrete syntax. The most important forms are the following, needed for each category  $C$  and function  $f$  defined in the abstract syntax:

$$\begin{array}{ll} \mathbf{lincat} C = T & \text{--- } C \text{ has the linearization type } T \\ \mathbf{lin} f = t & \text{--- } f \text{ has the linearization function } t \end{array}$$

A *grammar* is a pair

$$\langle \mathcal{A}, \mathcal{C} \rangle$$

of abstract and concrete syntax. A *multilingual grammar* is a pair

$$\langle \mathcal{A}, \{\mathcal{C}_1, \dots, \mathcal{C}_n\} \rangle$$

of an abstract syntax and a set of concrete syntaxes.

## 2.5 Type checking, linearization, and parsing

A GF grammar is a purely declarative definition of a language. But the formalism is so defined that it is always possible to derive the following algorithms automatically:

*Type checking* decides whether a given tree in abstract syntax has a given type.

*Linearization* takes a tree in abstract syntax to an object in the corresponding linearization type.

*Parsing* takes a string into a set of abstract syntax trees (empty set: parsing fails; more than one elements: the string is ambiguous).

The type checking algorithm is inherited from logical frameworks. The linearization algorithm can be straightforwardly derived from the linearization rules, and it is similar to expression evaluation in  $\lambda$ -calculus and functional programming. The parsing algorithm is based on a non-trivial inversion of the linearization rules. Altogether, deriving these algorithms from a declarative source gives an implementation gain analogous to implementing logics in a logical framework.

As a corollary of linearization and parsing, a multilingual grammar automatically has

*Translation* from  $\mathcal{C}_i$  to  $\mathcal{C}_j$  is parsing in  $\mathcal{C}_i$  followed by linearization in  $\mathcal{C}_j$ .

The translation goes via  $\mathcal{A}$ , which guarantees semantic equivalence but does not preclude ambiguity.

## 2.6 Example: a fragment of arithmetic

We illustrate GF with a very small multilingual grammar, whose subject matter is arithmetic. The abstract syntax introduces the categories of natural numbers and propositions, the number zero, and the predicate that a number is even.

```
cat Nat
cat Prop
fun Zero : Nat
fun Even : Nat  $\rightarrow$  Prop
```

An English concrete syntax can be given as follows:

```
lincat Nat = {s : Str}
lincat Prop = {s : Str}
```

**lin** *Zero* = {*s* = "zero"}  
**lin** *Even x* = {*s* = *x.s* ++ "is" ++ "even"}

The linearizations are *records* which may have many different fields. In the simplest case, as here, there is just one field holding a string. As we will see, much of the difference between concrete syntaxes comes from what types of records are assigned to each category.

Using the abstract syntax above, we can form the tree

*Even Zero*

of type *Prop*. Using the concrete syntax, we compute the linearization of this proposition:

{*s* = "zero" ++ "is" ++ "even"}

## 2.7 Arithmetic in French

Abstract syntax abstracts away from features depending on concrete syntax. One of these features is *morphological variation*, which is very different in different languages. To describe morphological variation in the concrete syntax, we introduce a form of judgement permitting the definition of *parameter types*. For instance, even in a tiny fragment of arithmetic, the French concrete syntax will need the parameters of mood and gender:

**param** *Mod* = *Ind* | *Subj*  
**param** *Gen* = *Masc* | *Fem*

Expressions for natural numbers are, linguistically, noun phrases, and have an *inherent* gender, that is, a gender element belonging to the record as an extra field.

**lincat** *Nat* = { *s* : *Str* }  
                  { *g* : *Gen* }  
**lin** *Zero* = { *s* = "zéro" }  
                  { *g* = *Masc* }

An inherent gender in GF is a formalization of gender information such as given in a traditional dictionary.

Expressions for propositions are, linguistically, sentences. Their formation depends on a mood parameter, which is inherited by the verb of the sentence. In a predication sentence formed by using an adjective, the adjective is inflected in the gender of the subject noun phrase.

**lin**cat Prop = {s : Mod ⇒ Str}  
**lin** Even x =  
{s = **table** {m ⇒ x.s ++  
**table** { Ind ⇒ "est"  
Subj ⇒ "soit" } ! m ++  
**table** { Masc ⇒ "pair"  
Fem ⇒ "paire" } ! x.g}}

The linearization of a sentence is thus a *table* that gives a value to each of the parameters of a parameter type. This notion is a formalization of inflection tables occurring in traditional grammar.

## 2.8 Translating between English and French

The following examples (from an enlarged fragment) show how French uses the gender to generate different forms of the adjective.

*zero is even = zéro est pair*  
*the sum of zero and zero is even = la somme de zéro et de zéro est paire*

The following example shows a French construction requiring subjunctive mode of a subordinate clause.

*there exists an x such that x is even = il existe un x tel que x soit pair*

All the translations above are obtained via parsing into abstract syntax and subsequent linearization into concrete syntax. They work, of course, in both directions.

## 3 The unity of a language

### 3.1 Unity of languages lost

If we write the grammar of each language game separately, we will end up having no common structure in the sentences *zero is even*, *smallpox is contagious*, and *the weather is beautiful*, unless we find a language game that covers them all. This is counterintuitive: one would prefer to say that all of these expressions have the same syntactic structure, which is that of a sentence formed by adjectival predication.

A more general aspect of the problem is: if we can only describe language games, then there is no such thing as English or French language. This is counterintuitive: when I learn French, there is something unified that I learn—not just a set of distinct language games, but something that enables me to play any of my old language games in a new language. What is this thing?

### 3.2 Resource grammars

The way we can do justice to the unity of languages, in the ordinary sense, is by raising the level of abstraction in concrete syntax. Instead of defining linearization directly into strings and records of strings, we use intermediate syntactic structures. For instance, instead of defining

$$\mathbf{lin} \text{ Even } x = \{s = x.s ++ "is" ++ "even"\}$$

we use an adjectival predication function and a lexical item representing the adjective *even*:

$$\mathbf{lin} \text{ Even } x = \text{PredAP } x \text{ even}$$

These intermediate structures are collected into a *resource grammar*, whose aim is to define a natural language, such as English, instead of a language game. A resource grammar is built by considering the language independently of the various language games in which it might be used. Although it is defined in GF just like the semantically motivated structures, no semantic explanation is expected. On the other hand, the linearization rules of a resource grammar will be intuitive and immediate, since the grammar has been formulated by directly observing the concrete usage of the language.

### 3.3 Example resource grammar

The abstract syntax of a resource grammar is built from traditional linguistic categories and rules. In the GF resource grammar project (Ranta 2005), we have found it useful to have, among other things, sentences, noun phrases, and adjectival phrases.

$$\mathbf{cat} \text{ } S; NP; AP$$

The functions formalize rules that combine such phrases, e.g.

**fun** *PredAP* :  $NP \rightarrow AP \rightarrow S$

tells that a noun phrase and an adjective phrase can be combined into a sentence.

To give an English concrete syntax to this resource grammar, we first define the agreement features of noun phrases and verb phrases, constructed from number and person:

**param** *Num* =  $Sg \mid Pl$

**param** *Person* =  $P1 \mid P2 \mid P3$

**param** *Agr* =  $Ag \ Num \ Person$

Then we define the linearization types of syntactic categories and the linearization rules of the syntactic functions:

**lincat** *S* =  $\{s : Str\}$

**lincat** *NP* =  $\left\{ \begin{array}{l} s : Str \\ a : Agr \end{array} \right\}$

**lincat** *AP* =  $\{s : Str\}$

**lin** *PredAP* *np adj* =  $\{s = np.s ++ be! np.a ++ adj.s\}$

The adjectival predication rule uses the copula *be*, which is defined as an *auxiliary operation*:

**oper** *be* :  $Agr \Rightarrow Str$  = **table**  $\left\{ \begin{array}{ll} Ag \ Sg \ P1 & \Rightarrow \text{"am"} \\ Ag \ Sg \ P3 & \Rightarrow \text{"is"} \\ - & \Rightarrow \text{"are"} \end{array} \right\}$

Finally, the resource grammar may also have a *lexicon*, which defines the linguistic properties of words without paying notice to their semantics. For instance, the adjective *even* is unlikely to have a uniform semantic interpretation, but it does have a uniform shape as an adjective.

**fun** *even* : *AP* ; **lin** *even* =  $\{s = \text{"even"}\}$

**fun** *zero* : *NP* ; **lin** *zero* =  $\{s = \text{"zero"}\}$

### 3.4 Using resource grammars

In contrast to resource grammars, we use the term *application grammar* for a grammar describing a semantically interesting language game. To define the concrete syntax of an application grammar, we can now give its linearization types and rules in terms of the categories and functions of the resource. In English arithmetic, for instance, we define:

**lincat** *Prop* = *S* ; **lincat** *Nat* = *NP*  
**lin** *Zero* = *zero* ; **lin** *Even x* = *PredAP x even*

Structures and words in the resource grammar do not have a deeper semantic meaning. In a sense, their semantics is given indirectly, through the different uses to which they are put in different application grammars. For instance, the adjective *even* is used to linearize the concept of divisibility by 2 in some applications, and the concept of smoothness of a surface in others.

### 3.5 Sharing resource grammar

Even though concrete syntax differs a lot from one natural languages to another, there is surprisingly much of the abstract syntax of resource grammars that can be shared. For instance, the above fragment can be easily reused for French, by just redefining the linearization types and rules. The linearization types assigned to semantic types are the same as in English,

**lincat** *Prop* = *S* ; **lincat** *Nat* = *NP*

where of course the definitions of *S* and *NP* is differ from language to language. The French linearization rules are

**lin** *Zero* = *zéro* ; **lin** *Even x* = *PredAP x pair*

If we have a German implementation of the resource, we can write

**lin** *Zero* = *null* ; **lin** *Even x* = *PredAP x gerade*

The lexical items *zero*, *even*, *zéro*, and *pair null*, and *gerade* are only defined (except for name coincidents) in the resource lexica of each language. But the syntactic structure *PredAP* is defined for every language. On this abstraction level, we can thus conclude with the anonymous 12th century author that the difference between languages “arises from the difference of words and their formations”.

### 3.6 Differences in concrete syntax structure

We have seen that raising the abstraction level in concrete syntax makes it possible for different languages to share rules even there, and not only

in the abstract syntax. But of course, we should not expect this always to hold. A well-known example from translation is the verb *miss*. where translating between English and French requires swapping the subject and the object:

$$x \text{ misses } y = y \text{ manque à } x$$

Only the abstract syntax predicate *Miss* can be shared: concrete syntax uses different structures. The English and French linearization rules are

$$\text{lin } Miss \ x \ y = \text{PredVP } x \ (\text{ComplV2 } miss \ y)$$

$$\text{lin } Miss \ x \ y = \text{PredVP } y \ (\text{ComplV2 } manquer \ x)$$

In this example, it is no longer possible to say that just the words are different in the two languages.

## 4 Conclusion

We have not tried to define a universal grammar that is both vertical and horizontal, i.e. covers everything that can be said in any language. Vertical horizontality does not seem realistic even within one language, if one aims at the complete formalization of semantics.

However, we have found it possible to achieve horizontal universality in semantically limited domains, by using type-theoretical abstract syntax as interlingua of translation. Moreover, we have found a meager variant of vertical universality, on the level of resource grammars where we don't require semantics interpretation of the syntactic constructions.

## References

Curry 1963 de Bruijn (1967) (Descartes 1629) Desclés (1994), Dowty (1982). Gilson (1922) Gödel (1931) Harper & al., 1986 Kalish and Montague (1964) Landin 1967 Lyons (1968): Magnusson & Nordström 1994. Martin-Löf 1984. Montague (1974) Nordström & al. 1990 Ranta 2004 (Ranta 2005) Rosetta (1995). (Sapir 1929) Shaumyan (1965) (Wittgenstein 1953).