Fiat Lingua

Title: Designing an Artificial Language: Syntax

Author: Rick Morneau

MS Date: 07-26-1994

FL Date: 05-01-2019

FL Number: FL-00005C-00

Citation: Morneau, Rick. 1994. "Designing an Artificial Language: Syntax." FL-00005C-00, *Fiat Lingua*, <http://fiatlingua.org>. Web. 01 May 2019.

Designing an Artificial Language:

Syntax

by Rick Morneau

August, 1992
Revised July 26, 1994

1.0 INTRODUCTION

This essay is aimed at budding language designers who would like to learn something about syntax in general, and about some of the syntactic variability that exists among the world's many natural languages. It is also aimed at those who would like to have a tool that they can use to describe the syntax of their creations. By no means is this essay intended to be comprehensive - any such attempt would be so long that no one would want to read it. Besides, I've got a life to live. :-)

To get this thing off to a good start, I'll first talk a little about some of the major features of syntax as they actually exist in natural languages. Next, I'll discuss a formalism that you can use to describe the syntax of your artificial language (henceforth AL), and I'll illustrate its use by describing the syntax of a small fragment of English. Finally, I'll use the formalism to describe a simple but powerful syntax for a hypothetical AL.

2.0 LINGUISTIC TYPOLOGY AND BASIC WORD ORDER

A lot of work is currently being done in the area of linguistic typology and universals. Basically, this area of research tries to find patterns that exist across all natural languages. As it turns out, many universals are not truly universal, since exceptions to most patterns can often be found. These exceptions, however, often indicate that a language is undergoing a change from one pattern to another. Also, the exceptions themselves exhibit patterns indicating that the forms that they can take are controlled by even more subtle universals.

Consequently, these patterns or tendencies can often reveal a lot about how languages change and, more importantly, how change can be limited into following along certain channels. The forms that human languages can take are actually quite limited, and the study of universals can show us what these limitations are.

A study of linguistic universals can also be helpful to AL designers. This is especially true if you are trying to develop a language that is quite different from the natural languages you are familiar with. A study of universals can help you design a language that ends up being speakable

and learnable; i.e., one that is compatible with the grey stuff most of us have between our ears. If you "break the rules", so to speak, your language may end up being unlearnable in any real sense, and will end up being just a coding game rather than a real language.

2.1 SUBJECT, VERB, AND OBJECT

One of the first patterns that typologists look at is the basic relationship between subject (S), verb (V) and object (O) in simple declarative sentences. Determining these patterns is not always that simple, because many languages are inflected in such a way that they have a great deal of freedom in ordering their words. But even these languages will have some restrictions, or will tend to have dominant, preferred or unmarked word orders.

There are six possible orderings: VSO, SVO, SOV, VOS, OVS, and OSV. It turns out that a very large majority of the world's languages fit within the first three categories; i.e., where the subject comes before the object. Here are some examples:

```
SOV - Turkish, Tamil, Japanese, Tibetan, Quechua
      This is the largest single grouping, and probably
          accounts for slightly more than 40% of all
          languages.
      Sample sentence: John fish ate.

SVO - English, Swahili, Chinese, Indonesian
      This is also a very large group, although not quite
          as large as SOV.  It probably accounts for
          slightly less than 40% of all languages.
      Sample sentence: John ate fish.

VSO - Welsh, Hawaiian, Berber, Classical Arabic
      This is not a very large group, but it is still quite
          significant.  It probably accounts for about 15%
          of all languages.
      Sample sentence: Ate John fish.
```

Note that in all of the above, the subject comes before the object - only the verb's position changes. Also note that the percentages are very approximate. (Until recently, most linguists felt that SVO was the most common type. However, more recent knowledge seems to give SOV a slight lead.)

The other groups contain relatively few languages, and many of them are likely to be languages you've never heard of. Here are a few examples:

```
OVS - Guarijio (Azteco-Tanoan family, Mexico)
      Hixkaryana (Carib family, Brazil)
      Sample sentence: Fish ate John.
```

```
VOS - Fijian (Austronesian family, Fiji)
      Terena (Arawakan family, Brazil)
      Malagasy (Austronesian family, Madagascar)
      Sample sentence: Ate fish John.

OSV - Jamamadi (Arawakan family, Brazil)
      Language of Yoda, Jedi Master (Unknown language
          family, Dagobah star system)
      Sample sentence: Fish John ate.
```

Thus, if you want your AL to be as typologically "typical" as possible, you will need to make it SOV. If you want to make it similar to most European languages, then use SVO. However, if you don't want to play favorites but still want something that's easy to learn, you can choose the more neutral VSO format (which, incidentally, is my own personal favorite :-). VSO is also easier for human brains and computers to parse. Finally, if you're more interested in the exotic or the weird, choose one of the last three word orders.

2.2 RELATIVE CLAUSES

Relative clauses are embedded sentences that modify nouns. Consider the following English example:

```
1. The shirt (that) you want is on the bed.
```

In this sentence, the relative clause is "you want" and it modifies the noun "shirt". (The parentheses indicate that the relative pronoun "that" is optional.) There are two important observations we should make about this example: first, the word "shirt" is, in effect, the direct object of the verb "want"; and second, the verb "want" does not have an explicit direct object. The position where an object would normally go is called a _gap_.

It's also possible for relative clauses to modify nouns that correspond to positions other than direct object in the clause. Here are some examples:

```
2. The police caught the man who robbed the bank.
   (Here, "man" is the effective subject of the verb
      "robbed".)

3. This is the hammer (which) he broke the window with.
  =This is the hammer with which he broke the window.
   (Here, "hammer" is the effective object of the
      preposition "with")

4. They examined the room (which) the fire started in.
```

```
=They examined the room in which the fire started.
 (Here, "room" is the effective object of the preposition
    “in")
```

And so on. In English, any verb argument (i.e., subject, object, indirect object or object of a preposition) can be relativized, and when this happens, a gap is left in the clause. Incidentally, as illustrated in examples 3 and 4 above, English has the extremely rare and confusing habit of splitting a compound relative pronoun into two parts, moving the first half to the end of the clause, and optionally dropping the second half. Thus, we can have "in which" or "(which)...in". I strongly recommend that you NOT allow such splitting in your design if one of your goals is ease-of-learning. Either use single words that are inherently unsplittable, or keep the pieces together.

Gaps in relative clauses appear to be required by slightly less than half of the world's languages. A slight majority, however, either do not allow gaps or severely restrict them. For example, Palestinian and Egyptian Arabic allow gaps only if the noun being modified is the effective subject of the relative clause, as in example 2 above. For all other verb arguments, a gap is not allowed and must be filled by what is called a _resumptive pronoun_. Here is what Arabic versions of sentences 1, 3 and 4 would look like using English words and English word order:

```
1. The shirt that you want it is on the table.
3. This is the hammer that he broke the window with it.
4. They examined the room that the fire started in it.
```

Note that in all three sentences, the resumptive pronoun "it" is required for the Arabic sentences to be grammatical.

[Incidentally, the above applies to Egyptian and Palestinian Arabic. In Standard Arabic, the use of resumptive pronouns is optional. However, in Standard Arabic, the relative pronouns are inflected for gender and number. In effect, the resumptive pronoun is built into the relative pronoun.]

In other languages, such as Irish, the use of resumptive pronouns is optional. Here, though, the relative pronoun that introduces the clause (such as "who", "which", "that", "in which", "with which", etc.) differs depending on whether the clause contains a gap or a resumptive pronoun. For example, languages like this will have two words or phrases for "in which": one for use when the clause contains a gap, and the other for use when the clause contains a resumptive pronoun. Unfortunately, I have no information on how widespread this phenomenon is.

Finally, a very small minority of languages, such as Persian, not only allow the use of resumptive pronouns, but also allow gaps to be filled by modified or coordinated nouns. Thus, the following sentence would be perfectly grammatical in Persian:

```
The police caught the man who he and his wife robbed the
  bank.
```

However, this usage seems to be very rare.

2.3 THE NOUN PHRASE

Another ordering that typologists are concerned with is the relationship between adjectives (A) and the nouns (N) they modify in noun phrases. (I also include simple and complex numbers within the adjective group, such as "five", "between ten and twenty", etc.) Since we're only talking about two items, "A" and "N", there are only two ways to order them: AN or NA. However, we can complicate things by including so-called **heavy modifiers** such as relative clauses, descriptive prepositional phrases ("girls with red hair"), and prepositional arguments of nominalized verbs ("destruction of the city"). I will include all such heavy modifiers in the same category as relative clauses (R), since they all seem to pattern in the same way. Here are some examples:

```
NAR - Thai, French, Hebrew, Swahili
       Sample phrase: men big who eat quiche

ANR - Quechua, English, Persian, Russian
       Sample phrase: big men who eat quiche

ARN - none
       Sample phrase: big quiche eat who men

NRA - none
       Sample phrase: men who eat quiche big

RNA - Basque, Abkhaz, Burmese
       Sample phrase: quiche eat who men big

RAN - Turkish, Tamil, Korean
       Sample phrase: quiche eat who big men
```

In the sample sentences, I have placed the object "quiche" and the relative pronoun "who" in their most likely positions, based on the most likely branching direction for the type (I'll have more to say about branching later). It can, however, appear before or after the relative clause. In fact, in some languages, such as Hindi and Bengali, the relative pronoun appears before the noun being modified and the clause appears after it! For languages like these, our sample sentence would look something like this: "big who men quiche eat". (For purists or students of Hindi who are reading this, the clause must also be followed by a demonstrative pronoun, which actually makes the gloss more like "big who men quiche eat that".)

Quite a few languages, such as Turkish and Quechua, do not use relative pronouns, but instead nominalize the verb (i.e., convert it to a participle) in the relative clause to achieve the same

effect. The resulting participle can be inflected for tense. Thus, an English phrase such as "boys who broke windows" would be glossed in Turkish as "windows broking boys". Neat, huh?

Notice that **no** languages place a relative clause between an adjective and a noun (types ARN and NRA). I believe that this is an honest-to-god linguistic universal. Disregard it at your own peril.

Incidentally, it is important to note that the above discussion applies only to languages that **have** adjectives. Since many languages use stative verbs instead of adjectives, "R" and "A" would be part of a single category. Some examples of this are Indonesian (where modifiers follow the noun), and Chinese (where modifiers precede the noun). Thus, an English phrase like "that stupid boy who breaks windows" would be glossed in Indonesian as "boy which be_stupid which break windows". In Chinese, it would sound something like "break windows which be-stupid which boy". (I have much more to say about stative verbs in my (very long!) monograph Lexical Semantics.)

The other components of the noun phrase that are of interest to typologists are called _specifiers_, and include articles (***the, a, an***), demonstratives (***this, those***, etc.), quantifiers (***each, all, every***, etc.) and possessives (***my, their, John's***, etc.). These are called specifiers because they precisely pinpoint one or more referents from among a set of possible referents. Adjectives and numbers, however, only narrow down the number of choices by creating a subset, but without specifying particular referents. Consider, for example, the difference between "big black dogs" and "those big black dogs".

Specifiers undergo the same variability in word order as adjectives with one exception: it **is** possible for a relative clause to appear between a specifier and the noun it modifies. Thus, even though types ARN and NRA do not seem to exist, the types SRN and NRS (where S = specifier) do occur. In Indonesian, for example, the expression "those men who just left" would appear as "men who just left those". Aside from this idiosyncrasy, specifiers can appear either before an adjective (as in English), after an adjective (as in Vietnamese), or in either position (as in Swahili).

Incidentally, relative clauses and prepositional phrases can act semantically as either general modifiers ("boys who like basketball") or specifiers ("boy with the scar over his left eye"). However, I'm not aware of any language that reflects this difference in syntax.

Finally, I hope I haven't given the impression that I've covered all of the possible forms of noun phrase and relative clause that can exist in natural language. We've seen a few unusual cases, such as English's splittable relative pronouns and Persian's odd resumptive structure. But this is only the tip of the iceberg.

2.4 BRANCHING DIRECTION

There are certain ordering relationships that are so common, and which have so few exceptions that they almost certainly indicate something very basic about the way our brains process language. In fact, when languages are found that appear to contradict such orderings, it is usually

the case that they are undergoing a transition from one pattern to another and haven't yet "settled down".

For example, in VSO languages, the subject and object follow the verb (by definition). But we also find that specifiers, adjectives, genitives and relative clauses almost always follow the nouns they modify, that adverbs and adjectival arguments almost always follow the adjectives they modify, and that noun phrases almost always follow the prepositions that govern them. In other words, with very few exceptions, modifiers and arguments almost always FOLLOW the words they modify or are governed by in VSO languages. (Linguists would say that modifiers follow their _heads_ in VSO languages.) For SOV languages, the principle applies just as rigorously, but in the opposite direction; i.e., modifiers and arguments almost always PRECEDE the words they modify, and the equivalent of prepositions (called _postpositions_) are preceded by the noun phrases they govern. In other words, modifiers precede their heads in SOV languages. Another common way of describing this ordering is to say that VSO languages are predominantly right-branching, and that SOV languages are predominantly left-branching.

Unfortunately, the relationship breaks down when we consider SVO languages, such as English. Are these languages undergoing a transition from one form to another, or have they become stuck in some kind of cul-de-sac that is neither left nor right-branching? For languages with long written histories, we can often answer this question, and the answer seems to be that SVO is transitional. For example, Old English (spoken before 1200 AD) was SOV and heavily inflected. It lost most of its inflections and made the switch to SVO at about the same time.

Furthermore, a lot has been learned recently about word order patterns. As a result, linguists can often make intelligent guesses about where SVO languages came from, simply by looking at where they are now. Predicting the future, however, is essentially impossible. Languages certainly have inertia, but it's doubtful they have momentum.

There are some things, though, that seem to be fairly certain. All natural languages are constantly changing - not just in word order, but also in ways that I haven't discussed here, such as in phonology, morphology and semantics. Changes in word order, though, seem to be bounded by two "pure" or "ideal" endpoints: SOV and VSO. This probably explains why there are so few truly pure VSO and SOV languages. Since all languages are constantly changing, it's much more likely that any particular language will be wandering around somewhere between the endpoints.

The point for AL designers is this: As a language designer, you have total control over what your language will be. But even you cannot predict where your language will go. And since it will be a NEW creation, it cannot have a history. The only freedom you have is to decide where it will start.

3.0 A FORMALISM FOR DESCRIBING SYNTAX

Originally, it had been my intent to provide a brief description of a few of the most widely used linguistic formalisms, and to show how they could be applied to the description of an artificial language. I actually started work on this project using Government/Binding Theory, which is

currently the most popular formalism among professional linguists. However, I quickly realized that even a _brief_ description would require several dozen pages for each theory, and I doubted very much if many people would have the fortitude to read it all. So, I had to come up with something simpler. Fortunately, we don't really need to delve into current linguistic theories, since they must deal with the complexities and oddities of natural languages. ALs tend to be simpler and more regular. Besides, and perhaps more importantly, current linguistic theories are continuously moving targets, always subject to change without notice.

So, I'm going to limit myself to a somewhat restricted view of syntax. First of all, I will not discuss agreement/unification aspects of syntax, since these are normally handled with inflection which may not even be present in your AL. Also, inflection is a morphological process, and representing it along with word-order would be messy, time-consuming, boring, and not very useful. Thus, I will limit myself primarily to a discussion of word order and how to represent it.

With this goal in mind, I will describe how to use simple, context-free, phrase structure rules. The notational system I will use will be extremely simple - I will use a modified version of Backus-Naur Form (BNF) because it is more powerful and less confusing than the system normally used by linguists. Those among you who have worked with BNF will not learn anything new here, especially if you've ever studied or worked in compiler design. You may find it somewhat odd, though, to see BNF applied to human language.

Finally, I apologize to those readers who might be expecting a more up-to-date treatment of syntax in terms of X-Bar Theory. Although X-Bar Theory provides desirable constraints on the syntax of a NATURAL language, it is highly abstract and I was afraid that a lot of readers would be turned off if I used it. Also, it's quite possible that some AL designers may want to design a language that is NOT constrained as natural languages are. Phrase structure rules will give them this freedom - X-Bar Theory will not. However, I have not entirely abandoned X-Bar Theory, since my sample syntax in section 4 was designed within the framework of X-Bar Theory, even though I do not discuss it in those terms.

3.1 THE SYNTAX OF A SIMPLE ENGLISH FRAGMENT

Our first goal will be to learn how to use BNF to describe a simple subset of English. I will not try to develop a complete BNF description of the entire English language because it is not necessary, it would take too long, and I don't think it's possible. Instead, I'll only work with a rather small subset of English, one that is no more complex than is absolutely necessary to illustrate all of the features of BNF needed to design an AL syntax. To achieve this goal, I will start with very simple sentences and gradually increase their complexity.

Incidentally, my purpose here is to illustrate BNF - NOT to teach linguistics! As a result, some of the analyses below are not as linguistically precise as I would like, but to do the job right would have taken much more time and would have required lots of digression into areas that are not really relevant here.

For starters, consider the following simple sentences:

```
Sailors cuss.
Dogs bark.
Billy fell.
```

These sentences consist of two words each: a noun and a verb. To describe the structure of these sentences in BNF, we would write:

```
sentence ::= noun verb
```

which can be read as "a sentence consists of a noun immediately followed by a verb". However, verbs can also have objects, as in the following examples:

```
Children like puppies.
Louise kissed Jimmy.
Tornadoes destroy buildings.
```

We can extend the syntax to deal with direct objects as follows:

```
sentence ::= noun verb (noun)
```

which can be read as "a sentence consists of a noun followed by a verb which may in turn be followed by an optional noun". Thus, in our BNF notation, we will use parentheses to indicate that an item is optional.

Now, the nouns that precede and follow the verb can be more complex than shown above, as in the following sentences:

```
Children like cute little puppies.
Silly dogs bark.
Little Billy stutters.
```

For these cases, the noun is modified by one or more adjectives. In other words, one or more adjectives are optional. In BNF notation, we would write:

```
sentence ::= {adjective} noun verb ({adjective} noun)
```

Here, the curly braces mean "zero or more". Thus, our definition of a sentence now reads: "a sentence consists of zero or more adjectives followed by a noun, then a verb, and then an optional 'thing' which itself consists of zero or more adjectives followed by a noun".
Now, let's make things a little more complicated by adding articles:

```
The little boys saw a big angry dog.
A sad little girl watched the birds.
An angry man shouted.
```

We can add the articles "a", "an" and "the" to our BNF representation as follows:

```
sentence ::= (article) {adjective} noun verb ((article)
 {adjective} noun)
```

Note though, that we are repeating ourselves in that the same "thing" appears on both sides of the verb. It consists of an optional article, zero or more adjectives, and a noun. As it turns out, this "thing" has a name - it's called a _noun phrase_. Let's take it out of the definition of sentence and describe it separately, as follows:

```
sentence ::= noun_phrase verb (noun_phrase)

noun_phrase ::= (article) {adjective} noun
```

Our syntax now reads like this: A sentence consists of a noun phrase followed by a verb followed by an optional noun phrase. A noun phrase consists of an optional article followed by zero or more adjectives, followed by a noun.

English allows demonstrative adjectives ("this", "that", "these" and "those"), possessive adjectives ("my", "his", "our", "their", etc.) and quantifiers ("each", "both", "every", "all", etc.) to appear in place of an article, as in the following sentences:

```
My mother read this book.
Her angry dog bit Danny.
That cat ate both mice.
Every good student does his homework.
```

We can deal with this by redefining the noun phrase, as follows:

```
noun_phrase ::= (specifier) {adjective} noun

specifier ::= article | demonstrative | possessive |
 quantifier
```

Here, the vertical bar "|" can be read as "or". Thus, a specifier can be an article or a demonstrative or a possessive or a quantifier. Also, if we really want to be thorough, we can refine the definition of possessives as follows:

```
possessive ::= possessive_adjective | proper_noun's
```

which can be read as: a possessive can be a possessive adjective (such as "his", "your", "my", etc.) or a proper noun followed by apostrophe-s (such as "John's", "Boston's", "IBM's", etc.)

Next, let's add numbers to our syntax. In English, numbers can only appear between a specifier and an adjective, as in the following noun phrases:

```
all five boys
these three girls
```

```
the two fat lazy cats
our seven goldfish
```

So, to account for numbers in English, we have to modify our definition of the noun phrase as follows:

```
noun_phrase ::= (specifier) (number) {adjective} noun
```

Now, what do we do about pronouns? As it turns out, an entire noun phrase can be replaced by a single pronoun, as illustrated in the following:

```
John and Billy ate the whole apple pie. They ate it.
    ("They" = "John and Billy", "it" = "the whole apple
      pie")

Five angry dogs chased those three foolish boys. They chased
 them.
    ("They" = "Five angry dogs", "them" = "those three
      foolish boys”)
```

And so on. In English, pronouns cannot be directly modified by articles (*the she), demonstratives (*that it), possessives (*her him), quantifiers (*each she) or numbers (*five they). In other words, an English pronoun stands alone and is equivalent to an entire noun phrase. Thus, to account for pronouns, we have to redefine the noun phrase as follows:

```
noun_phrase ::= pronoun | modified_noun

modified_noun ::= (specifier) (number) {adjective} noun
```

Now, let's summarize by showing the entire syntax of the fragment of English that we've dealt with so far:

```
sentence ::= noun_phrase verb (noun_phrase)

noun_phrase ::= pronoun | modified_noun

modified_noun ::= (specifier) (number) {adjective} noun

specifier ::= article | demonstrative | possessive |
 quantifier

possessive ::= possessive_adjective | proper_noun’s
```

In all, we've got five definitions. Each of these definitions is called a _production rule_, since it defines how a structure within the language can be produced. Thus, our fragment of English consists (so far) of five production rules.

However, we've still got a long way to go. By now, though, you should be getting the idea. So let's speed things up a bit by handling two additional items all at once: indirect objects and prepositional phrases, as in the following example:

```
John gave the children candy in the back of the bus on the
 way to the park.
```

Handling the indirect object "the children" is easy. We simply add an optional noun phrase to our definition of sentence:

```
sentence ::= noun_phrase verb ((noun_phrase) noun_phrase)
```

Note that the indirect object is doubly nested inside the parentheses, indicating that it must precede the direct object, and that an indirect object cannot occur without a direct object.
The prepositional phrases are not so easy, since we are really talking about two different kinds of phrase: a sentential prepositional phrase and a noun-modifying prepositional phrase. The phrase "in the back" indicates where the action took place. The phrase "of the bus" simply further defines the noun "back". Thus, the phrase "in the back" is a sentential prepositional phrase, while "of the bus" is a noun-modifying prepositional phrase. Similarly, "on the way" is sentential since it indicates when the action took place, while "to the park" modifies the noun "way". We can handle these phrases as follows:

```
sentence ::= subject verb (objects) {prepositional_phrase}

subject ::= noun_phrase

objects ::= (noun_phrase) noun_phrase

noun_phrase ::= pronoun | modified_noun

modified_noun ::= (specifier) (number) {adjective} noun
                     {prepositional_phrase}

prepositional_phrase ::= preposition noun_phrase
```

Note that I added the constituents "subject" and "objects" to make things a little easier to read. To keep things simple, we will consider compound prepositions such as "from under", "up to", "on top of", etc. as if they were single words.

Now, if you look carefully, you'll see that something unusual is happening here. A noun phrase can contain a prepositional phrase which, in turn, contains another noun phrase. This kind of circularity is called _recursion_, and is one of the features of language that makes it so flexible. Basically, recursion occurs when a lower level structure, such as a prepositional phrase, is defined in terms of a higher level structure, such as a noun phrase.

Another example of recursion is the embedded sentence, as illustrated in the following examples:

```
He told me (that) he wanted a new job.
That he needed so much money worried his friends.
Bill knew (that) she broke the window.
He told me (that) Bill knew (that) she broke the window.
```

Note that an embedded sentence can never occur as an indirect object - it can only be either the subject or direct object. Note also that the conjunction "that" is required for an embedded sentence that is the subject of a verb, even though it is optional for the direct object. Thus, we can represent this kind of embedded sentence as follows:

```
subject ::= noun_phrase | "that" sentence

objects ::= (noun_phrase) direct_object

direct_object ::= noun_phrase | ("that") sentence
```

Another recursive structure is the relative clause. Consider the following sentences:

```
The boy who broke the window apologized.
I saw the man who robbed the bank.
The textbook (that/which) he bought had a chapter on
 linguistics.
John played the piano (that/which) his brother gave him.
```

Note that there are two types of relative clause shown above. In the first type, the relative pronoun "who" links a noun phrase to the subject of the clause. Thus, "the boy" is the effective subject of "broke", and "the man" is the effective subject of "robbed". In this type, the relative pronoun is required. In the second type, the relative pronoun links a noun phrase to the object of the direct clause. Thus, "the textbook" is the effective direct object of the verb "bought", and "the piano" is the effective direct object of the verb "gave". In this type, the relative pronoun is optional.

Note also that both nouns and pronouns (e.g., "He who laughs last laughs best") can be modified by relative clauses. Thus, the addition to our syntax is at a very low level, as follows:

```
noun_phrase ::=
   modified_noun
 | pronoun (relative_clause)

modified_noun ::=
   (specifier) (number) {adjective} noun
   {prepositional_phrase} (relative_clause)

relative_clause ::=
```

```
   relative_pronoun verb (objects) {prepositional_phrase}
 | (relative_pronoun) subject verb direct_object
   {prepositional_phrase}
```

where a relative pronoun can be either "that", "who" or "which". I will not discuss how to handle other types of relative clause that can be formed with relative pronouns such as "whose", "with whom", "to which", etc., but will leave it as an exercise for the interested reader.

Anyway, you should now have a pretty good idea about how to describe the syntax of an AL using BNF. I won't go any further with my analysis of English, since things are getting kind of messy already, and I don't think there's much to gain by going any further. For those who are interested, here's a summary of the syntax of the English fragment that we've just analyzed:

```
sentence ::= subject verb (objects) {prepositional_phrase}

subject ::= noun_phrase | "that" sentence

objects ::= (noun_phrase) direct_object

direct_object ::= noun_phrase | ("that") sentence

prepositional_phrase ::= preposition noun_phrase

noun_phrase ::=
   modified_noun
 | pronoun (relative_clause)

modified_noun ::=
   (specifier) (number) {adjective} noun
   {prepositional_phrase} (relative_clause)

specifier ::= article | demonstrative | possessive |
      quantifier

possessive ::= possessive_adjective | proper_noun's

relative_clause ::=
   relative_pronoun verb (objects) {prepositional_phrase}
 | (relative_pronoun) subject verb direct_object
   {prepositional_phrase}
```

4.0 THE SYNTAX OF A HYPOTHETICAL ARTIFICIAL LANGUAGE

In this section, I'll describe a simple yet highly effective syntax for an AL. It encompasses all of the basic features, including recursion, needed for a language to be totally functional. The fact that it is so simple, however, means that this language will depend heavily on the lexicon to provide capabilities that are sometimes provided by syntax. For example, instead of changing word order and using auxiliaries to change a statement to a question (as is done in English), this language will simply append a question particle to the end of the sentence (as is done in Japanese). In other words, the syntax will be simple at the expense of the lexicon. As it turns out, this approach is inherently more flexible because the lexicon is infinitely expandable while the syntax is not.

Unfortunately, simplicity has a price, and that price is boredom. The syntax I am about to show you is extremely and undeniably boring. However, I am only trying to illustrate a minimal configuration. I'm sure that most AL designers would want to expand upon it and come up with a design that's a little more exciting.

4.1 BASIC PRODUCTION RULES

My sample language will be purely right-branching. That is, basic word order will be verb-subject-object (VSO), and arguments, modifiers and specifiers will always follow their heads (and the head of a sentence will always be a verb - by definition). I choose VSO mainly because it appeals to me and because it illustrates structures that are different from English. I also like it because it is inherently easier to parse, although this is not all that important unless you plan to write computer programs to parse it. Other than this, there are no inherent advantages or disadvantages over other word-orders. As we've already seen, every possible basic word-order (VSO, SVO, SOV, VOS, OVS and OSV) has counterparts among natural languages, and one is not inherently "better" than the others.

The form of the sentence in the sample language will be:

```
sentence ::= verb {verb_modifier} {verb_argument}
 {sentence_particle}

verb_modifier ::= adverb | tense_marker | etc.
```

Verb modifiers would be words equivalent to English adverbs, such as "quickly", "tomorrow", "just", etc. They would also handle the tense and aspect of the verb, such as past progressive ("he was going"), future perfect ("he will have gone"), simple present ("he goes"), etc.

Sentence particles would be used to modify the nature of the sentence. For example, they could convert a statement to a question or command, or they could indicate the attitude of the speaker towards what he is saying.

Verb arguments would correspond to English subjects, objects and objects of prepositions, and would take the forms of noun phrases, adjective phrases, and embedded sentences:

```
verb_argument ::= (case_tag) expression {argument_particle}

expression ::= noun_phrase | adjective_phrase | sentence
```

A case tag would be equivalent to many English prepositions that introduce verbal arguments, such as "to" in "I went to Boston". Note, though, that they can also introduce subordinate clauses, as in "I saw her when I was in Boston". Here, "when" is the case tag. A case tag is optional if the argument is the subject or object of the verb; i.e., it is optional if it is part of the argument structure of the verb.

The argument particle would perform a function similar to the sentence particle, but will apply only to the argument it follows. For example, it can convert the argument to an interrogative or add emphasis to it. Let me illustrate what we've got so far, using English words:

English	Sample language
I like Boston.	Like I Boston.
I went to Boston.	Go did I to Boston.
He doesn't know I went to Boston.	Know not he go did I to Boston.
Doesn't he know I went to Boston?	Know not he go did I to Boston huh?
Which book did he buy?	Buy did he book which?

where "did" is a verb modifier indicating that the preceding verb is in the simple past tense, "huh" is a sentence particle indicating that the entire sentence is to be interpreted as an interrogative, and "which" is a particle that converts the preceding noun phrase to an interrogative. Note that, for all particles, the scope of the particle will depend on the particular particle. In other words, each particle will define its own scope. Note also that a particle inherently terminates the item it applies to, unless it is immediately followed by another particle that applies to the same item.

In case you're wondering, I'm making a basic assumption about the nature of verbs in the above structure. The verb basically has two kinds of arguments: those like the subject and object that are required by the verb, and those that are marked by case tags (prepositions in English, postpositions in Korean, inflections in Hungarian, etc.). Arguments that are required by a verb are part of the valency and thematic structure of the verb. (I have much more to say about this topic in my (very long!) monograph on Lexical Semantics.)

Now, let's fill in some of the blanks:

```
noun_phrase ::= noun {simple_noun_modifier}
 {complex_noun_modifier}
```

```
simple_noun_modifier ::=  adjective_phrase | number
                        | article | demonstrative
                        | possessive | quantifier

complex_noun_modifier ::=  noun_phrase_tag noun_phrase
                        | noun_clause_tag sentence

adjective_phrase ::= adjective {adjective_modifier}
 {adjective_argument} {adjective_particle}

adjective_modifier ::= adverb {adverb_particle}

adjective_argument ::= adjective_phrase_tag noun_phrase
```

Note that noun phrase tags are the equivalent of English prepositions in constructions such as "Bring the book with the red cover" or "Get the book on the table".

Noun clause tags are relative pronouns that introduce relative clauses. For the embedded sentence, you can require that it contain a gap or a resumptive pronoun, as we discussed earlier. Take your pick. The above syntax allows the use of resumptive pronouns, but does not require them.

Note that this syntax DOES allow the type of relative clause that we discussed earlier and which occurs in Persian:

```
The police caught the man who he and his wife robbed the
  bank.
```

Whether or not you implement it in your own design is entirely up to you. Keep in mind though, that if you decide to disallow it, you will have to add to the existing production rules of your language. In other words, by disallowing a structure that makes perfectly good linguistic sense, your syntax will actually become more complex.

Adjective arguments would handle cases like English "red with fever" and "blowing in the wind". Note, though, that this syntax allows these arguments to be used more directly than in English. For example, if English had a counterpart to this form, one could make a sentence like "The blowing in the wind kite hit him on the head".

Adjective and adverb particles would include emphasizers and de-emphasizers such as English "very", "rather", "not too", "quite", etc.

Word order is somewhat freer than English, especially in simple noun modifiers. This is intentional. It allows continuous refinement without the need for clumsy prepositional phrasing, as is the case in English. For example, the expression "his three of those five black dogs" would

be rendered as "dogs black five those three his", which, with English word order, would sound like "his three those five black dogs".

Finally, note that case tags, noun phrase tags, noun clause tags and adjective phrase tags all correspond to English prepositions. However, each will be distinct, unlike English prepositions which often fill multiple roles, and which often lead to ambiguities. I'll have more to say later about such ambiguities and how to prevent them.

4.2 GENERAL RULES

Sometimes it's better to separately state a rule that applies to all of the structures of a language, rather than clutter up the production rules. For example, coordination is best described as a general rule of the form:

```
COORDINATION RULE:

    Where it makes sense, any constituent may be replaced by a
    coordinated constituent (using "and", "or", "but not",
    etc.). Thus, for a constituent "X", the following applies:

    X ::= X {coordinating_conjunction X}
```

where a coordinating conjunction can be "and", "or", "but", "but not", etc. With this general rule, our sample language can now handle constituents such as "John and Bill", "She washed and I dried the dishes", "He bought the steaks at Joey's Market or at Safeway", "She prepared the drinks and I washed the dishes.", etc. Coordination poses some special problems for syntax, and I'll have more to say about it later.

Another rule involves the use of particles:

```
PARTICLE RULE:

    Where it makes sense, any constituent may be followed by one
    or more particles that adds to the meaning of the
    constituent and/or terminates it.
```

This rule allows us to unclutter the production rules defined in the previous section by eliminating all references to particles, and also gives us the flexibility of adding particles where they were not explicitly allowed.

Another very useful general rule allows us to create complex structures that are the syntactic equivalents of lexical compounds. Consider the following English sentences:

```
    They held an over-the-fence conversation.
    Made-in-USA and built-with-confidence labels must be
        prominently displayed.
```

```
He needed a powerful pick-me-up.
Mercenaries-for-hire ads appeared in several places in the
    magazine.
```

In order to implement these hyphenated complexes, we could create the following general rule:

```
COMPLEX STRUCTURE RULE:

    Complex nouns, verbs, adjectives and adverbs can be created
    with an introductory tag followed by a verb argument, as
    follows:

        complex_X ::= X_tag verb_argument

    where X can be "noun", "verb", "adjective" or "adverb". For
    example:

        complex_adjective ::= adjective_tag verb_argument

    A complex X can be used wherever an X appears in the syntax.
    For example, a complex adjective can appear in the place of
    an adjective.  Also, as with other structures, a terminating
    particle can be appended if necessary.
```

Note that these complex structures are similar in nature to compound words. The main difference is that they have precise structure and can be as long as you want. Also, if your morphology is self-segregating (as I discussed in my earlier essay on morphology), then you could remove the spaces between the words in the complex, creating in effect a single, polysynthetic word.

4.3 THE ULTIMATE SYNTAX FOR AN AL

Here is a complete listing of the production rules and general rules discussed above:

```
sentence ::= verb {verb_modifier} {verb_argument}

verb_modifier ::= adverb | tense_marker | etc.

verb_argument ::= (case_tag) expression

expression ::= noun_phrase | adjective_phrase | sentence

noun_phrase ::= noun {simple_noun_modifier}
 {complex_noun_modifier}

simple_noun_modifier ::= adjective_phrase | number
```

```
                          | article | demonstrative
                          | possessive | quantifier

    complex_noun_modifier ::= noun_phrase_tag noun_phrase
                           | noun_clause_tag sentence

    adjective_phrase ::= adjective {adjective_modifier}
     {adjective_argument}

    adjective_modifier ::= adverb

    adjective_argument ::= adjective_phrase_tag noun_phrase
```

COORDINATION RULE:

Where it makes sense, any constituent may be replace by a coordinated constituent (using "and", "or", "but not", etc.). Thus, for a constituent "X", the following applies:

```
X ::= X {coordinating_conjunction X}
```

PARTICLE RULE:

Where it makes sense, any constituent may be followed by one or more particles that adds to the meaning of the constituent and/or terminates it.

COMPLEX STRUCTURE RULE:

Complex nouns, verbs, adjectives and adverbs can be created with an introductory tag followed by a verb argument, as follows:

```
    complex_X ::= X_tag verb_argument
```

where X can be "noun", "verb", "adjective" or "adverb". A terminating particle can be appended if necessary.

4.4 SAMPLE SENTENCES

Here are some sample sentences that illustrate the above rules. The "a" sentences are normal English sentences. The "b" sentences are the equivalents of the "a" sentences using the new syntax:

```
a. The dog bit the boy.
b. Bite did dog the boy the.          (did = past tense verb
                                      modifier)

a. Did the dog bite the boy?
b. Bite did dog the boy the huh?      (huh = sentential
                                      question particle)

a. The book is on the table.
b. Be book the on table the.          (on = "place on" = case
       OR                             tag)
b. Be_on book the table the.          (be_on = transitive
                                      verb)

a. Did the boy who broke the window
   run away?
b. Run_away did boy the who break     (who = agentive noun
   did window the huh?                clause tag)


a. I hate reading technical manuals.
b. Hate I read (I) manual technical
   lots.                              (lots = plural
                                      specifier)
```

For those of you who are uncomfortable with this word order, you can get something closer to English by moving verb modifiers, simple noun modifiers and adjective modifiers before the word they modify, rather than after. In other words, all light modifiers will precede their heads, and all heavy modifiers will follow them. You'll still have VSO word order, but it will be less "pure", and modification will be more like English.

5.0 MISCELLANEOUS TOPICS IN SYNTAX

In the following sections, I will discuss some of the loose ends that would not have fit well in any of the preceding sections.

5.1 COORDINATION

Coordination is the linking together of two or more constituents with the same structure, creating, in effect, a single complex structure. The linking elements are called _coordinating conjunctions_. A language can have simple coordinating conjunctions, such as English "and", "or" and "but", as well as compound ones, such as English "either...or...", "both...and..." and "not only...but also...and finally...".

My reason for discussing coordination is that coordinated structures are often ambiguous, and you may want to take steps to prevent such ambiguity in your design. Consider, for example, the following sentence:

```
John wants to buy the painting of the fireplace and the
  Persian carpet.
```

Does John want to buy one item (a painting in which one can see both a carpet and a fireplace), or two items (a carpet and a painting)? Note that the ambiguity is present only in the written sentence. When spoken, a different stress, timing and intonational pattern (i.e., a different _prosodic_ pattern) would be used for each meaning.

As a language designer, you must decide whether this kind of ambiguity is to be allowed in your AL. If your major goal is to create a _spoken_ language, then you may feel secure in ignoring the problem, and let context and prosody resolve potential ambiguities. If, however, the speakers of your AL will have different native languages, then a solution based on prosody may not be practical, since natural languages differ considerably in the ways they use intonation, stress and timing. You can, of course, define the prosodic features of your AL and force students to learn them, but in doing so, you will make your AL much harder to learn.

An additional consideration could be whether or not you want your AL to be computer-tractable. That is, are you trying to design an AL that is readily amenable to analysis on a computer? If you do want your AL to be computer-tractable, then you don't really have a choice. Unfortunately, computers are not very good at resolving ambiguity, whether spoken or written.

So, assuming you do NOT want coordination ambiguities in your AL (for whatever reason), how can you prevent them? As it turns out, one possible solution becomes readily apparent if we slightly paraphrase our problem sentence, as follows:

```
John wants to buy both the painting of the fireplace and
  the Persian carpet.
```

OR

```
John wants to buy not only the painting of the fireplace
  but also the Persian carpet.
```

In other words, use of complex coordinating conjunctions is inherently less ambiguous, because each constituent that is being linked is explicitly identified.

Before presenting a general solution, however, let's look at what happens when one coordinated structure is embedded inside another one. Consider the following example:

```
John wants to buy the painting of the fireplace and the
  Persian carpet or the rocking chair.
```

Obviously, when you embed a coordinated structure inside another one, the potential for ambiguity becomes even greater. Thus, if we want to eliminate any ambiguity, we must mark both the beginning and the end of a coordinated structure. Furthermore, if three or more items are coordinated, the inner one(s) cannot be marked in the same way as the outer ones. So, if we wish to eliminate all possible ambiguity, regardless of the number of constituents or the depth of embedding, then each conjunction must have three forms: a start form, a continuation form and an end form. There are many ways to implement this. An easy one to learn would be to have a root morpheme to indicate the basic meaning of the conjunction, and an affix to differentiate the three forms. For example:

```
bi- "begin" affix     ko- "continue" affix

no affix - "end"      and let "kwa" = “and"
```

and our new coordination rule would look like this:

```
X ::= (coordinating_conjunction) X
  {coordinating_conjunction X}
```

Examples:

```
apples and oranges = (bikwa) apples kwa oranges

apples and oranges and pears and plums =
   (bikwa) apples kokwa oranges kokwa pears kwa plums
```

As shown in the examples, you can specify in your syntax that the initial "bikwa" is optional unless it is needed to prevent an ambiguity. This has two implications: 1. "bikwa" must ALWAYS be used to commence an embedded coordinated structure; 2. when "kwa" appears alone, it always links the constituent that immediately follows it with the constituent that MOST CLOSELY precedes it. (Incidentally, note that the affixes can be re-used in creating other words. In fact, the "bi-" affix is very similar to Esperanto's "ek-" affix.)

Now, if these words existed in English, here's how we would have written our problem sentence:

```
a. If the carpet is in the painting:
   John wants to buy the painting of (bikwa) the fireplace
     kwa the Persian carpet.

b. If the carpet and painting are separate:
   John wants to buy bikwa the painting of the fireplace
     kwa the Persian carpet.
```

In the first sentence, "bikwa" is optional, since the default interpretation is to link "the Persian carpet" with the noun phrase that immediately precedes it, which, in this case, is "the fireplace".

If the second sentence sounds too strange to you, simply replace "bikwa" with the English term "not only", and "kwa" with the English term "but also". Thus:

```
John wants to buy not only the painting of the
    fireplace but also the Persian carpet.
```

Surprisingly, English CAN have unambiguous coordination, even when one coordinated structure is embedded in another, as in the following example:

```
He baked not one or two pies or even three or four pies
  but actually ten pies!
```

In natural languages, though, coordination syntax can be inconsistent, and, as a result, ambiguous. As an AL designer, you can enforce consistency and totally eliminate the possibility of coordination ambiguities, without in any way requiring unnatural syntax.

5.2 OTHER ATTACHMENT AMBIGUITIES

Coordinated structures are not the only ones that can suffer from ambiguity. In some cases, a syntactic marker can have more than one interpretation, as in the following sentence:

```
John saw the man with a telescope.
```

Did John use a telescope to see the man, or did the man have the telescope? In this case, the problem is with the English word "with", which can be interpreted in the sense of either _accompaniment_ or _instrument_. The easiest solution is simply to use different words for the different meanings. Thus, your AL could have a case tag meaning "using" and a noun phrase tag for "accompanied by".

There are other cases of ambiguity, however, where the solution is not as obvious or as simple. Consider the following example:

```
John watched the man observing the crowd with a
  telescope.
```

Here, the telescope is clearly being used as an instrument, but by whom? It's interesting and perhaps instructive to look at how natural languages deal with this ambiguity. In English, you could rephrase the sentence as follows (assuming John has the telescope):

```
John watched with a telescope the man observing the
  crowd.
```

I'm not even sure if this sentence is grammatical in English, but even so, there's no reason why you can't make it grammatical in your AL. The problem, though, is that we are FORCING the speaker to use a different word order to prevent an ambiguity, and I don't believe that this is a

practical solution. In my opinion, a design is flawed if structures are created to be used only when ambiguity can occur. A much better solution would be to design the syntax and lexicon so that the ambiguity can NEVER occur.

As it turns out, there are several natural languages in which the above ambiguity would not occur. These languages mark the verb (usually by specially inflecting it) to indicate that an instrumental object follows the direct object. In Swahili, for example, the above ambiguous sentence would take one of the following two forms depending on its meaning:

```
John watched-using the man observing the crowd a
  telescope.
                        OR
John watched the man observing-using the crowd a
  telescope.
```

(Although Swahili does not have articles, I've included them so that the results sound more natural to the English-speaking reader.) In effect, the instrumental marking on the verb changes the argument structure of the verb, making it ditransitive with an instrumental second object. This approach will solve the problem, but many designers may not like it, perhaps considering it too exotic or difficult to learn. It would also get very messy if a verb has several arguments. And even natural languages that use this approach, such as Swahili and Mokilese, do it for only a small number of case roles. Although interesting, this approach does not appear to be a practical, general solution. (Also, I've often wondered how Swahili speakers perceive the use of this construction. It appears to be inflectional and that's the way linguists treat it. However, it doesn't allow one to utter an afterthought instrumental role, as in "He broke the window....uh, with a hammer". Perhaps it's not really inflectional, but derivational.)

Another way of solving this problem is to explicitly terminate a structure with a particle. Such a particle would, in effect, be similar to a right parenthesis. Each structure would have its own terminating particle (e.g., one for sentences, one for verb arguments, one for relative clauses, etc.) Such a particle would terminate the immediately preceding applicable structure and prevent anything that follows from attaching to it. Thus, if "John" is using the telescope, our test sentence would be something like this:

```
John watched the man observing the crowd end-sen
  with a telescope.
```

where "end-sen" is a particle that terminates the immediately preceding (embedded) sentence. Thus, "with a telescope" can only attach to the main verb "watched". (I assume here that "with" is a case tag that can only attach to a verb - NOT a noun phrase tag that could attach to "man".)

However, I do not consider this a good solution for the reason I mentioned earlier - we are adding something that is only used to prevent ambiguity, and which won't be needed most of the time. Also, I have not been able to find a single example of this approach among natural languages. If it does, in fact, violate linguistic universals, then it may be effectively unlearnable. In other words, any form of explicit "parenthesization" could only be uttered with forethought

and analysis - it could not be uttered automatically. I imagine that this is the essential difference between a real language and a non-linguistic coding game.

Another possible solution to this problem is to use a different case tag (i.e., preposition or postposition) for each possible interpretation. Since this is a lexical solution rather than a syntactic or morphological one, some designers may find it easier to swallow. One way to implement it would be to specially mark the case tag (i.e., have a distinct case tag) if it attaches to the main verb of the sentence, and leave it unmarked if it attaches to the immediately preceding verb (or vice-versa). For example, an English gloss using this technique would sound something like this:

```
John watched the man observing the crowd main-with a
  telescope.
                        OR
John watched the man observing the crowd with a
  telescope.
```

Note that "with" in the second sentence is not marked - it's attachment defaults to the verb "observing" since it is closest.

However, by marking the case tag to indicate that it attaches to the MAIN verb, you will not be able to attach to a middle verb, as in the following triple embedding:

```
John saw the man who observed the lady who watched the
 crowd with a telescope.
```

You can, of course, provide markers for case tags that attach to inner verbs, but I wouldn't bother. (My feeling is that people who use sentences like this deserve to be misunderstood. :-) Note also that this approach is very much like the one used in Swahili and Mokilese, except that the case tag is marked rather than the verb, and that the inherent messiness of dealing with multiple case roles is avoided.

Probably the best way, one which would work even with multiple embeddings, would be to mark the case tag with a part of the verb it attaches to. Thus:

```
John saw the man who observed the lady who watched the
 crowd saw-with a telescope.

John saw the man who observed the lady who watched the
 crowd obs-with a telescope.
```

All of the above discussion provides fodder for some interesting observations about syntax in general. Since language allows recursion, syntax is inherently two-dimensional. Unfortunately, since words are uttered one after another, the speech stream is inherently linear. Thus, when we are using language, we are trying to force a two-dimensional entity into a one-dimensional mold. The result is sometimes ambiguity. Note though, that real ambiguity is quite rare. The human brain can bring other weapons to bear on syntactic ambiguity, such as semantic context and

world knowledge. Consequently, if you are designing your language for exclusive use by humans, you can afford to ignore this potential for syntactic ambiguity. If, however, you want your language to be computer-tractable, then you must deal with the problem.

5.3 DO WORDS HAVE SYNTAX?

In the preceding essay in this series, I discussed ways to design the surface morphology of an artificial language. As it turns out, syntax and morphology have quite a lot in common. In fact, some linguists are have proposed theories that describe morphology as if it were essentially an extension of syntax. In these theories, rules that apply to the relationship between heads and modifiers in syntax also apply to the relationship between heads and modifiers in morphology. The difference is that heads and modifiers in syntax are complete words, whereas in morphology they are morphemes.

I am not going to try to describe these theories here, since they are both complex and somewhat controversial. All I would like to do here is to very briefly show how the syntactic formalism we've been using can be applied to the shapes of words.

If your AL is at all agglutinating, as seems to be true to some degree of most natural languages, then you will want to be able to create new and more complex words by combining more basic primitives. One way that this is commonly done is through a process called _derivational morphology_, in which a basic or root concept is modified to create a new word by adding a morpheme. For example, English "sad" -> "sadden" -> "sadness", "critic" -> "criticism" -> "criticize", etc. I do not want to discuss the semantics of derivational morphology here, since I discuss it in much greater detail in my monograph Lexical Semantics. Instead, I just want to show you how we can apply the formalism described above to the shapes of words.

Let's start by looking at _open class_ words. In English, these are words such as nouns, adjectives, verbs and most adverbs. They are called _open class_ because new words can enter and leave these groups with relative ease. Their counterpart, _closed class_ words, are words like prepositions, particles and specifiers which change much more slowly. We could define an open class word of a hypothetical AL as having the following "syntax":

```
open_class_word ::= {modifier} root (classifier)
 part_of_speech
```

Such a structure is called _morphotactic_ because it describes the way the morphemes are arranged or "touch" each other. In the above example, a modifier would be a morpheme that narrows down the possible interpretation of the root. For example, a modifier could apply the meaning of "femaleness" to the basic word "dog" to create the new word "bitch". Different modifiers would be used to create other distinctions such as between "mutt", "puppy", "purebred", "male dog", etc. For nouns, a classifier can indicate a position of the root in a hierarchy. For example, "dog" is in the class "mammal". For verbs, it can indicate the thematic roles and argument structure. For example, the words for "escape" and "release" would have the

same root and part of speech, but would have different classifiers, because one is inherently reflexive and the other is inherently causative. Also, if each root has a default classifier associated with it, then the classifier does not have to be specified. This will help somewhat to keep common words relatively short.
We can also describe the phonological "shape" of the individual morphemes, which linguists would call the _phonotactics_ of the morphemes. For example:

```
modifier ::= CV

root ::= CVCC

classifier ::= VCC

part_of_speech ::= V
```

where C is a consonant and V is a vowel. Thus, if the modifier "mi" indicates "femaleness", the root "temb" means "dog" within the class of mammals indicated by the classifier "anc" (pronounce 'c' like 'ch' in 'church'), and "o" is the part-of-speech indicator for nouns, then the word "mitembanco" would mean "bitch". If the default classifier for the root "temb" is "anc", then "anc" can be dropped, shortening the result to “mitembo".

You can extend your word syntax to include other kinds of words. For example, an anaphor could be created from any open class word by using only the classifier and part of speech. Thus, the pronoun "it", when referring to a female dog (or any mammal), would be "anco". In fact, in this particular scheme, the word "anco" would also be the word for "mammal".

Finally, you can use different forms for closed class words. For example, specifiers could have the form:

```
specifier ::= {modifier} specifier_class

specifier_class ::= CSV
```

where S is a semivowel. Here, one possible modifier could be a numeral, and the specifier class could indicate ordinality. Thus, if the modifier "ku" means "seven", and "nya" indicates ordinality, then the word "kunya" would mean "seventh".
And so forth. I won't say any more about word design here, since I will talk much more about it in my monograph Lexical Semantics. I simply wanted to whet your appetite and show you how words can in fact have "syntax".

[Postscript: Many thanks to Jacques Guy and Dan Maxwell for their helpful comments!]

End of Essay