yield unacceptable results for the classes of cases for which they are
designed.

University of Chicago

REFERENCES

2 Similar results are obtainable without use of the erasure clause. Consider the sentence (1'}). An uncle is an aunt'. The negation of (1'), by clause (ii) of the entry for 'Neg' is the analytic sentence (2') 'An uncle is not an aunt'. Since 'Female' in the reading of the predicate of (1') has been replaced by A(Female), i.e., '(Male)', in forming the reading for the predicate of (2'), there is nothing in the reading for (2') to distinguish it from the reading for (3') 'An uncle is an uncle'. Katz's definitions thus lead to the conclusion that (3') and (2') are synonymous.
3 'R subj' and 'R pred' stand respectively for readings for the subject and predicate of S1. Similarly for 'R subj' and 'R pred'; and generally, 'R const' and 'R const' stand for readings for the same unspecified constituent (e.g., Subject) of S1 and S2.
4 Another route to the same conclusion is this: The sentence 'A spinster is a person' is analytic. Therefore, R subj ∣= R subj. Since pred = pred, R pred ∣= R pred. Therefore (4), which is analytic, entails (5) which is synthetic.

A PROGRAM FOR SYNTAX

The program for syntax which I describe here is not one I can claim as specially my own. The two basic ideas are due to Frege: analysis of an expression into a main functor and its argument(s), and distinction among categories of functors according to the categories of arguments and values. The development of a handy notation for categories, and of an algorithm to test whether a string of expressions will combine into a complex expression that belongs to a definite category, is due to the Polish logicians, particularly Ajdukiewicz. My own contribution has been confined to working out details. So my program is not original, but I think it is right in essentials; and I am making propaganda for it by working it out in some particular instructive examples. I think this is all the more called for because some recent work in syntax seems to have ignored the insights I am trying to convey.

I shall begin with some thoughts from Aristotle's pioneering treatise on syntax, the De Interpretatione. Aristotle holds that the very simplest sort of sentence is a two-word sentence consisting of two heterogeneous parts—a name, and a predicative element (rhēma). For example, 'petetai Sōkratēs', 'Socrates is flying'. This gives us an extremely simple example for application of our category theory:

```
  petetai
  Sōkratēs
  s
  n
  s
```

The two-word Greek expression as a whole belongs to the category s of sentences; 'petetai' is a functor that takes a single name (of category n) 'Sōkratēs' as argument and yields as a result an expression of category s. Ajdukiewicz represented functorial categories by a fractional notation: α/β would be the category of a function that operates upon a single argument of category β to yield an expression of category α, so that we have a "multiplying out" of category indices. This notation becomes awkward to print when indices become complex; so following a suggestion of my
Leeds colleague Dr. T. C. Potts I shall henceforth rather write ‘:\alpha:\beta’ for such a functorial category. (This device makes bracketing theoretically superfluous, but in practice I shall insert parentheses sometimes to facilitate reading.) Our first rule then is the multiplying-out rule:

: \alpha \beta \rightarrow \alpha \\
For instance, :sn n \rightarrow s.

Aristotle observed that one may get a sentence from a rhêma like ‘petetai’ not only by combining it with a name but also by combining it with a quantified phrase like ‘pâs anthrôpos’, ‘every man’. He further observed that these two types of sentence behave quite differently under negation; the negation of ‘petetai Sôkratês’ is ‘ou petetai Sôkratês’, when the negation ‘ou’ attaches to the rhêma ‘petetai’; the negation of ‘pâs anthrôpos petetai’ is ‘ou pâs anthrôpos petetai’, where the negative attaches to the quantified phrase ‘pâs anthrôpos’. This is a profound insight, ignored by those who would lump together proper names and phrases like ‘every man’ as Noun Phrases; we have two different syntactical categories. It is easy to find in the Ajdukiewicz scheme another category that will yield the category s when combined with the category :sn; for we shall have, by the general rule, :ss :sn :sn \rightarrow s. But this is not enough to exhaust the Aristotelian insight. We should wish to make ‘ou petetai’ ‘does not fly’ a syntactically coherent sub-string of ‘ou petetai Sôkratês’, and on the other hand to make ‘ou pâs anthrôpos’ ‘not every man’ a syntactically coherent sub-string of ‘ou pâs anthrôpos petetai’. But by the Ajdukiewicz criteria for a string’s being syntactically coherent (SC), neither string will come out as SC. To negation ‘ou’, we must assign the category :ss of a sentence-forming operator upon sentences; and neither the category-indices ‘:ss :sn’ of ‘ou petetai’ nor the indices ‘:ss :ss:n’ of ‘ou pâs anthrôpos’ multiply out by Ajdukiewicz’ rule to a single index of the whole expression. These are two particular cases of a general fact, noticed by medieval logicians: that a sentence may contain a *formale*, formal element – Ajdukiewicz’ main functor – negation of which is negation of the whole proposition.

Intuitions about the SC nature of sub-strings are fallible, but are *pro tanto* evidential; we need to check our general theories of syntax against such intuitions, and also to correct our intuitions against wider insights. By the two-way process we may hope to get steadily closer to truth. In this case, we can satisfy the demands of intuition if we supplement the Ajdukiewicz multiplying-out rule with a recursive rule:

If \( \alpha \beta \rightarrow \gamma \), \( \alpha :\beta \delta \rightarrow :\gamma \delta \).

This already covers the Aristotelian and medieval cases. For suppose the main functor of a sentence is of category :\beta, so that we have a sentence by adding a \( \beta \) expression. We then have by our recursive rule:

Since :ss s \rightarrow s, :ss :s\beta \rightarrow :\beta.

And this covers all cases in which negation, of category :ss, operates upon a sentence of structure :s\beta. The string of expressions categorized as:

:ss :s\beta,

may be split up in two ways into SC sub-strings; namely, we may regard negation (:ss) as operating on the whole sentence categorized as :s\beta; or, we may regard it as combining with the :s\beta expression to form a complex :s\beta expression, which then combines with the \( \beta \) expression to form a sentence. The two Aristotelian examples are covered by this account if we take \( \beta = :sn \) and \( \beta = :ss :sn \).

Such possibilities of multiple analysis do not mean that we have a syntactically ambiguous string. We have a single “proper series of indices”, as Ajdukiewicz calls it, for a given sentence; the different ways of multiplying out the indices reveal two different but equally legitimate ways of dissecting out an SC sub-string from a larger SC string.

The Ajdukiewicz scheme allows for functors that take more than one argument. In the present discussion it will be enough to consider functors that take two arguments of the same category: if this category is \( \beta \) and \( \alpha \) is the category of the functor plus its two arguments, I give the functor the category :\alpha(\beta). We get in Ajdukiewicz the rule for multiplying out with such category indices:

: \alpha(\beta) \beta \rightarrow \alpha.

Once again I add a recursive rule:

If \( \alpha \beta \rightarrow \gamma \), then \( \alpha :\beta \delta :\beta \delta \rightarrow :\gamma \delta \).

A medieval example may serve to illustrate the newly introduced
categories. 'John or James came' need not be transformed into 'John came or James came' before we investigate its SC character; we can show it to be SC as it stands. But we cannot treat it as having the same simple subject-predicate structure as 'John came', only having a complex subject 'John or James' instead of the single name 'John'. For whereas the negation of 'John came' attaches to the predicate 'came', 'John or James came' has to be negated by negating 'or' – 'neither John nor James came'. So my medieval writer justly took 'or' to be here the *formulae* or main functor. 'John or James' may be regarded as a restricted existential quantification – 'for some x in the universe {John, James}, x...'; so we assign to it, just as we do to 'pās anthropōs' or 'every man', the category :s:sn. The functor 'or' will then be assigned the category :s:sn (2n), which combines with two names of category n to yield an :s:sn expression; and this in turn combines with the predicate 'came' of category :s:sn to yield a sentence. Negation, of category :s:s, will combine with a functor of category :s:sn (2n) to yield a functor of the same category; we see this by twice applying our recursive rule:

\[
: ss \, s \rightarrow s \\
\text{ergo, } : ss : s : sn \rightarrow : s : sn \\
\text{ergo, } : ss : : (s : sn) (2n) \rightarrow : ((s : sn) (2n))
\]

I shall now briefly sketch how the traditional apparatus of Parts of Speech get reshaped in an Ajdukiewicz grammar. I shall consider only some of the traditional list.

1. VERBS

Intransitive verbs like 'come' or 'petetai' may be categorized as :s:n. A transitive verb along with its noun-object, a phrase like 'loves Socrates', will likewise be of category :s:n; 'loves' itself is thus most conveniently categorized as :s:n:n. 'Every Greek loves Socrates' then admits of a double dissection into SC sub-strings; we need this, because we need to recognize both 'loves Socrates' and 'every Greek loves' as SC expressions that may recur in other contexts e.g. in the relative clauses 'who loves Socrates' and 'that every Greek loves'. (When symbolizing a piece of argument stated in the vernacular, we might find it convenient to represent either recurrent phrase by the same one-place predicate letter each time it occurred.) In fact, 'loves Socrates' gets categorized as :s:n:n, which multiplies out to :s:n by the Ajdukiewicz rule; and then 'Every Greek loves Socrates' will be categorized as :s:n :s:n, which multiplies out to s. On the other hand, 'every Greek loves' gets categorized as :s:n :s:n; this multiplies out to :s:n by our recursive rule:


So 'Every Greek loves Socrates' comes out as :s:n n, and thus again as s. Once again, we have two equally legitimate analyses, not a syntactic ambiguity.

II. CONJUNCTIONS

The term 'connective' is preferable, since 'conjunction' is indispensable as a name for one of the truth-functions. Traditional grammar distinguishes subordinating and coordinating connectives; in one case, e.g. with 'if', the connective is felt to go with the clause that follows it; in the other case, e.g. 'and', 'or', the connective is felt to be joining two clauses, not going with one rather than the other. No such distinction is needed for the binary sentence-connectives in a formal system, which may very well be taken to be all of one category; but for analysis of the vernacular it seems better to recognize a syntactical distinction, between the two sorts of connectives. A subordinating connective would be of category :s:ss; so such a connective together with the clause following it would be of category :s:ss s, i.e. :s:s, which is the category of a sentence-forming operator upon a sentence. A coordinating connective, on the other hand, would be of category :s:2s. A string categorizable as :s:2s s has as a whole the category s; but just as the category indices ':s:2s' s do not multiply out to a single index, so we need not take either 'John ran and' or 'and Jane rode' to be an SC substring of 'John ran and Jane rode'.

Grammarians have often taken sentences in which a coordinating connective joins expressions other than sentences to be derived from sentences in which the same connective joins sentences. I regard this view as wholly erroneous. Our theory of categories does not restrict the possible arguments of an :s:2s connective to a pair of sentences; on the contrary, by our recursive rule we have that a pair of the category :s:β may also be so connected to form a third:

Since :s:2s s s \rightarrow s, :s:2s :s:β :s:β \rightarrow :s:β, whatever category β may be.
And so we obtain a correct analysis of a sentence like:

All the girls admired, but most boys detested, one of the saxophonists.

This is not equivalent, as a moment's thought shows, to:

All the girls admired one of the saxophonists, but most boys detested one of the saxophonists,

and cannot sensibly be regarded as a transformation of it. The expressions 'all the girls admired' and 'most boys detested' are in fact each assignable to the category :sn, as we saw before regarding 'every Greek loved'; so the coordinating connective 'but' can combine them to form a single string of category :sn. Since 'one of the saxophonists' is plainly a quantifying expression like 'every man', it is of category :ssn; this is the main functor, operating upon 'All the girls admired, but most boys detested', of category :sn, to yield a sentence. The change of intonation pattern marked by the second comma, as contrasted with the smooth run in the sentence:

All the girls were thrilled, but most boys detested one of the saxophonists,

is easily explained: 'most boys detested one of the saxophonists' is an SC substring (in fact a sentence) in the latter example but not in the former, and the change of intonation shows our feeling for this. (Just as 'Plato was bald' has a different intonation pattern when it stands by itself and when it comes as part of 'The man whose most famous pupil was Plato was bald'; in the latter context it is patently not an SC string.)

Similarly, a subordinating connective along with the clause following it will come out, as I said, in the category :ss, that of a sentence-forming operator upon sentences; but it does not follow that such a unit can be read only as attached to an entire main clause; on the contrary, we must sometimes so regard it as attached to an expression of another category. A good medieval example of syntactical ambiguity brings out this point:

Every man dies when just one man dies.

This could be true (and was once, in this sense, a presumption of English law) as denying the possibility of quite simultaneous deaths; in the other possible sense, it could be true only if there were just one man, so that his death was the death of every man. The first sense requires us to take the subordinating connective plus its clause, 'when just one man dies', as going not with 'Every man dies' but just with 'dies', as we may see from the paraphrase:

It holds of every man that he dies when just one man dies

(namely he himself and nobody else).

The second sense affirms that the universal death of mankind happens along with the death of one and only one man; here, the whole sentence 'Every man dies' is operated on by the sentence-forming operator 'when just one man dies'.

III. ADVERBS

Some adverbs, as the name suggests, are verb-forming operators upon verbs, and are thus of category :(:sn) (:sn). Thus 'passionately protested' comes out as of the same category with 'protested' (I am taking this as an intransitive verb of category :sn) but also 'passionately loved' comes out as of the same category with 'loved', namely ::ssn, for we have:

Since :(:sn) (:sn) :sn → :sn, (:sn) (:sn) ::ssn → ::ssn.

And as in the other example we have a double possibility of analysis that corresponds to no syntactical ambiguity: 'passionately/loved Mary' and 'passionately loved/Mary' alike work out as SC, and here once more we are just picking out subordinate SC strings in alternative ways from an SC string.

Two adverbs can be joined by a coordinating connective – 'passionately and sincerely', 'improbably but presumably'. On the other hand a combination like 'passionately and presumably' sounds like nonsense. It is nonsense; it involves a confusion of syntactical categories. For an adverb like 'improbably' or 'presumably' is to be taken, in at least some cases, not as modifying the verb, but as modifying the whole sentence – its category must thus be :ss. Two adverbs of category :ss can be joined with the connective 'but' of category :s(2s); for by our recursive rule:

Since :s(2s) s s → s, :s(2s) :ss :ss → :ss.
So 'improbably but presumably' comes out as a complex adverb of category :ss. Again, by our recursive rule:

\[
\text{Since } :s(2s) s s \rightarrow s, \quad :s(2s) :s n s n \rightarrow :s n \\
\text{Since } :s(2s) :s n s n \rightarrow :s n, \quad :s(2s) :s n n :s n (s :s n) (s :s n) :s n \\
\rightarrow :s n (s :s n).
\]

So 'passionately and sincerely' comes out as of category :s n (s :s n), like its component adverbs. But an operator of category :s (2s) can take only two arguments of like category; so if we attempt to join with 'and' the adverbs 'passionately', of category :s n (s :s n), and 'presumably', of category :s s, we get syntactical nonsense.

IV. PREPOSITIONS

A prepositional phrase may be an adverb of category :s (s :s n), like 'in London' in 'Raleigh smoked in London'; if so the preposition in the phrase is of category :s (s :s n)n. On the other hand, in the sentence 'Nobody except Raleigh smoked', 'nobody except Raleigh', like plain 'nobody', is a quantifying expression, of category :s n s n. So 'except Raleigh' is a functor turning one quantifying expression into another — thus, of category :s n :s s n; and 'except' itself is of category :s n :s n n. As before, expressions of the same category can be joined with coordinating connectives but not expressions unlike in category; for example, we may assume that 'before' and 'after' are both of category :s (s :s n)n, so 'before or after' is well-formed, as we may see:

\[
\text{Since } :s(2s) s s \rightarrow s, \quad :s(2s) :s n s n \rightarrow :s n \\
\text{ergo, } :s(2s) :s n n :s n (s :s n) (s :s n) :s n \\
\rightarrow :s n (s :s n).
\]

But though 'Nobody smoked before or after Raleigh' is well-formed, 'Nobody smoked before or except Raleigh' is syntactical nonsense, because 'before' and 'except' differ in category.

The preposition 'by' is of different category, again, in the use it has with the passive construction; 'was hit by' must be regarded as formed by a logical operation upon 'hit', and the functor is of category :s n (s :s n), since :s n is the category of 'hit'. The word "governed" by 'by' is thus not syntactically connected with it, since ':s n (s :s n)' and 'n' do not multiply out to give a single index. Why anyone should call a 'by' phrase an Adverbial of Manner I can only dimly imagine, calling to mind half-remembered school exercises in parsing. (How, in what manner, was Caesar killed? By Brutus. Very well then, 'by Brutus' is an Adverbial of Manner, just like 'brutally'!

The categorizing of prepositions, however, raises very serious difficulties for our whole theory of categories — difficulties which I think can be overcome only by introducing a further powerful, recursive, procedure for establishing that an expression is SC. For example, 'some city' like 'every man' is of category :s n s n; but if we assign 'in' to category :s n (s :s n)n, not only is the functor incapable of taking 'some city' as an argument as it can take 'London', but also the whole sentence 'Raleigh smoked in some city' cannot be made out to be SC by any way of multiplying out the category indices of 'Raleigh' (n), 'smoked' (:s n), 'in', and 'some city'. The only arrangement of the indices that multiplies out to 's' is this:

\[
:s n :s n (s :s n) n n :s n (\text{some city}) (\text{in}) (\text{Raleigh}) (\text{smoked})
\]

but this gives rather the syntactical analysis of 'Some city smoked in Raleigh'.

Our recursive procedure is supplied by the well-known logical device — well expounded e.g. in Quine’s Methods of Logic — of introducing a predicate as an interpretation of a schematic letter in a schema. If 'F' is of category :s n s n, the schema 'F(London)' will be SC and of category s. Now if 'F(London)' is SC, so will '(Some city) F' be — since ':s n :s n' gives 's'. We now reason thus: We have seen how to assign categories to the expressions in 'Raleigh smoked in London' so as to show it is SC and of category s. We may accordingly assign 'Raleigh smoked in London' as the interpretation of the one-place predicate letter 'F' in the SC schema 'F(London)'. But then also the corresponding interpretation of the SC schema 'Some city) F' will be SC; and this interpretation is the sentence 'Raleigh smoked in some city'; so this sentence is also SC.

Some quite short sentences require a number of steps like this to show
they are SC. I shall give an example presently; but I must first explain how to categorize the reflexive pronouns in 'self'. Such a pronoun can be attached to a transitive verb of category :snn to yield a one-place predicate of category :sn. We have already seen two ways of so attaching an expression to a transitive verb; both 's:sn :snn' and '::snn n' multiply out to ':sn'. But a reflexive pronoun plainly is not either a name, or a quantifying expression like 'every man'. Nor is it a mere proxy or substitute for an expression of one of these categories; we might take 'himself' in 'Judas hanged himself' to go proxy for 'Judas', but there is nothing 'himself' would be taken as proxy for in 'The Apostle who hanged himself went to Hell', and plainly 'hanged himself' is not syntactically different in the two sentences. The only category that gives the right result is :sn :snn, since ::sn :snn :snn :snn :snn :snn → :sn. We may now consider our example, recalling ones of medieval vintage:

Every number or its successor is even.

We begin with the clearly well-formed sentence: '8 or 3 is even'. If we give the numerals the category n of proper names (shades of Frege!) then 'is even' will be of category :sn and this sentence will be of the same syntax in essentials as our previous example 'John or James came'.

Since '8 or 3 is even' is SC, we may take '8 or 1 is even' as the interpretation of the one-place predicate letter 'F' (category :sn) in the SC schema 'F(n)'. Now if 'F(n)' is SC, then if we assign to 'n's successor' the quantifier category :snn (there are arguments for doing this, but I omit them for simplicity of exposition), the schema 'n's successor)' F' will be SC. But the corresponding interpretation of this schema will be the sentence:

8 or 5's successor is even.

So this sentence is SC.

We now treat '1 or 2's successor is even' as the interpretation of the two-place predicate letter 'R' in the schema 'R(8,5)'. If 'R' is of category :snn, and each of '8', '5' is of category n, this schema is SC. But then also the result of operating on 'R' with a reflexive pronoun, 'R(1, itself)', will be an SC one-place schematic predicate; since we just saw that is how the reflexive pronoun works, to turn a two-place predicate into a one-

place predicate. And the corresponding interpretation of 'R(1, itself)' will be:

1 or itself's successor is even.

So this counts as an SC one-place predicate. English accidence of course demands that one rewrite 'itself's as 'its'.

Finally, since we may treat '1 or its successor is even' as an interpretation of the one-place predicate letter G, and since with the quantifying expression 'Every number' prefixed we get an SC schema 'Every number' G', we get as the relevant interpretation of this schema:

Every number or its successor is even.

So this is an SC sentence; which was to be proved.

Grammarians may find my interpretation of this sentence extremely farfetched. They should consider, however, that it does correspond to the obviously correct paraphrase:

It holds of every number that it or its (own) successor is even.

Moreover, other analysis, more comformable to the ideas that come natural to grammarians, lead us into a bog of absurdity. We cannot construe our sentence on the model of:

Every man or every woman will be shot.

For this is equivalent to 'Every man will be shot or every woman will be shot'; but no such equivalence holds in our case – the irrelevant falsehood 'Every number is even' has nothing to do with the syntax of our example. (Nor need 'Every man or every woman will be shot' itself be construed as short for a disjunction of sentences, though it is equivalent to one; for it is easily shown by our rules that the two quantifying expressions 'every man' and 'every woman', of category :snn, can in their own right be joined by 'or', category :ss, to form an expression of that same category.) As for taking 'number or its successor' as a complex term, that lets us in at once, as my medieval predecessors noticed, for an absurd "syllogism":

Every number is a (number or its successor).
Every (number or its successor) is even.

ergo: Every number is even!
V. Relative Pronouns

Quine and I have both repeatedly argued that the use of relative pronouns may fruitfully be compared to that of bound variables. The question is, though, which kind of expressions throws light on the syntax of the other kind; the syntax of bound variables is very complicated and unperspicuous, as we may see e.g. from the need for rules in logic books to guard against unintended "captures" of variables in formulas introduced by substitution. Ajdukiewicz attempted to modify his scheme of categories so as to assign categories to quantifiers that bind variables; but his theory is manifestly inadequate – it takes no account of the fact that a variable is bound to a quantifier containing an equiform variable: for Ajdukiewicz '*(x) (Fxy)*' would not differ syntactically from '*(z) (Fzy)*', so far as I can see.

It occurred to me that some light might be thrown on the matter by constructing a simple combinatory logic, on the lines of Quine's paper 'Variables explained away'. I cannot claim any algorithmic facility in working with combinators, but I have reached results encouraging enough to be worth reporting.

To translate into a combinatory notation the English sentence:

> Anybody who hurts anybody who hurts him hurts himself.

I began with an obvious translation of this into quantifier notation (variables restricted to persons; 'H 1 2' = '2 hurts 1'):

> (x) ((y) (Hxy → Hyx) → Hxx)

and then devised the following set of combinators:

'Univ': when a predicate followed by a string of variables has prefixed to it a universal quantifier binding just the last variable of the string, we may instead delete the last variable and prefix 'Univ'; e.g. '*(x) (Fx)*' becomes 'Univ F' and '*(x) (Ryx)*' becomes 'Univ Ry'.

'Imp': if the antecedent of a conditional consists of a predicate followed by a string of variables, and the consequent consists of a predicate followed by just the same string, then we may instead write 'Imp' followed by the two predicates followed by the string of variables. E.g. 'Rxy → Sxy' becomes 'Imp RS xy'; 'Fz → Gz' becomes 'Imp FG z'.

'Ref': if a predicate is followed by a string of variables ending with repetition of a variable, we may instead delete the repetition and prefix 'Ref' to the predicate. E.g. 'Rxx' becomes 'Ref Rx', and 'Syxx' becomes 'Ref Sx'.

'Cnv': the result of prefixing 'Cnv' to a predicate followed by a string of two or more variables is tantamount to the result of switching the last two variables of the string. E.g. 'Ryx' may be rewritten as 'Cnv R xy', and 'Ryxxy' as 'Cnv R xxy'.

We now eliminate, step by step, the variables in the above formula. 'Hxy → Hyx' may be rewritten as 'Hxy → Cnv H xy', and then as 'Imp H Cnv H xy'.

So '*(y) (Hxy → Hyx)*' may be rewritten as '*(y) (Imp H Cnv H xy)*' and thus as 'Univ Imp H Cnv H x'.

'Hen' may be rewritten as 'Ref H x'; so since '*(y) (Hxy → Hyx)*' may be rewritten as 'Univ Imp H Cnv H x', '*(y) (Hxy → Hyx) → Hxx)*' may be rewritten as:

> Imp Univ Imp H Cnv H Ref H x.

Finally, to get an equivalent of the whole formula, we get the effect of the prenex '*(x)*' by deleting the final 'x' and prefixing 'Univ':

> Univ Imp Univ Imp H Cnv H Ref H.

It is fairly easy to see how the symbols of this string should be assigned to categories. 'Univ F', when 'F' is one-place, is a sentence of the same form as 'Everyone smokes'; 'Univ', like 'everyone', is of category :sn. 'H', like the transitive verb 'hurts' that it represents, is of category :sn. 'Imp' is a connective that combines two predicates to form a predicate with the same number of places; it is thus of category :sn(2:sn). 'Ref', like a reflexive pronoun, reduces a predicate of n + 1 places to a predicate of n places; it is thus of category :sn(:sn). And 'Cnv' turns a many-place predicate into one of the same number of places; it is thus of category :(:,snn). (It might seem as if these assignments of categories were too restrictive of the arguments these functors would be allowed to operate on. But in view of our recursive rules this is not so. For example, 'Imp' could combine two predicates of category :sn to form a third:

> :sn(2:sn) :sn :sn → :sn

We may now check that the above string is, as Ajudkiewicz would say, well-formed throughout and of category ::sn, since we have:

\[ (::sn)(::sn) ::sn \rightarrow ::sn. \]

So ‘Imp H Cnv H’ is of category ::sn, since we have:

\[ ::sn(2::sn) ::sn ::sn \rightarrow ::sn. \]

Hence, by the recursive rule:

\[ ::sn(2::sn) ::sn ::sn ::sn \rightarrow ::sn. \]

So ‘Univ Imp H Cnv H’ is of category ::sn, since we have:

\[ ::sn ::sn ::sn \rightarrow ::sn. \]

\[ \text{ergo,} \quad ::sn ::sn ::sn \rightarrow ::sn. \]

Now also ‘Ref H’ is of category ::sn, since we have:

\[ ::sn(::sn) ::sn \rightarrow ::sn. \]

Hence ‘Imp Univ Imp H Cnv H Ref H’ is of category sn

\[ ::sn(2::sn) ::sn ::sn ::sn \rightarrow ::sn. \]

Finally, since ‘Univ’ is of category ::sn, the category of the whole works out as s.

Now this string of predicates and combinators can at once be translated, word for word, into pidgin English:

\[ \text{Univ Imp Univ Imp H Cnv H Ref H} \]

anybody anybody who hurt get hurt by hurt self.

(Some small changes of word order were made to improve this mock-up of English. ‘Cnv’ was rendered by ‘get’ before the argument of the functor and ‘by’ after it, and ‘Ref’ by ‘self’ after rather than before the argument of this functor.) I suggest, therefore, on the strength of this example (and of others I have not space for here) that we may hope to get a good mock-up of the use of relative pronouns in the vernacular by exercises in combinatory logic.

An interesting confirmation of this conjecture comes to us when we observe that in the above sentence ‘Univ Imp’ is an SC sub-string:

\[ \text{Univ Imp} \]

\[ ::sn \quad (::sn)(2::sn) \rightarrow ::sn(2::sn), \]

by our recursive rule since ::sn ::sn \rightarrow s.

Accordingly, we could definitionally introduce a new combinator of category ::sn, say ‘Unimp’, and rewrite our string as ‘Unimp Unimp H Cnv H Ref H’. The new string may also be translated straight into pidgin English:

\[ \text{Unimp Unimp H Cnv H Ref H} \]

Whoever whoever hurt get hurt by hurt self.

And this seems to give a correct account of the logical syntax of the relative pronoun ‘whoever’. Of course these results are most unnatural from the point of view of school grammar; in ‘anybody who hurts...’ the major division would be taken to come not after ‘who’ but after ‘anybody’, and ‘who hurts...’ would be taken as an SC sub-string somehow “modifying” ‘anybody’. But if we are to get a scientific insight into syntax we mustn’t be afraid to break Priscian’s head. As Descartes said, manum feralae subductimus – we no longer need hold out our hand to be caned by pedants.

Such are some specimens of the work I have done to carry out this Polish program. Much more remains to be done; it is like fitting together a huge jigsaw puzzle. But I hope I may have persuaded some readers that further following of this path is worth while.

University of Leeds,
University of Pennsylvania