Contemporary Mathematics
Volume 92, 1989

Multicategories Revisited

J. LAMBEK[1]

Abstract: This is an account of the origin of multicategories and their relation to algebra, logic and linguistics. It is shown that a multicategory is completely characterized by its internal language and how this language may be exploited to obtain properties of tensor product and internal *hom*. There is a discussion of what happens when some or all of Gentzen's structural rules are imposed.

1. SYNTACTIC CALCULUS

This article is largely historical, describing my interest in a common thread to algebra, logic and linguistics over the last thirty years or so; but I hope that it contains at least one new idea. It all began when George Findlay and I were trying to study homological algebra, just prior to the appearance of the book by Cartan and Eilenberg. We were looking at bimodules ${}_RA_S$, ${}_SB_T$ and ${}_RC_T$, R, S and T being associative rings with unity, and became obsessed with the natural isomorphisms

$$Hom(B, A\backslash C) \cong Hom(A \otimes B, C) \cong Hom(A, C/B),$$

where $A \otimes B$ is the tensor product $A \otimes_R B$, $A\backslash C$ (read A under C) is the $S - T$ - bimodule $Hom_R(A, C)$ and C/B (read C over B) is the $R - S$ - bimodule $Hom_T(B, C)$. We had noticed an analogous relationship for two-sided ideals A, B and C of a ring R, namely

$$B \subseteq A \cdot .C \Leftrightarrow A \cdot B \subseteq C \Leftrightarrow A \subseteq C. \cdot B,$$

where $A \cdot C$ is the product ideal, and $A \cdot .C = \{r \in R \mid \forall_{a \in A}\, ar \in C\}$ and $C. \cdot B = \{r \in R \mid \forall_{b \in B}\, rb \in C\}$ are two kinds of residual quotients, distinct when R is not commutative. Although our ideas on this topic were never published, I tried to push the notational distinction between left and right division in my book on ring theory (1966), but most ring theorists rejected this notation.

1980 Mathematics Subject Classification (1985): 03B40; 03F05; 18D15; 68S05; 68S20.

[1]This research was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Quebec Department of Education. The author is attached to the Category Research Center in Montreal. He is indebted to Dr. Franz Guenthner for the hospitality of the Forschungsstelle für natürlich-sprachliche Systeme in Tübingen, where most of this was written.

© 1989 American Mathematical Society
0271-4132/89 $1.00 + $.25 per page

The manuscript with Findlay had already pointed out a possible application to linguistics, which first appeared in print in 1958. The idea was this: given syntactic types N for "name" and S for "statement", and perhaps other basic types, one can build compound types $A \cdot B, A\backslash C$ and C/B, where $A \cdot B$ is the type of expressions consisting of expressions of type A followed by expressions of type B, $A\backslash C$ is the type of expressions which when preceded by expressions of type A yield expressions of type C and C/B is the type of expressions which when followed by expressions of type B yield expressions of type C. If we write $A \to B$ to mean that every expression of type A may also be assigned the type B, we notice that

$$B \to A\backslash C \Leftrightarrow A \cdot B \to C \Leftrightarrow A \to C/B.$$

In my 1958 paper I also assumed associativity in the form

$$(A \cdot B) \cdot C \leftrightarrow A \cdot (B \cdot C),$$

but I had second thoughts about this in 1961.

To illustrate these ideas, note that the name *John* has type N and that the statement *John works* has type S. Since *John* can here be replaced by any other name, it follows that *works* has type $N\backslash S$, as has any other predicate. Indeed, if we work in the so–called *associative syntactic calculus*, a formal system which also postulates the reflexive and transitive laws for the binary relation $\to$ between types, we can formally prove that $N \cdot (N\backslash S) \to S$, as follows:

$$\frac{A\backslash C \to A\backslash C}{A \cdot (A\backslash C) \to C}.$$

Similarly, in *John (likes Jane)*, the predicate *likes Jane* has type $N\backslash S$, hence *likes* has type $(N\backslash S)/N$. Indeed, $((N\backslash S)/N) \cdot N \to N\backslash S$ and, more generally,

$$\frac{C/B \to C/B}{(C/B) \cdot B \to C}.$$

Had we bracketed the above sentence differently as *(John likes) Jane*, we would have obtained the type $N\backslash(S/N)$ for *likes*. Fortunately, this does not matter, as one can prove that $(N\backslash S)/N \leftrightarrow N\backslash(S/N)$, using associativity. For example, abbreviating $(A\backslash B)/C$ as D, we have the following proof in tree form of this result in one direction:

$$\dfrac{\dfrac{\dfrac{(A \cdot D) \cdot C \to A \cdot (D \cdot C) \qquad \dfrac{\dfrac{D \to (A\backslash B)/C = D}{D \cdot C \to A\backslash B}}{A \cdot (D \cdot C) \to B}}{(A \cdot D) \cdot C \to B}}{A \cdot D \to B/C}}{(A\backslash B)/C = D \to A\backslash(B/C)}$$

Readers with appropriate background will recall an analogous isomorphism in homological algebra, which in our notation is written

$$(A\backslash B)\varnothing C \cong A\backslash(B\varnothing C).$$

As it happened, the linguistic application discussed so far had been anticipated by Bar-Hillel (1953), based on still earlier work by Ajdukiewicz (1937). As opposed to the latter, Bar- Hillel distinguished between the rules $A \cdot (A\backslash C) \to C$ and $(C/B) \cdot B \to C$, although not in this notation. My methods went a bit further, as will be seen from the following example.

Consider the statement *he works* of type S. We cannot give *he* the type N, as **Jane likes he* is ungrammatical, but we can assign to it the type $S/(N\backslash S)$, as $(S/(N\backslash S)) \cdot (N\backslash S) \to S$. On the other hand, *him* in *(Jane likes) him* has type $(S/N)\backslash S$, as $(S/N) \cdot ((S/N)\backslash S) \to S$. Of course, as *John* can be written in place of *he* or *him*, *John* should also have types $S/(N\backslash S)$ and $(S/N)\backslash S$ in addition to type N. (Indeed, this becomes mandatory in coordinate structures such as *John and he*.) A case can be made for assigning infinitely many types to *John* by iterating this process. Must all these types be listed under *John* in the dictionary? Not at all, because we can prove that $N \to S/(N\backslash S)$ and $N \to (S/N)\backslash S$. For example, the former is proved as follows:

$$\cfrac{\cfrac{A\backslash C \to A\backslash C}{A \cdot (A\backslash C) \to C}}{A \to C/(A\backslash C)}$$

Linguists now call this process "type raising".

Another example of this sort of thing arises when we bracket *John (likes him)* in the more usual way. Since *likes* has type $N\backslash(S/N)$ and *likes him* has type $N\backslash S$, *him* can also be assigned type $(N\backslash(S/N))\backslash(N\backslash S)$, in addition to the type $(S/N)\backslash S$ determined earlier. Again, these two types are not independent, in view of the law $C\backslash A \to (C\backslash B)\backslash(C\backslash A)$, an easy consequence of the associative law, now known to linguists as "Geach's Law".

Let this suffice for illustrating what can be done in applying the associative syntactic calculus to natural languages. There are some obvious limitations to this method and my paper was ignored for many years. (The only thing that was ever quoted was an example pointing out that *time flies* can be construed in two different ways: as a statement and as a command.) It is only recently that linguists have become interested in the associative syntactic calculus, as witnessed by the work of van Benthem (1983, 1988), Buszkowski (1986) and others.

Let me now turn to another question. As long as only the rules $(C/B) \cdot B \to C$ and $A \cdot (A\backslash C) \to C$, already known to Bar-Hillel, were used, it was easy to decide whether, for given types A and B, one could prove that $A \to B$. For, as one proceeded from left to right, the types were getting progressively simpler. Such a decision procedure was not so obvious in the presence of such rules as $A \to B/(A\backslash B)$ and $A/B \to (A/C)/(B/C)$.

2. DECISION PROCEDURE

A decision procedure for the associative syntactic calculus was obtained by modifying a method by Gentzen (see, e.g., Kleene 1952 or Szabo 1969), who had replaced simple deductions $A \to B$ by so-called "sequents" $A_1 A_2 \ldots A_n \to B$.

Using capital Greek letters to denote strings of types, we may present the syntactic calculus in Gentzen–style by means of the following axiom scheme and rules of inference:

$$A \to A;$$

$$\frac{\Gamma \to A \qquad \Delta \to B}{\Gamma\Delta \to A \cdot B};$$

$$\frac{\Gamma AB\Delta \to C}{\Gamma A \cdot B\Delta \to C};$$

$$\frac{\Gamma B \to A}{\Gamma \to A/B}; \qquad \frac{B\Gamma \to A}{\Gamma \to B\backslash A}$$

$$\frac{\Theta \to B \quad \Gamma A\Delta \to C}{\Gamma A/B\Theta\Delta \to C}; \qquad \frac{\Theta \to B \quad \Gamma A\Delta \to C}{\Gamma\Theta B\backslash A\Delta \to C}.$$

Note that each of the rules of inference introduces one of the symbols $\cdot, /$ *or* $\backslash$ on the right or on the left of the arrow. These rules are easily verified if we interpret ABC, for example, as another way of writing $(A \cdot B) \cdot C$. The transivity of the arrow in the syntactic calculus gives rise to the so–called "cut":

$$\frac{\Theta \to A \qquad \Gamma A\Delta \to B}{\Gamma\Theta\Delta \to B}.$$

The *cut elimination theorem* says that any proof with cut can be replaced by one without cut. This is just as well, as the cut would cause difficulty when we are trying to look for a proof of $\Gamma\Theta\Delta \to B$, as we might have to try infinitely many choices for A. Here is an example of a cut– free proof:

$$\frac{\dfrac{N \to N \qquad S \to S}{N(N\backslash S) \to S}}{N \to S/(N\backslash S)}$$

The reader will also notice that the reflexive law $A \to A$ is used only for basic types.

The proof of the cut elimination theorem for the associative syntactic calculus is even easier than Gentzen's original proof for intuitionistic propositional logic, as for the latter he had to postulate three so–called structural rules, which caused some complications:

$$\frac{\Gamma AB\Delta \to C}{\Gamma BA\Delta \to C} \qquad \text{(interchange)},$$

$$\frac{\Gamma AA \to B}{\Gamma A \to B} \qquad \text{(contraction)},$$

$$\frac{\Gamma \to B}{\Gamma A \to B} \qquad \text{(thinning)}.$$

While all three structural rules are required for intuitionistic logic, only the first two are required for relevance logic (as pointed out by Minc 1977) and only interchange is required for a version of Girard's linear logic (1987). We repeat that none of Gentzen's structural rules are used for the syntactic calculus.

3. MULTICATEGORIES

I had pointed out in 1961 that Gentzen's sequent calculus was essentially the same as Bourbaki's (1958) method of bilinear maps. For argument's sake, let us assume we are dealing with $R - R$–bimodules for a given ring R. Bourbaki had described a canonical bilinear map $AB \to A \otimes B$ which induced a natural isomorphism between *bilinear* maps $AB \to C$ and *linear* maps $A \otimes B \to C$. This was exploited to obtain all properties of the tensor product.

More generally, if $\Gamma = A_1 \dots A_m$ and $\Delta = B_1 \dots B_n$ are sequences of $R - R$–bimodules (we leave out the commas), there is a natural isomorphism

$$Mult(\Gamma AB\Delta, C) \cong Mult(\Gamma A \otimes B\Delta, C)$$

between $m + n + 2-$ linear and $m + n + 1-$ linear maps. In the same spirit, we have natural isomorphisms

$$Mult(\Gamma B, C) \cong Mult(\Gamma, C \not\circ B),$$
$$Mult(A\Gamma, C) \cong Mult(\Gamma, A \not\backslash C).$$

Once the analogy between the methods of Gentzen and Bourbaki has been realized, it is quite natural to ask that the sequents in the Gentzen style presentation of the syntactic calculus be interpreted as some kind of multilinear maps. This was done in my 1968 paper, to be followed by a formal definition of " multicategories" in 1969.

A multicategory is supposed to be like a category, except that arrows $f : A \to B$ have been replaced by arrows $f : A_1 \dots A_n \to B$, sometimes called "multiarrows".

Formally, we first introduce a *multigraph* consisting of a class of arrows (also called "sequents") and a class of objects (also called "types") together with two mappings

$$\text{source: } \{\text{arrows}\} \longrightarrow \{\text{objects}\}^*,$$
$$\text{target: } \{\text{arrows}\} \longrightarrow \{\text{objects}\},$$

where $\{\text{objects}\}^*$ is the free monoid generated by the class of objects, its elements are strings $\Gamma = A_1 \dots A_n$ of objects. Note that n may be zero, in which case Γ is the empty string, which will here be denoted by a blank. An arrow $f : \ \to B$ is also called an *element* of B.

To obtain a Gentzen–style sequent calculus (but without his three structural rules) we impose the axiom

$$1_A : A \to A$$

and the rule of inference

$$\frac{f : \Theta \to A \qquad g : \Gamma A \Delta \to B}{g < f > : \Gamma \Theta \Delta \to B},$$

Gentzen's "cut". (Out of context, the notation $g < f >$ is ambiguous, as it does not indicate where f is "substituted" into g, but we shall use it nonetheless.) What we obtained up to now is a contextfree production grammar, but with arrows reversed, we shall call it a *contextfree recognition grammar*.

To obtain a *multicategory* it remains to impose an appropriate equivalence relation between arrows $f, g : \Gamma \to B$; we shall denote it by the equality symbol. We impose the following equations:

(1a) $$\frac{f : \Gamma \to A \quad 1_A : A \to A}{1_A < f > : \Gamma \to A} = f : \Gamma \to A;$$

(1b) $$\frac{1_A : A \to A \quad g : \Gamma A \Delta \to B}{g < 1_A > : \Gamma A \Delta \to B} = g : \Gamma A \Delta \to B;$$

(2) $$\frac{\dfrac{f : \Theta \to A \quad g : \Gamma A \Delta \to B}{g < f > : \Gamma \Theta \Delta \to B} \quad h : \Phi B \Psi \to C}{h < g < f \gg : \Phi \Gamma \Theta \Delta \Psi \to C}$$

$$= \frac{g : \Gamma A \Delta \to B \qquad \dfrac{f : \Theta \to A \qquad h < g > : \Phi \Gamma A \Delta \Psi \to C}{h < g >< f > : \Phi \Gamma \Theta \Delta \Psi \to C}}{}\ ;$$

(3) $$\frac{\dfrac{g : \Delta \to B \quad h : \Phi A \Theta B \Psi \to C}{h < g > : \Phi A \Theta \Delta \Psi \to C} \qquad f : \Gamma \to A}{h < g >< f > : \Phi \Gamma \Theta \Delta \Psi \to C}$$

$$= \frac{f : \Gamma \to A \quad h : \Phi A \Theta B \Psi \to C}{\dfrac{g : \Delta \to B \qquad h < f > : \Phi \Gamma \Theta B \Psi \to C}{h < f >< g > : \Phi \Gamma \Theta \Delta \Psi \to C}}\ .$$

These equations were called *identity* laws, *associative* law and *commutative* law respectively in my 1969 paper. (The inadequacy of the notation is apparent from the different proof–trees labelled $h < g >< f >$ in (2) and (3) above.)

Examples of multicategories are categories, trivially, but also monoidal categories and closed categories (Bénabou 1963), as we shall see below.

As a matter of notation: we shall write $Mult(\Gamma, A)$ for the set of (multi)arrows $\Gamma \to A$.

4. STRUCTURED MULTICATEGORIES

Instead of defining monoidal categories, closed categories etc., it is often more convenient to introduce operatives $\otimes, \emptyset$ and $\circ$ into a multicategory directly.

A *tensor product* in a multicategory is a binary operation $\otimes$ on objects together with a canonical family of (multi)arrows $m_{AB} : AB \to A \otimes B$ inducing a bijection

$$Mult(\Gamma A \otimes B \Delta, c) \xrightarrow{\sim} Mult(\Gamma A B \Delta, C).$$

Thus, to each arrow $f : \Gamma AB\Delta \to C$ there corresponds a unique arrow $f^{\S} : \Gamma A \otimes B\Delta \to C$ such that $f^{\S} < m_{AB} >= f$. The uniqueness of f may also be expressed equationally by saying that, for each $g : \Gamma A \otimes B\Delta \to C$, $(g < m_{AB} >)^{\S} = g$.

A *right internal hom* in a multicategory is a binary operation ϕ on objects together with a canonical family of arrows $e_{CB} : (C\phi B)B \to C$ inducing a bijection

$$Mult(\Gamma, C\phi B) \xrightarrow{\sim} Mult(\Gamma B, C).$$

Thus, to each arrow $f : \Gamma B \to C$ there corresponds a unique arrow $f^* : \Gamma \to C\phi B$ such that $e_{CB} < f^* >= f$. The uniqueness of f^* may be expressed equationally by saying that, for each $g : \Gamma \to C\phi B$, $(e_{CB} < g >)^* = g$.

A *left internal hom* is defined similarly by an operation $\backslash$ on objects, a family of arrows $e'_{AC} : A(A\backslash C) \to C$ and by assigning to each arrow $f : A\Gamma \to C$ an arrow $f^+ : \Gamma \to A\backslash C$ satisfying appropriate equations to ensure a bijection

$$Mult(\Gamma, A\backslash C) \xrightarrow{\sim} Mult(A\Gamma, C).$$

A *unity object* in a multicategory is an object I together with an arrow $i : \ \to I$ (an empty string on the left of the arrow) inducing a bijection

$$Mult(\Gamma I\Delta, C) \xrightarrow{\sim} Mult(\Gamma\Delta, C).$$

Thus, to each arrow $f : \Gamma\Delta \to C$, there corresponds a unique arrow $f^{\#} : \Gamma I\Delta \to C$ such that $f^{\#} < i >= f$. The uniqueness of $f^{\#}$ may be expressed by saying that, for each $g : \Gamma I\Delta \to C$, $(g < i >)^{\#} = g$.

We list some of the structured multicategories that have appeared in the literature: a *monoidal* multicategory has $\otimes$ and I, a *right closed* multicategory has ϕ, a *left closed* multicategory has $\backslash$, a *biclosed* multicategory has ϕ and $\backslash$, a *residuated* multicategory has $\otimes$, ϕ and $\backslash$.

If one wants to add the logical operations $\wedge, \top, \vee, \bot$ to the syntactic calculus, one may of course do so. For multicategories, these symbols are better replaced by $\times, 1, +, 0$, the *Cartesian product, terminal object, coproduct* and *initial object* respectively. The Cartesian product comes equipped with canonical projections $\pi_{AB} : A \times B \to A$, $\pi'_{AB} : A \times B \to B$ and the coproduct with canonical injections $\kappa_{AB} : A \to A + B$, $\kappa'_{AB} : B \to A + B$. The following bijections are assumed to be induced:

$$Mult(\Gamma, A \times B) \xrightarrow{\sim} \quad Mult(\Gamma, A) \times Mult(\Gamma, B),$$

$$Mult(\Gamma, 1) \xrightarrow{\sim} \{*\},$$

$$Mult(\Gamma A + B\Delta, C) \xrightarrow{\sim} \quad Mult(\Gamma A\Delta, C) \times Mult(\Gamma B\Delta, C),$$

$$Mult(\Gamma 0\Delta, C) \xrightarrow{\sim} \{*\}.$$

Here $\{*\}$ is taken to be a prototypical one-element set.

For expository purposes, let us now confine attention to residuated multicategories (as in my paper of 1968), almost the same as biclosed monoidal multicategories (as in my 1969 paper) but without a unity object. A residuated category is nothing else than a sequent presentation of the syntactic calculus with a suitable equivalence relation between sequents.

Given a multicategory $\mathcal{X}$, one may consider the *free* residuated multicategory $F(\mathcal{X})$ generated by $\mathcal{X}$. By now everybody knows how to construct this, so let us immediately turn to the problem of calculating $Mult(\Gamma, B)$ for given $\Gamma = A_1 \dots A_n$ and B in $F(\mathcal{X})$. The problem breaks into two parts:

(I) Find all sequents $\Gamma \to B$ in the Gentzen–style syntactic calculus freely generated by $\mathcal{X}$.
(II) Decide when two sequents $f, g : \Gamma \to B$ determine the same arrow in $F(\mathcal{X})$.

(I) is answered by the cut elimination theorem discussed above. My answer to (II) in 1968 was only partly correct, namely when $\mathcal{X}$ is *discrete*, that is, a multicategory with no arrows except the obligatory $1_x : x \to x$ for each object x. What I should have done was to reduce f and g to normal form. This approach was in fact followed by various authors who have been working on related multicategories since, to my knowledge Szabo, Mann, Minc and his students. I hope to return to this problem on another occasion.

Incidentally, the cut elimination theorem for residuated multicategories says that every arrow of the free residuated multicategory constructed with cut is actually equal to one constructed without cut, not just that it can be replaced by the latter. Similar results hold for other structured multicategories.

5. TYPE LANGUAGES

It has become fashionable in the past few years to look at internal languages of various categories, e.g., toposes and Cartesian closed categories (Lambek and Scott 1986) and, most recently, monoidal categories (Barry Jay 1987). One may just as easily consider internal languages of multicategories and even define multicategories linguistically, as we propose to do here.

DEFINITION 5.1. A *type language* consists of two classes, the class of *types* and the class of *terms*, together with a mapping from the latter to the former assigning a unique type to each term. Among the terms there are countably many *variables* of each type A, say $x_1^A, x_2^A, \dots$. (Instead of specifying $x = x_i^A$, we usually just say: let x be a variable of type A.)

The class of terms is built up inductively from the variables by means of a given class of *operation symbols* $f : A_1 \dots A_n \to B$, the A_i and B being types, as follows: if a_i is a term of type $A_i (i = 1, \dots, n)$ and f is an operation symbol as shown, then $f a_1 \dots a_n$ is a term of type B. (If $n = 0$, then f is just a *constant* term of type B.) Moreover, for any finite set X of variables, there is a *congruence relation* $=_X$ between terms which contain no (free) variables other than members of X. By "congruence relation" we mean an equivalence relation satisfying the following substitution rule (substituting equals for equals):

(EO) if $f : A_1 \dots A_n \to B$ is any operation symbol and a_i, a_i' are terms of type A_i such that $a_i =_X a_i' \quad (i = 1, \dots, n)$, then $f a_1 \dots a_n =_X f a_1' \dots a_n'$.

Finally, $=_X$ is supposed to satisfy the obvious rule:

(E1) if $X \subseteq Y$, then from $a =_X b$ one may infer $a =_Y b$,

as well as the following substitution rule (substituting terms for variables):

(E2) if $z \notin X$, then from $a =_{X \cup \{z\}} b$ one may infer $[c/z]a =_X [c/z]b$.

Here $[c/z]a$ denotes the result of substituting c of type C for each (free) occurrence of the variable z of type C in a, and z is assumed to be outside X.

The rules (E1) and (E2) permit, in particular, change of (free) variable. Thus, for example, from $a =_z b$ we may obtain $a =_{\{z,z'\}} b$ by (E1), whence $[z'/z]a =_{z'} [z'/z]b$ by (E2), assuming that z and z' are variables of the same type.

Is it really necessary to write $=_X$ in place of $=$? Indeed, suppose a and b to be terms of type A not containing any free variables. Certainly, from $a = b$ we can infer $a =_z b$, where z is a variable of type C, in view of (E1). Conversely, from $a =_z b$ we can infer $a = b$, by (E2), by substituting for z some term c of type C, assuming, of course, that there are terms of type C. (Here $=$ means $=_\emptyset$.)

Thus, if there exist constant terms of each type, it is not necessary to place the subscript X on the equality sign (hence rule (E1) need not be stated explicitly). There may be other cases where it is not necessary to write $=_X$ in place of $=$.

For future reference, it will be convenient to make the following definition.

DEFINITION 5.2. Two operation symbols $f, g : A_1 \dots A_n \to B$ are said to denote the same *operations* if

$$(*) \qquad f x_1 \dots x_n =_X g x_1 \dots x_n,$$

where $X = \{x_1, \dots, x_n\}$, the x_i being distinct variables of type $A_i (i = 1, \dots, n)$. (Even if $A_i = A_j$, we insist that $x_i \neq x_j$, as long as $i \neq j$.) We write $f = g$.

The reader will be able to think of many examples of type languages. The main example we shall be concerned with here is the internal language of a multicategory or, more generally, the type language associated with a contextfree recognition grammar.

DEFINITION 5.3. The *internal language* of a contextfree recognition grammar (or of a multicategory) is defined as follows. Its types are the objects of the grammar and its operation symbols are the arrows (sequents). The identity arrow $1_A : A \to A$ counts as an operation symbol and the cut

$$\frac{f : \Theta \to A \qquad g : \Gamma A \Delta \to B}{g < f > : \Gamma \Theta \Delta \to B}$$

produces a new operation symbol $g < f >$ from two given operation symbols f and g. Let $=_X$ be the smallest congruence relation (actually function from finite sets of variables to congruence relations) between terms of the same type; containing at most the variables in the set X, satisfying (E1) and (E2) as well as the following:

(E3) $$1_A x =_x x,$$

where x is a variable of type A, and

(E4) $$g < f > u_1 \dots u_m x_1 \dots x_k v_1 \dots v_n =_X g u_1 \dots u_m f x_1 \dots x_k v_1 \dots v_n,$$

where $X = \{u_1, \dots, u_m, x_1, \dots x_k, v_1, \dots, v_n\}$, the u_i, x_l and v_j being variables of the types occuring in the strings Γ, Θ and Δ respectively.

Note that the cut is thus interpreted as substituting $f x_1 \dots x_k$ for x in $g u_1 \dots u_m x v_1 \dots v_n$. (It appears that here is one situation where we might as well have written $=$ in place of $=_X$. Anyway, from now on, subscripts will be omitted.)

6. RESTRICTED FUNCTIONAL COMPLETENESS

Recalling how the terms of a type language are defined inductively from the variables by means of operation symbols, we infer the following completeness property for the internal language of a contextfree grammar.

THEOREM 6.1 The internal language of a contextfree recognition grammar, hence of a multicategory, satisfies *restricted functional completeness*: given any term $\varphi(x_1, \ldots, x_n)$ which contains no free variables other than x_i of type $A_i (i = 1, \ldots, n)$, assume that these variables occur

(a) in this order,
(b) without repetition,
(c) explicitly,

then there exists an operation symbol (sequent) $f : A_1 \ldots A_n \to B$ such that

$$(+) \qquad f x_1 \ldots x_n = \varphi(x_1, \ldots, x_n),$$

where the subscript $X = \{x_1, \ldots, x_n\}$ has been omitted. Moreover, the operation defined by f is uniquely determined.

The restrictions (a),(b) and (c) on functional completeness are due to the absence of Gentzen's three structural rules. We shall see later what happens when some or all of these restrictions are removed. In any case, restriction (b) can be relaxed somewhat for the existence of f. In view of (E2) of Section 5, we may substitute the same variable for two variables of the same type. Hence repetition of variables is permitted in (+) as long as each occurrence is counted separately.

Proof: Let $a \equiv \varphi(x_1, \ldots, x_n)$ be any term of type A in which the variables $x_1, \ldots, x_n$ occur explicitly in this order and different occurrences of the same variable are listed separately. To each such term a we assign a sequent $\bar{a} : A_1 \ldots A_n \to A, A_i$ being the type of $x_i (i = 1, \ldots, n)$, defined by induction on the length of a as follows:

if $a \equiv x$, a variable of type A, $\bar{a} \equiv 1_A$;

if $a \equiv f a_1 \ldots a_m$, $\quad \bar{a} \equiv f < \bar{a}_1 > \ldots < \bar{a}_m >$.

We prove by induction that

$$\bar{a} x_1 \ldots x_n = \varphi(x_1, \ldots, x_n).$$

Indeed, if $a \equiv x$,

$$\bar{a} x = 1_A x = x,$$

and, if $a \equiv f a_1 \ldots a_m$,

$$\begin{aligned} \bar{a} x_1 \ldots x_n &= f \bar{a}_1 x_1 \ldots x_{r_1} \bar{a}_2 x_{r_1+1} \ldots x_{r_2} \ldots \bar{a}_k x_{r_{k-1}+1} \ldots x_m \\ &= f a_1 a_2 \ldots a_m, \end{aligned}$$

by inductional assumption.

Finally, if f satisfies (+), then the operation defined by f is uniquely determined in view of Definition 5.2.

Let us also remark that in a language which satisfies restricted functional completeness we need not distinguish between operations $f : \quad \to B$ whose source is the empty string and constant terms b of type B. Indeed, it follows from (+) with $n = 0$ and X empty that, given any term b of type B, there is an operation symbol f such that $f = b$.

Moreover, equality of terms coincides with equality of operations in this case.

For future reference we require a lemma.

LEMMA 6.2. In the internal language of a multicategory, if a' results from a by replacing all k free occurrences of the variable z by the closed term c of type C, so that $a' \equiv [c/z]a$, then $\bar{a}' = \bar{a} < \bar{c} >^k$, the notation indicating k cuts.

Proof: If $a \equiv z$ then $a' \equiv c$, so $\bar{a'} = \bar{c} = 1_A < \bar{c} >= \bar{a} < \bar{c} >$. Suppose that $a \equiv f a_1 \dots a_m$, put $a_i' \equiv [c/z]a_i$, then $\bar{a_i'} = \bar{a_i} < \bar{c_i} >^{k_i}$, by inductional assumption, where k_i is the number of occurrences of z in a_i. Therefore

$$\begin{aligned} \bar{a'} &= f \bar{a_1'} \dots \bar{a_m'} \\ &= f \bar{a_1} < \bar{c} >^{k_1} \dots \bar{a_m} < \bar{c} >^{k_m} \\ &= f \bar{a_1} \dots \bar{a_m} < \bar{c} >^{k} \\ &= \bar{a} < \bar{c} >^{k}, \end{aligned}$$

using the equations of a multicategory.

7. THE MULTICATEGORY GENERATED

DEFINITION 7.1. A type language clearly generates a multigraph, its arrows being the operation symbols $f : A_1 \dots A_m \to B$. If the type language satisfies restricted functional completeness, this multigraph is a contextfree recognition grammar: the sequent $1_A : A \to A$ is the operation symbol 1_A defined by $1_A x = x$ and the cut

$$\frac{f : \Theta \to A \quad g : \Gamma A \Delta \to B}{g < f > : \Gamma \Theta \Delta \to B}$$

is defined by

$$g < f > u_1 \dots u_m \; x_1 \dots x_k \; v_1 \dots v_n$$

$$= \; g \; u_1 \dots u_m \; f \; x_1 \dots x_k \; v_1 \dots v_n.$$

In fact, this contextfree recognition grammar is a multicategory.

Indeed, the equations of a multicategory are easily checked. For example, the equation $g < 1_A >= g$ holds because

$$g < 1_A > u_1 \dots u_m \; x \; v_1 \dots v_n \quad = \; g \; u_1 \dots u_m \; 1_A \; x \; v_1 \dots v_n$$

and $1_A x = x$.

What if the type language was already the internal language of a multicategory, which satisfies restricted functional completeness by Theorem 6.1?

THEOREM 7.2. The multicategory generated by the internal language of a multicategory $\mathcal{M}$ is $\mathcal{M}$.

Proof: It suffices to show that, if two arrows $f, g : A_1 \dots A_n \to B$, when regarded as operation symbols of the internal language, define the same operations (see Definition 5.2.), then $f = g$ in $\mathcal{M}$.

We begin by defining an equivalence relation $\cong$ between terms of the same type in the internal language: $a \cong b$ means that a and b contain the same variables taken from the set X in the same order (repetition is allowed as long as each occurrence counts separately) and that $\bar{a} = \bar{b}$, according to the definition of $\bar{a}$ in the proof of Theorem 6.1, that is, a and b determine the same operations.

We show that $\cong$ is a congruence relation,that is, it satisfies (E0) of Section 5. For, if f, a_i, b_i are given such that $a_i \cong b_i$, then a_i and b_i contain the same variables in the same order and $\bar{a}_i = \bar{b}_i$. Therefore also $a = fa_1 \dots a_n$ and $b = fb_1 \dots b_n$ contain the same variables and

$$\bar{a} = f < \bar{a}_1 > \dots < \bar{a}_n >= f < \bar{b}_1 > \dots < \bar{b}_n >= \bar{b}.$$

(E1) is vacuous as long as we omit the subscripts X and Y.

To check (E2), suppose $a \cong b$. Then, a given variable $z \notin X$ will occur as often in a as in b, say k times, and in the same places. Moreover, $\bar{a} = \bar{b}$. Let $a' = [c/z]a$ and $b' = [c/z]b$, where c does not contain z. We claim that $a' \cong b'$. Indeed, a' and b' contain the same variables in the same order and, by Lemma 6.2,

$$\bar{a'} = \bar{a} < \bar{c} >^k = \bar{b} < \bar{c} >^k = \bar{b'}.$$

Finally, we must show that $\cong$ satisfies (E3) and (E4), namely

$$1_A x \cong x, \quad g < f > u_1 \dots u_m x_1 \dots x_k v_1 \dots v_n \cong g u_1 \dots u_m f x_1 \dots x_k v_1 \dots v_n.$$

To check the former, for example, we note that

$$\overline{1_A x} = 1_A < \bar{x} >= 1_A < 1_A >= 1_A = \bar{x}.$$

The latter is verified similarly.

To complete the proof of the theorem, suppose that $f, g : A_1 \dots A_n \to B$ define the same operation, that is

$$f\ x_1 \dots x_n = g\ x_1 \dots x_n,$$

x_i being of type A_i, without repetition. Recall that $=$ (actually $=_X$) was the smallest congruence relation satisfying (E1) to (E4) and that we have just proved that $\cong$ is also such. Therefore,

$$b \equiv fx_1 \dots x_n \cong gx_1 \dots x_n \equiv b'.$$

In particular, $\bar{b} = \bar{b'}$, that is,

$$f < 1_{A_1} > \dots < 1_{A_n} >= g < 1_{A_1} > \dots < 1_{A_n} >,$$

that is, $f = g$, as was to be proved.

At first sight, there is something puzzling about the above argument. Suppose $f, g : AA \to B$ and $fxx = gxx$, then the same argument will show that $f = g$. How come? The answer is that, from the manner in which $=$ was defined, it follows that any proof of $fxx = gxx$ will generalize to a proof of $fxx' = gxx'$.

To sum up, let $L(\mathcal{M})$ be the internal language of a multicategory $\mathcal{M}$ and $M(\mathcal{L})$ the multicategory generated by the type language $\mathcal{L}$, which is assumed to satisfy restricted

functional completeness. Then Theorem 7.2 asserts that

$$M(L(\mathcal{M})) = \mathcal{M}.$$

In other words, a multicategory is completely determined by its internal language. It would be nice if we could also assert that

$$L(M(\mathcal{L})) = \mathcal{L}.$$

This is certainly the case if $\mathcal{L}$ is the internal language of some multicategory. The following appear to be necessary and sufficient conditions on $\mathcal{L}$ for this to be so:

(a) $\mathcal{L}$ satisfies restricted functional completeness;
(b) equality between terms (of the same type) is the smallest congruence relation respecting (E2) to (E4);
(c) operation symbols defining the same operations are equal.

8. INTERNAL LANGUAGE AND TENSOR PRODUCT

Recall that a tensor product in a multicategory is a binary operation $\otimes$ between objects together with a family of arrows $m_{AB} : AB \to A \otimes B$ inducing a bijection

$$Mult(\Gamma A \otimes B\Delta, C) \xrightarrow{\sim} Mult(\Gamma AB\Delta, C)$$

for given strings Γ, Δ and objects A, B and C. Thus, for each arrow $f : \Gamma AB\Delta \to C$ there is given an arrow $f^{\S} : \Gamma A \otimes B\Delta \to C$ satisfying

$$f^{\S} < m_{AB} >= f, \quad (g < m_{AB} >)^{\S} = g,$$

for each $g : \Gamma A \otimes B\Delta \to C$.

In the internal language, it is convenient to abbreviate $m_{AB}xy$ as (x, y). (Some people write $x \otimes y$, but this may lead to confusion with the tensor product of arrows introduced below.)

The above bijection may now be expressed by saying that to each arrow $f : \Gamma AB\Delta \to C$ there is associated a unique arrow $f^{\S} : \Gamma A \otimes B\Delta \to C$ such that

$$f^{\S} x_1 \dots x_m (x, y) y_1 \dots y_n = f x_1 \dots x_m xy\ y_1 \dots y_n.$$

(As above, we omit the subscript on the equality sign when there is no likelihood of confusion.) From this the usual properties of the tensor product are easily deduced.

For example, if $f : A \to A'$ and $g : B \to B'$ are given arrows, we may define $f \otimes g : A \otimes B \to A' \otimes B'$ by putting $f \otimes g \equiv h^{\S}$, where $h : AB \to A' \otimes B'$ is the unique arrow such that $h\ xy = mfxgy$, whose existence is assured by restricted functional completeness. In other words, we define $f \otimes g$ by the equation

$$(f \otimes g)(x, y) = (fx,\ gy).$$

To show that $\otimes$ is a bifunctor we require, for example, that

$$(f' \otimes g')(f \otimes g) = f'f \otimes g'g,$$

whenever $f' : A' \to A''$ and $g' : B' \to B''$. We prove this by calculating

$$\begin{aligned}(f' \otimes g')(f \otimes g)(x, y) &= (f' \otimes g')(fx,\ gy) \\ &= (f'fx,\ g'gy) \\ &= (f'f \otimes g'g)(x,\ y).\end{aligned}$$

The associativity arrow $\alpha_{ABC} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$ is defined by the equation

$$\alpha_{ABC}((x, y), z) = (x, (y, z)).$$

It is easily checked that α is a natural transformation. Similarly one defines $\alpha^{-1}_{ABC} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$ and checks that $\alpha_{ABC}\alpha^{-1}_{ABC} = 1_{A\otimes(B\otimes C)}, \alpha^{-1}_{ABC}\alpha_{ABC} = 1_{(A\otimes B)\otimes C}$. MacLane's pentagonal coherence condition is the commutativity of the following diagram:

$$\begin{array}{ccccc} ((A\otimes B)\otimes C)\otimes D & \to & (A\otimes B)\otimes(C\otimes D) & \to & A\otimes(B\otimes(C\otimes D)) \\ \downarrow & & & & \uparrow \\ (A\otimes(B\otimes C))\otimes D & & \longrightarrow & & A\otimes((B\otimes C)\otimes D) \end{array}$$

The way to prove this in the internal language is to point out that there is a unique arrow $f : ((A \otimes B) \otimes C) \otimes D \to A \otimes (B \otimes (C \otimes D))$ satisfying the equation

$$f(((x, y), z), t) = (x, (y, (z, t))).$$

We thus see that the usual properties of the tensor product in a category follow from the canonical bijection between arrows $\Gamma A \otimes B \Delta \to C$ and $\Gamma AB\Delta \to C$ in a multicategory. Of course, the traditional procedure is the reverse: one postulates that $\otimes$ is a bifunctor with a coherent natural associativity defined in a category. One then turns the category into a multicategory by defining $f : ABCD \to E$ as $f : ((A\otimes B)\otimes C)\otimes D \to E$ (to take the case $n = 4$ for example). The equations of a multicategory are then easily checked. However, in the spirit of Gentzen and Bourbaki, I prefer to go in the opposite direction, beginning with a multicategory.

We leave it as an exercise to the reader to explore the properties of a unity object in the internal language as follows:

$$i : \ \to I\ ,\ \frac{f : \Gamma\Delta \to C}{f^{\#} : \Gamma I\Delta \to C}\ ,\ f^{\#} < i >= f,\ (g < i >)^{\#} = g.$$

Let me draw attention here to the internal language of a monoidal category recently developed by Barry Jay (1987) and to the interesting applications made by him. His approach is somewhat different from ours, as the terms of his language are not allowed to contain repetitions of the same variable.

9. INTERNAL LANGUAGE AND INTERNAL HOM

We recall that a right internal hom in a multicategory is a binary operation ϕ between objects together with a family of arrows $e_{CB} : (C\phi B)B \to C$ inducing a bijection

$$Mult(\Gamma, C\phi B) \xrightarrow{\sim} Mult(\Gamma B, C).$$

Thus, to each arrow $f : \Gamma B \to C$ one assigns an arrow $f^* : \Gamma \to C \phi B$ satisfying

$$e_{CB} < f^* >= f, \quad (e_{CB} < g >)^* = g$$

for each $g : \Gamma \to C \phi B$.

In the internal language it is convenient to abbreviate $e_{CB} uy$ as $u`y$, read "u of y", where u and y are variables of types $C \phi B$ and B respectively. It is also customary to write $f^* x_1 \ldots x_n$ as $\lambda_{y \in B} f x_1 \ldots x_n y$, where y is a so– called "bound" variable, assumed to be distinct from the x_i. The above equations then become:

$$(\lambda_{y \in B} f x_1 \ldots x_n y) `y' = f x_1 \ldots x_n y',$$

$$\lambda_{y \in B} (g x_1 \ldots x_n `y) = g\, x_1 \ldots x_n.$$

For the reader's convenience, we have here distinguished the free variable y' of type B from the bound variable y of the same type, but this was not necessary. In any case, rule (E2) of Section 5 allows us to replace y' by any term b of type B, so the first equation may be generalized to

$$(\lambda_{y \in B} f x_1 \ldots x_n y) `b = f x_1 \ldots x_n b.$$

While this resembles the usual λ–calculus, it differs from it inasmuch as λ –abstraction applies only to the rightmost free variable occurring in an expression and that this variable is assumed to be distinct from all the others.

We can now show that ϕ is a bifunctor, contravariant in its second argument. Thus, given $f : A \to A'$ and $g : B' \to B$, we define $f \phi g : A \phi B \to A' \phi B'$ by the equation

$$((f \phi g) u) `y' = f(u `(g y')),$$

where u and y' are variables of types $A \phi B$ and B' respectively. In other words,

$$(f \phi g) u = \lambda_{y' \in B'} u `(g\, y').$$

If furthermore $f' : A' \to A''$ and $g' : B'' \to B'$, one easily checks that

$$(f' \phi g')(f \phi g) u = ((f' f) \phi (g g')) u.$$

We can also obtain a canonical arrow $\beta_{ABC} : A \phi B \to (A \phi C) \phi (B \phi C)$, expressing Geach's law, by putting

$$\beta_{ABC} u = \lambda_{v \in B \phi C}\ \lambda_{z \in C} u `(v `z),$$

where u is a variable of type $A\ \phi B$. In this way, one can obtain all the properties of ϕ that may turn out to be useful.

Conversely, given a category with a bifunctor ϕ, covariant in the first and contravariant in the second argument, it is an exercise, albeit somewhat tedious, to write down all the properties of ϕ which are needed to show that one obtains a multicategory by defining $f : ABCD \to E$ as $f : A \to ((E \phi D) \phi C) \phi B$, to take the case $n = 4$ for example.

If one is given a multicategory with both right internal hom and tensor product, it is immediate that one has a natural bijection

$$Mult(A \otimes B, C) \cong Mult(AB, C) \cong Mult(A, C \phi B),$$

so that $- \otimes B$ is left adjoint to $- \phi B$.

A left internal hom may be introduced dually by an arrow $e'_{AC} : A(A\backslash C) \to C$ which induces a bijection

$$Mult(\Gamma, A\backslash C) \xrightarrow{\sim} Mult(A\Gamma, C).$$

We write $e'_{AC}xv$ as $x'v$, think of "x substituted into v", and $f^+x_1 \dots x_n$ as $\lambda'_{x\in A} f x\, x_1 \dots x_n$. λ'–abstraction thus applies to the leftmost free variable in an expression.

In a multicategory which has both right and left internal hom, one obtains canonical arrows $\tau_{AB} : A \to B\oslash(A\backslash B)$ and $\tau'_{AB} : A \to (B\oslash A)\backslash B$, type raising, given by

$$\tau_{AB}x = \lambda_{u\in A\backslash B}(x'u),$$

for example. Also one can define an associativity isomorphism $\gamma_{ABC} : (A\backslash B)\oslash C \to A\backslash(B\oslash C)$ by

$$\gamma_{ABC}\omega = \lambda'_{x\in A}\lambda_{z\in C}(x'(\omega' z)),$$

where ω is a variable of type $(A\backslash B)\oslash C$.

10. GENTZEN MULTICATEGORIES

We shall take another look at Gentzen's structural rules, which we had discussed earlier. (See Szabo 1974.)

DEFINITION 10.1. A *Gentzen deductive system* is a contextfree recognition grammar which satisfies the three standard rules:

$$\frac{f : \Gamma A B \Delta \to C}{f^i : \Gamma B A \Delta \to C} \quad \text{(interchange)},$$

$$\frac{f : \Gamma A A \to B}{f^c : \Gamma A \to B} \quad \text{(contraction)},$$

$$\frac{f : \Gamma \to B}{f^t : \Gamma A \to B} \quad \text{(thinning)}.$$

By a *Gentzen multicategory* we shall understand a multicategory with operation symbols f^i, f^c, f^t, introduced as above, satisfying the following equations:

$$f^i x_1 \dots x_m y x\; y_1 \dots y_n = f x_1 \dots x_m\; x y\; y_1 \dots y_n,$$
$$f^c x_1 \dots x_m x = f x_1 \dots x_m x\; x,$$
$$f^t x_1 \dots x_m x = f x_1 \dots x_m.$$

In particular, in a multicategory, from fxy, gxx' and hx, where x and x' have the

same type, we can form

$$f^i yx = fxy, \quad g^c x = gxx, \quad h^t xy = hx$$

thus allowing permutations of arguments, diagonal maps and projections. One might point out that the possibility to construct diagonal maps lies at the bottom of many interesting arguments in mathematics: for example, Cantor's diagonal argument, Gödel's incompleteness proof and the construction of a non–calculable numerical function.

PROPOSITION 10.2. The internal language of a Gentzen multicategory satisfies unrestricted *functional completeness:*

Given any term $\varphi(x_1, \ldots, x_n)$ which contains no variables other than the x_i of type $A_i (i = 1, \ldots, n)$, there exists a unique operation $f : A_1 \ldots A_n \to B$ such that

$$fx_1 \ldots x_n = \varphi(x_1, \ldots, x_n).$$

Here the x_i need not occur in this order, they may occur at several places and they need not occur explicitly.

Thus we have dropped the restrictions (a), (b) and (c) of Section 6 completely. We leave the proof to the reader.

A Gentzen deductive system is of course the underlying deductive system of intuitionistic logic. A Gentzen multicategory is nothing else than a *manysorted algebraic theory*, in the sense of Lawvere (1963) and Bénabou (1968). (See also Birkhoff and Lipson, 1970.) In retrospect, Michael Barr's (1978) use of algebraic theories as analogues of grammars in his illustration of syntax acquisition was not so farfetched.

PROPOSITION 10.3 If a Gentzen multicategory is equipped with a tensor product, this will be a Cartesian product. If a Gentzen multicategory is equipped with a unity object, this will be a terminal object. If a Gentzen multicategory is right closed, it is also left closed; in fact BoA may be taken to be $A \phi B$ and viewed as exponentiation A^B.

Proof: Given a tensor product in a multicategory, we may define $\pi_{AB} : A \otimes B \to A$ and $\pi'_{AB} : A \otimes B \to B$ by

$$\pi_{AB}(x, y) = x \quad , \pi'_{AB}(x, y) = y.$$

where x and y are variables of types A and B respectively. Also, given $f : \Gamma \to A$ and $g : \Gamma \to B$, we may define $< f, g >: \Gamma \to A \otimes B$ by

$$< f, g > x_1 \ldots x_n =< fx_1 \ldots x_n, \; gx_1 \ldots x_n >,$$

where $\Gamma = A_1 \ldots A_n$ and the x_i are variables of type $A_i (i = 1, \ldots, n)$. Then

$$\pi_{AB} < f, g > x_1 \ldots x_n = fx_1 \ldots x_n,$$

so that $\pi_{AB} < f, g >= f$, by Definition 5.2, and similarly $\pi'_{AB} < f, g >= g$. Also

$$\begin{aligned} < \pi_{AB}, \pi'_{AB} > (x, y) &= (\pi_{AB}(x, y), \pi'_{AB}(x, y)) \\ &= (x, y), \end{aligned}$$

so that $< \pi_{AB}, \pi'_{AB} >= 1_{A\otimes B}$. This is so in view of Section 8, because there is a unique arrow $m^{\S} : A \otimes B \to A \otimes B$ such that $m^{\S}(x, y) = mxy = (x, y)$.

This shows that $A \otimes B$ is a Cartesian product with projections π_{AB} and π'_{AB}. We leave the proof that a unity object is terminal to the reader.

Suppose our multicategory is right closed. Define $B\backslash\!\!\backslash A \equiv A \not\circ B$ and $e'_{BA} : B(B\backslash\!\!\backslash A) \to A$ by

$$e'_{BA}\, yu = e_{AB}\, uy,$$

where y and u are variables of types B and $A \not\circ B$ respectively. (Thus $e'_{BA} = e^{i}_{AB}$.) Also put

$$\lambda'_{y\epsilon B} f y\ x_1 \dots x_n \equiv \quad \lambda_{y\epsilon B} f^{i} x_1 \dots x_n y,$$

where f^{i} results from f by interchange. Then

$$\begin{aligned} e'_{BA}\, y' \lambda'_{y\epsilon B} f y\ x_1 \dots x_n &= e_{AB} \lambda'_{y\epsilon B} f y\ x_1 \dots x_n y' \\ &= e_{AB} \lambda_{y\epsilon B} f^{i} x_1 \dots x_n y\ y' \\ &= f^{i} x_1 \dots x_n y' \\ &= f\ y' x_1 \dots x_n. \end{aligned}$$

The other equation involving λ' may be shown similarly.

11. SEMANTICS OF BICLOSED MULTICATEGORIES.

The internal language of a *closed* Gentzen multicategory (right closed suffices, in view of interchange) is the usual λ–calculus. For example, we may define

$$\begin{aligned} \lambda_{y\epsilon B} f x y z &\equiv \lambda_{y\epsilon B} f^{i} x z y, \\ \lambda_{y\epsilon A} g x x &\equiv \lambda_{x\epsilon A} g^{c} x, \\ \lambda_{y\epsilon B} h x &\equiv \lambda_{y\epsilon B} h^{t} x y. \end{aligned}$$

While intuitionistic logic and the associated Cartesian closed multicategories make use of all three structural rules (and the associated equations), the syntactic calculus and the associated residuated (or even biclosed monoidal) multicategories use none of them. Two intermediate cases have appeared in the literature.

The linear logic of Girard (1987) and the associated symmetric monoidal multicategories allow the interchange rule but neither contraction nor thinning. This connection had been noticed by Minc (1977), who called it "relevance logic" in place of "linear logic". However, it seems that the usual relevance logic allows not only interchange, but also contraction, forbidding the thinning rule only (see Anderson and Belnap 1975).

Incidentally, what has been called "λ–calculus" here was called "λK–calculus" by Church (1941). He reserved "λ–calculus" for what corresponds to relevance logic: the thinning rule is absent and the definition of $\lambda_{y\epsilon B} hx$ is consequently not permitted, if x is distinct from y.

Let us now take another look at the linguistic application discussed in Section I, namely the attempt to describe English syntax by means of the syntactic calculus. We

shall consider types built up from basic types S, N and perhaps others with the help of two binary operations "over" and "under". It will do no harm to forget about the operation "times", as long as we stick to the Gentzen–style presentation of Section 2. If we take account of the actual derivations of sentencehood and if we specify when two derivations are to be regarded as equivalent, we are led to study a biclosed multicategory. To be precise, we are interested in the *biclosed multicategory freely generated by the dictionary* (of English).

The *dictionary* is assumed to list certain arrows, whose source is the empty string:

$$\begin{aligned} &\textit{John}: && \longrightarrow N, \\ &\textit{works}: && \longrightarrow N\backslash S, \\ &\textit{likes}: && \longrightarrow (N\backslash S)/N, \end{aligned}$$

etc. For argument's sake, we shall here ignore the ambiguity of words such as *works*, which could just as well be the plural of a count noun, as this problem can easily be taken care of with the help of subscripts.

By the *syntactic λ–calculus* of English we shall mean the internal language of the biclosed multicategory freely generated by the dictionary. English words appear as constant terms in the syntactic λ–calculus. For example, *John* is of type N and *works* is of type $N\backslash S$, from which we infer that *John' works* has type S.

So much about English syntax. Its semantics is best described by a closed Gentzen multicategory or, equivalently, its internal language, the usual λ–calculus, also known as combinatory logic. This had been advocated for a long time by Curry (see e.g. Curry and Feys 1958) and was carried out more recently by Montague (see his selected papers 1974).

Since both syntax and semantics are biclosed multicategories, an *interpretation* of English should be a "biclosed multifunctor" (for a definition of this term see below). As the syntax is freely generated by the dictionary, an interpretation is completely determined by the interpretation of the basic types and the English words listed in the dictionary.

In a first attempt to find an interpretation of English we might take the semantics to be the multicategory of sets. Then S is interpreted by the set $\overline{S} = \{\top, \bot\}$ of truth values and N is interpreted by a given set $\overline{N}$ of individuals. Thus *John* will be interpreted as a certain individual who bears that name and *works* as a function from $\overline{N}$ to $\overline{S}$, that is, as a property of individuals.

This attempt at interpreting English is however not very realistic. Thus, whether John works is not just true or false but depends on time and other circumstances. It may be more reasonable to take for the semantics not the multicategory of sets, but some topos, for example Sets $^{\mathcal{T}^{op}}$, where $\mathcal{T}$ is a tree (or even a forest), allowing the present to develop into a number of possible futures, the past being fixed.

But even this attempt at interpreting English ignores one aspect of Montague semantics, as he insists that the language in which the interpretation takes place is an intensional language. We shall not pursue this matter any further here. Instead we shall end this section with the definition that had been promised.

DEFINITION 11.1 A *multifunctor* F between multicategories $\mathcal{M}$ and $\mathcal{M}'$ is a mapping from the class of objects of $\mathcal{M}$ to that of $\mathcal{M}'$ together with a mapping from the class of arrows of $\mathcal{M}$ to that of $\mathcal{M}'$ such that, when $f : A_1 \dots A_n \to B$ in $\mathcal{M}$ then $F(f) : F(A_1) \dots F(A_n) \to F(B)$ in $\mathcal{M}'$ and $F(1_A) = 1_{F(A)}$ as well as $F(g < f >) = F(g) < F(f) >$. If both $\mathcal{M}$ and $\mathcal{M}'$ are biclosed multicategories, F is said to be a *biclosed*

multifunctor if it preserves the biclosed structure on the nose, thus

$$F(A \o B) = F(A) \o F(B),$$
$$F(e_{AB}) = e_{F(A)F(B)},$$
$$F(f^*) = F(f)^*, \qquad \text{etc.}$$

12. SOME FINAL REMARKS

An imminent deadline won't allow me to carry out everything I would have liked to do here, for example, to re–examine cut elimination and coherence from the point of view of the internal language, the latter along the lines pioneered by Minc. Instead, I shall consider some of the points raised in the discussion after my talk at the Boulder conference and make some observations about the recent literature related to multicategories, although I cannot claim to have completely digested all of this material.

More general than contextfree recognition grammars are *production grammars*, also called "rewriting systems" or "semi–Thue systems", which deal with arrows (productions)

$$A_1 \dots A_m \to B_1 \dots B_n.$$

In fact, my own recent work concerning natural languages makes use of production grammars rather than categorial grammars.

If a suitable equivalence relation is introduced between derivations of a production grammar, one obtains a strictly monoidal category, which may also be described as a 2–category whose underlying 1–category is a monoid, in fact a free monoid. Something like this was done by Hotz (1966) and pursued by Benson in several papers. In particular, Benson (1970) pointed out that an interpretation should be viewed as a functor from a category called "syntax" to a category called "semantics", where the former was a strict monoidal category, thus anticipating the ideas of Section 11 (and those in Section 4 of my paper in Oehrle et al, 1988).

Inasmuch as contextfree recognition grammars are special kinds of production grammars, one should be able to view multicategories as special kinds of strict monoidal categories. This was in fact pointed out by Linton and Reynolds, in developing ideas which Linton (1971) had first presented in his customary inimitable style. The trick was to use sequences of arrows $f_i : \Gamma_i \to B_i \quad (i = 1, \dots, n)$ of the multicategory to construct arrows $(f_1, \dots, f_k) : \Gamma_1 \dots \Gamma_n \to B_1 \dots B_n$ of the strict monoidal category, which Linton calls a "multilinear" category.

Incidentally, the productions $A_1 \dots A_m \to B_1 \dots B_n$ should not be confused with the sequents used by Gentzen in his treatment of classical logic, which happen to look the same. In Gentzen's sequents, juxtaposition on the left denotes conjunction, while juxtaposition on the right denotes disjunction. Gentzen's classical sequents have also been lifted to the level of categories in Szabo's "polycategories" (1975). They reappear in Girard's linear logic, where they may be interpreted as arrows

$$A_1 \otimes \dots \otimes A_m \to (B_1^* \otimes \dots \otimes B_n^*)^*,$$

the star representing some kind of involution.

The first time I came across the use of lambda calculus in connection with text processing was in a preprint by Masterman et al (1956). Van Benthem (1983,1988) studied a version of the syntactic calculus admitting the structural rule of interchange and considered the restricted form of the lambda calculus that goes with it. Probably his system is the same as that studied by Minc (1977), who established a cut elimination

theorem for his system and used the restricted form of the lambda calculus to solve the coherence problem for symmetric monoidal closed categories (see also MacLane 1982). The same calculus appears in the linear logic of Girard (1987), among many other interesting ingredients, and a cut elimination theorem may also be found in Girard and Lafont (1987).

A completeness theorem for the syntactic calculus was proved by Buszkowski (1986): a sequent $A_1 \dots A_n \to A_{n+1}$ is provable if and only if it is valid under every interpretation of the A_i as downward closed subsets of a partially ordered semigroup, much like the ideals of a ring in Section 1. A similar result for linear logic was obtained by Girard (1987).

Bibliography

Ajdukiewicz, K.: Die syntaktische Konnexität, Studia Philosophica I (1937), 1-27; translated in S. McCall, Polish Logic 1920 - 1939, Clarendon Press, Oxford 1967.

Anderson, A.R., and N.D.Belnap: Entailment: the logic of relevance and necessity, Princeton University Press, Princeton, N.J., 1975.

Bar–Hillel, Y.: A quasiarithmetical notation for syntactic description, Language 29 (1953), 47-58.

Barr, M.: The theory of theories as a model of syntactic acquisition, Theoretical Linguistics 5 (1978), 261-274.

Bénabou, J.: Catégories avec multiplications, C.R. Acad. Sci. Paris 256 (1963), 1887-1890.

Bénabou, J.: Introduction to bicategories, Report on the Midwest Category Seminar, Springer Lecture Notes in Mathematics 47 (1967), 1-77.

Bénabou, J.: Structures algébriques dans les catégories, Cahiers Topologie Géom. Differentielle 10 (1968), 1-126.

Benson, D.B.: Syntax and semantics: a categorical view, Information and Control 17 (1970), 145-160.

Benson, D.B.: The basic algebraic structures in categories of derivations, Information and Control 28 (1975), 1-29.

Birkhoff, G. and J.D. Lipson: Heterogeneous algebras, J. Combinatorial Theory 8 (1970), 115-133.

Bourbaki, N.: Algèbre multilinéaire, Hermann, Paris, 1948.

Buszkowski, W.: Completeness results for Lambek syntactic calculus, Zeitschr. f. math. Logik und Grundlagen d. Math. 32 (1986), 13-28.

Cartan, H. and S. Eilenberg: Homological Algebra, Princeton University Press, Princeton, N.J., 1956.

Church, A.: The calculi of lambda–conversion, Annals of Mathematics Studies 6, Princeton University Press, Princeton, N.J., 1941.

Curry, H.B. and R. Feys: Combinatory Logic I, North Holland, Amsterdam, 1958.

Eilenberg, S. and G.M. Kelly: Closed categories, in: Proc. Conf. Categorical Algebra La Jolla 1965, Springer Verlag, Berlin, 421- 562.

Findlay, G.D. and J. Lambek: Calculus of bimodules, McGill University, manuscript 1955.

Girard, J.Y.: Linear logic, Theoretical Computer Science 50 (1987), 1-102.

Girard, J.Y. and Y. Lafont: Linear logic and lazy computation, preprint 1987.

Hotz, G.: Eindeutigkeit und Mehrdeutigkeit formaler Sprachen, Elektronische Informationsverarbeitung und Kybernetik 2 (1966), 235-247.

Jay, C.B.: Languages for monoidal categories, Dalhousie University, preprint 1987.

Kleene, S.C.: Introduction to metamathematics, Van Nostrand, New York and Toronto, 1952.

Lambek, J.: The mathematics of sentence structure, Amer. Math. Monthly 65 (1958), 154-69.

Lambek, J.: On the calculus of syntactic types, Amer. Math. Soc. Proc. Symposia Appl. Math. 12 (1961), 166-78.

Lambek, J.: Lectures on rings and modules, Ginn and Co. Waltham Mass., 1966; Chelsea Publ. Co., New York, 1986.

Lambek, J.: Deductive systems and categories I, J. Math. Systems Theory 2 (1968), 278-318.

Lambek, J.: Deductive systems and categories II, Springer LNM 86 (1969), 76-122.

Lambek, J.: Categorial and categorical grammars, in: Oehrle et al, 1988.

Lambek, J. and P.J. Scott: Introduction to higher order categorical logic, Cambridge studies in advanced mathematics 7 (1986).

Lawvere, F.W.: Functional semantics of algebraic theories, Proc. Nat. Acad. Sci. U.S.A. 50 (1963), 869-872.

Linton, F.E.J.: The multilinear Yoneda lemmas, Lecture Notes in Mathematics 195 (1971), Springer Verlag.

Linton, F.E.J. and G. Reynolds: Pro–U–categories, undated manuscript.

MacLane, S.: Natural associativity and commutativity, Rice University Studies 49 (1963), 28-46.

MacLane, S.: Categories for the working mathematician, Graduate Texts in Mathematics 5 (1971), Springer Verlag.

MacLane, S.: Why commutative diagrams coincide with equivalent proofs, Contemporary Mathematics 13 (1982), 387-401.

Mann, C.R.: The connection between equivalence of proofs and cartesian closed categories, Proc. London Math. Soc. 31 (1975), 289-310.

Masterman, M., A.F. Parker Rhodes and M.T. Hoskyns: An account of the pilot project of the Cambridge Language Research Unit, Progress Report II, Annex V (1956), 1-58.

Minc, G.E.: Theory of proofs (arithmetic and analysis), translated from Itogi Nauki i Tekhniki (Algebra, topologiya, geometriya), 13 (1975), 5-49.

Minc, G.E.: Closed categories and the theory of proofs, translated from Zapiski Nauchnych Seminarov Leningradskogo Otdeleniya Mat. Instituta im. V.A. Stuklova AN SSSR 68 (1977), 83-114.

Montague, R.: Formal philosophy, selected papers of Richard Montague, edited by R.H. Thomason, Yale University Press, New Haven, 1974.

Oehrle, R.T., E. Bach and D. Wheeler (editors): Categorial grammars and natural language structures, Reidel, Dordrecht, 1988.

Szabo, M.E.(ed.): The collected papers of Gerhard Gentzen, Studies in Logic and the Foundations of Mathematics, North Holland, Amsterdam 1969.

Szabo, M.E.: A categorical equivalence of proofs, Notre Dame J. Formal Logic 15 (1974), 177-91.

Szabo, M.E.: Polycategories, Communications in Algebra 3 (1975), 663-689.

Szabo, M.E.: Algebra of Proofs, Studies in Logic and the Foundations of Mathematics 88, North Holland, Amsterdam, 1978.

Van Benthem, J.: The semantics of variety in categorial grammar, Simon Fraser University, Report 83-26, 1983.

Van Benthem, J.: The Lambek calculus, in: Oehrle et al. 1988.

Mathematics Department
McGill University
Montreal, Quebec H3A 2K6