Formalization of Classical Mathematics in Automath

J. Zucker[1] [2]

0. INTRODUCTION

Automath is a formal language, invented by N.G. de Bruijn, suitable for the formalization of (large parts of) mathematics, such that the correctness of any proof of a theorem written in Automath can be checked mechanically (by a computer).

A description of the Automath project (headed by Prof. de Bruijn), and its motivation, are given in [de Bruijn 73c]. Further information on the language and the project is given in [*van Daalen 73 (A.3)*], [*Zandleven 73 (E.1)*] and [van Benthem Jutting 73].

The present paper deals with an aspect of the project with which the author was involved, namely the formalization, in Automath, of a certain part of mathematics, viz. calculus or classical real analysis. This (together with the formalization of some other mathematical topics) will constitute an "Automath book".

Only a brief outline of this work is possible here; a more detailed account is planned.

1. GENERAL DESCRIPTION OF THE LANGUAGE

1.1. We will give a sketch of a version of Automath called AUT-II. (A detailed description of another version, AUT-QE, essentially a sub-language of AUT-II, is given in [*van Daalen 73 (A.3)*]. We will first give a general description of the language, in accordance with [*van Daalen 73 (A.3)*], and then, in Section 2, consider those aspects of AUT-II which differ from AUT-QE.)

[1]The author was previously employed in the Automath project at the University of Technology, Eindhoven, Netherlands, and supported by the Netherlands Organization for the Advancement of Pure Science (Z.W.O.).

[2]This article is based on two talks on Automath given at the Colloque International de Logique, Clermont-Ferrand, France, July 1975. (The first talk was by N.G. de Bruijn, adapted from his [de Bruijn 73c]). Thanks are due to my colleagues in the project, and to Roel de Vrijer, for helpful comments on earlier versions.

The language is based on the typed λ-calculus, incorporating the well-known correspondence between types and propositions, discovered by Curry, Howard, Martin-Löf and Tait (see e.g. [Martin-Löf 75a]), and independently by de Bruijn ([*de Bruijn 70a (A.2)*], [Scott 70]).

1.2. Notation

Throughout the paper, the symbols "E" and "F", possibly with subscripts, will denote arbitrary expressions of the language.

The colon is used for the typing relation; thus "$E : F$" may mean that "E is an *object* of *type* F" or "E is a *proof* of the proposition F"; more generally, to cover both cases, we say that "E has *category* F".

1.3. Automath lines

An Automath book consists of a sequence of *lines*. A line may be one of three kinds:

(1) **Context line.** A context is a string of *parameters* (or free variables)

$$a_1 : F_1 , a_2 : F_2 , \ldots , a_n : F_n$$

where each parameter a_i has category F_i (so that a_i may be a variable of type F_i, or an assumption of the proposition F_i). Each expression F_i may contain the a_j for $j < i$ (only).

This serves as the context of a mathematical argument.

A context line, then, introduces such a context, or lengthens one by adding new parameters.

The empty context will be denoted by "φ".

(2) **Definition line.** In a given context, then, we may introduce a constant by definition:

$$a_1 : F_1 , \ldots , a_n : F_n \vdash C \ := \ E : F \ .$$

Here C is the new constant, defined (in the context shown) to be equal to the expression E, with category F, where the expressions E and F may contain the parameters $a_1, \ldots, a_n$.

(For convenience, we display the full context on the left of the definition line, separated from the defined constant by "$\vdash$".)

(3) **Primitive notion line.** This is similar to the definition line, except that the constant C is introduced, outright, as a "primitive notion" of some category:

$$a_1 : F_1 , \ldots , a_n : F_n \vdash C \ := \ \text{prim} : F \ .$$

So (in the context shown) C may be a primitive constant of type F, or a primitive proof of the proposition F (i.e. an axiom).

1.4. Substitution

In a definition line or primitive notion line the constant C depends (of course) on the parameters $a_1, \ldots, a_n$ (cf. 1.3). We may exhibit this dependence by writing C as $C(a_1, \ldots, a_n)$.

Let $t_1, \ldots, t_n$ be a sequence of terms, and (with F and F_i as in 1.3) let F^* denote the result of simultaneously substituting $t_1, \ldots, t_n$ for $a_1, \ldots, a_n$ in F, and F_i^* that of simultaneously substituting $t_1, \ldots, t_{i-1}$ for $a_1, \ldots, a_{i-1}$ in F_i, for $1 \leq i \leq n$.

Now suppose further that the sequence $t_1, \ldots, t_n$ *instantiates* the given context, i.e., t_i has category F_i^* for $1 \leq i \leq n$. Then we may substitute these terms for the parameters in $C(a_1, \ldots, a_n)$, to form the expression $C(t_1, \ldots, t_n)$, with category F^*.

1.5. Abstraction and application

Complex expressions may be formed from simple expressions (free variables and constants) by (*inter alia*) the operations of abstraction and application.

If a is a parameter of category E, and F an expression, then we may "abstract over E" to form the *abstraction expression* $[x : E]\, F^*$ (where F^* results from replacing a by the bound variable x in F). This corresponds, in more usual notation, to "$(\lambda x \in E)F^*$".

Next, if E and F are two expressions (satisfying certain assumptions of type coherence) then we can form the *application expression* $\langle F \rangle\, E$. This is the "application of E to F". (Notice that the "argument" F is written *before* the function E. This unconventional order has certain technical advantages.)

1.6. An example

Suppose that "Na" has been introduced as the type of the natural numbers, and we want to define the successor as a primitive constant. This is done by a primitive notion line, in the context of an arbitraty natural number:

$$a : \text{Na} \vdash \text{Sc} \ := \ \text{Prim} : \text{Na} \ .$$

Then "Sc" (or "Sc(a)") represents the successor of the (arbitrary) natural number a, with type again Na.

If we now want the successor function (of type Na → Na), we must abstract over Na; so we define (now in the empty context):

$$\emptyset \vdash \mathrm{ScFn} \;:=\; [x : \mathrm{Na}]\,\mathrm{Sc}(x) : \mathrm{Na} \to \mathrm{Na}\ .$$

Suppose next that t is a term of type Na. Then we can apply this successor function to t, to form the expression "$\langle t\rangle$ ScFn", which is definitionally equal to "Sc(t)" (by β-reduction).

2. SPECIAL FEATURES OF AUT-Π

2.1. It must be remembered that our aim is to formalize (a part of) *classical* mathematics. The following question then arises: is an Automath-like language suitable for this? – since such a language would seem to be appropriate rather for formalizing something like the work in [Bishop 67].

The answer, as it turns out, is "Yes". (We return to this point in Section 3).

A consequence of the requirements to formalize classical mathematics is a loss of the perfect symmetry between propositions and types found, for example, in [Martin-Löf 75a].

In fact, we have two kinds of expressions:

(i) t-expressions ("t" for "type" or "term"),

(ii) p-expressions ("p" for "proposition" or "proof").

These two kinds of expressions will not always be handled in the same way, as we will see.

If $E : F$, then E and F are either both t-expressions or both p-expressions.

Further, to every expression E we attach a *degree* deg(E):1, 2 or 3. Expressions of degree 1 are (roughly) "large categories" (cf. the type "V" of [Martin-Löf 75a], and if $E : F$ then $\deg(E) = \deg(F) + 1$.

We give some details in 2.2 and 2.3. (The description will be informal and incomplete, for the sake of brevity and simplicity).

2.2. Formation of t-expressions

These may be formed in the following ways.

(t-1) We begin with a constant "τ" of degree 1, the "category of all types".

(t-2) Suppose $\alpha : \tau$ (so $\deg(\alpha) = 2$). Then α is a *type*. (Below, "α" will refer to this α).

(t-3) Suppose $a : \alpha$ (so $\deg(a) = 3$). Then a is an *object* of type α.

What other t-expressions of degree 1 are there?

(t-4) $\alpha \to \tau$ has degree 1. It is the category of functions from α to τ, or of "type-valued functions with domain α".

(t-5) Suppose $f : \alpha \to \tau$ (so $\deg(f) = 2$). Then f is a *type-valued function* with domain α. Now we can form a *type* from f in three ways:

(i) **Application.** If $a : \alpha$ then $\langle a \rangle f : \tau$.

(ii) **Cartesian product.** $\Pi(f) : \tau$. (Hence the name "AUT-Π"). This is the type of all functions g with domain α such that for all $a : \alpha$, $\langle a \rangle g : \langle a \rangle f$.

(iii) **Type of pairs.** $\Sigma(f) : \tau$. This is the type of all pairs $\langle a, b \rangle$ such that $a : \alpha$ and $b : \langle a \rangle f$.

Note. If f has the form $[x : \alpha]\beta$, where the type β does not contain the bound variable x, then we can write (in more usual notation)

$$\alpha \to \beta \qquad \text{for } \Pi(f)$$

and

$$\alpha \otimes \beta \qquad \text{for } \Sigma(f) \ .$$

2.3. Formation of p-expressions

We will try, as far as possible, to make the description here parallel to that in 2.2 for t-expressions.

(p-1) We begin, again, with a constant "π" of degree 1, the "category of all propositions".

(p-2) Suppose $A : \pi$ (so $\deg(A) = 2$). Then A is a *proposition*. (Below, "A" will refer to this A).

(p-3) Suppose $p : A$ (so $\deg(p) = 3$). Then p is an *proof* of A.

What other p-expressions of degree 1 are there?

(p-4a) $A \to \pi$ has degree 1. It is the category of functions from (proofs of) A to propositions.

(p-5a) Suppose $f : A \to \pi$ (so $\deg(f) = 2$). We can form a *proposition* from f in three ways:

(i) **Application.** If $p : A$ then $\langle p \rangle f : \pi$.

(ii) **Generalized implication.** $\Pi(f) : \pi$. This is the proposition proved by those q such that for all $p : A$, $\langle p \rangle q : \langle p \rangle f$.

(iii) **Generalized conjunction.** $\Sigma(f) : \pi$. This is the proposition proved by those pairs $\langle p, q\rangle$ such that $p : A$ and $q : \langle p\rangle f$.

Note. If f has the form $[x : A]B$, where the proposition B does not contain the bound variable x, then we can write (in more usual notation)

$A \to B$ for $\Pi(f)$ (ordinary implication)

and

$A \wedge B$ for $\Sigma(f)$ (ordinary conjunction) .

As further p-expressions we have (with α as in (t-2)):

(p-4b) $\alpha \to \pi$ has degree 1. It is the category of proposition-valued functions with domain α, i.e. *predicates on* α.

(p-5b) Suppose $P : \alpha \to \pi$ (so $\deg(P) = 2$). Then P is a *predicate on* α. Now we can form a *proposition* from P in two ways:

(i) If $a : \alpha$ then $\langle a\rangle P : \pi$. (Note that the convention for the order in an application expression works well here! We can read "$\langle a\rangle P$" as "a has property P" or "a satisfies P" or "a is P".)

(ii) **Universal quantification.** $\Pi(P) : \pi$.

This is the proposition proved by those p such that for all $a : \alpha$, $\langle a\rangle p : \langle a\rangle P$. We also write "$\forall(P)$" for "$\Pi(P)$".

Now what about "(iii) **Existential quantification** $\Sigma(P) : \pi$"? We return to this point below, in Section 4.

2.4. Notes

(1) The above description is incomplete (as stated previously). For example, there is also a *disjoint sum* of types $\alpha \oplus \beta$ (for all $\alpha : \tau$ and $\beta : \tau$). Further, we have (for all $\alpha : \tau$) a type $\alpha \to (\alpha \to \pi)$ of *binary predicates* or *relations* on α, etc.

(2) There is abstraction over categories of degee 2 only. Thus, for an expression of the form $[x : E]F$ or $E \to F$, E must have degree 2, while F may have any degree (except 3 in the case of $E \to F$); and the degree of the expression is then the same as $\deg(F)$.

3. IRRELEVANCE OF PROOFS

It should be clear from the above description that the language incorporates *minimal logic* in $\to$, $\wedge$ and $\forall$.

Further development is necessary to make the system suitable for the formalization of *classical* mathematics. This is achieved in three steps.

(1) We introduce $\bot$ as a primitive constant, and the *intuitionistic* $\bot$ *rule* as a primitive axiom. This gives intuitionistic logic in $\rightarrow$, $\wedge$ and $\forall$.

Now we define $\neg$ from $\rightarrow$ and $\bot$, and also $\vee$ and $\exists$ from $\wedge$, $\forall$ and $\neg$, in the well-known way.

(2) We introduce the *classical* $\neg\neg$ *rule* as a primitive axiom.

However, we still do not have classical logic!

The problem is this. Consider e.g. the logarithmic function, defined on the positive reals:

$$a : \mathrm{Rl}\ , \qquad p : a > 0 \vdash \log\ :=\ \ldots : \mathrm{Rl}$$

(where "Rl" is the type of reals, and "..." is an expression defining the log in the context of the two parameters a and p. So the log is essentially a function of two arguments, namely a real number, and a proof that it is positive.

This is a familiar situation in constructive mathematics, but how does it work in classical mathematics? Suppose, for example, we have two proofs p and q that $3 > 0$. We want to say *immediately* that

$$\log(3, p) = \log(3, q)\ .$$

(The same problem arises with the reciprocal function, etc.)

We therefore adopt the principle of

(3) **Irrelevance of proofs**. This principle (due to de Bruijn) can be stated in the form: any two proofs of the same proposition (or of definitionally equal propositions) are taken as *definitionally equal*.

(It follows, to continue the above example, that "$\log(3, p)$" is definitionally equal to "$\log(3, q)$".)

The above three steps, then, form our basis for formalizing classical mathematics. And, as it turns out, this approach permits quite a natural development of classical real analysis; and (I think) this is because it reflects, to some extent, how mathematicians actually reason.

(Even "generalized implication" and "generalized conjunction" (cf. (p-5a) in 2.3), which are not usually considered as part of "classical logic", turn up naturally in reasoning in classical mathematics.)

In fact, one of the most interesting aspects of the project (to the author) is that it demonstrates the feasibility of founding a large part of everyday mathematics on the typed λ-calculus, rather than on axiomatic set theory.

4. STRONG AND WEAK EXISTENCE, SUBTYPES

We return to the problem left open in Section 2, (p-5b): what about "strong existential quantification" of predicates? – i.e., to form, from a predicate P on α, the proposition $\Sigma(P)$, proved by those pairs $\langle a, p\rangle$ such that $a : \alpha$ and $p : \langle a\rangle P$ (by analogy with (iii) of (t-5) or (p-5a); cf. [Scott 73] or [Martin-Löf 75a]).

The problem here is that "strong existence" is *inconsistent* with the principle of irrelevance of proofs! For consider, e.g., the predicate on the naturals of "being less than 2"; i.e., define the predicate

$$\emptyset \vdash P \;:=\; [x : \mathrm{Na}]\,(x < 2) : \mathrm{Na} \to \pi$$

and let p_0 and p_1 be proofs that 0 and 1, respectively, satisfy P.

Then both $\langle 0, p_0\rangle$ and $\langle 1, p_1\rangle$ have category $\Sigma(P)$. If this is taken as a proposition ("there exists P") then they are proofs; so by the principle of irrelevance of proofs, they are definitionally equal, and hence (by taking left projections) so are 0 and 1.

In fact, we do allow the category $\Sigma(P)$, but as a *type*, not a proposition (i.e., its category is τ, and it is a t-expression). It is the "subtype of α on which P holds". Now there is no problem, since (to return to the above example) $\langle 0, p_0\rangle$ and $\langle 1, p_1\rangle$ are simply two distinct objects of this subtype.

Hence we stick to the negative fragment of predicate logic, and define $\exists$ from $\forall$ and $\neg$ ("weak existence"). So also, for the sake of uniformity, we define $\vee$ from $\wedge$ and $\neg$ ("weak disjunction") (although we do have a disjoint sum of types $\alpha \oplus \beta$, and strong disjunction would probably be harmless).

5. DISTINCTION BETWEEN t- AND p-EXPRESSIONS

We repeat that an important feature of AUT-II is the distinction between t- and p-expressions (made possible by the existence of two "large categories", τ and π) and differences in their treatment. We summarize some of these differences:

(1) The double negation law can be applied to propositions, but not to types.

(2) The principle of irrelevance of proofs applies (only) to proofs of a given proposition, not to objects of a given type. (Clearly, we would not want a principle of "irrelevance of objects"!)

(3) Although we have universal quantification of predicates corresponding to Cartesian products of type-valued functions (compare ii) of (p-5b) and of (t-5)) we do not have strong existential quantification corresponding to the type of pairs, as explained in Section 4.

6. PREDICATES AND QUANTIFIERS

We have, for each type α, a category of predicates on α (as indicated in Section 2),

$$\alpha : \tau \ \vdash \text{Pred} := \alpha \to \pi$$

and also (binary) relations on α

$$\alpha : \tau \ \vdash \text{Reln} := \alpha \to (\alpha \to \pi)$$

(also of degree 1), and so on.

Predicates play an important part in our work. Not only are they formed by abstracting from propositions, but they also occur as predicate variables or parameters, so that we can work in the context of an *arbitrary predicate* on α:

$$\alpha : \tau \,,\ P : \text{Pred}(\alpha) \vdash \ldots \,.$$

This is useful, for example, in permitting a general treatment of quantifiers. For we can view a quantifier on α as an operation from predicates on α to propositions (as was done for the universal quantifier in [Church 40]).

Thus, e.g., the existential quantifier is defined by:

$$\alpha : \tau \,,\ P : \text{Pred}(\alpha) \vdash \exists := \neg\forall\,[x : \alpha]\,(\neg\,\langle x\rangle\, P) : \pi \,.$$

Other examples of quantifiers are the following. We can define, for predicates P on the naturals, the propositions "P holds eventually" and "P holds infinitely often":

$$P : \text{Pred(Na)} \vdash \text{Evt} := \ldots : \pi$$
$$\text{"} \qquad \vdash \text{I0} \ := \ldots : \pi \,.$$

In fact, there are many situations, notably in the theory of convergence of series, where the "Evt" quantifier is more important than the universal quantifier.

7. SOME FURTHER FEATURES OF THE LANGUAGE

7.1. Equality

There is, for each type α, an equality relation on α, introduced as a primitive proposition:

$$\alpha : \tau \,,\ a : \alpha \,,\ b : \alpha \vdash \text{Eq} := \text{prim} : \pi$$

("a equals b"). So equality can only be expressed between objects of a given type, but not between types, or between any p-expressions. (But in any case, all proofs of the same proposition are definitionally equal.)

We also introduce primitive axioms for equality, namely *reflexivity, substitutivity* (for an arbitrary predicate on α), and *extensionality* for functions. (The last axiom is probably not essential for our results in elementary real analysis, but simplifies the work.)

7.2. Sets

Now we cannot express equality between predicates (as stated above), only logical equivalence. Nor can we quantify over all predicates of a given type α, since Pred(α) has degree 1.

Therefore we introduce, for each type α, its "power type":

$$\alpha : \tau \vdash \text{Powertype} \;:=\; \text{prim} : \tau$$

i.e., the type of all sets of objects of type α; and also, the membership relation as a new primitive proposition:

$$\alpha : \tau\,,\; a : \alpha\,,\; b : \text{Powertype}(\alpha) \vdash \text{El} \;:=\; \text{prim} : \pi$$

("a is an element of b"); and primitive axioms, including *set extensionality* and *comprehension* (for an arbitrary predicate on α), by means of which we can go back and forth between predicates on α and their extensions in Powertype(α).

Now "Powertype(α)" is a t-expression of degree 2, so we *can* express equality between sets of a given type, and also quantify over all such sets.

7.3. Definition of individuals

Given a predicate P on a type α, and the assumptions that an object satisfying P exists and is unique, we may introduce, as primitives, (a name for) an individual object of type α, and the axiom that this individual satisfies P, thus:

$$\alpha\colon \tau\,,\; P : \text{Pred}(\alpha)\,,\; p : \exists(P)\,,\; q : \text{Unq}(P) \vdash \text{Indiv} \;:=\; \text{prim} : \alpha$$

$$\text{"} \quad \text{"} \quad \text{"} \quad \text{"} \quad \vdash \text{AxIndiv} := \text{prim} : \langle \text{Indiv} \rangle\, P$$

("Unq" is the uniqueness quantifier).

So this functions rather like an "ι-symbol" (cf. e.g. [Church 40]).

8. DEVELOPMENT OF CLASSICAL REAL ANALYSIS

8.1. We give, below, an outline of the mathematical development in the book.

We are guided by the nature of Automath, and so a type-theoretical rather than set-theoretical basis is used. So we start with a few basic types, including "Na" for the naturals and "Rl" for the reals, and build up other types from these. For example, Na $\rightarrow$ Na is the type of number-theoretical functions of one argument; Na $\rightarrow$ Rl the type of sequences of reals; Rl $\rightarrow$ Rl the type of (total) real-valued functions of a real variable, and so on.

8.2. We start, then, with the *naturals*,

$$\emptyset \vdash \mathrm{Na} \ := \ \mathrm{prim} : \tau \ ,$$

and introduce primitive constants for 0 and successor, as well as Peano's axioms for these. There is also a primitive constant for recursion on every type α, to define functions of type Na $\rightarrow \alpha$, together with the corresponding axioms.

Next, we introduce (as primitive constants) the operations and relations fro the ordered fields of *reals*,

$$(\mathrm{Rl}, 0, 1, +, \times, <) \ ,$$

together with the axioms for this structure, including order-completeness. (More accurately, we start with the extended reals as an ordered structure, and cut down to the reals as a subtype).

Then we define, by recursion, the embedding of the naturals in the reals, and hence construct the *integers* and *rationals* as predicates on (or subtypes of) the reals.

Next, we construct the type of *partial functions* of a real variable as follows. We take a primitive type "ω-type" containing exactly one point "ω", and form the disjoint sum type

$$\emptyset \vdash \mathrm{R}\omega \ := \ \mathrm{Rl} \oplus \omega\text{-type} : \tau \ .$$

Then the type of partial functions is defined by:

$$\emptyset \vdash \mathrm{Pfn} \ := \ \mathrm{Rl} \rightarrow \mathrm{R}\omega : \tau \ .$$

The idea is that the partial function is "defined" at exactly those points where its value is not ω.

Now we can define, in the context of an arbitrary partial function and arbitrary real point, the predicate of "being a derivative" (of the function, at the point), and prove that the derivative (if it exists) is unique. Hence (using "Indiv" of 7.3) we can define the "derived function" of any partial function (as another partial function).

We can then compute the derivatives of, e.g., the polynomial and reciprocal functions.

Next, we develop the theory of power series, define the exponential function as the sum of such a series, and compute its derivative. Finally, by an application of (a version of) the inverse mapping theorem, we compute the derivative of the logarithmic function.

9. FURTHER MATHEMATICAL TOPICS

A. Kornaat, a colleague in the project, has written some sections for the book on other topics:

(1) *Set theory*, in which the equivalence is shown between various formulations of the axiom of choise (such as Zorn's lemma).

(2) *Combinatorics*, including a proof of the Hall-König theorem [Hall 67, Th. 5.1.5].

(3) *Metric spaces*, inclusing compactness and connectedness; hence the Heine-Borel and Bolzano-Weierstrass theorems for the reals.

Our work is interrelated; e.g., the work in 3) above depends on the theory of the real number field (Section 8), and, conversely, the proof of the inverse mapping theorem (mentioned in Section 8) uses the result, proved in 3) above, that the continuous image of a connected subset of a metric space is connected.

10. ABSTRACT STRUCTURES

10.1. An example

In the course of our work we come across the following problem. We have to deal with a linear order on the naturals, and one on the reals, and we do not want to have to prove the same theorems (on linear orders) twice for these separate structures.

Therefore we first consider an "abstract linear order", as follows. We define, in the context of an arbitrary type α and relation ρ on α:

$$\alpha : \tau\,,\ \rho : \mathrm{Reln}(\alpha) \vdash \mathrm{AxL0} \ := \ \ldots : \pi$$

where "..." is the conjunction of the axioms for a linear order. So "$\mathrm{AxL0}(\alpha, \rho)$" says that ρ is a linear order on α.

Next we take the context

$$\alpha : \tau\,,\ \rho : \mathrm{Reln}(\alpha)\,,\ p : \mathrm{AxL0}(\alpha, \rho)$$

(so p is the assumption that ρ is a linear order on α), and, in this context, prove general theorems about linear orders, which we can then apply to our two examples.

For example, we can prove that the inverse relation to ρ is again a linear order on α. Hence, from any theorem that we may have about increasing sequences on the reals, or about upper bounds (say), we can immediately infer the *dual* theorem for decreasing sequences, or lower bounds (respectively).

10.2. Telescopes

Now a general framework in which to view linear orders, or other algebraic structures, has been proposed by de Bruijn. It uses the notion of "telescope".

A telescope, rougly speaking, acts as a "category" for certain strings of expressions (or "structures"). More precisely: consider a context

$$a_1 : F_1 , a_2 : F_2 , \ldots , a_n : F_n .$$

We can associate with this context a telescope, i.e., the expression

$$[a_1 : F_1] [a_2 : F_2] \ldots [a_n : F_n]$$

which consists of n "modules", as shown. Now certain strings of length n will "fit into" or "belong to" to this telescope, namely those strings which *instantiate* the given context (cf. 1.4).

A telescope therefore functions like a "generalized Σ".

Now to apply this to the situation in 10.1, we define a specific 3-telescope (i.e. telescope of length 3), the "category of linear orders":

$$\mathrm{CatL0} \ := \ [\alpha : \tau] [\rho : \mathrm{Reln}(\alpha)] [p : \mathrm{AxL0}(\alpha, \rho)] .$$

Those strings of length 3 which fit into "CatL0" are called *linear orders*. Three such structures were mentioned in 10.1: the "abstract linear orders" (α, ρ, p), and the linear orders on the naturals and on the reals.

Next, we can define a proposition "$\mathrm{AxComp}(\alpha, \rho)$" which says that (α, ρ) satisfies the axiom of order-completeness, and then add another module on to "CatL0" to form a 4-telescope, the "category of complete linear orders":

$$\mathrm{CatCL0} \ := \ [\alpha : \tau] [\rho : \mathrm{Reln}(\alpha)] [p : \mathrm{AxL0}(\alpha, \rho)] [q : \mathrm{AxComp}(\alpha, \rho)] .$$

So modules can act like adjectives (e.g. "complete"). (This analogy works best in a language like French, where the adjective comes after the noun.)

Another example of a telescope used in the book is the "category of fields" (a 6-telescope). Again, further modules can be added on to this, to form, in turn, the "category of ordered fields", and of "complete ordered fields".

The formalism of telescopes may prove especially useful in an Automath treatment of abstract algebra.