Formalization of constructivity in Automath

Formalization of Constructivity in Automath

N.G. de Bruijn

1. INTRODUCTION

There are various systems in which a large part of mathematical activity is formalized. The general effect of the activity of putting mathematics into such a system is what one might call the unification of mathematics: different parts of mathematics which used to be cultivated separately get united, and methods available in one part get an influence in other parts.

Very typical for twentieth century mathematics is the unifying force of the concepts of set theory. And today one might say that the language of mathematics is the one of the theory of sets combined with predicate logic, even though one might disagree about the exact foundation one should give to these two.

Not everyone thinks of set theory and logic as being parts of a single formal system. Set theory deals with objects, and logic deals with proofs, and these two are usually considered as of a different nature. Nevertheless, there are possibilities to treat these two different things in a common system in a way that handles analogous situations analogously indeed.

A system that goes very far in treating objects and proofs alike, is the Automath system (see [*de Bruijn 80 (A.5)*]). In Automath there are expressions on three different levels, called degrees. Each expression of degree 3 has a "type" that is of degree 2, and each expression of degree 2 has a type of degree 1. Expressions of degree 1 do not have a type. There are two basic expressions of degree 1, viz. `type` and `prop`. The word `type` should not be confused with the word type used more or less colloquially when saying that each expression of degree 2 or 3 has a type.

We denote typing by a colon. If A has B as its type, we write $A : B$. So we can have

$$A : B : \texttt{type} \tag{1}$$

and also

$$C : D : \texttt{prop} \,. \tag{2}$$

The interpretation of (1) is that A is the name of an object (like the number 3), and that B is the name of the class from which that object is taken (it might be

a symbol for the set of integers). The interpretation of (2) is that C is a name for a proof, and that D somehow represents the statement that is proved by C.

The main profit we have from this way of describing proofs and objects is the matter of substitutivity. If we have described an object depending on a number of parameters, that description can be used under different circumstances by means of substitution: we replace the formal parameters by explicit expressions. The same technique is applicable to theorems: a theorem is intended for many applications, and such applications can be effectuated by substitution. The conditions of the theorem are modified by these substitutions too. If we study the matter more closely, we see that some of the parameters are object-like, and othere are proof-like. The substitution machinery is the same for both. All this is effectively implemented in the Automath system.

2. ORIENTATION ON GEOMETRICAL CONSTRUCTIONS

On the fringe of mathematics there are mathematical activities which seem to be of a kind that does not fit into the pattern of objects and proofs. One such thing is the matter of geometrical constructions, a subject that goes back to Greek mathematics. A construction is neither an object nor a proof, but constructions are discussed along with geometrical objects, and along with proofs that show that the constructions construct indeed what is claimed to be constructed.

Since these geometrical constructions can also admit substitution for formal parameters, there is a case for creating facilities which handle a new kind of things along with objects and proofs. So we can think of a system that handles objects, proofs and geometrical constructions in more or less the same way.

If we think of geometrical constructions, there is a peculiarity that may not arise easily with other kinds of constructions: it is the matter of observability. Let us study a particular example in order to stress this point. Let there be given four points A, B, C and D in the plane. We assume that A, B and C are not on a line. Let M be the centre of the circle through A, B and C. We wish to construct the point P that is defined as follows. P is obtained from D by multiplication, with M as the multiplication center, and multiplication factor 1, 2 or 3. The factor is 1 if D lies inside the circle, 2 if D lies on the circle, and 3 if D lies outside the circle. If we want to carry out the construction of P, we have to know whether we are allowed to observe what the position of D with respect to the circle is. In particular this problem comes up for the practical question what should happen if there is insufficient precision for concluding whether D is inside or outside.

If we think of a construction with actual physical means like paper, pencil,

ruler and compass, then the case of D lying exactly on the circle is, of course, undecidable.

The above construction problem may seem to be very artificial, but yet its main characteristic turns up in very many geometrical constructions: it is the fact that, at some point of the construction the result of some observation will decide the further course of the construction. An example where this will happen is the case of geometrical constructions that have to be carried out inside a given finite part of the plane.

The naive approach to observability may be formulated as the slogan "truth is observable" (see Section 4). Other possibilities will be sketched in Sections 8–10.

A further thing one might like to formalize is selectability: one wants to be able to select an object from a set of objects one has constructed. For example, a construction of the intersection of two circles may produce two points, and we may wish to be able to "take one of them". In this case such a selection principle is not indispensable: one might describe the effect of the construction of the intersection as giving a labelled "first point" and a labelled "second point". But there is a stronger reason for implementing a selection principle: so often we have to "take an arbitrary point" at some stage of a construction. It should be noted that in such cases the final result of the entire construction does not depend on the particular point that was taken. In Section 5 we come back to this, in particular to the matter of the difference between "giving" and "taking" arbitrary points.

A description of all these features is possible in Automath. We have various options for doing it. The way we present this matter is necessarily arbitrary.

It is certainly not the intention of this note to give a particular basis for geometrical construction theory. The only thing that will be attempted is to provide a framework into which such a basis might be placed.

If we formalize a thing like constructability we of course dislike to do it in the style of classical logic. We do not want to consider constructability of a point as a proposition in the ordinary sense. We do not want to admit arguments where we get a contradiction from the assumption that the point P is not constructable, and then conclude the constructability of P. Therefore we want to put constructability (and the same thing might apply to observability and selectability) in a framework of positive logic, where we have no negation at all. In fact we can be even more restrictive, and refrain from introducing the ordinary logical connectives (like $\wedge$, $\vee$, $\rightarrow$) for this logic. The only thing we want to do is to register statements about constructability, observability and selectability (possibly provided with a number of parameters), and to keep them available for later use.

We can provide facilities for such a positive logic in Automath by adding a new expression of degree 1, to be called `pprop` (the first p stands for "positive"). For this `pprop` we shall not proclaim any logical axioms, and we shall not introduce the notion of negation. Moreover, we do not feel the need to have abstraction in the world of `pprop`. That is, if u: `pprop` we shall not take abstractors [x:u] like we would have in cases with `prop` or `type`. Accordingly, in this `pprop` world we shall not consider application $\langle .. \rangle ...$ either. That means: we take `pprop` entirely in the style of PAL (see [*de Bruijn 80 (A.5)*]).

There is a case for doing something similar in the world of `type`. Let us create a new expression of degree 1, to be called `ctype` (the 'c' stands for 'construction', since we intend to use it in the world of constructions). The difference between `ctype` and `type` is similar to the difference between `pprop` and `prop`. In `ctype` we intend to be free from all the assumptions that might have been made about `type`. In particular we shall not necessarily implement set-theoretical notions. And we shall not even introduce the notion of equality. That is, if $a : C$: `ctype` and $b : C$: `ctype`, then we will not introduce the equality of a and b as a proposition. Moreover, we shall treat `ctype` entirely in the style of PAL: no application and no abstraction.

For a description of Automath versions where various sets of rules apply to various expressions of degree 1, we refer to [de Bruijn 74a].

It has to be admitted that geometry is not the easiest example for the study of constructions. It is not so much the fact that the geometrical universes like planes, spaces, are uncountable. The most troublesome thing is neither that in the geometrical plane there is no fixed origin and no fixed direction. The real course of trouble is that there are so many situations where we have to except some of the cases. If we want to say that points p and q have just one connecting line we have to exclude the case $p = q$. Such things cause a steady flow of exceptions, which even has distorted the meaning of the word "arbitrary". In past centuries the word "arbitrary" often had the meaning: "arbitrary, but avoiding some obvious exceptions", and these exceptions were usually unspecified. If one took an arbitrary point and an arbitrary line then the point should not be so arbitrary to lie accidentally on the line!

A full description of all these exceptions has the tendency to make geometrical construction theory unattractive. Yet there is still another source of irritation: so often we have to split into cases (two circles may have 0, 1 or 2 points of intersection), and these situations might pile up to an entangled mess.

Nevertheless we may be grateful to geometry for having confronted us with the notion of constructability. What we have learned from geometry might be applied to other areas. Computer science might be one of them.

Observability, as a formal element in geometrical construction theory, was considered by D. Kijne [Kijne 62]. That paper also attempts a formal treatment

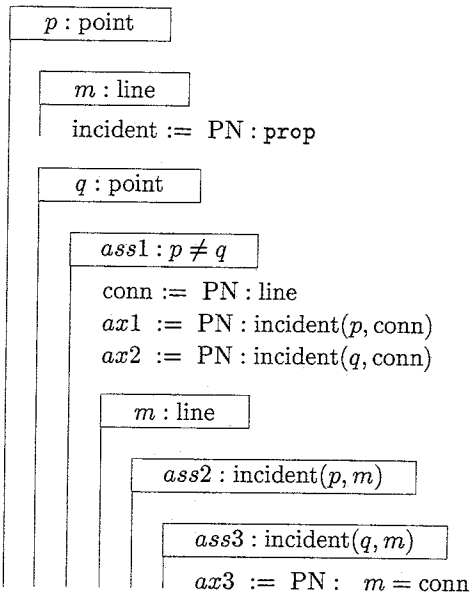
of selectability (with selection from finite sets only), and considers "giving arbitrary points" by means of a kind of algebraical adjunction operation.

3. THE BASIS OF FORMAL GEOMETRY

Before we discuss a formal basis for geometrical constructions we have to say what "formal geometry" or more generally, formal mathematics is. Here we are not concerned about the contents of formal geometry, but just about the spirit in which it is written. We may assume that it is written in an Automath book, using the full power of typed lambda calculus. And that it is written in a setting of logic and set theory, the details of which are still open to discussion. One might or might not take the rules of classical logic (e.g. in the form of the double negation law), and we might differ in taking or not taking a thing like the axiom of choice. Such distinctions hardly influence the spirit in which geometry is presented. They might influence the content, i.e. the set of all provable geometrical statements (but it should be remarked that there are areas of mathematics which are much more susceptible to foundational differences than classical geometry seems to be). Just to give an idea of the spirit, we give a small piece of Hilbert's axiomatization of geometry. Hilbert starts with: there are things we call points and there are things we call lines (in Hilbert's system the notion of a line is not presented as a special kind of point set). In Automath we say this by creating primitive types "line" and "point". These types are undefined, just introduced as primitive notions (PN's). As a primitive we also have the notion "incidence" of a point and a line. Next we can express axioms like: if two points are different, then there is exactly one line incident to both points. Something should be said about "different". We take it that our geometry text is written in a mathematics book in which for any two objects a, b of type A there is a proposition that expresses equality of a and b, and that for any proposition we can form the negation. In this way the fact that a and b are different can be expressed in Automath by means of $\mathrm{NOT}(\mathrm{IS}(A, a, b))$. But in order to keep this paper readable we shall just write $a \neq b$ instead of this.

We now give a piece of Automath text that can be considered as the start of a Hilbert-style geometry book (we display our Automath texts in a flag-and-flagpole format: the block openers are written on flags, and the poles indicate their range of validity).

point := PN : type
line := PN : type



So if p, q are points, and m is a line, then incident(p, m) is a proposition; if pr is a proof of $p \neq q$ then conn(p, q, pr) is the connecting line of p and q. In Axioms 1 and 2 we have expressed that this line is incident to p and q, in axiom 3 it is stated that if a line m is incident to both p and q then m is equal to the connecting line.

Although the above fragment is still a meagre piece of geometry it is hoped that it shows the spirit of a formalization. We shall refer to such a presentation of geometry as $\mathcal{G}$.

4. A NAIVE APPROACH TO OBSERVABILITY

What we shall call the naive approach is expressed by the slogan "Truth is Observable". Let us explain what this means by mentioning two cases.

In the first case we use knowledge obtained from geometrical theory $\mathcal{G}$ in order to prove that some object we have to construct is already in our possession. We do not bother whether that proof is "constructive" or not: truth is just truth. One might find this a poor example, since within the sope of usual geometrical theories and usual constructions it seems that "non-constructive" proofs can

always be replaced by very constructive ones, but it is easy to imagine fields where the situation is different.

In the second case we have a construction that started from a point that was chosen arbitrarily. At some stage of the construction we have a point P and a circle c, and subsequently our course of actions is depending on whether P lies inside c, outside c or on c. The naive point of view says that on the basis of the theory in $\mathcal{G}$ we have exactly one of the three alternatives. We can observe which one of the three occurs, and we act accordingly.

In Sections 6 and 7 we offer two different implementations for the naive point of view.

5. TAKING ARBITRARY OBJECTS

Before going on, we have to make it clear that there are two entirely different situations where in traditional geometry it was said that an arbitrary object (like a point) was taken. Let us call these situations $\mathcal{D}$ and $\mathcal{S}$, (these letters abbreviate "data" and "selection"). If we think of a problem where a teacher requires a pupil to construct something, then $\mathcal{D}$ is the case where the data have been chosen arbitrarily by the teacher. On the other hand, $\mathcal{S}$ is the case where the pupil, in the course of the construction, selects some point arbitrarily. Quite often the final result does not depend on the particular point that was chosen, but there may be other cases. It may happen that the final result itself has a kind of arbitrariness. An example: given points A, B and C, not on a line, construct a point inside the triangle formed by A, B and C.

In the opinion of the pupil, the points taken in situation $\mathcal{D}$ are not called "arbitrary": they are called "given", or possibly "arbitrarily given". The pupil has no freedom in case $\mathcal{D}$. In the $\mathcal{S}$-case, however, the pupil is completely free, and the teacher has no say in the matter.

In a formal presentation like in Automath the difference between $\mathcal{D}$ and $\mathcal{S}$ is very pronounced. $\mathcal{D}$ is effectuated by means of the introduction of a new variable, $\mathcal{S}$ is implemented by means of a primitive notion (PN). We shall show this in detail in Section 6 and 7.

There is something about the PN-implementation of the $\mathcal{S}$-situation that might be felt as strange. If we describe a construction by such a PN, then we select exactly the same point if we are requested to do the construction a second time. If the second time we would insist on selecting a point that is actually different from the one chosen the first time, then we have to do this on the basis of some new selection principle, of course. But if we just want to take a point again, without any restriction as to its being different from or equal to the first one, our PN provides us with the same point we had before. This means that we

get more information than we intended to have. Nevertheless, such information cannot possibly do any harm.

What shall we do about this weirdness of the PN-implementation? Shall we invent unpopular remedies in order to cure a completely harmless disease? Let us not prescribe a definite attitude in this, and admit that there are several ways to live with the situation. Either we leave the harmless disease for what it is, or we take one of the remedies. Let us mention two remedies. The first one is to take a notion of time t, and adhere a value of t to every construction step. The arbitrarily selected points will depend on t. If we have to repeat the construction some other day, t has a different value, so nothing is known about the selected point in comparison to the one selected the previous day.

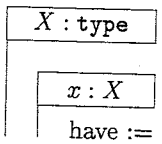
As a second remedy we suggest to implement arbitrary selection not by an axiom but by some axiom scheme. The scheme proclaims the right to create as many copies of the axiom as one might wish, each time with a different identifier.

We leave it at these scanty remarks. The author's opinion is that unless we invent a much simpler cure, we'd better learn to live with the harmless disease.

6. FIRST IMPLEMENTATION OF THE NAIVE POINT OF VIEW

We have to express in some way or other that some of our mathematical objects have been constructed. This can be thought of as a property of those objects, but for reasons sketched in Section 2 we prefer to take this property as a `pprop` rather than as a `prop`. We shall create, for every type X and for every x of type X, the expression have(X, x) with have(X, x) : `pprop`. In particular we can abbreviate have(point,x) to havep(x) and have(line,x) to havel(x). (Since we use "have" for points and lines only, one might think of taking just "havep" and "havel" as primitives, without taking "have" for general types.)

We now give some Automath text. It is supposed to be added to a book that contains geometrical theory $\mathcal{G}$ (see Section 3) already. First we introduce "have", and abbreviations "havep" and "havel".



u : point

havep := have(point,u) : pprop

v : line

havel := have(line,v) : pprop

Next we display how we take an arbitrary object in the sense of the $\mathcal{D}$-situation of Section 5 ("given objects"). In order to talk about a given point we need two block openers, expressing

(i) that u is a point, and

(ii) that havep(u) holds;

inside that context the point u can be considered as given. We shall now express: if u and v are given points and if $u \neq v$ then we can construct the line connecting u and v. According to our naive point of view the condition that u and v are different is simply expressed in the terminology of $\mathcal{G}$.

u : point

$ass11$: havep(u)

v : point

$ass12$: havep(v)

$ass13$: $u \neq v$

$ax11$:= PN : havel(conn($u, v, ass13$))

Next we describe a case of "taking arbitrary points" in the $\mathcal{S}$-situation of Section 5. We express that if m is a given line then we are able to take a point not on m (we use the identifier "ap" to suggest "arbitrary point").

m : line

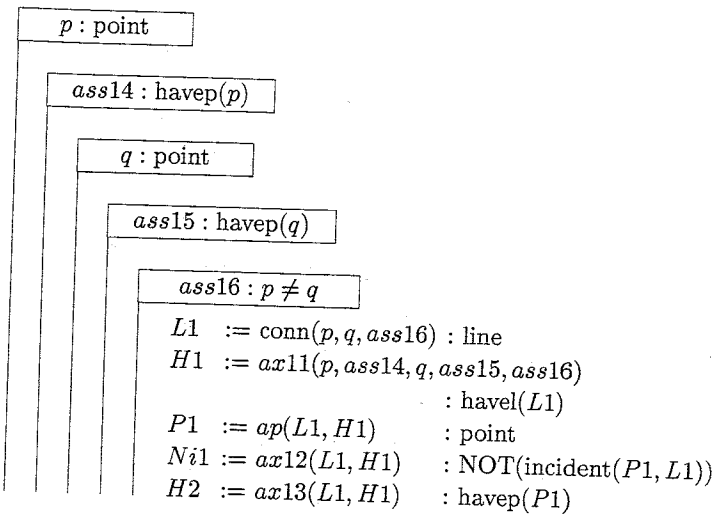
$ass14$: havel(m)

ap := PN : point

$ax12$:= PN : NOT(incident(ap, m))

$ax13$:= PN : havep(ap)

These pieces of text display the form in which the basic constructions are introduced. If we want to describe a more complicated construction, we mention the relevant objects one by one, in the order of the construction, and each time we express that we "have" them. We give a (still very simple) example.



Here $L1$ is an abbrevation for the line connecting p and q; $H1$ can be used as a reference for the fact that we actually have that line. $P1$ is the result of the construction, $Ni1$ assures us that $P1$ does not lie on $L1$, and $H2$ assures us that we actually have $P1$. Altogether the text line with identifier $P1$, $Ni1$, $H1$ represent the "derived construction" expressing that if p and q are given different points then we can take a point $P1$ such that p, q, $P1$ are not on one line. This derived construction can be applied later without referring to how it came about. It can be constructed as a kind of "subroutine".

The example of a derived construction we gave here is ridiculously simple, of course. Yet the pattern is the same as in more complicated cases. It shows the old idea of subroutines, which existed in constructive geometry many centuries before it came up in computer programming.

7. SECOND IMPLEMENTATION OF THE NAIVE POINT OF VIEW

In the second implementation we take a construction plane which we conceive as being different from the geometrical plane. We might think of the original geometrical plane as abstract, and the construction plane as concrete, consisting

of a piece of paper we can draw on. But, of course, our construction plane is still abstract: it is a mathematical model of a concrete plane. The objects in the constructed plane will be called cpoints and clines.

In the back of our minds we think of a one-to-one mapping between the two planes: every cpoint has a point as its companion, and every cline has a line as its companion. Yet we shall not express all of this in our mathematical formalism. We shall just talk about a mapping (to be called semp) of cpoints to points and a mapping (to be called seml) of clines to lines. The reason for this reticence lies in the interpretation. If $p1$ is a point, and if we are able to name a cpoint $cp1$ that is mapped to $p1$ in our mapping, then for us this means that we "have" $p1$. We do not want to say that every point in the geometrical plane is a point we "have" just by being able to express that point mathematically. Therefore we do not want to be able to express the inverse mapping.

Related to this reticence is the fact that we do not want to be able to discuss equality of two cpoints. Such equality has to be discussed for the companion points in the geometrical plane. And we do not want to admit as mathematical objects things like "the set of all cpoints" with some prescribed property. We achieve these restrictions by putting "cpoint" and "cline" into `ctype`, which is a world without equality, without set theory, without quantification. As a consequence we do not have constructability questions in our theory. A statement: "the point P is not constructable with ruler and compass" will not be a proposition in our Automath book. If we would be able to quantify over the construction plane we would be able to express that "there is no cpoint that is mapped onto P" and that would express the non-existence of the construction. Constructability questions belong to the meta-theory. They express that something "cannot be obtained on the basis of the PN's displayed thus far", and we cannot say such things in Automath itself.

What we call our second implementation starts with the introduction of cpoint, cline and the mappings semp and semf. The latter abbreviations suggest the word "semantics": we might say that the geometrical plane forms the semantics of the construction plane. If P is a cpoint then semp(P) is its semantics. Off we go:

```
cpoint  := PN : ctype
cline   := PN : ctype
```

```
cp : point
semp := PN : point
```

cl : cline
seml := PN : line

In order to take an arbitrary point in the construction plane, a single block opener "x : cpoint" plays the role of the pair "u : point", "$ass11$: havep(u)" of the first implementation. We show this with the fundamental construction that connects two points:

x : cpoint
y : cpoint
$ass21$: semp(x) $\neq$ semp(y)
cconn := PN : cline
$ax21$:= PN : seml(cconn) = conn(semp(x), semp(y), $ass21$)

The fact that cconn is the line we are looking for, is expressed (in $ax21$) by means of equality in $\mathcal{G}$.

If we have to take an arbitrary point in the $\mathcal{S}$-situation we again get one PN less than in the corresponding case of Section 6. In order to express that, we can take a point outside a line, we write

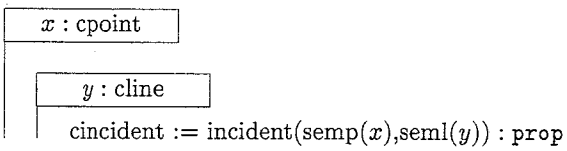
cm : cline
acp := PN : cpoint
$ax22$:= PN : NOT(incident(semp(acp), seml(cm)))

We also show the text corresponding to the one with $P1$, $Ni1$, $H2$ in Section 6:

p : cpoint
q : cpoint
$ass22$: semp(p) $\neq$ semp(q)
$CL1$:= cconn($p, q, ass22$) : cline
$CP1$:= $acp(CL1)$: cpoint
$Ni2$:= $ax22(CL1)$: NOT(incident(semp($CP1$),seml($CL1$)))
$Ni3$:= ... : NOT(incident(semp($CP1$), conn(semp(p),semp(q), $ass22$)))

We have not displayed the proof $Ni3$. It will depend on applying general axioms about equality, and will make use of $Ni2$ and $ax21$.

Passages like the one from $Ni2$ to $Ni3$ might be superfluous in many cases, since it is practical to keep the discussion as long as possible in the construction plane. To that end we might copy notions from $\mathcal{G}$ to the construction plane. The simplest example is



8. RESTRICTED OBSERVABILITY

In Sections 4, 6, 7 we described the naive point of view, where every truth in the geometrical theory is considered to be "observable". Observability has its meaning in the process of taking decisions about the course of our constructions.

Let us describe two different motives for restricting observability. One is practical, the other one is fundamentalistic. We shall discuss these in Section 9 and 10, respectively.

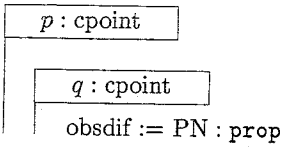
9. PRACTICAL RESTRICTIONS ON OBSERVABILITY

The practical point of view is connected to questions of precision. This can be compared to the matter of rounding off errors in numerical analysis. If in a construction two points turn out to be so close together that our construction precision does not guarantee that they are different, then we can not claim to be able to connect them by a line. And even if the points are different, the line will be ill-defined.

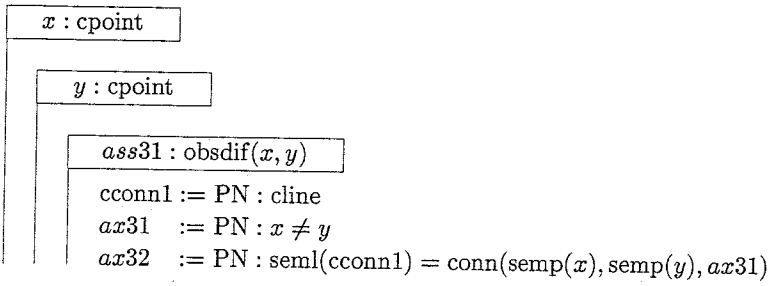
Although these practical matters give rise to quite complicated considerations, we cannot say that they are necessarily essentially different from what we did in Sections 6 and 7. One can still go on the basis that truth is observable: the question is just a matter of which propositions we consider the truth of. Instead of claiming the possibility to connect two points p, q if $p \neq q$ in the geometrical world G, we take a thing like $\mathrm{d}(p,q) > 1$ (distance exceeds unity) as our criterion.

Nevertheless we can make things a little livelier than this. Let us start from what we developed in the beginning of Section 7: just the four PN's that were called cpoint, cline, semp and seml. We now introduce a primitive notion

"obsdif" ("observationally different") in the construction plane:



And now instead of introducing the cconn, $ax21$, etc. of Section 7, we go on like this:



Knowledge about obsdif can come from different sources. In the first place we can axiomatize things like: if d(semp(x),semp(y)) > 1 then x and y are observationally different. A second source arises if we axiomatize in the construction plane, in some situations, that if cpoints u and v are observationally different, then the cpoints x and y, derived from u and v in one way or other, are observationally different. A very simple case of this is an axiom stating that obsdif(x, y) implies obsdif(y, x).

It will be clear that this subject will become very complicated without being very rewarding. Therefore it seems definitely unattractive.

10. FUNDAMENTALISTIC RESTRICTIONS ON OBSERVABILITY

In Section 9 we still had the uncritical acceptance of all truth that can be obtained in the geometrical world. There is a clear reason for restriction. If we have to use geometrical propositions for taking decisions in the world of constructions, it is reasonable to require that we also have a "constructive" way for actually deciding whether such propositions hold or do not hold.

We can implement such restrictions in Automath by selecting some "constructive" basis for logic and mathematics, like intuitionistic mathematics, and building our geometry G according to these principles. We might even mix

a constructive kind of mathematics with the ordinary kind, using `pprop` and `ctype` for the constructive kind. In particular it seems to be reasonable to take the "obsdif" we had in Section 9 as a `pprop` rather than as a `prop`.

The latter remark suggests that it might be simpler to shift life entirely to the construction plane, and to forget $\mathcal{G}$ altogether. But this is not what we usually want. Let us imagine that we want to describe the theory of Mascheroni constructions (constructions with compass but without ruler). The subject matter concerns both circles and straight lines, the constructions deal with circles only. This difference can be implemented by discussing both circles and straight lines in $\mathcal{G}$, but just "cpoints" and "ccircles" in the construction plane.

11. COMPARISON WITH COMPUTER PROGRAM SEMANTICS

It is very natural to compare the field of geometrical constructions with the one of computer programming. In both cases there is a number of actions that produce one or more objects, and in both cases it is very essential that it is proved that these objects satisfy the problem specification that was given beforehand.

In a computer program we usually think of a "state space"; the input is an element of that state space and the output is again such an element. In the case of geometrical constructions one would say that the input is (vaguely speaking) the given figure, and the output is the required figure. Let us admit different spaces for input space and output space, and try to describe at least the specification of a geometrical construction in terms of input and output. As an example we take the following (trivial) construction problem. Given two different points P, Q and a line m. Construct a line q that intersects m, passes through P but not through Q.

Let us talk in the style of Section 7, and let us moreover decide to introduce a name R for a cpoint of intersection of q and m (otherwise we would need existential quantification). An element of the input space is a triple (P, Q, m) where P : cpoint, Q : cpoint, m : cline, and where we have $\mathrm{semp}(P) \neq \mathrm{semp}(Q)$. An element of the output space is a pair (q, R) where q : cline and R : cpoint. The problem specification is given by the conditions that $\mathrm{seml}(q)$ is incident with $\mathrm{semp}(P)$ and $\mathrm{semp}(R)$ but not with $\mathrm{semp}(Q)$.

This kind of problem specification is entirely in the style of what is called "relational semantics" in computing science.

If we deal with geometrical constructions, the role of "subroutines" is more or less the same as in computer programming. In particular we can say that descriptive geometry consists of a large body of subroutines.

In computer programs we can have loops. Sometimes pieces of a program

have to be repeated until some condition is satisfied. The geometrical constructions we discussed in the previous sections have no such loops. This reveals essential restrictions on the class of constructions we can describe in the various systems that were suggested in these sections. An example of a construction with loop is the following one. Let A, B, C be given points on a given line, B between A and C. It is required to construct a point D on that line, such that C is between B and D, and such that the length of the line segment BD is an integral multiple of the length of the segment AB. This construction requires a loop.

Our treatment of geometrical constructions in Sections 3–10 might be called "operational" or anyway "functional". All the time uniquely determined outputs are obtained step by step, and in the slightly more sophisticated case of the use of subroutines the only thing we actually do is taking sequences of steps together and considering them as a single step. The reason is that the treatment is based on what we shall call the *interior* approach. In the interior approach we talk in terms of the constructed objects. The constructed objects are treated in the same style as ordinary mathematical objects and (but this is a typical Automath feature) proofs. In our Automath book we discuss the objects, but the action of construction is felt as subject matter of some metalanguage.

An entirely different way to deal with constructions is that we consider constructions as objects, seemingly more abstract than the ordinary objects, but nevertheless on the same linguistic level. Let us call this the *exterior* approach. (The name is suggested by the fact that if we work in the interior approach then the metalinguistic discussion of construction is felt as being something at the outside.)

With the exterior approach we can get rid of the limitations of our "functional style" of construction description. Anyway we can remove the last differences there might be between geometrical construction and computer programming.

We might try to start the exterior treatment with the introduction of a primitive notion "construction" like:

construction := PN : `ctype`

but it has to be more complicated than this. The notion of construction has to depend on the input space and the output space as parameters, and this is not so easy to describe.