THE JOURNAL OF SYMBOLIC LOGIC
Volume 52, Number 1, March 1987

A CONSTRUCTIVE INTERPRETATION OF THE FULL SET THEORY

VALENTIN F. TURCHIN[1]

§1. Introduction. The interpretation of the ZF set theory reported in this paper is, actually, part of a wider effort, namely, a new approach to the foundation of mathematics, which is referred to as *The Cybernetic Foundation.* A detailed exposition of the Cybernetic Foundation will be published elsewhere. Our approach leads to a full acceptance of the formalism of the classical set theory, but interprets it using only the idea of potential, but not actual (completed) infinity, and dealing only with finite objects that can actually be constructed. Thus we have a finitist proof of the consistency of ZF. This becomes possible because we set forth a metatheory of mathematics which goes beyond the classical logic and set theory and, of course, cannot be formalized in ZF, yet yields proofs which are as convincing—at least, from the author's viewpoint—as any mathematical proof can be.

Our metatheory is based on the following two ideas. Firstly, we define the semantics of the mathematical language using the cybernetical concept of knowledge. According to this concept, to say that a cybernetic system (a human being, in particular) has some knowledge is to say that it has some models of reality. In the Cybernetic Foundation we consider mathematics as the art of constructing linguistic models of reality. An analysis of the concept of model shows that a model is, essentially, a generator of predictions. We formalize a prediction as the statement that a given process is finite. We declare a proposition meaningful if, and only if, it can be interpreted as a generator of predictions.

The second idea behind the Cybernetic Foundation is the introduction of the user of mathematical machinery into the formalism of mathematics. This leads to a new kind of processes, which we call *metamechanical.* A metamechanical process is initiated and maintained by a mechanical device like a Turing machine in interaction with the subject of knowledge, i.e. the user of the device. Another way to characterize a metamechanical process is to say that it occurs in *real time* in which the user lives, as opposed to the *model time* of a Turing machine or any other autonomous mechanical device. The class of metamechanical processes is wider than the class of processes which can be generated by autonomous mechanical devices. A set is seen as a metamechanical generator of its elements.

Received December 27, 1983; revised March 24, 1986.
[1] Partly supported by NSF grants DCR-8007565 and DCR-8412986.

0022-4812/87/5201-0015/$04.00

Because of space limitations, we give only a brief outline of the general theory; it includes the basic definitions and principles, but not much of justification or discussion.

We start with the principle that the external world is, for us, a collection of *processes*. We shall not try to go beyond this picture of the world into any of the possible metaphysical constructions. We see around us, and sometimes construct, cybernetic devices which have the power to partially foresee the future by somehow *reflecting* the processes of the external world in their internal structure, or, in other words, creating *models* of those processes. By extension to ourselves, we postulate that our own knowledge of things is of the same nature, i.e., informally, a model of some of the world's processes, an instrument to produce predictions about those processes. A mathematical proposition, to be meaningful, must be interpreted in these terms.

Formally, if A is a process, we define the *prediction* $A!$ as the statement that A is *finite*, i.e. achieves a certain stage after which there is no change. Natural sciences produce predictions about different physical processes. Mathematics—at least its "arithmetized" part—produces predictions about processes in symbol-manipulating machines, such as Turing machines.

There are a number of formalisms to describe mechanical symbol manipulation: recursive functions, Turing machines, lambda-calculus, etc. For our purposes, the most convenient formalism is that of recursive functions on the set of strings in a certain alphabet. We require that among the symbols of the alphabet there must be two round parentheses; only the strings with properly matched parentheses, which we call *expressions*, are allowed. The machine that evaluates, step by step, recursive functions on expressions, is *the Refal machine* (Refal—which is the acronym for REcursive Functions Algorithmic Language—is, actually, a real computer language). Mathematical processes, about which we make predictions, are consecutive stages of a computation process by the Refal machine. A formal definition of Refal is given elsewhere [Turchin 1980]. Here we shall use the Refal machine informally; the reader will easily agree that the processes we shall discuss could be formalized in any of the more familiar formalisms, say, the Turing machines.

A recursive function will be often referred to as a *machine*, because in response to an input (the argument) it initiates a computation process (that may or may not be finite) and produce the value of the function. The Refal machine is, essentially, a programmable metamachine used to define recursive functions. Machines we define will often be dealing with machines; a Gödelization is implicit here, which we need not make more precise.

In addition to computation processes, which we shall refer to as *searches* (for the answer, the value), we consider *generators*, which originate generation processes. A generator, as it works, produces a chain of parenthesized expressions; this can also be seen as putting an expression into a certain place, which is then cleared for the next expression. Finally, the Refal machine has special *access functions*, which signify a radical departure from the autonomous Turing machine. This will be discussed later.

The definition of a *mathematical proposition*:

(a) a prediction is a proposition;

(b) a generator which generates only propositions is a proposition (to be referred to as a *P-generator*).

Thus a proposition may produce a whole hierarchy of propositions, but they must be such that ultimately they produce predictions. We say that *P hierarchically produces Q*, if there is a finite chain $P_1, P_2, \ldots P_n$, such that P_i produces P_{i+1}, $P_1 = P$, $P_n = Q$. A formal object has a meaning as a proposition only to the extent we know how to make it hierarchically produce predictions. If there is no way to obtain predictions from an object, it has no meaning as a proposition.

The statement that a given search A is infinite can be interpreted as a model (prediction generator). Indeed, to state that A is infinite is to state that:

- the initial stage A is not final;
- the next stage after the initial stage is not final;
- the next stage after the next stage after the initial stage is not final;

$\cdots$ and so on, infinitely.

An expression which represents a final stage of a process is *passive*, otherwise it is *active*. In Refal, a simple syntactic feature distinguishes active and passive expressions. Every one of the above statements can be formalized as a prediction by defining a process that checks whether a given expression is passive, and stating that this process, when applied to each next stage of A, is finite. Thus the infinity of a certain process is an infinite generator of predictions.

Because of the importance and frequent use of the infinity model we introduce a special notation for it. The proposition that a search A is infinite will be represented by the expression $A?$, and we treat such propositions, together with predictions, as certain elementary units, *atoms*. Thus, propositions $A!$ and $A?$ will be called *atomic*. One should bear in mind, however, that while $A!$ is a prediction, $A?$ is a *generator* of predictions.

Some composite propositions can be immediately interpreted as prediction generators if their constituents are. We define the machine $\underline{\text{and}}(P, Q)$ as a generator which produces P and Q. If P and Q are hierarchical prediction generators, then $\underline{\text{and}}(P, Q)$ conveys the meaning of the conjunction of P and Q. One can see that such propositions as $(\text{A}x)(x + 0 = 0)$ and $(\text{E}x)(5 + x = 8)$ can be interpreted as propositions in the sense of our theory. However, other logical connectives, e.g. disjunction, cannot be interpreted in this way. The meaning of $(P \vee Q)$ includes a reference to the concept of truth ("at least one of P and Q is *true*"), and there is no way to bypass it.

The criterion of truth. A prediction $A!$ is true if and only if the search A is finite; a generator-proposition P is true if and only if every prediction it hierarchically produces is true.

The predictions $A!$ and $B!$ are *incompatible* if they cannot be true simultaneously. It may happen that the incompatibility of two predictions is beyond any doubt, as in case of equality and inequality of two natural numbers. The direct use of the criterion of truth allows us to verify only one statement: that a given prediction $A!$ is true. If there is an incompatible prediction, then we may be able to verify that the prediction is false, i.e. $A?$ is true. For a generator producing infinitely many

predictions, the situation is much worse. We can only verify that it is false, but not that it is true (this was emphasized, with respect to natural sciences, by Karl Popper).

Yet the meaning of many of the logical constructs, if we want to interpret them as predictions or prediction generators, includes a reference to the process of establishing that a given general proposition (generator) is true. This becomes obvious if we think of the meaning of the proposition "if the search A is infinite, then P". This proposition is a generator that tries, somehow, to establish the infinity of A, and if it succeeds, it produces P. The disjunction $P \vee Q$ implies running in parallel two searches: those establishing the truth of P and Q. The disjunction is true if at least one of the searches succeeds.

The process through which we convince ourselves that a proposition P is true will be denoted as $\gamma(P)$. The pronoun "we" here is, essentially, impersonal. It stands for the *abstract user* of mathematical machines. To make this idea exact, we introduce a new symbol Γ, which is referred to as the *access function for the user's knowledge.* Whenever Γ is to be evaluated by the Refal machine, it asks the user to give the proposition that summarizes his total knowledge at the moment. After the user does so, the Refal machine goes on with computations. The search $\gamma(P)$ is defined now as $\underline{\text{imp}}(\Gamma \to P)$, where the function $\underline{\text{imp}}$ runs its first argument Γ as a hierarchical generator, and stops if and only if it finds that P is produced. It should be stressed that the Refal machine thus extended is still a completely and exactly defined object, and so are all machines and expressions defined through the Refal machine and the symbol Γ. It is true, though, that in different hands, or at different times, this instrument may give different results.

The function $\underline{\text{con}}(P)$ is a machine that runs P hierarchically and stops if and only if it finds that $A!$ and $A?$ are produced, with some A. The negation of a proposition P is interpreted as the finiteness of the search $\bar{\gamma}(P) = \underline{\text{con}}(\underline{\text{and}}(\Gamma, P))$, i.e. that P added to some true knowledge Γ will produce a contradiction. The functions γ and $\bar{\gamma}$ are referred to as *cognitive* functions.

The question presents itself immediately: how do we know that the user of the Refal machine does not supply a false proposition as Γ? Of course, we do not know that. But we make no assumptions about the real users.

One must distinguish between the *abstract* user and a *real* user. The abstract user is anybody who uses the interactive Refal machine, viewed from the machine's viewpoint, so to say. The design of the Refal machine calls for a certain way to use it, but tells us nothing about the identity, or character, or the extent of the knowledge of its user: the user remains *abstract*. A *real* user is, presumably, a human being, although it could be a Martian, or any other creature. A real user may make errors. Then he may come to a contradiction and retract to a safe part of his knowledge. The path of knowledge of a real user can be represented by the picture in Figure 1. The loops here represent the user's fallacies, later corrected. We call a user *ideal* if he never makes these loops, but manages to get through by the path shown as the bold line in Figure 1. This is a usual idealization, one of the cornerstones of the scientific method. The ideal user is not unique; there may be infinitely many ideal users moving towards truth by different paths.

It is important that our definitions depend only on the concept of the abstract user. But we assume, when actually constructing knowledge, that the knowledge we

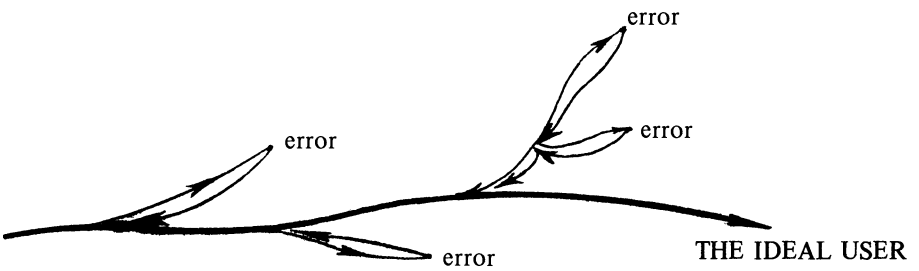

FIGURE 1. The path of knowledge

have at the moment is true, i.e. the abstract user is an ideal user. (This is only too natural: when we move forward we assume that all is right behind, otherwise we return and make a correction). There is no problem of "how do we know that the user is ideal", because in our initial definitions we use only the abstract user, about whom no assumptions are made. When I, as a real user, start accumulating knowledge, the user is myself. So I only need to convince myself (and the reader) that at every stage of the proof, the pieces of Γ I accept as true are really true. Then the whole proof is as sound as it could be.

Thus the following principle can be used in the construction of new knowledge: *The Consistency Principle.* Γ is always consistent, i.e. $\underline{\text{con}}(\Gamma)$ is infinite. For the abstract user of the Refal machine this becomes *The Consistency Rule.* If you have found that there is a contradiction in your current value Γ_i of Γ, you cannot leave it as it is; you must go back to a consistent part of Γ.

Processes that call access functions are *metamechanical* processes. They take place in the real time of the user, are inseparable from the user, and cannot be understood if we think of them as if they were autonomous (deterministic or nondeterministic) processes.

Metamechanical processes are certainly nondeterministic. Therefore the concept of being finite requires, for a metamechanical process, an additional definition. We define that a process P calling Γ is finite, symbolically $\gamma(P)!$, iff there is a true proposition Γ_i such that with the value of Γ replaced by Γ_i the process becomes finite. It follows immediately that P is true if and only if $\gamma(P)!$. This definition only relates the concept of truth to the finiteness of a metamechanical process, but gives us no independent definition of either. Indeed, our criterion of truth refers to the truth of a prediction, which is defined through the finiteness of a process; at the same time the finiteness of a process, if it is not mechanical, is defined through the concept of truth. The concept of *objective interpretability*, to be introduced below, gives a meaning to truth of *some* (namely, objectively interpretable) propositions referring to finiteness of metamechanical processes.

The *Completeness Principle* follows: Every true proposition is implied (i.e. produced) by Γ. Here we see the fundamental difference between mechanical and metamechanical processes, because for no mechanical process can the Completeness Principle be true. If the process Γ of knowledge production is mechanically defined, then by making a metasystem transition of the Gödel type we can construct a new

knowledge, i.e. a proposition that is definitely true but not in Γ. However, with a metamechanical process, "we" is a part of the process itself, which cannot be known in advance; so we cannot make a metasystem transition, we cannot separate ourselves from Γ. The moment we have constructed a new piece of knowledge, it is in Γ, by definition. The operationalist content of the Completeness Principle is the *Completeness Rule* for the abstract user: whenever you have proved that a proposition P is true, you must include it in your current value Γ_i of Γ; you cannot pretend that it is not there.

Both intuitionistic and classical logic, as used in first-order theories, can be interpreted from these positions. If the set of all objects is given by a generator which we shall denote as Λ, and the primitive predicates are somehow defined as predictions or prediction generators, then Table 1, where $[P]$ is our interpretation of the proposition P of formal logic, defines the usual logical constructs.

TABLE 1. Interpretation of logical constructs

$$
\begin{aligned}
[\sim P] &= \bar{\gamma}([P])! \\
[P \,\&\, Q] &= \underline{\text{and}}([P],[Q]) \\
[P \vee Q] &= \underline{\text{or}}(\gamma([P],\gamma([Q]))! \\
[P \to Q] &= \underline{\text{if}}\ \gamma([P])!\ \underline{\text{then}}\ [Q] \\
[P \equiv Q] &= [P \to Q \,\&\, Q \to P] \\
[(\text{A}x)P(x)] &= \underline{\text{all}}(x \in \Lambda\colon [P(x)]) \\
[(\text{E}x)P(x)] &= \underline{\text{sch}}(x \in \Lambda\colon \gamma([P(x)]))!
\end{aligned}
$$

The generator $\underline{\text{and}}(X, Y)$ produces exactly two expressions: X and Y. The search $\underline{\text{or}}(X, Y)$ runs in parallel the searches X and Y, and stops if either stops. The generator $\underline{\text{if}}\ X\ \underline{\text{then}}\ Y$ runs the search X and when (and if) it stops outputs Y. The generator $\underline{\text{all}}(x \in X\colon Y)$ runs the generator X, substitutes each produced object for every entry of x in Y, and outputs all results. The search $\underline{\text{sch}}(x \in X\colon Y)$ runs the generator X, substitutes each produced object for every entry of x in the search Y, and runs in parallel all resulting searches; it stops when at least one of the searches stops. P and $\gamma(P)!$ are equivalent.

With the definition of the functions $\gamma(P)$ and $\bar{\gamma}(P)$ we gave above, i.e.

$$\gamma(P) = \underline{\text{imp}}(\Gamma \to P), \tag{1}$$

$$\bar{\gamma}(P) = \underline{\text{con}}(\text{and}(\Gamma, P)) \tag{2}$$

the law of the excluded middle, which says that $\underline{\text{or}}\,(\gamma(P), \bar{\gamma}(P))!$ holds for all P, is not true. Indeed, after the first step in which Γ is replaced by the current knowledge Γ_i, γ and $\bar{\gamma}$ become purely mechanical; hence there always will be propositions which will neither be proved, nor refuted. With these definitions of the truth functions, our logic is intuitionistic. Note, that intuitionistic logic in our interpretation does not tie us to a definite method of proving Γ_i, it only assumes that this method stays *fixed* in the process of truth-seeking.

We can, however, modify these definitions so as to interpret the classical concept of truth in mathematics, which does not identify truth with some method of truth-

finding, but assumes that for every "meaningful" (whatever it may mean) P, *some* method Γ_i can somehow be created, and probably will be created at some point in time, such that it will be possible to prove or disprove P. The functions γ and $\bar{\gamma}$ corresponding to this notion run in parallel all the searches (1) and (2) with Γ changing in real time as our knowledge grows: $\Gamma_1, \Gamma_2, \ldots, \Gamma_i, \Gamma_{i+1}, \ldots$ and so on, endlessly. Thus, if there exists such a Γ_i which produces P, then $\gamma(P)$ will be finite, and analgously for $\bar{\gamma}(P)$. With these definitions, the Completeness Principle holds, and the law of excluded middle is true.

We have no space here for a more detailed discussion of logic. Our goal is the interpretation of set theory.

§2. Basic objects. Interpretability. The idea of our interpretation of a set theory which includes the concept of uncountability is to introduce another access function and, therefore, a real-time process, which would describe the construction, by the abstract user, of the set-theoretical universe of objects, much like Γ describes the construction of the universe of knowledge. We denote this access function as Λ. The consecutive stages of Λ, which will be called *subuniverses*, are $\Lambda_1, \Lambda_2, \ldots, \Lambda_i$, etc. Sets are interpreted as metamechanical generators of their elements which can call both Γ and Λ. We maintain that the reason for our intuitive conviction that classical logic is true and reliable is exactly its constructive interpretation that was outlined above. Then the interpretation of set theory along the same lines should be as convincing.

To avoid confusion between sets and propositions understood intuitively or as objects of the ZF set theory on one hand, and set generators and propositions (prediction generators) of our theory, on the other, we shall often refer to the latter as S-objects and P-objects, respectively. Collectively, they are addressed as SP-objects.

Definition of an S-object. (1) An empty expression is an S-object. (2) A generator producing only S-objects is an S-object.

Definition of a P-object. (1) A prediction is a P-object. (2) A generator producing only P-objects is a P-object.

The function Γ is used only through the functions γ and $\bar{\gamma}$ as defined above for the case of classical logic.

In our notation it becomes clear from the context, or is explicitly stated if necessary, whether a function symbol is *active*, i.e. represents a function call to be evaluated, or it is *passive*, i.e. just a symbol, a part of a symbolic expression. For example, in (2), con and Γ are active, while and is passive. It is easy to see why. The function $\bar{\gamma}$ checks whether the *formal expression* representing the logical conjunction of the value of Γ and P is contradictory. Hence and is passive here. In terms of recursive theory, con is a function of the Gödel number of its argument, not of its value (which, as a rule, does not exist). When it becomes necessary to use the meaning of conjunction, the symbol and will be activated by the controlling function con, and the definition of the function and will be used.

An SP-object which never calls Λ will be called *fixed*. Consider an S-object X. It is a machine. Run it, and whenever Λ becomes active replace it by a fixed S-object Λ_i (subuniverse). The machine thus modified is a *lambda-implementation* of X. Note that the passive entries of Λ are not replaced by Λ_i, but remain as were.

With every SP-object its *semantic graph* is associated. This is, essentially, the

history of the process represented by the object, with one significant alteration: when $\gamma(P)$ or $\bar{\gamma}(P)$ is encountered, a *metasystem transition* takes place. By this we mean that instead of recording the computation history of the cognitive function γ or $\bar{\gamma}$, the graph proceeds with the computation history of P. The nodes of the semantic graph are Refal expressions representing SP-objects; the arcs lead from one stage of the process to another. For a prediction $S!$, the process traced is the search S (so we may speak of searches as P-objects, meaning the corresponding predictions). Parallel processes (both in generation, and in search) are represented by parallel branches, i.e. arcs starting from the same node. So the semantic graph of an SP-object is a tree, which is growing (infinitely, in the general case) as time goes. It is a constructive object, whose infinity is potential.

Speaking of parallel branches, one must keep in mind that it takes some time for the Refal machine to initiate each new parallel process. Thus parallel branches are ordered by the process of their initiation. The number of parallel branches starting from a node may be infinite.

In graphic representation we show generator nodes (S-objects and P-generators) as bold dots ●, and predictions (searches) as ▼. The metasystem transition from $\gamma(P)$ to P will be shown as ○, where the circle denotes the call of γ, and the dot the proposition P; the same for $\bar{\gamma}(P)$ is □. A short vertical bar will show the end of a search. As an example, we give in Figure 2 the semantic graph for the P-object corresponding to the proposition $(\mathrm{A}x)(\mathrm{E}y) \sim (R(x, y) \to S(x, y)!)$, where $R(x, y)$ is some P-generator, and $S(x, y)!$ a prediction. We shall show how we arrive at this graph by using a more exact algebraic notation of semantic graphs.

In algebraic notation the semicolon ";" separates one stage of a process from another, and the arrow "→" is used to denote production of an object by a generator. The plus sign "+" separates parallel branches. When we have a potentially infinite number of parallel branches, we use x, y, z, possibly superscripted, as bound variables denoting the "typical" branch object. Some distinguished values of variables are marked by subscripts. Naturally, when we trace a process we do not necessarily write down each consecutive stage, but only some key stages that we use in argument. For convenience of notation we shall represent subexpressions of Refal expressions by numbers enclosed in angular brackets. The further evolution of a subexpression will be taken care of on a separate line.

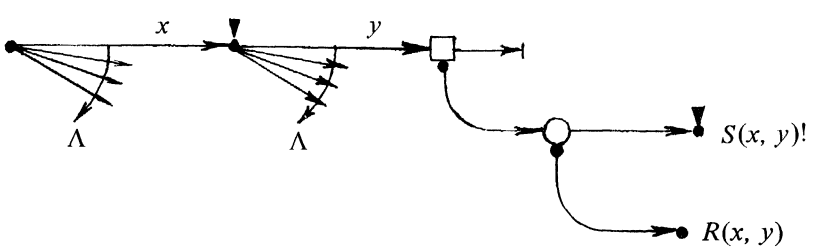

FIGURE 2. The semantic graph of the proposition
$(\mathrm{A}x)(\mathrm{E}y) \sim (R(x, y) \to S(x, y)!)$

The P-object corresponding to the proposition of Figure 2 is:

$$\underline{\text{all}}(x \in \Lambda\colon \underline{\text{sch}}(y \in \Lambda\colon \bar{\gamma}(\underline{\text{if}}\ \gamma(R(x,y))!\ \underline{\text{then}}\ S(x,y)!))!).$$

This expression (to be referred to as $\langle 1\rangle$) can be rewritten as

$$\underline{\text{all}}(x \in \Lambda\colon \langle 2\rangle!).$$

By the definition of the all machine, $\langle 1\rangle$ runs Λ, which produces the "typical" object x, then outputs $\langle 2\rangle!$ which uses x. We have the semantic graph:

$$\langle 1\rangle;\ \Lambda \to x;\ \to \langle 2\rangle!.$$

In Figure 2 this is represented by the first "fan" produced by Λ. The evolution of $\langle 2\rangle$ is given by

$$\langle 2\rangle;\ \underline{\text{sch}}(y \in \Lambda\colon \langle 3\rangle);\ \underline{\text{sch}}(\Lambda \to y;\ \langle 3\rangle).$$

Note the difference between all and sch. The former is a generator. It simply produces an object, and in the further evolution of this object there will be no call of the original all. sch is a parallel search. This function runs Λ, which produces y, and then initiates and runs the searches $\langle 3\rangle$ with each y, but retains control till the very moment (if ever) when one of the searches ends, after which it terminates the whole process. Thus, the call of sch stays till the end of the process. In Figure 2 this is represented by the second fan, which is a prediction node.

Now the remaining part of the graph:

$$\langle 3\rangle;\ \bar{\gamma}[\langle 4\rangle],$$
$$\langle 4\rangle;\ \underline{\text{if}}\ \gamma[R(x,y)]!\ \underline{\text{then}}\ S(x,y)!;\ \gamma[\langle 5\rangle];\ \to \langle 6\rangle!,$$
$$\langle 5\rangle;\ R(x,y);\ldots,$$
$$\langle 6\rangle;\ S(x,y);\ldots.$$

We used here the definition of the if machine, which runs the first argument, and when this process terminates, produces the second argument. In Figure 2 we see two nodes marked with a metasystem transition from a cognitive function to its argument.

The situation when a Refal expression E (possibly depending on some parameters) calls itself (possibly with different parameters) inside a cognitive function call, i.e. its semantic graph is of the form $E;\ \gamma[\cdots E\cdots]$ or the same with $\bar{\gamma}$, is referred to as *semantic recursion*. It should be distinguished from the usual, *computational*, recursion, where E simply calls itself in the process of computation: $E;\cdots E\cdots$.

A metamechanical process is not deterministic. It is not defined in advance for all future times, but only for the past, up to the present moment. Its further development depends on the decisions to be taken by the user. Moreover, if we want a theory that would be applicable to different users, then even the past of a metamechanical process is not really defined. Is there anything definite about metamechanical processes, or does the freedom of the user's will render every prediction about such processes meaningless?

It would do so if we put no constraints on the user's choices and decisions. But if we assume that the user is ideal, i.e. complies with the Consistency and Completeness

Rules, then for some processes we can define properties which will be *objective*, independent from the user as long as he complies with the rules. We call such processes *objectively interpretable*. They may be compared with physical quantities which are invariant with respect to the user's choice of a reference system. For other, *uninterpretable*, processes we cannot think of any property which would not depend on the user's current knowledge.

Before giving a formal definition, a brief preview. The concept of interpretability is applicable to all types of SP-objects. The interpretability of a search means that it is an objective fact whether the search is finite or infinite. The interpretability of a generator means that it is an objective fact whether it produces any given expression, or not. The interpretability of a proposition means that it can be objectively characterized as true or false. For SP-objects which are not interpretable, all these properties have no meaning, because, say, a search may terminate with one user, and be infinite with another.

We define objective interpretability inductively. The base of induction includes only such processes that do not call cognitive functions at all, and are deterministic (mechanical processes). If P is a proposition which involves only mechanical processes, then $\gamma(P)$ becomes objectibely interpretable: it must be finite for true and only for true P. Proceeding in this manner, we construct a whole hierarchy of interpretable objects.

We define interpretability as a property of semantic graphs. Our definition is based on the usual mathematical intuition of constructive potentially infinite processes, according to which every process—and when it splits into parallel processes, as is the case with our graphs, then every branch of it—is either finite or infinite, even though at the moment we may not be able to determine which of the two takes place for a given process. We present classification of semantic graphs and their parts as *labeling* of the nodes and branches constituting a graph. This does not imply that there is a decision procedure, an algorithm, which allows us to label every node in the graph. The labeling rules are definitions, not steps of an algorithm. This is only a convenient way to represent a series of intertwined inductive definitions. "Labeled T, F, finite, infinite" are the properties being defined.

We first consider the case when the root SP-object in the graph is fixed. The semantic graph of such an object is completely and uniquely defined.

LR1. A call $\gamma(P)$ with P labeled T is labeled finite.

LR2. A call $\gamma(P)$ with P labeled F is labeled infinite.

LR3. A call $\bar{\gamma}(P)$ with P labeled T is labeled infinite.

LR4. A call $\bar{\gamma}(P)$ with P labeled F is labeled finite.

LR5. If every branch starting from a P-generator either leads to a P-object labeled T, or is labeled infinite, this P-generator is labeled T.

LR6. If at least one branch starting from a P-generator leads to a P-object labeled F, then this P-generator is labeled F.

LR7. If at least one search starting from a prediction node is labeled finite, then this prediction is labeled T.

LR8. If every branch starting from a prediction node is labeled infinite, then this prediction is labeled F.

LR9. A branch is labeled finite if it ends, and all $\gamma/\bar{\gamma}$-calls on it are labeled finite.

LR10. A branch is labeled infinite if one of the following cases takes place:
(a) the branch has at least one $\gamma/\bar{\gamma}$-call which is labeled infinite;
(b) all the $\gamma/\bar{\gamma}$-calls on the branch are labeled finite, but the branch itself is infinite.
The following rule tells us when nodes and branches are uninterpretable.

ER. *Exclusion Rule.* A P-object which cannot be labeled T or F by applying the above rules is labeled U (uninterpretable). So is a branch which cannot be labeled finite or infinite. Thus every branch of an S-object is either labeled finite, labeled infinite, or uninterpretable.

As an example of an uninterpretable proposition we can take the famous Liar paradox: "The proposition expressed by this very sentence is false." Let us formalize it in our metatheory. We have here a proposition that states its own falseness, and nothing above that. Since it refers to itself, we should start with giving it some name, otherwise reference will be impossible. Let it be P^L. The statement that P^L is false is $\bar{\gamma}(P^L)!$. Then the proposition P^L which states this is defined by $P^L = \bar{\gamma}(P^L)!$.

The semantic graph for P^L is shown in Figure 3. It is an example of infinite semantic recursion. The proposition is clearly uninterpretable. In terms of our theory, it is not "paradoxical", but just devoid of any objective meaning.

We call our definitions inductive, and not recursive, because they are intended for the use "from the bottom up": the user can construct more and more levels of the hierarchy of interpretable objects and propositions. But it is impossible to give a single recursive definition of interpretability which would satisfy the requirement of meaningfulness in our theory, i.e. would itself be interpretable:

THEOREM 1. *There exists no parametrized process* $\underline{\text{int}}(P)$ *such that* $\underline{\text{int}}(P)$ *is objectively interpretable for every P, and is finite iff P is objectively interpretable.*

PROOF. Suppose that there is such a process $\underline{\text{int}}(P)$. Define the process A by

$$A = \underline{\text{if}}\ \underline{\text{int}}(A!)!\ \underline{\text{then}}\ \bar{\gamma}(A!).$$

Since $\underline{\text{int}}(A!)$ is interpretable, it is either finite, or infinite. Suppose it is finite. Then $A!$ must be interpretable. But if $\underline{\text{int}}(A!)$ is finite, A calls $\bar{\gamma}(A!)$, so it is uninterpretable. Suppose $\underline{\text{int}}(P)$ is infinite. Then $A!$ must be uninterpretable. But in this case A never

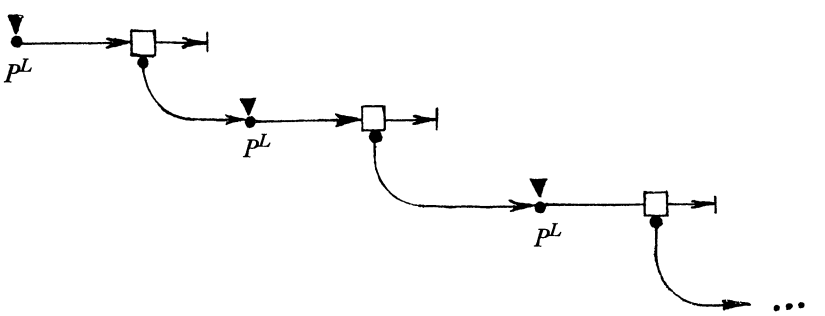

FIGURE 3. The Liar paradox

calls anything but $\underline{\text{int}}(A!)$, which is interpretable. So, A must be interpretable. The inescapability of a contradiction proves the theorem.

THEOREM 2. *There exists no objectively interpretable generator which produces all objectively interpretable expressions, and only them.*

PROOF. If there were such a generator G, we could immediately construct $\underline{\text{int}}(P)$, whose existence is denied by Theorem 1. One has only to run G and stop if and when it produces P.

Thus, the concept of interpretability cannot be formalized in our metatheory as an interpretable concept. This, of course, reminds one of Tarski [1933], even though our approaches are very different philosophically.

Intuitively, it is clear why we cannot objectively define $\underline{\text{int}}(P)$, or other universal functions like those establishing the truth or the falsity of a proposition. Such a function runs its argument A and wants to prove something about its own application to some expressions A' resulting from running A. To prove means to call a cognitive function. So we have a definition of the type:

$$\underline{\text{int}}(A) = \gamma(\cdots \underline{\text{int}}(A')\cdots) \tag{1}$$

or the analogous scheme with $\bar{\gamma}$. Since A will be, typically, recursive, A' may include A. Thus we have a semantic recursion which may be infinite, so we cannot guarantee the interpretability.

However, we can construct a hierarchy of interpretability concepts in our real time as Refal-machine users. We define 0-interpretability $\underline{\text{int}}^0$ using the process of checking that cognitive functions are never called. Then we define 1-interpretability $\underline{\text{int}}^1$ by modifying (1) so that in the left side we have the new concept, while the right side uses only the concepts already defined:

$$\underline{\text{int}}^1(A) = \gamma(\cdots \underline{\text{int}}^0(A')\cdots). \tag{2}$$

In this way we can define an infinite series of concepts: $\underline{\text{int}}^2$, $\underline{\text{int}}^3$, etc. All these concepts (parametrized processes) are interpretable.

This is not the end, of course. If a process is n-interpretable, the semantic length of the branches of its semantic graph never exceeds n. We can define the concept of ω-interpretability, which allows the branches on the semantic graph to be of arbitrary finite length, then $(\omega + 1)$-interpretability using ω-interpretability in the right side of (2), etc. (This is, essentially, how the ordinals of set theory emerge in our theory.)

Objectively interpretable propositions constitute our universe of discourse. Every expression inside this universe is meaningful. No one outside is. If there were an expression U which defined this universe by generation or recognition, then the subuniverse Λ_i that has been currently constructed, i.e. the generator of expressions known to be interpretable, could be simply obtained by replacing Γ in U by the current knowledge Γ_i. Our current universe of discourse would be a definite function of our current knowledge. But this is not the case (Theorems 1 and 2). We can expand the available domain of discourse by creating new levels in the never-to-be-fully-covered universe, even without expanding the current knowledge Γ. Thus the construction of the universe of discourse is a real-time process which, like the knowledge process, cannot be expressed by a fixed object. The access function Λ

represents this process. Its value is, at any moment in real time, an interpretable generator which produces all the expressions proved interpretable up to the current moment.

So, we have two real-time processes in mathematics: the sum-total of our knowledge, the "gnosis" Γ, and the sum-total of the expressions known to be meaningful, our universe of discourse, the "logos" Λ. They are interrelated: Γ is always a subset of Λ; on the other hand, our ability to prove interpretability depends on our current knowledge. But neither of them is a simple derivative of the other. We have two degrees of freedom here, not one.

In the classical set theory, as created by Cantor [1895–97] and Zermelo [1908], the construction of the universe of discourse is based on the concept of *regularity*.

Definition of regularity. A *production sequence* is a sequence of generators S_1, S_2, etc., such that for every $i > 1$, S_{i-1} produces S_i. An S-object is *regular* if it is interpretable and a production sequence starting with it may only be finite.

The universe of discourse in set theory consists of regular sets treated as objects. From now on we shall use Λ as the access function for this universe of objects. The universe of interpretable propositions can be defined in terms of Λ.

In the general case, an SP-object may not be fixed, i.e. some of the nodes of its semantic graph may include active references to Λ. The following rules extend the definitions listed above. By a *regular implementation* we mean a lambda-implementation with a regular Λ_i.

LR11. If a P-generator is labelled T in every regular implementation (of the Refal expression representing P-node), then this P-generator is labeled T.

LR12. If there exists a regular implementation in which a P-generator is labeled F, this P-generator is labelled F.

LR13. If there exists a regular implementation in which a prediction node is labelled T, this prediction node is labelled T.

LR14. If a prediction node is labelled F in every regular implementation, this prediction node is labelled F.

This concludes our classification of SP-objects. The definitions we gave are intertwined. But using them inductively we can construct more and more complex classes of interpretable objects. In the proofs of interpretability, whenever we apply LR11 and LR14 (universally quantified propositions), the only thing about Λ_i we use is its regularity. Applying LR12 and LR13, we actually construct the required Λ_i.

Objective interpretation is based on our intuition of the separability of the object and the subject of knowledge. When we deal with quantum-mechanical phenomena this intuition deceives us. The object and the subject of knowledge are not completely separable in the quantum-mechanical measurement. Our functions γ and $\bar{\gamma}$ can be seen as measurement procedures, of a kind. We took pains to separate the results of these "measurements", i.e. truth values of propositions, from our state of knowledge. Our theory allows objectively interpretable propositions only; this leads to the usual two-valued logic. It is possible that a more general theory can be built, which would not limit itself to those propositions we call interpretable, thus overstepping the boundaries of traditional logic. This possibility occurred to the author under the influence of the ideas of *the wave logic* developed by Yuri Orlov

[1978], [1982]. Orlov's ideas can probably be used in trying to expand the present theory aiming at description of subatomic phenomena.

§3. Comparable work. Some of the concepts of the Cybernetic Foundation bear resemblance to ideas known from the literature on logic and foundations of mathematics. Those the author is aware of are reviewed in this section.

First of all, the function $\gamma(P)$ will be immediately associated by the reader with the concept of an *oracle* widely used in recursive theory (see, e.g., Rogers [1967]). This association, however, is purely formal. It is correct only in the trivial sense that $\gamma(P)!$ is our formalization of the oracle telling that P is true, just as, say, $\underline{\text{or}}(\gamma(P), \gamma(Q))!$ is our formalization of the disjunction of P and Q. Otherwise, our concept is very different. When oracles are used in recursive theory, this adds nothing to the concept of truth or other aspects of the foundations of mathematics. An oracle only *tells* the truth, which is already there, somehow. Should the concept of oracle be formalized, it is expressed in terms of set theory.

Our aim, on the contrary, is the foundations. For us, $\gamma(P)$ is a *process* defined in terms of our cybernetical semantics and the concept of the user of mathematical machinery. We do not leave our "oracles" γ and $\bar{\gamma}$ undefined, nor do we define them in terms of set theory. On the contrary, we define set theory using the concept of cognitive functions.

Our concept of user, in its turn, recalls the "creating subject" used by Brouwer in his late papers [1949], [1952] (see also the discussion in FBL [1967] and Martino [1982]); this idea was further developed by Kreisel [1967]. The creating subject can be identified with what we called above "the real user": a person who actually creates mathematical objects, and proves new facts about them. Kreisel sees the creating subject as proceeding by stages and uses $\vdash_n P$ to denote "the creating subject has at stage n evidence for the proposition P". This is clearly analagous to the proposition $\gamma_n(P)!$ of our theory, where the function γ_n results from replacing Γ in γ by the nth stage of knowledge Γ_n. Kreisel formulates the following principles:

(i) $\vdash_n P \vee \sim \vdash_n P$.
(ii) $(\vdash_n P) \to P$.
(iii) $(\vdash_n P \;\&\; m > n) \to \vdash_m P$.
(iv) $P \to \sim\sim(\mathrm{E}n) \vdash_n P$.

A closely related principle is known as *Kripke's schema*:

$$(\mathrm{E}\alpha)[\{(\mathrm{A}x)(\alpha x = 0) \equiv \sim P\} \;\&\; \{(\mathrm{E}x)(\alpha x \neq 0) \to P\}]$$

where the function α for a given P is defined by

$$\alpha n = \begin{cases} 0 & \text{if } \sim \vdash_n P, \\ 1 & \text{if } \vdash_n P. \end{cases}$$

The crucial difference between these ideas and our theory is the difference in the concept of a mathematical object. For Brouwer, mathematical objects are found in human intuition. This approach is, therefore, often characterized as subjective, and even solipsist (see FBL [1967]). To quote Brouwer himself, "...the first act of intuitionism completely separates mathematics from mathematical language, in particular from the phenomena of language which are described by theoretical logic,

and recognizes that intuitionist mathematics is an essentially languageless activity of the mind ...." Our view, of course, is exactly the opposite: that all of mathematics, as well as logic, is essentially a linguistic activity. At the same time, we recognize, like Brouwer and unlike the formalists of the Hilbert school, the necessity of the concept of the subject of knowledge, or the user. This contradiction is resolved, in the Hegelian spirit, by making a synthesis: the concepts of abstract user and objective interpretability.

As a result, our approach is no less objective than the theory of relativity, which uses the concept of observer, closely analogous to our abstract user. For us a mathematical object is a machine, and the abstract user is only a way to describe the machine; it has nothing to do with human beings. We do not expel the subject of knowledge completely, because we admit that proof, as different from verification, is still based, in the last analysis, on intuition, and thus may be called belief. But our cybernetic semantics allows us to limit the subjective element severely by expelling it from the fundamental definitions. These include the criterion of truth and the definition of objective interpretability, from which follow all the formal differences between our theory and previous approaches.

Some of the formal differences can already be seen in the principles above. In (iv) the intuitionistic double negation is salient. But we can interpret and safely use classical logic. Another difference: Kreisel's principles use only the analogue of γ_n, not of $\bar{\gamma}_n$. There is not, and, apparently, there cannot be any analogue of $\bar{\gamma}(P)$! other than $\vdash_n \sim P$. This applies also to Kripke's schema, and follows from the syntactic character of formal logic, where connectives are no more than functions of truth-values, and truth-values are not interpreted at all. Our theory is semantic; logical connectives are machines. $\bar{\gamma}$ is the machine for negation. We simply could not do with γ alone.

There is a remarkable formal convergence between our objective interpretation and the work of S. Kripke [1975]. Kripke continues the line of research started by Tarski [1933], and generalizes his results. While Tarski's hierarchy of languages with formal definitions of truth includes only finite levels, Kripke includes transfinite levels. The most important difference is that Kripke uses a three-valued logic, first considered by Kleene [1952], instead of the usual classical logic in Tarski [1933]. These two factors allow him to construct a formal system containing its own truth predicate, which is impossible in the Tarski setting. Given a domain D, a predicate $P(x)$ is interpreted by a pair (S_1, S_2) of disjoint subsets of D; S_1 is the extension of $P(x)$, and S_2 its anti-extension (where $P(x)$ is false). In the gap between S_1 and S_2 the truth-value is "undefined". The ground-level system is a language L with a countable set of primitive predicates. Adding a truth predicate is a transformation of the system. The fixed point of this transformation is the language that contains its own truth predicate.

The valuation rules we use in labeling semantic graphs can also be described in terms of Kleene's three-valued logic. Technically, the way we label graphs is the same as computing the fixed point by iterations in Kripke's theory. Objects we define as interpretable correspond to those belonging to S_1 or S_2 in the least fixed point of Kripke's. In particular, Kripke's solution of the Liar paradox looks very much like ours.

With all that, Kripke's paper stays completely within set theory, and no attempt is made to use the formalism for solving foundational problems. Kripke's paper, like Tarski's and many other works in formal logic, shows what definitions of truth *could* exist, but says nothing about what mathematical objects and truths really are—which is, exactly, the endeavour of our theory. Even formally, though, our theory goes beyond Kripke's paper, where S_1 and S_2 are limited to *sets*, whereas in our theory the collections of all true and false propositions are not sets, but metatheoretical concepts.

To sum up, the similarities between our work and the work discussed in this section are mostly formal or superficial. We do not find in previous work the central element of this theory: the cybernetic concept of a mathematical proposition. Starting from this point one can logically derive all other features of the present theory, and this is how the Cybernetic Foundation has actually come into being. Incidentally, at the time when this theory was developed, the author was aware neither of the Brouwer-Kreisel concept of creating subject, nor of Kripke's work.

§4. Extensionality and regularity. We proceed to the interpretation of set theory. A set of ZF is interpreted as a regular S-object. We identify the concept of a set with the concept of a generator. Since we allow the use of real-time cognitive processes γ and $\bar{\gamma}$, we must limit set generators to *interpretable* processes—otherwise we shall not be able to interpret the membership of an object in a set. Regular sets (S-objects) are interpretable by definition. A proposition of ZF is interpreted as a P-object. Our main concern now will be to make sure that those P-objects are interpretable.

When an object and a set are given, we must be able to establish whether the object is an element of the set. Let the search $\underline{\text{el}}(x \in S)$ be finite iff x is an element of S. (The symbol "$\in$" is just a separator, as well as ":", etc., below). It would be easy to define the function $\underline{\text{el}}$ so that this search stops if and only if the expression x is among the expressions generated by S:

$$\text{(NONEXT)} \qquad \underline{\text{el}}(x \in S) = \underline{\text{sch}}(y \in S: \underline{\text{ident}}(y, x))$$

where $\underline{\text{ident}}$ simply checks the identity of the two expressions which are its arguments.

However, this staightforward concept of membership is not the one adopted in set theory. Set theory uses *the extensionality principle*, according to which two sets are identical, or equal, if and only if every element of one set is also an element of the other. Consequently, the equality of sets is not the same as the identity of the Refal expressions which represent them. Indeed, it is easy to define in Refal two different processes which will generate the same objects.

Thus we define membership by

$$\text{(EXT)} \qquad \underline{\text{el}}(T \in S) = \underline{\text{sch}}(y \in S: \gamma[y = T])$$

where the equality "$=$" is understood as the double inclusion:

$$(S = T) \equiv ([S \subseteq T] \,\&\, [T \subseteq S]).$$

The relation of inclusion (being a subset) is defined by

$$(S \subseteq T) \equiv \underline{\text{all}}(x \in S: \underline{\text{el}}(x \in T)!).$$

According to these definitions, the semantic graph for $\underline{\text{el}}(T \in S)!$ is:

$$\langle 1\rangle;\ \underline{\text{el}}(T \in S)!;\ \underline{\text{sch}}(S \to y;\ \gamma[\langle 2\rangle])!,$$
$$\langle 2\rangle;\ y = T;\ \langle 3\rangle + \langle 4\rangle,$$
$$\langle 3\rangle;\ y \subseteq T;\ y \to z;\ \underline{\text{el}}(z \in T)!,$$
$$\langle 4\rangle;\ T \subseteq y;\ T \to z;\ \underline{\text{el}}(z \in y)!$$

(see Figure 4). Thus $\underline{\text{el}}$ calls itself in $\langle 3\rangle$ and $\langle 4\rangle$. This recursion is *semantic*, because the path runs through $\langle 2\rangle$ inside the γ-function. This may cause uninterpretability. Indeed, we immediately are in trouble if S is among the elements of itself: $S \to S$. Take the case where there are no other elements in S except S. Let $T = S$ in $\langle 1\rangle$. Then for y and z there is only one value: S. We see that $\underline{\text{el}}(S \in S)$ semantically depends on itself, and is, therefore, uninterpretable. In the same way, uninterpretability results if $S \to y_1$ and $y_1 \to S$.

To guarantee interpretability, we use only *regular* sets.

THEOREM 3. *The process* $\underline{\text{el}}(T \in S)$, *where* T *and* S *are regular S-objects, is interpretable.*

PROOF. As we have seen, the semantic graph for $\underline{\text{el}}(T \in S)$ results from the semantic recursion over T and S that can be represented schematically by the formula

$$(T, S) \to (T', S') + (S'', T)$$

where (T, S) is the pair representing an argument of $\underline{\text{el}}$, and x' is some element of x. If T and S are regular, then they may start only finite production sequences, and any possible sequence of calls of $\underline{\text{el}}$ in the graph can only be finite. Therefore, all of them have a definite objective interpretation, which can be established, starting from the bottom, by labeling.

From now on, by an S-object we shall mean a *regular* S-object, if the opposite is not stated.

We do not exclude using the real-time process Λ in the interpretation of set-theoretical constructs; this calls for a more careful look into regularity. By definition, Λ produces all those sets that the user proved to be regular: this is the Completeness Rule with respect to Λ. Suppose that at a certain stage $\Lambda = \Lambda_i$ the user considers a new set S, not in Λ_i. If it is regular, it must be included in Λ. Therefore, to prove that it

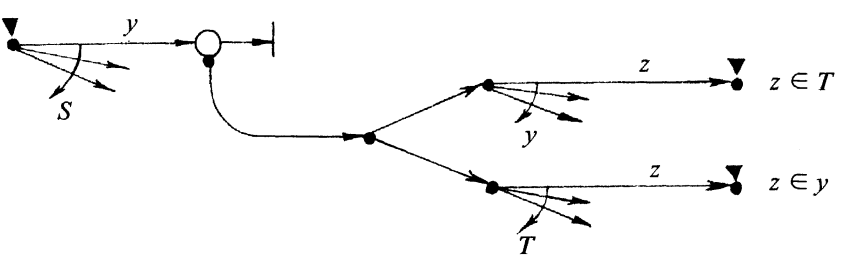

FIGURE 4. The semantic graph of $\underline{\text{el}}(T \in S)$

is regular, the user must prove that no infinite production sequences start from S *in the assumption that* Λ *produces* S.

Therefore, if S hierarchically produces Λ, it is possible that S is regular in every regular interpretation, as shown in Figure 5(a), where S is considered in isolation, but is still *not regular*, because of the infinite production sequence $S \to \cdots \to \Lambda \to S \to \cdots$, as shown in Figure 5(b). In particular, Λ itself is not regular, even though it is, obviously, regular in every regular implementation.

This may seem paradoxical, because at every moment in real time Λ produces only regular sets. It even may seem inconsistent. Indeed, is not Λ the union of all Λ_i? A union of regular sets is regular, hence Λ is regular.

This argument, however, is not valid, because it is based on a naive Platonist idea that our symbols are only *denotations* of some real (even if "abstract", or "ideal") things which exist independently of whether we denote them in our theories, or not.

In contrast, our theory deals with real linguistic objects and processes. The symbol Λ does not stand for any ideal entity. It stands for itself. Its meaning is in the rules by which it is manipulated by the user. As we have seen, these rules are such that Λ is not regular, even though it produces only regular sets.

THEOREM 4. *If an S-object is regular, all its elements are regular. If all elements of an interpretable S-object S are regular, S is also regular.*

PROOF. Should an element T of a regular set S be not regular, an infinite production sequence starting with T would exist. Then we have only to add S to it to prove that S is not regular either. To prove the second part, we notice that if $T \in S$, and T is regular, its regularity is established in the assumption that Λ produces T. Suppose that there is an infinite production sequence starting with $S \to T \to \cdots$. If this sequence does not include Λ, then we immediately have an infinite production sequence starting with T, which is impossible. If it includes Λ, then, again, there is an infinite sequence, because Λ produces T.

So, for a set to be regular, it is sufficient and necessary that all its elements are regular. Therefore, all regular sets can be constructed inductively starting with the empty set.

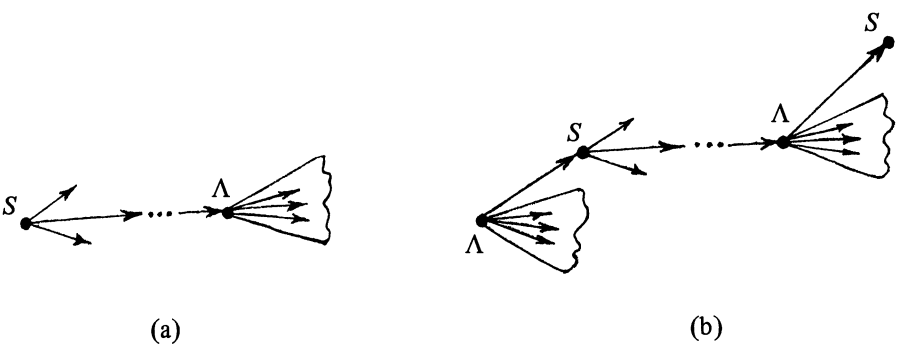

FIGURE 5. Including regular sets in Λ

It should be stressed that regularity becomes necessary only because the function el is defined according to the extensionality principle (EXT). The concept of a set which has itself as one of its elements is not contradictory as such. For instance, it is easy to define a generator that produces only itself. It will be interpretable, and, if we accept the definition (NONEXT) of membership, its only element will be itself.

§5. Basic constructors. The language of set theory is, in our interpretation, a *programming language*. Like other programming languages, such as FORTRAN or LISP, the set-theoretical language gives us the means to create linguistic processes (set generators in the case of set theory) that we use to model natural phenomena. Unlike computer languages, the language of set theory includes the means to communicate with the real-time processes Λ and Γ.

The role of basic operations of computer languages is played in set theory by *set constructors*. These are machines defined in the Refal metasystem and used to create new set generators. A set constructor must be such that when its arguments satisfy certain stated requirements, the generating process is interpretable and the set produced is regular. In the following we define the basic set constructors necessary to arrive at the present-time set theory. We define them one after another: this is the process of constructing the set theory itself. At every stage we have a certain subuniverse Λ_i, originally empty. Then we define a new constructor, and we prove that the new subuniverse Λ_{i+1}, which is a regular Λ_i extended by adding all possible S-objects that can be constructed with the use of the new constructor, is also regular.

We want, first of all, to form finite sets of objects which are already in existence. Since a finite set can be represented simply by the list of its elements, we define the trivial function fs (for "finite set") which takes a list as its argument and produces its members. Then the empty set is fs( ); the pair $\{a, b\}$ is fs(a, b).

It is easy to define a function $\bigcup(a)$ which interprets the union-set of a. It simply runs a, and whenever an element x is produced, it runs x as a generator in parallel with all other elements of a.

Set theory requires the existence of at least one infinite set, namely, the set which contains the empty set $\varnothing$ as its element, and together with any element x contains also the element formed as the union $x \cup \langle x \rangle$. Thus the elements of this set are:

$$\varnothing, \langle\varnothing\rangle, \langle\varnothing, \langle\varnothing\rangle\rangle, \langle\varnothing, \langle\varnothing\rangle, \langle\varnothing, \langle\varnothing\rangle\rangle\rangle, \ldots. \tag{1}$$

To construct a generator producing (1), we define the constructor inf(X), which in every step produces X and calls itself with $X \cup \{X\}$ as the next argument. Then inf(fs( )) is the desired infinite set. The following is obviously true:

THEOREM 5. fs( ) *and* inf(fs( )) *are regular. If* a *and* b *are regular then* fs(a, b) *and* $\bigcup(a)$ *are regular.*

Our next constructor will produce sets with elements selected for a certain property. Its format is

$$\text{set}(x \in S\colon H(x)!) \tag{2}$$

which is read: the set of all those elements x of the set S for which the search H depending on x as a parameter is finite. The search $H(x)$ will mostly be the proving of

a certain property P of x, or its negation, i.e. $\gamma(P)$ or $\bar{\gamma}(P)$. The $\underline{\text{set}}$ machine works as follows. G is run step by step. Each time that it produces an object x, this object is substituted into the search H and the search is run in parallel with the continuing running of G. Those and only those branches for which $H(x)$ stops produce the corresponding object x.

THEOREM 6. *If S is regular, and $P(x)$ is an interpretable P-object for every regular x, then $T = \underline{\text{set}}(x \in S: \gamma[P(x)]!)$ is regular.*

PROOF. The interpretability of T is guaranteed by the regularity of S and the interpretability of the selection process. The regularity of T follows from Theorem 4.

Can we use Λ in the role of S in the $\underline{\text{set}}$ constructor? Since Λ is not, as we have seen, regular, Theorem 6 does not legitimize such a construct; it must be considered for each $H(x)$ separately. That it is not always regular can be seen from Russell's paradox.

THEOREM 7. *The set $R = \underline{\text{set}}(x \in \Lambda: \bar{\gamma}[\underline{\text{el}}(x \in x)!]!)$ of the Russell paradox is uninterpretable.*

PROOF. To be regular, R must be regular in every regular implementation. Take some regular subuniverse Λ_i and add R to it. Consider the resulting semantic graph of R:

$$\langle 1\rangle;\ R;\ (\Lambda_i \to x;\ \bar{\gamma}[\underline{\text{el}}(x \in x)!];\ \to x + \bar{\gamma}[\underline{\text{el}}(R \in R)!];\ \to R).$$

Since x in the first branch is an element of Λ_i, it is regular, and $\text{el}(x \in x)!$ is false; thus the first branch is simply $\Lambda_i \to x$. The trouble comes from the proposition in the second branch:

$$\langle 2\rangle;\ \underline{\text{el}}(R \in R);\ \underline{\text{sch}}(R \to y; y = R);\ \underline{\text{sch}}(\Lambda_i \to y; y = R + \bar{\gamma}[\underline{\text{el}}(R \in R)!]; R = R).$$

We see that $\langle 2\rangle$ calls itself within $\bar{\gamma}$: an infinite semantic recursion. However, we have still to establish that the first group of branches in $\langle 2\rangle$ does not lead to a unique labeling. By the definition of equality, it is

$$\langle 3\rangle;\ \underline{\text{sch}}(\Lambda_i \to y;\ \gamma[\underline{\text{and}}(y \subseteq R, R \subseteq y)]).$$

If there existed a y for which this search is finite, $\langle 2\rangle$ would be labeled T, the infinite semantic recursion notwithstanding. This is not so, however. R cannot be a subset of y, because $\Lambda_i \subseteq R$; hence Λ_i must be a subset of y, but this is impossible because $\Lambda_i \to y$, and y must be an element of itself. Therefore, the infinite semantic recursion in $\langle 2\rangle$ makes it uninterpretable.

We saw that the $\underline{\text{set}}$ constructor with the universal generator Λ cannot be used to collectivize objects by an arbitrary property. However, if we specify the collectivizing property in a certain way, namely by putting $P(x) = x \subseteq S$ where S is a definite regular set, then we still can form a universal set. This set, i.e. the set of all subsets of S, known as the *powerset* of S, plays a most important role in Cantor's set theory. It deserves a special constructor:

$$\underline{\text{pow}}(S) = \underline{\text{set}}(x \in \Lambda: \gamma(x \subseteq S)!).$$

We must prove now that the subuniverse Λ_{i+1} which results from adding $\underline{\text{pow}}(S)$ is still regular if Λ_i was regular. Take some set S from Λ_i and add $\underline{\text{pow}}(S)$ to the

subuniverse Λ_i. The semantic graph of $\underline{\text{pow}}(S)$ is presented in Figure 6. It includes a semantically infinite path, but this does not prevent us from labeling T or F all the propositions involved. x_1, x_2, and other elements of $\underline{\text{pow}}(S)$ may or may not be elements of S, but S itself certainly is not an element of S, being regular. Thus $\underline{\text{pow}}(S) \subseteq S$ is interpretable and labeled F; $\underline{\text{pow}}(S)$ is not produced by $\underline{\text{pow}}(S)$, while all other x's produced by it are regular because they have been in Λ before the introduction of $\underline{\text{pow}}(S)$. This proves the regularity of $\underline{\text{pow}}(S)$ for any regular S.

However, this does not yet prove that the whole new subuniverse allowing the use of pow is regular (this observation was made by Karel Hrbáček, and I greatly appreciate it). Indeed, a new construct a may not produce infinite production sequences when it alone is added to Λ but may produce such a sequence when some construct $\zeta(a)$ is added to Λ, even though $\zeta(x)$ for any regular x is regular. For instance, if a is the set of all singlets, a does not produce itself because a is definitely not a singlet. But when $\underline{\text{fs}}(a)$ is added to Λ, a becomes nonregular because it produces $\underline{\text{fs}}(a)$, which in turn produces a. Thus we must prove that when any construct $\zeta(a)$ combined from the constructors introduced earlier is added to Λ, the new S-object $a = \underline{\text{pow}}(S)$ is still a regular set. The regularity of the graph in Figure 6 is only a special case of this general statement when $\zeta(a) = a$.

The idea of our proof for the general case of $\zeta(a)$ is this. We divide all constructs ζ into two categories: those for which $\zeta(a)$ is "equal to or higher than" a on the set-theoretical staircase, i.e. such that it produces a, or produces a set that produces a, etc.; and those which only use a as a generator, i.e. deal only with the elements of a, but not with a itself. For the constructors of the first category we shall establish, like we did in the case $\zeta(a) = a$, that they definitely are not subsets of S. As for the second category, it is intuitively clear that they should not create problems, because the elements of a are all "in the clear", and the property of being an element of a is equivalent to being a subset of S: a predicate which has no reference to a.

THEOREM 8. *If $\zeta(x)$ is a construct which is regular for every regular x, then adding $\zeta(a)$ to a subuniverse Λ_i leaves it regular. This means, of course, that $\zeta(a)$ is also regular.*

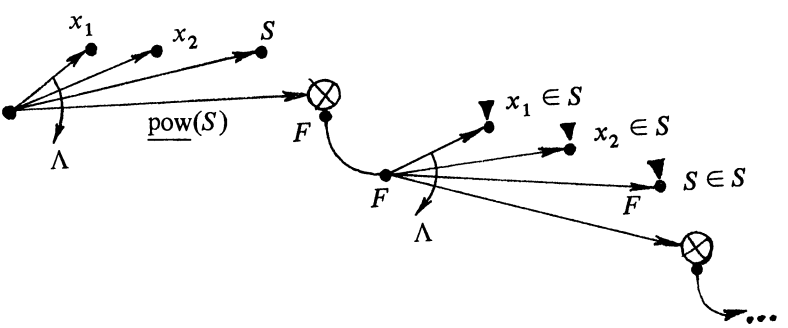

FIGURE 6. The semantic graph for $\underline{\text{pow}}(S)$

PROOF. We argue by structural induction on the number of the occurrences of a in $\zeta(a)$. Pick an occurrence of a. Our constructors are such that we can distinguish between running a and producing a (both may be conditional). The proposition P^h that $\zeta(a)$ hierarchically produces a is interpretable. Indeed, if our occurrence of a is running, then P^h is false. If the occurrence is producing, then whether a is actually produced is determined by the remaining parts of the structure of $\zeta(a)$, which, by the inductive assumption, call only regular sets, and are, therefore, interpretable.

Since P^h is interpretable, it is either true or false.

LEMMA 1. *If P^h is true, then $\zeta(a) \subseteq S$ (to be referred to as P^s) is interpretable and false.*

PROOF. By the condition of the lemma, there exists a sequence $x_1, \ldots, x_n$ such that

$$\zeta(a) \to x_1 \to \cdots \to x_{n-1} \to x_n, \quad \text{where } x_n = a$$

(the case when $n = 0$, i.e. $\zeta(a) = a$, was considered before). Consider the semantic graph for $\zeta(a) \subseteq S$:

$$\zeta(a) \to x; \underline{\text{el}}(x \in S)!.$$

By the definition of the predicates of membership $\underline{\text{el}}$ and equality, this goes on as

$$S \to y_1; x = y_1; (x \subseteq y_1 + y_1 \subseteq x).$$

Here x is universally quantified.

To prove that P^s is false we substitute x_1 for x, and show that $x_1 \subseteq y_1$ is false for every possible y_1 from S. If $n = 1$, then $x_1 = a$. But $a \subseteq y_1$ is false. Indeed, S is an element of a, hence if $a \subseteq y_1$, S must also be an element of y_1. But $S \to y_1$, so we have $S \to y_1 \to S$, which is impossible because of the regularity of S. If $n > 1$, we notice that refuting $\zeta(a) \subseteq S$, has been reduced by the preceding to refuting $x_1 \subseteq y_1$, where $\zeta(a) \to x_1$, and $S \to y_1$. We have for x_1 a production sequence ending with a which is shorter by one than the sequence for $\zeta(a)$; and y_1 is, like S, regular. By repeating this procedure n times we prove that P^s is false.

Note that in this proof we use universal quantification (the variable y_1 in the negative context) over S only, which is safe because S is known to be regular. The existential variables x_i exist by the premise of the lemma.

As an example of a construct satisfying the conditions of Lemma 1, the reader can analyze the semantic graph for $\underline{\text{fs}}(a)$.

LEMMA 2. *If P^h is false, i.e. $\zeta(a)$ may only run a, but never hierarchically produces it, then $\zeta(a)$ is regular, and the truth-value of $\zeta(a) \subseteq S$ after adding $\zeta(a)$ to the current subuniverse Λ_i is the same as before, i.e. in Λ_i.*

PROOF. Consider $\zeta(a)$ in the subuniverse $\Lambda_i + \zeta(a)$. Let us see how the generator $\zeta(a)$ works. Those operations in the definition of ζ which do not depend on a are outside of our concern. When a is used, it is being run. By the definition of a, we have a situation that can be schematically represented as follows:

$\langle 1 \rangle$ $\quad \zeta(a);$

$\langle 2 \rangle$ $\quad \zeta(\cdots \Lambda_i \to y; \gamma[y \subseteq S]; \to y + \gamma[\zeta(a) \subseteq S]; \to \zeta(a) \cdots).$

Of the two parts of the production of a, the first, running Λ_i, starts producing output immediately, and the definition of ζ starts using these branches immediately. The

second part must first check the condition P^s, and only then (and only in case of the true P^s) call $\zeta(a)$ recursively. We can write (2) shorter:

$$\langle 2' \rangle \qquad \zeta(\cdots y + \gamma[P^s]; \zeta(a)\cdots)$$

and keep in mind that y is, firstly, in Λ_i, and secondly, a subset of S.

In the call of $\gamma[P^s]$ the generator $\zeta(a)$ will again be used; it can be written in the form $\langle 2' \rangle$ again. The function γ will start "feeling" the branches represented by y; if they will not suffice to decide on P^s, then the next stage of recursion will take place, and $\langle 2' \rangle$ will be used again. The same applies to the call of $\zeta(a)$ in $\langle 2' \rangle$: it may happen that the top level ζ in $\langle 2' \rangle$ does not use the second term at all; if it does, $\zeta(a)$ must be replaced recursively by $\langle 2' \rangle$. We then have

$$\langle 3 \rangle \qquad \zeta(\cdots y + \gamma[P^s]; \zeta(\cdots y + \gamma[P^s]; \zeta(a)\cdots)\cdots).$$

So, the recursive call of a recedes further. The same phenomenon will occur in the call of P^s. We can represent the ultimate result as the infinite series

$$\zeta(\cdots y + \gamma[P^s]; \zeta(\cdots y + \gamma[P^s]; \zeta(\cdots y + \gamma[P^s]; \cdots)\cdots)\cdots)$$

which has no reference to a. This is the difference between the usual (computational) recursion and semantic recursion. Computational recursion does not create new objects. It only combines in various ways the nonrecursive parts of equations, and if there are no nonrecursive parts, it simply produces nothing. In our case, the nonrecursive parts are represented by the occurrences of y. Since the values of all the y's are in Λ_i, and all the ways to combine them leave the combinations in Λ_i, we conclude that all the expressions produced by the second branches of $\langle 2' \rangle$ are in Λ_i. Moreover, because of $\gamma(P^s)$, they all are subsets of S. But all possible subsets of S from Λ_i are already represented by the y's! The second branches can only reshuffle the same cards, represent the same sets in different forms. So we can ignore them, which proves the lemma.

As an example of a construct considered in Lemma 2, let us take $\bigcup a$. By the definition of union-set $\bigcup$, the production of x by $\bigcup a$ is represented by the graph

$$\bigcup a;\ a \to y;\ y \to x.$$

Using the definition of a, we have

$$\bigcup a;\ (\Lambda_i \to y;\ \gamma[y \subseteq S] \to y;\ y \to x + \gamma[\bigcup a \subseteq S] \to \bigcup a;\ \langle \bigcup a \rangle \to x).$$

The call of $\bigcup a$ enclosed in angular brackets is identical to the head call. So, even if the condition P^s in the second branch is satisfied (and in fact it is, because $\bigcup a = S$ is a subset of S), the second branch only produces again and again those and only those expressions which are already produced by the first branch.

Even more illuminating is the example of the construct $\bigcup\bigcup a$. In the case of $\bigcup a$, P^s is true for all S, i.e. for all a. But $\bigcup\bigcup a = \bigcup S$ may or may not be a subset of S. If it is, the second branches produce sets which could, at first glance, seem new, i.e. not produced by the first branches. However, a more careful analysis, which we leave to the reader, shows that they do not produce new sets. To prove this, one proves that if $\bigcup S \subseteq S$, then $\bigcup\bigcup S \subseteq S$, $\bigcup\bigcup\bigcup S \subseteq S$, etc.

Lemmas 1 and 2, together with the interpretability of P^h, prove Theorem 8.

THEOREM 9. *Adding* pow *to the constructors introduced before, we still have a regular subuniverse.*

PROOF. We proved above that adding pow(S) for some S, together with all constructs which can be produced using it, to the current subuniverse leaves the subuniverse regular. In our proof we used only the regularity of S. If Λ_i is the subuniverse without pow, we can include in it, step by step, pow(S) for all S of Λ_i. Let the resulting subuniverse be Λ_{i+i}. We then apply to it the same procedure, generating, in particular, such S-objects as pow(pow(S)), etc. This may be repeated indefinitely. The regularity of any construct using pow can be proven by induction on the number of pow's.

§5. Consistency of the ZF theory. We defined constructive SP-objects which have the properties of sets and propositions of the intuitive set theory. If our metatheory is intuitively trustworthy—and we believe it is—we have proved thereby the consistency of formal systems which include only axioms true in our interpretation, in particular, a considerable part of ZF. Now we want to make the proof more formal by reviewing its steps and making necessary additions.

1. We have defined interpretable P-objects and regular S-objects as certain metamechanical machines. The definitions are precise and formal. They refer only to the way these machines are used, but not to real users. Interpretable P-objects are either "labeled T" or "labeled F".

2. We have interpreted, in §2, every logical proposition as a P-object, if atomic predicates are interpreted as P-objects. Now we sum up the results of §4 as the following interpretation of ZF sets and atomic propositions:

TABLE 2

(a) *S-objects corresponding to sets*

$[\varnothing]$	=	fs( )
$[\{x \in A \mid P(x)\}]$	=	set$(x \in [A]: \gamma([P(x)])!)$
$[\{A, B\}]$	=	fs$([A], [B])$
$[\bigcup A]$	=	$\bigcup([A])$
[powerset of A]	=	pow$([A])$
[infinite set]	=	inf(fs( ))

(b) *P-objects corresponding to atomic propositions*

$[A \in B]$	=	el$([A] \in [B])!$
$[A = B]$	=	$[(Ax)(x \in A \equiv x \in B)]$

3. If a regular Λ_i and a true Γ_i are given, the *mechanical implementation* of an SP-object X is the machine which works as X, except that it calls Λ_i or Γ_i whenever Λ or Γ, respectively, is activated. If a P-object P itself is used for Γ, it is a *self-implementation.* We say that Γ_i *auto-produces* P if self-implementation of Γ_i produces P. The following theorem can be easily proven:

THEOREM 10 (Monotonicity theorem). *If γ_i is a mechanical implementation of γ, and $\overline{\gamma}_i$ that of $\overline{\gamma}$, then $\gamma_i(P)$ is finite only if P is labeled* T, *and $\overline{\gamma}_i(P)$ is finite only if P is labeled* F.

4. The usual logical identities can be shown to be true in our interpretation. There exists a P-object Γ_1 (our interpretation of the conjunction of logical axioms) such that if A is a conjunction of some extra-logical axioms, then Γ_1 & A auto-produces P iff P corresponds to a proposition derivable from A in classical logic.

5. On the other hand, it is easy to prove:

THEOREM 11 (Correctness theorem). *If A is a P-object labeled true, then any mechanical implementation, including self-implementation, of A can produce only propositions labeled* T.

6. To prove the consistency of ZF, we start by limiting ourselves to a recursively-enumerable Λ-implementation. We define the subuniverse Λ_{ZF} as a mechanical generator of all S-objects that can be formed using the set constructors introduced above. Note that when we use a Λ-implementation, this does not in the least alter the SP-objects: references to Λ are *not* replaced by references to Λ_{ZF}. This shows how the Skolem paradox is seen from our theory. Although the SP-objects definable in ZF can be enumerated, they still may refer to, and speak about, *all* possible objects of set theory, which, of course, cannot be enumerated.

If we now prove that all the P-objects corresponding to the ZF axioms are labeled T, we prove the consistency of ZF as a purely syntactical fact. Indeed, let A_{ZF} be the conjunction of the axioms. By step 4 above, if a proposition and its negation are derivable from A_{ZF}, then Λ_1 & A_{ZF} must auto-produce two P-objects, one of which is labeled F. This, however, is impossible because of the correctness theorem.

7. Thus we review the ZF axioms.

I. *Extensionality axiom.* Sets having the same elements are equal. This is one part of our definition of equality between sets. Using the reversed implication one can easily prove that the equality so defined is, as required, reflexive, symmetric, and transitive.

II. *Axiom of the empty set.* There is a set which has no elements. This set is $\underline{\text{fs}}(\,)$: a generator which produces nothing.

III. *Separation axiom.* For every set a and every property $P(x)$ of sets there exists a set whose elements are those and only those elements of a which have the property P. This set is

$$b = \underline{\text{set}}(x \in a: \gamma(P(x))!).$$

IV. *Pairing axiom.* Given any sets a and b, there exists a set c whose elements are exactly a and b. The set c is $\underline{\text{fs}}(a, b)$.

V. *Sum-set axiom.* For every set a there exists a set b whose elements are exactly those objects occurring in at least one element of a. This set is $\bigcup(a)$.

VI. *Powerset axiom.* For every set a there exists a set b the elements of which are exactly the subsets of a. This is $\underline{\text{pow}}(a)$.

VII. *Axiom of infinity.* There exists a set which includes the empty set and with every set x includes $x \cup \{x\}$. This is $\underline{\text{inf}}(\underline{\text{fs}}(\,))$.

VIII. *Axiom of replacement.* The image of a set under an operation (functional dependence) is again a set. More precisely, if a is a set and $F(x, y)$ is a formula such

that for every x from a there is exactly one y such that $F(x, y)$, then there exists a set the elements of which are exactly those y's for which an $x \in a$ exists such that $F(x, y)$:

$$\text{(REP)} \qquad (\mathrm{A}x)(\mathrm{E}y)(\mathrm{A}z)[F(x,z) \equiv y = z] \\ \rightarrow (\mathrm{E}b)(\mathrm{A}y)[y \in b \equiv (\mathrm{E}x)[x \in a \,\&\, F(x,y)]].$$

We shall prove this proposition in a limited form, namely in the assumption that the proof of the antecedent of (REP), i.e. the functional character of the predicate $F(x, y)$, is constructive: there exists an interpretable generator $G(x)$ that produces the required (regular) y, given an arbitrary (regular) x from a. Using $G(x)$, we construct b as an S-object which runs a, and for every produced x runs $G(x)$ and outputs its product. Clearly, it is regular and has the necessary property.

IX. *Axiom of regularity* (*or foundation*). Every nonempty set is disjoint from at least one of its elements. If every element of a has another element of a as its element, then there is an infinite (cyclic or acyclic) sequence of sets such that each next set is an element of the preceding one, which starts with a. Since a is regular this is impossible.

As for the axiom of choice, it does not hold in the interpretation we have defined above. However, it is possible to modify the definition of interpretability in such a way that the axiom of choice will be true. We have no space here to go into detail. Anyway, for the consistency of set theory this does not matter, because it was proved by Gödel [1940] that if ZF is consistent without the axiom of choice, it is consistent with it. We have come to

CONSISTENCY THEOREM. *The ZF system of axioms with limited Replacement Axiom* (*and without Axiom of Choice*) *is consistent.*

§6. Functions. Uncountable sets. In set theory, a *function* is a certain subset of the Cartesian product of certain sets. As everything in set theory, this definition is a static representation of an intuitive concept that is inherently dynamic, procedural. Our theory returns to the concept of function its intuitive procedural content.

An *objectively definable function* is a parametrized search which for every set of parameters (arguments) is objectively interpretable, and such that if it is finite on more than one parallel branch, then the final passive stage (the value of the function) is the same on all branches. It can be easily shown that this definition is equivalent to the usual set-theoretical definition when expressed in terms of our theory.

We identify a function with the procedure of its "computation"; the quotes are here because we extend the concept of computation by allowing reference to real-time processes Γ and Λ. Such a procedure may not be executable in a computer; it is a definition, not an algorithm. According to the current terminology, a function is *computable* if it can be defined by a purely mechanical process, an algorithm. We should better call such functions *mechanical.* Noncomputable functions should be properly called *nonmechanical.*

To compute a function $f(x)$ defined through a metamechanical process we have to replace the calls of Γ and Λ by the best of our "gnosis" and "logos" for today: Γ_i and Λ_i. The resulting mechanical function $f_i(x)$ will be referred to as an *implementation* of the function $f(x)$. From the monotonicity theorem we get the following proposition:

THEOREM 12. *Let $f(x)$ be an objectively definable function, and $f_i(x)$ its implementation. For a given x, if the search $f(x)$ is infinite, then $f_i(x)$ is also infinite; if*

$f(x)$ is finite, then $f_i(x)$ may be either finite or infinite, but if it is finite then its value $f_i(x)$ is equal to the value of $f(x)$.

One can see that our definition of function is, basically, constructive: a function is a computational procedure which may employ metamechanical processes. This extension of the concept of computation is fully justified. Metamechanical processes are distinguished from mechanical not in that they appeal to some "transcending" powers or use nonexistent omniscient creatures ("oracles"), but in that they include the activities of the subject of knowledge, the user of the mechanical device. Turing, who introduced machines on paper into mathematics, identified computation with the activity of an *autonomous* machine. But computational processes which we see around are autonomous only during some finite stretches of time: say, while a computer program is working. But it is the user of the computer who has written the program, and will throw it away and write a better one. It is the user who proves theorems and bases new algorithms on them. Mathematics is being done by human beings, not computers. Computation is a metamechanical process. Mechanical, algorithmic computation is only a special case.

There is a fundamental difference between the use of parallel processes in propositions and in functions. The interpretation of a parallel search in a proposition depends only on the finiteness or infiniteness of the branches, but not on their final stages. A functional parallel search is interpretable only if all finite branches produce one and the same result.

A functional search which runs an interpretable generator is, generally, uninterpretable, because it may cut off the process at some moment, and both this moment and the result of the search may depend on "the competition" of the branches, which is implementation dependent. In particular, the concept "the first member produced by the set generator" is uninterpretable if the generator involves metamechanical processes. So is, of course, "the second member", etc. While belonging to the set defined by a metamechanical generator is objectively interpretable, the order in which the members are produced is not. This explains why sets of set theory must be unordered if we want to consider not only recursively enumerable sets.

Tied closely to the concept of function is the concept of *numbering* set elements. Our interpretation leads naturally to these three types of sets:

1. A set produced by a mechanical generator is r.e. Its numbering function is mechanical.

2. A set generator which inescapably calls Γ, but does not call Λ, is not r.e., but is still *countable*. Its numbering function can be objectively defined, even though it is not mechanical.

3. A set generator which inescapably calls Λ (in ZF—through pow) is *uncountable*. For such a set there is no objectively definable numbering function.

The identification of sets with generators in constructive approaches to the foundation of mathematics usually stumbles over the interpretation of uncountable sets. In our theory, because of the introduction of metamechanical processes and the concept of objective interpretability, the constructive approach is compatible with the existence of genuinely uncountable sets. We should discuss how this becomes possible.

An uncountable set generator S produces its elements by "strata": for every subuniverse Λ_i, a subset S_i. This, in itself, is nothing new. To enumerate the resulting

set, there is the procedure of diagonalization, in use since Cantor. The members are arranged in the infinite rectangular table:

$$S_1 = \langle a_1, a_2, a_3, a_4, \ldots \rangle,$$
$$S_2 = \langle b_1, b_2, b_3, b_4, \ldots \rangle,$$
$$S_3 = \langle c_1, c_2, c_3, c_4, \ldots \rangle,$$
$$S_4 = \langle d_1, d_2, d_3, d_4, \ldots \rangle,$$
$$\ldots.$$

Then the diagonalization process generates all of them in the order $a_1, a_2, b_1, a_3, b_2, c_1, a_4, \ldots$. When an element appears repeatedly, we ignore it, thus counting only the first entry. So we come to an enumeration of the union set S.

This procedure, however, assumes (often tacitly) that there is a way *to define* the sequence S_1, S_2, etc., even though it is not necessary to be able to generate it mechanically. In our theory this is not true. The sets S_i are generated in real time by Λ. By Theorem 2, there exists no *objectively definable* generator which produces all the Λ_i. This is why diagonalization does not work. Although the limit of Λ_i for $i \to \infty$ (loosely understood) is the same for all subjects of knowledge, the specific stages in which it is achieved may be different. The order in which the interpretability of different expressions is proved may vary from user to user; it is not objectively defined. Therefore, the resulting enumeration is not objectively defined either. Moreover, even with a given sequence of S_i's, the enumeration, which must be a process in real time, will give different results depending on the frequency with which this process updates the current S_i. For example, if it makes readings twice as frequently as was assumed in the rectangular table above, then the table will be different, namely:

$$S_1 = \langle a_1, a_2, a_3, a_4, \ldots \rangle,$$
$$S_2 = \langle a_1, a_2, a_3, a_4, \ldots \rangle,$$
$$S_3 = \langle b_1, b_2, b_3, b_4, \ldots \rangle,$$
$$S_4 = \langle b_1, b_2, b_3, b_4, \ldots \rangle,$$
$$\ldots.$$

Now the ordering given by diagonalization is different from what it was above; e.g., a_3 precedes b_1, while they were in the opposite order before.

So, this is how uncountability is interpreted in our theory. Uncountable sets are generated by metamechanical processes, in the interaction between the user and the machine. The property of belonging to such a set is objectively definable. Since the only way a proposition of set theory can include a set is through the mediation of the function of membership el, propositions may refer to uncountable sets and still be objectively defined. But the order in which the elements of such a set are produced is not objectively definable; it depends on the user.

This interpretation is radically different from the one given by Cantor, according to which an uncountable set has "more" elements than a denumerable set. In our theory every set, and every element of every set, is represented by an expression. The set of all expressions (including uninterpretable) is, of course, denumerable (and even recursively enumerable). Thus all infinite sets in our theory are intuitively

perceived as having "less" elements than the denumerable set of all expressions. The relation of being a part (even if not a formal subset) between infinite sets is considered analogous to this relation between finite sets. But the impossibility of mapping one set onto another has nothing to do with the greater-less relationship between finite sets; it stems from the impossibility of defining an image for every given element in an *objective* way, i.e. avoiding any dependence on the user's individual sequence of subuniverses.

The theorem that a subset of a denumerable set is denumerable remains true, of course, but this does not lead to a contradiction, because the generator of all expressions is not a regular S-object. The fact that our universe of discourse stays always within the set of all expressions does not help us to enumerate it. By Theorem 1, it is impossible to separate interpretable expressions from uninterpretable ones, even if we are allowed to use metamechanical processes, not only mechanical. The ultimate universe is undefinable. Meaningful objects and propositions can only be constructed inductively, from bottom up, in the interaction between the user and the machine.

Acknowledgements. I appreciate many useful discussions of the Cybernetic Foundation with Karel Hrbáček, Angus Macintyre, and Robin Gandy. I am especially grateful to Angus Macintyre for guidance through the recent literature in logic and set theory, and constant attention to this work.

REFERENCES

L. E. J. BROUWER [1949], *Consciousness, philosophy, and mathematics*, ***Library of the tenth international congress of philosophy (Amsterdam, 1948)***, vol. I, North-Holland, Amsterdam, pp. 1235–1249; reprinted in his ***Collected works***, vol. 1 (A. Heyting, editor), North-Holland, Amsterdam, 1975, pp. 480–494.

——— [1952], *Historical background, principles and methods of intuitionism*, ***South African Journal of Science***, vol. 49, pp. 139–146; reprinted in his ***Collected works***, vol. 1 (A. Heyting, editor), North-Holland, Amsterdam, 1975, pp. 508–515.

G. CANTOR [1895–97], *Beiträge zur Begründungen der transfiniten Mengenlehre*. I, II, ***Mathematische Annalen***, vol. 46, pp. 485–512, and vol. 49, pp. 207–246; English translation (by P. E. B. Jourdain), Open Court, Chicago, Illinois, 1915; reprint, Dover, New York, 1952.

FBL [1967]: A. A. FRAENKEL, Y. BAR-HILLEL, and A. LÉVY, ***Foundations of set theory***, 2nd rev. ed., North-Holland, Amsterdam.

KURT GÖDEL [1940], ***The consistency of the axiom of choice and the generalized continuum hypothesis with the axioms of set theory***, Annals of Mathematics Studies, no. 3, Princeton University Press, Princeton, New Jersey.

S. C. KLEENE [1952], ***Introduction to metamathematics***, Van Nostrand, New York. (See especially pp. 332–340.)

G. KREISEL [1967], *Informal rigour and completeness proofs*, ***Problems in the philosophy of mathematics (proceedings of the international colloquium in the philosophy of science, London 1965)***, vol. 1 (I. Lakatos, editor), North-Holland, Amsterdam, pp. 138–186.

S. KRIPKE [1975], *Outline of a theory of truth*, ***Journal of Philosophy***, vol. 72, pp. 690–716.

E. MARTINO [1982], *Creative subject and bar theorem*, ***The L. E. J. Brouwer centenary symposium*** (A. S. Troelstra and D. van Dalen, editors), North-Holland, Amsterdam, pp. 311–318.

YURI F. ORLOV [1978], *Wave calculus based on wave logic*, ***International Journal of Theoretical Physics***, vol. 17, pp. 585–598.

——— [1982], *The wave logic of consciousness: a hypothesis*, ***International Journal of Theoretical Physics***, vol. 21, pp. 37–53.

H. ROGERS [1967], ***Theory of recursive functions and effective computability***, McGraw-Hill, New York.

A. TARSKI [1933], *The concept of truth in the languages of deductive sciences*, ***Prace Towarzystwa Naukowego Warszawskiego, Wydzial III***, no. 34 (in Polish); German translation, ***Studia Philosophica***, vol. 1 (1935), pp. 261–405; English translation in his ***Logic, semantics, metamathematics. Papers from 1923 to 1938***, Clarendon Press, Oxford, 1956, pp. 152–278.

V. F. TURCHIN [1980], ***The language Refal***, Courant Computer Science Report no. 20, New York University, New York.

E. ZERMELO [1890], *Untersuchungen über die Grundlagen der Mengenlehre*, ***Mathematische Annalen***, vol. 65, pp. 261–281; English translation in ***From Frege to Gödel: a source book in mathematical logic*** (J. van Heijenoort, editor), Harvard University Press, Cambridge, Massachusetts, 1967, pp. 200–215.

DEPARTMENT OF COMPUTER SCIENCES
THE CITY COLLEGE
CITY UNIVERSITY OF NEW YORK
NEW YORK, NEW YORK 10031