Understanding Intuitionism

by Edward Nelson
Department of Mathematics
Princeton University
http://www.math.princeton.edu/~nelson/papers.html

Intuitionism was the creation of L. E. J. Brouwer [Br], and I like to think that classical mathematics was the creation of Pythagoras. Imagine a conversation between a classical mathematician and an intuitionist, say a woman and a man respectively. She speaks first:

I have just proved $\exists x A$.
Congratulations! What is it?
I don't know. I assumed $\forall x \neg A$ and derived a contradiction.
Oh. You proved $\neg \forall x \neg A$.
That's what I said.

Or another:

I have proved $A \vee B$.
Good. Which did you prove?
What?
You said you proved A or B; which did you prove?
Neither; I assumed $\neg A$ & $\neg B$ and derived a contradiction.
Oh, you proved $\neg[\neg A$ & $\neg B]$.
That's right. It's another way of putting the same thing.

But he does not agree with her last statement; they have a different semantics and a different notion of proof. This paper is an attempt to understand the differences between them.

I am grateful to Mitsuru Yasuhara for stimulating discussions of this material and for pinpointing errors and obscurities in earlier drafts. I also wish to thank Simon Kochen and Per Martin-Löf for helpful comments.

1. Incomplete communications

For a classical mathematician, a closed formula, true in a given structure, is a complete communication. It expresses an objective state of affairs in the universe of discourse; it is an ontological assertion. But to an intuitionist, a closed true formula may be an incomplete communication. For an existential formula he wants an object n for which $A_x(n)$ holds and further, since $A_x(n)$ may itself be an incomplete communication, he wants the information needed to complete it. Similarly, for $A \vee B$ he wants to know which, together with the information needed to complete that communication.

To be specific, let us begin by discussing Arithmetic. The language $\mathcal{L}$ of Arithmetic consists of the the constant 0, the unary function symbol S (*successor*), $+$, and $\cdot$, together with the logical symbols: the variables, $=$, and the logical constants $\neg$ & $\forall$ $\rightarrow$ $\exists$ $\vee$. This language is the same for classical and intuitionistic arithmetic. Also the structure for the language, the universe of discourse to which the language refers, is the same: the universe is $\mathbb{N}$, and the function symbols denote the usual arithmetical operations. A *numeral* is a term containing only S and 0, and each element of $\mathbb{N}$ is denoted by a unique numeral and vice versa.

Observe that a closed atomic formula C of $\mathcal{L}$ is an equation $a = b$ where a and b are variable-free terms. Any variable-free term reduces to a unique numeral when the usual laws

$$\begin{aligned} x + 0 &= x \\ x + \mathrm{S}y &= \mathrm{S}(x + y) \\ x \cdot 0 &= 0 \\ x \cdot \mathrm{S}y &= (x \cdot y) + x \end{aligned}$$

are applied repeatedly from left to right. We say that C is *true* in case a and b reduce to the same numeral.

Negation can be eliminated in both classical and intuitionistic arithmetic: replace $\neg A$ by $A \rightarrow \mathrm{S}0 = 0$.

Here is an attempt, following Kleene [Kl, §82], to put into words the information an intuitionist needs to complete the communication in a true closed formula C of Arithmetic. The description is by recursion on the complexity of C:

(I1) Let C be atomic: no additional information.

(I2) Let C be A & B: the information needed to complete A and the information needed to complete B.

(I3) Let C be $\forall x A$: a general method yielding for any numeral n the information needed to complete $A_x(n)$.

(I4) Let C be $A \rightarrow B$: a general method yielding for any information completing A the information completing B.

(I5) Let C be $\exists x A$: a numeral n together with the information needed to complete $A_x(n)$.

(I6) Let C be $A \vee B$: 1 or 2, indicating first or second disjunct, together with the information needed to complete it.

Thus we see that the intuitionistic notion of truth is far richer than the classical: rather than one bit of information (true) it may specify numbers, choices of disjuncts, and general methods.

2. Codes

Kleene [Kl45] [Kl, §82] made an incisive analysis of intuitionistic semantics based on the notion of realization. I shall express this notion in a different formalism that avoids the intricacies of recursive partial functions and Gödel numbering.

We introduce ten *additional function symbols*, written here with variables to indicate the placement of arguments:

$$0 \quad 1 \quad 2 \quad \langle x, y \rangle \quad \pi_1 x \quad \pi_2 x \quad x\{y\} \quad \gamma(x, y, z) \quad \lambda x y \quad \rho(x, y)$$

called *zero*, *first position*, *second position*, *pair*, *first projection*, *second projection*, *evaluation*, *choice*, *lambda*, and *recursion*. Recursion is special to Arithmetic. Let $\mathcal{L}'$ be $\mathcal{L}$ together with the additional function symbols.

I have called λ a function symbol but in many respects it is like a quantifier symbol. A *code* is a term c of $\mathcal{L}'$ such that for every occurrence of λ in c, its first argument is a variable. The notions of λ-*bound* and λ-*free* occurrences of variables in codes, of λ-*closed* codes, of a code being λ-*substitutable* for a variable in another code, and of a λ-*variant* of a code are all defined by strict analogy with the corresponding notions for quantifier symbols. If a and b are codes and x is a variable, $a_x(b)$ is the code obtained by substituting b for each λ-free occurrence of x in a; when we use this notation, we assume that b is λ-substitutable for x in a (otherwise, first replace a by a λ-variant a' such that b is λ-substitutable for x in a').

It will not be necessary to distinguish notationally between the additional constant 0 and the constant 0 of $\mathcal{L}$. We call 1 and 2 *positions*. The projections are tightly binding, so $\pi_1 c\{n\}$ is parsed as $(\pi_1 c)\{n\}$. Parentheses are used when necessary for grouping. Unless indicated otherwise by parentheses, the scope of an occurrence of λx is the entire following expression.

The meaning of the additional function symbols is expressed by the following *reduction rules for codes*:

	replace:	by:
(R1)	$\pi_1\langle a, b\rangle$	a
(R2)	$\pi_2\langle a, b\rangle$	b
(R3)	$\gamma(a, b, \langle 1, c\rangle)$	$a\{c\}$
(R4)	$\gamma(a, b, \langle 2, c\rangle)$	$b\{c\}$
(R5)	$(\lambda x a)\{b\}$	$a_x(b)$
(R6)	$a + 0$	a
(R7)	$a + \mathrm{S}b$	$\mathrm{S}(a + b)$
(R8)	$a \cdot 0$	0
(R9)	$a \cdot \mathrm{S}b$	$(a \cdot b) + a$
(R10)	$\rho(\mathrm{S}n, c)$	$\pi_2 c\{n\}\{\rho(n, c)\}$
(R11)	$\rho(0, c)$	$\pi_1 c$

The rules (R1) and (R2) say that the first or second projection of a pair is the corresponding member of the pair; (R3) and (R4) give a choice of whether to evaluate a or b on c; the rule (R5) is the usual rule of the λ-calculus. The remaining rules are special to Arithmetic. If a is a variable-free term of $\mathcal{L}$, the rules (R6)–(R9) reduce it to a numeral n, and then applications of (R10) and (R11) eliminate this occurrence of ρ by recursion.

A code is *irreducible* in case none of these reduction rules applies to any code occurring in it. Notice that for any term of $\mathcal{L}'$, if the arguments of the term are irreducible then at most one reduction rule applies to it. The *reduction algorithm for codes* is this: repeatedly apply the reduction rules from right to left (that is, starting at the leaves of the tree structure of the term). Because of (R5), this algorithm may not terminate. A code is *terminating* in case the algorithm terminates, in which case we say that the code *reduces to* the irreducible code with which the algorithm terminates. Let r be the function with domain the set of all terminating codes c whose value $\mathsf{r}c$ is the code to which it reduces. In our notation r binds less tightly than all the function symbols of $\mathcal{L}'$, so that $\mathsf{r}c\{a\}$ is parsed as $\mathsf{r}(c\{a\})$.

An example of a non-terminating code is $(\lambda x x\{x\})\{\lambda x x\{x\}\}$, which keeps on repeating itself, and there are others with explosive growth. But if c is terminating, this can be effectively determined and the value $\mathsf{r}c$ can be effectively computed. This is a concrete, syntactical formalism, implementable by computer.

Now we describe the relation between a code c and a formula C. This can be expressed by a program B. I believe it to be a faithful expression of Brouwer's intent when he formulated intuitionism. It is not an algorithm but an interactive program, since in general it will prompt from time to time for input during its execution. Let c be a λ-closed code and C a closed formula of $\mathcal{L}$. Then the action of $\mathsf{B}(c, C)$ is described as follows:

(B1) Let C be atomic. Then $\mathsf{B}(c, C)$ reduces the two variable-free terms in the equation C to numerals and prints "true" if they are the same and "false" otherwise. This does not depend on c.

(B2) Let C be A & B. Then $\mathsf{B}(c, C)$ splits into two programs, running independently of each other, namely $\mathsf{B}(\pi_1 c, A)$ and $\mathsf{B}(\pi_2 c, B)$.

(B3) Let C be $\forall x A$. Then $\mathsf{B}(c, C)$ prompts for a numeral n and runs $\mathsf{B}\big(c\{n\}, A_x(n)\big)$.

(B4) Let C be $A \to B$. Then $\mathsf{B}(c, C)$ prompts for a λ-closed code a and forms a pipe: it runs $\mathsf{B}(a, A)$ and if this terminates (and since it is an interactive program, that may depend on the responses to prompts), then it runs $\mathsf{B}(c\{a\}, B)$.

(B5) Let C be $\exists x A$. Then $\mathsf{B}(c, C)$ prints $\mathsf{r}\pi_1 c$ (if $\pi_1 c$ is terminating; otherwise, the program does not terminate) and then runs the program $\mathsf{B}\big(\pi_2 c, A_x(\mathsf{r}\pi_1 c)\big)$ (if $\mathsf{r}\pi_1 c$ is a numeral; otherwise it prints an error message).

(B6) Let C be $A \vee B$. Then $\mathsf{B}(c, C)$ prints $\mathsf{r}\pi_1 c$ (if $\pi_1 c$ is terminating; otherwise, the program does not terminate) and then runs the program $\mathsf{B}(\pi_2 c, A)$ if $\mathsf{r}\pi_1 c$ is 1, $\mathsf{B}(\pi_2 c, B)$ if $\mathsf{r}\pi_1 c$ is 2, and otherwise prints an error message.

We call $\forall$ and $\to$ *input operators* and $\exists$ and $\vee$ *output operators.*

3. Realization

The rules (B1)–(B6) are concrete, syntactical rules implementable by computer. But now we ask, what is a good code for a formula? When does a code c express the intuitionistic truth of the formula C? Let c be a terminating λ-closed code and C a closed formula of $\mathcal{L}$. Here is Kleene's definition of c *realizes* C, abbreviated c *rz* C, by recursion on the complexity of C. It is expressed in our formalism and slightly modified in the first clause (the realization code for a true closed atomic formula can be arbitrary, not necessarily 0):

(K1) Let C be atomic. Then c rz C in case C is true.

(K2) Let C be A & B. Then c rz C in case $\pi_1 c$ rz A *and* $\pi_2 c$ rz B.

(K3) Let C be $\forall x A$. Then c rz C in case *for all* numerals n, $c\{n\}$ rz $A_x(n)$.

(K4) Let C be $A \rightarrow B$. Then c rz C in case *for all* λ-closed codes a, a rz A *implies* $c\{a\}$ rz B.

(K5) Let C be $\exists x A$. Then c rz C in case $\mathsf{r}\pi_1 c$ is a numeral *and* $\pi_2 c$ rz $A_x(\mathsf{r}\pi_1 c)$.

(K6) Let C be $A \vee B$. Then c rz C in case $\mathsf{r}\pi_1 c$ is a position, *and* $\mathsf{r}\pi_1 c$ is 1 *implies* $\pi_2 c$ rz A, *and* $\mathsf{r}\pi_1 c$ is 2 *implies* $\pi_2 c$ rz B.

(K*) For a formula C that is not closed, let $x_1, \ldots, x_i$ be the distinct variables occurring free in C, say in the order of first free occurrence. Then c *realizes* C in case $\lambda x_1 \ldots \lambda x_i c$ realizes $\forall x_1 \ldots \forall x_i C$ (so c is terminating and λ-closed). Equivalently, for all numerals $n_1, \ldots, n_i$, we have $c_{x_1,\ldots,x_i}(n_1, \ldots, n_i)$ rz $C_{x_1,\ldots,x_i}(n_1, \ldots, n_i)$.

Because of the quantifiers *for all* over infinite domains (and notice that in (K4) they can occur in the hypotheses of implications), realization is an abstract notion.

The following two lemmas are proved by induction on the complexity of C:

Lemma 1: Let C a closed formula of $\mathcal{L}$. Then c rz C if and only if $\mathsf{r}c$ rz C.

Lemma 2: Let A be a formula of $\mathcal{L}$ whose only free variable is x, and let a be a variable-free term of $\mathcal{L}$. Let $\mathsf{r}a$ be the numeral n. Then c rz $A_x(a)$ if and only if c rz $A_x(n)$.

A formula is *classical* in case there is no occurrence of an output operator $\exists$ or $\vee$ in it. Every formula is classically equivalent to a classical formula.

Theorem 1: Let C be a closed classical formula of $\mathcal{L}$ and let c be a terminating λ-closed code. Then c rz C if and only if 0 rz C.

Proof: By induction on the complexity of C, using (K1)–(K4). ▪ (The reasoning does not apply to non-classical formulas, since (K5)–(K6) restrict the form of realization codes for them.)

Consequently, we can simply say C is *true*, rather than c rz C, for C a closed classical formula. Then for closed classical formulas, A & B is true if and only if A is true and B is true; $\forall x A$ is true if and only if for all numerals n, $A_x(n)$ is true; $A \rightarrow B$ is true if and only if [not A is true] or [B is true].

In short, the classical and intuitionistic semantics are identical on classical formulas.

Let us remark that for any closed formula C, classical or not, $c \,\mathrm{rz}\, C$ is expressed, in the metalanguage, with only the classical logical constants *and*, *for all*, *implies*. Consequently, the classical and intuitionistic semantics of $c \,\mathrm{rz}\, C$ are identical.

A classical mathematician might say that for a classical formula A, $\neg\forall x\neg A$ means that there exists a numeral n such that $A_x(n)$ is true, but to her this means exactly the same as saying that not $\forall x\neg A$ is true, so there is no dispute over any matter of substance between her and an intuitionist. She need only defer to his sensibilities by rephrasing statements in ways that to her are entirely equivalent and then they will be in complete accord on the semantics of classical formulas.

We are led to the following conclusion: the intuitionistic semantics is ontological and Platonic. This is because the classical semantics of classical formulas is ontological and Platonic, but intuitionistic semantics is identical with classical semantics on classical formulas.

It appears that this conclusion can be challenged only by maintaining that realization is an unsatisfactory expression of intuitionistic semantics. We shall return to intuitionistic semantics, but first we need to discuss some syntactical questions.

4. Intuitionistic proofs

Let us list the axioms and rules of inference of Intuitionistic Arithmetic. What follows is a version of Heyting's formalization [He]. With each axiom or axiom scheme we associate a code; these codes will be discussed shortly.

The language is $\mathcal{L}$. We associate $\to$ from right to left. In the list, A, B, and C are formulas, x and y (possibly with subscripts) are variables, and a is a term (substitutable for x in A). The nonlogical axioms are the usual:

1.	$\neg\, \mathrm{S}x = 0$	0
2.	$\mathrm{S}x = \mathrm{S}y \to x = y$	0
3.	$x + 0 = x$	0
4.	$x + \mathrm{S}y = \mathrm{S}(x + y)$	0
5.	$x \cdot 0 = 0$	0
6.	$x \cdot \mathrm{S}y = (x \cdot y) + x$	0
7.	$A_x(0) \,\&\, \forall x[A \to A_x(\mathrm{S}x)] \to A$	$\lambda c\rho(x, c)$

The logical axioms are the identity and equality axioms:

8.	$x = x$	0
9.	$x = y \to \mathrm{S}x = \mathrm{S}y$	0
10.	$x_1 = y_1 \to x_2 = y_2 \to x_1 + x_2 = y_1 + y_2$	0
11.	$x_1 = y_1 \to x_2 = y_2 \to x_1 \cdot x_2 = y_1 \cdot y_2$	0
12.	$x_1 = y_1 \to x_2 = y_2 \to x_1 = x_2 \to y_1 = y_2$	0

together with axioms schemes for the propositional calculus:

13.	$A \to B \to A$	$\lambda a \lambda b a$
14.	$[A \to B] \to [A \to B \to C] \to A \to C$	$\lambda p \lambda q \lambda a q\{a\}\{p\{a\}\}$
15.	$A \to B \to A \mathbin{\&} B$	$\lambda a \lambda b \langle a, b\rangle$
16.	$A \mathbin{\&} B \to A$	$\lambda c \pi_1 c$
17.	$A \mathbin{\&} B \to B$	$\lambda c \pi_2 c$
18.	$A \to A \vee B$	$\lambda a \langle 1, a\rangle$
19.	$B \to A \vee B$	$\lambda b \langle 2, b\rangle$
20.	$[A \to C] \to [B \to C] \to A \vee B \to C$	$\lambda p \lambda q \lambda r \gamma(p, q, r)$
21.	$[A \to B] \to [A \to \neg B] \to \neg A$	$\lambda p \lambda q \lambda a q\{a\}\{p\{a\}\}$
22.	$\neg A \to A \to B$	0

and the substitution axiom schemes of the predicate calculus:

23.	$A_x(a) \to \exists x A$	$\lambda y \langle a, y\rangle$
24.	$\forall x A \to A_x(a)$	$\lambda y y\{a\}$.

A few comments on the axioms: (13) and (14) justify the use of deductions (introduction and discharge of hypotheses); with the elimination of $\neg$, (21) is the special case of (14) in which C is $\mathrm{S}0 = 0$; if the axiom (22) is changed to (22′): $\neg\neg A \to A$, then the resulting theory is classical Peano Arithmetic [Kl, §19].

There are three rules of inference:

25. $A,\ A \to B \ /\ B$
26. $A \to B \ /\ \exists x A \to B \quad (x \text{ not free in } B)$
27. $A \to B \ /\ A \to \forall x B \quad (x \text{ not free in } A)$.

To each of these rules of inference we associate a *rule for constructing codes*. For (25), let $y_1, \ldots, y_j$ be the variables occurring free in A but not in B; for (26) and (27), let y be a variable that does not occur in A or B. Then:

if a is a code associated with A and c is a code associated with $A \to B$, associate $(c\{a\})_{y_1 \dots y_j}(0 \dots 0)$ with B;

if c is associated with $A \to B$ and x is not free in B, associate $\lambda y(\lambda xc)\{\pi_1 y\}\{\pi_2 y\}$ with $\exists x A \to B$;

if c is associated with $A \to B$ and x is not free in A, associate $\lambda y \lambda xc\{y\}$ with $A \to \forall x B$.

We use Kleene's notation S for this intuitionistic theory. A simple algorithm checks whether a purported intuitionistic proof in S is indeed a proof, associates to each axiom its code, and associates to each formula derived by a rule of inference the code constructed according to the rules for constructing codes. Thus the algorithm associates to each proof in S a code for the theorem of which it is a proof. This is concrete and syntactical, implementable by computer.

The codes for the axioms and code construction rules for the rules of inference are due to David Nelson. (He and I are unrelated.) At Kleene's suggestion, Nelson undertook to establish that each intuitionistic theorem of Arithmetic is realizable, and achieved this in [Ne].

Let us give a proof in our formalism of Nelson's theorem. First we need to show that the associated codes realize the axioms. By (K*) and (K3), we assume that in each axiom the free variables are replaced by numerals, with the same replacements for the λ-free variables in the associated code.

Since each variable-free instance of (1)–(6) is true, 0 realizes each of them, and similarly for the identity and equality axioms (8)–(12).

7: By (K*) we assume that x is the only free variable in A. Suppose c *rz* $A_x(0)$ & $\forall x[A \to A_x(\mathrm{S}x)]$. To show: $\rho(n,c)$ *rz* $A_x(n)$ (where n is a numeral). The proof is by induction on n, and the beauty of the argument is that a single induction yields the realization of all induction axioms. By (R11), $\mathsf{r}\rho(0,c)$ is $\mathsf{r}\pi_1 c$. By (K2) and Lemma 1, this realizes $A_x(0)$. Suppose $\rho(n,c)$ *rz* $A_x(n)$. To show: $\rho(\mathrm{S}n,c)$ *rz* $A_x(\mathrm{S}n)$. By (R10), $\mathsf{r}\rho(\mathrm{S}n,c)$ is $\mathsf{r}\pi_2 c\{n\}\{\rho(n,c)\}$. By (K2) and (K3), $\pi_2\{n\}$ *rz* $A_x(n) \to A_x(\mathrm{S}n)$, so by (K4) and the induction hypothesis $\pi_2 c\{n\}\{\rho(n,c)\}$ *rz* $A_x(\mathrm{S}n)$. The proof is complete by Lemma 1.

In the following proofs, such frequent use is made of the reduction rule (R5), for $(\lambda xa)\{b\}$, and (K4), for $\to$, that no explicit reference is made to these rules.

13: Suppose a' *rz* A. To show: $\lambda ba'$ *rz* $B \to A$. Suppose b' *rz* B. To show: a' *rz* A. But it does, by hypothesis.

14: Suppose p' *rz* $A \to B$. To show: $\lambda q \lambda aq\{a\}\{p'\{a\}\}$ *rz* $[A \to B \to C] \to A \to C$. Suppose q' *rz* $A \to B \to C$. To show: $\lambda aq'\{a\}\{p'\{a\}\}$ *rz*

$A \to C$. Suppose a' *rz* A. To show: $q'\{a'\}\{p'\{a'\}\}$ *rz* C. But since a' *rz* A and q' *rz* $A \to B \to C$, we have $q'\{a'\}$ *rz* $B \to C$. Also, since q' *rz* $A \to B$, we have $p'\{a'\}$ *rz* B. Consequently, $q'\{a'\}\{p'\{a'\}\}$ *rz* C.

15: Suppose a' *rz* A. To show: $\lambda b\langle a', b\rangle$ *rz* $B \to A \mathbin{\&} B$. Suppose b *rz* B. To show: $\langle a', b'\rangle$ *rz* $A \mathbin{\&} B$. But it does, by (K2).

16: Suppose c' *rz* $A \mathbin{\&} B$. To show: $\pi_1 c'$ *rz* A. This holds by (K2).

17: Suppose c' *rz* $A \mathbin{\&} B$. To show: $\pi_2 c'$ *rz* B. This holds by (K2).

18: Suppose a' *rz* A. To show: $\langle 1, a'\rangle$ *rz* $A \vee B$. By (R1), $\mathsf{r}\pi_1\langle 1, a'\rangle$ is 1, so this holds by (K6).

19: Suppose b' *rz* B. To show: $\langle 2, b'\rangle$ *rz* $A \vee B$. By (R1), $\mathsf{r}\pi_1\langle 2, a'\rangle$ is 2, so this holds by (K6).

20: Suppose p' *rz* $A \to C$. To show: $\lambda q\lambda r\gamma(p', q, r)$ *rz* $[B \to C] \to A \vee B \to C$. Suppose r' *rz* $A \vee B$. To show: $\lambda r\gamma(p', q', r)$ *rz* $A \vee B \to C$. Suppose r' *rz* $A \vee B$. To show: $\gamma(p', q', r')$ *rz* C. By the reduction algorithm for codes (which works from right to left) and Lemma 1, we assume that p', q', and r' are irreducible. By (K6), $\mathsf{r}\pi_1 r'$ is a position, and there are two cases. First, suppose $\mathsf{r}\pi_1 r'$ is 1. Since r' is irreducible, by (R1) it is of the form $\langle 1, a\rangle$ where a is irreducible. By (K6), a *rz* A. Since p' *rz* $A \to C$, $p'\{a\}$ *rz* C. But $\mathsf{r}\gamma(p', q', r')$—that is, $\mathsf{r}\gamma(p', q', \langle 1, a\rangle)$—is $\mathsf{r}p'\{a\}$ by (R3). By Lemma 1, $\gamma(p', q', r')$ *rz* C. The second case, that $\mathsf{r}\pi_1 r'$ is 2, is similar.

21: This is the special case of (14) in which C is $\mathrm{S}0 = 0$.

22: Suppose c *rz* $\neg A$; that is, c *rz* $A \to \mathrm{S}0 = 0$. To show: $0\{c\}$ *rz* $A \to B$. That is, we need to show that if a *rz* A then $0\{c\}\{a\}$ *rz* B. But this holds vacuously since the hypothesis is untenable: if a *rz* A then $c\{a\}$ *rz* $\mathrm{S}0 = 0$, which is impossible by (K1).

23: Suppose b *rz* $A_x(a)$. To show: $\langle a, b\rangle$ *rz* $\exists x A$. Let n be $\mathsf{r}a$, so that n is a numeral. By Lemma 2, b *rz* $A_x(n)$. By (K5), $\langle a, b\rangle$ *rz* $\exists x A$.

24: Suppose b *rz* $\forall x A$. To show: $b\{a\}$ *rz* $A_x(a)$. Let n be $\mathsf{r}a$, so that n is a numeral. By (K3), $b\{n\}$ *rz* $A_x(n)$. By Lemma 1, $b\{a\}$ *rz* $A_x\{n\}$, and by Lemma 2, $b\{a\}$ *rz* $A_x(a)$.

Finally, we must prove that the rules of inference with the rules for constructing codes preserve realization.

25: Suppose a *rz* A and c *rz* $A \to B$. Let $y_1, \ldots, y_j$ be the variables occurring free in A but not B and let b be $c\{a\}_{y_1 \ldots y_j}(0 \ldots 0)$. To show: b *rz* B. Let a', b', c', A', and B' be the codes and formulas obtained by replacing the λ-free and free variables by numerals, but with the $y_1, \ldots, y_j$ replaced by 0. By (K*), we need to show that b' *rz* B'. But c' *rz* $A' \to B'$ and a' *rz* A', so $c'\{a'\}$ *rz* B'. But $c'\{a'\}$ is b'.

26: Suppose c *rz* $A \to B$, where x is not free in B. To show: $\lambda y(\lambda xc)\{\pi_1 y\}\{\pi_2 y\}$ *rz* $\exists xA \to B$. By (K*) we assume that x is the only free variable in $A \to B$. Suppose d *rz* $\exists xA$. Then we need to show: $(\lambda xc)\{\pi_1 d\}\{\pi_2 d\}$ *rz* B. By (K5), $\mathsf{r}\pi_1 d$ is a numeral n and $\pi_2 d$ *rz* $A_x(n)$. By (K*), $c_x(n)$ *rz* $A_x(n) \to B$ (this is where the assumption that x is not free in B is used), so $c_x(n)\{\pi_2 d\}$ *rz* B. But $\mathsf{r}(\lambda xc)\{\pi_1 d\}\{\pi_2 d\}$ is $\mathsf{r}c\{n\}\{\pi_2 d\}$ by (R5), so $(\lambda xc)\{\pi_1 d\}\{\pi_2 d\}$ *rz* B by Lemma 1.

27: Suppose c *rz* $A \to B$, where x is not free in A. To show: $\lambda y \lambda xc\{y\}$ *rz* $\forall xB$. By (K*) we assume that x is the only free variable in $A \to B$. Suppose a *rz* A. To show: $\lambda xc\{a\}$ *rz* $\forall xB$. Let n be a numeral. By (K3), we must show that $(\lambda xc\{a\})\{n\}$ *rz* $B_x(n)$. But $\mathsf{r}(\lambda xc\{a\})\{n\}$ is $\mathsf{r}c_x(n)\{a\}$ by (R5), since a is λ-closed (this is where the assumption that x is not free in A is used). By (K*), $c_x(n)$ *rz* $A \to B_x(n)$, so $(\lambda xc\{a\})\{n\}$ *rz* $B_x(n)$ by Lemma 1.

This concludes the proof of Nelson's theorem.

We can argue that since, by (K1), $S0 = 0$ is not realizable, it is not a theorem of S, so S is consistent. This is entirely analogous to the classical argument that Peano Arithmetic is consistent since the axioms are true and the rules of inference preserve truth. The argument cannot be formalized in S, if S is consistent, because of the arbitrarily large number of alternating quantifiers needed to express the realization predicate for formulas of high complexity. The realization predicate is not arithmetical.

We have seen that a classical mathematician can explain a theorem to an intuitionist in a manner that satisfies them both, but will he accept her proof? This question was answered affirmatively, for Arithmetic, by Gödel [Gö33] [Kl, §81]. The demonstration is by a very simple algorithm that transforms a classical proof of a classical formula into an intuitionistic proof of the same formula. Thus intuitionism is an extension of classical mathematics in its proof syntax as well as semantically.

In practice if not in profession, intuitionistic arithmetic takes classical arithmetic, with classical semantics and proof syntax, as its base. Brouwer created two new logical constants: the constructive $\exists$ and the constructive $\vee$. The intuitionistic semantics and proof syntax of $\exists xA$ and $A \vee B$ are quite different from those of $\neg\forall x \neg A$ and $\neg[\neg A \mathbin{\&} \neg B]$. Stricter standards of proof are required and richer information is obtained.

Let us call a formula C *unconditional* in case there is no occurrence of $\to$ in it. For an unconditional C, the program $\mathsf{B}(c, C)$ in noninteractive; it is an algorithm. An intuitionistic proof of C yields a realization code c, and this constructs recursive functions for the existentially quantified variables with the values of the universally quantified variables as arguments; the placement of the quantifiers in C indicates the dependencies. This suggests the possibility of using Nelson's algorithm as a means to construct and verify algorithms in a mode congenial to mathematical ways of thinking. This is potentially a powerful tool, but the practical applications of intuitionism as a method for obtaining constructive information have been meager for most of this century. Most often, important constructive results are obtained with no use of intuitionism, for example Tarski's decision procedure [Ta] for real algebra with inequalities, which played an essential role in Hörmander's classification [Hö] of hypoelliptic partial differential equations. But the situation is changing, spurred in part by the availability of fast computers. One indication of this is the work of Martin-Löf and his school [M-Lö] [NoPeSm].

For a conditional formula C, the program $\mathsf{B}(c, C)$ is interactive. Let C be $\exists x A \to \exists y B$, where A and B are unconditional, and let $c \mathrel{rz} C$. It may be difficult to find an a realizing $\exists x A$, and only those in the know would have access, via $c\{a\}$, to an n such that $B_y(n)$ holds. If A and B are themselves conditional, the situation is more complex. Kleene's realization predicate may be the natural tool for analyzing questions concerning complex interactive programs, such as problems of system security.

5. Kripke semantics

This is a very brief discussion of Kripke's analysis [Kr] of intuitionism.

Kleene was concerned with the intuitionistic semantics of Arithmetic under the intended interpretation; Kripke is concerned with the intuitionistic semantics of the predicate calculus under all possible interpretations.

In discussing the intuitionistic predicate calculus, rather than arithmetic, we must make two changes. First, we do not have a specific false formula $S0 = 0$ to use in eliminating $\neg$, so restore $\neg$. Second, in the comparison of classical and intuitionistic arithmetic implicit use was made of the fact that intuitionistically an atomic formula of arithmetic is equivalent to its double negation, but this is not so in the predicate calculus. Call a formula *strongly classical* in case it is classical and each occurrence in it of an atomic formula is doubly negated. Every formula is classically equivalent to a strongly classical formula. In part (d) of Theorem 60 in [Kl, §81], Kleene modifies the Gödel reduction so that the algorithm transforms a classical proof of a strongly classical formula into an intuitionistic proof of the same formula.

Roughly speaking, Kripke's analysis involves stages of investigation forming a tree where at each stage a classical semantics is imposed, with the proviso that if a formula is true at one stage it remains true at all later stages but if a formula is false at one stage it may become true at a later stage; a formula is valid in case it is true at the root of the tree in every structure. His completeness theorem establishes that a formula is valid if and only if it is provable in Heyting's predicate calculus (the "if" part being trivial).

Theorem 2: The classical and Kripke semantics are identical on strongly classical formulas.

Proof: Suppose that a closed strongly classical formula C is classically valid. By Gödel's completeness theorem for classical logic [Gö30], it is provable in the classical predicate calculus. By the reduction of classical proofs to intuitionistic proofs (Kleene's modification of Gödel's algorithm), it is provable in Heyting's intuitionistic predicate calculus, and so is valid in Kripke's intuitionistic semantics. Conversely, by Kripke's completeness theorem a formula valid in his semantics is provable in Heyting's intuitionistic predicate calculus, and therefore in the classical predicate calculus, and hence is classically valid. ∎

Kripke semantics leads to the same conclusions regarding the ontological Platonic nature of intuitionistic semantics, and intuitionism as an extension of classical mathematics, as those reached on the basis of Kleene's realization predicate. When an intuitionist states that a classically valid strongly classical formula is not constructively valid (as is frequently stated for the strongly classical translation of the axiom of choice), he implicitly states that the Kripke semantics is an inadequate description of intuitionistic semantics.

6. Epistemic semantics

Many writers on intuitionism base intuitionistic semantics on epistemology; on knowledge and proof. When Kleene [Kl, §82] introduces realization, he avoids all reference to the notion of proof. But in the informal discussion in §81, he writes (my italics):

> The intuitionists' use of negation and implication must then be understood as only requiring us to recognize, e.g., that a particular given *proof* is intuitionistically acceptable, or (when they *prove* a statement of the form $(A \to B) \to C$) that if one should produce an intuitionistically acceptable *deduction* of one statement B from another A, then on the basis of it one could by a given method surely construct an intuitionistically acceptable *proof* of a third C.

Heyting [He31] [BePu] says (again my italics):

> It is important to note that the negation of a proposition always refers to a *proof procedure* which leads to the contradiction, even if the original proposition mentions no proof procedure.

Dummett [Du] [BePu] confronts the problem directly:

> However, the distinction [between the notions of truth and assertability] is unavoidable if the explanations of universal quantification, implication and negation are to escape circularity. The standard explanation of implication is that a proof of $A \to B$ is a construction of which we can recognise that, applied to any proof of A, it would yield a proof of B.
>
> ...if the intuitionistic explanation of implication is to escape, not merely circularity, but total vacuousness, there must be a restricted notion of proof – canonical proof – in terms of which the explanation is given...
>
> The notion of canonical proof thus lies in some obscurity; and this state of affairs is not indefinitely tolerable, because, unless it is possible to find a coherent and relatively sharp explanation of the notion, the viability of the intuitionist explanations of the logical constants must remain in doubt.

Thus the standard explanation of the intuitionistic meaning of the logical constants is based on the notion of proof. Let us call this *epistemic semantics*. How can it be formalized?

Knowledge is most conveniently expressed in the first person singular, but we are interested only in communicable knowledge, so the task is to describe how I can communicate my knowledge to you, and what it is about my knowledge that will convince you once it has been communicated. Knowledge is closely related to syntactical questions, to the notion of proof. How else can I communicate my knowledge to you, other than by an appeal to a shared vision of a common ontology or by a demonstration? We have already discussed the former; the question now is whether it can be dispensed with in explaining the intuitionistic meaning of the logical constants.

Let us grant, for purposes of argument, that mathematical reasoning cannot be fully formalized. But consider all closed formulas of Arithmetic with l or fewer logical constants. There are only finitely many of them, modulo the infinity of atomic formulas. So if I am to communicate to you in a convincing way my knowledge of the truth of such a formula without appealing to ontology, it must be possible to formalize the argument in a formal system T_l. This system must be axiomatic, so that there is an effective method for deciding whether a putative proof is indeed a proof.

Consider a closed formula C of $\mathcal{L}$. If C is atomic, $a = b$, I communicate my knowledge by performing the computation and showing that a and b reduce to the same numeral. For a conjunction $A \mathbin{\&} B$, I communicate my knowledge of A and my knowledge of B. For $\exists xA$, I exhibit a numeral n and communicate my knowledge of $A_x(n)$, and for a disjunction $A \vee B$ I choose one of the disjuncts and communicate my knowledge of it.

But for the input operators $\forall$ and $\rightarrow$ there is a problem. The realization predicate for them was expressed by means of quantifications over infinite domains. This makes no sense epistemically; I cannot produce an infinitely long communication, not even introspectively. If I know $\forall xA$ without appeal to ontology, I must know it on the basis of a general argument, and that argument must be formalizable in T_l. This formalization requires an arithmetization, treating $\forall xA$ as an entity of T_l rather than as a formula. But it is neither necessary nor sufficient to prove in T_l the arithmetization $\ulcorner \forall xA \urcorner$; what I must do is prove in T_l an arithmetization of my knowledge of $A_x(n)$ for a general numeral n, something like $\ulcorner n$ is a numeral *implies* $K(n)$ is my knowledge that $A_x(n)\urcorner$, and similarly for implications $A \rightarrow B$. Thus we must also require that T_l be strong enough to arithmetize itself; we shall take each T_l to be an axiomatic extension of S.

We need some function symbols and predicate symbols in T_l. The function symbols have default value 0 if the arguments do not match the descriptions given. Let $\pi_1 x$ and $\pi_2 x$ be the first and second members of the pair x and let $\operatorname{Arg}(x, y)$ be the xth argument of the arithmetized formula y. Let "numeral(x)" express that x is a numeral in the arithmetized theory. Let "proof(x, y)" express that y is an arithmetized formula of T_l and x is an arithmetized proof in T_l of it; the dependence on T_l is not indicated in the notation.

Let C be a closed formula of $\mathcal{L}$. We introduce, by recursion on the complexity of C, a formula c *vf* C of T_l. This is read as c *verifies* C and its intended meaning is that c is a communication of my knowledge that C is intuitionistically true.

(V1) Let C be atomic. Then c vf C is C.

(V2) Let C be $A \mathbin{\&} B$. Then c vf C is $\pi_1 c$ vf $A \mathbin{\&} \pi_2 c$ vf B.

(V3) Let C be $\forall xA$. Then c vf C is $\operatorname{proof}(\pi_1 c, \ulcorner\operatorname{numeral}(w) \rightarrow \pi_2$ vf $A_x(w)\urcorner)$.

(V4) Let C be $A \rightarrow B$. Then c vf C is $\operatorname{proof}(\pi_1 c, \ulcorner z$ vf $A \rightarrow \pi_2 c$ vf $B\urcorner)$.

(V5) Let C be $\exists xA$. Then c vf C is $\pi_2 c$ vf $A_x(\pi_1 c)$.

(V6) Let C be $A \vee B$. Then c *vf* C is $\pi_2 c$ *vf* $\mathrm{Arg}(\pi_1 c, C)$.

In (V3) and (V4), w and z are variables not occurring previously and $\pi_2 c$ may, of course, depend on them.

For a general formula C of $\mathcal{L}$, let c *vf* C be c *vf* $\overline{C}$, where $\overline{C}$ is the closure of C. We say that C is *verifiable* in case there is a consistent axiomatic extension T_l of S and a variable-free term c of T_l such that $\vdash c \text{ vf } C$ (where $\vdash$ means $\vdash_{T_l}$).

For fixed C the formula x *vf* C represents a recursive predicate, so for each variable-free term c we have $\vdash c \text{ vf } C \vee \neg\, c \text{ vf } C$, and the occurrences of $\to$ can be eliminated in favor of $\neg \ldots \vee \ldots$. Thus epistemic semantics explains the intuitionistic meanings of the quantifier symbols, the constructive $\vee$, and the much-debated $\to$ in simpler terms.

But now we must ask whether this semantics is adequate for intuitionistic mathematics. The answer is no.

Theorem 3. $\neg\neg\, 0 = 0$ is unverifiable.

Proof: Suppose that $\vdash c \text{ vf } \neg\neg\, 0 = 0$, where c is a variable-free term. That is,

$$\vdash c \text{ vf } [\neg\, 0 = 0 \to \mathrm{S}0 = 0].$$

By (V4),

$$\vdash \mathrm{proof}(\pi_1 c, \ulcorner [z \text{ vf } \neg\, 0 = 0] \to [\pi_2 c \text{ vf } \mathrm{S}0 = 0] \urcorner).$$

Now $\pi_2 c$ *vf* $\mathrm{S}0 = 0$ is $\mathrm{S}0 = 0$ by (V1), so

$$\vdash \mathrm{proof}(\pi_1 c, \ulcorner [z \text{ vf } \neg\, 0 = 0] \to [\mathrm{S}0 = 0] \urcorner).$$

That is,

$$\vdash \mathrm{proof}(\pi_1 c, \ulcorner \neg\, [z \text{ vf } \neg\, 0 = 0] \urcorner),$$

which is

$$\vdash \mathrm{proof}\big(\pi_1 c, \ulcorner \neg\, \big[z \text{ vf } [0 = 0 \to \mathrm{S}0 = 0]\big] \urcorner\big).$$

By (V4),

$$\vdash \mathrm{proof}\big(\pi_1 c, \ulcorner \neg\, \mathrm{proof}\big(\pi_1 z, \ulcorner [w \text{ vf } 0 = 0] \to [\pi_2 z \text{ vf } \mathrm{S}0 = 0] \urcorner\big) \urcorner\big),$$

which by (V1) is

$$\vdash \mathrm{proof}\big(\pi_1 c, \ulcorner \neg\, \mathrm{proof}\big(\pi_1 z, \ulcorner [w \text{ vf } 0 = 0] \to [\mathrm{S}0 = 0] \urcorner\big) \urcorner\big).$$

That is,

$$\vdash \mathrm{proof}\big(\pi_1 c, \ulcorner \neg\, \mathrm{proof}(\pi_1 z, \ulcorner \neg\, w \text{ vf } 0 = 0 \urcorner) \urcorner\big).$$

Let u be a new variable and let d be $\pi_1 c$ followed by the (arithmetized) proof of the instance obtained by substituting $\langle u, 0\rangle$ for z and proving $\pi_1\langle u, 0\rangle = u$. Then

$$\vdash \operatorname{proof}\bigl(d, \ulcorner\neg \operatorname{proof}(u, \ulcorner\neg\, w \ \mathit{vf}\ 0 = 0\urcorner)\urcorner\bigr).$$

Let A be $\neg \operatorname{proof}(u, \ulcorner\neg\, w \ \mathit{vf}\ 0 = 0\urcorner)$. Then d is a variable-free term of T_l and we have $\vdash \operatorname{proof}(d, \ulcorner A\urcorner)$. Since $\operatorname{proof}(x, y)$ represents a recursive predicate, we have a proof in T_l of A; that is, we have a proof in T_l of

$$\neg \operatorname{proof}(u, \ulcorner\neg\, w \ \mathit{vf}\ 0 = 0\urcorner).$$

Thus we have a proof in T_l that a certain arithmetized formula is unprovable; that is, we have a self-consistency proof for T_l. But T_l is a consistent axiomatized extension of S, so this is impossible by the modern form of Gödel's second theorem [Gö31]. ▪

The trouble lies in the propositional calculus. Modus ponens, the rule of inference (25) of the propositional calculus, preserves verifiability. For suppose that T_l is a consistent axiomatic extension of S, and $\vdash a \ \mathit{vf}\ A$ and $\vdash c \ \mathit{vf}\ A \to B$. Then $\vdash \operatorname{proof}(\pi_1 c, \ulcorner z \ \mathit{vf}\ A \to \pi_2 c \ \mathit{vf}\ B\urcorner)$. Since $\vdash a \ \mathit{vf}\ A$ we can arithmetize the proof and substitute a for z. This gives us a variable-free term d such that $\vdash \operatorname{proof}(d, \ulcorner \pi_2 c \ \mathit{vf}\ B\urcorner)$. Since $\operatorname{proof}(x, y)$ represents a recursive predicate, we have $\vdash \pi_2 c \ \mathit{vf}\ B$.

Using Nelson's realization proofs as templates, we can establish that the propositional axioms (13)–(22) are verifiable, with one exception. The exception is (14),

$$[A \to B] \to [A \to B \to C] \to A \to C$$

and its special case (21).

This is used, together with (13), to justify deduction. Consider a deduction, introducing a hypothesis A. So we suppose that for some a we have $a \ \mathit{vf}\ A$. Then we establish B, with $b \ \mathit{vf}\ B$, and $B \to C$, with $e \ \mathit{vf}\ B \to C$. We want to find a c such that $c \ \mathit{vf}\ C$. Why can't we do this by modus ponens? The proof that modus ponens preserves verification used the fact that if $\operatorname{proof}(d, \ulcorner D\urcorner)$ for a variable-free term d, then $\vdash D$. But to prove this for a variable d, as would be necessary in a deduction, we would need to prove in T_l the consistency of T_l, and this we cannot do.

When an intuitionist makes a deduction, introducing and discharging a hypothesis, he implicitly reifies a hypothetical situation, projecting it onto an abstract ontology.

Epistemic semantics is adequate for mathematics shorn of the axiom scheme (14), with deductions not permitted. But this drastic prohibition would indeed deprive the boxer of the use of his fists.

7. Conclusions

This investigation has led to the following conclusions. Intuitionism is an extension of classical mathematics, both in its proof syntax and semantically. Intuitionistic semantics, being an extension of classical semantics, is ontological and Platonic. Attempts to give an epistemic basis for intuitionistic semantics, to explain the intuitionistic meaning of the logical constants in terms of proof, are not viable. What is genuinely new in intuitionism is Brouwer's creation of two new logical constants, the constructive $\exists$ and the constructive $\vee$, together with a rich notion of truth containing much constructive information. But so far intuitionism has not proved to be a powerful tool in constructive mathematics.

This is a somewhat negative assessment of intuitionism, but I want to end on a constructive note. Since the advent of digital computers, attention has turned from effective methods—functions computable in principle—to feasible algorithms and programs. There is strong evidence that polynomial time functions provide the correct formalization of the intuitive notion of a feasible computation, and unlike the situation for recursive functions there is a purely syntactical characterization [BeCo] [Be] [Le] of polynomial time functions. I am convinced that intuitionism reformulated in this context will become a powerful practical method for constructing and verifying feasible algorithms, and that Kleene's realization predicate will provide an incisive tool for analyzing problems concerning interactive programs.

References

[Be] Stephen J. Bellantoni, *Predicative Recursion and Computational Complexity*, Technical Report 264/92, Department of Computer Science, University of Toronto, 1992. Available on-line at
ftp://ftp.cs.toronto.edu/pub/reports/theory/cs-92-264.ps.Z

[BeCo] — and Stephen Cook, *A new recursion-theoretic characterization of the polytime functions*, Computational Complexity, vol. 2, pp. 97–110, 1992.

[BePu] Paul Benacerraf and Hilary Putnam, eds., *Philosophy of Mathematics: Selected readings*, 2nd ed., Cambridge University Press, Cambridge, 1983.

[Br] L. E. J. Brouwer, *Die onbetrouwbaarheid der logische principes*, Tijdschrift voor wijsbegeerte, vol. 2, 1908.

[Du] Michael Dummett, *The philosophical basis of intuitionistic logic*, Proceedings of the Logic Colloquium, Bristol, July 1973, ed. H. E. Rose and J. C. Shepherdson, pp. 5–40, North-Holland, Amsterdam, 1975. Reprinted in [BePu].

[Gö30] Kurt Gödel, *Die Vollständigkeit der Axiome des logischen Funktionenkalküls*, Monatshefte für Mathematik und Physik, vol. 37, pp. 349–360, 1930.

[Gö31] —, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, Monatshefte für Mathematik und Physik, vol. 38, pp. 173–198, 1931.

[Gö33] —, *Zur intuitionistischen Arithmetik und Zahlentheorie*, Ergebnisse eines mathematischen Kolloquiums, Heft 4, pp. 34–38, 1933.

[He] Arend Heyting, *Die formalen Regeln der intuitionistischen Logik*, Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse, pp. 42–56, 1930.

[He31] —, *Die intuitionistische Grundlegung der Mathematik*, Erkenntnis, vol. 2, pp. 106–115, 1931. English translation as *The intuitionist foundations of mathematics* in [BePu].

[Hö] L. Hörmander, *On the theory of general partial differential operators*, Acta. Math., vol. 94, pp. 161–248, 1955.

[Kl] S. C. Kleene, *Introduction to Metamathematics*, North-Holland, Amsterdam, 1971. First published 1952.

[Kl45] —, *On the interpretation of intuitionistic number theory*, Jour. Symbolic Logic, vol. 10, pp. 109–124, 1945.

[Kr] Saul A. Kripke, *Semantical analysis of intuitionistic logic I*, Formal Systems and Recursive Functions, Proceedings of the Eighth Logic Colloquium, Oxford, July 1963, ed. J. N. Crossley and M. A. E. Dummett, pp. 93–130, North-Holland, Amsterdam, 1965.

[Le] Daniel Leivant, *Ramified recurrence and computational complexity I: Word recurrence and poly-time*, in Feasible Mathematics II, ed. Peter Cole and Jeffrey Remmel, pp. 320–343, Perspectives in Computer Science, Birkhauser-Boston, New York, 1994.

[M-Lö] Per Martin-Löf, *Intuitionistic Type Theory*, Bibliopolis, Naples, 1984.

[Ne] David Nelson, *Recursive functions and intuitionistic number theory*, Trans. Amer. Math. Soc., vol. 61, pp. 307–368, 1947.

[NoPeSm] Bengt Nordström, Kent Petersson, Jan M. Smith, *Programming in Martin-Löf's Type Theory: An Introduction*, The International Series of Monographs on Computer Science 7, Clarendon Press, Oxford, 1990.

[Ta] Alfred Tarski, *A Decision Method for Elementary Algebra and Geometry*, Berkeley, 1951.