FEASIBLY CONSTRUCTIVE PROOFS AND THE PROPOSITIONAL CALCULUS

Preliminary Version

Stephen A. Cook
University of Toronto

1. Introduction

The motivation for this work comes from two general sources. The first source is the basic open question in complexity theory of whether *P* equals *NP* (see [1] and [2]). Our approach is to try to show they are not equal, by trying to show that the set of tautologies is not in *NP* (of course its complement is in *NP*). This is equivalent to showing that no proof system (in the general sense defined in [3]) for the tautologies is "super" in the sense that there is a short proof for every tautology. Extended resolution is an example of a powerful proof system for tautologies that can simulate most standard proof systems (see [3]). The Main Theorem (5.5) in this paper describes the power of extended resolution in a way that may provide a handle for showing it is not super.

The second motivation comes from constructive mathematics. A constructive proof of, say, a statement $\forall x A$ must provide an effective means of finding a proof of A for each value of x, but nothing is said about how long this proof is as a function of x. If the function is exponential or super exponential, then for short values of x the length of the proof of the instance of A may exceed the number of electrons in the universe. Thus one can question the sense in which our original "constructive" proof provides a method of verifying $\forall x A$ for such values of x. Parikh [4] makes similar points, and goes on to suggest an "anthropomorphic" formal system for number theory in which induction can only be applied to formulas with bounded quantifiers. But even a quantifier bounded by n may require time exponential in the length of (the decimal notation for) n to check all possible values of the quantified variable (unless *P* = *NP*), so Parikh's system is apparently still not feasibly constructive.

In section 2, I introduce the system PV for number theory, and it is this system which I suggest properly formalizes the notion of a feasibly constructive proof. The formulas in PV are equations $t = u$, (for example, $x \cdot (y+z) = x \cdot y + x \cdot z$) where t and u are terms built from variables, constants, and function symbols ranging over *L*, the class of functions computable in time bounded by a polynomial in the length of their arguments. The system PV is the analog for *L* of the quantifier-free theory of primitive recursive arithmetic developed by Skolem [5] and formalized by others (see [6]). A result necessary for the construction of the system is Cobham's theorem [7] which characterizes *L* as the least class of functions containing certain initial functions, and closed under substitution and limited recursion on notation (see section 2). Thus all the functions in *L* (except the initial functions) can be introduced by a sequence of defining equations. The axioms of PV are these defining equations, and the rules of PV are the usual rules for equality, together with "induction on notation".

All proofs in PV are feasibly constructive in the following sense. Suppose an identity, say $f(x) = g(x)$, has a proof Π in PV. Then there is a polynomial $p_\Pi(n)$ such that Π provides a uniform method of verifying within $p_\Pi(|x_0|)$ steps that a given natural number x_0 satisfies $f(x_0) = g(x_0)$. If such a uniform method exists, I will say the equation is polynomially verifiable (or p-verifiable).

The reader's first reaction might be that if both f ang g are in *L*, then there is always a polynomial p(n) so that the time required to evaluate them at x_0 is bounded by $p(|x_0|)$, and if $f(x) = g(x)$ is a true identity, then it should be p-verifiable. The point is that the verification method must be uniform, in the sense that one can see (by the proof Π) that the verification will always succeed. Not all true identities are provable, so not all are p-verifiable.

There is a similar situation in constructive (or intuitionistic) number theory. The Kleene-Nelson theorem ([8], p. 504) states that if a formula $\forall x A$ has a

constructive proof, then it is recursively realizable in the sense that there is a recursive function f which takes x_0 into a proof of $A\frac{x_0}{x}$ (more properly, $f(x_0)$ is a number which "realizes" $A\frac{x_0}{x}$). The converse is false. One can find a formula $\forall xA$ which is recursively realizable, but not constructively provable, since one cannot prove that the realizing recursive function works. Similarly, any true equation $f(x) = g(x)$ in PV is recursively realizable (in fact, L-realizable), but not all are p-verifiable (i.e. have feasibly constructive proofs).

I argue in section 2 that provable equations in PV are p-verifiable. I also conjecture the converse is true, which leads to

1.1 Verifiability thesis. An equation $t = u$ of PV is provable in PV if and only if it is p-verifiable.

This statement is similar to Church's thesis, in that one can never prove that PV is powerful enough, since the notion of p-verifiable is informally defined. We present evidence for the power of PV in this paper by giving examples of things that are provable in PV, and by presenting the system PV1 in section 3 which appears to be more powerful than PV, but isn't.

Another argument for the power of PV that can be made is this. There is evidence that intuitionistic number theory, as formalized by Kleene [8], is equivalent to a quantifier-free theory in which functions are introduced by ordinal recursion up to ϵ_0. From this point of view, PV is the same kind of quantifier-free theory, except the kind of recursion allowed is restricted so that only functions in L can be defined.

In section 2, the system PV is described in detail, and some simple examples of proofs in the system are given. The Valuation Theorem (2.18) states that all true equations in PV without variables are provable in PV.

In section 3, the system PV1 is presented. This system allows formulas to be truth functional combinations of equations, instead of just equations, and is much more convenient than PV for formalizing proofs. Nevertheless, theorem 3.10 states that any equation provable in PV1 is provable in PV.

The second Gödel Incompleteness theorem for PV, stating that the consistency of PV cannot be proved in PV, is proved in outline in section 4. I am aware of only one other treatment in the literature of this theorem for a free-variable system, and that is in [9]. (However, there seems to be a mistake in [9], since theorem 16, p. 134 fails when $f(s(x))$ is neither identically zero nor identically non-zero.)

In section 5, the proof system extended resolution is described, and the notion of a p-verifiable proof system for the propositional calculus is defined. The Main Theorem (5.5) states that a proof system f for the propositional calculus is p-verifiable iff extended resolution can simulate f efficiently, and the proof that the simulation works can be formalized in PV. The "if" part is proved in outline.

Section 6 describes how to develop propositional formulas which express the truth of equations $t = u$ of PV for bounded values of the variables in t and u. The ER Simulation Theorem (6.8) states that if $t = u$ is provable in PV, then there is a polynomial (in the length of the bound on the variables) bound on the length of the minimal extended resolution proofs of the associated propositional formulas. The "only if" part of the Main Theorem is then proved in outline from this.

In section 7, it is shown how the Gödel Incompleteness theorem implies that the system PV, as a proof system for the propositional calculus, is not itself p-verifiable.

Finally, section 8 offers some conclusions and directions for future research.

2. The System PV

I will use dyadic notation (see Smullyan [10]) to denote natural numbers.† The dyadic notation for the natural number n is the unique string $d_k d_{k-1} \ldots d_0$ over the alphabet {1,2} such that $\sum_{i=0}^{k} d_i 2^i = n$. In particular, the dyadic notation for 0 is the empty string. The dyadic successor functions $s_1(x)$ and $s_2(x)$ are defined by $s_i(x) = 2x + i$, $i = 1,2$, and correspond to concatenating the digits 1 and 2, respectively, on the right end of the dyadic notation for x. I shall thus abbreviate $s_i(x)$ by xi.

A function f comes from functions $g_1,\ldots,g_m$ by the operation of substitution iff some equation of the form

2.1 $$f(x_1,\ldots,x_n) = t$$

holds for all $x_1,\ldots,x_n$, where t is a

† The trouble with the more conventional binary notation is the necessity of proving the consistency of the analogs of equations 2.2 and 2.3 when $x = i = 0$.

syntactically correct term built up from the variables $x_1,\ldots,x_n$, numerals for the natural numbers, and the function symbols $g_1,\ldots,g_m$.

A function f comes from functions g,h_1,h_2,k_1,k_2 by the operation of limited recursion on dyadic notation iff

2.2 $$f(0,\bar{y}) = g(\bar{y})$$

2.3 $$f(xi,\bar{y}) = h_i(x,\bar{y},f(x,\bar{y})), \quad i = 1,2$$

2.4 $$f(x,\bar{y}) \le k_i(x,\bar{y}), \quad i = 1,2$$

for all natural number values of the variables, where $\bar{y} = (y_1,\ldots,y_k)$. We allow the case k = 0, in which g is a constant.

Cobham's class *L* can be defined to be the set of functions f on the natural numbers such that for some Turing machine Z and some polynomial p, for all natural numbers $x_1,\ldots,x_n$, Z computes $f(x_1,\ldots,x_n)$ within $p(|x_1|+\ldots+|x_n|)$ steps, where $|x|$ is the length of the dyadic notation for x.

2.5 Definition. The dyadic notation for $\circledast(x,y)$ is the dyadic notation for x concatenated with itself $|y|$ times.

2.6 Theorem (Cobham). *L* is the least class of functions which includes the initial functions s_1, s_2, and $\circledast$, and which is closed under the operations of substitution and limited recursion on dyadic notation.

Cobham stated this result in [7], in a slightly different form. I am not aware of any published proof of the theorem, although Lascar gave a proof in some unpublished seminar notes [11].

The formal system PV will have function symbols with defining equations of the forms 2.1, 2.2, and 2.3. I want only functions in *L* to be definable in PV, which means the inequalities 2.4 must be satisfied for some functions k_1,k_2 in *L*. It is not hard to see that the question, given g,h_1,h_2,k_1,k_2, of whether the function f defined by 2.2 and 2.3 satisfies 2.4 is recursively undecidable. I want, however, for the proof predicate in PV to be not only decidable, but definable in PV. Therefore, I shall require that before a function f can be introduced by 2.2 and 2.3, a proof must be available in PV that f does not grow too fast. It is awkward to require that 2.4 be proved directly in PV, because it obviously cannot be proved without using f, whose status in PV is uncertain until after the proof is carried out. Thus the proof will instead verify the inequality

$$|h_i(x,\bar{y},z)| \le |z*k_i(x,\bar{y})|, \quad i = 1,2$$

for some previously defined functions k_1 and k_2 (not those in 2.4), where * indicates concatenation. It is easy to see that this inequality guarantees that f is in *L* if k_1 and k_2 are in *L*, since then $|f(x,\bar{y})| \le |g(\bar{y})| + |k(0,\bar{y})| + |k(d_1,\bar{y})| + \ldots + |k(d_1\ldots d_k,\bar{y})|$ where $d_1\ldots d_{k+1}$ is the dyadic notation for x and $k(x,\bar{y}) = k_1(x,\bar{y}) + k_2(x,\bar{y})$.

In order to specify formally what constitutes a proof of this inequality, we must introduce enough initial functions in PV to define the relation $|x| \le |y|$. Thus we introduce a function TR(x) (TR for "trim") which deletes the right-most digit of x. From this, a function LESS(x,y) can be defined whose value is x with the right-most $|y|$ digits deleted. Thus $|x| \le |y|$ iff LESS(x,y) = 0. In addition, we need * (concatenation) as an initial function, and also $\circledast$ (see 2.5). The purpose of $\circledast$ is to allow formation of functions in PV by composition which grow sufficiently fast to dominate any function in *L*.

Function symbols in PV will be defined later to be certain strings of symbols which encode the complete derivation from initial functions of the function they stand for. In particular, the defining equation(s) and number of arguments (arity) for a function symbol can be determined by inspection from the symbol.

The set of terms of PV is defined inductively as follows. (i) 0 is a term, any variable x is a term, and any function symbol f of arity 0 is a term. (ii) If $t_1,\ldots,t_k$ are terms, and f is a function symbol of arity $k \ge 1$, then $f(t_1,\ldots,t_k)$ is a term. An equation is a string of the form t = u, where t and u are terms. A derivation in PV of an equation E from equations $E_1,\ldots,E_n$ is a string of equations of the form $D_1,\ldots,D_\ell$, such that D_ℓ is E, and each D_i, $1 \le i \le \ell$, is either some E_j, a defining equation for a function symbol, or follows from earlier equations in the string by a rule of PV (see below). If such a derivation exists, we shall write $E_1,\ldots,E_n \vdash_{PV} E$, or simply $\vdash_{PV} E$, if there are no hypotheses (the symbol PV here will sometimes be deleted). A derivation of E from no hypotheses is a proof of E.

RULES OF PV

(Here t,u,v are any terms, x is a variable, and $\bar{y}$ is a k-tuple of variables, $k \ge 0$.)

R1. $t=u \vdash u=t$

R2. $t=u,\ u=v \vdash t=v$

R3. $t_1=u_1,\ldots,t_k=u_k \vdash f(t_1,\ldots,t_k) = f(u_1,\ldots,u_k)$,
for any k-place function symbol f, $k\geq 1$.

R4. $t=u \vdash t\frac{v}{x} = u\frac{v}{x}$,
where $\frac{v}{x}$ indicates substitution of the term v for the variable x.

R5. (Induction on notation) $E_1,\ldots,E_6 \vdash f_1(x,\bar{y}) = f_2(x,\bar{y})$,
where $E_1,\ldots,E_6$ are the equations 2.2 and 2.3 with f replaced by f_1 and by f_2.

The definition of proof is not yet complete, because the notion of function symbol (and hence of term and equation) and associated defining equations has not yet been specified. These notions must actually be defined inductively simultaneously with the definition of proof, because of our requirement that the boundedness of functions be proved in PV. The arity of a function symbol is the number of arguments, and the order of the symbol is roughly the depth of nesting of recursion on notation used to define it. We define the order of a proof to be the greatest of the orders of the function symbols occurring in it. Now we can complete the definitions of all these notions simultaneously and recursively as follows:

The initial function symbols all have order 0. These are the symbol 0 (of arity 0), s_1,s_2,TR (each of arity 1) and *,⊛, LESS (each of arity 2). There are no defining equations for 0, s_1 and s_2, and the defining equations for the others are (here x1 means $s_1(x)$, x_2 means $s_2(x)$):

TR: $TR(0) = 0$
$TR(xi) = x, \quad i = 1,2$

*: $*(x,0) = x$
$*(x,yi) = s_i(*(x,y)), \quad i = 1,2$

⊛: $\circledast(x,0) = 0$
$\circledast(x,yi) = *(x,\circledast(x,y)), \quad i = 1,2$

LESS: $LESS(x,0) = x$
$LESS(x,yi) = TR(LESS(x,y)), \quad i = 1,2$

Note: We use infix notation for * and ⊛ after this.

It t is a term, and k is the maximum of the orders of the function symbols occurring in t, and all variables in t are among the variables $x_1,\ldots,x_n$, $n\geq 0$, then $\lambda x_1\ldots x_n t\rho$ is a function symbol of arity n and order k. The defining equation is $f(x_1,\ldots,x_n) = t$, if $n \geq 1$, and $f = t$, if $n = 0$, where f is $\lambda x_1\ldots x_n t\rho$.

If g,h_1,h_2,k_1,k_2 are function symbols of arity n-1, n+1, n+1, n, and n, respectively, ($n\geq 1$) and if k is the maximum of the orders of the five function symbols, and if Π_i, i=1,2 are proofs of order k or less of $LESS(h_i(x,\bar{y},z),z*k_i(x,\bar{y})) = 0$, i=1,2, then $\langle[g,h_1,h_2,k_1,k_2][\Pi_1][\Pi_2]\rangle$ is a function symbol of arity n and order k+1. If f denotes this function symbol, then the three defining equations for f are

$$f(0,\bar{y}) = g(\bar{y}) \quad (\text{or } f(0) = g, \text{ if } n = 1)$$

$$f(xi,\bar{y}) = h_i(x,\bar{y},f(x,\bar{y})), \quad i = 1,2$$

All function symbols must be formed in these ways. This completes the formal specification of the system PV.

As examples of proofs on PV, let us verify some simple properties of LESS or TR.

2.7 $\quad \vdash_{PV} TR(LESS(xi,y)) = LESS(x,y), \quad i = 1,2$

The strategy is to use R5 (induction on notation). To do this we introduce a new function symbol f (formally, f is $\lambda xyTR(LESS(yi,x))\rho$) with defining equation $f(x,y) = TR(LESS(yi,x))$. Also a function symbol LESS' is introduced with defining equation $LESS'(x,y) = LESS(y,x)$. Now the hypotheses of the induction rule can be verified, when f_1 is f and f_2 is LESS', and $g(y) = y$, and $h_j(x,y,z) = TR(z)$; j=1,2. Hence $f(x,y) = LESS'(x,y)$, from which 2.7 follows by R1, R2, and R4, and the defining equations for f and LESS'.

2.8 $\quad \vdash_{PV} LESS(x,x) = 0$

This is shown by induction on x, using 2.7 with y replaced by x.

2.9 $\quad \vdash_{PV} LESS(x,yi*z) = TR(LESS(x,y*z)), \quad i = 1,2$

This is proved by induction on z. Here $h_j(x,y,z,u) = TR(u)$.

2.10 $\quad \vdash_{PV} LESS(x,y*z) = LESS(x,z*y)$

Again this is proved by induction on y, using 2.9, and the same function h_j above.

2.11 $\quad \vdash_{PV} LESS(x,x*y) = 0$

The proof is induction on y, using 2.8.

The intended semantics of PV should be clear. Every function symbol f stands for a uniquely defined function in L, which we can denote by $\phi(f)$. (The reader can give a precise definition of $\phi(f)$ by induction on the length of the function symbol f.) An equation t = u in PV is true iff its universal closure is true in the domain of natural numbers, when all function symbols receive their standard interpretations.

We say a function F on the natural numbers is definable in PV iff $\phi(f) = F$ for some function symbol f of PV. By Cobham's theorem, every function definable in PV is clearly in L, but the converse is far from obvious, because of our requirement that the bounding inequalities be provable in PV. Nevertheless, the converse is true.

2.12 Theorem. Every function in L is definable in PV.

To prove this requires a reproving of half of Cobham's theorem, showing that the functions introduced by limited recursion on notation can have their bounding inequalities proved in PV. We will not give the argument here.

Below we introduce two functions in PV which we will use in the next section. The defining equations given do not strictly fit the format for recursion on notation, since the function symbols g, h_1, h_2, k_1, k_2 would have to be introduced explicitly. However, the reader should have no trouble doing this.

Note: $s_1(0)$ is abbreviated by 1, and $s_2(0)$ is abbreviated by 2.

2.13 $\overline{sg}(0) = 1$

$\overline{sg}(xi) = 0, \quad i = 1,2$

2.14 $sg(0) = 0$

$sg(xi) = 1, \quad i = 1,2$

2.15 $CON(0,y) = 0$

$CON(xi,y) = sg(y)$

The bounding inequalities for the above three functions are easily proved in PV from the defining equations for LESS and TR.

We now wish to argue in support of one part of the Verifiability Thesis (1.1), namely that only p-verifiable equations are provable in PV. Our argument includes an outline of a highly constructive consistency proof for PV, and it could be formalized in, say, primitive recursive number theory, to show there is no proof in PV of 0 = 1. An indication of how a similar argument showing the consistency of elementary arithmetic (in the sense of Kalmar) could be carried out in primitive recursive arithmetic was given in Rose [12].

2.16 Proposition. If $\vdash_{PV} t = u$, then the equation t = u is p-verifiable.

2.17 Corollary. Not $\vdash_{PV} 0 = 1$.

Our argument for establishing 2.16 proceeds by induction on the length of the proof of t = u (here length counts the length of the function symbols in the proof). Thus suppose $\ell \geq 1$, and the proposition holds for all proofs of length $< \ell$. Let Π be a proof of t = u of length ℓ. If t = u is a defining equation for a function symbol f, then the equation holds by definition of f. However, the time required to verify the equation for a particular value of the arguments is equal to the time to compute f at that value, so we must be sure that this computation time is bounded by a polynomial in the length of the arguments. Here we apply the induction hypothesis, both to be sure that f does not grow too fast (we know this partly because if f is defined by recursion, then there are proofs of length less than ℓ establishing a bound on the growth rate) and that all functions used in defining f can be computed in polynomial time.

Now suppose t = u follows from earlier equations in Π by one of the rules R1,..., R5.

We will consider R4 as an interesting example. Thus (changing the roles of t and u to be consistent with the notation of R4) we assume by the induction hypothesis that t = u is p-verifiable, and also that the equations defining the functions in the term v give a polynomial time method of evaluating v. Thus, to verify $t\frac{v}{x} = u\frac{v}{x}$ for particular values for the arguments, we first evaluate v at the argument values, obtaining v_0, and then (using the induction hypothesis) verify t = u at these argument values except we let x have the value v_0. Note that, by the induction hypothesis, we are confident that the equation will hold at the values. Further, since a composition of polynomials is a polynomial, the whole process is bounded in time by a polynomial in the length of the arguments.

We leave the other rules to the reader.

Notice that nothing is said about how the verification time grows with the length of the proof Π. In fact, it is easy to see that the naive bound on this time is at least exponential in the length of Π for fixed argument values, and we will prove in section 7 that PV itself is, in a sense, not p-verifiable.

The final result in this section is the following:

2.18 Valuation Theorem. If $t = u$ is a true equation of PV without variables, then $\vdash_{PV} t = u$.

2.19 Definition. The numeral $\bar{n}$ for the natural number n is the unique term in PV of the form $s_{i_1}(s_{i_2}(\ldots s_{i_k}(0)\ldots)$ whose value is n. In particular, the numeral for 0 is '0'.

2.20 Lemma. Every true equation in PV of the form $f(\bar{n}_1,\ldots,\bar{n}_k) = \bar{m}$ is provable in PV.

First let us note that the valuation theorem follows from the lemma. One shows by induction on the length of t (using the lemma) that if $t = \bar{n}$ is a true equation, then it is provable in PV (rules R1,R2,R3 are all that is needed for this). But $t = \bar{n}, u = \bar{n} \vdash_{PV} t = u$.

The lemma is proved by induction on the length of the function symbol f, where we take the lengths of s_1 and s_2 to be 0, and the lengths of TR, *, ⊛, and LESS to be 1, 2, 3, and 4, respectively. If f is s_1 or s_2, then our task is to show $\vdash_{PV} \bar{m} = \bar{m}$. But the identity function has defining equation $I(x) = x$, from which we may conclude $x = x$ by R1 and R2, and $\bar{m} = \bar{m}$ by R4.

Now suppose f is $\lambda x_1\ldots x_n t\rho$ for some term t. Then the defining equation for f is $f(x_1,\ldots,x_n) = t$. If $f(\bar{n}_1,\ldots,\bar{n}_k) = \bar{m}$ is true, then $t\frac{\bar{n}_1,\ldots,\bar{n}_k}{x_1,\ldots,x_k} = \bar{m}$ is true. Since the induction hypothesis applies to each function symbol in t, the argument made two paragraphs above can be applied to show this last equation is provable in PV. Hence, by R4 and R2, $\vdash_{PV} f(\bar{n}_1,\ldots,\bar{n}_k) = \bar{m}$.

Finally, suppose f is introduced by recursion on notation, so that it has defining equations 2.2 and 2.3. (I intend to include the initial functions TR, *, ⊛, and LESS in this case too.) Then one can see by induction on p that if $f(\bar{p},\bar{n}_1,\ldots,\bar{n}_k) = \bar{m}$ is true, it is provable in PV. (Notice that the main induction hypothesis holds for the function symbols g,h_1,h_2.)

3. The System PV1

The goal now is to construct a system PV1 in which it is easier to formalize proofs than in PV, and then show that every equation provable in PV1 is provable in PV, and conversely.

As a first step, we notice that it is often easier to define a function by simultaneous recursion on several variables at once, rather than on just one variable, as in 2.2 and 2.3. For example, addition is easily defined this way as follows:

$$x + 0 = 0 + x = x$$

$$xi + yj = \begin{cases} (x+y)2 & \text{if } i = j = 1 \\ (s(x+y))1 & \text{if } i \neq j \\ (s(x+y))2 & \text{if } i = j = 2 \end{cases}$$

where $s(x) = x + 1$.

More generally, $f(x,y,\bar{z})$ is defined from $g_{00},g_{01},g_{10},\{h_{ij},k_{ij} | i,j\in\{1,2\}\}$ by limited 2-recursion on dyadic notation iff

3.1 $f(0,0,\bar{z}) = g_{00}(\bar{z})$

3.2 $f(0,yj,\bar{z}) = g_{01}(y,\bar{z})$

3.3 $f(xi,0,\bar{z}) = g_{10}(x,\bar{z})$

3.4 $f(xi,yj,\bar{z}) = h_{ij}(x,y,\bar{z},f(x,y,\bar{z}))$, $i,j \in \{1,2\}$

3.5 $\mathrm{LESS}(h_{ij}(x,y,\bar{z},u),u*k_{ij}(x,y,\bar{z})) = 0$, $i,j \in \{1,2\}$

The reason for using three initial defining equations (3.1, 3.2, 3.3) instead of just two, defining $f(x,0,\bar{z})$ and $f(0,y,\bar{z})$, is to avoid the necessity of proving the consistency of the equations when $x = y = 0$.

3.6 Theorem. Suppose there are function symbols $g_{00},g_{01},g_{10},\{h_{ij},k_{ij} | i,j\in\{1,2\}\}$ in PV such that the four equations 3.5 are each provable in PV. Then there is a function symbol f in PV such that each of the equations 3.1,...,3.4 is provable in PV.

The proof will not be given here.

It is also useful to have a rule allowing induction on notation on several variables at once.

3.7 Theorem. Suppose the equations

3.8 $f(x_1,\ldots,x_n,\bar{y})\frac{0}{x_i} = g_i(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,\bar{y})$, $1 \le i \le n$

3.9 $f(x_1i_1,\ldots,x_ni_n,\bar{y}) = h_{i_1,\ldots,i_n}(x_1,\ldots,x_n,\bar{y},f(x_1,\ldots,x_n,\bar{y}))$, $(i_1,\ldots,i_n) \in \{1,2\}^n$

(2^n+n equations altogether) are each provable in PV when f is replaced by f_1 and again when f is replaced by f_2. Then

$f_1(x_1,\ldots,x_n,\bar{y}) = f_2(x_1,\ldots,x_n,\bar{y})$ is provable in PV.

The proof will not be given here.

The system PV1 can now be defined. Variables, function symbols, terms, and equations are the same as in PV. Formulas in PV1 are either equations, or truth-functional combinations of equations, using the truth-functional connectives $\&,\vee,\neg,\supset,\equiv$. The axioms and axiom schemes of PV1 are the following:

E1. $x=x$

E2. $x=y \supset y=x$

E3. $(x=y\&y=z) \supset x=z$

$(E4)_f$. $(x_1=y_1\&\ldots\&x_k=y_k) \supset f(x_1,\ldots,x_k)=f(y_1,\ldots,y_k)$
for each $k \geq 1$ and each k-place function symbol f in PV

E5. $(x=y) \equiv (xi=yi)$, $i = 1,2$

E6. $\neg x1=x2$

E7. $\neg 0=xi$, $i = 1,2$

DEF. The defining equations for any function symbol in PV are axioms in PV1. Further, the equations 3.1,...,3.4 are axioms in PV1, provided the equations 3.5 are provable in PV1 without using these instances of 3.1-3.4, and provided that the function symbol f is the one given by theorem 3.6. Finally, the defining equations of the initial functions TR, *, ⊛, and LESS, are axioms of PV1.

TAUTOLOGY. Any truth-functionally valid formula of PV1 is an axiom of PV1.

The rules of PV1 are the following:

SUBST. $A \vdash A\frac{t}{x}$, where A is any formula of PV1, t is any term, and x is any variable.

IMP. $A_1,\ldots,A_n \vdash B$, where the formula B is a truth-functional consequence of formulas $A_1,\ldots,A_n$.

n-INDUCTION, $n\geq 1$:

$$\{A\tfrac{0}{x_i} \mid 1\leq i\leq n\},\ \{A\supset A\tfrac{x_1{}^{i_1},\ldots,x_n{}^{i_n}}{x_1,\ldots,x_n} \mid (i_1,\ldots,i_n)\in\{1,2\}^n\} \vdash A$$

For example, 1-induction is the rule

$$A\tfrac{0}{x},\ A \supset A\tfrac{x1}{x},\ A \supset A\tfrac{x2}{x} \vdash A$$

Proofs and derivations in PV1 are described in a way similar to PV.

We use the notation Cl(A) to mean the universal closure of A. We say a formula A of PV1 is true if Cl(A) is true in the domain of natural numbers, when the function symbols receive their standard meanings. The reader is warned that if the terms t and u have variables, then this interpretation means that $\neg t = u$ is not the negation of $t = u$. For example, $sg(x) = 0$ and $\neg sg(x) = 0$ are both false, since their universal closures are both false in the natural numbers.

3.10 Theorem. An equation $t = u$ is a theorem of PV1 if and only if it is a theorem of PV.

The proof is omitted for lack of space.

As a measure of the power and usefulness of the system PV1, we prove the following result.

3.11 Theorem. If $A_1,\ldots,A_n,B$ are formulas in PV1, and Cl(B) can be derived from $Cl(A_1),\ldots,Cl(A_n)$ in the predicate calculus with equality, then $A_1,\ldots,A_n \vdash_{PV1} B$.

Proof. Suppose the hypotheses of the theorem are satisfied. Then Cl(B) is a logical consequence of $Cl(A_1),\ldots,Cl(A_n)$, $Cl(E_1),\ldots,Cl(E_4)$ in the predicate calculus (without the equality axioms). Thus $Cl(A) \supset Cl(B)$ is a quantificationally valid formula, where A is $(A_1\&\ldots\&A_n\&E_1\&\ldots\&E_4)$. By the Herbrand theorem (see [14]), there are substitutions $\sigma_1,\ldots,\sigma_k$ such that

$$3.12 \qquad \bigvee_{i=1}^{k} (A\sigma_i \supset B\tfrac{c_1,\ldots,c_r}{x_1,\ldots,x_r})$$

is truth-functionally valid where $c_1,\ldots,c_r$ are new distinct constant symbols, $x_1,\ldots,x_r$ are the variables occurring in B, and each σ_i is a substitution of "ground" terms (built from $c_1,\ldots,c_r$ and constant symbols of PV by applying function symbols of PV) for the variables in A. If we let σ_i' be the substitution resulting when σ_i is followed by the substitution $\frac{x_1,\ldots,x_r}{c_1,\ldots,c_r}$, then the formula

$$3.13 \qquad \bigvee_{i=1}^{k} (A\sigma_i' \supset B)$$

is "isomorphic" to 3.12, and hence it is also truth-functionally valid. It follows, since 3.13 is a formula of PV1, that it is an axiom of PV1 (by TAUTOLOGY). Furthermore, by the rule SUBST, each of the formulas $E_1\sigma_i',\ldots,E_4\sigma_i'$, $1\leq i\leq k$, is a theorem of

PV1, and $A_1\sigma_i^!,\ldots,A_n\sigma_i^!$, $1\le i\le k$, can be derived in PV1 from the hypotheses $A_1,\ldots,A_n$. Hence, by the rule IMP, we see that $A_1,\ldots,A_n \vdash_{PV1} B$.

4. The Gödel Incompleteness Theorem for PV

The main theorem in this section states that the consistency of PV cannot be proved in PV. This will be applied in section 7 to show that the system PV, as a proof system for the propositional calculus, is not p-verifiable.

It is easy to see that PV is incomplete, because the equivalence problem for functions in L is not recursively enumerable. But we need to know that a proof of this incompleteness can be given in PV itself so that we can follow Gödel's method of proving that a theory cannot have a proof of its own consistency.

The first step is to assign "Gödel numbers" to the terms, equations, and proofs in PV. Notice that an object of any of these three kinds has been defined to be a string of symbols. The underlying alphabet of symbols is infinite, because we assume there are an unlimited number of variables at our disposal. However, we can agree that a variable is just the symbol x followed by a finite string on the alphabet {1,2}. Hence any term, equation, or proof, is a finite string on some fixed alphabet A of at most 32 symbols. We can code each symbol σ in A by a unique five-digit code $\psi(\sigma)$ over the alphabet {1,2}. Then the Gödel number of a string $\sigma_1\ldots\sigma_k$ is the number whose dyadic notation is $\psi(\sigma_1)\ldots\psi(\sigma_k)$. The number of an object C is denoted by [C]. The important property of Gödel numbers from our point of view is that an object C and the dyadic notation for [C] can be obtained from each other within time bounded by polynomials in the lengths of [C] and C, respectively.

We define the function proof on the natural numbers by

$$\text{proof}(m,n) = \begin{cases} 1 \text{ if } m \text{ is the number of an equation } t = u, \text{ and } n \text{ is the number of a proof in PV of } t = u \\ 0 \text{ otherwise} \end{cases}$$

Next we define the function sub as follows: $\text{sub}(m) = n$ if $m = [t=u]$ and $n = [(t=u)^{\overline{m}}_{x}]$, for some equation $t = u$, where $\overline{m}$ is the numeral for m. If m is not of the form [t=u], then sub(m) = 0.

It is not hard to see that both the functions proof and sub can be computed in time bounded by a polynomial in the lengths of their arguments, so that both functions are in L. By theorem 2.12 there are function symbols PROOF and SUB in PV which define proof and sub, respectively. (We assume that the defining equations for these function symbols represent a straightforward algorithm for computing the functions.) Let

4.1 $\quad r = [\text{PROOF}(\text{SUB}(x),y)=0]$

Then

4.2 $\quad s = \text{sub}(r) = [\text{PROOF}(\text{SUB}(\overline{r}),y)=0]$

Thus equation number s says "I am not provable".

4.3 Theorem. Equation number s has no proof in PV.

Proof. Suppose, to the contrary, that p is the number of a proof of equation number s. By the valuation theorem (2.18), we have $\vdash_{PV} \text{PROOF}(\text{SUB}(\overline{r}),\overline{p}) = 1$. But by assumption, $\vdash_{PV} \text{PROOF}(\text{SUB}(\overline{r}),y) = 0$, so by the rules R4, R1, and R2 of PV, $\vdash_{PV} 0 = 1$. This contradicts the consistency of PV (theorem 2.17), establishing the present theorem.

Now let CON(PV) stand for the equation PROOF([0=1],y) = 0. This is a true equation of PV, asserting that the equation 0 = 1 has no proof in PV.

4.4 Theorem. CON(PV) has no proof in PV.

The idea, of course, is to show that the proof of theorem 4.3 can be formalized in PV. We will actually work in the system PV1, since this is easier. The first step is to formalize the valuation theorem (2.18) in PV1. The proof of 2.18 shows how to construct, for each function symbol f of PV, a function gen_f in L such that $\text{gen}_f(n_1,\ldots,n_k,m)$ is the number of a proof in PV of the equation $f(\overline{n}_1,\ldots,\overline{n}_k) = \overline{m}$, provided the equation is true, and $\text{gen}_f(n_1,\ldots,n_k,m) = 0$ otherwise. The function $\text{form}_f(n_1,\ldots,n_k,m) = [f(\overline{n}_1,\ldots,\overline{n}_k)=\overline{m}]$ is certainly in L. It should be possible to show

4.5 Lemma. $\vdash_{PV1} f(x_1,\ldots,x_k)=y \supset \text{PROOF}(\text{FORM}_f(x_1,\ldots,x_k,y),\text{GEN}_f(x_1,\ldots,x_k,y)) = 1$ for each function symbol f of PV, where FORM_f and GEN_f are the function symbols defining form_f and gen_f, respectively.

Now let us apply the lemma when f is PROOF, and substitute $\overline{s}$ (from 4.2) and 1 for two of the variables, to obtain

4.6 $\quad \vdash_{PV1} \text{PROOF}(\overline{s},y)=1 \supset \text{PROOF}(\text{FORM}(\overline{s},y,1),\text{GEN}(\overline{s},y,1))=1$

where we have left off the subscripts on FORM and GEN. By definition, $\text{form}_{PROOF}(s,n,1) = [PROOF(\overline{s},\overline{n})=1]$, and for each value of n, a proof (say number p) in PV of the equation in brackets together with a proof (say number q) in PV of formula number s (see 4.2) gives rise easily to a proof (say number contra(p,q,n)) in PV of 0 = 1. If we let CONTRA define the function contra, then one can prove the last statement in PV1.

4.7 Lemma.
$\vdash_{PV1}$ $(PROOF(FORM(\overline{s},y,1),z)=1 \& PROOF(\overline{s},u)=1)$
$\supset PROOF([0=1],CONTRA(z,u,y))=1$

Now lemma 4.7 with $GEN(\overline{s},y,1)$ substituted for z and y substituted for u, together with 4.6 gives us immediately by the rule IMP of PV1

4.8 $\vdash_{PV1} PROOF(\overline{s},y)=1 \supset PROOF([0=1],t)=1$

where t is $CONTRA(GEN(\overline{s},y,1),y,y))$.

By axiom E7 of PV1, $\vdash_{PV1} \neg 0 = 1$. Hence, by substituting t for y in the definition of CON(PV), we have by IMP and equality reasoning, from 4.8,

4.9 $\vdash_{PV1} CON(PV) \supset \neg PROOF(\overline{s},y)=1$

Simple reasoning shows $\vdash_{PV1} \neg PROOF(x,y)=1 \supset PROOF(x,y)=0$, and since by the valuation theorem, $\vdash_{PV} \overline{s} = SUB(\overline{r})$, we have by 4.9

4.10 $\vdash_{PV1} CON(PV) \supset PROOF(SUB(\overline{r}),y)=0$

Thus, if $\vdash_{PV} CON(PV)$, then $\vdash_{PV1} CON(PV)$ (by theorem 3.10), so $\vdash_{PV1} PROOF(SUB(\overline{r}),y) = 0$, so $\vdash_{PV} PROOF(SUB(\overline{r}),y) = 0$ (again by 3.10), which contradicts theorem 4.3. This completes our outline of the proof of theorem 4.4.

5. Propositional Calculus and the Main Theorem

Propositional formulas will be formed in the usual way from the connectives $\&,\vee,\neg,\supset,\equiv$, and from an infinite list of atoms. We will define an atom to be the letters ATOM followed by a string on {1,2}, so that formulas are certain strings on a certain fixed finite alphabet. We can assign Gödel numbers to the strings as in section 4, and we will write [A] for the number of the formula A. A tautology is a valid propositional formula, and we will use TAUT to denote the set of Gödel numbers of tautologies.

A proof system (for TAUT) is a function f in L from the set of natural numbers onto TAUT. (This differs from the definition in [3] in that numbers are used instead of strings.) If f is a proof system, and f(x) = [A], then x is (or codes) a proof of A.

The paper [3] describes a large number of standard proof systems, and compares them from the point of view of length of proof. The system we are interested in here is a very powerful system called extended resolution (ER), which can efficiently simulate any of the standard systems, except possibly Frege systems with a substitution rule. The idea of extended resolution is due to Tseitin [13].

The system ER can be defined as follows. A literal is an atom or a negation of an atom. The complement $\overline{L}$ of a literal L is given by $\overline{P} = \neg P$, $\overline{\neg P} = P$, where P is an atom. A clause is a disjunction $(L_1 \vee \ldots \vee L_k)$ of literals, $k \geq 0$, with no literal repeated. If k = 0 the clause (called the empty clause) is denoted by $\square$. If A is a propositional formula, then we associate a literal L_B with every subformula B of A by the conditions (i) if B is an atom, then L_B is B, (ii) if B is $\neg C$, then L_B is $\overline{L_C}$, and (iii) if B is $(C \vee D)$, $(C \& D)$, $(C \supset D)$, or $(C \equiv D)$, then L_B is some atom uniquely associated with B.

If F is a propositional formula, then CNF(F) denotes some set of clauses whose conjunction is equivalent to F (and which is not unnecessarily long). Now we associate with every propositional formula A a set def(A) of clauses by the conditions (i) $def(P) := \emptyset$ if P is an atom, (ii) $def(\neg B) = def(B)$, (iii) $def(B \circ C) = def(B) \cup def(C) \cup CNF(L_{B \vee C} \equiv (L_B \circ L_C))$, where $\circ$ is $\&$, $\vee$, $\supset$, or $\equiv$.

5.1 Lemma. a) Any truth assignment τ to the atoms of A has a unique extension τ' to the atoms of def(A) which makes (each clause in) def(A) true. In fact, $\tau'(L_B) = \tau(B)$ for each subformula B of A, so in particular, $\tau'(L_A) = \tau(A)$.
b) A is a tautology if and only if L_A is a truth-functional consequence of def(A).
c) There is a function f in L which satisfies $f([A]) = [def(A)]$.

Part a) is proved by induction on the length of A. Part b) follows immediately from a). For part c), observe that def(A) has at most three times as many clauses as A has connectives, and these clauses are easily found.

Notice that, in contrast to def(A), CNF(A) is not in general computable in polynomial time, simply because some formulas have a shortest conjunctive normal form which is exponential in their length. (For example, $(P_1 \& P_2 \vee \ldots \vee P_{2n-1} \& P_{2n})$).

If a clause C_1 is $(L_1 \vee \ldots \vee L_i \vee L \vee L_{i+1} \vee \ldots \vee L_k)$ and C_2 is $(M_1 \vee \ldots \vee M_j \vee \bar{L} \vee M_{j+1} \vee \ldots \vee M_\ell)$ then the resolvent of C_1 and C_2 is the clause which results from deleting repetitions of literals from $(L_1 \vee \ldots \vee L_k \vee M_1 \vee \ldots \vee M_\ell)$.

The extension rule for an atom P allows the introduction of the three or four clauses in $CNF(P \equiv (L_1 \circ L_2))$, where $\circ$ is $\&$, $\vee$, $\supset$, or $\equiv$, provided P and $\bar{P}$ are distinct from L_1 and L_2. An ER proof of a formula A is a string $A*C_1*\ldots*C_k**C_{k+1}*\ldots*C_n$, where C_n is L_A, $\{C_1,\ldots,C_k\}$ are the clauses in def(A), and each C_i for $i > k$ is either a resolvent of two earlier C_j's, or is introduced by the extension rule for some atom P which has no earlier occurrence in the string. Any string not of the above form is, by convention, an ER proof of $(P \vee \neg P)$, for some fixed atom P. The proof system ER is the function such that ER(n) = [A], provided the dyadic notation of n codes an ER proof of A. It is easy to see the function ER is in L.

It follows from lemma 5.1 and a slight modification of the usual completeness theorem for ground resolution (see [14], for example) that every tautology A has an ER proof in which the extension rule is not used. (The purpose of the extension rule is to give shorter proofs.) I now prove the converse explicitly, since I want to argue later that the proof can be formalized in PV.

5.2 Soundness of ER. If a formula A has an ER proof, then A is a tautology.

Proof. If A is $\neg P \vee P$, then A is obviously a tautology. Otherwise, the proof has the form $A*C_1*\ldots*C_k**C_{k+1}*\ldots*C_n$ described earlier. Let τ be any truth assignment to the atoms of A. Then, as mentioned in lemma 5.1, τ can be extended to a truth assignment τ' to the atoms L_B of def(A) such that τ' makes all clauses in def(A) true and $\tau'(L_A) = \tau(A)$. Hence τ' makes $C_1,\ldots,C_k$ true. Further, each time clauses D_1,D_2,D_3 are introduced by the extension rule for an atom P, τ' can be extended to τ'' whose domain includes P in such a way that D_1,D_2,D_3 are true under τ'' (for example, if the clauses are $CNF(P \equiv (L_1 \vee L_2))$, then $\tau''(P)$ is $\tau'(L_1 \vee L_2)$). Thus there is an extension τ_1 of τ' which makes all clauses C_i introduced by extension true. It is easy to see that any truth assignment which makes two clauses true must make any resolvent of those clauses true. Hence, by induction on i, we see that τ_1 makes C_i true for $1 \le i \le n$. In particular, τ_1 makes $C_n = L_A$ true. Since $\tau_1(L_A) = \tau'(L_A) = \tau(A)$, τ makes A true. Since τ is an arbitrary truth assignment to A, A is a tautology.

The above argument shows more than just the soundness of ER. It shows that an ER proof of A provides a uniform method of checking rapidly that a given truth assignment satisfies A; namely check that τ_1 satisfies successively $C_1,C_2,\ldots,C_n = L_A$, and check successively that $\tau_1(B) \equiv \tau_1(L_B)$ for larger and larger subformulas B of A, and finally check that $\tau(A) = \tau_1(L_A) = \text{true}$. Thus ER is a p-verifiable proof system in the following sense.

5.3 (Informal definition). A proof system F for TAUT is p-verifiable iff there is a polynomial p(n) such that given a proof x in the system of a formula A, x gives a uniform way of verifying within $p(|x|)$ steps that an arbitrary truth assignment to A satisfies A.

It is easy to see that all the usual "Frege" systems (see [3]) for the propositional calculus satisfy this definition, in addition to ER. On the other hand, if the substitution rule (from A conclude $A\sigma$, where σ substitutes formulas for atoms) is added to Frege systems, then it is no longer clear that the system is p-verifiable. A proof of A in such a system does provide a way of verifying that a given truth assignment τ satisfies A, but since a formula B in the proof may have several substitution instances in the proof, and each of these instances can again have several instances, and so on, we may end up having to verify B for exponentially (in the length of the proof) many truth assignments to check that A comes out true under the single assignment τ. Also, there is no reason to think that a proof system for TAUT which incorporates Peano number theory or set theory is p-verifiable.

To make the notion of p-verifiable proof system precise, let us code a truth assignment τ as a string $(P_1,\tau(P_1)),(P_2,\tau(P_2)),\ldots,(P_k,\tau(P_k))$ listing the atoms in its domain and the truth value assigned to these atoms. This string in turn can be coded as a string on {1,2}, and $[\tau]$ will denote the number whose dyadic notation is this last string. Then we can define a function tr in L such that

$$tr([A],[\tau]) = \begin{cases} 1 & \text{if } \tau(A) \text{ is true} \\ 0 & \text{if } \tau(A) \text{ is false} \end{cases}$$

We can make the convention that τ assigns false to all atoms of A for which a value is not explicitly given, so that $\tau(A)$ is

defined for any formula A and truth assignment τ and every number n codes some truth assignment. Let TR be a function symbol in PV which defines tr.

5.4 (Formal Definition). A proof system f for TAUT is p-verifiable iff there is some function symbol F in PV defining f such that $\vdash_{PV} TR(F(x),y) = 1$.

It is worth pointing out that this formal definition depends on the particular function symbol TR chosen to define tr. That is, it depends on the algorithm chosen to compute tr. Presumably, if TR and TR' both represent straightforward algorithms for computing tr, then $\vdash_{PV} TR(x,y) = TR'(x,y)$, so definition 5.4 would be the same for TR and TR'.

The formal definition requires that the soundness of f be provable in PV. If one believes the verifiability thesis (1.1), then it is easy to see that the formal definition captures the informal one.

In [3], a notion of one proof system simulating another is defined. Here I would like to sharpen that notion and say that a proof system f_1 p-simulates a proof system f_2 iff there is a function g in L such that $f_2(n) = f_1(g(n))$ for all n. Further, f_1 p-verifiably simulates f_2 iff there exist functions symbols F_1, F_2 in PV defining f_1, f_2, respectively, and a function symbol G such that $\vdash_{PV} F_2(x) = F_1(G(x))$.

Now I can state the main theorem of this paper, which characterizes the p-verifiable proof systems.

5.5 Main Theorem. A proof system f for tautologies is p-verifiable if and only if extended resolution p-verifiably simulates f.

5.6 Theorem. Extended resolution p-verifiably simulates any Frege system (see [3]).

5.7 Corollary. Every Frege system is a p-verifiable proof system.

Theorem 5.6 can be proved by formalizing in PV the proof in [3] which shows that ER simulates any Frege system. The argument will not be given here.

The following lemma is needed for the Main Theorem.

5.8 Lemma. ER is p-verifiable. That is, $\vdash_{PV} TR(EXTRES(x),y) = 1$, where EXTRES is a suitable function symbol in PV defining ER.

The proof amounts to showing the proof of 5.2 (Soundness of ER) can be formalized in PV. (Of course, in practice it is easier to work in PV1.) Thus one defines a function tauone(n) in L such that when $n = [\tau]$, then $tauone(n) = [\tau_1]$, where τ_1 is the truth assignment described in that argument. Then the formal versions of the equations $\tau_1(L_A) = \tau'(L_A) = \tau(A)$ are provable in PV, and TR(ER(x),y) = 1 follows. The details are omitted.

The "if" part of the Main Theorem follows easily from the lemma. For suppose ER p-verifiably simulates f. Then $\vdash_{PV} F(x) = EXTRES(G(x))$, where F defines f. If rule R3 of PV (with TR for f) is applied to this equation and the result applied with transitivity to 5.8 with G(x) for x, we obtain $\vdash_{PV} TR(F(x),y) = 1$. Hence f is p-verifiable.

The converse to the Main Theorem is more difficult and will be dealt with in the next section.

6. Propositional Formulas Assigned to Equations of PV

To prove the "only if" part of theorem 5.5 I propose to first prove that extended resolution can p-simulate any p-verifiable proof system, and then argue that this proof can be formalized in PV. This first proof is carried out by assigning, for each m, a propositional formula to each equation t = u which says, roughly speaking, "the equation holds when variables are restricted so that the dyadic notations for all relevant functions have length at most m". I then argue that if $\vdash_{PV} t = u$, then there is an ER proof of the formula whose length is bounded by a polynomial in n. Applying this result to the equation TR(F(x),y) = 1 (which is provable in PV if F represents a p-verifiable proof system f), one can see that there is an ER proof of formula number f(n) which is not much longer than the proof n.

Proceeding more formally, let us fix the integer $m > 0$. We associate with every term t of PV the atoms $P_0[t], P_1[t], \ldots, P_m[t]$ and $Q_0[t], Q_1[t], \ldots, Q_m[t]$. We will call these the atoms of t. The intended meanings are

$$P_i[t] \equiv \begin{cases} \text{true if ith dyadic digit (i.e. coefficient of } 2^i\text{) of t is 2} \\ \text{false if this digit is 1} \\ \text{irrelevant if the dyadic length of t is } < i+1 \end{cases}$$

$$Q_i[t] \equiv \begin{cases} \text{true if coefficient of } 2^i \text{ in t is defined (i.e. } t \geq 2^{i+1} - 1) \\ \text{false otherwise} \end{cases}$$

Now we can define, for each term t and each truth assignment τ to the atoms of t which

satisfies $Q_i[t] \supset Q_{i-1}[t]$, $1 \le i \le m$, a number $val_m(t,\tau)$ which is the number whose dyadic notation is determined by these intended meanings. Next we associate a propositional formula $prop_m[t]$ with the term t (the subscript m will sometimes be omitted). Among the atoms of this formula are some of the atoms of t and the atoms of the variables which occur in t. This formula has the following property:

6.1 Semantic Correctness of $prop_m$. Let the term t of PV with variables $x_1,\ldots,x_n$ define the function $f(x_1,\ldots,x_n)$, and let τ be a truth assignment which satisfies $prop_m[t]$ and such that when f is evaluated (according to the defining equations in PV) at $x_i = val_m(x_i,\tau)$, $1 \le i \le n$, no value of any number appearing in the computation exceeds m in dyadic length. Then $val_m(t,\tau) = f(val_m(x_1,\tau),\ldots,val_m(x_n,\tau))$.

To define $prop_m[t]$ in general in such a way that 6.1 holds, we start with the following special cases.

6.2 $prop_m[x]$ is $\underset{i=1}{\overset{m}{\&}} Q_i[x] \supset Q_{i-1}[x]$, for each variable x.

6.3 $prop_m[0]$ is $\underset{i=0}{\overset{m}{\&}} \neg Q_i[0]$

6.4 $prop_m[s_1(x)]$ is

$$(prop_m[x] \underset{i=0}{\overset{m-1}{\&}} (P_{i+1}[s_1(x)] \equiv P_i[x]) \;\&\neg P_0[s_1(x)] \;\&Q_0[s_1(x)] \underset{i=0}{\overset{m-1}{\&}} (Q_{i+1}[s_1(x)] \equiv Q_i[x]))$$

6.5 $prop_m[s_2(x)]$ is defined similarly.

Let $\sigma = \frac{t_1,\ldots,t_k}{x_1,\ldots,x_k}$ be a substitution (regarded as a transformation) of terms for variables. The function ψ takes a substitution and an atom of t into an atom of $t\sigma$, and is defined by the equations $\psi(\sigma,P_i[t]) = P_i[t\sigma]$, and $\psi(\sigma,Q_i[t]) = Q_i[t\sigma]$, where $t\sigma$ is the term resulting when σ is applied to t. ψ can be extended in an obvious way so that its second argument is any propositional formula in the atoms of various terms t. Thus $\psi(\sigma,\neg A) = \neg\psi(\sigma,A)$, and $\psi(\sigma,(A \circ B)) = (\psi(\sigma,A) \circ \psi(\sigma,B))$, where $\circ$ is &, $\vee$, $\supset$, or $\equiv$. The formulas $prop_m$ will satisfy the following property:

6.6 Substitution Principle. Let $\sigma = \frac{t_1,\ldots,t_k}{x_1,\ldots,x_k}$. Then

$$prop_m[t\sigma] \iff \underset{i=1}{\overset{k}{\&}} prop_m[t_i] \& \psi(\sigma, prop_m[t]),$$

where <=> can be read "is truth-functionally equivalent to".

For example, if t is $s_1(x)$ and σ is $\frac{0}{x}$, then this principle and 6.3, 6.4, say that $prop_m[s_1(0)]$ is a conjunction of formulas, including $\neg Q_i[0]$, $1 \le i \le m$, and $\neg P_0[s_1(0)]$, and $Q_0[s_1(0)]$, and $Q_{i+1}[s_1(0)] \equiv Q_i[0]$, $0 \le i \le m-1$. These formulas imply $\neg P_0[s_1(0)]$, $Q_0[s_1(0)]$, and $\neg Q_i[s_1(0)]$, $1 \le i \le m$, which completely specify the dyadic notation for $s_1(0)$ (=1).

Now suppose $prop_m[f(x_1,\ldots,x_k)]$ has been defined for all function symbols f is a certain set S. Then we can inductively define $prop_m[t]$ for each term t built from 0, variables, and function symbols in S by

6.7 $prop_m[f(t_1,\ldots,t_k)] =$

$$\underset{i=1}{\overset{k}{\&}} prop_m[t_i] \& \psi\left(\frac{t_1,\ldots,t_k}{x_1,\ldots,x_k}, prop_m[f(x_1,\ldots,x_k)]\right)$$

To complete the definition of $prop_m[t]$ for all terms t, it suffices to show how to define $prop_m[f(x_1,\ldots,x_n)]$ for each of the two ways of defining new function symbols. First, suppose f is $\lambda x_1 \ldots x_n t\rho$, where $prop_m[t]$ has been defined. Then the defining equation for f is $f(x_1,\ldots,x_n) = t$, and we define

6.8 $prop_m[f(x_1,\ldots,x_n)]$ is

$$(prop_m[t] \underset{i=0}{\overset{m}{\&}} (P_i[f(x_1,\ldots,x_n)] \equiv P_i[t]) \;\underset{i=0}{\overset{m}{\&}} (Q_i[f(x_1,\ldots,x_n)] \equiv Q_i[t]))$$

The case in which f is defined by recursion on notation is more complicated, and is omitted for lack of space. This completes the definition of $prop_m[t]$, for all terms t.

Now suppose $x_1,\ldots,x_r$ is a list of all the variables appearing in the terms t and u, and n,m are positive integers with $n \le m$. Then $|t=u|^n_m$ is the propositional formula

$$((\text{prop}_m[t]\&\text{prop}_m[u])\&(\neg Q_{n+1}[x_1]\&\ldots\&\neg Q_{n+1}[x_r])) \supset$$

$$\left(\&_{i=0}^{m} Q_i[t]\supset(P_i[t]\equiv P_i[u])\right) \left(\&_{i=0}^{m} (Q_i[t]\equiv Q_i[u])\right)$$

We say that m is a bounding value for n relative to t = u if the terms t and u can be evaluated by the relevant defining equations for all values of their variables of dyadic length n or less without having any value in the computation exceed m in dyadic length.

6.8 ER Simulation Theorem. Suppose Π is a proof in PV of $t = u$. Then there is a polynomial p(m) (depending on Π) such that for all n,m, if m is a bounding value for n relative to t = u, then $|t=u|^n_m$ has an extended resolution proof of length at most p(m).

The proof is by induction on the length of Π, and is omitted.

Using this theorem, we can sketch the proof of the "only if" part of theorem 5.5. Thus suppose f is a p-verifiable proof system, and suppose F is a function symbol in PV which defines f, such that $\vdash_{PV} TR(F(x),y) = 1$. Since all functions used in defining F and TR are in L, it follows that one can find a polynomial q with natural number coefficients such that for all n, q(n) is a bounding value for n relative to TR(F(x),y) = 1, and $q(n) \to \infty$. By theorem 6.8, there is a polynomial p(n) such that $|TR(F(x),y)=1|^n_{q(n)}$ has an ER proof of length at most p(q(n)), for all n.

Now let $P_1,\ldots,P_k$ be the atoms of some propositional formula A. A truth assignment τ to these atoms determines a number $[\tau]$ in a straightforward manner, and using a variable y for $[\tau]$, we can use the extension rule to introduce a set CL of clauses defining the atoms $P_i[y]$ and $Q_i[y]$ in terms of the atoms $P_1,\ldots,P_k$. Thus any truth assignment τ' which satisfies all clauses in CL must have the property that if τ is the restriction of τ' to $P_1,\ldots,P_k$, then $[\tau]$ is the value of y whose dyadic notation is represented by $\tau'(P_i[y])$, $\tau'(Q_i[y])$, $1\le i\le m$, for suitable m. Since any truth assignment τ'' satisfying def(A) must have $\tau''(A) = \tau''(L_A)$, one can see from the way TR is defined that there is an ER derivation of $L_A \equiv Q_0[TR(\overline{[A]},y)]$ from def(A), CL, and $\text{prop}_m[TR(\overline{[A]},y)]$, for suitable m. Further, there is a polynomial r(n) such that for all formulas A, this ER derivation has length at most r([A]).

Now suppose A has a proof a in the system f; that is, suppose f(a) = [A]. Then by the valuation theorem 2.18, $\vdash_{PV} F(\overline{a}) = \overline{[A]}$, and one can verify that $|F(\overline{a})=\overline{[A]}|^n_{q_1(n)}$ has an ER proof of length not exceeding $p_1(q_1(n))$, where $n = |a|$, for some polynomials p_1 and q_1. Putting this ER proof together with the ones in the preceding two paragraphs, and noting that the clauses in CL, $\text{def}(\text{prop}_m[t])$ for all terms t involved can be introduced by the extension rule, we come up with an ER proof g(a) of A of length not exceeding $p_2(|a|)$, for some polynomial p_2. Thus ER(g(a)) = f(a) for all a, and since g(a) is in L, ER p-simulates f.

To complete the proof of theorem 5.5 it is necessary to show $\vdash_{PV} ER(G(x)) = F(x)$, where G is a function symbol in PV defining g. This amounts to showing the above argument can be formalized in PV, which I will not do here. It is not hard to check, however, that the above argument is feasibly constructive, so that if one believes the verifiability thesis (1.1), the formalization is not necessary.

7. PV as a Propositional Proof System

Any formal system for number theory can be treated as a proof system for TAUT by regarding a proof of the formalization of tr([A],y) = 1 as a proof of A. In particular, if Π is a proof in PV of $TR(\overline{[A]},y) = 1$, then Π is a proof in PV of A. We can define a function pv in L which satisfies $pv([\Pi]) = [A]$ if Π is a proof of A. Thus pv is a proof system for TAUT in the general sense defined in section 5.

7.1 Theorem. The system pv is not p-verifiable.

7.2 Lemma. $\vdash_{PV1} TR(PV(x),y)=1 \supset PROOF([0=1],z)=0$

The lemma says that the statement "if pv is p-verifiable, then PV is consistent" is provable in PV1. I prove the lemma by giving an informal argument for the implication which is readily formalized in PV1, using the techniques of section 4.

By hypothesis,

(1) $tr(pv(m),n) = 1$ for all m and n.

It is not hard to see that

(2) $\vdash_{PV} TR([P\&\neg P],y) = 0$.

Now suppose PV is inconsistent, so that

(3) $\vdash_{PV} 0 = 1$.

Then from (2) and (3),

(4) $\vdash_{PV} TR([P\&\neg P],y) = 1$.

If Π is a proof in PV of (4), then

(5) $pv([\Pi]) = [P\&\neg P]$.

Combining (1) and (5), we have

(6) $tr([P\&\neg P],n) = 1$, for all n.

But (6) is absurd, since $tr([P\&\neg P],n) = 0$. Hence our assumption that PV is inconsistent is untenable, so PV is consistent.

From lemma 7.2, we see that if $\vdash_{PV} TR(PV(x),y) = 1$, then $\vdash_{PV1} CON(PV)$, so $\vdash_{PV} CON(PV)$, violating theorem 4.4. Therefore pv is not p-verifiable.

8. Conclusions and Future Research

(1) There should be alternative formalizations of PV. These would make the verifiability thesis (1.1) more convincing and make it easier to formalize arguments in PV. One such formalization should be a programming approach, where proving $f(x) = g(x)$ amounts to proving the equivalence of two programs.

(2) If one believes that all feasibly constructive arguments can be formalized in PV, then it is worthwhile seeing which parts of mathematics can be so formalized. I think that a good part of elementary number theory (such as the unique factorization theorem) can be formalized in PV, although the results will have to be formulated carefully. For example, the function $p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ is not in L and so it is not definable in PV. However, the relation $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ is an L-relation, and its characteristic function is definable in PV. As another example of formulation problems, it is hard to see at first how to formulate in PV the completeness of a proof system for TAUT such as ER, since there is no function g in L taking an arbitrary tautology number [A] into an ER proof of A (unless $P = NP$). However, there is a function g in L which takes a code for a tautology A together with a list $\tau_1,\ldots,\tau_k$ of all truth assignments to A into an ER proof of A, and the equation $ER(G([A*\tau_1*\ldots*\tau_k])) = [A]$ should be provable in PV. This formulation of completeness says that given a formula A, together with a verification that all truth assignments to A make A true, one can find an ER proof of A. This statement certainly incorporates the information that every tautology has an ER proof.

(3) The question that lead me to the system PV in the first place is the question of whether extended resolution is a super proof system. I conjecture that it is not. A possible approach to showing this is by proving some sort of converse to the ER simulation theorem (6.8). Specifically, I conjecture that the propositional formulas $|cap's\ (PV)|^n_{q(n)}$ have no ER proofs bounded in length by a polynomial in n, where q(n) is a bounding value for n relative to con(PV).

(4) It would be interesting to prove that a Frege system with substitution (see [3]) is not p-verifiable. A likely approach is to show that such a system p-verifiably simulates pv, which would mean that if such a system were p-verifiable, so would be pv, violating theorem 7.1.

ACKNOWLEDGMENTS

I would like to thank Robert Constable for helpful discussions concerning constructive mathematics, and for a critical reading of the first half of this manuscript. I would like to thank Martin Dowd for helping with the technical aspects of k-recursion and k-induction on notation.

REFERENCES

1. S.A. Cook. "The complexity of theorem proving procedures". Proceedings of Third Annual ACM Symposium on Theory of Computing, May, 1971.

2. R.M. Karp. "Reducibility among combinatorial problems" in Complexity of Computer Computations, R.M. Miller and J.N. Thatcher, editors, Penum Press, 1972.

3. S.A. Cook and R.A. Reckhow. "On the lengths of proofs in the propositional calculus". Proceedings of Sixth Annual ACM Symposium on Theory of Computing, May, 1974, pp 135-148. See also corrections for the above in SIGACT News, Vol 6, No 3 (July, 1974) pp 15-22.

4. R.J. Parikh. "Existence and feasibility in arithmetic". J. Symbolic Logic, Vol 36, No 3 (1971).

5. Th. Skolem. Begründung der Elementaren Arithmetik, 1923

6. R.L. Goodstein. Recursive Number Theory, North-Holland, Amsterdam, 1957.

7. Alan Cobham. "The intrinsic computational difficulty of functions". Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Sciences, North-Holland Publishing Co., Amsterdam, pp 24-30.

8. S.C. Kleene. Introduction to Mathematics, Van Nostrand, 1952.

9. H.E. Rose. "On the consistency and undecidability of recursive arithmetic". Zeitschr. f. math. Logik and Grundlogen d. Math, Bd. 7, S. 124-135 (1961).

10. R.M. Smullyan. Theory of Formal Systems, Princeton University Press, Revised Edition, 1961.

11. Daniel Lascar."Cobham's characterization of L", in Topics in the theory of Computation, notes of a seminar conducted by R.M. Baer and S.A. Cook, Dept. of Mathematics, University of California at Berkeley (March, 1967) (Unpublished)

12. J.P. Cleave and H.E. Rose. "E^n-arithmetic". Sets, Models, and Recursion Theory, Crossley, ed., North-Holland, Amsterdam, 1967.

13. G.S. Tseitin. "On the complexity of derivation in propositional calculus". Studies in Mathematics and Mathematical Logic, Part II, A.O. Slisenko, ed. (Translated from Russian).

14. C.L. Chang and C.T. Lee. Symbolic Logic and Mechanical Theorem Proving Academic Press, 1973.