Department of Philosophy - Utrecht University

A Δ_0 definition for Finite Sequences

Marc jumelet

Logic Group
Preprint Series
No. 79
September 1992

Department of Philosophy
University of Utrecht
Heidelberglaan 8
3584 CS Utrecht
The Netherlands

A Δ_0 definition for Finite Sequences.

Marc Jumelet
Department of Philosophy,
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
Marc.Jumelet@phil.ruu.nl

1. Introduction.

Throughout the following we will be interested in the solution of the problem whether finite sequences of natural numbers can be coded by means of Δ_0-formulae in such a way that the theory $I\Delta_0$ proves some basic facts about this way of coding. To be more precise: by "coding finite sequences in $\mathbb{N}$" the following two things are meant. First, a quintuple $< \in \text{Fseq}, \varnothing, *, <.>, \text{lth}(.) >$ exists such that all of its elements are expressions in the language of a given system of arithmetic, $\in$ Fseq is a unary predicate, $\varnothing$ is a constant, * is a binary function whereas <.> and lth(.) are unary ones. We require that the theory $I\Delta_0$ verifies these facts, which explains why we will use functional expressions like s*t = u. On top of that, we require $\in$ Fseq, *, <.> and lth(.) to be defined by Δ_0-formulae in the same language.
Secondly, we require the quintuple $< \in \text{Fseq}, \varnothing, *, <.>, \text{lth}(.) >$ to mimick as exactly as possible some operations one would be inclined to regard as natural conditions for reasoning with finite sequences. With $\in$ Fseq we intend a predicate which tells us whether a given number codes a finite sequence or not. The intended meaning of $\varnothing$ is the empty sequence, that of * is concatenation of two sequences, <.> is supposed to behave as a function that assigns a sequence of only one element, say n, to the number n and lth(.) should, if applied to a sequence, yield its length, that is, the number of numbers occurring in it. Making this informal explanation precise, the quintuple should have the property that the following facts are provable in $I\Delta_0$:

a. $\forall s \in \text{Fseq}\ s*\varnothing = \varnothing*s = s$;
b. $\text{lth}(\varnothing) = 0$;
c. $\forall s, t \in \text{Fseq}\ \text{lth}(s*t) = \text{lth}(s)+\text{lth}(t)$;
d. $\forall y, x\ (y = <x> \rightarrow \text{lth}(y) = 1)$;
e. $\forall x\ \exists!\ y\ (y = <x> \wedge y \in \text{Fseq})$;
f. $\forall s, s'\ (s, s' \in \text{Fseq} \rightarrow \exists!\ t \in \text{Fseq} \wedge s*s' = t)$;
g. lth(.) is total on all $s \in$ Fseq;
h. * is associative, in the sense that $(s*t)*u = s*(t*u)$ for all $s, t, u \in$ Fseq.

It could be argued that one should also have arbitrary projection functions at one's disposal. Although such functions have not been listed above, it will be clear from the following that they may be defined in an appropriate way with hardly any difficulty at all.

Why is it that this matter is not trivial?

Coding of sequences of a finite, given length can be done using a suitable iteration of some quadratic pairing function. In this case however, we want to deal with sequences of arbitrary length. A fine idea would be to use products of successive primes to code our finite sequences. Now any number would code a finite sequence, namely the sequence which is formed considering the exponents of the primes in the decomposition of that number following the order of the primes themselves. This way of coding is one of the most traditional ones in formalizing arithmetic (see e.g. Kleene [1974]). It is still not inconceivable that this way of coding may respect our demands. However there are some horrible obstacles. For example, in doing so, the Δ_0 -formulae that would now be required to express concatenation could only be verified to yield a total function if we can be sure that $I\Delta_0$ proves much more about the distribution of prime numbers than our present knowledge gives ground to assume. This becomes clear if we consider an example. Suppose we have coded the sequence 2, 4, 7 by the number $p_1^2 . p_2^4 . p_3^7$ and the sequence 3, 5 by the number $p_1^3 . p_2^5$. Naturally, the sequence 2, 4, 7, 3, 5 ought to be coded by the number $p_1^2 . p_2^4 . p_3^7 . p_4^3 . p_5^5$. Generally however, $I\Delta_0$ cannot tell us yet how big this number is going to be. Another example is provided by considering the singleton function. If, as is the case in the coding by means of exponents of primes, "singleton x" is coded by the number 2^x, we run into the difficulty that $I\Delta_0$ cannot prove the totality of this function "singleton x". This is a consequence of Parikh's theorem. In order to avoid such difficulties, we turn to a completely different method of coding.

2. Coding by means of two rulers.

A very organic way of coding is obtained by translating the numbers in our sequence to their binary expansions, putting these in the right order one next to the other and translating the obtained binary string back to the number it represents. If we want to do this properly, we also need a second number which tells us where our original numbers began and ended. This is the second ruler. Suppose we want to code the sequence 3, 2, 0, 0, 7. First, we write these numbers as binary words (from the left to the right):

11, 01, 0, 0, 111.

Next, we concatenate these words:

110100111.

As we want to know where the elements in this sequence actually are, we add an auxiliary ruler to contain this information:

010111001.

The idea is, that we code our sequence with this pair:

<110100111, 010111001>.

Now, we can "read" this pair by putting the second sequence underneath the first one and letting the 1's tell us where the words in the first sequence end.

11, 01, 0, 0, 111

01, 01, 1, 1, 001.

For the pairing function we can take: $\frac{1}{2}(x+y+1)(x+y)+x$.

An informal explanation of why it works.

The reason why this works the way we want it to, is basically due to the fact that $I\Delta_0$ is sufficiently strong to carry out these operations and the fact that these operations can be described by means of Δ_0-formulae.

As a matter of fact, our technique shows that the number of elements in our sequence corresponds precisely with the number of 1's that occur in the second ruler. Therefore it is sufficient that $I\Delta_0$ be able to count the number of 1's occurring in the binary expansion of any number if we want our length-function to be definable by a Δ_0-formula about which $I\Delta_0$ gives the right information. Concatenation of two sequences say, $<x_1, x_2>$ and $<y_1, y_2>$ can now be performed by forming the pair $<z_1, z_2>$ where z_1 is the number created by concatenating x_1 and y_1 (x_2 and y_2 will give the information necessary to do this properly) and z_2 the number created by concatenating x_2 and y_2.

The function that assigns to a number n a sequence that contains only the number n, can be defined in a very simple way. In our construction this sequence should be a pair of two numbers, the first of which is to be the number n itself and the second is to be the greatest power of 2 smaller than or equal to n itself. For instance, <13> should be:

$\frac{1}{2}(a+b+1)(a+b)+a$, where

a = 13 (1011 in binary),

b = 8 (0001 in binary).

What remains to be checked, of course, is the totality of our functions.

Preliminary definitions.

We will now make the effort of defining the operations that were sketched above in a more formal way. This will eventually show that the entire proces of coding finite sequences can take place within the framework of $I\Delta_0$. We already know that a Δ_0-definition for exponentiation exists, so we can use the shorthand notation $z = 2^a$. The same remark applies to the use of the remainder-function, abbreviated as rem(x, y). The first thing we need, is a suitable expression for "the a-th digit in b is a 1". The following will do:

$a \in_1 b := \exists x, k, z \leq b\ (z = 2^a \wedge b = z(1+2k) + x \wedge x \leq z)$. It needs no comment that this is indeed a Δ_0-definition.

Our next goal is to define the entier-logarithmic function. That is, the function that gives us the last digit in the binary representation of a number. Here it is:

$\ulcorner \log(b) \urcorner = c := (2^c \leq b \wedge b < 2^{c+1}) \vee (b = c = 0)$.

3. Counting zeroes and ones.

Since our coding of sequences of numbers uses a ruler that separates the elements that are stored in another number, our length-function will require the use of a function that tells us how many ones occur in the binary expression of a number. In fact, we will use it to count the number of ones that separate the elements in the first number occurring in the coding pair, which, of course, is exactly the amount of numbers coded by the pair. From Paris and Wilkie[1987] it is definitely clear that this function has a Δ_0-definition. However, we will give an alternative definition.

The first thing that we will need is a binary function that assigns to a given number n, a sum of powers of a given number m, such that m^j occurs in the result if and only if 2^j occurs in the binary representation of n.

First we will define by means of a Δ_0-formula what it means to be a sum of powers of a given number:

3.1 Definition. "x is a sum of powers of y" (Sumpow(x,y)):

$$\text{Sumpow}(x, y) := 1 \leq y \wedge \forall z, r, t \leq x\ ((\text{Pow}(t, y) \wedge x = t.z + r \wedge 0 \leq r < t) \rightarrow \text{rem}(z, y) = 0 \vee \text{rem}(z, y) = 1).$$

Now we are free to define what it means for a certain power of y to occur in a sum of powers of x. We will simply omit the case in which x is not a sum of powers of y at all.

3.2 Definition. "y^j is in the sum x of powers of y" (In(y, j, x)):

$In(y, j, x) := Sumpow(x, y) \wedge$

$\exists z \leq x\ (z = y^j \wedge \forall r, t \leq x\ ((x = t.z+r \wedge 0 \leq r < z) \rightarrow rem(t, y) = 1))$.

3.3 Definition. "z is the sum of powers of y resulting from x by replacing the occurrences of powers of 2 in the binary representation of x by equal powers of y" (Trans(x, y, z)):

$Trans(x, y, z) := Sumpow(z, y) \wedge \forall i \leq z\ (In(2, i, x) \leftrightarrow In(y, i, z))$.

Counting zeroes and ones now reduces to the following trick. We can replace the occurrences of powers of 2 by powers of some, sufficiently large number, say n+1 in the binary representation of x by means of the procedure Trans defined above. Next, if we take this result modulo n, then this remainder, provided n+1 was sufficiently large with respect to the logarithm of x, will coincide with the number of ones that did occur in the original binary representation of x.

Since, however, the growth rate of Trans is beyond polynomial size, we can only make a piece-wise use of this trick. That is, we will first chop up x in small parts before applying this trick to these parts, storing in the meantime the number of ones already counted this way, in an auxiliary sequence. First, we need to have some more equipment.

The entire construction is similar to the one in Hajek and Pudlak[1990], in as far as the number to be counted is chopped up in sufficiently small portions.

3.4 Definition. "y is the binary segment between the i-th and the j-th digit in x (Bseg(y, x, i, j))":

$Bseg(y, x, i, j) := \exists x_1, x_2 \leq x\ (i \leq j \wedge x = x_2 + y.2^i + x_1.2^j \wedge x_2 < 2^i \wedge y < 2^{j-i})$.

This definition is accompanied by the following claim:

3.5 Claim. $I\Delta_0$ proves the following statements:

i. $\forall$ x, y, i, j $Bseg(y, x, i, j) \rightarrow Bseg(y, 2.x, i+1, j+1)$.

ii. $\forall$ x, y, j $Bseg(y, x, 0, j) \rightarrow Bseg(2.y+1, 2.x+1, 0, j+1)$.

Proof. i. Reason in $I\Delta_0$: let $x_1, x_2 \leq x$ be such that

$(x = x_2 + y.2^i + x_1.2^j \wedge x_2 < 2^i \wedge y < 2^{j-i})$ holds. Let $x_3 = 2.x_1$, $x_4 = 2.x_2$ so,

$2.x = x_4 + y.2^{i+1} + x_3.2^j \wedge x_4 < 2^{i+1} \wedge y < 2^{(j+1)-(i+1)}$ will hold. But that is exactly what was needed for Bseg(y, 2.x, i+1, j+1) to be true.

ii. If Bseg(y, x, 0, j) holds, then there will be x_1 in, such that $x_1 \leq x$ and $x = y + x_1.2^j$ holds. As a consequence, we will also have $2.x+1 = (2.y+1) + x_1.2^{j+1}$, from which immediately follows that Bseg(2.y+1, 2.x+1, 0, j+1) holds.

The following claim, concerns the growth-rate of the procedure Trans on binary segments of a given length by considering the "worst case" that occurs whenever a given binary segment shows a one in all of its digits.

3.6 Claim. $I\Delta_0$ proves the following statements:

i. $\forall$ k, m, z, t ([Sumpow(z, m) $\wedge$ $\forall i \leq z$(In(m, i, z)$\rightarrow$i<k) $\wedge$ $t = m^k$]$\rightarrow$
$1+m.z \leq z+t$).

ii. $\forall$ x, y, i, n, z, t ([Bseg(y, x, i.n, (i+1).n) $\wedge$ Trans(y, n+2, z) $\wedge$ $t = (n+2)^n$]$\rightarrow$
$1+(n+1).z \leq t$).

Proof. i. Reason in $I\Delta_0$. The proof is by induction on k. If k = 0 and Sumpow(z, m) holds, then $\forall i \leq z \neg$In(m, i, z) will hold, so z = 0. But, since t = 1 will hold, we have $1 + m.0 \leq 0+1$. If k = k'+1, then observe that, if Sumpow(z, m) holds, then either z = 0, or there is some z' < z such that Sumpow(z', m) holds and z = m.z'+1 or z = m.z'. If z = 0 the statement is trivial since $1+m.z \leq z+t$ now reduces to $1 \leq t$ for t some power of m. If z = m.z'+1 for some z' being a sum of powers of m, then from $\forall i \leq z$(In(m, i, z)$\rightarrow$i<k'+1) we can infer: $\forall i \leq z'$(In(m, i, z')$\rightarrow$i<k'), so, by induction, we obtain: $1+m.z' \leq z'+m^k$. Multiplying by m on either side, we now obtain: $m.z \leq m.z'+m^{k'+1}$, from which $1+m.z \leq z+m^k$ readily follows. On the other hand, if z = m.z', for some z' being a sum of powers of m, then the proof is analogous.

ii. This follows from i, observing that if Bseg(y, x, i.n, (i+1).n) and Trans(y, n+2, z) hold, then Sumpow(z, n+2) will hold by definition and In(n+2, i, z) will only hold for numbers i strictly less than n.

This is sufficient if we want to count the number of occurrences of ones in a segment of x of a specific, sufficiently restricted length. Therefore, the following definition:

3.7 Definition. "z is the sum of powers of m resulting from the binary segment of x between the i-th and the j-th digit by replacing the occurrences of powers of 2 in the binary representation of that segment of x by equal powers of m, and w is the remainder of z modulo n (rsp(x, i, j, n, m, z, w))":

rsp(x, i, j, n, m, z, w):= $\exists t \leq x$ (Trans(t, m, z) $\wedge$ Bseg(t, x, i, j) $\wedge$ $w \equiv z \bmod n$).

The procedure described above, is to be used in the following manner: if we take the binary segment of x between the i-th and the i+ℓ-th digit that contains ℓ digits and we replace the occurrences of powers of 2 in this segment by powers of ℓ+2, then the procedure rsp(x, i, i+ℓ, ℓ+1, ℓ+2, z, w) will give us a number w which is exactly the number of ones in that segment. This can easily be seen by observing the fact that all powers of ℓ+2 are congruent 1 modulo ℓ+1 and that the segment contained exactly ℓ digits, so, the remainder of z modulo ℓ+1 corresponds with the number of occurences of powers of 2 in the original binary segment.

We will now formulate a lemma that shows us that this procedure actually satisfies some properties needed to count the ones in the binary representation of a number.

3.8 Lemma. $I\Delta_0$ proves the following statements:

i. $\forall$ x, i, n, z, z', w, w' ([rsp(x, i, i+n, n+1, n+2, z, w)$\wedge$
rsp(x, i, i+n, (n+1)+1, (n+1)+2, z', w')]$\rightarrow$ w = w').

ii. $\forall$ x, i, j, n, z, z', z'', w, w', w'' ([j $\leq$ n $\wedge$ rsp(x, i, i+j, n+1, n+2, z, w) $\wedge$
rsp(x, i+j , i+n, n+1, n+2, z', w') $\wedge$ rsp(x, i, i+n, n+1, n+2, z'', w'')] $\rightarrow$
w'' = w + w').

iii. $\forall$ x, i, n, z, z', w, w' ([rsp(x, i, i+n, n+1, n+2, z, w)$\wedge$
rsp(2.x, i+1, i+n+1, n+1, n+2, z', w')]$\rightarrow$ w=w').

iv. $\forall$ x, i, n, z, z', w, w' ([rsp(x, i, i+n, n+1, n+2, z, w)$\wedge$
rsp(2.x+1, i+1, i+n+1, n+1, n+2, z', w')]$\rightarrow$ w=w').

v. $\forall$ x, i, n, z, z', w, w' ([rsp(x, 0, n, n+1, n+2, z, w)$\wedge$
rsp(2.x+1, 0, n+1, n+2, n+3, z', w')]$\rightarrow$ w+1=w')

Proof. The proofs of the various parts of this lemma are carried out by a tedious kind of induction on x and n, using claims 3.5 and 3.6 whenever necessary.

As we had already pointed out above, this is not yet sufficient for a Δ_0 - definition. To obtain that result, we need to have bounds on the number z that occurs while carrying out the procedure. Our method will be the following: for every x we will simply take the greatest number n that has the property that the sum of powers of n+2 we have to use if the powers of 2 occurring in a binary segment of n digits are to be replaced by powers of n+2 is still smaller than x itself. Next, we will chop up the number x in segments that have length n. For convenience we will refer to such a number with the term "counting number". Observing that the worst case occurs if the binary representation of a segment containing n digits shows only 1's, the following definition meets our demands:

3.9 Definition. "n is a counting number for x (counts(n, x))":
counts(n, x):=

$$\exists\, z(z< (n+1).x+1 \wedge\ z=(n+2)^n) \wedge \neg\exists\, z'\ (z'< (n+2).x+1 \wedge z'=(n+3)^{n+1}).$$

What the definition of counting number actually says, is that the number n is the biggest number below x, such that $1 + (n+2) + ... + (n+2)^{n-1}$ is still below x.

This definition may look slightly suspicious, since expressions like $z=(n+2)^n$ occur in it. However, one should bear in mind the fact that this predicate is only applied to numbers below (n + 2).x. Therefore the definition is a sound Δ_0 -definition. However, we also require $I\Delta_0$ to prove some of the properties of this so-called counting number. This consideration gives reason to formulate the following claim.

3.10 Claim. $I\Delta_0$ proves the following statement:
"for all n, if n^n exists, then $(n+1)^{n+1}$ exists".

Proof. Observe that we only have to show that we can find a polynomial bound for $(n+1)^{n+1}$ expressed in n and n^n. The proof follows in a trivial way from the fact that we assumed our definition for exponentiation to behave properly. Reason in $I\Delta_0$: if $m = n^n$, then $m^2 = (n^2)^n$. But as $n + 1 < n^2$ holds for $n \geq 2$, we get: $(n+1)^n < m^2$.
So, $(n+1)^{n+1} < (n+1).m^2$ which clearly gives a polynomial bound.

The fact that $I\Delta_0$ is capable of reasoning about $(n+2)^{n+1}$ provided that n is a counting number for x, has an interesting consequence. This is formulated in the following claim.

3.11 Claim. $I\Delta_0$ proves the following statement:
for all n, x, "if $n \geq 3$ is a counting number for x, then $\lceil \log(x) \rceil < 2^n$ ".

Proof. From the last claim it follows that $I\Delta_0$ proves the existence of $(n+2)^{n+1}$. Since the definition of counting number stipulated that n was the biggest number such that $1 + (n+2) + ... + (n+2)^{n-1}$ was below x, we can conclude: $x \leq 1 + (n+3) + ... + (n+3)^n$.
From this follows: $(n+3)^{n+1} - 1 \geq (n+2).x$. Now we can infer:

$$\begin{aligned}\lceil \log(x) \rceil &\leq (n+1).\lceil \log(x) \rceil - \lceil \log(n+2) \rceil + 1 \\ &\leq n.\lceil \log(n+3) \rceil + 2 \\ &\leq 2^n \text{ for } n \geq 3.\end{aligned}$$

At this point we need some lemmas which relate the values of a counting number n of x with the length of x itself.

3.12 Lemma. $I\Delta_0$ proves: $\forall$ n, x$\neq$0 (counts(n, x)$\rightarrow$ $2^n \leq 2.x$).

The proof is straightforward by induction on n and x, observing the fact that any counting number for x will satisfy the property: $2^{n-1} \leq (n+2)^{n-1} \leq x$. Since all these values occur below x, the statement is provable by means of Δ_0-induction.

An even more trivial lemma is the following:

3.13 Lemma. $I\Delta_0$ proves: $\forall$ x $\exists$! n counts(n, x).

The proof of this lemma is also by straightforward induction on x.

Now we will define the number of segments of length equal to the counting number for x that emerges if we split up x in segments of that length.

3.14 Definition. "k bounds the counting of x by means of n (bound (k, x, n))":
bound(k, x, n):= $0 < k \wedge n.(k-1) \leq \lceil \log(x) \rceil < n.k$.

Note that, for any n, there can only be one such number k. Examining the proof of claim 3.11, we can infer the following:

3.15 Claim. $I\Delta_0$ proves the following statement:
$\forall$n, x ((counts(n, x) $\wedge$ bound(k, x, n))$\rightarrow$k<n).

Proof. This claim is a direct consequence of claim 3.11.

The next claim tells us that we can correctly apply the procedure "rsp" k times by means of number n to a number x if n is the counting number for x and if k bounds the counting of x by means of the same n.

3.16 Claim. $I\Delta_0$ proves the following statement:
$\forall$n, x, k, i $\exists$ z $\leq$x $\exists$ w $\leq$n [i<k$\wedge$bound(k, x, n)$\wedge$ counts(n, x)$\rightarrow$
rsp(x, i.n, (i+1).n, n+1, n+2, z, w)].

Proof. The proof is by putting together the definitions of the counting number and the lemmata about the result of the sum of powers and claim 3.6.

The next definition concerns the construction of a witness y that keeps track with the piece-wise counting of the number of ones in the binary representation of x.

3.17 Definition. "y is the progression of added sums of ones in the binary expansion of x, counted by means of n (countr(y, x, n))":

countr(y, x, n) :=

$\exists\, k \leq x$ (counts(n, x) $\wedge$ bound(k, x, n) $\wedge$ $y < 2^{k.n} \wedge$

$\forall\, i, j < k\ \forall t \leq y$ [Bseg(t, y, i.n, (i+1).n) $\leftrightarrow$

(i=0 $\wedge$ $\exists\, z \leq x$ (rsp(x, 0, n, n+1, n+2, z, t))$\vee$

(i=j+1 $\wedge$ $\exists t', t'' \leq y\ \exists\, z \leq x$ (Bseg(t', y, j.n, (j+1).n)

$\wedge$ rsp(x, i.n, (i+1)n, n+1, n+2, z, t'')$\wedge$ t = t' + t''))]).

The fact that, for each x, there is at most one y such that countr(y, x, n) holds if n is the counting number for x is formalized in the following lemma:

3.18 Lemma. $I\Delta_0$ proves the following statement:

$\forall$x, n, y, y' [(countr(y, x, n) $\wedge$ countr(y', x, n)) $\rightarrow$ y = y'].

Proof. Reason in $I\Delta_0$. Suppose countr(y, x, n) $\wedge$ countr(y', x, n) for x, n, y, y' $\in M$. If y$\neq$y', then there should be a least number, say i, such that Bseg(t, y, i.n, (i+1).n) $\wedge$ Bseg(t', y', i.n, (i+1).n) holds and also t$\neq$t'. We now proceed by induction below the number k, for which bound(k, x, n) holds. If i=0, then $\exists\, z \leq x$ (rsp(x, 0, n, n+1, n+2, z, t) should hold, as well as $\exists\, z' \leq x$ (rsp(x, 0, n, n+1, n+2, z', t'). But then we also have t=t' in (by lemma 3.8). On the other hand, if i is a successor, say j+1, then there is a u, such that Bseg(u, y, j.n, (j+1).n) and Bseg(u, y', j.n, (j+1).n) both hold, with as a result that, since there is a unique t'' such that $\exists\, z \leq x$ rsp(x, i.n, (i+1)n, n+1, n+2, z, t'') holds, we have t=u+t''=t'. In either case t$\neq$t' is contradicted.

The next definition gives the final result.

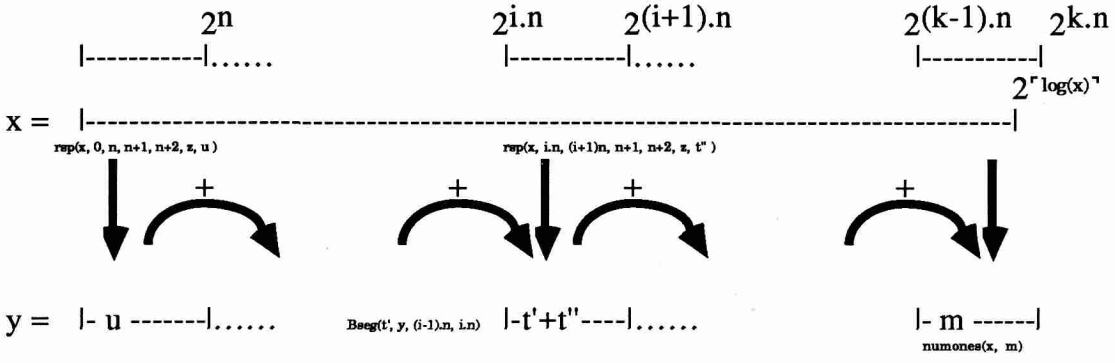
3.19 Definition. "m is the number of ones occurring in the binary representation of x (numones(x, m))":

numones(x, m):= (x = m = 0) $\vee$... $\vee$ (x = 31 $\wedge$ m = 5) $\vee$

$\exists\, y, n, k \leq 4.x^2$ [countr(y, x, n)$\wedge$ bound(k, x, n)$\wedge$

Bseg(m, y, (k-1).n, k.n)].

The first part of this definition sums up the actual number of ones in the binary representation of x for $0 \leq x \leq 31$. As to the second part, this definition only stipulates that the number of ones is exactly the number that is contained at the end of y, the witness in which these numbers were stored. The construction is visualized in the picture below:



What remains to be seen still, is whether this definition gives the right outcome on the standard model. If we use the symbol # naively, that is, to indicate the number of ones occurring in the binary representation of a number x, then we wish to prove the following theorem that represents uniqueness and soundness of the definition and verifies it's recursive properties:

3.20 Theorem.

i. $I\Delta_0 \vdash$ numones(0, 0) $\wedge$
$\forall x, y$ (numones(x, y) $\rightarrow$ numones(2.x, y) $\wedge$ numones(2.x + 1, y + 1));

ii. $I\Delta_0 \vdash \forall x, y_1, y_2$ (numones(x, y_1) $\wedge$ numones(x, y_2) $\rightarrow y_1 = y_2$) ;

iii. $\forall a, b \in \mathbb{N}, \mathbb{N} \models$ # (a) = b if and only if $I\Delta_0 \vdash$ numones(a, b).

Since iii follows from i by plain induction and ii is an immediate consequence of the last lemma, we will concentrate on i.

Proof. The fact that $I\Delta_0$ proves numones(0, 0) follows from the definition of numones. The same holds for all numbers up to 31. The fact that numones(32, 1) holds is easily verified. The induction procedure however, demands a very precise proof. From this point onwards we will reason in a (non-standard) model M of $I\Delta_0$. Suppose that numones(x, y) holds for $x > 31$, $y \in M$. Our strategy will be the following: we will distinguish two cases, the one in which the counting number for 2.x and 2.x+1 is the

same as it was for x and the case in which this counting number is bigger than that for x. We observe that the counting number n for x is the unique number with the property: $(n+1)^{n-1}-1<n.x<(n+1).x\leq(n+2)^n-1$. With some calculation we can now easily derive the fact: $(2.x+1).(n+2)\leq(n+3)^{n+1}-1$, which shows that, if the counting number is to go up, then it will go up only by one.

Assume numones(x, m) to hold in *M*. We have to prove that numones(2.x, m) and numones(2.x+1, m+1) will hold in *M*. We will pass immediately to the complicated case, that is the one in which the counting number for 2.x+1 is not the same as the counting number for x.

Suppose the counting number has gone up by one for 2.x+1. This means that there are y_1, $n \in M$ such that $M \models$ countr(y_1, x, n) $\wedge$ counts(n+1, 2.x+1). Now let k_1, $k_2 \in M$ be such that $M \models$ bound(k_1, x, n) $\wedge$ bound(k_2, 2.x+1, n+1). First, we have to prove that there is a y_2 in *M* such that countr(y_2, 2.x+1, n+1) holds in the same model. Given y_1, we will define y_2 inside *M* by stipulating that y_2 is the least element in *M* smaller than $2^{k_2.(n+1)}$ that has the following property:

$\forall j < k_2 \ \forall t \leq y_2$ Bseg(t, y_2, j.(n+1), (j+1).(n+1)) $\leftrightarrow$

$\exists u \leq y_1 \ \exists z, u' \leq x$ (Bseg(u, y_1, j.n, (j+1).n) $\wedge$

rsp(x, (j+1).n, (j+1).(n+1)-1, n+1, n+2, z, u') $\wedge$ t = 1+ u + u').

In order to show that such a number y_2 exists, we will verify that, if Bseg(u, y_1, j.n, (j+1).n) $\wedge$ rsp(x, (j+1).n, (j+1).(n+1)-1, n+1, n+2, z, u') $\wedge$ t=1+ u+u' holds for u, u' and z in *M*, then $t < 2^{n+1}$ will hold too. This follows from $u \leq 2^n - 1$, since u was a binary segment in y_1 of length n, combined with the fact that $u' \leq n$. Furthermore, $y_2 \leq 4.(2.x+1)^2$ follows from:

$y_2 \leq 2^{k_2.(n+1)} \leq 2^{(k_2-1).(n+1)+(n+1)} \leq (2.x+1).2^{n+1} \leq (2.x+1).2.(2.x+1) \leq 4.(2.x+1)^2$.

The next part of this proof concerns the verification that countr(y_2, 2.x+1, n+1) holds in *M*. This is proved by induction below k_2.

For i=0. Let $z \leq 2.x+1$, $t \leq n+1$ be such that rsp(2.x+1, 0, (n+1), (n+1)+1, (n+1)+2, z, t) holds in *M*. Since n+1 is the counting number for 2.x+1, this z must be in *M*. By lemma 3.16, there is such t. We have to show that, in *M*, Bseg(t, y_2, 0, (n+1)) will hold too. Since y_1 is the progression of added sums in the binary expansion of x in *M*, we can take $u \leq y_1$, $z' \leq x$ be such that rsp(x, 0, n, n+1, n+2, z', u) holds in *M*. By definition, Bseg(u, y_1, 0, n) holds in *M* and t = u+1 by lemma 3.8.

As $\exists z \leq x$ rsp(x, n, n, n+1, n+2, z, 0) holds *a fortiori*, we now have, in *M*, taking u'=0:

$\exists u \leq y_1 \ \exists z, u' \leq x$ (Bseg(u, y_1, 0, n) $\wedge$ rsp(x, n, n, n+1, n+2, z, u') $\wedge$ t = 1+ u + u').

So, from the definition of y_2 in *M*, we can infer: Bseg(t, y_2, 0, (n+1)).

For $i=j+1<k_2$. Let $z \leq 2.x+1$, $t \leq n+1$, $t' \leq 4.(2.x+1)^2$ be such that rsp(2.x+1, i.(n+1), (i+1).(n+1), (n+1)+1, ((n+1)+2), z, s) and

Bseg(t, y_2, j.(n+1), (j+1).(n+1)) hold in *M*. We have to show that, in *M*,
Bseg(s+t, y_2, i.(n+1), (i+1).(n+1)) holds too.
Let, to this end, u, v $\leq y_1, p_1, p_2, p_3, s' \leq x$, be such that the following statements hold in *M* (claim 3.16 guarantees the existence in *M* of these elements) :

Bseg(u, y_1, j.n, (j+1).n);
Bseg(v, y_1, i.n, (i+1).n);
$\exists z \leq x$ rsp(x, (j+1).n, (j+1).(n+1)-1, n+1, n+2, z, p_1);
$\exists z \leq x$ rsp(x, (j+1).(n+1)-1, (i+1).n, n+1, n+2, z, p_2);
$\exists z \leq x$ rsp(x, (i+1).n, (i+1).(n+1)-1, n+1, n+2, z, p_3);
$\exists z \leq x$ rsp(x, i.n, (i+1).n, n+1, n+2, z, s').

Clearly, v = s' + u will hold. Note that, by claim 3.15, p_2 exists, since $k_2 < n+1$, from which (j+1).(n+1)-1< (i+1).n follows. By lemma 3.8, we also have: $s'=p_1+p_2$. Let, on the other hand, p_2', p_3', $s'' \leq 2.x+1$ be such that the following three statements hold in *M* (as above, claim 3.16 guarantees the existence in *M* of these elements):

$\exists z \leq 2.x+1$ rsp(x, (j+1).(n+1)-1, (i+1).n, n+2, n+3, z, p_2');
$\exists z \leq 2.x+1$ rsp(x, (i+1).n, (i+1).(n+1)-1, n+2, n+3, z, p_3');
$\exists z \leq 2.x+1$ rsp(x, (j+1).(n+1)-1, (i+1).(n+1)-1, n+2, n+3, z, s'');

By definition of y_2, we will have that Bseg(1+v+p_3, y_2, i.(n+1), (i+1).(n+1)) is true in *M*. Now, by lemma 3.8, we have: $s''=p_2'+p_3'$, $p_2=p_2'$ and $p_3=p_3'$ and s=s'' hold in *M*. Therefore we can conclude with the following argument:

$$\begin{aligned} s+t &= s + (1+u+p_1) \\ &= (p_2+p_3)+(1+u+p_1) \\ &= 1+u+s'+p_3 \\ &= 1+v+p_3. \end{aligned}$$

Now that countr(y_2, 2.x+1, n+1) has been verified for the said y_2, it should still be checked that Bseg(m+1, y_2, (k_2-1).(n+1), k_2.(n+1)) holds in *M*.
Let q, v $\leq y_1$ be such that the following two statements hold in *M*:

$\exists z \leq x$ rsp(x, k_2.n, k_2.(n+1)-1, n+1, n+2, z, q);
Bseg(v, y_1, (k_2-1).n, k_2.n).

We can deduce: Bseg(v+q+1, y_2, (k_2-1).(n+1), k_2.(n+1)) holds in *M*. The fact that v+q=m in *M*, follows from the observation that there is no more one in the binary expansion of x beyond the k_2.(n+1)-1-th digit.
As to the other cases, if the counting number has gone up by one for 2.x, then the proof is analogous to the one for 2.x+1, except for the fact that the +1 clause is omitted in the definition of the progression of added sums of ones in the binary expansion of 2.x as well as in the rest of the proof. On the other hand, if the counting number did not change at all, then the progressions of added sums of ones in the binary expansion of 2.x and of 2.x+1 may be defined directly from the ones already existing for x.

From theorem 3.20, we obtain by a simple inductive argument the following corollary:

3.21 Corollary. $I\Delta_0$ proves the following statements:
i. $\forall x, z, y\ [(\text{numones}(x, y) \wedge \text{Pow}(z, 2)) \rightarrow \text{numones}(z.x, y)]$;
ii. $\forall x, z, y\ [\text{numones}(x, y) \wedge \text{numones}(z, y') \rightarrow \text{numones}(x + 2^{\ulcorner \log(2.x) \urcorner}.z, y+y')]$.

Proof. i. This statement follows by induction from the first part of theorem 3.20.
ii. Reason in $I\Delta_0$ for fixed z, y' such that numones(z, y') holds. We will prove by induction:
numones$(x + 2^{\ulcorner \log(2.x) \urcorner}.z, y+y')$ for all x such that numones(x, y).
$x = 0$. As $2^{\ulcorner \log(2.0) \urcorner} = 2^{\ulcorner \log(0) \urcorner} = 1$ by the preliminary definitions, this case is trivial.
Induction step. By induction hypothesis we have numones$(u + 2^{\ulcorner \log(2.u) \urcorner}.z, y+y'')$ for y'' such that numones(u, y'').
$x = 2.u$. By theorem 3.20, numones(x, y'') holds. If $u = 0$ then $x = u$ and the case reduces to the former one. In the other case we have: $2^{\ulcorner \log(2.x) \urcorner} = 2.2^{\ulcorner \log(2.u) \urcorner}$, hence $x + 2^{\ulcorner \log(2.x) \urcorner}.z = 2.(u + 2^{\ulcorner \log(2.u) \urcorner}.z)$ and numones$(x + 2^{\ulcorner \log(2.x) \urcorner}.z, y+y'')$ holds by theorem 3.20.
$x = 2.u + 1$. By theorem 3.20, numones(x, y''+1) holds. Since $2^{\ulcorner \log(2.x) \urcorner} = 2.2^{\ulcorner \log(2.u+1) \urcorner}$, we will get: $x + 2^{\ulcorner \log(2.x) \urcorner}.z = 1 + 2.(u + 2^{\ulcorner \log(2.u+1) \urcorner}.z)$. But also: $2^{\ulcorner \log(2.u+1) \urcorner} = 2^{\ulcorner \log(2.u) \urcorner}$. Therefore: $x + 2^{\ulcorner \log(2.x) \urcorner}.z = 1 + 2.(u + 2^{\ulcorner \log(2.u) \urcorner}.z)$. So, by theorem 3.20, we will obtain numones$(x + 2^{\ulcorner \log(2.x) \urcorner}.z, y+y''+1)$.

4. Formalization of a Δ_0 definition for finite sequences in $\mathbb{N}$.

At this point we may formalize the requested quintuple $< \in \text{Fseq}, \varnothing, *, <.>, \text{lth}(.) >$ and show that the demands imposed on it in the introduction are actually fulfilled.

4.1. Definition."z codes a finite sequence of natural numbers ($z \in \text{Fseq}$)":

$$z \in \text{Fseq} := \exists x_1, x_2 \leq z\ (z = \langle x_1, x_2 \rangle \wedge \ulcorner \log(2.x_1) \urcorner \leq \ulcorner \log(2.x_2) \urcorner \wedge \forall i \leq \ulcorner \log(x_2) \urcorner ((i+1 \in_1 x_2 \wedge \neg(i+1 \in_1 x_1)) \rightarrow i \in_1 x_2).$$

We recall that we took $\langle a, b \rangle := \frac{1}{2}(a+b+1)(a+b)+a$ to be the pairing function. It should be remarked that the last clause of this definition is intended to exclude the occurence of redundant zeroes in x_1.

4.2. Definition. "$\varnothing$ is the empty sequence ($\varnothing$)":
$\varnothing := 0$.

4.3. Definition."z codes the sequence that originates by concatenating x with y ($z = x*y$)":
$z=x*y := x\in Fseq \wedge y\in Fseq \wedge \exists z_1, z_2 \leq z\, \exists x_1, x_2 \leq x\, \exists y_1, y_2 \leq y$
$(x = \langle x_1, x_2\rangle \wedge y = \langle y_1, y_2\rangle \wedge z = \langle z_1, z_2\rangle \wedge$
$z_1 = x_1 + 2^{\lceil \log(2.x_2)\rceil}.y_1 \wedge z_2 = x_2 + 2^{\lceil \log(2.x_2)\rceil}.y_2)$.

4.4. Definition."y codes singleton x ($y = \langle x\rangle$)":
$y = \langle x\rangle := \exists y_1, y_2 \leq y\, (y = \langle y_1, y_2\rangle \wedge$
$y_1 = x \wedge y_2 = 2^{\lceil \log(x)\rceil})$.

4.5. Definition."y is the length of the finite sequence x ($y = lth(x)$)":
$y = lth(x) := x\in Fseq \wedge \exists x_1, x_2 \leq x\, (x = \langle x_1, x_2\rangle \wedge numones(x_2, y))$.

We will now verify that these definitions satisfy the demands that we had listed in the introduction.

4.6. Theorem. The following statements are provable in $I\Delta_0$:
a. $\forall s \in Fseq\ s*\varnothing = \varnothing*s = s$;
b. $lth(\varnothing) = 0$;
c. $\forall s, t \in Fseq\ lth(s*t) = lth(s)+lth(t)$;
d. $\forall y, x\ (y = \langle x\rangle \rightarrow lth(y) = 1)$;
e. $\forall x \exists!\ y\ (y = \langle x\rangle \wedge y \in Fseq)$;
f. $\forall s, s'\ (s, s' \in Fseq \rightarrow \exists!\ t \in Fseq \wedge s*s' = t)$;
g. lth(.) is total on all $s \in Fseq$;
h. * is associative, in the sense that $(s*t)*u = s*(t*u)$ for all $s, t, u \in Fseq$.

Proof. Reason in $I\Delta_0$.
a. First of all, it is readily observed that $\varnothing\in Fseq$, for $0 = \frac{1}{2}(0+0+1)(0+0)+0 = \langle 0,0\rangle$ and $\lceil \log(2.0)\rceil = 0 = \lceil \log(2.0)\rceil$ and there is no power of 2 that occurs in the number 0. Suppose $t = s*\varnothing$ and $u = \varnothing*s$, for $s\in Fseq$. Clearly, we will have:
$t_1 = s_1 + 2^{\lceil \log(2.s_2)\rceil}.0 \wedge t_2 = s_2 + 2^{\lceil \log(2.s_2)\rceil}.0$ and
$u_1 = 0 + 2^{\lceil \log(2.0)\rceil}.s_1 \wedge u_2 = 0 + 2^{\lceil \log(2.0)\rceil}.s_2$.
So we will have: $t_1 = s_1$, $t_2 = s_2$, $u_1 = s_1$ and $u_2 = s_2$. Therefore, $t = s = u$.
b. Immediate by theorem 3.20.

c. Let s, t , u $\in$ Fseq be such that u = s*t. We have to show that lth(u) = lth(s)+lth(t). Since u = s*t, there are u_1, u_2 such that u = $<u_1, u_2>$ and $u_1 = s_1 + 2^{\lceil \log(2.s_2) \rceil}.t_1$and $u_2 = s_2 + 2^{\lceil \log(2.s_2) \rceil}.t_2$. Let n_s and n_t be such that numones(s, n_s) and numones (t, n_t) hold. From corollary 3.21 numones(u_2, $n_s + n_t$) follows.

d. Suppose y = <x>. In this case Pow(y_2, 2) holds for y_2 such that there is a y_1 such that y = $<y_1, y_2>$. But then numones (y_2, 1) will also hold (this can directly be inferred from corollary 3.21 and the fact that numones(1, 1) is trivially provable.

e. As $I\Delta_0$ proves $\forall x \exists y \leq x+1 (y = 2^{\lceil \log(x) \rceil})$, we can find the numbers needed to construct the pair $<y_1, y_2>$ for <x> bounded by a polynomial in x. Uniqueness of this singleton function follows from uniqueness of the pairing function.

f. To prove that the concatenation function is total, we have to show that if z = x*y holds, then we can prove that such a z bounded by some polynomial in x and y can be found. Uniqueness follows from the fact that, given x= $<x_1, x_2>\in$ Fseq and y= $<y_1, y_2>\in$ Fseq, there can only be one pair z = $<z_1, z_2>$, if any, such that

$z_1 = x_1 + 2^{\lceil \log(2.x_2) \rceil}.y_1 \wedge z_2 = x_2 + 2^{\lceil \log(2.x_2) \rceil}.y_2$.

It is easily proved that $x_1 \leq 2.x_2$ holds if x = $<x_1, x_2>\in$ Fseq. This follows from the second clause in the definition of $\in$ Fseq which stipulates that $\lceil \log(2.x_1) \rceil \leq \lceil \log(2.x_2) \rceil$ holds. At this point, setting $z_1 = x_1 + 2^{\lceil \log(2.x_2) \rceil}.y_1$ and $z_2 = x_2 + 2^{\lceil \log(2.x_2) \rceil}.y_2$, for x = $<x_1, x_2>\in$ Fseq and y = $<y_1, y_2>\in$ Fseq we get:

$$z_1 \leq 2.x_2 + 2.x_2.2.y_2,$$

$$z_2 \leq x_2 + 2.x_2.y_2.$$

Therefore, applying the pairing function to the arguments z_1 and z_2:

$$z = <z_1, z_2> \leq \frac{1}{2}(3.x_2 + 6.x_2.y_2+1)(3.x_2 + 6.x_2.y_2)+2.x_2 + 4.x_2.y_2.$$

Hence, observing that $x_2 \leq x_2^2 \leq 2.x$ is a trivial consequence of the character of the pairing function, we obtain: z = $<z_1, z_2> \leq 16.x + 163.x.y$, which is indeed a polynomial in x and y. The last thing we have to check is that th z defined above does pertain to the set of numbers that code finite sequences.

First, we check: $\lceil \log(2.z_1) \rceil \leq \lceil \log(2.z_2) \rceil$. This follows from:

$\lceil \log(2.z_1) \rceil = \lceil \log(2.(x_1 + 2^{\lceil \log(2.x_2) \rceil}.y_1)) \rceil = \lceil \log(2.x_1) \rceil + \lceil \log(2.y_1) \rceil$ and

$\lceil \log(2.z_2) \rceil = \lceil \log(2.(x_2 + 2^{\lceil \log(2.x_2) \rceil}.y_2)) \rceil = \lceil \log(2.x_2) \rceil + \lceil \log(2.y_2) \rceil$.

Next, we check: $\forall i \leq \lceil \log(z_2) \rceil ((i+1 \in_1 z_2 \wedge \neg(i+1 \in_1 z_1)) \rightarrow i \in_1 z_2)$. There are two cases to distinguish. Let i be such that $i \leq \lceil \log(z_2) \rceil$.

1. $i < \lceil \log(x_2) \rceil$. If $i+1 \in_1 z_2 \wedge \neg(i+1 \in_1 z_1)$ holds, then we will also have: $i+1 \in_1 x_2 \wedge \neg(i+1 \in_1 x_1)$. So, $i \in_1 x_2$ since x= $<x_1, x_2>\in$ Fseq. But then, it will also be the case that $i \in_1 z_2$.
2. $\lceil \log(x_2) \rceil \leq i$. In this case, if $i+1 \in_1 z_2 \wedge \neg(i+1 \in_1 z_1)$ holds, we can infer: $(i+1) - \lceil \log(x_2) \rceil \in_1 y_2$ and $\neg((i+1) - \lceil \log(x_2) \rceil \in_1 y_1)$. In the same style as in the former case, we can now infer: $i - \lceil \log(x_2) \rceil \in_1 y_2$ and therefore $i \in_1 z_2$.

g. The fact that the length function is total is an immediate consequence of the first part of theorem 3.20, by means of which we can prove: $\forall x, y$ (numones(x, y) $\rightarrow y \leq x$).

h. Let z, y, x, u, v, r $\in$ Fseq, be such that $z = x*y$, $x = u*v$, $r = v*y$. In order to prove associativity we have to show: $z = u*r$. For $x= \langle x_1, x_2 \rangle$, $y= \langle y_1, y_2 \rangle$, $u= \langle u_1, u_2 \rangle$, $v= \langle v_1, v_2 \rangle$ and $r= \langle r_1, r_2 \rangle$ we have:

$$z_1 = x_1 + 2^{\lceil \log(2.x_2) \rceil}.y_1;$$
$$z_2 = x_2 + 2^{\lceil \log(2.x_2) \rceil}.y_2;$$
$$x_1 = u_1 + 2^{\lceil \log(2.u_2) \rceil}.v_1;$$
$$x_2 = u_2 + 2^{\lceil \log(2.u_2) \rceil}.v_2;$$
$$r_1 = v_1 + 2^{\lceil \log(2.v_2) \rceil}.y_1;$$
$$x_2 = v_2 + 2^{\lceil \log(2.v_2) \rceil}.y_2.$$

Rewriting this, we have:

$$z_1 = u_1 + 2^{\lceil \log(2.u_2) \rceil}.v_1 + 2^{\lceil \log(2.u_2) \rceil + \lceil \log(2.v_2) \rceil}.y_1;$$
$$z_2 = u_2 + 2^{\lceil \log(2.v_2) \rceil}.v_2 + 2^{\lceil \log(2.u_2) \rceil + \lceil \log(2.v_2) \rceil}.y_2, \text{ and thus:}$$
$$z_1 = u_1 + 2^{\lceil \log(2.u_2) \rceil}.(v_1 + 2^{\lceil \log(2.v_2) \rceil}.y_1);$$
$$z_2 = u_2 + 2^{\lceil \log(2.v_2) \rceil}.(v_2 + 2^{\lceil \log(2.v_2) \rceil}.y_2).$$

But this is exactly:

$$z_1 = u_1 + 2^{\lceil \log(2.u_2) \rceil}.r_1;$$
$$z_2 = u_2 + 2^{\lceil \log(2.v_2) \rceil}.r_2.$$

This completes the proof.

Concluding remarks.

It seems natural to consider some more demands that could be imposed on the predicates defined above to code finite sequences in $I\Delta_0$. Some demands are met by the method we have described, others surely are not. As to those that do not cause difficulties, we can remark that they include an important one, namely the provability of the following statement: $\forall x \in$ Fseq $(x = \emptyset \vee \exists y \in$ Fseq $\exists z\ (x= y*\langle z \rangle))$. As our definition of Fseq is not restricted to a specific class of numbers, we can also prove (in $I\Delta_0$) that the set of codes of finite sequences is unbounded. With our definition it is even possible to prove that finite sequences exist of length equal to x for any number x for which 2^x exists. These facts capture the natural character of our coding. On the other hand however, the hope to satisfy some other demands is destroyed by Parikh's theorem, like the one asserting that finite sequences of arbitrary length do exist. What the exact correspondance is between the impossibility to satisfy demands like the last one and the nature of the definitions used to code finite sequences is a matter of further research.

References.

Hájek, P. & Pudlák, P., 1990, *Metamathematics of first-order arithmetic,* draft, Mathematical Institute, Prague.

Kleene, S. C., 1974, *Introduction to Metamathematics*, Wolters-Noordhoff, Groningen.

Paris, J. B. & Wilkie, A. J., 1987, *Counting Δ_0 sets*, Fund. Math. Vol. 127, pp. 67-76.

Logic Group Preprint Series

Department of Philosophy, University of Utrecht
Heidelberglaan 8, 3584 CS Utrecht
The Netherlands

1 C.P.J. Koymans, J.L.M. Vrancken, *Extending Process Algebra with the empty process*, September 1985
2 J.A. Bergstra, *A process creation mechanism in Process Algebra*, September 1985
3 J.A. Bergstra, *Put and get, primitives for synchronous unreliable message passing*, October 1985
4 A. Visser, *Evaluation, provably deductive equivalence in Heyting's arithmetic of substitution instances of propositional formulas*, November 1985
5 G.R. Renardel de Lavalette, *Interpolation in a fragment of intuitionistic propositional logic*, January 1986
6 C.P.J. Koymans, J.C. Mulder, *A modular approach to protocol verification using Process Algebra*, April 1986
7 D. van Dalen, F.J. de Vries, *Intuitionistic free abelian groups*, April 1986
8 F. Voorbraak, *A simplification of the completeness proofs for Guaspari and Solovay's R*, May 1986
9 H.B.M. Jonkers, C.P.J. Koymans & G.R. Renardel de Lavalette, *A semantic framework for the COLD-family of languages*, May 1986
10 G.R. Renardel de Lavalette, *Strictheidsanalyse*, May 1986
11 A. Visser, *Kunnen wij elke machine verslaan? Beschouwingen rondom Lucas' argument*, July 1986
12 E.C.W. Krabbe, *Naess's dichotomy of tenability and relevance*, June 1986
13 H. van Ditmarsch, *Abstractie in wiskunde, expertsystemen en argumentatie*, Augustus 1986
14 A. Visser, *Peano's Smart Children, a provability logical study of systems with built-in consistency*, October 1986
15 G.R. Renardel de Lavalette, *Interpolation in natural fragments of intuitionistic propositional logic*, October 1986
16 J.A. Bergstra, *Module Algebra for relational specifications*, November 1986
17 F.P.J.M. Voorbraak, *Tensed Intuitionistic Logic*, January 1987
18 J.A. Bergstra, J. Tiuryn, *Process Algebra semantics for queues*, January 1987
19 F.J. de Vries, *A functional program for the fast Fourier transform*, March 1987
20 A. Visser, *A course in bimodal provability logic*, May 1987
21 F.P.J.M. Voorbraak, *The logic of actual obligation, an alternative approach to deontic logic*, May 1987
22 E.C.W. Krabbe, *Creative reasoning in formal discussion*, June 1987
23 F.J. de Vries, *A functional program for Gaussian elimination*, September 1987
24 G.R. Renardel de Lavalette, *Interpolation in fragments of intuitionistic propositional logic*, October 1987 (revised version of no. 15)
25 F.J. de Vries, *Applications of constructive logic to sheaf constructions in toposes*, October 1987
26 F.P.J.M. Voorbraak, *Redeneren met onzekerheid in expertsystemen*, November 1987
27 P.H. Rodenburg, D.J. Hoekzema, *Specification of the fast Fourier transform algorithm as a term rewriting system*, December 1987
28 D. van Dalen, *The war of the frogs and the mice, or the crisis of the Mathematische Annalen*, December 1987
29 A. Visser, *Preliminary Notes on Interpretability Logic*, January 1988
30 D.J. Hoekzema, P.H. Rodenburg, *Gauß elimination as a term rewriting system*, January 1988
31 C. Smoryński, *Hilbert's Programme*, January 1988
32 G.R. Renardel de Lavalette, *Modularisation, Parameterisation, Interpolation*, January 1988
33 G.R. Renardel de Lavalette, *Strictness analysis for POLYREC, a language with polymorphic and recursive types*, March 1988
34 A. Visser, *A Descending Hierarchy of Reflection Principles*, April 1988
35 F.P.J.M. Voorbraak, *A computationally efficient approximation of Dempster-Shafer theory*, April 1988
36 C. Smoryński, *Arithmetic Analogues of McAloon's Unique Rosser Sentences*, April 1988

37 P.H. Rodenburg, F.J. van der Linden, *Manufacturing a cartesian closed category with exactly two objects*, May 1988

38 P.H. Rodenburg, J.L.M.Vrancken, *Parallel object-oriented term rewriting : The Booleans*, July 1988

39 D. de Jongh, L. Hendriks, G.R. Renardel de Lavalette, *Computations in fragments of intuitionistic propositional logic*, July 1988

40 A. Visser, *Interpretability Logic*, September 1988

41 M. Doorman, *The existence property in the presence of function symbols*, October 1988

42 F. Voorbraak, *On the justification of Dempster's rule of combination*, December 1988

43 A. Visser, *An inside view of EXP, or: The closed fragment of the provability logic of* $I\Delta_0+\Omega_1$, February 1989

44 D.H.J. de Jongh & A. Visser, *Explicit Fixed Points in Interpretability Logic*, March 1989

45 S. van Denneheuvel & G.R. Renardel de Lavalette, *Normalisation of database expressions involving calculations*, March 1989

46 M.F.J. Drossaers, *A Perceptron Network Theorem Prover for the Propositional Calculus*, July 1989

47 A. Visser, *The Formalization of Interpretability*, August 1989

48 J.L.M. Vrancken, *Parallel Object Oriented Term Rewriting : a first implementation in Pool2*, September 1989

49 G.R. Renardel de Lavalette, *Choice in applicative theories*, September 1989

50 C.P.J. Koymans & G.R. Renardel de Lavalette, *Inductive definitions in COLD-K*, September 1989

51 F. Voorbraak, *Conditionals, probability, and belief revision (preliminary version)*, October 1989

52 A. Visser, *On the* Σ^0_1*-Conservativity of* Σ^0_1*-Completeness*, October 1989

53 G.R. Renardel de Lavalette, *Counterexamples in applicative theories with choice*, January 1990

54 D. van Dalen, *L.E.J. Brouwer. Wiskundige en Mysticus*, June 1990

55 F. Voorbraak, *The logic of objective knowledge and rational belief*, September 1990

56 J.L.M. Vrancken, *Reflections on Parallel and Functional Languages*, September 1990

57 A. Visser, *An inside view of EXP, or: The closed fragment of the provability logic of* $I\Delta_0+\Omega_1$, revised version with new appendices, October 1990

58 S. van Denneheuvel, K. Kwast, G.R. Renardel de Lavalette, E. Spaan, *Query optimization using rewrite rules*, October 1990

59 G.R. Renardel de Lavalette, *Strictness analysis via abstract interpretation for recursively defined types*, October 1990

60 C.F.M. Vermeulen, *Sequence Semantics for Dynamic Predicate Logic*, January 1991

61 M.B. Kalsbeek, *Towards the Interpretability Logic of* $I\Delta_0+EXP$, January 1991.

62 D. van Dalen, *I<R, Some Intuitionistic Elementary Equivalences*, February 1991.

63 M. Bezem, *Strong termination of Logic Programs*, March 1991.

64 A. Visser, *The Unprovability of Small Inconsistency*, March 1991.

65 C.M. Jonker, *On the Semantics of Conflict Resolution in Truth Maintenance Systems*, March 1991.

66 F. Voorbraak, *A Preferential Model Semantics for Default Logic*, July 1991.

67 M. Kracht, *Splittings and the finite model property*, October 1991.

68 M. Bezem, *Impredicative recursion: terms depending on order types (extended abstract)*, November 1991.

69 E. Barendsen, M. Bezem, *Bar recursion versus polymorphism (extended abstract)*, December 1991.

70 C.F.M. Vermeulen, *Merging without Mystery*, December 1991.

71 F. Voorbraak, *Generalized Kripke models for epistemic logic*, December 1991.

72 W. Veldman, M. Bezem, *Ramsey's theorem and the pigeonhole principle in intuitionistic mathematics*, January 1992.

73 M. Kracht, *Prefinitely axiomatizable modal and intermediate logics*, January 1992.

74 C. M. Jonker, G. R. Renardel de Lavalette, *A tractable algorithm for the wellfounded model*, February 1992.

75 M. Kracht, *The Theory of Syntactic Domains*, February 1992.

76 A. Visser, *Actions under Presuppositions*, February 1992.

77 P. Blackburn, E. Spaan, *A Modal Perspective on the Computational Complexity of Attribute Value Grammar*, April 1992.

78 F. Voorbraak, *Belief functions and inner measures*, June 1992.

79 M.D. Jumelet, *A Δ_0-definition for Finite Sequences*, September, 1992