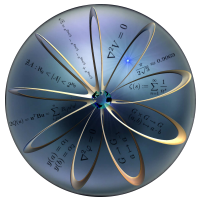

planetmath.org

Math for the people, by the people.

Δ_1 bootstrapping

Canonical name	Delta1Bootstrapping
Date of creation	2013-03-22 12:58:25
Last modified on	2013-03-22 12:58:25
Owner	Henry (455)
Last modified by	Henry (455)
Numerical id	10
Author	Henry (455)
Entry type	Example
Classification	msc 03B10

This proves that a number of useful relations and functions are Δ_1 in first order arithmetic, providing a bootstrapping of parts of mathematical practice into any system including the Δ_1 relations (since the Δ_1 relations are exactly the recursive ones, this includes Turing machines).

First, we want to build a tupling relation which will allow a finite sets of numbers to be encoded by a single number. To do this we first show that $R(a,b) \leftrightarrow a|b$ is Δ_1. This is true since $a|b \leftrightarrow \exists c \leq b(a \cdot c = b)$, a formula with only bounded quantifiers.

Next note that $P(x) \leftrightarrow$`x is prime` is Δ_1 since $P(x) \leftrightarrow \neg\exists y < x(\neg y = 1 \wedge y|x)$. Also $A_P(x,y) \leftrightarrow P(x) \wedge P(y) \wedge \forall z < y(x < z \rightarrow \neg P(x))$.

These two can be used to define (the graph of) a primality function, $p(a) = a + 1$-`th prime`. Let $p(a,b) = \exists c \leq b^{a^2}(\neg[2|c] \wedge [\forall q < b \forall r \leq b(A_P(q,r) \rightarrow \forall j < c[q^j|c \leftrightarrow r^{j+1}|c])] \wedge [b^a|c] \wedge \neg[b^{a+1}|c])$.

This rather awkward looking formula is worth examining, since it illustrates a principle which will be used repeatedly. c is intended to be a function of the form $2^0 \cdot 3^1 \cdot 5^2 \cdots$ and so on. If it includes b^a but not b^{a+1} then we know that b must be the $a+1$-th prime. The definition is so complicated because we cannot just say, as we'd like to, $p(a+1)$ is the smallest prime greater than $p(a)$ (since we don't allow recursive definitions). Instead we embed the series of values this recursion would take into a single number (c) and guarantee that the recursive relationship holds for at least a terms; then we just check if the a-th value is b.

Finally, we can define our tupling relation. Technically, since a given relation must have a fixed arity, we define for each n a function $\langle x_0, \ldots, x_n \rangle = \sum_{\leq n} p_i^{x_i+1}$. Then define $(x)_i$ to be the i-th element of x when x is interpreted as a tuple, so $\langle (x)_0, \ldots, (x)_n \rangle = x$. Note that the tupling relation, even taken collectively, is not total. For instance 5 is not a tuple (although it is sometimes convenient to view it as a tuple with "empty spaces": $\langle \_, \_, 5 \rangle$). In situations like this, and also when attempting to extract entries beyond the length, $(x)_i = 0$ (for instance, $(5)_0 = 0$). On the other hand there is a 0-ary tupling relation, $\langle\rangle = 1$.

Thanks to our definition of p, we have $\langle x_0, \ldots, x_n \rangle = x \leftrightarrow x = p(0)^{x_0+1} \cdot \cdots \cdot p(n)^{x_n+1}$. This is clearly Δ_1. (Note that we don't use the $\sum$ as above, since we don't have that, but since we have a different tupling function for each n this isn't a problem.)

For the reverse, $(x)_i = y \leftrightarrow ([p(i)^{y+1}|x] \wedge \neg[p(i)^{y+2}|x]) \vee ([y = 0] \wedge \neg[p(i)|x])$.

Also, define a length function by $\text{len}(x) = y \leftrightarrow \neg[p(y+1)|x] \wedge \forall z \leq y[p(z)|x]$ and a membership relation by $\text{in}(x,n) \leftrightarrow \exists i < \text{len}(x)[(x)_i = n]$.

Armed with this, we can show that all primitive recursive functions are Δ_1. To see this, note that $x = 0$, the zero function, is trivially recursive, as are $x = Sy$ and $p_{n,m}(x_1, \ldots, x_n) = x_m$.

The Δ_1 functions are closed under composition, since if $\phi(\vec{x})$ and $\psi(\vec{x})$ both have no unbounded quantifiers, $\phi(\psi(\vec{x}))$ obviously doesn't either.

Finally, suppose we have functions $f(\vec{x})$ and $g(\vec{x}, m, n)$ in Δ_1. Then define the primitive recursion $h(\vec{x}, y)$ by first defining:

$$\bar{h}(\vec{x}, y) = z \leftrightarrow ln(z) = y \wedge \forall i < y[(z)_{i+1} = g(\vec{x}, i, (z)_i)] \wedge [ln(z) = 0 \vee (z)_0 = f(\vec{x})$$

and then $h(\vec{x}, y) = (\bar{h}(\vec{x}, y))_y$.

Δ_1 is also closed under minimization: if $R(\vec{x}, y)$ is a Δ_1 relation then $\mu y.f(\vec{x}, y)$ is a function giving the least y satisfying $R(\vec{x}, y)$. To see this, note that $\mu y.f(\vec{x}, y) = z \leftrightarrow f(\vec{x}, z) \wedge \forall m < z \neg f(\vec{x}, m)$.

Finally, using primitive recursion it is possible to concatenate sequences. First, to concatenate a single number, if $s = \langle x_0, \ldots x_n \rangle$ then $s *_1 y = t \cdot p(\text{len}(s) + 1)^{y+1}$. Then we can define the concatenation of s with $t = \langle y_0, \ldots, y_m \rangle$ by defining $f(s, t) = s$ and $g(s, t, j, i) = j *_1 (t)_i$, and by primitive recursion, there is a function $h(s, t, i)$ whose value is the first j elements of t appended to s. Then $s * t = h(s, t, \text{len}(t)$.

We can also define $*_u$, which concatenates only elements of t not appearing in s. This just requires defining the graph of g to be $g(s, t, j, i, x) \leftrightarrow [\text{in}(s, (t)_i) \wedge x = j] \vee [\neg \text{in}(s, (t)_i) \wedge x = j *_1 (t)_i]$