CDM

Coding Systems

KLAUS SUTNER

CARNEGIE MELLON UNIVERSITY

FALL 2023

1 **Coding the World**

2 **Gödel's β Function**

3 **Coding Systems**

Faking Data structures

Our primitive recursive programming language has one glaring defect: it only supports one data type, $\mathbb{N}$. There are no lists, trees, graphs, hash tables and so on, only natural numbers.

As it turns out, all these discrete structures can be obtained from just integers if we are able to express sequences $a_0, a_1, \ldots, a_{n-1}$ of numbers as a single number $\langle a_0, a_1, a_2, \ldots, a_{n-1} \rangle$.

This is obviously not meant as a practical programming idea, it is purely conceptual: natural numbers already suffice in principle, and the ability to compute with them means that other computation involving, say, lists, are also possible.

Algorithms in the RealWorld$^{\text{TM}}$

We claim that any algorithm you will ever see, outside of a class dealing directly with logic and computability, is always primitive recursive. And, in fact, often trivially so.

There are two parts to this claim:

- All these algorithms operate on finitary data structures that can be coded as natural numbers, and
- given this coding, for input as well as output, the corresponding arithmetic functions are always primitive recursive.

Of course, there is no actual theorem here, just an observation. I'd be most curious to hear about anything that might contradict this claim[†].

[†] I will change my definition of RealWorld$^{\text{TM}}$

Leopold Kronecker, Semi-Constructivist

Die ganzen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk.

“Dear god” made the integers, everything else is the work of men.

Coding

Write $\mathbb{N}^\star$ for the set of all finite sequences of natural numbers and nil for the empty sequence.

We would like to express a sequence $a_0, a_1, \ldots, a_{n-1} \in \mathbb{N}^\star$ as a single number $\langle a_0, a_1, \ldots, a_{n-1} \rangle$. So we need a coding function, a multiadic map of the form

$$\langle . \rangle : \mathbb{N}^* \to \mathbb{N}$$

that allows us to decode: from $b = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ we can recover n as well as all the a_i.

Note that any coding function must necessarily be injective. Moreover, both the coding and decoding operations should be computationally cheap, at least primitive recursive.

Decoding

Suppose

$$b = \langle a_0, a_1, a_2, \ldots, a_{n-1} \rangle$$

is some code number. Note that we have used 0-indexing to simplify notation below.

We want a unary length function len : $\mathbb{N} \to \mathbb{N}$ that determines the length of the coded sequence

$$\mathsf{len}(b) = n$$

and a binary decoding function dec : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ that extracts the components:

$$\mathsf{dec}(b, i) = a_i$$

for all $i = 0, \ldots, n-1$. Traditionally, dec(b, i) is written $(b)_i$.

Computable Coding

Again, we need three functions:

$$\langle . \rangle : \mathbb{N}^\star \to \mathbb{N}$$

$$\mathsf{dec} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

$$\mathsf{len} : \mathbb{N} \to \mathbb{N}$$

In the set-theoretic universe, the existence of these functions is entirely trivial: $\mathbb{N}^\star$ is countable.

But we live in the computational universe: we need these functions to be easily computable, and in particular primitive recursive. More precisely, we want dec and len to be primitive recursive ($\langle . \rangle$ is multiadic and simply cannot be primitive recursive).

Sequence Numbers

The numbers of the form $\langle a_0, a_1, a_2, \ldots, a_{n-1} \rangle$ that appear as codes of sequences are called sequence numbers.

Note that a priori $\text{len}(x)$ need not be defined when x is not a sequence number. The same is true for $\text{dec}(x, i)$, plus i may be too large to make sense. Still, one usually insists that both decoding functions are total and return some default value like 0 for meaningless arguments.

Exercise

Suppose you are given dec *and* len*. Can you check whether a number is a sequence number?*

Pairs

The first step is to select a pairing function, an injective map $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

More precisely, we are looking for 3 functions $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\pi_i : \mathbb{N} \to \mathbb{N}$, $i = 1, 2$, such that

$$\pi_i(\pi(x_1, x_2)) = x_i$$

There are many possibilities, the following choice arguably yields one of the most intuitive coding functions.

$$\pi(x, y) = 2^x(2y + 1)$$

For instance

$$\pi(5, 27) = 32 \cdot 55 = 1760 = 110111\,00000_2$$

In Binary

Note that the binary expansion of $\pi(x, y)$ looks like so:

$$\underbrace{y_k y_{k-1} \dots y_0 \, 1}_{2y+1} \; \underbrace{00 \dots 0}_{x}$$

where $y_k y_{k-1} \dots y_0$ is the standard binary expansion of y (y_k is the most significant digit). Hence the range of π is $\mathbb{N}_+$ (but not $\mathbb{N}$).

This makes it easy to find the corresponding unpairing functions:

$$x = \pi_1(\pi(x, y)) \qquad\qquad y = \pi_2(\pi(x, y)).$$

Aside: Fueter-Pólya

Another popular pairing function is the quadratic polynomial due to Cantor:

$$p(x, y) = ((x + y)^2 + 3x + y)/2$$

Note that this function is a bijection (unlike our exponential pairing function which misses 0).

A surprising theorem by Fueter and Pólya from 1923 states that, up to a swap of variables, this is the only quadratic polynomial that defines a bijection $\mathbb{N}^2 \leftrightarrow \mathbb{N}$.

The proof is rather difficult and uses the fact that e^a is transcendental for algebraic $a \neq 0$.

It is an open problem whether there are other bijections for higher degree polynomials. Extra Credit.

Extending to Sequences

$$
\begin{aligned}
\langle \mathsf{nil} \rangle &\coloneqq 0 \\
\langle a_0, \ldots, a_{n-1} \rangle &\coloneqq \pi(a_0, \langle a_1, \ldots, a_{n-1} \rangle)
\end{aligned}
$$

Here are some sequence numbers for this particular coding function:

$$
\begin{aligned}
\langle 10 \rangle &= 1024 \\
\langle 0, 0, 0 \rangle &= 7 \\
\langle 1, 2, 3, 4, 5 \rangle &= 532754
\end{aligned}
$$

It's a Bijection

Lemma

$\langle . \rangle : \mathbb{N}^* \to \mathbb{N}$ is a bijection.

Proof. Suppose

$$\langle a_0, \ldots, a_{n-1} \rangle = \langle b_0, \ldots, b_{m-1} \rangle$$

We may safely assume $0 < n \leq m$ (why?).

Since π is a pairing function, we get $a_0 = b_0$ and $\langle a_1, \ldots, a_{n-1} \rangle = \langle b_1, \ldots, b_{m-1} \rangle$.

By induction, $a_i = b_i$ for all $i = 1, \ldots, n-1$ and $0 = \langle \mathsf{nil} \rangle = \langle b_n, \ldots, b_{m-1} \rangle$. Hence $n = m$ and our map is injective.

Exercise

Prove that the function is surjective.

Less formally ...

Here is a sequence number and its binary expansion:

$$
\begin{aligned}
\langle 2,3,5,1 \rangle &= 20548 \\
&= 1\underbrace{\;0\;}_{1}1\underbrace{00000}_{5}1\underbrace{000}_{3}1\underbrace{00}_{2}
\end{aligned}
$$

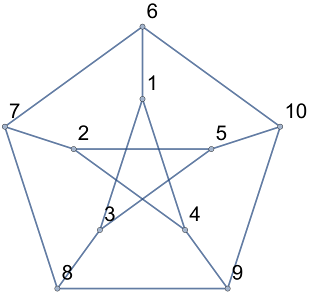
So the number of 1's (the digitsum) is just the length of the sequence, and the spacing between the 1's indicates the actual numerical values.

It follows that the coding function is injective and surjective, right?

That's It!

We can now code any discrete structure as an integer by expressing it as a nested list of natural numbers, and then applying the coding function.

For example, the so-called Petersen graph on the left is given by the nested list on the right.



$$((1,3),(1,4),(2,4),(2,5),(3,5),\\ (6,7),(7,8),(8,9),(9,10),(6,10),\\ (1,6),(2,7),(3,8),(4,9),(5,10))$$

Sequence Numbers

The edges have sequence numbers

$$\begin{gathered}34, 66, 258, 132, 260, 1028, 520, 4104, 16400,\\ 65568, 16448, 131136, 65664, 262400, 1049088\end{gathered}$$

Alas, the sequence number for this list is about

$$3.210742533937650 \times 10^{485597}$$

and has almost half a million digits. In the interest of brevity, we won't write it down.

Clearly, these coding techniques have nothing to do with algorithms, it is a tool to explore the power and limits of computation.

Everything is PR

Exercise

Show that the pairing function π *and both unpairing functions* $x = \pi_1(\pi(x, y))$ *and* $y = \pi_2(\pi(x, y))$ *are primitive recursive.*

Exercise

Show that the length and decoding functions len *and* dec *are primitive recursive.*

Exercise

Show that the coding function $\langle . \rangle$ *is primitive recursive when restricted to inputs of fixed length.*

Recording History

One neat application of sequence numbers is course-of-value recursion. First note that ordinary primitive recursion can be expressed in terms of sequence numbers like so:

$$f(x,\boldsymbol{y}) = z \iff \exists\, s \in \mathsf{Seq}\, \Big(\mathsf{len}(s) = x^+ \wedge (s)_0 = g(\boldsymbol{y}) \wedge \\ \forall\, 0 \leq i < x\, ((s)_{i^+} = h(i,(s)_i,\boldsymbol{y})) \wedge (s)_x = z\Big)$$

Here x^+ is shorthand for $x+1$. The sequence number s records all previous values of f. Now consider the following function $\widehat{f}$ associated with f:

$$\widehat{f}(x,\boldsymbol{y}) := \langle f(0,\boldsymbol{y}), f(1,\boldsymbol{y}), \ldots, f(x,\boldsymbol{y})\rangle$$

Lemma

f is primitive recursive iff $\widehat{f}$ is primitive recursive.

Course of Value Recursion

Thus, it is natural to generalize the primitive recursion scheme slightly by defining functions so that the value at x depends directly on all the previous values.

$$
\begin{aligned}
f(0, \boldsymbol{y}) &= g(\boldsymbol{y}) \\
f(x^+, \boldsymbol{y}) &= H(x, \widehat{f}(x, \boldsymbol{y}), \boldsymbol{y})
\end{aligned}
$$

Lemma

If g and H are primitive recursive then f is also primitive recursive.

Exercise

Prove the last two lemmata. You may safely assume that standard sequence operations such as append are primitive recursive.

Sequence Operations

As always, having a data structure by itself is not particularly interesting, we need to be able to implement operations. In our case, one can show that the following operations on sequences are primitive recursive.

- head, tail
- concatenate
- reverse
- sort
- map
- sum, product

In fact, it would be quite difficult to come up with any example of an operation used in a real program that fails to be primitive recursive.

Exercises

Exercise

Prove that all these functions are indeed primitive recursive.

Exercise

Explain how to implement search in binary search trees as a primitive recursive operation.

Exercise

Come up with yet another coding function based on repeated application of a pairing function (make sure your method really works).

1 Coding the World

2 Gödel's β Function

3 Coding Systems

Gödel's Approach

There is more elegant and slightly more elementary way to code sequence numbers due to Gödel that he used in his famous incompleteness theorem.

For the sake of completeness, here is a brief description of Gödel's method.

Gödel's Trick

Perhaps the most obvious way to encode a sequence $\boldsymbol{a} = a_0, \ldots, a_{n-1}$ is to use the uniqueness of the prime decomposition:

$$\langle \boldsymbol{a} \rangle = p_0^{a_0+1} p_1^{a_1+1} \ldots p_{n-1}^{a_{n-1}+1}$$

where p_i is the ith prime.

Alas, this requires the enumeration of primes and exponentiation, which may not be available in weak systems of arithmetic.

Here is a more basic approach that avoids these tools, and uses the Chinese Remainder Theorem instead. Define

$$\beta(x, y, i) = x \pmod{1 + (i{+}1)y}$$

The hope is that, for the right choice of x and y, $\beta(x, y, i) = a_i$.

CRT

To this end define

$$\begin{aligned} c &= \max(n, a_0, \ldots, a_{n-1}) \\ C &= c! \end{aligned}$$

Claim: The numbers $1 + (i{+}1)C$, $i < n$, are all coprime.

Proof. If some prime p divides $1 + (i{+}1)C$, it cannot divide C, so $p > c$. But if p divides two of these numbers, then p divides their difference dC, $d < n \leq c$. Contradiction. □

But then, by CRT, there is some a such that $a_i = a \bmod 1 + (i{+}1)C$, and we can choose $a < \prod 1 + (i{+}1)C$.

Sequence Numbers

We can then define a sequence number for $a_0, \ldots, a_{n-1}$ to be $s = \pi(a, C)$ and decode via

$$\mathsf{dec}(s, i) = \beta(\pi_1(s), \pi_2(s), i)$$

We can set up a primitive recursive coding scheme as above by conducting bounded searches for the various parameters. This involves factorials and is just as demanding as exponentiation.

However, in some applications we only need the existence a sequence number, and that can be handled more easily using Gödel's approach.

Defining Exponentiation

Here is the key application: in a formal system of arithmetic providing just addition and multiplication, we can also represent exponentiation.

$$\begin{aligned} E(a,b) &= \min\big(\, s \mid \mathsf{dec}(s,0) = 1 \wedge \forall\, i < b\, (\mathsf{dec}(s,i{+}1) = a \cdot \mathsf{dec}(s,i))\,\big) \\ a^b &= \mathsf{dec}(E(a,b),b) \end{aligned}$$

The $\min$ operator is easily definable: $\phi(x) \wedge \forall\, z < x\, \neg\phi(z)$.

Here existence matters, but bounds do not.

Gödel's Trick

To deal with sequences of arbitrary length one can use a clever divisibility argument.

Lemma (Gödel)

There exists a primitive recursive function dec : $\mathbb{N}^2 \to \mathbb{N}$ *such that*

$$\forall\, a_0, \dots, a_{n-1}\, \exists\, a\, \forall\, i < n\, (a_i = \mathsf{dec}(a, i)).$$

So a is a potential code number for $a_0, \dots, a_{n-1}$

Proof. Set

$$\mathsf{dec}(a, i) = \min\Big(\, x < a \mid \big((\pi(x, i) + 1)\pi_2(a) + 1\big) \text{ divides } \pi_1(a)\,\Big)$$

The idea is that the factors of $\pi_1(a)$ contain information about the a_i.

We need to establish the existence of the witness a.

Proof, contd.

Let $a_0, \ldots, a_{n-1}$ arbitrary and set

$$\begin{aligned} c &= \max\Big(\, \pi(a_i, i) \mid i < n \,\Big) \\ C &= (c-1)! \\ p &= \prod_{i<n} ((\pi(a_i, i) + 1)C + 1) \\ a &= \pi(p, C) \end{aligned}$$

Note that $\quad \forall\, i < j < c\, (iC + 1, jC + 1 \text{ coprime }).$

But then

$$\mathsf{dec}(a, i) = \min\Big(\, x < a \mid (\pi(x, i) + 1)C + 1 \text{ divides } p \,\Big)$$

□

Sequence Numbers

Definition

Define a coding function $\langle . \rangle$ by

$$\langle \boldsymbol{x} \rangle = \min\Big(a \mid \mathsf{dec}(a,0) = n \land \forall\, i \in [n]\, (\mathsf{dec}(a,i) = a_i) \Big)$$

Also set $\mathsf{lh}(a) = \mathsf{dec}(a,0)$ and $(a)_i := \mathsf{dec}(a,i)$.

Again, $\langle . \rangle$ is not primitive recursive, but we have:

- $\mathsf{Seq} = \{\, \langle \boldsymbol{x} \rangle \mid \boldsymbol{x} \in \mathbb{N}^* \,\} \subseteq \mathbb{N}$ is primitive recursive.
- The restriction to $\mathbb{N}^n$ is primitive recursive.
- dec is primitive recursive.

Exercise

Prove this claim in detail.

1 Coding the World

2 Gödel's β Function

3 Coding Systems

Coding in General

The examples of the coding function from the previous sections are by no means exhaustive, even if one is interested only in functions that are fairly easy to compute.

In general, one way to build a coding system is to

- First define a pairing function, and
- extend this pairing function to a full coding function.

Pairing Systems

Definition

A pairing system consists of

- an injective function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, the pairing function
- functions $\pi_i : \mathbb{N} \to \mathbb{N}$, $i = 1, 2$, the unpairing functions

where $\pi_i(\pi(x_1, x_2)) = x_i$ for all x_i.

The system is primitive recursive if the paring and unpairing functions as well as the range of the pairing function are all primitive recursive.

The unpairing functions are only of interest only on the range of π; we will generally assume that they are 0 everywhere else.

Note that this is not a problem for primitive recursive systems.

Orderings

Note that any pairing function induces a total order on $\mathbb{N} \times \mathbb{N}$:

$$(a, b) \prec (a', b') \iff \pi(a, b) < \pi(a', b')$$

This provides a convenient method to construct pairing functions: find some reasonable linear ordering on $\mathbb{N} \times \mathbb{N}$ and then engineer a corresponding function.

We would like the function to be primitive recursive, so the ordering should be fairly straightforward.

Cantor

For example, we could define the order to be

$$\begin{aligned}(a,b) \prec (a',b') \iff{} & (a{+}b < a'{+}b') \lor \\ & \big((a{+}b = a'{+}b' \land (a,b) <_p (a',b')\big)\end{aligned}$$

Here $<_p$ refers to the usual product order. So the first few pairs are

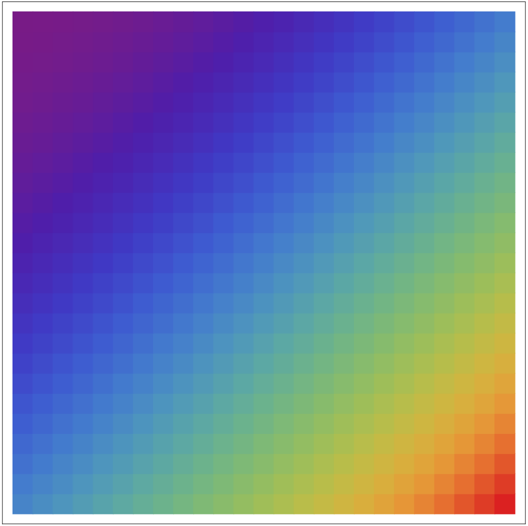
$$(0,0),(0,1),(1,0),(0,2),(1,1),(2,0),(0,3),(1,2),(2,1),(3,0),\ldots$$

The corresponding pairing function is a simple quadratic polynomial:

$$\pi(x,y) = ((x+y)^2 + 3x + y)/2$$

Note that this function is actually a bijection.

Picture



Exercises

Exercise

Explain the Cantor polynomial and show that function is indeed a bijection.

Exercise

Find simple descriptions for the corresponding unpairing functions.

Exercise

Show that the unpairing functions are primitive recursive.

Aside: Fueter-Pólya

A surprising theorem by Fueter and Pólya from 1923 states that, up to a swap of variables, this is the only quadratic polynomial that defines a bijection $\mathbb{N}^2 \leftrightarrow \mathbb{N}$.

The proof is rather difficult and uses the fact that e^a is transcendental for algebraic $a \neq 0$.

It is an open problem whether there are other bijections for higher degree polynomials.

Extra Credit.

Another Pairing Function

We could replace sum by max in the pair ordering:

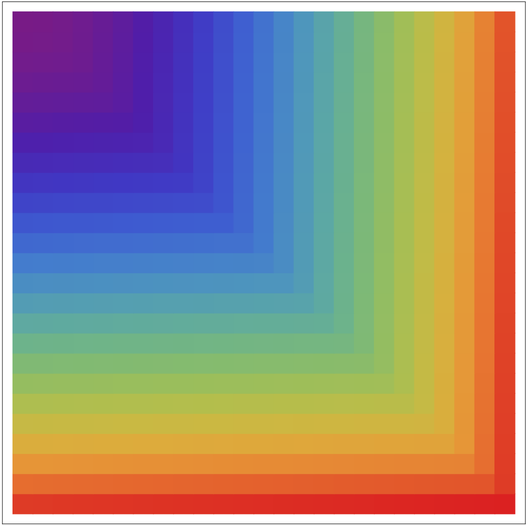
$$\begin{aligned}(a,b) \prec (a',b') \iff & (\max(a,b) < \max(a',b')) \lor \\ & \big(\max(a,b) = \max(a',b') \land (a,b) <_p (a',b')\big)\end{aligned}$$

Again, $<_p$ refers to the usual product order. he first few pairs this time are

$$(0,0),(0,1),(1,0),(1,1),(0,2),(1,2),(2,0),(2,1),(2,2),(0,3),\ldots$$

Somewhat similar to the last pairing function, but the picture looks quite different.

Picture



Exercise

Find the pairing function for the last order.

Exercise

Then determine the corresponding unpairing functions.

Exercise

Show that the unpairing functions are primitive recursive.

Even/Odd

The last two pairing functions are inspired by pair orderings. Then next one is based on basic arithmetic instead: decomposing a number into an even and on odd part.

$$\pi(x, y) = 2^x(2y + 1)$$

Hence the range of π is $\mathbb{N}_+$ (but not $\mathbb{N}$). The pairs this time look like so:

$$(0, 0), (1, 0), (0, 1), (2, 0), (0, 2), (1, 1), (0, 3), (3, 0), (0, 4), (1, 2), \dots$$

Much less intuitive than the previous cases.

To see why this pairing function is still quite natural consider the numbers in binary. For example

$$\pi(5,27) = 32 \cdot 55 = 1760 = 110111\,00000_2$$

In general, the binary expansion of $\pi(x,y)$ looks like so:

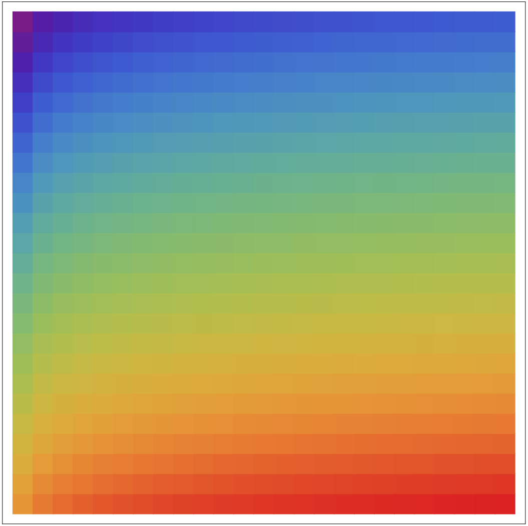
$$y_k y_{k-1} \ldots y_0 \, 1 \, \underbrace{00 \ldots 0}_{x}$$

where $y_k y_{k-1} \ldots y_0$ is the standard binary expansion of y (y_k is the most significant digit).

Exercise

Find the corresponding unpairing functions.

Picture (log plot)



Coding Systems

Definition

A coding system consists of three functions

$$\begin{aligned} \langle . \rangle &: \mathbb{N}^\star \to \mathbb{N} \\ \mathsf{len} &: \mathbb{N} \to \mathbb{N} \\ \mathsf{dec} &: \mathbb{N} \times \mathbb{N} \to \mathbb{N} \end{aligned}$$

For $b = \langle a_1, a_2, \ldots, a_n \rangle$ we have $n = \mathsf{len}(b)$ and $\mathsf{dec}(b, i) = a_i$ for all $i \in [n]$.

$\langle . \rangle$ is polyadic and called the coding function, len is the length function, and dec is the decoding function. The range Seq of the coding function is the set of sequence numbers.

Seq may be equal to $\mathbb{N}$, but in general it will not be.

Decoding

To be more explicit about the decoding process, suppose

$$b = \langle a_1, a_2, \ldots, a_n \rangle$$

is some sequence number. Then we can recover the length of the sequence via

$$\mathsf{len}(b) = n$$

and the actual entries via that extracts the components:

$$\mathsf{dec}(b, i) = a_i$$

for all $i = 0, \ldots, n-1$.

For simplicity we assume that len and dec are 0 outside of their relevant domain of definition.

Computable Coding

Definition

A coding system is primitive recursive if the length and decoding functions, and the sequence numbers are all primitive recursive.

Note that the coding function itself is polyadic, and thus cannot be primitive recursive. In all interesting cases, the restrictions

$$\langle . \rangle : \mathbb{N}^n \to \mathbb{N}$$

will be primitive recursive, though.

So the challenge is to come up with well-behaved primitive recursive coding systems.

Sequence Numbers

Exercise

Show how to check if a number is a sequence number given dec *and* len*.*

Extending Pairs

Suppose we have a pairing system. The first step is to extend the pairing function π to a map $\widehat{\pi}$ that is defined on all sequences of length at least 2:

$$\widehat{\pi} : \mathbb{N}^{\geq 2} \longrightarrow \mathbb{N}$$

This comes down to declaring π to be, say, right associative:

$$\begin{aligned} \widehat{\pi}(x_1, x_2) &= \pi(x_1, x_2) \\ \widehat{\pi}(x_1, x_2, \ldots, x_k) &= \pi(x_1, \widehat{\pi}(x_2, \ldots, x_k)) \end{aligned}$$

Note that this map is not injective: let $c = \pi(a, b)$, then $\widehat{\pi}(a, c) = \widehat{\pi}(a, a, b)$.

Length Encoding

To avoid this issue, define

$$\begin{aligned} \langle \mathsf{nil} \rangle &:= \pi(0,0) \\ \langle a \rangle &:= \pi(1,a) \\ \langle a_1,\ldots,a_n \rangle &:= \pi(n, \widehat{\pi}(a_2,\ldots,a_n)) \end{aligned}$$

Here are some sequence numbers for this particular coding function:

$$\begin{aligned} \langle 10 \rangle &= 1024 \\ \langle 0,0,0 \rangle &= 7 \\ \langle 1,2,3,4,5 \rangle &= 532754 \end{aligned}$$

It's Decodable

Lemma

$\langle . \rangle : \mathbb{N}^* \to \mathbb{N}$ *is injective.*

Proof. Suppose

$$\langle a_1, \ldots, a_1 \rangle = c = \langle b_1, \ldots, b_m \rangle$$

Since $\pi_1(c)$ is the length of the sequence we can conclude that $n = m$.

But then $\widehat{\pi}(a_1, \ldots, a_n) = \widehat{\pi}(b_1, \ldots, b_n)$ and it follows that $a_i = b_i$. □

Sans Length

Recall the even/odd pairing function

$$\pi(x, y) = 2^x \, (2y + 1)$$

The range here is $\mathbb{N}_+$, so we don't have a bijection. As it turns out, we can exploit this produce a rather elegant coding function:

$$\begin{aligned} \langle \mathsf{nil} \rangle &\coloneqq 0 \\ \langle a_1, \ldots, a_n \rangle &\coloneqq \pi(a_1, \langle a_2, \ldots, a_n \rangle) \end{aligned}$$

It’s Decodable

Lemma

$\langle . \rangle : \mathbb{N}^* \to \mathbb{N}$ *is bijective.*

Proof. Suppose

$$\langle a_1, \ldots, a_n \rangle = c = \langle b_1, \ldots, b_m \rangle$$

We may safely assume that $n \leq m$. If $n = 0$, then $c = 0$ and it follows that $m = 0$ because the range of π does not contain 0.

So suppose $0 < c$, $0 < n \leq m$. Since the range of π is all of $\mathbb{N}_+$ we have $a_1 = \pi_1(c) = b_1$, furthermore $\langle a_2, \ldots, a_n \rangle = \pi_2(c) = \langle b_2, \ldots, b_n \rangle$.

Done by induction. □

Less formally . . .

Here is a sequence number and its binary expansion:

$$
\begin{aligned}
\langle 2,3,5,1\rangle &= 20548 \\
&= 1\underbrace{0}_{1}1\underbrace{00000}_{5}1\underbrace{000}_{3}1\underbrace{00}_{2}
\end{aligned}
$$

So the number of 1's (the digitsum) is just the length of the sequence, and the spacing between the 1's indicates the actual numerical values.

It follows that the coding function is injective and surjective, right?

Everything is PR

Exercise

Show that the pairing function π and both unpairing functions $x = \pi_1(\pi(x, y))$ and $y = \pi_2(\pi(x, y))$ are primitive recursive.

Exercise

Show that the length and decoding functions len *and* dec *are primitive recursive.*

Exercise

Show that the coding function $\langle . \rangle$ is primitive recursive when restricted to inputs of fixed length.