1.4 Pairing Function and Arithmetization

Cantor Pairing Function

1.4.1 Pairing function. The *modified Cantor pairing function* is a p.r. function by the following explicit definition:

$$\langle x, y\rangle = \sum_{i=0}^{x+y} i + x + 1,$$

Figure 1.1 shows the initial segment of values of this modified pairing function of Cantor in tabular form.

$\langle x,y\rangle$	0	1	2	3	4	5	6	⋯
0	1	2	4	7	11	16	22	⋯
1	3	5	8	12	17	23	30	⋯
2	6	9	13	18	24	31	39	⋯
3	10	14	19	25	32	40	49	⋯
4	15	20	26	33	41	50	60	⋯
5	21	27	34	42	51	61	72	⋯
6	28	35	43	52	62	73	85	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Fig. 1.1 The modified Cantor pairing function

1.4.2 Basic properties of the pairing function. We have

$$\langle x_1, x_2\rangle = \langle y_1, y_2\rangle \to x_1 = y_1 \wedge x_2 = y_2 \tag{1}$$

$$x < \langle x, y\rangle \wedge y < \langle x, y\rangle \tag{2}$$

$$x = 0 \vee \exists y \exists z\, x = \langle y, z\rangle. \tag{3}$$

Property (1) is called the *pairing property* and it says that the function is an injection. Property (2) is needed for induction and/or recursion. From (2) we get that $0 \neq \langle x, y\rangle$ for every x and y. This means that 0 is not in the range of the pairing function and plays the role of the atom *nil* of Lisp. From this and (3) we can see that the pairing function is onto the set $\mathrm{N} \setminus \{0\}$, i.e. that 0 is the only atom.

1.4.3 Ordering properties of the pairing function. We have

$$\langle x_1, x_2\rangle \leq \langle y_1, y_2\rangle \leftrightarrow x_1 + x_2 < y_1 + y_2 \vee x_1 + x_2 = y_1 + y_2 \wedge x_1 \leq y_1$$
$$\langle x_1, x_2\rangle < \langle y_1, y_2\rangle \leftrightarrow x_1 + x_2 < y_1 + y_2 \vee x_1 + x_2 = y_1 + y_2 \wedge x_1 < y_1.$$

1.4.4 Pair representation of natural numbers. The class of *pair numerals* consists of terms obtained from 0 by finitely many pairing operations. It can be easily proved by complete induction that every natural number x can be uniquely presented as a pair numeral. We call this the *pair representation* of natural numbers. Pair numerals can be visualized as finite binary trees.

Zeroes are leaves and a pair numeral $\langle \tau_1, \tau_2 \rangle$ is a tree with two sons τ_1 and τ_2. Figure 1.2 enumerates the finite binary trees corresponding to the pair numerals. We will denote by $|x|_{\mathrm{p}}$ the number of nodes of the tree corresponding to the pair numeral $\tau = x$. In other words, $|x|_{\mathrm{p}}$ is the number of pairing operations needed to construct the pair numeral $\tau = x$.

In order to obtain a simple recursive characterization of subelementary complexity classes (such as PTIME) one should use a pairing function such that $|x|_{\mathrm{p}} = \Omega(\lg(x))$. The system CL uses such a function but for the purposes of this text this requirement is not important and we use a much simpler pairing function which does not satisfy the requirement.

1.4.5 Projection functions. From the basic properties of the pairing function we can see that every non-zero number x can be uniquely written in the form $x = \langle y, z \rangle$ for some y, z. The numbers y and z are called the *first* and the *second projection* of x, respectively.

The *first projection* function π_1 and *second projection* π_2 are unary functions satisfying

$$\pi_1(0) = 0 \qquad \pi_2(0) = 0$$
$$\pi_1\langle x, y \rangle = x \qquad \pi_2\langle x, y \rangle = y.$$

The projection functions are p.r. functions by bounded minimalization:

$$\pi_1(x) = \mu y < x[\exists z < x\, x = \langle y, z \rangle] \qquad \pi_2(x) = \mu z < x[\exists y < x\, x = \langle y, z \rangle].$$

Arithmetization of Tuples

1.4.6 Tuples. In this section we introduce a particular encoding of ordered n-tuples of natural numbers based on our pairing function $\langle x, y \rangle$. Our aim is to assign to each element $(x_1, \ldots, x_n)$ of the Cartesian product N^n a number $\ulcorner (x_1, \ldots, x_n) \urcorner^n$, called the *code* of $(x_1, \ldots, x_n)$, so that different codes are assigned to different n-tuples. Moreover we would like to have decoding effective. This means we can effectively decide whether a number is the code of an n-tuple and if it is, find that tuple. We will use this encoding throughout the rest of this text.

1.4.7 Arithmetization. Encoding of Cartesian products N^n, where $n \geq 0$, is defined inductively on n as follows:

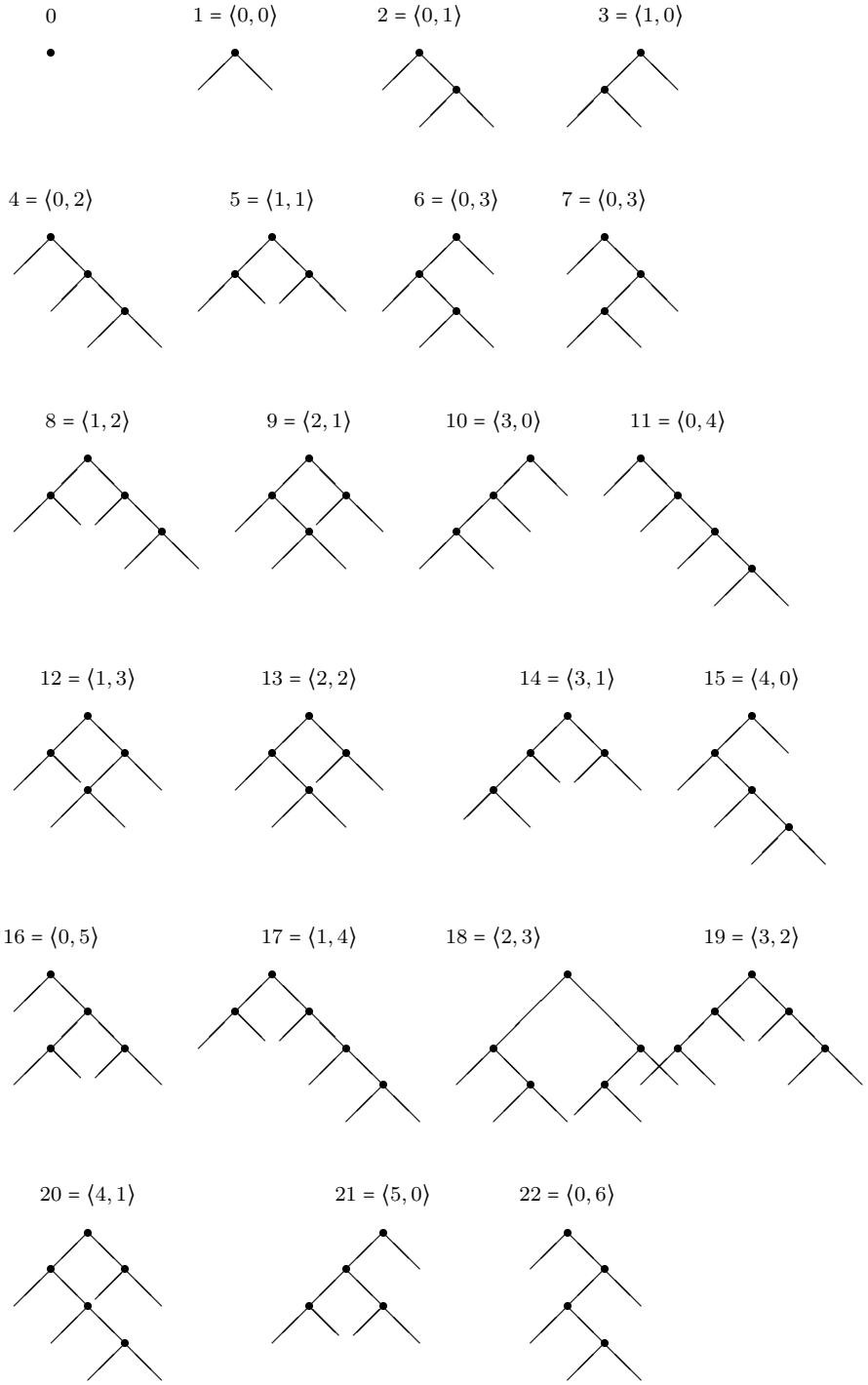

Fig. 1.2 Pair representation of natural numbers

$$\begin{aligned} \ulcorner \varnothing \urcorner^0 &= 0 \\ \ulcorner x \urcorner^1 &= x \\ \ulcorner (x_1, x_2, \ldots, x_n) \urcorner^n &= \big\langle x_1, \ulcorner (x_2, \ldots, x_n) \urcorner^{n-1} \big\rangle \qquad \text{if } n \geq 2. \end{aligned}$$

The reader will note that the code of an 1-tuple x is the number itself and the code of the empty tuple $\varnothing$ is the number 0. Note also that $\ulcorner (x, y) \urcorner^2 = \langle x, y \rangle$.

The reader will also note that these encodings may overlap. Consider, for instance, the number 2. We have $2 = \langle 0, 1 \rangle$ and $1 = \langle 0, 0 \rangle$. Therefore

$$\begin{aligned} 2 &= \langle 0, 1 \rangle = \ulcorner (0, 1) \urcorner^2 \\ 2 &= \langle 0, 1 \rangle = \big\langle 0, \langle 0, 0 \rangle \big\rangle = \big\langle 0, \ulcorner (0, 0) \urcorner^2 \big\rangle = \ulcorner (0, 0, 0) \urcorner^3. \end{aligned}$$

Hence, the number 2 is the code both of the ordered pair $(0, 1) \in \mathrm{N}^2$ and the ordered triple $(0, 0, 0) \in \mathrm{N}^3$.

1.4.8 Notational conventions. We will adopt the following conventions for the pairing function $\langle x, y \rangle$. We postulate that the pairing operator groups to the right, i.e. $\langle x, y, z \rangle$ abbreviates $\big\langle x, \langle y, z \rangle \big\rangle$. If $\vec{\tau} \equiv (\tau_1, \ldots, \tau_n)$ is an n-tuple of terms then the term $\langle \vec{\tau} \rangle$ stands for $\langle \tau_1, \ldots, \tau_n \rangle$ when $n \geq 2$, for τ_1 when $n = 1$, and for 0 when $n = 0$. Note that we then have

$$\ulcorner (x_1, \ldots, x_n) \urcorner^n = \langle x_1, \ldots, x_n \rangle$$

for every n and every element $(x_1, \ldots, x_n)$ of N^n.

1.4.9 Predicate holding of the codes of tuples. For $n \geq 2$, we have

$$\exists x_1 \ldots \exists x_n \, x = \langle x_1, \ldots, x_n \rangle \leftrightarrow \pi_2^{n-2}(x) \neq 0.$$

Consequently, the binary predicate $\mathit{Tuple}(n, x)$, which holds when x is the code of an n-tuple, is primitive recursive by the following explicit definition

$$\mathit{Tuple}(n, x) \leftrightarrow n = 0 \wedge x = 0 \vee n = 1 \vee n \geq 2 \wedge \pi_2^{n-2}(x) \neq 0.$$

1.4.10 Projection function for tuples. The ternary *projection* function $[x]_i^n$ selects the i-th element of the n-tuple coded by x, i.e.

$$[\langle x_1, \ldots, x_n \rangle]_i^n = x_i$$

for every $i = 1, \ldots, n$. We clearly have

$$x = \langle x_1, \ldots, x_n \rangle \to \bigwedge_{i=1}^{n-1} x_i = \pi_1 \, \pi_2^{i-1}(x) \wedge x_n = \pi_2^{n-1}(x)$$

for $n \geq 2$. Thus we can define $[x]_i^n$ explicitly as a p.r. function by

$$[x]_i^n = D\big(i \dot{*} n, \pi_1 \pi_2^{i \dot{-} 1}(x), \pi_2^{n \dot{-} 1}(x)\big).$$

The projection function satisfies

$$\begin{aligned} [x_1]_1^1 &= x_1 \\ [\langle x_1, x\rangle]_1^{n+2} &= x_1 \\ [\langle x_1, x\rangle]_{i+2}^{n+2} &= [x]_{i+1}^{n+1}\,. \end{aligned}$$

1.4.11 Contraction to unary functions. As a simple application of the arithmetization of n-tuples we obtain the following natural correspondence between n-ary and unary functions. If f is an n-ary function then its *contraction* is the unary function $\langle f\rangle$ such that

$$\langle f\rangle(x) = \begin{cases} f(x_1, \ldots, x_n) & \text{if } x = \langle x_1, \ldots, x_n\rangle \text{ for some numbers } x_1, \ldots, x_n, \\ 0 & \text{if there are no such numbers.} \end{cases}$$

Note that the contraction of an unary function is the function itself.

We can define the contraction of f explicitly by

$$\langle f\rangle(x) = D\Big(\mathit{Tuple}_*(n, x), f\big([x]_1^n, \ldots, [x]_n^n\big), 0\Big).$$

Vice versa, we can recover f from its contraction by

$$f(x_1, \ldots, x_n) = \langle f\rangle(\langle x_1, \ldots, x_n\rangle).$$

Thus a function is primitive recursive if and only if its contraction is.

Arithmetization of Finite Sequences

1.4.12 Finite sequences. Now we consider the problem of the arithmetization of finite sequences of natural numbers. Mathematically speaking, finite sequences are just tuples of variable length and so the set of all such sequences is the infinite union $\bigcup_{n\in \mathrm{N}} \mathrm{N}^n$. We cannot use the method of codings of tuples of fixed length since such encodings overlap. Our uniform encoding of finite sequences is based on the fact that the number 0 is the atom, i.e. it is not in the range of the pairing function $\langle x, y\rangle$.

1.4.13 Arithmetization. A uniform method for coding of finite sequences of numbers into N is obtained as follows. We assign the code 0 to the empty sequence $\varnothing$. A non-empty sequence $x_1, \ldots, x_n$ is coded by the number $\langle x_1, x_2, \ldots, x_n, 0\rangle$ as shown in Fig. 1.3. The number $\langle x_1, x_2, \ldots, x_n, 0\rangle$ is often called the sequence number of the sequence $x_1, \ldots, x_n$.

The reader will note that the assignment of codes is one to one: i.e. every finite sequence of natural numbers is coded by exactly one natural number, and vice versa, every natural number is the code of exactly one finite sequence of natural numbers.

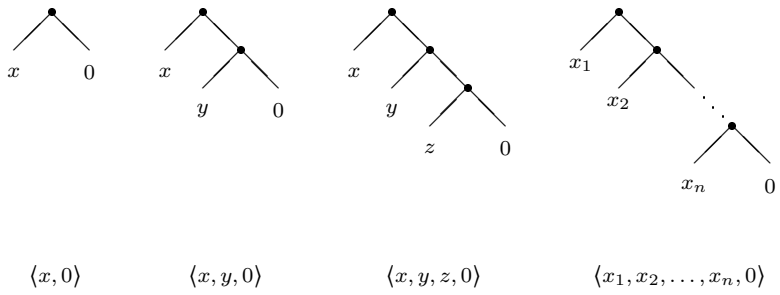

Fig. 1.3 Arithmetization of finite sequences

1.4.14 Length of sequences. The code $x = \langle x_1, x_2, \ldots, x_n, 0\rangle$ of the sequence $x_1, \ldots, x_n$ has the *length* n. The function $L(x)$ yielding the length of x is introduced by the bounded minimalization as a p.r. function:

$$L(x) = \mu n \leq x[\pi_2^n(x) = 0].$$

The function satisfies

$$\begin{aligned} L(0) &= 0 \\ L\langle v, w\rangle &= L(w) + 1. \end{aligned}$$

1.4.15 Indexing function. The *indexing* function $(x)_i$ yields the $(i+1)$-st element of the sequence x, i.e.

$$(\langle x_0, \ldots, x_i, \ldots, x_{n-1}, 0\rangle)_i = x_i.$$

The function is defined explicitly by

$$(x)_i = \pi_1\, \pi_2^i(x)$$

as a primitive recursive function.

The recurrent properties of the indexing function are:

$$\begin{aligned} (\langle v, w\rangle)_0 &= v \\ (\langle v, w\rangle)_{i+1} &= (w)_i\,. \end{aligned}$$