- Otherwise, the output is equal to the input, provided that it contains fewer than 2^{64} bytes, prefixed by the minimal-length byte array which when interpreted as a big-endian integer is equal to the length of the input byte array, which is itself prefixed by the number of bytes required to faithfully encode this length value plus 183.

Byte arrays containing 2^{64} or more bytes cannot be encoded. This restriction ensures that the first byte of the encoding of a byte array is always below 192, and thus it can be readily distinguished from the encodings of sequences in $\mathbb{L}$.

Formally, we define R_{b}:

$$R_{\mathrm{b}}(\mathbf{x}) \equiv \begin{cases} \mathbf{x} & \text{if} \quad \|\mathbf{x}\| = 1 \wedge \mathbf{x}[0] < 128 \\ (128 + \|\mathbf{x}\|) \cdot \mathbf{x} & \text{else if} \quad \|\mathbf{x}\| < 56 \\ \left(183 + \left\|\mathtt{BE}(\|\mathbf{x}\|)\right\|\right) \cdot \mathtt{BE}(\|\mathbf{x}\|) \cdot \mathbf{x} & \text{else if} \quad \|\mathbf{x}\| < 2^{64} \\ \varnothing & \text{otherwise} \end{cases} \tag{192}$$

$$\mathtt{BE}(x) \equiv (b_0, b_1, ...) : b_0 \neq 0 \wedge x = \sum_{n=0}^{\|\mathbf{b}\|-1} b_n \cdot 256^{\|\mathbf{b}\|-1-n} \tag{193}$$

$$(x_1, ..., x_n) \cdot (y_1, ..., y_m) = (x_1, ..., x_n, y_1, ..., y_m) \tag{194}$$

Thus BE is the function that expands a non-negative integer value to a big-endian byte array of minimal length and the dot operator performs sequence concatenation.

If instead, the value to be serialised is a sequence of other items then the RLP serialisation takes one of two forms:

- If the concatenated serialisations of each contained item is less than 56 bytes in length, then the output is equal to that concatenation prefixed by the byte equal to the length of this byte array plus 192.
- Otherwise, the output is equal to the concatenated serialisations, provided that they contain fewer than 2^{64} bytes, prefixed by the minimal-length byte array which when interpreted as a big-endian integer is equal to the length of the concatenated serialisations byte array, which is itself prefixed by the number of bytes required to faithfully encode this length value plus 247.

Sequences whose concatenated serialized items contain 2^{64} or more bytes cannot be encoded. This restriction ensures that the first byte of the encoding does not exceed 255 (otherwise it would not be a byte).

Thus we finish by formally defining R_{l}:

$$R_{\mathrm{l}}(\mathbf{x}) \equiv \begin{cases} (192 + \|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{if} \quad s(\mathbf{x}) \neq \varnothing \wedge \|s(\mathbf{x})\| < 56 \\ \left(247 + \left\|\mathtt{BE}(\|s(\mathbf{x})\|)\right\|\right) \cdot \mathtt{BE}(\|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{else if} \quad s(\mathbf{x}) \neq \varnothing \wedge \|s(\mathbf{x})\| < 2^{64} \\ \varnothing & \text{otherwise} \end{cases} \tag{195}$$

$$s(\mathbf{x}) \equiv \begin{cases} \mathtt{RLP}(\mathbf{x}[0]) \cdot \mathtt{RLP}(\mathbf{x}[1]) \cdot ... & \text{if} \quad \forall i : \mathtt{RLP}(\mathbf{x}[i]) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \tag{196}$$

If RLP is used to encode a scalar, defined only as a non-negative integer (in $\mathbb{N}$, or in $\mathbb{N}_x$ for any x), it must be encoded as the shortest byte array whose big-endian interpretation is the scalar. Thus the RLP of some non-negative integer i is defined as:

$$\mathtt{RLP}(i : i \in \mathbb{N}) \equiv \mathtt{RLP}(\mathtt{BE}(i)) \tag{197}$$

When interpreting RLP data, if an expected fragment is decoded as a scalar and leading zeroes are found in the byte sequence, clients are required to consider it non-canonical and treat it in the same manner as otherwise invalid RLP data, dismissing it completely.

There is no specific canonical encoding format for signed or floating-point values.

Appendix C. Hex-Prefix Encoding

Hex-prefix encoding is an efficient method of encoding an arbitrary number of nibbles as a byte array. It is able to store an additional flag which, when used in the context of the trie (the only context in which it is used), disambiguates between node types.

It is defined as the function HP which maps from a sequence of nibbles (represented by the set $\mathbb{Y}$) together with a boolean value to a sequence of bytes (represented by the set $\mathbb{B}$):

$$\mathtt{HP}(\mathbf{x}, t) : \mathbf{x} \in \mathbb{Y} \equiv \begin{cases} (16f(t), 16\mathbf{x}[0] + \mathbf{x}[1], 16\mathbf{x}[2] + \mathbf{x}[3], ...) & \text{if} \quad \|\mathbf{x}\| \text{ is even} \\ (16(f(t) + 1) + \mathbf{x}[0], 16\mathbf{x}[1] + \mathbf{x}[2], 16\mathbf{x}[3] + \mathbf{x}[4], ...) & \text{otherwise} \end{cases} \tag{198}$$

$$f(t) \equiv \begin{cases} 2 & \text{if} \quad t \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{199}$$

Thus the high nibble of the first byte contains two flags; the lowest bit encoding the oddness of the length and the second-lowest encoding the flag t. The low nibble of the first byte is zero in the case of an even number of nibbles and the first nibble in the case of an odd number. All remaining nibbles (now an even number) fit properly into the remaining bytes.