Chapter 2. Unconventional Fixed-Radix Number Systems

§2.1. Negative-Radix Number Systems

If we allow the radix to be a negative number, $r = -\beta, \beta \geqslant 2$, then we have a negative-radix number system, where a given value X can be represented as $X = (x_{n-1}, \ldots, x_0)_{-\beta}$, and its value can be given by

$$X = \sum_{i=0}^{n-1} x_i(-\beta)^i$$

where the weight is given by

$$(-\beta)^i = \begin{cases} \beta^i & \text{if } i \text{ is even} \\ -\beta^i & \text{if } i \text{ is odd.} \end{cases}$$

§2.1.1. Negative decimal number system

Example 1 *Let $\beta = 10$, then we have nega-decimal system.*

$$(182)_{-10} = 1 \times (-10)^2 + 8 \times (-10)^1 + 2 \times (-10)^0 = 22.$$

$$(123)_{-10} = 1 \times (-10)^2 + 2 \times (-10)^1 + 3 \times (-10)^0 = 83$$

For an integer in nega-decimal system of length three, $X = (x_2, x_1, x_0)_{-10}$, the maximal representable value is

$$\max\{(x_2, x_1, x_0)_{-10}\} = (909)_{-10} = 909,$$

and the minimal representable value is

$$\min\{(x_2, x_1, x_0)_{-10}\} = (090)_{-10} = -90.$$

So, the representation range for this number system is $[-90, 909]$.

- Representation of negative numbers with negative-radix systems are efficient. (No sign is needed to represent a signed number.)
- The sign of the number is decided by the first nonzero digit.
- The arithmetic operations with negative-radix representation are slightly more complex.

Arithmetic operations

Example 2 *Let $A = (182)_{-10}$ and $B = (123)_{-10}$ be two numbers in nega-decimal system. Then we have the following*

$$A + B = (182)_{-10} + (123)_{-10} = (105)_{-10}.$$
$$A - B = (182)_{-10} - (123)_{-10} = (079)_{-10} = (79)_{-10}.$$

What about changing A to $(192)_{-10}$, and then performing $A - B$?

Conversion between decimal and nega-decimal systems

Example 3 *Let $A = (1824)_{-10}$, $B = (1824)_{10}$ and $C = (43.6875)_{10}$ be three numbers in either nega-decimal or decimal system.*

(1). Convert A into decimal number system:

$$A = (1824)_{-10} = -1020_{10} + 804_{10} = -216_{10}.$$

(2). Convert B into nega-decimal number system (Method 1).

$$\begin{aligned} B &= (1824)_{10} \\ &= 10000_{10} - 8176_{10} \\ &= 10000_{10} - 9000_{10} + 824_{10} \\ &= 10000_{10} - 9000_{10} + 900_{10} - 76_{10} \\ &= 10000_{10} - 9000_{10} + 900_{10} - 80_{10} + 4_{10} \\ &= (19984)_{-10}. \end{aligned}$$

(3). Convert C into nega-decimal number system (Method 2)

Integer: Decimal-to-Nega-decimal		
Dividing-by-(-10)	*Quotient*	*Remainder (keep it non-negative)*
$44/(-10)$	-4	$4 = x_0$
$-4/(-10) = (-10+4)/(-10)$	1	$6 = x_1$
$1/(-10)$	0	$1 = x_2$

Fraction: Decimal-to-Nega-decimal		
Multiplying-by-(-10)	*Fractional part (keep it within the range)*	*Integral part*
$-0.3125 \times (-10)$	-0.875	$4 = x_{-1}$
$-0.875 \times (-10)$	-0.25	$9 = x_{-2}$
$-0.25 \times (-10)$	-0.5	$3 = x_{-3}$
$-0.5 \times (-10)$	0	$5 = x_{-4}$

In the second table the range is referred to an interval that a fractional number in nega-decimal form can represent. For nega-decimal system, the range is given by $[-10/11, +1/11] = [-0.909, +0.091]$.

§2.1.2. Negative binary number system

Example 4 *Let $X = (x_4x_3x_2x_1x_0)_{-2}$ be a nega-binary number system with $n = 5$ digits. Its range can be given as $[X_{Min}, X_{Max}]$ and*

$$\begin{aligned} X_{Max} &= 10101_{-2} = 16 + 4 + 1 = 21_{10}, \textit{ and} \\ X_{Min} &= 01010_{-2} = -8 - 4 = -12_{10}. \end{aligned}$$

Its arithmetic operations and conversions are similar to those for nega-decimal system discussed in the last section. Conversion from nega-binary to binary is simple and can be realized with a binary subtraction operation.

On the other hand, for a given binary number, conversion to nega-binary representation can be realized with steps similar to those shown in Part (3) in Example 3. The range for a fractional number in nega-binary system can also be obtained as $[-0.667, +0.333]$, which can be shown as follows: Let F be a fractional number in nega-binary system. Then

$$\begin{cases} F^{-}_{\text{max}} = -(2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} + \cdots) = -0.667_{10} \\ F^{+}_{\text{max}} = (2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + \cdots) = 0.333_{10} \end{cases}$$

§2.2. A General Class of Fixed-Radix Number Systems

Each n-digit number system is characterized by a positive radix β, and a vector Λ of length n, $\Lambda = (\lambda_{n-1}, \lambda_{n-2}, \dots, \lambda_0)$ where $\lambda_i \in \{-1, 1\}$. Such a system can be identified by $< n, \beta, \Lambda >$. Then the value of number

$$X = (x_{n-1}, \dots, x_0)$$

in the system $< n, \beta, \Lambda >$ can be decided by

$$X = \sum_{i=0}^{n-1} \lambda_i x_i \beta^i.$$

The use of λ_i allows us to select between positive and negative weights. For example,

- $\lambda_i = 1$ for every i: positive-radix system
- $\lambda_i = (-1)^i$ for every i: negative-radix system
- $\lambda_{n-1} = -1$, and $\lambda_i = 1$ for $i = 0, 1, \dots, n-2$: radix-complement system.

§2.3. Signed-Digit (SD) Number Systems

For a radix-r number system, the digit set is by default defined as $\{0, 1, \ldots, r-1\}$. If this digit set is extended to including some signed digits, say, $\{-(r-1), -(r-2), \ldots, -1, 0, 1, \ldots, r-1\}$, then we have a radix-$r$ signed-digit number system.

Let $\overline{x} = -x$, then the digit set for the radix-r signed-digit number system is given by

$$\{\overline{r-1}, \overline{r-2}, \ldots, \overline{1}, 0, 1, \ldots, r-2, r-1\}.$$

Example 5 *For $r = 10$, the allowed digits are $\{\overline{9}, \ldots, \overline{1}, 0, 1, \ldots, 9\}$, and when $n = 2$, $\overline{9}\,\overline{9} \le X \le 99$, which includes 199 numbers. However with two digits x_1 and x_2 each having 19 possibilities there are $19^2 = 361$ representations. So the number system is redundant. Some numbers can have more than one representation. For example, $(1) = (1\,\overline{9}) = 1, (0\,\overline{1}) = (\overline{1}\,9) = -1$. Out of the 361 representations, $361 - 199 = 162$ are redundant and thus there is $162/199 = 81\%$ redundancy.*

To reduce the amount of redundancy, we can choose digit set as

$$x_i \in \{\overline{a}, \overline{a-1}, \ldots, \overline{1}, 0, 1, \ldots, a-1, a\}, \text{ with } \left\lceil \frac{r-1}{2} \right\rceil \le a \le r-1.$$

At least r different digits are needed to represent a number in a radix-r number system. With $\overline{a} \le x_i \le a$ we have $2a+1$ digits.

$$2a + 1 \ge r \quad \Longrightarrow \quad a \ge \left\lceil \frac{r-1}{2} \right\rceil.$$

One advantage of performing addition/subtraction using SD representations is that the carry propagation chains can be eliminated. For the following operation:

$$(x_{n-1}, \ldots, x_0) \pm (y_{n-1} \ldots, y_0) = (s_{n-1} \ldots, s_0),$$

we want to break the carry chains by having s_i depend only on the four operand digits x_i, y_i, x_{i-1} and y_{i-1}.

Algorithm 1 *Perform radix-r signed-digit number addition*

$$(x_{n-1}, \ldots, x_0) + (y_{n-1} \ldots, y_0) = (s_{n-1} \ldots, s_0), \text{ where } x_i, y_i, s_i \in \{\overline{a}, \ldots, a\},$$

without generating carry chains.

Step 1. Compute an interim sum u_i and a carry digit c_i: $u_i = x_i + y_i - rc_i$, where

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geqslant a \\ \overline{1} & \text{if } (x_i + y_i) \leqslant \overline{a} \\ 0 & \text{if } |x_i + y_i| < a \end{cases}$$

Step 2. Calculate the final sum digit: $s_i = u_i + c_{i-1}$.

Example 6 *Let $r = 10$ and $a = 6$. The digit set is $x_i \in \{\overline{6}, \ldots, \overline{1}, 0, 1, 6\}$. Then we have $u_i = x_i + y_i - 10c_i$ and*

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geq 6 \\ \overline{1} & \text{if } (x_i + y_i) \leq \overline{6} \\ 0 & \text{if } |x_i + y_i| < 6 \end{cases}$$

Lets take as an example the following addition of two decimal numbers $4536 + 1466$.

$$\begin{array}{ccccccccc} & 4 & & 5 & & 3 & & 6 \\ + & 1 & & 4 & & 6 & & 6 \\ \hline & & \hookleftarrow & & \hookleftarrow & & \hookleftarrow & \\ & 6 & & 0 & & 0 & & 2 \end{array}$$

In the table, the symbol "$\hookleftarrow$" denotes that a carry digit is generated. Now we want to use Algorithm 1 to perform the addition so that the carry chain can be eliminated, which is shown as follows.

$$\begin{array}{cccccccl} & & & 4 & 5 & 3 & 6 & \\ + & & & 1 & 4 & 6 & 6 & \\ \hline & & 0 & 1 & 1 & 1 & & c_{i-1} \\ & & & 5 & \overline{1} & \overline{1} & 2 & u_i \\ \hline & & & 6 & 0 & 0 & 2 & s_i \end{array}$$

In the following we are going to show that Algorithm 1 can also be used for converting a conventional representation into the SD representation.

Example 7 *Convert the decimal number 27956 into a decimal SD representation with $a = 6$.*

$$\begin{array}{ccccccl} & 2 & 7 & 9 & 5 & 6 & x_i + y_i \\ \hline 0 & 1 & 1 & 0 & 1 & & c_{i-1} \\ & 2 & \overline{3} & \overline{1} & 5 & \overline{4} & u_i \\ \hline & 3 & \overline{2} & \overline{1} & 6 & \overline{4} & s_i \end{array}$$

Convert the SD representation back into the conventional decimal representation:

$$3\,\overline{2}\,\overline{1}\,6\,\overline{4} = 30060 - 2104 = 27956.$$

To guarantee the carry chains are broken, one has to make sure $s_i = u_i + c_{i-1} \le a$. So $|u_i| \le a - 1$ since $|c_{i-1}| \le 1$.

Let us check with the extreme cases for u_i so that we can find the condition for a which makes Algorithm 1 work properly. The largest value that $x_i + y_i$ can assume is $2a$. When $x_i + y_i = 2a$ we have $c_i = 1$ and $u_i = 2a - r$. Since $a \le r - 1$, it follows $u_i = 2a - r \le a - 1$ or $a \le r - 1$, which is obvious.

When $x_i + y_i = a$ we have $c_i = 1$ and $u_i = a - r < 0$. Then from $|u_i| = r - a \le a - 1$, it follows $a \ge \left\lceil \frac{r+1}{2} \right\rceil$.

So we conclude that the condition for choosing a radix-r SD number system to break carry chains using Algorithm 1 is

$$\left\lceil \frac{r+1}{2} \right\rceil \le a \le r - 1.$$

For decimal SD number system, the above condition is $a \ge 6$.

§2.4. Binary SD Number System

When $r = 2$ and $a = 1$, the digit set is given by $\{\bar{1}, 0, 1\}$. Since the condition $\left\lceil \frac{r+1}{2} \right\rceil = 2 \le a$ can not be satisfied, there is no guarantee that a new carry will not be generated using Algorithm 1.

Rewrite Algorithm 1 for the binary case:

Step 1. Compute an interim sum u_i and a carry digit c_i: $u_i = x_i + y_i - 2c_i$, where

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geqslant 1, \\ \bar{1} & \text{if } (x_i + y_i) \leqslant \bar{1}, \\ 0 & \text{if } |x_i + y_i| = 0. \end{cases}$$

Step 2. Calculate the final sum digit: $s_i = u_i + c_{i-1}$.

The above two steps are equivalent to the following table.

$x_i y_i$	00	01	$0\bar{1}$	11	$\bar{1}\bar{1}$	$1\bar{1}$
c_i	0	1	$\bar{1}$	1	$\bar{1}$	0
u_i	0	$\bar{1}$	1	0	0	0

Table 1: Binary case for Algorithm 1.

In the following example we show that using Table 1 does not guarantee that a new carry will not be generated.

Example 8 *Perform* $-9+29$.

$$
\begin{array}{cccccccccl}
 & & & 0 & \overline{1} & 1 & \overline{1} & 1 & 1 & \\
+ & & & 1 & 0 & 0 & \overline{1} & 0 & 1 & \\
\hline
 & & 1 & \overline{1} & 1 & \overline{1} & 1 & 1 & & c_{i-1} \\
 & & & \overline{1} & 1 & \overline{1} & 0 & \overline{1} & 0 & u_i \\
\hline
 & & * & * & * & * & 1 & 0 & 0 & s_i
\end{array}
$$

When $i=3$, $u_3=c_2=\overline{1}$ *so there is a new carry* $c_3=\overline{1}$ *is generated. Another possible place where a new carry may be generated is* $u_4=c_3=1$.

There are two scenarios that a new carry will be generated with Table 1:

Case 1: $c_{i-1}=u_i=1$

Case 2: $c_{i-1}=u_i=\overline{1}$.

For Case 1, when $c_{i-1}=1$ from Table 1 we have $x_{i-1}y_{i-1}=11$ or 01. When $u_i=1$ from Table 1 we have $x_iy_i=0\,\overline{1}$ and $c_i=\overline{1}$. In this case, when $x_{i-1}y_{i-1}=11$ or 01 we make the following changes:

$$
\left\{\begin{array}{lcl} u_i & = & 1 \\ c_i & = & \overline{1} \end{array}\right. \Longrightarrow \left\{\begin{array}{lcl} u_i & = & \overline{1} \\ c_i & = & 0 \end{array}\right.
$$

For Case 2, when $c_{i-1}=\overline{1}$ from Table 1 we have $x_{i-1}y_{i-1}=\overline{1}\,\overline{1}$ or $0\,\overline{1}$. When $u_i=\overline{1}$ from Table 1 we have $x_iy_i=01$ and $c_i=1$. In this case, when $x_{i-1}y_{i-1}=\overline{1}\,\overline{1}$ or $0\,\overline{1}$ we make the following changes:

$$
\left\{\begin{array}{lcl} u_i & = & \overline{1} \\ c_i & = & 1 \end{array}\right. \Longrightarrow \left\{\begin{array}{lcl} u_i & = & 1 \\ c_i & = & 0 \end{array}\right.
$$

After incorporating the above changes into Table 1, we have the following Table 2.

x_iy_i	00	01	01	$0\,\overline{1}$	$0\,\overline{1}$	11	$\overline{1}\,\overline{1}$
$x_{i-1}y_{i-1}$	–	neither is $\overline{1}$	at least one is $\overline{1}$	neither is $\overline{1}$	at least one is $\overline{1}$	–	–
c_i	0	1	0	0	$\overline{1}$	1	$\overline{1}$
u_i	0	$\overline{1}$	1	$\overline{1}$	1	0	0

Table 2: Modified Addition Algorithm for Binary SD Representations.

Example 9 *Repeating Example 8 using the algorithm shown in Table 2.*

$$\begin{array}{cccccccc|c}
 & & 0 & \bar{1} & 1 & \bar{1} & 1 & 1 & \\
+ & & 1 & 0 & 0 & \bar{1} & 0 & 1 & \\
\hline
 & 0 & 0 & 0 & \bar{1} & 1 & 1 & & c_{i-1} \\
 & & 1 & \bar{1} & 1 & 0 & \bar{1} & 0 & u_i \\
\hline
 & & 1 & \bar{1} & 0 & 1 & 0 & 0 & s_i
\end{array}$$

Clearly, there is no carry generated.