Computing with the Binary and Dyadic representation of numbers

General monadic discrimination

We write $\underline{n} \equiv \overbrace{S \cdots S}^{n}(0)$. For any number $n > 0$ exactly **one** formula holds:

$$x = 0 \vee x = \underline{1} \vee \cdots \vee x = \underline{n-1} \vee \exists! y\, x = y + \underline{n}$$

This can be used in clausal definitions. The sequence F_n of **Fibonacci** is defined:

$$F_0 = 1$$
$$F_1 = 1$$
$$F_{n+2} = F_{n+1} + F_n$$

F_n gives the number of **pairs** of rabbits at the beginning of the year $n = 0, 1, \ldots$ when we start with one pair of young rabbits. A pair of at least one year old rabbits breeds each year a new pair of rabbits.

year	young	old	total
0	$1 = F_1$	0	$1 = F_0$
1	0	$1 = F_1$	$1 = F_1$
2	$1 = F_1$	$1 = F_1$	$2 = F_2$
3	$1 = F_1$	$2 = F_2$	$3 = F_3$
4	$2 = F_2$	$3 = F_3$	$5 = F_4$
5	$3 = F_3$	$5 = F_4$	$8 = F_5$
$\dots$	$\dots$	$\dots$	$\dots$
$n+2$	F_n	F_{n+1}	F_{n+2}

F_n grows as **fast** as the exponential function 2^n. But the **computation** of F_n requires on the order of 2^{F_n} successor operations.

Consider the clausal definition of $Fa(n, y, o)$:

$$Fa(0, y, d) = y$$
$$Fa(n + 1, y, d) = Fa(n, d, y + d)$$

$$Fa(n, F_k, F_{k+1}) = Fa(n - 1, F_{k+1}, F_{k+2}) = \\ \dots = Fa(0, F_{k+n}, F_{k+n+1}) = F_{k+n}$$

Hence $F_n = Fa(n, F_0, F_1) = Fa(n, 1, 1)$

Binary representation of numbers

Every number $x > 0$ can be **uniquely** written as $x = \sum_{i=0}^{n} d_i{\cdot}2^i$ where $d_i \leq 1$ for $i = 0, \ldots, n-1$ and $d_n = 1$ are its **binary digits**. We also have $0 = \sum_{i=0}^{1} 0{\cdot}2^0$. The **binary length** $|x|_b$ of x is the number of its binary digits ($n + 1$ in the first case and 0 when $x = 0$).

Note: The number x in the monadic representation $\underline{n}$ takes x succesor operations which is its **monadic length**. The binary length of x is on the length of $\log(x)$.

Arithmetic operations on binary numbers are done similarly as the corresponding operations on the decimal notation as we know them from the elementary school.

Using the **primitive** recursion $0 + y = y$ and $S(x) + y = S(x + y)$ we need n recursions to compute $\underline{n} + \underline{m}$. We know that $\max(|n|_b, |m|_b)$ suffices for the binary addition.

Binary successor functions

Consider the functions $S_0(x) = 2 \cdot x + 0$ and $S_1(x) = 2 \cdot (x) + 1$. We have $0 = S_0(0)$ and $1 = S_1(0)$. For any $x \geq 2$ We have

$$x = \sum_{i=0}^{n+1} d_i \cdot 2^i = (\sum_{i=1}^{n+1} d_i \cdot 2^i) + d_0 =$$

$$(\sum_{i=0}^{n} d_{i+1} \cdot 2^{i+1}) + d_0 = 2 \cdot (\sum_{i=0}^{n} d_{i+1} \cdot 2^i) + d_0$$

Hence for the unique $y = \sum_{i=0}^{n} d_{i+1} \cdot 2^i$ we have $x = 2 \cdot y + 0$ or $x = 2 \cdot y + 1$, i.e. $x = S_{d_0}(y)$. Thus every number x is uniquely formed from its *binary predecessor* y by a **binary successor** function S_{d_0}. **Note** that if $x > 0$ then $y < x$.

We can repeat this as in the **Horner's** scheme for the evaluation of polynomials: Thus

$$\sum_{i=0}^{n} d_i{\cdot}2^i = S_{d_0} S_{d_1} \ldots S_{d_n}(0)$$
$$6 = S_0(3) = S_0 S_1(1) = S_0 S_1 S_1(0)$$

The indices 110 read from the end constitute the **binary representation** of 6.
For better **visualization** we write the binary successors in the **postfix** notation:
$S_0(x) \equiv x\mathbf{0} \qquad S_1(x) \equiv x\mathbf{1}$.
Thus $6 = 0\mathbf{110}$. note that $S_0 S_1 S_1(a) \equiv a\mathbf{110}$
For every x exactly one of the following holds:

$$\exists! y\, x = y\mathbf{0} \vee \exists! y\, x = y\mathbf{1}$$

This can be used in CL in the **binary discrimination** to compute **binary predecessors**:

$$Div_2(x) = y \leftarrow x = y\mathbf{0}$$
$$Div_2(x) = y \leftarrow x = y\mathbf{1}$$

We have $Div_2(x) = x \div 2$ and we can also write:

$$Div_2(x\mathbf{0}) = x$$
$$Div_2(x\mathbf{1}) = x$$

Arithmetic operations in binary

The **successor** function S can be clausally defined as $x +_b 1$ such that $S(x) = x +_b 1$ as follows:

$$x\mathbf{0} +_b 1 = x\mathbf{1}$$
$$x\mathbf{1} +_b 1 = (x +_b 1)\mathbf{0}$$

For the second clause **note**
$x\mathbf{1} + 1 = (2{\cdot}x + 1) + 1 = 2{\cdot}x + 2 = (x + 1)\mathbf{0}$
Also note:

$$\begin{array}{r r l} x\mathbf{0} & x\mathbf{1} & \\ 0\mathbf{1} & 0\mathbf{1} & \\ \hline x\mathbf{1} & (x + 1)\mathbf{0} & +1 \text{ is the } \textbf{carry} \end{array}$$

Addition in binary (I)

$$\frac{\begin{array}{r} x\mathbf{0} \\ y \end{array}}{y} \quad \text{if } x = 0$$

$$\frac{\begin{array}{r} x\mathbf{0} \\ y\mathbf{0} \end{array}}{(x+y)\mathbf{0}} \quad \text{if } x > 0\text{; note then } (\mathbf{0}x) > x$$

$$\frac{\begin{array}{r} x\mathbf{0} \\ y\mathbf{1} \end{array}}{(x+y)\mathbf{1}} \quad \text{if } x > 0\text{; note then } (\mathbf{0}x) > x$$

$$\frac{\begin{array}{r} x\mathbf{1} \\ y\mathbf{0} \end{array}}{(x+y)\mathbf{1}} \quad \text{note that } x\mathbf{1} > x$$

$$\frac{\begin{array}{r} x\mathbf{1} \\ y\mathbf{1} \end{array}}{(x+y+1)\mathbf{0}} \quad \text{note the } \textbf{carry} \text{ and } x\mathbf{1} > x$$

The recursion **descends** in the first argument.

Addition in binary (II)

Using **binary discrimination** we program $x +_b y = x + y$ in the **term** notation:

$$
\begin{array}{l}
x +_b y = \text{ if } x = \\
\qquad\qquad x_1\mathbf{0} \;\; \to \text{ if} \\
\qquad\qquad\qquad\qquad x{=}0 \to y \\
\qquad\qquad\qquad\qquad x{>}0 \to \text{ if } y = \\
\qquad\qquad\qquad\qquad\qquad\qquad y_1\mathbf{0} \to (x_1 +_b y_1)\mathbf{0} \\
\qquad\qquad\qquad\qquad\qquad\qquad y_1\mathbf{1} \to (x_1 +_b y_1)\mathbf{1} \\
\qquad\qquad x_1\mathbf{1} \;\; \to \text{ if } y = \\
\qquad\qquad\qquad\qquad y_1\mathbf{0} \to (x_1 +_b y_1)\mathbf{1} \\
\qquad\qquad\qquad\qquad y_1\mathbf{1} \to ((x_1 +_b y_1) +_b 1)\mathbf{0}
\end{array}
$$

and in CL

$$
\begin{array}{ll}
x\mathbf{0} +_b y = y & \leftarrow x = 0 \\
x\mathbf{0} +_b y = (x +_b y_1)\mathbf{0} & \leftarrow x > 0 \wedge y = y_1\mathbf{0} \\
x\mathbf{0} +_b y = (x +_b y_1)\mathbf{1} & \leftarrow x > 0 \wedge y = y_1\mathbf{1} \\
x\mathbf{1} +_b y\mathbf{0} = (x +_b y_1)\mathbf{1} & \\
x\mathbf{1} +_b y\mathbf{1} = ((x +_b y) +_b 1)\mathbf{0} &
\end{array}
$$

Dyadic representation of numbers

Take any $x > 0$ and express $x+1$ in **binary** as $x+1 = \sum_{i=0}^{n} d_i{\cdot}2^i$. We have $n > 0$ and $d_n = 1$. Hence

$$x+1 = 2^n + \sum_{i=0}^{n-1} d_i{\cdot}2^i = 1 + \sum_{i=0}^{n-1} 1{\cdot}2^i + \sum_{i=0}^{n-1} d_i{\cdot}2^i$$

Thus

$$x = \sum_{i=0}^{n-1} (d_i + 1){\cdot}2^i$$

Every *positive* number x can be thus uniquely written in **dyadic notation** as $x = \sum_{i=0}^{n} d_i{\cdot}2^i$ with the **dyadic digits** $d_i = 1, 2$. **Dyadic length** $|x|_d$ is the number of dyadic digits of x where we set $|0|_d = 0$.
We have

$$1 = (1)_d \quad 2 = (2)_d \quad 3 = (11)_d \quad 4 = (12)_d$$
$$5 = (21)_d \; 6 = (22)_d \; 7 = (111)_d \; 8 = (112)_d$$

Dyadic discrimination

Similarly as with binary representation we can use the **dyadic successors**

$S1(x) \equiv x\mathbf{1} = 2{\cdot}x + 1 \qquad S2(x) \equiv x\mathbf{2} = 2{\cdot}x + 2$

in the scheme of **Horner** and write every positive number $x = \sum_{i=0}^{n} d_i{\cdot}2^i$ as
$x = S_{d_0} S_{d_1} \ldots S_{d_{n-1}} S_{d_n}(0)$.
Thus $8 = S_2 S_1 S_1(0) \equiv 0\mathbf{112}$.

Dyadic discrimination in CL is based on the fact that exactly one of the following holds:

$$x = 0 \lor \exists! y\, x = y\mathbf{1} \lor \exists! y\, x = y\mathbf{2}$$

The troublesome **leading zeroes** problem of binary numbers does not exist.

We compute with dyadic numbers similarly as with binary. For instance, for the the dyadic successor $x +_d 1 = S(x)$ we have:

$$
\begin{array}{ccc}
x\mathbf{1} & x\mathbf{2} & \\
0\mathbf{1} & 0\mathbf{1} & \\
\hline
x\mathbf{2} & (x+1)\mathbf{1} & +1 \text{ is the } \textbf{carry}
\end{array}
$$

$0 +_d 1 = S1(0)$
$x\mathbf{1} +_d 1 = x\mathbf{2}$
$x\mathbf{2} +_d 1 = (x +_d 1)\mathbf{1}$

Dyadic multiplication $x \times_d y$ is

$0 \times_d y = 0$
$x\mathbf{1} \times_d y = z +_d z +_d y \leftarrow x \times_d y = z$
$x\mathbf{2} \times_d y = z +_d z \qquad \leftarrow x \times_d y +_d y = z$

Note that

$$x\mathbf{1}\cdot y = (2\cdot x + 1)\cdot y = 2\cdot x\cdot y + y$$
$$x\mathbf{2}\cdot y = (2\cdot x + 2)\cdot y = 2\cdot x\cdot y + 2\cdot y = 2\cdot(x\cdot y + y)$$