Redundant Radix Numeration Systems

Daniel S. Roche

December 5, 2006

1 Introduction

The idea of using redundant numeration systems to ease arithmetic computation has been around since at least 1726. John Colson, then Lucasian Chair of Mathematics at Cambridge, submitted a paper to the Royal Society outlining a method of what he terms "negativo-affirmative arithmetick" [3].

What Colson actually reccommends is an alternate numeration system. His system still uses the familiar base 10, but the digits are from $\overline{9}$ to 9 (I use the standard notation of $\overline{x} = -x$).

In Colson's system, the ease in arithmetic comes from the fact that operations are simpler when the digits are all small in absolute value. So, to perform an arithmetic operation, Colson's algorithm is to first convert the number (in standard base-10) to "small figures" (all digits at most 5 in absolute value), then perform the arithmetic, and then convert back to the standard representation.

For example, the addition of two numbers written with "small figures" will only produce a carry when we have two 5's of the same sign in the same position in each number. And if we require that no two 5's of the same sign be adjacent in any number in "small figures", then it is easy to see that a carry will never propagate further than one place.

For this number system, the biggest arithmetic advantages come when adding together a group of numbers all at once, and for multiplication. Notice that the number system is *redundant*, or *ambiguous*, in the sense that one number could have multiple representations. For example, $1234 = 2\overline{7}\overline{7}4$ in this system.

2 Theoretical Results

2.1 Preliminary Definitions

I use the notation introduced in [4].

Definition 1. A *number system* N is a $(k+1)$-tuple of integers $(r, d_1, d_2, \ldots, d_k)$ such that $v, |r| \geq 1$ and $d_1 < d_2 < \cdots < d_k$. We say that r is the *base* and $D = \{d_1, d_2, \ldots, d_k\}$ is the *digit set* of N.

Note that I will often write (r, D) to mean $(r, d_1, d_2, \ldots, d_k)$.

In particular, a number system, in the way that it is defined here, is specifically a *radix* number system, and the base r is also called the radix.

Definition 2. Let N be a number system with base r and digit set D. The *evaluation function* of N is $E_N : D^* \to \mathbb{Z}$ given by:

$$E_N(a_0a_1a_2 \cdots a_n) = a_0 \cdot r^n + a_1 \cdot r^{n-1} + \cdots + a_{n-1} \cdot r + a_n.$$

In particular, the empty string always evaluates to zero. That is, $E_N(\epsilon) = 0$.

Notice that the evaluation of a digit string amounts to polynomial arithmetic. This is the justification for performing arithmetic with arbitrary number systems. Throughout this paper, some implicit assumptions are made regarding the validity of digit set arithmetic; for a formal justification, see [5].

Definition 3. A number system $N = (r, D)$ is *complete* for the set $S \subseteq \mathbb{Z}$ iff

$$\forall n \in S,\ \exists \mathbf{w} \in D^* \text{ s.t. } E_N(\mathbf{w}) = n.$$

That is, a number system is complete whenever there is some representation in the number system for every number in the given set. To state this another way, (r, D) is complete if and only if there exists some subset $A \subseteq D^*$ such that the evaluation function $E_N : \ A \to S$ is onto.

Definition 4. A number system $N = (r, D)$ is *redundant* iff there exist two words $\mathbf{w}_1$ and $\mathbf{w}_2$ in D^*, which do not start with `0`, such that $E_N(\mathbf{w}_1) = E_N(\mathbf{w}_2)$.

So redundancy corresponds to the evaluation function E_N being one-to-one.

In most papers and surveys of number systems, problems are explicitly formulated to exclude redundancy — that is, the only number systems which are considered are those which are complete and *non*redundant for some set S. If these conditions hold, then the evaluation function E_N is a bijection and therefore its inverse is also a bijection. However, here we examine specifically those number systems which are both redundant and complete for either $\mathbb{N}$ or $\mathbb{Z}$.

2.2 Necessary and Sufficient Conditions

Now we examine briefly what the digit set D will look like if the number system (r, D) is redundant and complete.

First, another definition:

Definition 5. A set $S \subseteq \mathbb{Z}$ is a *complete residue system* (CRS) mod r iff $|S| = |r|$ and

$$\{s \bmod r | s \in S\} = \{m \bmod r | m \in \mathbb{Z}\}.$$

Now we can give a necessary condition for D:

Theorem 6. *If $N = (r, D)$ is redundant and complete for $\mathbb{N}$, then D contains a CRS mod r and $|D| > |r|$.*

Proof. Let $N = (r, D)$ be some number system which is complete for $\mathbb{N}$ and redundant.

First note that, for any nonempty word $\mathbf{w} \in D^*$, if $E_N(\mathbf{w}) \equiv m \pmod{r}$, then the last digit in $\mathbf{w}$ must be congruent to m mod r.

Then, since each of $\{1, 2, \dots, r\}$ is in $\mathbb{N}$, it is clear that D must contain a CRS mod r.

Also, since N is redundant, there exist two different, nonempty words $\mathbf{w}_1$ and $\mathbf{w}_2$ in D^+ such that $E_N(\mathbf{w}_1) = E_N(\mathbf{w}_2)$.

Let i be the index of the last position (least significant digit) in which the two words differ. Then the two digits at position i must be congruent mod r.

Then, since D contains a CRS *and* contains two digits in the same congruency class, $|D| > r$. $\square$

There is also a simple sufficient condition for the number system N to be redundant and complete:

Theorem 7. *If the following conditions hold, then $N = (r, D)$ is redundant and complete for the set $\mathbb{N}$:*

- $|r| \geq 2$
- $\exists\ D' \subsetneq D$ *such that* $N' = (r, D')$ *is complete for* $\mathbb{N}$

Proof. Since (r, D') is complete for $\mathbb{N}$, then D' must contain a complete residue system mod r, from the previous proof.

Then, since D' is a proper subset of D, then there is some digit in $D \setminus D'$, call if d. And since D' contains a CRS mod r, then there is some digit $d' \in D'$ such that $d \equiv d' \pmod{r}$.

This means that $d' - d = kr$, for some integer $k \neq 0$. First consider the case where $k > 0$. Then, since (r, D') is complete for $\mathbb{N}$, there exists some $\mathbf{w} \in D'^*$ such that $E_N(\mathbf{w}) = k$.

Then $E_N(\mathbf{w}d) = kr + d = d'$. And clearly $E_N(d') = d'$. Thus the number d' can be written in two different ways, so N is redundant.

If instead we have $k < 0$ above, then just consider $d - d' = kr$ instead and the above argument shows that there are two different representations for d. $\square$

From these conditions we can see, for example, that $(2, \overline{\mathtt{4}}, \mathtt{1})$ and $(-5, \overline{\mathtt{9}}, \overline{\mathtt{4}}, \mathtt{0}, \mathtt{1}, \mathtt{5}, \mathtt{6})$ are not complete for $\mathbb{N}$ and redundant, but $(3, \overline{\mathtt{1}}, \mathtt{0}, \mathtt{1}, \mathtt{2})$ is. However, we still cannot tell if, for example $(2, \overline{\mathtt{5}}, \overline{\mathtt{3}}, \mathtt{2})$ is complete and redundant.

2.3 Automaticity

There has been some study of automatic sequences defined over non-standard digit sets, for example in [1], but again the authors have chosen to restrict the number systems examined to those which are both complete and nonredundant.

So a natural question is to ask if the notion of automatic sequences can be extended to include redundant number systems, and if so, what is the relationship to the standard definitions of automatic sequences.

So first, we extend the definition of a k-automatic sequence given in [2] to allow for non-standard number systems:

Definition 8. Let $N = (r, D)$ be some number system which is complete for $\mathbb{N}$. The sequence $(a_n)_{n\geq 0}$ over a finite alphabet Δ is *N-automatic* iff there exists a DFAO $M = (Q, D, \delta, q_0, \Delta, \tau)$ such that $a_n = \tau(\delta(q_0, \mathbf{w}))$, for all $n \geq 0$ and all $\mathbf{w} \in D^*$ with $E_N(\mathbf{w}) = n$.

Note that the usual definition of k-automaticity is now equivalent, using this definition, to (k, Σ_k)-automaticity (here and for the remainder, Σ_k refers to the standard base-k digit set $\{\mathtt{0}, \mathtt{1}, \ldots, k-1\}$).

To illustrate some of the differences between k-automaticity and
N-automaticity for a redundant number system N, consider the Thue-Morse sequence $(t_n)_{n\geq 0} = \mathtt{01101001}\cdots$, where t_n is defined to be the sum of the digits in the standard binary representation of n mod 2.

The DFAO in Figure 1 is well-known to output t_n on input of the standard binary representation of n; thus $(t_n)_{n\geq 0}$ is $(2, \Sigma_2)$-automatic. Notice that this automaton has just two states, and (as is the case for any 2-automatic sequence) each state has exactly 2 exit edges.

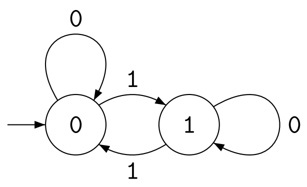

Figure 1: DFAO for $(t_n)_{n\geq 0}$ over $(2, \mathtt{0}, \mathtt{1})$

By contrast, it is not too difficult to verify that the DFAO in Figure 2 will output t_n on input of n represented in the number system $N = (2, \overline{\mathtt{1}}, \mathtt{0}, \mathtt{1})$. Notice the differences here. First, the number of states has doubled. However, the number of states has no bearing on the definition of automaticity.

What is significant is that, unlike any 2-automatic DFAO, each state now has 3 exit edges instead of two. This would seem to allow more freedom in the automaton. However, there is also the further restriction that each redundant representation of the same number (for instance, $\mathtt{100}\overline{\mathtt{1}} = \mathtt{111}$) must output the same value. In fact, the next theorem will show that 2-automaticity and N-automaticity are equivalent here, so these two differences "cancel each other out" in some sense.

In order to prove the equivalence of automaticity over redundant number systems, we first need a finite-state transducer to convert from a number in an arbitrary (radix) number system to the standard number system in the same base.

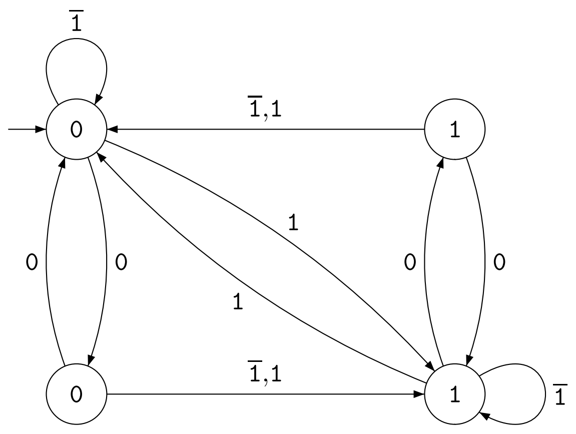

Figure 2: DFAO for $(t_n)_{n\geq 0}$ over $(2, \overline{1}, 0, 1)$

So let $N = (r, D)$ with $r \geq 2$, and define the finite-state transducer $T_N = (Q, D \cup \{0\}, \delta, 0, \Sigma_r, \lambda)$ with:

- $Q = \{-m, -m+1, \ldots, 0, 1, \ldots, n-1\}$ is the set of states, where m and n are the least non-negative integers such that $-m(r-1) \leq d \leq n(r-1)$ $\forall d \in D$.
- $D \cup \{0\}$ is the input alphabet (we must include 0 because we may need to pad the input with some leading 0's in order for the transducer to work.)
- $\delta : Q \times (D \cup \{0\}) \to Q$ is the transition function.
- Σ_r is the output alphabet.
- $\lambda : Q \times (D \cup \{0\}) \to \Sigma_r$ is the output function.

The transition and output functions δ and λ are defined as follows:
Let $q \in Q$ and $a \in (D \cup \{0\})$ be arbitrary. Since $q, a \in \mathbb{Z}$ and $r \geq 2$, we can write

$$q + a = br + c,$$

for some $b, c \in \mathbb{Z}$ with $0 \leq c < r$.

Then define

$$\delta(q, a) = b, \qquad \lambda(q, a) = c.$$

Since $0 \leq c < r$, $c \in \Sigma_r$, so λ is a valid function from $Q \times (D \cup \{0\})$ to Σ_r.

Now we need to show that $\delta(q, a) = b \in Q$. To see this, note that $-m \leq q \leq n-1$ and $-m(r-1) \leq a \leq n(r-1)$ from the definitions of Q, m, and n. Then

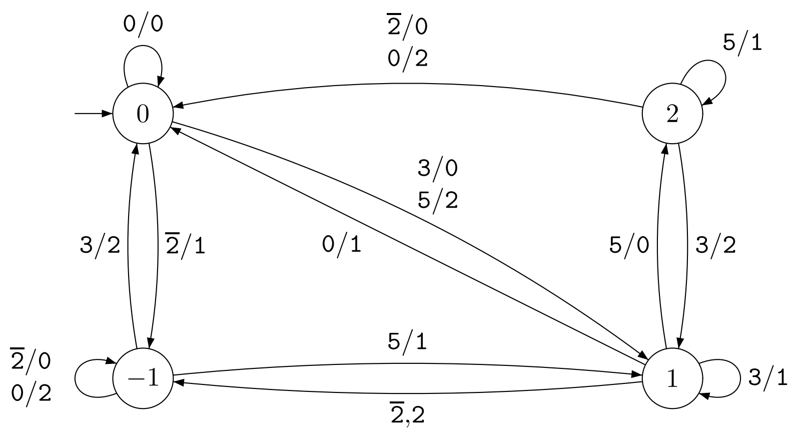

Figure 3: The transducer T_N for $N = (3, \overline{2}, 3, 5)$

we have:

$$\begin{array}{rcccl} -m - m(r-1) & \leq & q + a & \leq & (n-1) + n(r-1) \\ -mr & \leq & br + c & \leq & nr - 1 \\ -mr - (r-1) & \leq & br & \leq & nr - 1 \\ -(m+1)r & < & br & < & nr \\ -m & \leq & b & \leq & n - 1 \end{array}$$

Therefore $\delta(q, a) \in Q$, so δ is a valid function from $Q \times (D \cup \{0\})$ to Q.

And since D is finite, m and n are bounded, so then T_N is a valid finite-state transducer.

Before giving the proof that T_N actually maps non-negative integers represented in N to the corresponding representation in (r, Σ_r), it may be helpful to consider an example.

Figure 3 gives the transducer T_N when $N = (3, \overline{2}, 3, 5)$. So we have the zero state (which will exist in any transducer T_N), two positive states, and one negative state. The positive states can be thought of as "carry" states, and the negative ones as "borrow" states. Thus, we will always start in the zero state, and we always want to end in that state as well — this is why we may have to "pad" the input number in D^* with some leading zeroes. Also, note that the transducer works from the least significant figure first, thus reading numbers from right to left as opposed to the usual left-to-right order.

All of the reasoning behind this will be thoroughly explained in the proof to follow, but as an example, imagine we want to use the transducer in Figure 3 to convert the number $5\overline{2}3$ to the standard base-3 representation. So we feed the number into the transducer, reversed, and in this case we need one leading zero. It is easy to confirm that on input $3\overline{2}50$ the transducer outputs 0211. The output is also given least-significant digit first, so this indicates that $5\overline{2}3 = 1120$ in base 3, and in fact we can see that both of these are equal to 42 (written in base 10).

Before beginning the actual correctness proof for this transducer conversion, we need a lemma regarding T_N:

Lemma 9. *Let $N = (r, D)$ be some number system and $\mathbf{w} \in D^n$. Then if $T(\mathbf{w}^R) = \mathbf{u} = u_0 u_1 \cdots u_n$ and the transducer ends in state q, then*

$$E_N(\mathbf{w}) = qr^{n+1} + E_{(r,\Sigma_r)}(\mathbf{u}^R).$$

Proof. By induction on $|\mathbf{w}|$.

The base case, $\mathbf{w} = \epsilon$, is a simple consequence from the fact that the start state is 0 and $E_N(\epsilon) = 0$ for any number system N.

So assume the statement is true for $n = k$, $k \geq 0$.

Let $\mathbf{w} \in D^{k+1}$. Suppose $T_N(\mathbf{w}^R) = u_0 u_1 \cdots u_k u_{k+1}$.

Then, since $|\mathbf{w}| \geq 1$, we can write $\mathbf{w} = a\mathbf{w}'$ for some digit $a \in D$. And therefore $T_N((\mathbf{w}')^R) = u_0 u_1 \cdots u_k$.

Let q be the state of T_N after reading $(\mathbf{w}')^R$. Then we know $\lambda(q, a) = u_{k+1}$. And suppose $\delta(q, a) = b \in Q$. Then $a = br + u_{k+1} - q$.

So, using the induction hypothesis, we have

$$\begin{aligned}
E_N(\mathbf{w}) &= E_N(a\mathbf{w}') \\
&= ar^{k+1} + E_N(\mathbf{w}') \\
&= (br + u_{k+1} - q)r^{k+1} + qr^{k+1} + E_{(r,\Sigma_r)}\left((u_0 u_1 \cdots u_{k-1} u_k)^R\right) \\
&= br^{k+2} + u_{k+1}r^{k+1} + E_{(r,\Sigma_r)}(u_k u_{k-1} \cdots u_1 u_0) \\
&= br^{k+2} + E_{(r,\Sigma_r)}(u_k + 1 u_k u_{k-1} \cdots u_1 u_0) \\
&= br^{(k+1)+1} + E_{(r,\Sigma_r)}(\mathbf{u}^R) \qquad \square
\end{aligned}$$

This lemma explains why the transducer must end in state 0 for the input and output words to represent the same number. It also provides the following corollary giving a characterization of which words in D^* represent non-negative integers:

Corollary 10. *$E_N(\mathbf{w}) \geq 0$ iff $q \geq 0$, where q is the final state reached by T_N on reading $\mathbf{w}^R$.*

Now we can give the theorem showing T_N maps an arbitrary number system (r, D) to (r, Σ_r).

Theorem 11. *Let $N = (r, D)$ be any number system complete for $\mathbb{N}$ with $r \geq 2$. Then, for all $\mathbf{w} \in D^*$, $E_N(\mathbf{w}) \in \mathbb{N}$ implies*

$$E_N(\mathbf{w}) = E_{(r,\Sigma_r)}(T_N(\mathbf{w}^R \mathtt{0}^{n-1})^R).$$

Proof. From the corollary, we know that T_N must be in a non-negative state after reading $\mathbf{w}^R$. And from the transducer definition, we can see that $\delta(q, \mathtt{0}) < q$ whenever $q > 0$. Then, since there are only $n - 1$
positive states, T_N must end in state 0 after reading $\mathbf{w}^R \mathtt{0}^{n-1}$.

Clearly $E_N(\mathtt{0}^*\mathbf{w}) = E_N(\mathbf{w})$, so, from the lemma,

$$\begin{aligned} E_N(\mathbf{w}) &= 0 \cdot r^{k+1} + E_{(r,\Sigma_r)}(T_N(\mathbf{w}^R \mathtt{0}^{n-1})) \\ &= E_{(r,\Sigma_r)}(T_N(\mathbf{w}^R \mathtt{0}^{n-1})) \qquad \square \end{aligned}$$

Now we can use this transducer to prove the main result of this section, the equivalence of N-automaticity and k-automaticity.

Theorem 12. *If $N = (r, D)$ is any number system with $|r| \geq 2$ which is complete for $\mathbb{N}$, then a sequence over a finite alphabet is r-automatic if and only if it is N-automatic.*

The detailed proof of this theorem is identical to that given for Theorem 5.2.7 in [2], so I do not repeat it here in detail. All that needs to be shown to use that proof is that the language $\{\mathbf{w} \in D^* \mid E_N(\mathbf{w}) \in \mathbb{N}\}$ is regular. This is easily seen from Corollary 10 if we consider the transducer T_N as a simple finite automaton, with the non-negative states as the final states.

2.4 Further Results Derived from the Transducer

Now we use the transducer T_N to show some more theoretical results regarding redundant digit sets. First, we have a new sufficient condition:

Theorem 13. *Let $N = (r, D)$ be any number system containing a CRS mod r, with $r \geq 2$.*

Let the states of T_N be $\{-m, \ldots, 0, \ldots, n-1\}$.

Suppose that, for each $q \in \{-n+1, \ldots, 0, \ldots, m\}$, there exists a word $\mathbf{w} \in D^*$ such that $E_N(\mathbf{w}) = q$.

Then N is complete for $\mathbb{N}$.

Proof. Define a new transducer T'_N, which (informally) is the same as T_N, but with extra digits removed, and with the input and output swapped on each edge.

Formally, say $T_N = (Q, D \cup \{\mathtt{0}\}, \delta, 0, \Sigma_r, \lambda)$, and let D' be the subset of D which is a complete residue system mod r. Then set $T'_N = (Q, \Sigma_r, \delta', 0, D', \lambda')$, where δ' and λ' are defined as follows:

For any $q \in Q$ and $c \in \Sigma_r$, since D' is a CRS mod r, there is exactly one $a \in D'$ such that $q + a \equiv c \bmod r$. That is, $\lambda(q, a) = c$. Then define:

$$\begin{aligned} \delta'(q, c) &= \delta(q, a) \\ \lambda'(q, c) &= a \end{aligned}$$

Since there is only one $a \in D'$ satisfying $\lambda(q, a) = c$ for each q and c, these functions are well defined, and it is clear that $\delta'(q, c) \in Q$ and $\lambda'(q, c) \in D'$.

So T'_N is a valid finite-state trasducer mapping Σ_r to D'.

Now let $n \in \mathbb{N}$ be arbitrary. Suppose T'_N on input of $(n)_r^R$
outputs $\mathbf{w}^R$ and ends in state q (where $(n)_r$ is defined to be the standard base-r

representation of n). Then T_N on input of $\mathbf{w}^R$
outputs $(n)_r^R$ and ends in state q. So, from Lemma 9,

$$E_N(\mathbf{w}) = qr^{|\mathbf{w}|+1} + n.$$

Let $\mathbf{u} \in D^*$ be a word with $E_N(\mathbf{u}) = -q$. We know that such a word exists from the assumption of the theorem. Then

$$E_N(\mathbf{uw}) = (-q)r^{|\mathbf{w}|+1} + q \cdot r^{|\mathbf{w}|+1} + n = n.$$

Therefore N is complete for $\mathbb{N}$. □

To see at least one useful purpose for this new sufficient condition, recall Theorem 7 from earlier. It is a natural question to ask if this condition is also necessary. If so, then we have a very nice categorization of complete and redundant number systems.

However, 13 gives a counterexample to the necessity of 7.

Let $N = (2, \overline{5}, \overline{3}, 2)$. Then no 2-subset of D forms a complete number system for $\mathbb{N}$ with base 2. This is seen by examining all three such subsets:

$D' = \{\overline{5}, \overline{3}\}$ Let $\mathbf{w} \in D'^+$. Then $E_N(\mathbf{w}) = 2x - 3$ or $E_N(\mathbf{w}) = 2x - 5$, for some integer x. In either case, we can see that $\mathbf{w}$ cannot be even, so any even number cannot be represented with these digits.

$D' = \{\overline{5}, 2\}$ Suppose (r, D') is complete for $\mathbb{N}$, and choose $\mathbf{w} \in D'^*$ to be a representation of the number 5 with minimal length. Then the last digit of $\mathbf{w}$ must be $\overline{5}$, or else $E_N(\mathbf{w})$ is even. Then we have $\mathbf{w} = \mathbf{w}'\overline{5}$, and $2E_N(\mathbf{w}') - 5 = 5$. Therefore $E_N(\mathbf{w}') = 5$, so we have found a representation of 5 of shorter length, a contradiction.

$D' = \{\overline{3}, 2\}$ By the same logic as for the previous subset, there is no representation of 3 in (r, D').

And Theorem 13 tells us that $N = (2, \overline{5}, \overline{3}, 2)$ is complete for $\mathbb{N}$ as follows. The states of T_N are $\{-5, -4, \ldots, 0, 1\}$. That is, $m = 5$ and $n = 2$. And we have:

$$\begin{aligned}
-1 &= E_N(2\overline{5}) \\
0 &= E_N(\epsilon) \\
1 &= E_N(2\overline{3}) \\
2 &= E_N(2) \\
3 &= E_N(2\overline{3}2\overline{5}) \\
4 &= E_N(2\overline{3}2) \\
5 &= E_N(2\overline{3}2\overline{3})
\end{aligned}$$

So each number in $\{-n+1, \ldots, m\} = \{-1, 0, \ldots, 4, 5\}$ has a representation in N, so Theorem 13 tells us that N is complete for $\mathbf{N}$.

And therefore not every number system which is redundant and complete for $\mathbb{N}$ satisfies the conditions of Theorem 7.

3 Application: Carry-free addition

As is suggested by the methods of Colson outlined in the introduction, redundant number systems can be used to ease arithmetic computation. Recall that Colson's method was basically to renormalize each operand before each arithmetic operation, thus simplifying the computations.

However, to normalize a number n takes $O(\log n)$ time, which is the same as the time complexity just to add two numbers together, so this technique does not give any performance improvement if we are using computers.

Without normalization, we can see that it is impossible to completely eliminate carries. This is because, for any finite digit set, we will have some digit with greatest absolute value, so if this digit falls in the same position in two words (representing numbers) which are to be added, then the sum is not a digit, so we have a carry.

Instead, what we will eliminate is carry *propagation* — that is, any carry in one position will not produce a new carry in the next position. This will allow operations to be performed in parallel.

The method I will describe is given in [6]. My purpose in presenting it is only to provide a motivation for redundant number systems, so I will not rigorously prove everything or explain every derivation.

First, we need the following definition:

Definition 14. For a number system $N = (r, D)$, the *redundancy index* ρ is defined to be

$$\rho = |D| - r.$$

For the given algorithm to be correct for some number system $N = (r, D)$, we need to impose the following conditions on r and D:

- D is a set of consecutive integers including 0. That is, $D = \{-\alpha, -\alpha + 1, \ldots, -1, 0, 1, \ldots, \beta - 1, \beta\}$ for some non-negative integers α and β.
- $r \geq 3$
- If $\alpha = 1$ or $\beta = 1$, then $\rho \geq 3$. Otherwise, $\rho \geq 2$.

Then to find the sum of two numbers x and y represented in the number system N, calculate the following for each position i of the two numbers:

Position sum $p_i = x_i + y_i$

Transfer digit t_{i+1}

Interim sum $w_i = p_i - rt_{i+1}$

Final sum $s_i = w_i + t_i$

Then the final sum of x and y is just given by the concatenation of the final sum digits s_i.

All of the computations are straightforward (and given above) except for the selection of the transfer digit t_{i+1}. To show how these are chosen, we first need some constants:

$$\lambda = \left\lceil \frac{\alpha}{r-1} \right\rceil \qquad \mu = \left\lceil \frac{\beta}{r-1} \right\rceil$$

Then we define $C_{-\lambda} = -\infty$ and $C_{\mu+1} = \infty$.
For $-\lambda < k \leq \mu$, define C_k such that:

$$kr - (\alpha - \lambda) \leq C_k \leq (k-1)r + \beta - \mu + 1.$$

So we have the constants λ, μ, and C_k, for $-\lambda \leq k \leq \mu + 1$. Then we define $t_i = 0$, and for each position $i \geq 1$, define t_{i+1} to be k if and only if

$$C_k \leq p_i < C_{k+1}.$$

So each t_{i+1} is computed just from the position sum p_i and the pre-computed constants. Note that in the transfer digit selection algorithm, we are assuming that $kr-(\alpha-\lambda) \leq (k-1)r+\beta-\mu+1$ for all $-\lambda < k \leq \mu$, which is a non-trivial statement. The proof of this is rather tedious and arises from the restrictions on r, D, and ρ that were chosen above.

What we will prove is that, assuming that each C_k exists, then the stated algorithm actually does perform carry-free addition. It is clear from the definitions that the final sum formed from the concatenation of the s_i's is in fact the sum of x and y, but what we need to show is that each s_i is actually a valid digit. That is, we need the following lemma:

Lemma 15. *Given the algorithms for carry-free addition and transfer digit selection above, each final sum digit s_i is in the digit set D.*

Proof. From the way the transfer digits are chosen, we know that $-\lambda \leq t_i \leq \mu$ and $C_{t_{i+1}} \leq p_i < C_{t_{i+1}+1}$.

Since each position sum p_i is the sum of two digits in D, they must each satisfy:

$$\begin{array}{rcccl} -2\alpha & \leq & p_i & \leq & 2\beta \\ -\left\lceil \frac{\alpha}{r-1} \right\rceil (r-1) - \alpha & \leq & p_i & < & \left\lceil \frac{\beta}{r-1} \right\rceil (r-1) + \beta + 1 \\ -\lambda(r-1) - \alpha & \leq & p_i & < & \mu(r-1) + \beta + 1 \\ (-\lambda)r - (\alpha - \lambda) & \leq & p_i & < & ((\mu+1)-1)r + \beta - \mu + 1 \end{array}$$

So, instead of $-\infty$ and ∞, we could set $C_{-\lambda} = (-\lambda)r - (\alpha - \lambda)$ and $C_{\mu+1} = ((\mu+1)-1)r + \beta - \mu + 1$, and the algorithm would remain unchanged. So assume for the purposes of this proof that $C_{-\lambda}$ and $C_{\mu+1}$ are so defined.

Then we know that, for each k, $kr - (\alpha - \lambda) \leq C_k \leq (k-1)r + \beta - \mu + 1$.

And since $s_i = w_i + t_i = p_i - rt_{i+1} + t_i$, we have

$$\begin{array}{rcccl} C_{t_{i+1}} - rt_{i+1} - \lambda & \leq & s_i & < & C_{t_{i+1}+1} - rt_{i+1} + \mu \\ t_{i+1}r - (\alpha - \lambda) - rt_{i+1} - \lambda & \leq & s_i & < & t_{i+1}r + \beta - \mu + 1 - rt_{i+1} + \mu \\ -\alpha & \leq & s_i & < & \beta + 1 \\ -\alpha & \leq & s_i & \leq & \beta \end{array}$$

Then, since D contains all integers between $-\alpha$ and β, s_i is in D. $\square$

To see exactly why this algorithm can be run in parallel, simply notice that each sum digit s_i can be totally determined from the four digits x_i, y_i, x_{i-1}, and y_{i-1}. In fact, if each processor shares just one piece of information (the transfer digit), then the processor computing s_i only needs to look at x_i, y_i, and t_i.

In fact this parallelism is the primary advantage of this algorithm. This means that addition can be performed in constant time on a parallel machine, as opposed to addition in standard number systems, which is not parallelizable and takes $O(\log n)$ time.

Also notice that, perhaps contrary to intuition, greater redundancy does not give better performance. A high redundancy index ρ will mean that the values of α and/or β are high, and therefore the range of transfer digits $[-\lambda, \mu]$, will be increased, which makes the transfer digit selection step more complicated. Also, larger digit sets will necessarily require a larger amount of storage. So in general, to make the algorithm most efficient, α and β should be chosen to be as small as possible while still satisfying all the constraints.

According to [6], this algorithm has in fact been implemented at a low level in some specific systems where a large set of numbers must be added together all at once, or for multiplication, as these are where the biggest computational advantages lie.

However, using a redundant number system also poses a few problems. One problem is negation — that is, given a word $\mathbf{w} \in D^*$, with $E_N(\mathbf{w}) = n$, how to find a word $\overline{\mathbf{w}}$ such that $E_N(\overline{\mathbf{w}}) = -n$? Note that if we used the *balanced digit set* where $\alpha = \beta$, then we can just negate each digit to find the additive complement. This is a big part of the reason why the balanced digit set representation seems to be one of the most commonly used ones. If the digit set is not balanced, computing the additive inverse could be more complicated. And since usually subtraction is performed by adding the additive inverse, this means that subtraction could be slower when using redundant digit sets which are not balanced.

The other major problem is comparison: since one number may have multiple representations, it is not immediately obvious how to tell even if two numbers are equal. However, if we can perform subtraction quickly, and α and β are both strictly less than the base r, then we can compare to numbers easily by finding their difference and then examining the sign of the first (i.e. most significant) digit. This is due to the fact that, for any $k > 0$, we know that

$$r^k > r^k - 1 = \sum_{i=0}^{k-1} (r-1) r^k.$$

4 Conclusion and open problems

We can see that redundant number systems have some very fascinating theoretical properties, and they also have some practical use for computing. Specifically, some necessary and some sufficient conditions for a number system to be complete and redundant were examined, giving some characterization of what these number systems look like. We also expanded the notion of automaticity to include redundant number systems, and in fact showed that the new and the traditional definitions are equivalent. Finally, we examined how redundant number systems can facilitate fast, parallel arithmetic.

However, there are still a number of open questions raised here. First, it would be nice to have a relatively simple condition which is necessary *and* sufficient for a number system to be both redundant and complete. Such conditions do exist for certain number systems to be non-redundant and complete, for example basic digit sets (see [5]); however, the class of complete, redundant number systems seems more elusive to describe.

Another open question is whether the new equivalence of N-automaticity for a redundant, complete number system N to the traditional definition of k-automaticity is useful at all. Automatic sequences have been studied a great deal — does this new equivalence allow some new fact to be proved, or does it allow some other proof to be simplified? It seems that the new formulation is different enough that this should be possible, but an example of this has yet to be found.

Finally, it is well-known that redundant number systems can be used to aid certain arithmetic operations, most notably addition and multiplication. However, the tradeoffs inherent in redundant number systems which allow for faster arithmetic are perhaps applicable in different operations as well, for example, to aid in division or discrete logarithm. Also, while redundant digit set arithmetic has been implemented at a very low level for some specific uses, it could perhaps find some use in software as well. One possible example could be infinite-precision integer arithmetic, in which numbers are essentially implemented in base 2^{32} (each digit occupies one word of memory). Such software systems are (expectedly) much slower than standard integer arithmetic, especially for extremely large integers, and so redundancy used to speed up arithmetic could be desirable.

In any case, it should be clear that redundant number systems hold a great deal of interest, both from the theoretical and practical viewpoints, and there is much room for further study of their properties and applications.

References

[1] J.-P. Allouche, E. Cateland, W. J. Gilbert, H.-O. Peitgen, J. O. Shallit, and G. Skordev. Automatic maps in exotic numeration systems. *Theory Comput. Syst.*, 30(3):285–331, 1997.

[2] Jean-Paul Allouche and Jeffrey Shallit. *Automatic sequences.* Cambridge University Press, Cambridge, 2003. Theory, applications, generalizations.

[3] John Colson. A short account of negativo-affirmative arithmetick. In *Philos. Trans. Royal Soc. London*, volume 34. 1726.

[4] Juha Honkala. On number systems with negative digits. *Ann. Acad. Sci. Fenn. Ser. A I Math.*, 14(1):149–156, 1989.

[5] David W. Matula. Radix arithmetic: digital algorithms for computer architecture. In *Applied computation theory: analysis, design, modeling*, pages 374–448. Prentice-Hall, Englewood Cliffs, N.J., 1976.

[6] B. Parhami. Generalized signed-digit number systems: A unifying framework for redundant number representations. *IEEE Transactions on Computers*, 39(1):89–98, Jan. 1990.