ECONOMICAL ENCODING OF COMMAS BETWEEN STRINGS

by

S. Even and M. Rodeh

Technical Report No. 54
July 1975

TECHNION - Israel Institute of Technology
Department of Computer Science

HAIFA - ISRAEL
A method for insertion of delimiters between strings so that the result is uniquely decipherable is presented. As the lengths of the strings increase, the extra-cost, in terms of channel capacity, converges to zero.

Key words: string transmission, channel capacity, commas.
When transmitting a contiguous sequence of strings of arbitrary lengths, one has to insure the ability of parsing the transmitted data into its components. A possible method of achieving this end may be the addition of a new letter, which is used as a comma. As a result, the capacity of the channel is decreased by a constant. We describe a scheme that makes the parsing easy while, for long strings, the decrease in channel capacity tends to zero.

We use the binary alphabet \{0,1\} (the generalization to an arbitrary finite alphabet is straightforward). Let \(a,b,c,\ldots\) denote binary digits; \(\alpha,\beta,\gamma,\ldots\) denote finite strings of binary digits. Let \(\alpha = a_1 a_2 \ldots a_n\). \(\alpha <_{i,j}\) denote the substring \(a_1 \ldots a_j\). \(\ell(\alpha)\) is the length (= number of letters) of \(\alpha\). \(\alpha \beta\) is the concatenation of \(\alpha\) and \(\beta\).

For a positive integer \(n\), \(B(n)\) denotes that binary representation of \(n\), for which the most significant bit is 1. \(L\) is an abbreviation for \(\lceil \log(x+1) \rceil\) where, for binary case, the base of the logarithm is 2, and \(\lfloor x \rfloor\) is the least integer \(\leq x\); \(L^2 = L\) and \(L(L^2) = L(L^2)\).

Define \(R'(n)\) as follows:

\[
R'(n) = \begin{cases} 
  b_1 b_2 b_3 & n \leq 7 \\
  R'(L^2 n) B(n) & n > 7 
\end{cases}
\]

where \(b_1 b_2 b_3\) is a binary representation of \(n\).

Next we define \(R(n)\) by

\[
R(n) = R'(n) 0.
\]

* This problem is not the synchronizability problem [1]
The intuitive meaning of \( R(n) \), for \( n > 7 \), is as follows: on the left of the string generated so far we concatenate the \( R' \) representation of the number of bits used in the previous step. Also, we put an 0 at the right end of \( R \).

**Example** (blanks added for clarity)

\[
R'(2761) = 100 \ 1100 \ 101011001001
\]

\[
R(2761) = 100 \ 1100 \ 101011001001 \ 0
\]

**Lemma 1:** For \( n \geq 4 \) there exists an \( i \) such that \( \log^i n = 3 \).

**Proof** By induction. For \( n = 4, 5, 6, 7 \) a direct computation yields

\[
\log n = \left\lfloor \log (n+1) \right\rfloor = 3.
\]

Assuming the correctness of the Lemma for \( 4 \leq n < 2^m \) and \( 3 \leq m \), we shall prove its correctness for \( 2^m \leq n < 2^{m+1} \).

\[
\log n = \left\lfloor \log (n+1) \right\rfloor = \log (2^m+1) = m+1.
\]

But \( 4 \leq m+1 < 2^m \). Thus, by the induction hypothesis, there exists a \( j \) such that

\[
\log^j (m+1) = 3.
\]

Thus, \( \log^{j+1} n = 3 \).

**Q. E. D.**

**Corollary 1:** \( \ell(R'(n)) = \sum_{i=1}^{\log^i n = 3} \log^i n \).

Next, we shall find an upper bound for \( \ell(R(n)) \).

**Lemma 2:** For \( n \geq 4 \) \( \ell(R'(n)) \leq 2 \log n \).

**Proof:** By induction. First we shall check directly for two domains. For \( 4 \leq n < 2^j \):

\[
\ell(R'(n)) \leq \log n + \log^2 (2^j-1) = \log n + 3,
\]
where the inequality follows from the monotonicity of LG. For 
\( n \geq 4, \ L_n \geq 3 \). Thus
\[ \ell(R'(n)) \leq 2 \ L_n. \]

For \( 2^7 \leq n \leq 2^{10} \) we have
\[ \ell(R'(n)) = L_n + L_n^2 + L_n^3 \]
\[ \leq L_n + L_n^2 (2^{10}) + L_n^3 (2^{10}) \]
\[ \leq L_n + 7 \]
\[ \leq 2 \ L_n. \]

Now we proceed by induction. Assume that we have proven the Lemma
for \( n < m, m > 2^{10} \). For \( m \leq n < 2^{m-1} \) we have
\[ \ell(R'(n)) = L_n + \sum_{i=2}^{L_n^3} L_n^i \]
\[ = L_n + \sum_{i=1}^{L_n^i(L_n^3)} L_n^i(L_n^3) \]
\[ = L_n + \ell(R'(L_n^3)). \]

But \( 4 < L_n \leq L (2^{m-1} - 1) = m - 1 < m \).
Thus, by the induction hypothesis
\[ \ell(R'(n)) \leq L_n + 2 \cdot L_n^2. \quad (*) \]

Observing that for \( k \geq 10 \)
\[ 2 \cdot \log(k+1) \leq k - 2 , \]
we get that \( 2 \cdot (\log(k+1)+1) \leq k \).

Thus \( 2 \cdot \lceil \log(k+1) \rceil \leq k \),
and we conclude that for \( k \geq 10 \)
\[ 2 \ L_n \leq k. \]
Therefore, for \( n \geq 2^{10} \)
\[ 2 \log^2 n \leq \log n. \]
Substituting in (*) we get
\[ \ell(R'(n)) \leq 2 \log n. \]

Q. E. D.

**Corollary 2**  
\[ \ell(R(n)) \leq 2 \log n + 1 \]

Let \( \alpha_1, \ldots, \alpha_m \) be a sequence of strings that we wish to transmit contiguously. Denote \( \ell(\alpha_i) = n_i \). The string, for transmission is

\[ \sigma = \sigma(\alpha_1, \ldots, \alpha_m) = R(n_1)\alpha_1 R(n_2)\alpha_2 \ldots R(n_m)\alpha_m 000. \]

To parse \( \sigma \) we iterate on \( i = \) isolate \( R(n_i) \) and decode it to get \( n_i \) itself; then use \( n_i \) to identify \( \alpha_i \). We stop when \( n_i = 0 \). It is obvious that the crucial problem is the proper handling of \( R(n_i) \). The following algorithm implements a method for proceeding along \( \sigma \) and halting on the last character of \( R(n_i) \) giving \( n_i \) as a result.

i) (Initialization)
\[
\begin{align*}
& i \leftarrow 1; \\
& j \leftarrow 3; \\
& n_i \leftarrow \sigma < 1,3 >;
\end{align*}
\]
(we consider \( n_i \) as a number whose binary representation is \( \sigma < 1,3 > \).)

\[
\text{if } n_i = 0 \text{ then stop ;}
\]

ii) (Iterate)
\[
\begin{align*}
& \text{while } \sigma < j+1 > = 1 \text{ do} \\
& \quad (i \leftarrow j+1 ; \\
& \quad j \leftarrow j+n_i ; \\
& \quad n_i \leftarrow \sigma < i,j > ) ; \\
& \text{stop ;}
\end{align*}
\]
Example: Let $\beta$ be a string, $L(\beta) = 2761$. Let us apply algorithm to

$$\sigma = R(2761)\beta 000.$$ Writing $\sigma$ in more detail:

$$\sigma = 1001100101011001001000.$$ We see that

$$i = 1;$$
$$j = 3;$$
$$n_1 = 4;$$

Then

$$\sigma^{<4>} = 1$$ thus

$$i = 4;$$
$$j = 7;$$
$$n_1 = 12;$$

Then

$$\sigma^{<8>} = 1$$ thus

$$i = 8;$$
$$j = 19;$$
$$n_1 = 2761;$$

Since

$$\sigma^{<20>} = 0$$ the algorithm stops.

The algorithm has computed $n_1$. Using it we easily identify $\beta$ as

$$\beta = \sigma^{<j+2,j+n_1+1>}.$$ Notice that the algorithm stops on the last character of $R(n)$ without checking the next letter. The reason for this is that $B(n)$ is a representation of $n$ whose most significant bit is 1. Therefore, the first 0 we meet is the 0 appearing explicitly in the definition of $R(n) = R'(n)0$.

**THEOREM** If $m$ is a fixed positive integer then

$$\lim_{n \to \infty} \frac{L(\sigma(\alpha_1, \ldots, \alpha_m))}{\Omega(\alpha_1, \ldots, \alpha_m)} = 1$$
Proof

\[ \ell(\sigma(\alpha_1, \ldots, \alpha_m)) = \sum_{i=1}^{m} n_i \]

\[ 3 + \sum_{i=1}^{m} n_i + \sum_{i=1}^{m} \ell(R(n_i)) \leq \sum_{i=1}^{m} n_i \]

By corollary 2

\[ 3 + 2 \sum_{i=1}^{m} \log_2(n_i) + m \]

\[ + 1 \]

Thus, the limit is 1.

Notice that the convergence to 1 is rapid. For example, for strings of length \(2^{10}\) bits, the overhead is less than 2%.

Reference: