ECONOMICAL ENCODING OF COMMAS BETWEEN STRINGS

by

S. Even and M. Rodeh

Technical Report No. 54
July 1975
ABSTRACT

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,... denote binary digits; α,β,γ,... denote finite strings of binary digits. Let α = a_1 a_2 ... a_n. α <i,j> denote the substring a_i ... a_j. ℓ(α) is the length (= number of letters) of α. αβ is the concatenation of α and β.

For a positive integer n, B(n) denotes that binary representation of n, for which the most significant bit is 1. LGx 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 ≥ x; LG^2x = LG(LGx): LG^{i+1}x = LG(LG^i x).

Define R'(n) as follows:

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

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

$\log n = \lceil \log (n+1) \rceil = 3$.

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

$\log n = \lceil \log (n+1) \rceil$

$= \log (2^{m+1}) = m + 1$.

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 $n \geq 4 \log(R'(n)) \leq \log n$.

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

$\ell(R'(n)) \leq \log n + \log^2 \left(2^{2^j} - 1 \right) = \log n + 3$,
where the inequality follows from the monotonicity of LG. For \( n \geq 4 \), \( LGn \geq 3 \). Thus
\[
\ell(R'(n)) \leq 2 \cdot LGn.
\]
For \( 2^7 \leq n \leq 2^{10} \) we have
\[
\ell(R'(n)) = LGn + LG^2n + LG^3n \\
\leq LGn + LG^2 (2^{10}) + LG^3 (2^{10}) \\
\leq LGn + 7 \\
\leq 2 \cdot LGn.
\]
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
\[
\ell(R'(n)) = LGn + \sum_{i=2}^{LGn=3} LG^i n \\
= LGn + \sum_{i=1}^{LG(LGn)=3} LG^i (LGn) \\
= LGn + \ell(R'(LGn)).
\]
But \( 4 \leq LGn \leq LG (2^{m-1}-1) = m - 1 < m \).
Thus, by the induction hypothesis
\[
\ell(R'(n)) \leq LGn + 2 \cdot LG^2n . \quad (*)
\]
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 \( k \geq 10 \)
\[
2 \cdot LGk \leq k .
\]
Therefore, for \( n \geq 2^{10} \)
\[ 2 \log^2 n \leq \log n. \]

Substituting in (*) we get
\[ l(R'(n)) \leq 2 \log n. \]

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

\[ \sigma = \sigma(\alpha_1, \ldots, \alpha_m) = R(n_1)\alpha_1R(n_2)\alpha_2 \cdots R(n_m)\alpha_m 000. \]

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.

\begin{enumerate}
  \item \textbf{(Initialization)} \quad \( i \leftarrow 1; \)
      \[ j \leftarrow 1; \]
      \[ n_i \leftarrow 0 \]
      \( (j, n_i) ; \)
      \( \sigma < 1, 3 > ; \)
  \end{enumerate}

(we consider \( n_i \) as a number whose binary representation is \( \sigma < 1, 3 > \).

\hspace{1cm} \textbf{if} \quad n_i = 0 \quad \textbf{then} \quad \textbf{stop} ;

\begin{enumerate}
  \item \textbf{(Iterate)} \quad \textbf{while} \quad \sigma < j+1 > = 1 \quad \textbf{do}
    \begin{enumerate}
    \item \( i \leftarrow j+1 \)
    \item \( j \leftarrow j+n_i \)
    \item \( n_i \leftarrow \sigma < 1, j > \)
    \end{enumerate}
    \textbf{stop};
\end{enumerate}
Example: Let $\beta$ be a string, $\ell(\beta) = 2761$. Let us apply algorithm to $\sigma = R(2761)\beta000$. Writing $\sigma$ in more detail:

$\sigma = 10011001010110010010000$

We see that

\begin{align*}
i & = 1; \\
j & = 3; \\
n_1 & = 4;
\end{align*}

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

\begin{align*}
i & = 4; \\
j & = 7; \\
n_1 & = 12;
\end{align*}

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

\begin{align*}
i & = 8; \\
j & = 19; \\
n_1 & = 2761;
\end{align*}

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

$$\lim_{i=1\atop n \to \infty} \frac{\ell(\sigma(\alpha_1, \ldots, \alpha_m))}{\prod_{i=1}^{n} i} = 1$$
Proof

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

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

By corollary 2

$$3 + 2 \frac{\sum_{i=1}^{m} \log(n_i)}{\sum_{i=1}^{m} n_i} + m + 1$$

Q.E.D.

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: