Universal Lossless Data Compression
Via Binary Decision Diagrams

J. Kieffer, P. Flajolet, and E-h. Yang
Abstract

A binary string of length $2^k$ induces the Boolean function of $k$ variables whose Shannon expansion is the given binary string. This Boolean function then is representable via a unique reduced ordered binary decision diagram (ROBDD). The given binary string is fully recoverable from this ROBDD. We exhibit a lossless data compression algorithm in which a binary string of length a power of two is compressed via compression of the ROBDD associated to it as described above. We show that when binary strings of length $n$ a power of two are compressed via this algorithm, the maximal pointwise redundancy/sample with respect to any $s$-state binary information source has the upper bound $(4 \log_2 s + 16 + o(1)) / \log_2 n$. To establish this result, we exploit a result of Liaw and Lin stating that the ROBDD representation of a Boolean function of $k$ variables contains a number of vertices on the order of $(2 + o(1))2^{k/k}$.

Index Terms: lossless source coding, universal codes, Boolean functions, ROBDD representations
I Introduction

Let $S$(dyadic) denote the set of all binary strings $x$ such that

- The length of $x$ is a power of two.
- The substring of $x$ which forms the left half of $x$ does not coincide with the substring of $x$ which forms the right half of $x$.
- $x$ contains at least one entry of 1 and at least one entry of 0.

Let $x \in S$(dyadic) and let $k$ be the logarithm to the base two of the length of $x$. In a natural way, $x$ induces a Boolean function $f_x$ of $k$ variables. The function $f_x$ maps the set $\{0, 1\}^k$ into the set $\{0, 1\}$, and can be defined as follows. Let $u_1, u_2, \ldots, u_{2^k}$ be the lexicographical ordering of all binary strings of length $k$. For each $i = 1, 2, \ldots, 2^k$, define $f_x(u_i)$ to be the $i$-th coordinate of $x$. Following [1] [2], the Boolean function $f_x$ can be represented by a directed acyclic graph called a reduced ordered binary decision diagram (ROBDD). Since the Boolean function $f_x$ can be recovered from its ROBDD representation, $x$ can also be recovered from this representation. This means that we can losslessly compress a string $x \in S$(dyadic) by compressing the ROBDD representation of the Boolean function $f_x$ induced by $x$. It is the purpose of this note to investigate the compression performance that is achievable by such a compression algorithm (we obtain a redundancy bound).

There are public domain software packages (some of them on the Internet) for computing the ROBDD representation of a Boolean function. Such a package would be easily adaptable in order to provide compression of a data string in $S$(dyadic) according to the ROBDD-based compression algorithm that we shall present.

II ROBDD’s Representing Data Strings

Define $G$ to be the set of all finite graphs $G$ such that

(a) $G$ is directed and acyclic.

(b) $G$ has a unique nonterminal vertex $V_G^r$ such that for every other vertex $V$ of $G$, there is at least one directed path leading from $V_G^r$ to $V$. The vertex $V_G^r$ is called the root vertex of $G$.

(c) There are exactly two terminal vertices of $G$, which shall be denoted $T_G^0$ and $T_G^1$, respectively.

(d) From each nonterminal vertex of $G$, there emanate exactly two edges, one of which is labelled “0” and the other of which is labelled “1”. These edges terminate at different vertices (i.e., $G$ has no multiple edges).

(e) Each vertex $V$ of $G$ carries a positive integer label $L(V)$ which we shall call the level of $V$. The levels of the vertices satisfy the properties:

- $L(V_G^r) = 1$. 

obtained by concatenating together $j$ as indicated in Figure 2. (This is a “canonical ordering” of the vertices of $G$ explained later.) Since $y$ denotes the binary string obtained by concatenating $y$ strings of finite positive length. We define $\phi_G$ to satisfy the properties (a)-(e). Therefore, this graph is a member of $\mathcal{G}$.

Example 1. The graph given in Figure 1, in which each vertex is labelled by its level, is seen to satisfy the properties (a)-(e). Therefore, this graph is a member of $\mathcal{G}$.

Let $G \in \mathcal{G}$. Let $V(G)$ be the set of vertices of $G$. Let $\{0, 1\}^+$ denote the set of all binary strings of finite positive length. We define $\phi_G$ to be the unique mapping from $V(G)$ into $\{0, 1\}^+$ such that

- $\phi_G(T_G^0) = 0$ and $\phi_G(T_G^1) = 1$.
- If $V$ is a nonterminal vertex of $G$, if the edge labelled 0 emanating from $V$ terminates at vertex $V_0$, and the edge labelled 1 emanating from $V$ terminates at vertex $V_1$, then
  \[
  \phi_G(V) = \phi_G(V_0)^{(2^{L(V_0)}-L(V)}-1) \phi_G(V_1)^{(2^{L(V_1)}-L(V)}-1)
  \]
  (Notation: If $y$ is a binary string and $j$ is a positive integer, then $y^j$ denotes the binary string obtained by concatenating together $j$ copies of $y$. If $y_1$ and $y_2$ are binary strings, then $y_1y_2$ denotes the binary string obtained by concatenating $y_2$ onto the right end of $y_1$.)

Example 2. Let $A_1, A_2, \ldots, A_{16}$ denote the sixteen vertices of the graph $G$ in Figure 1, as indicated in Figure 2. (This is a “canonical ordering” of the vertices of $G$, which shall be explained later.) Since $A_8 = T_G^0$ and $A_{16} = T_G^1$, we have

\[
\begin{align*}
\phi_G(A_8) &= 0 \\
\phi_G(A_{16}) &= 1 \\
\phi_G(A_9) &= \phi_G(A_8)\phi_G(A_{16}) = 01 \\
\phi_G(A_{10}) &= \phi_G(A_8)^2\phi_G(A_{16})^2 = 0011 \\
\phi_G(A_{11}) &= \phi_G(A_9)\phi_G(A_{10})^2 = 0111 \\
\phi_G(A_{12}) &= \phi_G(A_8)^4\phi_G(A_{16})^4 = 00001111 \\
\phi_G(A_{13}) &= \phi_G(A_9)^2\phi_G(A_{16})^4 = 01011111 \\
\phi_G(A_{14}) &= \phi_G(A_{10})\phi_G(A_{16})^4 = 00111111 \\
\phi_G(A_{15}) &= \phi_G(A_{11})\phi_G(A_{16})^4 = 01111111 \\
\phi_G(A_{14}) &= \phi_G(A_8)^8\phi_G(A_9)^4 = 000000001010101 \\
\phi_G(A_{15}) &= \phi_G(A_{10})^2\phi_G(A_{11})^2 = 0011001101110111 \\
\phi_G(A_{16}) &= \phi_G(A_{12})\phi_G(A_{13}) = 0000111101011111 \\
\phi_G(A_{17}) &= \phi_G(A_{14})\phi_G(A_{15}) = 0011111101111111 \\
\phi_G(A_{18}) &= \phi_G(A_4)\phi_G(A_{5}) = \text{length 32 string} \\
\phi_G(A_{19}) &= \phi_G(A_6)\phi_G(A_{7}) = \text{length 32 string} \\
\phi_G(A_{11}) &= \phi_G(A_2)\phi_G(A_{3}) = \text{length 64 string}
\end{align*}
\]

The following is clear from the definition of $\phi_G$ and Example 2.
Lemma 1 Let $G$ be any graph in $\mathcal{G}$. Suppose $L(T_0^G) = L(T_1^G) = k + 1$. Then, for each vertex $V$ of $G$, the length of $\phi_G(V)$ is $2^{k+1-L(V)}$. In particular, the length of $\phi_G(V_r^G)$ is $2^k$.

**Definition.** We define $\mathcal{G}^*$ to be the set of all graphs $G \in \mathcal{G}$ such that the mapping $\phi_G$ is one-to-one.

Lemma 2 The following statements hold:

(a) For any $G \in \mathcal{G}^*$, the binary string $\phi_G(V_r^G)$ is a member of $S$(dyadic).

(b) For each $x \in S$(dyadic), there is a unique $G \in \mathcal{G}^*$ such that $\phi_G(V_r^G) = x$. In the language of [1] [2], this unique graph $G$ is the unique ROBDD representing the Boolean function $f_x$.

**Proof.** Part (a) is clear from Example 2. Part (b) (including the uniqueness of the ROBDD representation) may be seen to be true by consulting the papers [1] [2].

**Notation.** For each $x \in S$(dyadic), we let $G_x$ denote the unique graph in $\mathcal{G}^*$ which represents $x$ in the sense of Lemma 2(b).

**Example 3.** The graph $G$ in Figure 1 is $G_x$, where $x \in S$(dyadic) is found from Example 2 by the calculation

$$
x = \phi_G(A_4)\phi_G(A_5)\phi_G(A_6)\phi_G(A_7)
= 00000000101010100110011011011101110000111101011111101111110111111111
$$

### III Encoding Method

For each $G \in \mathcal{G}^*$, we shall define in this section a binary codeword $\sigma(G)$ from which $G$ can be recovered. Given $x \in S$(dyadic), we can then losslessly encode $x$ into the binary codeword $\sigma(G_x)$.

We need the following notation. If $G$ is a graph in $\mathcal{G}^*$, and $V$ is a nonterminal vertex of $G$, then the notation

$$V \rightarrow V_0, V_1$$

means that $V_0$ is the vertex of $G$ to which edge 0 from $V$ leads, and $V_1$ is the vertex of $G$ to which edge 1 from $V$ leads.

Fix $G \in \mathcal{G}^*$, and let $j$ be the number of vertices of $G$. We define a canonical ordering of the vertices of $G$. Let $A_1, A_2, \ldots, A_j$ be the enumeration of the vertices of $G$ which is uniquely determined by the two properties

**Property(i):** $A_1 = V^r$

**Property(ii):** If $q_1 < q_2 < \ldots < q_{j-2}$ are the integers in $\{1, 2, \ldots, j\}$ such that $A_{q_1}, A_{q_2}, \ldots, A_{q_{j-2}}$ are the nonterminal vertices of $G$, and if we write

$$
\begin{align*}
A_{q_1} & \rightarrow A_{r_1}, A_{s_1} \\
A_{q_2} & \rightarrow A_{r_2}, A_{s_2} \\
& \quad \ldots \\
A_{q_{j-2}} & \rightarrow A_{r_{j-2}}, A_{s_{j-2}}
\end{align*}
$$
then, if we list the distinct entries of the sequence

\[(A_{r_1}, A_{s_1}, A_{r_2}, A_{s_2}, \ldots, A_{r_j-2}, A_{s_j-2})\]

in order of their first left-to-right appearances in this sequence, we get the list \(A_2, A_3, \ldots, A_j\).

**Example 4.** The canonical ordering of the vertices of the graph \(G\) in Figure 1 is given in Figure 2. We can determine this ordering by generating the following relations one by one:

\[
\begin{align*}
A_1 & \rightarrow A_2, A_3 \\
A_2 & \rightarrow A_4, A_5 \\
A_3 & \rightarrow A_6, A_7 \\
A_4 & \rightarrow A_8, A_9 \\
A_5 & \rightarrow A_{10}, A_{11} \\
A_6 & \rightarrow A_{12}, A_{13} \\
A_7 & \rightarrow A_{14}, A_{15} \\
A_9 & \rightarrow A_8, A_{16} \\
A_{10} & \rightarrow A_8, A_{16} \\
A_{11} & \rightarrow A_9, A_{16} \\
A_{12} & \rightarrow A_8, A_{16} \\
A_{13} & \rightarrow A_9, A_{16} \\
A_{14} & \rightarrow A_{10}, A_{16} \\
A_{15} & \rightarrow A_{11}, A_{16}
\end{align*}
\]  

(3.1)

Notice that in (3.1), vertices \(A_8\) and \(A_{16}\) are missing from the left hand sides. This means that \(A_8\) and \(A_{16}\) are the terminal vertices of the graph in Figure 2. One of these vertices is equal to \(T^0_G\) and the other is equal to \(T^1_G\). We cannot determine which is the case from (3.1) alone. We would need an extra bit of information to determine which of the two possibilities

\[
\begin{align*}
A_8 &= T^0_G & A_{16} &= T^1_G \\
A_8 &= T^1_G & A_{16} &= T^0_G
\end{align*}
\]

holds.

Let \(G \in \mathcal{G}^*\), let \(k\) be the positive integer such that \(L(T^0_G) = L(T^1_G) = k + 1\), and let \(A_1, A_2, \ldots, A_j\) be the canonical ordering of the vertices of \(G\). We will generate strings \(S_1, S_2, \ldots, S_{k+1}\) in which

- \(S_1 = A_1\), and each entry of each \(S_i\) is a member of the set of symbols
  \[
  \{A^q_m : m = 1, 2, \ldots, j, \ q = 1, 2, \ldots\}
  \]
- The strings \(S_1, S_2, \ldots, S_{k+1}\), taken together, allow one to build the graph \(G\) (except for the determination of which of the two terminal vertices equals \(T^0_G\), and which equals \(T^1_G\), which takes one more bit of information, as discussed above).
• Each $S_i$ ($i \geq 2$) is generated recursively from $S_{i-1}$ and certain side information, and the side information from each recursive step is what is encoded to form the overall codeword $\sigma(G)$. From $\sigma(G)$, the decoder can then recursively generate the $\{S_i\}$, from which $G$ is obtained.

Fix $i$, where $2 \leq i \leq k+1$. The following procedure describes how $S_i$ is recursively generated from $S_{i-1}$:

**Step(i):** Write down the string $U$ consisting of the first appearances (from left to right) of each distinct symbol appearing in $S_{i-1}$.

**Step(ii):** For each entry of $U$ of form $A_{m}^q$, where $q > 1$, write below that entry the entry $A_{m}^{q-1}$.

**Step(iii):** For each entry of $U$ of form $A_{m}$, write down below that entry the two entries $A_{m_0}^q, A_{m_1}^q$, where $m_0, m_1$ are the respective vertices to which edges 0 and 1 from $A_m$ lead, and $q_0$ and $q_1$ are the positive integers

$$
q_0 = L(A_m) - L(A_{m_0}) \\
q_1 = L(A_m) - L(A_{m_1})
$$

**Step(iv):** Concatenate together the sequence of entries written below the entries of $U$ in Steps (ii) and (iii). The resulting sequence is $S_i$.

**Example 5.** For the graph $G$ in Figure 2, the strings $S_1, S_2, \ldots, S_7$ are as follows:

$$
S_1 = A_1 \\
S_2 = (A_2, A_3) \\
S_3 = (A_4, A_5, A_6, A_7) \\
S_4 = (A_8^4, A_9^3, A_{10}^2, A_{11}^2, A_{12}, A_{13}, A_{14}, A_{15}) \\
S_5 = (A_8^3, A_9^2, A_{10}, A_{11}, A_3, A_{16}, A_2^3, A_{16}^3, A_{16}^2, A_{16}, A_{16}^3, A_{16}^2) \\
S_6 = (A_8^2, A_9, A_{16}, A_{16}^2, A_2, A_{16}, A_{16}) \\
S_7 = (A_8, A_9, A_{16}, A_{16})
$$

Let $G \in G^*$, let $k$ be the positive integer such that $L(T_{k+1}^0) = k + 1$, and let $A_1, A_2, \ldots, A_j$ be the canonical ordering of the vertices of $G$. One easily determines from $S_1, S_2, \ldots, S_{k+1}$ the level of each vertex $A_1, A_2, \ldots, A_j$. For each $A_i$, find the unique $S_m$ such that $A_i$ is an entry of $S_m$. Then, $L(A_i) = m$. To illustrate, from $S_1, S_2, \ldots, S_7$ in Example 5, we determine that

$$
L(A_1) = 1 \quad L(A_2) = 2 \quad L(A_3) = 2 \quad L(A_4) = 3 \quad L(A_5) = 3 \quad L(A_6) = 3 \quad L(A_7) = 3 \quad L(A_8) = 7 \quad L(A_9) = 6 \quad L(A_{10}) = 5 \quad L(A_{11}) = 5 \quad L(A_{12}) = 4 \quad L(A_{13}) = 4 \quad L(A_{14}) = 4 \quad L(A_{15}) = 4 \quad L(A_{16}) = 7
$$

Referring to Figure 2, we see that this assignment is correct.
One also easily determines from \( S_1, \ldots, S_{k+1} \) where each edge of \( G \) begins and ends. For each nonterminal vertex \( A_i \), find the unique \( m < k+1 \) such that \( A_i \) is an entry of \( S_m \), and then look below in \( S_{m+1} \) to find the corresponding two consecutive entries \( A_{i_0}^0, A_{i_1}^0 \)—vertices \( A_{i_0} \) and \( A_{i_1} \) are then the respective vertices at which edges 0 and 1 from \( A_i \) terminate. To illustrate, from \( S_1, S_2, \ldots, S_7 \) in Example 5, we get the edge description given in (3.1), which we see is correct by referring to Figure 2.

For a graph \( G \in G^* \) such that \( L(T_G^0) = L(T_G^1) = k+1 \), we suppose that the strings \( S_1, S_2, \ldots, S_{k+1} \) have been generated. We now describe how these strings are encoded for transmission to the decoder. The decoder already knows that \( S_1 = A_1 \). In addition to this, the decoder needs to know:

(a) How to obtain \( S_i \) from \( S_{i-1} \), for each \( i = 2, \ldots, k+1 \). This information is transmitted to the decoder using \( M_i \) codebits. In the sequel, we shall explain what these \( M_i \) codebits consist of.

(b) For the two symbols \( A_{j_1} \) and \( A_{j_2} \) comprising the entries of \( S_{k+1} \), the decoder needs to know which of these symbols equals \( T_G^0 \). This information is transmitted to the decoder using one codebit.

From the above description, we see that a total of \( (M_2 + \ldots + M_{k+1}) + 1 \) codebits is transmitted to the decoder by the encoder. We need to further explicate Step (a) above, so that it is understood what \( M_i \) is. To do this, we need a number of definitions.

**Definition 1.** If \( u = (u_1, u_2, \ldots, u_J) \) is any nonempty sequence of finite length over any alphabet \( A \), we define

\[
H(u) \triangleq \sum_{j=1}^{J} - \log_2 \frac{n(u_j)}{J},
\]

where, for each \( a \in A \), \( n(a) \) is the number of \( 1 \leq j \leq J \) for which \( u_j = a \). If \( u \) is an empty sequence, we define \( H(u) = 0 \). The quantity \( H(u) \) is important for the following reason: If the set \( \{u_1, u_2, \ldots, u_J\} \) is known, and if the frequencies with which the symbols in this set appear in \( u \) are known, the sequence \( u \) can be losslessly encoded using \( \lceil H(u) \rceil \) codebits. This is because there are no more than \( 2^{H(u)} \) sequences having the known symbol frequencies.

**Definition 2.** If \( u \) is a sequence of finite length, \( |u| \) denotes the length of \( u \).

**Definition 3.** Let \( u \) be any nonempty sequence of finite length over any alphabet. We define \( \tilde{u} \) to be the (possibly empty) sequence obtained from \( u \) by striking out each term of \( u \) which is making its first left-to-right appearance in \( u \). For example, if

\[
u = (a, a, b, a, b, c, b, c, a),
\]

we strike out the first, third, and sixth terms, obtaining

\[
\tilde{u} = (a, a, b, b, c, a)
\]

It could be that \( u \) is empty. In this case, we define \( \tilde{u} \) to be the empty sequence.

**Definition 4.** If \( u \) is a sequence of finite length such that \( H(\tilde{u}) > 0 \), we define \( h(u) = |u| + H(\tilde{u}) \). If \( u \) is a sequence of finite length such that \( H(\tilde{u}) = 0 \), we define \( h(u) = 0 \). Here is
why the quantity $h(u)$ is important: If the frequencies with which the symbols appearing in $u$ are known, and if the list of these symbols in order of first left-to-right appearance in $u$ is known, then the sequence $u$ can be losslessly encoded using $\ceil{h(u)}$ codebits. To see this, one can encode $\tilde{u}$ using $\ceil{H(\tilde{u})}$ codebits. Then, one can obtain $u$ from $\tilde{u}$ with an additional $|u|$ codebits (these additional codebits tell the decoder the positions in $u$ where the first left-to-right appearances of the symbols in $u$ occur). This gives us a total of $|u| + \ceil{H(\tilde{u})} = \ceil{h(u)}$ codebits. (We have assumed $H(\tilde{u}) = 0$. The reader can treat the case $H(\tilde{u}) = 0$ separately.) For example, if $u$ is the sequence in (3.2), the additional $|u|$ codebits are $(1, 0, 1, 0, 0, 1, 0, 0, 0, 0)$, the ones indicating first appearances of $a, b, c$ in $u$ in positions 1, 3, 6, respectively.

**Definition 5.** For each $2 \leq i \leq k + 1$, we let $\hat{S}_i$ be the subsequence of $S_i$ that arises from substituting for the distinct entries of $S_{i-1}$ of form $A_m$. (Recall that each such entry of $S_{i-1}$ generates two entries of $S_i$.)

**Definition 6.** An entry of $\hat{S}_i$ of form $A_{m}^q$, where $A_{m}^{q+1}$ appears in $S_{i-1}$, shall be called a Type I entry of $\hat{S}_i$. We let $\pi_1^i$ denote the subsequence of $\hat{S}_i$ consisting of all the Type I entries of $\hat{S}_i$.

**Definition 7.** An entry of $\hat{S}_i$ of form $A_{m}^q$, where the symbol $A_m$ does not appear in $S_{i-1}$, shall be called a Type II entry of $\hat{S}_i$. We let $\pi_2^i$ denote the subsequence of $\hat{S}_i$ consisting of all the Type II entries of $\hat{S}_i$. Suppose that there are $r$ distinct entries of $\pi_2^i$, and that $A_m$ is the vertex of highest index $m$ that has appeared in the sequences $S_1, S_2, \ldots, S_{i-1}$. Then, if we list the distinct entries of $\pi_2^i$ in order of their first left-to-right appearances in $\pi_2^i$, this list will take the form

$$A_{m+1}^{q_1}, A_{m+2}^{q_2}, \ldots, A_{m+r}^{q_r}$$

(3.3)

**Definition 8.** We let $Q_i$ be the nonnegative integer consisting of the sum of all the powers $q$ as $A_m^q$ ranges through all of the distinct terms of $\pi_2^i$. (In other words, referring to (3.3), $Q_i$ is equal to $q_1 + q_2 + \ldots + q_r$.)

With the above definitions, we can now stipulate that

$$M_i = |S_i| + |\hat{S}_i| + Q_i + \ceil{H(\pi_1^i)} + \ceil{H(\pi_2^i)},$$

(3.4)

Here is how the different terms in $M_i$ arise:

(a.1) Encoder transmits to decoder $|S_i|$ codebits to let the decoder know the frequency with which each distinct element of $S_i$ appears.

(a.2) Encoder transmits to decoder $|\hat{S}_i|$ codebits so that the decoder will know which entries of $\hat{S}_i$ are of Type I and which entries are of Type II.

(a.3) Encoder transmits to decoder $Q_i$ codebits so that the decoder will know the powers $q$ appearing in the Type II entries $A_m^q$ of $\hat{S}_i$.

(a.4) The encoder transmits to the decoder $\ceil{H(\pi_1^i)}$ codebits, which tell the decoder what $\pi_1^i$ is.

(a.5) The encoder transmits to the decoder $\ceil{h(\pi_2^i)}$ codebits, which tell the decoder what $\pi_2^i$ is.
Definition. We let $\sigma(G)$ be the binary codeword of length $(M_2 + \ldots + M_{k+1}) + 1$ obtained by concatenating together the codebits from Steps (a.1)-(a.5), (b) above.

Example 6. We explain how the decoder can obtain $S_5$ from $S_4$ in Example 5. Initially, the decoder will know that $S_5$ takes the form

$$S_5 = (A_3^8, A_6^2, A_{10}, A_{11}, \hat{S}_5),$$

where the entries of $\hat{S}_5$ have to be filled in. The decoder knows that the length of $S_5$ is 12. The decoder looks at the first 12 codebits that are currently in its codebit buffer, to determine the frequencies of the distinct entries of $S_5$. In this case, these 12 codebits are

$$0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1$$

which tell the decoder that $A_3^8$ appears twice in $S_5$, $A_6^2$ appears twice in $S_5$, $A_{10}$ appears twice in $S_5$, $A_{11}$ appears twice in $S_5$, and an element of form $A_q^q$, with $q$ unknown, appears four times in $S_5$. The decoder now knows that $\pi_1^1$ is of length one and consists of one appearance of each of the symbols $A_3^8, A_6^2, A_{10}, A_{11}$, and that $\pi_2^2$ is of length four and consists of four appearances of the symbol $A_q^q$. The next $|\hat{S}_5| = 8$ codebits in the decoder’s buffer tell the decoder which entries of $\hat{S}_5$ are of Type I and which are of Type II. In this case, these codebits are

$$(0, 1, 0, 1, 0, 1, 0, 1),$$

which tell the decoder that the entries of $\hat{S}_5$ alternate between Type I entries and Type II entries, starting with a Type I entry. The decoder now needs to determine the power $q$ in the symbol $A_q^q$. In this case, $q = 3$, and the decoder will know this because the codebits

$$(0, 0, 1)$$

will appear at the start of the decoder’s codebit buffer at this point. The next $|H(A_3^3, A_6^2, A_{10}, A_{11})| = 8$ codebits tell the decoder that

$$\pi_1^1 = (A_3^3, A_6^2, A_{10}, A_{11})$$

The decoder already knows that

$$\pi_2^2 = (A_{16}^3, A_{16}^3, A_{16}^3, A_{16}^3),$$

so that, putting $\pi_1^1$ and $\pi_2^2$ together, the decoder has determined that

$$\hat{S}_5 = (A_3^3, A_{16}^3, A_6^2, A_{16}^3, A_{16}^3, A_{16}^3, A_{16}^3, A_{16}^3)$$

IV Performance Bound

Let $G \in G^*$, and let $L(T_G^0) = k+1$. The binary codeword $\sigma(G)$ results by encoding the sequences $S_2, S_3, \ldots, S_{k+1}$, plus the transmission of an extra codebit to signal the decoder which of the two terminal vertices of $G$ is equal to $T_G^0$. In this section, we want to upper bound the codeword length $|\sigma(G)|$, in order to see how good the encoder is.
From the previous section, it can be seen that

\[ |\sigma(G)| \leq 4|S_1| + |S_2| + \ldots + |S_{k+1}| + \sum_{i=2}^{k+1} \left[ |H(\pi_i^1)| + |H(\pi_i^2)| \right] \]  

(4.5)

The only tricky part in obtaining this bound is the observation that

\[ \sum_{i=2}^{k+1} Q_i \leq |S_2| + |S_3| + \ldots + |S_{k+1}| \]

To see this, notice that if a Type II symbol \( A_m^q \) appears in a sequence \( S_i \), and \( q > 1 \), then the \( q-1 \) symbols \( A_m^{q-1}, A_m^{q-2}, \ldots, A_m \) appear in subsequent sequences \( S_{i+1}, S_{i+2}, \ldots \). Summing the powers \( q \) for all such symbols \( A_m^q \), one must obtain a quantity \( Q_2 + \ldots + Q_{k+1} \) upper bounded by \( |S_2| + \ldots + |S_{k+1}| \).

Let \( x \) be the binary string of length \( 2^k \) represented by \( G \) (i.e., \( \phi_G(V_G^x) = x \)). Fix \( i \) satisfying \( 2 \leq i \leq k + 1 \). From left to right, partition \( x \) into disjoint substrings of length \( 2^{k-i+2} \), and let \( u_1, u_2, \ldots, u_M \) be the list of distinct substrings in this partition, listed in order of first left-to-right appearance in the partition. For each \( u_m \) in this list, let \( u_m(L) \) denote the prefix of \( u_m \) of length \( 2^{k-i+1} \), and let \( u_m(R) \) denote the suffix of \( u_m \) of length \( 2^{k-i+1} \). (In other words, when we bisect the string \( u_m \), we obtain \( u_m(L) \) on the left, and \( u_m(R) \) on the right.) Replace each \( u_m \) in the sequence \( (u_1, \ldots, u_M) \) for which \( u_m(L) \neq u_m(R) \) by the pair of strings \( u_m(L), u_m(R) \); otherwise, if \( u_m(L) = u_m(R) \), replace \( u_m \) by \( u_m(L) \). These replacements yield a new sequence \( v_i \) whose entries are substrings of \( x \) of length \( 2^k-i+1 \). The following properties can be proved (see [3]).

**Property 1:** The sequence \( S_i \) has the same length as the sequence \( v_i \).

**Property 2:** Writing

\[
S_i = (q_1, q_2, \ldots, q_M) \\
v_i = (r_1, r_2, \ldots, r_M)
\]

the sets \( \{q_1, q_2, \ldots, q_M\} \) and \( \{r_1, r_2, \ldots, r_M\} \) are of the same size, and there is a one-to-one mapping \( \alpha_i \) from the first set onto the second set in which

\[
v_i = (\alpha_i(q_1), \alpha_i(q_2), \ldots, \alpha_i(q_M))
\]

**Property 3:** There is a partition \( \Pi \) of \( x \) and disjoint subsequences \( s^2, s^3, \ldots, s^{k+1} \) of \( \Pi \) (some of which may be empty), such that

\[ s^i = \tilde{v}_i, \quad 2 \leq i \leq k + 1 \]

**Definitions.** We let \( \Lambda \) denote the family of all mappings \( \lambda : \{0,1\}^+ \to (0,1] \) such that for every sequence \( u \in \{0,1\}^+ \), and every partition \( (u_1, u_2, \ldots, u_r) \) of \( u \) into nonempty substrings of \( u \),

\[ \lambda(u) \leq \lambda(u_1) \lambda(u_2) \ldots \lambda(u_r) \]  

(4.6)
If $\lambda \in \Lambda$, we define

$$|\lambda| = \sup_{n=1,2,\ldots} \sum_{u \in \{0,1\}^n} \lambda(u)$$

**Lemma 3** Let $\lambda$ be a function in $\Lambda$ for which $|\lambda| < \infty$. Let $G \in \mathcal{G}^*$, let $x$ be the binary string of length $2^k$ represented by $G$, and let $S_1, S_2, \ldots, S_k$ be the strings defined for $G$ according to Section II. Then,

$$\sum_{i=2}^{k+1} [H(\pi_i^1) + H(\tilde{\pi}_i^2)] \leq \left(\sum_{i=2}^{k+1} |S_i|\right) \log_2 |\lambda| - \log_2 \lambda(x)$$

(4.7)

**Proof.** The sequences $\pi_i^1$ and $\tilde{\pi}_i^2$ are disjoint subsequences of $\tilde{S}_i$. Applying Property 2, we have

$$H(\pi_i^1) + H(\tilde{\pi}_i^2) \leq H(\tilde{S}_i) = H(\tilde{v}_i)$$

The entries of $\tilde{v}_i = (w_1, \ldots, w_T)$ are substrings of $x$ of length $2^{k-i+1}$. Let $\Sigma \leq |\lambda|$ be the positive constant such that

$$\mu(y) = \lambda(y)/\Sigma, \quad y \in \{0,1\}^{2^{k-i+1}}$$

defines a probability distribution on $\{0,1\}^{2^{k-i+1}}$. Then,

$$H(\tilde{v}_i) \leq -\log_2 \mu(w_1) - \log_2 \mu(w_2) - \ldots - \log_2 \mu(w_T)$$

$$\leq |S_i| \log_2 |\lambda| - \sum_{t=1}^T \log_2 \lambda(w_t)$$

Summing the preceding inequality over $i$ in the range $2 \leq i \leq k + 1$, and using Property 3 together with the property (4.6) of $\lambda$, we obtain (4.7).

**Lemma 4** There is a sequence of positive numbers $\{\epsilon_k : k = 1,2,\ldots\}$ converging to zero such that the following is true. For any $k = 1,2,\ldots$ and any $G \in \mathcal{G}^*$ representing a binary string of length $2^k$, if we let $S_1, S_2, \ldots, S_{k+1}$ be the strings defined from $G$ in Section II,

$$|S_1| + |S_2| + \ldots + |S_{k+1}| \leq \frac{2^{k+1}(2 + \epsilon_k)}{k}$$

(4.8)

**Sketch of Proof.** Suppose $G$ is any graph in $\mathcal{G}^*$ representing a binary string $x$ of length $2^k$. Let $S(x)$ be the set of all binary strings which lie in the partitions of $x$ into substrings of length $1,2,2^2,\ldots,2^k$. Define the graph $G'$ to be the graph in which:

- The set of vertices of $G'$ is $S(x)$. The set of terminal vertices of $G'$ is $\{0,1\}$.
- For each nonterminal vertex $u$ of $G'$, there are two edges emanating from $u$, one of which, labelled edge 0, terminates at the left half of $u$, and the other of which, labelled edge 1, terminates at the right half of $u$. 

11
The graph $G'$ is isomorphic to the graph termed by Liaw and Lin [4] the quasi-reduced ordered binary decision diagram corresponding to the ROBDD $G$. Let $V(G)$ be the set of vertices of $G$ and let $V(G')$ be the set of vertices of $G'$. It is proved in the paper [4] that there exists a sequence of positive constants $\{\epsilon_k\}$ tending to zero such that for any $k$ and any $G \in G^{*}$ representing a binary string of length $2^k$,

$$|V(G)| \leq |V(G')| \leq \frac{2^k(2 + \epsilon_k)}{k} \tag{4.9}$$

For the strings $S_1, S_2, \ldots, S_{k+1}$ defined for this same $G$ as in Section II, it can be shown (we omit the proof here) that

$$|S_1| + |S_2| + \ldots + |S_{k+1}| \leq |V(G')| + |V(G)| \tag{4.10}$$

Combining (4.9) and (4.10), we obtain (4.8).

Here is our main result.

**Theorem 1** Consider an arbitrary binary $s$-state information source. For each binary string $x$ of finite length, let $\mu(x)$ denote the probability assigned to $x$ by the given source. Then, for $n = 2, 4, 8, 16, \ldots$,

$$\max\{x \in \{0, 1\}^n \cap S(\text{dyadic}) : |\sigma(x)| + \log_2 \mu(x)\} \leq \left(\frac{n}{\log_2 n}\right) (16 + 4 \log_2 s + o(1))$$

*Proof.* Fix a $\lambda \in \Lambda$ such that

- $|\lambda| \leq s$.
- $\mu(y) \leq \lambda(y)$ for every binary string $y$.

Fix $n \in \{1, 2, 4, 8, \ldots\}$ and $x \in \{0, 1\}^n \cap S(\text{dyadic})$. Let $G \in G^{*}$ be the graph $G = G_x$. Let $k = \log_2 n$, and let $S_1, S_2, \ldots, S_{k+1}$ be the strings constructed from $G$ according to Section II. Applying Lemmas 3 and 4 to (4.5),

$$|\sigma(G)| \leq 4 \left[\frac{2^{k+1}(2 + \epsilon_k)}{k}\right] + \left[\frac{2^{k+1}(2 + \epsilon_k)}{k}\right] \log_2 s - \log_2 \mu(x)$$

which gives us our result.
Figure 1: A ROBDD $G$ from Bryant [1] (left edges labelled 0, right edges labelled 1)
Figure 2: Canonical ordering of vertices of ROBDD $G$ in Figure 1
References

[1] R. Bryant, “Graph-Based Algorithms for Boolean Function Manipulations,” *IEEE Transactions on Computers*, Vol. C-35, pp. 677–691, 1986.

[2] R. Bryant, “Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams,” *ACM Computing Surveys*, Vol. 24, pp. 293–318, 1992.

[3] J. Kieffer, E.-h. Yang, G. Nelson, and P. Cosman, “Universal Lossless Compression Via Multilevel Pattern Matching,” *IEEE Trans. Inform. Theory*, Vol. 46, pp. 1227–1245, 2000.

[4] H-t. Liaw and C-s. Lin, “On the OBDD-Representation of General Boolean Functions,” *IEEE Transactions on Computers*, Vol. 41, pp. 661–664, 1992.