PAPER

# Head-Tail Expressions for Interval Functions

**Infall SYAFALNI**[†a)] ***and*** **Tsutomu SASAO**[††b)], ***Members***

**SUMMARY** This paper shows a method to represent interval functions by using head-tail expressions. The head-tail expressions represent *greater-than* $GT(X : A)$ functions, *less-than* $LT(X : B)$ functions, and interval functions $IN_0(X : A, B)$ more efficiently than sum-of-products expressions. Let $n$ be the number of bits to represent the largest value in the interval $(A, B)$. This paper proves that a head-tail expression (HT) represents an interval function with at most $n$ words in a ternary content addressable memory (TCAM) realization. It also shows the average numbers of factors to represent interval functions by HTs for up to $n = 16$, which were obtained by a computer simulation. It also conjectures that, for sufficiently large $n$, the average number of factors to represent $n$-variable interval functions by HTs is at most $\frac{2}{3}n - \frac{5}{9}$. Experimental results also show that, for $n \geq 10$, to represent interval functions, HTs require at least 20% fewer factors than MSOPs, on the average.

***key words:*** *prefix sum-of-products, head-tail expressions, TCAM*

## 1. Introduction

Recent developments of network technology demand a high-speed processing of packets. Packet classification [1], [5] is a fundamental network primitive. The key device that supports this technology is a ternary content addressable memory (TCAM) [7], [11]. Since TCAMs check rules in parallel, they are *de facto* standard for high speed packet classification. However, inspite of its high-speed classification ability, the TCAMs dissipate high power and are expensive. These problems tend to be worse with the growth of the internet [17].

Thus, to overcome these drawbacks, reduction of TCAM size is essential. Since the problems of TCAM minimization is related to logic minimization, a logic minimizer, such as ESPRESSO can be utilized [2]. However, an exact minimization of a TCAM is extremely time consuming [6].

Table 1 shows an example of a classification function. This function has two fields that correspond to the source and the destination ports represented by intervals. In Table 1, values are tested in a sequential manner from the top to the bottom. In a TCAM, the operation is equivalent to testing rows in a sequential order [6]. When each port is specified by either * (*don't care*) or a single value, each rule corresponds to one word in a TCAM. However, when a port is specified by an interval such as $(0, 65536)$, the interval must be represented by multiple words in a TCAM [3]. For example, the interval $(0, 65536)$ requires 16 words. Suppose that the header of incoming packets with source port 1080 wants to access a destination with destination port 2080. As in Table 1, the header does not match to the first rule, but matches to the second rule, thus the action is *Accept* and the packet is sent to the destination.

**Table 1** Example of classification function.

| Rule | Source Port | Destination Port | Action |
|---|---|---|---|
| 1 | $(0, 65536)$ | 6790 | Accept |
| 2 | $(999, 2001)$ | $(0, 5590)$ | Accept |
| 3 | * | * | Deny |

Table 2 compares our work with previous works, where $n$ denotes the number of bits to represent the largest value in the interval. The first method [12] uses a special circuit to represent an interval directly. Thus, any interval can be represented by a single word. However, this method is the most expensive because it uses non-standard TCAMs*. The second method [10] uses an exact minimum sum-of-products expression (MSOP) to represent an interval. This method uses standard TCAM, and any interval function can be represented with at most $2(n - 2)$ products. Since we have to minimize TCAM words, this method is quite time consuming. The third method [8] uses output encoding. This method also uses a special circuit in addition to the TCAM, while it requires at most $n$ words to represent an interval. The method proposed in this paper uses a head-tail expression (HT) [4] to represent an interval. This method requires a RAM in addition to the TCAM. Since HTs can be generated from the binary representations of endpoints of the intervals, time to generate HTs is quite short. The third method and our methods require the same number of TCAM words to represent a field. However, our method uses only standard components such as TCAM and RAM. On the other hand the method of [8] requires special hardware, which would be very expensive.

In this paper, we show a method to represent an interval function using a head-tail expression (HT). The head-tail expressions efficiently represent *greater-than* $GT(X : A)$ functions, *less-than* $LT(X : B)$ functions, and interval functions

*Using non-standard LSIs is very expensive, since the development cost of such LSIs is very high, but the size of the market is not large enough to amortize the development cost.

Manuscript received March 17, 2014.
Manuscript revised June 12, 2014.

†The author is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.
††The author is with the Department of Computer Science, Meiji University, Kawasaki-shi, 214-8571 Japan.
a) E-mail: infall@aries30.cse.kyutech.ac.jp
b) E-mail: sasao@cs.meiji.ac.jp

DOI: 10.1587/transfun.E97.A.2043

**Table 2** Comparison with previous works.

| Parameter | Ref. [12] | Ref. [10] | Ref. [8] | This paper |
|---|---|---|---|---|
| Method | Comparator | MSOP | Output encoding | Head-tail expr. |
| Hardware | Special circuit | TCAM | TCAM + Special circuit | TCAM + RAM |
| Representation | Direct interval | $n$-bit non-prefix | $n$ + 1-bit prefix | $n$-bit prefix |
| Max. # of words to represent a field | 1 | $2(n-2)$ | $n$ | $n$ |
| Cost | High | Low | High | Low |

$IN_0(X : A, B)$. We prove that any interval function can be represented by an HT with at most $n$ factors. We also show the average numbers of factors to represent interval functions by HTs for up to $n = 16$, which were obtained by our heuristic minimization algorithm. And, we conjecture that, for sufficiently large $n$, the average number of factors in HTs to represent $n$-variable interval functions is $\frac{2}{3}n - \frac{5}{9}$. By this experiment, we also show that, for $n \geq 10$, to represent interval functions, HTs require at least 20% fewer factors than MSOPs, on the average.

This paper is organized as follows: In Sect. 2, important words are defined and the basic properties of interval function are explained. In Sect. 3, a head-tail expression (HT) is introduced to represent $GT$, $LT$ and $IN_0$ functions. In Sect. 4, experimental results are shown. Finally, in Sect. 5, the paper is concluded. A preliminary version of this paper was presented in [13].

## 2. Definition and Basic Properties

In this section, we present definitions and basic properties before we step into the main contribution of this paper i.e., **head-tail expression**. First, we define a prefix sum-of-products expression (**PreSOP**), and we give some examples to make it more understandable. Second, we define open interval and open interval functions; a **greater-than function** ($GT$), a **less-than function** ($LT$), and an **interval function** ($IN_0$). We also show examples for $GT$, $LT$ and $IN_0$ functions.

### 2.1 Prefix Sum-of-Products Expression

**Definition 2.1:** ${x_i}^{a_i}$ denotes $x_i$ when $a_i = 1$, and $\bar{x}_i$ when $a_i = 0$. $x_i$ and $\bar{x}_i$ are **literals** of a variable $x_i$. The AND of literals is a **product**. The OR of products is a **sum-of-products expression** (SOP).

**Definition 2.2:** A **prefix SOP** (PreSOP) is an SOP consisting of products having the form $x^*_{n-1}x^*_{n-2}\ldots x^*_{m+1}x^*_m$, where $x^*_i$ is $x_i$ or $\bar{x}_i$ and $m \leq n-1$.

**Example 2.1:** $f(x_2, x_1, x_0) = x_2 \vee \bar{x}_2 x_1 \vee \bar{x}_2\bar{x}_1 x_0$ is a PreSOP. $f(x_2, x_1, x_0) = x_2 \vee x_1 \vee x_0$ is an SOP, but it is not a PreSOP. ■

**Definition 2.3:** An SOP representing a given function $f$ with the fewest products is a minimum sum-of-product expression (MSOP). A PreSOP representing a given function $f$ with the fewest products is a minimum PreSOP (MPreSOP). An MSOP and an MPreSOP for $f$ are denoted by MSOP($f$) and MPreSOP($f$), respectively.

**Definition 2.4:** Let $\mathcal{F}$ be an SOP. $\tau(\mathcal{F})$ denotes the number of products in $\mathcal{F}$. $\tau_p(f)$ denotes the number of products in MPreSOP($f$).

In general, an SOP require fewer products than a PreSOP to represent the same function [10]. However, in the internet communication area, PreSOPs are used instead of SOPs, since PreSOPs can be quickly generated from the binary decision trees of the functions [16].

### 2.2 Interval Functions

**Definition 2.5:** Let $A$ and $B$ be integers such that $A < B$. An **open interval** $(A, B)$ does not include its endpoints.

**Definition 2.6:** Let $X$, $A$ and $B$ be integers. An $n$-input **open interval function** is

$$IN_0(X : A, B) = \begin{cases} 1, & \text{if } A < X < B \\ 0, & \text{otherwise.} \end{cases}$$

An $n$-input **greater-than function** ($GT$) function is

$$GT(X : A) = \begin{cases} 1, & \text{if } X > A \\ 0, & \text{otherwise.} \end{cases}$$

An $n$-input **less-than function** ($LT$) function is

$$LT(X : B) = \begin{cases} 1, & \text{if } X < B \\ 0, & \text{otherwise,} \end{cases}$$

where $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$.

**Lemma 2.1:** The number of distinct $n$-variable interval functions in $(A, B)$, where $-1 \leq A < B \leq 2^n$, is $N(n) = 2^{n-1}(2^n + 1)$.

*Proof:* Let the size of an interval $(A, B)$ be $C = B - A - 1$. For $C = 1, C = 2, \ldots, C = 2^n$, the number of distinct interval functions are $2^n, 2^n - 1, 2^n - 2, \ldots, 1$, respectively. Thus, we have $N(n) = 2^n + (2^n - 1) + (2^n - 2) + \ldots + 1 = 2^{n-1}(2^n + 1)$. □

**Lemma 2.2** ([15]): The minimum PreSOPs (MPreSOPs) of $GT$ and $LT$ functions can be represented as follows:

$$GT(X : A) = (x_{n-1}\bar{a}_{n-1}) \vee \bigvee_{i=n-2}^{0} \left( \bigwedge_{j=n-1}^{i+1} x_j^{a_j} \right) x_i \bar{a}_i,$$

$$LT(X : B) = (\bar{x}_{n-1} b_{n-1}) \vee \bigvee_{i=n-2}^{0} \left( \bigwedge_{j=n-1}^{i+1} x_j^{b_j} \right) \bar{x}_i b_i,$$

where $\vec{a} = (a_{n-1}, \cdots, a_0)$ and $\vec{b} = (b_{n-1}, \cdots, b_0)$ are the binary representations of $A$ and $B$, repectively. $GT$ and $LT$ have $\sum_{i=0}^{n-1} \bar{a}_i$ and $\sum_{i=0}^{n-1} b_i$ disjoint products, respectively.

**Example 2.2:** When $n = 4$ and $A = 0$, we have $\vec{a} = (0, 0, 0, 0)$. Thus, $GT(4:0) = x_3 \vee x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_0\bar{x}_1\bar{x}_2\bar{x}_3$. ■

**Example 2.3:** When $n = 4$ and $B = 15$, we have $\vec{b} = (1, 1, 1, 1)$. Thus, $LT(X:15) = \bar{x}_3 \vee \bar{x}_2 x_3 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_0 x_1 x_2 x_3$. ■

**Theorem 2.1** ([15]): Let $\vec{a} = (a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$ and $\vec{b} = (b_{n-1}, b_{n-2}, \cdots, b_1, b_0)$ be the binary representations of $A$ and $B$, respectively, and $A < B$. Let $s$ be the largest index such that $a_s \neq b_s$. Then, $IN_0(X : A, B)$ can be represented by

$$\bigvee_{i=s-1}^{0} \left[ \left( \bigwedge_{j=n-1}^{i+1} x_j^{a_j} \right) x_i \bar{a}_i \vee \left( \bigwedge_{j=n-1}^{i+1} x_j^{b_j} \right) \bar{x}_i b_i \right].$$

The number of products is

$$\tau_p(IN_0(X : A, B)) = \sum_{i=0}^{s-1} (\bar{a}_i + b_i).$$

When $A = B - 1$ or $A + 1 = B$, the interval $(A, B)$ has empty set, thus $IN_0(X : A, B)$ has no product (including the case when $s = 0$). When $A = -1$ and/or $B = 2^n$, these endpoints are called by **extremal endpoints**.

**Lemma 2.3** ([15]): In the extremal endpoints, we have $GT(X : -1) = LT(X : 2^n) = IN_0(X : -1, 2^n) = 1$, $IN_0(X : -1, B) = LT(X : B)$, and $IN_0(X : A, 2^n) = GT(X : A)$.

The optimality of $GT(X : A)$, $LT(X : B)$, and $IN_0(X : A, B)$ functions represented by PreSOPs has been discussed in the reference [15].

**Example 2.4:** Let $A = 0$, $B = 31$ and $n = 5$. In this case, $\vec{a} = (0, 0, 0, 0, 0)$ and $\vec{b} = (1, 1, 1, 1, 1)$. By Theorem 2.1, the PreSOP for $IN_0(X : 0, 31)$ is

$$\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1 x_0 \vee x_4 x_3 x_2 x_1 \bar{x}_0 \vee \bar{x}_4\bar{x}_3\bar{x}_2 x_1 \vee x_4 x_3 x_2 \bar{x}_1 \vee \bar{x}_4\bar{x}_3 x_2 \vee x_4 x_3 \bar{x}_2 \vee \bar{x}_4 x_3 \vee x_4 \bar{x}_3.$$

Figure 1(a) shows its map. The integers in the map denote decimal representations of minterms, where $X = \sum_{i=0}^{n-1} x_i \cdot 2^i$. The PreSOP requires $\tau_p(IN_0(X : 0, 31)) = 4 + 4 = 8$ products. ■

Note that an MSOP for $IN_0(X : 0, 31)$ is $\bar{x}_4 x_3 \vee \bar{x}_3 x_2 \vee \bar{x}_2 x_1 \vee \bar{x}_1 x_0 \vee \bar{x}_0 x_4$. Figure 1(b) shows its map.

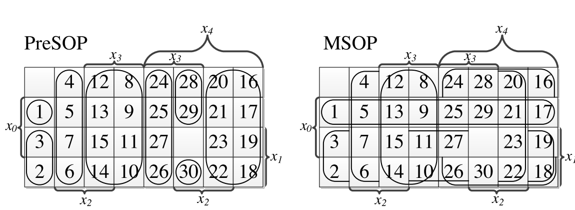



**Fig. 1** Maps for $IN_0(X : 0, 31)$.

## 3. Head-Tail Expressions for Interval Functions

In this section, we use head-tail expressions (HTs) to represent interval functions. HTs [4] were originally introduced to design NAND three-level networks. Lemma 2.2 shows that when the binary representation of $A$ has $t$ 0's, a PreSOP for $GT(X : A)$ requires $t$ products. Especially when $a_{n-1} = a_{n-2} = \cdots = a_0 = 0$, the PreSOP requires $n$ products. Similarly, it also shows that when the binary representation of $B$ has $t$ 1's, a PreSOP for $LT(X : B)$ requires $t$ products, and $n$ products when $b_{n-1} = b_{n-2} = \cdots = b_0 = 1$. Theorem 2.1 shows that when $a_{n-1} = a_{n-2} = \cdots = a_0 = 0$ and $b_{n-1} = b_{n-2} = \cdots = b_0 = 1$, the PreSOP for $IN_0(X : A, B)$ requires $2(n-1)$ products. Thus, if the PreSOP is used in a TCAM, we need up to $2(n-1)$ words.

However, the number of TCAM words can be reduced if we use the properties of a TCAM. We will show such a method in this section.

### 3.1 Derivation of Head-Tail Expressions for Interval Functions

**Definition 3.1:** A **head-tail expression** (HT) has a form

$$f = \bigvee_{i=t}^{0} \left[ \bigwedge_{j=s_i}^{0} (\bar{h}_{ij}) \right] \left[ \bigwedge_{k=v_i}^{0} (g_{ik}) \right], \quad (1)$$

where for $(i = 0, 1, \cdots, t)$, $(\bar{h}_{ij})$ is the **head factor** and $(g_{ik})$ is the **tail factor**, and $h_{ij}$ and $g_{ik}$ are represented by products. In this paper, (product) and $\overline{\text{(product)}}$ are called **factors**. Products are used for PreSOPs and MSOPs, while factors are used for HTs. Both products and factors are realized in the form of **words** in TCAMs. Note that an SOP is considered as a special case of an HT.

**Example 3.1:** $\overline{(\bar{x}_6\bar{x}_5\bar{x}_4)} \cdot \overline{(\bar{x}_6\bar{x}_5 x_4)} \cdot (x_3 x_2) \vee \overline{(\bar{x}_6\bar{x}_5\bar{x}_4)} \cdot \overline{(\bar{x}_6\bar{x}_5 x_4)} \cdot (\bar{x}_3\bar{x}_2)$ is an HT. ■

**Lemma 3.1:** The circuit in Fig. 2 consisting of a TCAM and a RAM implements an HT.

In Fig. 2, the circuit realizes the function $f = (\bar{h}_0)g_0 \vee$

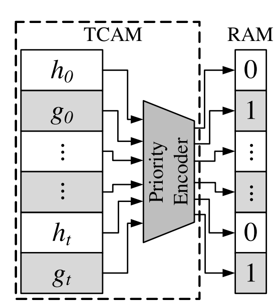



**Fig. 2** Circuit for a head-tail expressions.

$(\bar{h}_1)g_1 \vee \cdots \vee (\bar{h}_t)g_t$. Note that TCAM has a priority encoder in the output part [7], [11]. A factor corresponds to a word in a TCAM. Since the upper words have higher priority than the lower words, the TCAM will produce the action for the upmost matched word. Thus, in Fig. 2, if the input pattern mismatches $h_0$ and matches $g_0$, then the output is 1. However, if the input pattern matches both $h_0$ and $g_0$, then the output is 0. Thus, unlike Programmable Logic Arrays (PLAs) [9], the order of words stored in the TCAM is very important.

Any logic function can be represented by a canonical sum-of-products expression (i.e., minterm expansion). It is a special case of a PreSOP, and a PreSOP is a special case of an SOP, and an SOP is a special case of an HT. Thus, any logic function can be represented by an HT. In particular, any interval function can be represented by an HT. Unfortunately, the HT derived by Theorem 2.1 requires many factors.

In this part, we show a more efficient way to represent an interval function by an HT. The general idea is to decompose a given function into sub-functions, so that each sub-function require a small number of factors.

Consider the case of $GT(X : A)$. As shown in Lemma 2.2, the more 0's in the binary representation of $A$, the more product terms are necessary in the expression. First, we will show that when the binary representation of $A$ has a consecutive 0's, we have an efficient representation.

**Definition 3.2:** The **integer representation** of a binary number $\vec{a} = (a_{n-1}, a_{n-2}, \ldots, a_0)$ is $A = \sum_{i=0}^{n-1} a_i 2^i$, and denoted by $A = INT(\vec{a})$. The **complement** of an integer $A$ is defined as $2^n - 1 - A$, and denoted by $COMP(A)$.

**Example 3.2:** When $\vec{a} = (1, 0, 1, 1)$. We have $INT(\vec{a}) = 8 + 2 + 1 = 11$ and $COMP(11) = 4$. ■

To extract the least significant consecutive 0's in a binary vector, we use the 0**-extraction vector**.

**Definition 3.3:** Let $\vec{a}$ and $\vec{a}'$ be binary vectors of $n$ bits such that $INT(\vec{a}) \leq INT(\vec{a}')$. Further assume that

$$a_i = a'_i = 1 \text{ for } i = 0, 1, \ldots, m-d-1,$$
$$a_i = a'_i = 0 \text{ for } i = m-d, m-d+1, \ldots, m-1, \text{ and,}$$
$$a'_i = 1 \text{ for } i = m, m+1, \ldots, n-1.$$

Then, $\vec{e} = \vec{a} \vee \overline{\vec{a}'}$ denotes the 0**-extraction vector** for consecutive 0's in the least significant bits, where $\overline{\vec{a}'}$ is the complement of $\vec{a}'$, and $\vee$ denotes the bitwise OR operation. Note that $\vec{a}$ uniquely determines $\vec{a}'$.

**Example 3.3:** Let $\vec{a} = (1, 0, 1, 0, 0, 1, 1)$ and $\vec{a}' = (1, 1, 1, 0, 0, 1, 1)$. These vectors satisfy the properties of Definition 3.3, where $n = 7$, $m = 4$, and $d = 2$. Then, the 0-extraction vector is $\vec{e} = \vec{a} \vee \overline{\vec{a}'} = (1, 0, 1, 1, 1, 1, 1)$. ■

**Definition 3.4:** Let $\vec{b}$ and $\vec{b}'$ be binary vectors of $n$ bits such that $INT(\vec{b}') \leq INT(\vec{b})$. Further assume that

$$b_i = b'_i = 0 \text{ for } i = 0, 1, \ldots, m-d-1,$$
$$b_i = b'_i = 1 \text{ for } i = m-d, m-d+1, \ldots, m-1, \text{ and ,}$$
$$b'_i = 0 \text{ for } i = m, m+1, \ldots, n-1.$$

Then, $\vec{e} = \vec{b} \wedge \overline{\vec{b}'}$ denotes the 1**-extraction vector** for consecutive 1's in the least significant bits, where $\overline{\vec{b}'}$ is the complement of $\vec{b}'$, and $\wedge$ denotes the bitwise AND operation. Note that $\vec{b}$ uniquely determines $\vec{b}'$.

**Example 3.4:** Let $\vec{b} = (0, 1, 0, 1, 1, 0, 0)$ and $\vec{b}' = (0, 0, 0, 1, 1, 0, 0)$. These vectors satisfy the properties of Definition 3.4, where $n = 7$, $m = 4$, and $d = 2$. Then, the 1-extraction vector is $\vec{e} = \vec{b} \wedge \overline{\vec{b}'} = (0, 1, 0, 0, 0, 0, 0)$. ■

**Lemma 3.2:** Let $\vec{a} = (a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$ and $\vec{a}' = (a'_{n-1}, a'_{n-2}, \cdots, a'_1, a'_0)$ be binary vectors satisfying the property of Definition 3.3. Let $d$ ($d \geq 1$) be the number of 0's in the consecutive 0's in $\vec{a}'$. Let $\vec{e} = \vec{a} \vee \overline{\vec{a}'}$ be the 0-extraction vector. Then, $\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a}))$ can be represented by the HT with two factors:

$$\overline{\left(\bigwedge_{j=n-1}^{m} x_j^{a_j} \bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)} \cdot \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right).$$

When $n-1 < m$, the product $\bigwedge_{j=n-1}^{m} x_j^{a_j}$ is represented by the constant function 1. Note that, when $\vec{a}' = \vec{a}$, $\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a})) = GT(X : INT(\vec{a}))$.

*Proof:* In this case, we only consider the group of consecutive 0's specified by the vector $\vec{a}'$.

$$\begin{aligned}
&\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a})) \\
&= \bigvee_{i=m-1}^{m-d} \left(\bigwedge_{j=n-1}^{i+1} x_j^{a_j}\right) x_i \\
&= \bigvee_{i=m-1}^{m-d} \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right)\left(\bigwedge_{k=m-1}^{i+1} x_k^{a_k}\right) x_i \\
&= \bigvee_{i=m-1}^{m-d} \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right) x_i \\
&= \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right) \cdot \left(\bigvee_{i=m-1}^{m-d} x_i\right) \\
&= \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right) \cdot \overline{\left(\bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)} \\
&= \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right) \cdot \left(\overline{\left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right)} \vee \overline{\left(\bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)}\right) \\
&= \overline{\left(\bigwedge_{j=n-1}^{m} x_j^{a_j} \bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)} \cdot \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right).
\end{aligned}$$

Thus, we have the lemma. In this case: $(\bar{h}_1) = \overline{\left(\bigwedge_{j=n-1}^{m} x_j^{a_j} \bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)}$ is the head factor, and $(g_1) =$

$\left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right)$ is the tail factor. Note that, when $n-1 < m$, the product $\bigwedge_{j=n-1}^{m} x_j^{a_j}$ is represented by the constant function 1. Note that, when $\vec{a}' = \vec{a}$, $\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a})) = \overline{GT(X : 2^n - 1)} \cdot GT(X : INT(\vec{a})) = GT(X : INT(\vec{a}))$. □

**Example 3.5:** Let $\vec{a} = (1,0,1,1,0,0,0)$ and $\vec{a}' = (1,1,1,1,0,0,0)$. $\vec{a}$ and $\vec{a}'$ satisfy the properties of Definition 3.3, where $n = 7$, $m = 3$ and $d = 3$. In this case, the 0-extraction vector is $\vec{e} = \vec{a} \vee \overline{\vec{a}'} = (1,0,1,1,1,1,1)$. In $\vec{a}$, there is a group of consecutive 0's. By Lemma 3.2, the HT for $\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a}))$ is represented as

$$\overline{\left(\bigwedge_{j=n-1}^{m} x_j^{a_j} \bigwedge_{i=m-1}^{m-d} \bar{x}_i\right)} \cdot \left(\bigwedge_{j=n-1}^{m} x_j^{a_j}\right)$$
$$= \overline{(x_6\bar{x}_5x_4x_3\bar{x}_2\bar{x}_1\bar{x}_0)} \cdot (x_6\bar{x}_5x_4x_3).$$

It requires two factors. ■

**Lemma 3.3:** Let $\vec{b} = (b_{n-1}, b_{n-2}, \cdots, b_1, b_0)$ and $\vec{b}' = (b'_{n-1}, b'_{n-2}, \cdots, b'_1, b'_0)$ be binary vectors satisfying the property of Definition 3.4. Let $d$ ($d \geq 1$) be the number of 1's in the consecutive 1's in $\vec{b}'$. Let $\vec{e} = \vec{b} \wedge \overline{\vec{b}'}$ be the 1-extraction vector. Then, $\overline{LT(X : INT(\vec{e}))} \cdot LT(X : INT(\vec{b}))$ can be represented by the HT with two factors:

$$\overline{\left(\bigwedge_{j=n-1}^{m} x_j^{b_j} \bigwedge_{i=m-1}^{m-d} x_i\right)} \cdot \left(\bigwedge_{j=n-1}^{m} x_j^{b_j}\right).$$

When $n-1 < m$, the product $\bigwedge_{j=n-1}^{m} x_j^{b_j}$ is represented by the constant function 1. Note that, when $\vec{b}' = \vec{b}$, $\overline{LT(X : INT(\vec{e}))} \cdot LT(X : INT(\vec{b})) = LT(X : INT(\vec{b}))$.

*Proof:* The proof is similar to that of Lemma 3.2. □

Lemmas 3.2 and 3.3 are applicable to interval functions with a special property.

As explained before, a PreSOP is a special case of HTs, thus an interval function can be represented by an HT. Note that an interval function can be *segmented* into smaller interval functions which are represented by HTs.

**Lemma 3.4:** Let $-1 \leq A < B \leq 2^n$. An interval function $IN_0(X : A, B)$ can be represented by

$$IN_0(X : A, B) = GT(X : A) \cdot \overline{GT(X : B-1)}$$
$$= \overline{LT(X : A+1)} \cdot LT(X : B).$$

*Proof:* An interval function $IN_0$ can be represented by a AND of $GT$ and $LT$ functions

$$IN_0(X : A, B) = GT(X : A) \cdot LT(X : B).$$

Since $GT(X : A) = \overline{LT(X : A+1)}$ and $LT(X : B) = \overline{GT(X : B-1)}$, we have the lemma. □

**Example 3.6:** Represent $IN_0(X : -1, 4)$ and $IN_0(X : 3, 8)$ by $GT$ and/or $LT$ functions.

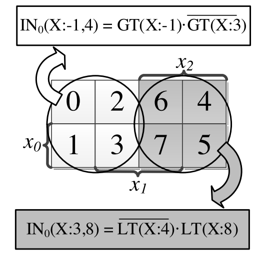



**Fig. 3** Example of Lemma 3.4.

$$IN_0(X : -1, 4) = GT(X : -1) \cdot \overline{GT(X : 3)}$$
$$= \overline{LT(X : 0)} \cdot LT(X : 4)$$
$$IN_0(X : 3, 8) = GT(X : 3) \cdot \overline{GT(X : 7)}$$
$$= \overline{LT(X : 4)} \cdot LT(X : 8)$$

Figure 3 illustrates Lemma 3.4. The white part corresponds to an expression for $IN_0(X : -1, 4)$, while the grey part corresponds to an expression for $IN_0(X : 3, 8)$. ■

**Theorem 3.1:** A $GT$ function can be represented as:

$$GT(X : A) = \bigvee_{i=r-1}^{0} \overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{a}_i)),$$

where $INT(\vec{a}_0) = A$, and $\vec{a}_{i+1} = \vec{e}_i = \vec{a}_i \vee \overline{\vec{a}'_i}$ $(i = 0, 1, 2, \ldots, r-1)$ are 0-extraction vectors, and $\vec{e}_{r-1} = (1, 1, 1, \ldots, 1)$.

*Proof:* Let $\vec{a}_0 = \vec{a}$. If there are groups of consecutive 0's in $\vec{a}_0$, then we extract the vectors by finding a group of consecutive 0's at the least significant bit and masking it by $\vec{a}'_i$. Then, we represent every 0's from the least significant bits of $\vec{a}_0$ by decomposing the $GT$ function into two parts, where $\vec{e}_0 = \vec{a}_0 \vee \overline{\vec{a}'_0}$:

$$GT(X : A) = \overline{GT(X : INT(\vec{e}_0))} \cdot GT(X : INT(\vec{a}_0))$$
$$\vee\, GT(X : INT(\vec{e}_0)).$$

Next, if there are groups of consecutive 0's in $\vec{a}_1 = \vec{e}_0$, we further decompose the last part into two, where $\vec{e}_1 = \vec{a}_1 \vee \overline{\vec{a}'_1}$:

$$GT(X : A) = \overline{GT(X : INT(\vec{e}_0))} \cdot GT(X : INT(\vec{a}_0))$$
$$\vee\, \overline{GT(X : INT(\vec{e}_1))} \cdot GT(X : INT(\vec{a}_1))$$
$$\vee\, GT(X : INT(\vec{e}_1)).$$

We decompose the function until $INT(\vec{e}_{r-1}) = 2^n - 1$, where $r$ is the number of groups of consecutive 0's in $\vec{a}$. Note that $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{a}_i))$ can be obtained by Lemma 3.2 or Lemma 2.2. □

**Example 3.7:** Let $\vec{a} = (0, 1, 1, 0, 0)$, where $n = 5$. Represent $GT(X : INT(\vec{a}))$ by HT. Let $\vec{a}'_0 = (1, 1, 1, 0, 0)$ and use Theorem 3.1 to decompose $GT(X : INT(\vec{a}))$. Note that $\vec{a}_0 = \vec{a}$ and $\vec{e}_0 = \vec{a}_1 = \vec{a}_0 \vee \overline{\vec{a}'_0} = (0, 1, 1, 1, 1)$. We have:

$$GT(X : INT(\vec{a})) = \overline{GT(X : INT(\vec{e}_0))}$$

$$\cdot GT(X : INT(\vec{d}_0)) \vee GT(X : INT(\vec{e}_0)).$$

By Lemma 3.2, we have $\overline{GT(X : INT(\vec{e}_0))} \cdot GT(X : INT(\vec{d}_0)) = \overline{(\bar{x}_4x_3x_2\bar{x}_1\bar{x}_0)} \cdot (\bar{x}_4x_3x_2)$. Next, let $\vec{d}'_1 = (0,1,1,1,1)$. Thus, $\vec{e}_1 = \vec{d}_1 \vee \overline{\vec{d}'_1} = (1,1,1,1,1)$. We have:

$$GT(X : INT(\vec{d}_1)) = \overline{GT(X : INT(\vec{e}_1))} \cdot GT(X : INT(\vec{d}_1)) \vee GT(X : INT(\vec{e}_1)).$$

Note that $GT(X : INT(\vec{e}_1)) = 0$. In this case, if we use Lemma 3.2, $GT(X : INT(\vec{d}_1))$ requires two factors, while, if we use Lemma 2.2, it requires only one product i.e., $GT(X : INT(\vec{d}_1)) = x_4$. Thus,

$$GT(X : INT(\vec{d})) = \overline{(\bar{x}_4x_3x_2\bar{x}_1\bar{x}_0)} \cdot (\bar{x}_4x_3x_2) \vee x_4.$$

It requires 3 factors. ■

**Theorem 3.2:** An *LT* function can be represented as:

$$LT(X : B) = \bigvee_{i=r-1}^{0} \overline{LT(X : INT(\vec{e}_i))} \cdot LT(X : INT(\vec{b}_i)),$$

where $INT(\vec{b}_0) = B$, and $\vec{b}_{i+1} = \vec{e}_i = \vec{b}_i \wedge \overline{\vec{b}'_i}$ $(i = 0, 1, 2, \ldots, r-1)$ are 1-extraction vectors and $\vec{e}_{r-1} = (0, 0, 0, \ldots, 0)$.

*Proof:* The proof is similar to that of Theorem 3.1. □

From Theorems 3.1 and 3.2, we have the following:

**Definition 3.5:** Let $h$ and $g$ be logic functions. If $g(x) = 1$ for all $x$ such that $h(x) = 1$, then $g$ **includes** $h$, denoted by $h \subseteq g$.

**Lemma 3.5:** If $h_0 \subseteq g_0 \subseteq h_1 \subseteq g_1 \subseteq \cdots \subseteq h_{p-2} \subseteq g_{p-2} \subseteq h_{p-1} \subseteq g_{p-1}$, then $Z = \bar{h}_0 g_0 \vee \bar{h}_1 g_1 \vee \cdots \vee \bar{h}_{p-2} g_{p-2} \vee \bar{h}_{p-1} g_{p-1}$ is represented by:

$$Z = \bar{h}_0(\bar{h}_1 \vee g_0)(\bar{h}_2 \vee g_1) \cdots (\bar{h}_{p-2} \vee g_{p-3})(\bar{h}_{p-1} \vee g_{p-2}) g_{p-1}$$

*Proof:* The grey area in the map of Fig. 4 indicates the covering of $Z$. Thus, we have the lemma. □

**Lemma 3.6:** Let $\vec{e} = (e_{n-1}, e_{n-2}, \ldots, e_1, e_0)$ be a binary vector. Consider two functions:

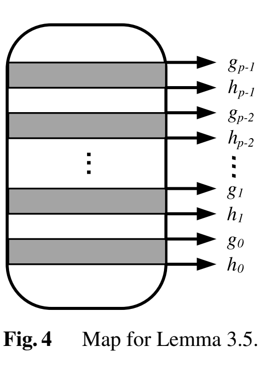



**Fig. 4** Map for Lemma 3.5.

$$g_{k-1} = \left(\bigwedge_{j=n-1}^{m} x_j^{e_j}\right) \text{ and } \bar{h}_k = \overline{\left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j}\right)}.$$

In this case, we can combine two factors into one:

$$\bar{h}_k \vee g_{k-1} = \overline{\left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j} \cdot x_m^{\bar{e}_m}\right)}.$$

*Proof:* $\bar{h}_k \vee g_{k-1}$ can be combined to a factor:

$$\begin{aligned}\bar{h}_k \vee g_{k-1} &= \overline{\left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j}\right)} \vee \left(\bigwedge_{j=n-1}^{m} x_j^{e_j}\right)\\ &= \overline{\left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j}\right)} \vee \left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j}\right) \cdot x_m^{e_m}\\ &= \overline{\left(\bigwedge_{j=n-1}^{m+1} x_j^{e_j} \cdot x_m^{\bar{e}_m}\right)}.\end{aligned}$$

Thus, we have the lemma. □

**Procedure 3.1:** A simplified HT for an arbitrary $GT(X : A)$ function can be derived as follows:

1. Use Theorem 3.1 to decompose the function.
2. In $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$, if there is only a single 0 between two groups of consecutive 1's in $\vec{d}_i$, and the groups of consecutive 1's have two or more 1's in $\vec{d}_i$, then represent $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$ by a product using Lemma 2.2.
3. In $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$, if there are a group or groups of consecutive 0's which are separated by a single 1 among each group of consecutive 0's in $\vec{d}_i$, and there exists at least one group that has more than one 0's in $\vec{d}_i$, then represent $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$ by an HT using Lemma 3.2 and the factors can be reduced by Lemma 3.6.
4. Otherwise, represent $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$ by a product using Lemma 2.2.

*Explanation:* 1) We expand $GT(X : A)$ by Theorem 3.1. 2) If the condition satisfies, we use Lemma 2.2 to represent each $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$, because it requires only a single product: $\vec{d}_i$ has one bit different from $\vec{e}_i$ i.e., $\vec{d}_i$ has one zero extra, thus each product in $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$ will cancel each other except for one product. Note that, by Lemma 3.2, it requires two factors. 3) If this condition satisfies, every $\overline{GT(X : INT(\vec{e}_i))} \cdot GT(X : INT(\vec{d}_i))$ can be represented by Lemma 3.2 and, every $\bar{h}_k$ and $g_{k-1}$ that satisfy Lemma 3.6 can be reduced into a factor. 4) If each group of consecutive 0's has only a single 0, then using Lemma 3.2 will cost more factors than that of by Lemma 2.2. Although, we can reduce some factors by Lemma 3.6, it still requires one factor more than that of by Lemma 2.2. The argument for $LT(X : B)$

function is similar.

If there exists only one group of consecutive 0's or 1's in $\vec{a}$ or $\vec{b}$, then Lemma 3.2 or Lemma 3.3 can be used to represent a function by an HT. Otherwise, when multiple groups of consecutive 0's or 1's exist in $\vec{a}$ or $\vec{b}$, Procedure 3.1 can be used to represent a function by an HT.

## 3.2 Examples of Head-Tail Expressions for Interval Functions

**Example 3.8:** Represent $IN_0(X : 0, 15)$ using an HT. The interval function can be represented by:

$$IN_0(X : 0, 15) = GT(X : 0) \cdot LT(X : 15).$$

Since the largest value is 15, we have $n = 4$. Binary representations of $A = A' = 0$ and $B = B' = 15$ are $\vec{a} = \vec{a}' = (0, 0, 0, 0)$ and $\vec{b} = \vec{b}' = (1, 1, 1, 1)$, respectively. By Lemma 3.2, we have $m = 4$, $d = 4$, and $\vec{e} = (1, 1, 1, 1)$:

$$\overline{GT(X : INT(\vec{e}))} \cdot GT(X : INT(\vec{a})) = GT(X : INT(\vec{a}))$$
$$= \overline{\left(\bigwedge_{i=3}^{0} \bar{x}_i\right)} \cdot (1) = \overline{(\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0)} \cdot (1).$$

By Lemma 3.3, we have $m = 4$ and $d = 4$, and $\vec{e} = (0, 0, 0, 0)$:

$$\overline{LT(X : INT(\vec{e}))} \cdot LT(X : INT(\vec{b})) = LT(X : INT(\vec{b}))$$
$$= \overline{\left(\bigwedge_{i=3}^{0} x_i\right)} \cdot (1) = \overline{(x_3x_2x_1x_0)} \cdot (1).$$

Finally, we have:

$$IN_0(X : 0, 15) = \overline{(\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0)} \cdot \overline{(x_3x_2x_1x_0)} \cdot (1).$$

The maps for $IN_0(X : 0, 15)$ are shown in Fig. 5. The top row shows the PreSOP, which requires 6 products. The bottom row shows the HT, which requires only 3 factors. This expression still needs a product to represent the universe, which is indicated by the constant 1 in the bottom row of Fig. 5. Table 3 shows realizations of the function, where the TCAM stores the words and the RAM stores the actions. Table 3(a) corresponds to the top row of Fig. 5, while Table 3(b) corresponds to the bottom row of Fig. 5. ∎

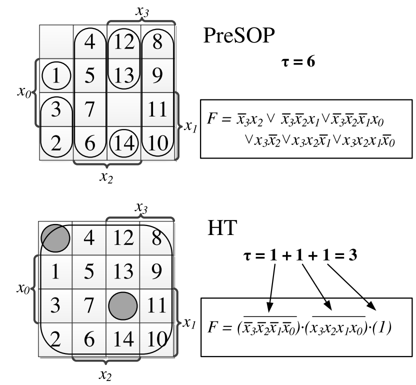



**Fig. 5** Maps for the PreSOP and the HT representing $IN_0(X : 0, 15)$.

**Table 3** Realization of $IN_0(X : 0, 15)$ by TCAM and RAM.

(a)

| TCAM | RAM |
|---|---|
| 0001 | 1 |
| 001* | 1 |
| 01** | 1 |
| 10** | 1 |
| 110* | 1 |
| 1110 | 1 |
| **** | 0 |

(b)

| TCAM | RAM |
|---|---|
| 0000 | 0 |
| 1111 | 0 |
| **** | 1 |

**Example 3.9:** Represent $IN_0(X : 0, 27)$ by an HT. Binary representations of $A = 0$ and $B = 27$ are $\vec{a} = (0, 0, 0, 0, 0)$ and $\vec{b} = (1, 1, 0, 1, 1)$, respectively. To represent the function, we use Lemma 3.4:

$$IN_0(X : 0, 27) = \overline{LT(X : 1)} \cdot LT(X : 27).$$

By Lemma 2.2, we know that $\overline{LT(X : 1)} = \overline{(\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0)}$. The next step is to derive $LT(X : 27)$. $\vec{b}$ has two consecutive groups of 1's and a single isolated 0 between them. By Theorem 3.2, first group of consecutive 1's starts from the index one where $\vec{b}'_0 = (0, 0, 0, 1, 1)$ and $\vec{e}_0 = \vec{b}_1 = \vec{b} \wedge \overline{\vec{b}'_0} = (1, 1, 0, 0, 0)$, while the second group starts from index four where $\vec{b}'_1 = (1, 1, 0, 0, 0)$ and $\vec{e}_1 = \vec{e}_0 \wedge \overline{\vec{b}'_1} = (0, 0, 0, 0, 0)$. By Lemma 3.3, we can represent the first group of consecutive 1's by an HT:

$$\overline{LT(X : INT(\vec{e}_0))} \cdot LT(X : INT(\vec{b}_0))$$
$$= \overline{\left(\bigwedge_{j=4}^{2} x_j^{b_j} \bigwedge_{i=1}^{0} x_i\right)} \cdot \bigwedge_{j=4}^{2} x_j^{b_j} = \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot (x_4x_3\bar{x}_2).$$

And, the second group of consecutive 1's can be represented by an HT:

$$\overline{LT(X : INT(\vec{e}_1))} \cdot LT(X : INT(\vec{b}_1))$$
$$= \overline{\left(\bigwedge_{i=4}^{3} x_i\right)} \cdot (1) = \overline{(x_4x_3)} \cdot (1).$$

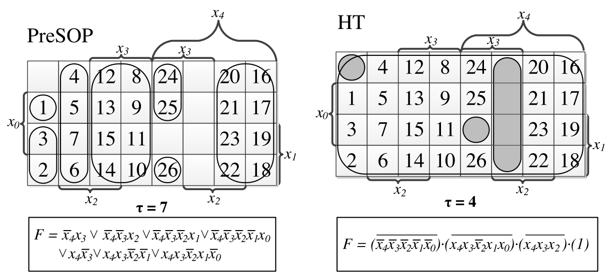
Note that $LT(X : INT(\vec{e}_1)) = 0$. Moreover, by Lemmas 3.5 and 3.6, we can reduce a factor such that:

$$LT(X : 27) = \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot (x_4x_3\bar{x}_2) \vee \overline{(x_4x_3)} \cdot (1)$$
$$= \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot \overline{(x_4x_3x_2)} \cdot (1).$$

The reduced HT for the interval function is

$$IN_0(X : 0, 27) = \overline{(\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0)} \cdot \overline{(x_4x_3\bar{x}_2x_1x_0)}$$
$$\cdot \overline{(x_4x_3x_2)} \cdot (1).$$

Figure 6 shows the maps for $IN_0(X : 0, 27)$. The top




(a) (b)

**Fig. 6** Maps for the PreSOP and the HT representing $IN_0(X:0,27)$.

**Table 4** Realization of $IN_0(X:0,27)$ in TCAM and RAM.

| TCAM | RAM |
|---|---|
| 00000 | 0 |
| 11011 | 0 |
| 111** | 0 |
| ***** | 1 |

row shows the PreSOP which requires 7 products. The bottom row shows the HT which requires only four factors. Table 4 shows the realization of the function by using a TCAM and a RAM with four words. ■

**Example 3.10:** Let $\vec{d}_0=(0,1,0,1,1,0,0,1,0,0,0,1,1,0)$. Represent $GT(X:INT(\vec{d}_0))$ by an HT. In this case, we have $n=14$, and $\vec{d}_0$ has five separate groups of consecutive 0's. To extract the consecutive 0's in the LSB, we select the vector $\vec{d}'_0=(1,1,1,1,1,1,1,1,1,1,1,1,1,0)$. The 0-extraction vector is $\vec{e}_0=\vec{d}_1=\vec{d}_0\vee\overline{\vec{d}'_0}=(0,1,0,1,1,0,0,1,0,0,0,1,1,1)$. Thus, by Theorem 3.1 (Procedure 3.1, Step 1), we have the following representation:

$$\begin{aligned}GT(X:INT(\vec{d}_0))&=\overline{GT(X:INT(\vec{d}_1))}\\&\quad\cdot GT(X:INT(\vec{d}_0))\vee GT(X:INT(\vec{d}_1)).\end{aligned}$$

Since, there is only a single 0 at the least significant bit, and two 1's in the more significant bits, we use Lemma 2.2 (Procedure 3.1, Step 2). Thus, $\overline{GT(X:INT(\vec{d}_1))}\cdot GT(X:INT(\vec{d}_0))=(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6\bar{x}_5\bar{x}_4\bar{x}_3x_2x_1x_0)$. Next, we go to the higher group of consecutive 0's in $\vec{d}_1$ and we select the vector $\vec{d}'_1=(1,1,1,1,1,1,1,1,0,0,0,1,1,1)$. Then, the 0-extraction vector is $\vec{e}_1=\vec{d}_2=\vec{d}_1\vee\overline{\vec{d}'_1}=(0,1,0,1,1,0,0,1,1,1,1,1,1,1)$. Thus, we have the following representation:

$$\begin{aligned}GT(X:INT(\vec{d}_1))&=\overline{GT(X:INT(\vec{d}_2))}\\&\quad\cdot GT(X:INT(\vec{d}_1))\vee GT(X:INT(\vec{d}_2)).\end{aligned}$$

Since, the number of 0's is three, we can represent it by Lemma 3.2 (Procedure 3.1, Step 3). In this case, we have $\overline{GT(X:INT(\vec{d}_2))}\cdot GT(X:INT(\vec{d}_1))=\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6\bar{x}_5\bar{x}_4\bar{x}_3)}\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6)$, where $m=6$ and $d=3$. Next, we select the vector $\vec{d}'_2=(1,1,1,1,1,0,0,1,1,1,1,1,1,1)$. Then, the 0-extraction vector is $\vec{e}_2=\vec{d}_3=\vec{d}_2\vee\overline{\vec{d}'_2}=(0,1,0,1,1,1,1,1,1,1,1,1,1,1)$. Thus, we have the following representation:

$$\begin{aligned}GT(X:INT(\vec{d}_2))&=\overline{GT(X:INT(\vec{d}_3))}\\&\quad\cdot GT(X:INT(\vec{d}_2))\vee GT(X:INT(\vec{d}_3)).\end{aligned}$$

Since, the number of 0's is two, we can represent it by Lemma 3.2 (Procedure 3.1, Step 3). In this case, we have $\overline{GT(X:INT(\vec{d}_3))}\cdot GT(X:INT(\vec{d}_2))=\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7)}\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9)$, where $m=9$ and $d=2$. According to Procedure 3.1, Step 3, we can reduce the factors by Lemma 3.6, such that

$$\begin{aligned}&\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6\bar{x}_5\bar{x}_4\bar{x}_3)}\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6)\\&\vee\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7)}\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9)\\&=\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7)}\cdot\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7\bar{x}_6)}\\&\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9).\end{aligned}$$

Next, the vector $\vec{d}'_3=(1,1,0,1,1,1,1,1,1,1,1,1,1,1)$ is selected. Then, the 0-extraction vector is $\vec{e}_3=\vec{d}_4=\vec{d}_3\vee\overline{\vec{d}'_3}=(0,1,1,1,1,1,1,1,1,1,1,1,1,1)$. Thus, we have the following representation:

$$\begin{aligned}GT(X:INT(\vec{d}_3))&=\overline{GT(X:INT(\vec{d}_4))}\\&\quad\cdot GT(X:INT(\vec{d}_3))\vee GT(X:INT(\vec{d}_4)).\end{aligned}$$

And finally, the vector $\vec{d}'_4=(0,1,1,1,1,1,1,1,1,1,1,1,1,1)$ is selected. Then, the 0-extraction vector is $\vec{e}_4=\vec{d}_5=\vec{d}_4\vee\overline{\vec{d}'_4}=(1,1,1,1,1,1,1,1,1,1,1,1,1,1)$. Thus, we have the following representation:

$$\begin{aligned}GT(X:INT(\vec{d}_4))&=\overline{GT(X:INT(\vec{d}_5))}\\&\quad\cdot GT(X:INT(\vec{d}_4))\vee GT(X:INT(\vec{d}_5)).\end{aligned}$$

In this case, we find two groups of consecutive 0's which are separated by a single 1 in $\vec{d}_3$, but each group has only a 0, thus, according to Procedure 3.1 Step 4, they can be represented by Lemma 2.2: $\overline{GT(X:INT(\vec{d}_4))}\cdot GT(X:INT(\vec{d}_3))=(\bar{x}_{13}x_{12}x_{11})$ and $\overline{GT(X:INT(\vec{d}_5))}\cdot GT(X:INT(\vec{d}_4))=(x_{13})$. Note that $GT(X:INT(\vec{d}_5))=0$. Finally, the HT for $GT(X:INT(\vec{d}_0))$ is

$$\begin{aligned}&(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7x_6\bar{x}_5\bar{x}_4\bar{x}_3x_2x_1x_0)\\&\vee\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7)}\cdot\overline{(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9\bar{x}_8\bar{x}_7\bar{x}_6)}\\&\cdot(\bar{x}_{13}x_{12}\bar{x}_{11}x_{10}x_9)\vee(\bar{x}_{13}x_{12}x_{11})\vee(x_{13}).\end{aligned}$$

In this case, the HT requires only 6 factors, while the PreSOP requires 8 products. ■

### 3.3 The Number of Factors to Represent an Interval Function by a Head-Tail Expression

**Definition 3.6:** Let $\zeta(f)$ be the minimum number of factors to represent a function $f$ by an HT.

From here, we assume that $X=(x_{n-1},x_{n-2},\ldots,x_1,x_0)$.

**Lemma 3.7:**

$$\zeta(GT(X:A)) \leq \left\lceil \frac{n+1}{2} \right\rceil, \text{and}$$

$$\zeta(LT(X:B)) \leq \left\lceil \frac{n+1}{2} \right\rceil.$$

*Proof:* Consider a binary representation $\vec{a}$ that makes the number of factors in an HT for a $GT$ function maximum. If there is three or more consecutive 0's in $\vec{a}$, then we can reduce the number of factors in the HT, by Theorem 3.1. Note that when more than one groups of consecutive 0's exist in arbitrary location in $\vec{a}$, we can use Theorem 3.1 to segment each group to form an HT by Procedure 3.1.

According to the third argument of Procedure 3.1, regardless the number of 0's in each group, if there are $p$ groups, then the number of factors is $p + 1$. For instance, if we have a group with two or more consecutive 0's, the number of factors is $p + 1 = 2$. Note that when only a group with two consecutive 0's exists, the number of factors is not reduced by Procedure 3.1. So, to avoid such reduction of the factors and to get the maximum number of factors in an HT, one possibility is by alternating 0 and 1 in the binary representation (the second argument of Procedure 3.1). Another possibility is by alternating two consecutive 0's and two consecutive 1's in the binary representation[†]. In these cases, we have at least $\lceil \frac{n}{2} \rceil$ zeros, and their numbers of factors in HTs are the same as the numbers of products in Pre-SOPs which are $\lceil \frac{n}{2} \rceil$. Thus, the maximum number of factors in an HT for a $GT$ function occurs when the number of 0's is equal to or greater than $\lceil \frac{n}{2} \rceil$, and Procedure 3.1 cannot be used to reduce the number of factors. The argument for $LT$ functions is similar.

**When $n$ is odd:** Suppose that the number of the factors to represent a $GT$ function takes its maximum when $\vec{a} = (0, 1, 0, 1, \cdots, 1, 0)$. The number of 0's is $\frac{n+1}{2}$, the number of 1's is $\frac{n-1}{2}$, and no consecutive 0's exist.

By Lemma 2.2, the number of products for $GT$ is bounded above by $\sum \bar{a}_i$. So, when the number of 0's is equal to or less than $\frac{n+1}{2}$, the lemma holds. When the number of 0's is greater than $\frac{n+1}{2}$, there exist consecutive 0's in the sequence of $a_i$. In this case, we can apply Procedure 3.1 to reduce the number of factors to construct $GT$. In both cases, the number of factors does not exceed $\frac{n+1}{2}$. Thus, we have

$$\zeta(GT(X:A)) \leq \frac{n+1}{2}.$$

The argument for the number of the factors for an $LT$ function is similar. When $\vec{b} = (1, 0, 1, 0, \cdots, 0, 1)$, the number of 1's is $\frac{n+1}{2}$, the number of 0's is $\frac{n-1}{2}$, and no consecutive 1's exist. When the number of 1's is equal to or greater than $\frac{n+1}{2}$, Procedure 3.1 is used to reduce the number of factors, we have

$$\zeta(LT(X:B)) \leq \frac{n+1}{2}.$$

**When $n$ is even:** When $\vec{a} = (0, 1, 0, 1, \cdots, 0, 1)$, the numbers of 0's and 1's in $GT$ are the same which are $\frac{n}{2}$. However, this does not make the number of factors in an HT maximum which is $\frac{n}{2} + 1$. The number of factors becomes its maximum when $\vec{a} = (0, 1, 0, 1, \cdots, 0, 1, 0, 0)$. The last two components are $a_1 = a_0 = 0$ $(d = 2)$ that produce two factors as explained in Lemma 3.2. Thus, we have

$$\zeta(GT(X:A)) \leq \frac{n}{2} + 1.$$

Likewise, for $LT$, the number of factors becomes its maximum when $\vec{b} = (1, 0, 1, 0, \cdots, 1, 0, 1, 1)$. Thus, we have

$$\zeta(LT(X:B)) \leq \frac{n}{2} + 1.$$

Combining these two cases, we have the lemma. □

To derive a main theorem, we need the following:

**Lemma 3.8:** If $\alpha \subseteq \bar{x}$ and $\beta \subseteq x$, then

$$\bar{\alpha}\bar{x} \vee \bar{\beta}x = \bar{\alpha}\bar{\beta}.$$

*Proof:* Note that $\bar{\alpha} \supseteq x$ and $\bar{\beta} \supseteq \bar{x}$.

$$\begin{aligned}
\bar{\alpha}\bar{\beta} &= (\bar{\alpha} \vee x)(\bar{\beta} \vee \bar{x}) \\
&= \bar{\alpha}\bar{\beta} \vee \bar{x}\bar{\alpha} \vee x\bar{\beta} \\
&= (x \vee \bar{x})\bar{\alpha}\bar{\beta} \vee \bar{x}\bar{\alpha} \vee x\bar{\beta} \\
&= x\bar{\alpha}\bar{\beta} \vee \bar{x}\bar{\alpha}\bar{\beta} \vee \bar{x}\bar{\alpha} \vee x\bar{\beta} \\
&= x\bar{\beta} \vee \bar{x}\bar{\alpha} \vee \bar{x}\bar{\alpha} \vee x\bar{\beta} \\
&= \bar{x}\bar{\alpha} \vee x\bar{\beta}.
\end{aligned}$$

Thus, we have the lemma. □

Lemma 3.7 can be extended to an interval function:

**Theorem 3.3:**

$$\zeta(IN_0(X:A,B)) \leq n$$

*Proof:* Exhaustive examination shows that the theorem holds for $n \leq 4$. Let $\vec{a} = (a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$ and $\vec{b} = (b_{n-1}, b_{n-2}, \cdots, b_1, b_0)$ be binary representations of $A$ and $B$, respectively, and $A < B$. According to Theorem 2.1, if the most significant bits (MSBs) are the same, then we can ignore the MSBs and consider the function with fewer variables. Assume that the MSB is different, i.e., $a_{n-1} = 0$ and $b_{n-1} = 1$. The function can be expanded into

$$IN_0(X:A,B) = \bar{x}_{n-1}GT(\hat{X}:A) \vee x_{n-1}LT(\hat{X}:\hat{B}),$$

where $\hat{B} = B - 2^{n-1}$ and $\hat{X} = (x_{n-2}, \ldots, x_1, x_0)$.

**When $n$ is odd:** Let $a_{n-1} = 0$ and $b_{n-1} = 1$. Consider the case when $\zeta(GT(\hat{X}:A))$ and $\zeta(LT(\hat{X}:\hat{B}))$ take their maximum values. Note that, according to Theorem

[†]The numbers of factors in HTs for $GT$ functions become maximum for various cases. Other cases occur when the binary representations are the combination of alternating 0, 1, two consecutive 0's, and two consecutive 1's in which the numbers of 0's are equal to or greater than $\lceil \frac{n}{2} \rceil$ and Procedure 3.1 cannot be used to reduce the numbers of factors.

2.1, we only consider bits after the $s$'th of the binary representation such that $a_s \neq b_s$. By Lemma 3.7 (the *even* part), the vectors are $\vec{a} = (0, 0, 1, 0, 1, \cdots, 0, 1, 0, 0)$ and $\vec{b} = (1, 1, 0, 1, 0, \cdots, 1, 0, 1, 1)$, where $s = n - 1$. In this case, we can apply Procedure 3.1 to obtain $GT$ and $LT$ functions. Note that when we apply Lemma 3.6 in Procedure 3.1, there are literals of the same variable in both HTs which are $g_1 = \bar{x}_{n-1}$ and $g_2 = x_{n-1}$. We have $(h_{1_p} \vee h_{1_{p-1}} \vee \ldots \vee h_{1_1}) \subseteq g_1$ and $(h_{2_q} \vee h_{2_{q-1}} \vee \ldots \vee h_{2_1}) \subseteq g_2$. By Lemma 3.8, we can combine the literals of both tail factors to form one factor as follows:

$$\begin{aligned}(\bar{h}_{1_p}\bar{h}_{1_{p-1}} \cdots \bar{h}_{1_1})\bar{x}_{n-1} \vee (\bar{h}_{2_q}\bar{h}_{2_{q-1}} \cdots \bar{h}_{2_1})x_{n-1} \\ = (\bar{h}_{1_p}\bar{h}_{1_{p-1}} \cdots \bar{h}_{1_1}) \cdot (\bar{h}_{2_q}\bar{h}_{2_{q-1}} \cdots \bar{h}_{2_1}) \cdot (1).\end{aligned} \qquad (2)$$

Thus, by Lemma 3.7, we have

$$\begin{aligned}\zeta(IN_0(X : A, B)) &\leq \zeta(GT(\hat{X} : A)) + \zeta(LT(\hat{X} : \hat{B})) \\ &\leq \left\lceil \frac{n}{2} \right\rceil + \left\lceil \frac{n}{2} \right\rceil = n + 1.\end{aligned}$$

Moreover, by Eq. (2), we have

$$\zeta(IN_0(X : A, B)) \leq n + 1 - 1 = n.$$

**When $n$ is even:** The HTs for both $GT$ and $LT$ functions contribute and we have

$$\begin{aligned}\zeta(IN_0(X : A, B)) &\leq \zeta(GT(\hat{X} : A)) + \zeta(LT(\hat{X} : \hat{B})) \\ &\leq \left\lceil \frac{(n-1)+1}{2} \right\rceil + \left\lceil \frac{(n-1)+1}{2} \right\rceil = n.\end{aligned}$$

Thus, we have the theorem. □

Next, we give an example of HTs for interval functions that require the maximum number of factors.

**Example 3.11:** When $n = 5$, there exist several interval functions that have the maximum number of factors in HTs. HTs for these functions can be reduced by Lemma 3.6. Table 5 shows these functions and their endpoints represented by binary numbers.

First, we expand the functions by:

$$IN_0(X : A, B) = \bar{x}_{n-1}GT(\hat{X} : A) \vee x_{n-1}LT(\hat{X} : \hat{B}),$$

where $\hat{B} = B - 2^{n-1}$ and $\hat{X} = (x_{n-2}, \ldots, x_1, x_0)$. Then, we use Procedure 3.1 to represent the $GT$ and the $LT$ functions by HTs. Thus, we have:

$$\begin{aligned}f_1 &= (\bar{x}_4\bar{x}_3\bar{x}_2x_1x_0) \vee \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot (\bar{x}_4) \\ &\quad \vee \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot \overline{(x_4x_3x_2)} \cdot (x_4), \\ f_2 &= (\bar{x}_4\bar{x}_3\bar{x}_2x_1x_0) \vee (x_4x_3x_2\bar{x}_1\bar{x}_0) \\ &\quad \vee \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot (\bar{x}_4) \vee \overline{(x_4x_3x_2)} \cdot (x_4), \\ f_3 &= \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot (\bar{x}_4) \\ &\quad \vee \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot \overline{(x_4x_3x_2)} \cdot (x_4), \\ f_4 &= \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot (\bar{x}_4) \\ &\quad \vee (x_4x_3x_2\bar{x}_1\bar{x}_0) \vee \overline{(x_4x_3x_2)} \cdot (x_4).\end{aligned}$$

**Table 5** Endpoints of interval functions that have the maximum number of factors in HTs for $n = 5$.

| Function | $\vec{a}$ | $\vec{b}$ |
|---|---|---|
| $f_1 = IN_0(X : 2, 27)$ | $(0, 0, 0, 1, 0)$ | $(1, 1, 0, 1, 1)$ |
| $f_2 = IN_0(X : 2, 29)$ | $(0, 0, 0, 1, 0)$ | $(1, 1, 1, 0, 1)$ |
| $f_3 = IN_0(X : 4, 27)$ | $(0, 0, 1, 0, 0)$ | $(1, 1, 0, 1, 1)$ |
| $f_4 = IN_0(X : 4, 29)$ | $(0, 0, 1, 0, 0)$ | $(1, 1, 1, 0, 1)$ |

Since $n$ is odd, the number of factors for each function is $\lceil \frac{5}{2} \rceil + \lceil \frac{5}{2} \rceil = 3 + 3 = 6$.

In this case, from $f_1$, we have $\bar{\alpha}_1 = \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)}$ and $\bar{\beta}_1 = \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot \overline{(x_4x_3x_2)}$. From $f_2$, we have $\bar{\alpha}_2 = \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)}$ and $\bar{\beta}_2 = \overline{(x_4x_3x_2)}$. From $f_3$, we have $\bar{\alpha}_3 = \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)}$ and $\bar{\beta}_3 = \overline{(x_4x_3\bar{x}_2x_1x_0)} \cdot \overline{(x_4x_3x_2)}$. And from $f_4$, we have $\bar{\alpha}_4 = \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)}$ and $\bar{\beta}_4 = \overline{(x_4x_3x_2)}$.

By Lemma 3.8, we can reduce the factors:

$$\begin{aligned}f_1 &= (\bar{x}_4\bar{x}_3\bar{x}_2x_1x_0) \vee \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot \overline{(x_4x_3\bar{x}_2x_1x_0)} \\ &\quad \cdot \overline{(x_4x_3x_2)} \cdot (1), \\ f_2 &= (\bar{x}_4\bar{x}_3\bar{x}_2x_1x_0) \vee (x_4x_3x_2\bar{x}_1\bar{x}_0) \\ &\quad \vee \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot \overline{(x_4x_3x_2)} \cdot (1), \\ f_3 &= \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \cdot \overline{(x_4x_3\bar{x}_2x_1x_0)} \\ &\quad \cdot \overline{(x_4x_3x_2)} \cdot (1), \\ f_4 &= (x_4x_3x_2\bar{x}_1\bar{x}_0) \vee \overline{(\bar{x}_4\bar{x}_3x_2\bar{x}_1\bar{x}_0)} \cdot \overline{(\bar{x}_4\bar{x}_3\bar{x}_2)} \\ &\quad \cdot \overline{(x_4x_3x_2)} \cdot (1).\end{aligned}$$

Note that each function has five factors. ■

## 4. Experimental Results

We developed a heuristic algorithm [14] to generate HTs for interval functions that uses the properties of Procedure 3.1. By the computer program, we represented all the interval functions for $n = 1$ to $n = 16$ by HTs. There are $N(n) = (2^n+1)(2^{n-1})$ distinct interval functions of $n$ variables. When $n = 16$, the total number of the distinct interval functions is approximately $2^{31} \approx 2.147 \times 10^9$.

Table 6 shows the distribution of $GT$ or $LT$ functions

**Table 6** Numbers of $GT(X : A)$ or $LT(X : B)$ functions requiring $\tau$ factors in HTs for $n = 1$ to $n = 16$ produced by a heuristic algorithm.

| $n$ | # Factors ($\tau$) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 2 | | | | | | | | |
| 2 | 3 | 1 | | | | | | | |
| 3 | 4 | 4 | | | | | | | |
| 4 | 5 | 9 | 2 | | | | | | |
| 5 | 6 | 16 | 10 | | | | | | |
| 6 | 7 | 25 | 28 | 4 | | | | | |
| 7 | 8 | 36 | 60 | 24 | | | | | |
| 8 | 9 | 49 | 110 | 80 | 8 | | | | |
| 9 | 10 | 64 | 182 | 200 | 56 | | | | |
| 10 | 11 | 81 | 280 | 420 | 216 | 16 | | | |
| 11 | 12 | 100 | 408 | 784 | 616 | 128 | | | |
| 12 | 13 | 121 | 570 | 1344 | 1456 | 560 | 32 | | |
| 13 | 14 | 144 | 770 | 2160 | 3024 | 1792 | 288 | | |
| 14 | 15 | 169 | 1012 | 3300 | 5712 | 4704 | 1408 | 64 | |
| 15 | 16 | 196 | 1300 | 4840 | 10032 | 10752 | 4992 | 640 | |
| 16 | 17 | 225 | 1638 | 6864 | 16632 | 22176 | 14400 | 3456 | 128 |

**Table 7** Numbers of $n$-variable interval functions requiring $\tau$ factors in HTs for $n = 1$ to $n = 16$ produced by a heuristic algorithm.

| $n$ | # Factors ($\tau$) | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 1 | 3 | | | | | | | | | | | | | | | |
| 2 | 7 | 3 | | | | | | | | | | | | | | |
| 3 | 15 | 16 | 5 | | | | | | | | | | | | | |
| 4 | 31 | 51 | 42 | 12 | | | | | | | | | | | | |
| 5 | 63 | 132 | 181 | 124 | 28 | | | | | | | | | | | |
| 6 | 127 | 307 | 574 | 644 | 364 | 64 | | | | | | | | | | |
| 7 | 255 | 672 | 1537 | 2384 | 2240 | 1024 | 144 | | | | | | | | | |
| 8 | 511 | 1419 | 3714 | 7220 | 9504 | 7424 | 2784 | 320 | | | | | | | | |
| 9 | 1023 | 2932 | 8405 | 19212 | 32204 | 35968 | 23520 | 7360 | 704 | | | | | | | |
| 10 | 2047 | 5979 | 18222 | 46844 | 93996 | 136016 | 129408 | 71744 | 19008 | 1536 | | | | | | |
| 11 | 4095 | 12096 | 38401 | 107504 | 247200 | 435200 | 544816 | 445504 | 211904 | 48128 | 3328 | | | | | |
| 12 | 8191 | 24355 | 79426 | 236444 | 603152 | 1236272 | 1910016 | 2080768 | 1476288 | 608768 | 119808 | 7168 | | | | |
| 13 | 16383 | 48900 | 162277 | 504700 | 1393148 | 3217408 | 5866784 | 7980416 | 7620096 | 4731904 | 1707264 | 293888 | 15360 | | | |
| 14 | 32767 | 98019 | 328926 | 1054932 | 3090572 | 7840416 | 16323584 | 26503616 | 31901824 | 26889216 | 14729728 | 4687872 | 711680 | 32768 | | |
| 15 | 65535 | 196288 | 663329 | 2173104 | 6655392 | 18175744 | 42109520 | 78930432 | 114450048 | 122574848 | 91806208 | 44679168 | 12634112 | 1703936 | 69632 | |
| 16 | 131071 | 392859 | 1333410 | 4431844 | 14023392 | 40562080 | 102439456 | 216008512 | 364880640 | 474360832 | 454498304 | 304347136 | 132431872 | 33488896 | 4038656 | 147456 |

**Table 8** Average numbers of factors to represent $n$-variable interval functions by HTs (near minimum) and exact MSOPs for $n = 1$ to $n = 16$.

| $n$ | Head-Tail Expression | | MSOP | Ratio ($\rho(n)$) |
|---|---|---|---|---|
| | $\mu_h(n)$ | $\frac{(\frac{2}{3}n-\frac{5}{9})}{\mu_h(n)}$ | $\mu_s(n)$ | $\frac{\mu_h(n)}{\mu_s(n)}$ |
| 1 | 1 | 0.1111 | 1 | 1 |
| 2 | 1.3 | 0.5983 | 1.3 | 1 |
| 3 | 1.7222 | 0.8387 | 1.7778 | 0.97 |
| 4 | 2.2574 | 0.9352 | 2.3971 | 0.94 |
| 5 | 2.8523 | 0.9739 | 3.1288 | 0.91 |
| 6 | 3.4822 | 0.9892 | 3.9433 | 0.88 |
| 7 | 4.1301 | 0.9954 | 4.8154 | 0.86 |
| 8 | 4.7873 | 0.9980 | 5.7267 | 0.84 |
| 9 | 5.4492 | 0.9991 | 6.6645 | 0.82 |
| 10 | 6.1135 | 0.9996 | 7.6203 | 0.80 |
| 11 | 6.7790 | 0.9998 | 8.5886 | 0.79 |
| 12 | 7.4450 | 0.9999 | 9.5654 | 0.78 |
| 13 | 8.1114 | 0.9999 | 10.5487 | 0.77 |
| 14 | 8.7779 | 0.9999 | 11.5362 | 0.76 |
| 15 | 9.4445 | 0.9999 | - | - |
| 16 | 10.1111 | 0.9999 | - | - |

that require $\tau$ factors in HTs for up to $n = 16$ produced by the heuristic algorithm [14]. As shown in the table, to represent a $GT$ or an $LT$ function, at most $\frac{n+1}{2}$ factors are necessary when $n$ is odd, and at most $\frac{n}{2} + 1$ factors are necessary when $n$ is even. For $GT$ and $LT$ functions, the heuristic program generates exact minimum HTs.

Table 7 shows the distribution of interval functions that require $\tau$ factors in HTs for up to $n = 16$ produced by the heuristic algorithm. It shows that with an HT, any interval functions can be represented with at most $n$ factors.

Let $\mu_h(n)$ be the average number of factors to represent $n$-variable interval functions by HTs produced by the heuristic algorithm. Table 8 shows $\mu_h(n)$ for $n = 1$ to $n = 16$. We represented all the interval functions by HTs generated by the heuristic algorithm [14]. Thus, they may not be minimum. Since $(\frac{2}{3}n - \frac{5}{9})/\mu_h(n)$ approaches to 1.00 with the increase of $n$, we have the following:

**Conjecture 4.1:** For sufficiently large $n$, the average number of factors to represent $n$-variable interval functions is at most $\frac{2}{3}n - \frac{5}{9}$.

We also obtained $\mu_s(n)$, the average numbers of products to represent $n$-variable interval functions by exact MSOPs, by using exact algorithm for $n = 1$ to $n = 14$. The fourth column of Table 8 shows values of $\mu_s(n)$. The first experiment, for $n = 1$ to $n = 13$, we used Intel Dual2Duo 3.0 GHz microprocessor with 8 GB memory. We generated all the interval functions and minimized them using ESPRESSO-EXACT [2] which obtains exact minimum SOPs. For $n = 13$, to obtain $\mu_s(13)$, it took one month. The second one, for $n = 14$, we used Intel Xeon 8-core 2.27 GHz microprocessors with 24 GB memory and paralleled the program into 8 parts, and the computation took nearly a month. By using the same method, for $n = 16$, it would take a few years to obtain $\mu_s(16)$. The rightmost column of Table 8 shows the ratio $\rho(n) = \frac{\mu_h(n)}{\mu_s(n)}$. It shows that $\rho(n)$ decreases with the increment of $n$. The experimental results also show that, for $n \geq 10$, HTs require at least 20% fewer factors than MSOPs, on the average.

Moreover, we can observe interesting sequences in Table 6. Let $C_\tau(n)$ be the value of the $\tau$-th column in Table 6. For $\tau = 1$ to $\tau = 6$, we have:

$$C_1(n) = n + 1$$
$$C_2(n) = (n-1)^2$$
$$C_3(n) = \frac{(n-3)(n-2)(2n-5)}{3}$$
$$C_4(n) = \frac{(n-4)^2[(n-4)^2-1]}{3}$$
$$C_5(n) = \frac{(n-7)(n-6)(n-5)(n-4)(2n-11)}{15}, \text{ and}$$
$$C_6(n) = \frac{4}{90}(n-7)^2[(n-7)^2-1][(n-7)^2-4].$$

The derivation of these formulas are future work.

## 5. Conclusion

In this paper, we introduced head-tail expressions (HTs) to represent interval functions. We showed that HTs efficiently represent $GT$, $LT$ and interval functions. We also showed that a pair of a TCAM and a RAM directly implements an

HT. Finally, we prove that an HT requires at most $n$ factors to represent any interval function $IN_0(X : A, B)$. By a heuristic algorithm, we obtained average numbers of factors to represent interval functions in HTs for up to $n = 16$. And, we conjecture that, for sufficiently large $n$, the average number of factors by HTs to represent $n$-variable interval functions is $\frac{2}{3}n - \frac{5}{9}$. We also show that, for $n \geq 10$, HTs generated by our heuristic program require at least 20% fewer factors than MSOPs, on the average.

## Acknowledgment

This work is partially supported by the Japan Society for the Promotion of Science (JSPS), Grant in Aid for Scientific Research, and by the Adaptable and Seamless Technology Transfer Program through target-driven R&D, JST.

### References

[1] F. Baboescu and G. Varghese, "Scalable packet classification," IEEE/ACM Trans. Netw., vol.13, no.1, pp.2–14, Feb. 2005.
[2] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, MA, 1984.
[3] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, "Packet classifiers in ternary CAMs can be smaller," ACM SIGMETRICS, pp.311–322, June 2006.
[4] J.F. Gimpel, "The minimization of TANT networks," IEEE Trans. Electron. Comput., vol.16, no.1, pp.18–38, Feb. 1967.
[5] P. Gupta and N. McKeown, "Packet classification on multiple fields," ACM SIGCOMM, pp.147–160, Sept. 1999.
[6] R. McGeer and P. Yalagandula, "Minimizing rulesets for TCAM implementation," INFOCOM, pp.1314–1322, April 2009.
[7] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE J. Solid-State Circuits, vol.41, no.3, pp.712–727, March 2006.
[8] O. Rottenstreich and I. Keslassy, "Worst-case TCAM rule expansion," INFOCOM Miniconference, pp.1–5, March 2010.
[9] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.
[10] T. Sasao, "On the complexity of classification functions," ISMVL, pp.57–63, May 2008.
[11] T. Sasao, Memory-Based Logic Synthesis, Springer, 2011.
[12] E. Spitznagel, D. Taylor, and J. Turner, "Packet classification using extended TCAMs," ICNP, pp.120–131, Nov. 2003.
[13] I. Syafalni and T. Sasao, "Head-tail expressions for interval functions," SASIMI 2012, pp.94–99, March 2012.
[14] I. Syafalni and T. Sasao, "A fast head-tail expression generator for TCAM–Application to packet classification," ISVLSI 2012, pp.27–32, Aug. 2012.
[15] I. Syafalni and T. Sasao, "On the numbers of products in prefix SOPs for interval functions," IEICE Trans. Inf. & Syst., vol.E96-D, no.5, pp.1086–1094, May 2013.
[16] D.E. Taylor, "Survey and taxonomy of packet classification techniques," ACM Computing Surveys, vol.37, no.3, pp.238–275, 2005.
[17] H. Yu, "A memory- and time-efficient on-chip TCAM minimizer for IP lookup," DATE, pp.926–931, March 2010.

**Infall Syafalni** was born in 1987. He received B.E. in electrical engineering from Bandung Institute of Technology (ITB), Indonesia in 2008, M.Sc. in electronics engineering from University of Science, Malaysia (USM) in 2010, and Ph.D. in electronics engineering from Kyushu Institute of Technology (KIT), Japan in 2014. He is currently a researcher in the Department of Computer Science and Electronics at Kyushu Institute of Technology, Japan. His research interests include logic design and VLSI design. He is a member of the IEEE.

**Tsutomu Sasao** received the B.E., M.E., and Ph.D. degrees in electronics engineering from Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan; IBM T. J. Watson Research Center, Yorktown Height, NY; the Naval Postgraduate School, Monterey, CA; and Kyushu Institute of Technology, Iizuka, Japan. Now, he is a Professor of Department of Computer Science, Meiji University, Kawasaki, Japan. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published more than nine books on logic design, including *Logic Synthesis and Optimization*, *Representation of Discrete Functions*, *Switching Theory for Logic Synthesis*, *Logic Synthesis and Verification*, and *Memory-Based Logic Synthesis*, in 1993, 1996, 1999, 2001, and 2011, respectively. He has served as Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the NIWA Memorial Award in 1979, Takeda Techno-Entrepreneurship Award in 2001, and Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVLs in 1986, 1996, 2003, 2004 and 2013. He has served as an Associate Editor of the *IEEE Transactions on Computers*. He is a Fellow of the IEEE.