Matching Integer Intervals by Minimal Sets of Binary Words with *don't cares*⋆

Wojciech Fraczak[1,3], Wojciech Rytter[2,4], and Mohammadreza Yazdani[3]

[1] Dépt d'informatique, Université du Québec en Outaouais, Gatineau PQ, Canada
[2] Inst. of Informatics, Warsaw University, Warsaw, Poland
[3] Dept. of Systems and Computer Eng., Carleton University, Ottawa ON, Canada
[4] Department of Mathematics and Informatics, Copernicus University, Torun, Poland

Abstract. We consider n-bit binary encodings of integers. An integer interval $[p, q]$ can be considered as the set X of binary strings corresponding to encodings of all integers in $[p, q]$. The most natural encoding is the usual binary representation of integers (lexicographic encoding) and another useful encoding, considered in the paper, the reflected Gray code. A word w with a *don't care* symbol is matching the set $L(w)$ of all words of the same length which can differ only on positions containing a *don't care*. A set Y of words with *don't cares* is *matching* X iff $X = \bigcup_{w \in Y} L(w)$. For a set X of codes of integers in $[p, q]$ we ask for a minimal size set Y of words with *don't cares* matching X. Such a problem appears in the context of network processing engines using *Ternary Content Addressable Memory* (TCAM) as a lookup table for IP packet header fields. Hence we use the name TCAM for sets of words with *don't cares*. A TCAM representation of intervals has been traditionally performed by a heuristic called "prefix match", which can produce the TCAM of the size approximately twice larger than the minimal one. In this paper we present two fast (linear time in the size of the input and the output) algorithms for finding minimal solutions.

1 Introduction and motivation

The *Ternary Content Addressable Memory* (TCAM), [8, 9], is a type of associative memory with a highly parallel architecture which is used for performing very fast (constant time) table look up operations. The problem of interval representation by TCAM appears in network processing engines where the header fields of each IP packet (e.g., source address, destination address, port number, etc.) should be matched under strict time constraints against the entries of an Access Control List (ACL) [3, 7, 12]. Often, an ACL is represented by a TCAM. Each entry of the ACL defines either a single value or an interval of values for the fields of the packet header. If an ACL entry defines only single values for all header fields, then it can be directly and very efficiently represented using a single TCAM rule. However, if the ACL entry defines some non-trivial intervals

⋆ The research of the second author was supported by the grant of the Polish Ministery of Science and Higher Education N 206 004 32/0806

for some header fields of the packet, then it may need more than one TCAM rule [10, 11].

The problem of finding a minimal TCAM representation of an interval corresponds to the problem of finding a minimal set of words over three letter alphabet $\{0, 1, *\}$ (where $*$ plays a role of *don't care*) covering the interval.

In this paper we show that in the case of lexicographic and reflected Gray encodings the problem can be solved very efficiently. We present two fast (linear time in the size of the input and the output) algorithms for finding minimal solutions.

2 Definition of the problem

A TCAM can be defined as a two dimensional array of cells, where each cell carries one of the three values 0, 1, or $*$. Each row of this array is called a TCAM rule. Examples of TCAMs are shown in Figure 1.

		a)	b)	c)	d)	e)
1		0001	0**1	01**	0*1*	01**
2		001*	10**	1*0*	10**	1**0
3		01**	*10*	*0*1	*1*0	*01*
4		10**	**10	**10	**01	**01
5		110*				
6		1110				

Fig. 1. TCAM representations for interval $[1, 14]$, where the integers are represented in the standard unsigned binary encoding over 4 bits. The first table corresponds to the TCAM generated by the "prefix match" heuristics.

A rule of a TCAM of width n is a sequence $r = e_1 e_2 \dots e_n$, where $e_i \in \{0, 1, *\}$ for $i \in \{1, \dots, n\}$. It defines the following non-empty language $L(r)$:

$$L(r) \stackrel{\texttt{def}}{=} L(e_1)L(e_2)\dots L(e_n),$$

where $L(0) \stackrel{\texttt{def}}{=} \{0\}$, $L(1) \stackrel{\texttt{def}}{=} \{1\}$, and $L(*) \stackrel{\texttt{def}}{=} \{0, 1\}$. For example, $L(\texttt{0*1}) = \{0\}\cdot\{0, 1\}\cdot\{1\} = \{001, 011\}$. A TCAM $\mathcal{R}$ consisting of k rules $r_1, r_2, \dots, r_k$ will be written as $\mathcal{R} = (r_1, r_2, \dots, r_k)$.. The language of $\mathcal{R} = (r_1, r_2, \dots, r_k)$ is the union of the languages defined by its rules, i.e., $L(\mathcal{R}) \stackrel{\texttt{def}}{=} L(r_1) \cup L(r_2) \cup \dots \cup L(r_k)$.

Let $E : \{0, 1\}^n \hookrightarrow \{0, 1, \dots, 2^n - 1\}$ be an encoding of integer values by n-bit strings. Any subset $X \subseteq \{0, 1, \dots, 2^n - 1\}$ can be represented by a TCAM $\mathcal{R}$ of width n. More precisely, $\mathcal{R}$ represents X iff $E^{-1}(X) = L(\mathcal{R})$. For two integers x, y, we denote by $[x, y]$ the set $\{x, x + 1, \dots, y\}$ and call it an *interval*. A set of binary strings X of length n is an *interval-set* of E if $E(X)$ is an interval.

The problem of finding a minimum size TCAM $\mathcal{R}$ (i.e., a TCAM with the minimum number of rules) for a given set X is known to be NP-hard (as it

corresponds to the problem of finding a minimal disjunctive normal form for a Boolean expression). However, in this paper we are interested in sets of binary strings which correspond to the encodings of the numbers in a given interval. A given interval can be represented by several TCAMs. For example, Figure 1 shows 5 TCAM representations for the interval $[1, 14]$ in the case of the lexicographic 4-bit encoding.

3 Notation and preliminary results

Let $\mathcal{R} = (r_1, r_2, \ldots, r_k)$ and $\mathcal{R}' = (p_1, p_2, \ldots, p_m)$ be two TCAMs of the same width d, i.e., $r_i, p_j \in \{0, 1, *\}^d$ for $i \in \{1, 2, \ldots, k\}$ and $j \in \{1, 2, \ldots, m\}$, and $w \in \{0, 1, *\}^n$ be a rule of length n. We define:

$$\mathcal{R} + \mathcal{R}' \stackrel{\texttt{def}}{=} (r_1, r_2, \ldots, r_k, p_1, p_2, \ldots, p_m)\ ; \qquad w \cdot \mathcal{R} \stackrel{\texttt{def}}{=} (wr_1, wr_2, \ldots, wr_k).$$

Intuitively, the TCAM $\mathcal{R} + \mathcal{R}'$ is the union of $\mathcal{R}$ and $\mathcal{R}'$, thus its width remains d and the number of its rules is $k + m$. The TCAM $w \cdot \mathcal{R}$ is of width $d + n$ and each of its rules is the concatenation of w with a rule of $\mathcal{R}$.

For $a \in \{0, 1\}$, by $\overline{a}$ we will denote the *complement* of a, i.e., $\overline{0} \stackrel{\texttt{def}}{=} 1$ and $\overline{1} \stackrel{\texttt{def}}{=} 0$. We define a *full* tree of height n as a perfect binary tree of height n such that each pair of sibling edges are labeled 0 and 1. The assignment of labels to the edges (two alternatives per each internal node) can be chosen arbitrarily. Let T_n be a full tree of height n. The label $w \in \{0, 1\}^n$ of the path from the root to i-th leaf defines the n-bit encoding of number i, with 0 corresponding to the furthest left leaf of the tree. In this way, T_n defines a bijection $T_n : \{0, 1\}^n \hookrightarrow \{0, 1, \ldots, 2^n - 1\}$, which we call *dense-tree encoding*. The lexicographic encoding (i.e., standard unsigned binary encoding) and the binary reflected Gray encoding [5, 6] are two important examples of dense-tree encodings. They are presented in Figure 2 in the forms of full trees. In the context

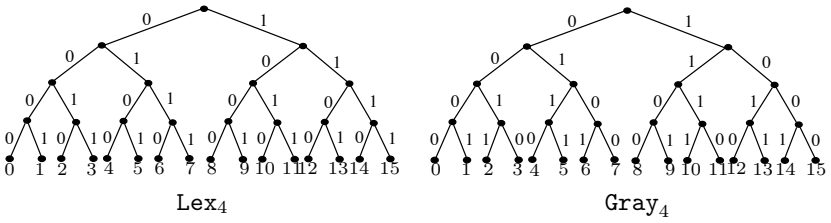


Fig. 2. Two 4-bit dense-tree encodings: the lexicographic encoding ($\texttt{Lex}_4$) and the reflected Gray encoding ($\texttt{Gray}_4$). Notice that in the case of `Gray` every pair of sibling sub-trees are labeled symmetrically.

of a dense-tree encoding T_n, a set $X \subseteq \{0, 1\}^n$ defines both the set $T_n(X)$ of integers and a subset of leaves of T_n. X can be represented by a *skeleton tree*

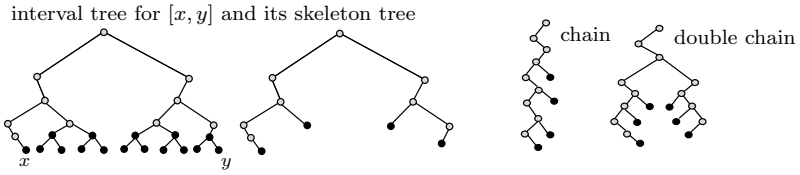


Fig. 3. The skeleton tree for an interval $[x, y]$, a chain, and a double-chain.

(see Figure 3). The skeleton tree of X is obtained from T_n by removing all edges which are not leading to the leaves of X and turning all full sub-trees into leaves.

By $\emptyset$ and $\bullet$ we will represent the empty tree and the single node tree, respectively. Let S be a tree and $w \in \{0,1\}^*$. By $S.w$ we will denote the corresponding sub-tree of S; the root of $S.w$ is the last vertex of the unique path starting from the root of S and labeled by w.

Let $X \subseteq \{0,1\}^n$. A TCAM rule r is called an *X-limited* rule if $L(r) \subseteq X$. An X-limited rule r is said to be *X-essential* if there is no other X-limited TCAM rule r' such that $L(r) \subseteq L(r')$. In the context of two-level logic minimization, an X-essential TCAM rule is called a "prime implicant" of X (see [1]). Any coverage of X by a TCAM with k rules can be turned into a coverage by k X-essential TCAM rules. Therefore, it is quite natural to consider only X-essential TCAM rules in the process of finding a minimal TCAM representation of X; representation of a set in TCAM by essential rules will be called a *canonical representation*.

We say that a skeleton tree S is a *chain*, if every vertex of S has at most one non-leaf child. A *double-chain* is a skeleton tree with at most one vertex v having two non-leaf children and such that all ancestors of v have only one child. Examples of a chain and a double-chain are illustrated in Figure 3. The skeleton tree of any interval in a dense-tree encoding is either a chain or a double-chain. A chain is called *left chain* (resp., *right chain*) if every right (resp., left) child is a unique child or it is a leaf. Intuitively, a left-chain C_L defines an range $[x, 2^n - 1]$, and a right-chain C_R a range $[0, y]$, where n is the width of the dense-tree encoding, and $x, y \in \{0, 1, \ldots, 2^n - 1\}$. We say that C_L and C_R are *complementary chains* if the intervals they define overlap, i.e., if and only if $x \leq y + 1$.

Lemma 1. [4] *Let the skeleton tree of $X \subseteq \{0,1\}^n$ in a dense-tree encoding T_n be a chain C with k leaves. There exists a unique minimal canonical TCAM for X, denoted by $ChainTCAM_n(C)$, with k rules, which can be computed in time $O(kn)$. (See Appendix A.)*

If the skeleton tree of a given interval is a chain C, we can directly use Lemma 1 to calculate its unique minimal canonical TCAM representation, denoted by $ChainTCAM_n(C)$, independently of the dense-tree encoding T_n.

4 Intervals in the lexicographic encoding

Let C_L, C_R be left and right chains, respectively. Denote by $Merge(C_L, C_R)$ the interval skeleton tree which results by creating a new root and connecting C_L to the left child and C_R to the right child of the root (see S' in Figure 7).

ALGORITHM $LexTCAM(S, n)$

1. if S is a chain then **return** $ChainTCAM_n(S)$;
2. if $v = root(S)$ has one child z and the edge $v \to z$ has label $a \in \{0, 1\}$ then **return** $a \cdot LexTCAM(S.a, n-1)$, where $S.a$ is the tree rooted at z;
3. k := number of grandchildren of the root;
   let L, R be the leftmost and rightmost grandchildren of v, and C_L, C_R the left and the right chains rooted in L and R, respectively;
   if $k = 2$ then **return** $01 \cdot ChainTCAM_{n-2}(C_L) + 10 \cdot ChainTCAM_{n-2}(C_R)$;
4. S' := $Merge(C_L, C_R)$; $\mathcal{R}'$:= $LexTCAM(S', n-1)$;
   Split rules of $\mathcal{R}'$ w.r.t. the first symbol, i.e., $\mathcal{R}' = 0 \cdot \mathcal{R}'_0 + 1 \cdot \mathcal{R}'_1 + * \cdot \mathcal{R}'_*$;
   if $k = 3$ and L is a right child then
   return $01 \cdot \mathcal{R}'_0 + 1* \cdot \mathcal{R}'_1 + *1 \cdot \mathcal{R}'_* + (10*^{n-2})$;
   if $k = 3$ and L is a left child then
   return $0* \cdot \mathcal{R}'_0 + 10 \cdot \mathcal{R}'_1 + *0 \cdot \mathcal{R}'_* + (01*^{n-2})$;
5. $(k = 4)$ $\mathcal{R}$:= $0* \cdot \mathcal{R}'_0 + *1 \cdot \mathcal{R}'_1 + ** \cdot \mathcal{R}'_* + (10*^{n-2})$;
   if (C_L, C_R) are *complementary* then **return** $\mathcal{R}$ else **return** $\mathcal{R} + (01*^{n-2})$;

Fig. 4. Algorithm $LexTCAM(S, n)$.

The recursive algorithm $LexTCAM(S, n)$, presented in Figure 4, returns a minimal canonical TCAM for an interval represented by its skeleton tree S in the $\texttt{Lex}_n$ encoding. It employs a *top-down-reduction* approach to reduce the TCAM representation problem to another problem with a smaller skeleton tree.

The algorithm consists of 5 parts which correspond to the treatment of 5 different types of interval skeleton trees described below (see also Figure 5):

ξ_0 – Chains;
ξ_1 – Double-chains whose roots have only one child;
ξ_2 – Double-chains whose roots have two children and two grandchildren;
ξ_3 – Double-chains whose roots have three grandchildren;
ξ_4 – Double-chains whose roots have four grandchildren.

Let us consider an interval $I = [19, 61]$ within $\texttt{Lex}_6$ encoding.

$$\texttt{Lex}_6(010011) = 19 \quad \text{and} \quad \texttt{Lex}_6(111101) = 61$$

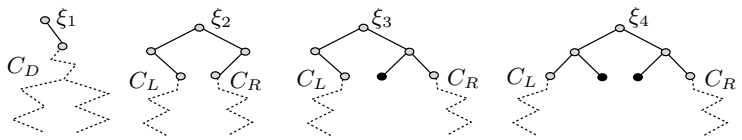


Fig. 5. Double-chains of types ξ_1, ξ_2, ξ_3, and ξ_4. The double-chain C_D, left chain C_L, and right chain C_R are non-empty. Notice that in some cases the chains cannot be trivial (i.e., a single node tree). For example, in ξ_4 both chains C_L and C_R are non-trivial.

The history of the execution of $LexTCAM(S, 6)$, where S is the interval skeleton tree for I, is presented in Figure 6. The final result of the computation is shown in the rightmost column of the table and consists of 6 TCAM rules.

The correctness of the algorithm is checked by analysing all five steps of the algorithm. In the proof we will need the following lemma.

Lemma 2. *Let I and I' be two intervals corresponding to the skeleton trees S and S' of Figure 7, respectively. A minimal TCAM for I has at least k rules more than a minimal TCAM for I', where $k \geq 1$ is the height difference between S and S'. (See Appendix A.)*

Step 1 and 2. For skeleton trees of type ξ_0, i.e., chains, we use the solution provided by Lemma 1. The case of a skeleton tree of type ξ_1 is simple since there is the one-to-one correspondence between all TCAM representations of (the interval of) S and all TCAM representations of (the interval of) $S.a$, where a is the label of the unique edge of the root of S.

Step 3. For the skeleton trees of type ξ_2 we have:

Proposition 1 (Skeleton trees of type ξ_2). *Let S be an interval skeleton tree in $\texttt{Lex}_n$ such that $S.00 = S.11 = \emptyset$, $S.01 = C_L \neq \emptyset$, and $S.10 = C_R \neq \emptyset$. If $\mathcal{R}_L$ is a minimal canonical TCAM for C_L on $n-2$ bits and $\mathcal{R}_R$ is a minimal canonical TCAM for C_R on $n-2$ bits, then $\mathcal{R} = 01\mathcal{R}_L + 10\mathcal{R}_R$ is a minimal canonical TCAM for S encoded on n bits.*

Proof. Every rule in a TCAM for S starts by 01 or 10. Therefore, any TCAM covering S can be written as $01\mathcal{R}_L + 10\mathcal{R}_R$, where $\mathcal{R}_L$ and $\mathcal{R}_R$ cover C_L and C_R, respectively. □

Step 4. If a skeleton tree S is of type ξ_3 then either $S.00 = \emptyset$ or $S.11 = \emptyset$. Since these two cases are very similar, in the following proposition we consider only one case, $S.00 = \emptyset$, which corresponds to the graphical representation of ξ_3 in Figure 5. Due to Lemma 2, for $k = 1$. we have the following fact.

Proposition 2 (Skeleton trees of type ξ_3). *Let S be an interval skeleton tree in $\texttt{Lex}_n$ such that $S.00 = \emptyset$, $S.01 = C_L \neq \emptyset$, $S.10 = \bullet$, and $S.11 = C_R \neq \emptyset$.*

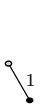
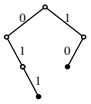
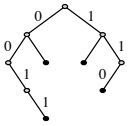
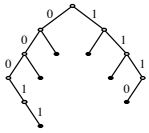
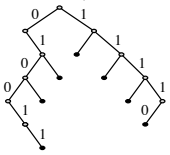
	$S'''', 1$	$S''', 3$	$S'', 4$	$S', 5$	$S, 6$
					
	ξ_0: i.e., chain	ξ_2: $\mathcal{R} = 01\mathcal{R}' + 10(*)$	ξ_4 (non-compl.): $\mathcal{R} = 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + (01*^2) + (10*^2)$	ξ_4 (complementary): $\mathcal{R} = 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + (10*^3)$	ξ_3 (L is right child): $\mathcal{R} = 01\cdot\mathcal{R}'_0 + 1*\cdot\mathcal{R}'_1 + *1\cdot\mathcal{R}'_* + (10*^4)$
$\mathcal{R}$	$1\boxed{1}$	$1\boxed{011}$ $2\boxed{10*}$	$1\boxed{0*11}$ $2\boxed{*10*}$ $3\boxed{01**}$ $4\boxed{10**}$	$1\boxed{0**11}$ $2\boxed{0*1**}$ $3\boxed{*10**}$ $4\boxed{**10*}$ $5\boxed{10***}$	$1\boxed{01**11}$ $2\boxed{01*1**}$ $3\boxed{1*0***}$ $4\boxed{*110**}$ $5\boxed{*1*10*}$ $6\boxed{10****}$
$\mathcal{R}_0$	$\emptyset$	$1\boxed{11}$	$1\boxed{*11}$ $2\boxed{1**}$	$1\boxed{**11}$ $2\boxed{*1**}$	
$\mathcal{R}_1$	$1\boxed{\varepsilon}$	$1\boxed{0*}$	$1\boxed{0**}$	$1\boxed{0***}$	
$\mathcal{R}_*$	$\emptyset$	$\emptyset$	$1\boxed{10*}$	$1\boxed{10**}$ $2\boxed{*10*}$	

Fig. 6. The history of the algorithm LexTCAM for interval $I = [19, 61]$ with $\texttt{Lex}_6$ encoding, presented in terms of skeleton trees together with intermediate TCAMs. The skeleton trees from the first row of the table correspond to the arguments of recursive calls (from right to left). The second row describes the type of the skeleton tree argument as well as the formula which is used to calculate the TCAM in which $\mathcal{R}' = 0 \cdot \mathcal{R}'_0 + 1 \cdot \mathcal{R}'_1 + * \cdot \mathcal{R}'_*$ corresponds to the result of the sub-recursive call. For convenience, all intermediate results $\mathcal{R}$ are presented explicitly (third row) and in form of $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}_*$ (remaining rows). For example, $\mathcal{R}'_0$ in the formula of column 3 (for recursive call $S'', 4$), refers to TCAM written in column 2 (call $S''', 3$) row 4 ($\mathcal{R}_0$).

Also let $S' = Merge(C_L, C_R)$, i.e., S' is an interval skeleton tree in $\texttt{Lex}_{n-1}$, such that $S'.0 = C_L$ and $S'.1 = C_R$. If $\mathcal{R}' = 0\mathcal{R}'_0 + 1\mathcal{R}'_1 + *\mathcal{R}'_*$ is a minimal canonical TCAM for S', then

$$\mathcal{R} = 01\mathcal{R}'_0 + 1*\mathcal{R}'_1 + *1\mathcal{R}'_* + 10(*^{n-2})$$

is a minimal canonical TCAM for S.

Step 5. In case of skeleton trees of type ξ_4, i.e., when all four grandchildren of the root are non-empty, we distinguish two cases; the chains rooted in the leftmost grandchild and the rightmost grandchild (i.e., chains C_L and C_R in ξ_4 of Figure 5) are *complementary* or are not.

Proposition 3 (Skeleton trees of type ξ_4). *Let S be an interval skeleton tree in $\texttt{Lex}_n$ such that $S.00 = C_L \neq \emptyset$, ($S.01 = \bullet$, $S.10 = \bullet$, and $S.11 = C_R \neq \emptyset$. Also suppose that $S' = Merge(C_L, C_R)$ is an interval skeleton tree in $\texttt{Lex}_{n-1}$ with a minimal canonical TCAM $\mathcal{R}' = 0\mathcal{R}'_0 + 1\mathcal{R}'_1 + *\mathcal{R}'_*$. If C_L and C_R are complementary chains then a minimal canonical TCAM for S is*

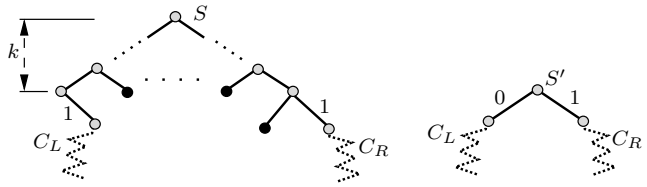


Fig. 7. Two skeleton trees S and S' from Lemma 2.

$\mathcal{R} = 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + 10(*^{n-2})$. *Otherwise a minimal canonical TCAM for* S *is:* $\mathcal{R} = 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + 10(*^{n-2}) + 01(*^{n-2})$.

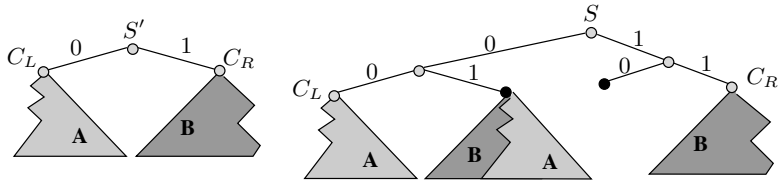


Fig. 8. Skeletons trees $S' = \mathit{Merge}(C_L, C_R)$ and S of Proposition 3 in case when C_L and C_R are complementary. Intervals corresponding to left chain C_L and right chain C_R are represented by trees A and B, respectively.

Proof. If C_L and C_R are complementary, then the construction in Figure 8 shows that TCAM $\mathcal{R} = 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + 10(*^{n-2})$ covers S, whenever $0\mathcal{R}'_0 + 1\mathcal{R}'_1 + *\mathcal{R}'_*$ covers S' and the number of rules in $\mathcal{R}$ is one more than that of $\mathcal{R}'$. Besides Lemma 2 guarantees that no smaller TCAM can cover S. Notice that another symmetric construction for $\mathcal{R}$ is also possible: $\mathcal{R} = *0\mathcal{R}'_0 + 1*\mathcal{R}'_1 + **\mathcal{R}'_* + 01(*^{n-2})$.

If C_L and C_R are non-complementary, then $01*\cdots*$ (resp., $10*\cdots*$) is an S-essential rule which cannot be covered by other S-essential rules. Therefore, $10*\cdots*$ and $01*\cdots*$ have to be in any minimal canonical solution for S. Notice that in case when C_L and C_R are non-complementary, there are four variants for construction of a minimal canonical TCAM for S from a minimal canonical TCAM for S':

$$\mathcal{R} = \alpha\mathcal{R}'_0 + \beta\mathcal{R}'_1 + **\mathcal{R}'_* + 10(*^{n-2}) + 01(*^{n-2}),$$

for $\alpha \in \{*0, 0*\}$ and $\beta \in \{*1, 1*\}$

Theorem 1. *The algorithm LexTCAM computes in time* $O(n + K)$ *a minimal canonical TCAM for an interval in the* n*-bit lexicographic encoding, where* K *is the total size (in bits) of the generated TCAM.*

5 Intervals in the reflected Gray encoding

Unlike for the lexicographic encoding, not every sub-tree of the reflected Gray dense-tree encoding $\texttt{Gray}_n$ is a reflected Gray dense-tree encoding. However, every pair of sibling sub-trees of the dense tree representing $\texttt{Gray}_n$ are the mirror copies of each other, $\texttt{Gray}_k$ and its mirror copy $\overline{\texttt{Gray}}_k$, $1 \leq k < n$.

Suppose that $I = [x, y]$ is an interval and $y < 2^n$. We say that I is *reciprocal* if $x = 2^n - 1 - y$. If I is reciprocal, then there is a $w \in \{0,1\}^{n-1}$ such that $x = \texttt{Gray}_n(0w)$ and $y = \texttt{Gray}_n(1w)$. (The same holds for $\overline{\texttt{Gray}}_n$ where 0 and 1 are interchanged.)

ALGORITHM *GrayTCAM*(S, n)

1. $v := LCA(x', y)$; $\alpha = \Pi_L(u, v)$;
2. $\mathcal{R}_1 := ChainTCAM_{n-1}(C_1)$; $\mathcal{R}_2 := ChainTCAM_{n-|\alpha|-2}(C_2)$;
   $\mathcal{R}_3 := ChainTCAM_{n-|\alpha|-1}(C_3)$;
3. **if** $(d_1 \leq d_2)$ **return** $(*\mathcal{R}_1 + 0\alpha*\mathcal{R}_2)$ **else return** $(*\mathcal{R}_1 + 0\alpha\mathcal{R}_3)$

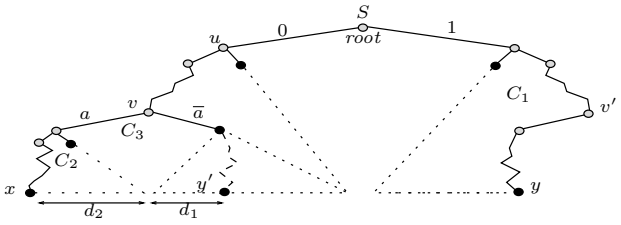


Fig. 9. The structure of the skeleton tree S for a non-reciprocal interval in $\texttt{Gray}$.

Lemma 3. *Let $I = [x, y]$ be a reciprocal interval with $x = \texttt{Gray}_n(0w)$ and $y = \texttt{Gray}_n(1w)$. Let $\mathcal{R}$ be a minimal canonical TCAM representation of the interval $I' = [\texttt{Gray}_{n-1}(w), 2^{n-1} - 1]$. The TCAM $*\mathcal{R}$ is a minimal canonical TCAM representation of the interval I. (See Appendix A.)*

Our algorithm for calculating the minimal canonical TCAM representation of intervals in Gray encoding relies on Lemma 3; it divides a given non-reciprocal interval into two overlapping sub-intervals where one of the intervals is reciprocal and the other one is not. Then it generates the minimal TCAM representation of the interval as the union of those of the sub-intervals.

The algorithm *GrayTCAM* (shown in Figure 9) generates a minimal canonical TCAM representation of an interval $[x, y]$ whose skeleton tree S has the form of Figure 9. In this skeleton tree y' is the mirror image of y in the left sub-tree and $y' > x$. Depending on the values of d_1 and d_2, the computation of the minimal TCAM is reduced to the computation of a minimal TCAM for the right chain C_1 and one of the left chains C_2 or C_3. $LCA(x, y')$ denotes the

lowest common ancestor of x and y', which is calculated by taking the greatest common prefix of x and y' in $\mathtt{Gray}_n$. $\Pi_L(u, v)$ is the label of the path from the left child of the root of S to node v, where every symbol of a left-going edge is replaced by $*$.

Theorem 2. *The algorithm GrayTCAM computes in time $O(n+K)$ a minimal canonical TCAM for an interval in the n-bit reflected Gray encoding, where K is the total size (in bits) of the generated TCAM.*

6 Conclusions

The problem of interval representation in TCAM consists of finding a representation of a set of consecutive integers by a number of TCAM rules. In this paper we presented two fast and simple algorithms that generate a minimal TCAM representation for a given interval: one algorithm for the binary reflected Gray encoding and the other for the lexicographic encoding. Our algorithms can also be used in the context of the "two-level logic minimization" [1, 2], since a minimal TCAM representation of an interval corresponds to a minimal size logical formula in DNF (disjunctive normal form) defining the characteristic function of the interval.

References

1. R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. S.-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.
2. O. Coudert. Two-level logic minimization: an overview. *Integr. VLSI J.*, 17(2):97–140, 1994.
3. G. Davis, C. Jeffries, and J. van Lunteren. Method and system for performing range rule testing in a ternary content addressable memory, April 2005. US Patent 6,886,073.
4. W. Fraczak, W. Rytter, and M. Yazdani. TCAM representations of intervals of integers encoded by binary trees. In *IWOCA'07*, p. 59–69, Australia, College Publications, 2007.
5. E. N. Gilbert. Gray codes and paths on the n-cube. *Bell Systems Technical Journal*, 37:815–826, 1958.
6. F. Gray. Pulse code communications, March 1953. US Patent 2,632,058.
7. H.-J. Jeong, I.-S. Song, T.-G. Kwon, and Y.-K. Lee. A multi-dimension rule update in a TCAM-based high-performance network security system. In *AINA'06*, p. 62–66, Washington, DC, USA, 2006. IEEE Computer Society.
8. R. Kempke and A. McAuley. Ternary CAM memory architecture and methodology, August 1996. US Patent 5,841,874.
9. T. Kohonen. *Content-Addressable Memories*. Springer-Verlag, New York, 1980.
10. K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with Ternary CAMs. In *SIGCOMM'05*, p. 193–204, 2005.
11. H. Liu. Efficient mapping of range classifier into Ternary-CAM. In *HotI'02*, p. 95, Washington, DC, USA, 2002. IEEE Computer Society.
12. V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *SIGCOMM'98*, p. 191–202, 1998. ACM.

A Appendix

Alternative presentation of algorithm *LexTCAM*. Assume $\alpha = (a_1, a_2, \ldots a_k), \beta = (b_1, b_2, \ldots b_k)$ are binary strings of a same length k, denote by $[\alpha]_2$ the number corresponding to α in binary. Assume $p = [\alpha]_2 \leq q = [\beta]_2$. The input interval $[p, q]$ is presented in the form (α, β). Denote by $size(\alpha, \beta)$ the length of the interval $[p, q]$. Denote by $\mathbf{1}_k$, $\mathbf{0}_k$ the sequence of k ones, and k zeros, respectively. It is convenient to assume later that we write $\mathbf{0}$, $\mathbf{1}$ without indices, the length is implied by the other string in a corresponding pair. The crucial is the notion of complementary interval. The pair $\alpha \leq \beta$ is called **complementary** iff

$$size(\alpha, \mathbf{1}_k) + size(\mathbf{0}_k, \beta) \geq 2^k$$

ALGORITHM $LexTCAM1(\alpha, \beta)$

1. if $[\alpha, \beta]$ is a chain then **return** $ChainTCAM(\alpha, \beta)$;
2. if $(\alpha = a \cdot \alpha',\ \beta = a \cdot \beta'$ for $a \in \{0, 1\}$ then **return** $a \cdot LexTCAM1(\alpha', \beta')$;
3. if $(\alpha = 01\alpha',\ \beta = 10\beta'$ then **return**
   $01 \cdot ChainTCAM(\alpha', \mathbf{1}) + 10 \cdot ChainTCAM(\mathbf{0}, \beta')$;
4. Let $\alpha = a_1a_2 \cdot \alpha',\ \beta = b_1b_2 \cdot \beta'$;
   $\mathcal{R}' := LexTCAM1(0 \cdot \alpha',\ 1 \cdot \beta')$; Represent $\mathcal{R}'$ as $0\mathcal{R}'_0 + 1\mathcal{R}'_1 + *\mathcal{R}'_*$;
   if $a_1a_2 = 00,\ b_1b_2 = 10$ then **return** $01 \cdot \mathcal{R}'_0 + 11 \cdot \mathcal{R}'_1 + **\mathcal{R}'_* + 10* \ldots$;
   if $a_1a_2 = 01,\ b_1b_2 = 11$ then **return** $0* \cdot \mathcal{R}'_0 + 10 \cdot \mathcal{R}'_1 + **\mathcal{R}'_* + 01* \ldots$;
5. (Now $a_1a_2 = 00,\ b_1b_2 = 11$;) $\mathcal{R} := 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + 10** \cdots *$;
   if α', β' are complementary then **return** $\mathcal{R}$ else **return** $\mathcal{R} + 01** \cdots *$;

The motivation for presenting the algorithm in terms of skeleton trees was to ease the proof of correctness. The skeleton tree is showing much better the structure of the interval from the point of view of constructing a minimal TCAM.

The proof of Lemma 1. Consider a chain C in a dense-tree encoding T_n. The minimal canonical TCAM representation of C, denoted by $ChainTCAM(C, n)$, can be calculated as follows:

- $ChainTCAM(\emptyset, n) = ()$, i.e., the empty TCAM;
- $ChainTCAM(\bullet, n) = (*^{\mathtt{n}})$, i.e., the single rule consisting n star symbols.

Otherwise, i.e., when $C \notin \{\emptyset, \bullet\}$:

$$ChainTCAM(C, n) = \begin{cases} 1ChainTCAM(C.1, n-1) & \text{if } C.0 = \emptyset \\ 0ChainTCAM(C.0, n-1) & \text{if } C.1 = \emptyset \\ *\mathtt{ChainTCAM(C.1, n-1)} + \mathtt{0}(*^{\mathtt{n-1}}) & \text{if } C.0 = \bullet \\ *\mathtt{ChainTCAM(C.0, n-1)} + \mathtt{1}(*^{\mathtt{n-1}}) & \text{if } C.1 = \bullet \end{cases}$$

The above characterization of the minimal canonical TCAM representation of a set X holds as long as the corresponding skeleton tree for X is a chain.

The proof of Lemma 2.

Lemma 4. [4] *A minimal TCAM representation of* $I_k = [1, 2^k - 2]$ *in* $\texttt{Lex}_k$ *needs exactly* k *rules.*

Let $\mathcal{R}$ be a minimal canonical TCAM for I and $\mathcal{R}_C$ be the set of those rules in $\mathcal{R}$ which enter into the chains C_L and C_R (i.e., rules that intersect with $0^k 1 * * \cdots *$ or with $1^{k+1} * * \cdots *$). The rules of $\mathcal{R}_C$ can be turned into a TCAM for I' by replacing $k+1$ initial symbols of each rule with one of 0, 1, or $*$.

If $k = 1$, then the I-essential rule $10 * * \cdots *$ cannot be covered by other I-essential rules. Therefore a minimal TCAM for I has exactly one rule more than a minimal TCAM for I'. For $k > 1$, since C_R is not a single node tree (i.e., at least $111 \cdots 1 \notin C_R$), no inner node of S at level k, except the parents of C_L and C_R, can be covered by rules from $\mathcal{R}_C$. More precisely, no string which has both a 0 and a 1 in its k first bits, 0 at position $k+1$, and then only 1 in all positions bigger than $k+1$, can be covered by rules from $\mathcal{R}_\mathcal{C}$. Thus, by Lemma 4, we need at least k rules outside of $\mathcal{R}_\mathcal{C}$ to cover all those k-level inner nodes.

The proof of Lemma 3.

The proof is illustrated in Figure 10.

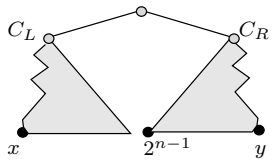


Fig. 10. Illustration of Lemma 3; y is the mirror image of x with respect to the root of the tree. If $\mathcal{R}$ is a minimal TCAM for left-chain C_L then $*\mathcal{R}$ is a minimal TCAM for the whole tree.