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

Wojciech Fraczak, Wojciech Rytter, Mohammadreza Yazdani

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 (Internet Protocol) 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.

We present two fast (linear time in the size of the input and the output) algorithms for finding minimal solutions.

a)

1	0001
2	001*
3	01**
4	10**
5	110*
6	1110

b)

1	0**1
2	10**
3	*10*
4	**10

c)

1	01**
2	1*0*
3	*0*1
4	**10

d)

1	0*1*
2	10**
3	*1*0
4	**01

e)

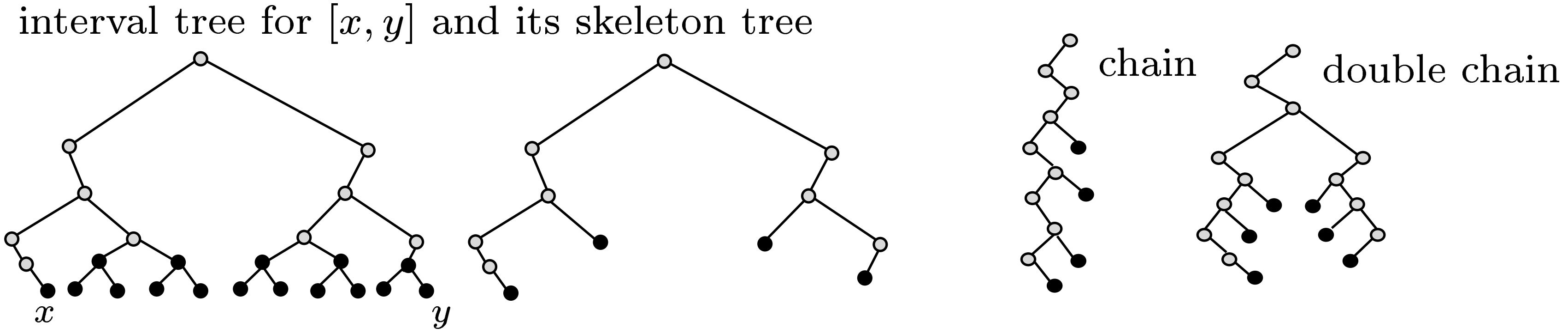
1	01**
2	1**0
3	*01*
4	**01

Rysunek 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.

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 are two important examples of dense-tree encodings.

In the context 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* . 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.



Rysunek 2: The skeleton tree for an interval $[x, y]$, a chain, and a double-chain.

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 2. 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

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 $O(kn)$.

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 .

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

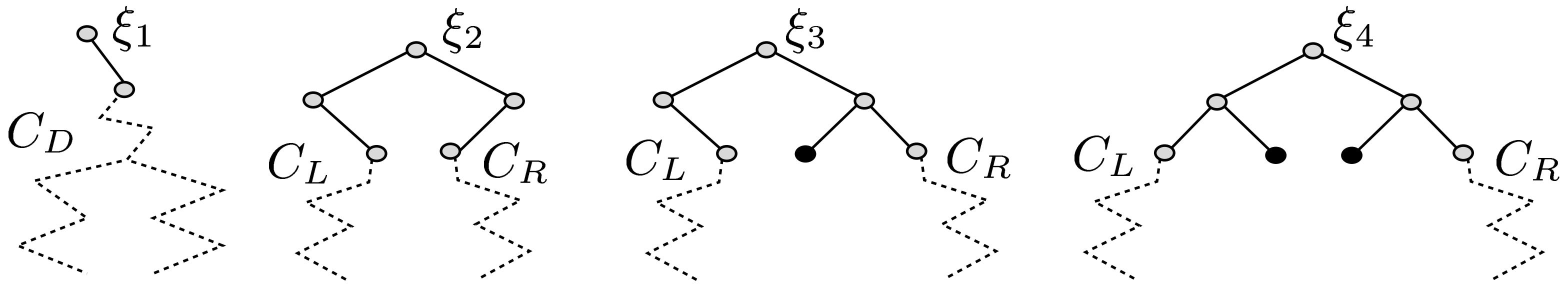
   return $01 \cdot \mathcal{R}'_0 + 1* \cdot \mathcal{R}'_1 + *1 \cdot \mathcal{R}'_* + (10*^{n-2})$;

   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** $\mathcal{R} + (01*^{n-2})$;

The algorithm consists of 5 parts which correspond to the treatment of 5 different types of interval skeleton trees described below



Rysunek 3: 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.

5 different types of interval skeleton trees

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.

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.*

Good example: an interval $I = [19, 61]$ within $\texttt{Lex}_6$ encoding.

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

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*(α, β)

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 $\alpha = a_1 a_2 \cdot \alpha', \; \beta = b_1 b_2 \cdot \beta'$;
   $\mathcal{R}' := LexTCAM1(0 \cdot \alpha', \; 1 \cdot \beta')$; Represent $\mathcal{R}'$ as $0\mathcal{R}'_0 + 1\mathcal{R}'_1 + *\mathcal{R}'_*$;
   if $a_1 a_2 = 00, \; b_1 b_2 = 10$ then **return** $01 \cdot \mathcal{R}'_0 + 11 \cdot \mathcal{R}'_1 + **\mathcal{R}'_* + 10* \ldots$;
   if $a_1 a_2 = 01, \; b_1 b_2 = 11$ then **return** $0* \cdot \mathcal{R}'_0 + 10 \cdot \mathcal{R}'_1 + **\mathcal{R}'_* + 01* \ldots$;
5. (Now $a_1 a_2 = 00, \; b_1 b_2 = 11$;) $\mathcal{R} := 0*\mathcal{R}'_0 + *1\mathcal{R}'_1 + **\mathcal{R}'_* + 10** \cdots *$;
   if α', β' are complementary then **return** $\mathcal{R}$ else **return** $\mathcal{R} + 01** \cdots *$;

Example Let us consider the same interval as before: $[19, 61]$, corresponding to the pair $(\alpha, \beta) = (010011,\ 111101)$. Below there is a part of the history of the computation (going back from recursion):

$$LexTCAM(011,\ 101) \ = \ 011 \ + \ 10* \ \stackrel{Step5}{\rightarrow}$$

$$LexTCAM(0011,\ 1101) \ = \ 0*11 \ + \ *10* \ + \ 10** \ + \ 01** \ \stackrel{Step5}{\rightarrow}$$

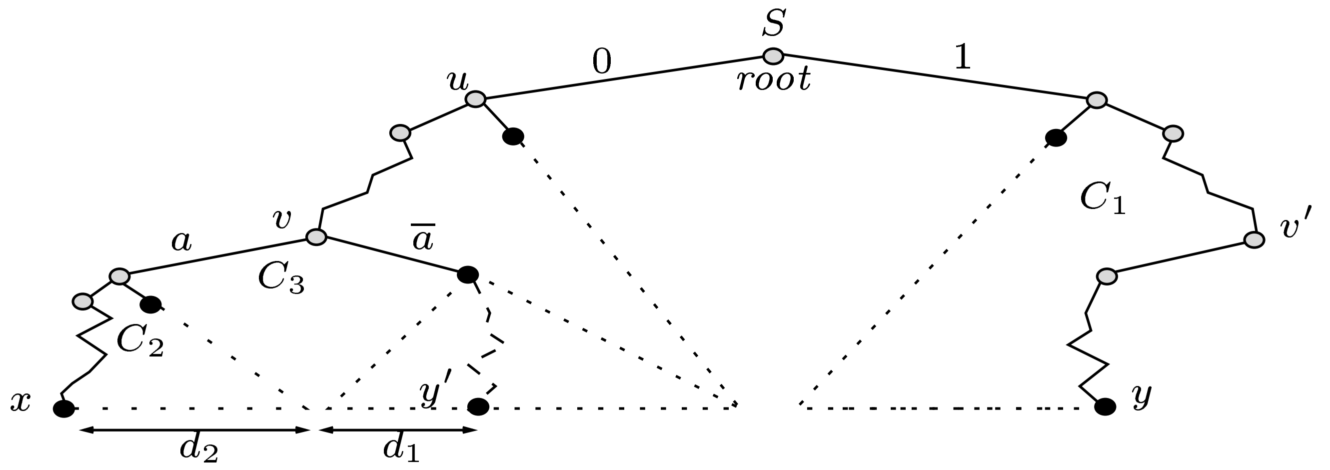
$$LexTCAM(00011,\ 11101) \ = \ 0{*}{*}11 + {*}{*}10{*} + {*}10{*}{*} + 0{*}1{*}{*} + 10{*}{*}{*} \stackrel{Step4}{\rightarrow}$$

$$LexTCAM(010011,\ 111101) \ = \ 01**11 \ + \ **1*10* \ + \ *110** \\ + \ 01*1** \ + \ 1*0*** \ + \ 10****$$

Unlike for the lexicographic encoding, not every sub-tree of the reflected Gray dense-tree encoding $\mathtt{Gray}_n$ is a Gray dense-tree encoding. However, every pair of sibling sub-trees of the dense tree representing $\mathtt{Gray}_n$ are the mirror copies of each other, $\mathtt{Gray}_k$ and its mirror copy $\overline{\mathtt{Gray}}_k$, $1 \leq k < n$. 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 $x = \mathtt{Gray}_n(0w)$ and $y = \mathtt{Gray}_n(1w)$. (The same holds for $\overline{\mathtt{Gray}}_n$ 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** $(*\mathcal{R}_1 + 0\alpha\mathcal{R}_3)$



Rysunek 4: The structure of the skeleton tree S for a non-reciprocal interval in `Gray`.

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.*

Theorem 3 *There is a dense-tree encoding T_n and an interval I such that the minimum number of prefix rules representing I is $2n-2$ and the minimum TCAM size is only n.*

Theorem 4
(a) *Let T_n be a lexicographic or reflected Gray encoding of length n. There is an interval $I = [x, y]$ which cannot be represented with less than $\max(n, 2n-4)$ TCAM rules.*
(b) *For each $n \geq 1$ there exists a dense-tree encoding T_n and an interval $I = [x, y]$ which needs $\max(n, 2n-3)$ TCAM rules.*

Theorem 5 *Let T_n be a dense-tree encoding of length n. Every interval $I = [x, y]$ can be represented by* $\max(n, 2n-3)$ *TCAM rules.*

Theorem 6 *Let $T_n \in \{\mathtt{Lex}_n, \mathtt{Gray}_n\}$ be the lexicographic or the reflected Gray encoding of length n. Every interval $I = [x, y]$ can be represented by* $\max(n, 2n-4)$ *TCAM rules.*