TT and ANF Representations of Boolean functions

Truth Table(TT)

The truth table representation is the default representation of a Boolean function as it directly translates the definition of a Boolean function. The TT of a Boolean function f on ${F_2}^n$ is a binary vector of length 2^n, each element of this binary vector is an image corresponding to a unique element in ${F_2}^n$. Now we introduce a notation that lets us order the elements of the TT lexicographically. We replace each element $(x_0, x_1, \ldots, x_{n-1})$ in ${F_2}^n$ by its decimal representation $x = x_0 2^{n-1} + x_1 2^{n-2} + ... + x_{n-1}$. So instead of writing $f(0, 0, \ldots, 0)$ we write $f(0)$, instead of $f(0, 0, \ldots, 1)$ we write $f(1)$, instead of $f(1, 1, \ldots, 1)$ we write $f(2^n - 1)$ and so on. This gives a lexicographical order on all the elements of ${F_2}^n$ and allows us to define a Boolean function as $f = [f(0)\ f(1)\ f(2)\ \ldots f(2^n - 1)]$. For instance, suppose we have a 3-variable Boolean function $f = [0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$. Then our TT will be as shown in Table 2.1.

x	x_0	x_1	x_2	$f(x)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Table 1: Truth table

Another representation that is closely related to the truth table is the polarity TT(PTT) or bipolar representation, and is widely used in telecommunications. It is defined as $(-1)^f = [(-1)^{f(0)}\ (-1)^{f(1)}\ \ldots\ (-1)^{f(2^n-1)}]$ which means that instead of 0's in TT we have 1's in the PTT and instead of 1's in TT we have -1's in the PTT. So it is a sequence of $\{1, -1\}$'s.

Algebraic Normal Form(ANF)

The ANF is one of the most used representations in cryptography. An ANF of a Boolean function on ${F_2}^n$ is a polynomial of the following form:

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{j=(j_0,\ldots,j_{n-1})\in F_2^n} a_j x_0^{j_0} x_1^{j_1} \cdots x_{n-1}^{j_{n-1}} \, (mod\ 2)$$

where $a_j \in F_2$.
The *algebraic degree of* f, denoted by *deg(f)*, is the number of variables in the longest term(s) of the ANF of f. If *deg(f)* ≤ 1, then f is called an *affine* function. An affine function without the constant term ($i.e.$ $a_0 = 0$) is often called a *linear* function. An affine function with *deg(f)* $= 0$, which is either $f(x) = 0$ or $f(x) = 1$, is called a *constant* function. The set of affine functions is denoted by $A(n)$.
Let $C = [c_0\ c_1\ \ldots\ c_{2^n-1}]$ be the coefficient vector of the polynomial representing the Boolean function f. If $c_j = 1$, where $0 \leq j \leq 2^n - 1$, then the monomial $x_0^{j_0} x_1^{j_1} \cdots x_{n-1}^{j_{n-1}}$ exists in the ANF of f and does not otherwise, where $(j_0, j_1, \ldots, j_{n-1})$ is the binary representation of index j. For instance, if $C = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$ then the ANF is $x_0 + x_0x_1x_2$. The following theorem, shows a relation between C and the truth table $f = [f(0)\ f(1)\ f(2)\ \ldots\ f(2^n - 1)]$.

Theorem

Let f be the truth table of an n-variable Boolean function. Let be as defined above. Then

$$C = fA_n$$

where

$$A_n = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{\otimes n}$$

That is A_n is the nth tensor power mod 2 of the matrix $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$*, or in other notation,*

$$A_n = \begin{bmatrix} A_{n-1} & A_{n-1} \\ 0 & A_{n-1} \end{bmatrix} \qquad \textit{and} \qquad A_0 = [1].$$

The above theorem helps us to convert from truth table to ANF and vice versa in almost 2^{2n} binary operations. The following algorithm reduces the conversion to only $O(n2^n)$ operations.

Algorithm: TT $\rightarrow$ ANF
Input: TT of a Boolean function f
Output: The coefficient vector of the ANF of f
For $0 \leq k \leq n$, define $f_{k,a} \in F_{2^k}$, where $0 \leq a \leq 2^{n-k} - 1$.

1. Set $f_{0,a} = f(a)$ for $0 \leq a \leq 2^n - 1$.
2. *for* $k = 0$ to $n - 1$ do
   for b = 0 to $2^{n-k-1} - 1$ do
   $f_{(k+1),b} = [f_{k,2b}\ f_{k,(2b+1)} + f_{k,2b}]$
3. $C = f_{n,0}$

The following example illustrates what the above algorithm does. Let us find the ANF of the Boolean function represented by the truth table in Table 2. We have $f = [0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$. Looking at the 1's positions in C we see that the ANF of f is $x_2 + x_1 + x_0x_1$.

f	0	1	1	0	0	1	0	1
$k = 0$	0	1	1	1	0	1	0	1
$k = 1$	0	1	1	0	0	1	0	0
$k = 2$	0	1	1	0	0	0	1	0
$C = f_{3,0}$	0	1	1	0	0	0	1	0

Table 2: Converting TT to ANF algorithm

The above algorithm can also be used to convert from ANF to TT by changing the input from TT to C (the coefficients) of the ANF of f.