INFORMATION TECHNOLOGY BY BOOLEAN CONSTANTS

Moh Moh Kyi [1]

Abstract

This paper expresses the applications of Boolean algebra by Boolean constants with the technical applications. First we have to express with rooted tree, binary tree, ordered tree and expression tree as the case of the graph in discrete version. Then we have to express the prefix code tree by encoding and decoding. According to the transformation of encoding to decoding and decoding to encoding, we can connect with discrete version and Boolean version over communications line. In addition probability version too.

Key Words: prefix code tree, encoding and decoding.

INTRODUCTION

This paper has three parts to be expressed that the connection of the discrete mathematics and ***Boolean*** algebra. For this paper, we have to start with rooted tree, binary tree, ordered tree and expression tree from the version of discrete mathematics, and then the prefix code tree in the second. In addition the prefix code tree is not only the foundation of rooted tree, binary tree, ordered tree and expression tree but also applications to ***Boolean*** algebra by using encoding and decoding. Over the communications line the technical applications are going to be informed by transformation of encoding to decoding and decoding to encoding. Here encoding is the length of bit string with ***Boolean*** constants 0,1 and decoding is the message of the alphabets according to the indicated path of the encoding. Finally, we have to express the relation of the prefix code tree and the tree diagrams of some probability trees.

OBJECTIVES

This paper is intended to be express as the technical information by the connection of the versions of Boolean algebra and discrete mathematics with encoding and decoding in the prefix code tree. Then probability version is too. The information appears over the communications line.

[1]Dr., Associate Professor, Mathematics Department, Co-operative University(Thanlyin)

INFORMATION TECHNOLOGY BY BOOLEAN CONSTANTS

Materials & Methods

1. Boolean Algebra

A Boolean algebra $(B, \wedge, \vee, ')$ is a non empty set B together with two binary operations $\wedge$ and $\vee$ on B and a urinary operation $'$ on B satisfying the following axioms for all *x*, *y*, *z* in B.

1. $x \wedge y = y \wedge x$ } Commutative law
2. $x \vee y = y \vee x$
3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ } Associative law
4. $x \vee (y \vee z) = (x \vee y) \vee z$
5. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ } Distributive law
6. $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
7. There is a zero element o in B such that $x \vee 0 = x$
8. There is a unit element 1 in B such that $x \wedge 1 = x$
9. $x \wedge x' = 0$
10. $x \vee x' = 1$

where zero element 0∈ B is a least element of B and unit element 1∈ B is a greatest element of B for all elements in B (i.e, $0 \le x \le 1, \forall x \in B$).

Example

Let A= { 1, 2 } be a non empty set and P(A) be the set of all subsets of A. Then P(A)= {*Φ*,{ 1 },{ 2 },{ 1,2 }} has $2^2 = 4$ elements.

Then { 1 } ∩ { 2 } = *Φ*,

{ 1 } ∪ { 2 } = { 1, 2 } = A

The above shows that the set P(A) has a least element *Φ* and a greatest element A. The complement of set { 1 } is { 2 } and the complement of { 2 } is { 1}.

Therefore $\langle P(A), \cap, U, ' \rangle$ is a Boolean algebra because all axioms of Boolean algebra are also satisfied.

Terminology

(i) $x \wedge y$ (meet of x and y)

(ii) $x \vee y$ (join of x and y)

(iii) x' (complement of x)

(iv) 0 (zero element (or) least element)

(v) 1 (unit element (or) greatest element)

Theorem

The complement of a Boolean algebra is unique.

(If $a \wedge b = 0,\ a \vee b = 1,$ then $b = a',\ \forall a, b \in B.$)

Proof

For any $a,b \in B$. Then

$$\begin{aligned} b &= b \vee 0 \\ &= b \vee (a \wedge a') \\ &= (b \vee a) \wedge (b \vee a') \\ &= 1 \wedge (b \vee a') \\ &= b \vee a' \end{aligned}$$

$$\begin{aligned} a' &= a' \vee 0 \\ &= a' \vee (a \wedge b) \\ &= (a' \vee a) \wedge (a' \vee b) \\ &= 1 \wedge b \\ &= b \end{aligned}$$

Example

Let A be the set of all positive integers which are integral divisions of 20. Then □ A, □, $\wedge$,′ □is a Boolean algebra.

Proof: A={1,2,4,5,10,20}

For any $x,y \in$ A. Let $x \wedge y$ be g.c.d (greatest common divisor), $x \vee y$ be l.c.m (least common multiple) and $x' = \dfrac{20}{x}$

Then

$4 \wedge 5 = 1,\ 4 \vee 5 = 20$ and $5' = 4$

$2 \wedge 4 = 1,\ 2 \vee 4 = 20$ and $2' = 10$

Thus $\langle A, \wedge, \vee, ' \rangle$ is a Boolean algebra.

2. Tree

Trees are connected graph with no closed trails. Here a trail is a walk with distinct edges.

3. Rooted tree

A tree with the additional structure of having one vertex designated as the root is called a rooted tree.

Example of Rooted Tree

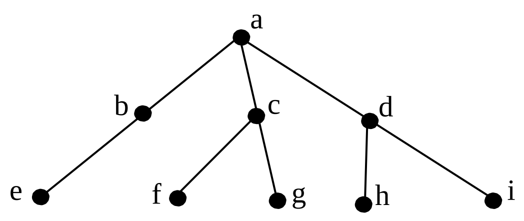

Figure 1: A rooted tree

4. Subtree of a Tree

Subtree of a Tree is a subgraph of the tree.

Example of Subtree

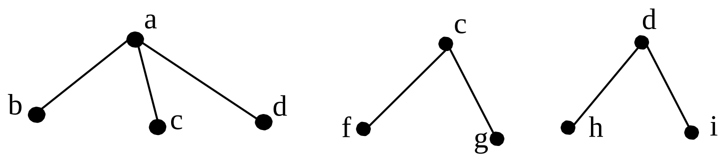

Figure 2: Three subtrees of figure 1.

The above figure shows three subtrees come from a rooted tree of figure 1.

5. Level of the vertex v in a Rooted Tree

Level of the vertex v in a rooted tree is the length of the unique path from the root to v, thus the root is at level o, the vertices adjacent to the root are at level 1 and so on.

Example of a Rooted Tree with level

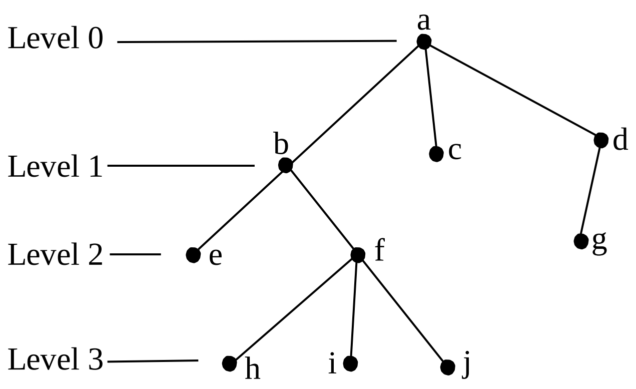

Figure 3: A rooted tree with level 3.

The above figure shows that there are 3 levels, here, '*a*' is a root at level '0'. The vertices b,c,d are at level '1' they are adjacent from '*a*'. The vertices e,f,g are at level '2' here the vertex e and f are adjacet from b and the vertex g is adjacent from d. level 3 vertices are h,i and j they are adjacent from f.

6. Binary Tree

A binary tree is a rooted tree witheach vertex has either two branches (both left and right) or only one branch (left or right) for every vertices at level i, i ≥ 1.

Example of a Binary Tree

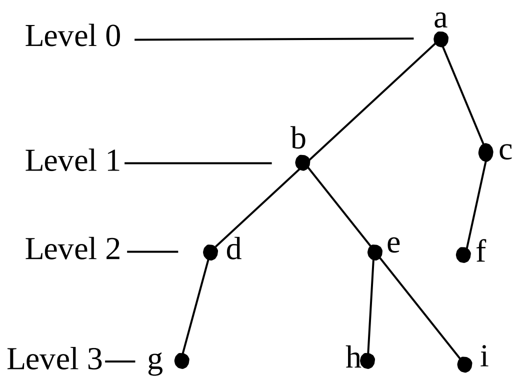

Figure 4: A binary tree

The above figure shows a vertex ‘a’ is at level 0, ab and ac are left branch and right branch of a. the vertices b and c are at level 1, bd and be are left and right branches of b, cf is only left branch of c.Similarly, the vertices d,e,f are at level 2 and the vertices g,h,i are at level 3 here dg and eh are two left branches of d and e respectively. But ei is a right branch of e.

7. Order tree

An ordered tree is a rooted tree in which the branches of each vertex are shown in order from left to right.

8. Expression Tree

Expression tree is an ordered tree in which leaves represent variables or constants and internal vertices represent operations being applied to the values of their branches.

Example of Expression Tree

The expression tree for (3+2) · 5 as the following figure. Consider the expression tree for the calculation (3 + 2) · 5. First we consider the addition operation (3 + 2) and then the multiplication operation for the remaining 5 as shown in the following figure from bottom to up position.

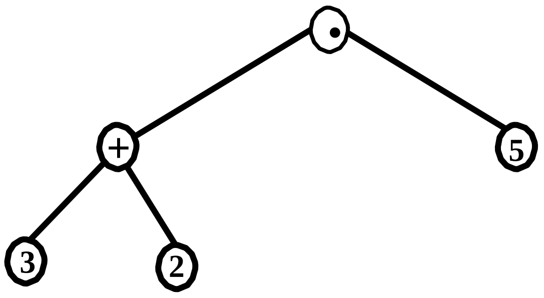

Figure 5 : Expression tree for **(3+2) · 5**

Example of Expression Tree

Consider the expression tree for Boolean expression ($a \wedge$ b) $\vee$ (b′ $\wedge$ c). It can be expressed as follows:

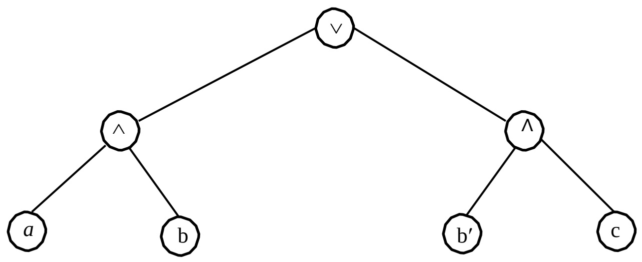

Figure 6 : Expression tree for $(a \wedge b) \vee (b' \wedge c)$

In figure 6, first we have to consider the two brackets $(a \wedge b)$ and $(b' \wedge c)$ and then ‘ v’ join operation for the given expression.

9. Calculation of Boolean Values (or) Boolean Constants

Let us denote Boolean constant 1 for leaf x and 0 for leaf x' .

Example

Consider the expression tree for the expression $(a \wedge b) \vee (a' \wedge c)$. It can be shown as the following figure.

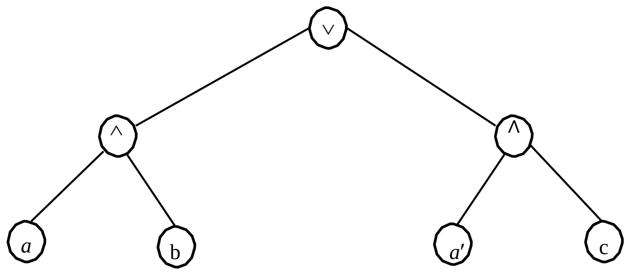

Figure 7(a) : Expression tree for $(a \wedge b) \vee (a' \wedge c)$

There are four leaves in the above figure. Let us denote for the leaf $a = b = c = 1$. Then $a' = b' = c' = 0$. Thus we obtain the following expression tree by substituting the corresponding Boolean constants in place of the given expression in figure 7.

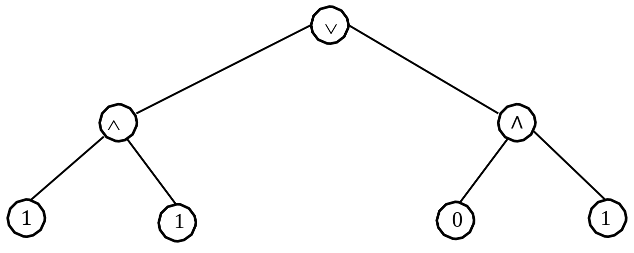

Figure 7(b) : Expression tree by Boolean constants

Figure 7 shows that the new expression tree of figure 7 by substituting Boolean constants 0, 1. Here $1 \wedge 1 = 1$ and $0 \wedge 1 = 0$. Thus $(1 \wedge 1) \vee (0 \wedge 1)$ gives $1 \vee 0 = 1$. Thus the value of the given expression tree by Boolean constant is 1.

Example

Consider the expression tree for the expression $(a \vee b) \wedge (c \wedge b')$. It can be represented by the following expression tree.

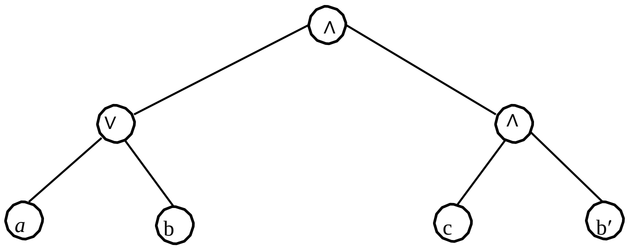

Figure 8 : Expression tree for $(a \vee b) \wedge (c \wedge b')$

The above expression tree as shown in figure 8 can be represented by the following expression tree as shown in figure 9. Here we have to replace the Boolean constant 1 in place of the leaves *a* and b. Similarly Boolean constant 0 takes place in leaf b′.

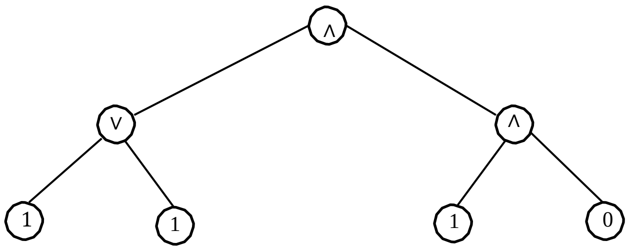

Figure 9 : Expression tree by Boolean constants

We have to calculate Boolean values for the above expression $(1 \vee 1) \wedge (1 \wedge 0) = 1 \wedge 0$ gives the value 0.

Example

Consider the expression tree for the expression $(a \wedge b) \vee (a \wedge c) \vee (a' \wedge b')$. It can be represented by the following expression tree.

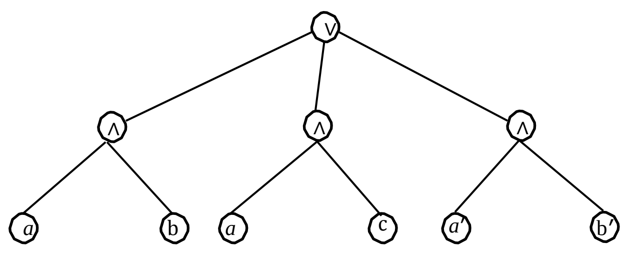

Figure 10: Expression tree for ($a \wedge$ b) ∨ ($a \wedge$ c) ∨ ($a' \wedge$ b′)

Here the leaves a, b the leaves a, c and the leaves a', b′ are level 2 vertices. They are separated from each level 1 vertices with 3 meet operation leaves. Then these three meet operations are separated from a level 0 vertex with the leaf join operation. Thus we obtain the given expression.

On the other hand, we have to substitute the value '1' in place of the leaves a, b, a, c and the value 0 in place of the leaves a', b′ . Therefore we get the following expression tree again.

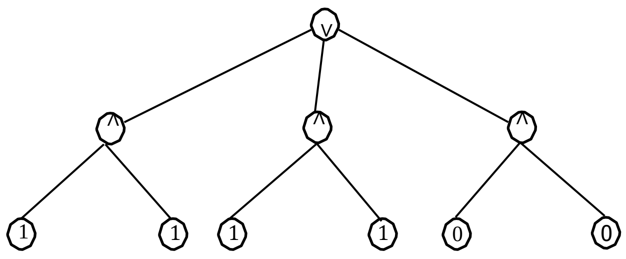

Figure 11: Expression tree by Boolean constants

From figure 11, we can calculate the value of the given expression $(1 \wedge 1) \vee (1 \wedge 1) \vee (0 \wedge 0) = 1 \vee 1 \vee 0 = 1$. So the figure 11 gives the value 1.

Example

Expression tree for the Boolean expression ($a' \vee$ b) ∧ (b′ ∨ c′) ∧ ($a \vee$ c′) can be represented by the following figure.

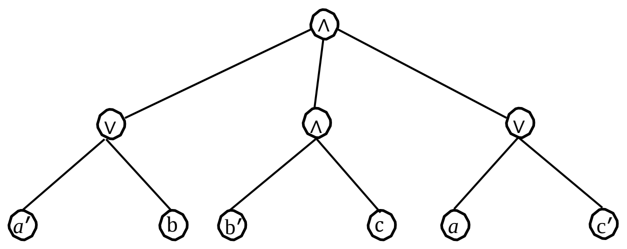

Figure 12: Expression tree for Boolean expression ($a' \vee$ b) ∧ (b′ ∨ c′) ∧ ($a \vee$ c′)

In figure 12, we need to express the given expression from bottom to up positions. If we put the value ‘1’ to the leaves a, b and the value ‘0’ to the leaves a′ ,b′ , c′ at each level 2 vertex. Therefore we obtain the following expression tree

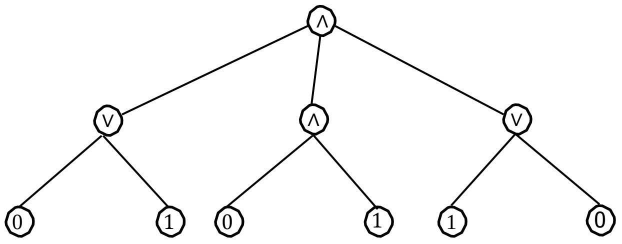

Figure 13: Expression tree by Boolean constants

We can calculate $(0 \vee 1) \wedge (0 \wedge 1) \wedge (1 \vee 0) = 1 \wedge 0 \wedge 1 = 0$. The above figure 13 gives the value 0.

Example

Expression tree for the set (A ⋂ B) ⋃ (C ⋂ D).

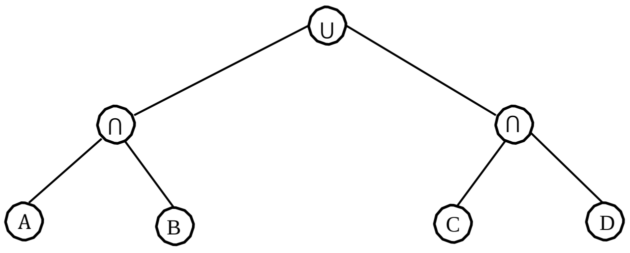

Figure 14: Expression tree for the set (A ⋂ B) ⋃ (C ⋂ D).

The above figure shows that the expression tree for the given expression , firstly, we use the intersection operation ‘⋂’ for each bracket and then the union operation ‘⋃’ for the whole set.

Here we notice that we can replace the set operations ‘⋂’ intersection and ‘⋃’ union on behalf of the Boolean operations ‘∧’ meet operation and ‘∨’ join operation respectively.

Remark

Every expression tree of Boolean expression gives the Boolean values.

Results & Discussion

1. Prefix Code Tree

A prefix code tree for an alphabet set V is a binary tree whose leaves are in one-to-one correspondence with the elements of V. If v is a symbol in V, then the encoding of v is the bit string that represents the path from the root of the tree to the leaf corresponding to v, where a '0' represents descent to a left branch and a '1' represents descent to a right branch.

2. Encoding

When information is stored in a computer or when information is sent over communications lines, it is usually as bit strings. On the other hand, an encoding method is a function from V^* to $\{0, 1\}^*$ where V^* is the set of bit strings over the alphabet V.

For instance, encodes 011 and 100 represents a bit string of length 3 for some alphabets A and B in V.

Example of a Prefix Code Tree

Let V={A, B, C, D, E, F, G, H, I} be the set of alphabets. We can show as the following prefix code tree.

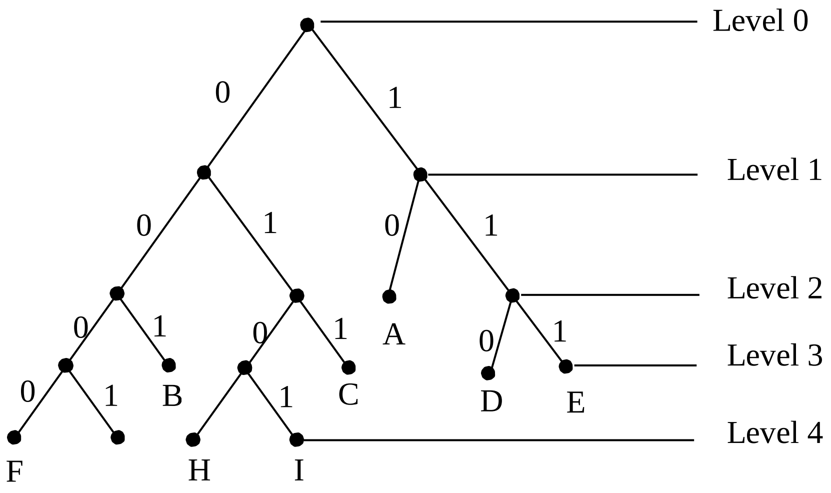

Figure 15: A prefix code tree with level 4.

The above prefix code has 4 levels with level 0 to level 4. Each vertex separates 2 branches with left branch '0' and right branch '1'. ā set V has 9 alphabets, each alphabet takes place from level 2 to level 4 and each alphabet has no new separating branches to the next level. Moreover each alphabet can be represented by the codes such as the codes 10 for A, 001 for B, 011 for C, 110 for D, 111 for E, 0000 for F, 0001 for G, 0100 for H, 0101 for I respectively. Clearly, we see that F, G, H, I have bit strings of length 4, alphabets B, C, D, E have bit string of length 3 and A has bit strings of length 2. The encoding for AGE can be represented by 10 0001 111. Similar way for any message of alphabets in V.

3. Decoding

Decoding is the message of the alphabets in V. It may be obtained by the indicated path from a prefix code tree of the encoding.

Example of Decoding

In figure 7, let's consider decoding from the bit string of encoding 110 01 10.

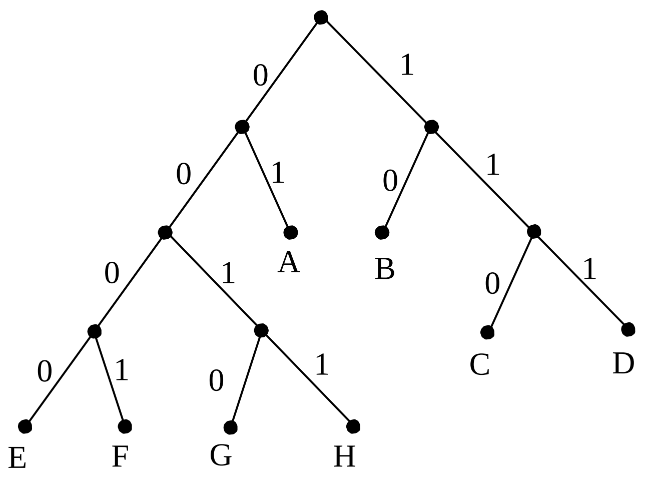

Figure 16: A prefix code tree in a set V

The given encode 110 01 10 begins with 1. In the above figure shows there are three possible alphabets begin with 1 namely B, C and D. By decoding, 10 refers to B, 110 refers to C and 111 refers to D. Similarly there are 5 possible letters begin with 0 namely A, E, F, G and H. By decoding A refers to 01, E refers to 0000, F refers to 0001, G refers to 0010 and H refers to 0011 respectively. Thus we get the decoding CAB for the given encoding 110 01 10. Therefor we can find message decoding from any given encoding.

Remark

(1) In any prefix code tree has the length of bit string and the number of level of the vertices are the same.

(2) In any prefix code tree, we can send the information from bit string of encoding to message of decoding and vice versa.

4. Connection of Probability Version

In study of probability, we have to know the basis of the probability such that experiment, outcomes, random experiment, sample space and event. An **experiment** is any process of observation. The result of an observation is called **outcomes**. **Random experiment** is an experiment if its outcome cannot be predicted. The set of all possible outcomes of an experiment is called **sample space**. Then a subset of a sample space is called an **event**.

Examples of Experiment (Random experiment)

Examples of random experiments are the roll of a die, the toss of a coin, drawing a card from a desk and selecting a message signal for transmission from several messages.

For instance, tossing a coin is an experiment with heads or tails as possible outcomes. Each time the coin is tossed is a **trial**.

Example of Sample Space and Events

In the experiment of drawing a number from the numbers 1 through 10. Then the sample space is S={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. The event of drawing an odd number is {1, 3, 5, 7, 9}, the event of drawing an even number is {2, 4, 6, 8, 10}, and the event of drawing a prime number is {2, 3, 5, 7}.

5. Some Tree Diagrams in Probability

In study of some tree diagrams in probability, we have to study the experiment of tossing in a coin. If a coin is tossed once in a time, we have two possible outcomes head and tail. Thus the sample space S={H, T} with the following tree diagram.

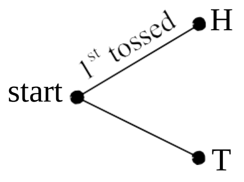

Figure 17: A tree diagram of a coin is tossed once in a time

If a coin is tossed twice, we have four possible outcomes HH, HT, TH, TT. So we get the sample S={HH, HT, TH, TT}as the following tree diagram.

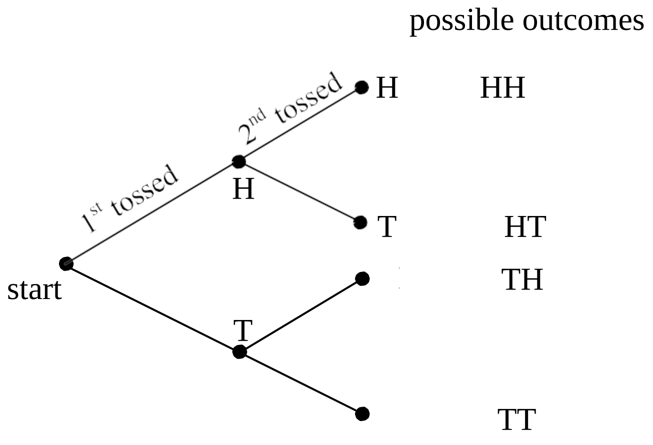

Figure 18: A tree diagram of a coin is tossed twice

Next if a coin is tossed three times, we have eight possible outcomes namely HHH, HHT, HTH, HTT, THH, THT, TTH, TTT. Therefore the sample space S={ HHH, HHT, HTH, HTT, THH, THT, TTH, TTT } as follows.

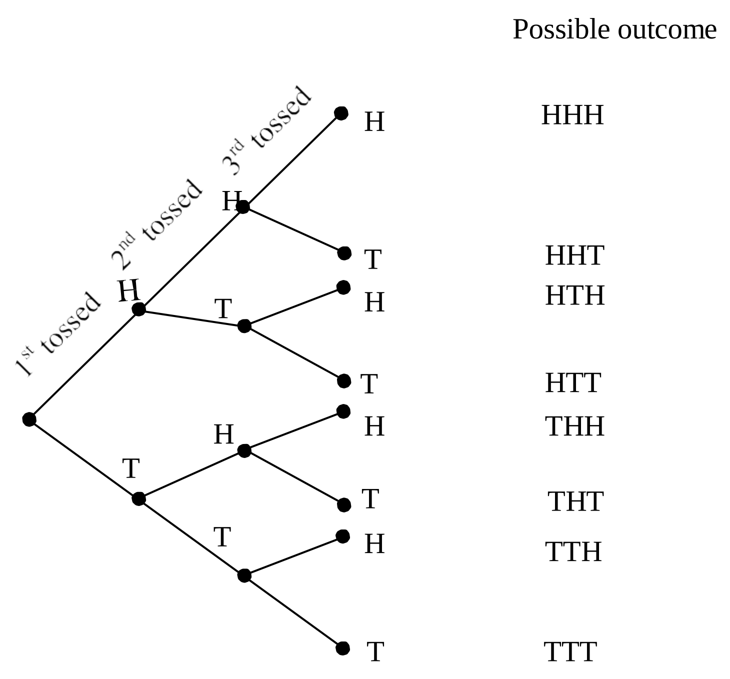

Figure 19: A tree diagram of a coin is tossed three times

Rooted Tree of Figure 17

Look at a tree diagram of figure 17, we can see the vertex at a root from left to right position to top to bottom as follows.

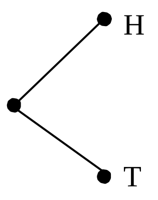

Figure 20(a)

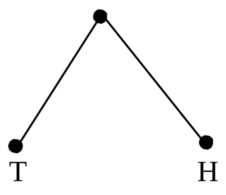

Figure 20(b)

Figure 20(a) and Figure 20(b) show that they are the same. Moreover they are rooted trees as well as binary trees with left and right branches.

Both Binary Tree and Prefix Code Tree of Figure 18

Clearly we can see that figure 18 from left to right position to from up to down position are as follows:

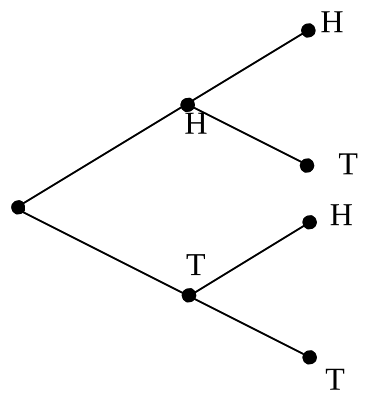

Figure 21(a)

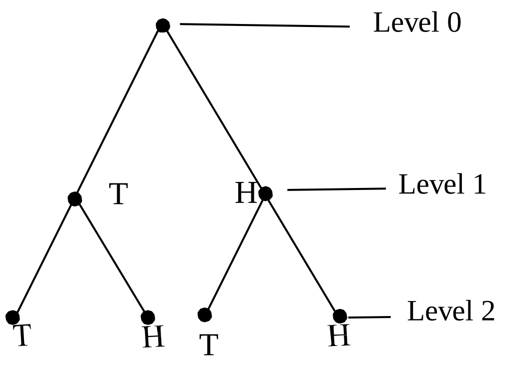

Figure 21(b)

We can transform figure 21(a) to figure 21(b) the root is at the top from left to right position. Therefore they are the same. Each vertex of them shows both left and right branches. Thus they are binary trees. Each binary tree shows 3 level of vertices. So they are also the prefix code tree as follows.

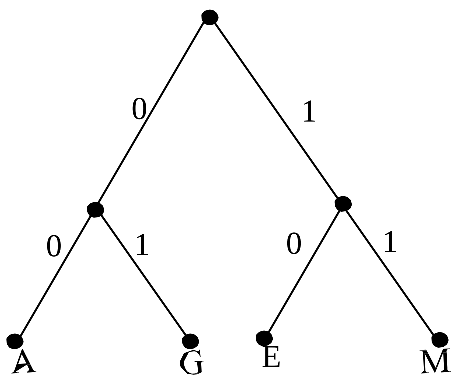

Figure 21(c)**:A** Prefix Code

Relation Between Tree Diagram in some probability trees (Tossing a Coin) and a Prefix Code Tree

In the experiment of some probability trees (tossing in a coin), if we tossed a coin, there are two possible outcomes head (H) and tail (T).

In study of prefix code tree, clearly we see that from upward to downward (top to bottom) position, each level of vertices (or) vertex will be separated left and right branches with Boolean constants 0 and 1 respectively.

Obviously we see that Boolean constant ‘0’ takes place in left branch along the possible outcome tail (T). Boolean constant ‘1’ takes place in right branch along the possible outcome Head (H).

Moreover in study of tossing in a coin. All the possible outcomes are represented by a sample space S. In study of prefix code tree. The vertices which are not separated to the next level of vertices are represented by the alphabets in the set of alphabets V.

Then we have to be continued the information by sending from bit string of encodes to message of decodes according to the encodes and the information by sending from message of decodes to bit string of encodes.

For instance in Figure 21, there are four alphabets A, G, E, M in a given prefix code tree. They are represented by encodes such that the code A is 00, the code G is 01, the code E is 10 and the code M is 11. Again if the given encode is 01 10 11 then we can find the message GEM, it is decode. If the given message is AGE, then we can show that the encode with bit strings 00 01 10.

Transformation of a prefix code tree from figure 19-

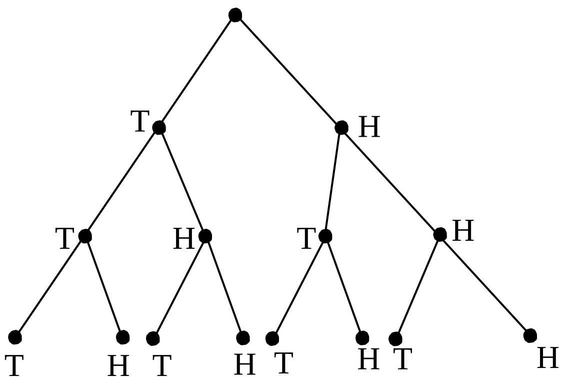

Figure 22: Binary Tree

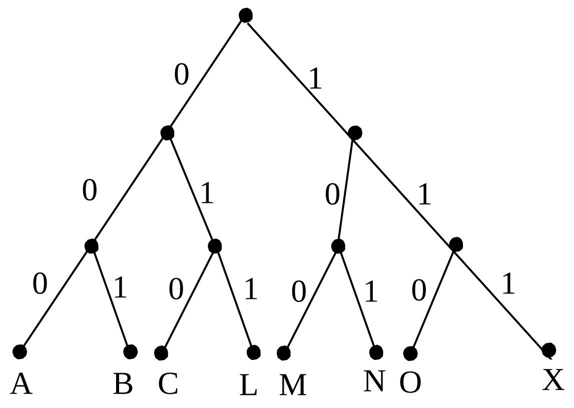

Figure 23: A Prefix Code Tree

Figure 22 shows that binary tree by transformation of figure 19 the root tip is from left to right position to top to bottom position. Figure 23 shows a prefix code tree of Figure 22 by substituting the Boolean constants 1, 0 in place of along the vertices H and T of Figure 22 respectively. For instance, the codes for A is 000, B is 011, C is 010, L is 011, the code M is 100, N is 101, O is 110 and X is 111. We can send the information from encode of bit string 100 000 101 to message of decod namely MAN. Next we can also send message of decode MOON to bit string of encode 100 110 110 101.

Observation

(i) The root tips of from left to right position and from top to bottom position are the same.

(ii) Replace the Boolean constant ‘1’ in place of along the vertex H (success) and the Boolean constant ‘0’ takes place in place of along the vertex T (fail) by transformation of binary tree to prefix code tree.

(iii) Any prefix code tree has at least level 2 vertices.

(iv) Both sender and receiver need to define the secret meaning of each alphabets when the information sends by encodes to decodes and vice versa.

Usefulness

This observation is useful when we would like to send some top secret information or some breaking news from one place to another place over the communications line.

Acknowledgements

My deepest gratitude to my parents and all my teachers in student life, and my Ph.D. supervisor Dr Aye Ko (Retire Professor) too.I would like to thank to rector, pro-rectors and professor (Mathematics Department) in Co-operative university, Thanlyin for the permission of this paper in the ceremony of Co-operative university, Thanlyin.I would like also to thank to all my participants in mathematics department in order to make their helpful for this paper necessary.