Reduction of Logic to Arithmetic

Ranganath G Kulkarni
E mail: kulkarni137@gmail.com
Address: R G Kulkarni
C/o G V Kulkarni
Jambukeshwar Street
Jamkhandi, INDIA
PIN: 587301

Abstract: It is possible to make decisions mathematically of first order predicate calculus. A new mathematical formula is found for the solution of decision problem. We can reduce a logical algorithm into simple algorithm without logical trees.

I
Introduction:

For n number of inputs, is there any mathematical formula that answers yes or no ?[1][2][3] Now it is possible, a mathematical formula[4] is found that can be used to reduce the logical algorithm into simple algorithm without logical trees. This converts logical operation to arithmetic operation. It is a solution for decision problem of first order predicate calculus or first order logic. Therefore in a computer programming we can replace logical test containing thousands of if-else by single mathematical equation using the proposed formula. We can convert decision algorithm into algebraic manipulation or math equation. This changes the design of algorithm and also programming.

II

Theory:

Consider the function $Y=[[(|x|+|c|+x)/(|x|+|c|+c)].(1/e)]$

Where $-\text{infinity}<c<\text{infinity}$

Now, $Y1=|Y/\log Y|$ and $Y2=|Y1/\log Y1|$

For n number of terms $Yn=|Yn\text{-}1/\log Yn\text{-}1|$

Taking limit as n tends to infinity yields, For $x>c$ then $Yn=e$ and $Z=f$

For $x=c$ then $Yn=1/e$ and $Z=g$, For $x<c$ then $Yn=0$ and $Z=h$

This result has no proof, but we can easily verify it. We call Yn as fundamental logic function. We consider Yn as standard math library function.

Where the function Z is given by $Z=f[(Yn-1/e)Yn/(e-1/e)e]+g[(Yn-e)Yn/(1/e-e)(1/e)]+h[(Yn-e)(Yn-1/e)]$

Here f, g and h are functions of x. In logical language, this formula is a solution for decision problem[5][6] of first order logic.This formula can be used to replace decision algorithm by simple algorithm without decision trees.

If a program contains many logical tests then we can convert into single math equation by replacing f, g, h by Z1, Z2, Z3 respectively. Where Z1, Z2 and Z3 are of the same form as function Z. This can be explained clearly as follows.

For the function Z the independent variable is x and is compared with constant c. Here the corresponding logic function is Yn depends on x and c. The mathematical formula Z is expressed as f, g and h with logic function Yn. Where f, g and h are functions of x. Similarly for the function Z1, Z2, Z3 the independent variables are x1, x2, x3 and constants are c1, c2, c3 respectively. The corresponding logic functions are Yi, Yj, Yk respectively. Now the function Z can expressed as

$$Z=Z1[(Yn-1/e)Yn/(e-1/e)e]+Z2[(Yn-e)Yn/(1/e-e)(1/e)]+Z3[(Yn-e)(Yn-1/e)]$$

Here $Z1=f1[(Yi-1/e)Yi/(e-1/e)e]+g1[(Yi-e)Yi/(1/e-e)(1/e)]+h1[(Yi-e)(Yi-1/e)]$

where f1, g1 and h1 are functions of x1. The logic function Yi depends on x1 and c1 respectively. Similarly the formula

$$Z2=f2[(Yj-1/e)Yj/(e-1/e)e]+g2[(Yj-e)Yj/(1/e-e)(1/e)]+h2[(Yj-e)(Yj-1/e)]$$

and $Z3=f3[(Yk-1/e)Yk/(e-1/e)e]+g3[(Yk-e)Yk/(1/e-e)(1/e)]+h3[(Yk-e)(Yk-1/e)]$

A program containing many logical test can be illustrated with the example as follows. In the above formula Z if $x>c$ is true then Z becomes Z=Z1 Here the formula Z1 has logical test, the independent variable x1 is compared with constant c1. The final result will be Z=f1 or Z=g1 or Z=h1 depending upon the value of x1 and c1. Therefore a program containing many logical test can be replaced by single mathematical equation. Therefore by using the proposed formula and optimization techniques we can make elegant programming.

III

Discussion:

Here, by using the function Yn it is possible to express non-closed form expression into closed form expression. It is a solution for decision problem of first order predicate calculus. This formula can be used for reduction of logical algorithm into simple algorithm without logical trees. This reduces logical operation to arithmetic operation. Hence the reduction of logic to arithmetic. By using the proposed formula and optimization techniques we can make elegant programming. It changes the style of designing algorithm and also we can make efficient programming.

IV

Conclusion:

I have found the decision fragment for first order predicate calculus. In a computer programming we can replace logical test by single mathematical equation. Therefore we can make elegant programming.

V

Concept implementation:

We can replace a logical test containing thousands of if else by single mathematical equation. In otherwords we can convert a logical algorithm into algebraic manipulation or math equation. This converts logical operation to arithmetic operation. Here the arithmetic operation is also done by logic circuits but the point is that by using the proposed formula and optimization techniques we can make elegant programming. A complex program canbe executable in an ordinary processor. It saves memory time and energy. We can develop software that works on ordinary processor . Therefore it can be useful for mobile tablet and all computers.

In C programming we can illustrate with example, how logical test can be replaced by a single mathematical equation. Consider a C program, where f, g and h are functions of x. We can write as

```
  main()

{

 float x, c, f, g, h, Z1, Z2,  Z3;

 printf(“Enter numbers x and c “);

 scanf(“%f%f”, &x, &c);

 if(x>c)

 Z1=f;

  printf(“Z1=%f”, Z1);

  else if (x==c)

   Z2=g;

    printf(“Z2=%f”, Z2);

    else

    Z3=h;

    printf(“Z3=%f, Z3);

}
```

Now by using the mathematical formula we can replace logical test by a single mathematical equation. The function Yn has only three values e, 1/e and 0 for values x>c, x=c and x<c respectively. This function is responsible for reduction of logic to arithmetic. We call this function as logic function. For simplicity of the program, we consider the logic function Yn as standard math library function.

```
   main()

 {

    float  x, c, f, g, h, Z;

      printf("Enter numbers x and c ");

        scanf("%f%f", &x, &c);

       Z=f[(Yn-1/e)Yn/(e-1/e)e]+g[(Yn-e)Yn/(1/e-e)(1/e)]+h[(Yn-e)(Yn-1/e)];

         printf("Z=%f", Z);

 }
```

In this program there is no logical test. The fundamental logic function Yn converts logical operation to algebraic manipulation or math equation.

i) Techniques of programming using logic functions:

Now we see in practice how programming can be made easily. Consider the reduction formula

Z=f[(Yn-1/e)Yn/(e-1/e)e]+g[(Yn-e)Yn/(1/e-e)(1/e)]+h[(Yn-e)(Yn-1/e)]

The fundamental logic function Yn(x, d) depends on variable x and constant d. Here , we have denoted constant as d, because we are taking c for the next section. We can express the reduction formula as

Z=af+bg+ch

Where a=[(Yn-1/e)Yn/(e-1/e)e] , b= [(Yn-e)Yn/(1/e-e)(1/e)] and c=[(Yn-e)(Yn-1/e)]

Here a, b and c are functions of the fundamental logic function Yn(x,d). Therefore a, b and c are also logic functions. They have only two values either 0 or 1. We consider logic functions a, b and c as standard math library functions. By using logic functions a, b and c we can replace logical trees by a single mathematical equation.

if x>d then a=1, b=0, c=0 and Z becomes Z=f

if x=d then b=1, c=0, a=0 and Z becomes Z=g

if x<d then c=1, a=0, b=0 and Z becomes Z=h

The logical test has three possibilities, greater than, equal to and less than. Here the term a indicates greater than , the term b indicates equal to and the term c indicates less than.

The programming can be made easily is illustrated with two examples.

1)Consider a C program, p, q, r, s and t are functions of x. d1, d2, d3 are constants, where d1>d2>d3

In this program we have not mentioned d1, d2, d3 in the declaration, because they are constant numbers.

```
         main()

  {

     float x, p, q, r, s, t, Z1, Z1', Z2, Z3, Z4;

     printf("Enter the number x");

     scanf("%f",  &x);

     if(x>d1)

      Z1=p;

      printf("Z1=%f",  Z1);

     else if (x==d1)

       Z1'=q;

      printf("Z1'=%f",  Z1');

      else if (x>d2)

      Z2=r;

      printf("Z2=%f", Z2);

      else if (x>d3)

       Z3=s;

       printf("Z3=%f", Z3);

       else

         Z4=t;

         printf("Z4=%f", Z4);

          }
```

We can make programming easily by using logic functions a, b and c, which have value either 0 or 1. In the above C program we have first logical test if x>d1, the fundamental logic function for the variable x and constant d1 is Yi(x, d1). The corresponding logic functions are a1, b1 and c1. Here if x>d1 then a1=1, b1=0, c1=0 else a1=0. In the second logical test if x=d1 then b1=1, c1=0, a1=0 else b1=0. In the third logical test if x>d2 the fundamental logic function for the variable x and constant d2 is Yj(x, d2). The corresponding logic functions are a2, b2 and c2. Here if x>d2 then a2=1, b2=0, c2=0 else a2=0 The fourth logical test is if x>d3, the fundamental logic function for the variable x and constant d3 is Yk(x, d3). The corresponding logic functions are a3, b3 and c3. Here if x>d3 then a3=1, b3=0, c3=0 else a3=0.

We can replace the logical trees of the above C program by single mathematical equation and is given by

$$Z=(a1p+b1q)+c1[a2r+(b2+c2)(a3s+(b3+c3)t)]$$

This equation looks complicated , but it is simple to formulate. Now, we see how the program can be expressed in a single math equation. We can express this formula as

$$Z =a1p+b1q+c1h$$

Here h=a2r+(b2+c2)l,   where l=a3s+(b3+c3)t

The equations Z, h and l are of the form of reduction formula Z=af+bg+ch

The reduction formula has three terms. Here the term a indicates greater than, the term b indicates equal to and the term c indicates less than.

In the above C program , we have first logical test if x>d1 is true then a1=1, b1=0, c1=0 therefore the first term is a1p. the second logical test is if x=d1 then b1=1, c1=0, a1=0. Therefore the second term is b1q. If both test fails we have the third term c1h. Here, the function h contains logical test . If x>d2 is true then the first term is a2r If x>d2 is not true then there is logical test either x=d2 or x< d2. But there is no logical test for both. Therefore the second and third term is b2l and c2l respectively. The function l contains logical test , if x>d3 is true then the first term is a3s. If this test fails , then there is logical test either x=d3 or x<d3. But there is no logical test for both. Therefore the second and third term is b3t and c3t respectively. The function t does not contain logical test. Therefore the program finally terminates.

Now, we can write C program in a single mathematical statement by using logic functions (a1, b1, c1), (a2, b2, c2), (a3, b3, c3)

```
      main()

   {

      float x, p, q, r, s, t, Z;

       printf("Enter the number x");

       scanf("%f", &x);

       Z=(a1p+b1q)+c1[a2r+(b2+c2)(a3s+(b3+c3)t)];

       printf("Z=%f", Z);

    }
```

2)Consider another example that illustrates how to write a C program in a single mathematical statement containing AND and OR operation.

Consider a C program , p, q, r, s are functions of x. The variables are x, x1, x2, x3, x4 and d, d1, d2, d3, d4 are constants.

In this program we have not mentioned d, d1, d2, d3, d4 in declaration , because they are constant numbers.

```
            main()

  {

           float x, x1, x2, x3, x4, p, q, r, s, Z1, Z2, Z3, Z4;

           printf("Enter the numbers x, x1, x2, x3, x4");

           scanf("%f%f%f%f%f",   &x, &x1, &x2, &x3, &x4);

          if (x>d && x1>d1)

             Z1=p;

            printf("Z1=%f",  Z1);

          else if (x2>d2  || x3>d3)

             Z2=q;

             printf("Z2=%f", Z2);

          else if (x4>d4)

             Z3=r;

             printf("Z3=%f", Z3);

              else

               Z4=s;

              printf("Z4=%f", Z4)  ;

               }
```

Now, we see how to write the above C program using logic functions. The fundamental logic functions for logical tests x>d, x1>d1, x2>d2, x3>d3 and x4>d4 are Yi(x, d), Yj(x1, d1), Yk(x2, d2), Yl(x3, d3) and Ym(x4, d4) respectively. The corresponding logic functions are (a, b, c), (a1, b1, c1), (a2, b2, c2), (a3, b3, c3), and (a4, b4, c4) respectively.

Consider the first logical test of the above C program which is if(x>d && x1>d1) Here, if both logical tests are true then the result is true . The comparison x>d indicates a and the x1>d1 indicates a1. Therefore the result which is true is given by aa1. This is the only term valid for AND operation. But the logical test has three possibilities greater than, equal to and less than. Therefore expanding a to a+b+c and a1 to a1+b1+c1 Now aa1 becomes

$$(a+b+c)(a1+b1+c1)=aa1+a(b1+c1)+a1(b+c)+(b+c)(b1+c1)$$

Here the only term aa1 is true if both logical tests are true. Therefore the remaining terms indicates false result of AND operation. We can express for both true and false result of AND operation as

$$Z=aa1p+[a(b1+c1)+a1(b+c)+(b+c)(b1+c1)]h \qquad \ldots\ldots\ldots\ldots\ldots\ldots..1)$$

Here the function h contains logical test OR operation. The logical test OR operation has if (x2>d2 || x3>d3) Here the comparison x2>d2 indicates a2 and the comparison x3>d3 indicates a3. In OR operation if either or both of the logical test is true then the result is true.

Assuming a2 is true, the second logical test of OR operation has three possibilities a3 or b3 or c3. For this combination the true result of OR operation has terms a2(a3+b3+c3). Now, assuming a3 is true, the first logical test of OR operation has three possibilities a2 or b2 or c2. For this combination the true result of OR operation has terms a3(a2+b2+c2). Therefore, the terms of OR operation for which the result is true is given by

$$a2(a3+b3+c3)+a3(a2+b2+c2)$$

Here the term a2a3 is repeated twice . Therefore we take only one term , this becomes

$$a2(a3+b3+c3)+a3(b2+c2)$$

The all terms containing true and false result of OR operation is given by expanding a2a3 to

$$(a2+b2+c2)(a3+b3+c3)=a2(a3+b3+c3)+a3(b2+c2)+(b2+c2)(b3+c3)$$

Now, we can express the function h for both true and false result of OR operation as

$$h=[a2(a3+b3+c3)+a3(b2+c2)]q + (b2+c2)(b3+c3)l \qquad \ldots\ldots\ldots\ldots\ldots..2)$$

Here the function l contains logical test if(x4>d4). The comparison x4>d4 indicates a4. If a4 is true then the result is given by a4r. If the test fails then the logical test has two possibilities either b4 or c4. But there is no logical test for both. Therefore if (x4>d4) is not true then the result is given by (b4+c4)s. Now, we can express the function l for both true and false result of the logical test is given by

$$l=a4r+(b4+c4)s \qquad \ldots\ldots\ldots\ldots\ldots..3)$$

Here the function s does not contain logical test . Therefore, the program finally terminates. Substituting the function l in the equation 2) and then substituting the function h in equation 1) The single mathematical statement of the above C program is given by

$$Z=aa1p+[a(b1+c1)+a1(b+c)+(b+c)(b1+c1)]\{[a2(a3+b3+c3)+a3(b2+c2)]q+(b2+c2)(b3+c3)(a4r+(b4+c4)s)\}$$

The above C program written in a single mathematical equation is given by

```
 main()

 {

      float x, x1, x2, x3, x4, p, q, r, s,  Z;

      printf("Enter the numbers x, x1, x2, x3, x4");

      scanf("%f%f%f%f%f",  &x, &x1, &x2, &x3, &x4);


Z=aa1p+[a(b1+c1)+a1(b+c)+(b+c)(b1+c1)]{[a2(a3+b3+c3)+a3(b2+c2)]q+(b2+c2)(b3+c3)(a4r+(b4+c4)s)};

      printf("Z=%f", Z);
```

```
  }
```

ii)Program is readable :

Consider the formula   Z=af+bg+ch

The fundamental logic function Yn(x, d) depends on the variable x and constant d. The logic functions are a, b and c, which have value either 0 or 1. Here the comparison x>d, x=d and x<d indicates a, b and c respectively. Therefore a1 indicates either x1>d1 or x>d1. Similarly c1 indicates either x1<d1 or x<d1 In otherwords the term a indicates greater than, the term b indicates equal to and the term c indicates less than. We can say in mathematical language that if a is true then Z=f. Here a is true means the logical test x>d is true. Therefore the value of a becomes a=1 and Z becomes Z=f

A logical test containing simple if-else or if-else with AND and OR operation can be readable, if we are familiar with how to write in a mathematical statement. For example, if we are not familiar with elementary formulas of calculus then we are unable to perform differentiation and integration. Therefore we should be familiar with the mathematical statement of simple if-else or if-else with AND and OR operation etc.

Now the question arises about how to read the program containing thousands of if-else written in a single mathematical statement. This can be resolved. It is known that if the program is readable then we can debug. To make the program readable written in a single mathematical statement, we have to divide the single math equation. The procedure is that, the mathematical statement of first logical test of the program is written in first line. The mathematical statement of second logical test is written in second line and so on. This makes the program readable. It can be explained clearly with the second example of C program containing AND and OR operation. Here, we write the C program in a readable format.

```
      main()

{

      float x, x1, x2, x3, x4, p, q, r, s,Z;

      printf(“Enter the numbers x, x1, x2, x3, x4”);

     scanf(“%f%f%f%f%f”,  &x, &x1, &x2, &x3, &x4);

     Z=aa1p+[a(b1+c1)+a1(b+c)+(b+c)(b1+c1)]*

     * {[a2(a3+b3+c3)+a3(b2+c2)]q+(b2+c2)(b3+c3)*

     *{(a4r+(b4+c4)s)}};

      printf(“Z=%f”, Z);

}
```

Here the * sign indicates multiplication and also it connects mathematical statement of next logical test. The mathematical statement of the first logical test is AND operation and it is written in first line. The mathematical statement of second logical test is OR operation and it is written in second line. The mathematical statement of third logical test is simple if-else and it is written in third line. In this way we can express a program containing many logical tests in a readable format. Therefore we can debug the program.

But the expression for single mathematical statement written in a readable format is not valid according to the rules of C programming. Therefore we have to change the rules of C programming for readable format of single mathematical statement.

References:

[1] Turing, A.M. 1936.‘On Computable Numbers, with an Application to the Entscheidungsproblem’. Proceedings of the London Mathematical Society , series 2, 42 (1936-37), 230-265.

[2] Church, A. "A Note on the Entscheidungsproblem." J. Symb. Logic 1 , 1936.

[3] Gödel, K. (1931), ‘On Formally Undecidable Propositions of Principia Mathematica and Related Systems I’, Monatschefte für Mathematik und Physik 38, pp. 173–198

[4] R G Kulkarni, Expression for the Mathematical Constant e, viXra:1109.0054
(see, http://vixra.org/abs/1109.0054)

[5] B.J. Copeland, The Essential Turing, Publisher Clarendon Press, 2004

[6] Mathematical Problems. Lecture delivered before the International Congress of Mathematicians at Paris in 1900, English translation of Hilbert (1900) by Mary W. Newson, Bull. Amer. Math. Soc. 8 (1902), 437–479