Marketing Transforms in the CIS Curriculum

Jerry Chin, Mary Chin and Michelle Hulette

Abstract

The traditional university business curriculum from accredited business programs is approximately one-half of the degree credits. In this paper, we hypothesize that class work outside of the college of school of business (General Education) can be a source of material for problems within the computer information curriculum. In particular, this paper uses the Truth Functional Form Algorithm from a logic course, normally targeted for students in philosophy, computer science, or mathematics. The problem is reduced to a string search problem with a computer programming solution and serves as a simple logic example for marketing students, broadening their university experience. The actual C# code is provided with the user interface in the appendix of the paper.

Introduction

The Northwest Center for Emerging Technologies (NWCET) published a report in 2000 titled "Building a Foundation for Tomorrow". In this report they list several categories of worker readiness skills sought by nearly every employer. In the Core knowledge category required skills include analytical skills and problem solving, business organization and environment, coordination and communication skills, core computer and hardware/software skills, and project and process flows.

The expansion of electronic commerce has elevated the importance of business insight; many computer and information systems managers are called on to make important business decisions. A graduate entering the workforce should be prepared to use technology, but they should understand that development of a strategy for solving a problem is the higher priority. Greater success can be found with focus on the solution first, then implementing technology to achieve the desired result, rather than finding a technology that appears to be a solution to a problem that is not well defined. In this paper, we suggest that understanding and knowing the basics of logic, can enhance problem-solving capability.

Mary's Problem

Mary has just left a Monday morning meeting where the usual marketing heads discuss upcoming events and strategies. Her notes indicate that there is some concern about the marketing data in the South. Given the history of the product, it is generally agreed that if the product is a success in Florida, then the southeast region has been properly analyzed. Sales figures for Jackson, Mississippi have been good and have historically been a leading success indicator for Florida. She quickly volunteers that the product is a success in Florida and that the southeast region is now in a stable position for the product.

Mary's background in the liberal arts exposed her to fundamental logic courses most philosophy department offers its undergraduates. A late night search of her old books in the attic brings back buried concepts of testing statements for truth. She wonders if there is a more systematic way to formulate more complex problems of this nature. What is her conclusion for next week?

A Solution For Business

One of the advantages of a presentation of formal logic is that there is an underlying structure of rules that the student must follow in order to judge whether the conclusion is valid. We normally look for structure in the sciences: chemistry, biology, especially, mathematics. In the business curriculum, students study systems in accounting, finance and information systems. The ultimate question remains: Is our decision valid?

Short of a major or even minor in a field of study, the discussion of problems of where structure and rules are present can arguably contribute to the strengthening of students' critical thinking skills, reasoning powers and creativity. Moreover, it can also provide the opportunity to students to appreciate other disciplines outside the business school. While this exercise may not always be practical to the general business management student, the intellectual process offers students that opportunity to see rigor and alternative strategies in problem solving from other academic fields.

In this paper, we present a string search problem that is based in first order logic. In particular, the application of this type of search and classification problem is the parsing and manipulation of character strings, so important now in information searches.

In addition, we present a short hand for this problem that provides better precision to the search for a solution to the problem. In summary, the process for our problem is:

- Reading the statement in question.
- Reformulating the statement-if needed - using symbolic logic.
- Using a simple C# program to transform the statement into an even simpler form.
- Determining if the transformed statement is a tautology: a logical statement that is always true.

Problem solving in business is fundamental to the survival to the business. In a very practical sense, the tool is not as important as the solution to the problem. There are many problem solving models based in mathematics, psychology, philosophy, and engineering. What is fundamental to these models is that there is always some analysis that offers the student or investigator some clarity of the process toward a result. What we propose is that critical thinking can be stimulated by borrowing from other fields. In addition, we also propose that the business curriculum include problems and techniques across the campus (Chin *et al.*, 2005).

Mary's Initial Analysis

The problem is whether Mary's marketing statement is true. Logically speaking, the question is: Is the statement a tautology? That is, is the question always true? We first make some observations:

- The problem at hand is well-defined.
- We see that this problem can be solved if we answer another question: Is the marketing problem a symbolic logic problem?
- We can visualize a solution by traversing through the string sentence and simplifying the problem.
- The problem has an algorithmic solution if we can derive a way to explicitly reformulate the problem by a series of steps.
- If this problem has a solution, then similar problems can be solved. Thus, we can add to the institutional memory.

We view a sentence under consideration as a string of characters. This is done in string search techniques in computer and information systems. For brevity and clarity, a discussion of logic rules and concepts involving tautologies is not done. The first order of business is to discuss a way to transform (logic) sentences into simpler forms, so that it can be tested for truthfulness at a later time.

Truth Functional Form Algorithm

In logic a tautology is a logic sentence that is always logically true. Given a logic sentence, a natural question arises: Is this a tautology? That is, is this sentence always true? One way to make the problem simpler is to reduce the size of the problem. One way is to reduce the form of the given logic sentence. This can be accomplished by the Truth Functional Algorithm. If we believe that a sentence is a tautology if and only if its truth-functional form is a tautology, then our problem is reduced to looking at truth-functional forms.

Consider the following string:

$$(\exists y(Py \vee Ry) \rightarrow \forall x(Px \wedge Qx)) \rightarrow (\neg\forall x(Px \wedge Qx) \rightarrow \neg\exists y(Py \vee Ry))$$

We verbalize the search process. "Start at the beginning of the string and proceed to the right. If there is a quantifier ($\exists, \forall$) or an atomic sentence (e.g. Py), begin to underline. If you start with a quantifier, underline it and its corresponding formula. This will either be enclosed in parentheses (e.g. $\neg\exists y(Py \vee Ry)$ or just an atomic sentence (e.g. $\exists yPy$). Assign each underlined constituent a letter (A, B, C...). If a prior identical constituent used the same letter, use that same letter. Otherwise, use a new letter." The resulting functional form is:

$$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A).$$

Notation and Convention

We need a convention to form sentences in logic. Predicate symbols are used to express a relation between objects. For example, Home(Mary) might mean "Mary is a home". If we abbreviate further where H = Home and m = Mary, we could just as easily write Hm. The relation "Bill is taller than Sue" might be expressed by Taller(Bill,Sue), or $T(b, s)$. The $\neg$ symbol, is used to mean "not". If Bill is not taller than Sue, then we can write $\neg T(b, s)$. Predicates will use capital Roman letters A to Z Constants will be represented by lower case Roman letters, a to v. Lowercase Roman letters w to z will be variables. An example using this notation is:

Lxy: x likes y
h: harry
s: sue

Quantifiers.

The Universal Quantifier?. The combination? w reads as "for each object w". If we add a predicate Lhs, "Harry like Sue", Harry likes everyone can be written as $(\forall x)$Lhx. Each x is such that Harry likes them.

The Existential Quantifier $\exists$. The combination $\exists w$ reads "for some object w". If we add a predicate Lhs, "Harry like Sue", we get something like $(\exists x)$Lxs. We read this as "Someone that likes Sue". There exists an object x that like Sue.

Using the above notation convention, we can construct string sentence such as:

$$\neg(Td \wedge \forall xSx) \rightarrow (\neg Td \vee \forall ySy)$$

Note that the term Td has no quantifier ($\forall$ or $\exists$).

The Algorithm

One step students are encouraged to do before writing code is to use the original flowchart. This preliminary step is to outline an attack on the problem at hand. In this paper, we suggest that this early head scratching be done with a collection of symbols and notation that provides a way to describe key characters and traversing along the string itself.

String functions and notation.

Let α = the current character of the string. We will use a LIFO stack to store and separate the string constituents as they are determined. $\Pi \downarrow (z)$ means push the string z down onto a stack. The stack retrieving order will be last in - first out. $\Pi \uparrow$ means pop up a string from the stack. $(\wedge X)$ means look for the character X. We have the following sets which are used to

$$\begin{aligned} \text{Quantifiers } L &= \{\forall, \exists\} \\ \text{Variables } \quad V &= \{t, u, v, w, x, y, z\} \\ \text{Constants } C &= \{a, , s\} \\ \text{Predicates } A &= \{A.B, C, \} \end{aligned}$$

Fundamental Analysis

Consider the string s. There are three distinct cases to consider.

1) If $s(\alpha) \in$ (read $\in$ to mean "is a member of") and $s(\alpha + 3) \in V$, Then$\Pi \downarrow (s(\alpha, \alpha + 3))$

For example suppose $s = \forall xRx$ and the current character indicator α is pointing at the character $\forall$. The three characters down the string would be $\alpha + 3$. So, $s(\alpha + 3) = x$, which is a member of V. In this case, we would we push onto the stack all the characters of the

string, $s = \forall xRx$. That is, $\Pi \downarrow (s(\alpha, \alpha + 3))$. The C# code is listed in the Appendix.

2) If $s(\alpha) \in L$ and $s(\alpha + 2) =$ '(' then λ(')'). Assume $s(\omega) =$ ')'. Then $\Pi \downarrow s(\alpha, \omega)$. Consider the string $s = \exists x(Px \vee Qx)$. Suppose at location α we have $s(\alpha) = \exists$ and $s(\alpha + 2) =$ '('. Then look for the closing parentheses, $s(\omega) =$ ')'. Push all the characters from α to ω onto the stack.

3) If $s(\alpha)\Box A$, $\Pi \downarrow (s(\alpha, \alpha + 1))$. If $s =$ Ba, the push Ba onto the stack. The last statement is:

$$\Pi \uparrow \text{ and assign letters from A.}$$

This means pop the stack and assign letters $\{A, b, c, \ldots\}$. At this point, the string s has been parsed and its components have been separated and pushed onto a stack. Now as each constituent of the original string s is popped off the stack, it is assigned an uppercase Roman letter. If a component matches a previous component, assign the same letter as before. Now, finally go through the original string and replace each component with its assigned letter. We are through with the transformation of the string.

A screen shot of a C# program that uses the truth-functional algorithm is provided in the appendix. In this example, the example string with the sentence,

$$(\exists y(Py\Box Ry) \rightarrow \Box x(Px\Box Qx)) \rightarrow (\neg\Box(Px\Box Qx) \rightarrow \neg\Box y(Py\Box Ry))$$

has the truth-functional form: $(A) \rightarrow (C \rightarrow B)$. The right-hand side shows the string constituents that are pushed and popped off the data stack.

With this shortening of the original logic sentence, the question of whether the original sentence is a tautology can be answered by asking if the shortened logic sentence is a tautology.

Mary's Analysis

Using the above notation, Mary's problem can be formulated with the following predicates:

$$\begin{aligned}[\text{Success(Florida)} &\rightarrow \text{Analysis (Southeast)} \\ &\wedge \text{Success(Mississippi)} \\ &\rightarrow \text{Success (Florida)}] \\ &\rightarrow [\text{Success(Florida)} \\ &\rightarrow \text{Analysis(Southeast)}].\end{aligned}$$

Using the Truth-Functional Algorithm, the above sentence can be reduced to the following representation:

$$[A \wedge B] \rightarrow A.$$

While truth-tables were not discussed in this paper, this sentence is a tautology established by its truth table. Mary was right all along.

Conclusion

We have shown that pursuit of the truth-value of a simple business question can stimulate critical thinking and problem solving. The search led to the formulation of a symbolic logic question. The analysis gave way to three cases when one parses the string statement. These three cases could be explicitly expressed using symbols and notations. This step indicated that the solution was algorithmic in nature and a C# coding implementation was at hand. Therefore, a tool has been devised to measure if a statement is a tautology, that is, true all the time.

References

Barwise, J. and Etchemendy, J. (2003) Language Proof and Logic in collaboration with G. Allwein, D. Barker-Plummer, and A. Liu. Stanford, CA: CSLI Publications.

Chin, Jerry M. and Chin, Mary H., A Marketing Syllogism for CIS Business Students. Ethics & Critical Thinking Journal. September **15**, 2005, pp 16–26.

Larson, Todd. The Practical Value of the Liberal Arts. Retrieved from `http://www.haverford.edu/publications/winter99/busarticle.htm` on June 15, 2006.

Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, 2006-07 Edition, Computer and Information Systems Managers, on the Internet at http://www.bls.gov/oco/ocos258.htm (visited June **30**, 2006).

Missouri State of the Workforce Report 2004. March 31, 2004 http://www.ded.mo.gov/employment/mtec/pdf/swr.pdf

Appendix

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace string_search
{
 public partial class Form1 : Form
 {
  Stack <string> fconstiuents = new Stack<string>();
  Stack<string> aconstiuents = new Stack<string>();
  int charPtr;
  string[] Letter = {"A", "B", "C", "D", "E"};
  string truthfunctionalform;
  int StrDelta;
  public Form1()
  {
     InitializeComponent();
  }
  private void TestData_Click(object sender, EventArgs e)
  {
     richTextBox1.Text =

            ∃ y(Py ∨ Ry) → ∀ x(Px ∧ Qx)) → (¬∀ x(Px ∧ Qx) → ¬∃ y(Py ∨ Ry));


  }

  private void Parse_Click(object sender, EventArgs e)
  {
     string s  = richTextBox1.Text;
     string hold = "";
     charPtr = 0;
     StrDelta = s.Length;
```

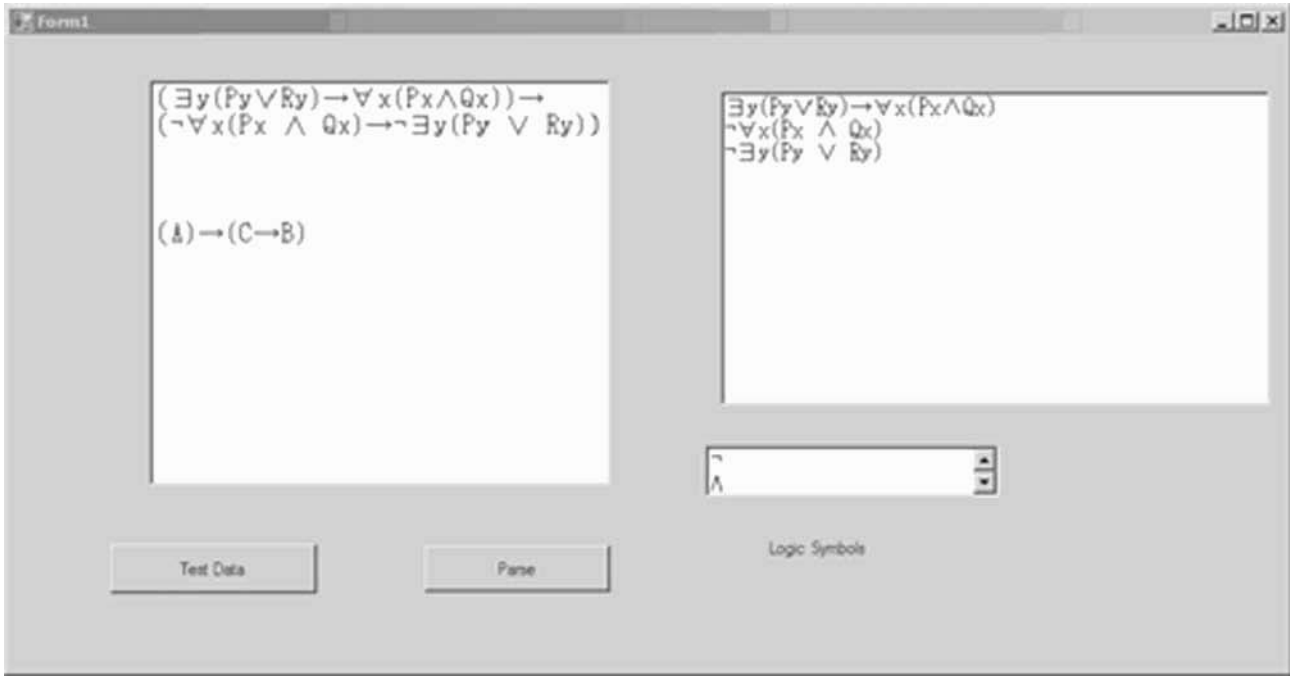
```
//"R(a)" is 4 characters long and smallest constituent. If less than 4,
//Then rest of string cannot have anymore atomic constients.
while (StrDelta > 3)
  {
     // check for formula
     Boolean formula = s[charPtr] == '(' &
     (s[charPtr+1] == '□' || s[charPtr+1] == '□');
     Boolean z = s[charPtr] == '□' || s[charPtr] == '□';
     if(formula)
      {// formula
         hold = do_formula(s);
         if(fconstiuents.Contains(hold) == false)
         {
            fconstiuents.Push(hold);  // push onto stack
            richTextBox2.Text = richTextBox2.Text + hold + '\n';
          }
      }
     else if (z)
      {
         //atomic with quantifier
         hold = atomic(s);
         if (aconstiuents.Contains(hold) == false)
          {
            aconstiuents.Push(hold);  // push onto stack
            richTextBox2.Text = richTextBox2.Text + hold + '\n';
          }
       }
        else if (z == false & s[charPtr] != '(' )
         {// atomic sentence with no quanifier
           hold = no_quantifier(s);
           if (aconstiuents.Contains(hold)== false)
            {
              aconstiuents.Push(hold);  // push onto stack
              richTextBox2.Text = richTextBox2.Text + hold + '\n';
            }
         }
          charPtr++;  // next character
       }
       // replace strings from 2 stacks
          int i = 0;
     while(fconstiuents.Count > 0)  //do formulas first
     {
       string s2 = fconstiuents.Pop();
       string truthfunctionalform = s.Replace(s2, Letter[i]);
       s = truthfunctionalform;
       i++;
```

```
          }
            int n = 0;
          while (aconstiuents.Count > 0)
          //now do atomic which may be in formulas
           {
              string s2 = aconstiuents.Pop();
              truthfunctionalform = s.Replace(s2, Letter[n+i]);
              // Letter[i]'s above used
              s = truthfunctionalform;
              n++;
           }
              richTextBox1.Text = richTextBox1.Text + truthfunctionalform;
}

   private string atomic(string s)
    {
            int I = s.IndexOf(')',charPtr);
            string hold = "";
            for ( int k = charPtr; k <= I; k++)
            {
                hold = hold + s[k];
            }
            StrDelta = s.Length - I-1;  // how far down the string ?
            charPtr = I+1;
            return hold;
    }
    private string no_quantifier(string s)
    {
            int I = s.IndexOf(')',charPtr);
            string hold = "";
            for (int k = charPtr; k <= I; k++)
            {
                hold = hold + s[k];
            }
            charPtr = I+1;
            StrDelta = s.Length - 2-I;
            return hold;
        }
private string do_formula(string s)
        {
            int I = s.IndexOf("))", charPtr);
            string hold = "";
            for (int k = charPtr+1; k <= I+1; k++)
            {
                hold = hold + s[k];
            }
```

```
                charPtr = I+2;
                StrDelta = s.Length - I-2;
                return hold;
            }
    }
}
```

Figure 1.
Screen Shot of Truth-Functional Algorithm C# program



ABOUT THE AUTHORS

Jerry Chin is Department Head and Professor of Computer Information Systems in the College of Business Administration at Missouri State University. He holds a Doctor of Business Administration from the University of Memphis and degrees from Delta State University and Mississippi State University. His current interests are computer applications and teaching methodologies within computer information systems.

Mrs. Mary H. Chin is currently Instructor of Marketing in the College of Business Administration at Missouri State University. She earned her MBA from Missouri State University and her BS in finance from the University of Memphis. Her current interests are marketing strategies and their implications within the undergraduate business curriculum.

Mrs. Michelle Hulett is currently Instructor of Computer Information Systems in the College of Business Administration at Missouri State University. She holds degree in MIS from the university of Arkansas and MBA from Missouri State University. Her teaching and research interests are Web development and computer applications. She is the recent author of several textbooks on Microsoft Office.