ABSTRACT

This paper is a demonstration of the use of logical spreadsheets in the Microsoft Excel environment. Rather than using the traditional IF statements, ARulesXL software uses rules that are entered directly on the worksheet. The rules and syntax are simple to understand, which allows makes the application easier to understand and maintain. Teaching this approach to business rule applications combines decision support with computational analysis.

Introduction

Custom Widgets has customers in the 25 towns and cities such as Boston, Los Angeles, and Miami. Each city has specific requirements for a widget. Therefore, a price per unit differential is calculated as well as shipping cost. Recently, a surcharge was also determined to cover the cost of training the sales personnel. Mary is a newly hired MBA with Excel skills and has been assigned to create an application that will automatically calculate costs for customers as they place their orders over the phone. She decides to keep this assignment out of the MIS group as all the users will be from the Sales department. Moreover, history has shown that many of the costs are calculated by the Sales group and change almost on a monthly basis. In addition, there is a moving discount based upon past business.

Mary has several alternatives available to her. She believes any of the approaches can provide the users with the information required by the Sales Department to serve its customers. Her analysis includes the use of a logical spreadsheet.
Logical Spreadsheets

The relationship between logic and programming is very close. The use of if-then statements are fundamental to transforming natural language statements into propositions and then to the equivalent programming if-then statements. The code in Figure 2 is a simple example of that process. This process to analyze and provide solutions is a common experience in the IT world. But, what about the user in another area such as finance or accounting or even biology where spreadsheets are a common tool?

The purpose is to introduce in a simple way the use of spreadsheets using a logic foundation built upon the logic Prolog and a commercial product named ARulesXL (“Add Business Rules to Your Excel Spreadsheets,” n.d.). The first workshop on the spreadsheet paradigm and symbolic logic was held at Stanford in the Fall 2005 term. The aim was to investigate the automation of the decision making process for users by symbolic reasoning just as current spreadsheet users make decisions based upon numeric data.

The proponents of the software ARulesXL claim that using rules is better because rules are:

• Easier to code,
• Easier to maintain,
• More reliable, and
• Easily audited

These claims will be demonstrated by our example below.

Mary’s Solution

Mary’s fact finding mission and analysis brings her to the following partial information:

<table>
<thead>
<tr>
<th>City</th>
<th>Cost per Unit</th>
<th>Surcharge</th>
<th>Shipping cost</th>
<th>Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>25</td>
<td>1125</td>
<td>375</td>
<td>1.25%</td>
</tr>
<tr>
<td>Dallas</td>
<td>10</td>
<td>150</td>
<td>500</td>
<td>2.19%</td>
</tr>
<tr>
<td>Miami</td>
<td>15</td>
<td>135</td>
<td>450</td>
<td>1.20%</td>
</tr>
</tbody>
</table>

Mary’s sample calculation for 100 units for Miami is:

\[
\begin{align*}
\text{Total} &= 15 \times 100 + 135 + 450 \\
\text{Discount} &= 2085 \times 0.0120 = 25.02
\end{align*}
\]

Since Excel allows up to seven nested If statements, Mary is certain she could implement this application as an Excel spreadsheet. Her spreadsheet looks like:
A quick trip to the IS department and the redemption of an owed favor results in a quick VBA solution for her to consider. The VBA code is stored as a macro (Gross, Akaiwa, & Nordquist, 2006; see also Jacobson, 2001) and executed after the user makes the appropriate entries in the cells designated for user input. She is not a coder and the behind-the-scenes process is totally transparent to her. An example of the type of code sample that was returned to her is below in Figure 2.

![Figure 2](image_url)

We now take a quick tour of the software. Some of the features of ARulesXL are briefly described below. We will cover the following points:

- Input
- Variables
- Rules
- Multiple Rules
- Rquery
  - When \( x = _1 \), D1
- RRRef

Figure 3 is a diagram to aid in the explanation of the concepts below.
**Property**

What is an object/property? This is a value that is to be found based upon information or rules. It can be input from the user or determined by reasoning from a query and the associated rule set. Properties are designated by a leading period. For example,

```
.price
product.price
product[sales].price
```

are examples of properties which can be assigned values. Input from a user can be captured at the cell level. One way to designate the cell location in CR notation is through a “query”.

**Query**

Queries can take on many forms. For example, such a query might look like:

```
RQUERY(RuleSetName,” FIND .price WHEN .tax = _1”, C2)
```

In this query, RuleSetName is actually what it is, the name of a set of rules that a query may use to calculate or find a specific property, in this case .price. Cell at address C2 holds the value for the property, .loc, which is used by one or more rules to find .price.
A rule set begins with the rule set name and ends with an “*”.

```
RULE1
.price = 25 WHEN .loc = “Boston”
taxrate = .08 WHEN .loc = “Boston”
.price = 10 WHEN .loc = “Dallas”
taxrate = .10 WHEN .loc = “Dallas”
*
```

The condition WHEN is used to form a rule equivalent to the familiar if-then conditional:

If .loc = “Boston” then .price = 25.

If there is an assignment without a WHEN condition, the RREF construct can be used:

```
RREF( .price = 40)
```

Queries can be used to find more than one property:

```
RQUERY(RuleSetName,”FIND .price AND .total WHEN .tax = _1 AND .units = _2”, C1, C2).
```

Here, values for .tax and .units are located at cell addresses at C1 and C2, respectively.

In addition, user input can be constructed at an array of consecutive row cells. It would look like:

```
RQUERY(RuleSetName,”FIND .price AND .total WHEN .tax = _1 AND .units = _2”, C1, C2).
```

This type of array query is constructed by use of prescribed key strokes.

A rule set called, RULE2, can point to a rule set, RULE1, by inserting the statement “INHERIT FROM RULE1” in RULE2. In other words, a RULE2 query will search RULE1 when looking for a property and it is not found in RULE2.

Queries can use rule variables by the variables of the form: ?x. A query might look like:

```
RQUERY(RULE2, “FIND .price WHEN .loc = _1”, C2).
```

Assume the cell C2 holds “hammer”. In the associated rule set RULE2, there would be a rule,
Our last feature shows variable rule sets. Consider the query:

\[
\text{RQUERY(MasterRules,"FIND .cost WHEN .plantname = _1", D2)}
\]

The rule sets are

- **MasterRules**
  - LoadRules(Plant1, Plant2)
  - .cost = ?x!.cost  WHEN ?x = . plantname
  - *

- **Plant1**
  - Inherit from MasterRules
  - .cost = 24
  - *

- **Plant2**
  - Inherit from MasterRules
  - .cost = 35
  - *

The ?x!.cost construct with the “!” means that .cost can be evaluated at a rule set specified by the variable as defined by ?x.

**Marketing Example**

We now return to Mary’s marketing problem. She decides to use the ARULESXL software. Four rule sets are formed.
DallasData
Inherit from Calculate
City[Dallas].unitcost = 10
City[Dallas].surcharge = 150
City[Dallas].delivery = 500
City[Dallas].note = "Dallas"

BostonData
Inherit from Calculate
City[Boston].unitcost=25
City[Boston].surcharge=125
City[Boston].delivery=375
City[Boston].note ="Boston"

MiamiData
Inherit from Calculate
City[Miami].unitcost=15
City[Miami].surcharge=135
City[Miami].delivery=450
City[Miami].note = "Miami"

Calculate
Loaded Rules: DallasData, BostonData, MiamiData
inherit from DallasData
inherit from BostonData
inherit from MiamiData
.Total = City[?x].unitcost * .units + City[?x].surcharge + City[?x].delivery when ?x = .Loc
.Note[?x] = City[?x].note when ?x = .Loc

Multiple rule sets have three distinct cities with different values assigned to the parameters of unitcost, surcharge, and delivery. The instruction, appears as Loaded Rules: DallasData, BostonData, MiamiData.

This forces the reasoning to apply to these other Rule sets when searching for data. The “inherit from DallasData” is used when a value is not found in the current rule set.

The one query is:

RQUERY(Calculate,"FIND .Total when .Loc =_1 AND .units =_2", B3,B4)

where cells B3 and B4 has the location and number of units.
Therefore, Mary has incorporated the all requirements of the decision making problem assigned to her. Moreover, now that has seen a prototype and can see the working solution, the addition of more locations as well as more properties (values) can be managed. The actual spreadsheet with the ARulesXL functionality incorporated into worksheet toolbar is below.

Conclusion

We have demonstrated a limited example in a ARulesXL logic spreadsheet. Spreadsheets are constructed by input and out cells, rule sets, and queries. Queries can search for property values through single or multiple rule sets. Rule sets are statements which relate the query to property values.

The advantage of this type of presentation is that the rules are stated on the spreadsheet and can be altered directly by the user rather than being imbedded in VBA code or Excel IF functions. In addition, the limitation of the number of Excel nested IF statements is avoided.

A limitation of this application is that logic usually conjures up concepts such as truth tables, symbolic logic, and proofs. This paper has attempted to introduce the reader to an approach which should give the user more insight into the decision making process as well as the rules and reasoning behind a decision.
REFERENCES

