Interactive Wang Notation Tool For Web Tables

Piyushee Jha
Rensselaer Polytechnic Institute

May 2007

Table of Contents

1. Introduction

The Semantic Web combines various technologies to supplement or replace the content of web documents with descriptive data that will assist the user in decision making and will address their specific needs and wants. This can only be accomplished with an abundance of ontologically annotated data. However, creating ontologies is a difficult process. We begin by attempting to create ontologies from data in tables. The first step in this process is to convert all the information in any given table into a standard form for easy comparison and manipulation. This report describes the creation of a tool in Matlab that does just that by converting HTML tables into Wang notation [1].

Successive versions of this tool were created, but early versions did not meet our criteria. There are two requirements for a successful tool – it must be robust, able to handle a variety of tables, both in shape and size, and it must be fast. The primary advantage of having a tool to generate Wang notation rather than manually writing it, is speed. Therefore, in the end we want a tool that is mostly automatic and able to handle numerous types of tables. The rest of the Introduction describes the Wang notation and what remains to be done. The next four sections describe the evolution of the tool.

a. Wang Notation

Now, we present a brief explanation of Wang notation. Wang Notation consists of two parts - category notation (using C) and delta notation (using δ). An abstract table is specified by an ordered pair (C,δ) where C is a finite set of labeled domains (header, sub headers of tables, etc) and δ represents each individual value within a table corresponding to C. Table 1 shows the Wang table from Wang's PhD thesis that was used as the test table during the creation of the Wang Notation Tool (WNT).

Table 1: Wang Table

Year	Term	Mark					
		Assignments			Examinations		Grade
		Ass1	Ass2	Ass3	Midterm	Final	
1991	Winter	85	80	75	60	75	75
	Spring	80	65	75	60	70	70
	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
	Spring	80	80	70	70	75	75
	Fall	75	70	65	60	80	70

The Wang table has three categories (making it a 3-d table). The categories are the broadest possible headings (headers) and everything within them is their subcategory (sub headers). All cells containing category information are called category cells. Following is the category notation for the above table.

(Year, {(1991,ϕ), (1992,ϕ)})
Year is the first category with 1991 and 1992 as the only choices.
(Term, {(Winter,ϕ), (Spring,ϕ), (Fall,ϕ)})
Term is the next category with winter, spring, and fall as the three choices.
(Mark, {(Assignments, {(Ass1,ϕ), (Ass2,ϕ), (Ass3,ϕ)}), (Examinations, {(Midterm,ϕ), (Final,ϕ)}), (Grade,ϕ)})

Mark is the most complicated category with three subcategories (Assignments, Examinations, and Grade) among which Assignments and Examinations have their own subcategories.

The delta notation shows which category cells are related to each of the individual values within the table. The cells containing information for the delta notation are referred to as either content cells or delta cells. Following is the delta notation for the first two rows of the Wang table.

δ({Year.1991, Term.Winter, Mark.Assignments.Ass1}) = 85
δ({Year.1991, Term.Winter, Mark.Assignments.Ass2}) = 80
δ({Year.1991, Term.Winter, Mark.Assignments.Ass3}) = 75
δ({Year.1991, Term.Winter, Mark.Examinations.Midterm}) = 60
δ({Year.1991, Term.Winter, Mark.Examinations.Final}) = 75
δ({Year.1991, Term.Winter, Mark.Grade}) = 75
δ({Year.1991, Term.Spring, Mark.Assignments.Ass1}) = 80
δ({Year.1991, Term. Spring, Mark.Assignments.Ass2}) = 65
δ({Year.1991, Term. Spring, Mark.Assignments.Ass3}) = 75
δ({Year.1991, Term. Spring, Mark.Examinations.Midterm}) = 60
δ({Year.1991, Term. Spring, Mark.Examinations.Final}) = 70
δ({Year.1991, Term. Spring, Mark.Grade}) = 70

b. Outstanding Issues

The tool described in this report can only deal with well-formed HTML tables. Since the tool is written in Matlab, we use Excel to transfer data between HTML web pages and Matlab. HTML is not a very rigid language and different people have different ways of coding the same thing. Excel can accommodate the different coding styles of different people and still correctly interpret them as the same table. However, using Excel is a step, which if eliminated, would simplify and speed up the process of getting a table into Matlab. We don't yet have a way of transferring tables directly from HTML code to Matlab.

User training is also an issue that needs to be addressed. The tool is not fully automatic and is designed to be interactive. The user has to be able to understand enough about a table to be able to differentiate between category and content cells, to be able to point out the category cells related to a given leaf cell, and to be able to point out which category cells correspond to a given content cell. In the final version of the tool, the user has an opportunity to correct the indented notation of the categories within the table. This means that the user also has to understand the indented notation well enough to decide if the indented notation is correct for the given category. We will have to develop some kind of test to see if our users really understand the aforementioned aspects of a table.

Another issue to be addressed is foreign tables. Is it possible for a user to differentiate between category and content cells in a foreign table? There are some cases where it might be, but in general it would not be. To demonstrate this further, I have shown both the actual tables and a version of the tables where the text is represented by symbols below. In Table 2, a user cannot differentiate between category and content cells in the first or second column. In the original table (a), it is easy to see that the first column is a category followed by its subcategories and the second column is a category followed by its values, but in the "foreign" table (b),

the user cannot tell if the columns are categories and their subcategories or categories followed by their values.

Similarly in Table 3, there is confusion in the first two columns between the original (a) and "foreign" (b) table. However, in the third category of both tables, 'Population' and 'Mark', the user can differentiate between the category and content cells because of the merged cells. A merged cell always implies that it is a category cell. Therefore the lowest category cell will be the first row of least merged cells. That would be the second row in table 1 and third row in table 2. The last category row and first content row have the same cell divisions and it's logical to assume that if there is no further division the second row of greatest division must contain content cells. From this we come to the conclusion that categories that appear at the top of the table can be differentiated but categories that appear on the sides of the table cannot.

Table 2: (a) – original table, (b) – "Foreign" table

country	area sq.km.	population	
		yearly growth	today
World	510,072,000	1.14%	6,563,077,034
China	9,596,960	0.59%	1,317,924,274
India	3,287,590	1.38%	1,103,054,870
United States of America	9,631,418	0.91%	299,828,179
Indonesia	1,919,440	1.41%	247,216,367
Brazil	8,511,965	1.04%	189,074,990
Pakistan	803,940	2.09%	167,569,436
Bangladesh	144,000	2.09%	148,934,854
Russia	17,075,200	-0.37%	142,624,117
Nigeria	923,768	2.38%	133,458,955
Japan	377,835	0.02%	127,476,602

Table 3: Left – (a) – original table, (b) – "Foreign" table

Year	Term	Mark				
		Assignments		Examinations		Grade
		Ass1	Ass2	Midterm	Final	
1991	Winter	85	80	60	75	75
	Spring	80	65	60	70	70
	Fall	80	85	55	80	75
1992	Winter	85	80	70	75	75
	Spring	80	80	70	75	75
	Fall	75	70	60	80	70

While version 3 of this tool is fairly robust and semi-automatic, it is still possible to further automate it. The goal is to have a tool that can generate Wang notation of tables automatically, with the user having an opportunity to correct mistakes made. The tool will also ask the user for help if it encounters something for which it cannot generate notation. This would require refining the tool so it will know when

something is incorrect. Also, the tool should be able to "learn" and not have the user do anything repetitive; once a mistake has been made, it should be avoided and once the tool has processed a certain type of table, the same type of table should be processed automatically.

2. WNT Version 1 – Basic Wang Notation Converter with Limitations

a. HTML to Matlab

The first step in making a tool to convert tables into Wang notation is to obtain the tables. Our tables will come from the domain of geo-political data. These tables, in general, are very large but have simple categories. For now we are working with HTML tables only. At a later date, scanned image tables will also be used.

An Internet browser can be used to copy the table and paste it into Excel. While this sounds quite straightforward, it requires some care. When copying the table from the Internet browser, one must select the entire table, not just the text within the table. Also instead of pasting the table into Excel, with the normal copy-paste, the user must use the 'HTML' option within the 'Paste Special' command. Some basic pre-processing has to be done in Excel to make it ready for MATLAB.

First, all the formatting has to be removed from the table. Next, and most important, all the cells of the table have to be designated 'text' type cells. The function *xlsread* in Matlab is used to transfer the table from Excel into Matlab. The xlsread function reads the text cells and the numeric cells of an Excel spreadsheet into different arrays, which is why we designate all cells to be text cells before using Matlab.

Due to the difficulties of transferring a table from a web page into Matlab through Excel in WNT v 3.0, we will be making use of open-source code (provided by Cui Tao of Dr. David Embley's lab) that parses an HTML table and outputs the table in ASCII.

b. Categories as Trees

One of the first observations I made was that each category could be represented as a tree. While I did not utilize this observation in any way in the first two versions of my tool, it was used extensively in version 3. This connection is mentioned here to make some things easier to explain.

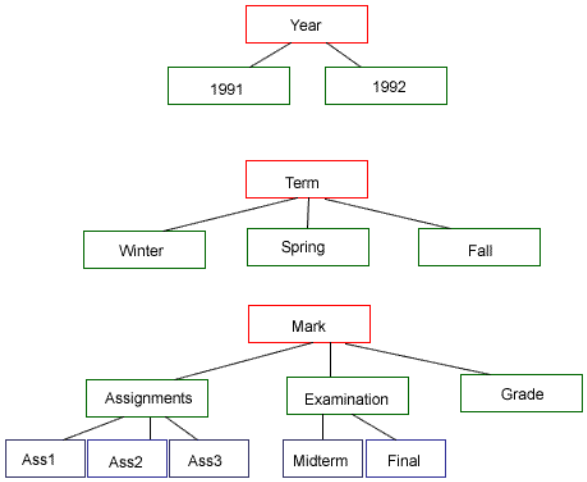


Figure 1: Categories as Trees

c. Prompting for Information

Version 1 of the Wang Notation Tool (WNT V.1) asks the user questions and has them type in responses. The main reason for making a MATLAB program was to make it able to incorporate a wide variety of tables. The category notation relies on knowing the lowest subcategories of the first branch before moving on to the next branch within that category. In relation to trees, this means that the category notation has a depth first traversal rather than a breadth first traversal. However, I did not think of tree traversal as an answer to my problems nor did I know how to program it then.

Therefore, I decided to limit my tables. The largest category that could be entered could contain 5 levels. A level is a row of blocks in Figure 1. The most complicated category in the Wang table has 3 levels. By limiting my tables, I was able to write a program containing numerous nested loops to generate the category notation. The program prompted the user for the number of categories first. Then the user was prompted for the name of the first category, then the number of subcategories within the first category and so on. All responses had to be typed into the Matlab command window.

The delta notation stemmed from the category notation responses. Again, looking at categories from a tree structure perspective, all the combinations between the branches of different categories makes up the delta notation. For example a branch from category three would be *Mark* > *Assignments* > *Ass1*. These combinations were generated by my program and were then displayed in the command window one by one and the user was asked to type in the value of each delta. See Appendix Part A for the command window output of WNT V.1.

WNT V.1, while functional, had many disadvantages. Firstly, it was not very general because the table was limited to 5 levels. Secondly, it takes a lot of time. It would be faster for a user who knows Wang notation to just type in the Wang notation. I had no ideas of how to remove the limitations of the tool at this time, but I could see that if the user did not have to type all that information and if the delta portion of the code could be fully automated, the tool would be much faster. This led to WNT V.2.

3. WNT Version 2 – Eliminating the Need to Type

a. Automatic Delta Using Symmetric Table

I started with the problem of automating the delta notation generation. Since the delta cells vastly outnumber the category cells, automating them would drastically reduce the time taken to generate Wang notation using this tool. I explored many avenues of automatic delta generation, but none worked and were simultaneously robust. Many of the original ideas I had about searching out the category cells corresponding to each delta cell could not be applied to all tables. Finally, I realized that if every table was made symmetric, generating the delta notation automatically would be simple.

A symmetric table is a table such that all the category cells pertaining to a delta cell are in either the same row or same column as that delta cell. The top and leftmost delta cell in Table 3 contains the number '85'. In the category cells in the same row and column as this delta cell, this delta cell, or this numeric value of '85', is associated with 1991, Winter, Mark, Assignments, and Ass1. It is not evident that this value '85' is also associated with Year and Term. To my prompting program, I added a few more questions about the locations of categories that were not symmetric to generate a symmetric table. Table 4 is a symmetric version of the Wang table.

Table 4: Symmetric Wang Table

Year	Year	Term	Term	Mark					
				Assignments			Examinations		Grade
				Ass1	Ass2	Ass3	Midterm	Final	
	1991		Winter	85	80	75	60	75	75
			Spring	80	65	75	60	70	70
			Fall	80	85	75	55	80	75
	1992		Winter	85	80	70	70	75	75
			Spring	80	80	70	70	75	75
			Fall	75	70	65	60	80	70

Once I had this symmetric table, I wrote a program to go through every delta cell, determine the category cells it is related to by simply looking at every category cell in the same row or column as itself and then generate the delta notation completely automatically. This, as predicted, greatly reduced the amount of time taken to generate Wang notation.

b. Graphical User Interface

The next step to reducing the time needed to generate the Wang notation was to make the table into a GUI so that the responses to the prompts of the program could be clicked rather than typed. This not only speeds up the process, but also reduces the chance of error (mostly spelling errors) because the user is no longer typing everything.

Using the *uicontrol* function in Matlab, I converted the table obtained via Excel into a GUI. The code for this is shown below. Two other functions that were used in the callback function and were very handy were *uiwait* and *uiresume*. Before these functions were used, the user had to click their response in the GUI and then go to the command window to hit ENTER to have their response

acknowledged by the program. The *uiwait* and *uiresume* functions made the WNT faster by letting the user click only a button for the entry to be acknowledged by the program.

```
for i = 1:s(1,2),
   for j = 1:s(1,1),
      c = c + 1;
      h(c) = uicontrol('Style', 'pushbutton', 'String', char(table(i,s(1,1)
          j+1)),'Position',[i*80 j*20 80 20],'callback','CW2_callback;');
   end
end
```

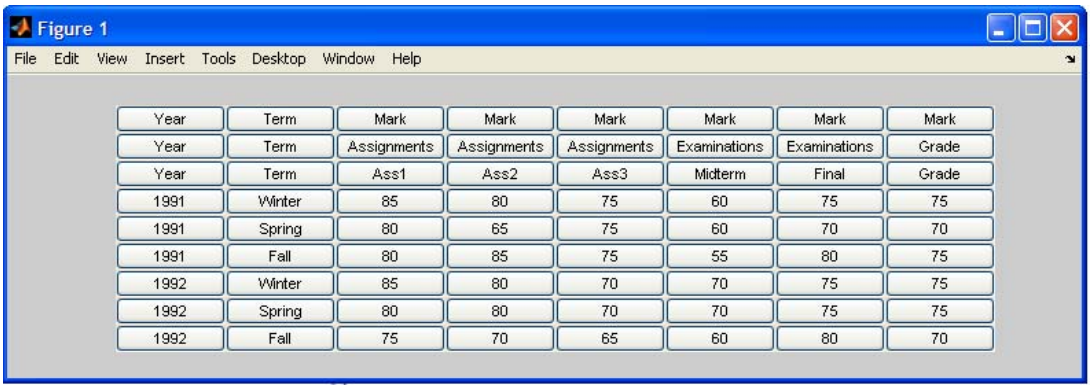


Figure 2: Wang Table As GUI

Once I had this GUI (Figure 2) the user could click responses every time the program prompted for the name of something. However, the program also prompted for various numbers relating to the categories. For these prompts, I created another GUI with buttons 0 to 10. These GUIs eliminated the need for any typing.

To summarize version 2: It can handle tables with up to 10 categories and up to 5 sub-levels. Once the program is run, two figures pop-up: one contains the table and the other contains the potential answers to questions asked by the program. Prompts are displayed in the command window and the user has to click on the appropriate button in the appropriate figure. There is no need for the user to type any answers, everything can be answered by clicking. After the prompts are over and the user has completed entering all the data, the category notation is generated by the program. Then, the user is asked a couple more questions that can be answered by clicking some buttons, the table is made symmetric, and the delta notation is generated automatically.

WNT V.2 reduced the time taken to enter the contents of the table, but it did nothing for robustness. V.1 and V.2 were programmed so that the largest number of levels the tool could accommodate was 5 and the largest number of categories and subcategories it could handle was 10.

4. WNT Version 3 – More Automatic and Robust

WNT V.3 removes many of limitations that were present in V.1 and V.2. To remove these limitations, I went back to my idea of representing categories as trees. Also the structure of tables makes it possible to reduce the amount of clicking, thus reducing time. In addition, a symbolic representation for discussing the various relations and defining nodes is created.

a. Symbolic Representation

To better discuss the nodes of a tree, the following convention is used. The table itself will be referred to as 0. In an n-dimensional table, the categories are called 1, 2, … , n in alphabetical order. The subcategories of each category are labeled in standard left to right or top to bottom order. The figure below shows the tree of the Wang Table, with the node numbers listed in red.

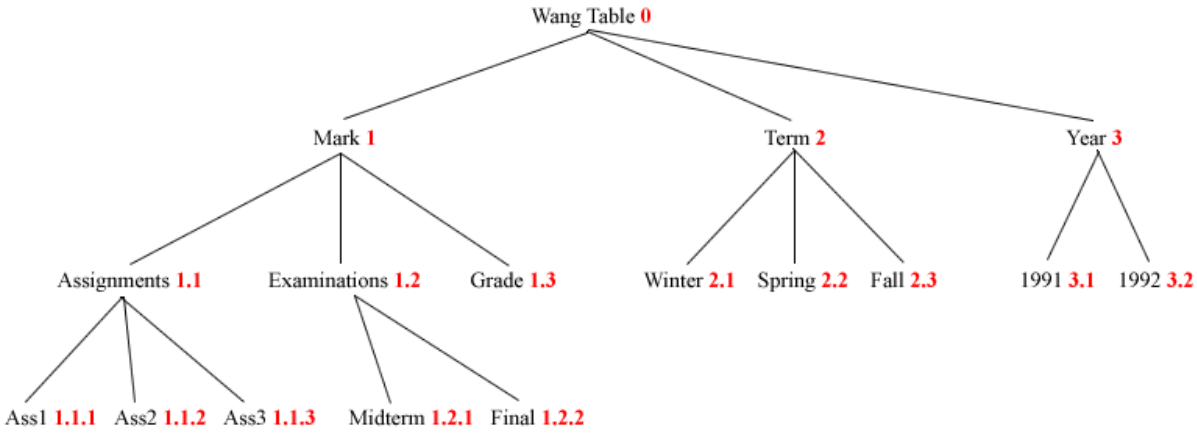


Figure 3: Wang Table in Tree Form

b. Category Notation and Trees

Before I thought about tying trees in with the category notations, I tried a different method of generating robust category notation. The user still clicks on all the cells containing category notation. Each category is made into a separate array with all extraneous rows and columns deleted. I tried to add parentheses and commas in the correct places to comply with the category notation. I had the program read the cells contents and insert them in the correct places. This method, however, only worked for the simplest categories – again, since this was a depth first traversal rather than breadth first traversal it had me stalled. Finally, I went back to using tree representation for categories. I realized that traversing a table in pre-order is the same as the Wang notation order. From this observation, I could create a robust generation method.

i. Indented Notation

The first thing I had to do was actually represent categories as trees. This process started out the same as the previous ones with the user clicking the cells containing category cells. From here, each category was separated and since the position of each cell compared to its root was known, I wrote a program in Matlab to create it. The following is the *indented notation* for the third category in the Wang table. It is represented as an array in Matlab. Appendix Part B contains the code.

```
'Mark'      ' '             ' '
' '         'Assignments'   ' '
' '         ' '             'Ass1'
' '         ' '             'Ass2'
' '         ' '             'Ass3'
' '         'Examinations'  ' '
' '         ' '             'Midterm'
' '         ' '             'Final'
' '         'Grade'         ' '
```

ii. Table of Contents with pointers

To perform actions on a tree structure in Matlab, I created a *table of contents* representation of each category from the indented table. The following is the table of contents representation for the Mark category in the Wang table. Appendix Part B contains the code.

```
Mark         1  0  0
Assignments  1  1  0
Ass1         1  1  1
Ass2         1  1  2
Ass3         1  1  3
Examinations 1  2  0
Midterm      1  2  1
Final        1  2  2
Grade        1  3  0
```

iii. Pre-order Traversal

For the pre-order traversal of trees, we need to be able to manipulate trees. Since it is difficult to manipulate general trees, we convert them into binary trees. Father/son relations are more efficient in binary trees than in general trees. The leftson in a binary tree is the firstson of the father in the general tree. The rightson in a binary tree is the rightsibling of the preceding son in the general tree. The order of the nodes does not matter, only the pointers. We represent binary trees in a structure array with fields *nodename* and *pointers*.

Dr. George Nagy developed some functions in Matlab that perform various operations on trees including a function to build a table structure from the table of contents representation. Appendix Part B contains the functions for tree manipulation.

Now that we had a tree structure that could be easily manipulated, I wrote a function for the pre-order traversal of trees. This function is also included in Appendix Part b. As predicted, the pre-order traversal matches the order of categories in Wang notation. We end up with a robust, unlimited, and fast method of generating category notation. Also recall that the delta notation has been automatic since version 2 of the Wang Notation Tool.

However, traversing a tree and putting all the categories in the correct order is not all that is required for generating the category notation. In addition to tree-traversal, the category notation contains other symbols such as parenthesis, curly parenthesis, and commas that delineate the relation between the cells of a category. To appropriately place additional symbols, I came up with a set of rules. Figure 3 is a nonsense example of the general tree. Figure 4 is the equivalent binary tree with the node numbers, pointers, and symbols shown. Figures 4 and 5 and the pseudo code that follows demonstrate the insertion of symbols into the category notation

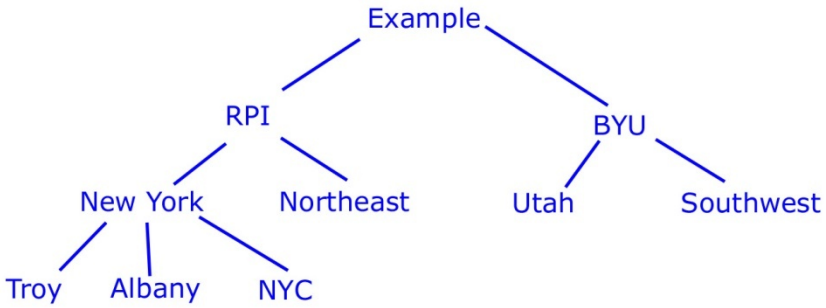


Figure 4: General Nonsense Table

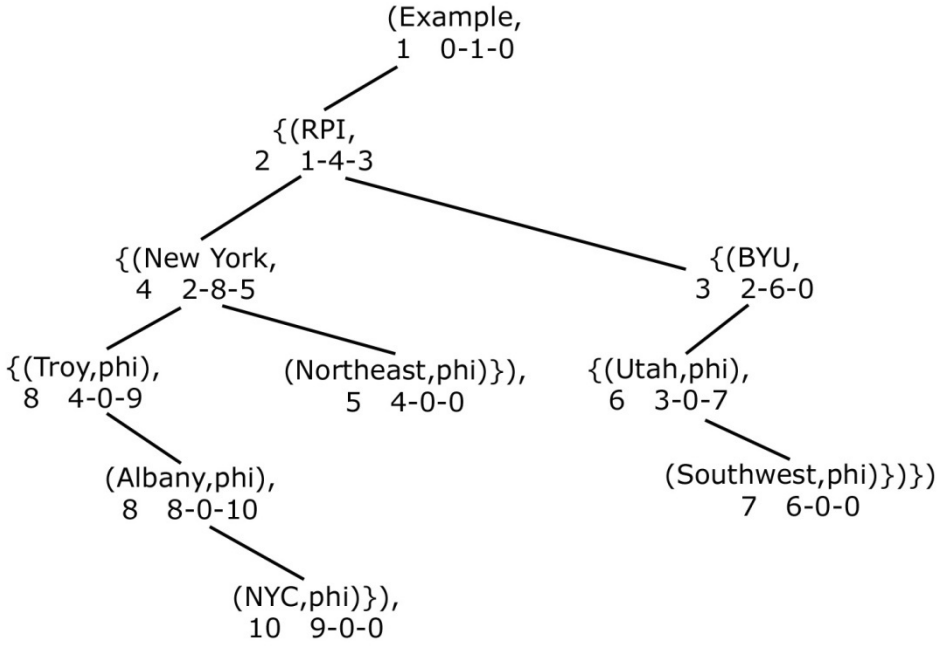


Figure 5: Equivalent Binary Table

```
i = index of current cell
if leftson of i ~= to 0 AND rightson of i ~= 0
        if leftson of the father of i == i
                {( cell content ,
        elseif leftson of the father of i ~= i
                ( cell content ,
elseif leftson of i ~= to 0 AND rightson of i == 0
        ( cell content ,
elseif leftson of i == to 0 AND rightson of i ~= 0
        if leftson of the father of i == i
                {( cell content ,phi),
        elseif leftson of the father of i ~= i
                ( cell content ,phi),
elseif leftson of i == to 0 AND rightson of i == 0
        ( cell content ,phi)}),
```

c. Further Reducing Clicks

The next thing to be done would be to reduce the number of clicks even further. If we make a table symmetric before we do any manipulation on it, all the cells pertaining to any one category fall within a specific rectangle within the table. (See Table 5) I explored this further and came to the conclusion that this was true for almost all tables. Using this information, I changed my program such that the user now only had to click on the top leftmost cell and the bottom rightmost cell.

Table 5: Wang Table with Category Rectangles Marked

Year	Year	Term	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Year	Term	Term	Assignments	Assignments	Assignments	Exams	Exams	Grade
Year	Year	Term	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
Year	1991	Term	Winter						
Year	1991	Term	Spring						
Year	1991	Term	Fall						
Year	1992	Term	Winter						
Year	1992	Term	Spring						
Year	1992	Term	Fall						

Once the user clicks on the two cells that define a category, my program sweeps the cells and determines which cells are relevant (it will delete all the repeated cells) and then creates the indented table, table of contents, and pre-order traversal as before. Version 3 is therefore much faster and much more robust that the previous versions.

d. Error Correction

An important part of making anything automatic is having ways of correcting errors. If we are able to correct errors then it is possible for us to make a more automatic program and then just have the user correct any errors later. Version 3 incorporated two methods of error correction that made it more robust and a better tool overall than the previous two versions.

a. Correct While Entering

The first way of catching and fixing errors arises from further manipulation within the *uicontrol* function. I changed the program so that each cell clicked would change color and the user would know what they had clicked. Also, I re-wrote the callback function such that if a cell that had been clicked was clicked again, it would act in standard Windows manner and un-select the cell.

Year	Year	Term	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Year	Term	Term	Assignments	Assignments	Assignments	Examinations	Examinations	Grade
Year	Year	Term	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
Year	1991	Term	Winter	85	80	75	60	75	75
Year	1991	Term	Spring	80	65	75	60	70	70
Year	1991	Term	Fall	80	85	75	55	80	75
Year	1992	Term	Winter	85	80	70	70	75	75
Year	1992	Term	Spring	80	80	70	70	75	75
Year	1992	Term	Fall	75	70	65	60	80	70

Figure 6: Wang Table GUI with Colors

The red cells are the ones that were clicked by the user and the blue cells were turned blue by the program. This gives the user an opportunity to see which cells the program considers category cells. If the program's interpretation is incorrect, then the user can click on a red cell again to un-select it. Note that in the GUI above all the cells are filled in, i.e. Year is repeated down the entire column, but they don't have to be. If they weren't filled in, all the extra filled cells would just be empty and the program would function the same way.

b. Post-Editing Tool

In addition to the error correction built in during category entry, I created a tool to correct errors after everything had been entered. This tool appears on-screen after the indented table is made.

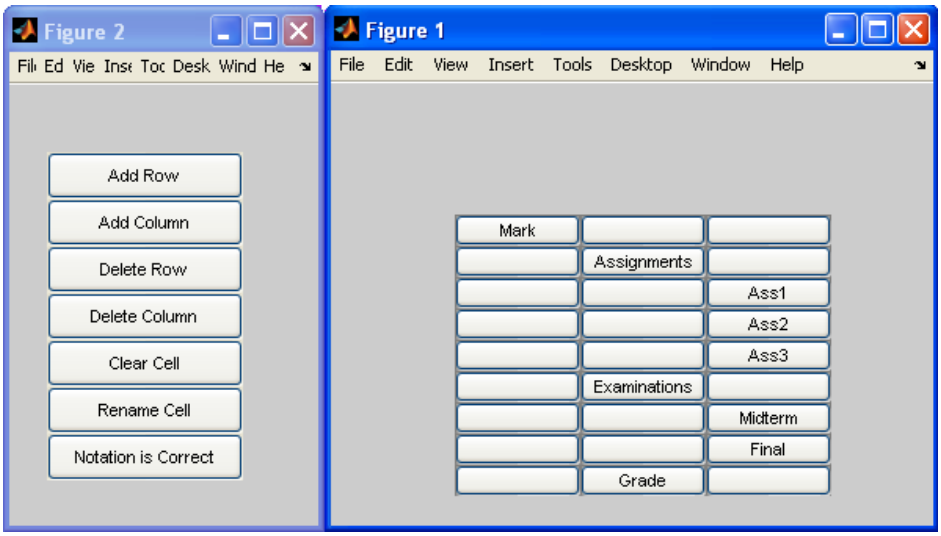


Figure 7: Error Correction Tool

The user can click on the cell they want to change and then click on what change they want. Using the indented notation and the error correction tool, all errors can be fixed. This error correction tool is contained within the GUI's; there is never a need for the user to look at the command window in Matlab. 'Rename Cell' is the only option that needs a written input by the user. When the 'Rename Cell' button is clicked, the cell to be renamed in the indented table turns into a textbox instead of a pushbutton where the user can type the new name. Once the user hits enter or clicks anywhere outside that textbox, the cell turns into a pushbutton again. After all errors are fixed, the user clicks on 'Notation is Correct' and the program will then continue on with generating category and delta notation.

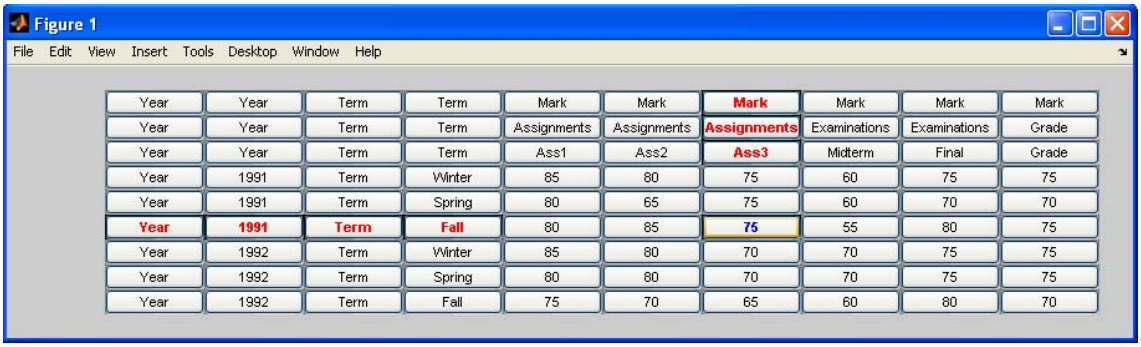
e. Experiment with Real Tables

Till now the Wang table was used for all testing purposes but since this is only one table, I had to test my program with various tables to determine its robustness and efficiency. I found many tables of geopolitical data on the following website: http://www.geohive.com.

One of the problems with these tables is that they are too large to display on the screen at the same time so it is annoying to have to click on something at the very top and bottom. Another problem I had was in transferring data from Excel to Matlab. Initially I had mentioned that before reading data from Excel, someone had to select all the cells and format them so that all the cells were text type cells. I found that this was not working and even after formatting cells as text type cells in Excel, Matlab was still reading them as numeric cells. The key to this problem is that all the cells in the worksheet have to be formatted as text cells before anything is pasted from the HTML file. Transferring files to Matlab via Excel is tedious because it takes much effort sometimes to get Excel to recognize that all cells are text cells. Therefore I need to think of a better way of getting HTML into Matlab. Other than the aforementioned problems, my tool has generated category and delta notation satisfactorily for the tables from GeoHive.

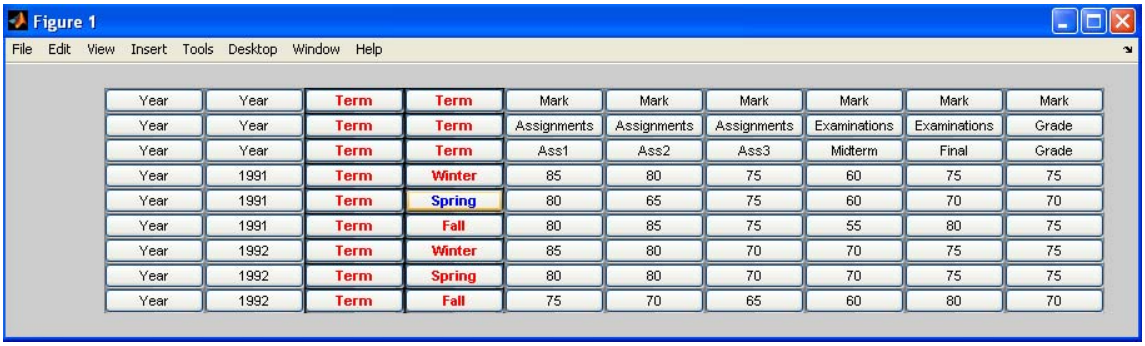
f. Connect Category and Delta Notation

Lastly, we wanted the user to be able to concretely connect the category and delta notation. To do this, at the end of all the processing, I display a GUI containing the table that was just processed. If the user clicks on a delta cell, that cell turns blue and its corresponding category cells turn red. See Figure 7. Similarly, if the user clicks on one category cell, that cell turns blue and all the other cells in that category turn red.



Year	Year	Term	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Year	Term	Term	Assignments	Assignments	Assignments	Examinations	Examinations	Grade
Year	Year	Term	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
Year	1991	Term	Winter	85	80	75	60	75	75
Year	1991	Term	Spring	80	65	75	60	70	70
Year	1991	Term	Fall	80	85	75	55	80	75
Year	1992	Term	Winter	85	80	70	70	75	75
Year	1992	Term	Spring	80	80	70	70	75	75
Year	1992	Term	Fall	75	70	65	60	80	70

Figure 8: Connecting A Delta Cell With Its Category Cells



Year	Year	Term	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Year	Term	Term	Assignments	Assignments	Assignments	Examinations	Examinations	Grade
Year	Year	Term	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
Year	1991	Term	Winter	85	80	75	60	75	75
Year	1991	Term	Spring	80	65	75	60	70	70
Year	1991	Term	Fall	80	85	75	55	80	75
Year	1992	Term	Winter	85	80	70	70	75	75
Year	1992	Term	Spring	80	80	70	70	75	75
Year	1992	Term	Fall	75	70	65	60	80	70

Figure 9: Connecting Category Cells

5. WNT Version 3.5 – Improvements

After V.3 was completed, we thought of new ways to further improve WNT. Firstly, the generation of symmetric tables was causing the user a host of problems. It is hard to explain the concept of symmetric tables to a user unfamiliar with tables and the earlier versions of WNT prompted the user for information to correctly generate the symmetric table. If a table is symmetric to begin with, WNT v 3.5 works just as well. After this we worked on improving the process of getting the original table from HTML into Matlab.

a. Removing Symmetric Tables

To start with I automated the process of generating symmetric tables automatic. The user picks rectangle corners in the non-symmetric table as before and then depending on the shape of the rectangle, a function makes the category symmetric. For example, a category that is either a 1 x n or n x 1 is assumed to be symmetric. The plan was to add to this symmetric table function as more categories were tested out. However, as I tested more tables I realized that I could not get the same piece of code to handle a variety of tables correctly. Instead of making the process automatic, it seemed like more user interaction would be required. Therefore, I decided to take out symmetric tables completely.

Originally, we thought to make table symmetric (a definition for which is: A table is symmetric when the category cells relevant to a delta cell are in the same row and column as that delta cell.) to make it more visually understandable to the user. Afterwards, the symmetric table was used to

generate delta notation. Thus, getting rid of symmetric tables meant that I had to come up with a new way to generate delta notation. Fortunately, this was fairly simple. I combined all the indented categories to obtain the indented notation for the entire table and from this, I determined the delta notation.

A note: Virtual headers used to be a problem, but after we got rid of symmetric tables, the user can just add a virtual header using the post-error correction GUI and the category and delta notation will be correct. A virtual header is needed if all the categories in the table do not show up in every line of the delta notation.

b. Tables in ASCII format

The original table is in HTML from where it is copied and pasted into Excel and then read into Matlab. The step with Excel, in addition to taking extra time and effort on the part of user, was never very good. This was because of the way Excel handled its formats; it was difficult to get Excel to recognize a certain table is all text rather than numeric, especially when the table was large. For these reasons, we at RPI, contacted BYU to ask if they had a way to get HTML tables into a form easily manipulated by Matlab. In return, we got a JAVA project that parses HTML files, detects tables within the file, and outputs the tables in ASCII format.

The ASCII version of the tables is sent to a text file that is then read in by Matlab. A Matlab function goes through the text file and takes the ASCII information and translates it to an array. However, we ran into a few problems with the original ASCII output. The original Java project does not take into account spanned cells, therefore this, along with a few other minor changes, were requested from BYU. The changes are detailed below. In the meantime, we used a manually synthesized version of the ASCII output to run our files.

The current output from the Java project for the Wang table is as follows:

```
total table number is 1
################error: undefined table type!

Year ** Term ** Mark **
Assignments ** Examinations ** Grade **
Ass1 ** Ass2 ** Ass3 ** Midterm ** Final **
1991 ** Winter ** 85 ** 80 ** 75 ** 60 ** 75 ** 75 **
Spring ** 80 ** 65 ** 75 ** 60 ** 70 ** 70 **
Fall ** 80 ** 85 ** 75 ** 55 ** 80 ** 75 **
1992 ** Winter ** 85 ** 80 ** 70 ** 70 ** 75 ** 75 **
Spring ** 80 ** 80 ** 70 ** 70 ** 75 ** 75 **
Fall ** 75 ** 70 ** 65 ** 60 ** 80 ** 70 **
**********************************************
**********************************************
**********************************************
```

The desired output is as follows:

```
**********************************************

Year [rowspan = 3] ** Term [rowspan = 3] ** Mark [colspan = 6] *****
Assignments [colspan = 3] ** Examinations [colspan = 2] ** Grade [rowspan
= 2] *****
Ass1 ** Ass2 ** Ass3 ** Midterm ** Final *****
1991 [rowspan = 3] ** Winter ** 85 ** 80 ** 75 ** 60 ** 75 ** 75 *****
Spring ** 80 ** 65 ** 75 ** 60 ** 70 ** 70 *****
Fall ** 80 ** 85 ** 75 ** 55 ** 80 ** 75 *****
1992 [rowspan = 3] ** Winter ** 85 ** 80 ** 70 ** 70 ** 75 ** 75 *****
Spring ** 80 ** 80 ** 70 ** 70 ** 75 ** 75 *****
Fall ** 75 ** 70 ** 65 ** 60 ** 80 ** 70 *****
**********************************************
**********************************************
**********************************************
```

The most notable change is the addition of the rowspan and colspan words in the desired output. In addition to that, the error message is suppressed; a line of asterisks is added at the beginning of each table and instead of two asterisks, there are 5 asterisks at the end of each row. With these changes, we can correctly and quickly convert an ASCII file to a Matlab array.

c. Examples of WNT v 3.5

Below are two examples of tables run through WNT v3.5. The first example is of the Wang table.

1. Domain - HTML version:

Year	Term	Mark					
		Assignments			Examinations		Grade
		Ass1	Ass2	Ass3	Midterm	Final	
1991	Winter	85	80	75	60	75	75
	Spring	80	65	75	60	70	70
	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
	Spring	80	80	70	70	75	75
	Fall	75	70	65	60	80	70

2. Original ASCII:

```
total table number is 1
################error: undefined table type!

Year ** Term ** Mark **
Assignments ** Examinations ** Grade **
Ass1 ** Ass2 ** Ass3 ** Midterm ** Final **
1991 ** Winter ** 85 ** 80 ** 75 ** 60 ** 75 ** 75 **
Spring ** 80 ** 65 ** 75 ** 60 ** 70 ** 70 **
Fall ** 80 ** 85 ** 75 ** 55 ** 80 ** 75 **
1992 ** Winter ** 85 ** 80 ** 70 ** 70 ** 75 ** 75 **
Spring ** 80 ** 80 ** 70 ** 70 ** 75 ** 75 **
Fall ** 75 ** 70 ** 65 ** 60 ** 80 ** 70 **
**********************************************
**********************************************
**********************************************
```

3. Manually Synthesized ASCII:

```
**********************************************

Year [rowspan = 3] ** Term [rowspan = 3] ** Mark [colspan = 6] *****
Assignments [colspan = 3] ** Examinations [colspan = 2] ** Grade [rowspan
= 2] *****
Ass1 ** Ass2 ** Ass3 ** Midterm ** Final *****
1991 [rowspan = 3] ** Winter ** 85 ** 80 ** 75 ** 60 ** 75 ** 75 *****
Spring ** 80 ** 65 ** 75 ** 60 ** 70 ** 70 *****
Fall ** 80 ** 85 ** 75 ** 55 ** 80 ** 75 *****
1992 [rowspan = 3] ** Winter ** 85 ** 80 ** 70 ** 70 ** 75 ** 75 *****
Spring ** 80 ** 80 ** 70 ** 70 ** 75 ** 75 *****
Fall ** 75 ** 70 ** 65 ** 60 ** 80 ** 70 *****
**********************************************
**********************************************
**********************************************
```

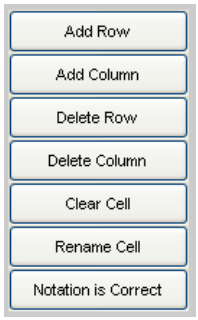
4. Table:

Year	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Term	Assignments	Assignments	Assignments	Examinations	Examinations	Grade
Year	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
1991	Winter	85	80	75	60	75	75
1991	Spring	80	65	75	60	70	70
1991	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
1992	Spring	80	80	70	70	75	75
1992	Fall	75	70	65	60	80	70

5. Categories:

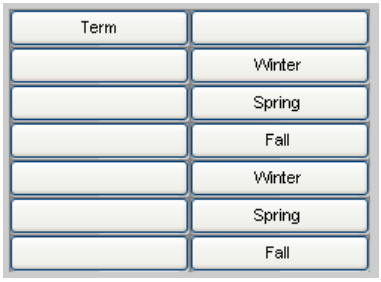
Year	Term	Mark	Mark	Mark	Mark	Mark	Mark
Year	Term	Assignments	Assignments	Assignments	Examinations	Examinations	Grade
Year	Term	Ass1	Ass2	Ass3	Midterm	Final	Grade
1991	Winter	85	80	75	60	75	75
1991	Spring	80	65	75	60	70	70
1991	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
1992	Spring	80	80	70	70	75	75
1992	Fall	75	70	65	60	80	70

6. Indented Table



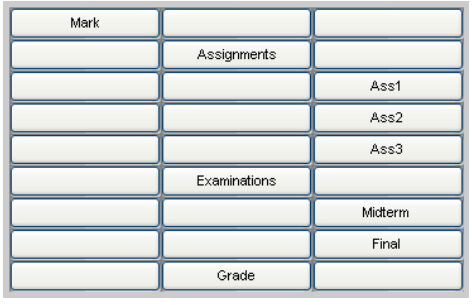
Error Correction GUI

Category 1



Category 2 – incorrect

Category 2 – corrected



Category 3

7. Category Notation:

(Year,{(1991,phi),(1992,phi)})
(Term,{(Winter,phi),(Spring,phi),(Fall,phi)})
(Mark,{(Assignments,{(Ass1,phi),(Ass2,phi),(Ass3,phi)}),(Examinations, ...
{(Midterm,phi),(Final,phi)}),(Grade,phi)})

8. Delta Notation:

```
delta({Mark.Assignments.Ass1 ,Year.1991,Term.Winter })=85
delta({Mark.Assignments.Ass2 ,Year.1991,Term.Winter })=80
delta({Mark.Assignments.Ass3 ,Year.1991,Term.Winter })=75
delta({Mark.Examinations.Midterm ,Year.1991,Term.Winter })=60
delta({Mark.Examinations.Final ,Year.1991,Term.Winter })=75
delta({Mark.Grade,Year.1991,Term.Winter })=75
delta({Mark.Assignments.Ass1 ,Year.1991,Term.Spring })=80
delta({Mark.Assignments.Ass2 ,Year.1991,Term.Spring })=65
delta({Mark.Assignments.Ass3 ,Year.1991,Term.Spring })=75
delta({Mark.Examinations.Midterm ,Year.1991,Term.Spring })=60
delta({Mark.Examinations.Final ,Year.1991,Term.Spring })=70
delta({Mark.Grade,Year.1991,Term.Spring })=70
delta({Mark.Assignments.Ass1 ,Year.1991,Term.Fall })=80
delta({Mark.Assignments.Ass2 ,Year.1991,Term.Fall })=85
delta({Mark.Assignments.Ass3 ,Year.1991,Term.Fall })=75
delta({Mark.Examinations.Midterm ,Year.1991,Term.Fall })=55
delta({Mark.Examinations.Final ,Year.1991,Term.Fall })=80
delta({Mark.Grade,Year.1991,Term.Fall })=75
delta({Mark.Assignments.Ass1 ,Year.1992,Term.Winter })=85
delta({Mark.Assignments.Ass2 ,Year.1992,Term.Winter })=80
delta({Mark.Assignments.Ass3 ,Year.1992,Term.Winter })=70
delta({Mark.Examinations.Midterm ,Year.1992,Term.Winter })=70
delta({Mark.Examinations.Final ,Year.1992,Term.Winter })=75
delta({Mark.Grade,Year.1992,Term.Winter })=75
delta({Mark.Assignments.Ass1 ,Year.1992,Term.Spring })=80
delta({Mark.Assignments.Ass2 ,Year.1992,Term.Spring })=80
delta({Mark.Assignments.Ass3 ,Year.1992,Term.Spring })=70
delta({Mark.Examinations.Midterm ,Year.1992,Term.Spring })=70
delta({Mark.Examinations.Final ,Year.1992,Term.Spring })=75
delta({Mark.Grade,Year.1992,Term.Spring })=75
delta({Mark.Assignments.Ass1 ,Year.1992,Term.Fall })=75
delta({Mark.Assignments.Ass2 ,Year.1992,Term.Fall })=70
delta({Mark.Assignments.Ass3 ,Year.1992,Term.Fall })=65
delta({Mark.Examinations.Midterm ,Year.1992,Term.Fall })=60
delta({Mark.Examinations.Final ,Year.1992,Term.Fall })=80
delta({Mark.Grade,Year.1992,Term.Fall })=70
```

The second example is a table from the geopolitical domain. It is fairly simple, yet very typical of the geopolitical domain. It also contains a footnote.

1. Domain - HTML version:

population per region

year	**Africa**	**Asia**
1750	106,000,000	502,000,000
1800	107,000,000	635,000,000
1850	111,000,000	809,000,000
1900	133,000,000	947,000,000
1950	221,000,000	1,402,000,000
1998	749,000,000	3,585,000,000
2050	1,766,000,000	5,268,000,000

source: United Nations, 1973. "The Determinants and Consequences of Population Trends, Vol.1" (United Nations, New York). United Nations, (forthcoming). "World Population Prospects: The 1998 Revision" (United Nations, New York).

2. Original ASCII:

total table number is 1
################error: undefined table type!

year ** population per region **
Africa ** Asia ** Europe ** Latin Am. & Caribbean ** Northern America ** Oceania ** World **
1750 ** 106,000,000 ** 502,000,000 ** 163,000,000 ** 16,000,000 ** 2,000,000 ** 2,000,000 ** 791,000,000 **
1800 ** 107,000,000 ** 635,000,000 ** 203,000,000 ** 24,000,000 ** 7,000,000 ** 2,000,000 ** 978,000,000 **
1850 ** 111,000,000 ** 809,000,000 ** 276,000,000 ** 38,000,000 ** 26,000,000 ** 2,000,000 ** 1,262,000,000 **
1900 ** 133,000,000 ** 947,000,000 ** 408,000,000 ** 74,000,000 ** 82,000,000 ** 6,000,000 ** 1,650,000,000 **
1950 ** 221,000,000 ** 1,402,000,000 ** 547,000,000 ** 167,000,000 ** 172,000,000 ** 13,000,000 ** 2,521,000,000 **
1998 ** 749,000,000 ** 3,585,000,000 ** 729,000,000 ** 504,000,000 ** 305,000,000 ** 30,000,000 ** 5,901,000,000 **
2050 ** 1,766,000,000 ** 5,268,000,000 ** 628,000,000 ** 809,000,000 ** 392,000,000 ** 46,000,000 ** 8,909,000,000 **
source: United Nations, 1973. "The Determinants and Consequences of Population Trends, Vol.1" (United Nations, New York). United Nations, (forthcoming). "World Population Prospects: The 1998 Revision" (United Nations, New York). **
****
****
****

3. Manually Synthesized ASCII:

****

year [rowspan = 2] ** population per region [colspan = 7] *****
Africa ** Asia ** Europe ** Latin Am. & Caribbean ** Northern America ** Oceania ** World *****
1750 ** 106,000,000 ** 502,000,000 ** 163,000,000 ** 16,000,000 ** 2,000,000 ** 2,000,000 ** 791,000,000 *****
1800 ** 107,000,000 ** 635,000,000 ** 203,000,000 ** 24,000,000 ** 7,000,000 ** 2,000,000 ** 978,000,000 *****
1850 ** 111,000,000 ** 809,000,000 ** 276,000,000 ** 38,000,000 ** 26,000,000 ** 2,000,000 ** 1,262,000,000 *****
1900 ** 133,000,000 ** 947,000,000 ** 408,000,000 ** 74,000,000 ** 82,000,000 ** 6,000,000 ** 1,650,000,000 *****
1950 ** 221,000,000 ** 1,402,000,000 ** 547,000,000 ** 167,000,000 ** 172,000,000 ** 13,000,000 ** 2,521,000,000 *****

1998 ** 749,000,000 ** 3,585,000,000 ** 729,000,000 ** 504,000,000 ** 305,000,000 ** 30,000,000 ** 5,901,000,000 *****
2050 ** 1,766,000,000 ** 5,268,000,000 ** 628,000,000 ** 809,000,000 ** 392,000,000 ** 46,000,000 ** 8,909,000,000 *****
source: United Nations, 1973. "The Determinants and Consequences of Population Trends, Vol.1" (United Nations, New York). United Nations, (forthcoming). "World Population Prospects: The 1998 Revision" (United Nations, New York). *****
****
****
****

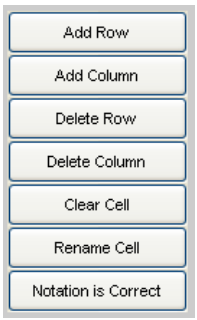
4. Table:

year	population per region	population per region	population per region	population per region	population per region	population per region	population per region
year	Africa	Asia	Europe	Latin Am. & Caribbean	Northern America	Oceania	World
1750	106,000,000	502,000,000	163,000,000	16,000,000	2,000,000	2,000,000	791,000,000
1800	107,000,000	635,000,000	203,000,000	24,000,000	7,000,000	2,000,000	978,000,000
1850	111,000,000	809,000,000	276,000,000	38,000,000	26,000,000	2,000,000	1,262,000,000
1900	133,000,000	947,000,000	408,000,000	74,000,000	82,000,000	6,000,000	1,650,000,000
1950	221,000,000	1,402,000,000	547,000,000	167,000,000	172,000,000	13,000,000	2,521,000,000
1998	749,000,000	3,585,000,000	729,000,000	504,000,000	305,000,000	30,000,000	5,901,000,000
2050	1,766,000,000	5,268,000,000	628,000,000	809,000,000	392,000,000	46,000,000	8,909,000,000
source: United Nations...							

5. Categories:

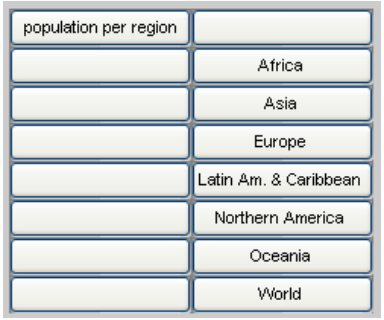
year	population per regi...	population per regi...	population per regi...	population per regi...	population per regi...	population per regi...	population per regi...
year	Africa	Asia	Europe	Latin Am. & Caribbe...	Northern America	Oceania	World
1750	106,000,000	502,000,000	163,000,000	16,000,000	2,000,000	2,000,000	791,000,000
1800	107,000,000	635,000,000	203,000,000	24,000,000	7,000,000	2,000,000	978,000,000
1850	111,000,000	809,000,000	276,000,000	38,000,000	26,000,000	2,000,000	1,262,000,000
1900	133,000,000	947,000,000	408,000,000	74,000,000	82,000,000	6,000,000	1,650,000,000
1950	221,000,000	1,402,000,000	547,000,000	167,000,000	172,000,000	13,000,000	2,521,000,000
1998	749,000,000	3,585,000,000	729,000,000	504,000,000	305,000,000	30,000,000	5,901,000,000
2050	1,766,000,000	5,268,000,000	628,000,000	809,000,000	392,000,000	46,000,000	8,909,000,000
source: United Nations...							

6. Indented Table



Error Correction GUI

Category 1



Category 2

7. Category Notation:

(year,{(1750,phi),(1800,phi),(1850,phi),(1900,phi),(1950,phi), … (1998,phi),(2050,phi)})
(population per region,{(Africa,phi),(Asia,phi),(Europe,phi), (Latin Am. … & Caribbean,phi),(Northern America,phi),(Oceania,phi),(World,phi)})

8. Delta Notation:

delta({population per region.Africa ,year.1750 })=106,000,000
delta({population per region.Asia ,year.1750 })=502,000,000
delta({population per region.Europe ,year.1750 })=163,000,000
delta({population per region.Latin Am. & Caribbean ,year.1750 })=16,000,000
delta({population per region.Northern America ,year.1750 })=2,000,000
delta({population per region.Oceania ,year.1750 })=2,000,000
delta({population per region.World ,year.1750 })=791,000,000
delta({population per region.Africa ,year.1800 })=107,000,000
delta({population per region.Asia ,year.1800 })=635,000,000
delta({population per region.Europe ,year.1800 })=203,000,000
delta({population per region.Latin Am. & Caribbean ,year.1800 })=24,000,000
delta({population per region.Northern America ,year.1800 })=7,000,000
delta({population per region.Oceania ,year.1800 })=2,000,000
delta({population per region.World ,year.1800 })=978,000,000
delta({population per region.Africa ,year.1850 })=111,000,000
delta({population per region.Asia ,year.1850 })=809,000,000
delta({population per region.Europe ,year.1850 })=276,000,000
delta({population per region.Latin Am. & Caribbean ,year.1850 })=38,000,000
delta({population per region.Northern America ,year.1850 })=26,000,000
delta({population per region.Oceania ,year.1850 })=2,000,000
delta({population per region.World ,year.1850 })=1,262,000,000
delta({population per region.Africa ,year.1900 })=133,000,000
delta({population per region.Asia ,year.1900 })=947,000,000
delta({population per region.Europe ,year.1900 })=408,000,000
delta({population per region.Latin Am. & Caribbean ,year.1900 })=74,000,000
delta({population per region.Northern America ,year.1900 })=82,000,000
delta({population per region.Oceania ,year.1900 })=6,000,000
delta({population per region.World ,year.1900 })=1,650,000,000
delta({population per region.Africa ,year.1950 })=221,000,000
delta({population per region.Asia ,year.1950 })=1,402,000,000
delta({population per region.Europe ,year.1950 })=547,000,000
delta({population per region.Latin Am. & Caribbean ,year.1950 })=167,000,000
delta({population per region.Northern America ,year.1950 })=172,000,000
delta({population per region.Oceania ,year.1950 })=13,000,000
delta({population per region.World ,year.1950 })=2,521,000,000
delta({population per region.Africa ,year.1998 })=749,000,000
delta({population per region.Asia ,year.1998 })=3,585,000,000
delta({population per region.Europe ,year.1998 })=729,000,000
delta({population per region.Latin Am. & Caribbean ,year.1998 })=504,000,000
delta({population per region.Northern America ,year.1998 })=305,000,000
delta({population per region.Oceania ,year.1998 })=30,000,000
delta({population per region.World ,year.1998 })=5,901,000,000
delta({population per region.Africa ,year.2050 })=1,766,000,000
delta({population per region.Asia ,year.2050 })=5,268,000,000
delta({population per region.Europe ,year.2050 })=628,000,000
delta({population per region.Latin Am. & Caribbean ,year.2050 })=809,000,000
delta({population per region.Northern America ,year.2050 })=392,000,000
delta({population per region.Oceania ,year.2050 })=46,000,000
delta({population per region.World ,year.2050 })=8,909,000,000

d. Further Automation

The next step after WNT v.3.5 would be to completely automate WNT. A future WNT will be given an input in the form of an HTML table. It will automatically generate the ASCII output and converts that to a Matlab array. (Note that this process is not automatic yet because the user has to run the java project manually, in the future one command should execute every portion of WNT). It will then determine which cells are category cells and which cells are delta cells without any user input. Its guesses can be displayed to a user for correction and approval. This version of WNT will also keep a detailed log, which will aid the program in not making the same mistake twice and in characterizing operator performance. After the category and delta cells are known, the rest of the program is already automatic. There will be one more instance of user correction with the post-editing tool used while generating the indented notation.

Referring to my brief discussion on foreign tables found in the introduction to this report, it can be seen that in at least some instances it is possible to differentiate between category and delta cells using solely the structure of the table. Below are some tables and some possible interpretations for them.

Table 6: Car Table

	1	2	3	4
1		2007 BMW 328i	2206 Audi S4 Quattro	2006 Maserati Coupe GT
2	PRICING			
3	Base Retail	$35,995	$60,970	$84,550
4	Base Invoice	$33,170	$56,082	N/A
5	New Car Blue Book	$35,815	$59,446	$84,550
6	New Car Incentives	Current Incentives	-	-
7	GENERAL INFO			
8	Country of Assembly	Germany	Germany	Italy
9	Country of Origin	Germany	Germany	Italy
10	EPA Class	Compact	Compact	Subcompact
11	Body Style	Coupe	Sedan	Coupe
12	Doors	2	4	2
13	Seating Capacity	4	5	4
14	POWERTRAIN			
15	Engine Code/Name	-	-	-
16	VIN	-	L	-
17	Cylinders	6	V8	V8
18	Displacement	3	4.2	4.2
19	Bore x Stroke	3.31 x 3.53	3.33 x 3.65	3.62 x 3.14
20	Compression Ratio	10.2	11.0	11.1
21	Fuel Type	Gas	Gas	-
22	Fuel Induction	Electronic Fuel	Sequential Fuel	-

Correct Interpretation: 2 categories. First category needs a virtual header and its subcategories are the names of cars (row 1). The other category also needs a virtual header and contains all the cells in column 1.

Possible Interpretations: First note that in the original table, rows 2, 7, and 14 were one merged cell. When the table is transferred into Matlab it breaks

those rows into 4 cells. This is the case regardless of whether we use Excel or Java. Also note that we can usually tell which cells were merged because they will now be empty. This is not always true, but is usually the case. Now recalling my discussion on foreign tables, I stated that the first row/column that contains the largest number of cells (or in the case of Matlab separated cells, the largest number of non-empty cells) is usually the last row/column of category cells. This, again, is only true for some categories, not all.

Using the above information, for this table, a program could guess that the first row and first column are category cells because all rows and columns are equally separated. It can also guess that locations (2,1), (7,1), and (14,1) are top-level headers due to empty cells next to them. It will need to ask the user for input on virtual categories because there is no apparent header in row 1. The empty cell (1,1) could be a problem. Having the program scan the table from the bottom up and from right to left will be much more useful than the usual raster scanning.

Wang Table

	1	2	3	4	5	6	7	8
1	Year	Term	Mark					
2			Assignments			Examinations		Grade
3			Ass1	Ass2	Ass3	Midterm	Final	
4	1991	Winter	85	80	75	60	75	75
5		Spring	80	65	75	60	70	70
6		Fall	80	85	75	55	80	75
7	1992	Winter	85	80	70	70	75	75
8		Spring	80	80	70	70	75	75
9		Fall	75	70	65	60	80	70

Correct: 3 categories. First: Year, 1991, 1992. Second: Term: Winter, Spring, Fall. Third: Mark, Assignments, Examinations, Grade, Ass1, Ass2, Ass3, Midterm, Final.

Interpretations: The Wang table is surprisingly difficult. Going by the foreign table rule, the first row that contains the largest number of non-empty cells is the fourth row, which would be incorrect. And the first column that contains the largest number of cells would be the third column, which is also incorrect. However, from looking at the empty cells, we see that all the rows and columns with empty cells are rows and columns containing category cells. Distinguishing between categories is also tough and there are many different guesses the program could make. It could guess that the first two columns are a category (because the third column is filled in all the way) and it could guess that the first two columns are separate categories. Once the first two columns have been decided, it can guess that the rectangle from (1,3) to (3,8) is a category. The only thing that could throw off a program guess at this point is that cell(3,8) is empty.

6. Table Processing Ontology

This report details the stages WNT has gone through in the course of its development. We have decided to create an ontology of table processing tools to catalog these stages and their relations for other researchers. We just started

looking at this; therefore we decided to start by creating an ontology for WNT v1, which was very simple. It started out with an HTML table that was copy/pasted into Excel and from Excel transferred into Matlab. Then, the Matlab program asked the user many questions regarding the table that the user answered by typing in the command window. Finally, the category and delta notation was given as an output.

The ontology for WNT v1 is a task ontology rather than a domain ontology. An ontology on geopolitical data would be a domain ontology. However since an ontology on table processing tools details how a system performs a task, it would be classified as a task ontology. From "Some answers to questions about ontology from the DEG point of view" by Yuji Tijerino, we know that task ontologies consists of four kinds of concepts:

- **Task roles**, roles played by the domain objects in the problem solving process
- **Task actions**, representing unit activities appearing in the problem solving process
- **States**, of the objects and
- **Other**, concepts specific to the task and not the domain

For WNT v.1:

- **Task roles**: domain object would be the table being used. The role played by the domain object is as an input. The HTML version is a domain object.
- **Task actions**: The activities used to process information would be: converting from HTML to Excel, converting from Excel to Matlab, asking questions and receiving answers from the user, processing data using the answers obtained.
- **States**: some objects in this ontology would be: the Excel version, MATLAB version, answers received from user, category notation (output), and delta notation (output)
- **Other**: none

The following is a graphical representation of the ontology.

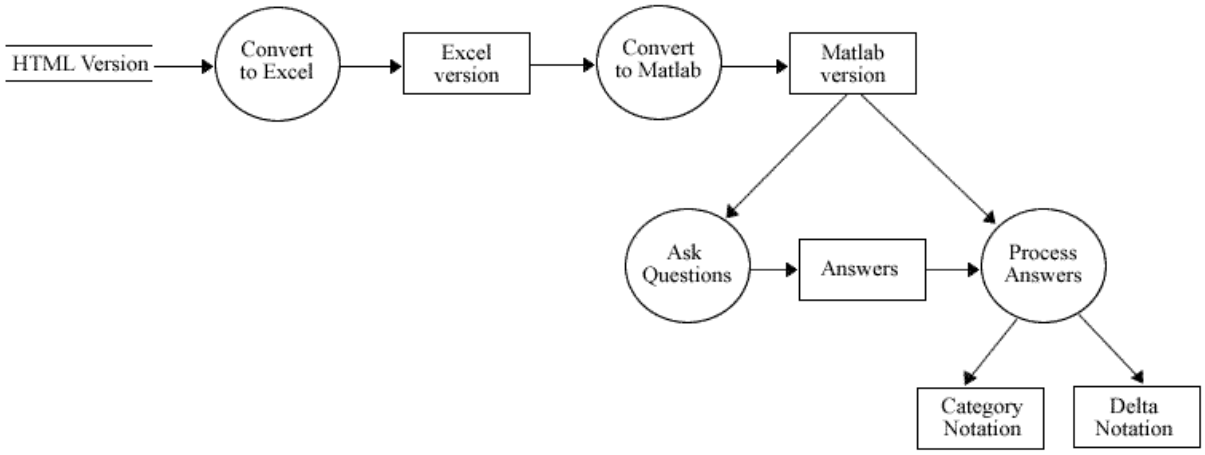


Figure 10: WNT v.1 Ontology

7. Appendix

a. Command Window Output for Version 1 of Wang Notation Tool

```
How many categories are in this table?: 3
Category name? 'Year'
How many subcategories are in this category?: 2
Subcategory name? '1991'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? '1992'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Category name? 'Term'
How many subcategories are in this category?: 3
Subcategory name? 'Winter'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? 'Spring'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? 'Fall'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Category name? 'Mark'
How many subcategories are in this category?: 3
Subcategory name? 'Assignments'
How many subcategories are in this subcategory? (answer 0 for none): 3
Subcategory name? 'Ass1'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? 'Ass2'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? 'Ass3'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
This level is complete.
Subcategory name? 'Examinations'
How many subcategories are in this subcategory? (answer 0 for none): 2
Subcategory name? 'Midterm'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
Subcategory name? 'Final'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
This level is complete.
Subcategory name? 'Grade'
How many subcategories are in this subcategory? (answer 0 for none): 0
No more subcategories
del({Year.1991, Term.Winter, Mark.Assignments.Ass1}) =
Input value for above: '85'
del({Year.1991, Term.Winter, Mark.Assignments.Ass2}) =
Input value for above: '80'
del({Year.1991, Term.Winter, Mark.Assignments.Ass3}) =
Input value for above: '75'
del({Year.1991, Term.Winter, Mark.Examinations.Midterm}) =
Input value for above: '60'
```

```
del({Year.1991, Term.Winter, Mark.Examinations.Final}) =
Input value for above: '75'
del({Year.1991, Term.Winter, Mark.Grade}) =
Input value for above: '75'
del({Year.1991, Term.Spring, Mark.Assignments.Ass1}) =
Input value for above: '80'
del({Year.1991, Term.Spring, Mark.Assignments.Ass2}) =
Input value for above: '65'
del({Year.1991, Term.Spring, Mark.Assignments.Ass3}) =
Input value for above: '75'
del({Year.1991, Term.Spring, Mark.Examinations.Midterm}) =
Input value for above: '60'
del({Year.1991, Term.Spring, Mark.Examinations.Final}) =
Input value for above: '70'
del({Year.1991, Term.Spring, Mark.Grade}) =
Input value for above: '70'
del({Year.1991, Term.Fall, Mark.Assignments.Ass1}) =
Input value for above: '80'
del({Year.1991, Term.Fall, Mark.Assignments.Ass2}) =
Input value for above: '85'
del({Year.1991, Term.Fall, Mark.Assignments.Ass3}) =
Input value for above: '75'
del({Year.1991, Term.Fall, Mark.Examinations.Midterm}) =
Input value for above: '55'
del({Year.1991, Term.Fall, Mark.Examinations.Final}) =
Input value for above: '80'
del({Year.1991, Term.Fall, Mark.Grade}) =
Input value for above: '75'
del({Year.1992, Term.Winter, Mark.Assignments.Ass1}) =
Input value for above: '85'
del({Year.1992, Term.Winter, Mark.Assignments.Ass2}) =
Input value for above: '80'
del({Year.1992, Term.Winter, Mark.Assignments.Ass3}) =
Input value for above: '70'
del({Year.1992, Term.Winter, Mark.Examinations.Midterm}) =
Input value for above: '70'
del({Year.1992, Term.Winter, Mark.Examinations.Final}) =
Input value for above: '75'
del({Year.1992, Term.Winter, Mark.Grade}) =
Input value for above: '75'
del({Year.1992, Term.Spring, Mark.Assignments.Ass1}) =
Input value for above: '80'
del({Year.1992, Term.Spring, Mark.Assignments.Ass2}) =
Input value for above: '80'
del({Year.1992, Term.Spring, Mark.Assignments.Ass3}) =
Input value for above: '70'
del({Year.1992, Term.Spring, Mark.Examinations.Midterm}) =
Input value for above: '70'
del({Year.1992, Term.Spring, Mark.Examinations.Final}) =
Input value for above: '75'
del({Year.1992, Term.Spring, Mark.Grade}) =
Input value for above: '75'
del({Year.1992, Term.Fall, Mark.Assignments.Ass1}) =
Input value for above: '75'
del({Year.1992, Term.Fall, Mark.Assignments.Ass2}) =
Input value for above: '70'
del({Year.1992, Term.Fall, Mark.Assignments.Ass3}) =
Input value for above: '65'
del({Year.1992, Term.Fall, Mark.Examinations.Midterm}) =
```

```
Input value for above: '60'
del({Year.1992, Term.Fall, Mark.Examinations.Final}) =
Input value for above: '80'
del({Year.1992, Term.Fall, Mark.Grade}) =
Input value for above: '70'
```

Correctly generated Wang Notation:

Cat1 = (Year, {(1991,phi), (1992,phi)})
Cat2 = (Term, {(Winter,phi), (Spring,phi), (Fall,phi)})
Cat3 = (Mark, {(Assignments, {(Ass1,phi), (Ass2,phi), (Ass3,phi)}), (Examinations, {(Midterm,phi), (Final,phi)}), (Grade,phi)})

delta_all =
del({Year.1991, Term.Winter, Mark.Assignments.Ass1}) = 85
del({Year.1991, Term.Winter, Mark.Assignments.Ass2}) = 80
del({Year.1991, Term.Winter, Mark.Assignments.Ass3}) = 75
del({Year.1991, Term.Winter, Mark.Examinations.Midterm}) = 60
del({Year.1991, Term.Winter, Mark.Examinations.Final}) = 75
del({Year.1991, Term.Winter, Mark.Grade}) = 75
del({Year.1991, Term.Spring, Mark.Assignments.Ass1}) = 80
del({Year.1991, Term.Spring, Mark.Assignments.Ass2}) = 65
del({Year.1991, Term.Spring, Mark.Assignments.Ass3}) = 75
del({Year.1991, Term.Spring, Mark.Examinations.Midterm}) = 60
del({Year.1991, Term.Spring, Mark.Examinations.Final}) = 70
del({Year.1991, Term.Spring, Mark.Grade}) = 70
del({Year.1991, Term.Fall, Mark.Assignments.Ass1}) = 80
del({Year.1991, Term.Fall, Mark.Assignments.Ass2}) = 85
del({Year.1991, Term.Fall, Mark.Assignments.Ass3}) = 75
del({Year.1991, Term.Fall, Mark.Examinations.Midterm}) = 55
del({Year.1991, Term.Fall, Mark.Examinations.Final}) = 80
del({Year.1991, Term.Fall, Mark.Grade}) = 75
del({Year.1992, Term.Winter, Mark.Assignments.Ass1}) = 85
del({Year.1992, Term.Winter, Mark.Assignments.Ass2}) = 80
del({Year.1992, Term.Winter, Mark.Assignments.Ass3}) = 70
del({Year.1992, Term.Winter, Mark.Examinations.Midterm}) = 70
del({Year.1992, Term.Winter, Mark.Examinations.Final}) = 75
del({Year.1992, Term.Winter, Mark.Grade}) = 75
del({Year.1992, Term.Spring, Mark.Assignments.Ass1}) = 80
del({Year.1992, Term.Spring, Mark.Assignments.Ass2}) = 80
del({Year.1992, Term.Spring, Mark.Assignments.Ass3}) = 70
del({Year.1992, Term.Spring, Mark.Examinations.Midterm}) = 70
del({Year.1992, Term.Spring, Mark.Examinations.Final}) = 75
del({Year.1992, Term.Spring, Mark.Grade}) = 75
del({Year.1992, Term.Fall, Mark.Assignments.Ass1}) = 75
del({Year.1992, Term.Fall, Mark.Assignments.Ass2}) = 70
del({Year.1992, Term.Fall, Mark.Assignments.Ass3}) = 65
del({Year.1992, Term.Fall, Mark.Examinations.Midterm}) = 60
del({Year.1992, Term.Fall, Mark.Examinations.Final}) = 80
del({Year.1992, Term.Fall, Mark.Grade}) = 70

b. Matlab Functions for Tree Manipulations

```
function [table] = build_table_from_tok(tok_t,cats)
% build_table_from_tok constructs a binary table tree using
% insert_left and insert_right functions

clear table*
table1(1).nodename=cats(1,:); % root node
table1(1).pointers=[0,2,0]; % leftpointer of root is always 2
sizetoc=size(tok_t);

%now add new nodes, one at a time`
for i=2:sizetoc(1)
   [father, lr]=backpoint(tok_t,i);
   if lr==-1
      table1 = insert_left(table1,cats(i,:),father,i);
      if father==0; table1(i).pointers=[0,2,i];end
   elseif lr==+1
      table1 = insert_right(table1,cats(i,:),father,i);
      if father==0; table1(i).pointers=[0,2,i];end
      else display 'error in tok'
   end;
end;
table=table1;

sizet=size(cat(1,table.pointers));
blanks=repmat('    ',sizet(1),1)
nodenames={table.nodename} % show new nodenames
display_table=[int2str([1:sizet(1)]'), blanks  ,cats, int2str(cat(1,table.pointers))]


function [father, lr] = backpoint(toc_ttt, tocline);
% father is the backpointer of item tocline in toc,
% lr shows whether it is left (-1) or right (+1) insert;
% for root, lr=0

sizetoc=size(toc_ttt);
if tocline==1,
% check for root node (must be in first line of toc)
   father=0; lr=0;
else
lastnonzero = find(toc_ttt(tocline,:),1, 'last');
if toc_ttt(tocline,lastnonzero)==+1
   lr=-1;
   header=toc_ttt(tocline,:);
%find the row that matches tocline except for lastnonzero
   header(lastnonzero)=0;
   tocheader=repmat(header,sizetoc(1),1);
   A=tocheader==toc_ttt; %the matching row has all 1's
   B=sum(A');     father=find(B==max(B));
else
   lr=+1;
   header=toc_ttt(tocline,:); %find the left sibling of tocline
   header(lastnonzero)=header(lastnonzero)-1;
    tocheader=repmat(header,sizetoc(1),1);
   A=tocheader==toc_ttt; %the matching row has all 1's
   B=sum(A');     father=find(B==max(B));
end;
end;
```

```
function [larger_table] = addnode(table_in, newnode, father, leftright)
% addnode add newnode to table_in, and produces table_out
% adds left son to father if leftright=-1, right son if +1, else error

if leftright==-1
   larger_table = insert_left_node(table_in, newnode, father)
   elseif leftright==+1
   larger_table = insert_right_node(table_in, newnode, father)
else
   display(leftright);
end

showtable(larger_table)


function [table_out] = insert_left_node(table_in,newnode,father);
% a table is represented as a binary tree
% table_in and table_out are struct array with fields: nodename, pointers
% table.pointers(1) is upointer, (2) is left pointer, (3) is right pointer
% creates a new sruct (table_out) for a tree,
% after inserting a new node into table_in below father,
% with up-pointer to father, and down-pointers inherited from father
size_table_in=size(table_in);
table_out=table_in;
table_out(size_table_in(2)+1).nodename = newnode; %add newnode to nodename

% update pointers:
% (1) father's leftpointer to last+1 (where we put newnode)
% (2) newnode's new uppointer to point to father
% (3) newnode's leftpointer to 0
% (4) fatherleftpointer (if any) newnode's rightpointer
table_out(father).pointers(2) = size_table_in(2)+1; %(1)
table_out(size_table_in(2)+1).pointers(1) = father; %(2)
table_out(size_table_in(2)+1).pointers(2) = 0; %(3)

if table_in(father).pointers(3)~=0; % (4)
   table_out(size_table_in(2)+1).pointers(3) = [table_in(father).pointers(2)]; %(4);
else table_out(size_table_in(2)+1).pointers(3) = 0; % (4)
end;


function [table_out] = insert_left(table_in,newnode,father,i);
% a table is represented as a binary tree
% table_in and table_out are struct array with fields: nodename, pointers
% table.pointers(1) is upointer, (2) is left pointer, (3) is right pointer
% creates a new sruct (table_out) for a tree,
% after inserting a new node into table_in below father,
% with up-pointer to father, and down-pointers inherited from father
   % size_table_in=size(table_in);
table_out=table_in;
% table_out(size_table_in(2)+1).nodename = newnode; %add newnode to nodename
table_out(i).nodename = newnode; %add newnode to nodename

% update pointers:
% (1) father's leftpointer to i (where we put newnode)
% (2) newnode's new uppointer to point to father
% (3) newnode's leftpointer to 0
% (4) fatherleftpointer (if any) to newnode's rightpointer
table_out(father).pointers(2) = i; %(1)
```

```
   % table_out(size_table_in(2)+1).pointers(2) = 0; %(3)
table_out(i).pointers(1) = father; %(2)
table_out(i).pointers(2) = 0; %(3)

if table_in(father).pointers(2)~=0; % (4)
   %    table_out(size_table_in(2)+1).pointers(3) = [table_in(father).pointers(2)];
%(4);
   table_out(i).pointers(3) = [table_in(father).pointers(2)]; %(4);
   % else table_out(size_table_in(2)+1).pointers(3) = 0; % (4)
   else table_out(i).pointers(3) = 0; % (4)
end;


function[cat_indent,cat_com_indent] =
indent_tbl(cat_newest,cat_com_newest)
% This function creates an indented table format from a cell array.

a = size(cat_newest);   t = 0;
for k3 = 1:a(2),
   for k4 = 1:a(1),
      if cat_com_newest(k4,k3) == 1;
         t = t + 1;
         cat_indent(t,k4) = cat_newest(k4,k3);
         cat_com_indent(t,k4) = t;
      end
   end
end
b = size(cat_com_indent);
for k3 = 1:b(1),
   for k4 = 1:b(2),
      if cat_com_indent(k3,k4) == 0;
         cat_indent(k3,k4) = {' '};
      end
   end
end


function[cats,toc] = create_toc(cat_indent,cat_com_indent)
% This function creates a table of contents format from the indented
% notation.

% Create table of contents format - DEPTH FIRST
a = size(cat_indent);
toc = zeros (a(1),a(2));
toc(1,1) = 1;      cats = char(cat_indent(1,1));
cat_count = zeros(a(1),a(2));
for k5 = 2:a(1),
   for k6 = 1:a(2),
      if cat_com_indent(k5,k6) ~= 0;
         cats = strvcat(cats,char(cat_indent(k5,k6)));
         for k7 = 1:k5,
            if cat_com_indent(k5-k7+1,k6-1) ~= 0;
               yo = cat_com_indent(k5-k7+1,k6-1);
               cat_count(k5-k7+1,k6-1) = cat_count(k5-k7+1,k6-1) + 1;
               toc(k5,:) = toc(yo,:);
               toc(k5,k6) = cat_count(k5-k7+1,k6-1);
               break
            end
         end
```

```
        end
    end
end


function[ptt] = preorder_cat_notation(ptt,table_in,i)
% This function gives the pre-order result of a tree traversal
% with the category notation in place.
% 'ptt' is the array that contains the pre-order traversal.
% Since this is a recursive function, ptt is present in both the
% input and the output.  'table_in' is the table, or tree, that
% is to be traversed and a struct array with nodename and pointers
% designated. 'i' is the cell/node within the table/tree.

global tv;  global h;   global s;   global b; global i_all;

% ADDING WANG NOTATION
if i == 1;
    wang = ['(',strtrim(table_in(i).nodename),','];
else

if table_in(i).pointers(1,2) ~= 0 && table_in(i).pointers(1,3) ~= 0;
    if table_in(table_in(i).pointers(1,1)).pointers(1,2) == i;
        wang = ['{(',strtrim(table_in(i).nodename),','];
    elseif table_in(table_in(i).pointers(1,1)).pointers(1,2) ~= i;
        wang = ['(',strtrim(table_in(i).nodename),','];
    end
elseif table_in(i).pointers(1,2) ~= 0 && table_in(i).pointers(1,3) == 0;
    wang = ['(',strtrim(table_in(i).nodename),','];
elseif table_in(i).pointers(1,2) == 0 && table_in(i).pointers(1,3) ~= 0;
    if table_in(table_in(i).pointers(1,1)).pointers(1,2) == i;
        wang = ['{(',strtrim(table_in(i).nodename),',phi),'];
    elseif table_in(table_in(i).pointers(1,1)).pointers(1,2) ~= i;
        wang = ['(',strtrim(table_in(i).nodename),',phi),'];
    end
elseif table_in(i).pointers(1,2) == 0 && table_in(i).pointers(1,3) == 0;
    wang = ['(',strtrim(table_in(i).nodename),',phi)}),'];
end

end

ptt = [ptt,wang];
i_all = [i_all,i];
tv = tv+1;

% TREE TRAVERSAL
while tv < s;

if table_in(i).pointers(1,2) ~= 0 && table_in(i).pointers(1,3) ~= 0;
    b = b + 1;
    h(b) = table_in(i).pointers(1,3);
    k = table_in(i).pointers(1,2);
    ptt = preorder_cat_notation(ptt,table_in,k);
elseif table_in(i).pointers(1,2) ~= 0 && table_in(i).pointers(1,3) == 0;
    k = table_in(i).pointers(1,2);
    ptt = preorder_cat_notation(ptt,table_in,k);
elseif table_in(i).pointers(1,2) == 0 && table_in(i).pointers(1,3) ~= 0;
    k = table_in(i).pointers(1,3);
    ptt = preorder_cat_notation(ptt,table_in,k);
```

```
elseif table_in(i).pointers(1,2) == 0 && table_in(i).pointers(1,3) == 0;
   for bc = 1:length(i_all),
      if i_all(1,bc) == h(b);
         blc = 1;
         break
      elseif i_all(1,bc) ~= h(b)
         blc = 0;
      end
   end
   if blc == 1;
      ptt = preorder_cat_notation(ptt,table_in,h(b-1));
   elseif blc == 0;
      ptt = preorder_cat_notation(ptt,table_in,h(b));
   end
end
end
```