Dynamic Translation of Spreadsheet Formulas to Problem Domain Narratives

Bennett Kankuzi
Department of Computer Science
North-West University, Mafikeng Campus
P/Bag X2046, Mmabatho 2735, South Africa
bennett.kankuzi@nwu.ac.za

Abstract

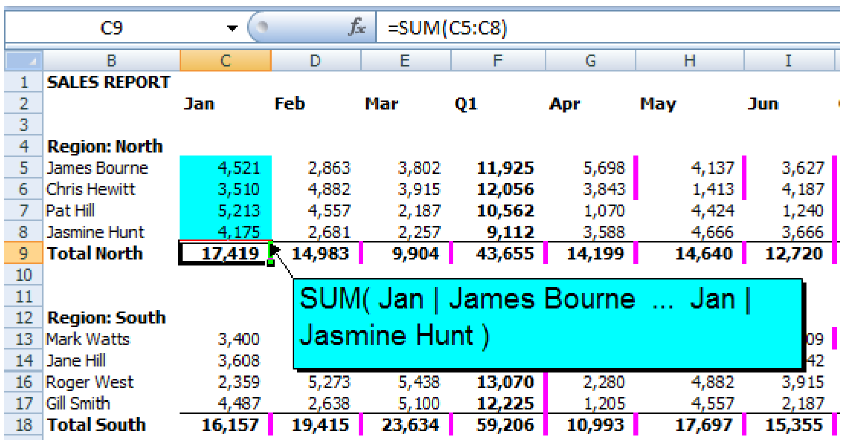
Most errors in spreadsheets are formula-based. Referenced cells in formulas are normally presented using the traditional A1 cell referencing style. A spreadsheet user has to therefore mentally map referenced cells to their corresponding labels in order to comprehend a formula in the context of the problem domain. In this paper, we give a detailed description of an algorithm that can be used to dynamically translate traditional spreadsheet formulas to their problem domain equivalents which are easier to understand. The translation is done as one accesses a formula cell in a spreadsheet. The formula translation is based on inferred labels of referenced cells in the formula. The aim of the translation is to ease the cognitive load on the spreadsheet user and hence improving the error-prone spreadsheet development process. The paper also highlights some factors that need to be taken into consideration when dynamically translating spreadsheet formulas. The number of referenced cells in a formula and the distance of labels from referenced cells will determine the speed of the translation process hence affecting system responsiveness as one navigates through a spreadsheet. Unpredictable spatial arrangement of data in spreadsheets (spreadsheet layout) can also pose a challenge to the translation algorithm which may lead to mis-translation of spreadsheet formulas. These challenges might increase the cognitive load on the spreadsheet user hence negating the purpose of dynamically translating spreadsheet formulas.

1. Introduction

Spreadsheets are one of the most popular end-user programming environments (Abraham, Burnett, & Erwig, 2008). Their simple and easy to use tabular visual interface makes it easy to manipulate data and instantly see the results of the data manipulation (B. A. Nardi & Miller, 1990). However, spreadsheets are plagued with the problem of being vulnerable to errors (Powell, Baker, & Lawson, 2009). In spreadsheet cells, one can enter either labels, data values or formulas (Abraham et al., 2008). A data value or formula without corresponding labels is meaningless as the labels carry the meaning of the cell value or formula in relation to the context of the problem being solved using a particular spreadsheet.

Computations in spreadsheets are entered through formulas. Research, however, has also shown that most errors in spreadsheets are formula based (Rajalingham, Chadwick, Knight, & Edwards, 2000). Formulas normally reference other cells in a spreadsheet. Referenced cells in formulas are normally presented using the traditional A1 cell referencing style such as B3, D7, G12, etc. A spreadsheet user has to therefore mentally map referenced cells to their corresponding labels in order to comprehend the meaning of a formula in the context of the problem domain. This increases the cognitive load of the spreadsheet user as the mental model of the problem domain has to be constantly mapped back and forth to its spreadsheet counterpart (Kankuzi & Sajaniemi, 2013). Cognitive load can be defined as the mental effort one has to make to learn new information in order to accomplish an intellectual task (Sweller, 1994). The higher the cognitive load, the greater the risk for one to commit an error when completing a task (Paas, Renkl, & Sweller, 2004). It is thus, imperative that even in the formula comprehension process, spreadsheet users do not have high cognitive load.

Usability is an important aspect of human-centred software engineering (Seffah & Metzker, 2004) and a recurring usability goal is to reduce cognitive load for users (Hollender, Hofmann, Deneke, & Schmitz, 2010). One technique of reducing cognitive load in user interfaces is by having users focus on recognition rather than on recall of interface elements (Hollender et al., 2010). One way to do this in spreadsheets is by translating a given traditional spreadsheet formula into its problem domain equivalent which



C9 =SUM(C5:C8)

	B	C	D	E	F	G	H	I
1	**SALES REPORT**							
2		**Jan**	**Feb**	**Mar**	**Q1**	**Apr**	**May**	**Jun**
3								
4	**Region: North**							
5	James Bourne	4,521	2,863	3,802	**11,925**	5,698	4,137	3,627
6	Chris Hewitt	3,510	4,882	3,915	**12,056**	3,843	1,413	4,187
7	Pat Hill	5,213	4,557	2,187	**10,562**	1,070	4,424	1,240
8	Jasmine Hunt	4,175	2,681	2,257	**9,112**	3,588	4,666	3,666
9	**Total North**	**17,419**	**14,983**	**9,904**	**43,655**	**14,199**	**14,640**	**12,720**
10								
11								
12	**Region: South**							
13	Mark Watts	3,400						09
14	Jane Hill	3,608						42
16	Roger West	2,359	5,273	5,438	**13,070**	2,280	4,882	3,915
17	Gill Smith	4,487	2,638	5,100	**12,225**	1,205	4,557	2,187
18	**Total South**	**16,157**	**19,415**	**23,634**	**59,206**	**10,993**	**17,697**	**15,355**

SUM(Jan | James Bourne ... Jan | Jasmine Hunt)

Figure 1 – The interactive spreadsheet visualization tool: formula in cell C9 is translated to a problem domain narrative "SUM(Jan | James Bourne ... Jan | Jasmine Hunt)" and referenced cells C5, C6, C7 and C8 are highlighted.

is easier to understand in relation to the problem being solved.

In previous work, Kankuzi and Sajaniemi (2014a, 2014b) developed an interactive spreadsheet visualization tool that dynamically translates a given traditional spreadsheet formula into its problem domain equivalent which is easier to understand in relation to the problem being solved. The formula translation is based on inferred labels of referenced input cells in the formula and is done as one accesses a formula cell. The inferred labels for a particular referenced input cell form symbolic names for the cell. The translated spreadsheet formula is called a "problem domain narrative" or in short a "domain narrative". For an active formula cell, a domain narrative is displayed in a box just right below it. Domain narratives are positioned a little bit lower than the active formula cell to avoid distractions when navigating through the cells. The tool also highlights all referenced cells in an active formula cell and marks a formula cell with a pink right border. For example, in the spreadsheet depicted in Figure 1, a formula in cell C9 given as "= SUM(C5:C8)", is automatically translated to "SUM(Jan | James Bourne ... Jan | Jasmine Hunt)". An empirical evaluation of the tool by Kankuzi and Sajaniemi (2014b, 2016) found that it helped in reducing cognitive load of spreadsheet users as it helped in mapping a spreadsheet user's spreadsheet specific mental model to the corresponding problem domain mental model. The tool also helped spreadsheet users to statically identify more errors in spreadsheets (Kankuzi & Sajaniemi, 2014b, 2016).

This paper has two aims. First, the paper gives a detailed description of the translation algorithm that is the engine of the interactive spreadsheet visualization tool presented and evaluated in Kankuzi and Sajaniemi (2014a, 2014b, 2016). Second, the paper also highlights factors that need to be taken into consideration when dynamically translating spreadsheet formulas. We therefore present a technical performance analysis of the algorithm in particular in relation to system responsiveness which is an important usability factor in interactive tools (Waloszek & Kreichgauer, 2009). We thus analyze the time and space complexity of the algorithm. We also discuss how the problem of unpredictable spreadsheet layouts (Abraham & Erwig, 2004; Koci, Thiele, Romero Moral, & Lehner, 2016; Roy, Hermans, Aivaloglou, Winter, & van Deursen, 2016) can affect the formula translation process.

The rest of the paper is organized as follows: in Section 2, we present our algorithm for dynamically translating spreadsheet formulas. In Section 3, we present and discuss factors that need to be considered when dynamically translating spreadsheet formulas to problem domain narratives. We present related literature in Section 4. Concluding remarks are given in Section 5.

2. The Algorithm

In this section, we present an algorithm for dynamically translating spreadsheet formulas to their corresponding problem domain narratives.

First, we get all the cells being referenced in a given active spreadsheet formula.

Second, for each referenced cell, we do the following in sequence:

(i) We extract the column label and row label for the referenced cell. A column label is extracted with respect to the relative position of the referenced cell in a spreadsheet. From the position of the referenced cell, a column label is extracted by visiting upwards all cells in the same column as the reference input cell and in the process concatenating contents of all label cells. A column label can thus be formed from more than one label cell. A "label cell" may be defined as a non-empty cell which does not have both precedent cells and dependent cells (Abraham & Erwig, 2004). In other words, a label cell is a non-empty cell which is not referencing any other cell and it is also not referenced by other cells. A label cell is therefore not just restricted to text but can also can also contain dates, numbers such as years, etc. We terminate the column label forming process in two situations. If we have encountered an empty cell after already visiting a label cell, that will signify an end of a spreadsheet logical block as spreadsheet users normally separate spreadsheet logical blocks in spreadsheets with empty cells. We also terminate the process if we reach the top boundary of the spreadsheet interface.

    Similarly, a row label is extracted with respect to the relative position of the referenced cell in a spreadsheet. From the position of the referenced cell, a row label is extracted by visiting leftwards all cells in the same column as the reference input cell and in the process concatenating contents of all label cells. Just as a column label, a row label can also be formed from more than one label cell. We also terminate the row label forming process in two situations. If we have encountered an empty cell after already visiting a label cell, that will signify an end of a spreadsheet logical block. We also terminate the process if we reach the left boundary of the spreadsheet interface.

(ii) A symbolic name for the referenced cell is then formed by concatenating the extracted column label with the extracted row label but separating them by a "|" to have *column label* | *row label* as a symbolic name. The naming follows a "column | row" direction because traditional spreadsheet cell naming follows the column-row convention. There are also some situations where a referenced cell can have an incomplete symbolic name: a referenced cell which does not have a corresponding column label has its row label as its symbolic name; a referenced cell which does not have a corresponding row label has its column label as its symbolic name; and a referenced cell that does not have both its corresponding row label and column label has "unnamed" as its symbolic name.

(iii) Each of the textual references of the referenced cell in the untranslated or partly translated spreadsheet formula are replaced with the symbolic name and thus progressively forming a domain narrative. Occurrences of the range operator ":" in a formula are also replaced by " ... ".

Third, a fully translated spreadsheet formula or domain narrative is displayed to the spreadsheet user. For example, for the formula in cell C9 in Figure 1 given by "=SUM(C5:C8)", its corresponding problem domain narrative is "SUM(Jan | James Bourne ... Jan | Jasmine Hunt)". The pseudo-code for the translation algorithm is given in Algorithm 1.

3. Challenges in Dynamic Translation of Spreadsheet Formulas

3.1. System Responsiveness

We analyze the time and space complexity of the translation algorithm given in Algorithm 1 to see the effect of the translation algorithm on system responsiveness. From the algorithm, we can deduce that the time complexity is $O(mn)$, where m is the number of referenced cells in a formula and n is the number of cells that have to be visited in order to form a symbolic name of each referenced cell. A large number

Algorithm 1 An algorithm for dynamically translating a spreadsheet formula into a problem domain narrative

```
procedure GENERATEDOMAINNARRATIVE(activeFormulaCell)
    referencedCellsList ← GETREFERENCEDCELLSLIST(activeFormulaCell)
    for each referencedCell in referencedCellsList do
        columnLabel ← GETCOLUMNLABEL(referencedCell)
        rowLabel ← GETROWLABEL(referencedCell)
        if isNotEmpty(columnLabel) and isNotEmpty(rowLabel) then
            symbolicName ← columnLabel + " | " + rowLabel
        else
            if isNotEmpty(columnLabel) and isEmpty(rowLabel) then
                symbolicName ← columnLabel
            else
                if isEmpty(columnLabel) and isNotEmpty(rowLabel) then
                    symbolicName ← rowLabel
                else
                    symbolicName ← "unnamed"
                end if
            end if
        end if
        Replace referencedCell with symbolicName in spreadsheet formula to progressively
        form a domain narrative
    end for
    Store or display generated domain narrative
end procedure

function GETREFERENCEDCELLSLIST(activeFormulaCell)
    return a list of all cells being referenced in activeFormulaCell
end function

function GETCOLUMNLABEL(referencedCell)
    position currentUpwardColumnCell indicator one cell up in the column of the referencedCell
    to get currentUpwardColumnCell
    while (not(topSpreadsheetBoundary)) do
        if hasNoPrecedents(currentUpwardColumnCell)
      and hasNoDependents(currentUpwardColumnCell) then
            columnLabel ← currentUpwardColumnCell + columnLabel
        end if
        if isBlank(currentUpwardColumnCell) and isNotEmpty(columnLabel) then
            exit while loop
        end if
        position currentUpwardColumnCell indicator one cell up in the column of the
        referencedCell to get currentUpwardColumnCell
    end while
    return columnLabel
end function
```

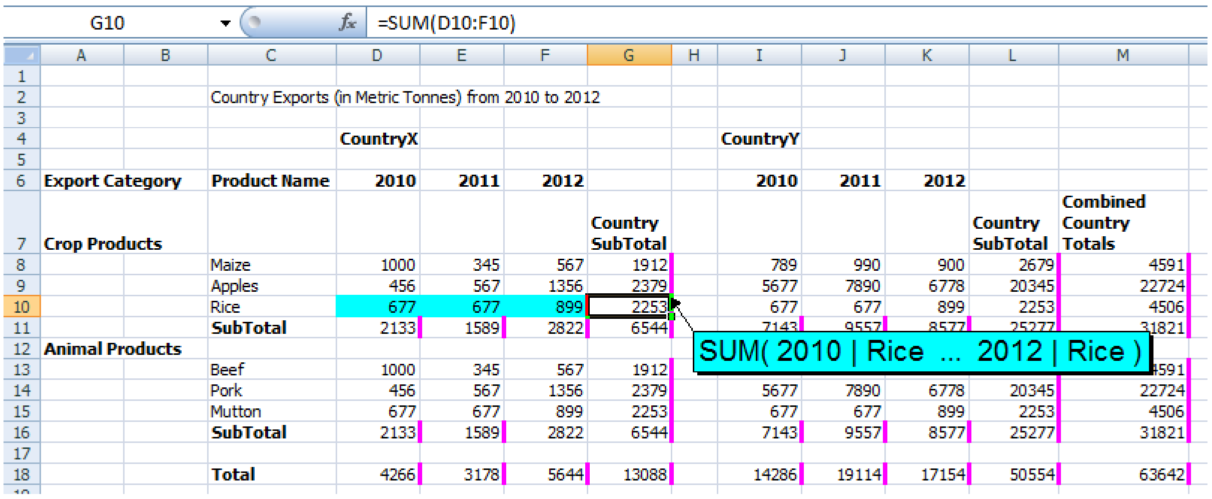
```
42: function GetRowLabel($referencedCell$)
43:     position $currentLeftwardRowCell$ indicator one cell left in the row of the $referencedCell$
44:     to get $currentLeftwardRowCell$
45:     while (not($leftSpreadsheetBoundary$)) do
46:         if hasNoPrecedents($currentLeftwardRowCell$)
      and hasNoDependents($currentLeftwardRowCell$) then
47:             $rowLabel \leftarrow currentLeftwardRowCell + rowLabel$
48:         end if
49:         if isBlank($currentLeftwardRowCell$) and isNotEmpty($rowLabel$) then
50:             exit while loop
51:         end if
52:         position $currentLeftwardRowCell$ indicator one cell left in the row of the $referencedCell$
53:         to get $currentLeftwardRowCell$
54:     end while
55:     return $rowLabel$
56: end function
```

of referenced cells will imply that the formula translation process will also take longer as there will be a large number of referenced cells that will need to be translated. If labels are located further away from the referenced cells, the translation process will take longer as there will be a large number of cells to traverse before reaching desired label cells. Longer translation times will imply slow system response as one navigates from one formula cell to the other. However, waiting for system response for more than an expected period affects short-term memory as information in short-term memory decays over time (Miller, 1968). Human beings also want to feel in control rather than having the pace of their task being controlled by a computer (Miller, 1968). For dynamic translation of spreadsheet formulas to be effective, the number of referenced cells in a spreadsheet formula therefore needs to be relatively small and the label cells should be located closer to the referenced cells.

The algorithm has also space complexity $O(n)$, where n is the number of referenced cells in a formula. A large number of referenced cells will imply that the formula translation process will need more memory to hold the collection of referenced cells. This might also affect system responsiveness as one accesses formula cells in a spreadsheet. The number of referenced cells in a spreadsheet formula therefore needs to be relatively small for dynamic formula translation to be effective.

3.2. Varying Spreadsheet Layouts

Varying spreadsheet layouts also pose a challenge to formula translations as spreadsheet creators can layout their spreadsheets in any way they want (Abraham & Erwig, 2004; Koci et al., 2016; Roy et al., 2016). This challenge applies to formula translations in general and not just to dynamic formula translations. Some spreadsheets can be quite simple while others can be complicated with hierarchical headers, repeated data blocks, vertical data blocks sharing same column headers, blank rows separating blocks, etc (Roy et al., 2016). Our algorithm produces symbolic names which are inferred by inspecting label cells, column wise or row-wise, from a particular referenced cell. This might lead to incomplete symbolic names in some translated formulas in complex spreadsheets which might end up confusing the spreadsheet user. To mitigate this problem, in the spreadsheet visualization tool which uses this algorithm, we also highlight the cells which are being referenced in an active formula cell. For example, in the spreadsheet in Figure 2, the formula in cell G10, given by "=SUM(D10:F10)" and has been dynamically translated to "SUM(2010 | Rice ... 2012 | Rice)". The same translation would occur for the formula in cell L10. This would be confusing to the spreadsheet user. A complete and correct translation for formula in cell G10 would have included the hierarchical headers and consequently would have been translated as "SUM(CountryX ~2010 | Crop Products ~Rice ... CountryX ~2012 | Crop Products ~Rice)". On the other hand, the formula in cell 10 would be translated as "SUM(CountryY ~2010 | Crop Products ~Rice ... CountryY ~2012 | Crop Products ~Rice)". To compensate for this shortfall in our

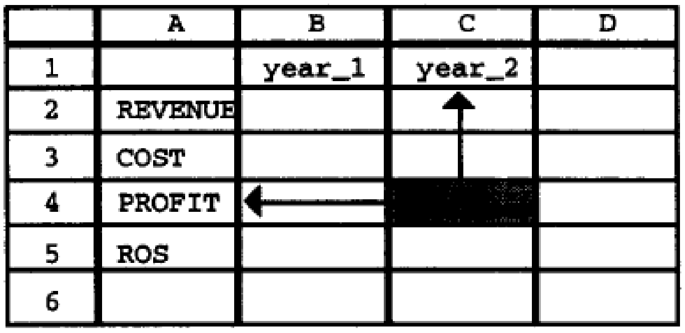


G10 =SUM(D10:F10)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2			Country Exports (in Metric Tonnes) from 2010 to 2012										
3													
4				**CountryX**					**CountryY**				
5													
6	**Export Category**		**Product Name**	**2010**	**2011**	**2012**			**2010**	**2011**	**2012**		
7	**Crop Products**						**Country SubTotal**					**Country SubTotal**	**Combined Country Totals**
8			Maize	1000	345	567	1912		789	990	900	2679	4591
9			Apples	456	567	1356	2379		5677	7890	6778	20345	22724
10			Rice	677	677	899	2253		677	677	899	2253	4506
11			**SubTotal**	2133	1589	2822	6544		7143	9557	8577	25277	31821
12	**Animal Products**												
13			Beef	1000	345	567	1912						4591
14			Pork	456	567	1356	2379		5677	7890	6778	20345	22724
15			Mutton	677	677	899	2253		677	677	899	2253	4506
16			**SubTotal**	2133	1589	2822	6544		7143	9557	8577	25277	31821
17													
18			**Total**	4266	3178	5644	13088		14286	19114	17154	50554	63642

SUM(2010 | Rice ... 2012 | Rice)

Figure 2 – A spreadsheet with hierarchical headers and repeated data blocks depicting country exports for years 2010 to 2012 for CountryX and CountryY. Formula in cell G10 has been translated as "SUM(2010 | Rice ... 2012 | Rice)". A correct and complete translation would have been "SUM(CountryX ~2010 | Crop Products ~Rice ... CountryX ~2012 | Crop Products ~Rice)".



	A	B	C	D
1		year_1	year_2	
2	REVENUE			
3	COST			
4	PROFIT			
5	ROS			
6				

Figure 3 – An illustration of how a symbolic name is assigned to a cell in the tool by D. Nardi and Serrecchia (1994). Cell C4 is referred as "PROFIT(year_2)".

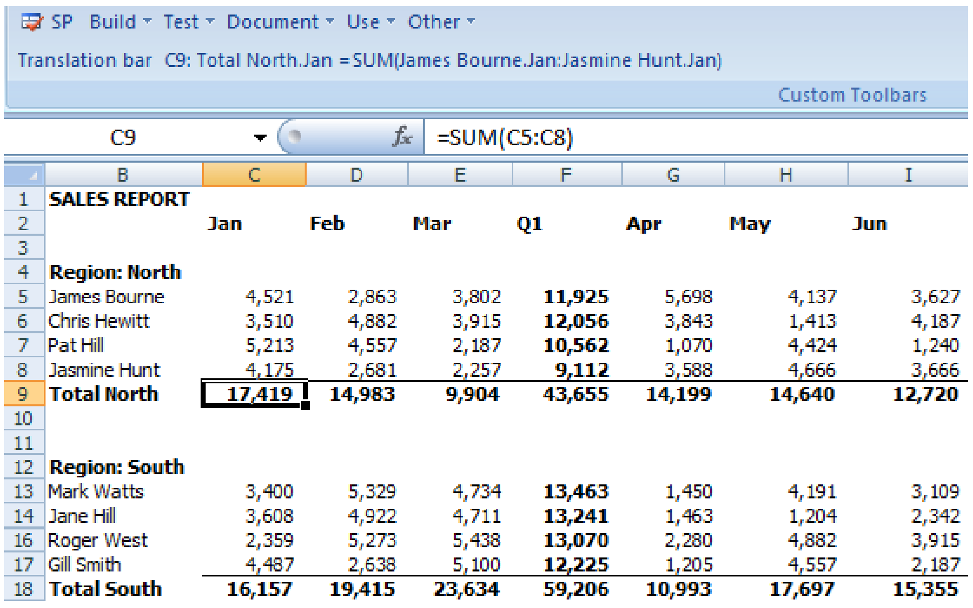
algorithm, the spreadsheet visualization tool (Kankuzi & Sajaniemi, 2014a, 2014b, 2016) highlights the cells being referenced in the active formula cell. In Figure 2, referenced cells D10, E10 and F10 have been highlighted accordingly for formula in cell G10.

4. Related Work

Symbolic names and spreadsheet formula translations have also been used in some spreadsheet auditing tools as well as some spreadsheet systems. A notable difference among the tools, however, being in syntactical notation and location of display of a translated spreadsheet formula.

D. Nardi and Serrecchia (1994) developed a spreadsheet visualization tool that translates spreadsheet formulas. In this tool, labels for input cells to a formula are given out as symbolic names as illustrated in Fig. 3 in which cell C4 is translated as "PROFIT(year_2)" and are displayed in an interface separate from the native spreadsheet system. If in Figure 3, the formula in cell B4 was =B2-B3, the tool by D. Nardi and Serrecchia (1994) would generate an alternative corresponding explanation for the formula as "REVENUE(year_1) - COST(year_1)".

Spreadsheet Professional, a spreadsheet add-in developed by Spreadsheet Innovations Ltd (2017), translates formulas whereby input cell references in a formula are replaced by symbolic names obtained by searching for the first textual cell leftwards (or optionally rightwards) and/or upwards (or downwards) from the referenced cell. The drawback with finding symbolic names by just searching for the first textual cell (leftwards, etc) is that it leaves out the neighbouring text which might form a "whole" symbolic name as in many spreadsheets, symbolic names may occupy more than one cell. In Figure 4, Spreadsheet Professional translates the formula, "=SUM(C5:C8)" in the active cell given in cell C9 in the spreadsheet captured as "= SUM(James Bourne.Jan:Jasmine Hunt.Jan)". And in the translation bar, Spreadsheet Professional displays "C9: Total North.Jan = SUM(James Bourne.Jan:Jasmine Hunt.Jan)". Spreadsheet Detective, a spreadsheet add-in developed by Southern Cross Software (2017), also translates spreadsheet formulas into a form similar to that of Spreadsheet Professional. Numbers, a spreadsheet application developed by Apple Inc. (2017), also automatically translates spreadsheet formulas by creating named ranges over given data after adding data and headers.



Translation bar C9: Total North.Jan =SUM(James Bourne.Jan:Jasmine Hunt.Jan)

C9 fx =SUM(C5:C8)

	B	C	D	E	F	G	H	I
1	**SALES REPORT**							
2		**Jan**	**Feb**	**Mar**	**Q1**	**Apr**	**May**	**Jun**
3								
4	**Region: North**							
5	James Bourne	4,521	2,863	3,802	**11,925**	5,698	4,137	3,627
6	Chris Hewitt	3,510	4,882	3,915	**12,056**	3,843	1,413	4,187
7	Pat Hill	5,213	4,557	2,187	**10,562**	1,070	4,424	1,240
8	Jasmine Hunt	4,175	2,681	2,257	**9,112**	3,588	4,666	3,666
9	**Total North**	**17,419**	**14,983**	**9,904**	**43,655**	**14,199**	**14,640**	**12,720**
10								
11								
12	**Region: South**							
13	Mark Watts	3,400	5,329	4,734	**13,463**	1,450	4,191	3,109
14	Jane Hill	3,608	4,922	4,711	**13,241**	1,463	1,204	2,342
16	Roger West	2,359	5,273	5,438	**13,070**	2,280	4,882	3,915
17	Gill Smith	4,487	2,638	5,100	**12,225**	1,205	4,557	2,187
18	**Total South**	**16,157**	**19,415**	**23,634**	**59,206**	**10,993**	**17,697**	**15,355**

Figure 4 – A Spreadsheet Professional translation of the formula in the active cell, C9, displayed in a special translation bar.

Except for the tool by D. Nardi and Serrecchia (1994), in the other tools and spreadsheet system just stated above, the algorithms on how the formula translations are done, are not explicitly stated.

Hermans, Pinzger, and Van Deursen (2011), however, also presented a spreadsheet formula translation algorithm in which A1 cell references in a spreadsheet formula are also replaced by cell labels. However, their algorithm differs from our algorithm in many aspects. First, in their algorithm, classification of cells is done before the translation process starts i.e. all cells are inspected and marked as formula cell, data cell or label cell. In our case, our algorithm determines whether a cell is a label cell, during the formula translation process as one accesses a formula cell in a spreadsheet. We also do not initially classify formula or data cells. And our algorithm, determines a label cell by simply checking whether it does not have both precedent and dependent cells. This way, label cells can contain text, numbers, dates, etc. Second, in the algorithm by Hermans et al. (2011), cell references in a formula are replaced by labels obtained by searching for the first label in a row or a column. The drawback with this approach is that it leaves out neighbouring labels which might form a complete symbolic name as in many spreadsheets, symbolic names may occupy more than one cell. For example, if a name of a person is split into "first name" and "last name" columns, the algorithm by Hermans et al. (2011) will only pick the first name as the label. In our case, our algorithm forms complete symbolic names by concatenating neighbouring label cells. In other words, our algorithm does not just pick the first label cell in a row or column. Third, the environment of the algorithm by Hermans et al. (2011) is a static spreadsheet tool that is used to generate leveled high level data flow diagrams to aid spreadsheet comprehension. In our case, our algorithm is the engine of an interactive spreadsheet visualization tool, within the spreadsheet environment, whose purpose is to ease cognitive load on the spreadsheet user by helping to map a spreadsheet user's spreadsheet specific mental model to the corresponding problem domain mental model (Kankuzi & Sajaniemi, 2014a, 2014b) through the dynamic spreadsheet formula translations. Lastly, in our work, we have also gone further to give a detailed description of the algorithm as well as a technical analysis of the performance of the algorithm in relation to system responsiveness.

Symbolic names in formula translations or problem domain narratives may appear to be similar to standard named ranges in spreadsheets. The usefulness of standard named ranges has, however, been subject to debate. For example, McKeever, McDaid, and Bishop (2009) found that the use of named ranges may lead to a reduction in debugging performance, particularly for novice spreadsheet users. Standard named ranges, however, are statically defined by a spreadsheet user hence being more error-prone. On the other hand, domain narratives or symbolic names in formula translations as used in our context are generated automatically as one accesses a formula cell and they are accompanied by a clear visualization of the referenced cells. Panko and Ordway (2005) also argue that in selecting ranges for standard named ranges, pointing errors can assign the wrong range to a range name and thus although the range will be wrong, but it will appear to be correct in formulas that merely reference the range names. This can not happen with domain narratives as they are automatically generated and each narrative corresponds to the referenced range area and as such a "correct range name but an incorrect range" type of error (Panko & Ordway, 2005) is impossible with dynamic translations of spreadsheet formulas. An empirical evaluation of dynamic formula translations by Kankuzi and Sajaniemi (2014b, 2016) also found that they helped in reducing cognitive load of professional spreadsheet users as it helped in mapping a spreadsheet user's spreadsheet specific mental model to the corresponding problem domain mental model. The dynamic translations of spreadsheet formulas also helped spreadsheet users to statically identify more errors in spreadsheets (Kankuzi & Sajaniemi, 2014b, 2016).

5. Conclusion

We have presented a detailed description of an algorithm that can be used to dynamically translate traditional spreadsheet formulas into problem domain narratives which are easier to understand. The formula translation is based on inferred labels of referenced input cells in a given formula. The aim of the formula translations is to ease the cognitive load of the user in the spreadsheet formula comprehension process. We have also presented challenges that need to be considered when dynamically translating spreadsheet formulas to problem domain narratives. This paper has three key contributions:

i. We have given a detailed description of an algorithm that can be used to dynamically translate spreadsheet formulas to problem domain narratives right within the spreadsheet environment as the spreadsheet user accesses formula cells.

ii. We have demonstrated, through an analysis of the time and space complexity of the translation algorithm, how the number of referenced cells in a formula and the distance of labels from referenced cells will determine the speed of the translation process and consequently system responsiveness. System responsiveness is an important usability factor in interactive environments.

iii. We have demonstrated how varying spreadsheet layouts also pose a challenge to dynamic spreadsheet formula translation algorithms and have shown how we can mitigate the impact of this challenge in interactive spreadsheet visualization tools through measures such as highlighting of referenced cells.

As part of our future work, we intend to investigate on how problem domain narratives can be used to automatically detect errors in spreadsheets.

6. References

Abraham, R., Burnett, M., & Erwig, M. (2008). Spreadsheet programming. *Wiley Encyclopedia of Computer Science and Engineering*.

Abraham, R., & Erwig, M. (2004). Header and Unit Interference through Spatial Analyses. In *2004 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 165–172). Washington, DC, USA.

Apple Inc. (2017). *Numbers for Mac.* (URL:http://www.apple.com/mac/numbers/, Accessed March 2017)

Hermans, F., Pinzger, M., & Van Deursen, A. (2011). Supporting professional spreadsheet users by generating leveled dataflow diagrams. In *Proceedings of the 33rd International Conference on Software Engineering* (pp. 451–460).

Hollender, N., Hofmann, C., Deneke, M., & Schmitz, B. (2010). Integrating cognitive load theory and concepts of human–computer interaction. *Computers in Human Behavior*, *26*(6), 1278–1288.

Kankuzi, B., & Sajaniemi, J. (2013). An empirical study of spreadsheet authors' mental models in explaining and debugging tasks. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on* (pp. 15–18).

Kankuzi, B., & Sajaniemi, J. (2014a). A domain terms visualization tool for spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on* (pp. 209–210).

Kankuzi, B., & Sajaniemi, J. (2014b). Visualizing the problem domain for spreadsheet users: A mental model perspective. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on* (pp. 157–160).

Kankuzi, B., & Sajaniemi, J. (2016). A mental model perspective for tool development and paradigm shift in spreadsheets. *International Journal of Human-Computer Studies*, *86*, 149–163.

Koci, E., Thiele, M., Romero Moral, Ó., & Lehner, W. (2016). A machine learning approach for layout inference in spreadsheets. In *IC3K 2016: Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management: volume 1: KDIR* (pp. 77–88).

McKeever, R., McDaid, K., & Bishop, B. (2009). An exploratory analysis of the impact of named ranges on the debugging performance of novice users. In *Proceedings of European Spreadsheet Risks Interest Group 2009 Conference.* Paris, France.

Miller, R. B. (1968). Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, part I* (pp. 267–277).

Nardi, B. A., & Miller, J. R. (1990). *The spreadsheet interface: A basis for end user programming.* Hewlett-Packard Laboratories.

Nardi, D., & Serrecchia, G. (1994). Automatic Generation of Explanations for Spreadsheet Applications. In *Proceedings of the Tenth Conference on Artificial Intelligence for Applications* (pp. 268–274). Washington, DC, USA.

Paas, F., Renkl, A., & Sweller, J. (2004). Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional science*, *32*(1-2), 1–8.

Panko, R. R., & Ordway, N. (2005). Sarbanes-Oxley: What about all the spreadsheets? Controlling for errors and fraud in financial reporting. In *Proceedings of the European Spreadsheet Risks Interest Group 2005 Conference.* London, UK.

Powell, S. G., Baker, K. R., & Lawson, B. (2009). Errors in operational spreadsheets: A review of the state of the art. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on* (pp. 1–8).

Rajalingham, K., Chadwick, D., Knight, B., & Edwards, D. (2000). Quality control in spreadsheets: A software engineering-based approach to spreadsheet development. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on* (pp. 1–9).

Roy, S., Hermans, F., Aivaloglou, E., Winter, J., & van Deursen, A. (2016). Evaluating automatic spreadsheet metadata extraction on a large set of responses from mooc participants. In *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on* (Vol. 1, pp. 135–145).

Seffah, A., & Metzker, E. (2004). The obstacles and myths of usability and software engineering.

Communications of the ACM, *47*(12), 71–76.

Southern Cross Software. (2017). *Spreadsheet Detective.* (URL: http://www.spreadsheetdetective.com/, Accessed March 2017)

Spreadsheet Innovations Ltd. (2017). *Spreadsheet Professional.* (URL: http://www.spreadsheetinnovations.com/, Accessed March 2017)

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, *4*(4), 295–312.

Waloszek, G., & Kreichgauer, U. (2009). User-centered evaluation of the responsiveness of applications. *Human-Computer Interaction–INTERACT 2009*, 239–242.