Rule based machine translation of spreadsheet formulas to natural language expressions

B Mariro

orcid.org 0000-0002-6723-0228

Dissertation submitted in fulfilment of the requirements for the degree *Master of Science in Computer Science* at the North West University

Supervisor: Dr B Kankuzi

Examination: November 2018

Student number: 28502760

Declaration

I, BENSON MARIRO hereby declare that this dissertation report titled, "Rule based machine translation of spreadsheet formulas to natural language expressions" is my own work carried out at North-West University and has not been submitted in any form for the award of a degree to any other university or institution of tertiary education or published earlier. All the materials used as source of information have been acknowledged in the text.

Signature	Date

APPROVAL:

Signature	Date

SUPERVISOR:
DR. BENNETT KANKUZI
Department of Computer Science
North-West University
South Africa

Dedication

This dissertation is dedicated to my fiancée, Malebogo Setlanabo and son, Aidan Lefika Setlanabo for their patience and endless support during this study.

Acknowledgments

First and foremost, I would like to thank the Almighty God for giving me the strength, knowledge, ability and opportunity to undertake this research study. I also would like to thank my family with sincere gratitude for their unconditional support.
I also wish to express my sincere gratitude to my supervisor, Dr Bennett Kankuzi, for his patience, advice and guidance during this research.
Lastly, I would like to thank NWU Masters Bursary for sponsoring my research studies.

Abstract

Errors in spreadsheets are a pervasive problem both in business and other real-life settings. Most errors in spreadsheets are formula based. Spreadsheet formulas are normally specified using alphanumeric cell addresses and can only be understood if one associates referenced cells to their labels. This increases mental effort (cognitive load) in a person comprehending a spreadsheet formula and hence makes spreadsheet formula comprehension to be error prone. Translating traditional spreadsheet formulas to structured higher level problem domain oriented forms based on labels of referenced cells in a formula is one way that has been proposed to ease formula comprehension. This research work, however, provides a technique to automatically translate traditional spreadsheet formulas to natural language expressions in English using rule based machine translation. Rule-based machine translation is knowledge based machine translation that retrieves rules from bilingual dictionaries to translate source language expressions to target language expressions.

This research work has three key contributions. First, an algorithm was designed for automatically translating traditional spreadsheet formulas into natural language expressions based on devised translation rules. Second, a prototype software tool, that implemented the designed algorithm for translating traditional spreadsheet formulas into natural language expressions, was developed. Lastly, through a user study, the utility of having spreadsheet formulas in natural language expressions with respect to spreadsheet debugging, was demonstrated. In the study, it was found that natural language representation of formulas results in a non-statistically significant improvement in debugging performance in terms of percentage of errors found, with $Z = -1.414$, $p = 0.157$. However, despite this being the case, it was also found that natural language representation of formulas results in statistically significant improvement in debugging performance in terms of the speed in detecting each error as spreadsheet users took significantly less time in detecting each error than without translations, $Z = -2.521$, $p = 0.012$. The mean time for locating each error with translations was 97.1 seconds per error while without translations was 201.7 seconds per error.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

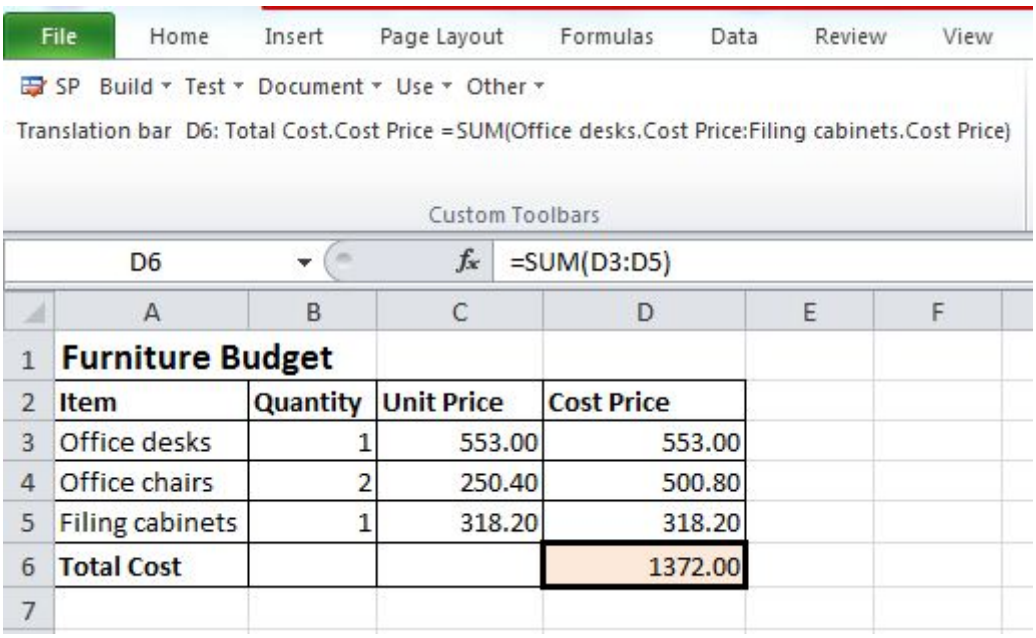
1.1 Background

Spreadsheets are an important tool used in decision making and problem solving [4]. Many companies rely on spreadsheets as a key tool in their financial reporting and operational processes [5]. As a result, the use of spreadsheets is an integral part of the information and decision-making framework for these companies. Several studies have also demonstrated that spreadsheets are extensively used to make decisions in many professional fields such as business, education, engineering and science [6,7]. Spreadsheets are used in quantitative and statistical analysis of data from databases with hundreds of thousands of records as spreadsheet offer user-friendly front ends [8]. Because spreadsheets are easy to learn and capable of sophisticated analysis, they have been accepted by users spanning a broad continuum from beginner to expert [8].

The flexibility of spreadsheets also allow them to be used without great discipline [8]. Due to poor design practices, errors are therefore easily introduced and these errors are not easily detected [8]. Although many decisions are based on the analysis of spreadsheet models, many spreadsheets have data quality problems, i.e. underlying formulas and resulting numbers are frequently wrong [4]. A growing body of empirical evidence indicates that these errors are a pervasive problem both in business and other real life settings [9]. However, spreadsheet errors are not trivial and can be costly to organizations that use spreadsheets [10].

Research has also shown that most errors in spreadsheets are mostly formula based [11]. Spreadsheet formulas are however normally specified using alpha numeric references such as “=SUM(A1:A4)”. Referenced cells in spreadsheet formulas have column and row labels. The column and row labels are then used by spreadsheet users to understand the meaning of the formula in relation to the problem

being solved in the spreadsheet. However, mapping of cell and range references into problem domain interpretations is difficult as it increases cognitive load in comprehending a spreadsheet formula [12, 13]. Cognitive load refers to the total amount of mental effort one uses in the working memory to solve an intellectual problem [14]. As cognitive load increases, so are the chances of committing errors when solving a problem [15]. Therefore, it is important to reduce cognitive load when comprehending spreadsheet formulas. It is better to create an environment which allows the user to focus on the problem domain rather than memorizing spreadsheet formula mappings, considering that a single spreadsheet can have more than one formula [16]. Several researchers have therefore proposed translating spreadsheet formulas to structured higher level expressions based on labels of referenced cells in a formula [2, 16–18]. Some commercial spreadsheet debugging tools have also provided formula translations [1, 2]. For example in a sample spreadsheet given in Figure 1.1, Spreadsheet Professional, a commercial spreadsheet debugging tool by Spreadsheet Innovations Ltd [2], translates the formula "=SUM(D3:D5)" in cell D6 to "SUM(Office desks.Cost Price:Filing cabinets.Cost Price)".



	A	B	C	D	E	F
1	**Furniture Budget**					
2	**Item**	**Quantity**	**Unit Price**	**Cost Price**		
3	Office desks	1	553.00	553.00		
4	Office chairs	2	250.40	500.80		
5	Filing cabinets	1	318.20	318.20		
6	**Total Cost**			1372.00		
7						

Figure 1.1: A Spreadsheet Professional [2] translation of formula in cell D6 displayed in the translation bar.

Based on [2], the formula would have been translated as "SUM(Cost Price | Office desks ... Cost Price | Filing cabinets)". In [18], the formula would have been translated as "SUM (Cost Price Office desks : Cost Price Filing cabinets)". The translations, however, can be distinguished through the notation and location of display of the translated formulas.

[1] http://www.spreadsheetdetective.com/

[2] https://www.auditexcel.co.za/download/spreadsheet-testing-tool/

1.2 Problem statement

Spreadsheet formulas are normally specified using A1 references (alphanumeric cell addresses) such as in the formula "=SUM(D3:D5)" where D3, D4 and D5 are referenced cells. Spreadsheet formulas can be understood only if one associates the referenced cells to their labels. This increases mental effort (cognitive load) in a person comprehending a spreadsheet formula [12, 19] and hence makes spreadsheet formula comprehension to be error prone [15]. Research has also shown that most errors in spreadsheets are formula based [11]. Several researchers have therefore proposed translating traditional spreadsheet formulas to higher level problem domain oriented forms based on labels of referenced cells in a formula [2, 17, 18]. Several commercial tools such as Spreadsheet Detective [1] and Spreadsheet Professional [2] also offer formula translations. High level formula translations have also been found to reduce mental effort of spreadsheet users in interpreting formulas and have also been found to be helpful in debugging spreadsheets [19]. Nevertheless, all the high-level formula translations from different researchers and commercial debugging tools, are not in pure natural language expressions. Human beings, however, communicate through natural languages such as English. A natural language is defined as a human language used as means of communication [20]. This research work, therefore, attempted to translate traditional spreadsheet formulas to natural language expressions using rule based machine translation. For example, in a sample spreadsheet given in Figure 1.1, the traditional spreadsheet formula "=SUM(D3:D5)" in cell D6 can be automatically translated to a natural language expression such as "The sum of cost price for office desks to cost price for filing cabinets". Rule-based machine translation is knowledge based machine translation that retrieves rules from bilingual dictionaries to translate source language sentences to target language sentences [21]. The translation requires the use of syntactic, morphological and semantic regularities of each language.

In this research work, source language sentences are the traditional spreadsheet formulas defined using the syntax (grammar) of a spreadsheet language such as Microsoft Excel. The target language sentences are the natural language expressions which conform to the syntax and morphology of the English language. The resulting natural language expressions also need to match with the intended semantics of the spreadsheet formula.

[2] https://www.auditexcel.co.za/download/spreadsheet-testing-tool/

[1] http://www.spreadsheetdetective.com/

[2] https://www.auditexcel.co.za/download/spreadsheet-testing-tool/

1.3 Research aim and objectives

1.3.1.1 Aim

The aim of this research work was to translate traditional spreadsheet formulas into natural language expressions using rule based machine translation.

1.3.1.2 Objectives

The objectives of this research work were to:

a) Design a rule-based machine translation algorithm that can be used to translate spreadsheet formulas to natural language expressions.

b) Develop a software tool that automatically translates spreadsheet formulas to natural language expressions using an identified rule-based machine translation algorithm.

c) Evaluate empirically the resulting natural language expressions, in particular, on their effect on how they can help spreadsheet users to debug their spreadsheets.

1.3.1.3 Research questions

For the identified objectives, this research work had the following research questions;

a) **RQ1**: What rule based machine translation algorithm can be used to translate spreadsheet formulas to natural language expressions?

b) **RQ2**: How can we automatically translate spreadsheet formulas to natural language expressions using a given rule based machine translation algorithm?

c) **RQ3**: What is the effect of natural language expressions on spreadsheet users when debugging or locating errors in spreadsheets?

1.4 Significance of study

Spreadsheets are widely used in most organisations and by individuals in decision making and calculation-based problems, although they are error-prone [6]. Many companies use spreadsheets in their day to day operations as a key tool in their financial reporting and operational processes [5]. Thus, the use of spreadsheets is an integral part of the information and decision-making framework for these companies. Several studies have also demonstrated that spreadsheets are extensively used to make

decisions in many professional fields such as business, education, engineering and science [6]. This has resulted in some organisations realizing big financial losses due to errors generated in spreadsheets [6, 11, 22, 23]. In other researches, in [18] spreadsheet formulas were translated into static context data flow diagrams to make it easier for spreadsheet users to understand the flow of data in spreadsheet formulas, however, the translations were not pure natural language expressions. In [2] also translated spreadsheet formulas into "problem domain narratives" using an interactive visualization tool. The translations helped to reduce the cognitive load in spreadsheet users when mapping the users' mental models to their corresponding mental model of the problem domain. However, the translations were not pure natural language expressions as was the case in this study. This research work, therefore, addressed this problem by presenting spreadsheet formulas in a form spreadsheet users can easily understand (natural language expressions), hence helped them in locating spreadsheet errors.

1.5 Structure of dissertation

The dissertation consists of six chapters as follows: Chapter 1 introduces the research problem. A review of related literature is presented in Chapter 2. The chosen research methodology is specified in Chapter 3 while in Chapter 4, the designed algorithm and its implementation are presented. The evaluation of the proposed solution is presented in Chapter 5. Lastly, the conclusion of the dissertation is presented in Chapter 6.

1.6 Conclusion

In this chapter, the research problem was presented. The specific objectives of this research work have also been presented. In the next chapter, a review of related literature is presented.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents related literature relevant to this research. An in-depth analysis on current thinking and research on machine translation, spreadsheet formula translation and spreadsheet errors is provided.

2.2 Machine translation

Machine translation is an automatic way of translating one language to another language using computer software [21]. Many approaches have been used to develop machine translation systems. There are two main categories for machine translation: rule-based machine translation and corpus-based approach [24, 25]. In rule-based machine translation (RBMT), human experts specify a set of rules to describe the translation process. Under the corpus-based approach, knowledge is automatically extracted by analysing translation examples from a parallel corpus built by human experts. Combining features of these two major classifications of machine translation systems gave birth to the Hybrid Machine Translation Approach. This research work used a rule-based machine translation approach in order to translate spreadsheet formulas to natural language expressions.

Table 2.1: Sample machine translation approaches [3]

Rule-based machine translation	**corpus-based machine translation**
Direct approach	statistical approach
Transfer based approach	Exampled based approach
Interlingual approach	

2.2.1 Rule-based machine translation

Rule-based machine translation involves using a set of rules to translate source language to target language [21]. The source language is mapped to a set of rules to identify the corresponding target language matches [21]. Rule-based machine translation can be implemented through three approaches: direct machine translation, interlingua machine translation and transfer machine translation.

2.2.1.1 Direct machine translation

Direct machine translation involves use of, syntactical processing or morphological analysis, a bilingual dictionary and local reordering rules to produce meaningful target language expressions [21]. The approach is illustrated in Figure 2.1

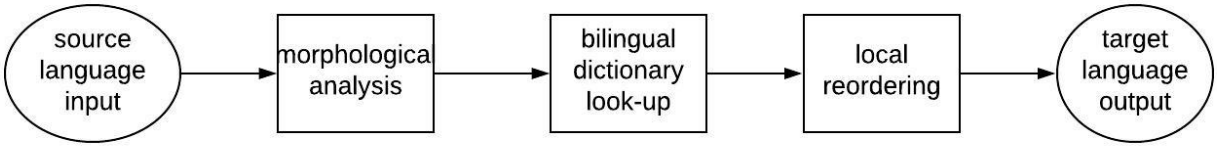


Figure 2.1: Stages of direct machine translation [1].

Direct machine translation has disadvantages in that it involves word for word translation and the coding is difficulty [1]. It is also not suitable for multilingual translations. The approach, however, can take advantage of similarities between languages [1].

2.2.1.2 Interlingua machine translation

The approach involves two stages. In the first stage, the source language is processed into an intermediate representation called interlingua [1]. In the second stage, the interlingua is then translated into the target language. The approach has a computational advantage compared to direct translation in multilingual translations, since the interlingua is independent of any language and can be translated to any target language [21]. The approach can be illustrated as shown in Figure 2.2

The main challenge to this approach is the difficulty in creating the intermediate language (interlingua) [1], as the interlingua has to be totally independent of any language and should be able to be translated to any target language. Another disadvantage of this approach, is that it does not utilize the advantage of similarities between languages [1].

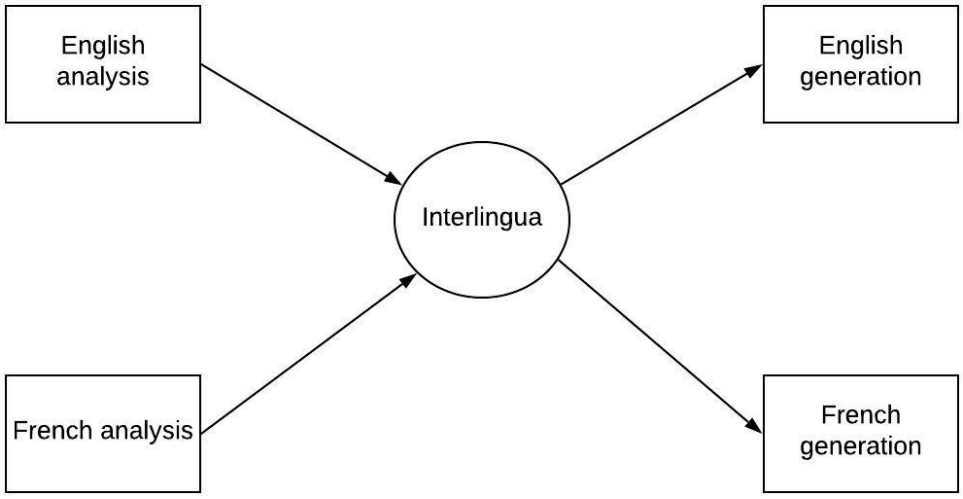


Figure 2.2: Stages of Interlingua machine translation [1].

2.2.1.3 Transfer machine translation

This approach comprises of three stages; analysis, transfer and generation [21]. In analysis, the source language expression is processed into its syntactic representation. In transfer the resulting syntactic representation is translated into corresponding target language representations [21]. The last stage involves using a morphological analyzer to generate the target language expressions [21]. The stages of transfer machine translation are illustrated in Figure 2.3.

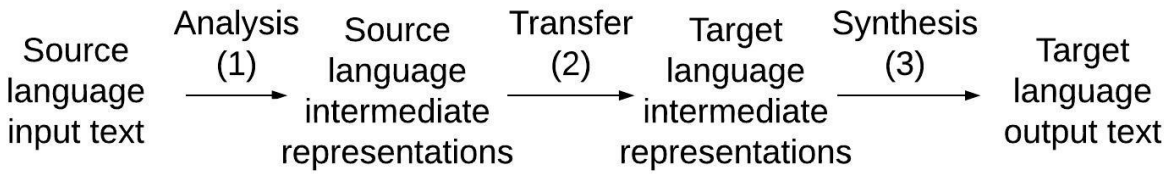


Figure 2.3: Stages of transfer machine translation [1].

Transfer-based machine translation has a challenge in multilingual translations since the intermediate language representation is language dependent unlike the intermediate in the interlingua translation approach [1]. However, the intermediate representation in the transfer approach is easier to implement compared to the intermediate from interlingua approach.

2.2.2 Corpus-based machine translation

Corpus-based machine translation approaches rely on large parallel bilingual or multilingual corpora [21]. Statistical machine translation and example based machine translation are the two main cate-

gories of corpus-based machine translation [21]. A large parallel corpus consisting sample bilingual or multilingual translations is used by the approaches to determine appropriate current translation.

2.2.2.1 Statistical machine translation

Statistical methods are applied to carryout bilingual or multilingual translations based on parallel corpora [21]. Statistical tables from the parallel corpora are generated, which are used to determine appropriate translation of the source language based on characteristics of well-formed sentences and correlation between the source and target languages [21]. Statistical machine translation can be implemented based on three categories; word, phrase and hierarchical phrase based translations [21].

2.2.2.2 Word based translation

Extracted words from the source language sentences are translated individually to their equivalent target language words based on statistical outcomes from the parallel bilingual corpus [21]. The translated words are then sorted to obtain target language sentences [21]. A disadvantage of this approach is low performance when dealing with large documents translations [21].

2.2.2.3 Phrase based translation

Phrases from source language sentences are translated to their equivalent target language phrases unlike word based translations [21]. Although the approach is better in performance than the word based translation but there is a challenge in word ordering in complex word ordering translations [21].

2.2.2.4 Hierarchical phrase based translation

This approach uses phrases that contain sub-phrases, which allow learning of reordering, translations of multi-word expression or insertions and deletions sensitive to the current translations [26]. The advantages of this approach is better reordering and better performance due to recursive nature of hierarchical phrases [21, 26].

2.2.3 Example-based machine translation

Translation process is by analogy through the use of a bilingual corpus as the knowledge base. The translation uses a collection of similar source and target language translations to translate the current source language into its target language equivalent [21]. The concept of this approach is that similar translations are more likely to reoccur in future [21]. Its major benefit is that it does not need very large parallel corpus for the translation process [21].

In this research work, we have used the direct rule-based machine translation approach as it is simpler to implement considering the time frame of the research.

2.3 Translations of spreadsheet formulas to natural language expressions

In [17], a tool was created for automatic generation of explanations in spreadsheets. The tool was developed to make it easier for end-users to comprehend spreadsheet formulas. The approach was based on the construction of a knowledge base containing information on the mathematical relations from a spreadsheet and the generation of explanations concerning the quantities used in spreadsheet and their relations.

D2 f_x =(B2*C2)

	A	B	C	D	E
1	**Client Name**	**Credit Amount**	**Interest**	**Interest Amount**	
2	John Smith	5000.00	20.00%	1000.00	
3	Peter Crouch	3000.00	20.00%	600.00	
4					
5					
6					
7					

Figure 2.4: A sample spreadsheet for calculating Credit interest.

In generating explanations, the spreadsheet formula "=B2*C2" in D2 as shown in Figure 2.4 would be translated to "John Smith (Credit Amount) * John Smith (Interest)". Although the approach successfully generated explanations from spreadsheet formulas but the explanations are not full translation of spreadsheet formulas into pure natural language expressions [16], since human beings communicate better in their natural language [20].

In [18], formulas are translated in a static context involving automatic extraction of leveled dataflow diagrams from Microsoft Excel spreadsheets. The aim of the tool was to also make it easier for spreadsheet users to understand how data flows in spreadsheets. Cell references in a data flow diagram are replaced with names obtained from labels of the referenced cells as illustrated in Figure 2.5. The resulting translations, however, are not pure natural language expressions which will make it easier for spreadsheet users to comprehend spreadsheet formulas.

In [2] and [27], an interactive spreadsheet visualization tool which translates spreadsheet formulas into what are termed as "problem domain narratives" is presented. The tool extracts column and row

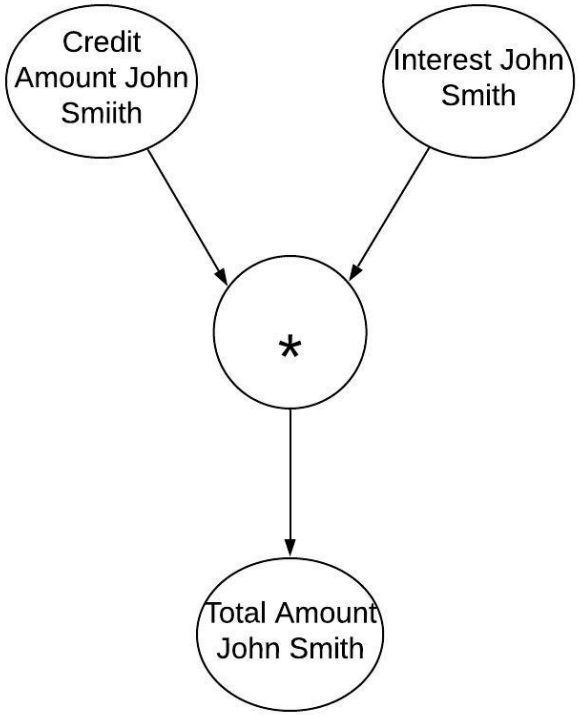


Figure 2.5: A corresponding leveled dataflow diagram for the formula "=B2*C2" in cell D2 of the spreadsheet given in Figure 2.4.

labels for a particular referenced input cell in a formula to form a symbolic name for that cell. For example in Figure 2.6, the spreadsheet formula in cell C9 is translated as "SUM(Jan | James Bourne ... Jan | Jasmine Hunt)"

The tool also highlights all referenced cells in an active formula cell and marks each formula cell with a pink right border [2]. The tool proved to be effective in reducing the cognitive load in spreadsheet users as it helped in mapping a user's spreadsheet mental model to their corresponding mental model of the problem domain [19]. The translated spreadsheet formulas were also found to be learnable and helped spreadsheet users to locate more errors in their spreadsheets. The translations, however, were also not pure natural language expressions which will make it easier for spreadsheet users to comprehend spreadsheet formulas. In this research work, we have adapted and extended this tool in the implementation of translation of spreadsheet formulas to natural language expressions based on translation rules.

In another research, [23], introduced the named ranges approach which involved naming a cell or a range of cells in a spreadsheet formula. The naming of a cell or range of cells followed the same approach used in software programming when naming variables. The name used for naming a cell or range of cells has to be descriptive of its purpose. The approach, when implemented to sample formula "D3=B3-C3", would generate the formula "netSalary = gross - tax" with named cells.The

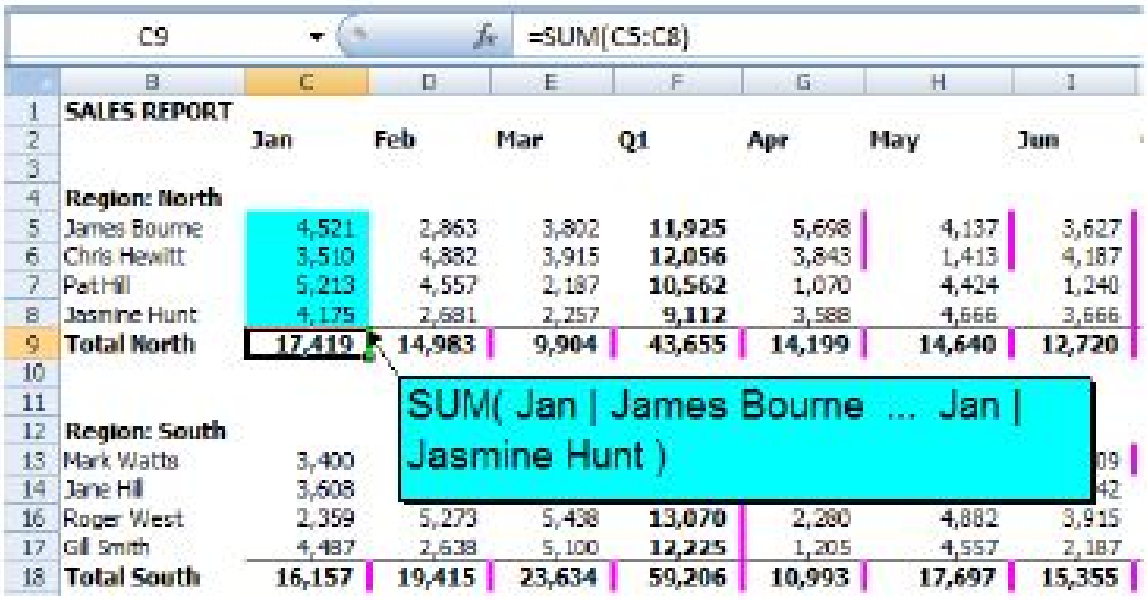


Figure 2.6: Formula in cell C9 with a corresponding translation displayed below it [2].

approach proved to be effective in helping spreadsheet users in understanding spreadsheet formulas. However, the approach does not adapt well with complex spreadsheet formulas and does not transform operators to expressions for spreadsheet users to comprehend spreadsheet formulas easily, therefore prompting for the need of spreadsheet formulas translation into pure natural language expressions.

2.4 Errors in spreadsheets

Spreadsheet usage has grown tremendously in organizations [28]. However, research has shown that most spreadsheets contain non-trivial errors [9]. Errors in spreadsheets have dire consequences for organizations since they are used in decision making, budgeting, daily operations, etc [9]. In some instances, they have cost organizations millions of dollars [10].

Errors in spreadsheets occur due to many factors. Spreadsheet users experience high cognitive load when apprehending referenced cells of a spreadsheet formula meaning in the context of problem domain [19]. This task involves constant back and forth mapping of the spreadsheet user's mental model and the spreadsheet [19]. This constant mapping thus increases the cognitive load of spreadsheet users [19]. The increased cognitive load results in spreadsheet users making errors in spreadsheets. Most spreadsheet users lack training and tend not to follow any standard in the process of developing spreadsheets, even though they use spreadsheets daily. Thus, most spreadsheets end up with high rate of errors since their development process tend not to follow formal development methodologies [29]. Research has also shown that, even though spreadsheet errors are prevalent,

most companies or organizations which use spreadsheet on daily basis, have not implemented strong policies for spreadsheet development and testing to reduce errors [9]. The "ease of use" nature of spreadsheets tends to tempt end users into developing spreadsheet programs without training and following formal development methodologies. Many spreadsheet users also lack appreciation of the risks posed in casual programming approach [22].

2.4.1 Classification of spreadsheet errors

Spreadsheet errors can be classified into two main categories: quantitative and qualitative errors [9]. Quantitative errors generate incorrect bottom-line values in spreadsheets [9]. Qualitative errors arise from poor design layout of spreadsheet programs which may later lead to qualitative errors [9]. Quantitative errors can further be divided into mechanical, omission and logic errors. Mechanical errors are caused by mistyping a value or referencing wrong cell range in a formula. Typing a wrong operation in a formula is also classified as mechanical error [9]. Mechanical errors are due to mistakes or slips when typing a formula or entering a values in spreadsheets. Omission errors occur when a spreadsheet user omits a factor or fact in the solution model of spreadsheet program due to misinterpretation of the situation [9]. Logical errors occur when a spreadsheet users designs a completely wrong algorithm in a formula [9].

Qualitative errors include formatting errors, update errors, hard-coding errors and semantic errors [28].

2.4.2 Techniques for detecting spreadsheet errors

Error detection is the process of finding errors for correction in a program. This involves identifying defects, errors, and deficiencies [30]. It is a significant stage in program development to determine the capabilities, limitations, and quality of a program [30]. Error detection is also used in spreadsheets to identify errors and make appropriate corrections to improve the quality of spreadsheets. In spreadsheets, error detection is also referred to as spreadsheet auditing [31]. Error detection in spreadsheets can be done manually or using automated approaches.

2.4.2.1 Manual error detection techniques in spreadsheets

These techniques involve spreadsheet developers or peer reviewers identifying errors in a spreadsheet program without using any tool. The process of identifying errors may be done during or after spreadsheet development. Experienced spreadsheet users have been found to be more efficient and effective at spreadsheet debugging than novices [32]. Some researchers have also indicated the need

for group work in auditing spreadsheets as it improves the rate of identifying errors in group work than as individuals [31].

2.4.2.2 Review during spreadsheet construction

This involves spreadsheet users or developers inspecting spreadsheet formulas during development. This allows early detection of errors before they cascade through the spreadsheet [9, 33]. During the construction of a spreadsheet it is natural to check for errors. Every input value and formula should be checked for the presence of errors in spreadsheets [34]. This can be achieved through designed test cases that can reveal errors on spreadsheet formulas.

2.4.2.3 Usage of cross-checks and validation points

Spreadsheet users may save time and effort of reviewing a spreadsheet by carrying out cross-checks of bottom line values for correctness. By summing up values across rows and columns, using alternative calculation methods to the same problem and so on [35, 36].

2.4.2.4 Manual testing of spreadsheets

This process goes beyond just reviewing spreadsheet models, it involves the running of models and examining the results [9, 35, 36]. Test cases are design including input of expected ranges and boundary conditions, which will be used to verify the model and examine the results against expected outcome. Testing is intended to build confidence in the spreadsheet to interested parties.

2.4.2.5 Inspection of all formulas

Spreadsheet users or peers reviewers may carry out a comprehensive examination of all inputs and formulas cell by cell in a spreadsheet [33, 35]. Although, this process can be effective in small spreadsheet, but it is time-consuming and tedious in very large and complex spreadsheets. The reviewers may become less thorough before the task is completed, thus leading to the process being ineffective in large spreadsheets.

2.4.2.6 Automatic error detection

Automatic error detection involves the use of software tools to automatically find errors for correction [5, 7, 37]. Spreadsheet users or reviewers may engage software tools to detect errors as spreadsheets may become large and complex. Manual error detection in large and complex spreadsheets is tiresome and less thorough, thereby reducing chances of identifying more errors, to avoid these

challenges users engage software tools to automate the detection process [35]. Several auditing tools have been developed to automate error detection process and have proved to be useful [35]. Some of the commercial auditing tools include:

a) Spreadsheet Detective is another commercial tool used to identify errors for correction in spreadsheets. It provides automated documentation that highlights mistakes [1]. The tool creates graphical annotations to help the spreadsheet user to visualize the layout of the spreadsheet [1].

b) Spreadsheet Professional is a commercially available tool which provide a set of functionalities such as building, testing, analyzing and comparing spreadsheets [38]. The tool provides a formal testing process which is more thorough and takes less time by up to 80% than manual testing [2]. The building functionality is used for creating accurate spreadsheets quickly and the analyzing part is used for generating spreadsheet documentation [2]. The tool can also compare 2 spreadsheets from one version to another to identify changes using the analyzing module.

c) Operis Analysis Kit is similar to Spreadsheet Professional. The tool provides functionalities for generating reports, visualization of formula models, and comparing of spreadsheets from one version to another for changes [39]. The tool also allows searching of particular cells with specific attributes [39].

d) Microsoft Excel built-in tool also provide auditing functions. The tool provides standard functions for testing the spreadsheet for errors [40]. The functions can perform cell by cell inspection and highlight cells with errors [40].

There are also non-commercial debugging tools and techniques such as:

a) In [41], a constraint-based debugging approach was developed. The approach converts a faulty spreadsheet and test cases into a constraint satisfaction problem in order to diagnose faulty cells in the spreadsheet.

b) CACheck from [42], an approach which detects and repairs cell arrays with distortion or ambiguity in data or formula meaning. A cell array, in this case, refers to cell range with same computational semantics. The tool checks for inconsistencies in formula patterns of cells with formulas and generates new patterns to repair the formulas that are distorted or ambiguous.

c) In [43], a tool that automatically detects potentially erroneous cells (smells) in spreadsheets is presented. The tool uses a two-stage technique to automatically cluster cells based on their features.

[1] http://www.spreadsheetdetective.com/

[2] https://www.auditexcel.co.za/download/spreadsheet-testing-tool/

d) In [44], a tool referred to as Melford uses neural networks to locate errors in spreadsheets. The tool predicts whether a cell should contain a formula if a spreadsheet user had erroneously entered a number instead of a formula.

e) ExceLint from [45], is an approach which uses static analysis to find errors in spreadsheet formulas by exploiting the rectangular nature of spreadsheets.

f) Smellsheet detective was used to detect error smells in spreadsheets [5]. Error smells are characteristics that an error might occur in a spreadsheet.

2.5 Conclusion

This chapter provided an analysis of different machine translation and also considered their strengths and weaknesses. A review of spreadsheet errors and current state of the art in automatic spreadsheet debugging has also been presented. The research methodology used in this work is presented in the next chapter.

Chapter 3

Methodology

3.1 Introduction

This chapter focuses on the research methodology used in this research work. Constructive research methodology [46] is the approach in this work due to the nature of our research. This research work sought to find a solution to a practical and relevant problem.

3.2 Constructive research methodology

The constructive research methodology follows an idealized model which comprises the following steps:

i) finding a practically relevant problem that has research potential.

ii) obtaining a general, comprehensive understanding of the topic.

iii) designing a new construct.

iv) demonstrating that the new construct (solution) works.

v) showing the theoretical connections and the research contribution of the solution concept.

vi) examining the scope of applicability and generalizability of the solution.

3.2.1 Finding a practically relevant problem

The research problem in this research work was initially identified through a literature review about errors in spreadsheets and how others have tried to solve this problem. A detailed description of the research problem is presented in Chapter 1.

3.2.2 Obtaining comprehensive understanding of topic

Extensive literature review was conducted to obtain comprehensive understanding about errors in spreadsheets and existing solutions to the problem. An extensive review of machine translation techniques was also conducted. This is presented in Chapter 2.

3.2.3 Designing a new construct

This involves designing a solution to the identified problem. In this research work, an algorithm to translate traditional spreadsheet formulas to natural language expressions was developed. A corresponding software tool, that implements the algorithm, was also developed. This is presented in Chapter 4.

3.2.4 Demonstrating that the solution works

The developed software tool that automatically translates spreadsheet formulas to natural language expressions was evaluated in a user study. The user study involved selected professional Accountant participants who were to evaluate two real-life non-trivial Microsoft Office Excel spreadsheets sourced from EUSES spreadsheet corpus [47]. The participants were selected using the purposeful sampling technique. This technique was relevant for this study since the evaluation required homogeneous sample population and familiarity in spreadsheets [48]. Participants from the accounting field proved to be ideal for this research work as they use spreadsheets in their daily operations. In the evaluation study both spreadsheet were seeded with real life errors and both spreadsheets were translated in separate occasions. This was to gauge the effectiveness of having formulas presented as natural language expressions in spreadsheet debugging tasks. The evaluation is presented in Chapter 5.

3.2.5 Theoretical connections and research contribution

The contributions of this research work are presented in Chapter 6.

3.2.6 Examining the scope of applicability and generalizability

The applicability of findings from this research work are presented in Chapter 6.

In the context of this research work, the constructive research methodology is summarized in a process flow diagram as illustrated in Figure 3.1.

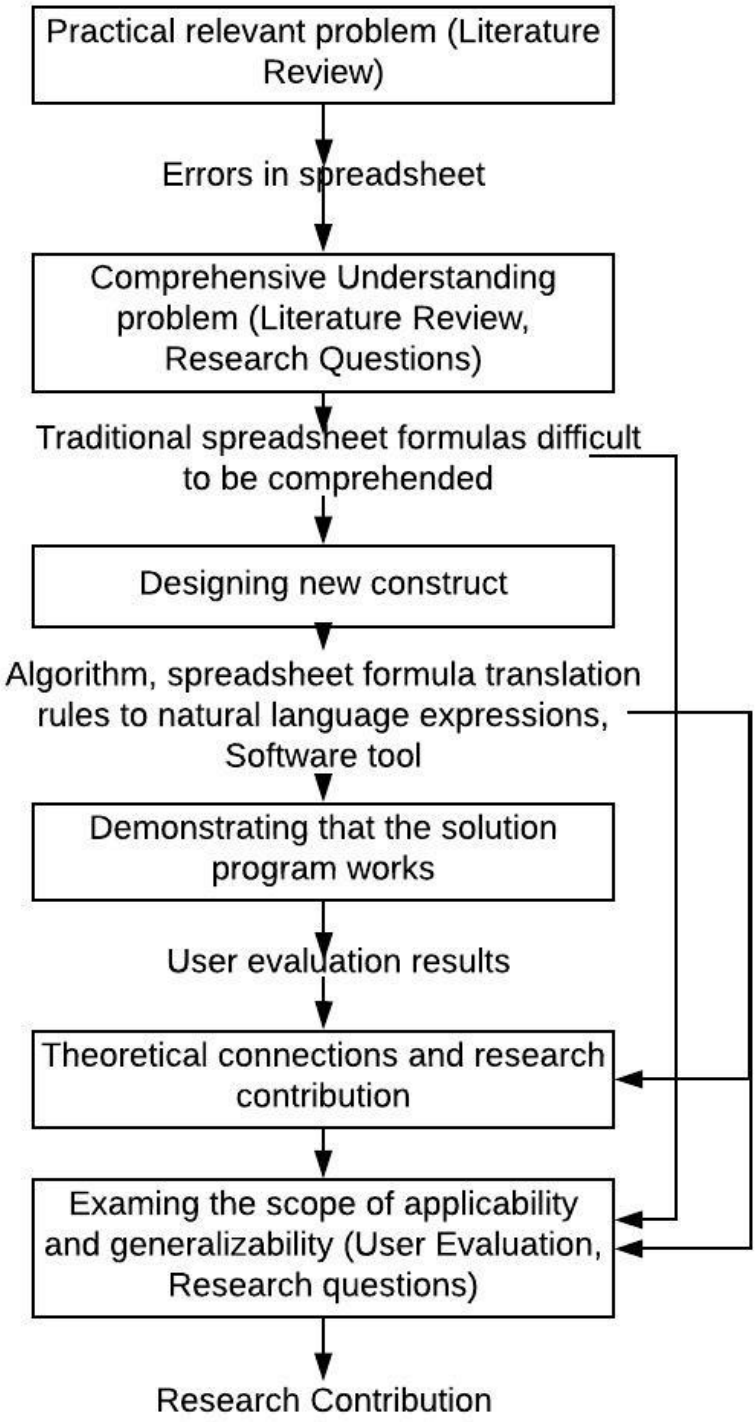


Figure 3.1: Stages of constructive research methodology as used in the context of this research work.

3.3 Justification of the research methodology

The constructive research methodology was deemed suitable for this research as there was a need to solve a relevant practical problem. The solution was intended to make it easier for spreadsheet users to comprehend formulas as most spreadsheet errors are formula based. Unlike other methodologies, constructive research methodology provided general steps through which this research work could be conducted.

3.4 Conclusion

This chapter presented a description of the constructive research methodology and how it has been used in the context of this research work. Constructive research methodology was chosen as it was deemed to be an effective and relevant approach to achieve the research objectives of this research work. In the next Chapter, the algorithm design and its implementation, as a solution to the research problem, is presented.

Chapter 4

Algorithm Design and Implementation

4.1 Introduction

This chapter presents the design and implementation of algorithms used to translate traditional spreadsheet formulas into natural language expressions in order to answer two research questions:

RQ1: *What rule based machine translation algorithm can be used to translate spreadsheet formulas to natural language expressions?*

RQ2: *How can we automatically translate spreadsheet formulas to natural language expressions using a given rule based machine translation algorithm?*

4.2 Algorithm design

The translation process is based on a rule based machine translation approach, which requires the generation of a bilingual dictionary (rules tables) which is used to map source language expressions to their corresponding target language expressions and reordering of the resulting translated expressions to form meaningful target language expressions.

The translation process consists of the following two steps:

a) Transforming a traditional spreadsheet formula into an intermediate form called a problem domain narrative.

b) Transforming a problem domain narrative to a natural language expression (NLE) in English by applying rules from a bilingual dictionary (rules tables).

The translation process steps are illustrated in Figure 4.1

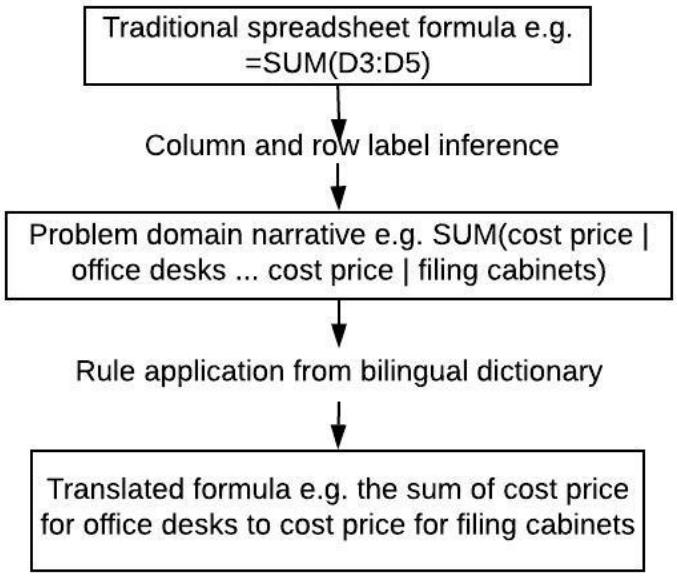


Figure 4.1: Process flow diagram illustrating the translation steps.

4.2.1 Factors affecting algorithm design

The translation process takes into consideration of the grammatical rules of the English language, which is the natural language used in this research work. These rules impact the approach and structure of the algorithm design.

4.2.1.1 Grammar for spreadsheet language

Spreadsheet application programs use formulas to compute values from specified data. Due to its popularity, Microsoft Excel was the spreadsheet application program used in this research work. A spreadsheet formula contains elements such as function calls (built-in or user-defined functions), function arguments, data, and references, which need to be specified following spreadsheet language grammatical rules. The grammatical rules specify the structural elements and their arrangement in a spreadsheet formula expression. Microsoft published grammatical rules also known as production rules for Excel formulas [49]. These rules provide guidelines on the structural arrangement of the formula elements and are published in [50]. Some of the production rules for Microsoft Excel grammar, specified using the augmented Backus–Naur (ABNF) notation, are presented in Appendix A:

In this research work, a spreadsheet formula is thus parsed for particular formula elements such as functions and translated accordingly based on identified tokens and corresponding translation rules. A formula can also be expressed as a parse tree using to easily understand the grammatical structure of the formula. For example, formulas“=D3+D4+D5”, “=SUM(D3:D5)” and

"=IF(D3 > 300, "Low", "High")" are expressed in parse tree structures, based on the grammar of Microsoft Excel, as shown in Figure 4.2, Figure 4.3 and Figure 4.4 respectively. The parse trees were generated following guidelines on the structural arrangement of the formula elements and production rules for Microsoft Excel grammar, specified using the augmented Backus-Naur (ABNF) presented in Appendix A.

Figure 4.2: Parse tree for Microsoft Excel formula "=D3+D4+D5".

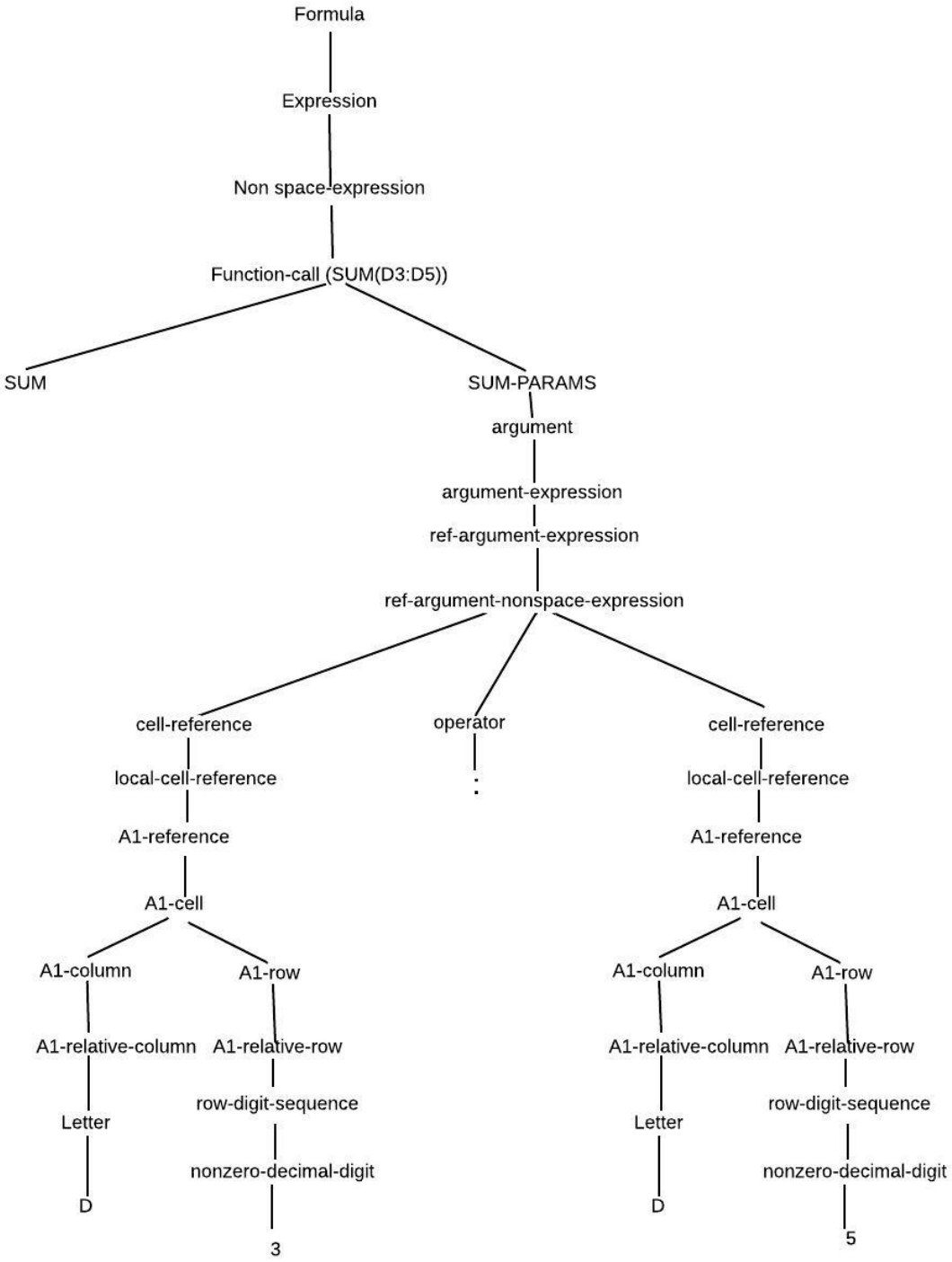


Figure 4.3: Parse tree for Microsoft Excel formula “=SUM(D3:D5)”.

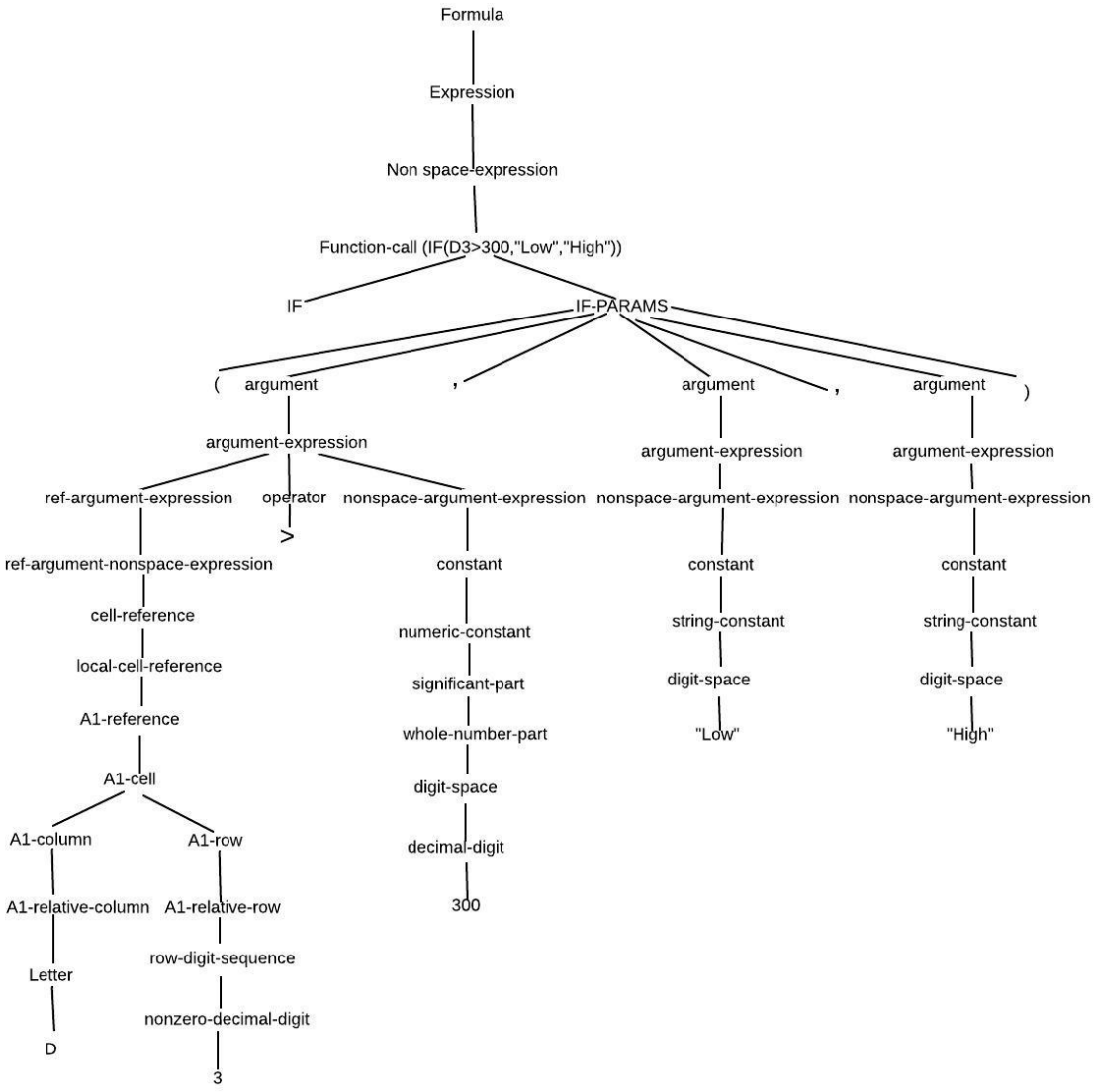


Figure 4.4: Parse tree for Microsoft Excel formula “IF(D3 > 300, “Low”, “High”)”.

4.2.1.2 Grammar for natural language (English)

English contains grammatical units such as words, phrases, clauses and sentences [51]. A word in English can be a verb, noun, adjective, adverb, preposition, determiner, pronoun and conjunction. A phrase can be verb phrase. noun phrase, adjective phrase, adverb phrase and preposition phrase. Clauses are built from phrases through the following syntax;

- Subject (noun phrase) + verb (verb phrase) + complement (noun phrase)

A clause may contain prepositional phrase as follows;

- Adverbial (prepositional phrase) + Subject (noun phrase) + verb (verb phrase) + object (noun phrase)

On the other hand, sentences are made up of elements such as subjects, verbs, objects, complements and adverbials. A sentence can also be constructed from a single clause or a combination of two or more clauses. In this research, elements of sentences are used to construct English language expressions. The elements of sentences are extracted from row and column labels of references cells from a spreadsheet formula and corresponding English language expression mappings of formula tokens from rules tables. The extracted expressions from column and row labels and corresponding mappings from rules tables form the subjects(noun phrases) or objects in the context of the English language.

4.2.2 Transforming a traditional spreadsheet formula into a problem domain narrative

The algorithm for translating spreadsheet formulas into problem domain narratives was adapted from [27]. In this research the same algorithm was followed as specified in the algorithm steps and the result was further transformed by the developed algorithm into pure natural language expressions. The algorithm for translating a spreadsheet formula into a problem domain narrative has the following steps:

i) Obtain all referenced cells in a given spreadsheet formula.

ii) Find the corresponding column label and row label for each referenced cell in the formula.

iii) Form a symbolic name for each referenced cell by joining the corresponding column label and row label while separating them by a special character "|".

iv) Replace each referenced cell in the original spreadsheet formula expression with a corresponding symbolic name.

v) The resulting formula expression is now stored as a domain narrative expression after replacing the operator ":" with " ... " if it exists in the expression.

The pseudocode for the algorithm for translating a spreadsheet formula into a problem domain narrative is presented in Algorithm 1.

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

```
1:  procedure GENERATEDOMAINNARRATIVE(cellFormula)
2:      precedentsList ← getReferencedCells(cellFormula)
3:      for each precedentCell in precedentsList do
4:          columnLabel ← getColumnLabel(precedentCell)
5:          rowLabel ← getRowLabel(precedentCell)
6:          symbolicName ← columnLabel "|" rowLabel
7:          cellFormula ← replace(cellFormula, precedentCell, symbolicName)
8:      end for
9:      narrative ← cellFormula
10:     store(narrative)
11: end procedure
```

4.2.3 Transforming a problem domain narrative to a natural language expression in English

The algorithm for transforming a problem domain narrative to a natural language expression takes the problem domain narrative from the previous algorithm and through a series of steps, transforms a given problem domain narrative into a natural language expression (English). The algorithm has the following steps:

i) Basic operators in the problem domain narrative are replaced using the mappings from the rules table for basic operators shown in sample table, Table 4.1. If there are no basic operators in the formula expression, the original problem domain narrative is not changed.

ii) All the functions in the problem domain narrative are then identified using regular expression pattern matching and then translated to natural language expressions by repeatedly following the sub-steps below until there are no untranslated functions in the problem domain narrative.

   a) All functions without nesting are identified i.e. functions which do not have any other functions inside them are identified. A string representing a function without nesting is identified through the regular expression pattern $[a - zA - Z] + \backslash([\ \hat{}\ \backslash(\backslash)] + \backslash)$.

   b) Each of the strings representing a function without nesting is then split into tokens based on commas in the string. The tokenization is based on commas because Microsoft Excel separates arguments in functions using commas.

   c) Each token is then translated by looking up the corresponding mappings in the functions rules table and extracting the corresponding equivalent natural language expressions of the token. A sample function rules table is presented in Table 4.2. The equivalent expression for the token is then used to replace the token expression in the original untokenized string representing the function without nesting. The original untokenized string representing the function without nesting is fully translated after all the token expressions in the string are translated. To avoid confusion in the further tokenization processes, the commas in the original untokenized string are temporarily replaced with special character " ~ ".

   d) The fully translated string representing the function without nesting is then used to replace the original expression in the problem domain narrative expression.

iii) The sub-steps, above, are repeated until there are no untranslated functions in the string representing the problem domain narrative.

iv) The special character " ~ " is replaced with the original commas in the fully translated problem domain narrative.

v) If necessary, special English grammatical rules are then applied to get a fully translated natural language expression in English.

The pseudocode for the algorithm for translating a problem domain narrative to natural language expression (English) is given in Algorithm 2.

Algorithm 2 Algorithm for translating a problem domain narrative to a natural language expression (English)

procedure TRANSLATETONLE($narrative$)
  for each $operator$ in $narrative$ **do**
    $narrative \leftarrow$ replace($narrative, operator, operatorTranslationRules$)
  end for
  $regexPattern \leftarrow [a-zA-Z] + \backslash([\hat{}\backslash(\backslash)] + \backslash)$
  while hasPattern($narrative, regexPattern$) **do**
    $matchList \leftarrow$ getFunctionsWithoutNesting($narrative, regexPattern$)
    for each $match$ in $matchList$ **do**
      $tokenList \leftarrow$ tokenize($match$,“,”)
      $translatedMatch \leftarrow match$
      for each $token$ in $tokenList$ **do**
        $tokenTranslation \leftarrow$ translate($token, functionTranslationRules$)
        $translatedMatch \leftarrow$ replace($translatedMatch, token, tokenTranslation$)
      end for
      $translatedMatch \leftarrow$ replace($translatedMatch$,“,”,“ ~ ”)
      $narrative \leftarrow$ replace($narrative, match, translatedMatch$)
    end for
  end while
  $narrative \leftarrow$ replace($narrative$,“ ~ ”,“,”)
  $narrative \leftarrow$applySpecialEnglishGrammarAdjustments($narrative$)
  $fullyTranslatedExpression \leftarrow narrative$
end procedure

The algorithm for transforming a problem domain narrative to natural language expressions uses a bilingual dictionary or rules tables to translate spreadsheet formulas into natural language expressions. The purpose of the bilingual dictionary is to provide rules for translating spreadsheet formulas into natural language expressions.

4.2.3.1 Generation of bilingual dictionary

The bilingual dictionary is presented using the table format where each row contains a pair of equivalent source language expression and target language expression. In this research, three tables were designed and are referred to as rule tables. The first rules table is for spreadsheet basic operators. The table contains operator symbols and their corresponding natural language expressions. The table also includes translations of two special operators "|" and "..." which are specially used in domain narratives. Table 4.1 illustrates the sample rules table fo spreadsheet basic operators;

Table 4.1: Sample rules table for basic operators

Rule No.	**Operator Symbol**	**Natural language Expression**
1.	*	multiply by
2.	+	plus
3.	-	minus
4.	|	for
5.	...	to
6.	>	is greater than
7.	>=	is greater than or equal to
8.	<	is less than
9.	<=	is less than or equal to
10.	<>	is not equal to

The second table in the bilingual dictionary is the function rules table. This table contains mappings of tokens of spreadsheet function expressions and their corresponding English expressions. The table has five columns; rule number, function expression, expression type, token and natural language expression. Sample table, Table 4.2 shows the mappings of tokens of spreadsheet function expressions into natural language expressions of three common spreadsheet functions used in formulas [52].The function expression column contains the general forms of a given function expression. The expression type column shows the classification of each function expression into complex form or simplest form. A function expression is in complex form if, at least, one optional argument is specified. For

example, “SUM(D3,D4)” is in complex form because D4 is an optional argument according to the general syntax of the SUM function, “SUM(number1, [number2], [number3], ...)”. A function expression is in simplest form if it has only the minimal required arguments. For example, the function “SUM(D3)” is in the simplest form of the SUM function because it has no optional arguments and thus takes the simplest general form “SUM(number1)”. The third column presents the general form of the tokens extracted from the function expression. The tokenization is carried out using the comma as a delimiter to extract the required tokens. The natural language expression column provides the general equivalent translation of each token in the natural language form. To get a translation for a given token, the function expression and its type also need to be specified. This translation technique can be applied to any Microsoft Excel function. For purposes of this write-up, translation rules of some sample functions is presented in Table 4.2. This translation technique, however, can be applied to any Microsoft Excel function. For example, in Appendix B, translation rules for the VLOOKUP function are provided.

Table 4.2: Sample rules table for functions.

Rule No.	Function Expression	Expression Type	Token	Natural language Expression
1.	SUM(number1, [number2], [number3], ...)	complex	SUM(number1	the sum of number1
2.	SUM(number1, [number2], [number3], ...)	complex	[number2]	[number2]
3.	SUM(number1, [number2], [number3], ...)	complex	[number3]	[number3]
4.	SUM(number1, [number2], [number3], ...)	complex	...)	and ...
5.	SUM(number1)	simplest	SUM(number1)	the sum of number1
6.	IF(condition, [value_if_true], [value_if_false])	complex	IF(condition	if condition
7.	IF(condition, [value_if_true], [value_if_false])	complex	[value_if_true]	[the value is value_if_true]
8.	IF(condition, [value_if_true], [value_if_false])	complex	[value_if_false]	[otherwise the value is value_if_false]
9.	IF(condition,)	simplest	IF(condition	if condition
10.	IF(condition,)	simplest	)	empty space
11.	AVERAGE(number1, [number2], [number3], ...)	complex	AVERAGE(number1	the average of number1
12.	AVERAGE(number1, [number2], [number3], ...)	complex	[number2]	[number2]
13.	AVERAGE(number1, [number2], [number3], ...)	complex	[number3]	[number3]
14.	AVERAGE(number1, [number2], [number3], ...)	complex	...)	and ...
15.	AVERAGE(number1)	simplest	AVERAGE(number1)	the average of number1

The third rules table is for special English grammatical rules. This table provides mappings for cases where there is a need to add propositions to attain meaningful natural language expressions. The table has three columns; rule number, the expression column and the replacement column. The expression column shows the extracted expression from the spreadsheet formula expression which needs to be mapped to an equivalent natural language expression. The replacement column present the equivalent mapping of the expression provided in the expression column. Table 4.3 shows the rules table for special grammatical rules. For example, a comma can not follow an "and" and therefore the comma needs to be deleted. Similarly, if a comma follows "otherwise" in an expression, we replace the comma with a semi-colon. This is particularly useful when translating IF function.

Table 4.3: Sample special English grammatical rules

Rule No.	Expression	Replacement
1.	, and	and
2.	, otherwise	; otherwise

4.3 Examples of spreadsheet formula translations

This section presents examples of translations of various spreadsheet formulas into natural language expressions using the designed algorithms.

4.3.1 Example translation of spreadsheet formula "=D3+D4+D5"

Given a formula "=D3+D4+D5" in cell D6 in the spreadsheet shown in Figure 4.5;

D6 f_x =D3+D4+D5

	A	B	C	D	E	F
1	**Furniture Budget**					
2	**Item**	**Quantity**	**Unit Price**	**Cost Price**	**Rank 1**	**Rank 2**
3	Office desks	1	553.00	553.00	High	High
4	Office chairs	2	250.40	500.80	High	High
5	Filing cabinets	1	250.00	250.00	Low	Low
6	**Total Cost**			1303.80		
7	Average			434.60		

Figure 4.5: Sample spreadsheet formula "=D3+D4+D5" for translation in cell D6.

The transformation of the formula "=D3+D4+D5" will be carried out by first implementing the

steps specified in algorithm 1 which transforms the original spreadsheet formula "=D3+D4+D5" into a problem domain narrative as shown in the following steps;

step i) Referenced cells D3, D4 and D5 are extracted from the formula "=D3+D4+D5".

step ii) Cell D3 has column label "Cost Price" and row label "Office desks ". D4 has column label "Cost Price" and row label "Office chairs ". Cell D5 has column label "Cost Price" and row label "Filing cabinets ".

step iii) Symbolic names for D3, D4 and D5 are formed by joining their corresponding column label and row label while separating them by a special character "|". D3 will have symbolic name "cost price | office desks", D4 have symbolic name "cost price | office chairs " and D5 will have "cost price | filing cabinets ".

step iv) Each referenced cell in the original spreadsheet formula "=D3+D4+D5" is replaced by the corresponding symbolic name. D3 will be replaced by "cost price | office desks", D4 by "cost price | office chairs " and D5 by "cost price | filing cabinets" to produce the expression "cost price | office desks + cost price | office chairs + cost price | filing cabinets"

step v) The expression "cost price | office desks + cost price | office chairs + cost price | filing cabinets" is now stored as a domain narrative expression.

The output of algorithm 1 in this case, the problem domain narrative is then used as input to algorithm 2 and processed as indicated in the following steps;

step i) The translation process starts by searching for basic operators in the domain narrative "cost price | office desks + cost price | office chairs + cost price | filing cabinets". Basic operators "|" and "+" will be replaced by their corresponding mapping expressions as defined in rule number 4 and 2 respectively from Table 4.1. Thus, the resulting domain narrative expression will be "cost price for office desks plus cost price for office chairs plus cost price for filing cabinets" .

step ii) A regular expression pattern is used to search for functions in the domain narrative and then translate them to natural language expressions by repeatedly following the sub-steps below until there are no untranslated functions in the problem domain narrative.

a) The regular expression pattern $[a - zA - Z] + \backslash([\ \hat{}\ \backslash(\backslash)] + \backslash)$ is used to extract the function matching the pattern. In this case, no function will be returned as the domain narrative does not contain any function.

b) The problem domain narrative “cost price for office desks plus cost price for office chairs plus cost price for filing cabinets” therefore does not use function rules as it does not contain any function expression.

c) The translated string “cost price for office desks plus cost price for office chairs plus cost price for filing cabinets” is then used to replace the original expression in the problem domain narrative expression.

step iii) In this case, the formula expression does not use any function rules as it does not contain any function expression.

step iv) The formula expression does not contain any “ ˜ ” since there were no commas.

step v) Special English grammatical rules are then applied to “the sum of cost price for office desks to cost price for filing cabinets.” to get a fully translated natural language expression in English for display.

4.3.2 Example translation of spreadsheet formula “=SUM(D3:D5)”

Given a formula “=SUM(D3:D5)” in cell D6 in the spreadsheet shown in Figure 4.6;

D6 f_x =SUM(D3:D5)

	A	B	C	D	E	F
1	**Furniture Budget**					
2	**Item**	**Quantity**	**Unit Price**	**Cost Price**	**Rank 1**	**Rank 2**
3	Office desks	1	553.00	553.00	High	High
4	Office chairs	2	250.40	500.80	High	High
5	Filing cabinets	1	250.00	250.00	Low	Low
6	**Total Cost**			1303.80		
7	Average			434.60		

Figure 4.6: Sample spreadsheet formula (=SUM(D3:D5)) for translation in cell D6.

The transformation of the formula “=SUM(D3:D5)” will be carried out by first implementing the steps specified in algorithm 1 which transforms the original spreadsheet formula “=SUM(D3:D5)” into problem domain narrative as shown in the following steps;

step i) Referenced cells D3 and D5 are extracted from the formula “=SUM(D3:D5)”.

step ii) Cell D3 has column label “Cost Price” and row label “Office desks ”. Cell D5 has column label “Cost Price” and row label “Filing cabinets ”.

step iii) Symbolic names for D3 and D5 are formed by joining their corresponding column label and row label while separating them by a special character "|". D3 will have symbolic name "cost price | office desks" and D5 will have "cost price | filing cabinets ".

step iv) Each referenced cell in the original spreadsheet formula "=SUM(D3:D5)" is replaced by the corresponding symbolic name. D3 will be replaced by "cost price | office desks" and D5 by "cost price | filing cabinets" to produce the expression "SUM(cost price | office desks ... cost price | filing cabinets)"

step v) The expression "SUM(cost price | office desks ... cost price | filing cabinets)" is now stored as a domain narrative expression.

The output of algorithm 1 in this case, the problem domain narrative is then used as input to algorithm 2 and processed as indicated in the following steps;

step i) The translation process starts by searching for basic operators in the domain narrative "SUM(cost price | office desks ... cost price | filing cabinets)". Basic operators "|" and "..." will be replaced by their corresponding mapping expressions as defined in rule number 4 and 5 respectively from Table 4.1. Thus, the resulting domain narrative expression will be "SUM(cost price for office desks to cost price for filing cabinets)".

step ii) A regular expression pattern is used to search for functions in the domain narrative and then translate them to natural language expressions by repeatedly following the sub-steps below until there are no untranslated functions in the problem domain narrative.

a) The regular expression pattern $[a-zA-Z]+\backslash([\hat{}\backslash(\backslash)]+\backslash)$ is used to extract the function matching the pattern. In this case, only one function is identified "SUM(cost price for office desks to cost price for filing cabinets)", since there are no nested functions.

b) The problem domain narrative "SUM(cost price for office desks to cost price for filing cabinets)" does not have commas to be broken into more than one token and therefore the function expression is in simplest form. Thus, it is returned as one token.

c) The returned token "SUM(cost price for office desks to cost price for filing cabinets)" is translated using its equivalent mapping expression in rule number 5 from Table 4.2 since it is in simplest form, resulting in the expression "the sum of cost price for office desks to cost price for filing cabinets" .

d) The fully translated string “the sum of cost price for office desks to cost price for filing cabinets” representing the function without nesting is then used to replace the original expression in the problem domain narrative expression.

step iii) In this case, the formula expression is only executed once since there are no nested functions.

step iv) The formula expression does not contain any “ ~ ” since there were no commas.

step v) Special English grammatical rules are then applied to “the sum of cost price for office desks to cost price for filing cabinets.” to get a fully translated natural language expression in English for display. In this, none of the rules is applicable.

4.3.3 Example translation of spreadsheet formula “=AVERAGE(D3,D4,D5)”

Given a formula “=AVERAGE(D3, D4, D5)” in cell D7 in the spreadsheet shown in Figure 4.7;

D7 f_x =AVERAGE(D3,D4,D5)

	A	B	C	D	E	F
1	**Furniture Budget**					
2	**Item**	**Quantity**	**Unit Price**	**Cost Price**	**Rank 1**	**Rank 2**
3	Office desks	1	553.00	553.00	High	High
4	Office chairs	2	250.40	500.80	High	High
5	Filing cabinets	1	250.00	250.00	Low	Low
6	**Total Cost**			1303.80		
7	Average			434.60		

Figure 4.7: Sample spreadsheet formula (=AVERAGE(D3,D4,D5)) for translation in cell D7.

The translation of “=AVERAGE(D3,D4,D5)” will first pass through the steps specified in algorithm 1 which transforms the formula “=AVERAGE(D3,D4,D5)” into problem domain narrative as demonstrated in the following steps;

step i) Cells D3, D4 and D5 are extracted from the formula “=AVERAGE(D3,D4,D5)”.

step ii) The column and row labels for cells D3, D4, D5 are inferred from the spreadsheet. D3 has column label “Cost Price” and row label “Office desks ”. D4’s column is also “Cost Price” and row label “ Office chairs ”. Referenced cell D5’s column label “Cost Price” and row label “Filing cabinets ”.

step iii) Symbolic names for D3, D4 and D5 are formed by joining their corresponding column label and row label while separating them by a special character “|”. Symbolic name for D3 “cost

price | office desks ". Symbolic name for D4 is "cost price | office chairs" and for D5 is "cost price | filing cabinets ".

step iv) Symbolic names from the previous step are used to replace D3, D4 and D5 from the spreadsheet formula. D3 by "cost price | office desks ". D4 by "cost price | office chairs" and D5 by "cost price | filing cabinets ". The resulting text is "AVERAGE(cost price | office desks, cost price | office chairs, cost price | filing cabinets)"

step v) The resulting expression "AVERAGE(cost price | office desks, cost price | office chairs, cost price | filing cabinets)" will be stored as a domain narrative.

The stored problem domain narrative is then used as input to algorithm 2 and transformed as it passes through the following steps;

step i) "AVERAGE(cost price | office desks, cost price | office chairs, cost price | filing cabinets)" is searched for the presence of basic operators. The only operator present is "|" will be replaced by its corresponding mapping expression in rule number 5 from Table 4.1. Thus, the resulting expression will be "AVERAGE(cost price for office desks, cost price for office chairs, cost price for filing cabinets)".

step ii) The formula expression from the previous step is searched for presence of inner functions as it goes through sub-steps a to d and the functions identified are then translated.

a) The regular expression pattern $[a-zA-Z]+\backslash([\hat{}\backslash(\backslash)]+\backslash)$ is used to extract inner functions. The narrative "AVERAGE(cost price for office desks, cost price for office chairs, cost price for filing cabinets)" does not contain other functions inside it, thus it will be the expression that is returned.

b) The problem domain narrative "AVERAGE(cost price for office desks, cost price for office chairs, cost price for filing cabinets)" is in the complex form of the AVERAGE function as it has at least one optional argument and will be tokenized using comma to break it into tokens. The following tokens are extracted; "AVERAGE(cost price for office desks", "cost price for office chairs" and "cost price for filing cabinets)".

c) Each token from the previous step is translated using its equivalent mapping from Table 4.2. The equivalent mapping expression rule number 11 for the token "AVERAGE(cost price for office desks" is the expression "the average of cost price for office desks", then for the second token "cost price for office chairs" the same expression is returned as is the case with mapping expression in rule number 12 and the third token

"cost price for filing cabinets)" its corresponding mapping in rule number 14 "and cost price for filing cabinets" is used. The extracted mappings are then used to replace the token expressions from the original problem domain narrative and the resulting fully translated formula expression "the average of cost price for office desks, cost price for office chairs, and cost price for filing cabinets" is formed. The commas in the original untokenized string are temporarily replaced with special character " ∼ " to form "the average of cost price for office desks ˜ cost price for office chairs ˜ and cost price for filing cabinets"

d) The fully translated string "the average of cost price for office desks ˜ cost price for office chairs ˜ and cost price for filing cabinets" representing the function without nesting is then used to replace the original expression in the problem domain narrative expression.

step iii) The formula expression is only executed once since there are no nested functions.

step iv) The special character " ˜ " in the fully translated formula expression is replaced by commas. The resulting expression "the average of cost price for office desks, cost price for office chairs, and cost price for filing cabinets." is formed.

step v) Special English grammatical rule number 1 from Table 4.3 is then applied to get a fully translated natural language expression "the average of cost price for office desks, cost price for office chairs and cost price for filing cabinets" in English for display.

4.3.4 Example spreadsheet formula translation with nested functions

An example formula with nested functions, "IF(D3 < 300, "Low",
IF(D3 < 500, "Medium", IF(D3 < 700, "High", "Very High")))", shown in Figure 4.8 was transformed as shown in the following steps;

F3 =IF(D3<300,"Low",IF(D3<500,"Medium",IF(D3<700,"High","Very High")))

	A	B	C	D	E	F	G	H	I
1	**Furniture Budget**								
2	**Item**	**Quantity**	**Unit Price**	**Cost Price**	**Rank 1**	**Rank 2**			
3	Office desks	1	553.00	553.00	High	High			
4	Office chairs	2	250.40	500.80	High	High			
5	Filing cabinets	1	250.00	250.00	Low	Low			
6	**Total Cost**			1303.80					
7	Average			434.60					

Figure 4.8: Sample nested formula for translation in cell F3.

step i) Cell reference D3 is extracted from the formula "IF(D3 $<$ 300, "Low", IF(D3 $<$ 500, "Medium", IF(D3$<$700, "High", "Very High")))"

step ii) Column label and row label for cell D3 are inferred from the active spreadsheet. D3 has column label "Cost Price" and row label "Office desks ".

step iii) Symbolic name "cost price | office desks" for D3 is formed.

step iv) D3 is replaced by its symbolic name from the spreadsheet formula. D3 will be replaced by "cost price | office desks ". The resulting expression is "IF("cost price | office desks" $<$ 300, "Low", IF("c price | office desks" $<$ 500, "Medium", IF("cost price | office desks" $<$ 700, "High", "Very High")))"

step v) The resulting expression will be stored as a domain narrative.

The resulting problem domain narrative is then used as input to algorithm 2 and transformed as it passes through the following steps;

step i) Basic operators are extracted from the narrative "IF("cost price | office desks" $<$ 300, "Low", IF("cost price | office desks" $<$ 500, "Medium", IF("cost price | office desks" $<$ 700, "High", "Very High")))". The only operators present are "|" and "$<$" will be replaced by their corresponding mapping expressions from rule numbers 4 and 8 respectively from Table 4.1. Thus, the resulting expression will be "IF("cost price for office desks" is less than 300,"Low", IF("cost price for office desks" is less than 500,"Medium", IF("cost price for office desks" is less than 700,"High", "Very High")))".

step ii) The formula expression from the previous step is searched for presence of inner functions as it goes through sub-steps a to d and functions identified are then translated.

a) The regular expression pattern $[a-zA-Z]+\(\[\^\(\)\]+\)$ is used to extract the inner most function. The expression "IF("cost price for office desks" is less than 700,"High", "Very High")" from the problem domain narrative is extracted as it is the inner most function and does not contain other functions inside it, and thus it will be the string returned.

b) The expression "IF(cost price for office desks is less than 700,"High", "Very High")" will be tokenized using comma to break it into tokens as it is in complex form of the IF function. The following tokens are extracted; "IF(cost price for office desks is less than 700", "High" and "Very High")".

c) Each token from the previous step is translated using its equivalent mapping from Table 4.2. The equivalent mapping from rule number 6 for the token "IF(cost price for office desks is less than 700" is the expression "if cost price for office desks is less than 700", then for the second token "High" the expression "the value is High" is returned as indicated on rule number 7 and the third token "Very High)" its corresponding mapping "otherwise Very High." from rule number 8 is used. The extracted mappings are then used to replace the token expressions from the original problem domain narrative and the resulting fully translated formula expression "if cost price for office desks is less than 700, the value is High, otherwise Very High." is formed. The commas in the original untokenized string are temporarily replaced with special character " ˜ " to form "if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High" .

d) The fully translated string "if cost price for office desks is less than 700 ˜ the value is High õtherwise Very High" representing the function without nesting is then used to replace the original expression in the problem domain narrative expression to form "IF("cost price for office desks" is less than 300,"Low","IF(cost price for office desks is less than 500,"Medium", if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High))"

step iii) The formula expression is executed second time since there are more nested functions.

a) The regular expression pattern $[a-zA-Z]+\(([\hat{}\(\)]+\)$ is used to extract the inner most function. The expression IF("cost price for office desks" is less than 500,"Medium", "if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High)" from the problem domain narrative "IF("cost price for office desks" is less than 300,"Low","IF(cost price for office desks is less than 500,"Medium", if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High))" is extracted as it is the inner most function and does not contain inner functions thus it will be the one returned.

b) The expression "IF("cost price for office desks" is less than 500,"Medium", if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High)" is tokenized using comma to break it into tokens. The following tokens were extracted; "IF(cost price for office desks is less than 500", "Medium" and "if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High)".

c) Each token from the previous step is translated using its equivalent mapping from Ta-

ble 4.2. The equivalent mapping for the token “IF(cost price for office desks is less than 500” is extracted from rule number 6, the expression “if cost price for office desks is less than 500”, then for the second token “Medium” the expression “the value is Medium” is returned as shown on rule number 7 and the third token “if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High)” its corresponding mapping “otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High” from rule number 8 is used. The extracted mappings are then used to replace the token expressions from the original problem domain narrative and the resulting fully translated formula expression “if cost price for office desks is less than 500, the value is Medium, otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ Very High.” is formed. The commas in the original untokenized string are temporarily replaced with special character “ ˜ ” to form “if cost price for office desks is less than 500 ˜ the value is Medium ˜ if cost price for office desks is less than 700 ˜ the value is High ˜ Very High” .

d) The fully translated string “if cost price for office desks less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High” representing the function without nesting is then used to replace the original expression in the problem domain narrative expression “IF(“cost price for office desks” is less than 300,“Low”,“if cost price for office desks less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High”)”.

step iv) The formula expression is executed third time since there are still untranslated functions.

a) The regular expression pattern $[a-zA-Z]+\backslash([\hat{}\ \backslash(\backslash)]+\backslash)$ is used to extract the inner most function. The expression “IF(cost price for office desks is less than 300, “Low”, “if cost price for office desks is less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks less than 700 ˜ the value is High ˜ otherwise Very High)” from the problem domain narrative is extracted as it is the inner most function and does not contain other functions inside it anymore thus it will be the string returned.

b) The expression is tokenized using comma to break it into tokens. The following tokens were extracted; “IF(‘cost price for office desks is less than 300”, “Low” and “if cost price for office desks less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks less than 700 ˜ the value is High ˜ otherwise Very High)”.

c) Each token from the previous step is translated using its equivalent mapping from Table 4.2. The equivalent mapping for the token “IF(cost price for office desks is less than 300” from rule number 6 is “ if cost price for office desks is less than 300”, then for the second token, “Low”, the expression “the value is Low” is returned from rule number 7 and the third token “if cost price for office desks is less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ High ˜ otherwise Very High)”, its corresponding mapping “otherwise if cost price for office desks is less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High” from rule number 8 is used. The extracted mappings are then used to replace the token expressions from the original problem domain narrative and the resulting fully translated formula expression “if cost price for office desks is less than 300, the value is Low, otherwise if cost price for office desks is less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High.” is formed. The commas in the original untokenized string are temporarily replaced with special character “ ˜ ” to form “if cost price for office desks is less than 300 ˜ the value is Low ˜ otherwise if cost price for office desks is less than 500 ˜ the value is Medium ˜ otherwise if cost price for office desks is less than 700 ˜ the value is High ˜ otherwise Very High” .

d) The fully translated string representing the function without nesting is then used to replace the original expression in the problem domain narrative.

step v) The formula expression is not iterated again since there are no more nested functions

step vi) The special character “ ˜ ” in the fully translated formula expressions is replaced by commas. The resulting expression “if cost price for office desks is less than 300, the value is Low, otherwise if cost price for office desks is less than 500, the value is Medium, otherwise if cost price for office desks is less than 700, the value is High, otherwise Very High” is formed.

step vii) Special grammatical rule number 2 is then applied to get a fully translated natural language expression “if cost price for office desks is less than 300, the value is Low; otherwise if cost price for office desks is less than 500, the value is Medium; otherwise if cost price for office desks is less than 700; the value is High; otherwise Very High” .

4.4 Tool Implementation

The designed algorithms were then implemented in a software tool using Visual Basic for Applications (VBA). VBA was used because of it is the native application programming language for Microsoft Office Excel and other Microsoft Office products, giving easy access to the Microsoft Excel object model. The source code for the software tool and installation instructions are freely available[1].

4.4.1 Software tool architecture

The developed software tool has five interlinked components as illustrated in Figure 4.9;

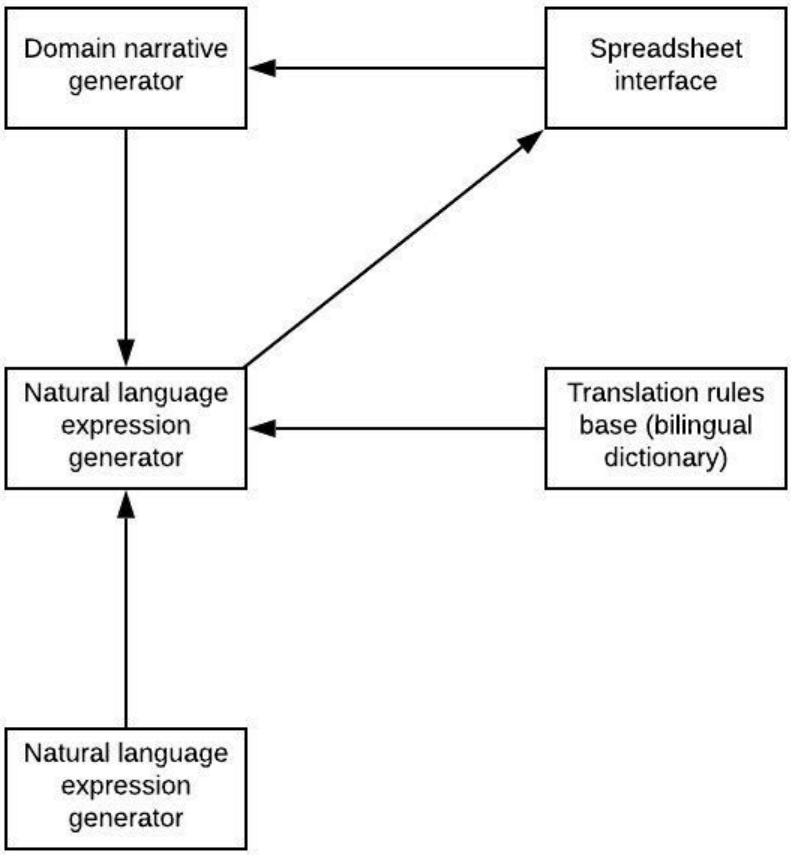


Figure 4.9: System architecture of the translation tool.

The system architecture can be broken down into the following components;

i) Spreadsheet interface: contains cells in which data and formulas are entered. Each cell is an intersection of a specific column and row.

[1] https://github.com/Benson-Mariro/rule_translation/tree/master

ii) The domain narrative generator: generates a domain narrative for an active formula cell based on column labels and row labels for referenced cells in the formula. The domain narrative generator was adapted from [2, 27].

iii) The translation rules base (bilingual dictionary): provides the rules for mapping domain narrative tokens into corresponding English language equivalents

iv) The natural language expression generator: transforms a given domain narrative into a natural language expression by using a parser to identify domain narrative elements that need to be tokenized and translated based on translation rules in the translation rules base (bilingual dictionary). The resulting natural language expression is then displayed by the side of the active formula cell in the spreadsheet interface. The tool also highlights some translated keywords in the display to enhance readability of the translated expressions. The tool also highlights referenced cells for an active formula.

4.5 Translation tool in action

The software tool was run on various spreadsheet formulas to demonstrate its functionality in the translation process.

4.5.1 Tool translation of spreadsheet formula, "=D3+D4+D5"

The translation tool was implemented on the formula "D3+D4+D5" to translate it into natural language expressions as shown in Figure

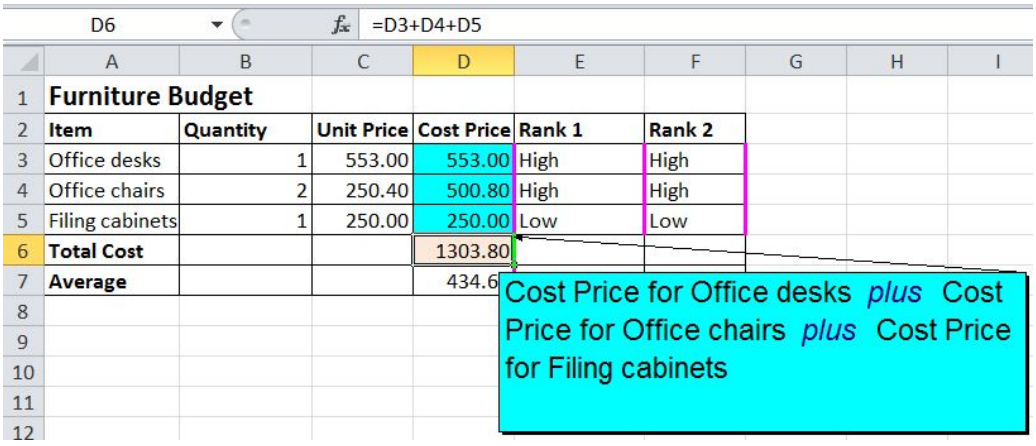


Figure 4.10: Translation of a spreadsheet formula of the form "=D3+D4+D5" in cell D6.

4.5.2 Tool translation of spreadsheet formula, "=SUM(D3:D5)"

A spreadsheet formula of the form "=SUM(D3:D5)" was transformed by the translation tool into "the sum of cost price for office desks to cost price for filing cabinets of quantity. " as shown in Figure 4.11;

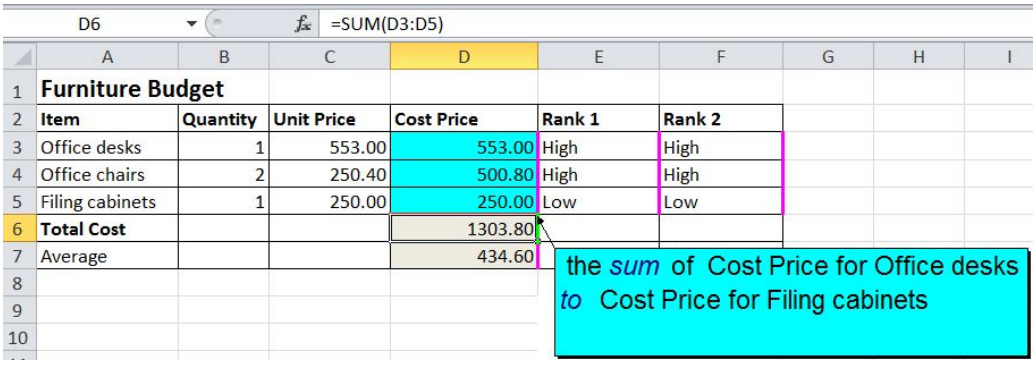


Figure 4.11: Translation of a spreadsheet formula of the form "=SUM(D3:D5)" in cell D6.

4.5.3 Tool translation of spreadsheet formula, "=AVERAGE(D3,D4,D5)"

The formula "AVERAGE(D3,D4,D5)" was translated as illustrated in Figure 4.12;

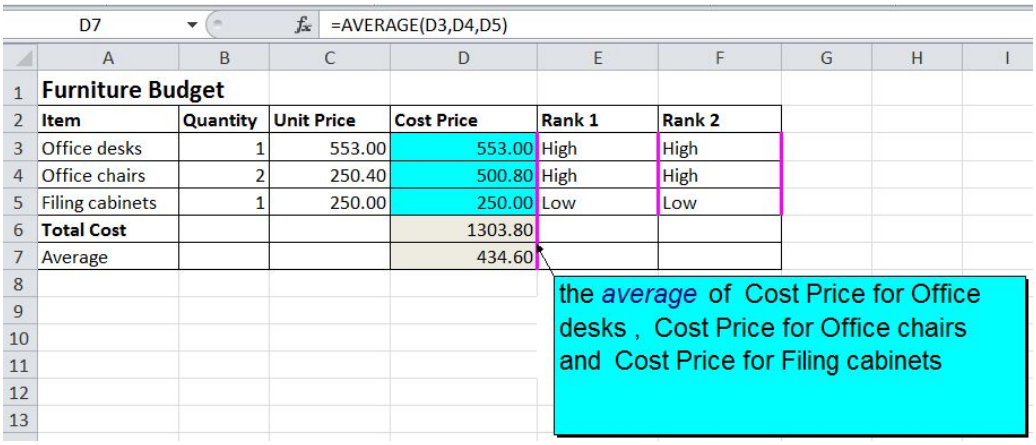


Figure 4.12: Translation of a spreadsheet formula "=AVERAGE(D3,D4,D5)" in cell D7.

4.5.4 Tool translation of spreadsheet formula with nested functions

The translation of nested functions implements parser tree approach to breakdown a nested function into individual inner functions. The translation tool demonstrates how a sample nested function is transformed e.g. a formula of the form "IF(D3 $<$ 300,"Low",
IF(D3 $<$ 500, "Medium", IF(D3 $<$ 700, "High", "Very High"))" was translated into "if Cost price for office desks is less than 300, the value is Low; otherwise if Cost price for office desks is less than 500, the value is medium; otherwise if Cost price for office desks is less than 700, the value is high; otherwise very high " as illustrated in Figure 4.13;

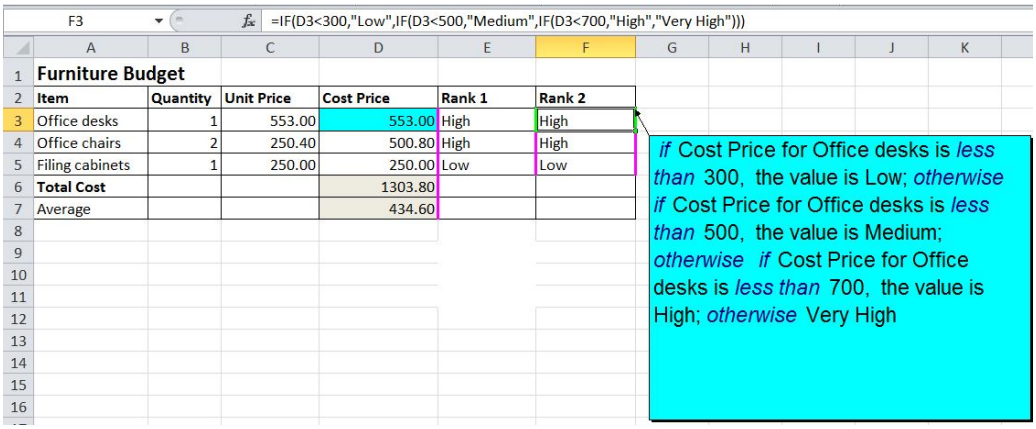


Figure 4.13: Translation of a sample spreadsheet nested formula in cell F3.

4.6 Conclusion

This chapter presented the algorithm design and implementation in translating traditional spreadsheet formulas into natural language expressions. Factors affecting the algorithm design and implementation were also specified including step by step pseudocode of the algorithm and example transformations of the sample traditional spreadsheet formulas into English language expressions. This was to answer research question RQ1: *What rule based machine translation algorithm can be used to translate spreadsheet formulas to natural language expressions?* and the solution to that research question was through the adoption of the direct approach machine translation. The research question RQ2: *How can we automatically translate spreadsheet formulas to natural language expressions using a given rule based machine translation algorithm?* was answered by the development of the translation tool which implemented the direct approach machine translation. In the next chapter, the translation tool which was implemented from the algorithm is evaluated to determine its effectiveness in spreadsheet debugging.

Chapter 5

Evaluation of Software Tool

5.1 Introduction

This chapter focuses on the evaluation of the software tool that translates spreadsheet formulas into natural language expressions based on devised translation rules. In particular, the tool is evaluated with respect to how translated formulas affect spreadsheet debugging in order to answer research question **RQ3**: *What is the effect of natural language expressions on spreadsheet users when debugging or locating errors in spreadsheets?* In other words, do spreadsheet formulas expressed in natural language make it easier to comprehend and understand them and consequently make it easier to locate errors in spreadsheets? With respect to spreadsheet debugging, the tool is evaluated on two metrics: (a) efficiency of spreadsheet users when debugging spreadsheets using the tool (b) satisfaction of users when using the tool as they debug spreadsheets. An evaluation study based on a within-subjects design (repeated measures) was thus conducted.

5.1.1 Selection of participants

The participants involved in the evaluation study were identified considering their background in accounting and frequency in using spreadsheets. Eight participants volunteered to participate in the study. Two of the participants were female while the remaining six participants were male. All the eight participants are accountants by profession who frequently use spreadsheets in their day to day work activities. They all use Microsoft Excel in their spreadsheet tasks. The participants' experience was summarized as below in Table 5.1:

Table 5.1: Participants' experience in years

Participant	Experience(years)
P1	7
P2	8
P3	5
P4	9
P5	6
P6	4
P7	8
P8	9

5.2 Efficiency of the translation tool

5.2.1 Method

Two real-life non-trivial Microsoft Office Excel spreadsheets were sourced and adapted from the EUSES spreadsheet corpus [47]. The two spreadsheets contained real-life accounting information. In this evaluation study, both spreadsheets were seeded with real-life errors as presented in Appendix C, which could be classified as mechanical and logical errors. Mechanical errors are generated from wrong typing or pointing in spreadsheets [9], [22]. Logical errors occur when using the wrong function or applying wrong formula in spreadsheets [9], [22]. In this study, five sample errors were identified representing each category of common real life errors in spreadsheets, the following five examples of errors were seeded:

a) Misuse of built-in functions e.g instead of using AVERAGE(), the user types it as AVERAGEA(). This is a mechanical error.

b) Sign errors e.g instead of * , user types +. This is a logical error.

c) Hard-coding values in formula cells instead of using cell references. This ia a mechanical error.

d) Referencing wrong cells in formulas e.g. whereby a formula would refer to a neighbouring range. This is a mechanical error.

e) Using text values instead of numeric data values, e.g. an error whereby a formula would reference cells including labels (text), This is a mechanical error.

The five seeded errors were identified due to the fact that they fall under quantitative errors which are mostly reported in spreadsheet errors [53].

Four participants debugged their first spreadsheet without translations (without tool) and the second spreadsheet with the translations (two starting with a spreadsheet denoted as SP1 and two starting with a spreadsheet denoted as SP2); the other four participants debugged their first spreadsheet with translations (with tool) and the second without the tool (and again two of them starting with spreadsheet SP1 while the remaining two starting with spreadsheet SP2). The splitting of the participants into smaller groups of twos was done randomly as the sample population was homogeneous. Each participant performed the tasks individually at their place of work after being briefed on the nature of the task. Each participant was told before hand that the spreadsheets to be used in the tasks had seeded errors. The specific number of seeded errors, however, was not told before hand. The participants were told that their task was to find the errors in the two scenarios. Participants who started doing the task with the tool were first briefed on what the tool does. Some participants (three participants out of the eight participants) carried out the evaluation study under observation but the rest of participants carried out the evaluation without observation due to constraints of time and distance. After all the two tasks, the spreadsheets were saved.

5.2.2 Results

The minimum number of identified errors in spreadsheets without translations was 60% and maximum number of identified errors in spreadsheets without translations was 100%. The minimum and maximum number of identified errors in spreadsheets with translations was 80% and 100% respectively.

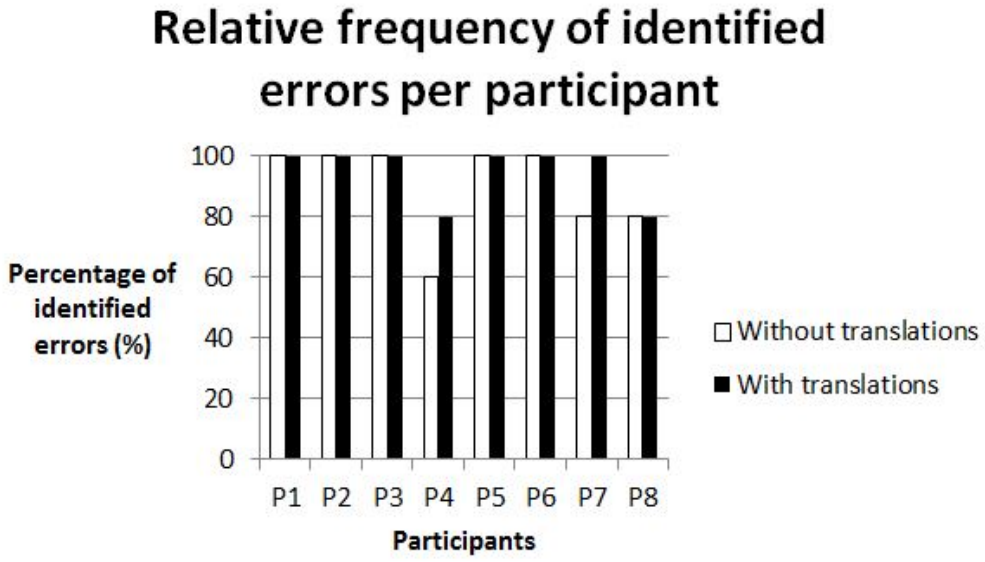


Figure 5.1: Relative frequency of identified errors per participant without translations and with translations.

The mean relative frequency of errors located in spreadsheets without translations was 90% (S.D. 14.14) and with translations was 95% (S.D. 8.66). The relative frequency of errors located in spreadsheets without translations and with translations for each participant are illustrated in Figure 5.1. As indicated in Figure 5.1 participants were given more time such that most (six participants) of them managed to identify all the 5 seeded errors whilst the remaining (two participants) totally failed to locate some of the errors. The two participants who failed totally to identify all errors had some incidents where they located errors that were not actual errors but because we were timing the time taken to identify each error, these participants were simply given more time to locate the seeded errors.

Using the Shapiro-Wilk's test normality test, the data for the relative frequency of errors identified by each participant was found not to be normally distributed. A non-parametric test which can be used for small sample size [54], Wilcoxon signed-rank test, was therefore used to test if the difference in the mean relative frequency of errors identified by each participant without translations and with translations was statistically significant. At significance level of 0.05, the Wilcoxon signed-rank test indicated that the mean relative frequency of errors identified (95%) with translations was not significantly higher than the mean relative frequency of errors identified (90%) without translations, $Z = -1.414$, $p = 0.157$.

The minimum time taken to debug a spreadsheet without translations was 420 seconds and maximum time taken to debug a spreadsheet without translations was 1560 seconds. The minimum and maximum time taken to debug a spreadsheet with translations was 240 seconds and 900 seconds respectively. The mean time taken to debug a spreadsheet without translations was 907.5 seconds (S.D. 6.8) and with translations was 495 seconds (S.D. 3.56). To better understand how much time was spent to debug each error in the two scenarios, the error detection rate per participant was also calculated. The error detection rate was derived by dividing the total amount taken (in seconds) by a participant to debug a spreadsheet in each scenario and the number of identified errors by the participant. The minimum error detection rate to debug a spreadsheet without translations was 96 seconds per error and maximum error detection rate to debug a spreadsheet without translations was 400 seconds per error. The minimum and maximum error detection rate to debug a spreadsheet with translations was 48 seconds per error and 150 seconds per error, respectively. The mean error detection rate for the spreadsheet without translations was 201.7 seconds per error and with translations was 97.1 seconds per error for all participants. The error detection rates for the study participants are illustrated in Figure 5.2.

Similarly, using the Shapiro-Wilk's normality test, the data for the error detection rate by each

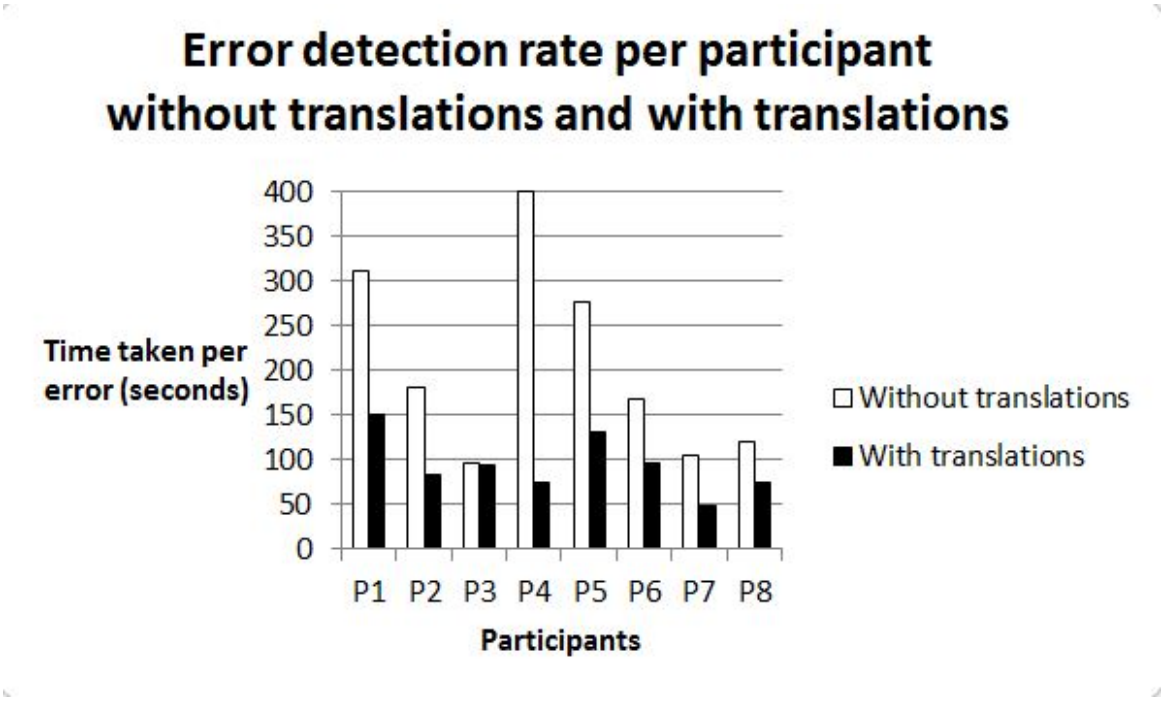


Figure 5.2: Error detection rate per participant without translations and with translations.

participant was found not to be normally distributed. A non-parametric test, Wilcoxon signed-rank test, was therefore used to test if the difference in the mean error detection rate by each participant without translations and with translations was statistically significant. At significance level of 0.05, the Wilcoxon signed-rank test indicated that the mean error detection rate (97.1 seconds per error) with translations was significantly lower than the mean error detection rate (201.7 seconds per error) without translations, Z = -2.521, p = 0.012.

5.2.3 Discussion

The results indicate that having translations in spreadsheets did not help spreadsheet users to locate more errors in spreadsheets as compared to when not using translations as the difference in the means of the relative frequency of identified errors was not statistically significant Z = -1.414, p = 0.157. However, despite this being the case, the results also indicate that translations speeded up the debugging process as spreadsheet users took significantly less time in detecting each error than without translations, Z = -2.521, p = 0.012. As already reported, the mean time for locating each error with translations was 97.1 seconds per error while without translations, it took 201.7 seconds per error. Furthermore, the results also indicate that all participants took less time to locate errors with translations. The results therefore indicate that translations improve efficiency in debugging, particularly, in terms of the time taken to locate an error. These results suggest that translations to natural language expressions could make spreadsheet debugging to be faster than without translations.

5.3 Satisfaction

5.3.1 Method

After completing the debugging tasks, each participant was asked to write down their opinion on whether they found the tool helpful and the reasons for their opinion.

5.3.2 Results

All the eight participants in the study indicated that they found the translations useful. One participant noted that the translations made it easier to comprehend the formulas in terms of the problem domain and also helped in identifying errors faster than without translations. Another participant also highlighted that initially had problems identifying errors without translations which was not the case when using the tool as the spreadsheet with the tool made it easier to comprehend the formulas and understand the calculations. Another participant also indicated that the tool assisted to take lesser time locating errors than when locating errors on the spreadsheet without translations. The participant also indicated that the tool provided a "user friendly environment" than the spreadsheet without translations. Another participant also mentioned that debugging a spreadsheet with the tool was more efficient than without translations. Another one indicated the the tool was important and necessary in debugging, although did not indicate how relevant was the tool in debugging.

5.3.3 Discussion

The opinions from the participants indicated that the tool was effective in helping spreadsheet users when debugging the spreadsheets. All 8 participants expressed that the tool was useful in debugging spreadsheets. The tool translated spreadsheet formulas into natural language expressions, and this helped the participants to easily comprehend spreadsheet formulas in terms of the problem domain. The opinions of participants also agree with the results that indicate that translations speeded up the debugging effort as spreadsheet users took significantly less time in detecting each error than without translations, $Z = -2.521$, $p = 0.012$. This confirms that participants found utility in using the tool.

5.4 Limitations of study

The study has some limitations that also need to be considered. First, the number of participants was small to generalize the results to all spreadsheet users. However, the results can be considered to be providing insights in the usefulness of formula translations to natural language expressions in spreadsheets. Second, the spreadsheets used were not created by the participants themselves. However, the

spreadsheets were adapted to be easy enough to be understood by any professional accountant without necessarily being the creator of the spreadsheets. Third, the spreadsheets used in the debugging tasks were seeded with errors and thus making them to be artificially introduced in the spreadsheets. It would also be interesting to also have study participants debug spreadsheets without seeded errors. Fourth, the formulas in the spreadsheets used in the debugging tasks were relatively simple. It would also be interesting to see how translations of complex formulas could affect the efficiency of study participants in debugging tasks.

5.5 Conclusion

This chapter presented the evaluation of the software tool that translates spreadsheet formulas into natural language expressions based on devised translation rules. The tool was evaluated with respect to how translated formulas affect spreadsheet debugging in order to answer research question **RQ3**: *What is the effect of natural language expressions on spreadsheet users when debugging or locating errors in spreadsheets?* With respect to spreadsheet debugging, the tool was evaluated on two metrics:

a) efficiency of spreadsheet users when debugging spreadsheets using the tool.

b) satisfaction of users when using the tool as they debug spreadsheets.

It was found that the tool improved the efficiency of spreadsheet users when debugging as study participants spent less time in finding errors when using the tool unlike when not using the tool. Tool users also expressed satisfaction that they found the translations by the tool useful. In the next chapter, final concluding remarks are presented.

Chapter 6

Conclusion and Recommendations

6.1 Summary of research work

The prevalence of spreadsheet errors and their implications in organizations motivated this research work. A lot of empirical evidence has indicated that these errors are a pervasive problem both in business and other real life settings [9]. Research has shown that most errors in spreadsheets are formula based [11]. Spreadsheet formulas are normally specified using A1 references (alphanumeric cell addresses). Spreadsheet formulas can be understood only if one associates the referenced cells to their labels. This increases mental effort (cognitive load) in a person comprehending a spreadsheet formula [12, 19] and hence makes spreadsheet formula comprehension to be error prone [15]. Several researchers have therefore proposed translating traditional spreadsheet formulas to higher level problem domain oriented forms, based on labels of referenced cells in a formula [2, 17, 18]. This research work involved translating spreadsheet formulas into natural language expressions (English) based on devised translation rules. The translation of spreadsheet formulas into natural language expressions is meant to help spreadsheet users to easily comprehend spreadsheet formulas in relation to the problem domain. The three specific objectives of this research work were to:

a) Design a rule-based machine translation algorithm that can be used to translate spreadsheet formulas to natural language expressions.

b) Develop a software tool that automatically translates spreadsheet formulas to natural language expressions using an identified rule-based machine translation algorithm.

c) Evaluate empirically the resulting natural language expressions, in particular, on their effect on how they can help spreadsheet users to debug their spreadsheets.

The constructive research methodology was used to accomplish the three specific objectives. First, a rule based translation algorithm was developed and is presented in Chapter 4 of this dissertation. The

algorithm outlined steps on how spreadsheet formulas can be translated into natural language expressions using devised translation rules (bilingual dictionaries). This provided the answer to the research question corresponding to the first objective. The corresponding question to the first objective was **RQ1**: *What rule based machine translation algorithm can be used to translate spreadsheet formulas to natural language expressions*? In order to achieve the second objective, a software tool was also developed, based on the previously specified algorithm, and its description is also presented in Chapter 4. This provided the answer to the research question corresponding to the second objective. The corresponding question to the second objective was **RQ2**: *How can we automatically translate spreadsheet formulas to natural language expressions using a given rule based machine translation algorithm*?

Through a user study, the developed software tool was then evaluated to determine the effect of natural language expressions on spreadsheet users when debugging spreadsheets. The user study is presented in Chapter 5. In the study, it was found that translations improved the efficiency of spreadsheet users when debugging with respect to time spent in debugging spreadsheets. This provided the answer to the research question corresponding to the third objective. The corresponding question to the third objective was **RQ3**: *What is the effect of natural language expressions on spreadsheet users when debugging or locating errors in spreadsheets*?

6.2 Key contributions of research work

This research work has provided the following three unique contributions to spreadsheet research:

a) An algorithm for translating traditional spreadsheet formulas into natural language expressions based on devised translation rules.

b) A prototype software tool, that implemented the designed algorithm for translating traditional spreadsheet formulas into natural language expressions. The source code for the software tool and installation instructions are freely available[1].

c) Through a user study, demonstrated the utility of having spreadsheet formulas in natural language expressions with respect to spreadsheet debugging.

6.3 General limitations of research work

This research work has the following two major limitations:

[1] https://github.com/Benson-Mariro/rule_translation/tree/master

a) Microsoft Excel, the spreadsheet language considered in this research work, has more than 450 functions as of the 2016 version. However, in this work, the translations of only three functions (SUM, AVERAGE and IF) have been presented as it will be tedious to present the translations of all the functions in Microsoft Excel. However, the algorithm and design of translation rules can be applied to any Microsoft Excel function. For example, in Appendix B, translation rules for the VLOOKUP function are provided.

b) The number of participants in the evaluation study was small to generalize the results to all spreadsheet users. However, the results can be considered to be providing insights in the usefulness of formula translations to natural language expressions in spreadsheets.

c) Due to small sample size, there was threat to validity of the evaluation results. The results from the evaluation study could have been biased and unreliable but considering that the sample size was further split into smaller groups to avoid bias of the evaluation outcome and managed to generate replicating results thus, we can conclusively state that increasing the sample size would still produce the same replicating results.

6.4 Recommendations and future work

This research has shown that translation of traditional spreadsheet formulas to natural language expressions can be useful to spreadsheet users when debugging their spreadsheets. This finding can be applicable to other end-user programming environments where instead of presenting information to users in cryptic form, corresponding natural language expressions could be presented. This could improve the usability of the programming environment. We thus recommend the automatic generation of corresponding natural language expressions in situations where information displayed to users might be cryptic. As part of future work, the research intends to devise translation rules for more Microsoft Excel functions. A study will also be conducted to determine how translations of more complex formulas could affect the efficiency of study participants in debugging tasks. The translation tool can also be integrated with an error detection component to help spreadsheet users in identifying errors. The combined effect of translating the spreadsheet formulas into natural language expressions and error detection can be effective in minimizing errors in spreadsheets.

Bibliography

[1] W. J. Hutchins and H. L. Somers, *An introduction to machine translation*. Academic Press London, 1992, vol. 362.

[2] B. Kankuzi and J. Sajaniemi, "A domain terms visualization tool for spreadsheets," in *Proceedings of the 2014 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2014, pp. 209–210.

[3] S. Tripathi and J. K. Sarkhel, "Approaches to machine translation," *Annals of Library and Information Studies*, vol. 57, pp. 388–393, 2010.

[4] S. E. Kruck and S. D. Sheetz, "Spreadsheet accuracy theory," *Journal of Information Systems Education*, vol. 12, no. 2, pp. 93–108, 2001.

[5] R. Abreu, J. Cunha, J. P. Fernandes, P. Martins, A. Perez, and J. Saraiva, "Smelling faults in spreadsheets," in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 111–120.

[6] J. P. Caulkins, E. L. Morrison, and T. Weidemann, "Spreadsheet errors and decision making: Evidence from field interviews," *Journal of Organizational and End User Computing*, vol. 19, no. 3, p. 1, 2007.

[7] J. Cunha, J. P. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *International Conference on Computational Science and Its Applications*. Springer, 2012, pp. 202–216.

[8] B. R. Lawson, K. R. Baker, S. G. Powell, and L. Foster-Johnson, "A comparison of spreadsheet users with different levels of experience," *Omega*, vol. 37, no. 3, pp. 579–590, 2009.

[9] R. R. Panko, "Spreadsheet errors: What we know. what we think we can do," *In:Proceedings of the European Spreadsheet Risk Interest Group Symposium, EuSpRIG, Greenwich, UK.*, 2008.

[10] S. G. Powell, K. R. Baker, and B. Lawson, "Impact of errors in operational spreadsheets," *Decision Support Systems*, vol. 47, no. 2, pp. 126–132, 2009.

[11] K. Rajalingham, D. Chadwick, B. Knight, and D. Edwards, "Quality control in spreadsheets: A software engineering-based approach to spreadsheet development," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.* IEEE, 2000, p. 10.

[12] D. G. Hendry and T. R. Green, "Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model," *International Journal of Human-Computer Studies*, vol. 40, no. 6, pp. 1033–1065, 1994.

[13] B. Kankuzi and J. Sajaniemi, "An empirical study of spreadsheet authors' mental models in explaining and debugging tasks," in *Proceedings of the 2013 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2013, pp. 15–18.

[14] J. Sweller, "Cognitive load theory, learning difficulty, and instructional design," *Learning and Instruction*, vol. 4, no. 4, pp. 295–312, 1994.

[15] F. Paas, A. Renkl, and J. Sweller, "Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture," *Instructional Science*, vol. 32, no. 1-2, pp. 1–8, 2004.

[16] K. Rajalingham and D. Chadwick, "Integrity control of spreadsheets: Organisation & tools," in *Integrity and Internal Control in Information Systems*. Springer, 1998, pp. 147–168.

[17] D. Nardi and G. Serrecchia, "Automatic generation of explanations for spreadsheet applications," in *Proceedings of the Tenth Conference on Artificial Intelligence for Applications.* IEEE, 1994, pp. 268–274.

[18] F. Hermans, M. Pinzger, and A. Van Deursen, "Supporting professional spreadsheet users by generating leveled dataflow diagrams," in *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 451–460.

[19] B. Kankuzi and J. Sajaniemi, "A mental model perspective for tool development and paradigm shift in spreadsheets," *International Journal of Human-Computer Studies*, vol. 86, pp. 149–163, 2016.

[20] H. Kamp and U. Reyle, *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*. Springer Science & Business Media, 2013, vol. 42.

[21] P. Antony, "Machine translation approaches and survey for indian languages," *International journal of computational linguistics and Chinese language processing*, vol. 18, no. 1, pp. 47–78, 2013.

[22] S. G. Powell, K. R. Baker, and B. Lawson, "Errors in operational spreadsheets," *Journal of Organizational and End User Computing (JOEUC)*, vol. 21, no. 3, pp. 24–36, 2009.

[23] R. McKeever, K. McDaid, and B. Bishop, "An exploratory analysis of the impact of named ranges on the debugging performance of novice users," *arXiv preprint arXiv:0908.0935*, 2009.

[24] F. J. Och, "Statistical machine translation: from single-word models to alignment templates," Ph.D. dissertation, Bibliothek der RWTH Aachen, 2002.

[25] D. Stein, "Machine translation-past, present, and future," *Translation: Computation, Corpora, Cognition*, vol. 3, no. 1, 2013.

[26] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 263–270.

[27] B. Kankuzi, "Dynamic translation of spreadsheet formulas to problem domain narratives," *Proceedings of the 2017 Annual Workshop of the Psychology of Programming Interest Group(PPIG)*, pp. 86–95, 2017.

[28] E. Zournatzidou, "Spreadsheet error detection performance: an empirical examination in greece." *International Conference on Applied Business and Economics*, pp. 128–143, 2015.

[29] Y. Ayalew, "Spreadsheet testing using interval analysis," *PHD Thesis, Universität Klagenfurt, Universitätsstrasse, A-9020 Klagenfurt, Austria*, vol. 1, pp. 1–2, 2001.

[30] M. Mor, "Software testing techniques for faults or errors detection," *International Journal of Computer Science and Communication Networks*, vol. 6, pp. 143–147, 2016.

[31] D. F. Galletta, K. S. Hartzel, S. Johnson, J. Joseph, and S. Rustagi, "An experimental study of spreadsheet presentation and error detection," in *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on*, vol. 2. IEEE, 1996, pp. 336–345.

[32] B. Bishop and K. McDaid, "An empirical study of end-user behaviour in spreadsheet error detection & correction," *arXiv preprint arXiv:0802.3479*, 2008.

[33] R. R. Panko and R. H. Sprague, "Hitting the wall: errors in developing and code inspecting asimple'spreadsheet model," *Decision Support Systems*, vol. 22, no. 4, pp. 337–353, 1998.

[34] Y. Ayalew, M. Clermont, and R. T. Mittermeir, "Detecting errors in spreadsheets," *arXiv preprint arXiv:0805.1740*, 2008.

[35] S. K. McKay, "Reducing spreadsheet errors," Corps of Engineers Washington DC Ecosystem Management and Restoration Research Program, Tech. Rep., 2009.

[36] R. R. Panko, "Recommended practices for spreadsheet testing," *arXiv preprint arXiv:0712.0109*, 2007.

[37] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa, "Avoiding, finding and fixing spreadsheet errors–a survey of automated approaches for spreadsheet qa," *Journal of Systems and Software*, vol. 94, pp. 129–150, 2014.

[38] S. G. Powell, K. R. Baker, and B. Lawson, "An auditing protocol for spreadsheet models," *Information & Management*, vol. 45, no. 5, pp. 312–320, 2008.

[39] O. G. plc, "Operis analysis kit," 2018, uRL: https://www.operisanalysiskit.com/, Accessed September 2018.

[40] D. Nixon and M. O'Hara, "Spreadsheet auditing software," *arXiv preprint arXiv:1001.4293*, 2010.

[41] R. Abreu, A. Riboira, and F. Wotawa, "Constraint-based debugging of spreadsheets." in *CIbSE*, 2012, pp. 1–14.

[42] W. Dou, C. Xu, S. C. Cheung, and J. Wei, "Cacheck: Detecting and repairing cell arrays in spreadsheets." *IEEE Trans. Software Eng.*, vol. 43, no. 3, pp. 226–251, 2017.

[43] S.-C. Cheung, W. Chen, Y. Liu, and C. Xu, "Custodes: automatic spreadsheet cell clustering and smell detection using strong and weak features," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 464–475.

[44] R. Singh, B. Livshits, and B. Zorn, "Melford: Using neural networks to find spreadsheet errors," Technical Report MSR-TR-2017-5. Microsoft, Tech. Rep., 2017.

[45] D. W. Barowy, E. D. Berger, and B. Zorn, "Excelint: automatically finding spreadsheet formula errors," *Proceedings of the ACM on Programming Languages*, vol. 2, no. OOPSLA, p. 148, 2018.

[46] E. Kasanen, K. Lukka, and A. Siitonen, "The constructive approach in management accounting research," *Journal of Management Accounting Research*, vol. 5, p. 243, 1993.

[47] M. Fisher and G. Rothermel, "The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, 2005, pp. 1–5.

[48] H. Suri, "Purposeful sampling in qualitative research synthesis," *Qualitative research journal*, vol. 11, no. 2, pp. 63–75, 2011.

[49] E. Aivaloglou, D. Hoepelman, and F. Hermans, "Parsing excel formulas: A grammar and its application on four large datasets," *Journal of Software: Evolution and Process*, vol. 29, no. 12, p. e1895, 2017.

[50] Microsoft Corporation, "Structures," 2018, uRL: https://msdn.microsoft.com/en-us/library/dd922181(v=office.12).aspx, Accessed October 2018.

[51] S. Greenbaum, *The Oxford English grammar*. Oxford University Press Oxford, 1996, vol. 652.

[52] E. Aivaloglou, D. Hoepelman, and F. Hermans, "A grammar for spreadsheet formulas evaluated on two large datasets," in *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on*. IEEE, 2015, pp. 121–130.

[53] R. R. Panko, "Two corpuses of spreadsheet errors," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. IEEE, 2000, pp. 8–pp.

[54] F. S. Nahm, "Nonparametric statistical tests for the continuous data: the basic concept and the practical use," *Korean journal of anesthesiology*, vol. 69, no. 1, p. 8, 2016.

Appendix A

Some of the production rules for Microsoft Excel grammar, specified using the augmented Backus–Naur (ABNF) notation

formula = expression

*expression= ref-expression / *whitespace nospace-expression *whitespace*

*ref-expression= *whitespace ref-nospace-expression *whitespace*

nospace-expression = "(" expression ")" / constant / prefix-operator expression / expression infix-operator expression / expression postfix-operator / function-call

ref-nospace-expression = "(" ref-expression ")" / ref-constant / ref-expression ref-infix-operator ref-expression / cell-reference / ref-function-call / name-reference / structure-reference

constant = error-constant / logical-constant / numerical-constant / string-constant / array-constant
ref-constant = "#REF!"

*operator = " : "/comma/space/"/" * "/"/"/" + "/" − "/"&"/" = "/" <> "/" < "/" <= "/" > "/" >= "/"%"*

function-call = ref-function-call / future-function-call / cell-function-call / user-defined-function-call / "ABS" abs-params / "ACCRINT" accrint-params / "ACCRINTM" accrintm-params / "ACOS" acos-params / "ACOSH" acosh-params / "ADDRESS" address-params / "AMORDEGRC" amordegrc-params / "AMORLINC" amorlinc-params / "AND" and-params / "AREAS" areas-params / "ASC" asc-params / "ASIN" asin-params / "ASINH" asinh-params / "ATAN" atan-params / "ATAN2" atan2-params / "ATANH" atanh-params / "AVEDEV" avedev-params / "AVERAGE" average-params / ...

*average-params = "(" (argument-expression / (argument 1*254("," argument))) ")"*

*argument = *whitespace [argument-expression]*

*argument-expression= ref-argument-expression / *whitespace nospace-argument-expression *whitespace*

*ref-argument-expression= *whitespace ref-argument-nospace-expression *whitespace*

nospace-argument-expression = “(“ expression ”)” / constant / prefix-operator argument-expression / argument-expression argument-infix-operator argument-expression / argument-expression postfix-operator / function-call

ref-argument-nospace-expression = “(“ ref-expression ”)” / ref-constant / ref-argument-expression ref-argument-infix-operator ref-argument-expression / cell-reference / ref-function-call / name-reference / structure-reference

argument-infix-operator = ref-argument-infix-operator / value-infix-operator

Appendix B

Rules table for the VLOOKUP function

Table 1: Rules table for the VLOOKUP function

Rule No.	**Function Expression**	**Expression Type**	**Token**	**Natural language Expression**
1	VLOOKUP(value, table, col_index, [range_lookup])	complex	VLOOKUP(value	look up for value
2	VLOOKUP(value, table, col_index, [range_lookup])	complex	table	in the range table
3	VLOOKUP(value, table, col_index, [range_lookup])	complex	col_index	col_index
4	VLOOKUP(value, table, col_index, [range_lookup])		[range_lookup])	[with approximate match being range_lookup]
5	VLOOKUP(value, table, col_index)	simplest	VLOOKUP(value	look up for value
6	VLOOKUP(value, table, col_index)	simplest	table	in the range table
7	VLOOKUP(value, table, col_index)	simplest	col_index)	and return corresponding value in column col_index

Appendix C

Spreadsheets used in the evaluation study

	B	C	D	E	F	G	H	I	J	K	L	M	N
1													
2													
3						**Group Profit Summary**								
4													
5			1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	**1999**
6													
7		Group trading profit	128	139.1	108.4	90.9	118.5	156.3	216.5	272.7	343.5	436.4	**=651.6**
8		Share of trading profit in joint ventures	3.8	8.8	7.2	9.7	7.2	16.9	8.1	10.8	14.2	15.4	**11.8**
9		Exceptional items											**64.2**
10		Total trading profit, including share of joint ventures	=SUM(D7:D9)	=SUM(E7:E9)	=SUM(F7:F9)	=SUM(G7:G9)	=SUM(H5:H9)	=SUM(I7:I9)	=216.5+8.1	=SUM(K6:K9)	=SUM(L7:L9)	=SUM(M7:M9)	**=SUM(N7:N8)**
11		Interest payable and similar charges (net)											
12		*- Group*	-20	-33.3	-28.9	-23.7	-28.1	-23.4	-19.1	-24.3	-32.1	-37.5	**-91.8**
13		*- share of joint ventures*	-5.2	-7.2	-6.3	-3.7	-2.3	-1.6	-1.6	-3.3	-4.1	-5.4	**-0.9**
14		Profit before taxation	=SUM(D10:D13)	=SUM(E10:E13)	=SUM(F10:F13)	=SUM(G10:G13)+0.1	=SUM(H10:H13)	=SUM(I10:I13)	=SUM(J10:J13)	=SUM(K10:K13)	=SUM(L10:L13)	=SUM(M10:M13)	**=SUM(N10:N13)**
15		Taxation	-22.1	-21	-14.7	-13.2	-17.6	-27.7	-41.8	-58.3	-75.7	-99.9	**-152**
16		Taxation on exceptional items											**-25.7**
17		Profit after taxation	=SUM(D14:D16)	=AVERAGE(E14:E16)	=SUM(F14:F16)	=SUM(F14:F16)	=SUM(H14:H16)	=SUM(I14:I16)	=SUM(J14:J16)	=SUM(K14:K16)	=SUM(L14:L16)	=SUM(M14:M16)	=SUM(N14:N16)

Figure 1: Formula view of spreadsheet 1 (SP1) used in the debugging task. The spreadsheet has seeded errors in cells G14, G17, H10, I17, J10.

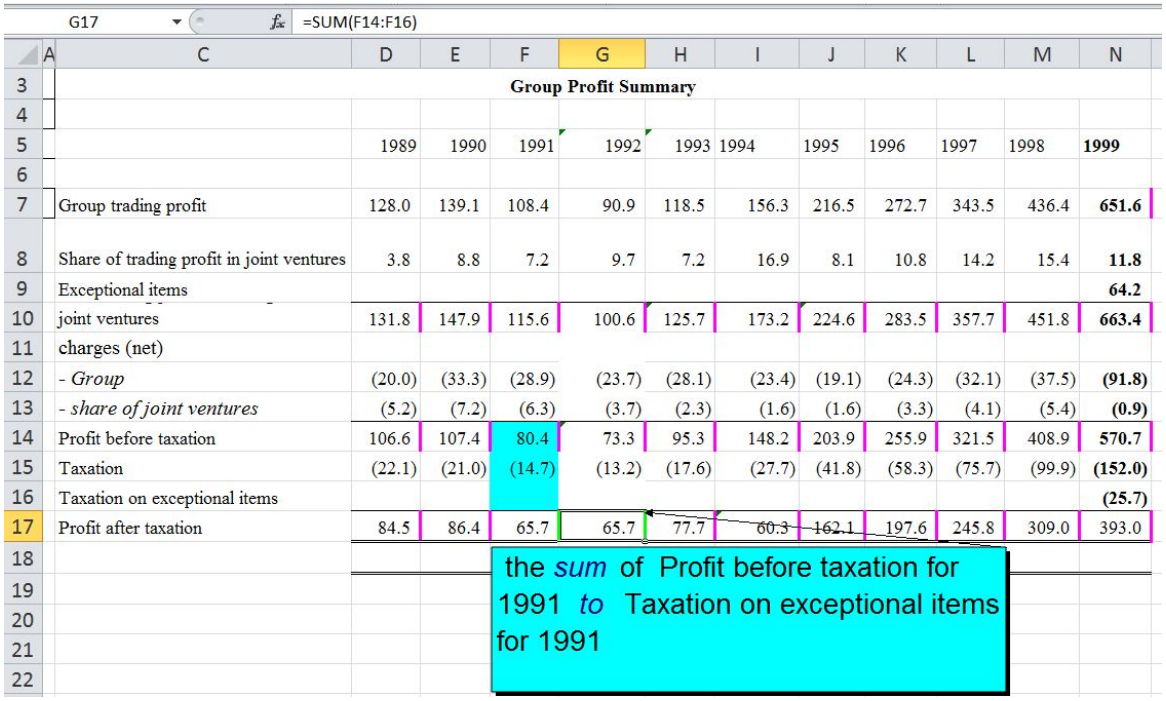


Figure 2: Numeric view of spreadsheet 1 (SP1) used in the debugging task with translation of erroneous cell G17.

	A	B	C	D	E	F	G	H
1								
2	Consolidated Statements of Shareholders' Equity							
3	[DOLLARS IN THOUSANDS]							
4								
5								
6		Common Shares Number	Common Shares Par Value	Additional Capital	Retained Earnings	Treasury Shares	Other	Total
7	**Balance as at 1 January 1999**	69494483	86868	43281	604227	-21902	-549	=SUM(C7:G7)
8								
9	Net income				128856			=SUM(C9:G9)
10	Translation adjustment							=SUM(C9:G9)
11	Pensions							=SUM(C11:G11)
12	Unrealized loss on investment securities							=SUM(C12:G12)
13	Other comprehensive income							=SUM(C13:G13)
14	Comprehensive income							=SUM(C14:G14)
15	Stock options exercised	108104	134	1918				=SUM(C15:G15)
16	Unearned compensation	149799	188	3933			-3485	=188+3933-3485
17	Performance shares	20397	26	686				=SUM(C17:G17)
18	Procomp and Nexus acquisitions	1710214	2138	37351		9487		=SUM(C18:G18)
19	Dividends declared and paid				-41668			=SUM(C19:G19)
20	**Balance as at 31 December 1999**	=AVERAGE(B7:B19)	=SUM(C7:C19)	=SUM(D7:D19)	=SUM(E7:E19)	=SUM(F6:F19)	=SUM(G7:G19)	=SUM(H7:H19)

Figure 3: Formula view of spreadsheet 2 (SP2) used in the debugging task. The spreadsheet has seeded errors in cells B20, F20, H7, H10, H16.

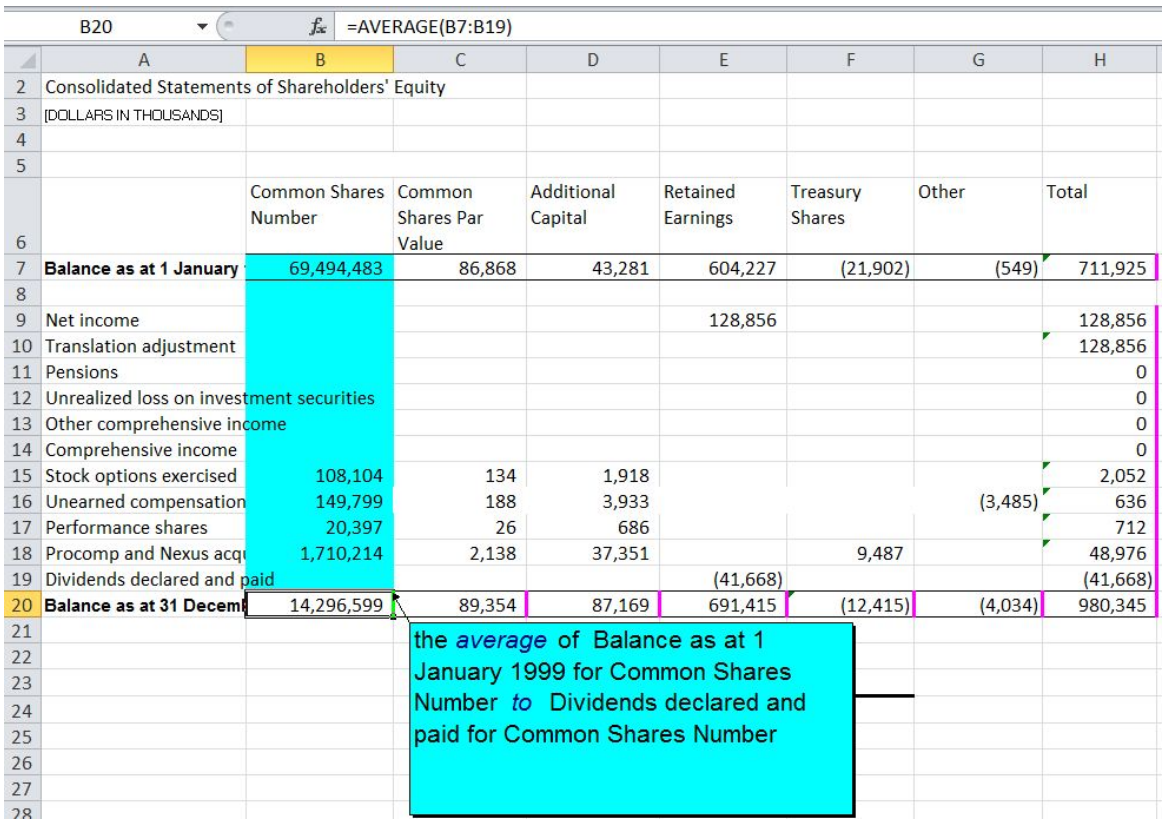


Figure 4: Numeric view of spreadsheet 2 (SP2) used in the debugging task with translation of erroneous cell B20.