AN ABSTRACT OF THE THESIS OF

Mihai Dan for the degree of Master of Science in Computer Science presented on June 12, 2019.

Title: Spreadsheet Explanation Through Table Abstraction

Abstract approved: _____

Martin Erwig

Spreadsheets are a pervasive technology throughout personal and industrial use. Often times, the user is not the author, contributing to a lack of understanding of the purpose and functionality of a spreadsheet. Furthermore, the lack of understanding is a major reason for mistakes in the use and maintenance of spreadsheets.

I present an approach, called explanation sheets, which eases the understanding and maintenance of spreadsheets. I identify the notion of explanation soundness and show that explanation sheets which conform to simple rules of formula convergence provide sound explanations. I also present a practical evaluation of explanation sheets based on samples drawn from widely used spreadsheet corpora and based on a small user study.

In addition to facilitating the understanding of spreadsheets, I describe the process of inferring explanation sheets from a spreadsheet. By means of assessing example spreadsheets, I present a set of inference rules to describe the relationship between a spreadsheet and its explanation.

Spreadsheet Explanation Through Table Abstraction

by

Mihai Dan

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 12, 2019
Commencement June 2019

Master of Science thesis of Mihai Dan presented on June 12, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Mihai Dan, Author

ACKNOWLEDGEMENTS

I would like to thank my adviser Martin Erwig for giving me the opportunity to pursue higher learning while providing me with guidance and help through any problems I had faced. I would also like to thank Jácome Cunha, Danila Fedorin, and Alex Grejuc for their help in shaping explanation sheets.

Lastly, I'd like to thank my parents and friends for giving me encouragement and being there when I needed advice or a helping hand.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Algorithm Page

Chapter 1: Introduction

Studies estimate that software maintenance costs make up 60% [26] to 80% [28] of the total cost of software over its life cycle. Complementary research also shows that software developers spend most of their time trying to understand source code [29].

Spreadsheets are a versatile tool used by companies and individuals with a wide range of applications. However, many of the users are not the original creators of the spreadsheets they work with, incurring overhead cost in trying to parse and understand the content presented by the spreadsheets. Studies focused on spreadsheet usage have shown that 85% of study participants did not create the spreadsheet they frequently use, instead had them handed down from other colleagues or past coworkers [16]. The same study shows that 70% of those who use these legacy spreadsheets have trouble understanding and using spreadsheets, spending hours browsing them.

Previous research has sought out solutions to some of these problems. Kankuzi extensively studied the mental model of spreadsheet users and promotes techniques and tools that reflect these models, such as the abstraction of cell references in formulas to user defined names [23]. Other approaches in this regard have tried to support the maintenance of spreadsheets by systematizing their evolution [7, 11, 8]. However, these techniques require learning and adoption of new languages and tools to be effective, which is likely to cause unnecessary overhead cost. Moreover, applying these techniques to legacy spreadsheets is difficult as they exhibit an inherent structure.

Therefore, I propose an alternative approach to understanding spreadsheets, so-called *explanation sheets*. Explanation sheets are themselves a type of spreadsheet, with certain details abstracted to highlight the underlying computational structure. An explanation sheet does not require the user to adopt new languages, rather uses the inherent structure of a spreadsheet to aid understanding, leading to less difficulty while maintaining and using spreadsheets. Explanation sheets can also be retroactively applied to existing spreadsheets, tackling the issue of misunderstanding of legacy spreadsheets.

When people try to understand a spreadsheet, they commonly face the time-consuming and error-prone task of resolving cell references to make sense of what formulas in a

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	Adams	8.9	40	5	=B3*C3	=B3*1.5*D3	=E3+F3
4	Baker	12.55	35	0	=B4*C4	=B4*1.5*D4	=E4+F4
5	Carlton	9.6	40	2	=B5*C5	=B5*1.5*D5	=E5+F5
6	Daniels	10.2	35	0	=B6*C6	=B6*1.5*D6	=E6+F6
7							
8	Totals		=SUM(C3:C6)	=SUM(D3:D6)	=SUM(E3:E6)	=SUM(F3:F6)	=SUM(G3:G6)

Figure 1.1: Payroll Spreadsheet

spreadsheet do. This task is often exacerbated in spreadsheets which contain a distracting and overwhelming volume of data and by the fact that referenced cells often contain formulas that reference other cells in different parts of the spreadsheet.

This process of "reference chasing" in spreadsheets is necessary, since references are specified using row and column indexes, which do not provide any information about the value or meaning of the referenced cell. Therefore, to make sense of what a reference represents, the user has to scan the spreadsheet for the indicated row and column. This may require a user to scroll a long distance and in many cases jump across multiple cells to make sense of the initial reference.

To address this issue, I introduce *label abstraction* in the spreadsheet explanation model, which replaces raw index references with labeling information when possible. With this abstraction, formula references become more clear, thus making the spreadsheet easier to understand.

Consider Figure 1.1, which shows the formula view of a spreadsheet with payroll information for employees within a company. The spreadsheet was adapted from a study on spreadsheet error detection and correction [4] and is a simplified version of a real-world scenario; one can imagine a similar spreadsheet be used for hundreds of employees and tracking other data such as commission earned. The regular pay for each employee is calculated by multiplying their regular hours and pay rate. Similarly, the overtime pay is calculated by multiplying overtime hours, 1.5, and pay rate. The total is simply calculated by adding the results of regular and overtime pay computations. Columns C-G contain a sum of the respective category in row 8.

Applying label abstraction to the payroll spreadsheet results in the spreadsheet shown

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	Adams	8.9	40	5	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
4	Baker	12.55	35	0	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
5	Carlton	9.6	40	2	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
6	Daniels	10.2	35	0	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
7							
8	Totals		=SUM(Regular Hours)	=SUM(Overtime Hours)	=SUM(Regular Pay)	=SUM(Overtime Pay)	=SUM(Total)

	A	B	C	D	E	F	G
1				Payroll Spreadsheet			
2	Name	Pay Rate	Regular Hours	Overtime Hours	Regular Pay	Overtime Pay	Total
3	*[Adams...Daniels]*	*[8.9...12.55]*	*[35...40]*	*[0...5]*	=Pay Rate*Regular Hours	=Pay Rate*1.5*Overtime Hours	=Regular Pay+Overtime Pay
4							
5	Totals		=SUM(Regular Hours)	=SUM(Overtime Hours)	=SUM(Regular Pay)	=SUM(Overtime Pay)	=SUM(Total)

Figure 1.2: *Top*: Payroll Spreadsheet with Label Abstraction; *Bottom*: Explanation Sheet for the Payroll Spreadsheet, demonstrating label abstraction and a zoom.

in Figure 1.2. For example, the original formula to compute overtime pay for Adams is

```
=B3*1.5*D3
```

where cell B3 refers to the Pay Rate and cell D3 refers to the Overtime Hours. The equivalent formula in the explanation sheet is

```
=Pay Rate*1.5*Overtime Hours
```

Label abstraction increases readability by giving context to computation, especially as formula complexity increases. Labels are more expressive than cell references in providing meaning for a formula, removing the need for reference chasing.

We can observe that while the values in rows 3-6 differ, the formulas with labels are all identical. Such a pattern occurs quite frequently. The different values in the different rows do not contribute much to the understanding of the represented computation. Furthermore, the redundancy that results from the repetition is rather distracting.

To address this shortcoming, I introduce the concept of a *zoom*, which is the result of compressing sections of a spreadsheet with similar content. This compression can be observed in Figure 1.2, where rows 3-6 from the original spreadsheet have been compressed into one row. Such compressed rows contain two kinds of information. First, columns (such as E-G) that contain in the uncompressed sheet a single repeated formula contain just that single formula. Second, columns (such as A-D) with different values in different

rows contain a range that captures all values found in the respective column. The same zoom compression technique can, of course, be applied to repeated columns.

It is important to note that both transformations preserve the essential structure and key computing elements of the spreadsheet. This preservation is intentional and significant to explaining spreadsheets, since users will already be familiar with the structure. A spreadsheet that is the result of label abstraction and zoom compression is called an *explanation sheet*.

This thesis makes the following contributions.

1. A new approach to facilitating understanding of spreadsheets through the concept of explanation sheets, discussed in Chapter 3, along with principles observed through the process of creating explanation sheets which can be applied to explanations in other domains.

2. A study focused on assessing the expressiveness of explanation sheets, as well as the applicability to real-world examples, discussed in Chapter 4.

3. A set of inference rules used to automate the process of creating explanation sheets, discussed in Chapter 5.

I will discuss related works in Chapter 2 and provide a comparison with explanation sheets in Chapter 6. Lastly, I discuss conclusions drawn from this thesis and potential future work in Chapter 7.

Chapter 2: Literature Review

In this chapter, I introduce and describe research related to explanation sheets. The related works mostly reside within the spreadsheet domain, however there are aspects of explanations in other domains which can be applied to spreadsheets. A comparison to explanation sheets is provided in Chapter 6.

The most prominent feature expressed by explanation sheets and not found in the related works is the facilitating of spreadsheet explanation while retaining the original structure of the spreadsheet. While these works, as well as explanation sheets, provide an alternate representation of the computational structure found within a spreadsheet, explanation sheets play on the user familiarity of the spreadsheet form. This structure preservation is an essential key to providing good explanations as it allows the user to remain in the same context as the original spreadsheet and create a mental mapping of the computational structure.

2.1 Calculation View

Sarkar et al. describe an approach to preventing spreadsheet errors by presenting the underlying computational structure to a user [27]. While spreadsheets are a great tool for displaying data, they come with the cost of abstracting away from the computation that created the data. For example, a user may see the value `42` in cell `E7`, but the underlying formula is `B7*C7+D7`. This aspect makes spreadsheets harder to understand, debug or explain.

While this research does not directly aim at providing explanations, it offers a more concrete representation of the computational structure of the spreadsheet. The research focuses on capturing this structure in a so-called Calculation View, presented to the user in addition to the original spreadsheet. The Calculation View can be used to set range assignments to specific formulas, define and reference names for cells, and view calculations more directly. Figure 2.1 shows an example of cell naming and range assignments in the Calculation View. While some of the functionality presented by the

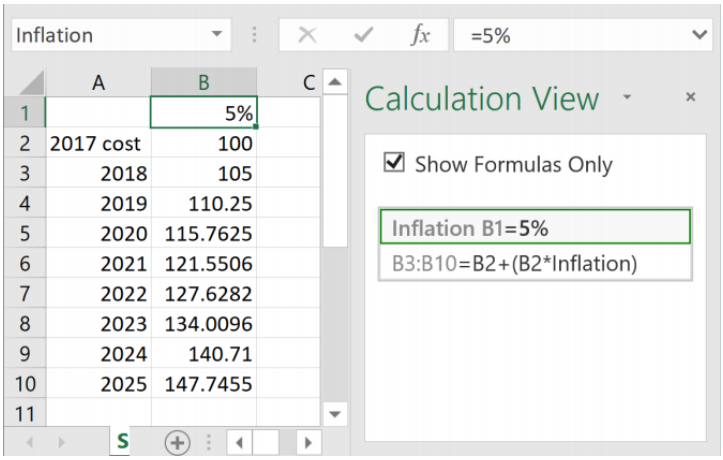


Figure 2.1: Calculation View

Calculation View already exists in spreadsheets, it is hard to perform with the current tools such as Excel.

The Calculation View presents the user with an alternative representation of the spreadsheet, which can often be useful when dealing with complicated formulas and large sheets. While formulas can be viewed in the traditional Excel representation, it is more verbose and difficult to parse as spreadsheet complexity and size increase. This system encapsulates the computation in a potentially more usable interface. By allowing user-defined names in the Calculation View, this approach provides the user with a familiar representation of their data. A user may also notice reoccurring patterns throughout their spreadsheet structure that would be otherwise missed with the traditional Excel representation. These factors can play a significant role in understanding the computational structure of a spreadsheet, thus helping to prevent errors.

The Calculation View is a preventive approach to dealing with spreadsheet errors, rather than trying to fix them after they have happened. The preemptive approach has the potential to prevent time-consuming errors from occurring while editing or creating spreadsheets. When tested in a user study, the Calculation View scored better than the traditional Excel formula view in task efficiency while creating and debugging a spreadsheet, cognitive load, and self-efficacy. Sarkar et al. showed that the Calculation View helps users better understand spreadsheets, even if they were not the original author.

2.2 A Domain Terms Visualization Tool for Spreadsheets

Kankuzi and Sajaniemi introduced a visualization tool for spreadsheets intended to relieve users from spreadsheet details and allow them to focus on the application domain [21]. In earlier research, Kankuzi and Sajaniemi discovered that spreadsheet authors have (at least) three mental models of a spreadsheet: the real-world model, which is comprised of general knowledge, the domain model, which contains knowledge about the problem domain, and the spreadsheet model, which contains knowledge of the expressions and data relationships in a spreadsheet [20].

The tool presented by the authors aims to ease the mapping between the problem domain and the spreadsheet by injecting the domain narrative into the spreadsheet. The domain narrative is inferred using the labels found throughout the spreadsheet.

The tool achieves this abstraction through a series of five steps. First, the tool displays formulas with cell references abstracted to label information in a box below the original formula cell. As shown in Figure 2.2, the formula in cell `C9`, originally `SUM(C5:C8)`, is shown as `SUM(Jan | James Bourne ... Jan | Jasmine Hunt)`, with column and row names being separated by "|". Second, a change in the domain narrative, such as renaming a label, is automatically reflected in the spreadsheet visualization. Third, in an effort to aid recognition of related cells, any cells referenced in a formula are automatically highlighted such that their background color matches that of the formula, also shown in Figure 2.2.

Fourth, all formula cells are marked with a pink right border to help distinguish between plain text cells and formula cells. The pink highlighting can also give a general overview of the computational structure found in the spreadsheet. Fifth, the tool is superimposed on the spreadsheet display, instead of existing as a separate entity. This feature alleviates the need to consult a separate visualization and determine its correspondence to the spreadsheet.

This tool was implemented as an add-on to Microsoft Excel in order to take advantage of users familiarity with existing tools. The tool has potential in aiding a user while creating or editing a spreadsheet, by highlighting the computational flow and underlying structure of the spreadsheet in terms derived from the user-created domain narrative. Leveraging the familiarity expressed through the visualization, the tool also has the potential to aid with spreadsheet comprehension. However, this aspect was not assessed

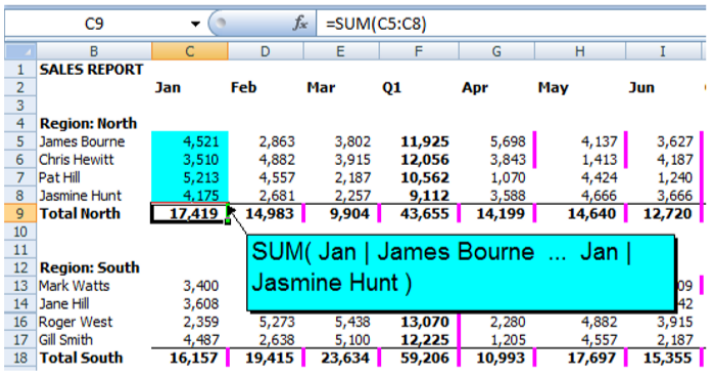


Figure 2.2: Visualization of a Sales Report spreadsheet.

in the authors' research.

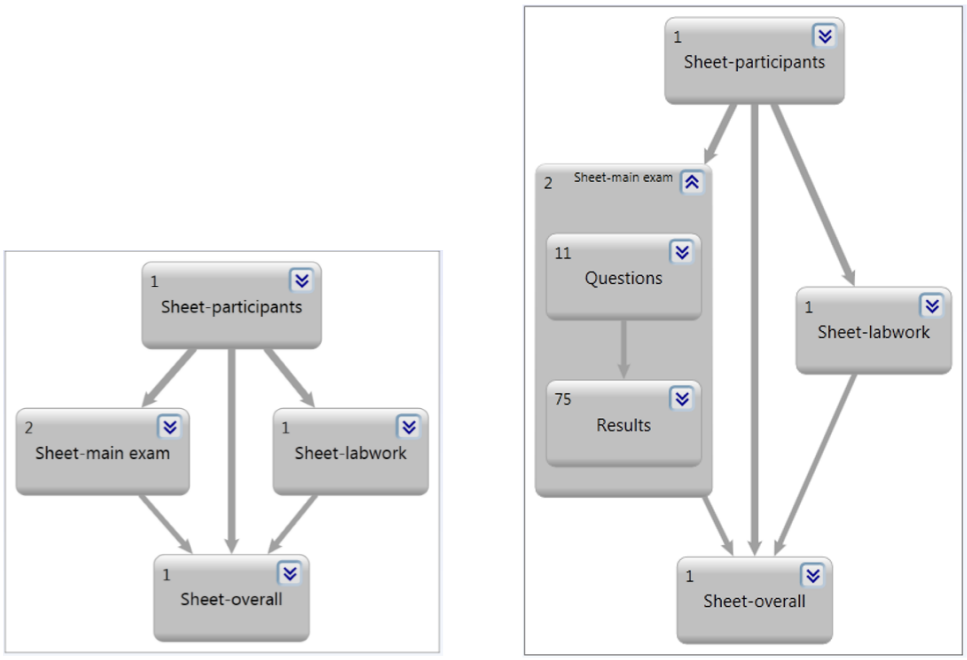
2.3 Visualizing Spreadsheets using Dataflow Diagrams

Hermans et al. describe an alternative approach to aid the understanding of spreadsheets by using data-flow diagrams [17]. The authors were able to discern how to best represent a spreadsheet in a compact and easy-to-understand way by studying the problems and information needs of professional spreadsheet users via data-flow diagrams [18].

Data-flow diagrams model how data moves from one process to another, as well as display the relationships amongst processes and data. The authors describe three different data-flow views: global, worksheet, and formula. The global view shows worksheets within a spreadsheet and the dependencies between them. An example of this can be observed in Figure 2.3a. An arrow from worksheet A to worksheet B indicates that a formula in B refers to a cell found in A.

The authors noticed that data tends to be segmented into data blocks within a worksheet, usually separated by empty space. The worksheet view shows data blocks within a worksheet and the dependencies between them. An example of this view can be observed in Figure 2.3b, where the worksheet *main exam* was expanded to worksheet view. The formula view explicitly shows the relationship between a formula and all cells it depends on. Each formula in the worksheet can be expanded to this view.

The data-flow diagrams are created through a series of steps, using the spreadsheet structure and labeling information. The labeling information is used to create a mapping



(a) Global view for exam spreadsheet. (b) Worksheet View of *main exam*.

Figure 2.3: Example Data Flow Diagrams

from cell references to associated label names, later used for abstracting away from cell references to names familiar to the user. The spreadsheet structure is used to infer the relationships between different entities found within the spreadsheet, more specifically cells, formulas, data blocks, and worksheets, as well as create data-flow diagrams. The cell references found in the data-flow diagrams are then replaced with their corresponding labels, creating a standalone representation of the computational flow of the spreadsheet.

The authors conducted a study to evaluate the usefulness of data-flow visualization [18] and found that it has potential to help users understand spreadsheets more easily than just using the traditional Excel tools. The study also concluded that data-flow diagrams can be used to observe the computational structure of the spreadsheet, without having to click on each individual cell to determine dependencies. On top of helping understand spreadsheets, data-flow diagrams could also be used to spot mistakes in spreadsheets through observing the various dependencies between spreadsheet entities.

2.4 Excel Extensions

Microsoft Excel is a widely used interface for creating, editing, and visualizing spreadsheets. Due to its ubiquity, users are often familiar with the interface and find it easier to use compared to other platforms [15]. Therefore, it is reasonable for an application with the purpose of aiding understanding to be implemented as an extension to Excel.

Canteiro and Cunha present SpreadsheetDoc, an Excel add-in aimed at creating documentation to improve spreadsheet maintenance and transition between several authors [5]. The application allows users to identify input and output cells, document ranges or sections of a spreadsheet, as well as describe computation that occurs in formula cells. SpreadsheetDoc does not directly alter the spreadsheet structure to provide its functionality, but is instead integrated into the Excel layout.

Through SpeadsheetDoc, spreadsheet authors are given the ability to document their computational flow and intended purpose, which could be helpful for future users. Furthermore, the documentation provided by SpreadsheetDoc has the potential to save users time in understanding the design and functionality of a spreadsheet.

Amalfitano et al. introduce a tool, named EXACT, intended to support the comprehension of spreadsheets heavily dependent on Visual Basic for Applications (VBA) macros [3]. VBA is an event-driven programming language that extends Excel's built-in functionality to perform more complex calculations. Spreadsheets are created with the intent of encoding domain-specific policies and allowing a simple structure to provide complex functionality. To this end, many users create VBA macros to accomplish various specific tasks. Similarly to spreadsheets, VBA programs increase in complexity over time and as they evolve to accommodate to the problem domain.

EXACT has the ability to extract information about the computation existing within a spreadsheet, as well as dependencies between cells, and expose it in an interactive interface. This tool has the potential to explain the relationships between a spreadsheet and its corresponding computational structure, specifically computation achieved through VBA. While it does not directly explain the purpose of the data, this tool could serve as a means of documentation and facilitate the understanding of computation.

2.5 Explanations in Other Domains

Explanations have been sought after in other domains as well. Walkingshaw and Erwig present a domain-specific embedded language (DSEL) that models complex casual relationships between events and functions [30]. Cause and effect have been extensively studied by philosophers for over 2000 years and are fundamental concepts upon which science and our concept of understanding and explanation rely on. The DSEL allows for the creation of visual representations of casual relationships to aid researchers examine causation, as well as model related scenarios with respective outcomes. Furthermore, the DSEL allows users to extend the notation and add constructs specific to the problem they are facing.

The DSEL is derived from neuron diagrams, a visual notation used in causation explanation and research. Since researchers in this field are already familiar with neuron diagrams, the transition process to using the DSEL should be fairly straightforward. The DSEL can be a powerful explanatory tool because it provides an alternative representation of complex relationships, and does so by using artifacts from the domain, with which an intended user should be familiar with.

Erwig and Walkingshaw employ explanation-oriented programming to create Probula, a domain-specific, visual language for explaining probabilistic reasoning [13]. Explanation-oriented programming refers to the paradigm of assessing not just *what* result a program produced, but *how* it achieved the result. Probula uses story telling techniques to guide the user through a probabilistic reasoning problem, from some initial state to a final state. Furthermore, the language can create equivalent explanations from one explanation instance, in the event that the user is not satisfied or does not understand the problem with the current explanation visualization.

Similar to the previous domain-specific language, Probula can be a powerful explanatory tool as it creates an alternative representation of complicated probabilistic reasoning problems. Additionally, Probula presents the problem in a sequential and easily digestible format, such that users should not be overwhelmed with information. Moreover, Probula provides additional explanatory value with the ability to create equivalent explanations when the one presented does not suffice.

In order to further develop the paradigm of explanation-oriented programs, Erwig and Walkingshaw present a visual language for explaining strategies in game theory

[12]. The visual language is based off of the normal form representation of game theory, which is well known in the economical and social sciences domains. The normal-form is a matrix representation of possible outcomes dependent on a finite set of possible choices available to the players of a game. Additionally, the visual language employs the notion of game traces, which represent a sequence of decisions that result in a certain state.

The language design focuses on the cognitive dimension of traceability, which measures a notations ability to represent and to relate to its semantics. A notation exhibits a high degree of traceability when it maps closely to the semantics it is derived from. Moreover, a notation with high traceability will provide insight in understanding not just what decision was made, but what steps were taken to get there.

This language can be used as an explanatory tool for game theory as it uses notation familiar to a user, and provides a sequential flow of how the end result was achieved.

Chapter 3: Explanation Language for Spreadsheets

The findings and experiments described in Chapters 3 and 4 were conducted with the help of my co-authors in [6]. In Sections 3.1 and 3.3 we define spreadsheets and explanation sheets, respectively. In 3.2 we discuss the principles observed that should guide the development of explanations. In Section 3.4 we introduce a relationship between the two languages that captures the notion of explainability.

3.1 Spreadsheets

A spreadsheet is a rectangular grid of cells that contain formulas and values. We can represent spreadsheets $s \in S$ as partial mappings from addresses $A = \mathbb{N} \times \mathbb{N}$ to formulas. Formulas are either plain values ($v \in \mathit{Val}$), application of operations (ω) to other formulas, or references to cells ($a \in A$). The set of values includes an empty value $\sqcup$, which allows us to distinguish undefined cells that are part of the spreadsheet from undefined cells on the outside.

$$f \in \mathit{Fml} \quad ::= \quad v \mid \omega(f, \ldots, f) \mid a$$

Abstracting from the contents of cells, we use the type constructor $\boxplus_\alpha = A \to \alpha$ to represent sheets indexed by addresses and storing values of type α. A spreadsheet $\boxplus_{\mathit{Fml}}$ is then simply a sheet of formulas. Formulas are evaluated to values *Val*, and we call the result of the evaluation of a spreadsheet a *value sheet*, which is a sheet of values $\boxplus_{\mathit{Val}}$. The semantics of a spreadsheet language are given by a function $[\![\cdot]\!] : \boxplus_{\mathit{Fml}} \to \boxplus_{\mathit{Val}}$ that maps spreadsheets to value sheets.

3.2 Explanation Principles

Informed in part by previous work on explanations [10, 30], but also by the experience during the creation of spreadsheet explanations, we have identified a number of general principles that we believe should guide the development of explanations. These principles are presented in a separate section, so that they can also inform the design of explanations

for other languages.

In general, explanations can take on many different forms. Taking a programming language perspective, we have found it useful to conceptualize an explanation system as consisting of two languages: (A) the language whose programs are to be explained and (B) the language in which explanations are expressed. We call the former the *subject language* and the latter the *explanation language*. Correspondingly, we call programs of the subject language *subject programs* (or *programs* for short), and we call programs of the explanation language *explanation programs* (or *explanations* for short). In the context of spreadsheet explanations, this means that we refer to spreadsheets sometimes as subject (spread)sheets and that we call their explanations explanation (spread)sheets.

The following four principles were derived from the work on explanation sheets, but can and should be applied to the design of explanation languages in other domains.

(1) **Structure Preservation**. *An explanation language should retain key subject language structures.* Subject language structures can provide easy access to an explanation, since users are already familiar with these structures. Moreover, reused structures facilitate the alignment of explanations with subject programs.

(2) **Abstraction**. *An explanation language should aim at high-level descriptions that abstract from details of the subject language.* Abstraction makes explanations faster to absorb. It also allows explanations to provide summaries of subject programs.

(3) **Partiality**. *An explanation language should support partial explanations.* In other words, an explanation should not be required to cover all of a subject program. Partiality supports a gentle-slope approach to explanations, since it allows the incremental construction of more and more complete explanations. Moreover, partiality allows one to ignore parts that cannot be explained (because they are not understood) or are trivial or unimportant.

(4) **Compositionality**. *An explanation language should support constructing bigger explanations from smaller ones.* This requires composition operators for explanations. Compositionality supports the systematic construction of explanations and the reuse of explanations. Together with partiality, compositionality supports the distributed creation of explanations by different people who understand different parts of the subject program.

Note that the first two principles are sometimes in conflict with each other because abstraction calls for ignoring structures in the subject language. Moreover, there is a trade-off between the benefits that can be gained from abstraction and the explicitness and simplicity offered by a detailed and concrete description of a computation. As illustrated in [10], this problem can be addressed by providing for one subject program a set of explanations that are related and can be explored in a systematic way.

3.3 Explanation Sheets

Following the structure preservation principle from Section 3.2, we design an explanation of a spreadsheet to be itself a kind of spreadsheet, a so-called *explanation sheet* that stores formula explanations in cells. Following the abstraction principle, an explanation sheet should abstract from some of the details of the spreadsheet and should thus be smaller in size. We therefore need a definition that allows one cell in an explanation sheet to explain many cells in a spreadsheet.

To explain formulas, we need explanations for values, references, and expressions built by operations applied to other formulas. Since the values in a spreadsheet are either numbers or strings, which are both ordered domains, we can summarize a set of values from different cells by a *value range* ($\bar{v} \in \overline{Val} = Val \times Val$). Similarly, a set of references can be summarized by an *address range* $\bar{a} \in \overline{A} = A \times A$). The two addresses of a range represent opposing corners of a rectangular area, and the region denoted by a range is given by the function $\rho : \overline{A} \to \mathcal{A}$, which is defined as follows ($\downarrow/\uparrow$ compute the minimum/maximum of two numbers).

$$\rho(((x_1, y_1), (x_2, y_2)) = \{(x, y) \mid x_1{\downarrow}x_2 \leq x \leq x_1{\uparrow}x_2 \\ \wedge\ y_1{\downarrow}y_2 \leq y \leq y_1{\uparrow}y_2\}$$

Since labels have been successfully employed in the past for annotating and explaining cells [9, 2, 24, 22], we use labels to explain a set of references by one or two (row and/or column) labels $\ell \in Lab = Val \cup Val \times Val$. More precisely, we can define a binary relationship $\mathcal{L} \subseteq A \times A$ where $(a, a') \in \mathcal{L}$ whenever the value $S(a)$ in cell a is considered to be a label for cell a'. Moreover, we can define a partial *labeling function* $L : A \to Lab$ which identities values as labels for cells.

	A	B	C	D	E
1			Earnings Per Share ($)		
2		Quarter 1	Quarter 2	Quarter 3	Quarter 4
3	Company A	4.25	6.32	4.96	3.22
4	Company B	7.98	8.25	6.46	7.95
5	Company C	2.55	1.36	4.26	3.38

Figure 3.1: Earnings Per Share Spreadsheet

$$L(a') = \begin{cases} S(a) & \text{if } \mathcal{L}^{-1}(a') = \{a\} \\ (S(a_1), S(a_2)) & \text{if } \mathcal{L}^{-1}(a') = \{a_1, a_2\} \end{cases}$$

The first case of the labeling function represents examples in which cell a' is labeled by a single value $S(a)$. This case occurs when a cell is labeled by either a column or row label. As a concrete example, recall the payroll spreadsheet in Figure 1.1 in Chapter 1. Applying the labeling function to cell `A3` yields the following.

$$L(\texttt{A3}) = S(\texttt{A2}) = \texttt{"Name"}$$

Alternatively stated, the cell `A3` is labeled by the value of cell `A2`, which is `"Name"`.

The second case of the labeling function represents examples in which cell a' is labeled by a pair of values $(S(a_1), S(a_2))$. This case occurs when a cell is labeled by both a column and row label. Consider a simplified version of an Earnings Per Share (EPS) spreadsheet depicting the EPS for three companies over four quarters, shown in Figure 3.1. Applying the labeling function to cell `D4` yields the following.

$$L(\texttt{D4}) = (S(\texttt{D2}), S(\texttt{A4})) = (\texttt{"Quarter 3"}, \texttt{"Company B"})$$

Alternatively stated, the cell `D4` is labeled by the values of cells `D2` and `A4`, which are `"Quarter 3"` and `"Company B"`, respectively. Lastly, $L(a')$ is undefined whenever $\mathcal{L}^{-1}(a') = \varnothing$. This only occurs when there are no cells which label cell a'.

We explain sets of formulas that share a common structure and differ only in their references by a formula with labels abstracting the references. Finally, we represent unexplained areas using the special value $\perp$ ("unexplained"), which allows us to reduce potentially large chunks of a spreadsheet to a single row, column, or cell.

Thus we obtain the following definition of explanation formulas and the derived notion of explanation sheets $\boxplus_{Xpl}$.

$$x \in Xpl \quad ::= \quad v \mid \bar{v} \mid a \mid \bar{a} \mid \ell \mid \omega(x, \ldots, x) \mid \bot$$

The structure preservation embraced by $\boxplus_{Xpl}$ aligns the structure and composition of an explanation sheet with that of the explained spreadsheet.

3.4 Explaining Spreadsheets with Explanation Sheets

The principal idea of explaining spreadsheets with explanations sheets is to decompose the spreadsheet into different rectangular areas and then associate with each area a smaller area of the explanation sheet. In the following we formalize this idea.

A spreadsheet explanation is captured by a so-called *zoom* $X \overset{\eta}{\lhd} S$, which consists of an explanation sheet X, a spreadsheet S, and a total function $\eta \subseteq A \times A$ that embeds the spreadsheet into the explanation, that is, $dom(\eta) = dom(S) \wedge rng(\eta) = dom(X)$. We also require that the explanation formulas in the zoom explain the formulas of the spreadsheet. The totality of η ensures that every cell in S is covered by a cell in X. We don't require zooms to be surjective to allow for "filler cells" in the explanation sheets that serve no other purpose than to turn explanation sheets into rectangular areas. In many cases η will actually be surjective, and then it follows that $|dom(S)| \geq |dom(X)|$.

The purpose of zooms is to explain a number of similar cells by one cell. Specifically, when $\eta^{-1}(a) = \{a_1, \ldots, a_k\}$, we use cell a to summarize, or explain, all the cells $a_1, \ldots, a_k$. We can formalize this idea through the notion of *formula explanation*, which is defined as a binary relationship $x \lhd f$ that says an explanation formula x explains a spreadsheet formula f, see Figure 3.2.

$a \lhd a_1, \ldots, a \lhd a_k$

Rule Value states that any value explains itself. Rule Value Range requires that a value v explained by a value range (v_1, v_2) is within the lower-bounding value v_1 and upper-bounding value v_2. Similarly, Rule Address Range requires that a cell address a explained by an address range (a_1, a_2) resides within the lower-bounding address a_1 and the upper-bounding address a_2.

Rule Formula requires that the explanation and explained formulas have the same

$$\textsc{Value}\quad v \lhd v \qquad \textsc{Value Range}\quad \frac{v_1 \leq v \leq v_2}{(v_1, v_2) \lhd v} \qquad \textsc{Address Range}\quad \frac{a_1 \leq a \leq a_2}{(a_1, a_2) \lhd a}$$

$$\textsc{Formula}\quad \frac{x_1 \lhd f_1 \quad \ldots \quad x_n \lhd f_n}{\omega(x_1, \ldots, x_n) \lhd \omega(f_1, \ldots, f_n)} \qquad \textsc{Label}\quad \frac{L(a) = \ell}{\ell \lhd a}$$

$$\textsc{Empty Value}\quad (v_1, v_2) \lhd \sqcup \qquad \textsc{Empty Formula}\quad \omega(x_1, \ldots, x_n) \lhd \sqcup \qquad \textsc{Unexplained}\quad \bot \lhd f$$

Figure 3.2: Formula Explanations

structure. The premise in the rule Label ensures that a label exists. The rules Empty Value and Empty Formula allow empty values to be explained by ranges and formulas, respectively, and the rule Unexplained allows any formula to be left unexplained. The necessity of the rules Empty Value and Empty Formula is evident when compressing rows or columns which contain empty cells amongst values. This pattern can be observed when data is missing for certain entries or when empty values are purposely used for presentation.

For a zoom $X \overset{\eta}{\lhd} S$ we require that every formula in X explain all formulas in S that are mapped to it, that is:

$$\forall a' \in \mathit{dom}(X), \forall (a, a') \in \eta : X(a') \lhd S(a)$$

Based on the semantics of spreadsheets, we can define the semantics for explanation sheets as follows. Since explanation formulas include ranges of values and addresses, they will generally evaluate to ranges of values.[1] To resolve references the semantics need access to the explanation sheet. Since we also have to account for $\bot$ formulas, the semantics of explanation formulas is of type $[\![\cdot]\!] : \mathit{Xpl} \to \boxplus_{\mathit{Xpl}} \to \overline{\mathit{Val}} \cup \{\bot\}$. The definition is shown in Figure 3.3. We use the function $\updownarrow V = (\downarrow V, \uparrow V)$ to compute the minimally enclosing range for a set of values V. We also use it for addresses.

The semantics of explanation sheets are then given by the following function

[1] A single value v can always be represented by a trivial range (v, v).

$$[\![v]\!]_X = (v, v) \qquad [\![\bar{v}]\!]_X = \bar{v} \qquad [\![a]\!]_X = [\![X(a)]\!]_X \qquad [\![\bar{a}]\!]_X = \updownarrow\{[\![X(a)]\!]_X \mid a \in \rho(\bar{a})\}$$

$$[\![\ell]\!]_X = \updownarrow L^{-1}(\ell) \qquad \frac{[\![x_i]\!]_X = (v_i^1, v_i^2) \qquad v_i^1 \leq v_i \leq v_i^2}{[\![\omega(x_1, \ldots, x_n)]\!]_X = \updownarrow\{[\![\omega(v_1, \ldots, v_n)]\!]_X\}} \qquad [\![\bot]\!]_X = \bot$$

Figure 3.3: Explanation Semantics

$[\![\cdot]\!] : \boxplus_{Xpl} \to \boxplus_{\overline{Val} \cup \{\bot\}}$.

$$[\![X]\!] = \{(a, \bar{v}_\bot) \mid (a, x) \in X \wedge [\![x]\!]_X = \bar{v}_\bot\}$$

Note that the semantics also depends on the underlying subject sheet S and a labeling relationship $\mathcal{L}$ to resolve labels (ℓ) in explanation formulas.

Next we introduce the notion of *zoom soundness*. This is essentially the $\lhd$ relationship for value ranges and values applied to whole sheets that are connected via a function η. We say that an explanation X is *sound* for a spreadsheet S under η if $[\![X]\!] \overset{\eta}{\lhd} [\![S]\!]$. This relationship captures the notion that an explanation sheet X covers all cases of the explained spreadsheet S and that the evaluation of S holds no surprises.

Now we can present our main result, which says that zooms are sound.

Theorem 1 (Soundness) $X \overset{\eta}{\lhd} S \implies [\![X]\!] \overset{\eta}{\lhd} [\![S]\!]$

Note that for any spreadsheet S we always can find a trivial explanation through the zoom $S \overset{id}{\lhd} S$,[2] which means that any spreadsheet trivially explains itself. However, such a zoom is not really useful, since it does not achieve any abstraction. Employing a straightforward ordering on zooms based on the size of the explanation sheet, we can define that a zoom $X_1 \overset{\eta_1}{\lhd} S$ achieves a *higher explanatory reduction* than a zoom $X_2 \overset{\eta_2}{\lhd} S$ if $|dom(X_1)| < |dom(X_2)|$. Note that this relationship defines a partial order, and there isn't necessarily a single smallest explanation.

[2]Here *id* denotes the identity function.

Chapter 4: Artifact Evaluation and User Study

4.1 Artifact Evaluation

We employed real-world spreadsheets from two spreadsheet corpora in the design and evaluation of our approach. We used a 2-step process comprised of refining a preliminary definition of explanation sheets, discussed in Section 4.1.1, and evaluating the resulting definition, discussed in Section 4.1.2.

4.1.1 Guiding the Design of Explanations

The design of our spreadsheet explanations was guided in part by real-world example spreadsheets. With a preliminary definition of explanation formulas and zooms we set out to explain existing spreadsheets from two repositories.

Specifically, we analyzed 20 randomly selected spreadsheets from [25], which are generally well-designed spreadsheets created by experts, plus 20 randomly selected spreadsheets from the Enron spreadsheets corpus [16], which includes more than 15,000 spreadsheets.

We manually created an explanation spreadsheet for each of the 40 spreadsheets. The main purpose of this exercise was to see whether explanation formulas are general enough or maybe even unnecessarily too general and whether our definition of zooms worked as anticipated.

During this testing phase, the explanation model was revised several times. Specifically, we removed a number of explanation formulas that we originally thought to be useful because the anticipated situations did either not occur at all or only once or twice and thus were not justifying a more elaborate notion of explanation formulas. We also simplified the definition of zooms, which originally were defined recursively allowing for nested zooms to explain nested loop structures in spreadsheets. But since such a nested loop structure occurred only in one of the selected examples, we traded the more general definition for a simpler one.

4.1.2 Applicability and Impact

In order to analyze the applicability and effect of spreadsheet explanations, we randomly selected a new set of spreadsheets from the two sources and used 41 worksheets from 36 different spreadsheets.

We observed that 78% (32/41) of worksheets contained areas that could be compressed and explained by zooms (20 worksheets contained row zooms, 10 worksheets contained column zooms, and 2 worksheets contained both row and column zooms). Ignoring three huge spreadsheets that were basically used as databases and that would lead to a misleadingly high average, the average size compression achieved by zooms was 64%, and the minimal/maximal achieved compression was 25%/99%.

N	**Error Type**
4	Value (value in S is mapped to a different value in X)
7	Range (value in S is not covered by the range in X)
4	Reference (undefined references in mappings)
1	Label A (value of the label does not match the value in S)
8	Label B (labeling does not correctly abstract reference)

Figure 4.1: Errors found by the Explanation Checker

To verify that the generated explanations were correct, we developed an explanation checker that implements the definitions from Section 3.4, specifically Figure 3.2. This explanation checker was applied to all explanations and helped to correct at least one error in 24 of the 41 manually created explanation sheets. Most of the errors were due to simple typos, but the checker also found several incorrect range mappings in zooms and other reference errors. A summary of the kinds of errors that were detected is shown in Figure 4.1.

4.2 User Study

In this section we describe an initial empirical evaluation we performed to assess the extent to which explanation spreadsheets help facilitate understanding of computation in spreadsheets. We describe the design of the user study in Section 4.2.1. The results are shown in Section 4.2.2 and discussed in Section 4.2.3.

4.2.1 Design

Spreadsheets. For this study, we have semi-randomly selected 4 spreadsheets from 3 different sources. The selection process was semi-random in the sense that after randomly choosing a spreadsheet from the source we analyzed it to verify if we could work with it. The process of analyzing and choosing appropriate spreadsheets is discussed in the following paragraph. If the spreadsheet was unfit for our purpose, then we would randomly select another one from the set. The random mechanism used was an Excel formula on the names of the spreadsheets.

We selected two spreadsheets from the EUSES spreadsheet corpus [14] which has been extensively used for evaluation of spreadsheet techniques and tools. In this case several spreadsheets were selected until two appropriate spreadsheets were found. The first two spreadsheets were discarded as they contained labels, but were otherwise empty, namely `inventory/Inventory-ChangeForm.xls` and `inventory/MoveEquip.xls`. Also, some were simple databases, which again, are not interesting. Obviously, we can work with such spreadsheets, but without computations or formulas they are of little interest for this study. We also obtained a couple of spreadsheets, namely `grades/i01.0203.xls` and `financial/ExampleFinancial#A8079.xls`, that have references to unlabeled ranges of cells which we cannot yet cope with. We discarded `inventory/poult_budg.XLS` as it is too big to create its explanation by hand. Finally, the first worksheet from spreadsheet `database/ProQuestUsage2002-03.xls` and the only worksheet with content from spreadsheet `homework/G140W04.xls` have no restrictions and were used for the study. The former will be called *database* and the latter *homework*.

We initially selected another spreadsheet from [25], namely `C9/Butson.xls`. However, this spreadsheet was discarded as it contains references to unlabeled ranges of cells. We then randomly selected `C7/DataSets/Applicants.xls`, but since it contains no formulas as it is mostly a database, it was also discarded. Finally, the random selection elected `C9/Data.xls`, from which we used the first worksheet, removing the remaining one from the workbook so we could direct the participants focus to be in line with the scope of the experiment. We will call this spreadsheet the *book* spreadsheet.

Finally, we have selected a spreadsheet from the Enron corpus, namely `john_arnold__15184__Daily_Vega.xlsx`. This first spreadsheet has no limitations and so we used its first worksheet, which required adding the final worksheet as it was a

				Spreadsheets Created			
ID	**Gender**	**Age**	**Occupation**	**High school**	**College**	**Work**	**Personal**
Group 1							
11	Male	29-31	researcher	2-3	5-10	>20	>20
12	Male	>32	post-doc	>5	>10	>20	>30
3	Female	26-28	phd student	6	1	0	5
14	Male	29-31	researcher	20	5	20	15
17	Male	29-31	researcher	>20	>10	>10	>10
Group 2							
2	Male	>32	researcher	0	0	>100	>100
5	Male	23-25	sw. developer	>10	>10	>100	>100
6	Female	>32	post-doc	many	many	many	few
7	Male	26-28	phd student	0	4-5	>30	>30
8	Male	29-31	researcher	5	5	0	20

Table 4.1: Data about the study participants.

dependency. We will call this spreadsheet the *enron* spreadsheet.

With these spreadsheets we try to have a representative set of real-world spreadsheets. Indeed, we are using spreadsheets from the Enron corpus, a former energy company, from EUSES, which contains spreadsheets gathered from the internet, and also from a book on modeling spreadsheets, which presents well designed spreadsheets.

Subjects Participants were recruited from two different universities. All the invitations were done via email and participants conducted the study on their own. They are all in the computer science graduate school or are already post-doctoral (referred to as researcher or post-doc in Table 4.1), except one which as already finished his studies.

Most of them are quite experienced in the use of spreadsheets as they declared to have created from a few spreadsheets to more than 100 in different stages of their lives (last four columns in Table 4.1). The total number of participants resulted in 13, which were divided into two groups, as discussed in the following subsection.

Procedure For each spreadsheet, we created the corresponding explanation. We then created Package 1 with four spreadsheets: subject *database*, explanation of *homework*, subject *book*, and explanation of *enron*. We also created Package 2 with four complementary spreadsheets: explanation of *database*, subject *homework*, explanation of *book*,

		average time		average score	
		subject	explanation	subject	explanation
database	Q1	1.3	2.1	2.2	2.4
	Q2	1.1	2.2	3.0	2.8
homework	Q1	3.1	2.9	2.0	2.6
	Q2	2.5	3.7	2.0	1.8
book	Q1	2.1	1.8	3.0	1.0
	Q2	1.0	2.9	2.4	1.4
enron	Q1	3.6	5.4	1.2	1.4
	Q2	6.8	3.3	1.8	2.0

Table 4.2: Average times and scores in the empirical study

and subject *enron*. We randomly assigned Package 1 or Package 2 to participants until we got 6 participants for Group 1 and 7 for the Group 2, so we could have an even distribution (as much as possible).

This distribution of the spreadsheets was meant to have some participants starting with explanation sheets (Package 2) and others with subject spreadsheets (Package 1) so we could eliminate learning effects, if any. We obtained an equal amount of responses from both groups, providing an even distribution of results.

Furthermore, we prepared a one-page tutorial about spreadsheet explanations. The participants were instructed to review the tutorial before answering any questions.

For each of the four spreadsheets we asked two questions (instantiated for each particular case):

Q1 **What** is being calculated in row/column/cell X?

Q2 **How** are the values in row/column/cell X calculated?

4.2.2 Results

We present the results of the study in Table 4.2. For each spreadsheet, we present the average time (in minutes) participants took to answer each of the two questions and the average score of the answers. We scored each answer with a value from 0 to 3, 0 meaning wrong answer and 3 meaning entirely correct.

4.2.3 Discussion

The user evaluation produced mixed results. Explanation sheets led to higher scores in 3 out of the 4 scenarios, with the exception of *book*, which produced significantly lower results. Here we note that the explanation sheet for *book* employed a column header (S) as a label where none was provided by the subject spreadsheet. As the participants had no prior knowledge of explanations, this could have made it hard to infer the meaning of the column reference, thus impacting understandability.

There is no significant difference between the average times it took participants to answer the questions. With the exception of *enron*, participants were able to determine *how* a computation was performed faster using the explanation. However, explanations were only faster at explaining *what* a computation calculated in cases *homework* and *book*.

For spreadsheet *enron*, participants that received the explanation sheet got a better score. Moreover, the participants were also faster at answering the questions presented in the experiment. This was the most difficult spreadsheet, as participants were asked to discern the following formula:

```
IF(
    ISERROR(
        ROUND(
            INDEX(Vega.(Post ID&Ref Period&Vega);
            MATCH(Month;Vega.Ref Period;0);
            3)/1000;
        4
        )
    );
    0;
    ROUND(INDEX(Vega.(Post ID&Ref Period&Vega);
            MATCH(Month;Vega.Ref Period;0);
            3)/1000;
    4
    )
)
```

Interestingly, participants took longer to answer questions for simple spreadsheets using explanations. This can possibly be attributed to the fact that the participants have had extensive experience with spreadsheets, while none with explanations. This also seems to indicate that explanation sheets are probably more useful for complex spreadsheets.

These results indicate the potential of explanation sheets for providing a better understanding of complex spreadsheets, given that a user has some existing knowledge of how explanation sheets abstract data. In their answers to a post-study survey eight out of ten participants said that they found explanation sheets somewhat or very helpful. For the two other participants, they did not make a difference. Moreover, eight of the participants would want to use explanation sheets in the future.

Chapter 5: Explanation Inference

In this chapter, I introduce a set of inference rules to infer explanation sheets from a spreadsheet, discussed in Section 5.1. More specifically, the inference rules infer the total mapping function $\eta \subseteq A \times A$, which maps cells in an subject spreadsheet S to the corresponding cells in a explanation sheet X. The judgment $(\mathtt{a}, \overline{\mathtt{a}}) \in \eta$ states that a cell $\mathtt{a} \in S$ is explained by a cell $\overline{\mathtt{a}} \in X$. Cell addresses in explanation sheets are marked with an over-line (i.e. $\overline{\mathtt{A1}}$) to differentiate from those cells which belong to subject sheets.

The inference rules rely on the assumption that the labeling relationships $\mathcal{L}_S$ and $\mathcal{L}_X$ are given. These labeling relationships could be obtained by using previous work on automatic label inference [1].

Then, in Section 5.2, I present an artifact evaluation which evaluates the correctness of the inference rules on examples of real-world spreadsheets.

5.1 Development of the Inference Rules

Explanation sheets have the potential to aid spreadsheet understanding, but are tedious and error prone to create manually. Therefore, I introduce a set of inference rules to infer explanations from a spreadsheet. These rules rely on the inherent structure and relative positioning of cells in the spreadsheet, as well as the content within each cell. The inference rules describe the binary relationship between some cell addresses $\mathtt{a}$ and $\overline{\mathtt{a}}$, part of the total mapping η that embeds a spreadsheet S into an explanation sheet X.

Consider the example in Figure 5.1, which depicts a simplified version of a gradebook spreadsheet, accompanied by its explanation sheet. Through the notion of formula explanation described in Section 3.4, rows 2 and 3 of the subject sheet S_1 have been compressed to row 2 in the explanation sheet X_1, which is captured by a zoom $X_1 \overset{\eta_1}{\lhd} S_1$, where $\eta_1 = \{(\mathtt{A1}, \overline{\mathtt{A1}}), (\mathtt{A2}, \overline{\mathtt{A2}}), (\mathtt{A3}, \overline{\mathtt{A2}}), (\mathtt{B1}, \overline{\mathtt{B1}}), (\mathtt{B2}, \overline{\mathtt{B2}}), (\mathtt{B3}, \overline{\mathtt{B2}})\}$.

To formulate rules for inferring explanation sheets, we can make several observations that help to establish necessary conditions. Examining the contents of cells $\mathtt{A3}$ and $\overline{\mathtt{A2}}$,

	A	B
1	Name	Grade
2	Aaron	C
3	Bill	A

(a) Subject Sheet S_1

	A	B
1	Name	Grade
2	*[Aaron...Bill]*	*[A...C]*

(b) Explanation Sheet X_1

Figure 5.1: Grade-Book Example

we can observe that $S_1(\texttt{A3}) = \texttt{Bill}$ and $X_1(\overline{\texttt{A2}}) = \texttt{[Aaron...Bill]}$. By using the VALUE RANGE rule from Figure 3.2, we can derive that the explanation formula $X_1(\overline{\texttt{A2}})$ explains the spreadsheet formula $S_1(\texttt{A3})$, since `Bill` is contained within the range `[Aaron...Bill]`. This observation leads to the formulation of the condition $X(\overline{\texttt{a}}) \lhd S(\texttt{a})$, which says the explanation formula in cell $\overline{\texttt{a}}$ must explain the spreadsheet formula in cell `a`.

The following observation relies on the labeling relationships for the subject and explanation sheets, $\mathcal{L}_S$ and $\mathcal{L}_X$, respectively. By looking at the structure of both sheets in Figure 5.1, we can notice that every cell in X_1 shares the same label content with the cell it explains in S_1. For example, the content of the cell that labels `A3` is `Name`, which is the same as the content of the cell that labels $\overline{\texttt{A2}}$. This observation leads to the requirement that there exist addresses `b` and $\overline{\texttt{b}}$ such that `b` labels `a`, $\overline{\texttt{b}}$ labels $\overline{\texttt{a}}$, and the contents of `b` and $\overline{\texttt{b}}$ are identical. Formally, $(\texttt{b}, \texttt{a}) \in \mathcal{L}_S$, $(\overline{\texttt{b}}, \overline{\texttt{a}}) \in \mathcal{L}_X$, and $S(\texttt{b}) = X(\overline{\texttt{b}})$ must hold true for X to explain S.

Thus far, we have only considered cases in which the cell has a label, but this notion does not capture label cells themselves. To this extent, we need a new rule to differentiate between the inference of label cells and those cells which have a label.

According to a case study on best spreadsheet practices, each column or row should contain a single unique label [19]. Based on this assumption, the relationship between label cells in η is quite straightforward. A label cell in the explanation sheet explains a label cell in the subject sheet if they have identical content. This observation leads to the first version of the explanation inference rules, shown in Figure 5.2. Rule IDENTICAL CONTENT captures the relationship between label cells, while rule SAME LABEL captures the relationship between cells that have labels.

The assumption that each label is unique does not hold when assessing real world spreadsheets, as many tend not to follow best practices. To this end, we can eliminate

$$\frac{X(\overline{\mathtt{a}}) \lhd S(\mathtt{a}) \qquad (\mathtt{b},\mathtt{a}) \in \mathcal{L}_S \qquad (\overline{\mathtt{b}},\overline{\mathtt{a}}) \in \mathcal{L}_X \qquad S(\mathtt{b}) = X(\overline{\mathtt{b}})}{(\mathtt{a},\overline{\mathtt{a}}) \in \eta}\ \textsc{Same Label} \qquad \frac{S(\mathtt{a}) = X(\overline{\mathtt{a}})}{(\mathtt{a},\overline{\mathtt{a}}) \in \eta}\ \textsc{Identical Content}$$

Figure 5.2: Explanation Inference, Version 1

	A	B
1	Name	Grade
2	Aaron	C
3	Bill	A
4		
5	Name	Grade
6	Carl	A
7	Dan	B

(a) Subject Sheet S_2

	A	B
1	Name	Grade
2	*[Aaron...Bill]*	*[A...C]*
3		
4	Name	Grade
5	*[Carl...Dan]*	*[A...B]*

(b) Explanation Sheet X_2

Figure 5.3: Modified Grade-Book Example

this assumption by assessing the relative positioning of label cells in the subject sheet to label cells in the explanation sheet.

We can employ a row-wise ordering of addresses when comparing relative positioning of cells in a spreadsheet, which means the row value takes priority over the column value. As a concrete example, a cell with the address `B2` is considered to be smaller than a cell with the address `A3`, but greater than a cell with the address `C1`, that is `A3` $>$ `B2` $>$ `C1`. This was an arbitrary decision inspired by the examples studied in Section 4.1.

Consider the modified grade-book spreadsheet shown in Figure 5.3, where the author added a separate table to record grades of students from another class. The explanation relationship is captured by a zoom $X_2 \overset{\eta_2}{\lhd} S_2$, where

$$\eta_2 = \{(\mathtt{A1},\overline{\mathtt{A1}}),(\mathtt{A2},\overline{\mathtt{A2}}),(\mathtt{A3},\overline{\mathtt{A2}}),(\mathtt{A5},\overline{\mathtt{A5}}),(\mathtt{A6},\overline{\mathtt{A6}}),(\mathtt{A7},\overline{\mathtt{A6}}),(\mathtt{B1},\overline{\mathtt{B1}}),(\mathtt{B2},\overline{\mathtt{B2}}),$$
$$(\mathtt{B3},\overline{\mathtt{B2}}),(\mathtt{B5},\overline{\mathtt{B5}}),(\mathtt{B6},\overline{\mathtt{B6}}),(\mathtt{B7},\overline{\mathtt{B6}})\}$$

Notice that by using the rule Identical Content as described in Figure 5.2, it is possible to deduce that $(\mathtt{A1},\overline{\mathtt{A4}}) \in \eta_2$, which says that cell $\overline{\mathtt{A4}}$ explains cell `A1`. The

	A	B	C
1	Name	Grade	Grade
2	Aaron	C	B
3	Bill	A	B

(a) Subject Sheet S_3

	A	B	C
1	Name	Grade	Grade
2	*[Aaron...Bill]*	*[A...C]*	*[B]*

(b) Explanation Sheet X_3

Figure 5.4: Modified Grade-Book Example

derivation is as follows:

$$\frac{S(\texttt{A1}) = X(\overline{\texttt{A4}})}{(\texttt{A1}, \overline{\texttt{A4}}) \in \eta_2}$$

Even though the two cells have the same content, this conclusion is incorrect as they label different sections of the spreadsheet, a notion which needs to be captured in the relationship.

Note that one of the ways an explanation sheet X achieves abstraction of a spreadsheet S is through compression, thus the size of X cannot exceed the size of S, i.e. $|dom(X)| \leq |dom(S)|$. It follows that the address of an explanation cell in X is less than or equal to the address of all cells it explains in S. More generally,

$$\forall(\mathtt{a}, \overline{\mathtt{a}}) \in \eta : \overline{\mathtt{a}} \leq \mathtt{a}$$

With this new premise, $(\texttt{A1}, \overline{\texttt{A4}}) \notin \eta_2$, since address $\overline{\texttt{A4}} > \texttt{A1}$. However, $(\texttt{A1}, \overline{\texttt{A1}}) \in \eta_2$ will hold true because $\overline{\texttt{A1}} \leq \texttt{A1}$.

Another issue has to be addressed before we can obtain a complete definition of explanation inference. Consider another variant of the grade-book spreadsheet shown in Figure 5.4, where the author added grades for a new semester. The explanation relationship is captured by a zoom $X_3 \overset{\eta_3}{\lhd} S_3$, where

$$\eta_3 = \{(\texttt{A1}, \overline{\texttt{A1}}), (\texttt{A2}, \overline{\texttt{A2}}), (\texttt{A3}, \overline{\texttt{A2}}), (\texttt{B1}, \overline{\texttt{B1}}), (\texttt{B2}, \overline{\texttt{B2}}), (\texttt{B3}, \overline{\texttt{B2}}), (\texttt{C1}, \overline{\texttt{C1}}), (\texttt{C2}, \overline{\texttt{C2}}), (\texttt{C3}, \overline{\texttt{C2}})\}$$

Using the modified version of the rule Identical Content, it is possible to conclude that cell $\overline{\texttt{B1}}$ explains cell $\texttt{C1}$ because they share the same content and $\overline{\texttt{B1}} \leq \texttt{C1}$. The

derivation is as follows:

$$\frac{S(\texttt{C1}) = X(\overline{\texttt{B1}}) \qquad \overline{\texttt{B1}} \leq \texttt{C1}}{(\texttt{C1}, \overline{\texttt{B1}}) \in \eta_3}$$

However, this does not capture the true notion of explanation as cell $\overline{\texttt{B1}}$ does not explain cell C1, but rather explains cell B1. This issue can be alleviated by comparing the addresses of the candidate explanation cells and selecting the *closest* to the address of the cell being explained.

The set of cells $\{\overline{\texttt{a}}, \overline{\texttt{a}}', \ldots, \overline{\texttt{a}}''\}$ which explain some cell a under the current definition of the rule IDENTICAL CONTENT are referred to as candidate explanation cells. As a concrete example, the candidate explanation cells for cell C1 are $\{\overline{\texttt{B1}}, \overline{\texttt{C1}}\}$, since they all share the same content Grade, $\overline{\texttt{B1}} \leq \texttt{C1}$, and $\overline{\texttt{C1}} \leq \texttt{C1}$. For clarity as to why both cells $\overline{\texttt{B1}}$ and $\overline{\texttt{C1}}$ explain cell C1, the derivation is as follows:

$$\frac{S(\texttt{C1}) = X(\overline{\texttt{B1}}) \qquad \overline{\texttt{B1}} \leq \texttt{C1}}{(\texttt{C1}, \overline{\texttt{B1}}) \in \eta_3} \qquad \frac{S(\texttt{C1}) = X(\overline{\texttt{C1}}) \qquad \overline{\texttt{C1}} \leq \texttt{C1}}{(\texttt{C1}, \overline{\texttt{C1}}) \in \eta_3}$$

As per the previous observation, an explanation cell $\overline{\texttt{a}}$ can only explain a spreadsheet cell a if $\overline{\texttt{a}} \leq \texttt{a}$, therefore the address of each candidate explanation cell $\{\overline{\texttt{a}}, \overline{\texttt{a}}', \ldots, \overline{\texttt{a}}''\}$ will be less than or equal to the address of the cell a being explained. Referring back to the example, this claim holds true as $\overline{\texttt{B1}} \leq \texttt{C1}$ and $\overline{\texttt{C1}} \leq \texttt{C1}$.

Sorting the candidate explanation cells in ascending order and comparing them to the subject sheet cell a results in

$$\overline{\texttt{a}} < \overline{\texttt{a}}' < \cdots < \overline{\texttt{a}}'' \leq \texttt{a}$$

Applying this observation to the example results in

$$\overline{\texttt{B1}} < \overline{\texttt{C1}} \leq \texttt{C1}$$

From this comparison, it is easy to see that $\overline{\texttt{a}}''$ (or $\overline{\texttt{C1}}$ in the example) is closest to the address of the cell a (or C1 in the example) being explained. Therefore, the notion of *closest* address can be defined as the maximum address of all candidate explanation cells. With this condition, explanation inference can conclude that $(\texttt{C1}, \overline{\texttt{C1}}) \in \eta_3$, since $\overline{\texttt{C1}}$ is the maximum cell address out of the candidate explanation cells $\{\overline{\texttt{B1}}, \overline{\texttt{C1}}\}$.

$$\frac{X(\overline{\mathtt{a}}) \lhd S(\mathtt{a}) \qquad (\mathtt{b},\mathtt{a}) \in \mathcal{L}_S \qquad (\overline{\mathtt{b}},\overline{\mathtt{a}}) \in \mathcal{L}_X \qquad S(\mathtt{b}) = X(\overline{\mathtt{b}}) \qquad \overline{\mathtt{a}} \leq \mathtt{a}}{(\mathtt{a},\overline{\mathtt{a}}) \in \gamma}\ \textsc{Same Label}$$

$$\frac{S(\mathtt{a}) = X(\overline{\mathtt{a}}) \qquad \overline{\mathtt{a}} \leq \mathtt{a}}{(\mathtt{a},\overline{\mathtt{a}}) \in \gamma}\ \textsc{Identical Content}$$

Figure 5.5: Explanation Inference

The final version of the explanation inference rules is shown in Figure 5.5. The candidate mapping $\gamma \subseteq A \times A$ is a mapping from cells in a subject sheet to their respective candidate explanation cells in an explanation sheet. Rule Same Label describes the relationship between cells in a subject sheet that have labels and their candidate explanation cells, captured by the candidate mapping γ. Rule Identical Content describes the relationship between label cells in a subject sheet and their candidate explanation cells, also captured by the candidate mapping γ.

Based on γ, we can define η as follows:

$$\eta = \{(\mathtt{a},\overline{\mathtt{a}}) \in \gamma \mid (\mathtt{a},\overline{\mathtt{a}}') \in \gamma \Rightarrow \overline{\mathtt{a}}' < \overline{\mathtt{a}}\}$$

Due to the variety of spreadsheet structures found in real-world examples, explanation inference can derive several different explanation sheets for a spreadsheet, that is, the relationship between spreadsheets and explanations is one-to-many. We can impose an ordering on explanations based on their resulting size. The goal of explanations is to present the computational structure of a spreadsheet such that a user is not overwhelmed with data and can easily understand desired information. Therefore, an explanation X achieves more explanatory power than an explanation X' if $|dom(X)| \leq |dom(X')|$.

5.2 Evaluation of Explanation Inference

In this section, I present an evaluation of explanation inference conducted on the corpus described in Section 4.1. The goal of this evaluation is to assess the expressiveness of explanation inference, as well as check the correctness of the manually created explana-

tions.

The corpus consists of 41 worksheets which exemplify real-world usage of spreadsheets. Each worksheet is accompanied by a manually created explanation. In Section 4.1, an explanation checker was used to decide if the generated explanations were correct. The formula errors indicated by the explanation checker were fixed before employing inference checking in order to reduce redundancy in the results.

Moreover, explanation checking also discovered errors with the spreadsheet labeling relationship $\mathcal{L}_S$. As part of data preparation, this relationship was corrected so that it accurately captured the intended structure of the spreadsheet. From the explanation, I was able to manually create labeling relationship $\mathcal{L}_X$ for all sheets in the corpus. These labeling relationships, along with the sheets themselves, yielded all the necessary information to derive the total cell mapping η using explanation inference. Additionally, I manually created cell mappings for all sheet and explanation pairs to serve as the ground truth for comparison.

I encoded the explanation inference rules in Prolog, shown in Appendix A. The flags `SL` and `IC` are used to differentiate between the rules Same Label and Identical Content. Using this encoding of explanation inference, I created a script to infer the mapping between all cells in a spreadsheet to their respective explanation cells. Subsequently, I compared the inferred mapping to the ground truth to gauge the accuracy of inference. In this instance, accuracy is defined as the number of correctly inferred explanation cells divided by the total number of cells in the spreadsheet.

Results In most cases, explanation inference correctly derived the cell mapping between the spreadsheet and explanation. The average inference accuracy was 98.90%, with a minimum of 81.29% and a maximum of 100%. In fact, there were only 5 worksheets in which explanation inference did not achieve 100% accuracy. The cases in which explanation inference failed were due to poor labeling practices and discussed in further detail in the following section.

It is important to note that this evaluation excludes three of the spreadsheets from the corpus as each of them contains 52,000+ cells. The sheer size of these sheets caused the script to crash due to lack of memory. However, I can assume that explanation inference would have derived the correct mapping by looking at examples of sheets with similar structure. These sheets are structured similarly to a relational database, with

each column containing some values with a label at the top. Within the corpus, there are 7 other spreadsheets which follow the same structure pattern, all of which resulted in 100% inference accuracy. Since explanation inference is proven to capture this structure pattern correctly, it is safe to assume that it would have derived the correct mapping for the larger spreadsheets.

The explanation inference evaluation did not yield any sub-optimally compressed explanation sheets. While this was not the direct goal of the explanation inference evaluation, it is important to note. The generated explanations were aligned with the manually created ones and conformed to the most compressed version of possible explanation sheets for each spreadsheet.

Discussion The cases in which explanation inference failed were due to poor labeling practices found in the spreadsheets. More specifically, explanation inference failed in sections of the subject sheet that lacked a proper label. This discovery came as no surprise, since explanation inference relies on the labeling information to determine the mapping between the subject and explanation.

However, this finding shows that explanation inference can be used to detect labeling errors in a spreadsheet. Many of these errors can go by unnoticed especially as spreadsheet size and complexity increase. Explicitly pointing out instances of missing labels can potentially save a spreadsheet user from having to manually make assumptions of what the data represents.

The evaluation also brought to light some structural mistakes within the subject and explanation sheets. For example, a common mistake was a column or row being offset by some amount of indices. The evaluation found structural mistakes in 6 of the 38 worksheets. These structural mistakes were introduced when manually encoding the spreadsheets in Prolog from the Excel environment and do not represent mistakes made by the original spreadsheet authors. However, for the purposes of this evaluation, it is important that these mistakes were caught and adjusted for.

The structural mistakes were not caught by explanation checking, since only the contents of the cells were checked. Since the explanations were created based on the subject sheets, the mistakes persisted; so while the positioning was incorrect, the cells in the subject sheet still 'correctly' mapped to their explanation counterpart. These structural discrepancies were identified through the manually created cell mappings which served

as the ground truth. The mappings were created based on the original Excel representation of the spreadsheets, thus pointing out any flaws in the Prolog encoding. Moreover, the mistakes were more evident with inference checking, as the positioning of cells is as important as the contents.

Based on this artifact evaluation, we can conclude that explanation inference is applicable to real-world spreadsheets and can determine the correct mapping if the labeling information on the subject sheet is accurate.

Chapter 6: Comparison to Related Works

In this chapter, I provide a comparison between explanations and the related works discussed in Chapter 2. The comparison is based on the explanation principles discussed in Section 3.2 as well as the explanatory value brought forth by each piece of work.

Calculation View Though the Calculation View [27] does not directly aim at facilitating understanding in a spreadsheet, it exhibits similarities to explanations. Most importantly, the Calculation View presents the user with an abstract representation of the underlying computation found within a spreadsheet. Abstraction from the subject language, in this case a spreadsheet, provides a user with an alternative view that omits certain details which may make it difficult to understand the purpose of the computation. Similarly, explanations abstract from details of the spreadsheet to provide a high-level description of the intended purpose of the spreadsheet.

However, this abstraction is presented differently in the two approaches. Explanations use labeling information within a spreadsheet to replace cell references. The Calculation View does not restrict this abstraction to information found in the spreadsheet, but instead allows the user to define names for cells or a range of cells. The Calculation View offers more granularity when replacing cell references with user-defined names, as the names do not rely on the inherent structure of the spreadsheet.

A subtle difference between these approaches can be observed by assessing their intended purpose. The Calculation View focuses on error prevention during the creating of a spreadsheet, while explanations focus on discerning the intended meaning of a spreadsheet after it has been created. These approaches are complementary, as they facilitate understanding at different parts of a spreadsheet's life cycle.

Calculation View potentially provides a more welcoming spreadsheet environment to users with a programming background as it offers a more systematic representation of the underlying computational structure. However, the representation completely abstracts away from the spreadsheet structure and does not play on the user's familiarity with the spreadsheet form to facilitate explanations.

A Domain Terms Visualization Tool For Spreadsheets The tool presented by Kankuzi and Sajaniemi [21] facilitates understanding by overlaying additional information on the original spreadsheet, while explanations do so by providing a separate entity. However, these two approaches are very similar in how they achieve abstraction of a spreadsheet. Both approaches use the labeling information found within a spreadsheet to abstract from cell references to names recognizable by a user.

Similarly to explanations, the tool offers partiality of explanations by allowing a user to only look at specific cells or sections of a spreadsheet. This attribute is essential, especially as spreadsheets increase in size and complexity, since it allows a user to focus on sections of the sheet which evoke confusion. Moreover, both approaches display structure preservation as the original spreadsheet is not altered.

Visualizing Spreadsheets using Dataflow Diagrams Hermans et al. use data-flow diagrams in their approach to explaining the relationships between entities in a spreadsheet [17]. Similarly to explanations, data-flow diagrams focus on providing an abstract representation of the computational flow in a spreadsheet. Data-flow diagrams preserve the inherent computational structure of the spreadsheet, but provide an alternative view of how cells and worksheets interact with one another. Moreover, data-flow diagrams use labeling information of the spreadsheet to replace cell references with names familiar to a user.

Data-flow diagrams differ from explanations in the granularity expressed through the visual representation. Data-flow diagrams explain the relationships of not only cells in a worksheet, but data blocks within a worksheet and even worksheets themselves. Explanations focus on displaying the underlying computational structure and relationship between cells in a single worksheet.

Data-flow diagrams employ partiality by allowing a user to only examine parts of the spreadsheets which evoke confusion. For example, a user can expand a single formula cell into a data-flow diagram view to see what computation is taking place, without considering parts of the spreadsheet which do not directly influence said cell.

Moreover, the data-flow diagram approach provides the user with information on dependencies between worksheets in a spreadsheet, which explanation sheets cannot do. These dependencies become important when data is shared between worksheets, a feature not captured by explanation sheets.

Explanations and data-flow diagrams can be used complementary to get a clear representation of the intent of a spreadsheet.

Explanations in Other Domains The principles discovered through creating spreadsheet explanations can be observed in explanation design for other domains explained in Section 2.5. Similarly to explanations, these approaches retain certain aspects of the subject language to facilitate understating, aspects which an intended user should be familiar with. For example, the DSEL that models complex casual relationships between events and functions [30] is derived from neuron diagrams, a notation familiar with researchers in the intended field. The ubiquitous nature of these approaches can provide easy access to explanations, as users do not have to accustom to a whole new language.

However, these approaches do not include all aspects of the underlying subject language, but rather provide an abstracted view. For example, the previously mentioned DSEL provides the user with a simplified version of the neuron diagrams, as to not overwhelm them with information that is not pertinent to the problem at hand. Abstraction from unnecessary details allows the user to more easily digest an explanation, and potentially lead to better understanding.

Explanations allow for partiality, meaning that sections of a spreadsheet can be explained individually from another. The aforementioned approaches exhibit partiality by allowing only certain parts of the subject language to be explained. For example, the DSEL for explaining strategies in game theory [12] allows a user to view segments of the decision making which may be hard to understand, without having to explain the whole sequence of decisions.

Moreover, partiality allows for the incremental construction of more and more complete explanations, which leads to the principle of compositionality. As a concrete example, the game theory DSEL allows for the composition of multiple representations of decisions to be combined to create a sequence of decisions which took place in a game. Similarly, compositionality is exhibited in explanation sheets by allowing the explanation of parts of a spreadsheet to be combined to create an abstract representation of the whole spreadsheet.

Furthermore, an interesting observation can be made about these approaches to explanations in other domains. Each approach makes the underlying computation visible via some representation ubiquitous to targeted users. To this extent, explanation sheets

are similar to these approaches as they also provide a user with a different representation of the computation structure found within a spreadsheet.

Chapter 7: Conclusion

In this thesis, I have introduced an approach to aid spreadsheet understanding through explanation sheets and describe their creation process, guided by examples of real-world spreadsheets. An explanation sheet provides an abstract view of a subject sheet, with the intent to expose the computational structure without diving into the details of each formula. Additionally, I have described explanation principles derived from studying and creating explanation sheets, which can guide the creation of explanations in other domains.

Moreover, I have presented a set of inference rules which describe the relationship between a spreadsheet and an accompanying explanation sheet, along with an evaluation of the expressiveness of these rules. The evaluation shows the applicability of explanation inference to real-world spreadsheets, but also exposes its limitations. Shown through the user study, explanation sheets have the potential to aid spreadsheet understanding and reduce the overhead incurred as spreadsheets grow and change authors.

Spreadsheets are ubiquitous and widely used at a personal and industrial level. In some cases, users are not the original author of a spreadsheet, leading to confusion and lack of understanding while trying to understand the intended purpose and computation structure of a spreadsheet.

Many approaches have intended to document and prevent errors during the creation of a spreadsheet [5, 27], but none which can be retroactively applied to existing sheets to ease understanding and maintenance. To this extent, explanations are a novel first step towards mitigating the aforementioned problems.

However, explanations have some short-comings which limit their use to spreadsheets with well-defined labeling information. A possible solution to address this issue could lie in the inclusion of external information found in a spreadsheet, such as notes left by the author. Another possible approach to resolving this issue is requesting additional input from the user before creating an explanation.

Another potential area for future work is the integration of explanation sheets into a platform ubiquitous to spreadsheet users, such as Excel. The implementation could

employ a human-in-the-loop model to allow for live feedback from the user in the creation of explanation sheets. Such an integration could allow a user to easily transition to using explanation sheets to further improve their understanding of spreadsheets.

Bibliography

[1] R. Abraham and M. Erwig. UCheck: A Spreadsheet Unit Checker for End Users. *Journal of Visual Languages and Computing*, 18(1):71–95, 2007.

[2] R. Abraham, M. Erwig, and S. Andrew. A Type System Based on End-User Vocabulary. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 215–222, 2007.

[3] Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino, and Porfirio Tramontana. Exact: A tool for comprehending vba-based excel spreadsheet applications. *Journal of Software: Evolution and Process*, 28(6):483–505, 2016.

[4] Brian Bishop and Kevin McDaid. An empirical study of end-user behaviour in spreadsheet error detection & correction. *CoRR*, abs/0802.3479, 2008.

[5] Jácome Cunha and Diogo Canteiro. Spreadsheetdoc: An excel add-in for documenting spreadsheets. In *6th National Symposium on Informatics (INForum 2015)*, 2015.

[6] Jácome Cunha, Mihai Dan, Martin Erwig, Danila Fedorin, and Alex Grejuc. Explaining spreadsheets with spreadsheets (short paper). In *Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, GPCE 2018, pages 161–167, New York, NY, USA, 2018. ACM.

[7] G. Engels and M. Erwig. ClassSheets: Automatic Generation of Spreadsheet Applications from Object-Oriented Specifications. In *20th IEEE/ACM Int. Conf. on Automated Software Engineering*, pages 124–133, 2005.

[8] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger. Automatic Generation and Maintenance of Correct Spreadsheets. In *27th IEEE Int. Conf. on Software Engineering*, pages 136–145, 2005.

[9] M. Erwig and M. M. Burnett. Adding Apples and Oranges. In *4th Int. Symp. on Practical Aspects of Declarative Languages*, LNCS 2257, pages 173–191, 2002.

[10] M. Erwig and E. Walkingshaw. A Visual Language for Explaining Probabilistic Reasoning. *Journal of Visual Languages and Computing*, 24(2):88–109, 2013.

[11] Martin Erwig, Robin Abraham, Steve Kollmansberger, and Irene Cooperstein. Gencel: A program generator for correct spreadsheets. *J. Funct. Program.*, 16(3):293–325, May 2006.

[12] Martin Erwig and Eric Walkingshaw. A visual language for representing and explaining strategies in game theory. page 101108, 2008.

[13] Martin Erwig and Eric Walkingshaw. A visual language for explaining probabilistic reasoning. *J. Vis. Lang. Comput.*, 24(2):88–109, April 2013.

[14] Marc Fisher and Gregg Rothermel. The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, May 2005.

[15] T. A. Grossman. Source Code Protection for Applications Written in Microsoft Excel and Google Spreadsheet. *arXiv e-prints*, Jan 2008.

[16] Felienne Hermans and Emerson Murphy-Hill. Enron's Spreadsheets and Related Emails: A Dataset and Analysis. In *37th Int. Conf. on Software Engineering*, pages 7–16, 2015.

[17] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Breviz: Visualizing spreadsheets using dataflow diagrams. *CoRR*, abs/1111.6895, 2011.

[18] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Supporting professional spreadsheet users by generating leveled dataflow diagrams. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 451–460, New York, NY, USA, 2011. ACM.

[19] Amy Hodge. Case study: Spreadsheets. `https://library.stanford.edu/research/data-management-services/case-studies/case-study-spreadsheets`.

[20] B. Kankuzi and J. Sajaniemi. An empirical study of spreadsheet authors' mental models in explaining and debugging tasks. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 15–18, Sep. 2013.

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

[22] B. Kankuzi and J. Sajaniemi. Visualizing the problem domain for spreadsheet users: A mental model perspective. In *IEEE Symp. on Visual Languages and Human-Centric Computing*, pages 157–160, 2014.

[23] Bennett Kankuzi and Jorma Sajaniemi. A mental model perspective for tool development and paradigm shift in spreadsheets. *International Journal of Human-Computer Studies*, 86:149 – 163, 2016.

[24] B. A. Nardi and J. R. Miller. Int. journal of man-machine studies. pages 161–184, 1991.

[25] Stephen G. Powell and Kenneth R. Baker. *The Art of Modeling with Spreadsheets.* John Wiley & Sons, Inc., New York, NY, USA, 2003.

[26] R. Pressman. *Software Engineering: A Practitioner's Approach (5th ed.).* McGraw-Hill, New York, NY, 2001.

[27] Advait Sarkar, Andrew D. Gordon, Simon Peyton Jones, and Neil Toronto. Calculation view: multiple-representation editing in spreadsheets. pages 85–93, 10 2018.

[28] C. Verhoef. How to Implement the Future. In *26th Euromicro Conference*, pages 32–47, 2000.

[29] A. von Mayrhauser, M. Vans, and A. Howe. Understanding Behaviour During Enhancement of Large-scale Software. *Journal on Software Maintenance: Research and Practice*, 9(5):299–327, 1997.

[30] Eric Walkingshaw and Martin Erwig. A dsel for studying and explaining causation. In Proceedings IFIP Working Conference on *Domain-Specific Languages,* Bordeaux, France, 6-8th September 2011, volume 66 of *Electronic Proceedings in Theoretical Computer Science*, pages 143–167. Open Publishing Association, 2011.

APPENDICES

Appendix A: Explanation Inference in Prolog

```
/* SAME LABEL */
ex(AX, AS, SL, X, S, LX, LS) :- labels(BX, AX, LX),
                                labels(BS, AS, LS),
                                label_eq(BX, BS, X, S),
                                bor(AS, AX),
                                explains(AX, AS).

/* IDENTICAL CONTENT */
ex(AX, AS, IC, X, S, _, _) :- label_eq(AX, AS, X, S),
                              bor(AS, AX).

/* Finding max(AX) out of possible cell addresses */
forallAX(AX, AS, Flag, X, S, LX, LS) :-
                     findall(AXp, ex(AXp, AS, Flag, X, S, LX, LS), AXs),
                     maxAX(AXs, AX).


/* (L, A) are part of the set Ls */
labels(L, A, Ls) :- member((L, A), Ls).

/* X[A] = S[B] */
label_eq(A, B, X, S) :- member((A, L), X),
                        member((B, L), S).

/* AX =< AS*/
bor((C, R), (XC, XR)) :- XC =< C,
                         XR =< R.
```

```
/* Maximum address in a list */
maxAX([(C, R)], (C, R)).
maxAX([(C, R), (_ , Ro)|S], Max) :- R > Ro,
                                    maxAX([(C, R)|S], Max).
maxAX([(C, R), (Co, Ro)|S], Max) :- C > Co,
                                    R = Ro,
                                    maxAX([(C, R)|S], Max).
maxAX([(C, R), (Co, Ro)|S], Max) :- C < Co,
                                    R = Ro,
                                    maxAX([(Co, Ro)|S], Max).
maxAX([(_, R), (Co, Ro)|S], Max) :- R < Ro,
                                    maxAX([(Co, Ro)|S], Max).
```