Exploration of Spreadsheet Formulae with Fency

Andrea Kohlhase and **Alexandru Toader**
Jacobs University Bremen
D-28717 Bremen, Germany
a.kohlhase and a.toader @jacobs-university.de

Abstract

Spreadsheets are well-known to be frequently-used but error-prone communication devices. They are useful since they are active (e.g., automatic computation), provide a cognitive notation system drawing on visualizing values, meanings and relations at the same time (enabled by labeled, color-coded grids), and provide easy-to-use domain-specific operations (e.g., computational functions). The latter, in particular, is enabled by the text-style formula format in spreadsheets, in which variables are replaced by cell references. For simply-structured formulae this works very well. To keep the formulae simple, computations are modularized into subformulae and as such distributed over and beyond the spreadsheet. This makes the provenance (tree) of spreadsheet values difficult to understand – a probable cause for the high error rate in spreadsheets.

To explore and navigate the subformulae involved in the computation of a cell value we present the subformula explorer "Fency", a tree-based, explorative interface: Whenever a user clicks on a cell its formula becomes the root of a cell-dependency graph. Each child node displays the formula of a cell (or range) reference used in the parent formula. Moreover, each node represents a direct link to the respective cell (or range), so that it can be used for formula navigation as well.

1 Introduction

What is a mathematical formula? According to Wikipedia, in mathematics it is "*an entity constructed using the symbols and formation rules of a given logical language*". Even though there are multiple mathematical communities of practice which use a partly different set of symbols and slightly varying formation rules, there is a common understanding how to encode several information levels into formulae by extending the linear form of text.

On the one hand, this construction of a formula, O'HALLORAN calls a "*grammatical strategy for encoding meaning efficiently [. . . which is achieved . . .] through spatial and positional notation in a form that is not found in language.*" [O'H05, p. 112]. In Fig. 1 we can see some common typographical line elements. The spatial information needed to characterize the form of a typical English

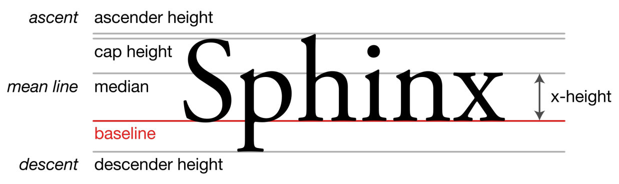


Figure 1: Typographical Line Elements[1]

text can be characterized via these line elements. But very often formulae need more space.

Accommodating our running example in Fig. 3, the equation

$$\sigma_4 = \frac{1}{3}\sum_{j=4}^{7}\delta_{4j}^2 \tag{1}$$

with variables σ_4 and δ_{4j} represents the simple formula used in cell [B4].

Here, if we take a closer look (Fig. 2), we realize that the equation transcends the ascender and descender height with respect to the typographical baseline of the used font quite a bit. If we look closely, we also realize right away that not only specific spatial and positional notation is used, the common font type is also broken, there are, for example, greek letters. For mathematicians these are not unexpected and hardly something to think about since they have internalized the notational naming convention within formulae, that is the relation between fonts and functional status of objects. This common mathematical practice of authoring and interpreting formulae evolved over centuries and proved to be effective and efficient for mathematicians.

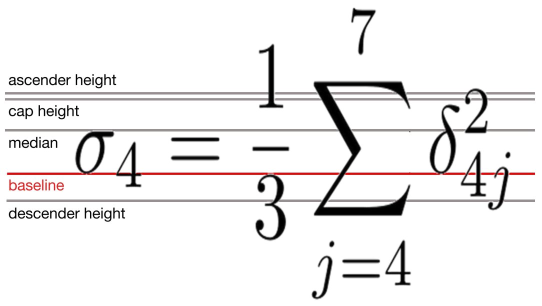
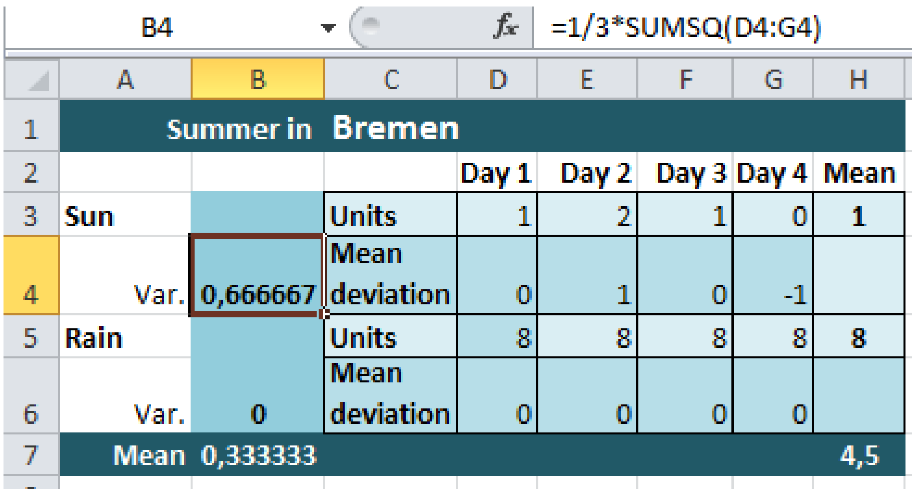


Figure 2: Equation (1) with Typography

On the other hand, in a spreadsheet there are also mathematical formulae. We can, for instance, reformulate Equation(1) as a computational formula in a spreadsheet like this:

$$\text{[B4]} = 1/3 * \texttt{SUMSQ}(D4 : G4) \tag{2}$$


B4 =1/3*SUMSQ(D4:G4)

	A	B	C	D	E	F	G	H
1	Summer in	Bremen						
2				Day 1	Day 2	Day 3	Day 4	Mean
3	Sun		Units	1	2	1	0	1
4	Var.	0,666667	Mean deviation	0	1	0	-1	
5	Rain		Units	8	8	8	8	8
6	Var.	0	Mean deviation	0	0	0	0	
7	Mean	0,333333						4,5

Figure 3: The Spreadsheet "Summer in Bremen"

The differences between the different representations is obviously vast. In this paper we use the example given in Sect. 2 as a running example. In particular we discuss the differences in Sect. 3 to motivate the design of our (sub)formula explorer "Fency" described in Sect. 4. We consider related work in Sect. 5 and conclude in Sect. 6 with an outlook on further work.

2 Running Example "Summer in Bremen"

Let us suppose that we want to describe the summer in Bremen statistically. Real-world distributions are typically not fully known, e.g. the rain could stop for 5 minutes when the observer went to the coffee bar to get some more coffee. In this case, the variance of the whole distribution is *estimated* by computing the **variance of a sample** of n observations drawn suitably randomly from the whole sample space according to Equation (3) where $x_1, \ldots, x_k$ represent the measurements and $\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$ their **arithmetic mean**.

$$\sigma = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})^2 \qquad (3)$$

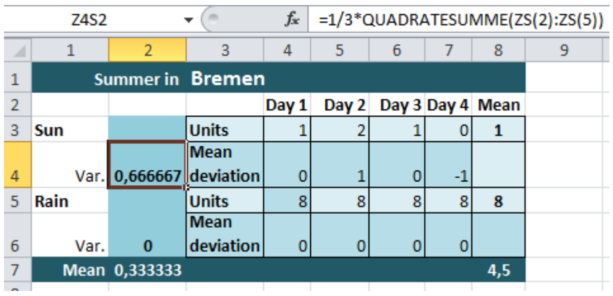
In the spreadsheet seen in Fig. 3, observed half-an-hour periods of full sunshine resp. rain in Bremen, i.e., the measurements, on four days in June are noted in ranges [D3:G3] resp. [D5:G5]. The difference $x_k - \bar{x}$ is called the **mean deviation of** x_k. The mean deviation of those measurements can be found in ranges [D4:G4] resp. [D6:G6]. The sample variance for sunshine in Bremen, for example, in cell [B4] is calculated from the mean deviation according to Equation (3) with the spreadsheet formula in Equation (2). Finally, the arithmetic mean of the sample variances is presented in cell [B7].

We use this example throughout the paper as running example.

3 Readability of Spreadsheet Formulae

In general, the set of symbols used in spreadseet formulae consists of given functions like `SUM`, individual macro extensions, numbers, and cell references like [B4] (in A1 referencing style referring to the cell in column B and row 4) or [R4C2] (in R1C1 referencing style pointing to the same cell). In `MS Excel`'10, for example, the set of symbols enlists 339 functions and $2^{20} \times 256$ cell references per worksheet. An essential component of spreadsheet players is their computational foundation: they can compute values from formulae, that is, they can simplify formulae to values. It is important to note that – even though it acts like a programming language – "*the formula language itself is entirely textual*" [Nar93, p. 49].

The formation rules are rather simple: concatenate the ingredients into a string of ASCII characters. From the user perspective NARDI points out that authoring and understanding formulae "*the user must master only two concepts: cells as variables and functions as relations between variables*" [Nar93, p. 42]. This is suspected to be the underlying reason for spreadsheets being the world's most used programming environment: the task of writing formulae (program scripts) is transformed into the task of writing text in a well-understood domain language consisting of typically 3-5 [SP88], at most 10 [Nar93, p. 43] and potentially – in `MS Excel` e.g. – 339 functions. It is rather interesting that the formula language hasn't changed at all since the very first appearance of spreadsheet applications, therefore we can call it a successful formula language.


Z4S2 =1/3*QUADRATESUMME(ZS(2):ZS(5))

	1	2	3	4	5	6	7	8	9
1	Summer in	Bremen							
2				Day 1	Day 2	Day 3	Day 4	Mean	
3	Sun		Units	1	2	1	0	1	
4	Var.	0,666667	Mean deviation	0	1	0	-1		
5	Rain		Units	8	8	8	8	8	
6	Var.	0	Mean deviation	0	0	0	0		
7	Mean	0,333333						4,5	

Figure 4: German R1C1 Notation of Equation (4)

Note that the ease of writing down spreadsheet formulae comes at the cost of reading them. For a simple formula, there is no problem in interpreting this linear notation of a formula – if the reader is very familar with the used naming convention for cells in spreadsheets.

The confusion begins if the spreadsheet author used the rather uncommon R1C1 referencing style, e.g., for Equation (2):

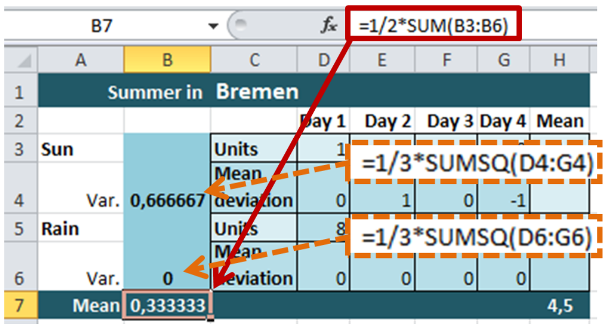
$$[\text{R4C2}] = 1/3 * \texttt{SUMSQ}(RC(2) : RC(5)) \qquad (4)$$

Here, the cell referencing is relative to the cell that will contain the calculated value, e.g. $RC(2) = R(0)C(2)$ refers to [D4] (with D=B+2,4=4+0).

It gets even more confusing if the spreadsheet author used e.g. the German `MS Excel` version with R1C1 referencing style (where "Z(eile)" stands for "R(ow)", "S(palte)" replaces "C(olumn)", and "`SUMSQ`" translates to "`QUADRATESUMME`") as in Fig. 4.

Besides this specific representation format knowledge, the reader might also get easily overwhelmed if the formula is complex. As readers are typically experts in their specific fields, but laymen in spreadsheet technology, this is in analogy to command line interfaces which work very well for simple commands used by laymen or for complex commands used by power users. Therefore, one explicit aim of a spreadsheet author has to be the optimal reduction of complex formulae.

This can be done via *modularization*, in particular by collapsing parts of formulae into variables by using these parts as autonomous formulae to calculate different cell values.



B7 =1/2*SUM(B3:B6)

	A	B	C	D	E	F	G	H
1	Summer in	Bremen						
2				Day 1	Day 2	Day 3	Day 4	Mean
3	Sun		Units	1				
4	Var.	0,666667	Mean deviation	0	1	0	-1	
5	Rain		Units	8				
6	Var.	0	Mean deviation	0	0	0	0	
7	Mean	0,333333						4,5

Figure 5: Modularization: Mean and Variance in [B7]

In Fig. 5 we can see a version of Fig. 4, this time in the more common A1 referencing style. The mean of all cell values in range [B3:B6] is calculated in [B7].

Moreover, the cells in the ranges [D4:G4] and [D6:G6] contain formulae of the kind (as shown in Fig. 6):

$$[D4] = \texttt{SUM}(D3; -\$H\$3) \tag{5}$$

Thus, in [B7] as seen in Fig. 5 we have the recursively resolved equations as shown in Fig. 9.

Note that even though the underlying formulae are one of the most simple ones, already the concatenated formula turns out to look rather complex to grasp. The reason consists of the fact that the cell references in Equation (10) can still be resolved easily by a reader, but the cell references in Equation (11) are more distributed and thus much harder to follow. HERMANS ET AL. report that nested formulae are hard to understand for end-users, which was also speculated in [Bre08]. "*We conclude that users find it difficult to work with long calculation chains*" [HPD12, p. 10]. Somewhat surprisingly they continue that this difficulty "*does not influence their perceived understanding of the formula or their ability to explain it*" [HPD12, p. 10]. A closer read reveals that their users are spreadsheet professionals, thus spreadsheet authors that not only do have the background knowledge for the specific spreatsheet at hand, they also know of the data architecture they created. They do not need to understand the concrete formula any longer as they trust in the underlying (hopefully) sound architecture.

As it is well-known that human short-term memory is rather limited (7 +/- 2 items can be kept in short term memory at any given time), the modularization of formulae is not an option, but rather a requirement for authoring readable spreadsheets. It is obvious that this modularization enables at the same time a high error rate with errors that are hard to debug.

The formula explorer Fency is based on the idea that the cell references can be automatically resolved into a cell-independent format e.g. presentation MathML [Aus+10] with variables that have mnemonic names, that is, names that hint at their meaning. For example, it is a quasi-standard to index a set of data points by a counter variable in $\{i, j, k, l, m, n\}$, to assign the name $\bar{y}$ to the mean of data points y_k, to name variances σ, and to name differences δ. Now look at Equations (10) to (12) in common mathematical notation:

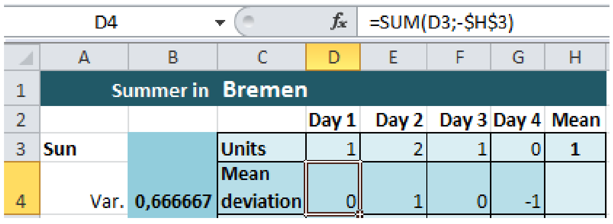
$$0,333333 = \bar{\sigma} \tag{6}$$

$$= \frac{1}{2} \sum_{i=3}^{6} \sigma_i \tag{7}$$

$$= \frac{1}{2} \sum_{i=3}^{6} \left(\frac{1}{3} \sum_{j=4}^{7} \delta_{ij}^2 \right) \tag{8}$$

$$= \frac{1}{6} \sum_{i=3}^{6} \sum_{j=4}^{7} (x_{ij} - \bar{x}_i)^2 \tag{9}$$

Note that typically a reader familiar with math notation will have noticed at the latest in Equation (7), that there is something strange going on with the mean being a sum of 4 numbers divided by the normalizing term 2. Looking at Fig. 5 we notice why the effect is correct, but the formula isn't. Therefore, math notation might also help to discover semantic errors in formulae.



D4 =SUM(D3;-H3)

	A	B	C	D	E	F	G	H
1	Summer in	Bremen						
2				Day 1	Day 2	Day 3	Day 4	Mean
3	Sun		Units	1	2	1	0	1
4	Var.	0,666667	Mean deviation	0	1	0	-1	

Figure 6: Modularization: Deviation in [D4]

The modularization can be kept, if we visualize the formula dependencies in form of a graph, where every node contains information about a formula.

4 The (Sub)Formula Explorer Fency

To keep spreadsheet formulae simple, computations are modularized into subformulae and as such distributed over and beyond the spreadsheet. Even though the modularization simplifies the formula itself, it resolves in a very complex provenance (tree) of spreadsheet values. The basic idea of Fency consists in an interactive visualization of the modularization of a formula. To explore and navigate the subformulae involved in the computation of a cell value we developed a semantically supported, tree-based, explorative interface: Whenever a user clicks on a cell its formula becomes the root of a "**formula graph**", i.e., a graph with cell/range nodes and cell/range-dependency edges. Each child node displays the formula of a cell (or range) reference used in the parent formula.

For example, in Fig. 7 we can see an entire formula graph developed after the user clicked cell [B7]. This

$$
\begin{aligned}
0,333333 &= 1/2 * \texttt{SUM}(B3:B6) && (10)\\
&= 1/2 * \texttt{SUM}(1/3 * \texttt{SUMSQ}(D4, G4) : 1/3 * \texttt{SUMSQ}(D6, G6)) && (11)\\
&= 1/2 * \texttt{SUM}(1/3 * \texttt{SUMSQ}(\texttt{SUM}(D3; -\$H\$3), \texttt{SUM}(G3; -\$H\$3))\\
&\quad : 1/3 * \texttt{SUMSQ}(\texttt{SUM}(D5; -\$H\$5), \texttt{SUM}(G5; -\$H\$5))) && (12)
\end{aligned}
$$

Figure 9: Recursively Solving Equations for cell [B7]

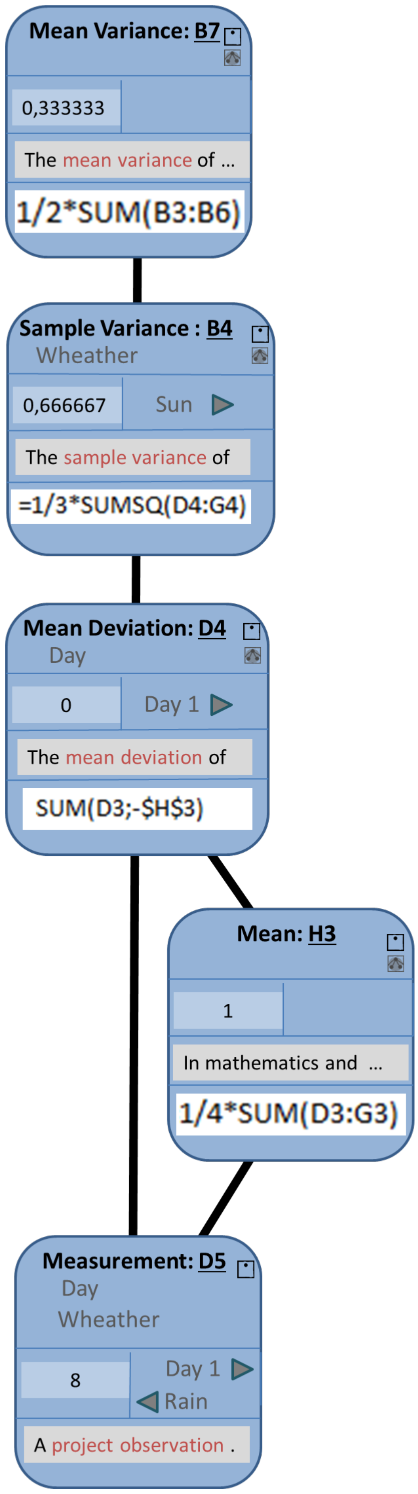


Figure 7: The Expanded Formula Tree in Cell [B7] (with Spreadsheet Formulae)

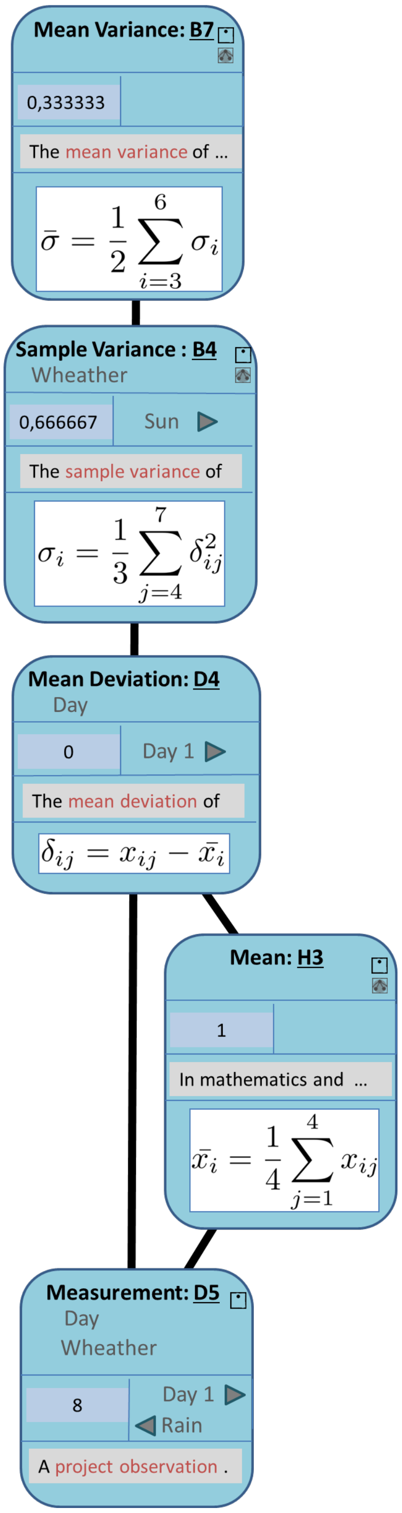


Figure 8: The Expanded Formula Tree in Cell [B7] (with Math Formulae)

cell contains the formula $1/2 * \texttt{SUM}(B3 : B6)$, that is Equation (10). The values in the cells in the cell range [B3:B6] are computed by equivalents of the formula $1/3 * \texttt{SUMSQ}(D4, G4)$ taken from cell [B4][2]. With Fency, if the user clicked cell [B7], the root node as in Fig. 10 would be created and the cell-dependency of the underlying formula on range [B3:B6] would give rise to a child node representing it in the formula graph. If the user wanted to see the child node of this, then she could click the expand button on the upper right and a node for the functional block in range [D4:G4] would appear.

On a more technical note, the formula explorer Fency is a semantic service integrated into the open source Semantic Alliance Framework [Dav+12]. This framework allows to superimpose semantic services over an existing (and possibly proprietary) application provided that it gives open-API access to user events. Elements in the application are connected to according concepts in structured background ontologies, which, for instance, contain a representation of the respective domain and some instance specific information. Semantic services can draw on the ontology information to offer intelligent services, which are offered to the user via the Semantic Alliance framework in local, but application independent windows. For the most common spreadsheet applications `MS Excel` and `LibreOffice` there are already existing Semantic Alliance APIs.

Fency offers more than a tree-based visualization of the (sub-)formulae in a spreadsheet. In a nutshell, every node of the formula graph consists of a list of elements:

- The **title** expressing the underlying meaning of a cell value or a range of values,
- a **link** to the corresponding cell/range in the spreadsheet,
- the **dependencies** this cell/range depends on,
- its data **value**,
- an **explanation** of its meaning,
- the spreadsheet **formula** (or its equivalent math formula), and
- **iterators** to move through the cells with their resp. values of a range.

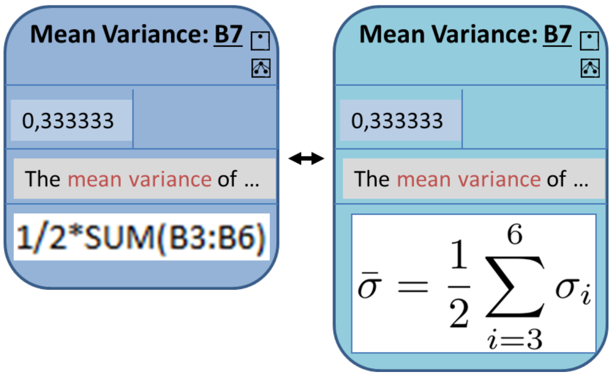
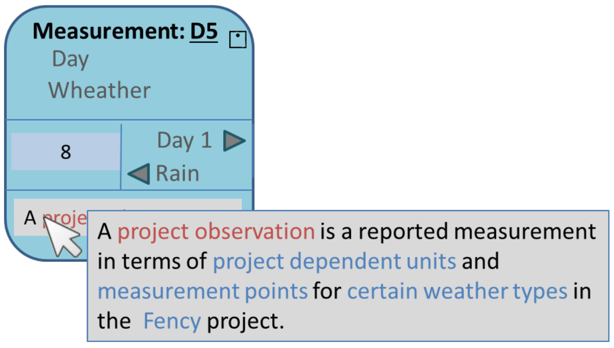


Figure 10: Node Variants in Cell [B7]

Let us have a closer look, for example, at a node like the left one in Fig. 10. The cell [B7] is associated with the ontology concept "mean variance". The title of this concept followed by the cell reference "B7" itself is used as a title for the node. The underline of the cell reference indicates that it represents a link to this cell. On the upper right-hand side we can see a collapse and an expand button, which collapses or expands the formula graph respectively if clicked. The cell value of cell [B7] is $0,333333$ and is shown in the node as well. In the grey box the beginning of the explanation of the concept "mean variance" given in the ontology is visible. Hovering over the grey box will trigger the expansion of it, so that the entire definition will be visible (see an example in Fig. 11). By using the JOBAD framework[JOBAD], the user can even interact with the information items within this explanation: If other concepts are referenced in this definition (indicated by blue font usage), a click will open another window with the according concept definition. This way, a user can explore the background ontology and comprehend the meaning of the formula much deeper. The lower part of the node contains the formula, here the formula for [B7], if existent; see an empty formula example in Fig. 11. The hovering effect kicks in here as well, in particular, if the formula exceeds a certain size, the entire formula will only be visible while hovering over the formula box.



A project observation is a reported measurement in terms of project dependent units and measurement points for certain weather types in the Fency project.

Figure 11: Expansion on Hover over Definition Box

Cell [B7] itself is not part of a functional block, but e.g. cell [D5] is. As the value in [B7] depends down in the formula tree on the value in this cell, we can find the node for [D5] as the last one in the formula graph in Fig. 7 or more conveniently in Fig. 11. This functional block covers the observed and summarized data. Each measurement depends on which day it was taken and what wheather condition is reported, in other words the measurement functional block depends on the day functional block [D3:G3] and the wheather functional block [[A3], [A5]]. This dependency is noted in the node directly under the title (in grey font). Moreover, we can see that cell [D5] contains the value for "Day 1" and "Rain". The triangular buttons allow a user to skim through the values in the respective functional blocks, and navigate to the respective spreadsheet cells via the link "D5" right after the title. This feature allows the user to easily navigate through related information items while abstracting away from the concrete structure. If any of the information items presented above are missing, the UI of the node adapts.

In a future prototype, if the user double clicks on the formula in a node, then the spreadsheet formula is converted into a math formula using MathML (see right node in Fig. 10). The option of presenting both variants seems sensible as a switch of formats should always be easily reversible to avoid confusion. The ontology concept "mean variance" includes knowledge about the symbol notation $\bar{\delta}$. Moreover, as the range [B3:B6] is associated with the concept "sample variance" with its symbol notation δ, a parser should be able to figure the math formula as seen in the

[2]Ranges used as cell references in formulae are typically **functional blocks**, i.e., cell ranges that have the same *functional* content, see [KK13] for more details.

right node in Fig. 10. To give a taste of the potential of this conversion, we include Fig. 8. Another idea, we want to pursue shortly is that the user can even edit the formula and push the changes back to the spreadsheet.

5 Related Work

The visualization of data-flows within spreadsheets is not a new idea. In `MS Excel` itself there is a *tracing tool* that visualises precedents and dependents of a selected cell. The visualization breaks if the dependencies are beyond the worksheet or even more so beyond the workbook.

In [CKR01] the authors studied the comprehension factor of formulae visualized in distinct ways. They frame formula understanding in terms of the reader's cognitive load and thus as a visual memory problem. They find that the "*ideal organization is the simple tree. It is the easiest to chunk. In the simple tree the surface organization of the formula tree is in harmony with its deep structure.*" [CKR01, p. 487].

KANKUZI and AYALEW presented in [KA08] a graph-based visualization of spreadsheets. Based on a Markov Clustering algorithm they generate a data-flow graph which visualizes cell cluster dependencies in an extra window aside the spreadsheet application window and provides semantic navigation similar to the one presented in Fency. Instead of using functional blocks, i.e., sets of cells that belong together semantically, these authors use statistical clustering. Even though this probably provides a similar grouping effect, the spreadsheet reader won't know why the cells are grouped. With Fency we cannot only offer the reader this reason, i.e., the semantic relating concept, we also allow the reader to dig into the definition of this concept.

In [Raj+00] a tree representation for formulae is suggested according to predominant Software Engineering techniques. In particular, a formula is divided into a structure tree containing operators and functions and an arguments tree containing cell addresses and constants. This tree visualization of a formula is suggested to be done when authoring a spreadsheet, whereas Fency is a tool that supports reading a spreadsheet. In [JMS06] a tool for generating formulae in several formats (possibly spreadsheet format) is presented. Again, the sole focus is given to the developper or author of formulae, nothing is said about the enhanced readibility or comprehendibility of a formula.

`http://www.spreadsheetstudio.com/` offers another type of formula explorer. The modularity of `MS Excel` formulae is made use of as is in Fency. This formula explorer offers a modal pop-up window that presents the formula of the selected cell. The formula is automatically segmented into sensible parts like cells, ranges, function plus function parameters, constants etc. If the user hovers over the formula shown then the corresponding value is presented. If a segment corresponding to a cell or range is left-clicked, then the formula of that `MS Excel` object is shown as before. Thus, this formula explorer allows a similar navigation thru a formula via its subformulae. Moreover, the `MS Excel` cursor also moves to the `MS Excel` object selected in the formula window.

ASUNCION suggests in [Asu11] to capture the provenance of cell values by unobtrusively document their history and to make this set of data available for later querying. This kind of provenance capture certainly is appealing because of its automation facility, but the provenance is not stored on a semantic level. Thus, the author has to recognize data to be able to interpret the provenance correctly. Otherwise this kind of data handling seems to be very tedious.

6 Conclusion and Further Work

In this paper we have presented Fency, a (sub)formula explorer for spreadsheets, that allows readers to deeper understand what formulae, which concrete calculated values, what underlying concepts are spread how and where over the document.

We hope that Fency will prove to be a useful service, especially as we are planning to extend its capability towards a light formula resp. concept editor, that allows to *update* existing formulae resp. ontology items. Even though the cell values are shown in the resp. formula nodes, we believe that the provenance of cell values is still not enough covered. The graph structure gives a hint where the data originally come from, but very often outside data bases are used for data input of spreadsheets. In particular, the spreadsheet author is typically a data architect. For him the primitives are data resources. Therefore, a set of new information objects could be introduced to spreadsheets. If they were present, then Fency could visualize it as well, to obtain a formula visualization that not only keeps all relevant information in one place, it also uses the notation that is most efficient.

Acknowledgements We want to thank the anonymous reviewers for their constructive suggestions. This work has been funded by the German Research Council under grant KO-2484-12-1.

References

[Asu11] Hazeline U. Asuncion. "In Situ Data Provenance Capture in Spreadsheets". In: *eScience*. IEEE Computer Society, 2011, pp. 240–247. ISBN: 978-1-4577-2163-2.

[Aus+10] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0.* W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: `http://www.w3.org/TR/MathML3`.

[Bre08] Andrej Bregar. "Complexity Metrics for Spreadsheet Models". In: *CoRR* abs/0802.3895 (2008).

[CKR01] David Chadwick, Brian Knight, and Kamalasen Rajalingham. "Quality Control in Spreadsheets: A Visual Approach using Color Codings to Reduce Errors in Formulae". In: *Software Quality Journal* 9.2 (2001), pp. 133–143.

[Dav+12] Catalin David et al. "`Semantic Alliance`: A Framework for Semantic Allies". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM) (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 49–64. ISBN: 978-3-642-31373-8. URL: `http://kwarc.info/kohlhase/papers/mkm12-SAlly.pdf`.

[HPD12] Felienne Hermans, Martin Pinzger, and Arie van Deursen. “Measuring Spreadsheet Formula Understandability”. In: *CoRR* abs/1209.3517 (2012).

[JMS06] Sven Jörges, Tiziana Margaria, and Bernhard Steffen. “FormulaBuilder: a tool for graph-based modelling and generation of formulae”. In: *Proceedings of the 28th international conference on Software engineering*. ICSE ’06. Shanghai, China: ACM, 2006, pp. 815–818. ISBN: 1-59593-375-1.

[JOBAD] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. URL: `http://jobad.omdoc.org` (visited on 02/18/2012).

[KA08] Bennett Kankuzi and Yirsaw Ayalew. “An end-user oriented graph-based visualization for spreadsheets”. In: *Proceedings of the 4th international workshop on End-user software engineering*. WEUSE ’08. Leipzig, Germany: ACM, 2008, pp. 86–90. ISBN: 978-1-60558-034-0.

[KK13] Andrea Kohlhase and Michael Kohlhase. “Spreadsheets with a Semantic Layer”. In: *Electronic Communications of the EASST: Specification, Transformation, Navigation – Special Issue dedicated to Bernd Krieg-Brückner on the Occasion of his 60th Birthday* (2013). Ed. by Till Mossakowski, Markus Roggenbach, and Lutz Schröder. in press. URL: `Http://kwarc.info/kohlhase/papers/easst11.pdf`.

[Nar93] Bonnie A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*. 1st. Cambridge, MA, USA: MIT Press, 1993. ISBN: 0262140535.

[O’H05] K.L. O’Halloran. *Mathematical discourse: language, symbolism and visual images*. Continuum, 2005. ISBN: 9780826468574. URL: `http://books.google.com/books?id=5LsAJaBRKRcC`.

[Raj+00] K. Rajalingham et al. “Quality control in spreadsheets: a software engineering-based approach to spreadsheet development”. In: *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. 2000, 10 pp. vol.1.

[SP88] Jorma Sajaniemi and Jari Pekkanen. “An empirical analysis of spreadsheet calculation”. In: *Softw. Pract. Exper.* 18.6 (June 1988), pp. 583–596. ISSN: 0038-0644.