Compensating the Semantic Bias of Spreadsheets

Andrea Kohlhase and Michael Kohlhase
German Center for Artificial Intelligence (DFKI)
{Andrea,Michael}.Kohlhase@dfki.de

Abstract

Spreadsheets are heavily employed in administration, financial forecasting, education, and science because of their intuitive, flexible, and direct approach to computation. They represent examples for "active documents" which are considered very worthwhile in the field of Knowledge Management. But these activity traits also lead to usability and maintenance problems, as spreadsheet-based applications grow evermore complex and long-lived. We argue that these difficulties come from a fundamental bias of spreadsheets towards computational aspects of the conceptual models underlying the applications, i.e., a semantic bias. Concretely, we present the software architecture of the SACHS project geared towards a semantic enhancement of MS Excel spreadsheets that aims at compensating this semantic bias.

1 Introduction

In 1979 Dan Bricklin and Bob Frankston co-authored the famous first spreadsheet program: "VisiCalc" [BF79]. Terry Winograd considers the key innovation of **spreadsheet programs** to be that they virtualize ledger sheets. In particular, "*spreadsheet virtuality combines the regular structure of the familiar ledger sheet with an underlying structure of interlinked formula*" [Win06, p. 230]. Ledger sheets have been well used in financial accounting scenarios and as spreadsheets they obtain the additional quality of being computational: the spreadsheet user can manipulate data directly or via the abstract level of formulas in order to appropriate the data setting, to forecast developments, and to test financial scenarios.

The management qualities led to wide spread use *and* reuse of spreadsheets (see [AE06] and its references for a discussion). But unfortunately, the intensity with which they were used yielded more and more complex documents intended to be shared among people who needed to understand the same (or similar) data sets. In turn, this very complexity now often results in errors on the data level and misinterpretation or misapprehension of the underlying model.

The work reported here originates in the SACHS project (Semantic Annotation for a Controlling Help System) at the German Center for Artificial Intelligence (DFKI), Bremen. It addresses the usability problem that even though MS Excel spreadsheets have the potential to serve well as an interface for a financial controlling system, DFKI employees experience them as too complex in practice — even longtime users cannot interpret all data and are not certain about its origin. A controlling system is a means for the organization to control finances, i.e., to *understand* profits and losses and draw conclusions, thus a lack of overview hampers the process: if users are not sufficiently informed they cannot optimize the next data to be gathered.

A key observation in our approach to solving these usability problems is that spreadsheets are *active documents*, we take this to mean that they are documents whose surface structure can adapt to the environment and user input. "Activeness" in documents can only be realized by making some of the relations between the underlying objects explicit to the software layer presenting the document. In the case of spreadsheets these relations are given as the spreadsheet formulae. In this paper we take a foundational stance and analyze spreadsheets as "semantic documents", where the formula representation is only the computational part of the semantic relations like how values were obtained. In this sense the focus on the computational aspects constitutes a semantic bias in spreadsheets. In particular, spreadsheets disregard other semantic relations that are important for understanding and relegate them to informal treatment, which we claim accounts for a good part of the usability problems lamented above.

To compensate the diagnosed semantic bias we propose to augment the two existing semantic layers of a spreadsheet — the surface structure and the formulae by one that makes the intention of the spreadsheet author explicit. In the SACHS project we encode the intention as an accompanying OMDoc [Koh06] document and can thereby provide multi-layered, semantic help services.

2 Semantic Layers in Spreadsheets

Instead of developing the general theory, we will expose the salient parts of our approach using Winograd's example spreadsheet from [Win06] (re-represented in MS Excel in Figure 1) as a running example. We will differentiate the three semantic layers in turn and draw conclusions viewing this spreadsheet as both an active and a semantic document.

	A	B	C	D	E	F
1	**Profit and Loss Statement**					
2						
3	(in Millions)		Actual		Projected	
4		1984	1985	1986	1987	1988
5						
6	**Revenues**	3,865	4,992	5,803	6,022	6,481
7						
8	**Expenses**					
9	Salaries	0,285	0,337	0,506	0,617	0,705
10	Utilities	0,178	0,303	0,384	0,419	0,551
11	Materials	1,004	1,782	2,046	2,273	2,119
12	Administration	0,281	0,288	0,315	0,368	0,415
13	Other	0,455	0,541	0,674	0,772	0,783
14						
15	Total Expenses	2,203	3,251	3,925	4,449	4,573
16						
17	**Profit (Loss)**	1,662	1,741	1,878	1,573	1,908
18						

Figure 1: A Simple Spreadsheet after [Win06]

2.1 Active and Semantic Documents

We call a document **semantic**, iff it contains an infrastructure that distinguishes between content and form. Note that such an infrastructure can range from superficial styling information in PowerPoint slide masters or LaTeX document classes, over RDFa annotations in web pages to formal logic specifications of program behaviors. The idea is that this infrastructure makes relations between the objects described in the document explicit, so that they can be acted upon by machines. In particular, semantic documents can be interpreted by "presentation engines" that operationalize the semantic relations by allowing the reader to interact with various aspects of the semantic properties. We call the combination of a semantic document with a presentation engine that can adapt the surface structure of the document to the environment and user input an **active document**. Our definition is between the concept of embedding semantic networks into hyperdocuments employed by Gaines and Shaw in [GS99] and the rather visionary notion of documents that can answer arbitrary questions about their content proposed by Heinrich and Maurer [HM00]. Crucially both presuppose some kind of content representation in or near the document and a suitable "presentation engine".

For the purposes of this paper, we will neglect the fact that most presentation engines also incorporate editing facilities and concentrate on the interaction with active documents for reading and exploration. This view is similar to what [UCI+06] call "*intelligent documents*".

A paradigmatic example of an active document is a Mathematica notebook [Wol02], where equations and mathematical objects can be inspected, visualized, and manipulated by the user. Here, the semantic document is written in the Mathematica markup language which includes a content markup scheme for mathematical formulae and a high-level programming language. The presentation engine is the Mathematica front-end which presents interactive documents to the user and calls the Mathematica kernel for evaluation of program fragments and computation of mathematical properties.

Spreadsheets are another paradigmatic class of active documents. Here the semantic document contains representations of the cell values or formulae together with display information like cell color, font information, and current viewport. The presentation engine is a spreadsheet program like MS Excel, which presents the semantic document to the user by giving it a grid layout and recalculates values from formulae after each update. But what is the underlying semantic model, i.e. what is the "activeness" of spreadsheets based on?

2.2 The Surface/Data Layer

If we look at the example in Figure 1, we see that the grid of cells can be roughly divided into three areas. The darker, ochre area in the center contains values of actual and past expenses and revenues, the lighter, yellow box on the right contains values projected from these. The white region that surrounds both boxes supplies explanatory text or header information hat helps users to interpret these numbers. Generally, non-empty cells that do not contain input or computed values usually contain text strings that give auxiliary information on the cells that do; we call these cells collectively the **legend** of the spreadsheet, since in this respect they serve the same purpose as the legend of a map.

Observe that row 17 displays the central values of the spreadsheet: the profit/loss situation over time (i.e., in the years 1984-1988 as indicated by the values in row 4). Moreover note that the meaning of the values in row 17 is that they represent profits and losses as a *function* π of time: remember that a function is a right-unique relation — i.e., a set of pairs of input values and output values. In our example the pair $\langle 1984, 1.662\rangle$ of values of the cells $[B4]$ and $[B17]$ is one of the pairs of π. We will call such a grid region a **functional block**, and the function it corresponds to its **intended function**. Empirically, all non-legend , semantically relevant cells of a spreadsheet can be assigned to a functional block, so we will speak of *the* functional block and *the* intended function of a cell.

In the general case a functional block consists of multiple rows and columns, then it represents a binary (i.e., two-argument) function, i.e. its values depend on two (main) parameters which are usually in the cells of the column on the left and the row on top of the block. We will call the cell ranges of arguments of the function of a block its **input row/column**. In

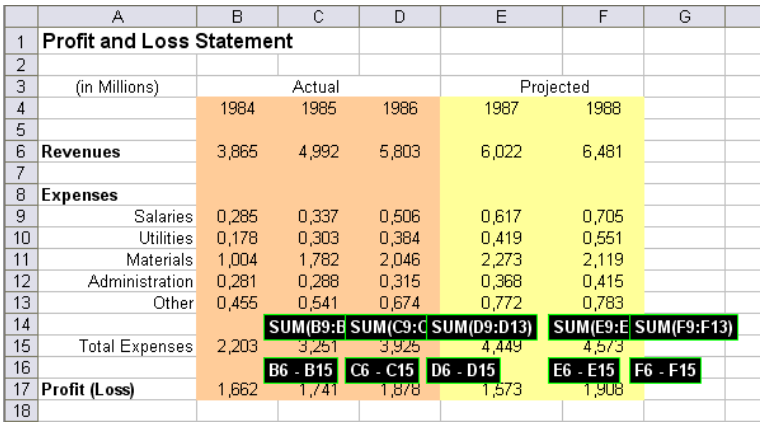

Figure 2: The Spreadsheet from Figure 1 with Formulae

our example the block with cells $[B9{:}D13]$ ranges over expense types (given in $[A9{:}A13]$) and years (given in $[B4{:}D4]$) and therefore represents a binary function.

Of course the actual values can also depend on more parameters, but these are given lower salience by the restriction to a two-dimensional grid. Alternatively, an n-ary function is split up into a sequence of binary functions and the spreadsheet grid contains blocks for all of these. If a functional block has only one row or column, then it represents a unary (i.e. one-argument) function, and in the case where it only consists of a single cell, it represents a constant (i.e., a function without arguments.)

Our notion of a functional block is related to, but different from Abraham and Erwig's of a "*table*" [AE04; AE06], which also contains row and column header and footer blocks. Our functional blocks roughly correspond to their "*table core*", whereas we would consider their header blocks as input blocks or legend cells (but there may be non-header input blocks in our model) and their footer blocks which contain aggregation cells as separate functional blocks.

2.3 The Formula Layer

A spreadsheet cell c may not only be associated with a simple data item (or **value**) $[\![c]\!]$, it may also be connected with a **formula** $[\![\![c]\!]\!]$, i.e., an expression built up from constants, an extended set of arithmetic operators, and references to other cells, which evaluates to the cell value. Without loss of generality we assume that cells with a constant value given by the spreadsheet author carry the pseudo-formula `input`.

In our example, the value of π can be computed from the yearly revenues in $\mathcal{R} := [B6{:}F6]$ and the total expenses in $\mathcal{E} := [B15{:}F15]$ by a simple subtraction, the total expenses can in turn be computed by summing up the various particular expense types listed in cells $[A9{:}A13]$. In Figure 2 we have exposed the respective formulae in the example spreadsheet.

Note that the formulae of cells in a functional block have to be "*cp-similar*" [BSRR02], i.e., they can be transformed/copied into each other by adjusting the respective rows and columns. We call a functional block **computed** if all of its formulae are cp-similar. In our example, the functional block $\mathcal{P} := [B17{:}F17]$ is computed: let γ range over the columns B to F in $\mathcal{P}$. Note that the formulae $[\![\![\gamma 17]\!]\!] = \gamma 6 - \gamma 15$ in cells $[\gamma 17]$ are indeed cp-similar. Together, they make up the function

$$\mathcal{F}(\mathcal{P}) := \{\langle [\![\gamma 6]\!], [\![\gamma 15]\!], [\![\gamma 17]\!]\rangle | \gamma \in \{B, \ldots, F\}\}$$

We call $\mathcal{F}(\mathcal{P})$ the function **induced by** the (formulae in) block $\mathcal{P}$. But we also observe that not all functional blocks in a spreadsheet are computed, for instance the block $[B9{:}D13]$ only has be pseudo-formula `input`, since all the values are provided by the spreadsheet author. We call such blocks **data blocks** and note that the property of being a functional block only depends on a functional correspondence (a conceptual aspect of the data) and not on the existence of formulae (a property of the spreadsheet).

With spreadsheet formulae, users can express data dependencies on a generic level, so that the spreadsheet program can do much computational work in the background. By this virtualization of the traditional ledger sheet (see above), the user's role is lifted to a layman programmer and offers according potential. But Abraham and Erwig report an error rate of up to 90% (!) in spreadsheets [AE06], which shows that this potential comes with a substantial risk. They analyze the source of many of these errors to be in a mismatch between what the spreadsheet author wants to express and the formulae he writes. They try to address this situation by static analysis techniques (type checking) of the formulae and supplying the author with "*spreadsheet templates*". To understand this mismatch better, let us now turn to model the author intends to convey.

2.4 The Intention Layer

Note that even though $\mathcal{F}(\mathcal{P})$ and π compute the same values, they are completely different functions. π is *defined* via the actual or projected profits or losses of an organization, while $\mathcal{F}(\mathcal{P})$ is a finite, partial binary arithmetic function. Even when we compose $\mathcal{F}(\mathcal{P})$ with $\mathcal{F}(\mathcal{R})$ and $\mathcal{F}(\mathcal{E})$ and restrict them to the years 1984-86 yielding $\mathcal{F} := \mathcal{F}(\mathcal{P}) \circ \langle \mathcal{F}(\mathcal{R}), \mathcal{F}(\mathcal{E})\rangle\big|_{[\![B4:D4]\!]}$, the functions $\mathcal{F}$ and π are only *extensionally* equal (they are equal as input/output relations) and still differ *intensionally*.

Surprisingly, only $\mathcal{F}$ is explicitly represented in the spreadsheet of Figure 1, moreover, this explicit representation is invisible to the user if she doesn't look at the formula box — thus, leaving the user to figure out the 'intention' (the function π) from the implicit information given in the white part by herself. This is why we speak of a **semantic bias** towards computational

aspects of spreadsheet program, as some layers of the semantics are explicated but others are not.

Generally, we can assume that spreadsheet program authors use spreadsheets to express and compute (measurable) properties of situations; if we look a little closer then we see that these are not properties of the world as such, but of a high-level, abstract, or mental model of the world, which we subsume under the term **intention of the spreadsheet**. In our example, the function π could be seen as a concept from the intention, whereas the function $\mathcal{F}$ can be seen as its implementation. In our simple example the intention is easy to deduce from the text in the legend and basic financial accounting knowledge.

But even here, some parts of the spreadsheet's intention remain unclear: e.g. for what company or department are the profits and losses computed or what are the methods of projection for the years 1987/8. Let us now take stock of what the cells in the spreadsheet mean and what information we need infer from this. As we already remarked above, the values of cells $[B17{:}D17]$ are (the scalar parts of) the actual profits/losses of a company in the years 1984-1986. We need information from cell $[A3]$ for the unit of measurement, from cells $[B3{:}D3]$ that they are actual, and from $[A17]$ for the interpretation as a 'profit/loss'. To understand the full meaning of these cells, we also need to know about profits and losses of companies — e.g. that high profits of a company I am employed by or that I own stock in are good for me, the fact that the company is based in the USA and therefore calculates finances in US$, and that values that are actual are computed from measured values. Finally, we need to know that the profit/loss of an organization over a time interval is *defined* as the difference between its revenues and expenses over this interval. This knowledge allows to compute the values of cells in $\mathcal{P}$ with the respective formulae from the values of cells in $\mathcal{R} \cup \mathcal{E}$ (i.e., using the function $\mathcal{F}$). The values of the cells in $\mathcal{E}$ can be similarly computed from the values of the cells $[B9{:}D13]$. Note that while the definition of profits and losses above is general accounting knowledge, this definition is probably particular to the particular company, as the applicable expenses vary with the organization.

A similar account can be given for the projected profits/losses in cells $[E17{:}F17]$, only that the interpretation of the cells wrt. the intention is even more difficult — even though the situation is simple if taken at face value. Cell $[\![E17]\!]$ is the projected profit in the year 1987, which is computed from the revenue and expenses in column E. But in contrast to the values in the actual block $[B6{:}D6] \cup [B9{:}D13]$, the values in the projected block $[E6{:}F6] \cup [E9{:}F13]$ are not measured, but projected from the actual values by some financial forecasting method that is reflected in the respective formulae. Note that the correspondence of the formula need not be as direct as in the case of the total expenses above. It might be that the forecasting method is defined abstractly, and the concrete formula is derived from that making some simplifying assumptions. Furthermore, we need to know what assumptions the forecasting method makes, what parameter values are employed and why, how reliable it is, etc. to fully understand the values. All of these concerns are not addressed at all in the spreadsheet as an active document. Abraham and Erwig describe this as follows:

> *There is a high level of ambiguity associated with spreadsheet template inference since spreadsheets are the result of a mapping of higher-level abstract models in the user's mind to a simple two-dimensional grid structure. Moreover, spreadsheets do not impose any restrictions on how the users map their mental models to the two-dimensional grid (flexibility is one of the main reasons for the popularity of spreadsheets). Therefore the relationship between the model and the spreadsheet is essentially many-to-many [...].* [AE06, p. 5]

3 Compensating the Semantic Bias

Our analysis of the example above has shown us that large parts of the intention of a spreadsheet is left implicit, even though it is crucial for a user's comprehension. In particular, a user needs to know the following for a spreadsheet:

- The **ontology**[1], i.e., background information about relations between concepts and objects from the intention. The objects in the intention include the functions represented in the spreadsheets e.g. π, their properties, e.g. the *units* of their arguments and values, and thus of the values in the cells.
- The **provenance** of data in a cell, i.e., how the value of this data point was obtained, e.g. by direct measurement, by computation from other values via a spreadsheet formula, or by import from another source; see [MGM+08] for a general discussion of provenance.
- The **interpretation**, i.e., a correspondence between functional blocks and concepts or objects of the intention. We distinguish three parts here
  - The **functional interpretation**, that specifies the intended function of the functional block.
  - The **value interpretation**, i.e. a bijective function that specifies how to interpret the values of the block cells as ontology objects.
  - The **formula interpretation** that links the formulae of a block to an object in the ontology. This mapping must be a *refinement* in the sense that the interpretations of proper formulae compute the same value as the formulae itself and the pseudo-formulae `input` is mapped to a provenance object.

In some spreadsheets that are the digital equivalent to "back-of-the-envelope calculations", the interpretation, provenance, and ontology information is simple to infer, so that the un-documented situation is quite

[1] In this paper we use the word "ontology" in its general, philosophical meaning as the study of the existence of objects, their categories and relations amongst each other, and not in the computer science usage, where it is restricted to formal systems with tractable inference properties (description logics).

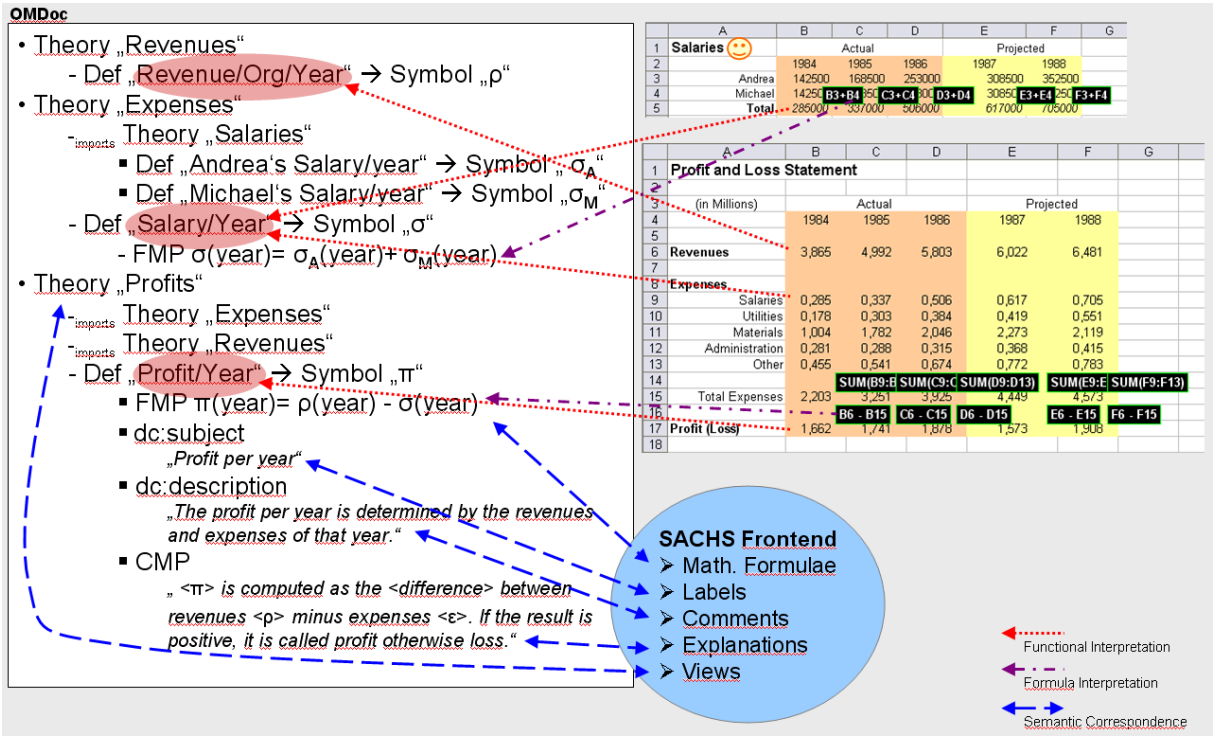

Figure 3: An OMDoc Ontology and Interpretation for the Spreadsheet from Figure 1

tolerable. Indeed this shows the cognitive strength of the table metaphor, in our example it is no problem for the human reader to interpret the legend item "*(in Millions)*" as a specification of the value interpretation of the cells $[B6{:}F17]$ (but not of the years $\mathcal{Y} := [B4{:}D4]$).

In many cases spreadsheets have developed into mission-critical tools that are shared amongst whole departments, because encapsulate important, non-trivial, institutional knowledge. The intention of such spreadsheets is much harder to infer, a fact that is witnessed by the fact that companies spend considerable energy to train employees in the usage (and intention) of such spreadsheets.

In this situation, it would be natural to extend the activity of spreadsheets to support the user's comprehension of the spreadsheet intention. Thus, in light of the discussion of Section 2.1, we suggest to compensate for the semantic bias diagnosed above by extending the underlying semantic document of a spreadsheet. Concretely, we propose to represent the intention (as the provenance and ontology) and to tie the cells in the spreadsheets to concepts in the intention (via an interpretation function).

3.1 Fixing the Ontology

We have (at least) two possibilities to extend spreadsheets with an ontology and provenance component: we can either extend spreadsheets by ontology and provenance facilities or we can extend them by interpretation facilities that reference external representations of the intention. As we have seen above, the intention contains quite a lot of information, and making it explicit in a software framework means a large investment. Therefore we contend that an external representation of the intention is more sensible, since it can leverage pre-existing tools and profit from interoperability.

We use the OMDoc format (Open Mathematical Documents, see [Koh06]) to represent the intention model. OMDoc is an XML-based format for representing semi-formal, semantically structured document collections. It allows to factorize knowledge into "*content dictionaries*" that serve as constitutive contexts for knowledge elements. OMDoc provides a mathematically inspired infrastructure for knowledge representation: document fragments are classified by their epistemic role as e.g. axioms, definitions, assertions, and proofs and mathematical formulae are represented in the OpenMath [BCC+04] or MathML [ABC+03] formats. Furthermore, OMDoc provides a strong, logically sound module system based on structured "theories" (content dictionaries extended by concept inheritance and views) [RK08]. Finally, the language has been extended to deal with units and measurable quantities [HKS06] as a prerequisite for interacting with the physical world. We make use of all of these features for modeling the intentions of spreadsheets. In contrast to other ontology modeling languages like Owl [MvH04], the OMDoc format does not commit to a formal logical language, and therefore lacks a native concept of inference but also does not force the author to fully formalize the spreadsheet intention and to work around the expressivity limitations of the underlying logical system. Instead, OMDoc allows to locally formalize elements — and thus provide partial inference — with whatever formal system is most suitable; in our application, we mainly use an formal system for arithmetics as a counterpart for spreadsheet formulae.

For the intention model in our example we divide the background knowledge into theories that inherit functionalities from each other via the `imports` relation. At the very basis we would have a content dictionary Revenues that defines the concept of the revenue of an organization over a time interval. This theory defines

the concept of a binary revenue function ρ, such that given an organization o and a natural number n the value $\rho(o, n)$ is the revenue (as a monetary quantity) of o over the span of the year n (AD) in the Gregorian calendar. Note that we use this very naive notion of revenues for didactic purposes only. For instance ρ was chosen as a binary function to highlight that there is no automatic agreement between functional correspondences in the table and objects of the intention. We would be perfectly free to analyze the concepts more thoroughly, embarking into representing monetary systems, theories of time, etc. For the purposes of this paper, we assume that we can either appeal to the intuition of the user or inherit these representations from a suitable foundational ontology.

In the same way we proceed with a content dictionary Expenses, which imports from a content dictionary Salaries. Finally, we build a content dictionary Profits that imports from both. In the OMDoc document pictured in Figure 3 we have summarized some of the relevant content dictionaries and the concepts they introduce.

3.2 Fixing the Provenance

We enrich our ontology with provenance information: As we have required the formula interpretation to be a refinement, we need to represent an abstract notion of spreadsheet computation in the ontology. This can be readily done making use of e.g. the CASL libraries [CoF04]. For modeling the provenance of user inputs in spreadsheets, we can be extremely minimalistic, just establishing a stub content dictionary that lifts the concept of "user input" to the ontology level. But in our example we can already see what a more elaborate provenance model could give us: We could specify that the salary values in $[B9{:}F9]$ are not only user inputs, but really manually copied over from another spreadsheet — the top spreadsheet "Salaries" in Figure 3. To take advantage of this (see details in the next section) we have to develop content dictionaries for provenance, adapting and extending first formalizations reported on in [MGM+08]. As this goes beyond the scope of this paper, we leave this to future work.

3.3 Fixing the Interpretation

To interpret the cells in $\mathcal{P}$ for example, we need to

- *fix the functional interpretation*: Identify that $\mathcal{P}$ and $\mathcal{R}$ form functional blocks with input row $\mathcal{Y}$. In Section 2.2 we have already discussed the functional interpretation of $\mathcal{P}$: it is just the intended function π. Similarly, we link the revenues block $\mathcal{R}$ with a unary function $\rho(c, \cdot)$, where c is the representation of our example company. Then we have to express the semantic counterpart of the spreadsheet formulae. In our OMDoc format we can simply represent this function as $\lambda y.\rho(c, y)$ in OpenMath or MathML.
- *fix the value interpretation*: In our example we observe that the values in $\mathcal{P}$ are actually only the scalar parts of measurable quantities, in this case measured "in millions" (of US$ presumably) according to the spreadsheet legend. Similarly, the (string) values `Salaries`, `Utilities`, ... in $[A9{:}A13]$ have to be interpreted as objects in the ontology. Thus in our example, we choose $i := \{y \mapsto y(AD)\}$ as the value interpretation of block $\mathcal{Y}$ and $j := \{x \mapsto 10^6 x\text{US\$}\}$ for block $\mathcal{P}$; obviously both are bijective.
- *fix the formula interpretation*: our example the formulae $\gamma 6 - \gamma 15$ in the functional block $\mathcal{P}$ would be linked to the formula $\pi(year) = \rho(year) - \epsilon(year)$ in the `FMP` element of the Profit/Year definition. Furthermore, we would link $\mathcal{R}$ to a semantic provenance object "imported from `Salaries.xsl`".

In Figure 3 we show the functional interpretation mappings as red, dotted arrows and the formula interpretation as purple, dot-dash-arrows. The totality of cell interpretations induces a set of content dictionaries that is associated with a spreadsheet, we call these content dictionary the **intention model**. Note that we can see the intention model as a representation of the intention of this spreadsheet.

We can think of the value interpretations as parser/generator pairs that mediate between the scalar function represented by the formulae in the spreadsheet and the intended function in the intention — which is usually a function between measurable quantities. In particular the functions $\mathcal{F}$ and π are related via the following commutative diagram, where the function $\mathcal{F}$ is induced by the spreadsheet formulae as discussed in Section 2.4 above.

$$\begin{array}{ccc} \mathcal{Y} & \xrightarrow{\mathcal{F}} & \mathcal{P} \\ {\scriptstyle i}\uparrow & & \downarrow{\scriptstyle j^{-1}} \\ \text{year} & \xrightarrow{\pi} & \text{profit} \end{array} \qquad \begin{array}{l} \textit{spreadsheet} \\ \\ \textit{intention} \end{array}$$

We see that the three components of the interpretation fully specify the correspondence between functional blocks in the spreadsheet and objects induced by the intention model. To see the strength of this construction let us return to our example and look at the import of salaries from `Salaries.xsl`. There we have a different value interpretation for the functional block $[F6{:}F6]$: this spreadsheet does not calculate in millions, so we chose $k := \{x \mapsto x\text{US\$}\}$ and get the import functions $k \circ j^{-1} = \{x\text{US\$} \mapsto 10^{-6} x\text{US\$}\}$ in the intention and $j^{-1} \circ k = \{x \mapsto 10^{-6} x\}$ in the spreadsheet.

In conclusion we observe that setting the ontology, provenance, and the interpretation of a cell gives us a full and explicit account of its intention, and we can relegate all further semantic services to the intention model. For instance we can verify (using inference in the intention model) this part of the spreadsheet by establishing the equality of $j^{-1} \circ \mathcal{F} \circ j$ and $\rho(c)$.

3.4 Semantic Services in SACHS

The `SACHS` system is a work-in-progress add-in for MS Excel (written in Visual Basic for Applications) that aims at providing semantic help facilities for spreadsheets. Though it has been developed for the DFKI controlling system, it works for any spreadsheet whose intention has been encoded as an OMDoc document, e.g. our running example. We have designed `SACHS` as *invasive technology* that extends well-used (and therefore well-known) software systems from within to overcome usage hurdles — see [Koh05b] for a discussion. Following [BWK00] we design the `SACHS` interface as a

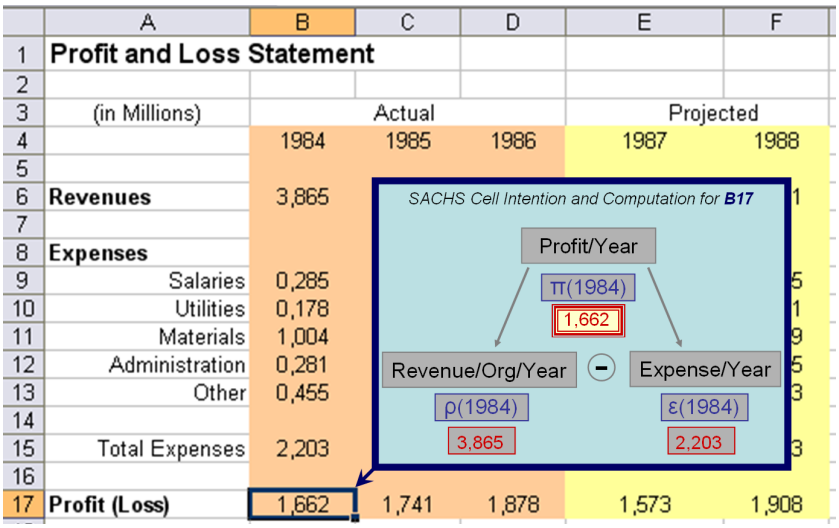

Figure 4: Theory Graph Example for the Spreadsheet from Figure 1

multi-view interface, where semantic services in SACHS allow the user to explore the intention of a spreadsheet. In the rest of the section we will present actual and envisioned services and discuss their realization in SACHS.

A very basic service that is missing in MS Excel is an overall visualization of the formulae in the spreadsheet together with the computed values. SACHS provides this as a multi-view interface — see Figure 2. For services that visualize the intention, the cells in the spreadsheet must be interpreted, i.e., linked to elements of the accompanying OMDoc document as e.g. shown in Figure 3. Generally, all cells in a functional block are linked to an OMDoc definition — the definition of its intended function, while OMDoc assertions justify their formulae. This assignment is internally represented by an extra worksheet within the spreadsheet we call the "SACHS interpretation". This is manually maintained by the spreadsheet author. In the case of the DFKI controlling system, spreadsheets are not hand-authored but generated from a controlling database, therefore the interpretation in the SACHS system is mediated by a level of symbolic keywords that insulate the setup against changes in the concrete spreadsheet generation scheme.

Once the interpretation is established we can directly make use of the various elements of the OMDoc information for the respective objects (see the dashed arrows in Figure 3). For instance, the Dublin Core metadata element `dc:subject` of a definition can be used as a *SACHS label* for the cell it interprets.

Moreover, MS Excel's comment functionality is hijacked to create *SACHS comments* that draw on the respective `dc:description` element, which contains a concise description of the object in question. In contrast, the CMP element of the OMDoc definition contains a detailed explanation using semantically marked up representations of objects from the intention model. These can be mathematical formulae encoded as OpenMath objects like the revenue function ρ or technical terms like "difference" which we have decorated in angle brackets in Figure 3. The added value of semantic annotation here is that the meaning of both can be further explored: The frontend item "*SACHS explanations*" allows this by providing "jump points" from within the text to those cells that are assigned to the definitions of those symbols via the SACHS interpretation sheet. Once jumped the user can look up the available semantic information of that particular cell and so on.

A *formula* underlying a cell is mirrored in the formula element FMP of the respective definition in the semantic document (see Figure 3) in the OpenMath format, this allows us to present it to the user in math notation: $\sum_{i=1}^{5} \epsilon_i(1984)$ is more readable than "`=SUM(B9:B13)`".

From a semantic point of view, the possibility of offering contextual views as the one in the blue pop-up in Figure 4 is the most interesting and probably helpful one. Such views allow a user to understand the connections between spreadsheet cells and background information. This view aggregates information about

- the intention of a cell in terms of the intended function of its functional block, and the intended arguments.
- how the cell value is computed from the values of its arguments.
- and the intentions of the arguments.

All information points in the pop-up are hyperlinked to their respective sources in the OMDoc document or in the spreadsheet. In a future version of SACHS, these hyperlinks could pre-instantiate the intention model with the argument values and allow an exploration from the view of the current cell — in our example in Figure 4 the intention specialized to the year 1984. Note that our example here only shows the situation for a formula-computed cells. For other provenances, the pop-up would visualize the provenance object given by the formula interpretation. For instance, for cell $[B9]$ the provenance object is "imported from `Salaries.xls`$[B5]$", so we can visualize the data from that cell using the existing SACHS pop-up.

As OMDoc is organized by theories, the provision of multi-level theory graphs as in the CPoint system [Koh05a] are nice-to-have services one can think of.

4 Conclusion and Outlook

We have analyzed the reasons for users' difficulties in understanding and appropriating complex spread-

sheets, as they are found e.g. in financial controlling systems. We claim that the ultimate cause is that spreadsheets are weak as active documents, because their underlying semantic documents are biased to computational aspects and fail to model the provenance, interpretation, and ontological relations of the objects and concepts operationalized by the system. To remedy the situation we propose to explicitly model the intention of a spreadsheet as an intention model in a collection of OMDoc content dictionaries which serve as an explicit knowledge base for the spreadsheet. Finally, we present the work-in-progress `SACHS` system that draws on such an intention model to offer various semantic services that aid the user in understanding and interacting with the spreadsheets.

In essence, our approach makes double use of the following duality identified by Fensel in [Fen08]

- *Ontologies define* formal *semantics for information, consequently allowing information processing by a computer.*
- *Ontologies define* real-world *semantics, which makes it possible to link machine processable content with meaning for humans based on* consensual *terminologies.*

In the analysis we look at the formal semantics of spreadsheets and find a semantic bias that hampers understanding since it fails to model consensual terminologies and therefore leads to real-world usability problems. In our proposed solution we extend the formal semantics of spreadsheets to draw on explicitly represented consensual terminologies.

While a semantic help system for a spreadsheet-based controlling system was the original motivation for our analysis, we feel that an explicit representation of the intention model of a spreadsheet has many more applications: it can be used for verification of the formulae, for change management along the lines of [MW07], and automated generation of user-adapted spreadsheets.

Our approach seems to be related to "*Class Sheets*" [EE05] introduced by Engels and Erwig. Their class descriptions can also be counted as an external, structured intention model, which is referenced by an interpretation mapping. We will have to study the relation to our work more closely, but it seems that their work suffers the same semantic bias towards computational issues as the spreadsheets themselves. But classes with extensive documentation or UML diagrams might go some ways towards generating a help system like `SACHS`.

In the future we plan to extend the repertoire of semantic services of the `SACHS` system. For instance, we envision a dual service to the one in Figure 4 which could visualize where the value of a cell is used to get an intuition for the relevance of a cell value.

At the moment, our approach presupposes that the spreadsheet author documents the spreadsheet intention as an OMDoc document and also annotates the spreadsheet with interpretation information. Both tasks place a heavy burden on the author and currently restrict our approach to mission-critical or extremely complex spreadsheets. But we expect the burden of specifying the ontology to decrease as more and more OMDoc content dictionaries for common models (e.g. standard accounting techniques) appear. For the interpretations, we plan to adapt techniques of header, unit, and template inference [AE04; AE06] to partially automate the annotation process via suggestions.

But future extensions are not limited to spreadsheets. Note that the semantic analysis in Section 2 is largely independent of the computational information that is at the heart of spreadsheets. The semantic information we are after only pertains to the use of data grids as a user interface to complex functions. In the future we plan to generalize the intention model architecture presented in Section 3 to the case of data grids — e.g. tables of experimental results or raster data from satellite images. Moreover, we want to develop a native infrastructure for representing "data grids" as a *user interaction feature* in OMDoc: In accordance with the format, the provenance and interpretation functionality would allow to link grid presentations to their intention without leaving the language. OMDoc-based systems could then pick up the semantics and offer complex interactions based on them. This would lead to much more general active documents. We could even turn the approach presented in this paper around and generate `SACHS` spreadsheets from OMDoc documents as active documents.

As a theoretical extension, we conjecture that the interpretation mappings can actually be seen as logic morphisms [RK08], which would allow us to re-use much of the higher-level OMDoc functionalities based on these.

Finally, we admit that we are still at the very beginning of understanding the repercussions of viewing spreadsheets as active and semantic documents. We were quite surprised at the depth of meaning in even our very simple example and are looking forward to further spelunking.

References

[ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium, 2003. Available from World Wide Web: `http://www.w3.org/TR/MathML2`.

[AE04] Robin Abraham and Martin Erwig. Header and unit inference for spreadsheets through spatial analysis. In *IEEE International Symposium on Visual Languages and Human-Centric Computing*, pages 165–172, 2004.

[AE06] Robin Abraham and Martin Erwig. Inferring templates from spreadsheets. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 182–191, New York, NY, USA, 2006. ACM.

[BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. Available from World Wide Web: `http://www.openmath.org/standard/om20`.

[BF79] Dan Bricklin and Bob Frankston. VisiCalc: Information from its creators, Dan Bricklin and Bob Frankston. `http://www.bricklin.com/visicalc.htm`, 1979.

[BSRR02] M. M. Burnett, A. Sheretov, B. Ren, and G. Rothermel. Testing homogenous spreadsheet grids with the "what you see is what you test" methodology. *IEEE Transactions on Software Engineering*, 29(6):576–594, 2002.

[BWK00] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI 2000*, pages 110–119. ACM, 2000.

[CoF04] CoFI (The Common Framework Initiative). CASL *Reference Manual*. LNCS 2960 (IFIP Series). Springer, 2004.

[EE05] Gregor Engels and Martin Erwig. ClassSheets: Automatic generation of spreadsheet applications from object oriented specifications. In *20th IEEE/ACM International Conference on Automated Seofware Engineering*, pages 124–155. IEEE Computer Society, 2005.

[Fen08] Dieter Fensel. Foreword. In Martin Hepp, Pieter De Leenheer, Aldo de Moor, and York Sure, editors, *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*, Semantic Web and beyond: Computing for Human Experience, pages 9–11. Springer Verlag, 2008.

[GS99] Brian R. Gaines and Mildred L. G. Shaw. Enbedding formal knowledge models in active document; creating problem-solving documents. *Communications of the ACM*, 42(1):57–63, 1999.

[HKS06] Eberhard Hilf, Michael Kohlhase, and Heinrich Stamerjohanns. Capturing the content of physics: Systems, observables, and experiments. In Jon Borwein and William M. Farmer, editors, *Mathematical Knowledge Management, MKM'06*, number 4108 in LNAI. Springer Verlag, 2006.

[HM00] E. Heinrich and H. Maurer. Active documents: Concept, implementation, and applications. *Journal of Universal Computer Science*, 6(12):1197–1202, 2000.

[Koh05a] Andrea Kohlhase. Cpoint, 2005. http://kwarc.info/projects/CPoint/.

[Koh05b] Andrea Kohlhase. Overcoming Proprietary Hurdles: CPoint as Invasive Editor. In Fred de Vries and Graham Attwell and Raymond Elferink and Alexandra Tödt, editor, *Open Source for Education in Europe: Research and Practise*, pages 51–56. Open Universiteit of the Netherlands, Heerlen, 2005.

[Koh06] Michael Kohlhase. OMDOC – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[MGM+08] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. The provenance of electronic data. *Communications of the ACM*, 51(4):52–58, 2008.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, February 2004. Available from World Wide Web: `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

[MW07] Normen Müller and Marc Wagner. Towards Improving Interactive Mathematical Authoring by Ontology-driven Management of Change. In Alexander Hinneburg, editor, *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) conference proceedings*, pages 289–295, 2007.

[RK08] Florian Rabe and Michael Kohlhase. A web-scalable module system for mathematical theories. Manuscript, to be submitted to the Journal of Symbolic Computation, 2008.

[UCI+06] Victoria Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a state of the art. *Web Semantics: Science, Services, and Agents on the World Wide Web*, 4(1):14–28, Jan. 2006.

[Win06] Terry Winograd. The spreadsheet. In Terry Winograd, John Bennett, Laura de Young, and Bradley Hartfield, editors, *Bringing Design to Software*, pages 228–231. Addison-Wesley, 1996 (2006).

[Wol02] Stephen Wolfram. *The Mathematica Book*. Cambridge University Press, 2002.