A Model for Representing Variational Spreadsheets

Martin Erwig
Oregon State University
erwig@eecs.oregonstate.edu

Duc Le
Oregon State University
ledu@eecs.oregonstate.edu

Eric Walkingshaw
Oregon State University
walkiner@eecs.oregonstate.edu

***Abstract*—TBD**

I. Introduction

New motivation needed

We first give a motivating example for VarSheet. Figure 1a shows a conventional spending estimate of a college student. Suppose the student is not happy with it, they would adjust the costs of some categories based on available options. After some adjustments, the student ends up with the spreadsheet in Figure 1b, for which they pay $50 less. In the updated spreadsheet, the housing cost is $50 less since the rented house is further from campus, but that would increase the cost of transportation. The student also decides to pay $100 less on their loan. Considering both versions, the student would probably go after the latter one, and by doing this, they lose the chance to reduce another $50. Had the student chosen the original housing option and kept Loan payment to be $500, the monthly cost would have been $1450.

Figure 1c shows a possible user interface of VarSheet that could support the student in their decision making process. There are three variation points in the spreadsheet. The *Housing&Transportation* categories are related and thus grouped into a red box with a dashed line separating the two available options. In VarSheet, *Housing&Transportation* is called a *dimension*—a choice users have to make. A dimension contains a set of options, each being called a *tag*. *Housing&Transportation*'s tags are *Close* and *Far*. The *Loan Payment* dimension represents a different variation point with two tags *500* and *600* and thus is colored differently (green). The last variation point, the *Total* category, does not contain variational formulas by itself. The formula being used is `SUM(B2:B6)` and is non-variational, yet the cell inherits the variational structures of its referred cells and hence contains four available options. This cell is therefore colored purple to indicate that it contains *induced variation*. Showing all the four alternatives of *Total* gives the student an overview of all the different options they have. Moreover, if the student selects the $1450 alternative, VarSheet will automatically make decisions for *Housing&Transportation* and *Loan Payment*, and displays those decisions and the resulting spreadsheet to them. We call this feature *goal-directed selection*, the process of selecting spreadsheet variants based on certain goals.

The study of variational spreadsheets brings up several insights to current research on software variation. While traditional variation mechanisms focus on either the syntax or semantics domain, spreadsheets' immediate semantics computation expands the application of variational constructs to both domains, enabling the realization of goal-directed selection. In the example above, Loan payment varies syntactically while Total varies semanticallly, and one could even define cells that vary on both domains.

	A	B
1	Category	Monthly cost
2	Housing	500
3	Transportation	50
4	Food	300
5	Entertainment	100
6	Loan payment	600
7		
8	Total	1550

(a) Before

	A	B
1	Category	Monthly cost
2	Housing	450
3	Transportation	150
4	Food	300
5	Entertainment	100
6	Loan payment	500
7		
8	Total	1500

(b) After

	A	B	
1	Category	Monthly cost	
2	Housing	450	500
3	Transportation	150	50
4	Food	300	
5	Entertainment	100	
6	Loan payment	500	600
7			
8	Total	1500	1550
9		1450	1600

(c) A Possible User Interface

Figure 1: A Monthly Spending Spreadsheet

4	Food	Grocery	200
5		Dining	100

Figure 2: A Different Way to Represent the Food Category

Existing variational contructs mainly work on linear or tree structures, which localizes their scope of impact. Spreadsheets have a special two dimensional structure that makes localization hard to achieve. For instance, in Figure 1a, one could replace row 4 by the spreadsheet in Figure 2 and expects this variation introduction to be local. This action unfortunately has a global impact on the spreadsheet's structure and the addresses/values of several unrelated cells. The Monthly Cost column has to be shifted to column C, while the Total column has to be shifted one row down. In our variational spreadsheet model, we provide mechanisms to localize the effect of structural changes.

Lastly, by letting users actively define variation in spreadsheets, we remove the need for using spreadsheet diffing algorithm, which can be imperfect and misleading at times.

II. Background and Related Work

To be rewritten

Talk about the prototype of the VLHCC 2011 paper

Existing empirical research demonstrates the need for an effective approach to deal with spreadsheet variation. Spreadsheet reusing is common, but users often have to choose from various options [citation needed]. Once a spreadsheet is chosen and several modifications have been made to it, if users recognize that it was not the right one to begin with, they will have to start all over again on a different one. This could happen for many times until users are happy with their choice. To mitigate this problem, VarSheet gives users the ability to modify multiple versions/spreadsheets at the same time on a single representation and select a desired version later. Another reason for spreadsheet variation is due to spreadsheet errors and debugging. Spreadsheets contain errors [7] [8], many of which are introduced in the process of reusing and modifying existing spreadsheets. When debugging, users often need to show the differences between multiple versions, so a framework for systematically managing changes is needed.

In the area of spreadsheet change support, existing tools can be classified into two big categories: change tracking tools and spreadsheet diffing tools. One representative example of a change tracking tool is Microsoft Excel's change tracking feature, which provides users the ability to track spreadsheet edits such as inserting rows, updating equations, etc. This tool is useful and effective in helping users understand versioning information of spreadsheets but is not without limitation. The entire variational spreadsheet is represented using only the time dimension. It is not possible to group changes into categories or groups such that they can be undone or applied again. Another problem arises when two or more users copy and modify the same original spreadsheet. When trying to merge the modified copies, it is unclear which change includes or excludes other changes. For VarSheet, grouping changes could simply be resolved by changing dimension/choice names, which will be defined in the next sections. Research and commercial tools for diffing spreadsheets are prevalent, including CC DiffEngineX [1] and Synkronizer [2]. These tools are effective in comparing spreadsheets and producing accurate results. However, they do not reveal the original purposes of users' changes and do not provide a way to document those.

On a broader topic, there has been extensive research on the topic of representing and managing software variation. The two big pillars of this topic is the compositional approach [6], which modularizes software product line features [4] into individual folders and describes variability at a higher level using feature models [3], and the annotated approach [6], where variability is encoded and represented inside source code. Since there are advantages and disadvantages for each approach, Erwig and Walkingshaw [5] designed the Choice Calculus to shorten the gap between them and to take advantage of the approaches' benefits. The Choice Calculus's design is based on the idea that software variation should be done at both source code and higher levels with not-too-restrictive and not-too-relaxed constraints, making it highly applicable for tree-like structures. VarSheet expands Choice Calculus's ideas of dimensions and choices to the spatial, two dimensional structure of spreadsheets.

III. Dimensions and Decisions

A *dimension* describes one way in which something varies. For example, *Housing&Transportation* in our example in Figure 1 is one dimension of variation. A *dimension definition* assigns a *dimension name* to a non-empty set of *tags*, which correspond to the alternatives in that dimension. A dimension definition is written as $D = \{t_1, \ldots, t_n\}$, for example, $Housing\&Transportation = \{Close, Far\}$. A *qualified tag* is a tag prefixed by the name of the dimension it is taken from, written $D.t_i$. Qualified tags are used to distinguish between tags of the same name from different dimensions. Given a qualified tag $q = D.t$, we can extract the dimension name with the function $dim(q) = D$.

A *decision space* of *degree* n is given by a set of n dimension definitions $D^n = \{D_1 = T_1, \ldots, D_n = T_n\}$ where T_i is the set of tags for dimension D_i. We define the function $dims(D^n) = \{D_1, \ldots, D_n\}$ to return the set of all dimension names in a decision space. The *tag universe* of decision space D^n, written Q_{D^n}, is the set of all qualified tags in D^n, defined as $Q_{D^n} = \{D.t \mid D \in dims(D^n) \land t \in D\}$.

A *decision* in a decision space D^n is a set of qualified tags $\delta \subseteq Q_{D^n}$ that contains at most one tag for each dimension, that is, $q, q' \in \delta \implies q = q' \lor dim(q) \neq dim(q')$. We overload the function *dims* to also denote the dimension names of a decisions, that is, $dims(\delta) = \bigcup_{q \in \delta} dim(q)$. A decision $\delta \subseteq Q_{D^n}$ is *complete* if it contains a qualified tag from every dimension in D^n, that is, if $dims(\delta) = dims(D^n)$, otherwise it is *partial*.

IV. Variational Spreadsheets

A variational spreadsheet represents a family of related plain spreadsheets, each being called a *variant*. Figure 1a and 1b show two of the four variants of the monthly spending spreadsheet. Each variant contains a subset of a universe set of *variational cells*.

Variational cells encode information about which variants the cells belong to, where they appear in the two-dimentional grid structure, and what types of content they store. Each variational cell has an unique address a from the set A. We define a function $V : A \to 2^{Q_{D^n}}$ to map the address to a subset of the tag universe. The purpose of V is to decide whether a cell belongs to a variant. Each cell also contains a *formula* $f \in F$, which can be a value, an address reference, or a function on other formulas.

$$
\begin{array}{rcll}
f \in F & ::= & v & \textit{values} \\
 & | & a & \textit{identity references} \\
 & | & \omega(f, \ldots, f) & \textit{functions}
\end{array}
$$

In the above definition, F represents the set of all formulas, v ranges over primitive values (integers, etc.), and ω stands for the set of all possible functions on formulas. We define a function $\varphi : A \to F$ to map cell addresses to formulas and a function $\pi : A \to P$ to generate *relative positions* given cell addresses. Relative positions $p = (\mathbb{N}, \mathbb{N})$ are stored as pairs of natural numbers and are used to aid a pretty printing algorithm in computing cells' absolute positions on two-dimentional grids. The first number in p represents relative vertical positions while the second represents horizontal positions.

A variational spreadsheet combines a decision space D^n and the universe set of variational cells and is defined by the tuple (D^n, V, φ, π).

In Figure 3 we provide two variants of a variational spreadsheet containing a single dimension $D = \{D.1, D.2\}$. Relative positions are shown on the top-right corner of each cell, and cell addresses are shown on the top-left corner. The universe set of cells contains all the cells with addresses #1–10. We leave the contents of several cells blank as they are not important for our discussion. The cell at address #2's formula is `#5 + 1`, yet the formula is pretty printed as `C1 + 1` and `D1 + 1` since cell #5's grid position changes from one variant to another. Two different types of cells exist in Figure 3, *non-variational* and *variational* cells. Non-variational cells are cells that appear in all variants whereas variational cells do not. Non-variational cells' tags are the tag universe set Q_{D^n}, and variational cells' tags are proper subsets of Q_{D^n}. We provide the cells' tags below.

$$\begin{array}{ll} \#1,\#2,\#5,\#6 & : \{D.1, D.2\} \\ \#3,\#4 & : \{D.1\} \\ \#7,\#8,\#9,\#10 & : \{D.2\} \end{array}$$

The pretty printing algorithm places the cell with the lowest horizontal and vertical positions at the position `A1` and recursively adds cells to the grid based on the following conventions.

- Cells with the same vertical position are on the same column. Cells with the same horizontal position are on the same row.
- For all pairs of cells x and y, if x's horizontal position is less than y's, x's row number has to be less than y's.
- For all pairs of cells x and y, if x's vertical position is less than y's, x's column number has to be less than y's.

V. Semantics

The pretty printing algorithm works on individual variants, which are selected by picking a subset of cells from the cell universe. A natural question is how do we know which cell to pick. To answer this question, in this section we describe *variation semantics*, a mapping between complete decisions and spreadsheet variants.[1]

The steps involved in computing variation semantics are: (1) generating all complete decisions, (2) performing *tag selection* on those decisions to mark several cells as excluded (✗), and (3) collect all remaining cells into variants and map corresponding decisions to those variants.

Dimensions and Complete Decisions

The set of complete decisions of a variational spreadsheet s is defined below with D^n being s's decision space.

$$\begin{aligned} decisions(s) = \{\{D_1.t_1, \ldots, D_n.t_n\} \mid \{D_1, \ldots, D_n\} = dims(D^n) \\ \wedge \forall i \in \{1 \ldots n\}, t_i \in D_i\} \end{aligned}$$

Tag Selection

Tag selection applies complete decisions to variational spreadsheets and mark several cells as excluded (✗). First, each cell at address a's *label* is instantiated as the cell's tags $V(a)$. During the tag selection process, the cell's label either becomes ✗ or get reduced to a smaller set of tags. The set of labels L is defined as $L = 2^{Q_{D^n}} \cup \{✗\}$.

The single tag selection operation $\lfloor\rfloor^s : L \times Q_{D^n} \to L$ is defined as

$$\begin{aligned} \lfloor ✗ \rfloor^s_t &= ✗ \\ \lfloor l \rfloor^s_t &= \begin{cases} \{t' \mid t' \in l \wedge dim(t') \neq dim(t)\} & \text{,if } t \in l \\ ✗ & \text{,otherwise} \end{cases} \end{aligned}$$

The complete tag selection operation $\lfloor\rfloor : L \times 2^{Q_{D^n}} \to L$ is then defined as

$$\begin{aligned} \lfloor l \rfloor_{\varnothing} &= l \\ \lfloor l \rfloor_{\{t\} \cup ts} &= \lfloor \lfloor l \rfloor^s_t \rfloor_{ts} \end{aligned}$$

[1]Note that we ignore the discussion about the semantics of individual variants since they are basically the semantics of plain spreadsheets.

Variation semantics is thus defined as

$$VS(s) = \{\lfloor s \rfloor_{ts} \mid ts \in decisions(s)\}$$

VI. Language Properties

VII. Conclusion and Future Work

TBD

References

[1] Florencesoft diffenginex compares microsoft excel worksheets. http://www.florencesoft.com/exceldiff.html, 2012.

[2] Synkronizer compares excel files faster than you can. http://www.synkronizer.com/, 2012.

[3] D. Batory. Feature Models, Grammars, and Propositional Formulas. In *Int. Software Product Line Conf.*, volume 3714 of *LNCS*, pages 7–20. Springer-Verlang, 2005.

[4] P. C. Clements and L. M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, 2001.

[5] M. Erwig and E. Walkingshaw. The Choice Calculus: A Representation for Software Variation. *ACM Trans. on Software Engineering and Methodology*, 2011.

[6] C. Kästner and S. Apel. Integrating Compositional and Annotative Approaches for Product Line Engineering. In *GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering*, pages 35–40, 2008.

[7] R. R. Panko. What we know about spreadsheet errors. *Journal of End User Computing*, 10:15–21, 1998.

[8] S. G. Powell, K. R. Baker, and B. Lawson. A critical review of the literature on spreadsheet errors. *Decision Support Systems*, 46(1):128 – 138, 2008.

	A	B	C
1	1 (1,1)	3 (2,1)	5 (4,1)
2	2 (1,2) =C1+1	4 (2,2)	6 (4,2)

(a) Variant 1

	A	B	C	D
1	1 (1,1)	7 (2,1)	9 (3,1)	5 (4,1)
2	2 (1,2) =D1+1	8 (2,2)	10 (3,2)	6 (4,2)

(b) Variant 2

Figure 3: Two variants of a variational spreadsheet