

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

In this paper we address the problem of working with variation in spreadsheets. Variation in software is an area of growing interest [?], [?], [?]. One reason is that the demands on modern software to adapt to different environments is constantly increasing. The individualization of software to the needs of a particular company, person, or even with regard to when or where it is used is one important driving force. As a software for particular application domains, spreadsheets are no different and also face the need to be flexible and adaptable.

So far the question of variation in spreadsheets has only been touched tangentially in research. There has been some work on computing differences between spreadsheets [?], [4], but this work has focused on identifying variation and does not offer solutions to the problem of how to represent, work with, and reason about variation on spreadsheets.

In this paper we propose a formal model for variational spreadsheets, analyze some of its formal properties, and demonstrate how this model can be used as a basis for future spreadsheet systems. Specifically, we will demonstrate how we can systematize the work with variation through the two concepts of *dimension* and *choice*. This systematic approach to the representation of variation in spreadsheets allows the identification of consistency criteria that can be exploited to develop tools for finding errors in spreadsheets. Moreover, we will illustrate how variational spreadsheets can support the exploratory design of spreadsheets. In many situations the definition of a spreadsheet model offers alternative ways of representing data and formulas, and a decision of what the best way is to do that is often not clear. Variations allow the spreadsheet designer/programmer to develop different model variants in parallel and defer the decision to a later point.

Continue with an example. Make sure that each aspect of the example points to a technical problem/issue and say where in the paper this issue will be addressed.

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,

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

(a) Before

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

(b) After

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

(c) A Possible User Interface

Figure 1: A Monthly Spending Spreadsheet

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 $\text{SUM}(B2:B6)$ 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

4	Food	Grocery	200
5		Dining	100

Figure 2: A Different Way to Represent the Food Category

domains, enabling the realization of goal-directed selection. In the example above, Loan payment varies syntactically while Total varies semantically, and one could even define cells that vary on both domains. Existing variational constructs 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 [9] [10], 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 [7], which modularizes software product line features [5] into individual folders and describes variability at a higher level using feature models [3], and the annotated approach [7], where variability is encoded and represented inside source code. Since there are advantages and disadvantages for each approach, Erwig and Walkingshaw [6] 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 $D = \{t_1, \dots, t_n\}$, 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 $D^n = \{D_1 = T_1, \dots, D_n = T_n\}$ where T_i is the set of tags for dimension D_i . We define the function $dims(D^n) = \{D_1, \dots, D_n\}$ to return the set of all dimension names in a decision space. The *tag universe* of decision space D^n , written Q_n , is the set of all qualified tags in D^n , defined as $Q_n = \{D.t \mid D \in dims(D^n) \wedge t \in D\}$.

A *decision* in a decision space D^n is a set of qualified tags $\delta \subseteq Q_n$ that contains at most one tag for each dimension, that is, $q, q' \in \delta \implies q = q' \vee dim(q) \neq dim(q')$. 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 $\delta \subseteq Q_n$ 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* S .

Variational cells encode information about which variants the cells belong to, where they appear in the spreadsheet grid structure, and what types of content they store. Each variational cell has a unique address a from the set A . We define a label

function $V : A \rightarrow 2^{Q_n} \cup \{\mathbf{X}\}$ to label each cell with either a decision δ or the \mathbf{X} symbol to provide support for defining variational spreadsheets' semantics. V helps 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.

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

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 $\varphi : A \rightarrow F$ to map cell addresses to formulas and a function $\pi : A \rightarrow P$ to generate *spatial embeddings* given cell addresses. Spatial embeddings $p = (\mathbb{N}, \mathbb{N})$ are stored as pairs of natural numbers and are used to aid a pretty printing algorithm in computing cells' concrete embeddings or positions on the two-dimensional grid. The first number in p represents relative vertical embeddings while the second represents horizontal embeddings.

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 embedding 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 concrete embedding 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 empty set \emptyset , and variational cells' tags are either partial or complete decisions δ . We provide the cells' tags below.

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

Note that we can always allow tags like $\{A.1, B.1, B.2\}$ since, due to the semantics definitions, $\{A.1, B.1, B.2\}$ is the same as $\{A.1\}$ in the case the B has only two tags. If B has more than two tags, $\{A.1, B.1, B.2\}$ is “syntactic sugar” for the two labels $\{A.1, B.1\}$ and $\{A.1, B.2\}$

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

- Cells with the same vertical embedding are on the same column. Cells with the same horizontal embedding are on the same row.
- For all pairs of cells x and y , if x 's horizontal embedding 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 embedding 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 cells to pick. To answer this question, in this section we describe *variation semantics*, a mapping between complete decisions and spreadsheet variants.¹ Each variant corresponds to a complete decisions δ , and a cell c belongs to the variant if $V(c) \subseteq \delta$. For instance, a cell whose label is $\{A.1\}$ belongs to all the variants whose decisions are supersets of $\{A.1\}$.

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 (\mathbf{X}), 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.

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

Tag Selection

Tag selection applies complete decisions to variational spreadsheets and mark several cells as excluded (\mathbf{X}). 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 \mathbf{X} or get reduced to a smaller set of tags. The set of labels is $L = 2^{Q_n} \cup \{\mathbf{X}\}$.

The single tag selection operation $\llbracket \cdot \rrbracket : L \times Q_n \rightarrow L$ is defined as

$$\llbracket \mathbf{X} \rrbracket_t = \mathbf{X} \\ \llbracket l \rrbracket_t = \begin{cases} l \setminus \{t\} & , \text{if } t \in l \\ l & , \text{if } t \notin l \wedge \text{dim}(t) \notin \text{dims}(l) \\ \mathbf{X} & , \text{otherwise} \end{cases}$$

We overload the complete tag selection operation $\llbracket \cdot \rrbracket : L \times 2^{Q_n} \rightarrow L$ as below.

$$\begin{aligned} \llbracket l \rrbracket_{\emptyset} &= l \\ \llbracket l \rrbracket_{\{t\} \cup ts} &= \llbracket \llbracket l \rrbracket_t \rrbracket_{ts} \end{aligned}$$

Variation semantics is thus defined as

$$VS(s) = \{ (ts, \{c \mid c \in s \wedge \llbracket V(c) \rrbracket_{ts} \neq \mathbf{X}\}) \mid ts \in \text{decisions}(s) \}$$

With the semantics defined above, we can even remove the condition $ts_1 \cap ts_2 = \emptyset$

Lemma 1. If $ts_1 \cap ts_2 = \emptyset$,

$$\llbracket l \rrbracket_{ts_1 \cup ts_2} = \llbracket \llbracket l \rrbracket_{ts_1} \rrbracket_{ts_2} = \llbracket \llbracket l \rrbracket_{ts_2} \rrbracket_{ts_1}$$

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

	A	B	C
1	1 (1,1)	3 (2,1)	5 (4,1)
2	2 =C1+1 (1,2)	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 =D1+1 (1,2)	8 (2,2)	10 (3,2)	6 (4,2)

(b) Variant 2

Figure 3: Two variants of a variational spreadsheet

VI. VARIATIONAL CHOICES

While each cell in a variational spreadsheet is labeled individually, the entire spreadsheet exhibits a common pattern—several sets of cells are alternatives of one another. For instance, in Figure 3, $\{ \#3, \#4 \}$ is an alternative of $\{ \#7, \#8, \#9, \#10 \}$. Since presenting cells and tags in a linear fashion does not reveal this relationship, in this section we define the notion of *choices* to explicitly represent alternatives.

A choice denotes one or more dimensions/decisions to be made and contains a set of alternatives, each being a pair of a decision and a set of cells. Formally, choices are defined as a pair (ds, as) satisfying the following conditions.

I did not include the position conditions in this section since they seem to be a bit arbitrary. We can add those later if we want to

$$\begin{aligned}
ds &\subseteq \text{dims}(D^n), ds \neq \emptyset \\
as &= \{ (\delta_1, s_1), \dots, (\delta_k, s_k) \} \\
\forall i \in \{1..k\}, s_i &\subseteq S, \forall i \neq j, \delta_i \neq \delta_j \\
\forall i \in \{1..k\}, \forall c \in s_i, V(c) &\subseteq \delta_i \\
|as| &= \prod_{D_i \in ds} |D_i|
\end{aligned}$$

Each element in as represents a complete decision with regard to ds and the alternative obtained by applying that decision. The aforementioned example can be represented as the choice

$$(\{A\}, \{(\{A.1\}, \{ \#3, \#4 \}), (\{A.2\}, \{ \#7, \#8, \#9, \#10 \})\})$$

To simplify this notations, from now on we assume each dimension contains two tags, $\forall A, A = \{A.1, A.2\}$, and we use the notation $A\langle s_1, s_2 \rangle$ to represent $(\{A\}, \{(\{A.1\}, s_1), (\{A.2\}, s_2)\})$. Using the same convention, the notation $A\langle B\langle s_{11}, s_{12} \rangle, B\langle s_{21}, s_{22} \rangle \rangle$ is equivalent to the choice

$$\begin{aligned}
&(\{A, B\}, \{(\{A.1, B.1\}, s_{11}), (\{A.1, B.2\}, s_{12}), \\
&(\{A.2, B.1\}, s_{21}), (\{A.2, B.2\}, s_{22})\})
\end{aligned}$$

Any variational spreadsheet can be partitioned into sets of choices and non-variational cells. In the following spreadsheet, every cell is represented by a tuple of three elements: address, label, and position. The set of five cells can be split into two choices $A\langle \{ \#1 \}, \{ \#2 \} \rangle$ and $A\langle B\langle \{ \#3 \}, \emptyset \rangle, B\langle \emptyset, \#4 \rangle \rangle$ and a set $\{ \#5 \}$.

$$\begin{aligned}
&(\#1, \{A.1\}, (1,1)) \\
&(\#2, \{A.2\}, (1,1)) \\
&(\#3, \{A.1, B.1\}, (2,1)) \\
&(\#4, \{A.2, B.2\}, (2,1)) \\
&(\#5, \emptyset, (3,1))
\end{aligned}$$

We can talk about the algorithm to derive the set of choices here, but it's really simple

This enables us to look at spreadsheets from a different angle where choices and sets of non-variational cells are combined into variational spreadsheets. We define the \uplus operator to combine choices and sets. The above spreadsheet can be represented as

$$A\langle \{ \#1 \}, \{ \#2 \} \rangle \uplus A\langle B\langle \{ \#3 \}, \emptyset \rangle, B\langle \emptyset, \#4 \rangle \rangle \uplus \{ \#5 \}$$

\uplus is commutative, associative and performs similarly to set union when being applied to two sets.

Given this *choice-directed* representation of spreadsheets, we can define an alternative way to compute variation semantics. For every complete decision δ , each choice in the choice-directed representation is resolved to an alternative, which is simply a set. For example, given the decision $\{A.1, B.1\}$, $A\langle \{ \#1 \}, \{ \#2 \} \rangle$ is resolved to $\{ \#1 \}$, while $A\langle B\langle \{ \#3 \}, \emptyset \rangle, B\langle \emptyset, \#4 \rangle \rangle$ is resolved to $\{ \#3 \}$. We can then compute the union of the produced alternatives and the set of non-variational cells to form the corresponding variant for the decision δ .

We can formalize this semantics if we want. This shouldn't be too hard

We can now talk about several properties of choices and variants. The transformations below are semantics-preserving. Note that all the conditions for choices mentioned above have to be satisfied.

$$A\langle s, s \rangle = s$$

This transformation is also true when $A\langle s, s \rangle$ is an alternative of a different choice B .

$$s_1 \uplus A\langle a_1, a_2 \rangle = A\langle s_1 \uplus a_1, s_2 \uplus a_2 \rangle$$

In this equation, a_1 and a_2 can either be sets of cells or choices.

$$A\langle a_1, a_2 \rangle \uplus B\langle a_3, a_4 \rangle = \begin{cases} A\langle a_1 \uplus a_3, a_2 \uplus a_4 \rangle, & \text{if } A = B \\ A\langle B\langle a_1 \uplus a_3, a_1 \uplus a_4 \rangle, B\langle a_2 \uplus a_3, a_2 \uplus a_4 \rangle \rangle, & \text{otherwise} \end{cases}$$

VII. VARIATION DESIGN THEORY

In this section we provide a list of principles/guidelines for variation languages in general and for our specific VarSheet language. We will also provide how the design of VarSheet follows the principles and guidelines.

(1) Any variation model should provide ways to encode the variation structure. In our model, the variation structure is encoded in the decision space D^n as well as choices.

(2) Any variation model should enable collaborative decision process, that is, dimensions or decisions can be made independently. There is no ordering constraints on decisions. Whether a decision in D is made before dimension D' or vice versa should not change the variation semantics. Lemma 1 verifies this principle.

(3) Variation has to be composable. Given two variational spreadsheets, we should be able to compose them into a single one. In our model, since spreadsheets are represented as sets of cells, composing them requires renaming, matching of dimensions and the set union operation. We can also use the \uplus operator to combine choice-directed representation of spreadsheets.

Another spreadsheet composing operator that our model supports is the *alternative composition* operation. Given two spreadsheets s_1 and s_2 , we can add $\{A.1\}$ to all of s_1 's cell labels and $\{A.2\}$ to all of s_2 's cell labels to produce a choice $A\langle s_1, s_2 \rangle$ with A being a fresh dimension, $A \notin \text{dims}(s_1), A \notin \text{dims}(s_2)$.

(4) Any variation model should allow automatic diffing between variants. Given two variants v_1 and v_2 and the corresponding decision δ_1 and δ_2 in the variational spreadsheet s , the set of common cells is

$$v_1 \cap v_2 = \{c \mid c \in s \wedge V(c) \subseteq \delta_1 \cap \delta_2\}$$

The set of cells in v_1 and not in v_2 is

$$v_1 \setminus v_2 = \{c \mid c \in s \wedge V(c) \subseteq \delta_1 \wedge V(c) \not\subseteq \delta_2\}$$

(5) Variational spreadsheets should enable structural variation, in which choice alternatives have different widths and heights. For instance, in Figure 3, the choice A contains one 2-by-1 alternative and one 2-by-2 alternative.

(6) Variational spreadsheets should localize the effect of structural variation, which might change cell embeddings and thus change cell references. For instance, in Figure 3, #5's embeddings are C1 and D1 in different variants, affecting the reference in #2. In our model, the reference in #2 is not affected by the structural choice since we store address references. #2 always points to #5, and the pretty printer ensures embedding references are shown correctly.

(7) Variational spreadsheets should allow filtering variants based on certain criteria. The variant filtering operation is applied using tag selection and is the basis for defining goal-directed selection. Goal-directed selection is the selection of all variants containing a certain cell. Given a variational spreadsheet s and a cell c , the set of s 's variants containing c is represented by the spreadsheet.

$$(D^n \setminus \text{dims}(V(c)), \{c' \mid c' \in s \wedge [V(c')]_{V(c)} \neq \emptyset\})$$

(8) Variational spreadsheets should allow easy integration into existing spreadsheet tools such as Excel. In our model, the dimension space can be represented by a dimension panel as in Figure 4. Each variant can be stored in a separated tab (in Excel for example). To select a variant, users make decisions in the dimension panel to show the corresponding tab. Our previous work [8] provides evidences that using dimension panel and showing individual C code variants improved program comprehension.

(9) Variational spreadsheets should allow analysis operations on cells. Several operations are supported in our models:

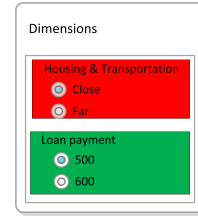


Figure 4: Dimension Panel

computing number of variants, checking exclusion relationship between cells, computing the most selective dimensions, etc.

(10) Our model of variational spreadsheets should be general enough to be applicable in other domains such as variational trees and graphs.

(11) Variational spreadsheets should provide set of transformation and equivalent rules to help reasoning about variation as well as provide chance for improving variational structure of spreadsheets. Several transformation laws were mentioned in section VI to support grouping and distributing cells over choices and removing spurious choices..

VIII. CONCLUSION AND FUTURE WORK

TBD

REFERENCES

- [1] Florecesoft diffenginex compares microsoft excel worksheets. <http://www.florecesoft.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-Verlag, 2005.
- [4] C. Chambers, M. Erwig, and M. Luckey. SheetDiff: A Tool for Identifying Changes in Spreadsheets. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 85–92, 2010.
- [5] P. C. Clements and L. M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, 2001.
- [6] M. Erwig and E. Walkingshaw. The Choice Calculus: A Representation for Software Variation. *ACM Trans. on Software Engineering and Methodology*, 2011.
- [7] 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.
- [8] D. Le, E. Walkingshaw, and M. Erwig. #ifdef confirmed harmful: Promoting understandable software variation. In *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*, pages 143–150, 2011.
- [9] R. R. Panko. What we know about spreadsheet errors. *Journal of End User Computing*, 10:15–21, 1998.
- [10] 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.