UNIVERSITY OF JOENSUU
DEPARTMENT OF COMPUTER SCIENCE
Report Series A

Modeling Spreadsheet Audit: A Rigorous Approach to Automatic Visualization

Jorma Sajaniemi

Report A-1998-5

ACM H.4.1, H.1.2, H.5.2
ISSN 0789-7316
ISBN 951-708-646-6

Modeling Spreadsheet Audit:
A Rigorous Approach to Automatic Visualization

Jorma Sajaniemi

University of Joensuu
Department of Computer Science
P.O. Box 111, FIN-80101 Joensuu, Finland

E-mail: Jorma.Sajaniemi@Joensuu.FI

Abstract:

Computations in spreadsheets are hard to grasp and consequently many errors remain unnoticed. The problem with the hidden errors lies in the invisibility of the structure of calculations. As a result, auditing and visualization tools are required to make spreadsheets easier to comprehend and errors easier to be detected. This paper presents a theoretical model of spreadsheets, and describes various spreadsheet auditing mechanisms employing the model. Moreover, two new visualization mechanisms are introduced.

The model reflects not only current spreadsheet systems but also the way people actually use spreadsheets. Theoretically, it is impossible to check the correctness of a spreadsheet without a formal definition of its computations, but our hope is to find visualizations that point out parts of spreadsheets that contain anomalies, i.e., potential locations of errors. The model helps us to understand how such anomalies can be defined.

Keywords: spreadsheets, auditing, visualization, formal models, user interfaces

Contents

1. Introduction 1

2. The FFR Model 2
   2.1 Strings 2
   2.2 Spreadsheets 3
   2.3 General properties 8
   2.4 Spreadsheet-Use Based Properties 9

3. Auditing Mechanisms 17
   3.1 Excel 7.0 Auditing 17
   3.2 Arrow Tool 19
   3.3 S2 Visualization 20
   3.4 S3 Visualization 22
   3.5 On-line Data Dependency Diagrams 24
   3.6 CogMap 25
   3.7 Spreadsheet Schemes 26

4. Concluding Remarks 27

Acknowledgments 28
References 28

1. Introduction

Spreadsheet calculation is one of the main office automation applications that is widely used in various application areas. Many decisions are based on results obtained by spreadsheets and hence the correctness of computations in spreadsheets is of vital importance. The quality of spreadsheets is, however, poor. Brown & Gould (1987) found that almost half of the simple spreadsheets created by experienced users contained errors. As a consequence, the results given by spreadsheets are often just wrong. But this incorrectness of results goes unnoticed and people use wrong results as if they were correct (see, e.g., Ditlea, 1987).

The problem with the hidden errors lies in the invisibility of the deep structure of spreadsheets. A spreadsheet calculates its results using a large set of formulae, each in its own cell. At one time, it is only possible to see a single formula and the others are not shown. Many spreadsheet systems can display a spreadsheet so that cells show their formulae and not the values yielded by these formulae. Such a view helps to reduce working memory load but the checking of all the formulae is still a demanding and tedious task, and our experiments on spreadsheet debugging have shown that it is not uncommon to judge a formula correct even if there is an obvious error.

Further, when an existing spreadsheet needs to be modified, its internal structure must be understood before any changes can be made. Even if the modification concerns a single cell only, it is hard to find out what other cells depend on that value and should those cells be changed, too. Davis (1996) gives an extensive treatment on the need of auditing tools for spreadsheets.

Spreadsheets are easy to build but their correctness is hard to verify. The apparent ease of creating a spreadsheet seems to make people think that the probability of errors is insignificant. Therefore, they skip verification and errors remain unfound. For this reason it is important that the existence of potential errors could be automatically checked and pinpointed to users. The exact place of an error is not so important because if a user knows that there is an error in a spreadsheet he or she can find it, though it may take some time. So the most important issue is to let a user know whether there are errors. However, if the system is not able to exactly say whether an error exists then it should provide the user an exact hint on the location of the potential error. Otherwise, the user may not find the error fast enough and conclude that the system has only made a false alarm.

Theoretically, it is impossible to check the correctness of a spreadsheet without further knowledge of the purpose of the spreadsheet, i.e., without a formal definition of the computations, because any computation implemented in a spreadsheet can be the wish of some user. But in practice, the computational structures can be visualized in ways that make the presence of potential errors evident. Computations in spreadsheets are not random but follow some rules of "good" structure (Tukiainen & Sajaniemi, 1996). Constant values and formulae form larger structures that are usually harmoniously located and any formula violating these rules can be suspected to contain an error. Therefore we expect that automatically created visualizations of computational structures can be used to highlight anomalies that show that a potential error exists and where it can be found.

Spreadsheet visualization tools (e.g., Davis, 1996; Hendry & Green, 1993; Isakowitz & al., 1995; Lakshmanan & al., 1998; Ronen & al., 1989) have been developed for various purposes: error detection, debugging, comprehension, documenting, etc. As a result, the amount of user assistance may vary from tool to tool. For example, Lakshmanan & al. (1998) ask the user to specify the lay-out of the spreadsheet in order to build semantic abstractions of the spreadsheet, Isakowitz & al. (1995) ask the user to identify and name the functional relations (in the sense of relational data bases) that make up the schema behind a spreadsheet, and Hendry & Green (1993) ask the user to build the whole visualization.

In the following, we will be interested in auditing tools that visualize the deep structure of spreadsheets at a given moment for the purposes of error detection, debugging, and comprehension of spreadsheets. We hope that auditing mechanisms would have the following properties:

- the visualization should be superimposed on the spreadsheet display: users may find it tedious and confusing to determine the correspondence between separate visualization and the spreadsheet itself (Davis, 1996), so superimposition is needed
- the visualization does not depend on input values: as the structure of calculations does not depend on specific values, neither should a visualization depend on them
- the visualization can be constructed automatically: users will probably not use tools that require much user intervention

We will first introduce a model of spreadsheet calculation that captures the essential features of spreadsheet systems and applications that are needed in describing the properties of auditing tools. In chapter 3, we will then use the model to describe various spreadsheet auditing mechanisms, both implemented and projected.

2. The FFR Model

This chapter introduces the FFR (formulae, formats, relations) model of spreadsheet calculation that abstracts formally the structure and fundamental features of spreadsheets without paying attention to the detailed semantics of operations and functions. The FFR model is distantly related to the Unisheet model (Hassinen, 1994) which describes a set of requirements that are supposed to guarantee that a spreadsheet is free from errors and contradictions. The FFR model is, however, more general and abstracts a larger set of features.

We will start with some basic definitions concerning strings, and then introduce spreadsheet applications. Spreadsheet systems are not defined explicitly but their properties are covered by spreadsheet applications and two relations: formula equivalence and format equivalence. We will then be interested in properties of formulae and areas. Section 2.3 defines properties that are derived from the way operations, notably copying, are implemented in spreadsheet systems, while section 2.4 deals with properties derived from the way people use spreadsheets.

2.1 Strings

We will use strings to represent, e.g., formulae. To make our model completely defined, we will start with the properties of strings.

Definition 2.1: Let X be an enumerable set. A *string* over the set X is a finite ordered sequence of elements, also called *characters*, of X. The *length* of a string s, denoted by length(s), is the number of characters in the string s. The *empty string* consisting of no characters is denoted by ε. The set of all strings over set X is denoted by X^*.

In the following, we will see that formulae are strings consisting of "ordinary" characters and special characters that represent cell and area references. As the number of cells is theoretically unlimited, the above definition of strings starts with a possibly infinite set of characters.

2.2 Spreadsheets

We will start defining spreadsheets with the definition of formulae. Note, that the definition of a formula does not suppose any spreadsheet where the formula should reside. In practice, formulae do live within spreadsheets only, but we want to abstract formulae as far from spreadsheets as possible.

Definition 2.2: Let W be a finite set of characters. Then a *spreadsheet formula* in W is a (possibly empty) string over the *formula element set*, denoted by $\mathcal{FS}_W$, consisting of the following:

- characters belonging to the set W
- quadruples $<cs,c,rs,r>$, called *cell references*, where c and r are integers, cs and rs may be either $\mathbb{A}$ or $\mathbb{R}$, and if cs is $\mathbb{A}$ (rs is $\mathbb{A}$) then $c>0$ (resp. $r>0$)
- pairs $<cr_1,cr_2>$, called *area references*, where both cr_1 and cr_2 are cell references

The first and second components of a *cell reference* are called collectively a *column reference*, its third and fourth components are called collectively a *row reference*, its second component is called a *column index* and its fourth component is called a *row index*.

A column or row reference (index) is said to be *absolute* if the first (resp. preceding) component is $\mathbb{A}$, otherwise it is said to be *relative*.

For example, the following is a spreadsheet formula with two cell references and a single area reference:

$$=2*<\mathbb{R},3,\mathbb{R},17>*<\mathbb{A},3,\mathbb{R},-3>+@SUM(<<\mathbb{R},11,\mathbb{R},11>,<\mathbb{R},14,\mathbb{R},14>>)$$

Another example of a formula is 32, i.e., constant values are special cases of formulae.

Spreadsheet systems may use different syntax for cell references, and impose syntax restrictions on the form of formulae. As we suppose that in practice all formulae emerge from real spreadsheets, all formulae satisfy these syntax restrictions and there is no need to consider these restrictions in the model.

In spreadsheet systems the terms absolute and relative reference are not properties of the reference itself but properties of the copying action. When a formula is copied from one cell to another, absolute references refer to the same column or row as in the original location, and relative references refer to the column or row with same offset as in the original location.

A reference, whether absolute or relative, can be represented in the formula as a *direct reference* giving the address of the referenced cell or as an *offset reference* giving the offset with respect to the location of the formula. When the formula is copied, references may need to be changed depending of the *reference representation mechanism* adopted. Figure 1 gives the various possibilities.

In the FFR model reference representation mechanism, we have adopted direct reference representation for absolute references and offset representation for relative references (henceforth called *the ADRO representation* for Absolute-Direct & Relative-Offset). As a consequence, copying is simple: the original formula and the copy are exactly same. The chosen reference representation mechanism is different from that in the user interfaces of most spreadsheet systems, i.e., direct reference representation for both absolute and relative references (*the ADRD representation*, Absolute-Direct & Relative-Direct) but that poses no problems: the FFR model does not say anything about user interfaces and any reference representation mechanism may be used for user interfaces while the FFR model is used. In the following, we will use the ADRD representation, i.e., the standard spreadsheet system user interface, in many examples.

Reference \ Representation mechanism	Direct	Offset
Absolute	no changes	must be changed
Relative	must be changed	no changes

Figure 1: Effects of the copying action to references.

The FFR reference representation mechanism isolates formula information from location information. The address of a cell referenced with an absolute reference is known even if the location of the formula is unknown, but address of a cell referenced with a relative reference is unknown until the location of the formula is given. On the other hand, the direction (up, left, ...) of an absolute reference is not known, but the direction of a relative reference is known without knowledge of the location of the formula. Thus, a formula bears direction information (of relative references) even though it hides location information.

If some other reference representation mechanism (direct reference representation for relative references or offset representation for absolute references) had been chosen, the location and direction information contained in a formula would be different. For example, given a formula and its copy (without knowing their location), it would be possible to tell their relative positions in a spreadsheet. Further, if the location of the copy were given, the location of the original formula could be deduced. In the ADRO representation, such deductions cannot be made.

The rationale for the selection of this particular reference representation mechanism is based on our goal to visualize computational relationships, i.e., relationships between formulae. The ADRO representation isolates formula information from location information enabling us to distinguish between visualizing principles that can work with direction information and those that need full location information.

It would be possible to choose any other reference representation mechanism but, as we will later see, the ADRO representation provides a clean theory of formula relationships. In the following, we will point out definitions that could not be transformed to use other referencing representation mechanisms without changing the essential content of those definitions.

We will now turn to other aspects of spreadsheets.

Definition 2.3: Let V be a finite set of characters. A *cell format* in V is a (possibly empty) string consisting of characters belonging to the set V.

For example, the following is a cell format:

099999.99

Just like in the case of formulae, spreadsheet systems may impose syntax restrictions on the form of cell formats. As we suppose that all cell formats emerge from real spreadsheets, they satisfy these syntax restrictions and there is no need to consider these restrictions in the model.

The definition of spreadsheet applications grasps the properties of single spreadsheets: they have a fixed number of columns and rows with a formula and format for each cell. Of course, formulae and formats may be empty.

Definition 2.4: A *spreadsheet application* is a 6-tuple $SA = (V, W, n, m, T, F)$ where

- V and W are finite sets of characters
- $n > 0$ is an integer (*number of columns*)
- $m > 0$ is an integer (*number of rows*)
- T (*format contents*) is a total function $T{:}(1..n, 1..m) \rightarrow V^*$
- F (*formula contents*) is a total function $F{:}(1..n, 1..m) \rightarrow \mathcal{FS}_W{}^*$ where for each spreadsheet formula $F(c,r)$, $1 \le c \le n$, $1 \le r \le m$
  - $1 \le c' \le n$ for each absolute column index c' in $F(c,r)$
  - $1 \le c+c' \le n$ for each relative column index c' in $F(c,r)$
  - $1 \le r' \le m$ for each absolute row index r' in $F(c,r)$
  - $1 \le r+r' \le m$ for each relative row index r' in $F(c,r)$

As an example of a spreadsheet application consider the 6-tuple ({A,…,Z,a,…z,0,…,9,.},{ A,…, Z,a,…z,0,…,9,+,-,*,/,.},2,3, T,F) where the functions T and F are defined as follows:

$T(1,1)$ = AlignLeft	$F(1,1)$ = January
$T(2,1)$ = 099999.99	$F(2,1)$ = 3708.2
$T(1,2) = \varepsilon$	$F(1,2) = \varepsilon$
$T(2,2) = \varepsilon$	$F(2,2) = \varepsilon$
$T(1,3) = \varepsilon$	$F(1,3)$ = Estimate
$T(2,3)$ = 09999999	$F(2,3) = 12 * \langle \mathbb{R}, 0, \mathbb{R}, -2 \rangle$

Let us next define the constituents of spreadsheets: cells and areas, and their basic attributes formats and formulae.

Definition 2.5: A *cell* of a spreadsheet application $SA=(V,W,n,m,T,F)$ is a pair $a=<c,r>$ where c and r are integers, $1\le c\le n$, $1\le r\le m$. The *format* of the cell a is $T(a)$, and the *formula* of the cell a is $F(a)$.

Definition 2.6: An *area* of a spreadsheet application $SA=(V,W,n,m,T,F)$ is a pair $A=<<c_1,r_1>,<c_2,r_2>>$ where $<c_1,r_1>$ and $<c_2,r_2>$ are cells of the spreadsheet application SA, $c_1\le c_2$ and $r_1\le r_2$. The area A is said to *contain* cells $<c,r>$ where $c_1\le c\le c_2$ and $r_1\le r\le r_2$.

Next we will define how formulae refer to cells and areas. The ADRO representation mechanism (as opposed to ADRD) requires that the location of a formula is known in order to know the location of the referenced cells.

Definition 2.7: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $<c_1,r_1>$ and $<c_2,r_2>$ its cells, and $<<c_3,r_3>,<c_4,r_4>>$ its area. The cell $<c_1,r_1>$ *refers to the cell* $<c_2,r_2>$ if $F(<c_1,r_1>)$ contains a cell reference $<cs,c,rs,r>$ such that $c_2=ac$ and $r_2=ar$ where

- ac = (if $cs=\mathbb{A}$ then c else $c+c_1$)
- ar = (if $rs=\mathbb{A}$ then r else $r+r_1$)

or $F(<c_1,r_1>)$ contains an area reference $<<cs,c,rs,r>,<cs',c',rs',r'>>$ such that $cl\le c_2\le cr$ and $rl\le r_2\le rr$ where

- cl = min(if $cs=\mathbb{A}$ then c else $c+c_1$, if $cs'=\mathbb{A}$ then c' else $c'+c_1$)
- cr = max(if $cs=\mathbb{A}$ then c else $c+c_1$, if $cs'=\mathbb{A}$ then c' else $c'+c_1$)
- rl = min(if $rs=\mathbb{A}$ then r else $r+r_1$, if $rs'=\mathbb{A}$ then r' else $r'+r_1$)
- rr = max(if $rs=\mathbb{A}$ then r else $r+r_1$, if $rs'=\mathbb{A}$ then r' else $r'+r_1$)

The cell $<c_1,r_1>$ *refers to* the *area* $<<c_3,r_3>,<c_4,r_4>>$ if $F(<c_1,r_1>)$ contains an area reference $<<cs,c,rs,r>,<cs',c',rs',r'>>$ such that

- c_3 = min(if $cs=\mathbb{A}$ then c else $c+c_1$, if $cs'=\mathbb{A}$ then c' else $c'+c_1$)
- c_4 = max(if $cs=\mathbb{A}$ then c else $c+c_1$, if $cs'=\mathbb{A}$ then c' else $c'+c_1$)
- r_3 = min(if $rs=\mathbb{A}$ then r else $r+r_1$, if $rs'=\mathbb{A}$ then r' else $r'+r_1$)
- r_4 = max(if $rs=\mathbb{A}$ then r else $r+r_1$, if $rs'=\mathbb{A}$ then r' else $r'+r_1$)

In practice, the formula of a cell always yields some *value*, possibly by using values in other cells. The FFR model does not include values because they are not needed for the purposes of visualizations. In the introduction, we postulated that visualizations should not depend on input values and, as a consequence, visualizations should not depend on the results of computations either. We are interested in the structure of computations, and not in the particular values yielded.

Theoretically, it is possible to extend the model to cover values, but then the semantics of formulae should be modeled, too. As formulae can contain any number of cell and area references, such a semantics modeling is a tedious task. Since values are not needed for our purposes, they are excluded from the FFR model.

A typical approach to the *type* concept in spreadsheet systems is to consider types as properties of cell values propagated from cell to cell through formulae. Values of any type can be formatted using the specific format of the cell provided the type and the format match. If they do not match, a type-specific default format is used. For example, in Excel 7.0, the type of a value is independent of the format of the cell where the value resides. A numeric value can be formatted to appear as text or a date, and a text may have the format "numeric with two decimal places" but it is formatted as text. There is a set of rules that govern how a value of certain type is formatted in each case so that errors do not arise. Thus, types are not tied to cell formats.

To decide the type of a cell value, one must know the semantics of formulae. For example, in Excel 7.0, the type of a constant or a formula not containing cell or area references is defined by the content of the formula, e.g., 23 and =5+17 are numeric whereas January is text. The type of a value produced by a formula containing references may depend on the types of the referenced cells, e.g., =A3 results in a numeric value if the value of A3 is 23 and in a string value if the value of A3 is January.

The type of a value may depend not only on the types of the referenced cells but on their values, too. For example, let the cell A2 contain the formula =MID(A1,1,1) which returns the first character of the value of A1. If the value of A1 is 3x then the value of A2 is 3 and its type is numeric, but if A1 contains x3 then the value of A2 is x and its type is text. Interestingly, Excel 7.0 and Applixware Spreadsheet 4.37 treat these cases similarly but differ in the case where the value of the cell A1 is 111. Excel 7.0 selects the first character (i.e., the function MID is defined also for numbers) but Applixware returns an error. However, this difference is not a result of differences in the general approach to types but a difference in the definition of the semantics of the MID function.

Thus, types are not bound to formulae but to cell values and, moreover, the type of a cell value is not directly determined by the associate formula but the types and values of the referenced cells are needed, also. Therefore, at least in Excel 7.0, the *concept of type is a property of the formula contents of spreadsheet applications together with the semantics of the formula language*.

As the FFR model does not cover formula semantics, it cannot handle types in full. On the other hand, we have seen that type is a dynamic property and can change depending on the input values. But we have postulated that visualizations should not depend on input values. Therefore, visualizations should not depend on types, at least when they are dynamic. When we analyze a static situation, i.e., visualize a spreadsheet with given input values, types are static and can be attached to formats, even though this is theoretically a wrong approach.

Spreadsheet systems provide also mechanisms to *name* cells, areas, or sometimes arbitrary collections of cells by giving a name for the collection of the cell and area references. The name can then be used in formulae instead of the named collection of cell references, e.g., =VAT*PRICE where VAT is the name for the cell reference $A$3 and PRICE is the name for the cell reference $B7. Most spreadsheet systems allow names to consist of both absolute and relative references (with the default being absolute). To understand the effects of copying, it is best to think that each name is replaced by the associated cell or area reference in the formula, and that the occurrence of a name in a formula is a user interface aspect only.

The FFR model contains no specific provision for naming as names are just mnemonics for ordinary references. We thus consider names to be part of user interface, only. It is possible, though, that a user gives names to semantically important and consistent areas, and so names may help us in understanding the structures of spreadsheets and effect on the form of the resulting visualization. Apparently, the most important occasions are names containing absolute references only, as they reveal the position of data forming a meaningful entity for the user.

The FFR model can describe names containing absolute references only. This can be done by attaching the name as part of the format information of each cell belonging to the named collection of cells. For example, the cell A3 could have the format 09.99#VAT. We will later see a relation called format equivalence that is used to resolve whether two formats are the same. If names are combined with the format information, the format equivalence must be defined to discard names when comparing two format strings.

Spreadsheet systems provide also mechanisms to *protect* cells in order to prevent accidental changes. Usually, cells containing formulae as well as constant, parameter-like values should be protected, and cells containing input values should not be protected. Empty cells should also be protected unless they are supposed to be later filled with input data.

The FFR model contains no specific provision for describing protection but it can be described as part of the format information just like naming above. For example, the cell A3 could have the format 09.99#VAT! stating that it is protected.

2.3 General Properties

Next we will define properties that are derived from the way operations, notably copying, are implemented in spreadsheet systems.

In the following definitions, we will use relations that concern cell formats, as well as relations between spreadsheet formulae. Even though the context of the definitions is some particular spreadsheet application, we will suppose that all relations are defined for all cell formats and all spreadsheet formulae.

> ***Definition 2.8:*** Let $SA = (V, W, n, m, T, F)$ be a spreadsheet application, a_1 and a_2 its two cells, $\equiv_t$ (*format equivalence*) an equivalence relation on V^*, and $\equiv_f$ (*formula equivalence*) an equivalence relation on $\mathcal{FS}_W^*$. The cells a_1 and a_2 are *copy equivalent with respect to* $\equiv_t$ *and* $\equiv_f$, if $T(a_1) \equiv_t T(a_2)$ and $F(a_1) \equiv_f F(a_2)$.

For example, the cells <1,5> and <2,5> are copy equivalent (with respect to equality as both format equivalence and formula equivalence) in the following case:

$$T(\langle 1,5\rangle) = 099 \qquad F(\langle 1,5\rangle) = 2*\langle \mathbb{R},3,\mathbb{R},17\rangle*\langle \mathbb{A},3,\mathbb{R},3\rangle$$
$$T(\langle 2,5\rangle) = 099 \qquad F(\langle 2,5\rangle) = 2*\langle \mathbb{R},3,\mathbb{R},17\rangle*\langle \mathbb{A},3,\mathbb{R},3\rangle$$

Both the format equivalence and the formula equivalence are supposed to be properties of the spreadsheet system behind our model. Usually both of them are normal equality, but the definition takes a more general approach by allowing some transformations to occur during copying. For example, the spreadsheet system might change the case of letters in function names, etc. Further, if naming is included in cell formats, the format equivalence should discard all naming information. If protection is included in cell formats, the format equivalence should not discard that information unless the copying operation of the spreadsheet system discards protection.

Due to the ADRO representation, copying is a trivial operation: there is no need to make any changes in the formula. Consequently, the definition of copy equivalence is very simple. If some other reference representation mechanism had been chosen, copy equivalence should be defined to consider changes in the absolute or relative references of the formula.

Copying may lead to an invalid spreadsheet application as a relative reference may point outside the application in its new location. The FFR model does not, however, define copying as an operation but as a relation between cells in a (valid) spreadsheet application and, therefore, there is no need to consider such invalid references in the model.

The formula equivalence is a relation between two formulae. Due to the ADRO representation, this relation can check, e.g., the column and row position of an absolute reference and the direction (up, left, ...) of relative references but not vice versa. Had some other reference representation mechanism been chosen, the strength of copy equivalence would be different. For example, in the ADRD representation, it is possible to test whether a formula is copied an even number of cells away from the original cell (i.e., the formula =2*A4 can be defined to be copy equivalent to =2*A6 but not to =2*A5) though one hardly would like to make use of such possibilities. Later, in the case of origin relation, these differences between reference representation mechanisms will be a more important issue.

It is easy to show that copy equivalence is a reflexive, symmetric and transitive relation, i.e., an equivalence relation.

2.4 Spreadsheet-Use Based Properties

In this section, we will define properties of areas that stem from the way people use spreadsheets. There is no pure theoretical basis for these properties but they are based on our understanding of the cognition that directs human spreadsheet construction (e.g., Sajaniemi & Pekkanen, 1988; Saariluoma & Sajaniemi, 1991; Saariluoma & Sajaniemi, 1994).

We will start with similarity of two cells. It states that the cells must either have the same formula or they both must be constants. In practice, people often use cells as calculators, e.g., they enter the formula =365*13 to a cell in the area containing other yearly costs. Thus, input data areas contain a mixture of single numbers and constant formulae, and similarity allows this to occur. Moreover, similarity requires that the formats of the cells must be the same.

> ***Definition 2.9:*** Let $SA = (V, W, n, m, T, F)$ be a spreadsheet application, a_1 and a_2 its two cells, $\equiv_t$ an equivalence relation on V^*, and $\equiv_f$ an equivalence relation on $\mathcal{FS}_W{}^*$. The cells a_1 and a_2 are *similar with respect to* $\equiv_t$ *and* $\equiv_f$, if they are copy equivalent with respect to $\equiv_t$ and $\equiv_f$, or $T(a_1) \equiv_t T(a_2)$ and both $F(a_1)$ and $F(a_2)$ contain no cell (or area) references.

For example, the cells <1,5> and <2,5> are similar, and so are the cells <13,36> and <25,11>, if their formats and formulae are the following:

T(<1,5>) = 099	F(<1,5>) = 2*< $\mathbb{R}$,3,$\mathbb{R}$,17>*< $\mathbb{A}$,3,$\mathbb{R}$,3>
T(<2,5>) = 099	F(<2,5>) = 2*< $\mathbb{R}$,3,$\mathbb{R}$,17>*< $\mathbb{A}$,3,$\mathbb{R}$,3>
T(<13,36>) = 099.99	F(<13,36>) = 36
T(<25,11>) = 099.99	F(<25,11>) = 178*15

It is easy to see that similarity is a reflexive, symmetric and transitive relation, i.e., an equivalence relation.

Similarity is used to define homogeneity: a homogeneous area is an area consisting of similar cells.

Definition 2.10: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, A its area, $\equiv_t$ an equivalence relation on V^*, and $\equiv_f$ an equivalence relation on $\mathcal{FS}_W{}^*$. The area A is *homogeneous with respect to* $\equiv_t$ *and* $\equiv_f$, if all cell pairs a_1, a_2 contained in the area are similar with respect to $\equiv_t$ and $\equiv_f$.

As similarity is a reflexive and transitive relation, an area can be shown to be homogeneous by considering its cells in any order $a_1, a_2, a_3, \ldots$, and checking that cells a_i and a_{i-1} are similar.

A homogeneous area contains either constants or it can be constructed by copying a single formula to all cells in the area. In practice there are, however, many areas that represent a single computation even though the cells contain different formulae. For example, in cumulative sum the formula in the first cell may be different from that of the others. The concept of top-originating areas abstracts and generalizes such calculations.

Definition 2.11: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W{}^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W{}^* \times V^* \times \mathcal{FS}_W{}^* \times V^* \times \mathbf{N}$. The area A is *top-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if there exists an integer k such that all of the following hold:

- $1 \le k \le r_2 - r_1 - 1$
- each area $<<c_1,r_1+r>,<c_2,r_1+r>>$, $0 \le r < k$, is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- area $<<c_1,r_1+k>,<c_2,r_2>>$ is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- $T(<c_1,r_1+r>) \equiv_t T(<c_1,r_1+r+1>)$, $0 \le r < k$
- $\Omega(F(<c_1,r_1+r>),T(<c_1,r_1+r>),F(<c_1,r_1+r+1>),T(<c_1,r_1+r+1>),r)$, $0 \le r < k$
- not $F(<c_1,r_1>) \equiv_f F(<c_1,r_2>)$ or not $T(<c_1,r_1>) \equiv_t T(<c_1,r_2>)$

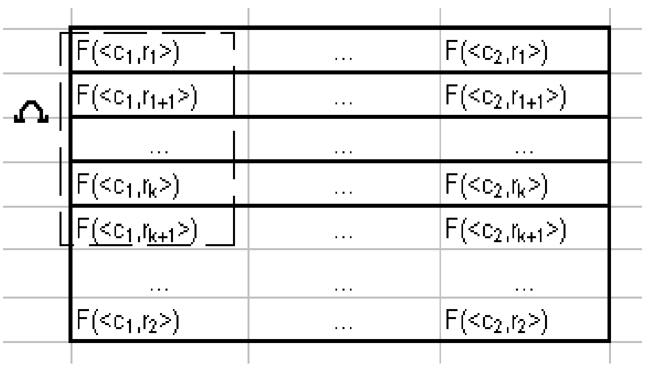

Figure 2: Homogeneous parts of a top-originating area.

Figure 2 illustrates this definition. In a top-originating area, the first row consists of similar cells that take care of the first step of the computation. In a simple case, there is a single column and consequently a single cell, but in the general case there may be multiple equivalent computations, each in its own column. The next row takes care of the second step of the computation, etc. After k steps, the computation must stabilize so that at least two rows at the end of the area must use the same formula. The origin relation Ω tells what kinds of formula series are allowed at the beginning of a computation.

Figure 3 contains two examples of top-originating areas (with the ADRD representation). The area B2:B5 is a cumulative sum. In the notation of the previous definition, k is 1, both $\equiv_t$ and $\equiv_f$ are equality, and $\Omega(f_1,t_1,f_2,t_2,r)$ holds if f_1 can be obtained from f_2 by removing all cell references containing a relative row reference to the cell in the previous row (together with the associated operator). The area G2:H5 is a generalization of cumulative sum with several columns, each computing its own sum.

Figure 3 contains further examples of top-originating areas. The area B8:B13 is the Fibonacci sequence that can be seen to be top-originating by selecting k=2, both $\equiv_t$ and $\equiv_f$ to be equality, and $\Omega(f_1,t_1,f_2,t_2,r)$ to hold if either f_1 contains no cell references or f_1 and f_2 are the same formula.

The area G8:G12 in Figure 3 is a sequence of titles 1980, 1985, 1990, etc. Now k is 1, both $\equiv_t$ and $\equiv_f$ are equality, and $\Omega(f_1,t_1,f_2,t_2,r)$ holds if either f_1 and f_2 are the same formula, or f_2 contains no row references to the cell in the previous row.

	A	B	C	D	E	F	G	H
1								
2	13	=A2		11	14		=D2	=E2
3	7	=A3+B2		12	8		=D3+G2	=E3+H2
4	12	=A4+B3		5	12		=D4+G3	=E4+H3
5	8	=A5+B4		14	7		=D5+G4	=E5+H4
6								
7								
8		1					1980	
9		1					=G8+5	
10		=B8+B9					=G9+5	
11		=B9+B10					=G10+5	
12		=B10+B11					=G11+5	
13		=B11+B12						
14								

Figure 3: Examples of top-originating areas.

An origin relation concerns formulae and formats, but within a top-originating area format equivalence is required, also. Thus, an origin relation is basically a relation between formulae. Formats are included just to cover cases where the format equivalence discards, e.g., naming information, and the origin relation needs this information.

Had some other reference representation mechanism been chosen for the FFR model, the meaning of the previous definition would change notably. Due to the ADRO representation, the formulae in the cells $<c_1,r_1>$, ..., $<c_2,r_1>$, i.e., in cells in the first homogeneous area, are exactly same (presuming formula equivalence is equality). Likewise, the formulae in the cells $<c_1,r_2>$, ..., $<c_2,r_2>$, i.e., in cells in the second homogeneous area, are exactly same Hence $\Omega(F(<c_1,r_1>),T(<c_1,r_1>),F(<c_1,r_1+1>),T(<c_1,r_1+1>),r)$ implies that for each c, $c_1 \le c \le c_2$, also $\Omega(F(<c,r_1>),T(<c,r_1>),F(<c,r_1+1>),T(<c,r_1+1>),r)$ holds. Thus, even though the definition of top-originating areas requires Ω to hold on the first column only, it holds also for the other columns. But if some other reference representation mechanism were in use (like in Figure 3), this implication would not be true as the formulae in different columns would differ.

For example, Figure 4 shows two pairs of cumulative sums, the first one using the ADRO representation mechanism used in the FFR model, and the other using ADRD. In the case of ADRO, the two columns are exactly the same and hence Ω holds for the second column if it holds for the first. But in ADRD, the columns differ. Consider, e.g., Ω defined as " $\Omega(f_1,t_1,f_2,t_2,r)$ holds if both f_1 and f_2 start with =D". Now Ω holds for the first column but not for the second.

If the FFR model were using ADRD, there would be a need to define some kind of "position independence of the origin relation" to overcome the problem. However, with ADRO as the selected reference representation mechanism, the FFR model avoids the problem totally.

Further, as the ADRO representation implies that formulae have no knowledge of their location in the spreadsheet, an origin relation cannot check, e.g., whether two relative references, one in each of the formulae, refer the same cell.

=<ℝ,-3,ℝ,0>	=<ℝ,-3,ℝ,0>
=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>	=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>
=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>	=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>
=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>	=<ℝ,-3,ℝ,0>+< ℝ,0,ℝ,-1>

=D2	=E2
=D3+G2	=E3+H2
=D4+G3	=E4+H3
=D5+G6	=E5+H4

Figure 4: Cumulative sums with the ADRO and ADRD representations.

An origin relation is not a property of the underlying spreadsheet system but a relation we must invent. In the previous examples, the various relations were defined to suit the specific spreadsheet application. As a result, the relation we used with the Fibonacci sequence can be used with the cumulative sum, but not vice versa. Now, an interesting question arises: can we define a single origin relation that will accept as many error-free calculations as possible while at the same time disallowing as many erroneous calculations as possible. Here the term "error" refers to users' intention of the calculation. Therefore, there is no formal way to define how well an origin relation works but empirical research is needed to evaluate various relations.

We will now define some special cases of origin relations. The basic idea is not to accept any kinds of relations but only relations that, when traversing through the formulae from bottom to top, retain other parts of the formulae but cut off references that stop referring to the area itself. For example, in the cumulative sum, the formula in the first cell lacks the reference that refers to earlier subsums in other formulae.

There will be two cases: a start-pruning origin relation cuts off such references together with parts of the formulae around the references, while a start-compressing origin relation cuts off such references and allows any other changes as long as other references remain untouched.

Definition 2.12: An origin relation Ω is *start-pruning* , if for all spreadsheet formulae f_1 and f_2, formats t_1 and t_2, and integer $r \geq 0$ for which $\Omega(f_1,t_1,f_2,t_2,r)$ holds, f_1 can be written in the form $s_1 s_2 \ldots s_q$ and f_2 can be written in the form $s'_0 s_1 s'_1 s_2 s'_2 .. s_q s'_q$ so that each s'_i is either ε or contains a relative row reference $-r$-1, and no s_i contains relative row references $-r$-1.

Definition 2.13: An origin relation Ω is *start-compressing*, if for all spreadsheet formulae f_1 and f_2, formats t_1 and t_2, and integer $r \geq 0$ for which $\Omega(f_1,t_1,f_2,t_2,r)$ holds, f_1 can be written in the form $s'_0 s_1 s'_1 s_2 s'_2 .. s_q s'_q$ and f_2 can be written in the form $s''_0 s_1 s''_1 s_2 s''_2 .. s_q s''_q$ so that

- each s_i is either ε or a cell reference containing either an absolute row reference or a relative row reference different from $-r$-1
- each s'_i contains no row references
- each s''_i contains neither absolute row references nor other relative row references but $-r$-1

For example, the origin relation used above with cumulative sum is start-pruning, e.g., =<ℝ,-3,ℝ,0> can be written as s_1 where s_1 is the whole formula, and =<ℝ,-3,ℝ,0>+<ℝ,0,ℝ,-1> can be written as $s'_0 s_1 s'_1$ where s'_0 is ε, and s'_1 is +<ℝ,0,ℝ,-1>.

The origin relation used with the sequence of titles 1980, 1985, 1990, etc. is not start-pruning but it is start-compressing. For example, 1980 can be written as $s'_0 s_1 s'_1$ where s'_0 is ε, s_1 is ε and s'_1 is 1980, and =<ℝ,0,ℝ,-1> can be written as $s''_0 s_1 s''_1$ where s''_0 is =<ℝ,0,ℝ,-1>, s_1 is ε and s''_1 is ε.

It is easy to show that every start-pruning origin relation is also start-compressing, but not vice versa. The following definitions consider cases where the pruning or compression part is always at most k cells long.

Definition 2.14: A start-pruning origin relation Ω is *k-start-pruning*, $1 \leq k$, if for all spreadsheet formulae f_1 and f_2, formats t_1 and t_2, and integer r, for which $\Omega(f_1,t_1,f_2,t_2,r)$ holds, $r<k$.

Definition 2.15: A start-compressing origin relation Ω is *k-start-compressing*, $1 \leq k$, if for all spreadsheet formulae f_1 and f_2, formats t_1 and t_2, and integer r, for which $\Omega(f_1,t_1,f_2,t_2,r)$ holds, $r<k$.

For example, the origin relation used with cumulative sum is 1-start-pruning.

Earlier we defined top-originating areas to abstract computations where the start of a calculation is at the top of the area. Next we will give definitions for the symmetrical cases.

Definition 2.16: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=\langle\langle c_1,r_1\rangle,\langle c_2,r_2\rangle\rangle$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^* \times V^* \times \mathcal{FS}_W^* \times V^* \times \mathbf{N}$. The area A is *bottom-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if there exists an integer k such that all of the following hold:

- $1 \leq k \leq r_2 - r_1 - 1$
- each area $\langle\langle c_1,r_2-r\rangle,\langle c_2,r_2-r\rangle\rangle$, $0 \leq r<k$, is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- area $\langle\langle c_1,r_1\rangle,\langle c_2,r_2-k\rangle\rangle$ is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- $T(\langle c_1,r_2-r\rangle) \equiv_t T(\langle c_1,r_2-r-1\rangle)$, $0 \leq r<k$
- $\Omega(F(\langle c_1,r_2-r\rangle),F(\langle c_1,r_2-r\rangle),F(\langle c_1,r_2-r-1\rangle),T(\langle c_1,r_2-r-1\rangle),r)$, $0 \leq r<k$
- not $F(\langle c_1,r_1\rangle) \equiv_f F(\langle c_1,r_2\rangle)$ or not $T(\langle c_1,r_1\rangle) \equiv_t T(\langle c_1,r_2\rangle)$

Definition 2.17: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *left-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if there exists an integer k such that all of the following hold:

- $1\le k\le c_2-c_1-1$
- each area $<<c_1+c,r_1>,<c_2+c,r_1>>$, $0\le c<k$, is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- area $<<c_1+k,r_1>,<c_2,r_2>>$ is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- $T(<c_1+c,r_1>)\ \equiv_t\ T(<c_1+c+1,r_1>)$, $0\le c<k$
- $\Omega(F(<c_1+c,r_1>),F(<c_1+c,r_1>),F(<c_1+c+1,r_1>),T(<c_1+c+1,r_1>),c)$, $0\le c<k$
- not $F(<c_1,r_1>)\equiv_f F(<c_2,r_1>)$ or not $T(<c_1,r_1>)\equiv_t T(<c_2,r_1>)$

Definition 2.18: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *right-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if there exists an integer k such that all of the following hold:

- $1\le k\le c_2-c_1-1$
- each area $<<c_1-c,r_2>,<c_2-c,r_2>>$, $0\le c<k$, is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- area $<<c_1,r_1>,<c_2-k,r_2>>$ is homogeneous with respect to $\equiv_t$ and $\equiv_f$
- $T(<c_1-c,r_2>)\ \equiv_t\ T(<c_1-c-1,r_2>)$, $0\le c<k$
- $\Omega(F(<c_1-c,r_2>),F(<c_1-c,r_2>),F(<c_1-c-1,r_2>),T(<c_1-c-1,r_2>),c)$, $0\le c<k$
- not $F(<c_1,r_1>)\equiv_f F(<c_2,r_1>)$ or not $T(<c_1,r_1>)\equiv_t T(<c_2,r_1>)$

An edge-originating area is any of the previous, and a corner-originating area is such that it can be constructed in two directions as a collection of edge-originating areas.

Definition 2.19: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *edge-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if it is either top-originating, bottom-originating, left-originating, or right-originating with respect to $\equiv_t$, $\equiv_f$ and Ω.

Definition 2.20: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *corner-originating with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if both of the following hold:

- area $<<c,r_1>,<c,r_2>>$ is edge-originating with respect to $\equiv_t$, $\equiv_f$ and Ω, $c_1\le c\le c_2$
- area $<<c_1,r>,<c_2,r>>$ is edge-originating with respect to $\equiv_t$, $\equiv_f$ and Ω, $r_1\le r\le r_2$

The area A is called *x-y-originating* if $<<c_1,r_1>,<c_1,r_2>>$ is x-originating (where x is top or bottom) and $<<c_1,r_1>,<c_2,r_1>>$ is y-originating (where y is left or right).

For example, the area D1:F3 in Figure 5 is corner-originating but not edge-originating for any start-pruning relation (unless a very strange formula equivalence is used).

	A	B	C	D	E	F
1	1	1	1	=A1	=B1+D1	=C1+E1
2	1	1	1	=A2+D1	=B2+D2+E1-D1	=C2+E2+F1-E1
3	1	1	1	=A3+D2	=B3+D3+E2-D2	=C3+E3+F2-E2

Figure 5: A corner-originating area.

As an example of a computation that is neither edge-originating nor corner-originating consider the computation of moving average in Figure 6. The reason is that the computation has a starting phase but is has an ending phase as well, and edge-originating areas can have formula differences at one end only.

	A	B
1	30	=(A1+A2)/2
2	20	=(A1+A2+A3)/3
3	28	=(A2+A3+A4)/3
4	20	=(A3+A4+A5)/3
5	20	=(A4+A5+A6)/3
6	22	=(A5+A6+A7)/3
7	17	=(A6+A7)/2

Figure 6: An area that is neither edge-originating nor corner-originating.

We have now defined the forms of areas that we consider to reflect error-free calculations. An area is next defined to be consistent if it is of any of those forms.

Definition 2.21: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, A its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *consistent with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if it is either

- homogeneous with respect to $\equiv_t$ and $\equiv_f$,
- edge-originating with respect to $\equiv_t$, $\equiv_f$ and Ω, or
- corner-originating with respect to $\equiv_t$, $\equiv_f$ and Ω

In visualizations, we are interested to find as large consistent areas as possible. This notion is captured by the definition of maximal areas.

Definition 2.22: Let $SA=(V,W,n,m,T,F)$ be a spreadsheet application, $A=<<c_1,r_1>,<c_2,r_2>>$ its area, $\equiv_t$ an equivalence relation on V^*, $\equiv_f$ an equivalence relation on $\mathcal{FS}_W^*$, and Ω (*origin relation*) a 5-ary relation on $\mathcal{FS}_W^*\times V^*\times\mathcal{FS}_W^*\times V^*\times\mathbf{N}$. The area A is *maximal with respect to* $\equiv_t$, $\equiv_f$ *and* Ω, if A is consistent with respect to $\equiv_t$, $\equiv_f$ and Ω and there is no area A' of the spreadsheet application SA such that all of the following hold:

- A' contains all the cells A contains
- A' contains at least one cell that A does not contain
- A' is consistent with respect to $\equiv_t$, $\equiv_f$ and Ω

It should be noted that given a cell in a spreadsheet, the maximal consistent area that contains the cell is not necessarily unique. For example, in Figure 7, the areas B1:B3 and B3:D3 are both maximal consistent areas even though they both contain the cell B3. In practice, such situations are rare, and a visualization may select any of the possible areas.

	A	B	C	D
1	1	=A1		
2	1	=A2+B1	1	1
3	1	=A3+B2	=B3+C2	=C3+D2
4				

Figure 7: Overlapping maximal consistent areas.

Finally, we define breaking areas as maximal areas that refer to other areas in an anomalous way. Acceptable ways are to refer to all cells of the other area, and to refer cells at one edge (or corner) provided that the computation in the referenced area starts at the opposite edge (or corner).

Definition 2.23: Let $SA = (V, W, n, m, T, F)$ be a spreadsheet application and $A = \langle\langle c_1, r_1\rangle, \langle c_2, r_2\rangle\rangle$ and B its two areas that are consistent with respect to equivalence relations $\equiv_t$ and $\equiv_f$, and an origin relation Ω. The area B *breaks* the area A, if at least one cell in the area B refers to at least one cell in the area A, and either

- A is homogeneous with respect to $\equiv_t$ and $\equiv_f$ and at least one cell in the area A is not referred to by any cell in the area B,
- A is top-originating (bottom-originating, left-originating, right-originating) with respect to $\equiv_t$, $\equiv_f$ and Ω, and B refers to all the cells in the area $\langle\langle c_1, r_2\rangle, \langle c_2, r_2\rangle\rangle$ (resp. $\langle\langle c_1, r_1\rangle, \langle c_2, r_1\rangle\rangle$, $\langle\langle c_2, r_1\rangle, \langle c_2, r_2\rangle\rangle$, $\langle\langle c_1, r_1\rangle, \langle c_1, r_2\rangle\rangle$) but not any other cell of A, or
- A is top-left-originating (bottom-left-originating, top-right-originating, bottom-right-originating) with respect to $\equiv_t$, $\equiv_f$ and Ω, and B refers to $\langle c_2, r_2\rangle$ (resp. $\langle c_2, r_1\rangle$, $\langle c_1, r_2\rangle$, $\langle c_1, r_1\rangle$) but not any other cell of A.

The area B is said to be a *breaking area* if it breaks any area A' and both A' and B are maximal with respect to $\equiv_t$, $\equiv_f$ and Ω.

For example, in Figure 8 there are four maximal areas A1:A4, B1:B2, B3:B3, and B4:B4. Each of the areas on the column B are breaking areas as they all break the area A1:A4. In Figure 9 there are three maximal areas A1:A4, B1:B4, and C4:C4. None of these areas breaks any other area.

	A	B
1	22	=10*A1
2	33	=10*A2
3	44	=440
4	55	=10*A4
5		

Figure 8: Breaking areas.

	A	B	C
1	22	=A1	
2	33	=A2+B1	
3	44	=A3+B2	
4	55	=A4+B3	=10*B4
5			

Figure 9: Non-breaking areas.

3. Auditing Mechanisms

Spreadsheet systems provide users with various tools to trace how values are computed. In the following, we are interested in *auditing mechanisms* that we define to consist of a visualization, a visualization construction process, and other user interface features.

A *visualization* is some graphical representation of a spreadsheet supposed to make the structure of the computations in the spreadsheet more readily available for users. A visualization can be superimposed on the spreadsheet display or it can be a separate entity. In the latter case, the visualization may have some user interface mechanism to make the connection between the visualization and the spreadsheet more evident.

A *visualization construction process* consists of the possibilities given to users to build the visualization in a stepwise manner or instantly. Users may be allowed to govern the construction process by limiting it to some parts of the spreadsheet (usually by selecting the core cells from which the visualization is gradually accumulated), or by giving the direction of the growing process (usually towards referenced cells or towards referencing cells). The final visualization may depend on the order of its construction.

Other user interface features may include, e.g., the use of the visualization to navigate in spreadsheets, or automatic repositioning of the visualization to show the currently active cell.

In the following, we will describe various auditing mechanisms, both implemented and projected. We will use the division described above and end with a statement of implementation status and a discussion part consisting of a comparison of the features of the mechanism with respect to the FFR model.

3.1 Excel 7.0 Auditing

Microsoft Excel 7.0 contains an auditing mechanism intended to help users to "understand the relationships between cells [...] and find mistakes. The tracers [...] graphically display the flow of computations," as the on-line help states.

	A	B	C	D	E	F	G	H
1								
2				25				
3								
4		11		=$D$2*B4/100		=D4		=B4
5		12		=$D$2*B5/100		=F4+D5		=B5+H4
6		13		=$D$2*B6/100		=F5+D6		=B6+H5
7		14		=$D$2*B7/100		=F6+D7		=B7+H6
8		15		=$D$2*B8/100		=F7+D8		=B8+H7
9								
10		=SUM(B4:B8)						=D2*H8/100

Figure 10: An example spreadsheet.

Visualization. Let us consider as an example the spreadsheet in Figure 10. This spreadsheet is shown in Figure 11 with Excel 7.0 total visualization, i.e., all cells have been audited (towards referenced cells). This visualization highlights the data flow by superimposing graphics on the spreadsheet display. If the formula of a cell, say *a*, contains a cell reference to another cell, say *b*, then the visualization has a blue arrow starting from a small blue circle located in the cell *b* and pointing to the cell *a*. In the case of an area reference, the circle is located in the top-left corner cell of the area, the whole area has a blue rectangle around it, and the arrow is thicker.

Arrows are normally blue, but an arrow starting form a cell containing the special error value is red.

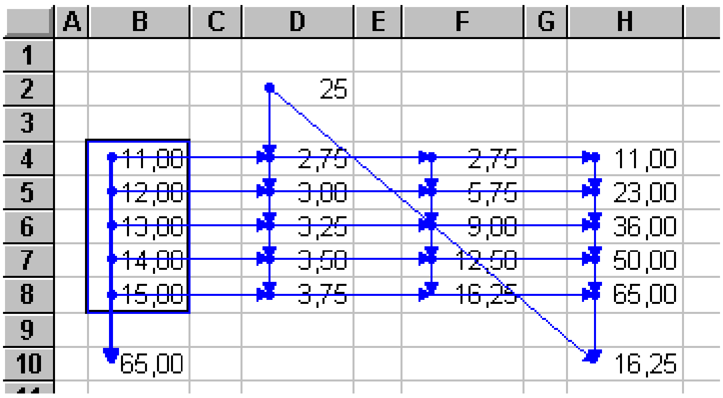

Figure 11: The spreadsheet of Figure 10 with the Excel visualization.

The arrows end and start at the same point within a cell yielding vertical and horizontal lines having arrow heads in those cells containing formulae, and circles in cells that are referenced in those formulae. The location of the start/end point of arrows within a cell is fixed implying that arrows may be on top of others, e.g., arrows D4 →F4 and B4 →H4 in figure 11 .

Representation construction process. The construction of a visualization starts always from a single cell and the user can request visualization construction in either of two directions: towards precedents (i.e., towards referenced cells or areas) or towards dependents (i.e., towards referencing cells). In each step the visualization consists of a set of cells and areas together with arrows connecting them. It is also possible to draw back the arrows added during the latest step, and this process can be repeated to clear the whole visualization.

When the visualization is constructed towards precedents, the system expands the set with new cell and area references in the formulae of the cells and areas in the set, and draws arrows as described above.

When the visualization is constructed towards dependents, the set consists of cells only. In each step, the system expands the set with cells that refer to cells already in the set, and draws arrows. In the case of visualization construction towards dependents, area references are treated as collections of cell references and no area visualization is used.

The cell forming the origin of visualization construction can be changed during the process, and so can the direction be changed, too. The next step is, however, performed just as there would be a new visualization construction process: even though the old visualization is not removed from the screen, the set of cells and areas that is the base of the visualization is reset to contain the new origin cell only. Each arrow tree is treated separately when arrows are drawn back, but the user can also remove all arrows with a single command.

Due to differences in the representation construction process depending on the direction of the process, the total visualization obtained by working towards precedents may differ from that obtained by working towards dependents.

Other features. If a user double-clicks an arrow, a cell at other end of the arrow will become the currently active cell. Thus, users can follow arrows by double-clicking them.

Implementation. The Excel 7.0 auditing mechanism is fully implemented and integrated in the Microsoft Excel 7.0 spreadsheet system.

Discussion. The Excel 7.0 auditing mechanism is based on the cell and area references of formula contents, and on the occurrence of the special error value among the cell values. The mechanism visualizes the following aspects of spreadsheet applications:

- cell references including the referenced cell and the referencing cell
- area references including the referenced area and the referencing cell
- cell references where the referenced cell contains the special error value
- area references where at least one cell in the referenced area contains the special error value

The Excel 7.0 auditing mechanism does not consider other aspects of formula contents, neither does it consider the format contents, naming or protection of spreadsheet applications. Moreover, it pays no attention to any relations between formulae.

3.2 Arrow Tool

Davis (1996) introduces two auditing tools for spreadsheets: the on-line data dependency diagram that will be described later, and the arrow tool. The arrow tool is closely related to the Excel 7.0 auditing.

Visualization. The arrow tool visualization is a variant of the Excel 7.0 visualization. It colors precedent and dependent cells in addition to using arrows. Area references are not treated separately from cell references, and all arrows have the same color.

Representation construction process. The construction of a visualization occurs as with the Excel 7.0 auditing except that the removal of the visualization cannot be done stepwise.

Other features. Arrows can be used for navigation by clicking them like with the Excel 7.0 auditing. Moreover, the arrow tool facilitates cyclic examination, one cell at a time, of a collection of dependent cells or precedent cells.

Implementation. The arrow tool is implemented as Excel macros.

Discussion. The arrow tool is based on the cell and area references of formula contents. The mechanism visualizes cell references including the referenced cell and the referencing cell. It does not consider other aspects of formula contents, neither does it consider the format contents, naming or protection of spreadsheet applications. Moreover, it pays no attention to any relations between formulae.

Davis (1996) has proposed enhancing the arrow tool by labeling each arrow with the reference and the name relating to the cell at the other end of the arrow. This labeling scheme makes references easier to understand especially when the other end of an arrow is outside the screen.

Davis (1996) has also proposed adding new commands to construct the visualization in larger steps: the whole precedents chain, the whole dependents chain, or the total graph with a single command.

3.3 S2 Visualization

We have developed a visualization that highlights the data flow of a spreadsheet as well as areas containing formulae or data that seems to form a logical entity. As a consequence, highlighted areas are supposed to describe the plan structure (Hassinen & al., 1988) of the spreadsheet. Any deviation from this structure shows clearly up in the visualization. The visualization is called S2 and it is the third in a series of visualizations (the former being called S0 and S1 with differences in the visualization of broken areas).

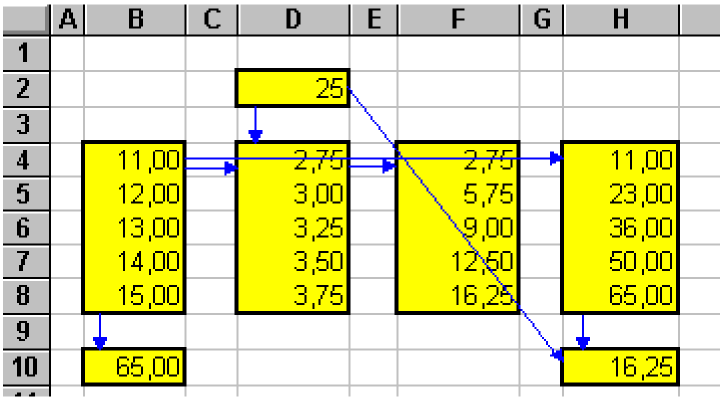

Figure 12: The spreadsheet of Figure 10 with the S2 visualization.

Visualization. Figure 12 contains the same spreadsheet as in Figure 11, but now with total S2 visualization. This visualization marks with yellow coloring and a thick border areas (so called *S areas*) that contain values having equal type and equal format as well as areas having a formula that could be obtained by copying from each other. Moreover, a S area can contain at any edge cell a formula that can be obtained by removing from the copy in the neighbor cell all references to the cells at that edge and copying that formula. For example, The formula in the cell H4 can be obtained from the formula =B5+H4 in the cell H5 by removing the reference to the cell H4 (together with the preceding operator) and then copying this formula to H4.

In the S2 visualization, blue arrows connect areas and not individual cells as in the previous visualizations. There is an arrow from an area *A* to an area *B* if any formula in the area B refers to any cell in the area A. When needed, arrows are shifted a little to prevent overlapping.

If some cells in the area A are not referenced in the area B, then the area A is split into smaller subareas so that in each resulting subarea either each cell is referenced in the area B or no cell is referenced. As a consequence, each cell at the starting area of any arrow is referenced by some formula at the head of the arrow. The area B, that caused the splitting, is colored orange. For example, consider a variant of the previous example spreadsheet having in the cell H7 the formula =D7+H6. In the place of the S area H4:H8 there are now three S areas H4:H6, H7:H7, and H8:H8 (as the formula in the cell H7 differs from neighboring cells). These three areas split both the area B4:B8 and the area D4:D8 because, e.g., the area H7:H7 contains a reference to D7 but not to other cells in the area D4:D8. As a result, the visualization of this spreadsheet is more complicated as shown in Figure 13.

It is noteworthy that, in Excel 7.0 auditing, the total visualization of the new spreadsheet with the change in the cell H7 has, as perceived by a user, the same appearance as the original one in Figure 11, because the only change considers the lengths of the arrows but these changes coincide with other arrows and hence they do not show up. Thus, the change in the spreadsheet can be detected during the representation construction process but not in the final visualization.

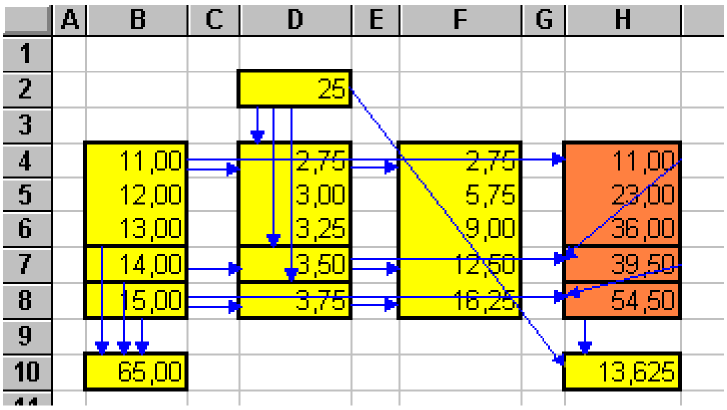

Figure 13: Splitting area in the S2 visualization.

In the S2 visualization, an area A is not split by an area B, if the area A has at one edge formulae with some references removed (as described above) and cells in the area B refer to cells at the opposite edge of the area A but not to other cells. For example, in the area H10:H10 there is a reference to the cell H8, but not to other cells in the area H4:H8, but this does not split the area H4:H8.

The splitting does not propagate, i.e., the subareas that form the split area A are treated as a single area when further splitting effects are considered.

Representation construction process. The construction of a visualization can start with any area and the mechanism automatically expands the set of cells to include all cells belonging to partially included S areas. The user can request visualization construction in either of two directions: towards precedents (i.e., towards referenced S areas) or towards successors (i.e., towards referencing S areas). In each step the visualization consists of a set of (possibly split) S areas.

When the visualization is constructed towards precedents, a S area is not colored orange until the S area it splits is included in the visualization.

When the visualization is constructed towards successors, a S area is not split until the S area that splits it is included in the visualization.

The whole visualization can be removed with a single command.

The total visualization does not depend on the direction of the representation construction process.

Other features. If a user double-clicks an arrow, the upper-left corner cell at other end of the arrow will become the currently active cell.

Implementation. The S2 visualization is partially implemented as Excel macros. Representation construction towards successors and arrow double-clicking are not implemented. Moreover, the positioning of arrows is simplified.

The S2 visualization macro is available for download by anonymous ftp at ftp://cs.joensuu.fi/pub/Software/S-visualization.

Discussion. The S2 visualization is originally defined by a procedure that finds S areas and not by defining the properties of S areas. Yet S areas fit quite well with the FFR model.

The S2 visualization is based on highlighting maximal consistent areas and references between these areas. Formats include type information, and format equivalence is equality with naming and protection information discard. Formula equivalence is case independent equality comparison with case dependence within strings embedded in formulae. The origin relation is defined by the phrase "a formula that can be obtained by removing from the copy in the neighbor cell all references to the cells at that edge, and copying that formula". This relation is 1-start-pruning.

Any breaking area, say B, is highlighted with different color than other consistent areas, and an area that it breaks, say A, is split into subareas so that B is not breaking these subareas.

The S2 visualization does not recognize corner-originating areas but they are treated as collections of consistent columns or rows. The S2 visualization pays no attention to naming or protection information.

3.4 S3 Visualization

The S2 visualization and the FFR model suggest together a more coherent visualization mechanism, the S3 visualization, that exploits the full power of maximal consistent areas, generalizes the origin relation from 1-start-pruning to start-pruning, includes protection information, and makes a visual distinction between split areas and maximal areas.

Visualization. The S3 visualization uses the same visual units, i.e., yellow and orange areas together with blue arrows to highlight references between the areas. In contrast to the S2 visualization, broken areas are now visualized with dashed lines between subareas treated as unbroken when arrows are drawn. Figure 14 gives the S3 visualization of the same spreadsheet as in Figure 13.

Each highlighted area is a maximal consistent area. The format equivalence is equality with naming information discard (but protection information retention), and the formula equivalence is case independent equality comparison with case dependence within strings embedded in formulae. The origin relation is the most general start-pruning relation, i.e., $\Omega(f_1,t_1,f_2,t_2,r)$ holds if f_1 can be obtained from f_2 by removing all cell references containing a relative row reference - r-1 together with any amount of preceding and succeeding characters (but not references).

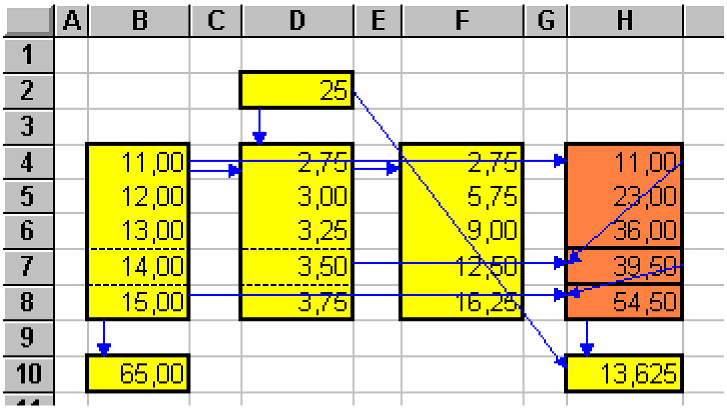

Figure 14: The S3 visualization of the spreadsheet in Figure 13.

Representation construction process. The construction of a visualization occurs as with the S2 visualization.

Other features. If a user double-clicks an arrow, the upper-left corner cell at other end of the arrow will become the currently active cell like in the S2 visualization. Moreover, clicking an arrow with another mouse button opens a text box that describes the areas at both ends of the arrow. An area description consists of the corner indexes, possible name of the area, protection of the area (all cells protected, none protected, mixed), and one or two formulae depending on the character of the area. If the area is homogeneous, the description contains the formula of the upper left corner cell together with its address, or the word “Constants ” if the area contains no cell or area references. Otherwise formulae and addresses of two cells are presented, the first being the cell from which the computation originates, and the second being the respective corner cell of the homogeneous part of the area. Figure 15 shows an example of an arrow description.

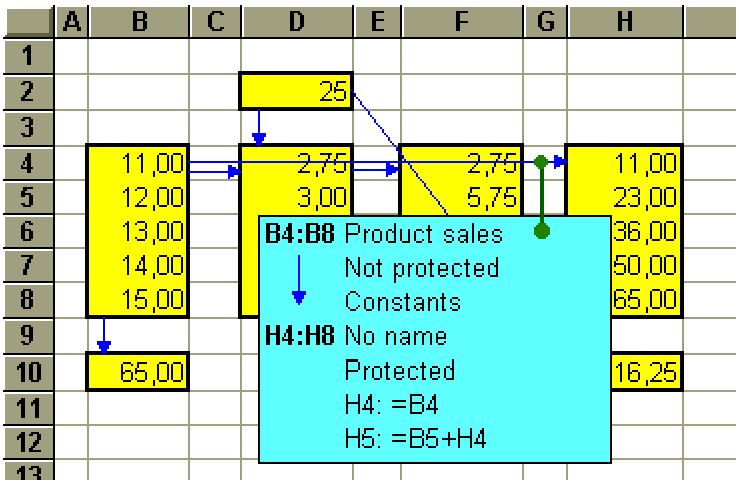

Figure 15: An arrow description in the S3 visualization.

Implementation. The S3 visualization can be implemented as Excel macros but this has not yet been done.

Discussion. The S3 visualization is defined in the terms of the FFR model above. It does not make use of any naming or protection information except in arrow descriptions.

3.5 On-line Data Dependency Diagrams

The other auditing tool suggested by Davis (1996), on-line data dependency diagrams, is based on spreadsheet flow diagrams introduced by Ronen & al. (1989) for documenting spreadsheet designs. While spreadsheet flow diagrams are supposed to be constructed by users, on-line data dependency diagrams are supposed to be generated automatically.

Visualization. The visualization is not superimposed on the spreadsheet display but is presented as separate graphics in its own window. The visualization consists of a flowchart-like graph with cells as nodes and arrows showing data dependencies. There are distinctive symbols for nodes to represent cells that function as inputs, outputs, decision variables, parameters, or formulae. Nodes contain textual information about the role of the cell and its formula. Figure 16 gives a partial example of the visualization.

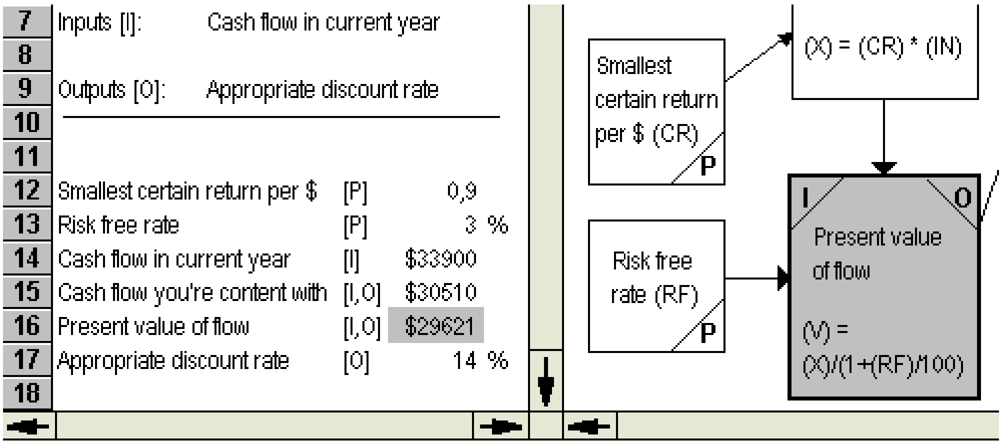

Figure 16: Excerpt of an on-line data dependency diagram (Davis, 1996).

Representation construction process. The visualization is constructed in a single step.

Other features. A user can activate the corresponding cell in the spreadsheet by giving a command when a node in the graph is selected, and vice versa. The selected cell and node are marked with the same color, and the display of the graph is centered on that node.

Implementation. The on-line data dependency diagram tool is not implemented and no implementation is planned.

In the only on-line data dependency diagram given by Davis (1996), nodes represent single cells. As a consequence, the set of nodes and arrows can be constructed automatically, and graph algorithms can be used to find a suitable layout. With large spreadsheets the graph becomes, however, huge and its usability may be questionable. Moreover, to be able to automatically construct the textual information in the cells, new theory about spreadsheets must be developed.

In spreadsheet flow diagrams, proposed by Ronen & al. (1989), nodes do not represent single cells but areas that are identified by users. As a result, the sizes of graphs remain manageable. But to be able to automatically find the areas, some theory about the properties of areas is needed. The FFR model above is the first attempt to give such a theory.

Discussion. On-line data dependency diagrams are based on the cell and area references of formula contents. The part whose construction is not automated, may use any other information but we cannot be more precise as there is no clear description of the basics of the visualization in literature.

3.6 CogMap

Hendry & Green (1993) introduce CogMap as a mechanism for spreadsheet users to describe their spreadsheets to co-workers. It is not an automatic audit tool but rather a way to document spreadsheets.

Visualization. The basic idea of CogMap is to give users the possibility to annotate spreadsheets with textual tags that are attached to colored areas and with links that connect such areas. Each tag and link may have a textual description. Users can set up tags and links, select colors, and write the contents of texts freely. Links are represented as lines, their thickness is adjustable, and they can have arrow heads. A cell may belong to many tags and, therefore, may have multiple colors. Simulated transparency is used to describe two colors on top of each other. Figure 17 contains an excerpt of a CogMap visualization.

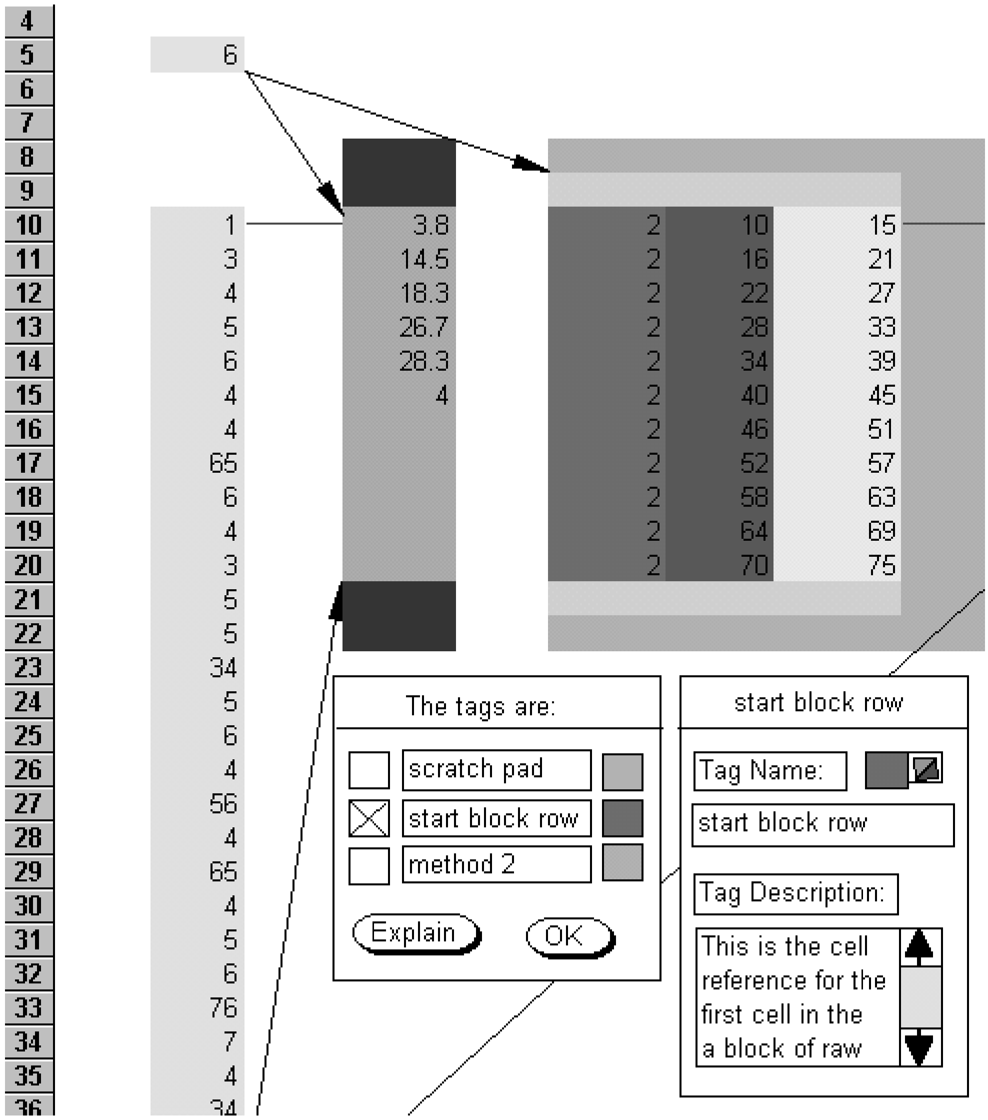

Figure 17: Excerpt of a CogMap visualization (Hendry & Green, 1993).

Representation construction process. CogMap assists users to construct annotations but it does not automatically suggest any part of the visualization.

Other features. CogMap allows users to list tags attached to an area, to display the texts, to control which tags and links are shown, and to search for particular tags and links.

Implementation. CogMap is implemented in the scripting language of the Claris Resolve spreadsheet system.

Discussion. CogMap annotations are attached to areas identified by users. All information is based on users' interpretation of the contents of spreadsheet applications. The tool does not analyze spreadsheets in any manner.

Hendry & Green (1993) have sketched an extension to CogMap consisting of short arrows starting from the upper-left cell of an area and pointing to a box with the tag of an area that is referenced in the cell. In contrast to links, the direction of arrows is now opposite, and they do not reach to the other area but a name box is used instead.

Hendry & Green (1993) have proposed also extending CogMap to apply a background pattern to cells containing absolute (column or row) references. The background pattern, either vertical, horizontal or diagonal lines, is supposed to reveal areas having copy equivalent formulae (or "the direction that the formula was filled [into the area with the Edit/Fill operation]"). Unfortunately, due to area borders not being visualized, this effect is not achieved. It is unclear whether such patterns are supposed to be applied manually by users or automatically by CogMap. If the background pattern is going to express more than just the form (one column wide, one row high, other) of a copy equivalent area, user assistance is needed.

3.7 Spreadsheet Schemes

Isakowitz & al. (1995) propose a textual representation of spreadsheets that captures the computations between logical structures in a spreadsheet. The process is not totally automatic: the user must identify and name the functional relations (in the sense of relational data bases) that make up the mathematical model behind the spreadsheet. The resulting logical schema is supposed to lend itself to identifying design anomalies, either by human auditor or by automated techniques.

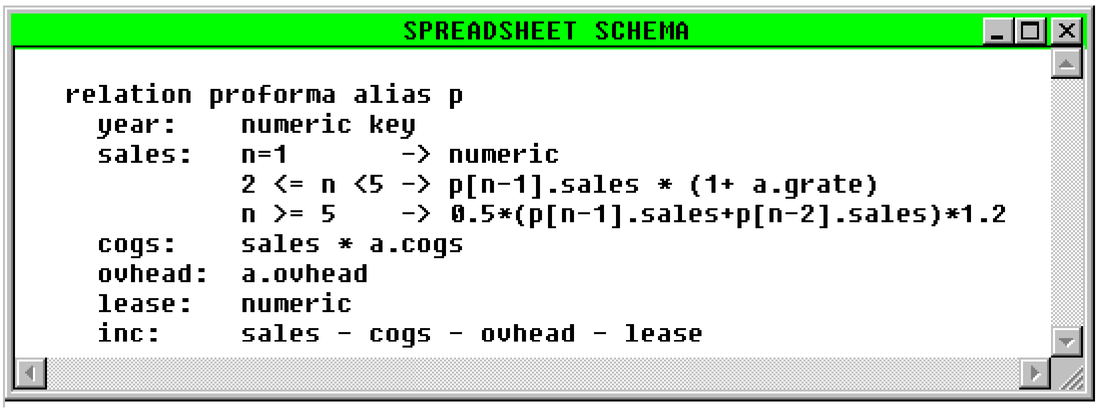

Figure 18: Excerpt of a spreadsheet schema (Isakowitz & al., 1995).

Visualization. Spreadsheet schemes are not graphical but textual representations that treat vector input areas as entities used in the calculations.

For example, Figure 18 gives an excerpt of a schema that describes a spreadsheet for sales forecast. The entities year, sales, cogs, ovhead, lease, and inc correspond to rows with those names in column A and values for six years in columns B..G. The spreadsheet schema has no reference to physical columns but uses indexes to refer to individual elements, and it applies vector operations when possible.

Representation construction process. The representation is constructed in a single step consisting of two phases. First, the user is asked to identify and name contiguous blocks of cells that represent either a singular or a repetitive entity. Then the spreadsheet schema is automatically constructed.

Other features. Spreadsheet schemes can be stored in external files. There is a reverse operation to create a spreadsheet from a schema, thus enabling schema reuse.

Implementation. Spreadsheet schemes are implemented as an Excel macro that enables users to identify and name entities, and a Pascal program that does rest of the representation construction. An implementation of the reverse operation is projected.

Discussion. Spreadsheet schemes consider areas identified by the user, formula and format contents, and types of cells containing constant formulae. Within a user identified entity, consecutive copy equivalent formulae are recognized and replaced by a single formula. Copy equivalence tests for formula equivalence but not for format equivalence.

Spreadsheet schemes do not take advantage of naming or protection information. Edge-originating calculations are not particularly recognized.

Spreadsheet schemes cannot be constructed automatically unless some theory about the properties of areas, like the FFR model, is introduced.

4. Concluding Remarks

We have introduced the FFR model to be able to theoretically analyze and compare visualization mechanisms. The model has demonstrated its practicality as we could easily describe various auditing tools with it. Moreover, the FFR model provides a systematic way to operate with different origin relations describing how calculations within areas can begin. The application of the model to describe the S2 visualization suggested immediately an improved version, the S3 visualization.

We used the FFR model to analyze 7 auditing tools. On-line data dependency diagrams, CogMap and spreadsheet schemes require user assistance in the visualization construction process making their usability limited. Of the remaining four tools, the S2 and S3 visualizations turned out to be essentially more expressive than the Excel 7.0 auditing mechanism and the arrow tool. The power of the S2 and S3 visualizations stems from the recognition of edge-originating areas by using specific start-pruning origin relations.

In the introduction we claimed that visualizations can be used to highlight anomalies that imply potential errors. The FFR model enables us to rephrase this claim: we expect that *errors in spreadsheet applications manifest themselves as inconsistent and breaking areas*. Provided that the claim holds for some origin relation Ω in most cases, a spreadsheet application audit tool can be based on visualizing such areas.

Thus, future research should concentrate on the formulation of new origin relations and their validation with empirical research. What is needed is an origin relation that rejects as many erroneous computations as possible and at the same time accepts practically all correct computations. We feel that such a relation will be start-compressing but not start-pruning.

Secondly, we need research into the psychological aspects of visualizations: how should we highlight various areas and their interconnections to best match the cognitive capabilities of users. Empirical experiments are needed to understand how details of visualization mechanisms affect users' comprehension of computational structures within spreadsheets.

Finally, when the FFR model is applied to new cases, there will certainly arise a need to improve the model itself. For example, it may turn out that naming and protection information should not be embedded in formats, or that types should be explicitly included in the model. But the further development of the FFR model must wait until we have more experiences about its use.

Acknowledgments

I would like to thank Markku Tukiainen for lengthy discussions about the nature of spreadsheets and the manifestation of users' goals and plans in spreadsheets, Pertti Saariluoma for his encouragement on visualization development, and Tero Koistinen for his programming efforts on the S2 visualization. This work was inspired by *carpets and wallpaper*.

References

Brown, P. S. & Gould, J. D. (1987) An Experimental Study of People Creating Spreadsheets. *ACM Transactions on Office Information Systems*, **5**(3), 258-272.

Davis, J. S. (1996) Tools for Spreadsheet Auditing. *International Journal of Human-Computer Studies*, **45**(4), 429-442.

Ditlea, S. (1987) Spreadsheets Can be Hazardous to Your Health. *Personal Computing*, 11(1), 60-69.

Hassinen, K., Sajaniemi, J. & Väisänen J. (1988) Structured Spreadsheet Calculation. 1988 *IEEE Workshop on Languages for Automation*, Computer Society Press, 129-133.

Hassinen, K. (1994) *Luokitteluperustainen menetelmä taulukko laskenta - sovellusten analysointiin* (A Classification-Based Method for the Analysis of Spreadsheet Applications). In Finnish. Lic.Phil. Thesis, University of Joensuu, Department of Computer Science.

Hendry, D. G. & Green, T. R. G. (1993) CogMap: A Visual Description Language for Spreadsheets. *Journal of Visual Languages and Computing*, **4**, 35-54.

Isakowitz, T., Schocken, S. & Lucas, H. C. Jr. (1995) Toward a Logical/Physical Theory of Spreadsheet Modeling. *ACM Transactions on Information Systems*, **13**(1), 1-37.

Lakshmanan, L. V. S., Subramanian, S. N., Goyal, N. & Krishnamurty, R. (1998) On Querying Spreadsheets. *14th International Conference on Data Engineering*, IEEE Computer Society, Los Alamitos, Ca, 134-141.

Ronen, B., Palley, M. A. & Lucas, H. C. Jr. (1989) Spreadsheet Analysis and Design. *Communications of the ACM*, **32**(1), 84-93.

Saariluoma, P. & Sajaniemi, J. (1991) Extracting Implicit Tree Structures in Spreadsheet Calculation. *Ergonomics*, **34**(8), 1027-1046.

Saariluoma, P. & Sajaniemi, J. (1994) Transforming Verbal Descriptions into Mathematical Formulas in Spreadsheet Calculation. *International Journal of Human-Computer Studies*, **41** (6), 915-948.

Sajaniemi, J. & Pekkanen, J. (1988) An Empirical Analysis of Spreadsheet Calculation. *Software - Practice and Experience*, **18** (6), 583-596.

Tukiainen, M. & Sajaniemi, J. (1996) *Spreadsheet Goal and Plan Catalog: Additive and Multiplicative Computational Goals and Plans in Spreadsheet Calculation*. University of Joensuu, Department of Computer Science, Technical Report, Series A, Report A-1996-4.