UNIVERSITY OF JOENSUU
DEPARTMENT OF COMPUTER SCIENCE
Report Series A

Spreadsheet Goal and Plan Catalog: Additive and Multiplicative Computational Goals and Plans in Spreadsheet Calculation

Markku Tukiainen, Jorma Sajaniemi

Report A-1996-4

ACM H.1.2, H.5.2, D.m
UDK 681.3.02, 519.68
ISSN 0789-7316
ISBN 951-708-461-7

Spreadsheet Goal and Plan Catalog:
Additive and Multiplicative Computational Goals and Plans in Spreadsheet Calculation

Markku Tukiainen and Jorma Sajaniemi
mtuki@cs.joensuu.fi, saja@cs.joensuu.fi

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

Abstract

Spreadsheeting knowledge can be characterized in the form of goals and plans that describe what must be achieved and how this is done. We have analyzed a set of real-life spreadsheets and extracted a collection of computational goals and plans. This report contains a descriptive catalog of goals and plans consisting of addition, subtraction, multiplication, and division. The catalog is based on actual spreadsheet usage and is not intended to be a prescriptive catalog of what spreadsheeting goals and plans should be like.

Keywords: Spreadsheet calculation, Empirical studies of programming

CR categories: H.1.2, H.5.2, D.m

Contents

1. INTRODUCTION	2
2. THE SCOPE OF THIS REPORT	3
3. NOTATIONS	4
3.1 Goal Notation	4
3.2 Plan Notation	5
4. GOAL CATALOG	6
4.1 General Goals	6
4.2 Formula Goals	11
4.3 Application Specific Goals	12
4.4 Merged Goals	14
5. PLAN CATALOG	16
5.1 General Plans	16
5.2 Formula Plans	23
5.3 Application Specific Plans	23
5.4 Merged Plans	26
References	28

1. INTRODUCTION

In the domain of programming expert tacit programming knowledge has been argued to appear in the form of plans which represent many of the stereotypic action sequences in programs (Ehrlich & Soloway, 1984). This notion of programming plans comes from the text comprehension research notion of schemas (Soloway & Ehrlich, 1984). The plan in this thinking is a procedure or strategy in which the key elements of the process have been abstracted and represented explicitly. This approach is often called schema-based, emphasizing the semantic structures of programming and contrasted to control-flow approach emphasizing the syntactic structures (Detienne, 1990; Rist, 1989).

Programming involves knowledge of different domains, at least those of the programming domain and the problem domain (Brooks, 1983; Bonar, 1985). The problem domain description of a programming problem represents the desired goal, or decomposition of this goal into several subgoals, and forms the highest level description of the problem. This composition can be expressed as a goal and plan tree (GAP tree) (Spohrer & al., 1985) where every second level describes (sub)goals to be fulfilled and every other level describes alternative plans for each of the goals. A plan implements the goal in terms of programming constructs and further subgoals. Each program that solves the programming problem obeys a pruned tree obtained from the GAP tree by selecting one special plan for each goal.

Programming knowledge can be divided into knowledge of general programming concepts and knowledge of a certain programming language or tool. Soloway & al., (1982) make a distinction between strategic plans which represent language independent programming knowledge, tactical plans which are kind of semi language independent programming knowledge, and implementation plans which are programming language dependent realizations of higher level plans. Soloway & al. have studied plans more accurately in the lower level of this hierarchy, i.e., plans in Pascal programming. They identified a set of plans (serving as expert programmers) they thought was used in simple Pascal looping programs. The plans were divided into two main kinds: control flow plans representing looping structures and variable plans representing variable usage (Ehrlich & Soloway, 1984).

Although the spreadsheet model has no provision for structuring data or computations, spreadsheets created within this model do contain groups of data items and formulae that have some structure. These structures are caused by programmers (or, more conventionally, users) who have created the spreadsheets. With this in mind we set out to explore actual spreadsheets to see if there are reoccurring uses of computations. We did this by writing a program to extract connected formula graphs from spreadsheets and by analyzing 101 spreadsheets used by Finnish business companies and governmental agencies. (See Sajaniemi & Pekkanen (1988) for a more detailed description of this data.)

We went through all these formula graphs, looked for reoccurring computational structures, and tried to figure out the intentions of the programmer. With this method we soon found out a collection of computational goals (e.g., cumulative sum of a set of numbers) and plans to achieve these goals (e.g., a certain set of formulae in consecutive cells). This report contains a descriptive catalog of the goals and plans we found in the spreadsheets. These goals and plans were found in real-life spreadsheets and thus describe methods utilized by people in their normal work. The catalog is based on actual spreadsheet usage and is by no means intended to be a prescriptive catalog of what spreadsheeting goals and plans should be like.

The goals and plans in this catalog describe spreadsheeting knowledge, i.e., knowledge that can be used in potentially any spreadsheet, and concerns the low-level end of goal and plan hierarchy. A certain spreadsheet, on the other hand, has its own goal and plan structure. This structure uses the given goals and plans adapted to the requirements of the specific spreadsheet.

The goals and plans are grouped in the catalog with respect to their generality but this grouping is by no means an absolute one. Goals and plans can be related to each other in many ways which are not reflected in this grouping. For example, two plans may differ only by the fact that one uses a constant value and the other uses a cell to find the value. These plans could be considered to be some kind of transformations of some general plan template but they are still given in the catalog as two independent plans. The grouping of the plans is not based on any formal notion of plan relationships but it is made to increase the readability of this report only.

2. THE SCOPE OF THIS REPORT

This report describes additive and multiplicative computational goals and plans found in a set of real-life spreadsheet applications (Sajaniemi & Pekkanen, 1988). Additive computational goals and plans are strategies or procedures that contain addition and/or subtraction, and multiplicative computational goals and plans are strategies or procedures that contain multiplication and/or division. We will not list goals and plans involving other operators and functions. For example the IF function is often used in spreadsheets and one of its major uses is to guard for division by zero.

The goals are arranged to four groups: application specific goals, general goals, formula goals and merged goals. Application specific goals are related to specific application, ie, the computations are not applicable accross different applications. The main goals of each application are not articulated here, because we were loooking for more general goals. General goals are goals which can be found (and applied) in many applications. Formula goals are implicit goals, they are used only at the level of formulas. Merged goals are goals that try to achieve two or more goals at the same time.

Spreadsheets contain other than computational goals and plans, also. We have found, for example, depictive goals and plans which represent data structures and other space reservations like titles and data labeling. Such goals and plans are not covered in this catalog. Further, we will not consider other aspects of spreadsheet systems like macros or graphics.

3. NOTATIONS

We have developed a special notation to characterize goals and plans both at the level of general spreadheeting knowledge and at the level of the spreadsheet language, i.e., the formula templates written at the abstracted spreadsheet programming language. In this spreadsheet language, the actual cell references have been abstracted to refer to data structures of the types chamber, sequence or table.

When computation refers to a single cell we will use term chamber and write the reference as Cham1, Cham2, etc. When the computation refers to larger data structures (areas) then the terms sequence and table are used. A sequence is one-dimensional uniform collection of cells, i.e., it lies either in a row or in a column of a spreadsheet. Sequences are named as Seq1, Seq2, etc. A table is a two-dimensional uniform collection of cells, i.e., it has at least two rows of cells and two columns of cells. Tables are noted as Tab1, Tab2, etc. In some computations, structure types are overloaded, i.e., a chamber can act as a sequence or a sequence can play the role of a table. Computations which can be applied to all structure types use the names Input and Output for their arguments.

3.1 Goal Notation

Goal descriptions have the following slots:

G: Goal name, the name reflects the intension of the goal

D: Description: Output = Goal_name(Input), an overview of the intentional aspects of the goal. Output tells what type of data structure can result from realization of this goal. Input tells what type of data structure can act as initial structure for this goal (if any needed).

P: Plan_names, the names of the plans that fulfil the intensions of this goal

E: Examples, of the actual spreadsheet applications from which the goal was extracted

3.2 Plan Notation

Plan descriptions have the following slots:

P: Plan_name, the name reflects the computation performed by this plan

D: Description: Output = Plan_name(Input), a description of the computational aspects of the plan. Output tells what type of data structure can result from application of this plan. Input tells what type of data structure can act as input structure for this plan (if any needed).

T: Template, the computation written at the abstracted spreadheet programming language. Template shows the formula which lies in the cell(s) of the output. The formula begins with plus (+) or minus (-) operator. The formula is applied to all cells of the output structure once, ie, over all indexes. If the computation needs to refer specific cells of the input then it is noted like Input.c_i where i is the index of Input cell. If an index n is used, like Input.c_n it means that n runs from the first index of input to the last index of input. If for some reason we have to tell all the formulae of the output then the ouput cells are noted like Output. c_n: +Input. c_n. Language contains build-in functions @SUM and @AVE for sums and averages.

C: Constrains, spatial or usually size constrains, eg. argument structures must be of equal size

S: SpatialVariations, some of the plans can be laid out horizontally or vertically, eg. IndexSums can be rowwise or columnwise

O: Observations, other remarks

R: Required subgoals, if computation uses other goals

E: Examples of the actual spreadsheet applications from which the plan was extracted

4. GOAL CATALOG

4.1 General Goals

Goal: TotalSum

D: Cham1 = TotalSum (Input)

TotalSum sums up all the cells in input structure to the output chamber Cham1.

P: _default

_splitting

_cumulative

E: 048, 049, 051, 052, 059, 060, 062

Goal: IndexSum

D: Seq1 = IndexSum (Tab1)

IndexSum sums up all the cells over the indexes of one dimension of the input table to the output sequence.

P: _default

P: _splitting

E: 002, 046, 050, 062

Goal: CumulativeSum

D: Seq2 = CumulativeSum (Seq1)

CumulativeSum sums over all the cells of the input sequence Seq1 to the output sequence Seq2.

P: _default

E: 047, 091

Goal: DifferencePairwise

D: Seq3 = DifferencePairwise (Seq1, Seq2)

DifferencePairwise subtracts cells of Seq2 from cells of Seq1 pairwise

P: _default

E: 049, 090, 065, 066

Goal: Duplication

D: Output = Duplication (Input)

Duplication duplicates values of the input structure to the output structure. Input and Output are of equal size.

P: _default

E: 024, 030, 033, 046, 047

Goal: IncompleteDuplication

D: Seq2 = IncompleteDuplication (Seq1)

IncompleteDuplication duplicates some but not all values of the input sequence to the output sequence.

P: _default

E: 090

Goal: Negation

D: Output = Negation (Input)

Negation changes the values of the input structure to their negations to the output structure. Input and Output are of equal size.

P: _default

E: 024, 030, 033

Goal: SubstractionByConstant
D: Output = SubstractionByConstant (Input)
SubstractionByConstant subtracts a constant value from the original cell.
P: _default
E: 090

Goal: SubstractionFromConstant
D: Output = SubstractionFromConstant (Input)
SubstractionFromConstant subtracts the referenced cell value from a constant.
P: _default
E: 090

Goal: MultiplyByConstant
D: Output = MultiplyByConstant (Input)
MultiplyByConstant transforms a value by multiplying the original cell value
by constant.
P: _default
E: 016

Goal: UnitProportion
D: Output = UnitProportion (Input1,Input2)
UnitProportion gives a proportion of a measure by another measure.
Typical examples are marks/hour, kg/h or income/capita etc.
P: _default
E: 059, 060, 072

Goal: Percentage
D: Output = Percentage (Input)
Percentage gives a percentage of a cell value from total of input structure.
P: _default
E: 065, 072

Goal: PercentagePairwise
D: Seq3 = PercentagePairwise (Seq1, Seq2)
PercentagePairwise gives a percentage proportion
between pairs of cells of input sequences Seq1 and Seq2.
P: _default
E: 048

Goal: PercentageProportion
D: Seq2 = PercentageProportion (Seq1, Cham1)
PercentageProportion gives a percentage proportion between pairs of cells of input sequence Seq1 and chamber Cham1.
P: _default
_usingConstantReference
E: 072

Goal: IncreaseByPercentage
D: Seq2 = IncreaseByPercentage (Seq1, Cham1)
IncreaseByPercentage increases the values of cells of input sequence Seq1 with a percentage proportion constant in chamber Cham1.
P: _default
E: 046

Goal: StepwiseIncrement

D: Seq2 = StepwiseIncrement (Cham1)

StepwiseIncrement increases the values of cells of sequence Seq2 with a constant value in chamber Cham1. This looks like CumulativeSum.

P: _default

E: 047, 076

Goal: CumulativeDifference

D: Seq3 = CumulativeDifference (Seq1, Seq2)

CumulativeDifference sums cumulatively over all the cells of the input sequences Seq1 and Seq2 and then takes a pairwise difference of these cumulated sequences to the output sequence Seq3

P: _default

E: 047

Goal: Duplicates

D: Output = Duplicates (Cham1)

Duplicates duplicates a value in the input chamber to the output structure

P: _default

E: 092, 075

Goal: NumberSeries

D: Output = NumberSeries (Input)

NumberSeries cumulates a constant value of 1 to the original cells

P: _default

E: 092

Goal: CorrectionByIndexClause

D: Output = CorrectionByIndexClause (Input, Cham1, Cham2)

CorrectionByIndexClause changes the values in input structure to the corrected values of output structure (eg. changes currency values according one base year) by dividing with base value Cham1 and multiplying by current year value Cham2 of the original cell

P: _default

E: 033, 030, 049

Goal: Average

D: Cham1 = Average (Input)

Average calculates an average value of the original cells

P: _default
_byHand

E: 059, 054

Goal: AdditionByConstant

D: Seq2 = AdditionByConstant (Seq1, Constant)

AdditionByConstant adds a constant value to the cells of input sequence. This is not Calculator because it is used for forecasting purposes in planned budget application.

P: _default

E: 076

Goal: FormatToThousands

D: Seq2 = FormatToThousands (Seq1)

FormatToThousands divides the cells of input sequence by 1000.

P: _default

E: 048, 049

Goal: ComputedTitles
D: Seq2 = ComputedTitles (Seq1)
ComputedTitles computes titles out of other numeric year titles.
P: _default
E: 048

Goal: PartialTotalSum
D: Cham1 = PartialTotalSum (Input)
PartialTotalSum sums up all the cells in input structure to the output chamber Cham1 and then substracts one of the cells out of this total sum.
This is usually done because one of the values is so dominating that further analysis are affected most by this value.
P: _default
E: 065

Goal: SumPairwise
D: Seq3 = SumPairwise (Seq1, Seq2)
SumPairwise sums cells of Seq2 from cells of Seq1 pairwise. Usually Seq1 and Seq2 are of equal size.
P: _default
E: 065, 076, 057

Goal: DifferenceBetweenPairs
D: Seq2 = DifferenceBetweenPairs (Seq1)
DifferenceBetweenPairs takes a difference between two consecutive cells in Seq1 so that first cell in Seq2 yields a difference between Seq1.c_1 and Seq1. c_2 and so on
P: _default
E: 072

Goal: MultiplyPairwise
D: Output = MultiplyPairwise (Input1, Input2)
MultiplyPairwise multiplies values of cells of Input1 by values of cells of Input2 pairwise
P: _default
E: 108, 075

Goal: DividePairwise
D: Output = DividePairwise (Input1, Input2)
DividePairwise divides values of cells of Input1 by values of cells of Input2 pairwise. This is probably always UnitProportion.
P: _default
E: 073, 075

Goal: RepairFormatToThousands
D: Output = RepairFormatToThousands (Input)
RepairFormatToThousands repairs FormatToThousand's division by multiplying values by 1000.
P: _default
E: 075

Goal: ProportionByPercentage

D: Seq2 = ProportionByPercentage (Seq1, Cham1)

ProportionByPercentage gives a percentage proportion Cham1 from input sequence Seq1.

P: _default

_usingConstantReference

E: 046, 059, 062

Goal: PercentageGrowth

D: Seq3 = PercentageGrowth (Seq1, Seq2)

PercentageGrowth gives a percentage growth rate from input sequence Seq1 to input sequence Seq2.

P: _default

_usingPercentageFormat

E: 057, 060

Goal: DivisionByConstant

D: Seq2 = DivisionByConstant (Seq1, Constant1)

DivisionByConstant divides values in input sequence Seq1 by Constant1.

P: _default

E: 059

Goal: DecreaseByPercentage

D: Seq3 = DecreaseByPercentage (Seq1, Seq2)

DecreaseByPercentage decreases values in input sequence Seq1 by percentages in Seq2.

P: _default

E: 062

Goal: PercentageDecrease

D: Seq3 = PercentageDecrease (Seq1, Seq2)

PercentageDecrease gives a percentage decrease rate from input sequence Seq1 to input sequence Seq2.

P: _default

E: 062

Goal: ProportionPartition

D: Seq2 = ProportionPartition (Cham1, Cham2, Seq1)

ProportionPartition gives a proportion parts of Cham1 by proportions in input sequence Seq1, eg. Cham1 is to be divede to parts which are proportions in Seq1.

P: _default

E: 061

Goal: PartlySum

D: Seq1= PartlySum (Constant1, Seq2)

PartlySum sums a size Constant1 portions of sequence Seq2.

P: _default

E: 106

Goal: FormatTo100Kilometers
D: Seq2 = FormatTo100Kilometers (Seq1)
FormatTo100Kilometers divides the cells of input sequence by 100.
O: Used to format vehicle's usage kilometers to hundreds.
P: _default
E: 063

Goal: FormatToHours
D: Seq2 = FormatToHours (Seq1)
FormatToHours divides the cells of input sequence by 60.
O: Used to format minutes to hours.
P: _default
E: 063

4.2 Formula Goals

Goal: SumClause
D: Cham1 = SumClause (Input)
SumClause sums up all the values in input structure to the output chamber with a single addition
P: _addition
_sumFunction
E: 049

Goal: Calculator
D: Output = Calculator (Input)
Calculator calculates value of a cell using constant operands and basic operators +,-,* and /.
P: _default
E: 001, 090, 075

4.3 Application Specific Goals

Goal: MeansEndsAnalysis

D: Seq3 = MeansEndsAnalysis (Seq1, Seq2, Cham1)

MeansEndsAnalysis calculates a productivity value using means/ends values of the input sequence Seq1 and corrects it by index values Seq2 and Cham1.

P: _default

E: 016

Goal: TurnTime

D: Cham4 = TurnTime (Cham1, Cham2, Cham3)

TurnTime computes a turn time of store.

P: _default

E: 064

Goal: VacationPay

D: Seq3 = VacationPay (Seq1, Seq2, 50)

VacationPay calculates a vacation time extra payment by dividing the monthly payment in Seq1 with 50 and multiplying this amount by the number of vacation days in Seq2.

P: _default

E: 046

Goal: XeroxCopyPrice

D: Cham6 = XeroxCopyPrice (Cham1, Cham2, Cham3, Cham4, Cham5)

XeroxCopyPrice calculates an approximate xerox copy price for square metre taking account of employee's salary and price of materials.

P: _default

E: 054

Goal: AverageSquareMetreAge

D: Cham1 = AverageSquareMetreAge (Seq1, Seq2)

AverageSquareMetreAge calculates an average age for a number of buildings weighted by the area of buildings. Seq1 holds the areas and Seq2 holds the ages.

P: _default

E: 075

Goal: TemperatureIndex

D: Seq4 = TemperatureIndex (Seq1, Seq2, Seq3, Constant1, Constant2)

TemperatureIndex calculates a monthly normalized temperature index. Seq1 holds the normalized degree-day for current month, Seq2 holds a sum of energy consumption for last twelve months, Seq3 holds a sum of degree-days for last twelve months.

P: _default

E: 106

Goal: EnergyConsumption

D: Seq3 = EnergyConsumption (Seq1, Seq2, Constant1, Constant2, Constant3)

EnergyConsumption calculates a monthly normalized energy consumption. Seq1 holds a sum of energy consumption for last twelve months, Seq2 holds a sum of degree-days for last twelve months.

P: _default

E: 106

Goal: ObtainableSales

D: Cham3 = ObtainableSales (Cham1, Cham2, Constant1, Constant2)

ObtainableSales calculates obtainable sales for a year. Cham1 holds a premis for sales for year, Cham2 holds a turnover for year.

P: _default

E: 069

Goal: DepreciatedValues

D: Seq4 = DepreciatedValues (Seq1, Seq2, Seq3)

DepreciatedValues calculates depreciated values for different things. Seq1 and Seq2 hold values of things, Seq3 holds depreciation value factors.

P: _default

E: 071

Goal: CirculationPeriodFactors

D: Seq2 = CirculationPeriodFactors (Seq1, Cham1, Cham2, Constant1, Constant2)

TurnoverTimeFactors calculates circulation period factors. Seq1 holds the values to be circulated. Cham1 holds a turnover for year, Cham2 holds the number of months in an accounting period.

P: _default

E: 071

Goal: ComputedAreaValue

D: Seq2 = ComputedAreaValue (Seq1, Cham1, Cham2)

ComputedAreaValue calculates a value of build area for company's assets. Seq1 holds the areas to be valuated. Cham1 holds a total sum of the areas, Cham2 holds the weighted value of the area according to geographic location.

P: _default

E: 087

Goal: YearlyCustomerSales

D: Seq5 = YearlyCustomerSales (Seq1, Seq2, Seq3, Seq4, Constant1)

YearlyCustomerSales calculates a yearly customer sales of animal food. Sales are calculated taking into account of percentage of customer sales (Seq1), yearly sales in kilograms (Seq2), price for 100 kilograms (Seq3) and profit percentage (seq4).

P: _default

E: 062

4.4 Merged Goals

MGoal: PercentagePairwise_&_SumClause
D: Seq3 = PercentagePairwise_&_SumClause (Seq1, Tab1)
PercentagePairwise_&_SumClause gives a percentage proportion between pairs of cells of input sequences Seq1 and Seq2. Seq2 is a IndexSum of Tab1 which can be computed only in formula of goal.
P: _default
R: Seq2 is IndexSum of Tab1. Seq1 and Seq2 of equal sizes.
E: 019

MGoal: UnitProportion_&_MultiplyByConstant
D: Output = UnitProportion_&_MultiplyByConstant (Input1, Input2, Constant1)
UnitProportion_&_MultiplyByConstant gives a proportion of a measure by another measure repaired with constant.
P: _default
E: 016

MGoal: FormatToThousands_&_CorrectionByIndexClause
D: Seq2 = FormatToThousands_&_CorrectionByIndexClause (Seq1)
FormatToThousands_&_CorrectionByIndexClause gives index clause corrected sequenced measured in thousands.
P: _default
E: 049

MGoal: DifferencePairwise_&_IndexSum
D: Seq2 = DifferencePairwise (Seq1,_IndexSum(Tab1))
DifferencePairwise_&_IndexSum gives a difference pairwise of input sequence Seq1 and formula goal IndexSum output sequence.
P: _default
E: 073

MGoal: Duplicates_&_CorrectionByIndexClause
D: Seq1 = Duplicates_&_CorrectionByIndexClause (Cham1, Constant1.. Constantn)
Duplicates_&_CorrectionByIndexClause gives a serie of duplicates of Cham1 and corrects some of them with constants Constant1.. Constantn.
P: _default
E: 075

MGoal: UnitProportion_&_ RepairFormatToThousands
D: Cham3= UnitProportion_&_ FormatToThousands (Cham1, Cham2, 1000)
UnitProportion_&_ FormatToThousands gives a proportion of Cham1 and Cham2 and multiplies this value by 1000 in order to correct previous FormatToThousands usage.
P: _default
E: 075

MGoal: ProportionByPercentage_&_ FormatToThousands

D: Seq2= ProportionByPercentage_&_ FormatToThousands (Seq1, Cham1, 100000)

ProportionByPercentage_&_ FormatToThousands gives a percentage proportion in Seq1 of Cham1 and divides this value by 1000 in order to change value to thousands.

P: _default

E: 059

MGoal: Input_&_CorrectionByIndexClause

D: Seq2= Input_&_CorrectionByIndexClause (Cham1, Constant1, Constant2)

Input_&_CorrectionByIndexClause transforms the original input in Constant2 by index correction Cham1 and Constant1. This is novice merged goal. Probable cause could be that novice does not want to see the original inputs and does not know how to hide cells so he puts the inputs to the formulas.

P: _default

E: 048, 049

MGoal: TotalSum_&_FormatToThousands

D: Seq2= TotalSum_&_FormatToThousands (Seq1, 1000)

TotalSum_&_FormatToThousands calculates a total sum of Seq1 and formats it to thousands.

P: _default

E: 033

MGoal: MultiplyPairwise_&_ DifferencePairwise

D: Seq4= MultiplyPairwise_&_ DifferencePairwise (Seq1, Seq2, Seq3)

MultiplyPairwise_&_ DifferencePairwise calculates a pairwise difference of Seq1 and Seq2 and multiplies result pairwise with Seq3.

P: _default

E: 108

MGoal: MultiplyPairwise _&_FormatToThousands

D: Seq3= MultiplyPairwise_&_FormatToThousands (Seq1, Seq2, 1000)

MultiplyPairwise_&_FormatToThousands multiplies Seq1 and Seq2 pairwise and formats Result to thousands.

P: _default

E: 115

5. PLAN CATALOG

5.1 General Plans

Plan: TotalSum_default
D: Cham1 = TotalSum_default (Input)
TotalSum_default sums up all the values of cells in the input structure directly
T: SumClause(Input)
E: 049, 051, 052, 059, 060

Plan: TotalSum_cumulative
D: Cham1 = TotalSum_cumulative (Input)
TotalSum_cumulative cumulates sum over all the values of cells in the input structure
R: Seq1 = CumulativeSum(Input)
T: +Seq1.c_{last}
E: 091

Plan: TotalSum_splitting
D: Cham1 = TotalSum_splitting (Part_1_of_Input)
:
Chamn = TotalSum_splitting(Part_n_of_Input)
TotalSum_splitting sums up all the cells in all the parts of the input structure individually to the output cells
T: SumClause(Input)
E: 059

Plan: IndexSum_default
D: Seq1 = IndexSum_default (Tab1)
IndexSum_default sums up all the cells over one of the indexes of the input table to the output sequence
T: SumClause(Tab1.index1)
S: Rowwise
S: Columnwise
E: 002, 047, 050

Plan: IndexSum_splitting
D: Seq1 = IndexSum_splitting (Tab1)
IndexSum_splitting sums up all the cells in all the parts of the input table over one of the indexes of the input table to the output sequence
T: SumClause(Tab1.index1)
S: Rowwise
S: Columnwise
E: 002, 046

Plan: CumulativeSum_default
D: Seq2 = CumulativeSum_default (Seq1)
CumulativeSum sums over all the cells of the input sequence Seq1 to the output sequence Seq2
T: + Seq1.c_n+Seq2.c_{n-1}
S: Rowwise: 047
S: Columnwise: 091
E: 047, 091

Plan: Duplication_default

D: Cham2 = Duplication_default (Cham1)

Duplication_default duplicates value of the input cell to the output cell

T: +Cham1

E: 024, 030, 033

Plan: IncompleteDuplication_default

D: Seq2 = IncompleteDuplication_default (Seq1)

IncompleteDuplication_default duplicates some but not all values of the input sequence to the output sequence

T: $Seq2.c_1$: $+Seq1.c_1$

:

$Seq2.c_{n-1}$: $+Seq1.c_{n-1}$

O: Usually all but the last value are duplicated.

E: 090

Plan: SubstractionByConstant_default

D: Cham2 = SubstractionByConstant_default (Cham1, Constant)

SubstractionByConstant_default repairs a value by subtracting the duplicated original by constant

R: Duplication

T: + Cham1-Constant1

O: in 090 cell: B32 template was -(Calculator)+cell reference

E: 090

Plan: SubstractionFromConstant_default

D: Cham2 = SubstractionFromConstant_default (Cham1, Constant)

SubstractionFromConstant_default repairs a value by subtracting a duplicate from the original constant

R: Duplication(Cham1)

T: +Constant1- Duplication(Cham1)

E: 090

Plan: MultiplyByConstant_default

D: Cham2 = MultiplyByConstant_default (Cham1, Constant)

MultiplyByConstant_default repairs a value by multiplying it with constant

T: + Cham1*Constant1

E: 016

Plan: UnitProportion_default

D: Output = UnitProportion_default (Input1,Input2)

UnitProportion_default gives a proportion of a measure by another measure for instance money divided by time marks/hour.

T: $+Input1.c_x/Input2.c_x$

E: 059, 060

Plan: Percentage_default

D: Output = Percentage_default (Input)

Percentage_default gives a percentage of a cell value from total of input structure.

T: $+ Input.c_x/SUM(Input)*100$

E: 065, 072, 062

Plan: DifferencePairwise_default

D: Seq3 = DifferencePairwise_default (Seq1, Seq2)

DifferencePairwise_default subtracts cells of Seq2 from cells of Seq1 pairwise

T: $+Seq1.c_n-Seq2.c_n$

E: 049, 090, 065

Plan: Negation_default

D: Output = Negation_default (Input)

Negation_default changes the values of the input structure to their negations to the output structure

T: $-Input.c_n$

C: Input and Output of equal sizes

E: xxx

Plan: PercentagePairwise_default

D: Seq3 = PercentagePairwise_default (Seq1, Seq2)

PercentagePairwise_default gives a percentage proportion between pairs of cells of input sequences Seq1 and Seq2.

T: $+Seq1.c_n/Seq2.c_n*100$

E: 048

Plan: PercentagePairwise_usingPercentageFormat

D: Seq3 = PercentagePairwise_usingPercentageFormat (Seq1, Seq2)

PercentagePairwise_ usingPercentageFormat gives a percentage proportion between pairs of cells of input sequences Seq1 and Seq2.

T: $+Seq1.c_n/Seq2.c_n$

O: Output values must be formatted with Percentage-format

E: 105

Plan: PercentageProportion_default

D: Seq2 = PercentageProportion _default (Seq1, Cham1)

PercentageProportion _default gives a percentage proportion between pairs of cells of input sequence Seq1 and chamber Cham1.

T: $+Seq1.c_n/Cham1*100$

E: 072

Plan: PercentageProportion_usingConstantReference

D: Seq2 = PercentageProportion _default (Seq1, Cham1)

PercentageProportion _default gives a percentage proportion between pairs of cells of input sequence Seq1 and chamber Cham1.

T: $+Seq1.c_n/Cham1*Constant1$

R: Constant1 = Duplication(100)

E: 070

Plan: IncreaseByPercentage_default

D: Seq2 = IncreaseByPercentage _default (Seq1, Cham1)

IncreaseByPercentage _default increases the values of cells of input sequence Seq1 with a percentage proportion constant in chamber Cham1.

T: $+Seq1.c_n*(1+Cham1/100)$

E: 046

Plan: StepwiseIncrement_default
D: Seq2 = StepwiseIncrement_default (Cham1)
StepwiseIncrement_default increases the values of cells of sequence Seq2 with a constant value in chamber Cham1. This looks like CumulativeSum.
T: $+Cham1+Seq2.c_{n-1}$
E: 047, 076

Plan: CumulativeDifference_default
D: Seq3 = CumulativeDifference_default (Seq1, Seq2)
CumulativeDifference_default sums cumulatively over all the cells of the input sequences Seq1 and Seq2 and then takes a pairwise difference of these cumulated sequences to the output sequence Seq3
T: + DifferencePairwise(Seq1, Seq2)
R: CumulativeSum(Seq1), CumulativeSum(Seq2)
E: 047

Plan: Duplicates_default
D: Output = Duplicates_default (Cham1)
Duplicates_default duplicates a value in the input chamber to the output structure
P:
T: $+Output.c_1$: +Cham1
$+Output.c_n$: $+Output.c_{n-1}$
E: 092

Plan: NumberSeries_default
D: Output = NumberSeries_default (Input)
NumberSeries_default cumulates a constant value of 1 to the original cells
T: $+Output.c_n$: $+Output.c_{n-1}+1$
E: 092

Plan: CorrectionByIndexClause_default
D: Output = CorrectionByIndexClause_default (Input, Cham1, Cham2)
CorrectionByIndexClause_default changes the values in input structure to the corrected values of output structure (eg. changes currency values according one base year) by dividing with base value Cham1 and multiplying by current year value Cham2 of the original cell
T: $+Input.c_n$*Cham1/Cham2
E: 033, 030, 049, 052÷

Plan: Average_default
D: Cham1 = Average_default (Input)
Average_default calculates an average value of the original cells using build in arithmetic average function
T: @AVE(Input)
E: 059

Plan: Average_ byHand
D: Cham1 = Average_ byHand (Input)
Average_ byHand calculates an average value of the original cells using TotalSum and division
T: +TotalSum(Input)/count of cells in Input
E: 054

Plan: AdditionByConstant_default

D: Seq2 = AdditionByConstant_default (Seq1, Constant)
AdditionByConstant_default adds a constant value to the cells of input sequence. This is not Calculator because it is used for forecasting purposes in planned budget application.
T: $+Seq1.c_n+Constant$
E: 076

Plan: FormatToThousands_default

D: Seq2 = FormatToThousands_default (Seq1)
FormatToThousands_default divides the cells of input sequence by 1000.
T: $+Seq1.c_n/1000$
E: 048, 049

Plan: ComputedTitles_default

D: Seq2 = ComputedTitles_default (Seq1)
ComputedTitles_default computes titles out of other numeric year titles. This plan strips the centuries out of year by substracting 1900 from original.
T: $+Seq1.c_n-1900$
E: 048

Plan: PartialTotalSum_default

D: Cham1 = PartialTotalSum_default (Input)
PartialTotalSum_default sums up all the cells in input structure to the output chamber Cham1 and then substracts one of the cells out of this total sum.
This is usually done because one of the values is so dominating that further analysis are affected most of this value.
T: $+TotalSum (Input)-Input.c_x$
E: 065

Plan: SumPairwise_default

D: Seq3 = SumPairwise_default (Seq1, Seq2)
SumPairwise_default sums cells of Seq2 from cells of Seq1 pairwise. Seq1 and Seq2 of equal sizes.
T: $+Seq1.c_n+Seq2.c_n$
E: 065

Plan: DifferenceBetweenPairs_default

D: Seq2 = DifferenceBetweenPairs_default (Seq1)
DifferenceBetweenPairs_default takes a difference between two consecutive cells in Seq1 so that first cell in Seq2 yields a difference between $Seq1.c_1$ and $Seq1.c_2$ and so on
T: $Seq2.c_1$: $+Seq1.c_2-Seq1.c_1$
$Seq2.c_m$: $+Seq1.c_n-Seq1.c_{n-1}$
O: m = n-1 ie. Seq2 is one cell shorter than Seq1
E: 072

Plan: MultiplyPairwise_default

D: Output = MultiplyPairwise_default (Input1, Input2)
MultiplyPairwise_default multiplies values of cells of Input1 by values of cells of Input2 pairwise
T: $+ Input1.c_n* Input2.c_n$
E: 108

Plan: DividePairwise_default

D: Output = DividePairwise_default (Input1, Input2)

DividePairwise_default divides values of cells of Input1 by values of cells of Input2 pairwise. This is probably always UnitProportion.

T: + Input1.c_n/ Input2.c_n

E: 073

Plan: RepairFormatToThousands_default

D: Output = RepairFormatToThousands_default (Input)

RepairFormatToThousands_default repairs FormatToThousands's division by multiplying values by 1000.

Is this computational goal?

T: Input.c_n*1000

E: 075

Plan: ProportionByPercentage_default

D: Seq2 = ProportionByPercentage_default (Seq1, Cham1)

ProportionByPercentage_default gives a percentage proportion Cham1 from input sequence Seq1.

T: +Seq1.c_n*Cham1/100

E: 046, 062

Plan: ProportionByPercentage_usingConstantReference

D: Seq2 = ProportionByPercentage_default (Seq1, Cham1)

ProportionByPercentage_default gives a percentage proportion Cham1 from input sequence Seq1.

T: +Seq1.c_n*Cham1/Constant1

R: Constant1 = Duplication(100)

E: 087

Plan: PercentageGrowth_default

D: Seq3 = PercentageGrowth_default (Seq1, Seq2)

PercentageGrowth_default gives a percentage grow rate from input sequence Seq1 to input sequence Seq2.

T: +Seq2.c_n/Seq1.c_n*100-100

E: 057

Plan: PercentageGrowth_usingPercentageFormat

D: Seq3 = PercentageGrowth_usingPercentageFormat (Seq1, Seq2)

PercentageGrowth_usingPercentageFormat gives a percentage grow rate from input sequence Seq1 to input sequence Seq2.

T: +(Seq2.c_n-Seq1.c_n)/Seq1.c_n

O: Output values must be formatted with Percentage-format

E: 060

Plan: DivisionByConstant_default

D: Seq2 = DivisionByConstant_default (Seq1, Constant1)

DivisionByConstant_default divides values in input sequence Seq1 by Constant1.

T: +Seq1.c_n/Constant1

E: 059

Plan: DecreaseByPercentage_default

D: Seq3 = DecreaseByPercentage_default (Seq1, Seq2)

DecreaseByPercentage_default decreases values in input sequence Seq1 by percentages in Seq2.

T: $+Seq2.c_n*(1-Seq1.c_n/100)$

E: 062

Plan: PercentageDecrease_default

D: Seq3 = PercentageDecrease_default (Seq1, Seq2)

PercentageDecrease_default gives a percentage decrease rate from input sequence Seq1 to input sequence Seq2.

T: $+(Seq1.c_n-Seq2.c_n)/Seq1.c_n*100$

E: 062

Plan: ProportionPartition_default

D: Seq2 = ProportionPartition_default (Cham1, Cham2, Seq1)

ProportionPartition_default gives a proportion parts of Cham1 by proportions in input sequence Seq1, eg. Cham1 is to be divede to parts which are proportions in Seq1.

R: Cham2 = TotalSum(Seq1)

T: $+Cham1/Cham2*Seq1.c_n$

E: 061

Plan: PartlySum_default

D: Seq1= PartlySum_default (Constant1, Seq2)

PartlySum_default sums a size Constant1 portions of sequence Seq2.

T: $Seq1.c_1 = TotalSum(Seq2.c_1..Seq2.c_{(Constant1)})$

$Seq1.c_n = TotalSum(Seq2.c_{(n-Constant1)}..Seq2.c_n)$

0: The size of TotalSum argument area is Constant1 cells.

E: 106

Plan: FormatTo100Kilometers_default

D: Seq2 = FormatTo100Kilometers_default (Seq1)

FormatTo100Kilometers_default divides the cells of input sequence by 100.

O: Used to format vehicle's usage kilometers to hundreds.

T: $+Seq1.c_n/100$

E: 063

Plan: FormatToHours_default

D: Seq2 = FormatToHours_default (Seq1)

FormatToHours_default divides the cells of input sequence by 60.

O: Used to format minutes to hours.

T: $+Seq1.c_n/60$

E: 063

5.2 Formula Plans

Plan: SumClause_addition

D: Cham1 = SumClause_addition (Input)

SumClause_addition sums up all the values of cells in the input structure using addition operator

T: $+Input.c_1+...+Input.c_n$

E: 047, 49

Plan: SumClause_sumFuntion

D: Cham1 = SumClause_sumFuntion (Input)

SumClause_sumFuntion sums up all the values of cells in the input structure using sum function

T: : @SUM(Input)

E: 047, 090

Plan: Calculator_default

D: Output = Calculator_default(Input)

Calculator_default calculates value of a cell using constant operands and basic operators +,-,* and /.

T: No particular form

O: No other operators or functions.

E: 001, 090

5.3 Application Specific Plans

Plan: MeansEndsAnalysis_default

D: Seq3 = MeansEndsAnalysis_default (Seq1, Seq2, Cham1)

MeansEndsAnalysis calculates a productivity value by means/ends values of the input sequence Seq1 and corrects it by index values Seq2 and Cham1.

T: $+Seq1.c_n/Seq2.c_n*Cham1$

R: Input Seq1 contains previously calculated means/ends values.

Plan: TurnTime_default

D: Cham4 = TurnTime_default (Cham1, Cham2, Cham3)

TurnTime_default computes a turn time of store.

T: +2* Cham1/((2* Cham2)+ Cham3)

R: Cham1 = Yearly need of products

Cham2 = Safety resource of products

Cham3 = Amount to be acquired at time

E: 064

Plan: VacationPay_default

D: Seq3 = VacationPay_default (Seq1, Seq2, 50)

VacationPay_default calculates a vacation time extra payment by dividing the monthly payment in Seq1 with 50 and multiplying this amount by the number of vacation days in Seq2.

T: $+Seq2.c_n*Seq1.c_n/50$

E: 046

Plan: XeroxCopyPrice_default

D: Cham6 = XeroxCopyPrice_default (Cham1, Cham2, Cham3, Cham4, Cham5)

XeroxCopyPrice_default calculates an approximate xerox copy price for square metre taking account of employee's salary and price of materials.

T: +Cham5*Cham2/(Cham4*Cham2+Cham3*Cham1)+Cham2

R: Cham1 = price of paper copy material÷
Cham2 = price of plastic copy material
Cham3 = amount paper used in a year
Cham4 = amount plastic used in a year
Cham5 = employee's salary

E: 054

Plan: AverageSquareMetreAge_default

D: Cham1 = AverageSquareMetreAge_default (Seq1, Seq2)

AverageSquareMetreAge_default calculates an average age for a number of buildings weighted by the area of buildings. Seq1 holds the areas and Seq2 holds the ages.

T: +Cham2/Cham3

R: Cham2=TotalSum(Seq1*Seq2)
Cham3=TotalSum(Seq3)
Seq3=MultiplyPairwise(Seq1,Seq2)

E: 075

Plan: TemperatureIndex_default

D: Seq4 = TemperatureIndex (Seq1, Seq2, Seq3, Constant1, Constant2)

TemperatureIndex calculates a monthly normalized temperature index. Seq1 holds the normalized degree-day for current month, Seq2 holds a sum of energy consumption for last twelve months, Seq3 holds a sum of degree-days for last twelve months.

T: $+\mathrm{Seq1.c_n}*100*\mathrm{Seq2.c_n}/(\mathrm{Seq3.c_n}*11890)$

R: Seq1 = normalized degree-day for current month
Seq2 = TotalSum(energy consumption last twelve months)
Seq3 = TotalSum(degree-days for last twelve months)
Constant1 = 100
Constant2 = 11890

E: 106

Plan: EnergyConsumption_default

D: Seq3 = EnergyConsumption_default (Seq1, Seq2, Constant1, Constant2, Constant3)

EnergyConsumption_default calculates a monthly normalized energy consumption. Seq1 holds a sum of energy consumption for last twelve months, Seq2 holds a sum of degree-days for last twelve months.

T: $+(\mathrm{Seq1.c_n}-12*145)*5267/\mathrm{Seq2.c_n}+12*145$

R: Seq1 = TotalSum(energy consumption last twelve months)
Seq2 = TotalSum(degree-days for last twelve months)

E: 106

Plan: ObtainableSales_default

D: Cham3 = ObtainableSales_default (Cham1, Cham2, Constant1, Constant2)

ObtainableSales_default calculates obtainable sales for a year. Cham1 holds a premis for sales for year, Cham2 holds a turnover for year.

T: +Cham1/360*Cham2*11

R: Cham1 = Premis for sales

Cham2 = Turnover

E: 069

Plan: DepreciatedValues_default

D: Seq4 = DepreciatedValues_default (Seq1, Seq2, Seq3)

DepreciatedValues_default calculates depreciated values for different things. Seq1 and Seq2 hold values of things, Seq3 holds depreciation value factors.

T: $+(Seq1.c_n+Seq2.c_n)*Seq3.c_n$

R: Seg3 = Depreciation value factors

E: 071

Plan: CirculationPeriodFactors_default

D: Seq2 = CirculationPeriodFactors_default (Seq1, Cham1, Cham2, Constant1, Constant2)

TurnoverTimeFactors_default calculates circulation period factors. Seq1 holds the values to be circulated. Cham1 holds a turnover for year, Cham2 holds the number of months in an accounting period.

T: $+Constant1*Seq1.c_n/Cham1*Constant2/Cham2$

R: Cham1 = Turnover

Cham2 = Number of months

Constant1 = Duplication(360)

Constant2 = Duplication(12)

E: 071

Plan: ComputedAreaValue_default

D: Seq2 = ComputedAreaValue_default (Seq1, Cham1, Cham2)

ComputedAreaValue_default calculates a value of build area for company's assets. Seq1 holds the areas to be valuated. Cham1 holds a total sum of the areas, Cham2 holds the weighted value of the area according to geographic location.

T: $+Seq1.c_n/Cham1*Cham2$

R: Cham1 = TotalSum(Seq1)

Cham2 = (Cham1/15000*32450)*40/1000

E: 087

Plan: YearlyCustomerSales_default

D: Seq5 = YearlyCustomerSales_default (Seq1, Seq2, Seq3, Seq4, Constant1)

YearlyCustomerSales_default calculates a yearly customer sales of animal food. Sales are calculated taking into account of percentage of customer sales (Seq1), yearly sales in kilograms (Seq2), price for 100 kilograms (Seq3) and profit percentage (seq4).

T: $+(Seq1.c_n/100*Seq2.c_n)*Seq3.c_n/(100-Seq4.c_n)$

E: 062

5.4 Merged Plans

MPlan: PercentagePairwise_&_SumClause_default

D: Seq2 = PercentagePairwise_withSumClause (Seq1, Tab1)

T: Seq1.c_n/SUM(Tab1.index1)*100

C: Seq1 and Tab1.index1 of equal sizes

E: 019

MPlan: UnitProportion_&_MultiplyByConstant _default

D: Output = UnitProportion_withMultiplyByConstant_default (Input1, Input2, Constant1)

UnitProportion_with MultiplyByConstant _default gives a proportion of a measure by another measure repaired with constant.

T: +Input1.c_x/Input2.c_x*Constant1

E: 016

MPlan: FormatToThousands_&_CorrectionByIndexClause_default

D: Seq2 = FormatToThousands_&_CorrectionByIndexClause_default (Seq1)

FormatToThousands_&_CorrectionByIndexClause_default gives index clause corrected sequenced measured in thousands.

T: +Seq1.c_n/1000*Cham1/Cham2

E: 049

MPlan: DifferencePairwise_&_IndexSum

D: Seq2 = DifferencePairwise (Seq1,_IndexSum(Tab1))

DifferencePairwise_&_IndexSum gives a difference pairwise of input sequence Seq1 and formula goal IndexSum output sequence.

T: +Seq1.c_n-IndexSum(Tab1).c_n

E: 073

MPlan: Duplicates_&_CorrectionByIndexClause_default

D: Seq1 = Duplicates_&_CorrectionByIndexClause_default (Cham1, Constant1.. Constantn)

Duplicates_&_CorrectionByIndexClause_default gives a serie of duplicates of Cham1 and corrects some of them with constants Constant1.. Constantn using formula goal.

T: +Cham1*Constantx

E: 075

MPlan: UnitProportion_&_ RepairFormatToThousands_default

D: Cham3= UnitProportion_&_ FormatToThousands_default (Cham1, Cham2, 1000)

UnitProportion_&_ FormatToThousands_default gives a proportion of Cham1 and Cham2 and multiplies this value by 1000 in order to correct previous FormatToThousands usage.

T: +Cham1/Cham2*1000

E: 075

MPlan: ProportionByPercentage_&_ FormatToThousands _default

D: Seq2= ProportionByPercentage_&_ FormatToThousands_default (Seq1, Cham1, 100000)

ProportionByPercentage_&_ FormatToThousands_default gives a percentage proportion in Seq1 of Cham1 and divides this value by 1000 in order to change value to thousands.

T: $+Seq1.c_n/100*Cham1/1000$

(in matter of fact written $Seq1.c_n*Cham1/100000$)

E: 059

MPlan: Input_&_CorrectionByIndexClause_default

D: Seq2= Input_&_CorrectionByIndexClause_default (Cham1, Constant1, Constant2)

Input_&_CorrectionByIndexClause_default transforms the original input in Constant2 by index correction Cham1 and Constant1. This is novice merged goal. Probable cause could be that novice does not want to see the original inputs and does not know how to hide cells so he puts the inputs to the formulas.

T: +Cham1/Constant1*Constant2

E: 048, 049

MPlan: TotalSum_&_FormatToThousands_default

D: Seq2= TotalSum_&_FormatToThousands_default (Seq1, 1000)

TotalSum_&_FormatToThousands_default calculates a total sum of Seq1 and formats it to thousands.

T: +TotalSum(Seq1)/1000

E: 033

MPlan: MultiplyPairwise_&_ DifferencePairwise_default

D: Seq4= MultiplyPairwise_&_ DifferencePairwise_default (Seq1, Seq2, Seq3)

MultiplyPairwise_&_ DifferencePairwise_default calculates a pairwise difference of Seq1 and Seq2 and multiplies result pairwise with Seq3.

T: $+Seq3.c_n*(Seq1.c_n-Seq2.c_n)$

E: 108

MPlan: MultiplyPairwise _&_FormatToThousands_default

D: Seq3=MultiplyPairwise_&_FormatToThousands_default (Seq1, Seq2, 1000)

MultiplyPairwise_&_FormatToThousands_default multiplies Seq1 and Seq2 pairwise and formats Result to thousands.

T: $+Seq1.c_n*Seq2.c_n/1000$

E: 115

References:

Bonar, J. G. (1985). "Understanding the Bugs of Novice Programmers." Ph.D. Thesis, University of Massachusetts.

Brooks, R. (1983). "Towards a Theory of the comprehension of computer programs." International Journal of Man-Machine Studies 18 : 543-554.

Detienne, F. (1990). Expert Programming Knowledge: A Schema-based Approach. Psychology of Programming Ed. J.-M. Hoc Green, T. R. G., Samurcay, R., Gilmore, D. J. London, Academic Press. 205-222.

Ehrlich, K., Soloway E. (1984). An Empirical Investigation of the Tacit Plan Knowledge in Programming. Human Factors in Computer Systems Ed. J. T. &. M. Scheider. Ablex Publ. Co. 113-133.

Rist, R. S. (1989). "Schema Creation in Programming." Cognitive Science 13 : 389-414.

Sajaniemi, J., Pekkanen, J. (1988). "An empirical Analysis of Spreadsheet Calculation." Software-Practice and Experience 18 (6): 583-596.

Soloway, E., Ehrlich, K. (1984). "Empirical Studies of Programming Knowledge." IEEE Trans. on Software Engineering 10 (5): 595-609.

Soloway, E., Ehrlich, K., Bonar, J., Greenspan, J. (1982). What Do Novices Know About Programming. Directions in Human/Computer Interaction Ed. A. Badre Shneiderman, B. Ablex Publ. Co. 27-54.

Spohrer, J. C., Soloway, E., Pope, E. (1985). "A Goal/Plan Analysis of Buggy Pascal Programs." Human-Computer Interaction 1 (2): 163-207.