Advanced Splitter Algorithm of the Content-Driven Template-Based Layout System

István Albert, Hassan Charaf and László Lengyel

Budapest University of Technology and Economics
Magyar Tudósokkörútja2, 1117 Budapest, Hungary

Abstract: With the immense popularity of mobile devices, there is an increasing need for adaptive layout solutions that allow for the automatic layout calculation of online magazines and newspapers. The Content-Driven Template-Based Layout System (CTLS) is a template-based online magazine layout approach. This paper introduces the advanced splitter algorithm of the CTLS approach. The splitter algorithm allows for applying both horizontal and vertical splitter components, embedding them into each other and in doing so build hierarchical layout definitions. The approach focuses on the adaptation methods of the layout elements themselves. Applying the splitter algorithm we deduce the resulting adaptation method at any level of the layout element hierarchy. The paper discusses the motivation, capabilities and boundaries of the algorithm, which include the conditions and requirements that make the algorithm more effective compared to the brute force equation solution. We investigate both the homogeneous and mixed splitter hierarchies. Furthermore, we introduce the operational state machine of the splitter algorithm.

Keywords: Online Magazine Layout, Adaptive Layout, Content-Driven Layout, Template-Based Layout, Splitter Algorithm

1. INTRODUCTION

The role of the mobile devices, including tablet devices, is relevant [1] [2] to the IT sector. Mobile device users continuously consume digital content; but the diversity of mobile platforms and the mobile device capabilities requires providing automatic layout solutions for online content. Consumers want to utilize those features which make their specific device unique. This requires that the magazine content be adaptable to the various tablets. We know that high-quality automatic document formatting is a difficult problem [3]. However, through our work and the motivations of our research activities, the answer to this problem is automatic layout approaches.

The main challenge is to automatically adapt the whole digital magazine content in order for articles to look as good on a tablet display, of any size, as they do in printed media.

Content-Driven Template-Based Layout System (CTLS) [4] [5] templates are column templates. The height of a column is fixed, based on the display properties of the device. Then the ideal width of the column is automatically calculated, based on the text size and horizontal scrolling is also made possible.

Templates assist in manipulating each different layout and are also applied to define one or more columns, i.e., a template which covers the whole column, from top to bottom. Layouts are composed of one or more templates. Rules (constraints) relate to and are applied in both templates and layouts. The basic composition of a template consists of rectangular areas that are arranged in the template and filled with content. Text, image and caption comprise the basic layout elements. In [6], we have demonstrated the reference layouts of the Content-Driven Template-Based Layout System.

The adaptation methods are the focus of the layout elements and size is not considered heavily. The behavior of the layout is decided by the methods during the layout calculation. Layout templates are often hierarchical which means the adaptation of these templates must be handled. This issue of layout component handling is addressed through equation sets which describe their behavior. The paper discusses the Advanced Splitter Algorithm, i.e., the common application of the horizontal splitter and vertical splitter components, during the template definition. Based on the equation forms of the adaptation methods, our automatic layout related results are created. The approach makes it possible to use the domain-specific properties of the area in order to effectively calculate the adaptation method of compound layout elements. The algorithm of our approach accounts for the behavior of the contained layout elements. As a result we can efficiently manage the layout calculation of hierarchically defined compound layout templates.

Section 2 discusses the basic elements of the layout approach. Section 3 introduces the details of the mixed splitter algorithm and we also define the hierarchy of the layout elements and discuss the boundaries of the algorithm. Section 4 elaborates the operational state machine of the splitter algorithm. Finally, we conclude the paper.
2. THE BASICS OF THE LAYOUT APPROACH

The CTLS approach makes it possible to define the requested layout through hierarchical layout elements. The layout elements are embedded into each other. The sizing and layout of the elements, contained by a template, is determined by their equation sets and the rules related to the template (e.g., the fixed height of the template in the case of a root template).

This approach, in essence, differs from other solutions (e.g. Windows Presentation Foundation [7] [8] and the table calculation of HTML [9]), which apply measure-arranged algorithms and handles the layout elements as black boxes without any extensive knowledge of the resizing method. The approach accounts for the adaptation method of all layout elements, not only minimum and maximum sizes of elements for certain size constraints. In this way, the approach handles the layout hierarchy as a whole system.

In the full equation set of a template, there are the equation sets of the primitive elements (zero, one or two equations for every element), and every splitter component has a further two and two equations.

The basic layout elements (text, image and caption) can contain the following behavior (adaptation) methods (also provided in Table 1):

1. Free text; Free (O): it has optional width and height.
2. One text column; Fixed Width (W): the width is fixed, but the height is optional. (x=Const.)
3. Header image; Fixed Height (H): the height is fixed, but the width is optional. (y=Const.)
4. Fixed image; Fixed (F): both the height and the width are fixed. (x=Const.1, y=Const.2)
5. Resizable image; Fixed Ratio (X): the ratio is constant. (x/y=Const.)
6. Caption (finite text); Fixed Area (+): the consumed area is constant. (x*y=Const.)

The domain-specific equation set of the Horizontal Splitter Component:

\[ x_{splitter} = x_1 = \cdots = x_n \]  (1)
\[ y_{splitter} = \sum_{i=1}^{n} y_i \]  (2)

The domain-specific equation set of the Vertical Splitter Component:

\[ x_{splitter} = \sum_{i=1}^{n} x_i \]  (3)
\[ y_{splitter} = y_1 = \cdots = y_n \]  (4)

In order to make the layout calculation more efficient, the following domain-specific equation sets are utilized by the splitter algorithms of our layout approach.

3. THE ADVANCED SPLITTER ALGORITHM

Instead of applying a brute force equation set solving solution, we utilize the domain-specific properties of the area and suggest a more efficient substitution-based approach. The approach applying substitutions reduces the complexity of the original equation set (Set). The resulted equation set (Set') contains fewer equations than the original equation set: #Set'<#Set, where # denotes the cardinality of the set. In an optimal case, the algorithm ends with a single solvable equation.

The algorithm identifies those points of the layout hierarchy in which the substitution cannot be applied. Either the points should be released with additional constraints, e.g. providing fixed height or fixed width values for certain layout elements, or it is necessary to solve the resulted, nonlinear equation set by applying classical numerical methods.

3.1 The Hierarchy of the Layout Elements

The basic, building blocks of the hierarchical templates are the primitive elements. They represent the leaves of the template tree. A splitter components composed of leafs. These splitter components can contain not only primitive elements, but further splitter components as well. The meta-model (a UML class diagram) of the Template Tree language is displayed in Figure 1. Green nodes represent the primitive layout elements, and blue nodes indicate the two types of splitter components.

The domain-specific equation set of the Horizontal Splitter Component:

\[ x_{splitter} = x_1 = \cdots = x_n \]  (1)
\[ y_{splitter} = \sum_{i=1}^{n} y_i \]  (2)

The domain-specific equation set of the Vertical Splitter Component:

\[ x_{splitter} = \sum_{i=1}^{n} x_i \]  (3)
\[ y_{splitter} = y_1 = \cdots = y_n \]  (4)

In order to make the layout calculation more efficient, the following domain-specific equation sets are utilized by the splitter algorithms of our layout approach.

Figure 1: The metamodel of the Template Tree language

In a template tree, the equations of the leaf elements correspond to the equation sets 1-6 (Table 1), and the further tree nodes are based on the equation sets 7-26 (Table 1). The supplemental constraints of the splitter components (e.g. fixed height or fixed width values) are defined by additional equations. These additional equations most often simplify the equation solving procedure.

Definition (Template Tree). The template tree is a model built from primitive layout elements (text, image and caption) and splitter components (horizontal and vertical splitter) according to the Template Tree language meta-model.

Figure 2 depicts an example hierarchical layout definition and the corresponding template tree. The template
contains two splitter components ($S1$ and $S2$), an image (1) and splitter $S2$ are embedded into splitter $S1$. Splitter $S2$ contains a text area (2), an image (3), and a caption (finite text) component (4).

**Figure 2** Example hierarchical layout definition and the corresponding template tree

Regarding the online magazine we have defined the following requirements:

1. The magazine should be displayed properly on a variety of screens, i.e., the approach should support optional screen resolution. This means that, vertically, the layout should automatically adapt to the device properties.
2. The content can be scrolled horizontally, therefore, the calculated layout has no horizontal restrictions.

The above requirements are applied for the root template. Within their constraints it should be possible to parameterize the root template in the vertical direction, and the template should be either a calculated or fixed size in the horizontal direction. If the horizontal direction values are calculated, then they are based on the vertical settings. These considerations provide possible adaptation methods of the root template element: Fixed Width (W), Fixed Ratio (X), Fixed Area (+), Calc Ratio (C) or Free (O). In the case of the Free (O) adaptation method, the approach can determine the width of the column template, e.g., it can apply the default column, text element width.

**Note.** If, instead of the horizontal scrolling feature, we wanted to provide a page-based layout, then the root template should have a Free (O) adaptation method in order to be applicable on optional screen sizes. In a similar way, we can conclude that the vertical scrolling could be supported with the following adaptation methods: Fixed Height (H), Fixed Ratio (X), Fixed Area (+) or Calc Ratio (C).

**Definition (Basic Layout Element).** The adaptation method of a basic layout element is one of the following: Free (O), Fixed Width (W), Fixed Height (H) or Fixed (F).

These adaptation methods correspond to the equation forms 1-4 in Table 1.

**Proposition.** The number of the equations related to a template tree is linear to the number of the layout elements.

**Proof.** Table 1 defines the equation sets related to simple and compound layout elements. Each equation set contains a maximum of two equations. Therefore, the total number of the equations is linear to the number of the layout elements.

### 3.2 The Steps of the Algorithm

Splitter components facilitate to place layout elements one below another (or one next to another) and calculate the resulting adaptation method. The algorithm calculates the equation set of the splitter component, based on the equations related to the behavior of different layout elements. The resulting equation set determines the adaptation method based upon the related equation forms.

Supposing that the algorithm can accomplish this for all splitter elements, the substitutions result in a single equation set. In order to end with a closed form, final equation set, regarding the splitter, the algorithm applies the following steps:

1. Based on the domain-specific equation ($x_{splitter} = x_1 = \cdots = x_n$), the equations of the children element $x_{splitter}$ values are substituted with $x_{splitter}$.
2. The algorithm substitutes the $y$ values with the equation provided from the appropriate child element into the $y_{splitter} = \sum_{i=1}^{n} y_i$ equation. Because of the first step, the child element equations contain only the $x_{splitter}$ variable. The resulting equation set contains only $x_{splitter}$ and $y_{splitter}$ variables. Therefore, this equation set accurately describes the behavior of the splitter component.

The first step of the algorithm can be always performed. The second step can be achieved if can be expressed, based upon , in a closed form.

This approach allows the embedding of splitter components into another, therefore, the above two steps of the algorithm are applied in a recursive way. The algorithm results in an equation set, in which the number of unknown variables corresponds to the number of independent equations. Such a solvable equation set is related to the root template as well, but an additional constraint is also considered: $\text{Screen} \times \text{Splitter} = \text{Height}$. We regard this constraint as domain-specific which simplifies the solving of the equation set. In a similar way, if a layout template cell has a fixed height or width, then the related equation set is often simplified based on these additional constraints.

**Proposition.** The Splitter Algorithm results in an equation set for every level of the template hierarchy.

**Proof:** It follows, from the above considerations, that the steps of the algorithm, and the variable substitution
method (which is applied by the algorithm).

Our concepts and algorithms containing the appropriate modifications, can be applied for vertical splitter components as well. The related condition is the following: The second step can be achieved if the can be expressed based on in a closed form.

Proposition. The Splitter Algorithm can be applied for both horizontal and vertical splitter elements.

Proof. Horizontal and vertical splitter components perform similarly. The difference lies in the element order: in a horizontal splitter, the layout elements are placed one below another, whereas in a vertical splitter, they are placed one next to another. The application of the algorithm for horizontal splitters is discussed above. Commuting the and variables of the algorithm is applicable for the vertical splitter in a similar way.

The above considerations depend upon certain adaptation methods and layout definitions. Free (O) and Fixed (F) adaptation methods always satisfy the above conditions. The cases which require further investigation are the followings:

- Calc. Ratio (C) adaptation method, and
- Layout definitions resulting Calc. Ratio (C) adaptation method splitter component.

Real world examples most often fall into this category. Therefore, we further investigate this branch of the algorithm. Other cases are covered by either the horizontal splitter component or vertical splitter component-related domain-specific, equation sets (1-4 equations), and variables are either constant or have no constraints, therefore, the substitution procedure can be performed in a systematic way.

The pressing question is: What will be the resulting adaptation method of two layout elements below or next to each other, if both of these elements have Calc. Ratio (C) adaptation method? Essentially, this means the contraction of two Calc. Ratio (C) layout elements. Thus, we investigate the simplification process of these and similar types of layout elements.

Figure 3 shows the contraction of layout elements with equation sets 5 and 6. The resulting equation set is 9. It can be written in the following way: 5+6=9. Note, that equation set 5 is related to the Fixed Ratio (X) and equation set 6 is related to the Fixed Area (+) adaptation methods (Table 1). These two adaptation methods are the two simplest Calc. Ratio (C) methods. In fact, every case is related to the Calc. Ratio (C) adaptation method, in which the relation between the and is defined by a function.

3.3 The Boundaries of the Algorithm

Definition (Degree of an equation). The degree of an equation is the maximum number of times any variable or variables can be multiplied together in any single term.

The degree of an equation is used to help decide how to solve an equation or whether or not an equation has a solution.

In order to identify the boundaries of the layout algorithm, we define those conditions and requirements that make the algorithm more effective when compared to the brute force equation solving method.

- **Condition 1 (Closed Form Conditions).**
  Horizontal Splitter related condition: The $\frac{X}{Y}$ values should be expressed based on $x_1$ values in a closed form and with the help of the equation set describing the relation of $x$ and $y$ values.
  Vertical Splitter related condition: The $x_2$ values should be expressed based on $y$ values in a closed form and with the help of the equation set describing the relation of $x$ and $y$ values.

- **Condition 2 (Substitution Condition).** During the layout calculation, we accept those substitutions that result in equations containing a maximum degree of 2. There exist formulas to solve cubic and quartic functions, but involving these methods into the algorithm does not provide a globally efficient solution.

- **Condition 3 (Single Equation Condition).** The result of the algorithm is also accepted if the result is a single equation (not an equation set), having a degree more than 2. Applying the current, effective numerical methods, solving such a single equation can be quickly solved even on mobile devices, in which resources are limited.

Regarding the closed form condition, we take interest in the forms of the equations and concentrate on the equation forms and not on specific, constant values. The algorithm contracts the layout elements of the template tree. The algorithm starts with the leaf nodes and, in a systematic way, reaches the top of the tree. This is a bottom-to-top method.

Definition (Light node of the template tree). In a light node, contracting the layout elements according to the splitter algorithm, the resulting equation set, the degree of all of the equations is maximum 2.

Definition (Heavy node of the template tree). In a heavy node, contracting the layout elements according to the
splitter algorithm, in the resulting equation set - there is at least one equation with degree \( \geq 3 \).

By default, every leaf of the template tree is light node.

The splitter algorithm performs layout element contractions and calculates the resulting adaptation method of the compound element. Below, we demonstrate the complexity of the differing contractions.

**Definition (Contraction complexity).**

- **Complex contraction**: increases the degree of the original equation.
- **Neutral contraction**: preserves the degree of the original equation.
- **Simplifying contraction**: decreases the degree of the original equation.

### 3.4 Homogeneous Hierarchy

Horizontal splitter components result the following equation form: \( \sum \mathbf{HSE} = \sum \mathbf{HSE} \). This means that the Condition 1 (Closed Form Conditions) and Condition 2 (Substitution Condition) always remain true if we embed only horizontal splitter components into each other. Analogously, this consideration is also true if we embed only vertical splitter components into each other. The resulting equation has the following form:

\[
\sum \mathbf{HSE} = \sum \mathbf{HSE}.
\]

**Proposition.** If we embed only one type of splitter components into each other, then the Condition 1 (Closed Form Conditions) and Condition 2 (Substitution Condition) are always true.

**Proof.** The proof follows from the equation forms introduced above.

We have already introduced the Horizontal Splitter Equation Set (HSE Set) and the Vertical Splitter Equation Set (VSE Set) [10]. In case of the HSE Set the resulting equation has the following form: \( y = c_1 + x + c_2 \cdot \frac{z}{2} \). Similarly, in the case of the VSE Set, the resulting equation has the following form: \( x = c_1 + y + c_2 \cdot \frac{z}{2} \).

**Proposition.** The equation forms of the HSE Set and the VSE Set satisfy the Condition 2 (Substitution Condition).

**Proof.** This results from the above introduced equation forms, because these forms define quadratic equations.

### 3.5 Mixed Hierarchy

In real world templates, horizontal and vertical splitters are embedded into each other in a variety of ways. The question related to these types of templates is as follows: If we apply both horizontal and vertical splitters, embedded into each other, then which cases fulfill Condition 1 (Closed Form Conditions)?

The substitution method of the algorithm works till we can express variables by closed form equations, having a maximum degree of 2. When this cannot be achieved, the algorithm marks these heavy nodes of the template tree. Based on the marked layout elements, the online magazine editor can introduce further constraints, e.g., fixed height or fixed width values. These constraints can simplify the layout contraction process and transform the heavy nodes into light nodes.

**Proposition.** The number of the contraction operations is linear with the number of the tree nodes: \( O(n) \), where \( n \) is the number of the tree nodes.

**Proof.** Tree nodes are either simple layout elements or splitter components. The adaptation methods related to simple layout elements are based on equations 1-4 (Table 1), and the adaptation methods related to splitter components are based on equations 1-4 (Table 1). Calculating the resulting adaptation method of a splitter component is based on a contraction operation. A splitter component contains at least two layout elements, therefore each contraction decreases the number of the unprocessed layout elements. This means that the number of required contraction operations is based on the number of the tree nodes: \( O(n) \).

**Proposition.** Assume a template tree \( N \), where the Condition 1 (Closed Form Conditions) and Condition 2 (Substitution Condition) are true for each layout element contraction. This implies that the template tree can be evaluated in \( d \) steps, where \( d \) is the number of the template tree levels.

**Proof.** The number of template tree levels is the depth of the template tree. The proposition states that we can evaluate a tree level in one step. The hierarchy of the template tree is organized from primitive layout elements (text, image and caption) and from splitter components. Based on the assumptions (Condition 1 and Condition 2), all layout contradictions defined by the splitter components can be performed and the layout contradictions produce usable results for the preceding hierarchy level. Therefore, in \( d \) steps we reach the final result, i.e., we receive the adaptation method of the whole template.

Table 1 provides the equation sets related to simple and compound layout elements. Equation sets 1-6 are related to the primitive elements (text, image and caption), and equation sets 7-26 are related to the combination of the equations 1-6. The equations with degree greater than 2 are omitted, because they are not supported by the substitution mechanism of the approach. The solution works with equations having a maximum degree of 2.
Furthermore, based on Condition 3 (Single Equation Condition) the approach can handle single equations with degree more than 2.

4. THE OPERATIONAL STATE MACHINE OF THE SPLITTER ALGORITHM

The adaptation, methods-related, equation forms table (Table 1) defines the equation forms resulted from the contraction of certain layout components. The emphasis is on the form of the equations. In the case of the horizontal splitter component, we define the following equation elements:

- A: c (Constant)
- B: \frac{c}{x}
- C: \frac{c}{x^2 + 1}
- D: \frac{c}{x^3 + 2x + 3}

The equations can be assembled based on these equation elements. For example equation 8 is AB, and equation 12 is ABC. We indicate the signs of the equation elements (A, B, C and D) and place them next to each other. The operation between the equation elements is the addition operation, because the addition operation is utilized in the splitter equations. We can convert the equation element signs, because the addition operation is commutative.

Assume that we contract two layout elements with Calc. Ratio (C) adaptation methods in a horizontal splitter. The closed form of variable based on variables is provided for both layout elements, therefore, we have to process the following operation: \frac{c}{x^2 + 1} + \frac{c}{x^3 + 2x + 3}. The beginning state is related to the equation form and the transition is in accordance with equation form. The contraction can be performed step-by-step, according to the rules of the addition operation. Adding the elements of either results in a new equation form or remains the same.

These considerations provide solutions for online magazine editors as well. Editors define the magazine layouts through the building of hierarchical templates. If the resulting adaptation method cannot be calculated based on the conditions we require (Conditions 1-2-3), then the algorithm marks the heavy nodes in the template hierarchy. Next, the editor should add further constraints to make the template valid (free of heavy nodes).

5. CONCLUSION

In this paper we have introduced the advanced splitter algorithm of the Content-Driven Template-Based Layout System. The algorithm allows applying both horizontal and vertical splitter components and includes their embedding into each other. With the help of this algorithm, we can deduce the resulting adaptation method on any level of the layout hierarchy.

The approach utilizes the domain-specific properties of the area. These properties are the equation set of the horizontal splitter component and the equation set of the vertical splitter component. We have discussed the hierarchy of the layout elements and have also defined the Template Tree.

We have analyzed the boundaries of the algorithm, which include the conditions and requirements that make the algorithm more effective when compared to the brute force equation solving method. Additionally, we have provided the Closed Form, Substitution and Single Equation conditions.

We have investigated both the homogeneous and mixed splitter hierarchies. Finally, we have introduced the operational state machine of the splitter algorithm. This state machine simplifies the implementation of the approach, because it substitutes the application of equation forms introduced in Table 1.

We believe that the results of the introduced advanced splitter algorithm contribute to address many relevant and previously unaddressed areas of the current online magazine layouts.
Table 1: The adaptation methods related equation forms

<table>
<thead>
<tr>
<th>Eq. ID</th>
<th>Adaptation mode</th>
<th>Equations (x)</th>
<th>Equations (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Free (O)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Fixed Width (W)</td>
<td>$x = c$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Fixed Height (H)</td>
<td>$y = c$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Fixed (F)</td>
<td>$x = c_1 + c_2$</td>
<td>$y = \frac{1}{c}$</td>
</tr>
<tr>
<td>5</td>
<td>Fixed Ratio (X)</td>
<td>$x = c + y$</td>
<td>$y = \frac{1}{c}$</td>
</tr>
<tr>
<td>6</td>
<td>Fixed Area (Y)</td>
<td>$x = c + \frac{1}{y}$</td>
<td>$y = \frac{c}{x}$</td>
</tr>
<tr>
<td>7</td>
<td>Calc. Ratio (C)</td>
<td>$x$ is expressed from equation y.</td>
<td>$y = c_1 * x + c_2 * \frac{1}{x}$</td>
</tr>
<tr>
<td>8</td>
<td>Calc. Ratio (C)</td>
<td>$x = c_1 * y + c_2$</td>
<td>$y = c_1 * x + c_2$</td>
</tr>
<tr>
<td>9</td>
<td>Calc. Ratio (C)</td>
<td>$x = c_1 + \frac{1}{y}$</td>
<td>$y = c_1 * x + c_2$</td>
</tr>
<tr>
<td>10</td>
<td>Calc. Ratio (C)</td>
<td>$x = c_1 + y + c_2 * \frac{1}{y}$</td>
<td>$y$ is expressed from equation x.</td>
</tr>
<tr>
<td>11</td>
<td>Calc. Ratio (C)</td>
<td>$x = c_1 + \frac{1}{y} + c_2$</td>
<td>$y = c_1 * x + c_2$</td>
</tr>
<tr>
<td>12</td>
<td>Calc. Ratio (C)</td>
<td>$x$ is expressed from equation y.</td>
<td>$y = c_1 * x + c_2 * \frac{1}{x}$</td>
</tr>
<tr>
<td>13</td>
<td>Calc. Ratio (C)</td>
<td>$x = c_1 * y + c_2 * \frac{1}{y} + c_3$</td>
<td>$y$ is expressed from equation x.</td>
</tr>
<tr>
<td>14</td>
<td>Calc. Ratio (C)</td>
<td>$x = \frac{c_1}{y + c_2} + c_3$</td>
<td>$y = \frac{c_1}{x + c_2} + c_3$</td>
</tr>
<tr>
<td>15</td>
<td>Calc. Ratio (C)</td>
<td>$x$ is expressed from equation y.</td>
<td>$y = c_1 * x + c_2 * \frac{1}{x} + c_3$</td>
</tr>
<tr>
<td>16</td>
<td>Calc. Ratio (C)</td>
<td>$x = \frac{c_1}{y + c_2} + c_3 * y$</td>
<td>$y$ is expressed from equation x.</td>
</tr>
<tr>
<td>17</td>
<td>Calc. Ratio (C)</td>
<td>$x = \frac{c_1}{y + c_2} + c_3 * y$</td>
<td>$y$ is expressed from equation x.</td>
</tr>
<tr>
<td>18</td>
<td>Calc. Ratio (C)</td>
<td>$x = \frac{c_1}{y + c_2} + c_3 * y$</td>
<td>$y$ is expressed from equation x.</td>
</tr>
</tbody>
</table>

ACKNOWLEDGEMENTS

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) organized by VIKING Zrt. Balatonfüred.

References