The Horizontal Splitter Algorithm of the Content-Driven Template-Based Layout System

ISTVÁN ALBERT, HASSAN CHARAF, LÁSZLÓ LENGYEL
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
HUNGARY
{ialbert, hassan, lengyel}@aut.bme.hu

Abstract: - We live in an information consumer society; day by day people consume lots of digital content. The reason behind it is that almost all of us have mobile devices that support web browsing, multimedia, and reading different document formats, like HTML, ePub or PDF files. Based on the current trends we can say that tablets are going to be dominant consumer devices. The diversity of the tablets and tablet screens require adaptive layout solutions that utilize the different device capabilities. This paper discusses the Horizontal Splitter Algorithm of the Content-Driven Template-Based Layout System (CTLS). The CTLS is a template-based online magazine layout approach with a unique property: the automatically created magazine layout is driven not only by templates and device properties but also by the actual content. The Horizontal Splitter Component of the approach facilitates to place layout elements one below another and calculate the resulting adaptation method. This paper discusses the equations related to different layout elements and introduces the way how the splitter component derives the resulting adaptation method based on the equation forms of the contracted layout elements. As a result the approach can handle hierarchical templates in an efficient way.

Key-Words: - Adaptive Layout, Content-Driven Layout, Template-Based Layout, Online Magazine Layout.

1 Introduction
Mobile devices are significant part of our daily life, we use them to reach and consume digital data. According to different surveys [1] [2] the role of the tablet devices is already quite relevant and they are going to determine the close future of the mobile devices. Tablet owners continually consume digital content, for example web portals, online newspapers and online magazines. A reasonable part of online documents have a fixed layout, like Adobe’s Portable Document Format (PDF). This inflexibility leads to a poor online reading experience, since the size and resolution of monitors and mobile devices require readers to scroll around in order to read a page [3] [4].

The diversity of mobile platforms and the device capabilities requires providing automatic layout solutions for online newspapers and magazines. Readers would like to utilize the special capabilities of their own devices. This requires that the magazine content adapt to the actual tablet, e.g. take into account the actual screen size, aspect ratio and preferred font size. This requirement highlights the importance of automatic layout approaches.

The challenge is to automatically adapt the whole digital magazine content, text and graphics, in order to articles look as good on tablet displays of any size as they do in printed media. The layout adaptation is required when the reader opens the magazine article or when she modifies e.g. the text size. This means that not only different devices but also different settings require rendering a new layout.

Online quality layout and design is mainly based on different table/grid solutions, e.g. HTML tables [5] and CSS [6] [7] or the different layout engines of certain programming frameworks, like Windows Presentation Foundation [8].

We have to admit that it is certainly possible to produce grid-based page design using HTML, but designers mostly apply fixed-size tables that do not adapt to the reader’s display. Still we want to improve the quality of online publication design. If we take into account the diversity of screens, e.g. sizes and resolution, preparing the appropriate layout for all of them means that the complexity and expense of quality publication design are high.

To improve the limitations of current online reading experience, we need adaptive layout solutions that support automatic layout preparation for a wide range of displays. The Content-Driven Template-Based Layout System (CTLS) [9] [10] proposes a layout approach that targets the tablet devices. The solution facilitates to define column-
based templates and the rules (constraints) of the required layout. Both templates and rules can be expressed in an intuitive way, using a high-level, editor-friendly language. The layout engine takes into account the properties of the actual display, the applied templates, required text size, the actual content, and prepares the appropriate layout. A remarkable property is that not only the templates, but the actual content also has an effect on the resulted layout.

In this paper we introduce the **Horizontal Splitter Algorithm** of the Content-Driven Template-Based Layout System. A motivation of this approach is the following: we should not concentrate on the size of the elements, i.e. we do not ask their requested size, but we work with their adaptation method. The method defines the behavior of the layout elements during the resizing. A layout template is often hierarchical thus we have to handle the adaptation of these types of templates. We address the layout component handling by providing equation sets describing their behavior. The paper discusses that the horizontal splitter algorithm is based on the adaptation methods related equation forms.

Section 2 briefly introduces the main features of our content-driven template-based layout approach. Section 3 discusses the details of the horizontal splitter algorithm including the algebraic structure of our approach. Finally, we conclude the paper.

### 2 The Content-Driven Template-Based Layout System

The Content-Driven Template-Based Layout System (CTLS) [9][10] provides an adaptive document layout approach. CTLS facilitates to design layout templates that are flexible to many different display capabilities and user settings, such as text size. With this approach magazine editors can define layout preferences and conditions with minimal effort. This section summarizes the main properties of our adaptive layout approach.

In CTLS, templates refer to column templates. The height of the column is fixed based on the display properties of the device. The ideal width of the column is automatically calculated based on the text size. The screen can be scrolled in the horizontal direction.

The layout is defined with the help of the templates. We use templates to define one or more columns, i.e. a template covers the whole column from the top till the bottom. A layout is assembled from one or more templates. Rules (constraints) are related to both templates and layouts.

The basic building blocks of templates are **rectangular areas** that are arranged in the template and filled with content. Each area receives content from one of the document streams. Figure 1 introduces a sample layout. This layout contains a highlighted text area, four images, and three further text areas that present the body text. We use grey background to indicate that a text area is a highlighted text area. This means that the length of the text in this area is fixed, i.e. the text does not flow into this area and flow out from it. The arrows show the flow of the body text among text areas.

![Sample basic layout](image)

**Figure 1. Sample basic layout.**

Templates can be hierarchical and a template may have substitute templates. They are ordered, and if the original template cannot be applied for the actual content or screen size/resolution then the layout engine uses the next substitute template.

Beside these elements the approach supports the following features: different content streams, managing aspect ratio of images, non-rectangular areas, handling empty spaces, calculating margin and padding, and managing backgrounds [9].

### 3 The Horizontal Splitter Algorithm

This section introduces the way how the Horizontal Splitter Component of our approach works. This component facilitates to place layout elements one below another and calculate the resulting adaptation method. We provide the equations related to the behavior of different layout elements and discuss the way how the component derives the resulting adaptation method based on the equation forms of the combined layout elements.

#### 3.1 Basics of the Horizontal Splitter Component

We build layout templates from basic layout elements: text, image and caption. A *splitter component* contains two or more elements ordered in one direction (horizontal or vertical direction). In
a Horizontal Splitter Component elements are arranged one below another. Figure 2 provides an example, where three elements are arranged one below another. The red lines indicate the borders between the different elements.

![Example horizontal splitter component.](image)

Figure 2. Example horizontal splitter component.

Every primitive content type has its default adaptation mode. The actual questions are: What happens with the splitter component if we put different elements one below another and define their height calculation method? What will be the resulted adaptation method of the horizontal splitter component?

Layout templates contain hierarchically assembled cells. Every cell has two basic properties: (i) the adaptation method (resizing mode) of the contained element, and (ii) the height calculation method. The contained element can be a simple layout element or a complex one, e.g. another horizontal splitter component.

The basic layout elements (text, image and caption) have the following behavior (adaptation) methods:

- **Free (O)**: it has adjustable width and height.
- **Fixed Width (W)**: the width is fixed, but the height is optional.
- **Fixed Height (H)**: the height is fixed, but the width is optional.
- **Fixed (F)**: both the height and the width are fixed.
- **Fixed Ratio (X)**: the ratio is constant \((x/y = \text{Const.})\).
- **Fixed Area (+)**: the consumed area is constant \((x*y = \text{Const.})\).
- **Calc. Ratio (C)**: for a given width it calculates the appropriate height, and vice versa. This is a generalized version of the last two options.

The behavior of the elements related to certain adaptation methods are defined with equations. Table 1 summarizes the adaptation methods related equation forms.

<table>
<thead>
<tr>
<th>Equation ID</th>
<th>Adaptation method</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Free (O)</td>
<td>(-)</td>
</tr>
<tr>
<td>2</td>
<td>Fixed Width (W)</td>
<td>(x = c)</td>
</tr>
<tr>
<td>3</td>
<td>Fixed Height (H)</td>
<td>(y = c)</td>
</tr>
<tr>
<td>4</td>
<td>Fixed (F)</td>
<td>(x = c_1) (y = c_2)</td>
</tr>
<tr>
<td>5</td>
<td>Fixed Ratio (X)</td>
<td>(\frac{x}{y} = c)</td>
</tr>
<tr>
<td>6</td>
<td>Fixed Area (+)</td>
<td>(x*y = c)</td>
</tr>
<tr>
<td>7-10</td>
<td>Calc. Ratio (C)</td>
<td>(y = c_1 + x + c_2) (7) (y = \frac{c_1}{x+c_2}) (8) (y = c_1 + x + c_2 + \frac{1}{x} + c_3) (9) (y = c_1 + x + c_2 + \frac{1}{x} + c_3) (10)</td>
</tr>
</tbody>
</table>

Besides the above considerations, our approach suggests the following height definition methods:

- **FixedH (FH)**: The height is defined with a fixed value. It can be a screen rate (e.g. 30% of the screen height) or expressed with ideal row height (the number of the text rows). Allowed cell content adaptation methods: Free and Fixed Height (the contained element determines the height).
- **FixedHW (FHW)**: The fixed height value is defined like in the case of FixedH, but this cell determines the width of the template as well. Therefore, only one with this type of cells can be added into the table. Allowed cell content adaptation methods: Fixed Ratio, Fixed Area, Calc. Ratio, Fixed Width and Fixed.
- **Auto (A)**: The height is determined by the contained element. For a given width the height is automatically calculated. The width can be defined by the splitter component (e.g. by a nearby Fixed Width type cell) or by a surrounding condition. Allowed adaptation methods: Fixed Ratio, Fixed Area and Calc. Ratio.
- **AutoN (AN)**: AutoN stands for \(1...n*Auto\). This means that the cell height is proportioned to an automatically sized \((Auto)\) cell height, i.e. \(N\) is a multiplication factor to calculate the final height of the element. Allowed adaptation methods: Free and Fixed Width.
- *N*: This method defines how to utilize the remaining space. *N* is a multiplication factor to calculate the final height of the element. Allowed adaptation methods: *Free* and *Fixed Width*.

Based on the introduced concepts, we emphasize the main motivation of our layout algorithm. We do not concentrate on the size of the elements, i.e. we do not ask their size or requested size, but we are interested in their adaptation method. This method defines their behavior during layout calculation. The algorithms of the approach are providing the answers for the following questions:

- Which cell containment type and cell height adaptation method pairs are feasible and which are invalid (contradicting) ones?
- What is the behavior method of a splitter component? Depending on the contained element what is the resulted adaptation method (e.g. *Free*, *Fixed Ratio*, or *Fixed Height*)?
- How to calculate the certain cell heights for a specific device?

All of the possible layout element combinations should be examined. In order to minimize the number of the combinations we normalize them and then we should handle only the differently behaving cases. The combinations and their possible normalizations are discussed in the next section.

### 3.2 Normalizing Layout Combinations and Calculating the Resulting Equations

In order to end with a manageable number of cases we consolidate the similarly behaving height definition methods and adaptation methods.

We assume that all horizontal splitter combinations can be composed, i.e. the layout element equations can be combined and the result is again an equation. This requires calculating the resulting equation for optional existing equation forms (e.g. $x+y=c$) and not only simple ones (e.g. $x=c$). The questions we are about to answer are the followings. What will be resulted by the possible combinations of the layout equations? Will we remain within the scope of the already introduced equation forms (Table 1) or we have to introduce new equation forms? If the combinations introduce new equation forms, what happens during the combination of these new equations? When will we reach (if we can reach) a closed list of the equation forms?

The above questions raise the topic of hierarchically applied horizontal splitter components. The approach has to know the resulting equation forms, because these forms should be applied during the combination of two or more horizontal splitter components. Form the algorithms point of view this represent a recursive calculation method.

The key motivation of the layout element composition is the following. A layout template can contain several layout elements, often they are hierarchically defined. In case of a horizontal splitter component we would like to replace it and the contained layout elements with a similarly behaving single element. The replacing element should have the same layout behavior as the original horizontal splitter. In this way we can efficiently handle the layout calculation of hierarchically defined compound layout templates, even if the hierarchy is deep. In the next section, Table 3 summarizes the possible composition of the layout elements related equations.

The consolidations of *N*, *AutoN*, *FixedH* (FH), *Fixed Area* (+), *Fixed Ratio* (X) and *Calc. Ratio* (C) adaptation methods are discussed in [11]. Here we summarize the results of the layout definition normalization rules. Figure 3 shows the possible containment type and height definition method pairs. We use slash (../..) as a separator to enumerate the different input and output cases of the layout normalization. For example $X/+ /C$.

```
<table>
<thead>
<tr>
<th>Containment</th>
<th>Height Method</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X / + / C / F</td>
<td>FHW</td>
<td>1</td>
</tr>
<tr>
<td>X / + / C</td>
<td>A</td>
<td>5 / 6 / 7-10</td>
</tr>
<tr>
<td>O / W</td>
<td>*</td>
<td>1 / 2</td>
</tr>
<tr>
<td>H</td>
<td>FH</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Figure 3. Summary of the normalization.

When the algorithm has to determine the resulting behavior method of a splitter component it is enough to contract the appropriate equations. There is no need to take into account the actual height definition methods. The height definition methods affect the validation (what type of primitive layout elements can be placed into certain cells), and the calculation of the final layout. This corresponds to our goal: based on the equation forms be able to determine the behavior of compound elements (splitter components).

The composition of layout elements is commutative (see next section). This means that the order of the elements in a splitter component is optional. This is also part of our motivation, because if we can prove this statement then the layout editor
should not take care about the order of certain layout elements, i.e. the layout elements can be optionally commute.

### 3.3 Algebraic Structure

In this section we discuss the algebraic structure related considerations of the splitter components. We handle layout equations as a Set of equation sets. Furthermore, we prove certain properties of this Set.

The elements of the Set are based on the equations describing the behavior of the layout elements. Actually we are interested in the form of the equations. The elements are set of equations (zero, one or two equations). The name of the Set is Horizontal Splitter Equation Set (HSE Set). The elements of the Set formed by our splitter component are provided in Table 2. The Set has 11 elements.

The Horizontal Splitter Equation Set is equipped with a single binary operation $M \times M \rightarrow M$. A binary operation is closed by definition, but no other axioms are imposed on the operation [12].

<table>
<thead>
<tr>
<th>Equation ID</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$y = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$-$</td>
</tr>
<tr>
<td>2</td>
<td>$x = c$</td>
</tr>
<tr>
<td>3</td>
<td>$y = c$</td>
</tr>
<tr>
<td>4</td>
<td>$x = c_1$</td>
</tr>
<tr>
<td>5</td>
<td>$y = c_2$</td>
</tr>
<tr>
<td>6</td>
<td>$x \ast y = c$</td>
</tr>
<tr>
<td>7</td>
<td>$y = c_1 \ast x + c_2$</td>
</tr>
<tr>
<td>8</td>
<td>$y = c_1 \ast x + c_2$</td>
</tr>
<tr>
<td>9</td>
<td>$y = c_1 \ast x + c_2 + \frac{1}{x} + c_3$</td>
</tr>
<tr>
<td>10</td>
<td>$y = c_1 \ast x + c_2 + \frac{1}{x} + c_3$</td>
</tr>
</tbody>
</table>

The binary operation of the HSE Set is the contraction operation $(\ast)$. The contraction simplifies two sets of equations with the help of a third equation set. For the horizontal splitter component, the third equation set is always the following:

$$x_3 = x_2 = x_1$$

$$y_3 = y_2 + y_1$$

The result of the operation is set of equations (an element of the HSE Set). The operational table of the contraction operation is summarized by the Table 3.

**Proposition 1.** The contraction operation is commutative for the Horizontal Splitter.

**Proof.** The operation has two operands (two sets of equations). The variables of the first operand are $x_1$ and $y_1$. The variables of the second operand are $x_2$ and $y_2$. Regarding to the contraction operation, the order of the two operands affects only the indexes of the variables. In the third set of equations (the hard-wired helper equations) the $y$ variables are added ($y_2 + y_1$). The addition is a commutative operation, therefore, the indexes can be commuted. This means that the order of the original set of equations is optional.

**Consequence:** Commutability means that the order of the layout elements in a horizontal splitter component does not affect the resulting behavior of the splitter.

**Proposition 2.** The contraction operation can be applied for Vertical Splitter component and the contraction operation is commutative for it.

**Proof.** The vertical splitter is similar to the horizontal splitter with the difference that the elements are placed one next to another. Therefore, the statement follows from the Proposition 1.

**Proposition 3.** The contraction operation cannot be applied for the composition of horizontal and vertical splitter components.

**Proof.** The composition of horizontal and vertical splitter components can result new forms of equations. This contradicts the definition of the contraction operation: the binary operation of the HSE Set is closed.

**Proposition 4.** The HSE Set and the contraction operation are associative.

**Proof.** The HSE Set is associative, if the contraction operation $(\ast)$ is an associative operation. The contraction operation is associative, if $\forall x, y, z \in M$, where $M$ is the HSE Set, $(x + y) + z = x + (y + z)$. Based on the operational table (Table 3), we can deduce all of the cases. For example:

$$(1 + 2) + 3 = 1 + (2 + 3) \quad 2 + 3 = 1 + 2 \quad 2 = 2$$

Because of the space restrictions, we do not deduce all of the combinations.

**Proposition 5.** The contraction operation is idempotent.

**Proof.** Based on the operational table (Table 3), the contraction operation is idempotent: $A \ast A = A$. There are two forbidden cases ($2 \ast 2$, and $4 \ast 4$), where the layout algorithm would result inconsistency.

The identity element (or neutral element) of the HSE Set is a special type of element of a set with
respect to a binary operation on that set. It leaves other elements unchanged when combined with them. By default we do not have an identity element (Table 1) but we defined it for the HSE Set \((y = 0)\).

<table>
<thead>
<tr>
<th>Eq. ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

4 Conclusion

In this paper we have introduced the horizontal splitter algorithm related considerations of our content-driven template based layout system. The approach provides a language, a tool set and the related layout algorithms for online magazine editors.

The discussed approach facilitates to calculate the adaptation method of complex elements. Therefore, we can build hierarchical templates, where every cell can contain not only a single element but a whole template as well. The layout algorithm calculates the resulting adaptation method on every level of the template hierarchy, thus finally we get the resulting adaptation method of the full compound component as well.

We have discussed the basics of the Horizontal Splitter Component, including the adaptation method (resizing mode) of the layout elements, and the approach provided height calculation methods. We have introduced the adaptation methods related equation forms.

We have underpinned the key motivations of the splitter components and the layout element composition provided by them. In a regular online magazine template, there are hierarchically defined layouts. To provide a practical solution we replace compound elements (splitter components) and the contained layout elements with similarly behaving single elements. We ensure that the replacing elements have the same layout behavior as the original splitter components. In this way we can efficiently manage the layout calculation of hierarchically defined compound layout templates.

We have introduced the algebraic structure of the provided approach. We have defined the Horizontal Splitter Equation Set and we have proved certain properties of the Set.

Based on the actual surveys [1] [2] the relevance of tablet devices and the number of the device owners is continuously growing. We believe that the relevance of layout solutions providing automatic adaptation, including the presented one, is quite relevant.

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

References: