Spatial Graph Grammars for Web Information Transformation

Mei Kang QIU   Guang Lei SONG   Jun KONG   Kang ZHANG

Department of Computer Science, University of Texas at Dallas
Richardson, Texas 75083-0688 USA
kzhang@utdallas.edu

Abstract

This paper presents an approach to spatial specifications for Web information transformation[1]. Extended from the Reserved Graph Grammar (RGG), a Spatial Graph Grammar (SGG) is proposed. The paper illustrates a detailed example that applies the SGG to transform a XML Web document to a WML structure for the display on mobile devices. The SGG formalism is general enough for a wide range of applications such as multimedia interfaces, electronic publishing and XML document conversion.

1. Introduction

With the rapid development of the Internet technology, more graphs and media-rich contents are delivered on the Web. There are various kinds of viewing conditions when surfing the Internet, such as varying screen sizes, style preferences, and different device capabilities. In order to adapt to different clients, we need an executable mechanism to automatically transform the presentation layout. There are increasing demands for the ability of automatic transformation and visualization to meet the client side requirements.

Visual programming languages (VPLs) are capable of expressing and communicating structural information more effectively than textual languages. As the underlying theory of VPLs, graph grammars provide a sound and well-established foundation in defining logic relations among the language components [13]. The recently developed Reserved Graph Grammar (RGG) formalism is powerful in expressing various types of diagrams, with a parsing complexity of polynomial time under a non-ambiguous condition [17][18]. Zhang et al. presents a visual approach to XML document design and transformation, which uses RGG to define the XML syntax and specify the transformation among different XML formats [20].

Although RGGs are expressive and efficient, they cannot be used in document layout transformations without support for spatial specifications. This paper presents a spatial extension to the RGG, called the *spatial graph grammar* (SGG), and illustrates its application in Web information transformation. The process of Web transformation is illustrated in Figure 1, where there are

two types of input documents: XML/XSL or XHTML specified, or graphs representing envisaged document structures. We first obtain the tree structure of the input Web document, which is then transformed to a host graph to be processed by a spatial graph grammar. The SGG is defined to transform the host graph to the desired presentation layout. The layout graph is finally automatically translated into a WML (Wireless Markup Language) document for displaying on mobile devices or XML/XSL/XHTML document for desktop displaying. This paper presents our approach to this process but omitting the conversion part for textual or graphical documents to their tree structures (i.e. the white boxes in Figure 1). WML [16] is a markup language based on XML, and is intended for use in specifying the content and user interface for narrowband devices, including cellular phones and pagers, and more recently PDAs.
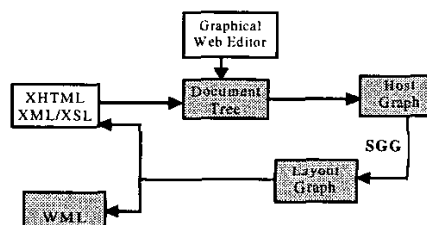

Figure 1 Web page transformation process

Section 2 briefly introduces the RGG formalism. Section 3 describes the Spatial Graph Grammar, in particular, the notations for spatial specifications and their application in layout transformation. Sections 4, 5, and 6 walk through an example to illustrate the application of the SGG to the transformation of a XML/XSL or XHTML tree into a WML document. Section 7 discusses related works, followed by the conclusion and future work in Section 8.

2. Reserved Graph Grammars

Most graph grammars consist of a set of rewriting rules called *productions* as shown in Figure 2. Each production consists of two sub-graphs, called *left graph* and *right graph*. Graph transformation is a sequence of applications of productions. Applications are classified into L-applications and R-applications. An L-application (or R-application) is to replace a sub-graph (called a *redex*) in the host graph, which is isomorphic to the left (or right) graph

of a production, with the right (or left) graph of a production. One of the most difficult problems with graph transformation systems is to decide which applications are allowed and which are disallowed. Even for the most restricted classes of graph grammars, the membership problem is NP-hard [12].
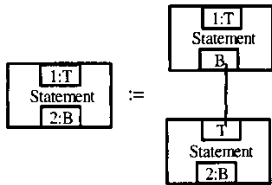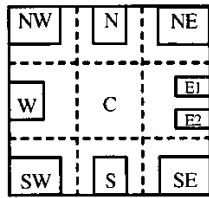

Figure 2 A graph grammar production

Figure 3 Node structure with directions

The Reserved Graph Grammar (RGG) combines the approaches of embedding rules and context elements to solve the embedding problem. A RGG is a collection of productions represented as labeled graphs. It is context-sensitive and its right and left graphs can have an arbitrary number of nodes and edges. The grammar uses an enhanced node structure with a marking mechanism in its graph representation, as shown in Figure 2. The outer rectangle of a node is called a *super-vertex* and each small rectangle embedded inside a super-vertex is called a *vertex*. Semantically, there is no difference between a vertex and a super-vertex. It is this structure with the marking mechanism that makes a RGG effective in specifying a wide range of visual languages and efficient in parsing the visual programs in such languages [18].

The RGG handles the context information with simple embedding rules and is sufficiently expressive to handle complicated programs. In order to identify any graph elements that should be reserved during transformation, we simply mark each involved vertex in a participating production by prefixing its label with a unique integer. The purpose of marking a vertex is to preserve the context and to avoid ambiguities. If a super-vertex or a vertex is marked, it will retain its outgoing edges connected to the vertices outside the redex after the application of a production.

3. Spatial Notations and Applications

The Spatial Graph Grammar (SGG) is an enhanced RGG with notations for spatial specifications. This section presents the SGG's three sets of notations and their uses in some typical Web transformations.

3.1. Spatial Specifications

The spatial graph grammar consists of three categories of specifications: direction, topology, and alignment, as described below.

3.1.1. Direction specification

In order to specify the direction between two nodes, one of the most important spatial relationships, the SGG defines a node's super-vertex as a grid of three rows by three columns, occupying nine areas as shown in Figure 3. The central area represents the super-vertex itself. Surrounding the center area, the eight areas represent eight directions: N (North), NE (Northeast), E (East), SE (Southeast), S (South), SW (Southwest), W (West), NW (Northwest), in clockwise direction. Each of these directions indicates the relative position of a node connected to the current node.

Each of the eight areas surrounding the central area may contain more than one vertex. The nodes connected to the vertices in the same area are in the same direction. For instance, in Figure 3, the East area of the node has two vertices, namely E1 and E2, indicating that the nodes connected to E1 or E2 are both on the East side of the current node.

3.1.2. Topology specification

We can generally define four topological relationships between two nodes: *non-overlapping, overlapping, touching,* and *containing.* Assume that D_x is the set of all the points on an object x, and B_x $(\subseteq D_x)$ is the boundary point set of x. Considering a primary object a and a reference object b and $D_a \cap D_b = C$, four topological relationships are defined as the following:

- a is *non-overlapping* with b iff $C = NULL$;
- a is *overlapping* with b iff $C \neq NULL$, and further
  - a is *touching* with b iff $C \subseteq (B_a \cap B_b)$; or
  - a is *containing* b iff $D_b \subseteq D_a$.

Using a rectangle to represent an object, Figure 4 shows the four types of topological relationships. *Non-overlapping* indicates that there is no common point on both involved objects. *Overlapping* means that there are common points between the two objects. It is represented by dotted lines on the boundary of the

(a) Notation	(b) Illustration
Non-overlapping	
Overlapping	
Touching	
Containing	

Figure 4 Topological relations

overlapped area. We define *touching* and *containing* as two special cases of overlapping. If common points only exist on the boundaries of two objects, the objects are *touching* with each other. The touched part is represented by a dotted
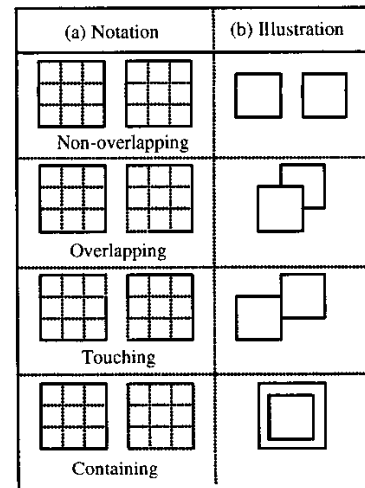
line. *Containing* means that all the points on one object belong to the other. In Figure 4, the boundary of an object is totally dotted, indicating that the object is contained in the other object.

3.1.3. Alignment specification

Two objects may be aligned vertically or horizontally. Figure 5 illustrates three alignment relations in the horizontal direction: the alignment on the top, bottom or the center of the involved objects.
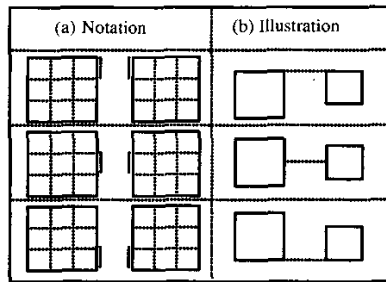

Figure 5 Alignment relations in horizontal direction

We use a bold line segment to represent the part that needs to be aligned at the same horizontal level. The alignment relations in vertical direction are similarly defined.

3.2. Layout Transformations

This subsection relates the above spatial specifications to some typical examples of Web transformations. We will focus on transforming Web pages for small-screen display on mobile devices, such as PDAs. To reduce the presentation space while maintaining the original contents, the simplest method is *linear scaling* (or normal zooming), which does not usually produce satisfactory results. A more elaborate technique is *differential scaling* [8], in which different components of a document are scaled differently. For example, as illustrated in Figure 6, each white space is compressed, while the box sizes are unchanged. We will discuss three types of transformations, i.e. distance, zooming and location transformations, in the context of spatial graph grammars.

3.2.1. Distance transformations

To represent a distance change between two nodes, we postfix a "+" to the vertex label to indicate a distance increase, "-" to indicate a distance decrease, and blank to represent no distance change. For example, the transformation in Figure 6 can be specified as in Figure 7. There is a postfix "-" in vertices A, B and C, implying a decreased distance between the three nodes.
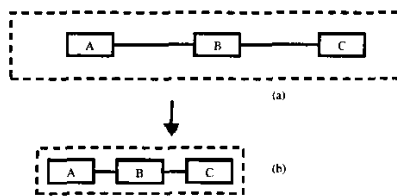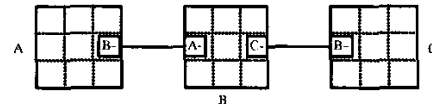

Figure 6 Differential scaling

Figure 7 Grammatical representation of distance transformation for differential scaling in Figure 6

When transforming a Web presentation to suit a PDA screen, we may use a viewing technique known as *semantic zooming* [8]. For varying interests, the presentation may show a particular area and/or level of details. Semantic zooming allows the viewer to zoom in hierarchically, while adapting the layout level of each individual component or group of components to the available screen size or to the viewer's preference. For example, we may need to enlarge one part, in which the user is particularly interested, while compressing unrelated parts, as illustrated in Figure 8. We need to look into the detail of object A first (since a mobile device screen cannot display all the details in one page, so we may view the details of A and B separately. Figure 9 depicts the combined use of distance, zooming, and location transformations (see below) to achieve this effect.
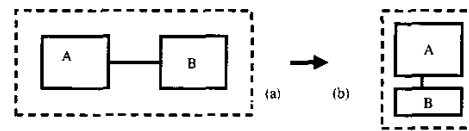

Figure 8 Semantic zooming

3.2.2. Zooming transformations

Transforming a Web page from the desktop presentation to the PDA presentation may involve many size changes. To represent the change of a node, we use "+" in the node's center box to indicate that the node will zoom in (becoming larger) in the transformation, "-" for zoom out (smaller), and blank for unchanged size.
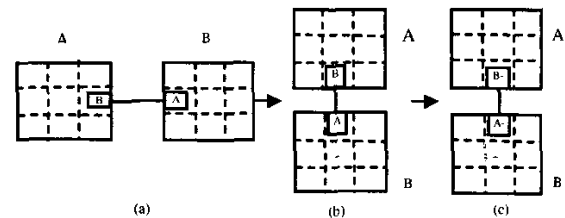
Figure 9 Grammatical transformations achieving semantic zooming

3.2.3. Location transformations

When transforming the layout of a Web presentation, not only may the distance between two objects and the size of an object change, but also the relative positions between the two objects may be changed. As shown in Figures 8(a) and 9 (a), originally node B is on the right of node A. After transformation, as in Figures 8(b) and 9(b), node B is at the

bottom of node A, thus the locations of vertices A and B have also changed.

4. A Web Example

In order to illustrate our approach, we use a popular page in Figure 10 as an example. Our graph grammar based approach is able to adjust the appearance intelligently to different displaying environments. The approach has two major advantages. First, a graphical transformation tool can be automatically generated by a visual language generator, such as VisPro [19]. Second, the generated transformation tool can be used by novice users who have no computing knowledge. We will transform this Web page into the WML format to be displayed on mobile devices. The XML description for the above Web page is as the following:

Figure 10 The original NASA home page

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<page>
    <section1>
        <block1>
            <Logo>
                <pic>
                    <id> nasa </id>
                    <source> ./images/nasa.bmp </source>
                </pic>
            </Logo>
            <text>
                02.01.03 Building Planets in Cyberspace
            </text>
        </block1>
        <block2>
            <theme>
                <pic>
                    <id> shuttle </id>
                    <source> ./images/shuttle.bmp </source>
                </pic>
            </theme>
            <button>
                <link>
                    <pic>
                        <id> missions </id>
                        <source> ./images/missions.gif </source>
                    </pic>
                    <href>
                            http://www.nasa.gov/missions/current/
                    </href>
                </link>
                ......
            </button>
        </block2>
    </section1>
    <section2>
        <pic>
            <id> improve life </id>
            <source> ./images/improvelife.bmp </source>
        </pic>
        ...
    </section2>
</page>
```


Figure 11 Resulting presentation as four pages on a PDA

Assume the desirable outcome as illustrated in Figure 11. We divide the original Web page into four small pages based on the four images. Each page contains three parts: the top part contains date and title (tagged "Text") and NASA logo ("Logo"), the middle part is an image ("Theme" or "Picture") and the bottom part contains three hyperlinks ("Link").

The output tree structure is translated into a WML document. Each page or a single interaction between a user agent and a user is known as a *card*. The beauty of this design is that multiple screens can be downloaded to a client in a single retrieval and vice versa. Our task is simply to transform the XML description into several cards, each to be displayed as a PDA page. The transformation from the XML structure to the WML PDA pages demonstrates the power of the SGG.

Following is part of the WML document for the PDA presentation in Figure 11, where "card" represents a separate page:

```
<wml>
    <card id="section1" Title="nasa">
        <p>
            <img src="./images/nasa.jpg" alt="earth"/>
            02.01.03 Building Planets in Cyberspace
        </p>
        <p>
            <img src="./images/shuttle.bmp" alt="shuttle"/>
        </p>
        <p>
            <img src="./images/missions.bmp" alt="shuttle"/>
            <img src="./images/multimedia.bmp" alt="shuttle"/>
            <img src="./images/events.bmp" alt="shuttle"/>
        </p>
```

```
</card>

<card id="improve" Title="improve life">
    <p>
        <img src="./images/nasa.jpg" alt="earth"/>
        02.01.03 Building Planets in Cyberspace
    </p>
    <p>
        <img src="./images/improvelife.bmp" alt="improve life"/>
    </p>
    <p>
        <img src="./images/missions.bmp" alt="shuttle"/>
        <img src="./images/multimedia.bmp" alt="shuttle"/>
        <img src="./images/events.bmp" alt="shuttle"/>
    </p>
</card>

</wml>
```

5. Structural Transformation

We first analyze the structure of this Web page. Each Web page has a layout constructed by many objects. In our Spatial Graph Grammar, each object is presented by a node. If we consider all the relationships between every pair of nodes, there will be $O(n^2)$ relationships for a total of n nodes. When n is large enough, the number of relationships will become prohibitively large. A Web page in XML is a tree structure whose elements can be grouped in a hierarchy. We can obtain the tree structure of the Web page from the XML description as shown in Figure 12. We adopt a hierarchical approach by grouping the nodes. A large group is further divided into smaller groups until they will be conveniently processed. Thus we obtain a hierarchy of the Web page objects. The tree structure will greatly reduce the total number of relationships among the objects of a Web page. To convert the tree to a more structured arrangement suitable for transformation, we need to introduce the concepts of logical nodes and grouping.
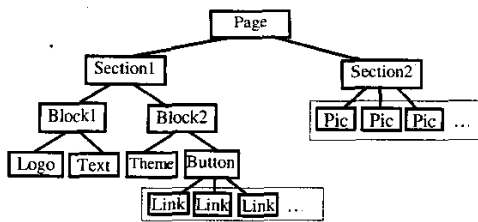

Figure 12 Tree structure of the Web page

The tree contains several logical nodes (LNs) such as *Page, Section1, Section2*, etc. *Page* is the root and contains two *Section* nodes. *Section1* contains two *Block* nodes. *Block2* contains *Theme* and *LN Button*. *Button* contains three *Link* nodes. *Section2* has a number of child nodes, called *Pictures* (three in the example). Such hierarchical relationships can be automatically derived from the XML document and used to generate the data structure in Figure 13.

We can add a *Group Header* if a group has many objects of a single type. A *Group Header* has a generic set of attributes applicable to the whole group. It inherits from its parents attributes such as vertices with spatial information. This will greatly improve the presentation

efficiency. Using the concepts of groups and *LNs*, we only consider spatial relations of a node with its parent, child and sibling (i.e. direct relatives). For example, since *LN Block1* and *Block2* are siblings, we will consider the relationship between *LN Block1* and *LN Block2*, but not the relationships between the members of *N(Block1)* (such as *Logo, Text*) and those of *N(Block2)* (such as *Theme, N(Button)*). Combining the spatial information from Figure 10 and above logical and hierarchical information from Figure 13, the host graph as in Figure 14(a) can be automatically generated to be processed by the spatial graph grammar. The application of the SGG generates the new layout structure in Figure 14(b) for PDA presentations, which will be explained in the next Section.
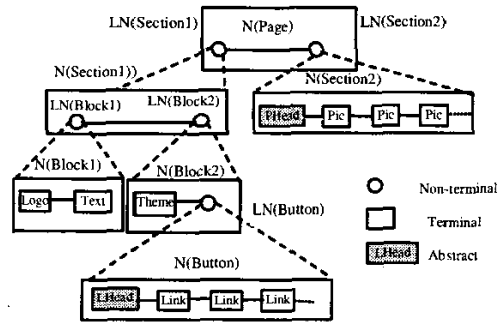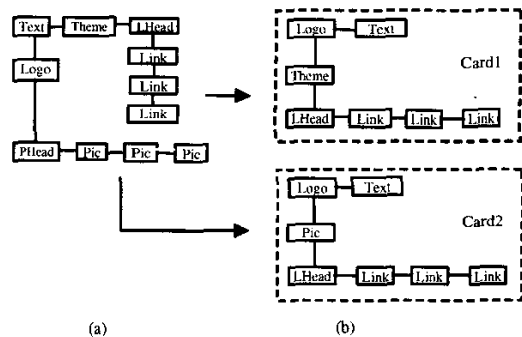

Figure 13 A hierarchical data structure

(a)   (b)

Figure 14 (a) Host graph of the original structure
(b) The resulting layout structure

6. Spatial Grammatical Specification

In order to perform the desired transformation, we define a set of productions as illustrated in Figure 15. There are two right graphs for some productions. The right graph not enclosed in a dashed box participates in syntactical parsing, and, together with the left graph, will be called a *syntax production* or simply S in the following description. The right graph enclosed in a dashed box is used for the layout transformation, and, together with the left graph, will be called a *layout production* or simply L.
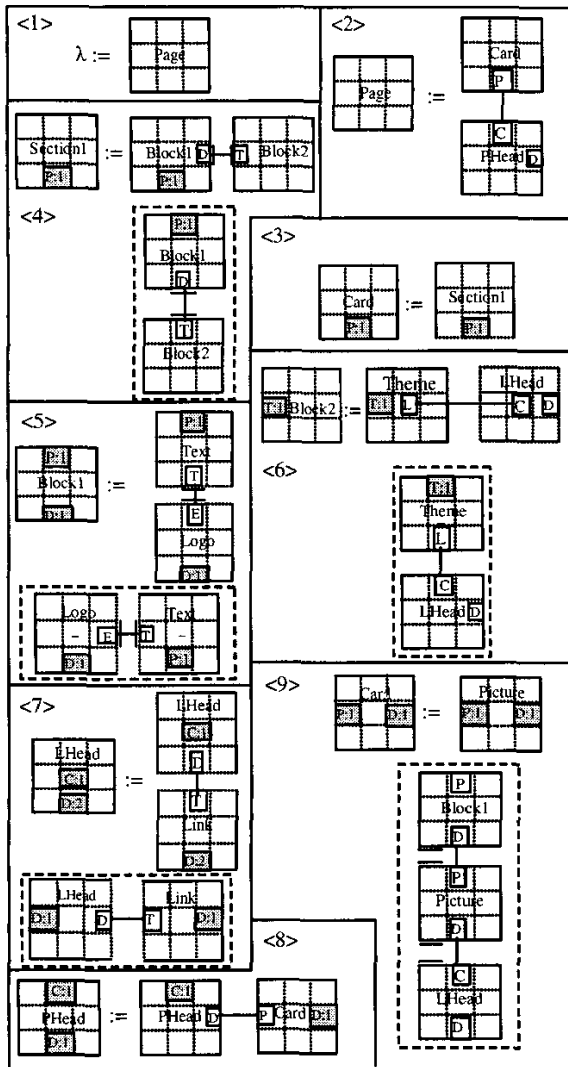
88

Figure 15 Productions for the transformation from the presentation in Figure 10 to the one in Figure 11

6.1. Syntax Productions.

Syntax production <1> (or simply S<1>) expresses the initial state. If a parsing eventually reaches the state λ (initial state), it is regarded as successful [17][18].

S<2> illustrates that such a page (NASA Homepage) consists of *Card* and *PHead*, and *Card* is on the top of *PHead*. S<3> abstracts a *Card* from *Section1*.

S<4> specifies that *Section1* contains two blocks, and *Block1* is side by side with *Block2*. The vertex in gray color in a node means that it is marked and will be reserved during parsing. For example, the vertex labeled *P* is marked, and will stay unchanged after parsing.

S<5> specifies that *Block1* consists of *Text* and *Logo*, and *Text* is directly on the top of *Logo*. The vertices labeled *P* and *D* are marked.

S<6> indicates that *Block2* includes *Theme* and *LHead*. *LHead* is a *Group Header* in the *Link* structure, and used to inherit the attributes from its parents. If the *Link* structure contains many members, using *LHead* will significantly improve the efficiency of the graphical presentation. To represent the containing relationship between *Theme* and *LHead*, we use dotted boundary in *LHead*, and connect the two nodes' central grids, which represent the super-vertices.

S<7> specifies that the *Link* structure consists of several terminal nodes of *Link*, stacked on top of each other.

S<8> and S<9> indicate that *Section2* includes several *Pictures*. In S<8>, *PHead* and *Card* can be reduced to *PHead*. *Card* is an intermediate node and can be abstracted from *Picture* (*Pic* for short) by using S<9>. We can apply S<9> continuously until no terminal node exists.

The R-application in the SGG is a parsing process, which in general consists of: selecting a production from the grammar and applying an R-application of the production to the host graph, and the process continues until no productions can be applied. If the host graph is transformed into an initial graph λ, the parsing process is successful and the host graph belongs to the language defined by the graph grammar. We first use S<9> and S<8> to reduce the *Picture* structure to *PHead*. S<7> is used to reduce the *Link* structure to *LHead*. S<6> is then used to reduce *LHead* and *Theme* to *Block2*, and S<5> to reduce *Logo* and *Text* to *Block1*. Then we use S<4> to obtain *Section1*. Finally, S<2> reduces *Card* and *PHead* to *Page* and S<1> to λ. Therefore the parsing process is successful.

6.2. Layout Productions

Based on the above syntax productions for parsing the original graph, we add several extended productions enclosed in dotted boxes called layout productions for transforming the presentation in Figure 10 to the one in Figure 11. The layout productions are thus an additive set to the syntax productions. Combining these two sets of productions, we can generate the desirable layout.

Layout production <4> (or simply L<4>) transforms *Block1* and *Block2* from the horizontal relationship to vertical relationship with *Block1* on top of *Block2*.

L<5> transforms *Text* and *Logo* from a vertically touching relationship to a horizontally touching relationship.

L<6> specifies how to transform two objects from a containing relationship to a vertical relationship. Before the transformation, *Theme* contains *LHead*. After the transformation, *Theme* is on the top of *LHead*.

L<7> transforms a sequence of *Links* from vertically touching relationships to horizontally touching relationships it is repeatedly applied.

In L<9>, when *Picture* with left and right vertices is matched, it is converted to a *Block1-Pic-LHead* structure,

89

whose three nodes are vertically aligned along the left edges.

We first parse the host graph in Figure 14(a) to λ. During parsing, a stack is used to record the sequence of the productions being used. Then from λ we retrieve the original parsing tree. At each step, the corresponding layout productions are popped from the stack to perform layout transformations. For example, when *Card* with its southern vertex is matched, S<3> is used to generate *Section1*. Then, we use L<4> to obtain a new layout in which *Block1* and *Block2* hold a vertical relationship. For *Block1*, L<5> is used to derive a horizontal relationship between *Logo* and *Text*. Using L<6>, *Theme* is moved to the top of *LPHead*. L<7> is used to obtain the horizontal *Link* structure. Now we obtain the first PDA page, represented as *Card1* in Figure 14(b). L<9> is used to expand *Card* to the *Block1-Pic-LHead* structure. *Logo* and *Text* are then generated using L<5> and the *Link* structure generated using L<7>. We therefore obtain the second PDA page (marked *Card2* in Figure 14(b)). The third and fourth pages, also of the *Card2* structure, are generated in the same fashion. The layout in Figure 14(b) can be automatically transformed to the final layout illustrated in Figure 11.

7. Related Work

Much research has been conducted in the areas of text summarization, graph compression, hierarchical and dynamical interfaces, and graph grammar transformations, as summarized below.

In text summarization, Buyukkokten *et al.* [4] presents important ideas of extracting semantics from the Web text yet greatly shortening the length of text. Usually, each text page is broken into a number of text units that can be hidden, partially displayed, fully visible, or summarized. Some research has been done on dynamic text presentation on mobile device using Rapid Serial Visual Presentation (RSVP) [10].

Our work is partly inspired by Six's work on graph compression [14] and Brandenburg's layout graph grammars [3]. Six *et al.* proposed a post-processing technique (after some major graph layout process), called *refinement*, which can significantly improve the quality of orthogonal drawings by reducing a graph's area, bends, crossings, and total edge length. Layout graph grammars [3] are context-free. With layout specifications, they can be used to draw limited classes of graphs.

Hierarchical menu structure has been used in user interface design based on spatial organization of information [6]. For a mobile device, content hierarchy or Hierarchical Atomic Navigation (HANd) has been proposed as a new philosophy to improve Web navigation on small displays [5]. The idea is to divide an original page into zones and make the navigator page as a reduced overview of the original page.

Dynamical interface constraints can be used to specify the desired presentation of a Web document through layout adaptation. Borning *et al.* [2] proposes a constraint-based system architecture in which both the author and the viewer can negotiate for the layout of a Web document. Marriott *et al.* [8] extends Scalable Vector Graphics (SVG) with constraint-based specification. Such an extension supports client-side adaptation of documents to different viewing conditions. These approaches rely on constraint solvers. There are also a number of systems and approaches for the presentation and dynamic authoring. Here, "authoring" refers to creating the content for any kind of presentation or document [9]. Dynamic authoring advocates that capture-based systems should support flexible hypertext structures generated by linking through interactive operations [Pim00]. Some user interface toolkits use the approach of recognition and mediation by constructing a library of reusable error correction and mediation tools, that can resolve ambiguity at the input event level [7].

Weitzman and Wittenburg applied a kind of graph grammar formalism, Relational Grammar, to the automatic presentation of multimedia documents [15]. Another related work is the RGG approach to XML document design and transformation [20]. Rather than using DTD and XSL, the Reserved Graph Grammar formalism is used to define the XML syntax and specify the transformations to other markup languages. Recently, we proposed a spatial extension of the RGG, and used it to transform adaptive multimedia presentations [21].

8. Conclusion and Future Work

This paper has presented the Spatial Graph Grammar (SGG) and demonstrated its application in the transformation Web presentations to suit small screen displays, such as a PDA screen. To graphically represent this kind of transformations, we have proposed the notation of grid nodes, and spatial relationships about direction, topology, and alignment. We have also presented three types of transformations for location, zooming and distance. A detailed example illustrates the transformation of a desktop Web page to the WML cards in three steps: transforming an XML file into a host graph automatically or using a Web graph; using the SGG to transform the layout of the host graph into a presentation suitable for multiple small pages; and finally, generating the equivalent WML document.

There are increasing demands for interactively changing the detail of specific parts of a Web page when viewing it. Such a mechanism is called *interactive semantic zooming* [8]. For example, two nodes, such as A and B, may expand their sizes alternatively. The viewing interface is changed dynamically. Temporal specifications determine the sequence of presentation. Allen presented some common temporal relations such as during, before, meet relations [1], which are potentially adaptable to Web info

transformations. We will investigate how to equip our spatial graph grammar with temporal specification capability and apply our SGG to the areas such as intelligent adaptation and dynamic interfaces.

References

[1]   F. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, 1983, Vol.26, No.11, pp.832 – 843.

[2]   A. Borning, R. K. Lin, K. Marriott, "Constraint-based Document Layout For The Web", *Multimedia system*, Vol.8, 2000, pp.177-189.

[3]   F. J. Brandenburg, "Layout Graph Grammars: the Placement Approach", *LNCS 532, Graph Grammars and Their Application to Computer Science*, Springer-Verlag, Berlin, 1991, pp. 144-156.

[4]   O. Buyukkokten, H. Garcia-Molina and A. Paepcke, "Text Summarization of Web Pages on Handheld Devices", *Proc. Workshop on Automatic Summarization 2001*, June 2001, Pittsburgh, PA.

[5]   F. J. González-Castaño, L. Anido-Rifón and E. Costa-Montenegro, "A New Transcoding Technique for PDA Browsers Based on Content Hierarchy", *Proc. Mobile HCI'2002 - 4th International Symposium*, LNCS 2411, Pisa, Italy, Sep. 18-20, 2002, pp.69-80.

[6]   G. Lorho, J. Hiipakka and J. Marila, "Structured Menu Presentation Using Spatial Sound Separation", *Proc. Mobile HCI'2002 - 4th International Symposium*, LNCS 2411, Pisa, Italy, Sep. 18-20, 2002, pp.419-424.

[7]   J. Mankoff, G. D. Abowd and S. E. Hudson, "OOPS: A Toolkit Supporting Mediation Techniques for Resolving Ambiguity in Recognition-Based Interfaces", *Computers and Graphics*, Vol.24, No.6, Dec. 2000, pp.819-834.

[8]   K. Marriott, B. Meyer, and L. Tardif, "Fast and Efficient Client-side Adaptivity for SVG", *Proc WWW'2002*, 2002, pp.496-507.

[9]   B. A. Myers, "Authoring Interactive Behaviors for Multimedia", *Proc. 9th NEC Research Symposium, Nara*, Japan, Aug.-Sep., 1998

[10] G. Öquist and M. Goldstein, "Toward an Improved Readability on Mobile Devices: Evaluating Adaptive Rapid Serial Visual Presentation", *Proc. Mobile HCI'2002 - 4th International Symposium*, Pisa, Italy, Sep. 18-20, 2002, pp. 255-240.

[11] M. Pimental, G. Abowd, and Y. Ishiguro, "Linking by Interacting: A Paradigm for Authoring Hypertext and Hypermedia", *CACM*, May 2000, pp. 39-48.

[12] G. Rozenberg and E. Welzl, "Boundary NLC Graph Grammars – Basic Definitions, Normal Forms, and Complexity", *Information and Control*, Vol.69, 1986, pp.136-167.

[13] G. Rozenberg (Ed.), *Handbook on Graph Grammars and Computing by Graph Transformation: Foundations*, Vol.1, World Scientific, 1997.

[14] J. M. Six, K. G. Kakoulis and I. G. Tollis, "Techniques for the Refinement of Orthogonal Graph Drawings", *Journal of Graph Algorithms and Applications*, Vol.4, No.3, 2000, pp.75-103.

[15] L. Weitzman and K. Wittenburg, "Automatic Presentation of Multimedia Documents using Relational Grammars", *Proc. ACM Multimedia'94*, San Francisco, USA, Oct. 15-20, 1994.

[16] Wireless Application Protocol Forum, Ltd, *Wireless Markup Language, Version 2.0*, Sep. 2001.

[17] D. Q. Zhang, "Generation of Visual Programming Languages", Ph.D. Thesis, Macquarie University, 1998.

[18] D. Q. Zhang, K. Zhang, and J. Cao, "A Context-Sensitive Graph Grammar Formalism for the Specification of Visual Languages", *The Computer Journal*, Vol. 44, No. 3, 2001, pp.187-200.

[19] K. Zhang, D-Q. Zhang, and J. Cao, "Design, Construction, and Application of a Generic Visual Language Generation Environment", *IEEE Transactions on Software Engineering*, Vol.27, No.4, April 2001, 289-307.

[20] K. Zhang, D-Q. Zhang, and Y. Deng, "A Visual Approach to XML Document Design and Transformation", *Proc. 2001 IEEE Symposium on Human-Centric Computing Languages and Environments*, Stresa, Italy, 5-7 Sep. 2001, pp.312-319.

[21] K. Zhang, J. Kong, and M. K. Qiu, "Multimedia Layout Adaptation Through Grammatical Specifications", *Technical Report*, UTDCS-08-03, University Of Texas at Dallas, 2003.