# Transforming Web Graphics for Mobile Devices

Meikang Qiu Kang Zhang

The University of Texas at Dallas

***Abstract***

*As the Web has been developed quickly, there are more and more demands for accessing Web applications from mobile devices such as PDAs. Since the screen size of a mobile device is very limited and varies from each other, we need a capability of transformation from Web graphs to suit mobile device displays. Such a transformation usually includes location change, differential scaling and semantic zooming. This paper presents a visual approach to the layout transformation of graphical objects from the Web to mobile device screens. The underlying theory of our approach is a context-sensitive graph grammar formalism. We use an enhanced node-edge diagram with a spatial partitioning mechanism to represent layout structures and support automatic transformations. Several examples of graph transformations are provided to demonstrate the conciseness and expressiveness of our context-sensitive graph grammar formalism. Theses extended context-sensitive graph grammars with spatial specifications can also be used in a wide range of applications such as multimedia interfaces, electronic publishing and XML document conversion.*

**Keywords**: Graph Transformation, Layout, Mobile Device, PDA, Visual Languages, Graph Grammars, Parsing.

## 1. Introduction

With the rapid development of the Internet technology, there are more and more graphs to be delivery on the Web. At the client side, there are various kinds of viewing conditions, such as varying screen sizes, style preferences, and different device capabilities. For example, consider the case of a user viewing a diagram representing an organizational structure on the Web, the fully expanded diagram is of considerable complexity and may be unsuitable for small displays [11]. Thus, if the diagram is to be viewed on the screen of a mobile device such as a PDA (Personal Digital Assistant), the original layout may not be appropriate. The small size and forms of display on PDAs introduce several new constraints for human computer interaction design. Further, the standard components of traditional graphical user interfaces, such as scrollbars, buttons and menus, which on a desktop only take a small percentage of the available screen estate, take up a considerable percentage of screen space on a PDA [5].

Of various systems that address the above problems, there are primarily two approaches: static and dynamic (interactive). This paper discusses only the static approach. In the static approach, the techniques include:

- **Alternative layout**. To adapt to the style of a PDA screen, there is usually a need to change the position of some object. This implies an alternative layout. For example, a vertical alignment presents a different visual perception and requires a different screen estate from a horizontal alignment, as shown in Figure 1.
- **Scaling**. The simplest solution to the problem of the limited screen size is *linear scaling* (or *normal zooming*), but this is often not the best way. The more elaborate technique is

*differential scaling*, in which different components of a document are scaled differently. *Differential scaling* is effective in compressing white spaces. For example, rather than performing just a linear scaling, each white space is compressed, while the box sizes are maintained [11], as illustrated in Figure 2.

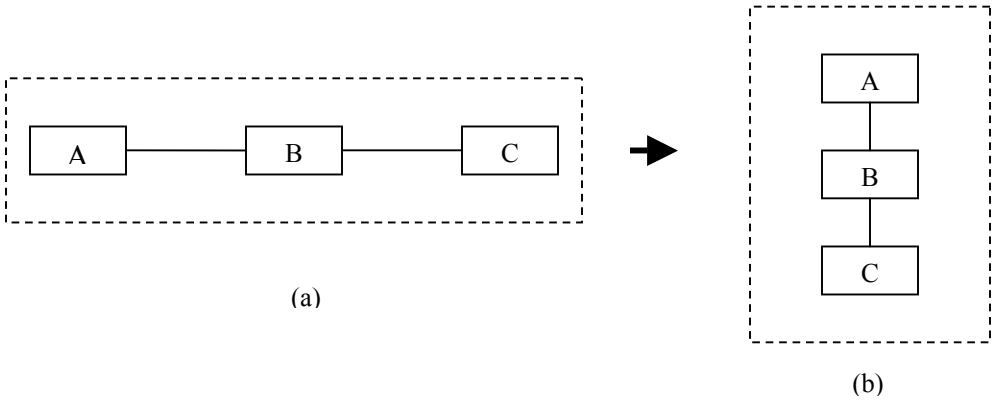



Figure 1 Alternative layout

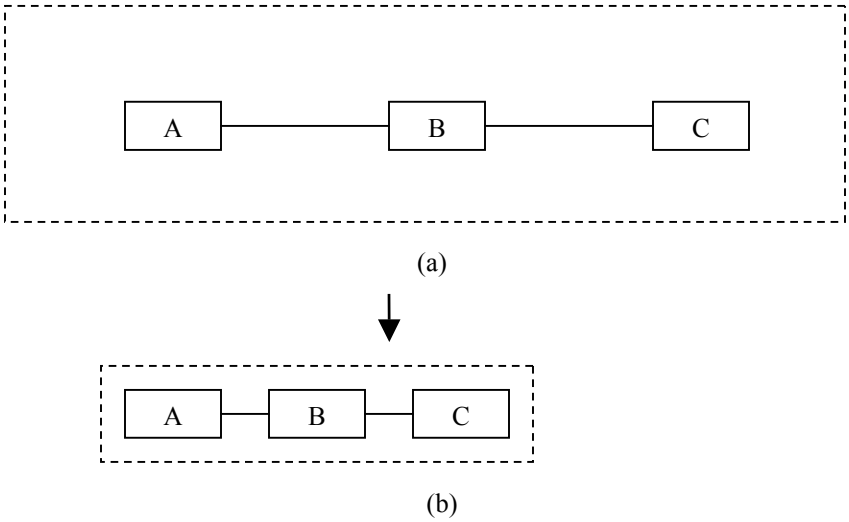



Figure 2 Differential scaling

- **Semantic zooming**. For varying interest in detail, an adapted layout may initially show one level of details. It 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. This viewing technique is called *semantic zooming* [11]. For example we may need to enlarge one part, in which the user is particularly interested, while compressing unrelated parts, as illustrated in Figure 3.

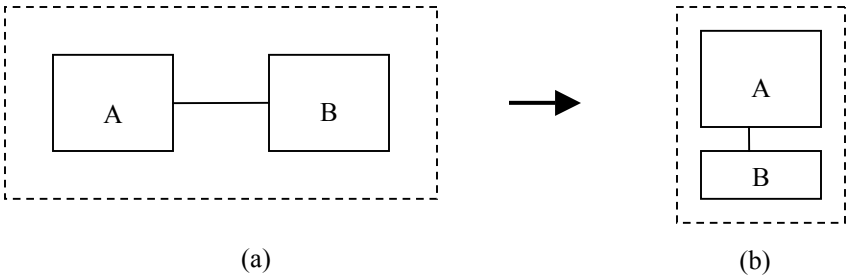



Figure 3 Semantic zooming and layout change

The central theme of this paper is to demonstrate how to use a graph grammar formalism to specify and support automatic transformations of graph layout, to reduce Web-based graphical images to suit mobile devices, such as PDAs, without losing the essence of the images.

The remaining part of the paper is organized as the following: we summarize related work in Section 2, introduce the grammar abstraction for graphical layout in Section 3. Section 4 discusses a context-sensitive graph grammar formalism, known as the Reserved Graph Grammar. Section 5 shows the Web information transformation mechanism. Section 6 concludes the paper, followed by Section 7 that proposes the future work.

# 2. Related Work

Much research has been conducted in the areas of text summarization and graph compression, as summarized below.

### *2.1. Text Summarization:*

Buyukkokten *et al*. [6] 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. There are mainly three methods for text summarization.

- **Page summarization**. This technique extracts an overview of a Web page by summarizing it.
- **Macro-level summarization**. This technique first partitions the page into "Semantic Textual Units" (STUs). STUs are page fragments such as paragraphs, lists, or ALT tags that describe images. It then uses fonts and other structural information to identify a hierarchy of STUs.
- **Micro-level summarization**. This technique explores in more detail a particular page.

Users can combine these three methods, using keywords, then most significant sentences, finally the entire STUs, to retrieve what they need.

### *2.2. Graph Compression*

Six *et al*. [15] 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 [3]. Our work is partly inspired by this work, which is helpful in our layout transformation from the Web to a mobile device

screen. For example, the techniques for handling the problems of stranded nodes, and poor placement of degree-two nodes are shown in Figures 4 and 5.

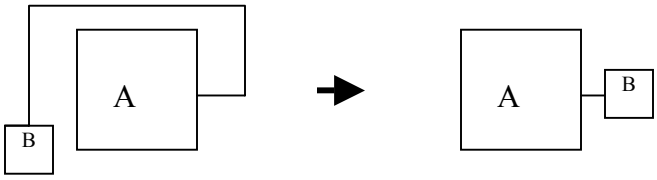



Figure 4 Stranded nodes problem

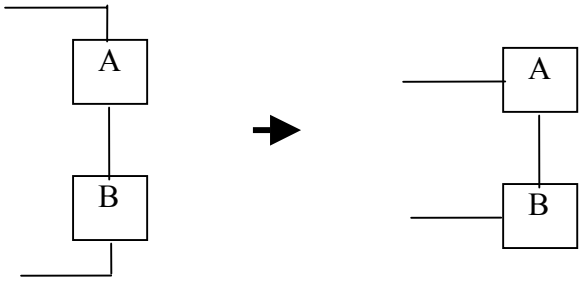



Figure 5 Poor placement of degree-two nodes

### *2.3. Dynamic Interfaces*

In a dynamic interface, the attributes of elements are defined in terms of other elements and attributes of the viewing environment:

- **Information links** indicate a (semantic) connection between two pieces of information, which can belong to different information domains [5].
- **Information view** is a collection of correlated objects displayed together to help the user to perform some activities on the objects [5].
- **Interactivity** allows the display to be dynamically adapted to the user's requirement. Borning *et al*. presents a system architecture in which both the author and the viewer can impose page layout constraints. The final appearance of a Web page is thus the result of negotiation between the author and the viewer [4][11].
- **Ubiquitous computing** includes three interaction themes: natural interfaces, context-aware applications, and automated capture and access [1].
- **Recognition and mediation.** Some user interface toolkit, consist of a library of reusable error correction, or mediation, can provide structured support for solving ambiguity at the input event level [10].
- **Dynamic authoring.** "Authoring" means creating the content for a given kind of presentation or document [12]. For authoring hypertext structures, it was advocated that capture-based

systems should support flexible hypertext structures generated by *linking by interacting* operations [13].

### *2.4. Graph Transformation:*

Research has been done for the graph grammar support for Web information transformations. To support automatic layout of flowcharts, recently Zhang *et al*. [21] presents a visual approach to XML document design and transformation, which uses RGG(Reserved Graph Grammar) [20] to define the XML syntax and to specify the transformation between different XML formats. In a graphical layout, maintaining a consistent view by automatically beautifying the display is desirable [16]. Zhang *et al*. presents an approach to combining RGG with constraint rules to support automatic layout of orthogonal graphs [22].

## 3. Spatial Abstraction for Graphical Layout

We will illustrate our ideas of graph transformations of Web graphics by several examples. As discussed above, Figure 1 illustrates an alternative layout. For a PDA screen, transforming a horizontal layout of graphical objects on a Web page to the vertical layout for a PDA screen is one of the simplest examples of transformations.

We perform the following steps to achieve the desired transformations. First, we translate the horizontal layout diagram into a graphical form whose syntax is suitable for our grammar interpretation. We will call such a graphical form a *node-edge diagram* [21]. A node is a two level structure: the super vertex is the bounding box of the node, the small rectangles embedded in a super vertex form the second level, called vertices. An edge is uniquely determined by two vertices in the involved nodes. There is no semantic difference between connecting to a vertex and connecting to a super vertex. All vertices should be labeled [20].

In order to represent the direction between two nodes, which is one of the most important relations in a graph layout, we define the super-vertex as a grid of three rows by three columns, occupying nine areas as shown in Figure 6. The central area represents the super-vertex itself. Surrounding the center area, the eight areas represent eight directions: N (North), S (South), E (East), W (West), NW (Northwest), NE (Northeast), SW (Southwest), SE (Southeast), in clockwise direction. Each of these directions indicates the relative position of the node connected to the current node.

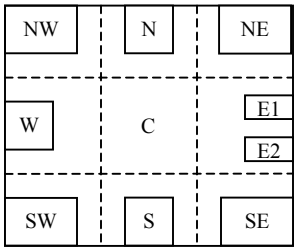



Figure 6 Node structure for location rule

Each of the eight areas surrounding the central area may include more than one vertex. The nodes connected to the vertices in the same area are in the same direction. For instance, in Figure 6, the East area of the node has two vertices, E1 and E2, which implies that the nodes connected to E1 and E2 are both on the right side of this node.

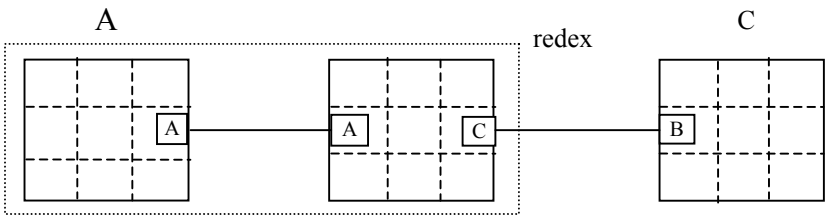



Figure 7(a) Node-edge representation of the graph in Figure 1(a)

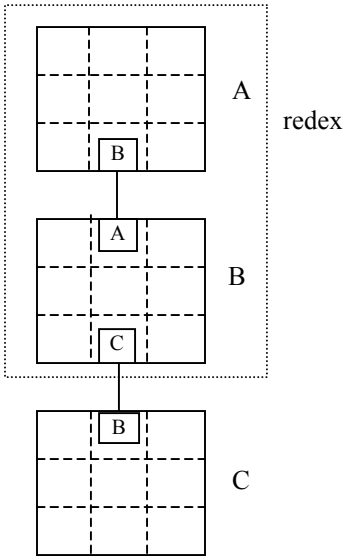



Figure 7(b) Node-edge representation of the graph in Figure 1(b)

We propose the following three sets of general graphical representation rules.

- **Location rules**. We divide the super vertex into three rows by three columns, totally nine areas. The central area represents the super vertex itself. Around the center area, the eight areas represent eight directions around the super vertex. For example, in Figure 7 (a), the vertex B in node A indicates that node B is on the right of node A. Similarly, in node B, the vertex A indicates that node A is on the left of node B, and vertex C indicates that node C is on the right of node B.
- **Zoom rules**. The transformation from a Web page to a PDA screen may involve many size changes. To represent the changes, we use "+" in the super vertex's center box to indicate that the super vertex will zoom in (becoming larger) in the transformation, "-" for zoom out (smaller), and blank for unchanged size.
- **Distance rules**. To represent a distance change between two nodes, we postfix a "+" to the vertex label to indicate a distance increase to the node it connects to, "-" to indicate a distance decrease, and blank to represent no distance change. When the distance between two nodes is zero, we consider the two nodes having a touch relationship. If two nodes have a negative distance between them, we consider the two nodes having an overlapping relationship. Overlapping relations can be further divided into partial overlapping, full overlapping, and containing relations.

For example, for the transformation in Figure 2, we use distance rules as shown in Figure 8.

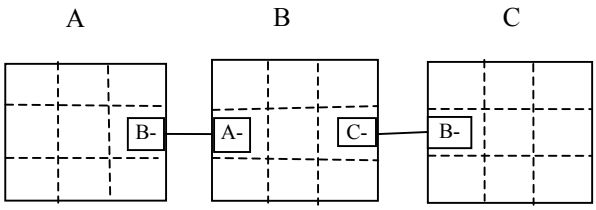



Figure 8 Graph representation of differential scaling,
Node-edge diagram for Figure 2(b)

As a layout transformation from Web page to mobile device screen usually involves these three types of changes, we will apply these three rules simultaneously. Below we will use an example to illustrate how to apply these transformation rules.

For the example transformation shown in Figure 3, we combine these three rules, and generate the result as in Figure 9. To explain the transformation in more details, we first translate the graph in Figure 3 to a node-edge diagram, as shown in Figure 9(a), and then we use the location rule to generate the graph in Figure 9 (b). Finally applying the zooming and distance rules generates the result as shown in Figure 9(c). Figure 10 illustrates another useful graphical information compression technique.

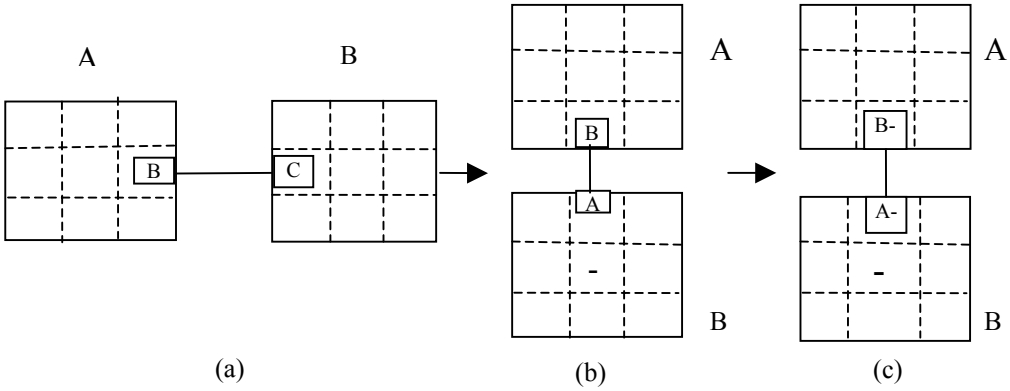



Figure 9 Application of semantic zooming and distance rules for the
transformation from Figures 3(a) to 3(c)

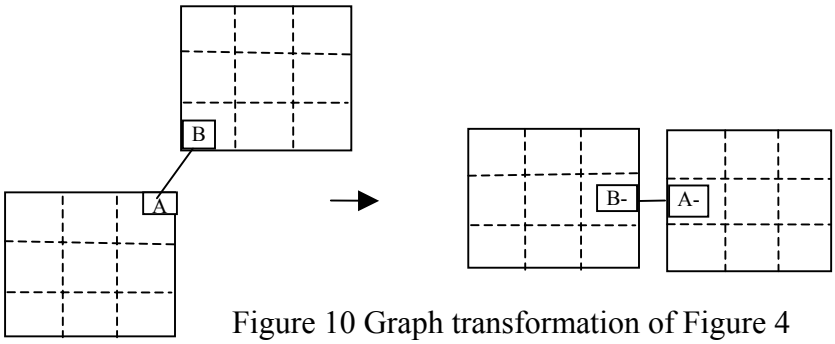



Figure 10 Graph transformation of Figure 4

# 4. A Graph Grammar

Information visualization often plays a crucial part in an application [8]. Visual programming aims at effectively improving the programming productivity by applying visual technologies to support program construction. Popular visual languages include UML, automata, Petri nets [19] etc. Compared with text, graphs can represent semantic and structural information more intuitively.

A graph grammar is made up of a set of rewriting rules called productions as shown in Figure 11. Each production consists of two sub-graphs, called left graph and right graph. The 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 in the host graph, which is isomorphic to the left (or right) sub-graph of a production, with the right (or left) sub-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 [14].

Another obstacle to restrict applications of graph grammars is that most proposed parsing algorithms [7][9][17] are based on context-free graph grammars. Many interesting graphs, however, cannot be specified by pure context-free grammars. Additional control mechanisms are necessary for context-sensitivity. In many applications, the logic structure of a graph is too complex to be defined by a context-free grammar.

Zhang *et al*. [20] proposed a context-sensitive graph grammar called RGG (Reserved Graph Grammar). It combines the approaches of embedding rules and context elements to solve the embedding problem. RGG is a collection of graph rewriting rules represented 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. It is this structure that makes an RGG effective in specifying a wide range of visual languages and efficient in parsing a certain class of visual languages. Although the time complexity of the parsing algorithm for a general RGG is exponential, parsing a RGG that satisfies a constraint can be done in polynomial time [20].

The RGG introduces context information with simple embedding rules and is thus sufficiently expressive to handle complicated programs. In order to identify any graph elements that should be reserved during the transformation process, we mark each isomorphic vertex in a production graph by painting it gray or 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 reserve its outgoing edges connected to vertices outside the replaced sub-graph (called a *redex*) in the application of a production.

Figure 7(a) shows a node-edge diagram for a Web graph, which can be called a *host graph*, we wish to transform it into the one in Figure 7(b), i.e. a node-edge diagram for PDA layout as a resulting graph. Figure 11 depicts the rewriting rule (production) for this required transformation.

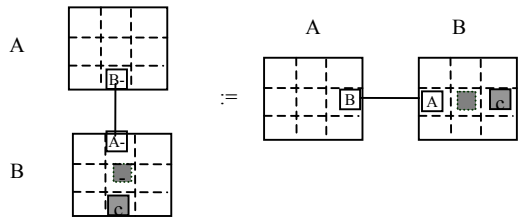



Figure 11 Marking mechanism in a production

Since in the host graph, the node B has two vertices A and C, we need to mark the super vertex (represented here as the center of a node) and vertex C by painting them gray. This means that during transformation, the edge connected to node B and vertex C will be reserved, so to satisfy the application requirement in Figure 7(a). Also, the size will be reduced (zoomed out).

The original RGG defines the logical relations among constructs of a graph. To support graphic layout, it may be extended to take constraint rules [16] represented by (x, y) coordinates [22]. Doing so however would greatly limit the intuitiveness and visualization power of the RGG. In our paper, we follow the general principle of context-sensitivity while increasing the expressiveness, and propose a set of graph transformation rules to enhance RGG with the capability of spatial (direction and distance) and size specifications. The enhanced RGG formalism is consistent with the original RGG but powerful enough to be used for in a wide range of applications, such as Web page design, graph layout, multimedia interface design and PCB design.

# 5. Web Information Transformation

This section will explain the application of spatial graph grammar rules by going through two examples graphs.

### *5.1. Transforming a Typical Web Drawing*

Assuming a Web graph layout as shown in Figure 12(a), we will transform it to the one in Figure 12(b), which may be displayed on a limited mobile device screen such as PDA.

The graph in Figure 12(a) contains six objects: Picture, Title, Contents, Notes, Menu and Link, which are displayed on a desktop computer screen whose horizontal dimension is usually larger than the vertical dimension. To transform the graph objects and their connections to suit a small screen, whose screen typically has a relatively larger vertical side and a smaller horizontal side. Assume that the central object, Picture, will maintain its original size, while other five objects will shrink their sizes to fit only the first letters of the object names. We now use our spatial graph grammar to specify this kind of transformation.

The complete set of graph rewriting rules (productions) for the required transformations are illustrated in Figure 13, that show the graph grammar defined for the type of graphs with dotted lines excluded, and the translation mechanism from the Web graph layout to mobile display such as a PDA screen layout when the sub-graphs (redex) with dotted lines are included. We refer to the former rules as the *grammar* (representing graph layout of Web drawings), and the later,

extended from the grammar, as the *translator* (from Web graph layout to mobile display layout) [21].

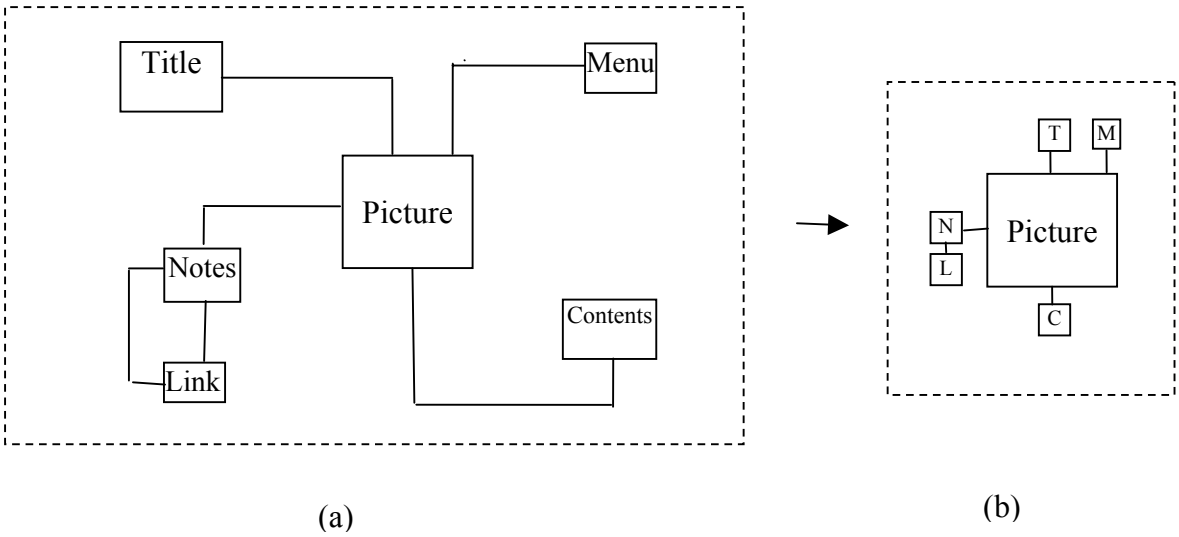



Figure 12 An example of Web graph transformation

Figure 13 shows seven rewriting rules, including 1 axiom rule (P1), 3 sets of representation rules (P2, P3, P4), and 3 Web graph compression rules (P5, P6, P7).

Productions <2> to <4> are three sets of general representation rules. They contain a number of variations, and are used as both the grammar and the translator.

Production <2> (P2) specifies a location change that can be applied to all the transformations. It is an instance of transformation from the horizontal relation to the vertical relation. At the right side, the two nodes are connected between vertex E of A and vertex W of node B. After transformation, the two nodes are connected between A's S vertex and B's N vertex. Since there are eight directions for each node, and the number of direction relations between any two nodes is 8*8 (=64). We describe the location changes as one production that can be mirrored or modified to suit other 63 direction relations.

Production <3 > (P3) represents a general rule on zooming. It is an instance that one node zooms in (become smaller) yet another node's size remains unchanged. The center of B has a "-"sign, which means that node B should be zoomed in (shrink). Similarly, we could use a blank or "+" sign to indicate that the node should remain unchanged in its size or be zoomed out (enlarged). We specify the zooming changes as one production which can be modified to suit other situations (including the change of a single node's size and the sizes of both nodes).

The marker (gray) at a super vertex or a vertex means that: after the transformation the edge connected to the outside of the redex will be preserved. This simple marking mechanism can avoid ambiguities [20].

Production <4> (P4) specifies distance changes. It is an instance of shortening distance between two nodes. In the left graph, if the two vertices connecting two nodes are both marked "-",the

distance between the two nodes should be decreased. Similarly, we use a blank and a "+" to indicate that the distance should remain unchanged or enlarge. We describe the distance changes in one production which can be modified to suit two other situations.

Production <5> to <7> are used to reduce the area size of Web-based graphical images (such as shrinking the layout structure and compressing white spaces), without losing the essence of the images.

Production <5> (P5) is used to delete a superfluous edge between two nodes. We define that only one edge can exist between two nodes. It is obvious that a long and folded edge between two nodes should be replaced by a short and straight edge. In Figure 14, we first apply P5 to the pair of nodes: *Notes* and *Link*, to delete the folded edge. We then apply P4 to shorten the distance

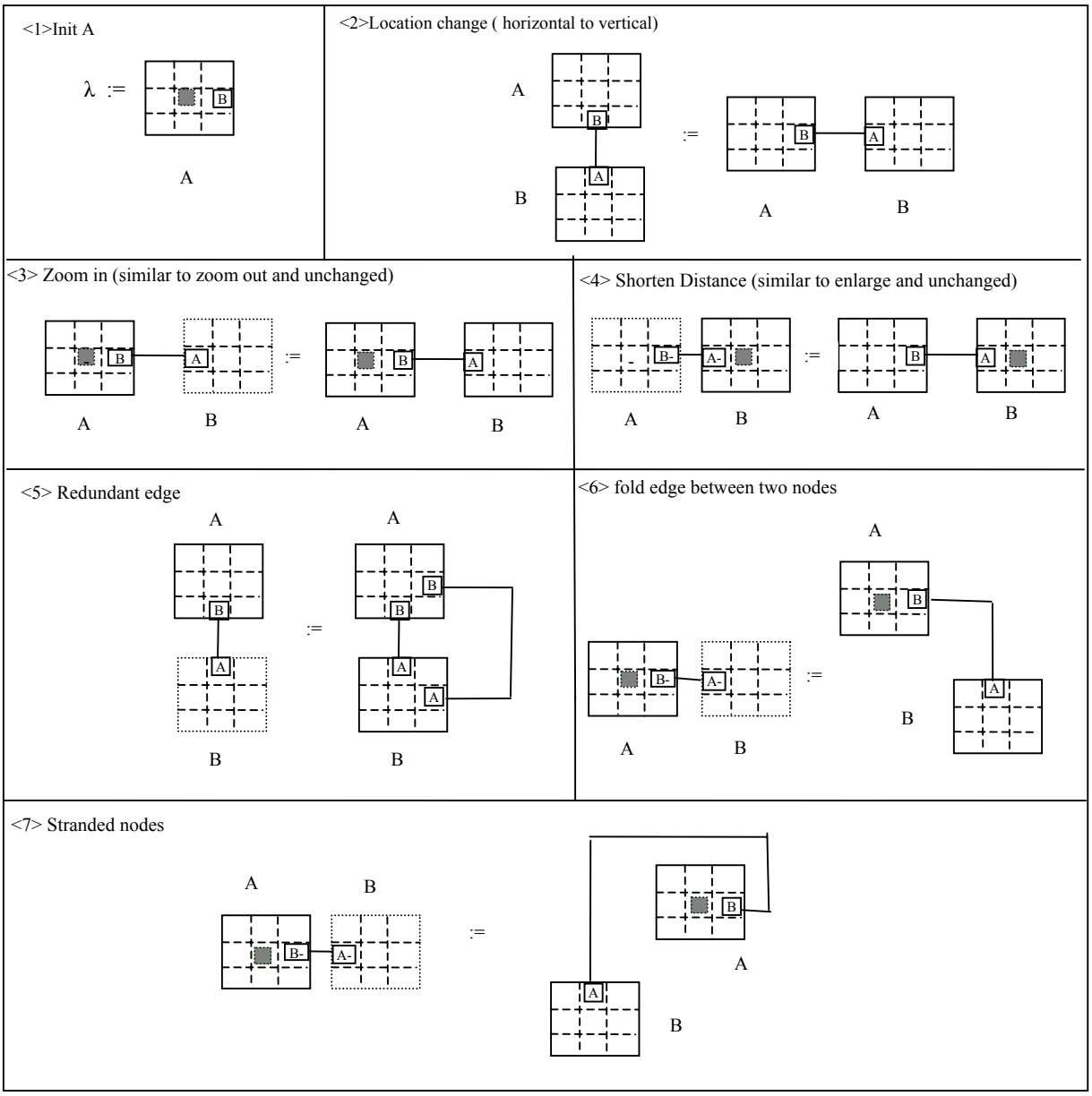



Figure 13 A graph grammar for Web graph layout compression

between them, and use P3 to zoom in both nodes. The number on the edge between *Notes* and *Link* stands for the productions that applied to them.

Production <6> (P6) describes the situation that there is a folded edge between two nodes. We will replace the folded edge by a straight edge. Productions P5, P2 and P4 describe the desired change, which include location changes and shortened distances. We apply P6 to the following pairs of nodes: *Title-Picture, Menu-Picture* and *Notes-Picture*.

Production <7> (P7) transforms a stranded node, which can be considered a special case of P6. In this case, the folded edge between two nodes is longer, and the two nodes are far away from each other. We apply P7 to the pair of nodes: *Contents-Picture*.

To avoid infinite applications of rules, we impose a constraint: each production can only be used once between any pair of nodes. We will explain this later in Figure 17.

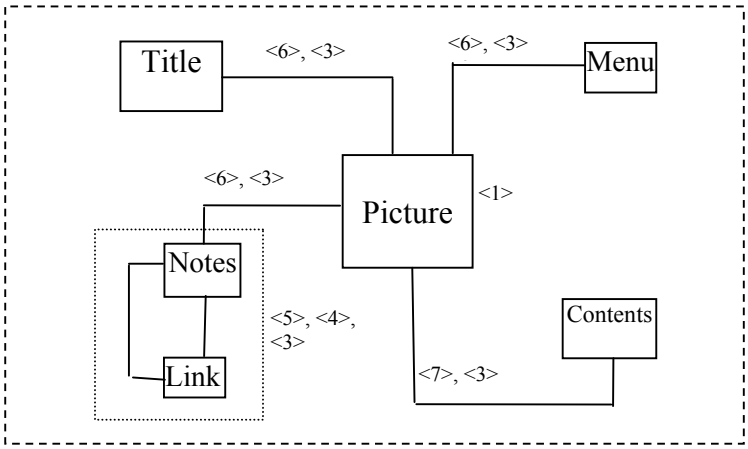
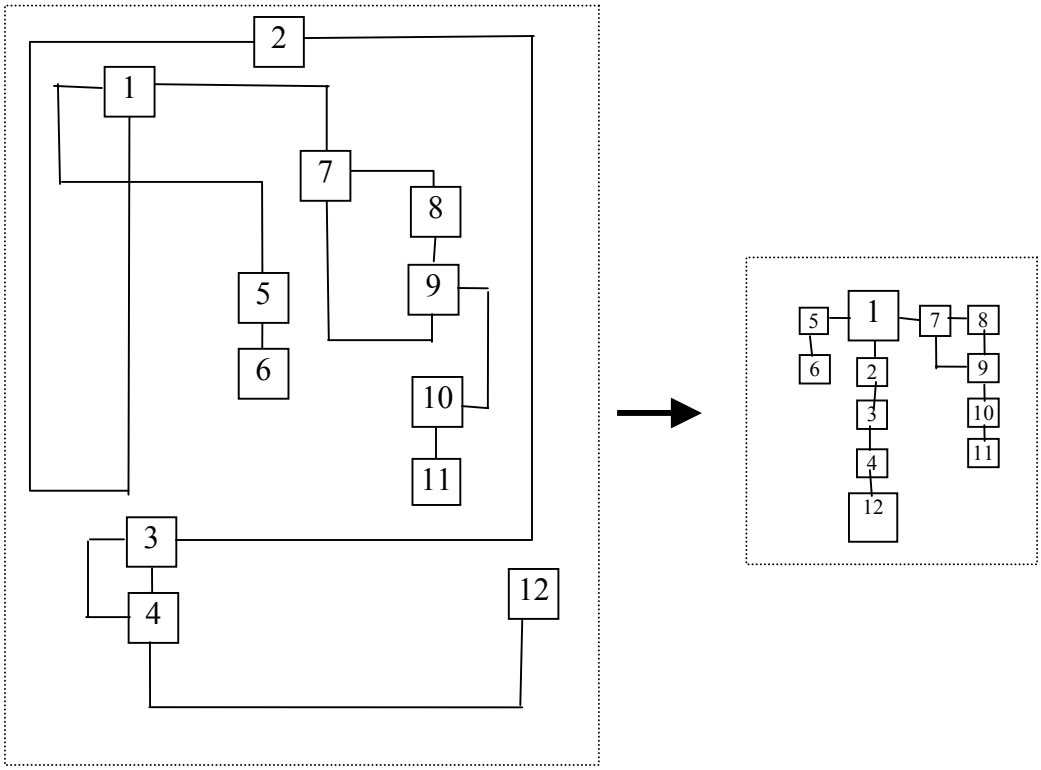



Figure 14 R-application of transformation formalism

The R-application in the RGG 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 layout, and the process continues until no productions can be applied. If the host graph is transformed into an initial graph, the parsing process is successful and the host graph belongs to the language defined by the graph grammar.

### *5.2. Transforming an Orthogonal Graph*

The content of Figure 15(a) is a typical graph that can be found in many areas such as a circuit layout and a Web drawing. Figure 15(b) shows a graph transformed from Figure 15(a), and becomes more compact with less white spaces than the latter. The steps of applying the transformation rules to change from Figure 15(a) to (b) are depicted in Figure 16.




(a) (b)

Figure 15 An example of graph layout transformation

Although we can apply the production rules to all the node pairs at the same time, we show the transformation process in four steps.

In Figure 16(a), we apply P7 and P3 to nodes 12 and 4 (for simplicity, we use (12, 4) to indicate the pair of nodes 12 and 4), P4 and P3 to node pairs (5,6), (10,11). We use a dashed rectangle to show the node pair to which a transformation is applied, and the number stands for the corresponding production. The result of the transformations becomes the graph in Figure 16(b). The later steps can be similarly interpreted.

Notice the sub-graph involving three nodes (7, 8, 9) showed in Figure 16(a). The transformations of the sub-graph are detailed in Figure 17. We first apply P6 to node pair (7, 8), transforming from Figure 17(a) to (b); then apply P7 to (7, 9), obtaining Figure 17(c). For the graph structure in Figure 17(c), we again apply P6 to (8, 9) to obtain Figure 17(d), which is a reflective of Figure 17(c). The area sizes of Figure 17(c) and (d) are the same. Further applying P6 will result in cyclic transformations that are apparently undesirable. Therefore, we impose a constraint that each production can be used only once between any pair of nodes. According this constraint, after applying P6 to Figure 17(c) to obtain Figure 17(d), the transformation process stops. Finally, we get the sub-graph shown in Figure 17(d). We may also apply P7 to node pair (7, 9) first, then P6 to (7, 8), finally apply P6 to (7, 9) to obtain the result.

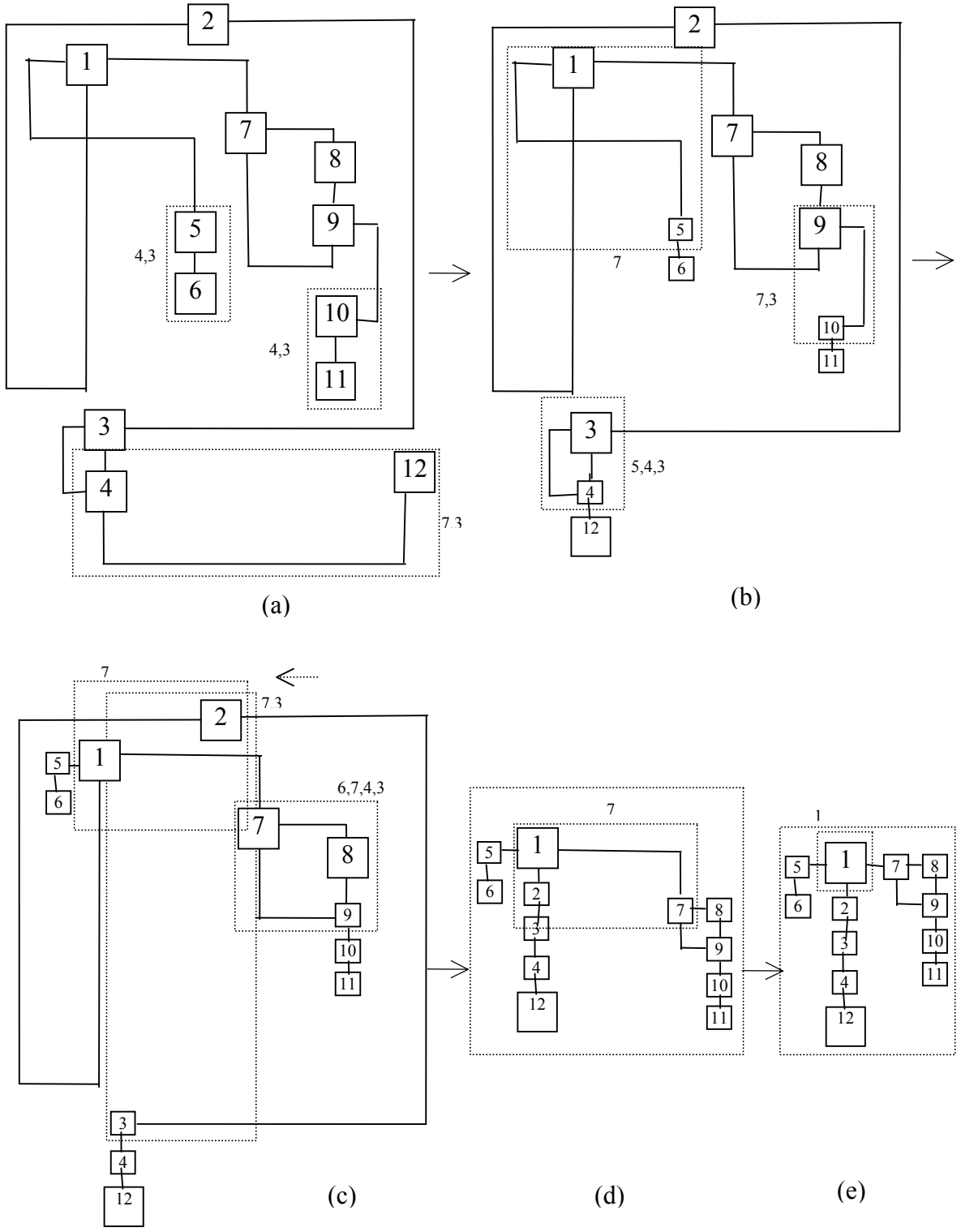



Figure 16 The snapshots of the transformation process

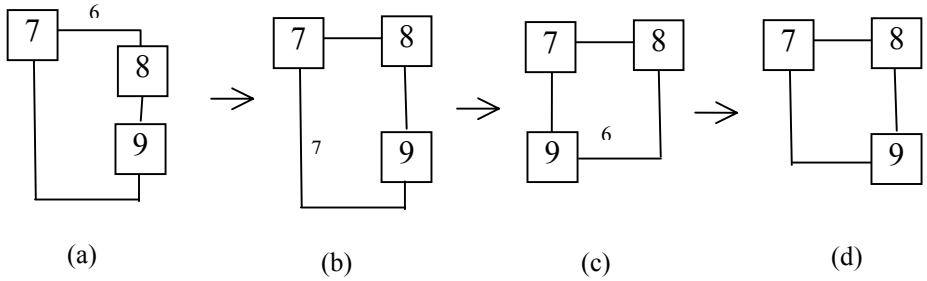



Figure 17 Possibly cyclic transformations

In a graph grammar, general difficulty in the process of parsing is to select an appropriate production when multiple choices exist, i.e. how to process ambiguities during parsing. Since the selection will affect the later parsing and final result, if the current path fails we must trace back to test other paths, which costs computational time. The RGG is equipped with a deterministic parsing algorithm, called *selection-free parsing algorithm* (SFPA), which only tries one parsing path [20]. Zhang has proved that the time complexity of SFPA is polynomial [18].

When defining transformation productions for graph layout where edges represent only geometric relations, we allow only one relation between any pair of nodes. Such relationships can be efficiently handled by the original RGG formalisms. Our graph grammar formalism with spatial specification capability is sufficiently expressive in specifying the multi-link relations. For example, when defining the graph structure of a multimedia document, apart from geometric relations, different media components should also have logical relationships and even temporal relationships. We will show the powerfulness of our grammar applying to this area in the future work.

# 6. Conclusion

This paper has presented the idea of applying graph grammars to the transformation of Web information to suit small displays, such as a PDA screen. Such transformations usually involve location change, differential scaling and semantic zooming. To graphically represent these three types of changes, we have proposed the notation of grid nodes, and three rules: location rule, zooming rule and distance rule. We use a context-sensitive graph grammar formalism with the spatial extension to explicitly and completely describe the syntax of a wide range of Web application layouts and transformation methods. The parsing algorithm of our extended reserved graph grammar (RGG) has polynomial time complexity in most cases, the parser performs an automatic validation on the layout structure.

The set of rules combine the layout structure, differential scaling, semantic zooming and marking mechanism. It offers the following features:

- **Visualization**. The description of the page layout can be visualized, and is easy to understand. A novice user can quickly catch the semantics.

- **Expressiveness**. The grammar can express most geometrical relations. Each grid node includes 8 directions, 3 distance relations and 3 size-changing conditions.
- **Conciseness**. The number of defined geometrical relations is minimum, yet powerful enough to express many spatial relations among graph objects.
- **Efficiency**. The definition of the spatial specification, zooming specification and marking mechanism can be easily embedded into a graph grammar without increasing the time complexity of parsing.
- **Adaptation**. A graph layout can be transformed as will according to the defined grammar. Graph objects can be zoomed to satisfy the user's requirement and particular interests.
- **Automation**. The graph transformation tool can be automatically generated, and syntax check and design validation are also automatically performed.

# 7. Future Work

The above discussion has shown that an extended RGG is promising in providing a powerful mechanism to represent the layout structure graphically and to perform an online transformation validation through an automatically generated parser. The context-sensitive property makes the grammar more powerful for layout transformations than a context-free grammar. It can also be used in other areas such as the development of multimedia interfaces, graph layout and PCB design. We will use our graph grammar formalism to represent dynamic interactive interfaces in the future research. More specifically, we will focus on the following topics in the future work:

**More Extensions**: We classify geometrical relations into three categories: distance specification, direction specification and size specification. We will design rules for more special relations, such as touch, overlapping and containing. We will illustrate the full use of these extension rules to our context-sensitive graph grammars and use realistic examples to demonstrate the conciseness and expressiveness of the extended graph grammar formalism.

**Animation**: Designing dynamic mobile interfaces is a hot topic, and there are great demands for interactive communication. In the field of dynamic capture and access, authoring, the attributes of page elements are defined in terms of those of other document elements and attributes of the viewing environment. We will combine the time and spatial specifications extended to our context sensitive graph grammar and explore the full power of spatial/temporal graph grammars for PDA interface transactions.

**Temporal Aspects**: Allen presented some common temporal relations such as *during, before, meet* relations [2], which are potentially adaptable to Web info transformations. Temporal specifications determine the sequence of presentation. There are increasing demands for interactively changing the detail of one part of a Web page when viewing it. Such a mechanism is called *interactive semantic zooming* [11]. For example, two nodes A and B, expand their sizes alternatively. The viewing interface is changed dynamically. We will investigate how to equip our GG with temporal specification capability.

## References:


[1] G. D. Abowd and E. D. Mynatt, Charting Past, Present, and Future Research in Ubiquitous Computing, *ACM Transactions on Computer-Human Interaction*, Vol.7, No.1, March 2000, 29–58.

[2] J. F. Allen, Maintaining Knowledge About Temporal Intervals, *Communications of the ACM* Vol.26, No.11, 1983, 832 – 843.

[3] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing Algorithms for the Visualization of Graphs,* Prentice Hall, Engle wood Cliffs, NJ, 1999.

[4] O. W. Bertelsen and C. Nielsen, Augmented Reality as a Design Tool for Mobile Interfaces. *Proc. ACM DIS '00 Conference*, Brooklyn, New York, Aug. 2000.

[5] S. Bjork, J. Redstrom, P. Ljungstrand and L.E. Holmquist, PowerView Using Information Links and Information Views to Navigate and Visualize Information on Small Displays, P.Thomas and H.-W. Gellersen (Eds.): *Proc. HUC'2000*, 2000, 46-62.

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

[7] E. J. Golin, A Method for the Specification and Parsing of Visual Languages, *Ph.D. Thesis*, Brown Univ., May 1991.

[8] I. Herman, G. Melançon and M. S. Marshall, Graph Visualization and Navigation in Information Visualization, *IEEE Transactions on Visualization and Computer Graphics*, Vol.6, No.1, 2000. 24-43.

[9] M. Kaul, Parsing of Graphs in Linear Time, *Proc. 2nd Int. Workshop on Graph Grammars and Their Application to Computer Science*, LNCS 153, 1982, 206-218.

[10] 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 (El.), SICI*. Vol.24, No.6, December, 2000. 819-834.

[11] K. Marriott, B. Meyer, and L.Tardif, Fast and Efficient Client-Side Adaptability for SVG, *Proc. WWW 2002*, May 7-11, Hawaii, USA, 2002, 496-507.

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

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

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

[15] 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, 75-103.

[16] M. Minas and G. Viehstaedt, Specification of Diagram Editors Providing Layout Adjustment with Minimal Change, *Proc. 1993 IEEE Symposium on Visual Languages*, 1993, 324-329.

[17] L. M. Wills, Automated Program Recognition by Graph Parsing, *Ph.D. Thesis*, MIT AI Lab, 1992.

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

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

[20] 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, 186-200.

[21] K. Zhang, D. Q. Zhang, and Y. Deng, Graphical Transformation of Multimedia XML Documents, *Annals of Software Engineering*, 12, 2001, 119-137.

[22] K. B. Zhang and K. Zhang, An Incremental Approach to Graph layout Based on Grid Drawing, *Proceedings of the Third Workshop on Software Visualization (SoftVis'99),* University of Technology, Sydney. December 3-4, 1999.