Web Interface Interpretation Using Graph Grammars

Jun Kong, Omer Barkol, Ruth Bergman, Ayelet Pnueli, Sagi Schein, Kang Zhang, and Chunying Zhao

***Abstract*—With the advent of the Internet, it is desirable to interpret and extract useful information from the Web. One major challenge in Web interface interpretation is to discover the semantic structure underlying a Web interface. Many heuristic approaches have been developed to discover and group semantically related interface objects. However, those approaches cannot solve the problem of nonuniformity satisfactorily and are not able to tag the semantic role of each object. Distinct from existing approaches, this paper develops a robust and formal approach to recovering interface semantics using graph grammars. Because of the distinct capability of spatial specifications in the abstract syntax, the spatial graph grammar (SGG) is selected to perform the semantic grouping and interpretation of segmented screen objects. Instead of analyzing HTML source codes, we apply an efficient image-processing technology to recognize atomic interface objects from the screenshot of an interface and produce a spatial graph, which records significant spatial relations among recognized objects. A spatial graph is more concise than its corresponding document object model structure and, thus, facilitates interface analysis and interpretation. Based on the spatial graph, the SGG parser recovers the hierarchical relations among interface objects.**

***Index Terms*—Data extraction, graph grammar, image processing, page segmentation.**

I. Introduction

WITH the large amount of heterogeneous data on the Web, it is desirable to automatically interpret a Web interface and extract useful information [4]. One major challenge in Web interface interpretation is to discover the Web interface semantics, i.e., *page segmentation*, which groups semantically related interface objects in a hierarchical structure and accordingly tags the semantic role of each object [46].

Since Web interfaces are created autonomously, the irregularities caused by different designers and organizations make it challenging to extract interface semantics. Many researchers have explored heuristic approaches [1], [7], [10]–[12], [14], [15], [21], [22], [35], [44]–[46] to discovering the information organization underlying a Web page. These heuristic approaches in general perform page segmentation by analyzing the document object model (DOM) structure through a set of heuristic rules. However, HTML is a very flexible language and different designers may use the HTML language completely differently. For example, tables in HTML are designed to organize and display tabular data, implying that information in a table is closely related. However, by not displaying the table border, many developers use a table as an organization grid to layout pictures and texts. In this case, information enclosed in a table may not necessarily be semantically relevant. Ahmadi and Kong [1] have evaluated six heuristic rules on three genres of Web sites (e.g., news, travel, and shopping) and concluded that heuristic rules have different accuracies on different genres. In addition to the diversity of HTML usages, the complexity of DOM structures also negatively affects the performance of page segmentation. Furthermore, heuristic approaches can group closely related information, but they are not capable of tagging semantic roles.

Recently, visual language formalisms have been proposed to analyze Web interfaces [30]. Distinct from heuristic approaches, this approach formalizes a common Web pattern as a graph grammar, which formally and visually specifies the information organization underlying a Web page. The grammar-based approach interprets a Web page from bottom to top and, thus, needs to first recognize atomic information objects before page segmentation. However, it is challenging to recognize atomic information objects from the DOM structure. For example, in an HTML Web page, a line of texts (i.e., an atomic interface object) may be separated by several HTML tags, and the seperation is content dependent.

Based on the preliminary work [48], this paper proposes a novel approach to page segmentation, taking advantage of graph grammars to provide robust page segmentation without relying on DOM structures. Because of the unique spatial specification capability in the abstract syntax, the spatial graph grammar (SGG) [29] is used in our approach to analyze Web interfaces. Spatial specifications in the abstract syntax enable designers to model interface semantics with various visual effects (e.g., a topological relation between two interface objects). Our approach interprets a Web page, or any interface page, directly from its image, instead of DOM structures. Image-processing techniques [16] are used to divide an interface image into different regions and recognize and classify atomic interface objects, such as texts, buttons, etc., in each region. The object recognition produces a spatial graph, in which nodes represent recognized atomic objects and edges indicate some significant spatial relationships (such as touch and containment). Finally, the SGG parser parses the spatial graph to discover the hierarchical relations among those interface objects based on a predefined graph grammar.

Manuscript received March 15, 2011; revised July 27, 2011; accepted September 22, 2011. Date of publication December 7, 2011; date of current version June 13, 2012. This work was supported in part by HP Labs Innovation Research Program (2009-1047-1-A and 2009-1048-1-A). This paper was recommended by Associate Editor A. M. Tjoa.

J. Kong is with the Department of Computer Science, North Dakota State University, Fargo, ND 58108 USA (e-mail: jun.kong@ndsu.edu).

O. Barkol, R. Bergman, A. Pnueli, and S. Schein are with HP Labs, Haifa 32000, Israel (e-mail: omer.barkol@hp.com; ruth.bergman@hp.com; ayelet.pnueli@hp.com; sagi.schein@hp.com).

K. Zhang is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: kzhang@utdallas.edu).

C. Y. Zhao is with the Department of Computer Science, Western Illinois University, Macomb, IL 61455 USA (e-mail: c-zhao@wiu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCC.2011.2171335

To our knowledge, the aforementioned approach is the first to combine image processing with graph grammar, which effectively addresses the issue of nonuniformity and simplifies page segmentation. The discovered interface semantics is useful in many Web-based applications, such as content adaptation, information retrieval, or usability evaluation. For example, based on the discovered semantics, we can adapt the information presentation to mobile devices or retrieve records by combining the interface semantics with a data model and a query language. Furthermore, the interface interpretation can verify whether different Web pages conform to a common pattern or not. In addition to Web interfaces, our work can be applicable to normal GUI applications and is useful to complement public application programming interfaces to extract useful data.

The rest of this paper is organized as follows. Section II introduces the background. Section III reviews related work. Section IV gives an overview of our approach. Section V illustrates our approach through a small example. Section VI goes through a case study. Section VII discusses the evaluation results of the case study, followed by conclusion and future work in Section VIII.

II. Background

This section first introduces the image-processing technique that is used to divide an interface image into several regions and recognize atomic interface objects in each region, and then introduces the SGG which is used to analyze the semantics underlying a Web page.

A. Image Processing

We break the task of classifying the functional parts of a GUI image into two phases. The first phase segments the image into candidate elements. These elements are usually of simple geometric shapes (i.e., rectangles and circles), and the second phase assigns a label to each candidate element. While some of the objects correspond to functional elements, such as links, buttons, and edit boxes, other elements are not classified and are considered as frames that can reflect the layout of a GUI. Inferring an inclusion relation between these frames can be utilized by the SGG parser to simplify the parsing process.

1) Graphical User Interface Segmentation: The first step starts by finding edge points based on a linear line-detector filter [27]. We convolve an image with a linear edge detector kernel in the horizontal and vertical directions and threshold the result. Then, long edges that are directed in the horizontal and vertical directions are considered. Pairs of parallel edges are aggregated to identify rectangles. Rectangle detection effectively identifies both frames and object boundaries.

There are some GUI components with a very distinct appearance, e.g., radio buttons and check buttons. For this kind of GUI components, we apply custom detectors. The detector analyzes the morphological edges in the image (a morphological edge is the difference between the maximal and the minimal pixel values in a given neighborhood, usually 3×3 or 5×5 pixels) [27]. It looks for symmetrical and small edge blobs. Circular blobs are detected as radio buttons, and square blobs are detected as check boxes.

Additionally, many objects have a uniform color and a high contrast with their background. Using this observation, even in the absence of sharp edges, basic objects within a screen image can be identified. The current algorithm relies mainly on edge information and uses the primary colors to enhance the segmentation when needed.

2) Graphical User Interface Object Recognition: The output of the image segmentation is a set of partially classified GUI elements, such as text and radio buttons. The task of the classification phase is to further classify GUI elements into one of the known classes of GUI objects or decide that a segment is not a GUI object. We employ a three-step approach for GUI element recognition.

1) *Represent each element as a vector in a high-dimensional feature space.* Selecting the proper object representation is crucial in any object recognition system. Features should be discriminative, and two different objects should be represented as two different feature vectors, but they should also be robust to uninformative differences. For example, in the case of GUI objects, color is usually regarded as an uninformative trait. We represent objects using two types of features that are concatenated into a single vector: (1) basic geometric features and (2) projection features. The first part represents the geometry of the object, e.g., width and height. The second part is based on the notion of projection [25], [27].
2) *Define a distance function to compare the similarity between pairs of elements.* Since we have two different types of features in a single unified vector, we also need two different comparison measures. The dimension-based distance measure corresponds to the amount of deviation between the dimensions of the objects. Let O_i and O_j be two objects of dimensions $n_i \times m_i$ and $n_j \times m_j$ respectively. We define our geometric distance measure to be
$$M_1(O_i, O_j) = 1 - \frac{\min(m_i, m_j)}{\max(m_i, m_j)} \cdot \frac{\min(n_i, n_j)}{\max(n_i, n_j)}.$$
Note that $M_1(O_i, O_j) \in [0, 1]$. To compare two projection-based feature vectors $[S_{\text{vert}}(O_i), S_{\text{horiz}}(O_i)]$ and $[S_{\text{vert}}(O_j), S_{\text{horiz}}(O_j)]$, we compute their edit-distance [33], which is defined as the minimal number of insertions, deletions, and swaps of characters that is needed to turn one string into the other. The edit-distance measure is robust to nonuniform scaling of GUI objects. In other words, the most prevalent change between the appearances of a database object and an instance is the nonuniform scaling in one of the orthogonal directions. Thus, edit-distance allows cheap changes such as stretches (e.g., wider button), but pays costly for shape changes (e.g., rectangle with rounded corners which is usually a button versus rectangle with sharp corners which is usually an edit box).
3) *Search a database of tagged examples for the nearest neighbor to classify a query object into a specific class.* To save execution time, we employ a cascading approach

where each distance function filters the database so that more expensive features are computed for a smaller number of database objects. First, the database is scanned to find all objects which are closer than a threshold T_1 relative to the shape measure M_1, and then, these objects are compared using the projection measure M_2 relative to a threshold T_2. The selection of the threshold can be tuned per-application/database to minimize classification errors. According to the thresholds, the classification appears in two stages, where the first stage is comparing features that are cheap to compute, e.g., the object's aspect ratio. This crude filtering prevents from doing expensive computations on objects that are very dissimilar. The ones that pass this first crude stage will have to go through the more expensive edit-distance computation with T_2. The small set of candidates is compared with the test object, and the class of the database object which minimizes the weighted sum is selected as the nearest neighbor [8] class. The output of the classification phase is a set of GUI objects, each with its class type and auxiliary information (e.g., object location, object geometry, and embedded text). A GUI image might contain regions that are not part of any token.

B. Spatial Graph Grammar

Graph grammars with their well-established theoretical background can be used as a natural and powerful syntax-definition formalism [39] for visual languages, which model structures and concepts in a 2-D fashion [18]. It defines computation in a multidimensional fashion based on a set of rewriting rules, i.e., *productions*. Each production consists of two parts: a left graph and a right graph; the difference of which visually indicates the changes caused by a computation. Most of graph-grammar formalisms use nodes to represent objects and edges to model relations between objects in the abstract syntax. Different from other graph-grammar formalisms, the SGG [29] introduces spatial notions to the abstract syntax. The direct representation of spatial information in the abstract syntax makes productions easy to understand since grammar designers often design rules with similar appearances as the represented graphs. In other words, using spatial information to directly model relationships in the abstract syntax is coherent with the concrete representation. The SGG production in Fig. 1(a) models the composition of an information block. In this example, an edge between two nodes indicates a semantic relation. According to the production in Fig. 1(a), a *product* information block is made of three information blocks, i.e., *link*, *image*, and *text*. The *link* has a close semantic relation with the *image*, which in turn has a semantic relation with the *text*. Furthermore, the *link* is placed above the *image*, and the *image* above the *text*. SGG supports the syntax-directed computation through *action code*. An action code is associated with a production and is executed when the production is applied. Writing an action code is like writing a standard event handler in Java. In the aforementioned example, the action code specifies that the size of the *product* information block is merged from three information blocks, i.e., *Link*, *Image*, and *Text*. A complete description of the SGG is discussed in [29].

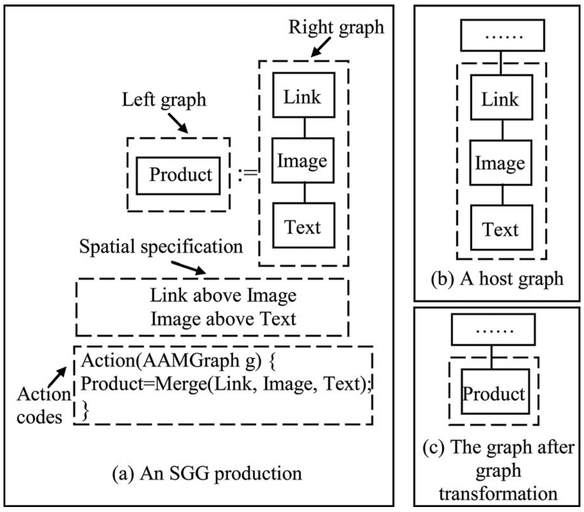


Fig. 1. (a)–(c) SGG formalism.

Applying a production to an application graph, which is usually called a *host graph*, is referred to as a *graph transformation*. A *redex* refers to a subgraph in the host graph which is isomorphic to the right graph in a parsing process. In other words, in the graph transformation in a parsing process, we search in the host graph for a subgraph that matches the right graph of a production and replace this subgraph with the left graph in the production. For example, Fig. 1(b) shows a host graph, i.e., an abstraction of a Web page. The redex that matches the right graph in Fig. 1(a) is highlighted in the dotted rectangle. After one graph transformation that groups related information, the new host graph is updated as shown in Fig. 1(c). In the graph transformation, SGG uses the marking technique to handle the embedding issue and dangling edges.

III. Related Work

As a challenging issue in Web interface interpretation, page segmentation has attracted much attention. Distinct from previous work, this paper develops a novel approach, which integrates the advantages of image-processing and graph-grammar techniques. The image-processing technique efficiently recognizes information objects and abstracts the original Web pages as a concise spatial graph, while graph grammar provides a solid foundation for interpreting a Web interface in terms of its spatial configuration. Currently, various page segmentation methods have been proposed for different applications. Those methods have been evaluated on different Web sites. In order to have an objective and quantitative comparison among different approaches, it is necessary to set up several benchmark Web sites for the evaluation and comparison purpose. Especially, those benchmark Web sites should cover different categories so that we can compare both the accuracy and generality.

In order to allow users to find information of interest quickly, a good Web designer often observes design guidelines to render information on the Web. For example, some HTML tags are commonly used to imply a boundary between interface objects. Those observations motivate various heuristic approaches, which discover blocks of closely related contents by analyzing the visual appearance or the HTML DOM structure of a Web page. Those approaches are automatic and efficient in grouping relevant information. Different from our grammar-based approach, they lack a formal basis and do not recover the semantic role of each interface object, which is useful in many applications.

Many heuristic approaches [11], [12], [28], [35] use HTML structural tags (like *Table*) to partition a Web page. Kaasinen *et al.* [28] proposed an HTML/WML conversion proxy server, which converts HTML-based Web contents to WML by mapping HTML structures to WML specifications. For example, it converts an HTML table to a WML table, an indexed subtree or a list according to the table size and viewing capacity. Buyukkokten *et al.* [11], [12] divided a Web page into several semantic textual units through HTML tags, e.g., the tag *P* might serve as the boundary between two semantic textual units. This method, however, only focuses on texts without supporting graphics. SmartView [35] used a thumbnail to provide a visual overview of a page and partition a page into logic units according to table tags. Opera [36] offers a small-screen rendering technology, which stacks Web contents vertically to avoid horizontal scrolling. This method may falsely separate closely related contents and combine unrelated information together. The DOM-structure-based analysis is limited by the complexity of DOM structures.

Recently, visual analysis has attracted more and more attention. Yang and Zhang [45] evaluated the visual similarities of HTML contents, detected the pattern of visual similarity, and then generated a hierarchical representation of the HTML page. Chen *et al.* [14] first divided a Web page into several high-level information blocks according to their sizes and locations, and, then, identified explicit and implicit separators inside each high level block. Based on the partition, a Web page is adapted to several subpages with a two-level hierarchy: a thumbnail at the top level for an index of contents and a set of subpages at the bottom level for detailed reading. CMo [9] utilizes geometrical alignment of frames to segment a Web page. Paterno and Zichittella [37] dynamically split the presentation of a desktop page by calculating the cost (e.g., the number of pixels of images or font sizes) of information objects. The vision-based page segmentation (VIPS) [13], [46] utilizes useful visual cues and DOM structures to obtain the partition of a Web page at the semantic level. Xia *et al.* [43] adjusted the VIPS algorithm to produce an SP-tree that represents the hierarchical structure of information blocks in a Web page. Hattori *et al.* [23] calculated the strength of connections between content elements based on the structural depth of HTML tags and analyzed the layout to segment a page. Ahmadi and Kong [1] analyzed both the DOM structure and the visual layout to divide the original Web page into several subpages, each including closely related contents and suitable for small-screen display. This approach supports automatic generation of a table of contents to facilitate the navigation between different subpages.

Visual language formalisms have been applied to analyzing patterns of Web queries [47]. Given a grammar in the form of a variant of the attributed multiset grammar [20], which specifies commonly used Web query patterns, a best-effort parser analyzes a query form by parsing the spatial arrangement of visual objects inside the form. This paper emphasizes on query interface, instead of a whole Web page. Kong *et al.* [30] uses SGG to analyze the semantics of a Web page. This approach is based on DOM specifications, instead of an image analysis.

HTML scraping [34] has been widely used to scrape HTML Web pages. Based on predefined regular expressions, the HTML scraping technique can efficiently extract useful data from Web pages. *Wrapper induction* [2], [19], [31] is used to extract structured data from Web pages or semistructured documents. Labský *et al.* [32] combined the hidden Markov models with image classification to extract structured information. Wong and Lam [42] proposed a novel framework, which can automatically adapt a previously learned wrapper from a source Web site to a new unseen site in the same domain. Instead of analyzing the organization of all information in a Web page, those approaches emphasize on extracting structured knowledge in response to a submitted query. Some researchers [3], [4] analyzed the detailed contents in an HTML Web page to extract semistructured data. Ashraf and Alhajj [3] proposed a novel approach, called *ClusTex*, which applies a clustering technique for information extraction from HTML pages. This approach first uses a clustering technique to divide raw data into clusters, which are then refined to eliminate irrelevant information. Ashraf *et al.* [4] later applied ClusTex to a number of Web sites from different domains. The evaluation results show good performance. Different from ClusTex, our approach interprets a Web page from its layout, instead of detailed contents.

Recently, the visual perception technique has been applied to extract structured data, since it is independent from the detailed implementation underlying a Web page. These approaches [17], [49] basically calculate the visual similarity among different Web pages to group semantically related information. Zheng *et al.* [49] introduced a *template independent* system to identify news articles based on visual consistency. This approach summarizes a set of visual features to present news stories and then automatically generates a template-independent wrapper based on those visual features. Chen and Xiao [17] proposed a system to extract news stories based on visual perception. First, it identifies the areas that contain news stories based on content functionality, space continuity, and formatting continuity. After detecting the news areas, news stories are extracted based on the position, format, and semantics. The two aforementioned approaches [17], [49] are limited to extract news stories and are not applicable to other domains. Distinct from those two approaches, our approach is general to different application domains.

IV. System Design

Fig. 2 shows an overview of our approach. Page segmentation proceeds in two steps: 1) recognize interface objects in an

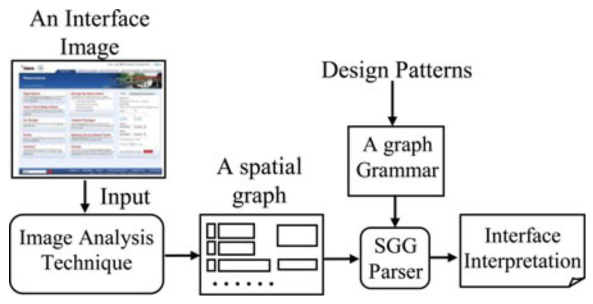


Fig. 2. Web interface interpretation.

interface image; and 2) interpret the interface through graph grammars.

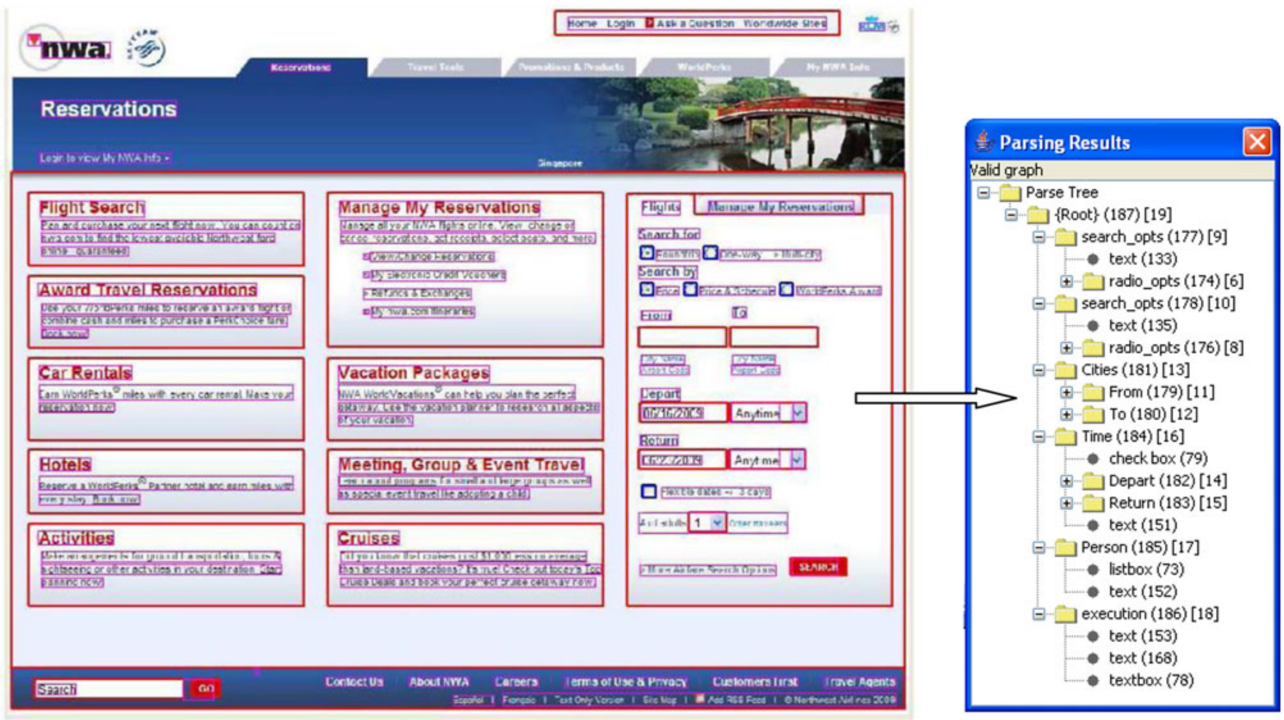
Our approach interprets an interface from bottom to top. Therefore, the first step is to recognize and classify interface objects. HTML tags may separate an atomic interface object into several pieces. For example, in order to highlight several words, a sentence, i.e., an atomic interface object, may be separated into several pieces by the HTML tag $<b>$. Since the separation is completely content dependent, it is challenging to derive some general rules to consolidate those pieces into atomic interface objects from the perspective of DOM structures. In our approach, image segmentation techniques, such as line detection, text detection, and template matching, are applied to identify interface objects in an interface image. In addition, object recognition techniques are used to recognize the interface object types, such as buttons, text boxes, images, etc. Object recognition and classification divide an interface into several regions and have the following advantages.

1) *Incremental analysis.* Those regions provide natural boundaries between composite interface objects. In other words, we can interpret interface objects incrementally—first, interpret interface objects within one region, and then consider the relations between different regions. Such an incremental analysis is inherently consistent with the hierarchical design of Web interfaces. Designers, in general, first divide an interface into several layout regions, and then determine the organization and layout of interface objects within each region.
2) *Handling irregularity.* If a region contains style exceptions, not captured by a graph grammar, we can simply interpret those exceptional interface objects as the direct children of the node representing the corresponding region.
3) *Performance.* Limiting the spatial parsing in a small region can reduce the search space and improve performance.

The visual analysis on the interface image generates a spatial graph. In a spatial graph, nodes represent recognized interface objects, and edges indicate siginicant spatial relations, each indicating a close semantic relationship. We have evaluated different Web sites and discovered that four spatial relations strongly imply a close semantic relation between two objects, i.e., touching, containment, vertical, and horizontal relations within a small distance. Accordingly, each edge in a spatial graph corresponds to one of those four relations. Meanwhile, a spatial graph also records the coordinates of each recognized interface object. Therefore, other additional spatial relations can be derived from those coordinates. Compared with the source HTML file, the spatial graph facilitates page segmentation by: 1) consolidating information pieces together; and 2) removing all redundant contents (such as empty columns or tables, which are used for adjusting layout).

After visual analysis, a graph grammar is applied to a spatial graph to discover the interface semantics. In our appraoch, page segmentation can be considered as a graph transformation issue. The input is a spatial graph that is abstracted from a concrete Web interface, and the output is a tree that reveals the hierarchical relations among interface objects. Therefore, graph grammars are a natural computational model for page segmentation. More specifically, the left graph in a production may include a composite interface object, which is made of a set of atomic/composite interface objects in the right graph. Each production groups related objects locally and a complete graph grammar provides a systematic specification to glue low-level groups into a higher level group. Based on the graph grammar, a parser constructs a hierarchical parsing tree, in which a leaf node indicates an atomic interface object and an intermediate node represents a composite object, bottom up as a coherent interpretation for a Web interface. Our approach is not limited to a specific graph-grammar formalism. This paper uses the SGG [29] as the specification formalism to analyze Web interfaces because of its distinct capacity of spatial specification in the abstract syntax. This unique feature enables us to extend our approach to combine DOM analysis with image processing in the future. The incremental parsing in our approach supports reusing a portion of a graph grammar in different Web sites. Even though two Web pages from different Web sites may be different at a high level, some low-level patterns, such as the organization of a paragraph, may be used repetitively across different Web sites.

Following the human–computer interaction principle that consistent layouts can improve the usability of an interface [40], Web designers commonly use similar layouts to present the same type of information. In other words, designers, in general, follow previous successful experiences, which can be summarized as guidelines, to present interface objects [24]. Some researchers [26] summarized common design patterns across different Web sites. We have evaluated 21 commercial Web sites and found that all those Web sites use two common patterns to demonstrate product information [38]. The usage of common patterns or standard guidelines makes our approach applicable in practice. A graph grammar can be applied to different Web sites that conform to one common pattern or standard. In order to reduce the manual efforts of designing a graph grammar, we plan to introduce grammar induction technique to automate the grammar design process in the future. The automatic grammar induction is especially useful when a content management system is used to generate Web pages. The grammar induction algorithm can automatically extract generation rules, and then, a grammar parser can perform a reverse processing to discover the underlying interface semantics.



```
Valid graph
Parse Tree
  {Root} (187) [19]
    search_opts (177) [9]
      text (133)
      radio_opts (174) [6]
    search_opts (178) [10]
      text (135)
      radio_opts (176) [8]
    Cities (181) [13]
      From (179) [11]
      To (180) [12]
    Time (184) [16]
      check box (79)
      Depart (182) [14]
      Return (183) [15]
      text (151)
    Person (185) [17]
      listbox (73)
      text (152)
    execution (186) [18]
      text (153)
      text (168)
      textbox (78)
```

(a) (b)

Fig. 3. NWA example. (a) Interface Image. (b) Interpretation.

In summary, our approach uses image analysis to recognize atomic interface objects and applies the SGG to specify patterns underlying Web pages. Based on the grammar, a graph parser takes a spatial graph, which is abstracted from an interface image, as input and produces a semantic interpretation of the interface.

V. Interpreting Web Interfaces

Having given an overview of our approach, this section uses the Northwest Airlines (NWA) example (see Fig. 3) to illustrate our approach.

A. *Northwest Airlines Web Page*

Consider an example of NWA as shown in Fig. 3(a). The top and the bottom include some isolated texts, which direct users to other Web pages in the NWA. The central region, which includes the main content, can be further divided into several regions. It includes several topics, such as flight search and hotels. Those topics are organized and displayed with the same pattern: Each topic has a title and several lines of texts. A search interface is displayed on the right of the central region.

B. *Image Analysis*

The input for the image analysis phase is the rendered Web page of NWA Reservation Center. In the image analysis phase, the image is first segmented using a suite of layout and object detection algorithms. These include rectangle detection, radio button detection, and text detection and recognition. We construct a hierarchical tree based on object containment. Thus, at

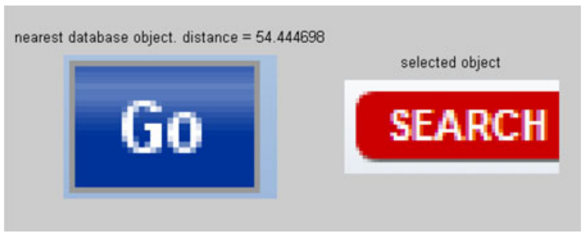


Fig. 4. Nearest neighbor result for a SEARCH button.

this point, we have a partially classified tree. In Fig. 3(a), each recognized object is highlighted with a rectangle. Objects are coded by the type with different colors. For example, blue indicates a *radio button* or *check box*, magenta indicates *text*, and red indicates a rectangle of *unknown* type.

Next, the GUI object recognition algorithm classifies each of the recognized rectangles based on a database, which includes a variety of manually classified GUI objects with a typical Web look and feel. In order to correctly recognize the type of an interface object, classified interface objects in the database should have a similar appearance as the elements in a page being analyzed. Since many Web sites keep consistent appearances, the database could be reused across different Web sites. In the aforementioned example, all the text boxes, list boxes, and buttons were classified correctly. Fig. 4, for example, shows the nearest match found for the "Search" button. Although the colors, the sizes, the aspect ratio, and the text on the buttons are different, the nearest match is of the correct classification. An even better match could be obtained if we use objects from the relevant site in the database.

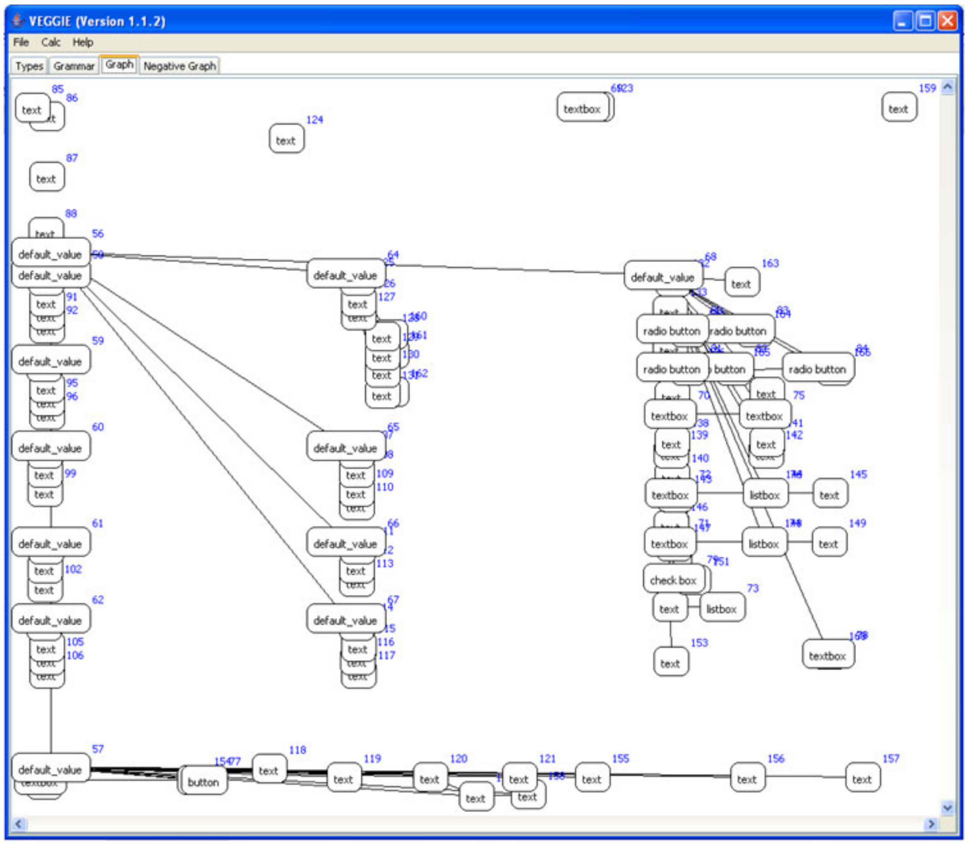


Fig. 5. Spatial graph.

C. Page Segmentation

After image analysis, we have a fully classified set of layout segments (frames) and GUI objects: their location on the page, associated type, text value, and other spatial attributes. In Fig. 3(a), the left side of the central region includes nine different topics about reservation. Each topic is displayed within a rectangle, recognized as a *Default_Value* object, and includes several lines of texts, each being recognized as a *text* object. In the search interface at the right side, five types of atomic interface objects are recognized as *text*, *radio button*, *check box*, *text box*, and *list box*. Based on recognized interface objects and their spatial features, a spatial graph is constructed by calculating significant spatial relations among those objects. Fig. 5 presents the spatial graph corresponding to the interface image in Fig. 3(a).

Based on the spatial graph, we formalize the pattern of information layouts as a graph grammar. In a graph grammar, the spatial relations of containment, touching, and vertical/horizontal relation within a short distance are visualized as edges, while other spatial relations are defined through spatial specifications [see Fig. 1(a)] in an SGG production. Fig. 6 presents the productions used to derive a topic on the left side of the central region. Production P1 in Fig. 6 combines two adjacent lines of texts as a composite interface object *texts*. The node *default_value* in P1 serves as a context object. Production P2 is useful when a topic includes an odd number of lines of texts. Production P3 further combines two composite interface objects *texts* as a larger composite object. Finally, Production P4 abstracts all lines of texts as a composite interface object *topic*. The search interface on the right of the central region in Fig. 3(a) can be analyzed based on the complete graph grammar in Appendix 1 at http://viscomp.cs.ndsu.nodak.edu/Grammar/grammar.pdf. In the future, we will integrate the concept of negative application conditions in the parsing process to enforce a proper match.

Each graph transformation reveals a local composition, and a sequence of graph transformations, i.e., the parsing process, assembles local compositions into a global hierarchical structure. In other words, the nonterminal object that is defined in the left graph is made up of (non)terminal objects in the right graph, while context objects are not counted to the composition. Such a hierarchical structure, in which leaf nodes represent atomic interface objects while intermediate nodes indicate composite objects, reveals the composition of interface objects. The application of the graph grammar in Appendix 1 (refer to http://viscomp.cs.ndsu.nodak.edu/Grammar/grammar.pdf) produces the hierarchical structure [see Fig. 3(b)] as an interpretation for the search interface in Fig. 3(a).

VI. Case Study

A. System Implementation

We have implemented a prototype for page segmentation. Our prototype mainly includes two subsystems: one supports

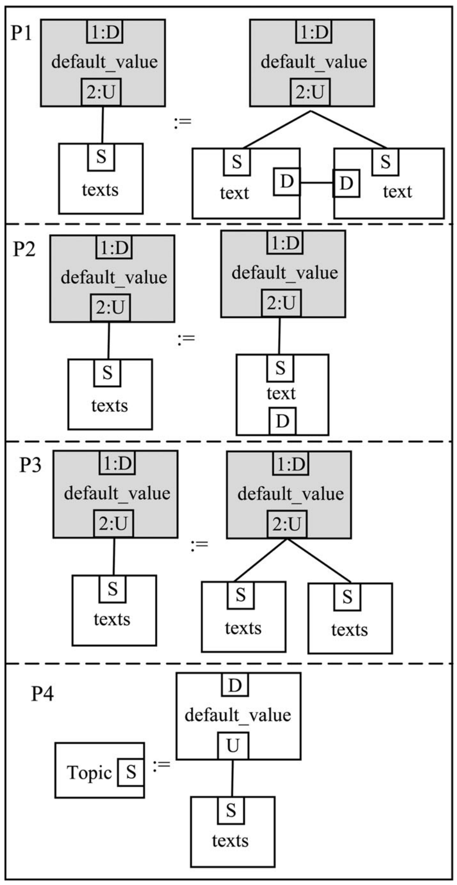


Fig. 6. Productions used to analyze topics.

image analysis to recognize atomic interface objects and the other supports spatial parsing for page segmentation.

For image analysis, we developed the CompDetect application, which supports loading either an image or directly from a URL. Once the image is loaded, CompDetect activates the stages of segmentation and classification. Segmentation can be done either incrementally or all at once automatically. The output of the segmentation stage is presented visually on the screen to allow manual inspection. The classification stage can be divided into three stages. First, create a database, which is either loaded from a previously created file or created by incrementally fixing initial classification results. The second stage allows the user to classify interface objects. Once again, the user can supervise the classification result and fix them manually. The last stage allows the user to save the classified objects into an XML file in order to constitute an input for page segmentation in the next step.

After image analysis, the automatically generated spatial graph is sent to a visual programming environment, i.e., Visual Environment for Graph Grammars: Induction and Engineering (VEGGIE) [5], [6], for page segmentation. VEGGIE supports the SGG specification and parsing. VEGGIE mainly consists of three independent editors (i.e., the Type Editor, the Grammar Editor, and the Graph Editor) and an SGG parser. The three editors provide GUIs for designers to visually design a graph grammar and are seamlessly working together in VEGGIE. The combined views ease the switching between different editors with a consistent look and feel, which enhances a coherent understanding. Grammar designers can visually create visual objects, i.e., node types, in the Type Editor, or import existing node types from a file in the form of GraphML. Then, based on these defined nodes, the designer can define productions in the Grammar Editor. In the Graph Editor, the designer can visually draw or import a spatial graph to be analyzed by the SGG parser. The data files storing nodes, grammar, and graph are shared and interoperated by all editors.

B. Commercial Web Site

We evaluated our approach on the Web site of Marks and Spencer (M&S), which is a major British retailer. The M&S Web site (http://www.marksandspencer.com/) hosts thousands of different products. Presenting products to customers is one of the most important functions on an online retailer Web site. Therefore, we emphasize on analyzing the presentation of product information. After investigating different M&S Web pages, we have identified two common patterns, which are used across the whole Web site.

1) *Pattern 1—Presenting a product in detail*. In order to present the detailed information of a product, an image of the selected product is displayed. At the right side and/or below the image, textual description is provided for a detailed description. The textual description may include several paragraphs, each of which has several lines of texts. In order to differentiate paragraphs, there are spaces between paragraphs. In addition to displaying product information, a variant of this pattern is used to display navigation links, which allow users to quickly switch between different Web pages.
2) *Pattern 2—Listing products*. An online retailer Web site needs to display a list of products so that a customer can choose any of them for details. At the M&S Web site, the overview of each product, in general, includes three parts. The top shows a small picture describing the product; below the small picture is a brief textual description, which may include several lines of texts; the bottom contains the price and customers' rating, which are presented in text. A list of products is presented from left to right and top down.

Although the Web pages of the M&S Web site have different contents, they are all made up of the aforementioned two patterns. Fig. 7 presents one Web page from the M&S Web site. Our approach uses the image-processing technique to divide each page into several regions. In Fig. 7, each recognized region (i.e., a *default_value* object) at a high level is highlighted in a dotted rectangle, and the pattern applicable to each region is marked with a circled pattern number: the navigation links on the left and at the bottom follow Pattern 1, and the center region which displays a list of products is consistent with Pattern 2.

In our approach, those patterns are formalized as graph grammars. We use the Type Editor of VEGGIE to define nodes. The Type Editor allows developers to specify various properties of a node, such as types, attributes, and vertices. The left panel of the

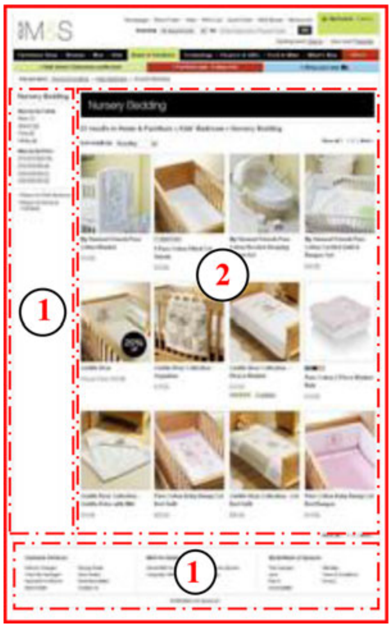
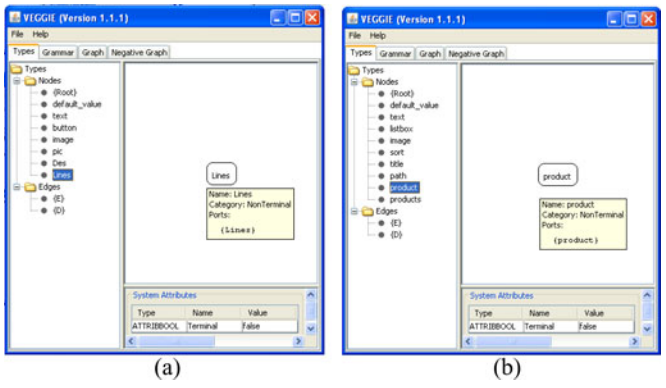


Fig. 7. M&S Web page.



Name: Lines
Category: NonTerminal
Ports:
{Lines}

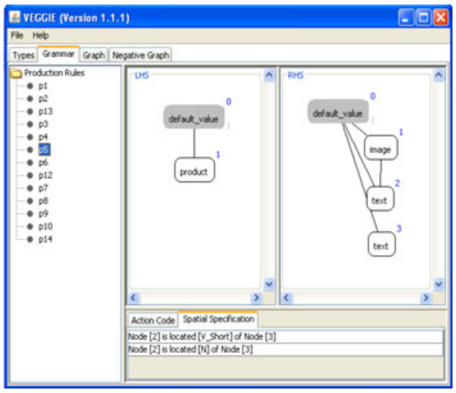
Type	Name	Value
ATTRIBBOOL	Terminal	false

Name: product
Category: NonTerminal
Ports:
{product}

Type	Name	Value
ATTRIBBOOL	Terminal	false

Fig. 8. Node type Editor in VEGGIE. (a) Node types in Pattern 1. (b) Node types in Pattern 2.

Type Editor displays all node types, and the right panel shows the detailed properties of a selected node and allows designers to modify those properties. In order to improve the efficiency of the design process, VEGGIE supports a direct import of terminal nodes from a spatial graph, instead of manually designing them. To avoid cluttering the display space, in the current version of VEGGIE, we only explicitly display nodes without their details. If the mouse moves over a node, detailed information of the node (e.g., vertices) will be displayed as depicted in Fig. 8. For example, Fig. 8(a) shows the node types in Pattern 1, while Fig. 8(b) displays the node types in Pattern 2. In Fig. 8(a), node types of *default_value*, *text*, *button*, and *image* are automatically recognized atomic interface objects, and the rest are composite interface objects. Object *pic* is made of a picture and button; object *Lines* represents multiple lines of texts which belong to the same paragraph; and object *Des* models different paragraphs. In Fig. 8(b), node types of *default_value*, *text*, *listbox*, and *image* are automatically recognized atomic interface objects. Composite object *title* is made of *text* objects, indicating a title. Object *path* represents the current location in the Web site and composes of several *text* objects, object *product* models an overview of a product, including both graphical and textual descriptions, and object *products* indicates several products.



Node [2] is located [V_Short] of Node [3]
Node [2] is located [N] of Node [3]

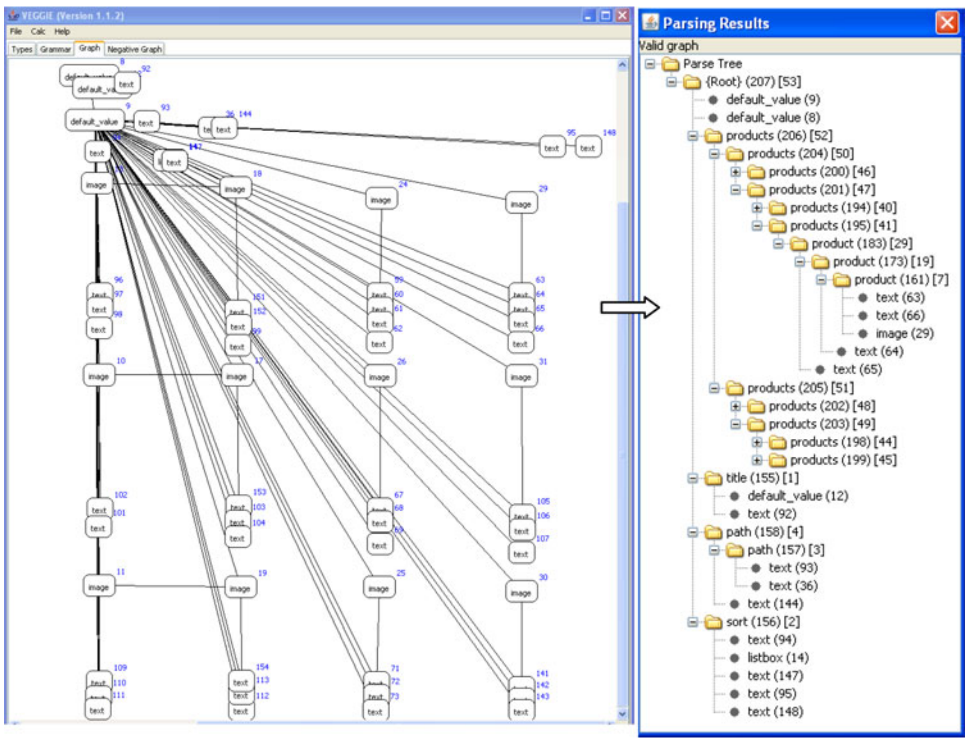
Fig. 9. Grammar Editor in VEGGIE.

The VEGGIE Grammar Editor as presented in Fig. 9 enables designers to visually manipulate productions. The leftmost panel of the Grammar Editor lists all the productions. By clicking on one of the productions, the corresponding details are displayed in the middle and right panels, representing the left graph and right graph of the selected production, respectively. Fig. 9 shows a production defined in Pattern 2. SGG is a context-sensitive graph grammar and a gray node represents a context object. Because of limited space, complete definitions for Patterns 1 and 2 can be found at http://viscomp.cs.ndsu.nodak.edu/Grammar/grammar.pdf.

Based on the defined graph grammar, page segmentation is implemented through a spatial parsing. For example, the screenshot on the left of Fig. 10 presents a spatial graph, corresponding to the product list in Fig. 7. By applying the graph grammar of Pattern 2, we can produce its interpretation in the right side in Fig. 10.

VII. Discussion

According to the case study in the previous section, we found that even a complex Web site is made of several simple patterns. This observation is also confirmed by other researchers [47]. For example, the M&S Web site mainly includes two patterns (refer to Section VI-B), and each Web page is presented based on the composition of those patterns. The existence of common patterns across a Web site makes our approach feasible in practice. Instead of designing a graph grammar for each page, we only need to consider a few patterns, which can significantly reduce the effort of designing a graph grammar. In addition to M&S, we have investigated some other popular online retailer Web sites, including buy.com, amazon.com, and newegg.com, and found that similar patterns are also used in those Web sites. In other words, the graph grammar that is designed for M&S can be reused in other similar Web sites with slight modifications. In the case study, we do find that some Web pages, e.g., the customer login page, do not follow any pattern. In this case,



```
Parse Tree
  {Root} (207) [53]
    ● default_value (9)
    ● default_value (8)
    products (206) [52]
      products (204) [50]
        products (200) [46]
        products (201) [47]
          products (194) [40]
          products (195) [41]
            product (183) [29]
              product (173) [19]
                product (161) [7]
                  ● text (63)
                  ● text (66)
                  ● image (29)
                ● text (64)
              ● text (65)
      products (205) [51]
        products (202) [48]
        products (203) [49]
          products (198) [44]
          products (199) [45]
    title (155) [1]
      ● default_value (12)
      ● text (92)
    path (158) [4]
      path (157) [3]
        ● text (93)
        ● text (36)
      ● text (144)
    sort (156) [2]
      ● text (94)
      ● listbox (14)
      ● text (147)
      ● text (95)
      ● text (148)
```

Fig. 10. Spatial Graph representing the product list in Fig. 7 and its interpretation.

we use the best effort parsing, which tries to recover as many structures as possible.

Each pattern describes the layout and organization of information at an abstract level. This implies that the instances of a pattern may be different. For example, in Pattern 1, each paragraph may include different lines of texts. Using recursive productions, the graph grammar is powerful to handle those variations. With the Web patterns, the graph grammar approach provides a powerful and visual means to specify and analyze Web pages.

One unique feature of our approach is the complementary roles of image analysis and graph grammar techniques in analyzing Web interfaces. The image analysis is critical to page segmentation, since it can significantly simplify the original Web page by eliminating redundant and blank contents and consolidating information pieces together. Currently, most approaches analyze a Web interface according to its DOM structure. However, even a simple Web page may have a complex DOM structure. We compare the spatial graph produced from image analysis with the DOM structure in terms of their complexity, as presented in Table I. Given the Web page in Fig. 7, an image analysis in our approach will produce a spatial graph with 176 nodes. On the other hand, the corresponding DOM structure has 1050 nodes, including both HTML elements and scattered information pieces. The image analysis approach produces the information that is of only 15% of the DOM's complexity. We can find similar observations in other Web pages in the M&S Web site in Table I. Furthermore, image analysis can divide a Web interface into several regions. Accordingly, the spatial parsing can be limited to a small scope with a reduced search

TABLE I
NUMBER OF NODES IN SPATIAL GRAPHS VERSUS DOM STRUCTURES

Webpage	Spatial Graph	DOM structure
Page 1	176	1050
Page 2	154	1026
Page 3	120	1357
Page 4	193	980
Page 5	196	1169

space. Thus, our approach performs more efficiently. Table II presents the running time[1] of spatial parsing in each region in five Web pages that are selected from the M&S Web site. The regions in each Web page are indexed from left to right and top down. The longest running time is 25 ms (Region 2 in page 4), which is apparently tolerable as parsing needs to be performed only once. The time complexity of the SGG parser is critical to the overall system performance. The SGG parser converts a 2-D graph to an ordered sequence of nodes based on spatial features of those nodes and uses an efficient string matching in the parsing process. SGG parser is developed based on confluent graph grammars. Informally, the SGG parser only tries one parsing path. Under the confluence condition, the SGG parser has a polynomial time complexity [29].

Compared with DOM analysis, image processing is efficient to extract atomic information objects. On the other hand, a DOM structure does provide valuable clues about information

[1]All tests were performed on a Windows PC with an Intel Pentium 3.2GHz Dual Core CPU and 4G bytes of memory.

TABLE II
RUNNING TIME OF SPATIAL PARSING

	Page 1			Page 2			Page 3			Page 4			Page 5		
	Region 1	Region 2	Region 3	Region 1	Region 2	Region 3	Region 1	Region 2	Region 3	Region 1	Region 2	Region 3	Region 1	Region 2	Region 3
Running Time (Second)	0.001	0.020	0.001	0.001	0.016	0.001	0.001	0.001	0.004	0.008	0.025	0.001	0.001	0.023	0.001

organizations, such as the hierarchical structure among containers and actual contents. In the future, we plan to combine image processing with DOM analysis. More specifically, image processing is first used to detect regions and basic information objects. Then, those recognized containers and information objects are mapped to HTML tags in the DOM structure based on their spatial properties, i.e., location and size. During the mapping, the original DOM structure is simplified by removing empty tags and consolidating related tags together. The mapping can also help refine the result of object recognition from image analysis. In the example in Fig. 3(a), one paragraph may cross several lines. In the image analysis, each line of texts is recognized as one individual interface object. By integrating the image analysis results with the corresponding DOM structure, we can combine those individual lines of texts together if all those lines of texts are enclosed in an HTML tag <*P*>. In the simplified DOM structure, information objects are classified based on both the HTML tags and image recognition. Finally, we can apply a graph grammar to the simplified DOM structure to interpret its semantics.

Compared with existing heuristic approaches, our approach provides fine-grained page segmentation. Heuristic approaches segment a Web page into several regions from top to bottom, without tagging the semantic role of each object. Instead, our approach interprets a Web interface from bottom to top. Our approach first recognizes and classifies atomic information objects and then uses graph parsing to discover the semantics among those recognized objects. Therefore, our proposed work actually interprets a Web interface, not just dividing. While most existing approaches depend on the source code to segment a Web page, our approach directly analyzes the screenshot of a Web page without relying on any source code. Consequently, our solution is applicable to the situations when source codes are not available, such as surfing the Web through a remote desktop. Finally, our approach supports the customization of the page segmentation results. A user can adjust a graph grammar to tune the page segmentation. In general, a general graph grammar has a higher generality while a specific graph grammar, which includes detailed application-dependant semantics (such as *cities* in the grammar in appendix 1 at http://viscomp.cs.ndsu.nodak.edu/Grammar/grammar.pdf), provides more detailed interpretation but it has limited generality, since it is application dependant.

VIII. CONCLUSION

In the Web interface interpretation, it is challenging to discover the semantic structure underlying a Web interface. Different from existing heuristic approaches, this paper develops a novel approach for page segmentation based on image analysis and graph grammar. Instead of analyzing DOM structures, our approach uses advanced image processing to recognize atomic interface objects and divide an interface into several regions. The image analysis produces a spatial graph, in which nodes represent recognized interface objects and edges model spatial relations, which imply a close semantic relation. Based on the spatial graph, a spatial parsing is performed to recover the semantics of the corresponding Web page. Because of the distinct capability of spatial specifications in the abstract syntax, the *SGG* is selected as the definition formalism for page segmentation. We have tested our approach on the M&S Web site, which shows promising results.

As the future work, we will conduct more experiments to investigate issues like generality and efficiency. Especially, we will investigate how to improve the efficiency of designing a graph grammar. Although the reuse of patterns across different Web sites can reduce the efforts of designing a graph grammar, we plan to further improve the efficiency by applying a grammar-induction algorithm to semiautomate the grammar design. Given sample interface snapshots, human experts first highlight most important substructures and give them some user-friendly names. Then, based on those manually highlighted substructures, a grammar induction algorithm produces a graph grammar. In addition to elaborating an induced graph grammar, human experts can also add spatial specifications to the generated graph grammars. Once the automatically produced graph grammar is evaluated and refined, it is used to interpret Web interfaces. Future work also includes evaluating the generality of the image recognition algorithm and improving the accuracy. An error in the image recognition can affect the interpretation of a Web interface. We will investigate an iterative processing. The objects that cannot be interpreted in the graph parsing are returned to the image recognition for the second round of recognition, which is then passed to the graph parser for further analysis. This process can be supervised by human experts to identify the recognition errors and improve the accuracy. Future work also includes exploring the idea of "a cascade of classifiers" [41], which has multiple stages, in the image processing to improve the performance.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful and constructive comments that have helped us to significantly improve the presentation.

REFERENCES

[1] H. Ahmadi and J. Kong, "Efficient web browsing on small screens," in *Proc. ACM Int. Working Conf. Adv. Visual Interfaces*, 2008, pp. 23–30.

[2] A. Arasu and H. Garcia-Molina, "Extracting structured data from web pages," in *Proc. Special Interest Group Manage. Data Conf.*, 2003, pp. 337–348.

[3] F. Ashraf and R. Alhajjt, "ClusTex: Information extraction from HTML pages," in *Proc. 21st Int. Conf. Adv. Inf. Netw. Appl. Workshops*, May 2007, pp. 355–360.

[4] F. Ashraf, T. Ozyer, and R. Alhajj, "Employing clustering techniques for automatic information extraction from HTML documents," *IEEE Trans. Syst., Man, Cybern.—Part C: Appl. Rev.*, vol. 38, no. 5, pp. 660–673, Sep. 2008.

[5] K. Ates, J. Kukluk, L. Holder, D. Cook, and K. Zhang, "Graph grammar induction on structural data for visual programming," in *Proc. IEEE 18th Int. Conf. Tools Artif. Intell.*, Nov. 2006, pp. 232–242.

[6] K. Ates and K. Zhang, "Constructing VEGGIE: Machine learning for context-sensitive graph grammars," in *Proc. IEEE 19th Int. Conf. Tools Artif. Intell.*, Oct. 2007, pp. 456–463.

[7] S. Baluja, "Browsing on small screens: Recasting web-page segmentation into an efficient machine learning framework," in *Proc. World Wide Web*, 2006, pp. 33–42.

[8] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, Jun. 2008, pp. 1–9.

[9] Y. Borodin, J. Mahmud, and I. V. Ramakrishnan, "Context browsing with mobiles—When less is more," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Services*, 2007, pp. 3–15.

[10] R. Burget, "Visual HTML document modeling for information extraction," in *Proc. Reconfigurable Architectures Workshop*, 2005, pp. 17–24.

[11] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Accordion summarization for end-game browsing on PDAs and cellular phones," in *Proc. ACM Special Interest Group Comput.–Human Interaction*, 2001, pp. 213–220.

[12] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the whole in parts: Text summarization for web browsing on handheld devices," presented at the Proc. of the World Wide Web, Hong Kong, 2001.

[13] D. Cai, S. Yu, J. Wen, and W. Ma, "Extracting content structure for Web pages based on visual representation," in *Proc. 5th Asia Pac. Web Conf.*, 2003, pp. 406–417.

[14] Y. Chen, W. Y. Ma, and H. J. Zhang, "Detecting web page structure for adaptive viewing on small form factor devices," in *Proc. World Wide Web*, 2003, pp. 225–233.

[15] J. Chen, X. Xie, W. Ma, and H. Zhang, "Adapting web pages for small-screen devices," *IEEE Internet Comput.*, vol. 9, no. 1, pp. 50–56, Jan./Feb. 2005.

[16] S. C. Chen, S. H. Rubin, M. L. Shyu, and C. C. Zhang, "A dynamic user concept pattern learning framework for content-based image retrieval," *IEEE Trans. Syst., Man, Cybern.: Part C Appl. Rev.*, vol. 36, no. 6, pp. 772–783, Nov. 2006.

[17] J. Chen and K. Xiao, "Perception-oriented online news extraction," in *Proc. 8th ACM/IEEE-CS Joint Conf. Digital Libraries*, 2008, pp. 363–366.

[18] P. T. Cox and T. Smedley, "Building environments for visual programming of robots by demonstration," *J. Visual Languages Comput.*, vol. 11, no. 5, pp. 549–571, 2000.

[19] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards automatic data extraction from large web sites," in *Proc. Very Large Data Bases Conf.*, 2001, pp. 109–118.

[20] E. J. Golin, "Parsing visual languages with picture layout grammars," *J. Visual Languages Comput.*, vol. 4, no. 2, pp. 371–394, 1991.

[21] X. D. Gu, J. L. Chen, W. Y. Ma, and G. L. Chen, "Visual based content understanding towards web adaptation," in *Proc. Int. Conf. Adaptive Hypermedia*, 2002, pp. 164–173.

[22] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "DOM-based content extraction of HTML documents," in *Proc. World Wide Web*, 2003, pp. 207–214.

[23] G. Hattori, K. Hoashi, K. Matsumoto, and F. Sugaya, "Robust web page segmentation for mobile terminal using content-distances and page layout information," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 361–370.

[24] A. Holzinger, "Usability engineering for software developers," *Commun. ACM*, vol. 48, no. 1, pp. 71–74, 2005.

[25] T. H. Hou and M. D. Pern, "A computer vision-based shape-classification system using image projection and a neural network," *Int. J. Adv. Manuf. Technol.*, vol. 15, pp. 843–850, 1999.

[26] M. Y. Ivory and R. Megraw, "Evolution of web site design patterns," *ACM Trans. Inf. Syst.*, vol. 23, no. 4, pp. 463–497, 2005.

[27] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[28] E. Kaasinen, M. Aaltonene, J. Kolari, S. Melakoski, and T. Laakko, "Two approaches to bringing internet services to WAP devices," *Comput. Netw.: Int. J. Comput. Telecommun. Netw.*, vol. 33, pp. 231–246, 2000.

[29] J. Kong, K. Zhang, and X. Q. Zeng, "Spatial graph grammar for graphic user interfaces," *ACM Trans. Human-Comput. Interaction*, vol. 13, no. 2, pp. 268–307, 2006.

[30] J. Kong, K. L. Ates, K. Zhang, and Y. Gu, "Adaptive mobile interfaces through grammar induction," in *Proc. IEEE 20th Int. Conf. Tools Artif. Intell.*, 2008, pp. 133–140.

[31] N. Kushmerick, D. S. Weld, and R. B. Doorenbos, "Wrapper induction for information extraction," in *Proc. Int. Joint Conf. Artif. Intell.*, 1997, pp. 729–737.

[32] M. Labský, V. Svátek, O. Šváb, P. Praks, M. Krátký, and V. Snášel, "Information extraction from HTML product catalogues: from source code and images to RDF," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2005, pp. 401–404.

[33] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–884, 1965.

[34] N. Mavridis, W. Kazmi, and P. Toulis, "Friends with faces: How social networks can enhance face recognition and vice versa," in *Computational Social Networks Analysis: Trends, Tools and Research Advances*. Berlin, Germany: Springer-Verlag, 2009.

[35] N. Milic-Frayling and R. Sommerer, "SmartView: Flexible viewing of web page contents," presented at the Proc. of the 11th World Wide Web Conf. (poster paper), New York, 2002.

[36] Opera Software ASA. (2008). [Online]. Available: http://www.opera.com/products/mobile/smallscreen

[37] F. Paterno and G. Zichittella, "Desktop-to-mobile web adaptation through customizable two-dimensional semantic redesign," in *Proc. 3rd Int. Conf. Human-Centered Softw. Eng.*, 2010, pp. 79–94.

[38] A. Roudaki and J. Kong, "Graph grammar based web data extraction," Tech. Rep. NDSU-CS-TR-10-002, North Dakota State Univ., Fargo, USA, 2010.

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

[40] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, MA: Addison-Wesley Longman, 2009.

[41] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. Comput. Visi. Pattern Recognit.*, 2001, pp. I-511–I-518.

[42] T. L. Wong and W. Lam, "Adapting web information extraction knowledge via mining site-invariant and site-dependent features," *ACM Trans. Internet Technol.*, vol. 7, no. 1, art no. 6, 2007.

[43] X. Y. Xiao, Q. Luo, D. Hong, H. Fu, X. Xie, and W. Y. Ma, "Browsing on small displays by transforming web pages into hierarchically structured subpages," *ACM Trans. Web*, vol. 3, no. 1, art no. 4, 2009.

[44] Y. D. Yang, J. L. Chen, and H. J. Zhang, "Adaptive delivery of HTML contents," in *Proc. World Wide Web*, 2000, pp. 24–25.

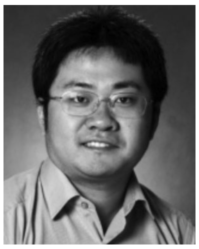
[45] Y. D. Yang and H. J. Zhang, "HTML page analysis based on visual cues," in *Proc. 6th Int. Conf. Document Analysis Recognit.*, 2001, pp. 859–864.

[46] S. P. Yu, D. Cai, J. R. Wen, and W. Y. Ma, "Improving pseudo-relevance feedback in web information retrieval using web page segmentation," in *Proc. Int. Conf. World Wide Web*, 2003, pp. 11–18.

[47] Z. Zhang, B. He, and K. C.-C. Chang, "Understanding web query interfaces: Best-effort parsing with hidden syntax," in *Proc. Int. Conf. ACM Special Interest Group Manage. Data*, 2004, pp. 107–118.

[48] K. Zhang and J. Kong, "Exploring semantic roles of web interface components," in *Proc. 2010 IEEE Int. Conf. Mach. Web Intell.*, 2010, pp. 8–14.

[49] S. Zheng, R. Song, and J. Wen, "Template-independent news extraction based on visual consistency," in *Proc. 22nd Nat. Conf. Artif. Intell.*, 2007, vol. 2, pp. 1507–1512.



Jun Kong received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1998, the M.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the University of Texas at Dallas, Richardson, in 2005, all in computer science.

He has been an Assistant Professor of computer science with North Dakota State University, Fargo, since 2006. His research and teaching interests include human–computer interaction, visual languages, software modeling and design, and pervasive computing.

Omer Barkol received the B.Sc. degree in mathematics and computer science, and the Ph.D. and M.Sc. degrees in computer science, all from the Technion–Israel Institute of Technology, Haifa, Israel, where he was focused on theoretical computer science, coding theory, and cryptography.

He is currently a Research Manager with HP Labs, Haifa. His main research deals with analytics and collaboration in large organizations, with a focus on IT management. In addition, he leads research on graph-data mining. Since 2008, he has been with HP Labs, involved both in the area of imaging and printing, and in the research area of IT information management. Prior to joining HP Labs, he has led a software team in Charlotte's Web Networks, dealing with routing protocols.

Ruth Bergman received the B.S. and M.S. degrees in computer science from the University of California, San Diego, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge.

She is currently the Director of HP Labs, Haifa, Israel. She joined HP Labs in 2001, where she leads a research agenda in analytics, collaboration, and automation in large organizations, with a focus on IT management and serviceability of printing systems. Prior to joining HP Labs, she was a Researcher with the NASA Jet Propulsion Laboratory, Pasadena, CA, and with the Lincoln Laboratory, MIT. She has authored six book chapters and ten conference papers. She holds seven patents and 17 patent applications. Her research interests include data mining, machine learning, and computer vision.

Ayelet Pnueli is currently a Research Scientist with HP Labs, Haifa, Israel.

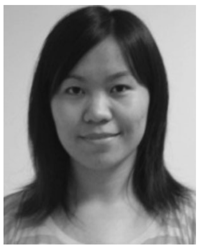
Sagi Schein received the Ph.D. degree in computer graphics and geometric computing from the Technion–Israel Institute of Technology, Haifa, Israel, where he was focused on the applications of multivariate B-spline functions to computer graphics, computer-aided geometric design, and medical imaging.

He is currently a Senior Research Scientist with HP Labs, Haifa. His main research interests include developing image processing, computer vision, and data-processing algorithms on commodity graphics cards.

Kang Zhang received the B.Eng. degree in computer engineering from the University of Electronic Science and Technology, Chengdu, China, in 1982, and the Ph.D. degree from the University of Brighton, Brighton, U.K., in 1990.

He is currently a Professor and Director of Visual Computing Lab, Department of Computer Science, University of Texas at Dallas (UT-Dellas), Richardson. He is also an Adjunct Professor of the UT-Dallas Computer Engineering Program and GIS Program. Prior to joining UT-Dallas, he held academic positions in the U.K., Australia, and China. His current research interests include visual languages, aesthetic computing, and software engineering. He has published more than 180 papers in these areas. He has authored and edited five books.

Dr Zhang is on the Editorial Boards of the *Journal of Visual Languages and Computing*, the *International Journal of Software Engineering and Knowledge Engineering*, and the *International Journal of Advanced Intelligence*.



Chunying Zhao received the B.E. and M.E. degrees in computer engineering from Nankai University, Tianjin, China, in 2002 and 2005, respectively, and the Ph.D. degree from the University of Texas at Dallas, Richardson, in 2010.

She is currently an Assistant Professor with the School of Computer Sciences, Western Illinois University, Macomb. Her research interests include software visualization, program comprehension, visual programming languages, and web services.