Diagram Analysis using Context-Based Constraint Grammars

Robert P. Futrelle and Nikos Nikolakis

Biological Knowledge Laboratory, College of Computer Science
161 Cullinane Hall, Northeastern University, Boston, MA 02115,
{futrelle,nikos}@ccs.neu.edu

phone: 617-373-2076, FAX: 617-373-5121

Technical Report NU-CCS-96-01

February 1996

1 Work supported in part by grant from the Department of Energy, Award No. DE-FG02-93ER61718.
ABSTRACT

This report presents a constraint-based grammar formalism which can be used for the specification and efficient syntactic analysis of a variety of types of complex diagrams. The significant concepts developed for this work are: the use of constraints and inherited attributes to successively delimit the objects considered until the desired solution sets are obtained, the performance of top-down analysis to effectively direct the parsing process, and the fact that objects are allowed to participate in more than one structure. The above ideas, along with the addition of set types to the grammar and spatial indexing of the data, make it possible to efficiently parse real diagrams of substantial complexity. The current work demonstrates efficient diagram parsing coupled to simple domain retargeting by writing alternate grammars. The classes of diagrams that we have analyzed include x,y data graphs and genetic diagrams drawn from the biological literature, as well as finite state automata diagrams (states and arcs).

Keywords – Diagram analysis, constraint grammars, visual parsing, spatial index
I. INTRODUCTION

In certain cases, people perceive quantitative information easier when it is presented in a graphic form [1]. In particular, visualized data sparks interest, enhances analytical thinking, saves space and becomes understood faster than corresponding text representations [2, 3]. Given the comprehensive advantages of pictorial representations, it is not surprising that non-textual data are a major part of all scientific documents. There are millions of diagrams that appear in scientific documents each year. In [4], it is stated that about one trillion statistical graphs are printed every year; 2.5 million distinct ones appear in biological papers. Newspapers regularly use statistical graphs to illustrate stock market figures, population trends, etc.

Diagrams are a critical part of graphical information and an intrinsic part of the way we think, express ourselves and understand the world around us. Since diagrams encode knowledge, computer-based methods are needed that can represent, analyze and reason about the content of diagrams. Such methods will be applicable to diagrams contained in electronic documents as well as the ones that exist in hardcopy form. The latter would require appropriate scanning and pre-processing before the analysis process.

Unfortunately, little has been done to incorporate diagrams into information systems. In current information systems, diagrams are almost always represented as bitmaps or two-dimensional arrays of gray-level values. In the future, they will most likely be represented by something more structured, ranging from graphics files to more knowledge-based representation. Incorporating structured graphics into future information systems will require progress on many fronts, including, systems that analyze store and retrieve graphic contents from a computerized knowledge base.

In this report we present a framework for efficient syntactic analysis of diagrams. Based on such techniques, computer systems could understand diagrams, starting from the graphics primitives such as lines and polygons and produce a symbolic representation of the diagram structure. The research displayed is directly applicable to documents existing in hardcopy form and provides a general framework for analysis and interaction with graphics available in electronic form. Descriptions of certain aspects of our system have been published [5-7].

Our work is closely related to grammar-based approaches for parsing of visual languages (VLs). Those approaches provide a declarative grammar-based framework suitable for modeling the structure of diagrams. Under a such scheme, the analysis of a diagram consist of identifying the exact correspondence between the diagram and the underlying grammar. The most noticeable models are the Relation Grammars [8, 9], Graphical F-PATR Grammars [10], Picture Layout Grammars [11-13], and Constraint Set Grammars [14]. While they are capable of describing a variety of domains, these systems do not appear to be efficient enough to parse diagrams of any real complexity (e.g., N=100 to 200 elements), as they are more or less based on
exhaustive techniques used to parse string languages. In addition, all the prior work we have
referred to deals with simple "toy" images instead of real ones. To provide applications of any
use, it is essential to develop systems that process a collection of typical diagrams drawn from
the technical literature [15].

Moreover, there are a number of systems that perform interpretation of engineering
drawings, circuits diagrams and maps [16]. In general, those systems use complex domain-
specific knowledge representations, making it difficult to apply the methods to different
domains.

II. DIAGRAM ANALYSIS AND VISUAL PARSING

Diagram analysis is the process of identifying the knowledge contained in a diagram. Such
analysis will enable accurate response to questions regarding the displayed data. Diagram
analysis systems can be viewed as the first step towards the development of knowledge-based
systems for large corpora.

In this work, we primarily deal with issues of syntactic analysis of diagrams. Syntactic
analysis focus on purely geometrical relations (syntax) without concerning itself with semantics.
This is particularly appropriate for diagrams because low-level graphical elements do not have
the arbitrary, yet precise meanings that words have in natural language. Given appropriate
domain information, syntactic analysis will be able to produce simple semantic descriptions.
Such a description would be the coordinates of the data-points that are present in a data graph or
the state-transition table of a finite state automaton (FSA) diagram.

The first step in the syntactic analysis is the development of specification languages for
various classes of diagrams. The next step is the identification of the underlying structure of the
picture as defined by the corresponding specification mechanism (parsing). Two-dimensional
objects can interact in more than one way and from any distance. In many cases one has to
search the whole domain in order to identify those interactions. As a result, parsing strategies
must be designed carefully in order to avoid potential complexity problems. The fact that
diagrams can be very complex makes things even worse. It is common to have diagrams that are
made up of more than 150 primitive objects. The following are the basic concepts of our
formalism:

Representation of Diagram Images: Diagrams are represented as a flat collection of graphics
primitives: lines, polygons, circles, bezier-curves and text, as might be derived from processing
the Postscript file of a document. Such representation preserves all spatial relations among the
graphical objects.

---

2 In the worst case, spatial parsing can take exponential time.
Notice that we do not address the full processing sequence starting from pixel images of hardcopy. The assumption that diagrams are available as a collection of primitives is appropriate for future electronic documents or for well-vectorized diagrams converted from scanned images.

**Specification Language:** The specification language proposed is a constraint-based grammar formalism\(^3\) that encodes the conventional aspects of diagram structure as well as the informational aspects related to perceptual grouping. Such organizations are *shared objects*, *sets*, *maximal-sets*, *aligned* objects and other *collections* of graphical objects (see Sections III and IV).

**Spatial Inference:** The addition of constraints to a grammar formalism requires the development of efficient procedures for verifying and solving these constraints. As mentioned earlier, geometrical relations are repeatedly used to define structure in VLs. Therefore, it is essential to have a system that, at least, determines various types of relations between graphical objects (e.g., *near*, *above*) and performs appropriate spatial operations (e.g., *intersection* or *union* of objects). This is called a *spatial inference* system\(^4\) as it provides constructive proofs of geometrical relations. From a practical point of view, the expressiveness of the inference engine and the efficiency of the corresponding computations play an important role in the parsing process.

The analysis system must be able to respond to three different types of queries for any particular relational constraint. Let \(r\) be a binary relation specified on the set \(D \times R\), where \(D, R\) are sets of primitive objects. The three types of queries related to the specific relation, \(r\), are defined as follows [20]:

- **Predicate Query:** Given \(a, b\) in \(D\), is \((a, b) \in r\) ?
- **Range Query (generation):** Given \(a\) in \(D\), give all \(y\) in \(R\) such that \((a, y) \in r\).
- **Domain Query (generation):** Given \(b\) in \(R\), give all \(x\) in \(D\) such that \((x, b) \in r\).

**Parsing Algorithm:** Parsing proceeds *top-down* and *depth-first*. A user can write grammar rules that lead to efficient parsing by specifying constraints that cut down on the number of elements that need to be examined or that are passed to lower rules. The solution strategy is more in the spirit of constraint satisfaction [21] than classical parsing — limited solution spaces are generated and then further restricted by the application of constraints.

### III. Spatial Indexing

Spatial inference systems are essential to any informational systems that manipulate spatial information (e.g., CAD and geographical systems). However, in the context of diagram analysis,

---

\(^3\) Constraint grammars are particularly useful in analysis of VLs [6, 17, 18].

\(^4\) Wang [19] uses the term *graphical inference* as he associates a picture description language with an "associated engine of graphical inference".
we have to rely on an inference system that fits the nature of diagrammatic representations. In the spatial system employed, all spatial relations can be queried for efficient generation of related objects.

A specific indexing method, *base language* (BL), based on a pyramid data structure presented in [5], is used to efficiently compute spatial relations between graphic objects. BL supports the efficient computation of spatial relations. The BL has a *pyramid* data structure where each level is a coarse representation of space at various degrees of resolution. Formally, a pyramid structure of *n* levels (layers) is a collection of *n* two-dimensional arrays (level\(_0\),...,level\(_{n-1}\); where level\(_i\) is an array of \(2^i \times 2^i\) cells). Each layer represents the same square area in an integer-space, nominally 8K x 8K, to which all figures are scaled for analysis purposes. Each cell is linked to its four children and to its parent cell and has references to all the geometrical objects it contains. In addition, at each level there exist two one-dimensional arrays (*projection arrays*) that contain the union of all cell contents in the corresponding row or column. Each projection cell is also linked to its two children and to its parent cell. The pyramid structure can be viewed as a multi-level spatial index that maps from spatial-positions to graphical objects.

![Fig. 1. The top three layers of a pyramid data structure that contains one object (line L). Only the references associated with the upper-right cell of the two-dimensional array and the right cell of the x-projection array are shown (both cells are specified for the Level 1). Each of those cells is linked with its parent (area) and children cells (area) and has references to all the contained objects.](image)

Each primitive graphic object is installed in every level of the pyramid structure. During a pre-computation phase (installation), each object is examined and a reference to the object is placed in any cell that is touched by or contains the object. In addition, each object has references to each of the cells it occupies and to all of its 8-neighbors (one collection of references for each level). Thus, given a point in space, all the objects at that point can be found
immediately and given an object, all nearby objects can be directly determined (the pyramid acts as a *spatially associative* index).

The BL provides efficient implementation of spatial relations such as near, inside, above, below, left, right and aligned. In particular, it is designed to answer both spatial predicates and generation questions. Spatial predicates for the inside, above, below, left, right and aligned relations can be answered by looking at the corresponding relation between their bounding-boxes. Such computation is even more efficient because every graphic object is associated with a bounding-box attribute.

![Image](n=4)

**Fig. 2.** An example that illustrates the efficient computation of the left-of relation. The objects that are left of pt correspond to the union of the cells shown in **. The arrows indicate the path followed during the search process.
Because the near and aligned relations are pre-computed during the installation phase, the near and aligned spatial generations depend linearly $O(N)$ on the number of related objects [22]. However, the efficient computation of the rest of the generation queries is a more challenging task. The spatial index can also be used to rapidly find all objects that are left, right, above or below a given object. The idea behind such algorithms is to find the minimum number of projection cells that cover exactly the desired space. In general, such cells correspond to different pyramid layers and the union of their contents gives the appropriate result (Fig. 2). The computation requires performing the union of the contents of at most $n$ cells, where $n$ is the depth of the pyramid, e.g., $n=6$. The union computation is linear in the total number of the (not necessarily distinct) objects in the $n$ cells. Variation of this procedure can be used for the computation of the above-of, below-of, left-of, right-of, between, within or outside relations for horizontal or vertical strips of rectangles. In general, the running time of a spatial generation depends on the number of related objects and the number of pyramid cells used in the corresponding computation.

The BL not only provides an efficient implementation of spatial predicates but also gives us the ability to efficiently generate sets of graphical objects specified by spatial relations. The power of spatial index is that it directly implements the "spatial storage of information" paradigm alluded to in [1]. Information about objects is literally stored in space. So if the system is looking at a certain small set of cells, it can discover relations about them that are stored there.

IV. THE GRAMMAR MODEL

The formalism employed, context-based constraint grammars (CCGs), is an attributed and constraint-based context-free grammar model. A CCG is a collection of rules. Each rule consists of a production (LHS $\rightarrow$ RHS) with a left-hand-side symbol (LHS), one or more right-hand-side (RHS) symbols, and a body that contains constraints on the RHS symbols (constituents). Constraints specify spatial relations among constituents or refer to a variety of geometric properties such as position, shape, etc.

Constituents are ordered, as specified in the body of the corresponding rule. In addition, CCGs are augmented with set-rules. Such rules relate sets of objects of a specific type. LHS symbols of set-rules correspond to objects that consist of a collection of graphical objects of a specific type. Appropriate constraints are defined that operate across entire sets of objects, such as requiring that all the objects in a set be horizontally aligned, or connected. These relations allow the parser to rapidly collect together large sets of related items, reducing the effective size of the problem. Although many subsets of a set may satisfy a relation, our algorithm chooses the
maximal set (*maximality principle*\(^5\)). The spatial index is used to make the computation of the constraints more efficient.

The attributes employed by a CCG can be distinguished in two classes: *synthesized* and *inherited* as defined in [25]. Inherited attributes, while they do not add more power into the grammar formalism, are suited for simple semantic specification of complex relations among nodes of the parse tree. With appropriate use of inherited and synthesized attributes "the value of any attribute of any node in the derivation tree depend in any desired way on the entire tree" [25]. Therefore, inherited attributes can be used to pass references to a particular node, in other nodes in the parse-structure. When the synthesized attributes of a LHS symbol depend only on the synthesized attributes of the corresponding RHS symbols, and the inherited attributes of a RHS symbol depend on either the inherited attributes of the corresponding LHS symbol, or the inherited and synthesized attributes of the preceding constituents, the attribute values can be computed for any derivation tree of every well-structured grammar [22].

Terminal and non-terminal symbols are required to contain the attributes *object* and *context* respectively. The *context* attribute, in general, specifies the set of primitive objects that have to be considered when generating candidates for the corresponding constituents. Thus, the already computed constituents can be used to reduce the space needed to be searched for the computation of unidentified ones, by altering appropriate context attributes. Therefore, when looking for data-points in a data graph, one can concentrate on objects that are above the corresponding x-axis and when looking for automata states, one should only consider objects that are near the arrow endpoints. The *object* attribute can be seen as a non-attributed representation of the corresponding spatial object and is used in the computation of the relevant spatial relations. The synthesized attribute *object* is defined for every terminal symbol, while the inherited attribute *context* is also associated with the start symbol. If the context attribute of a RHS is not specified, it is inherited from the corresponding LHS symbol. Moreover, a RHS inherits all the inherited attributes of the corresponding LHS (it is up to the writer of the grammar to avoid name conflicts).

![Fig. 3. When looking for automata state-circles, one should only consider objects that are near arrow endpoints.](image)

---

\(^5\) The Maximality Principle is a direct consequence of the *Minimum Principle* states that the preferred interpretation of a pattern is the simplest description of that pattern [23, 24].
Each node of a derivation tree is associated with attribute-value pairs for all the inherited and synthesized attributes of the corresponding symbol. When the values of the object attribute are represented once per analysis, derivation-trees correspond to DAGs as two distinct nodes can share the same value for the object attribute (e.g., an graphic object that is a solution of two nodes with different context attribute values). As a result, graphical objects can be used (shared) more than once per derivation. The nature of graphical forms makes sharing an significant issue in diagram analysis. Consider for example, the syntactic analysis of the FSA shown in Fig. 9. The state that corresponds to the circle labeled by b is part of four different structures, the four transitions that start or end from it. This holds, assuming that the grammar contains the following productions: Transition $\rightarrow$ State-1 Arrow State-2; State $\rightarrow$ circle.

Solutions of a rule are generated by finding tuples of constituents in the diagram that satisfy all the constraints in the body of a rule. For each solution, one LHS object is created with the corresponding constituents. Each LHS object has full status as a graphical object, with region, bounding box, center, etc., so it can participate as a constituent in other rules.

![Fig. 4. The x-axis part of a data graph.](image)

In the next paragraphs, an example of a sample CCG grammar is presented which can be used to identify the x-axis in data graphs (see Fig. 4).

**Rule-1:**

```
X-Axis $\rightarrow$ X-Axis-Line X-Ticks X-Labels X-Text 
    (:optional X-Text) 
    (X-Axis-Line (X-Line (get-val axis))) 
    (X-Ticks (touch X-Axis-Line ?) 
              :constraints (>= (size X-Ticks) 2)) 
    (X-Labels (below ? X-Axis-Line :strip t)) 
    (X-Text (below-nearest ? X-Labels));
```

**Rule-2:**

```
X-Ticks $\rightarrow$ Set ( Line ) 
    (:element-constraints (vertp Line) (short Line)) 
    (:constraint horiz-aligned));
```

Rule-1 specifies an X-Axis object as a structure that consists of the axis-line (X-Axis-Line constituent), the tickmarks (X-ticks constituent), the labels (X-labels constituent) and the corresponding axis-text (X-text constituent). The X-Text constituent is an optional one, and as a
result, the rule can be satisfied in the case where a corresponding text does not exist. The X-Axis-Line is set directly to the corresponding line (obtained through the inherited attribute axis which is specified in a rule not shown above). get-val is the accessor function for the inherited attributes. The context attribute of the X-ticks constituent is restricted to the objects that are near the X-Axis-Line, and so on. In every case, the context-specification is printed in bold format.

Rule-2 specifies the X-ticks as a maximal sets of horizontally aligned, vertical and short lines. As no context is specified, each set member (Line) should be a member of the value of the context attribute of the LHS symbol (X-ticks), which, as specified in the X-Axis rule consists of all objects that are close to the corresponding axis-line.

V. PARSING

The parsing process can be divided in two phases. In the first phase, all derivation trees of the particular symbol in CFG(G) (the corresponding non-attributed context free grammar) are generated. Then, for each derivation tree the corresponding solutions are determined. In general, a derivation tree of CFG(G) corresponds to many derivation trees of G as the object attribute of the leaf nodes ranges over the domain of image tokens. A derivation tree of a CFG(G) is seen as a hierarchy of CSPs where a distinct CSP corresponds to the rule associated with every node in the derivation tree. In such scheme, different instances of the same problem may appear at different places in the constraint hierarchy. The hierarchy is traversed in depth-first fashion: top to bottom and left to right. As we proceed from top to bottom, the values of the inherited attributes of the corresponding LHS and RHS symbols are determined. At every node, solutions to the corresponding CSP are generated from left to right. Every partial constituent assignment invokes a corresponding CSP instance that returns the candidate set for the next assigned constituent. At the end of such calls, we find instances of terminal CSPs. Those CSPs, when reached, impose the restrictions applied on the context attribute of the corresponding terminal symbol.

Information contained in the values of inherited attributes is passed down the search tree (see Fig. 5). Thus, when a CSP instance is invoked, this information is used to direct the parsing process. Context information is propagated to the terminal CSPs where it is used to generate limited solution spaces. Those spaces are then further restricted by the application of constraints.

In general, the analysis is efficient because of the continued restriction of the context as it is passed down the search tree. Parsing a grammar that does not restrict the context attribute for any constituent, in a sense, involves a complete search in the space of all possible solutions. However, in most visual languages, one can identify particular spatial relations that can be used in the restriction of the context attribute. After all, visual languages of interest contain human
produced drawings with well defined semantics. Therefore, a user can write grammar rules that lead to efficient parsing by specifying constraints that cut down on the number of elements that need to be examined or that are passed to lower rules. In this case, the usual combinatorial explosion met in visual domains is tamed a great deal by adopting the hierarchical view. The hierarchical view factors the problem into a set of small, interdependent problems that can be solved efficiently through top-down analysis.

**Fig. 5.** In the top-down parsing strategy, inherited attributes are passed down the search tree, while synthesized attributes are passed up. For example, the context attribute is passed down as an inherited attribute and the bounding-box is passed up as a synthesized one.

Set-rules while expressive, can lead to exponential complexity. The enforcement of the maximality principle and the use of specific spatial indexing methods enable us to efficiently parse set-rules. To parse a set-rule, all objects within the current context are identified and filtered according to the specified type and element constraints. Then, the set is searched for maximal sets of related objects. In the case of near and aligned relations, sets of related objects can be identified in linear time on the size of related objects [22]. Such ability along with appropriate context restriction makes the efficient parsing of CCGs that contain set-rules possible.

The sample grammar given earlier illustrates our strategy:

- In Rule 1, "X-Axis-Line" appears first in the body, so it is processed first. It refers to X-Line attribute of the its axis attribute.
- A solution space for X-Axis-Line is generated which consists of all corresponding X-Lines. There is one such line in Fig. 4, leading to one potential solution.
- For each X-Axis-Line solution, the X-Ticks rule, Rule-2, is entered, inheriting the context attribute determined by the form "(touch X-Axis-Line ") in Rule-1. The value of context...
in this case, the value of "?", is all graphical objects which touch the given X-Axis-Line. Rule-2 states that every X-Ticks solution is a set, in this case a set of Lines. The Lines must be drawn from the set of objects in the context inherited by Ticks. The constraints on each member of the set are that they are vertical and short. The constraints on each set as a whole is that the elements are horizontally aligned with one another. There is one such set in Fig. 4.

- The processing returns to Rule-1 where the size constraint is imposed. Because the given line set satisfies the size constraint, the corresponding partial solution is appropriately extended.
- For each of the remaining constituents, partial assignments are extend based on the corresponding context specifications. In the example given, one solution is identified.

VI. EXPERIMENTAL RESULTS

A prototype system has been implemented based on the framework presented. The system is written in Macintosh Common Lisp and uses the CLOS object system extensively. In this research, we pay special attention to the analysis of real diagrams. For this reason, all of our experiments were performed on actual diagrams from the scientific literature. A large number of diagrams is available on line thanks to the BKL Scientist's Assistant project [26]. So far, the diagrams appearing in 132 biological papers have been scanned and traced over on the screen using Canvas and are available in electronic form (Postscript files). The diagrams appearing in those biological papers (a total number of 660) consist our working database. All examples of diagrammatic representations given in this report are taken from this database. A pre-processing step is required to transform the diagram-files from Postscript to the form required by the analysis system (a file of S-expressions).

Working in the context of BKL Scientist's Assistant project, we have concentrated on the analysis of diagrams from the domain of data graphs and gene diagrams. We also worked on the domain of finite state automata (FSA), a well-understood domain with a completely different structure from the domains of data graphs and gene diagrams. The current system is capable of analyzing a variety of real diagrams. The grammars used for the syntactic analysis of theses domains can be found in [7, 22].

The efficient analysis of data graphs through CCGs is a result of the regularity of their background components [27] and appropriate restriction of the context attribute. For example, the data graph in Fig. 6 (N=88) required 28 sec to be analyzed (plus 41 sec for precomputing the spatial index, seven layers). In Fig. 7, the same diagram appears while the world-coordinates

---

6 The biological papers were selected by biologists as a broad representation of the field of bacterial chemotaxis.
7 All the analysis examples were performed on a Macintosh Quadra 700.
and the perpendicular lines from two selected data-points to the axis have been drawn through the corresponding post-processing. Another example of syntactic analysis is given in Fig. 8. The corresponding diagram (N=144) was analyzed in 45 sec plus 72 sec for precomputing the spatial index (seven layers). Once more, the world-coordinates and the perpendicular lines from two selected data-points to the axis have been drawn through the corresponding post-processing.

![Figure 6](image1.png)

**Fig. 6.** An example of a data graph taken from [28].

![Figure 7](image2.png)

**Fig. 7.** The world-coordinates and the perpendicular lines from the two selected data-points to the diagram axes have been drawn by post-processing of the syntactic analysis of the above diagram.
Fig. 8. The world-coordinates and the perpendicular lines from the selected data-points to the diagram axes have been drawn by post-processing of the syntactic analysis of the diagram.

Fig. 9. An example of a complex finite-state automata sketch which the system can easily parse. Parsing the diagram (N=124) required 69 sec plus 41 sec for precomputing the spatial index (7 layers).

Diagrams from the two other domains have been also parsed with our system. For example, parsing the FSA diagram in Fig. 9 (N=124) required 69 sec plus 41 sec for precomputing the
spatial index (7 layers). The fact that the arc-ends and arrowheads were drawn roughly and did not line up accurately posed no problems for the parser because the near constraint was used. The arrows are recognized from their constituent lines, rather than assumed as primitives. In addition, there are many graphical objects that are shared within the returned FSA structure. For instance, each circle and a-state object is directly a part of each different transition that leaves or reaches the corresponding state.

Simple post-processing of the parse gives the entire state-transition table so that the FSA can be run. In particular, one can write a domain specific function (dependent on the FSA grammar) that traverses the parsing output and produces the formal specification of the given FSA. A post-processing of the analysis of the above diagram results in the following interpretation (the obtain-fa function prints the inferred specification).

> (obtain-fa (parse 'fa))
The formal specification of the given FA is:
FA = < Q=(a b c d e f g h i k),
    S=(0 1),
    g0=a,
    F=(d h k),
    D=((f 0 g) (e 0 d) (e 1 f) (f 1 e) (g 0 f) (g 1 g) (h 0 g) (a 0 b)
      (b 0 a) (d 1 a) (a 1 a) (c 1 b) (b 1 c) (c 0 d) (d 0 d) (d 0 i)
      (i 0 i) (i 1 k) (k 1 e) (k 0 e) (h 1 d))>

VII. CONCLUSIONS AND DISCUSSION

Viewing visual parsing as a constraint satisfaction process provides a coherent framework for object sharing and efficient solution strategies. The grammar formalism supports both inherited and synthesized attributes. As a result, relations among objects that are distant in the parse tree can be easily specified. The BL and the associated spatial algorithms support the idea of context-basis analysis because searches can be efficiently confined to areas of interest and geometric relations can be computed rapidly. Although the installation process is computationally expensive, it is domain independent. As a result, the corresponding pyramid data-structure can be considered as an alternative representation of the input diagram. Due to the nature of the application domains, we have mostly focused on deterministic analysis of non-recursive grammars. However, we have also experimented with recursive CCGs and the analysis has been successful in the case where context is restricted at each recursive call.

The approach pursued here combines the flexibility of domain retargeting (by writing alternate grammars) with the efficiency of more domain-specific systems. Such a system can provide the basis for a general information retrieval system which can be used to retrieve the knowledge associated with diagrams, and ultimately respond to queries about the displayed data. The analysis system appears to succeed because of the following features:
• Matching all aspects of the system to the types of organization that people perceive in diagrams and use in drawing diagrams.

• Building spatially associative indexes of all primitives and derived objects so that searches can be efficiently confined to areas of interest and geometric relations can be computed rapidly.

• Using sets as a fundamental component of the grammatical formalism.

• Using equivalence relations (*near, aligned*) to partition object collections into classes, typically in linear time.

• Using constraints to successively restrict the objects considered until the desired solution sets are obtained.

• Performing top-down analysis to effectively direct the parsing process.

• Allowing both inherited and synthesized attributes.

• Allowing objects to participate in more than one structure, e.g., a shared wall between two rooms.

ACKNOWLEDGMENT

This research was supported in part by the Department of Energy, Award No. DE-FG02-93ER61718.

REFERENCES


