Appeared in IEEE Symposium on Visual Languages, VL99, Tokyo, 13-16 September 1999.
http://dlib.computer.org/conferen/vl/0216/pdf/02160172.pdf

Ambiguity in Visual Language Theory and its Role in Diagram Parsing

Robert P. Futrelle
Biological Knowledge Laboratory, College of Computer Science 161CN
Northeastern University, 360 Huntington Ave., Boston, MA 02115
futrelle@ccs.neu.edu http://www.ccs.neu.edu/home/futrelle

Abstract

To take advantage of the ever-increasing volume of diagrams in electronic form, it is crucial that we have methods for parsing diagrams. Once a structured, content-based description is built for a diagram, it can be indexed for search, retrieval, and use. Whenever broad-coverage grammars are built to parse a wide range of objects, whether natural language or diagrams, the grammars will overgenerate, *giving multiple parses. This is the ambiguity problem. This paper discusses the types of ambiguities that can arise in diagram parsing, as well as techniques to avoid or resolve them. One class of ambiguity is* attachment, *e.g., the determination of what graphic object is labeled by a text item. Two classes of ambiguities are unique to diagrams:* segmentation *and* occlusion. *Examples of segmentation ambiguities include the use of a portion of a single line as an entity itself. Occlusion ambiguities can be difficult to analyze if occlusion is deliberately used to create a novel object from its components. The paper uses our* context-based constraint grammars *to describe the origin and resolution of ambiguities. It assumes that diagrams are available as vector graphics, not bitmaps.*

1. Introduction

Diagrams are used to illustrate complex relations, to present data, to document designs, and to otherwise provide schematic views of information. In the future, virtually every document of importance, and all the diagrams they contain, will be available in digital form. In order to index, search, and manipulate digital diagrams, it is important that we develop automated procedures for parsing and analyzing them. As techniques for analyzing diagrams are scaled up to deal with a greater volume and variety of diagrams, there will be many challenges. One of these is ambiguity, the subject of this paper. Our own work has focused on diagrams drawn directly from the published research literature, e.g., biology journals. Our emphasis, and that of this paper, has been on diagrams that are highly schematic, such as data graphs, rather than drawings.

All communication is potentially ambiguous, whether it uses natural language or graphics. Most speakers and viewers resolve potential ambiguity problems quickly and almost unconsciously, selecting the "preferred reading" by using a broad collection of strategies based on context, conventions of communication, and domain-specific knowledge. Broad coverage grammars inevitably *overgenerate*, producing many alternate analyses in cases that are otherwise clear and unambiguous to a human.

The resolution of ambiguities in natural language parsing is a difficult and unsolved problem [4]. As this paper will demonstrate, diagrams are replete with a variety of ambiguities, many of them as subtle and difficult to resolve as those in natural language. Visual language parsing systems on the whole have not tried to deal with the ambiguity issue. Even a recent authoritative review of visual language analysis contains no discussion of ambiguity [6]. Though there can be no single strategy that will solve the diagram ambiguity problem, we do discuss techniques that can help to resolve specific classes of ambiguities. The techniques include choosing grammar rules to apply in context-specific ways, using preferences based on norms, choosing minimally complex descriptions, designing grammars to reflect graphics conventions, and examining surrounding text, e.g., captions. (In this paper we will not discuss any details of ambiguity resolution that requires the analysis of text.)

2. An example diagram

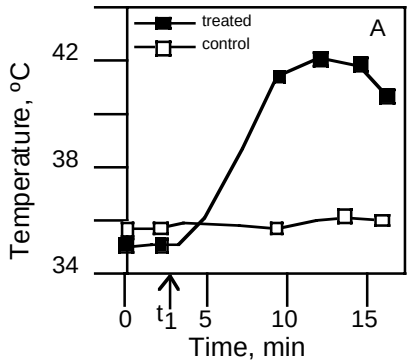

Figure 1. This x,y data graph illustrates a number of ambiguities that are discussed in the text. One of the most obvious ones is the analytic ambiguity raised by the fact that the data key at the top, labeled "treated" and "control" looks very much like actual data, but is not. Others include the occlusion of data lines and the y-axis by data points, and the problem of associating various labels with the items or structures that they label.

3. Plan of this paper

We first discuss the general nature of parsing as a grammar-driven constraint satisfaction problem. Then, the various types of ambiguities that can arise in diagrams are enumerated. Some of these ambiguities can be resolved by the local context, others require information from a broader context. We discuss the two major classes: *lexical* and *structural* ambiguity. A primary type of structural ambiguity is *attachment*, e.g., the association of a label with its correct referent. The other major class of structural ambiguity discussed is *analytic ambiguity*. In these, the categorization of a structure is itself in doubt, e.g., is a short vertical line at the x-axis of a data graph an error bar or a tick mark? The composition of elements, including splitting, joining, and occlusion also can create structural ambiguities. The final Discussion section covers issues such as knowledge-based disambiguation, future graphics standards, metadata for diagrams, and intelligent authoring systems that should reduce the need for complex diagram analysis systems.

4. Formulation of the parsing problem

- A diagram is vector-based, made up of two-dimensional primitives such as lines and curves that are open or closed (e.g., polygons), and positioned, oriented text.
- All constituents have attributes whose values describe their geometrical and logical properties. For primitives, attributes include such properties as line widths, region color, or fill pattern, and layer order which can lead to occlusion.
- The allowable interpretations of a diagram are specified by a grammar in which each production describes a constituent, its members, and constraints over the member attributes. Some of the constraints are geometric, e.g., near, or horizontally aligned. Other constraints may be higher-order, e.g., that all members of a set-valued constituent are short vertical lines.
- In general, the result of the analysis is a graph of constituents, not a tree.
- If the constraint system has more than one solution for a given diagram, then the diagram is ambiguous with respect to the grammar. Proper design of the grammars or subsequent analysis of the multiple solutions are used to reduce or eliminate ambiguities.

The formulation we have used in our diagram parsing work is the *Context-based Constraint Grammar* [1, 2]. One production drawn from a working grammar for x,y data graphs is shown below [1]. It defines an `Axis` in the conventional way as made up of two perpendicular lines such that the left endpoint of the `X-Line` touches the `Y-Line`. `left-endpoint` is an attribute of a line. `X-Line` and `Y-Line` are defined in turn by their own rules (not shown). Our parser examines the constraints in the body of the rule sequentially, so if the `touch` constraint is satisfied for a set of `Y-Lines`, the parser then enforces the more demanding `coincide` constraint to further narrow the set of legal `Y-Line` candidates.

```
(Axis -> X-Line  Y-Line
   (X-Line)
   (Y-Line
      (touch
         (left-endpoint X-Line) '?)
   :constraints
      (coincide
          (left-endpoint X-Line)
          (bottom-endpoint Y-Line))))
```

Other rules may be set-based, e.g., ones involving highly repeated elements such as data points or gene segments.

The diagram grammars that we and most others have developed are often formulated in domain-specific terms, e.g., using objects such as `Axis` and `Data-Points` for data graph grammars. This is in contrast with the more abstract syntactic formulations used for natural language. The reason for such domain-specific grammars is that the totality of diagrams breaks down naturally into a collection of classes, e.g., the classes we have studied: x,y data graphs, linear gene diagrams, and finite-state diagrams. We will argue below that more abstract approaches are possible for diagram parsing.

5. The classification of ambiguities

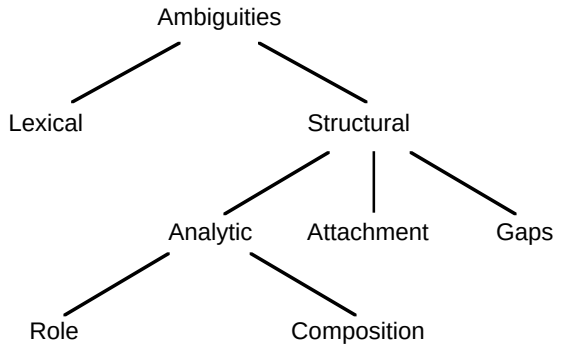

Figure 2. The relations between the classes of diagram parsing ambiguities that are discussed below. These closely parallel the ambiguities found in natural language [4]. Lexical ambiguities involve alternate senses for simple items. Structural ambiguities involve generating alternative parse structures for a given collection of elements.

6. Lexical ambiguity

The lowest level constituent normally considered in natural language analysis is the *lexeme* (roughly, the word). In diagrams, there is nothing that is strictly analogous to a lexeme, but it is useful to consider certain classes of items, which we will call *graphemes*, from this point of view. A grapheme may be ambiguous, just as a word may.

As an example, consider the arrow or directed line as an ambiguous grapheme (a homonym or homograph). An arrow has (debatably) three distinct senses:

- Vector (with position, orientation, and magnitude)
- Transition (as in a finite-state diagram)
- Designator (pointing to an object)

One method for resolving ambiguity is to design the grammar so that it only expects a single sense within a restricted context. In Fig. 1, the arrow below the x-axis is a designator for a time, t_1. The grammar fragment below that focuses on the appropriate context is a production, `X-Axis`, that includes an `X-Annotation` constituent that in turn includes a labeled arrow [1]. At the point at which the `X-Annotation` constituent is introduced in the `X-Axis` production, it is constrained to be drawn from a set of elements that lie within 700 units of the `X-Axis-Line`, but with all items comprising `X-Ticks` and `X-Labels` excluded by a set difference operator:

```
(X-Annotation
  (difference*
     (near X-Axis-Line 700)
     (union* X-Ticks X-Labels)))
```

(The "700 units" referred to above are the normalized units used in our Diagram Understanding System [2].)

Another way to resolve arrow ambiguities is to look at the objects near the ends of the arrow and adjacent to the shaft. For example, if the object at the tail is text, and the object near the head is not, the arrow is most likely a designator. If the objects near the two ends are of the same type, e.g., both are labeled rectangles, then the arrow sense could be a transition or it could be a vector indicating flow, such as packet flow in a network. The resolution of this type of ambiguity would require examining the broad topical context or specific text describing the diagram.

7. Attachment ambiguity

There are fourteen text labels in Fig. 1, which despite their apparently obvious referents could cause difficulties for a parser. For example, "Time, min" might apply only to the 5 to 10 region of the x-axis, analogous to the role of "t_1", which applies only to the arrow above it. Adjacency and alignment properties of labels are sufficient to resolve most ambiguous attachments for conventional diagrams. The "**A**" label in Fig. 1 might label the region of the data maximum or the diagram as a whole (contrasting with other diagrams: **B**, **C**, ...). This ambiguity would have to be resolved at a high level in the parse graph or by reference to text descriptions.

An interesting example of an attachment ambiguity arises for repeated patterns, as shown in Fig. 3. One solution to the ambiguity problem in Fig. 3 is to *prefer* assignments that lead to a solution of *minimal complexity* [5], one in which each of the five items in the label sets is paired with exactly one item in the tick set, leaving no item unpaired. Minimal complexity is a powerful principle that is useful in all aspects of diagram parsing [2]. It is a well-known technique in pattern recognition. Preference approaches are also used extensively in natural language disambiguation strategies.

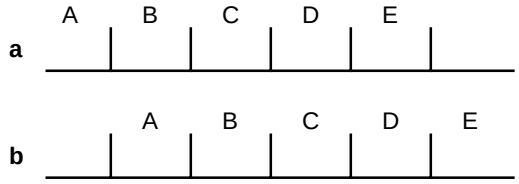

Figure 3. In this figure the intent is that each character labels one tick mark. It is obvious to a human that in **a**, the characters label the ticks to their right, and in **b**, the ticks to their left. This is in spite of the fact that the characters are placed midway between the ticks, so that nearness alone is not a decisive criterion.

8. Gaps

In natural language, when an element is omitted, creating a gap, the presumption is that the reader will choose the correct filler. Gaps are common in diagrams, e.g., the labels that are omitted for three of the y ticks in Fig. 1. Fortunately, there appear to be few if any *ambiguous* gaps in common diagrams.

9. Analytic ambiguities

9.1. Role determination

In natural language analysis, the role of a constituent may be ambiguous, e.g., in "I went to the store in her car.", "in her car" can attach to the verb "went" (correct) or to "store" (most probably incorrect!). An example of this for diagrams appears in Fig. 1 for the data key at the top of the figure, which has the appearance of data, but which is in fact a key. Virtually without exception, such data keys involve minimal graphics showing data point shapes or data line styles, each labeled with text and arranged in a tabular layout. These constraints are not easy to specify in a grammar and may best be handled in a separate disambiguation step.

9.2. Composition

9.2.1. Segmentation. In Fig. 1, the x and y axes appear to extend in the lower left corner to form y and x tick marks respectively. A good deal depends on exactly how the diagram was actually constructed: All tick marks might be short lines, or the tick marks at the lower left might be formed from the ends of the long axis lines, a potentially ambiguous construction. The former case presents no problems for parsing. For the latter, one approach would be to define the ticks in terms of the alignment of their outer termini which is not difficult in our approach, because line termini are instantiated as distinct objects before parsing begins. Another would be to allow the parser to hypothesize segments of long lines as short lines. This would lead to serious overgeneration, since two adjacent parallel lines

could then have any number of aligned equal length segments.

Since any figure that appears visually simple could be constructed from an arbitrary number of subparts, the parsing problem might seem insuperable. At this point we have to assume that diagrams obey certain well-formedness constraints, following Grice's *Cooperative Principle* for communication [3].

9.2.2. Occlusion. In Fig. 1, the data point markers (the small filled and unfilled squares) occlude the data lines in certain places, and even part of the y axis. In most diagrams, line objects such as the y axis are not altered by the presence of items that exist in a distinct overlying layer, so no ambiguity exists.

A difficult class of occlusion problems arises when the author of a diagram actively employs occlusion to create a complex object, a process we call *synthetic occlusion*. This is often done for author's convenience or to overcome some limitation of the diagram creation application. In Fig. 4 the title box at the top of the object could be created as a small rectangle that is then carefully aligned to abut the top of the larger rectangle. A simpler and quicker way to do this would be to create two rectangles, one overlaying the other as shown in the figure.

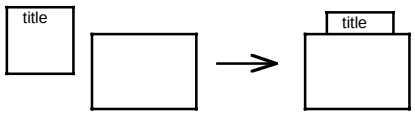

Figure 4. In *synthetic occlusion*, occlusion is used to create a visually distinct object. Most of the underlying small square is meant to be ignored in the analysis. Only the small visible rectangular portion of it that contains the label is relevant.

One problem with synthetic occlusion is that it may produce complex objects that are outside the range of the grammar, e.g., an object whose boundary is a mix of straight and curved segments.

The phenomenon of occlusion has also been discussed in studies of the logic of diagrammatic reasoning [7].

10. Discussion

This brief paper has attempted to catalogue the major ambiguities met in diagrams, as well as methods such as context restrictions and minimal complexity to resolve them.

The more difficult ambiguity problems must be solved using knowledge-based methods. These will involve more knowledge of the domain and its conventions as well as an analysis of the text within the diagrams, in captions, and in discussions of the diagrams in the body of the document.

For diagram parsing to succeed on a large scale, the representations chosen for diagrams must be matched by the design of the grammars used for their analysis. This is not just a matter of graphics file formats, but involves issues such as how lines are segmented, how rectangles are represented, etc. (Sec. 9.2.1). The pressure to achieve a uniform format will be intense, because all providers of diagrams will want their diagram contents indexed and available for search and use.

Though the Web is dominated by pixmap formats today (GIF and JPEG) there is a major effort to develop a vector standard for the web, SVG [8]. We predict that such vector representations will dominate in the future.

The analysis of a diagram ultimately leads to the production of *metadata* describing diagram content. In our view, the proper way to proceed in the future is not to develop more sophisticated analysis procedures, but to develop better authoring tools, *intelligent authoring systems*, that allow the human or machine creator of a diagram to embed the appropriate metadata in the diagram at its inception.

One revelation of the work here can be found in our assertion that even objects apparently as general as an arrow have a limited number of senses. We assert that the same is true of even simpler entities such as lines and rectangles -- they have a limited number of senses. Many think that an isolated rectangle is entirely without meaning; we do not now think that this is true. This insight will allow us to build more purely syntactic theories of diagram parsing. It will allow us to develop more concise characterizations of ambiguity in diagram parsing and open the way to more systematic methods for resolving ambiguities.

Acknowledgements. The anonymous reviews of the original draft were quite helpful. Thanks to the ERC for a productive working environment and to my wife, Carolyn Scott Futrelle, for her skilled editorial assistance.

References

[1] R. P. Futrelle, "The Diagram Understanding System Demonstration Site," *http://www.ccs.neu.edu/home/futrelle/diagrams/demo-10-98/*, 1998.

[2] R. P. Futrelle and N. Nikolakis, "Efficient Analysis of Complex Diagrams using Constraint-Based Parsing," presented at ICDAR-95 (Intl. Conf. on Document Analysis & Recognition), Montreal, Canada, 1995.

[3] H. P. Grice, "Logic and conversation," in *Syntax and Semantics 3: Speech Acts*, P. Cole and J. Morgan, Eds. New York: Academic Press, 1975, pp. 41-58.

[4] G. Hirst, *Semantic interpretation and the resolution of ambiguity*. Cambridge: Cambridge University Press, 1987.

[5] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag, 1993.

[6] K. Marriott, B. Meyer, and K. Wittenburg, "A Survey of Visual Language Specification and Recognition," in *Visual Language Theory*, K. Marriott and B. Meyer, Eds.: Springer Verlag, 1998, pp. 5-85.

[7] D. Wang, J. Lee, and H. Zeevat, "Reasoning with Diagrammatic Representations," in *Diagrammatic Reasoning. Cognitive and Computational Perspectives*, B. Chandrasekaran, J. Glasgow, and N. H. Narayanan, Eds. Menlo Park, CA, Cambridge, MA: AAAI Press, MIT Press, 1995, pp. 339-393.

[8] WWW-Consortium, "SVG -- Scalable Vector Graphics," *http://www.w3.org/Graphics/SVG/*, 1999.