

Summarization of Diagrams in Documents

Robert P. Futrelle

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

Documents are composed of text and graphics. There is substantial work on automated text summarization but almost none on the automated summarization of graphics. Four examples of diagrams from the scientific literature are used to indicate the problems and possible solutions: a table of images, a flow chart, a set of x,y data plots, and a block diagram. Manual summaries are first constructed. Two sources of information are used to guide summarization. The first is the internal structure of the diagram itself, its topology and geometry. The other is the text in captions, running text, and within diagrams. The diagram structure can be generated using the author's constraint-based diagram understanding system. Once the structure is obtained, procedures such as table element elision or subgraph deletion are used to produce a simpler summary form. Then automated layout algorithms can be used to generate the summary diagram. Current work on parsing and automated layout are briefly reviewed. Because automated diagram summarization is a brand-new area of inquiry, only the parsing phase of the approach has been fully implemented. Because of the complexity of the problem, there will be no single approach to summarization that will apply to all kinds of diagrams.

1. Introduction

Documents are composed of text and graphics. But virtually all summaries and excerpts of documents available today are in text form exclusively. Including graphics in summaries would address this imbalance, but raises obvious questions as to how such graphics summaries could be generated. There is essentially no literature on the summarization of graphics. This paper is a first foray into this unexplored territory. It focuses on *diagrams*, which are line drawings such as data plots or block diagrams.

The overall plan of this paper is to first present a set of four examples of diagram summarization, all constructed manually, using diagrams from the science and engineering literature. Since the summaries are constructed manually, they represent, in a sense, the ideal summarization that we would like an automated system to achieve. We then go on to discuss what would be involved in automating the process of summarizing diagrams, illustrated with the same four example diagrams. As part of this we discuss the computational results we have achieved so far on

extracting structural information from diagrams by constraint-based parsing. The four diagrams have been chosen to represent a cross-section of the types of problems that arise in diagram summarization.

Assuming that structural descriptions can be obtained, either from parsing or as metadata furnished by the author of the document, the problem then becomes one of *selecting* one or a few figures from a document, or *distilling* a figure (simplifying it), or *merging* multiple figures. The last two approaches require the generation of a new figure that did not previously exist, so results in the fields of automated layout and graph drawing are discussed to see how those methods can be brought to bear on distillation and merging. The overall process then is one of *analysis* to develop structural descriptions, the production of *summaries of the descriptions*, and finally the *generation* of the graphical form of the summary diagram.

Throughout we assume that the diagram of interest is vector-based, as contrasted with a raster image. Summarization of raster images involves an entirely different set of problems including image processing and segmentation, topics not treated here.

Our work on diagram summarization so far has been to develop systems that can automatically discover diagram content by visual parsing. It is this content that must be analyzed, along with related text, to produce summary figures that then have to be laid out and displayed. Because we have not yet developed automated systems to produce reduced or summary versions of diagram content, our discussion has to be viewed as an initial exploration of the nascent field of diagram summarization. It attempts to lay out the research and implementation problems that must be solved in order for working systems to be built. We hope that this paper serves as both a catalyst and a guide for this new domain. One inspiration for this work is the rapidly increasing availability of electronic documents on the World-Wide Web and elsewhere as well as the profusion of formats, such as PDF (Adobe Systems Incorporated, 1996) for whole documents, PGML for vector graphics, XML (World-Wide Web Consortium, 1998) for metadata, that make it possible to access and manipulate the contents of these documents for information extraction and summarization. The availability of these documents and the needs of users to access them efficiently motivates the current work on summarization in general, and this paper on diagram summarization in particular. A

useful discussion of the structure needed to fully exploit scientific documents appears in (Fateman, 1997), with additional discussion and web links at <http://http.cs.berkeley.edu/~fateman/MVSD.html>. Fateman uses mathematical notation as his example and discusses the problems of representing visual appearance, syntax (structure), and semantics, all issues that have to be dealt with in diagram summarization.

Our work on diagram parsing and this discussion of summarization focuses on the scientific and technical literature. The figures in such documents tend to be strongly information-bearing, as contrasted with figures in newspapers and magazines that are often more topical and indicative, e.g., a picture of some unidentified people applauding at a concert.

At first blush it might seem that the text accompanying figures could be exploited to guide diagram summarization, but as we explain below, this is often not the case — the diagram itself must be analyzed. This means that we cannot always take advantage of lexical resources and text statistics that are so helpful in text summarization.

2. Manually Constructed Diagram Summarization Examples

2.1 Example #1: Distilling a table of images

The first example is drawn from a paper about OCR (Ho and Baird, 1997) that has eleven figures in it, many multi-part and one occupying an entire page. The paper is about building parameterized defect models for the appearance of characters and training systems on very large collections of characters generated by the models, and assessing their performance. Some of the figures show the degraded characters that are synthesized, but most show statistical results of their analyses. An informative summary should probably include one figure that shows examples of the degraded characters. The figure we have chosen is their Fig. 7, shown in our Fig. 1a below. The running text in the article offers the following support for this decision,

"...The key contributors to scanning and printing noise are the three parameters blur ..., thrs ..., and sens These will be our primary concern. Figure 7 illustrates these effects."

The choice of this figure is a *selection process* but we will not dwell on that process at this point. On examining the figure, we can see that it is possible to reduce it to a more compact summary form, by manipulating its graphics content and its caption. The graphics content, with ten sample images in each of three rows, offers more detailed information than is necessary in a summary. The summarized form, shown in Fig. 1b, elides the eight intermediate examples, and indicates them with ellipsis. In summarizing the caption, the detailed numerical values can be omitted, giving the caption in Fig. 1b. This collapsing of the set of graphical items is a *distillation process*.

Example #1 does include *images* in tabular positions, but our approach to summarization for this example treats them as anonymous items, with no internal structure.

In Fig. 1 and throughout the remainder of the paper, we have adopted the convention that figures reproduced from other sources are surrounded by a border; if the original caption is included, it is contained within the same border. This is also true of figure/caption pairs that represent summaries such as Fig. 1b. Additional captions of our own are placed outside the borders. The full text of the original captions is included in all four example figures.

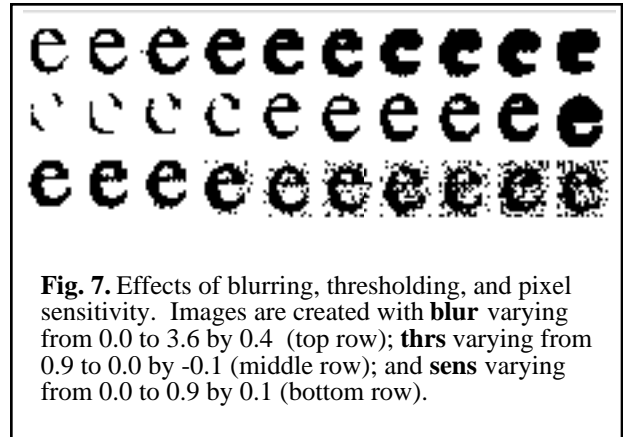


Fig. 7. Effects of blurring, thresholding, and pixel sensitivity. Images are created with **blur** varying from 0.0 to 3.6 by 0.4 (top row); **thrs** varying from 0.9 to 0.0 by -0.1 (middle row); and **sens** varying from 0.0 to 0.9 by 0.1 (bottom row).

Figure 1a, Example #1. This figure shows three monotonic sequences of degraded characters arranged in tabular form (Ho and Baird, 1997). The appearance of the characters is not discussed in the original paper, since their appearance is evident on viewing the figure.

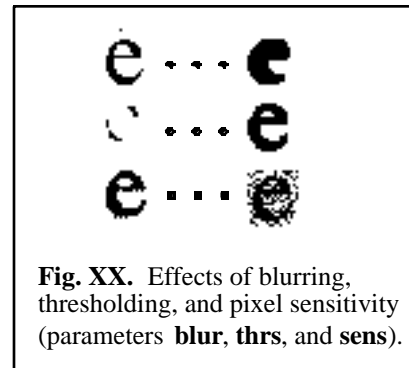


Fig. XX. Effects of blurring, thresholding, and pixel sensitivity (parameters **blur**, **thrs**, and **sens**).

Figure 1b. A manually constructed summary figure and caption for Fig. 1a. Each horizontal sequence in Fig. 1a shows a monotonic progression that is captured by the extreme values in this figure summary.

2.2 Example #2: Distilling a Flow Chart

Flow charts are common diagrams in many fields. They are often candidates for distillation because they contain minor steps that can be omitted in a summary. The flow chart in Fig. 2a shows the link of events leading from DNA to the synthesis of a protein. A normal path is shown on the left and an abnormal one on the right.

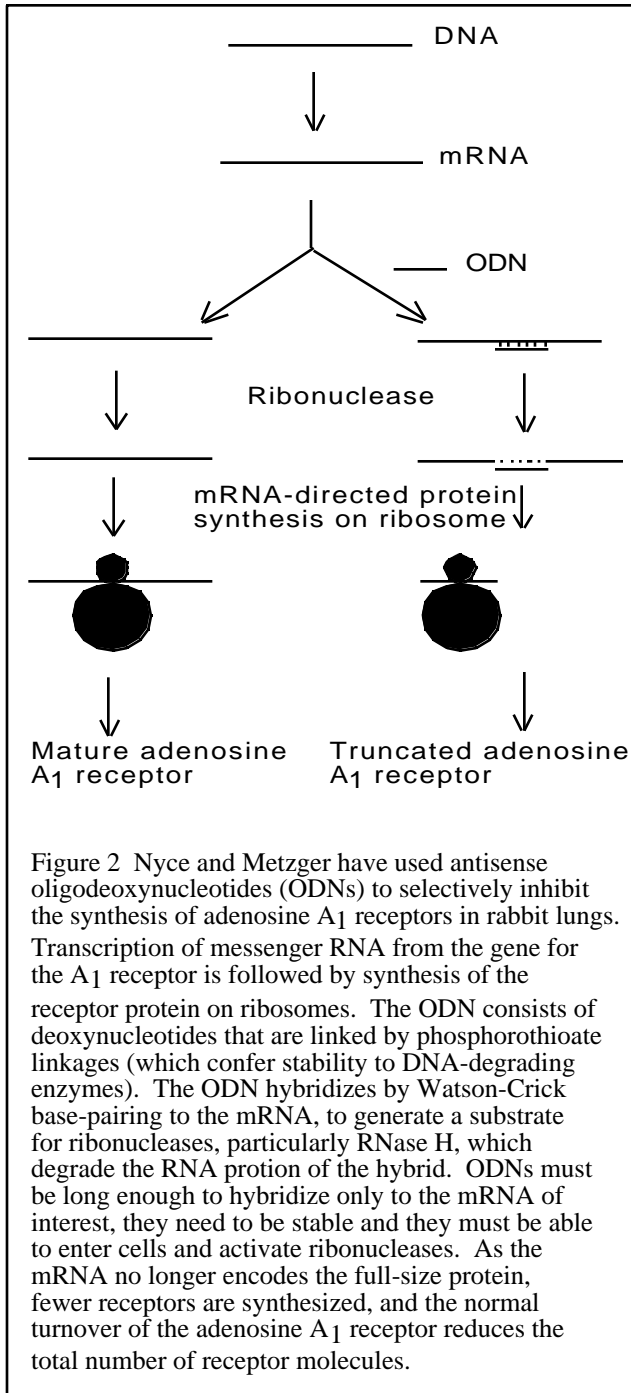


Figure 2a, Example #2. The topology of this flow

chart (Richardson, 1997) includes two initial stages, a binary split, and then four additional stages in each of the two pathways. Biologically, the key element is the addition of a certain substance, "ODN", in the right-hand pathway which leads to the synthesis of an abnormal and non-functioning receptor protein compared to the normal left-hand pathway.

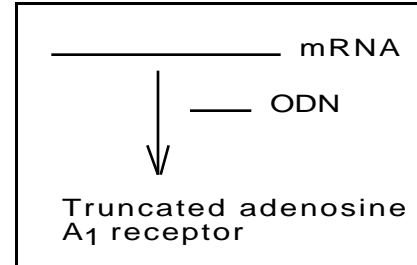


Figure 2b. A manually generated summary of Fig. 2a. The essence of the process is that ODN interferes with subsequent processing of the mRNA, leading to an abnormal protein. That is all that is retained in this simple summary.

Both the caption and the running text mention "ODN" extensively. Apposition tells us that ODN is the abbreviation used for "antisense oligodeoxynucleotide", and "antisense" is another prominent term, included also in the title of the article. This makes the right-hand branch of the flow chart, and ODN in particular, the most salient part of the diagram. The summary retains the arc with the ODN label, omits the common initial node and omits all intermediate states between the mRNA node and the final node. Since the left-hand path in Fig. 2a is normal and conventional, it is not particularly informative and is omitted in the summary.

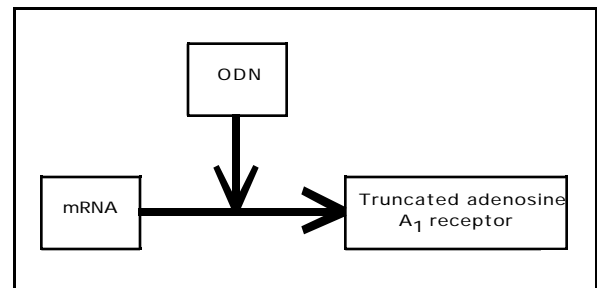


Figure 2c. A stylistically distinct rendering of the summary in Fig. 2b. This contains essentially the same information as the Fig. 2b. But the different style makes it difficult for a reader to first look at the summary and then go to the full paper where the full figure (2a) is presented, because of the horizontal organization versus the vertical organization of the original, rectangular nodes instead of the horizontal line nodes of the original, etc. A summary in the style

of Fig. 2b would be strongly preferred on these grounds.

2.3 Example #3: Merging for Contrast

Summarizing large collections of numerical data from experiments or complex simulations is an intrinsically difficult problem. The example, in Figs. 3a and 3b shows the *merging* parts of two different diagrams into a single one.

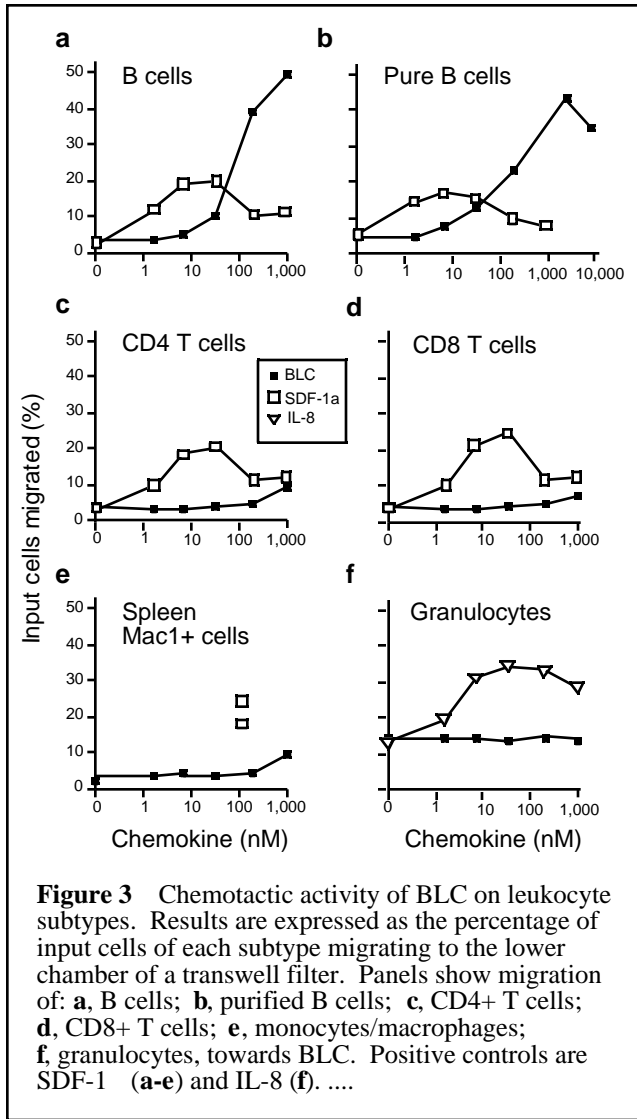


Figure 3a, Example #3. This contains the first six of eight subfigures from Fig 3 in (Gunn, et al., 1998) and the corresponding portions of the original caption.

The primary point made in the article in which the material in Fig. 3a appears is that the researchers have

discovered a substance that specifically affects (attracts) B cells (a class of motile blood cells) but not T cells and other major classes of blood cells. This can be pointedly summarized in a single figure that selects the portions of two of the data plots that show the contrasting behavior, e.g., plots **a** and **c**. The merged graph is shown in Fig. 3b.

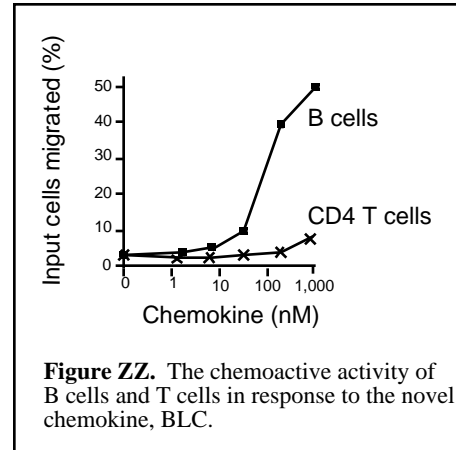


Figure 3b. This data plot contains the significant contrasting portions of Fig. 3a, parts **a** and **c**. We created the caption within this summary figure in the style normally used in biology research papers.

2.4 Example #4: Distilling a Large Block Diagram

Flow charts are typically DAGs, whereas block diagrams can have numerous interconnections and cycles. The example here is a block diagram for a video signal compression system (Bhatt, Birks and Hermreck, 1977), Fig. 4b. It is interesting to note that the original figure is in color, with eight distinct colors used for the various blocks. The colors for the various blocks are chosen essentially at random, with little regard to grouping them by functionality. This is apparent for example in a set of three source-to-transmitter figures earlier in the original article. In two of the figures the video camera icon is light blue and in the third it is a mustard color, an arbitrary difference for items with identical roles.

3. Automated Summarization of Diagrams

This section discusses the goals of automation and the important concepts that underlie the work. It also gives a brief overview of the important aspects of diagram content, from syntactic structure to semantic associations with real-world entities.

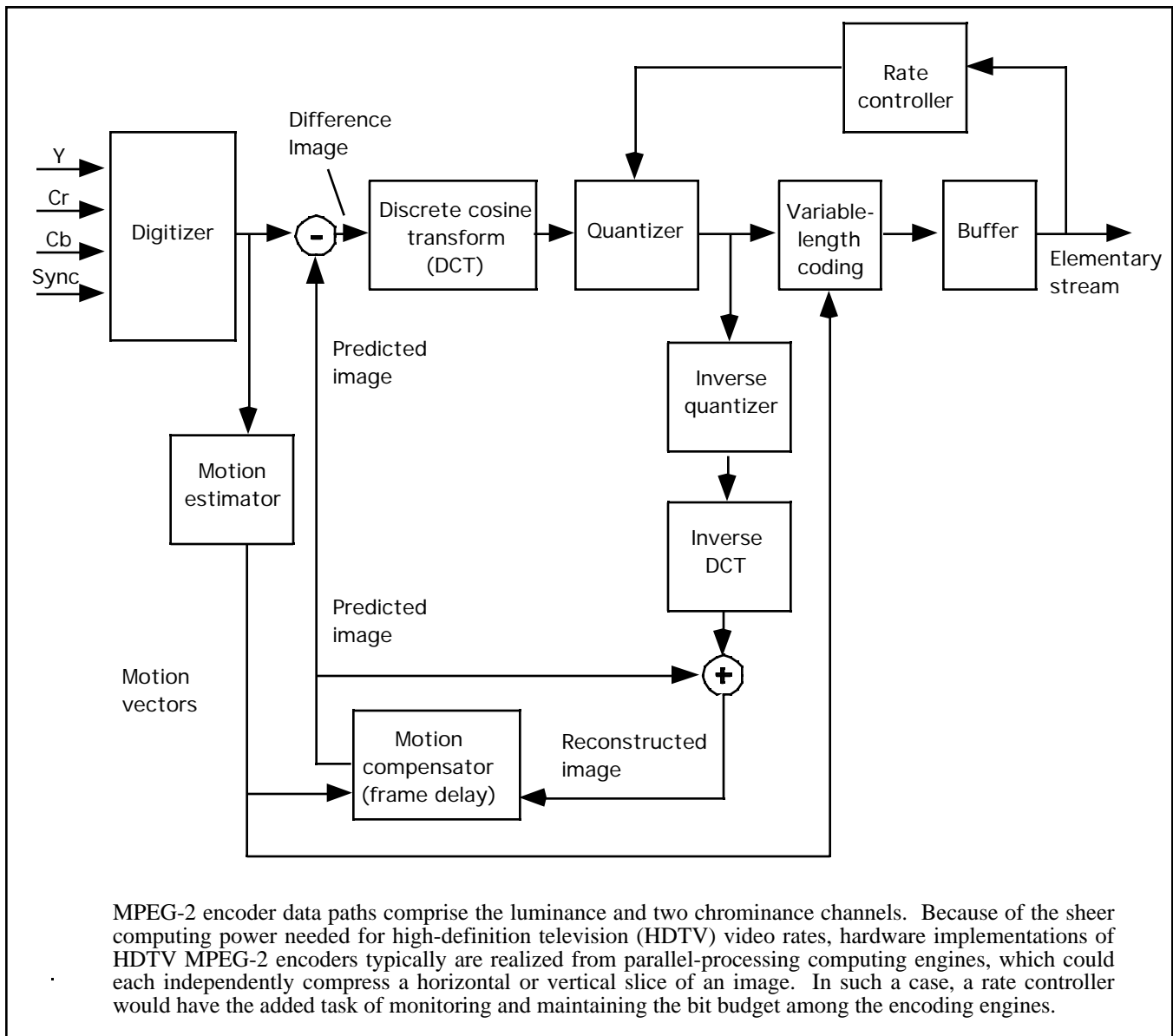


Figure 4a, Example #4. A large block diagram. The summary in Fig. 4b merges all the units below the primary upper backbone into a single composite unit.

3.1 Goals of Automation

One of the goals of any summarization system is to generate summaries in real time or, operating in batch mode, produce summaries at the rate at which the corpus of interest grows. That is, the methods should scale to realistically useful tasks. Our diagram parsing system currently parses a typical diagram (100 to 200 vector elements), in about 10 seconds. Let us assume, for the sake of argument, that the full summarization process would take about 30 seconds. Since there are 32M seconds/year,

such a system could process 1M diagrams/year. This would be a sizable fraction of the scientific and technical diagrams published per year, estimated from article counts gathered from databases such as *BIOSIS* and the *Engineering Index* (totaling over 1M documents/year).

Another goal is to develop summarization techniques that are relatively domain-independent. This is possible in text summarization, because corpus statistics are able to reveal major terms in the discussion, and the linear organization of the text can also be used, e.g., selecting the first sentences of section-leading paragraphs. It is more difficult to do domain-independent summarization for diagrams. We do suggest below that in some cases, the

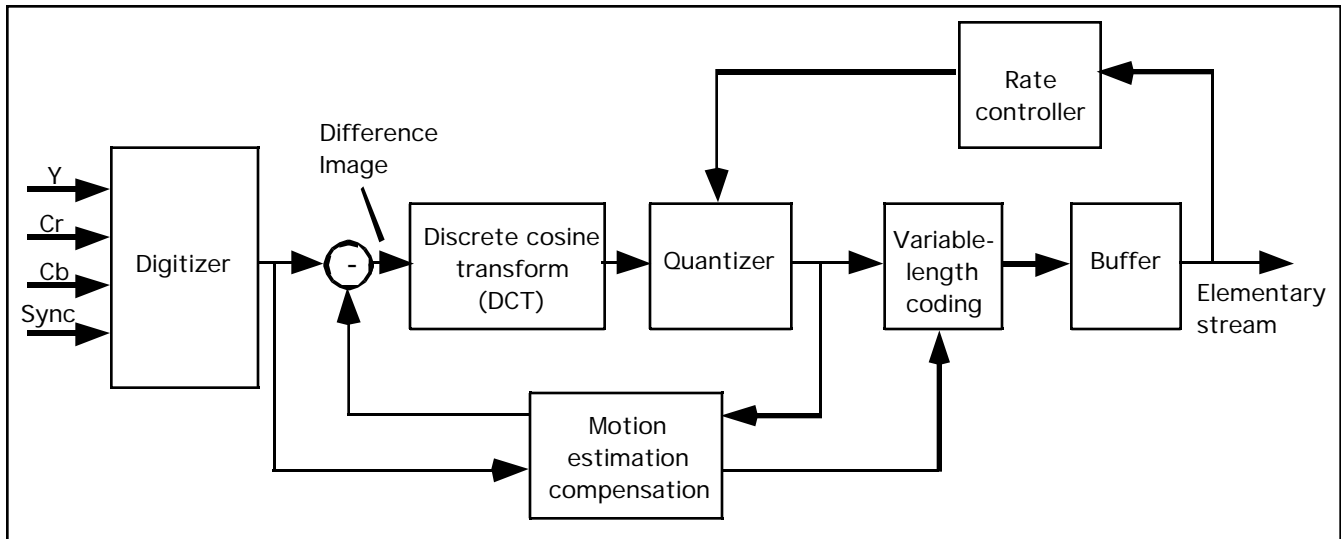


Figure 4b. A summary of Fig. 4a focusing on the horizontal "backbone" connecting the system input on the left to the output on the right. The choice of the text in the "black box" node at the bottom is a text summarization task.

overall topological and geometrical organization of a diagram may be exploited for domain-independent summarization.

3.2 Background for Automation

There is a collection of concepts and a standard terminology for them, some of it derived from text-based document analysis, that we will use in the discussions which follow. Documents can be characterized by their *genre* (journal article, newspaper article, book, etc.) and their *domain* (operating systems, microbiology, etc.). Text is essentially *propositional*, presenting statements in natural language that can be mapped into formalisms such as the predicate calculus. Language draws from a huge store of lexical entities that map words to meaning. In contrast, graphics is *analogical*, using spatial structures as the information-carrying elements (Sloman, 1995). Graphics and language are different *modalities*, because they require distinct interpretation functions, whereas both may rendered in the same *medium*, e.g., on the printed page (Stenning and Inder, 1995). Graphics can be divided into *veridical* entities, such as photographs and drawings of real-world structures, and *abstract diagrams* such as flow charts or data plots, but for the most part, diagrams are built up from relatively content-free items such as lines, curves, and closed regions such as polygons, which are (only roughly) analogous to the orthographic characters making up words. In graphics, there are also *conventional* classes of symbols, such as arrows, tick marks, error bars, corporate logos, and standard traffic signs (FHWA, 1997) that are learned elements of the visual lexicon.

It is useful to distinguish the general term, *graphics*, which contrasts with *text*, from specific subclasses such as

figures which are instances of graphics in documents; *diagrams* which are line drawings; and *images* which are raster-based. There are distinct *classes* of diagrams such as x,y data plots, pie charts, vertex/edge graphs, gene diagrams, etc.

The text associated with figures can be in *captions* or in the *running text*, or *included* within the figures. *Metadata* is normally propositional material that is available in an index or other data structure or is generated, that gives additional information about diagram structure and content, but is not always made available for viewing by the reader of an electronic document. *Tables* are hybrids of text and graphics. Summarization of text or graphics can be *indicative* or *informative* (Paice, 1990). The former presents material that indicates the subject domain (such as the audience picture we mentioned earlier), while the latter contains information which plays a substantive role in the document.

Additional concepts include the *purpose* of summarization which can include the intended *audience*, and a chosen *emphasis*, which can focus on methods, results, judgments, or matters in dispute. The choice of these can influence the type of summarization that is constructed.

3.3 Diagram Content

It is possible to define grammars and parsing strategies to produce structured descriptions of the contents of diagrams. In this section the basic structure of diagrams and its interpretation at the syntactic and semantic levels is described, using the examples in Fig. 5. Further details on the parsing approach are given in the next major section on diagram understanding.

The raw content of a diagram is the collection of vector primitives that make it up, the lines, curves, polygons, and text, each with their specific numerical parameters. For summarization to work, a description of the diagram is needed that codifies the objects, and their functional, geometrical, and topological relations, e.g., the fact that an object is a node in a block diagram that contains certain text and that is contacted on left by an oriented edge (indicated by an arrowhead). The edge may in addition emanate from another node or have a free end, etc.

The syntax and semantics of diagrams has strong overlaps with the related natural language concepts, but there are major differences. In language, the *model* of the logical form of an utterance is normally a collection of real-world entities existing outside language per se. But the basic objects in a figure can be viewed as having a physical existence of their own, e.g., marks on paper. More formally, the axioms of geometry and their associated computational algorithms (for length, distance, orthogonality, etc.) can be used to extend the meaning of geometrical objects in the formalism, allowing arbitrary geometrical calculations to discover or test relations among sets of objects. Graphical knowledge has been characterized as *vivid knowledge* (Levesque, 1987; Levesque, 1992), first-order languages which only contain ground atomic sentences, are universally quantified over the domain, and have other "concrete" properties. Entailment in such languages has been shown to be tractable. The limitation of this view is that though the graphical elements themselves are so restricted, the real-world entities to which they refer have no such constraints.

The approach above forms the basis for grammars that describe complex graphical objects. Fig. 5a shows a simple data plot, whose analysis by such a grammar identifies the scale lines with their tick marks, tick mark labels, and axis labels. The less orderly curve in the space delimited by the two axes is the data.

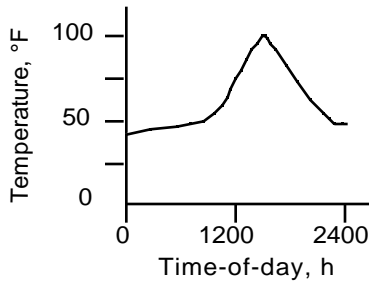


Figure 5a. Basic syntax: E.g., identification of the basic components, the x and y scale structures, and the data curve.

At the next level, in Fig. 5b, the world coordinates of the data points can be extracted, i.e., in units of degrees Fahrenheit and the time in hours. At this level there need be no explicit representation of the semantics of time or temperature.

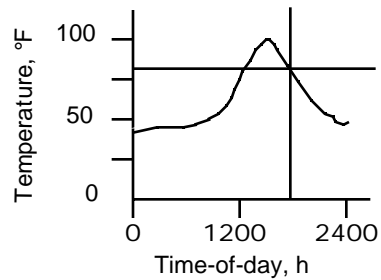


Figure 5b. Lowest semantic level: Numerical values of world coordinates, e.g., 79° at 1800 h.

Additional computations can yield useful information that is still domain-independent, e.g., the maximum value achieved by the data set, Fig. 5c. (The operations in Figs. 5b and 5c could be applied in either order.)

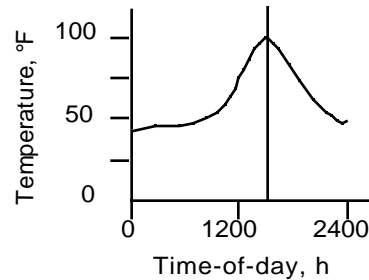


Figure 5c. Numerical analysis semantics: "Day's maximum of 100° reached at 1520 h"

Finally, an interpretation can be developed at a much higher level in which the information in the plot is related to real-world situations outside the diagram proper, in this case a statement about the affect of such temperatures on crop viability, Fig. 5d. Such an interpretation goes beyond diagram analysis per se, but could play a significant role in deciding whether or not to include information from particular diagrams in a document summary.

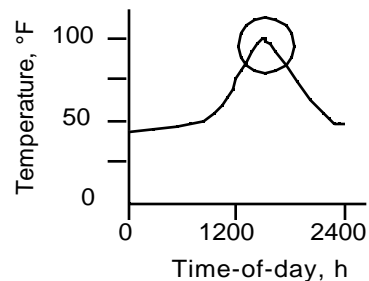


Figure 5d. Real-world semantics: "Reaching this temperature may cause some crop damage because of the current drought conditions"

4. Diagram Understanding

This section discusses our work on diagram parsing as well as work by others on the problem. Parsing is the first stage

of the diagram summarization process, producing the structural descriptions that are the input to later stages.

4.1 The Diagram Understanding System (DUS)

Parsing a diagram generates a syntactic description, a parse tree. The parse structures we consider here are based on approaches to parsing that we have pursued using *context-based constraint grammars* (Futelle, 1985; Futelle, 1990; Futelle, Carriero, Nikolakis and Tselman, 1992a; Futelle and Kakadiaris, 1991; Futelle, et al., 1992b; Futelle and Nikolakis, 1995). Our system is called the *Diagram Understanding System* or DUS. A typical production in a DUS grammar for x,y data plots describes a collection of x-axis tick marks and their associated horizontal scale line, such as in our figures 3a,b and 5a-d, by the following production,

```
X-Ticks -> Ticks X-Line
  (X-Line)
  (Ticks (touch X-Line ?)
    :constraints
    (> (number-of Ticks) 2))
```

In the X-Ticks production, the body of the production states the order of analysis (X-Line followed by Ticks) and states the constraints on the constituents—in this case that the number of Ticks be greater than 2. The term (touch X-Line ?) specifies that the items that can be considered in analyzing the Ticks production are restricted to a *context*, the set of objects that *touch* the X-Line already identified. The context functions as an *inherited attribute* in the sense of (Knuth, 1968). X-Line is defined by another production, as is Ticks, a *set object*,

```
Ticks -> Set ( Line )
  (:element-constraints
    (vertp Line)
    (short Line))
  (:constraint horiz-aligned))
```

In the Ticks production there are constraints on the individual elements of the set given by :element-constraints, as well as a constraint on the set as a whole, that its members be horizontally aligned with one another. A potential ambiguity of the Ticks rule is that the any subset of the vertical lines touching the horizontal one would be a legal instance. This ambiguity is avoided by a meta-rule that chooses the *maximal* (largest) set satisfying the constraints.

A simple diagram that would be an instance of this rule is shown in Fig. 6.

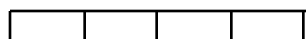


Figure 6. This structure is an instance of the X-Ticks grammar rule that describe it as a set of short vertical

lines that are horizontally aligned and touching a long horizontal line.

In addition to inherited attributes, there are *synthesized attributes* (Knuth, 1968) in the DUS formulation that are passed from a production to a higher-level node in which the constituent appears. An example would be the computation of the midpoint of a line, passed upwards as the value of an explicitly named attribute.

The efficiency of our system is aided by the use of context which restricts the search, as well as the order of construction of the constituents as stated in the rules. There are many computations of geometrical relations such as *near* and *aligned* that have to be done as parsing proceeds. These are speeded up by a preprocessing phase that builds an index, a spatially associative structure (SPAS). SPAS is a pyramid of square arrays covering the diagram space at varying levels of resolution, down to resolution of about one character width (Futelle, 1990). It is created in a preprocessing stage before parsing proper begins. Each square element lists all graphical objects that occupy or pass through the region. Thus, to find the objects near to a given one, only a local search in the SPAS index is needed. Computations of geometrical relations between simple objects such as straight lines or complex objects such as Bezier curves are all done in the same uniform way when using SPAS. Memoization is also used to enhance performance (Norvig, 1991).

Parsing in the DUS is implemented as a constraint satisfaction problem which, among other things, makes it straightforward to allow objects to be *shared*, i.e., to participate in more than one distinct structure in the same parse. Sharing is quite common in graphics. For example, the y-axis (ordinate) tick mark numerical labels in Fig. 3a are shared across a pair of plots, and the y-axis label on the left is shared by all six plots. Sharing allows us to analyze each of the six plots as a complete plot with full tick marks and a label without having to explicitly include sharing in the grammar. The parse of each of the six subplots in Fig. 3a can be treated as a tree structure even though it is technically a DAG because of the shared constituents.

Sharing allows another powerful type of analysis in which a single diagram is simultaneously analyzed in two different ways, all in terms of common objects rather than by two distinct analyses that later have to be brought into concordance. This is useful for example if one part of the analysis is stated in topological terms, e.g., the connectivity of a flow chart or block diagram, and another simultaneous analysis describes the spatial organization, e.g., the two lower vertical columns of nodes in the flow chart of Fig. 2a.

One issue that arises in a general-purpose retargetable system such as the DUS is how it should behave when presented with an arbitrary unknown diagram. We could attempt to parse the figure with a succession of grammars, each specialized to a different class of diagrams. A more efficient way to proceed would be to write a grammar that had in it a collection of simple productions that describe common "signatures" of certain classes of diagrams, such

as scale lines, arrows, tick-marks, back-bone structures, labeled boxes, etc. A preponderance of any one of these or of some subset of them in the resulting parse would greatly restrict the full-fledged grammars that would then be appropriate to use for more detailed analyses.

The DUS is implemented in Common Lisp / CLOS (Macintosh Common Lisp) and every constituent, whether a primitive (Line) or a higher-level one (Data-Plot) is an instance of a class of that name, with slots that include its constituents (RHS). The parsing proceeds top-down and the result is one or more solution objects of the start-node, which in turn contain constituents, etc., all CLOS instances. Detailed performance figures exist only for older machines (Futrelle and Nikolakis, 1995; Nikolakis, 1996), and scaling up from these, we expect that current machines should parse a typical diagram of 100 to 200 primitives, which is the size of a typical published data plot, in 10 seconds or less. The system is also being ported to Unix. So far, we have working grammars for x,y data plots, linear gene diagrams, and finite-state automata (Nikolakis, 1996).

Genesis of the DUS. Before reviewing other approaches to diagram parsing, it is worth explaining the strategy we employed to develop the DUS system, which gives it a very different character than other systems. The reason this discussion is relevant to the diagram summarization problem is that we intend to apply the same philosophy to the development of efficient and effective approaches to automated diagram summarization. Practically all other systems for parsing visual languages have grown from the extensive conceptual base and formalism developed for string languages, e.g., natural language and programming languages. Our approach began by studying thousands of published diagrams in a variety of scientific and technical publications. From that study we saw the great importance of repeated elements and thus the need for sets in our formalism. We also realized that there is a major distinction between *background* and *foreground* elements in informational diagrams. Background elements are those that are arranged in simple low-complexity patterns, e.g., with aligned repeated elements, such as tick marks. The background is called the *framework* in (Kosslyn, 1994). We then wrote algorithms that could discover and give structure to background elements very efficiently. Items not in the background constitute the foreground, the high-content or informational items, e.g., the non-uniformly positioned data points in a data plot. Our computational strategy was then extended to handle the foreground elements, ignoring the previously discovered background elements. Once this efficient computational system was in place, its procedural face was systematically transformed, using Lisp macros, to the essentially declarative form of the current DUS. Thus our system is the result of placing a declarative face on a fundamentally efficient underlying parsing engine, rather than attempting to efficiently parse grammars using strategies that are based ultimately on string language techniques.

4.2 Previous Work on Diagram Understanding

Now that we have described the DUS approach to diagram structure discovery, we will briefly discuss some of the other work in the field, generally known as *visual language parsing*. Reviews of this field can be found in Nikolakis' thesis (Nikolakis, 1996) and in the extensive recent review (Marriott, Meyer and Wittenburg, 1998) with 201 references. Given this amount of past work, our discussion is necessarily quite selective. Our approach falls under the general rubric of *attributed multiset grammars*. The constituents in these grammars may have geometric and semantic attributes associated with them. Productions contain constraints over the attributes of RHS constituents. Other approaches to parsing include logical and algebraic formalisms (Marriott, Meyer and Wittenburg, 1998). The primary statement that can be made about visual language parsing is that the demands of the problem are so great that the approaches needed to deal with them go beyond those that are theoretically and computationally tractable. This means that the utility of the approaches is very dependent on their design, their implementation, and the class of graphical items that they are expected to analyze. This is precisely the reason that our own approach was developed by focusing first on the efficient parsing of real diagrams, and only later on declarative formalisms and formal analysis (Nikolakis, 1996). Interestingly, in the extensive review we have already mentioned (Marriott, Meyer and Wittenburg, 1998), the analysis of published diagrams is hardly touched on as an application, in spite of their great volume and their importance in all of scientific communication. In the same vein, there has been little concern over parsing diagrams with hundreds of elements and essentially no systematic performance studies in this domain. The great utility of parsing systems such as the DUS is that they can be retargeted, applied to a variety of diagram types by simply changing the grammar. The only systems that have been built for truly large diagrams are hand-crafted, non-retargetable systems for problems such as interpreting drawings of telephone central office equipment racks ("distributing frames") (Luo, Kasturi, Arias and Chhabra, 1997), architectural drawings (Ah-Soon and Tombre, 1997), etc.

Comparison of a DUS and CMG example. Though developed from different points of view, it is interesting to compare a portion of a DUS grammar for a state diagram with the corresponding production in constraint multiset grammars (CMG) (Marriott, 1994). The DUS grammar includes context restrictions and sets and explicitly states the sequence of constituent processing. The CMG grammar is purely declarative.

DUS:

```
***** < TRANSITIONS > *****
Transition -> A-state_1 Labeled-arrow
              A-state_2
(Labeled-arrow)
(A-state_1
 (touch (leave-pt (arrow Labeled-arrow)) '?))
(A-state_2
 (touch (reach-pt (arrow Labeled-arrow)) '?));
Transitions -> Set ( Transition );
```

CMG:

```
TR:transition ::= A:arrow, T:text
  where exists R:state, S:state where
  T.midpoint close_to A.midpoint
  R.radius=distance(A.startpoint,R.midpoint),
  S.radius=distance(A.endpoint,S.midpoint)
  and TR.from=R.name, TR.to=S.name.
```

4.3 Diagram Syntax, Semantics, and Style

Syntax and Semantics. Stripped to its essentials, the DUS generates a parse tree for a diagram. But as the earlier examples make clear, both the DUS and CMG grammars are couched in terms of constituents whose semantic roles are evident from their names. Here is a list of some of the constituents in our grammars that makes this clear. For state machines: Init-state, Transitions, Final-states, Labeled-arrow, A-state. For gene diagrams: Gene, Genebody, Gene-title, Backbone, Segment. For data plots: XY-Data-Plot, X-Axis, Y-Axis Data, Data-Lines, Data-points, etc. Though writing grammars in this way might appear to solve the semantics problem, extending the coverage to a greater variety of diagram classes and domains could result in a proliferation of grammars that each try to solve the semantics role problem in a single step. For example, a graph could represent an organizational chart, a network diagram, a chemical process flow chart, etc. A better way to deal with such variety would probably be to have a grammar with more neutral terms, followed by a semantic interpretation process tuned to the domain. This would parallel natural language semantic formalisms more closely.

Style. In Fig. 2c we demonstrated what could happen if the details of the style and layout of the original diagram were not followed in constructing a summary. The reader who then went from the summary to the full document and its figures would find the transition disorienting. This situation could be even worse for a complex diagram such as the block diagram summary of Fig. 4b. Due to its complexity there are a large number of different ways in which the summary could be laid out on the page (vertically for example!) which would have the same topology but a radically different appearance.

In order to maintain as much style constancy as possible, it is necessary to capture the stylistic and layout

information of the original in the diagram understanding step and preserve it during the summarization and generation process. A related problem is discussed in the automated graph drawing literature. Graph drawing sometimes proceeds incrementally, as the user adds new vertices. If the automated layout of the graph were recomputed from scratch each time a new vertex was added, its appearance could change radically at each step. So some graph drawing algorithms attempt to preserve as much of the layout as possible when redrawing after insertion which allows the user to maintain a rather constant "mental map" of the graph (Misue, Eades, Lai and Sugiyama, 1995).

Most visual language parsing systems pay little or no attention to the stylistic aspects of diagrams. In our DUS work, the vector-based diagram data that we parse is derived from postprocessing Postscript files, themselves generated from diagrams that we have created using drawing applications such as Canvas. These retain all style information such as line widths, fonts, color, etc. Our parsing work has for the most part ignored the style information, but clearly it is available and could be used in the later generation phase. The locations and therefore the relative locations of various objects is an integral part of diagram parsing, so that layout information is always available.

There is a more complex problem that must be faced in capturing layout information for later use in generation, which is that the description of the necessary topological, content, and layout information might be difficult to represent simultaneously in a single parse. It might be necessary to build additional tools to extract the variety of types of information needed from a single parse. But as we described earlier, the constraint-based approach of the DUS, and sharing in particular, allows more than one type of analysis to proceed simultaneously, with the various results all using the same objects where appropriate.

5. Automated Diagram Generation

Diagram summarization often involves the *generation* of a diagram, once the summarized structure for the diagram is developed. There are three important aspects of the problem of generating diagrams. The first is the art and science of good graphics design which prescribes guidelines for the use of graphics elements and their layout in space in ways that make the presentation of information clear and unambiguous (Cleveland, 1994; Kosslyn, 1994; Tufte, 1985; Tufte, 1990; Tufte, 1997). The second is the collection of common systems that generate much of the graphics produced today. These are exemplified by systems such as spreadsheets that have built-in data plot generation capabilities. In these systems, the style of the plot is selected from a small set of examples and some portion of the data in the spreadsheet is used to generate a data plot containing it. This is normally not a difficult task, because the system designers retain total control over the

specifications of each of the supported styles. The third aspect is the work of the graph drawing community, which is primarily concerned with drawing large vertex/edge graphs under certain constraints such as minimizing the number of crossings or minimizing the number of right-angle bends (for drawings using only vertical and horizontal edge segments). There have been a number of annual graph drawing symposia, dealing with these issues, e.g., (DiBattista, 1997). Beyond these approaches to the generation of graphics, there is a good deal of research today on scientific visualization and the presentation of large datasets such as the collections of documents returned by retrieval systems. This work is less directly applicable to the problem of summarizing already existing diagrams, the primary concern here.

The field of automated graphics design and layout is even more active than visual language parsing. An excellent recent review of just the constraint-based approaches to such problems has more than 200 references (Hower and Graf, 1996). The early work in this field used algebraic (Mackinlay, 1987) and rule-based approaches (Seligmann and Feiner, 1991), but there is now substantial work based on constraints (Graf, 1998). The readings volume containing the last-mentioned reference reprints nine other papers on the topics of automated graphics design and layout (Maybury and Wahlster, 1998).

Constraint-solving algorithms have been developed with special attention to the problems of interactive graphics input including constraint hierarchies (Borning, Freeman-Benson and Wilson, 1992), and incremental constraint solving (Freeman-Benson, Maloney and Borning, 1990).

Generation for Summarization. The important and obvious point to note at this juncture is that the DUS diagram parsing system is itself constraint-based, so the grammars already go a long way towards specifying the constraints on generation. The major difference between diagram summarization and most of the constraint-based layout and design problems is that we focus on the *reduction* of diagrams rather than *additions* to them, though this is not a profound difference. In fact, it is reasonable, particularly with complex graphs, to reduce them a node at a time, allowing the system to maintain the same general layout, solving only an incremental constraint problem at each step. One major addition to our formalism that would be required to move it from parsing to generation would be to add quantitative preferences. A simple example of where this would be needed would be in the generation of tick marks in a data plot. In parsing a data plot, any number of tick marks would be acceptable, but in generation, some minimal spacing of a few mm and a total number of the order of ten to twenty tick marks would be typical design parameters.

The problem of automated layout can be computationally quite demanding, e.g., for label placement (Christensen, Marks and Shieber, 1995), but the diagram summarization problem should be able to take account of the many details embodied in the original diagram(s) to be summarized. From the original diagram it can obtain box

dimensions, font size, line widths, arrowheads, alignments, etc., and reuse these as much as possible. In the discussion below of automated techniques that could be applied to our example diagrams, we will make more specific suggestions.

6. The Relations of Text and Graphics

One of the first suggestions that researchers in text summarization make in discussions about the problem of diagram summarization is to exploit the text associated with diagrams, especially captions. But this approach is fraught with problems, as we explain and illustrate in this section.

6.1 Figure Captions.

The style and content of figure captions varies widely across the range of scholarly journals. Surveying current academic periodicals revealed the following: The biological literature, and to a large extent all scientific literature that reports the results of experiments, has substantial captions, averaging over 100 words in length. The first portion of each caption is typically a noun phrase that serves as a title. A few publications, such as the *Biochemical Journal*, separate the figure title from the caption text. Engineering and Physics tend to have only noun phrase captions from about 3 to 15 words in length. Mathematics is the extreme, in many cases having only figure numbers and no captions whatsoever. Linguistics journals often number figures (such as parse trees) just as all other presented items, with sequential parenthesized numbers in the margins and no captions. The humanities and social sciences, business, and clinical medicine have few figures but may include tables. Examples of typical brief figure captions from the information retrieval literature include, "System Architecture", "Hierarchical feature map", and "Average Improvement in Precision", each quoted in its entirety.

In spite of the paucity of significance information in many figure captions, they can represent a useful division of the set of figures in a document, and can be useful in the *selection* form of summarization, when only one or two figures are selected to be included in the summary, either in their original form or in summarized form. For example, the title portions of the four figures in the original article corresponding to our example #3 are: "Expression pattern of BLC mRNA in mouse tissues.", "BLC sequence and alignment with other protein sequences.", "Chemotactic activity of BLC on leukocyte subtypes.", and "BLR-1 mediated calcium mobilization and chemotaxins in response to BLC." In a system in which the reader states some particular aspect that he or she would like to see emphasized in a summary, titles such as this could be useful guides for an automated selection process.

It might seem that lengthy figure captions, where they are available, would be excellent indicators of figure content and significance. Surprisingly, this is rarely the

case. Virtually all lengthy captions that we have studied in the scientific literature are associated with the presentation of experimental data in the accompanying figures. These captions focus on the detailed conditions under which the data was gathered, or label the specific data subsets shown. They rarely contain any language that evaluates or explains the significance of the figure content.

Publications intended for wide audiences, such as *National Geographic*, are the only ones that appear to put significant content in figure captions. This is presumably done to present information in smaller "chunks" by allowing individual figures and their captions to be understood by the casual reader without their having to read the accompanying article in detail. Many popular publications do not even use figure numbers, making it difficult to refer to specific figures from the running text. Many popular magazines such as *Time* or *Scientific American*, and most newspapers, have "stand-alone" figure-caption pairs and never specifically refer to the figures in the running text at all.

The conclusion of the preceding analysis is that for most of the scientific and technical literature we can look for certain details in captions, but all comments as to the substance and significance of figure content will have to be found in the running text. As we have already emphasized, there will be important aspects of figure content that are not explained anywhere in the text of a document, since their content is obvious when looking at the figure.

6.2 Examples of Figure Captions

Example #1. In Fig. 1a, the "Effects" in the original caption are self-evident when viewing the figure. It is far more appropriate to simply look at them than to read a textual description of them.

Example #2. Fig. 2a appeared in an article in the *news and views* section of the journal *Nature*. These short pieces appear in the early pages of each issue and discuss the significance and interesting points of full technical papers that appear later in the same issue. As such, each *news and views* piece is written in an accessible and somewhat pedagogical style. The original 200 word caption in Fig. 2a explains the overall process of protein synthesis and how it was blocked by the addition of ODN in the experiments reported. Natural language analysis of this caption could markedly aid the summarization processing of the figure.

Example #3. The original 100 word caption in Fig. 3a has a somewhat informative initial title phrase, "Chemotactic activity of BLC on leukocyte subtypes." Most of the remainder of the caption deals with the experimental procedures and specifies the particular cell types used. The only particularly informative phrases in the caption refer to the control experiments whose results are contained in subfigures g and h, which we have not reproduced. The most significant results, which are presented in subfigures a-f of Fig. 3a and summarized in our Fig. 3b, are not explicated in the caption at all. In this regard, this caption

is quite typical of figure captions throughout the biological literature.

Example #4. *IEEE Spectrum* is that organization's most widely distributed publication, covering topics of interest to all members in carefully edited form with extensive color illustrations. It is a popular technical magazine. Consistent with this role, the captions are normally rather informative. The caption in Fig. 4a begins abruptly with a sentence that serves as a less than optimal figure title. It begins, "MPEG-2 encoder data paths comprise the luminance and two chrominance channels." It does not state that the figure is in fact a block diagram, though this is immediately obvious from the figure. The caption does not discuss the contents of the diagram per se but mentions the hardware implementation of it as well as changes that would need to be made to accommodate specialized hardware. We have to conclude that this caption would not be particularly helpful in the diagram summarization process.

6.3 What Running Text Says About Diagrams

When figure content is explicated in text, we have seen that little of it is to be found in figure captions; it is in the running text (if anywhere!). So we turn to a discussion of the running text of our four examples where we will find more useful information referring to figure content. The utility of this information is tempered by the complexity of the text, the same complexity met in attempting to do text summarization. Often there is only one or a few explicit references in the running text to a specific figure. But the discussion of the topics related to the figure may be still be extensive and distributed throughout the document.

Example #1. We have already quoted a significant item drawn from the running text in our first discussion of this example, Fig. 1a, viz.,

"...The key contributors to scanning and printing noise are the three parameters blur ..., thrs ..., and sens These will be our primary concern. Figure 7 illustrates these effects."

There are two marker phrases in the running text excerpt that emphasize the importance of the material in their Fig. 7 (our Fig. 1a): "The key contributors .. are ...", and, "... our primary concern." This is followed by the definitively worded figure reference sentence, "Figure 7 illustrates these effects." If this text is to be used to evaluate the importance of the figure, the referent of "these effects" would have to be identified. This is not a trivial problem because there is no obvious "effect-like" term in the preceding sentences, only "contributors", "parameters" and a prior use of "These", which itself needs to have its referents identified.

Example #2. Since this figure (2a) has an extensive and useful caption, the content of the running text is less critical. The article has two figures, of which Fig. 2a is the second. The first mention of it in the running text is in the initial sentence of the third paragraph, about 1/3 of the way

through the 900 word article. The sentence is, "ODNs act by sequence-specific hybridization to messenger RNA (Fig. 2)." The sentence which follows it parallels the content of our manually generated summary, Fig. 2b. The sentence is, "They then selectively prevent the translation of the RNA message into protein, and promote degradation of the message by ribonucleases." Some of the details presented in the original figure, Fig. 2a, are omitted in this summary sentence, suggesting that they can be omitted in a summary figure. Though this informal analysis is reasonable on the surface, it has inherent dangers – for example there may be important material in a figure that should be included in a summary figure but is so visually obvious that it is not discussed in the text.

Example #3. This figure (3a) has a weakly informative caption, as we have discussed, so the running text needs to be looked at closely. The article is a *letter to nature*, so it is short and packed with information, with figures covering micrography, gels results, DNA and amino acid sequences and comparisons, the eight-part figure we are discussing, and a final seven-part figure. The first two sentences that mention our figure (their Fig. 3) contain exactly the contrast displayed in our summary figure, 3b, viz.,

"BLC induced a strong chemotactic response in B cells in experiments using either total spleen lymphocytes (Fig. 2a) or purified B cells (Fig. 2b)."

and

"In contrast, BLC showed limited activity towards T cells carrying CD4 or CD8 antigens (Fig. 2c, d)."

In our summary, we chose to display the first item of each pair, **a** and **c**, but any pairing would have sufficed (one from a or b and the other from c-f). This is evident from the plots we have shown in Fig. 3a. So in this case at least, the running text has elements in it with significant marker items, "strong response", "In contrast", and "limited activity". These could guide an automated diagram summary system in achieving the summary diagram we produced manually in Fig. 3b.

Example #4. Our Fig. 4a appears in a two-page spread, a sidebar (distinguished by a light blue background), in the middle of a full article. The article figures are numbered, but the four in the sidebar have no figure numbers. Some are referred to in the running text positionally, e.g., "[see top left diagram, opposite]", but the figure we have chosen is not referred to explicitly in the running text at all. Without explicitly stating that the figure is being discussed, about one-third of the way through, it mentions the "digitizer", the left-most item in the block diagram. The discussion next mentions the DCT block, then the quantizer. The discussion then moves to the form of the signals within the encoder, with occasional mentions of other blocks in the diagram, e.g., the motion estimator and the two "inverse" blocks. Eventually, most of the blocks are described. There is no obvious emphasis on particular blocks or links in the running text, either by frequency of occurrence or by emphasis terms, so there is little to guide

automated summarization of the block diagram.

6.4 Text Included within Diagrams

Text within diagrams is common, ranging from data plot axis labels to text overlaid on images. It is distinct from captions and running text in that it is normally positioned within the diagram to indicate its role. Included text can play a number of distinct roles. Our example #1 has no included text so it is omitted from this part of the discussion.

Example #2. The various text items within Fig. 2a play four distinct roles. The top two text items label vertices (nodes) in the flow chart, the two horizontal lines, indicated by their adjacency and horizontal alignment with the lines. The next text item, "ODN", which labels an edge of the graph, indicates that a certain substance is introduced at the point in the process between the mRNA and the topmost right node below "ODN". The text "Ribonuclease" is aligned with and between the two arrows, indicating that it is associated with the transition between the vertices on each side (shared), rather than vertices proper. The same analysis applies to "mRNA-directed protein synthesis on the ribosome" below it. Finally, the two text items at the very bottom of the figure constitute vertices in themselves and both identify and describe the vertices (protein products as the end result of processing). In our summary diagram, Fig. 2b, the three text items play three distinct roles, the same roles they play in the original, Fig. 2a.

Example #3. In this set of data plots, Fig. 3a, the text items play five distinct roles. The six characters, a-f, label the six separate plots to allow efficient reference in the caption and running text. Their position is a *customary* one, lying slightly outside and above each plot. This prevents them from being confused with the shared ordinate label for the six plots, "Input cells migrated (%)". Another customary position for such letter labels is in the position occupied by the cell descriptor text, e.g., "B cells". The text within the key box identifies the three different types of data points, "patches" in (Kosslyn, 1994). The abscissa labels, "Chemokine (nM)", are duplicated, each being shared by the three plots above them. The numerical text labeling the ordinate tick marks, 0-50, is shared across two horizontally aligned plots, but the abscissa tick labels are repeated for all six plots (and differ for graph b). Our summary plot, Fig. 3b, is a single item requiring no sharing of text and has text in three roles. The role of the cell descriptor text has changed from one associated with an entire (sub)plot to one in which the two data sets are separately labeled. (A key could also have been used.)

Example #4. A block diagram, like a flow chart, is often replete with text, labeling every block. Of course, we have seen a flow chart, example #2, in which some nodes are purely graphical. In the block diagram of Fig. 4b, text labels either the blocks (vertices) or the links between them (directed edges). In one case, in the upper left, the text "Difference Image" labels a *call-out*, used because there is

no space in which to insert the text appropriately. Call-outs are frequently used to solve these layout "real estate" problems. Our summary diagram, Fig. 4b, uses text in the identical roles, including using the same call-out. Unlike the prior flow chart example, #2, the function of the link labels is to denote the data flowing between the processing or buffer nodes. In the flow chart, the link labels play a complementary role, indicating the *processing* occurring between the nodes that leads from one substance or configuration to the next.

7. Diagram Summarization Procedures

In this section we will discuss the structures (parses) generated for each original diagram of the four examples, the transformations done to produce summary structures, and the process of generating the resulting summarized diagram. Each diagram presents us with a new set of problems. The algorithms and other automation procedures used below have not yet been implemented in running code. Rather, the procedures are spelled out explicitly and the manipulations are done by hand. Of course the DUS has been used to produce full parses of similar diagrams in the past.

We have chosen to ignore the problem of text summarization required to synthesize new captions for diagram summaries. This is a topic unto itself, discussed in all the other papers in this volume. Text summarization in the context of diagram summarization is not simple, because the text summaries have to be concordant with the material that is presented in the summary diagrams, which may be a subset of the old or contain newly synthesized items, e.g., the new composite vertex at the bottom of the block diagram summary, Fig. 4b. The referent problem also arises whenever any summarization of the running text refers to figures that no longer are included or refers to material within a figure that has been altered or deleted.

7.1 Distilling Figures

Numerically parameterized sequences, Example #1. Fig. 1a contains three parallel sequences of items (ten simple images) each parameterized by a monotonic sequence of numerical parameter values.

In order for our diagram parsing approach to produce a useful structure for Fig. 1a, we will assume that the figure is made up of 30 small images. A grammar rule would be written describing a regular rectangular array using the horizontally and vertically aligned constraints. At this point the knowledge from the caption would need to be used, the fact that each row contains results for changes in a single degradation parameter. Combining the structure and caption information would allow the system to perform a sequence elision procedure, retaining only the extreme instances (and possibly the fifth or sixth instance to represent the intermediate appearances). The elided structure would be built using the same parse representation as the original. Using quantitative

parameters from the original figure, the summary figure could be constructed, Fig. 1b. An alternate approach would be to specify the summary as the original minus a sequence of deletions of the intermediate items in each row. Then an incremental constraint solver would "slide" the outer items toward the middle, producing a reduced form through a series of minimal changes.

The manual summary elides the eight intermediate images in each row, yielding Fig. 1b. This procedure could fail to produce correct results in many instances. It is often the case for such parameterized sequences that the most important items appear at intermediate values of the parameter, rather than at the extremes (minimum or maximum) values. For example, for living systems, optimal viability and growth is achieved at intermediate values of temperature, salinity, pH, etc. Ordinary data plots such as in example #3, Fig. 3a, commonly show such behavior, so an algorithm that produced a summary by only reporting values of the dependent data items at extreme values of the independent parameter value would often fail.

The article in which Fig. 1a appeared is concerned with synthesizing defective character images and then evaluating the performance of OCR systems trained on the images. Conceptually, it is clear to a manual summarist that increasing values of the image defect parameters induce a monotonic change in the quality of the images. Thus the extreme values, the "best" and "worst" items, are quite adequate as a summary. On reading the text it appears that it would be difficult to extract this type of information from it to guide an automated summarization process. The conservative approach would be to include the extreme parameter value items as well as one item corresponding to an intermediate parameter value. Choosing the value appropriately in an automated way, rather than just choosing the mean or median, may be problematic.

7.2 Graph Reduction

When graphs such as flow-charts or block diagrams are represented as standard directed vertex-edge structures, there are topological reduction procedures that can be applied to distill them to simpler form. Because they are based entirely on topology, these methods are domain independent. We define one such reduction procedure, *link-subgraph-deletion (LSD)*, and apply it to two of our diagrams. In LSD we identify certain subgraphs of a larger graph as shown in Fig. 7a. Each such subgraph is a *meganode*, a set of vertices which is allowed to have only a single entering edge and a single exit edge. Otherwise it may have arbitrary internal connectivity. The vertices that precede and follow the subgraph, a and b, can have arbitrary additional connectivity, indicated schematically in Fig. 7a. The graph is reduced by deleting the entire subgraph, resulting in Fig. 7b. The new edge between a and b now receives an ordered pair of labels, e and f. The LSD procedure uses the *maximal 2-connected* subgraphs between nodes since, for example, a simple linked list would contain many 2-connected subgraphs.

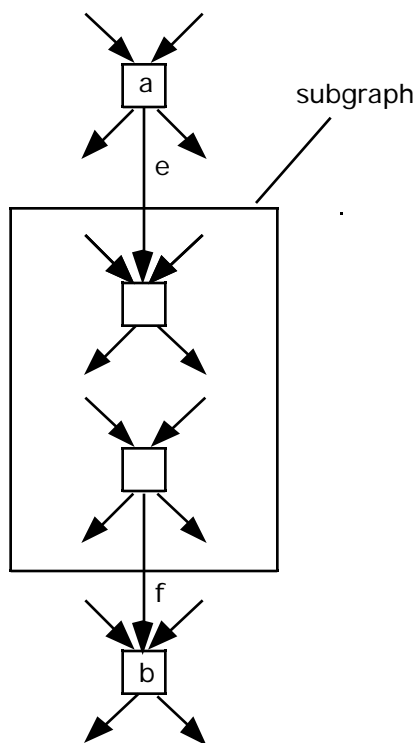


Fig. 7a. A subgraph of restricted connectivity that is deleted by the link subgraph deletion procedure (LSD), resulting in the graph shown in Fig. 7b.

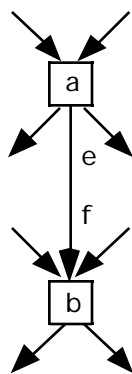


Figure 7b. The graph resulting from the deletion of the subgraph in Fig. 7a. The a-to-b edge is now labeled with the ordered pair of labels from the original.

Application of LSD to Example #2. The application of the link subgraph deletion procedure to the molecular flow chart of example #2 results in the distilled version of Fig. 8a. Parsing the original graph requires a grammar describing a graph, one that allows a vertex to be a text item (the two vertices at the bottom of the original graph). This would require a slight extension of the grammars previously used in the DUS for parsing state-machine diagrams. Another portion of the parse would be needed to

capture the vertical geometric arrangement of the diagram. The parse result would be walked to discover its topology, which could then be reduced by LSD. The resultant structure would be used to generate the summary diagram using the geometrical arrangement and numerical parameters of the original. Note that the object-based formulation in the DUS makes it relatively easy to reposition items and sets of items in space, so that for example, the first vertex and its "DNA" label could be repositioned as a unit to its final position in the summary diagram, as could the mRNA vertex and label, the branching edge and its ODN label, etc.

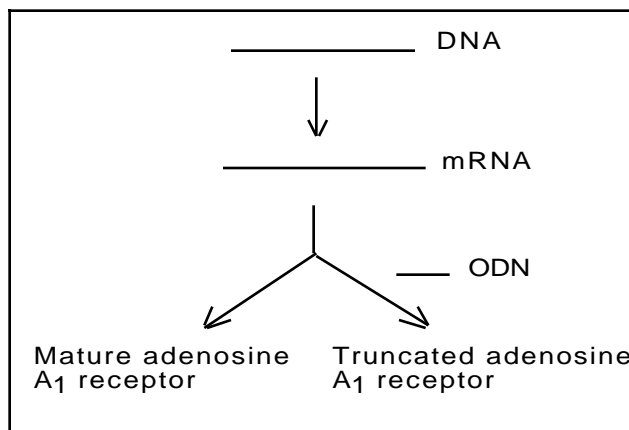


Figure 8a. Summary of Fig. 2a (example #2) produced by application of the link subgraph deletion (LSD) procedure, a domain-independent algorithm based entirely on graph topology. Compare with the simpler manual summary in Fig. 2b.

The original figure, 2a, contained 10 vertices and 9 edges. The algorithm-based summary in Fig. 8a contains 4 vertices and 3 edges, whereas the "optimal" manual summary of Fig. 2b contains only 2 vertices and 1 edge. The material in Fig. 8a that does not appear in the manual summary are the vertices for DNA at the top and the "Mature ..." item on the lower left, as well as their two connecting edges. The caption and running text in the original article contain 10 references to ODN as well as 5 to RNA, but only 1 reference to DNA. This could be taken as grounds for further automated pruning of Fig. 8a to delete the DNA node and its link as was done in the manual summary. It is also the case that in biology, the synthesis (transcription) of RNA from a DNA template is so commonplace that it hardly bears mention. The text describing the normal versus the abnormal (truncated) receptors is more complex and does not yield useful word-count data for, e.g., "mature" versus "truncated". This suggests that the best that an automated summarization procedure might be able to accomplish is the diagram shown in Fig. 8b, with 3 vertices and 2 edges.

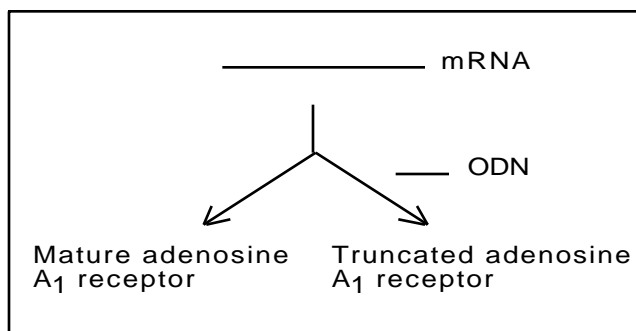


Figure 8b. A automated summary that might be produced using text analysis to discover salient concepts (terms) in the caption and running text for the original figure. Starting with the automated summary in Fig. 8a, the lack of evidence for the importance of the DNA-labeled vertex has been used to delete it and its accompanying edge.

7.3 Merging Figures

Compare and Contrast — Data in Example #3. The datasets in example #3, Fig. 3a are significant because they demonstrate a contrast between two types of behavior that is evident when certain subsets of the data are compared. The running text of the article places great emphasis on the compound, "BLC", mentioning it explicitly more than 50 times in the 3000 word article. It is introduced in the abstract in the sentence, "Here we describe a novel chemokine, B-lymphocyte chemoattractant (BLC), that is strongly expressed in the follicles of Peyer's patches, the spleen and lymph nodes." This simple statistic by itself could lead an automated system to focus on the BLC-related data sets within Fig. 3a (the black squares). Even the simplest statistical analysis of the BLC-related data sets would reveal that the data behavior in plots a and b, related to B cells, is significantly different from the corresponding behavior in plots c, d, e, and f. This would allow an automated procedure to select a pair of strongly contrasting data sets such as the two shown in the manually constructed summary of Fig. 3b, taken from Fig. 3a, plots a and c.

Though the preceding argument is attractive, it will be difficult to test the utility of such approaches without extensive application of automation algorithms to a variety of examples. Many published datasets are more equivocal than those in Fig. 3a, making it difficult to find clear-cut contrasts. But scientists continually strive to design experiments and data presentation procedures that make the significance of data strikingly clear.

The automated procedure for building the summarization in this example would start with a parse of the six plots. The parse distinguishes the background scale lines and their annotations from the data points and lines proper. We'll assume that analysis of the text and data has resulted in the choice of the appropriate datasets in a and c. Constructing the resultant summary plot could then be

done by bringing together and aligning a few large constituents: the x and y scale lines from plot a or c, the BLC dataset from plot a, the BLC dataset from plot c, the y scale label from the left, the x scale label from the bottom below plot e, the "B cells" label from plot a, and the "CD4 T cells" label from plot c.

7.4 Role of Layout in Summarization Decisions

Example #4, a Large Block Diagram. This block diagram, Fig. 4b, is difficult to deal with because as a graph it has a high level of connectivity (only one small link subgraph). This makes it difficult to argue that some particular portion of it should be favored for retention or for reduction/deletion. As we stated earlier, the running text accompanying the figure discusses most of the blocks with equal emphasis.

What can be used to drive the decision-making process for an automated summarization algorithm for this figure is its *geometrical layout*. Viewing the diagram as a complex flow chart, a signal processor, it has signal input on the upper left, the video stream entering the digitizer, and signal output, the compressed signal, on the upper right, after the buffer. There are six horizontally aligned blocks spanning the space between signal input and output (we must count the "+" item as a legitimate vertex). The edges between them are all directed to the right, making the set a linear chain, with numerous "side" connections. All blocks above and below the horizontal sequence are involved with feedback and feedforward controls related to image motion and rate throttling.

Initially the automated analysis would proceed much as in the previous graph example, in this case allowing vertices to be rectangles with internal text labels. Walking the parse structure would again construct a representation of the graph topology. Using the geometrically identified horizontal backbone, a dual analysis of the geometry and topology would indicate that there are four edges leaving the backbone from below and two from above. This would identify the subgraphs to be collapsed. The summary redrawing could use all the elements from the backbone, unchanged as well as the block above the backbone, all in unmodified form. The only actual synthesis required would be that of the new lower block. Even in this case, the identical contact points of the four lower edges with the backbone elements could be used. That is, the algorithm would attempt to maintain the "mental map" of the original as much as it could in the summary. Incremental constraint solving could aid in the process.

8. Discussion and Summary

Clearly, the problem of summarizing diagrams in documents is a complex and multi-faceted one. The variety and complexity of diagrams and their multifarious relations to text will require a variety of summarization strategies — there will be no single approach that will

apply across the board. Reviewing the problems of automated summarization for our four examples, we see that example #1 required the collapsing of a tabular set of images that was numerically parameterized. Example #2 allowed us to delete an internal "meganode" in a graph. Example #3 dealt with the selection and merging of numerical data from a collection of x,y data plots. Example #4 was a complex block diagram in which the summary was guided by the geometrical structure (layout) of the original. The role of text in guiding the automation for these examples was itself complex, and we did not attempt to investigate that aspect in much detail.

Essentially nothing has been done in this new field of diagram summarization, save some interesting work on multimedia, e.g., (Maybury and Merlino, 1997). Our own work has progressed to the point that we can parse a wide variety of diagrams efficiently, producing structured descriptions that are the input required by any automated diagram summarizer. This paper has used four examples, explored in depth, to describe how the summarization process might work, including the description of simple topological and elision algorithms. We felt that a thorough-going discussion of examples was needed because diagram summarization is such a new idea that most readers would not be able to fall back on any prior knowledge of the topic — this paper is in many ways, a "first" in its field. We hope that it has laid out a practical agenda for exploring the problem in greater breadth and depth and building practical systems.

The world is hardly ready for the application of automated diagram summarization because too few documents contain graphics in vector form that could serve as input to the first parsing stage. That is, most diagrams available today in PDF and HTML formatted documents are in raster form, as images, even though many were originally created as vector diagrams! But in 1998 there has been a surge in proposals to the World-Wide Web Consortium for various vector graphics standards (World-Wide Web Consortium, 1998), so that documents could contain more structured graphics, including *metadata* which describes the internal content of graphics items, using, for example, XML (World Wide Web Consortium, 1997).

Indeed, once we consider the possibility of including extensive metadata about diagrams within electronic documents, we can question how much of the complex analysis strategies we have discussed would be necessary. It should be possible to build intelligent authoring systems (IAS), applications that would allow the author to develop documents that include metadata about diagrams, and text for that matter. For a discussion of IAS strategies for text, see (Futrelle and Fridman, 1995). There is nothing new about metadata. Virtually anything beyond normal sequential natural language text can properly be called metadata. Web hyperlinks are an obvious example, but so are bibliographic citations, references to specific figures, etc. Authors already include such metadata in the papers they write. The only question is, what other types of

metadata might they agree to include in the future? Diagram metadata could include information that would indicate which diagram would be the one to retain in a summary, if only one were to be included, or at a more detailed level, which parts of a diagram would be suitable for a summary. The goal of a good IAS would be to make the inclusion of this type of additional information as effortless as possible for the already overworked author.

More and more major publications in science and various technical fields are moving online. Because of this, the desire to have the graphics in them treated as first-class content will spur the development of better formats, more metadata, and graphics manipulation systems such as diagram summarizers.

Acknowledgments

The author wishes to thank Tyler Chambers who was involved with and very helpful in developing the first draft of this paper. Thanks also to three anonymous reviewers whose insight and knowledge markedly transformed the original. Thanks finally to Carolyn S. Futrelle for her editing suggestions.

References

- Adobe Systems Incorporated (1996). *Portable Document Format Reference Manual*. Version 1.2.
- Ah-Soon, C. and Tombre, K. (1997). Variations on the Analysis of Architectural Drawings. In *ICDAR 97 (Fourth Intl. Conf. on Document Analysis and Recognition)*, (pp. 347-351). Ulm, Germany: IEEE Computer Soc.
- Bhatt, B., Birks, D. and Hermreck, D. (1977). Digital television: making it work. *IEEE Spectrum*, **34**(10), 19-28.
- Borning, A., Freeman-Benson, B. and Wilson, M. (1992). Constraint Hierarchies. *LISP and Symbolic Computation*, **5**(3), 223-270.
- Christensen, J., Marks, J. and Shieber, S. (1995). An Empirical Study of Algorithms for Point-Feature Label Placement. *ACM Trans. on Graphics*, **14**(3), 203-232.
- Cleveland, W. S. (1994). *The Elements of Graphing Data* (Revised Edition ed.). Summit, NJ: Hobart Press.
- DiBattista, G. (Ed.). (1997). *Graph Drawing*: Springer.
- Fateman, R. J. (1997). More Versatile Scientific Documents. Extended Abstract. In *ICDAR 97 (Fourth Intl. Conf. on Document Analysis and Recognition)*, (pp. 1107-1110). Ulm, Germany: IEEE Computer Soc.
- FHWA (1997). *Manual on Uniform Traffic Control Devices (MUTCD)*. Washington, DC: Federal Highway Administration — U. S. Dept. of Transportation. <http://www.ohs.fhwa.dot.gov/devices/mutcd.html>.
- Freeman-Benson, B., Maloney, J. and Borning, A. (1990).

- An incremental constraint solver. *Communications of the ACM*, **33**(1), 54-63.
- Futrelle, R. P. (1985). A Framework For Understanding Graphics In Technical Documents. In Expert Systems in Government Symposium (pp. 386-390): *IEEE Computer Society*.
- Futrelle, R. P. (1990). Strategies for Diagram Understanding: Object/Spatial Data Structures, Animate Vision, and Generalized Equivalence. In *10th International Conference on Pattern Recognition*, (pp. 403-408): IEEE Press.
- Futrelle, R. P., Carriero, C., Nikolakis, N. and Tselman, M. (1992a). Informational Diagrams in Scientific Documents. In *AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, (pp. 185-188). Stanford University.
- Futrelle, R. P. and Fridman, N. (1995). Principles and Tools for Authoring Knowledge-Rich Documents. In *DEXA 95 (Database and Expert System Applications) Workshop on Digital Libraries*, (pp. 357-362). London, UK.
- Futrelle, R. P. and Kakadiaris, I. A. (1991). Diagram Understanding using Graphics Constraint Grammars. In S. G. Tzafestas (Eds.), *Engineering Systems with Intelligence - Concepts, Tools and Applications* (pp. 73-81). Boston, MA: Kluwer Academic Press.
- Futrelle, R. P., et al (1992b). Understanding Diagrams in Technical Documents. *IEEE Computer*, **25**(7), 75-78.
- Futrelle, R. P. and Nikolakis, N. (1995). Efficient Analysis of Complex Diagrams using Constraint-Based Parsing. In *ICDAR-95 (Intl. Conf. on Document Analysis & Recognition)*, (pp. 782-790). Montreal, Canada.
- Graf, W. H. (1998). Constraint-Based Graphical Layout of Multimodal Presentations. In M. T. Maybury & W. Wahlster (Eds.), *Readings in Intelligent User Interfaces* (pp. 263-285). San Francisco: Morgan Kaufmann.
- Gunn, M. D., et al (1998). A B-cell-homing chemokine made in lymphoid follicles activates Burkitt's lymphoma receptor-1. *Nature*, **391**(1669), 799-803.
- Ho, T. K. and Baird, H. S. (1997). Large-Scale Simulation Studies in Image Pattern Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**(10), 1067-1079.
- Hower, W. and Graf, W. H. (1996). A Bibliographical Survey of Constraint-Based Approaches to CAD, Graphics Layout, Visualization, and Related Topics. *Knowledge-Based Systems*, **9**(7), 449-469.
- Knuth, D. E. (1968). Semantics of Context-Free Languages. *Mathematical Systems Theory*, **2**, 127-145.
- Kosslyn, S. M. (1994). *Elements of Graph Design*. New York: W. H. Freeman.
- Levesque, H. J. (1987). Making Believers out of Computers. *Artificial Intelligence*, **30**, 81-108.
- Levesque, H. J. (1992). Logic and the Complexity of Reasoning. *J. of Philosophical Logic*, **17**, 355-389.
- Luo, H., Kasturi, R., Arias, J. F. and Chhabra, A. (1997). Interpretation of Lines in Distributing Frame Drawings. In *ICDAR 97 (Fourth Intl. Conf. on Document Analysis and Recognition)*, (pp. 66-70). Ulm, Germany: IEEE Computer Soc.
- Mackinlay, J. D. (1987). Automating the Design of Graphical Presentations of Relational Information. *ACM Trans. Computer Graphics*, **5**(2), 110-141.
- Marriott, K. (1994). Constraint Multiset Grammars. In *Visual Languages 94: IEEE Computer Soc.*
- Marriott, K., Meyer, B. and Wittenburg, K. (1998). A Survey of Visual Language Specification and Recognition. In K. Marriott & B. Meyer (Eds.), *Visual Language Theory* : Springer Verlag.
- Maybury, M. and Merlino, A. (1997). Multimedia Summaries of Broadcast News. In *Conference on Intelligent Information Systems*.
- Maybury, M. T. and Wahlster, W. (Ed.). (1998). *Readings in Intelligent User Interfaces*. San Francisco: Morgan Kaufmann.
- Misue, K., Eades, P., Lai, W. and Sugiyama, K. (1995). Layout Adjustment and the Mental Map. *J. Visual Languages and Computing*, **6**, 183-210.
- Nikolakis, N. (1996) *Diagram Analysis using Equivalence and Constraints*. Ph.D. thesis, Northeastern University.
- Norvig, P. (1991). Techniques for Automatic Memoization with Applications to Context-Free Parsing. *Computational Linguistics*, **17**(1), 91-98.
- Paice, C. D. (1990). Constructing literature abstracts by computer: Techniques and prospects. *Information Processing & Management*, **26**(1), 171-186.
- Richardson, P. J. (1997). Blocking adenosine with antisense. *Nature*, **385**(20 February), 684-685.
- Seligmann, D. D. and Feiner, S. (1991). Automated Generation of Intent-Based 3D Illustrations. *Computer Graphics*, **25**(4), 123-132.
- Sloman, A. (1995). Musings on the Roles of Logical and Non-Logical Representations of Intelligence. In B. Chandrasekaran, J. Glasgow & N. H. Narayanan (Eds.), *Diagrammatic Reasoning. Cognitive and Computational Perspectives* (pp. 7-32). Menlo Park, CA, Cambridge, MA: AAI Press, MIT Press.
- Stenning, K. and Inder, R. (1995). Applying Semantic Concepts to Analyzing Media and Modalities. In B. Chandrasekaran, J. Glasgow & N. H. Narayanan (Eds.), *Diagrammatic Reasoning. Cognitive and Computational*

Perspectives (pp. 303-338). Menlo Park, CA, Cambridge, MA: AAAI Press, MIT Press.

Tufte, E. R. (1985). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

Tufte, E. R. (1990). *Envisioning Information*. Cheshire, CT: Graphics Press.

Tufte, E. R. (1997). *Visual Explanations*. Cheshire, CT: Graphics Press.

World Wide Web Consortium (1997). *Extensible Markup Language (XML)*. <http://www.w3.org/XML/>.

World-Wide Web Consortium (1998). *Technical Reports (covering XML, PGML)*. <http://www.w3c.org/TR/>.