General Diagram-Recognition Methodologies

Dorothea Blostein

Computing and Information Science
Queen's University, Kingston Ontario, Canada, K7L 3N6
blostein@qucis.queensu.ca

The field of diagram recognition faces many challenges, including the great diversity in diagrammatic notations, and the presence of noise and ambiguity during the recognition process. To help address these problems, research is needed into methods for acquiring, representing, and exploiting notational conventions. We review several frameworks for diagram recognition: blackboard systems, schema-based systems, syntactic methods, and graph rewriting. Next we discuss the need for a computationally-relevant characterization of diagrammatic notations, the need to exploit soft constraints during diagram recognition, and the possibility that diagram generators may provide a useful source of information about notational conventions.

Keywords: diagram recognition, notational conventions, diagrammatic notations, contextual information, diagram classification, blackboard systems, schema-based systems, graph rewriting

1. INTRODUCTION

A variety of diagrammatic notations are widely used in society; examples include notations for music, math, architecture, and logic circuits. These diagrammatic languages use two-dimensional arrangements of symbols to transmit information. Understanding a diagram involves two activities: symbol recognition and symbol-arrangement analysis. Here we focus on the latter: analysis of the spatial arrangement of a group of symbols, relative to the given notational conventions of a 2D language, to recover the information content of a diagram. Important open questions in this area include:

- How should notational conventions be represented? We need systematic methods for representing conventions in a format suitable for computation. Current diagram recognition systems tend to use ad-hoc methods for representing notational conventions (e.g. [BlBa92] reviews music-notation understanding). Comprehensive expositions of notational conventions have been published (e.g. [Read79] for music notation); these present information in a style suited for human readers, but unsuited for computation. Where computationally-oriented descriptions of notations have been developed, most are oriented toward *generation* of the notation rather than *recognition* of the notation (e.g. [Rous88], [Knut79]).
- How can noise and uncertainty be handled? How should context be applied to reduce errors and uncertainty in segmentation and symbol-recognition?
- How can the great diversity of diagram types be handled? We need general, systematic methodologies for organizing recognition systems, representing notational conventions, and using contextual information to reduce errors. Ideally, such research would eventually result in a recognition system that is reconfigurable for various diagrammatic notations.

Much of the current research in diagram recognition is directed at recognizing some particular type of diagram. We hope that over time this work gives rise to a technology for general diagram recognition. An appealing analogy is provided by compiler technology, where general techniques for parsing and code generation greatly simplify the task of constructing compilers for new source and target languages. However, diagram-recognition methods may not prove as portable or generalizable, due to the great diversity among diagrammatic notations, and due to the tremendous problems arising from noise and uncertainty in the input.

2. SIX DIAGRAM-RECOGNITION PROCESSES

Diagram recognition consists roughly of the following processes:

(1) early processing -- noise reduction, de-skewing, etc
(2) segmentation, to isolate symbols
(3) symbol recognition
(4) identification of spatial relationships among symbols
(5) identification of logical relationships among symbols   symbol-arrangement analysis
(6) construction of meaning

All of these processes require knowledge of notational conventions. For example, noise reduction methods (1) must preserve the presence of small or thin symbols, like decimal points in math and duration dots in music. Segmentation (2) needs information about how symbols overlap; an example is the separation of notes from staff-lines in music. Symbol recognition (3) needs information about symbol appearance; perhaps a font defining fixed symbols, and structural descriptions for parameterized symbols. The identification of spatial relationships (4) requires information about which spatial relationships are significant for encoding information. The last two processes (identification of logical relationships among symbols, and construction of meaning), rely heavily on knowledge of notational conventions Further research is required to clarify how notational conventions should be represented and used.

World knowledge plays an important role in diagram recognition. For example, knowledge of disassembly and kinematics is used for the recognition of engineering drawings [VaTo94]. Similarly, human recognition of music notation uses the reader's knowledge of music theory and compositional styles. The acquisition, representation, and use of world knowledge is a broad and interesting topic, but one that lies beyond the scope of this paper.

While we find it useful to discuss diagram recognition in terms of the above six processes, the processes are not necessarily clearly delineated in an implementation, and they need not be performed in the indicated order. For example, partial identification of spatial and logical relationships can be performed prior to symbol recognition, as in math layout analysis [OkMi91]. After symbol identity is known, more detailed identification of these relationships is possible. (The identity of a symbol helps to determine which spatial and logical relationships are most meaningful. For example, in music notation the vertical location of a notehead or accidental conveys critical information, whereas the vertical location of a stem-end does not.)

Concurrency among the six recognition processes is possible, and can be useful in handling ambiguity, noise, and uncertainty. Two basic control strategies for handling uncertainty are:

- <u>Sequential processing with lists of alternatives.</u> All possible interpretations are carried along from one recognition stage to the next. For example, the symbol recognizer can produce a list of alternatives (including "noise") for each symbol it recognizes. During symbol-arrangement analysis, constraints are applied to eliminate alternative interpretations. Interleaving of constraint-application and information recovery may be required, since some constraints can only be applied once partial recognition results are obtained [Fahm95].

- <u>Contextual feedback.</u> Concurrent execution of recognition processes permits higher-level, contextual feedback to compensate for noisy input or to reject erroneous input. Blackboard systems, for example, provide such a control mechanism.

Sequential processing is used in [BaIt94], with some discussion of limitations imposed by the need to carry forward all surviving alternatives. Sequential processing does offer the advantage of isolating portions of the computation for intensive study; we use this approach to focus on symbol-arrangement analysis without the need to study symbol recognition [Fahm95]. A contextual feedback organization, on the other hand, offers increased flexibility in handling uncertainty, and may be necessary when the space of possible interpretations is extremely large. Further research is needed to develop general methods of providing contextual feedback.

3. CLASSES OF DIAGRAM-RECOGNITION KNOWLEDGE

The following classes of knowledge are present in a diagram-recognition system, either procedurally or declaratively. The implementation may mix these together, or may enforce a separation between them.

- Knowledge specific to the current diagram. A unified data structure can be used to represent the spectrum of interpretation stages, including the original image, partial interpretations, and the final interpretation. Practical advice about such a data structure is provided by [BaIt94].
- Knowledge about the diagrammatic notation. The recognition system needs knowledge of notational conventions, but how best to represent notational conventions is very much an open question. Selected types of domain knowledge have well-studied data structures associated with them, such as dictionaries used in text recognition. The acquisition of domain knowledge is discussed in Section 6.
- Knowledge about diagram recognition. Generally applicable operations include thresholding, line finding, and vectorization. Such general knowledge can form the kernel of a diagram recognition system, with the addition of domain-specific knowledge bases to configure the system for recognition of particular diagrammatic notations [Past94].

4. FRAMEWORKS FOR DIAGRAM RECOGNITION

Various system organizations and various processing techniques have been proposed for general diagram recognition. The appendix provides a brief review of various recognition frameworks, and how they have been applied to diagram recognition and document image analysis. Frameworks that are discussed include blackboard systems, schema-based systems, syntactic methods, computational vision models, and graph rewriting.

The recognition frameworks make different provisions for handling various aspects of the recognition problem. Here we consider three aspects: representing levels of interpretation, processing noise and ambiguity, and representing knowledge about notational conventions.

<u>Representing Levels of Interpretation</u>

Levels of interpretation figure prominently in blackboard systems: recognition hypotheses on the blackboard are stored at different levels of interpretation. Schema-based systems use instance hierarchies to represent composition of objects into more complex objects. The class hierarchy represents specialization of knowledge about objects. Syntactic methods create levels of interpretation through the application of rewriting rules. Ordered graph rewriting similarly applies rewrite rules to create levels of interpretation; rule ordering offers explicit control over this process.

<u>Processing Noise and Ambiguity</u>

In a blackboard system, noise and ambiguity are processed by posting alternate recognition hypotheses on the blackboard, and eliminating inconsistent hypotheses as contextual information becomes available. This is a very general framework. The implementer is free to choose any search procedures and ambiguity-reduction algorithms; these are coded into the knowledge sources and the control component of the blackboard system. Schema-based systems add various mechanisms to handle noise and ambiguity. For example, the Mapsee system [MuMH88] uses schemata to construct a scene constraint graph; then constraints are propagated by a network consistency algorithm. The Anon system [JoPr92] uses a strategy grammar to direct schema-based construction of recognition hypotheses. Syntactic methods can handle noise through error-correcting parsing (typically a time-consuming operation for diagrams), or through stochastic grammars (as in [Chou89]). Ordered graph rewriting can handle noise in various ways, including explicit error-correcting production rules [Bunk82], or discrete relaxation [Fahm95].

<u>Representing Knowledge about Notational Conventions</u>

A central problem is representation of knowledge about notational conventions. In a blackboard system, this knowledge is stored in the knowledge sources; knowledge sources can contain an arbitrary mixture of declarative and procedural code. Schema-based system use schema classes (with inheritance) to define notational conventions. Additional control mechanisms may be used as well. Syntactic methods store notational conventions in rewrite rules, which typically have associated attributes and attribute computations (e.g. [Ande77]). Ordered graph rewriting encodes notational conventions both in the graph rewrite rules, as well as in the ordering imposed on rule application.

Criteria for Choosing a Recognition Framework

We are not aware of criteria which can be used to choose the most appropriate recognition framework for a given application. Existing recognition systems illustrate that a general framework (e.g. blackboard system, schema-based system) can be adapted in many different ways. Unfortunately, it is difficult to determine the relative merits of the various adaptations. Also, the recognition frameworks are not mutually exclusive, but can be combined in various ways. For example, the general blackboard architecture could be combined with the use of schemata or graph rewriting. No matter what recognition framework is chosen, the creation of a diagram recognizer remains a difficult and time-consuming task.

This concludes our discussion of frameworks for diagram recognition. We now turn to two issues that are important to the advancement of diagram-recognition technology: the need for a computationally-relevant characterization of diagrams, and the need to exploit soft constraints during diagram recognition.

5. COMPUTATIONAL CHARACTERIZATION OF DIAGRAMS

It is difficult to formulate generalizations based on existing diagram-recognition research. One reason for this is the lack of a characterization or classification of diagram types. Without a characterization of diagrams, it is difficult to delineate the range of applicability of a particular diagram-recognition technique. Several different diagram characterizations could be computationally relevant:

- Classification of diagram types. Based on such a classification, a researcher could state that a recognition method applies to all diagrams of type “line drawing”, “graph based”, or whatever else.
- Classification of notational elements common to various diagram types. Researchers could then identify techniques to solve common subproblems. For example, many diagrams include lines (so “line detection” and “separation of lines and overlapping symbols” are general subproblems). Symmetry is important in some diagrams (the “symmetry detection” problem). In many notations, points or lines are labeled by placing alphanumeric strings nearby (the “label association” problem). Various notations use coordinate axes -- full coordinate systems for maps and y=f(x) plots, partial coordinate systems for pitch and duration in music -- making “coordinate system recognition and interpretation” a common problem. Various symbol-recognition problems can be identified, some arising from notational primitives with fixed appearance (e.g. a note-head in music), others arising from parameterizable primitives (e.g. a beam or slur in music), and yet others arising from highly-complex, irregularly shaped primitives (e.g. lines in maps and y=f(x) plots).

Unfortunately, existing characterizations and classifications of diagrams are not computationally-oriented. For example, the recent article by Lohse et al. [LBWR94] provides many interesting examples of diagrams, and classifies these by asking human subjects to group together diagrams that seem similar. While the resulting classification is interesting, it is not directly useful for analyzing or generalizing diagram recognition methods. Another classification of diagrammatic notations is presented by Bertin [Bert83], but again not in a computationally-useful form.

Most diagrammatic notations are only semi-standardized, allowing many variations and drawing styles. For example, Tufte [Tuft83] describes numerous methods for displaying quantitative information, and encourages the designer of an illustration to be creative in the use of diagrammatic notations.

Computational characterizations of diagrammatic notations will emerge over time, as a result of ongoing work in the recognition of diverse diagrammatic notations. However, it is difficult to devise precise and impartial characterizations of notations based on experience with particular diagram recognizers: our impressions about diagram recognition often depend as much on the implementation of the given recognizers as on the inherent characteristics of the diagrammatic notation. Keeping these shortcomings in mind, we suggest a few dimensions along which notations could be characterized. We illustrate these ideas with comments based on our work in recognizing music and math notation [FaBl93] [Fahm95] [GrBl95].

- Subtlety of relative symbol placement. This characteristic of a notation affects how difficult it is to determine logical relationships among symbols from spatial relationships among symbols. In math notation, subtleties in the relative placement of symbols convey critical information. For example, minute changes in placement change our perception of whether two symbols are in a *subscript* or an *implied multiplication* relationship. In reading music notation, spatial relationships are less of a problem: staff lines provide a clear coordinate system for interpreting symbol placement. Thus, white-space plays an important role in the interpretation of math, more so than in music.
- Redundancy in the notation. Most diagrammatic notations have some redundancy in the representation of information. This provides a recognition system with constraints and cross-checks, particularly useful for the interpretation of noisy images. Although we cannot quantify this, it is our impression that music notation has more redundancy than does math notation. For example, image noise may make it difficult to determine whether a notehead is filled or unfilled; music notation offers redundant clues, such as "the number of beats in a measure matches the time signature", and "in multi-voice music, notes are placed so that beats line up across voices". Math notation offers less redundancy for resolving symbol-recognition ambiguities.
- Use of non-local symbol interactions. Both math and music notation rely on non-local symbol interactions, but in different ways. Math notation has a recursive structure, which results in relationships among symbol pairs -- such as [and], or ∫ and dx -- that are separated by subexpressions. Due to this recursive structure, bottom-up processing of math notation can automatically construct many of the long-distance relationships. Math interpretation is complicated by the fact that symbols in many different positions can potentially affect the meaning of a given symbol; there is a dense web of interdependencies among symbols. Music notation, on the other hand, lacks a recursive structure, requiring more explicit processing of non-local symbol interactions (such as between a clef and a note, or between a sharp and all similarly-pitched notes following in the measure). The search for non-local symbol interactions is more constrained in music than in math; for example, only one direction must be searched to find the clef associated with a note.

The above comparisons between music and math notation are preliminary and imprecise. Interesting research issues are involved in improving these characterizations. Precise definitions are needed for terms such as *redundancy* or *non-local symbol interactions* in a notation. Means are needed to measure these quantities in a manner that is independent of a particular recognition-technique.

6. ACQUIRING AND EXPLOITING NOTATIONAL CONVENTIONS

Notational conventions supply us with the syntactic and semantic constraints needed to disambiguate and interpret a noisy document-image. Thus, the acquisition, representation and exploitation of notational conventions is central to diagram recognition.

6.1 What can we learn from Diagram *Generators*?

Computer processing of diagrammatic information includes recognition, editing, and generation of diagrams. Traditionally diagram recognition and generation systems have been written independently of one another. Current generation technology is more advanced than recognition technology -- many usable diagram generators are on the market (packaged with diagram editors), whereas diagram recognizers are in the research stage. Can diagram recognizers can be improved by exploiting the knowledge and experience embodied in diagram generators?

A recent communication-theoretic approach to document-image decoding uses the explicit modeling of document-image generation as a central part of a recognition system [KoCh94]. To control the search space, the image-generation models are limited to finite state machines. Although this results in simplistic image-generation models, impressive recognition results are obtained. Our current suggestion is to exploit far more sophisticated image-generation models, as are embodied by the notational conventions used in current commercial diagram-generators.

There is significant overlap in the notational conventions used for generation and recognition. However, complications arise due to the following differences in the two diagram processing tasks. First, diagram generation involves aesthetic decisions, knowledge of which are not crucial to recognition. Second, diagram recognition must deal with noise and uncertainty, problems which do not arise in generation. This is illustrated in Figure 1.

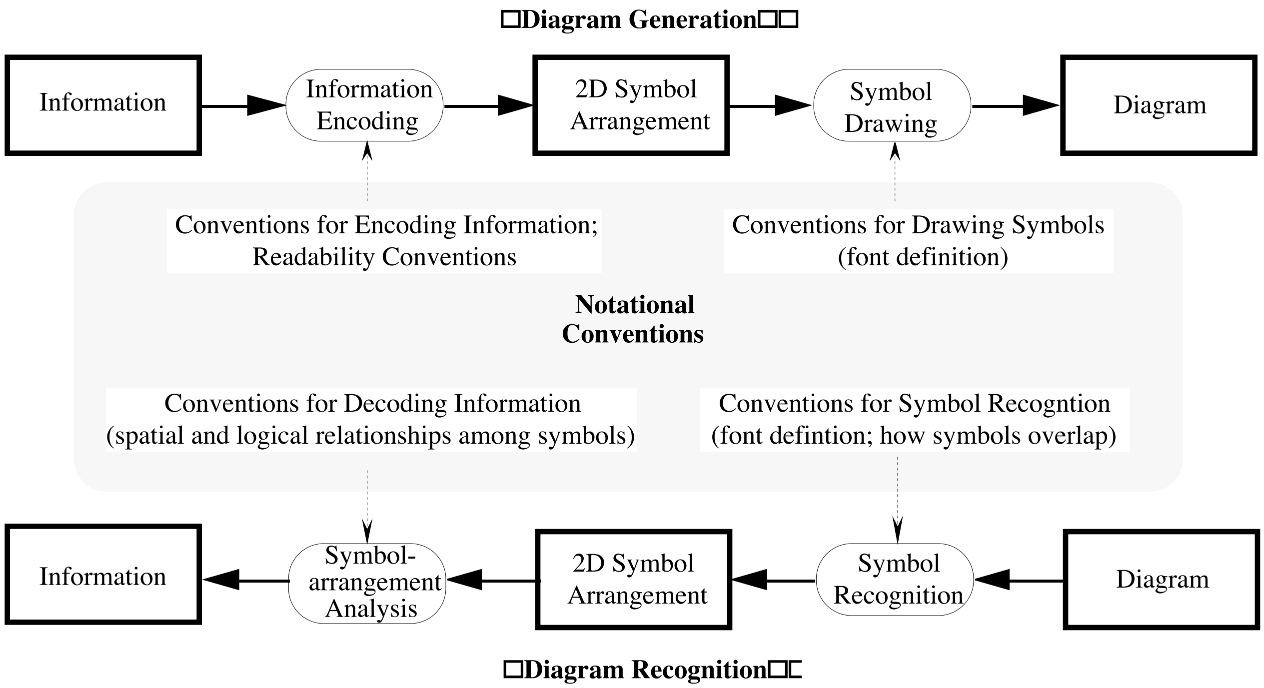

Figure 1 Diagram recognition and diagram generation systems both use notational conventions. For diagram generation, notational conventions dictate how to create an aesthetically pleasing diagram that encodes the given information. For diagram recognition, notational conventions dictate how to find logical relationships among symbols, and how to interpret these to recover the information content of a given diagram. (For simplicity, sequential processing is shown; in practice, diagram recognition can include feedback and concurrency between symbol recognition and symbol-arrangement analysis.)

Diagram-aesthetics are critical for human readability of a diagram, but current computer recognition systems do little to exploit these aesthetics. Ignorance of readability conventions simplifies the recognition task, but makes the recognizer less robust. In the following sections, we argue for the importance of exploiting readability conventions, and for the possibility that existing diagram generators are a valuable source of information about readability conventions.

6.2 Semantics and Readability: Hard and Soft Notational Conventions

Historically, diagrammatic languages arose as vehicles for communication, geared toward encoding information in a form that is easily readable by humans. For example, the lengthy evolution of music notation was shaped by the need to transmit pitch, duration, and phrasing information in a form that could be quickly read by performers [Slob81]. Thus notational conventions can be roughly characterized as being either *semantic conventions*, geared toward encoding information, or *readability conventions*, geared toward presenting the information in a form suited to human perception.

The distinction between semantic and readability conventions is fuzzy: any particular notational convention contributes both toward information-transmission and readability, but in varying proportions. Semantic conventions are small in number, and most of them can be described as local constraints on the diagrammatic notation. Readability conventions, which are sometimes referred to as aesthetic conventions, are large in number, difficult to state precisely, and complex to encode computationally. We illustrate this by considering notations for graphs and for music.

The diagrammatic notation for graphs has only few semantic conventions, such as: (1) graph structure is encoded by lines that connect nodes, and (2) node labels are written inside nodes. Readability conventions for graphs concern layout aspects such as limiting the number of edge crossings or edge bends, and spacing nodes appropriately.

The diagrammatic notation for music has a rich set of semantic and readability conventions [Read79]. Most semantic conventions are spatially localized, such as the use of noteheads, dots, flags, and beams to encode duration. An example of a non-local semantic convention is the propagation of accidentals from one note to a same-pitch note later in the measure. Numerous readability conventions govern the placement and spacing of music symbols. Of course, many notational conventions (such as "non-staff-line music symbols generally don't overlap") have important effects on both information-encoding and readability.

Two basic approaches are possible in using readability conventions for diagram recognition. The first is to simply ignore these conventions. This is the approach taken by most current diagram recognizers: they rely on hard semantic conventions and remain ignorant of readability conventions that are only "generally true". The second approach is to build knowledge of soft conventions into diagram recognizers. This provides the recognizer with much more contextual information for disambiguating noise and uncertainty in the input image. Since readability conventions strongly reinforce the information content of a diagram, a diagram-recognizer will be more robust if it is capable of interpreting the soft information provided by layout and spacing cues. These observations echo frequent comments in the diagram-recognition literature regarding the need to use context to resolve noise and ambiguity in the input image.

6.3 Sources of Information about Notational Conventions

Diagrammatic notations have complex notational conventions. Identifying and encoding these notational conventions is difficult. Currently, descriptions of notational conventions are available in the following forms.

- <u>Written descriptions oriented toward human use</u>: Comprehensive expositions of notational conventions have been published (e.g. [Read79] for music notation), but these present information in a style suited for human readers. (The typical reader of such a book is engaged in solving problems of typesetting the notation, not in solving problems of reading the notation.) Information is presented by example, with various notational conventions mixed together in each example. Unfortunately, no preconditions are stated for the rules and examples -- humans "use their judgment" in deciding when and how to apply the rules that are illustrated by example. This use of judgment is very difficult to mimic computationally.
- <u>Written descriptions oriented toward computation</u>: Where computationally-oriented descriptions of notations have been developed, most are oriented toward *generation* of the notation rather than *recognition* of the notation (e.g. [Rous88], [Knut79]).
- <u>Algorithmic and declarative descriptions built into diagram-recognition systems</u>: Current diagram recognition systems tend to use ad-hoc methods for representing notational conventions. The emphasis is on semantic constraints; few readability constraints are included.
- <u>Algorithmic and declarative descriptions built into diagram-generation systems</u>: Diagram generators contain comprehensive, though usually proprietary, descriptions of notational conventions. Much knowledge and experience is embodied in diagram-generation systems. For example, it is common for

music-editing systems to be under development for a decade or more [Belk94]. Both semantic and readability constraints are included in diagram generators.

These existing descriptions of notational conventions provide valuable resources for the further development of diagram generators and recognizers. Given the large variety of diagrammatic notations [Bert83] [Tuft83] [LBWR94], it would be desirable to develop general approaches to recognition and generation, applicable to whole classes of diagrams. Many interesting research problems are involved.

7. CONCLUSION

This paper has discussed the following research problems, aimed at furthering the emergence of a technology for general diagram recognition.

- Develop improved methods for acquiring, representing, and exploiting notational conventions. Notational conventions dictate how a given diagrammatic notation encodes information as a two-dimensional arrangement of symbols.
- Engage in comparative study of existing diagram-recognition frameworks. Here we briefly reviewed blackboard systems, schema-based systems, syntactic methods, and graph rewriting. Interesting commonalities and differences emerge when existing systems are studied.
- Develop a computationally-relevant characterization of diagrammatic notations. This will permit us to identify and exploit commonalities among diagrammatic notations.
- Develop methods to exploit soft constraints during diagram recognition. Many notational conventions concern diagram readability, and thus are observed "most of the time". These soft conventions provide important cues, but are unused by current diagram recognition systems.
- Consider diagram generators (built into diagram-editing systems) as a possible source of information about notational conventions, particularly about soft conventions.

Advances in the acquisition and exploitation of notational conventions can do much to further the state-of-the-art in diagram recognition.

APPENDIX

This appendix reviews existing frameworks for diagram recognition, including blackboard system, schema-based systems, syntactic methods, and graph rewriting.

A.1 Blackboard Systems

The blackboard architecture provides a general and flexible framework for combining diverse knowledge sources. (The name derives from an analogy to human experts communicating via a blackboard.) Here we discuss selected blackboard systems for document image analysis, including both text and diagram recognition. Existing blackboard-based recognition systems provide a wealth of experience in designing the detailed configuration of a blackboard system.

The three main parts of a blackboard system are a blackboard data structure (which contains recognition hypotheses), knowledge sources, and a control component [JaDB89]. Multiple, conflicting recognition hypotheses can be represented; knowledge sources commonly assign confidence values to the hypotheses. Authors mention various advantages of blackboard systems. These include the ability to integrate new knowledge sources into the system, the support for multiple hypotheses, the use of contextual feedback from high-level processing steps to low-level processing steps, and the ability to locally redo processing with new premises [Senn94]. Blackboard systems are able to handle both structure and randomness in the input, by acquiring evidence from diverse knowledge sources, without putting undo emphasis on any single source [WaSr89]. The control structure can invoke knowledge sources such that the effort expended in gathering evidence is in accordance with input complexity [WaSr89]. The blackboard architecture provides a flexible and adaptive order of processing; for example, clues can be opportunistically exploited to relate and interpret the text and diagram constituting a physics problem [NoBu93].

A blackboard is commonly divided into multiple levels of abstraction. Here we summarize the levels used in a sampling of systems. Kato and Inokuchi use five levels (image, primitive, symbol, meaning, goal) for music recognition [KaIn90a], and four levels (input diagram, symbol hypotheses, diagram hypotheses, recognition result) for circuit-diagram analysis [KaIn90b]. Vaxivière and Tombre use one level for each phase in engineering drawing analysis: lines and blocks, shafts, symmetric entities, functional setups [VaTo94]. Novak and Bulko use five levels (picture, text, picture-model, text-model, problem-model) to interpret the text and diagram defining a physics problem [NoBu93]. Sennhauser's blackboard for text recognition contains hypotheses related by a tree structure, with hypotheses at the page, block, chunk, and symbol level [Senn94]. Wang and Srihari use five levels (raw image, thresholded image, labeled image, text-line, and block levels) for address-block location in images of mail pieces; the main levels are divided into sublevels (seventeen in all) distinguishing e.g. machine-generated and hand-generated addresses [WaSr89]. The hierarchy of levels, models and algorithms used in text recognition are discussed in [Srih93]. Four main levels are used: the image level (divided into binary and grayscale levels), the character level, the linguistic level (divided into word, phrase, and sentence levels), and the textual structure level (divided into paragraph and document levels). Pasternak uses blackboard-driven building of aggregates in an adaptable drawing-interpretation kernel [Past94]; declarative geometrical constraints are applied to iteratively combine graphical objects into higher-level objects.

Knowledge sources can be organized in various ways. For example, Novak and Bulko suggest five types of knowledge-sources for symbol-arrangement analysis [NoBu93], which group related diagram elements, represent expectations about related objects, make inferences from common-sense rules and domain rules, relate diagram elements to *a priori* knowledge, and infer missing items.

Blackboard architectures provide a flexible control structure, which can be used to handle noise and uncertainty. In text recognition, character segmentation can be repeated if the classification results are unclear [Senn94]. A music recognition system can restore data to lower levels in the blackboard (for re-interpretation) if there is a contradiction at a higher level [KaIn90a]. This results in fast processing for clean input, with slower processing and more backtracking for noisy input. Similarly, in circuit-diagram recognition, the symbol hypotheses with the highest certainty factors are extracted first [KaIn90b]. Such processing allows cleaner parts of the input image to provide contextual information for interpreting noisier parts of the image.

Extensions to the basic blackboard system (knowledge sources, blackboard, and control component) are common. Here we summarize one such extension: Wang and Srihari's blackboard framework for object recognition in a highly variable set of images, including structured, partially structured, and unstructured images [WaSr89]. (The particular domain of interest is locating address blocks on mail pieces.) The six system components in this framework are a set of knowledge sources, the blackboard, a control mechanism (to select knowledge sources, combine evidence, update the focus-of-attention, and check termination), data for the control mechanism (a dependency graph that restricts the ordering of knowledge-source invocations, acceptance criteria for the object being recognized), a database of domain statistics (e.g. statistics about geometric features of logical blocks in a large sample of mail pieces), and a rule-based inference engine for interpreting the rules constituting a knowledge source. Each knowledge source contains rules for estimating the benefit and cost of using it, selecting parameters, and evaluating and interpreting results. Knowledge-source utility is estimated as a function of efficiency (fraction of knowledge-source-invocations that generate object candidates), effectiveness (fraction of generated candidates that turn out to be the desired object), average processing time, and the proportion of the input images to which the knowledge-source is designed to apply. The evidence generated by various knowledge-sources is combined using Dempster-Shafer's rules of combination, with the restriction that only singleton labels are assigned belief. Many aspects of this very interesting blackboard framework rely on the relatively simple structure of the object-recognition problem, but could perhaps be adapted to the diagram-recognition problem.

Blackboard architectures provide a means of coordinating fairly independent knowledge sources. This helps reduce the difficulties encountered in scaling up a recognition system. It is interesting to compare the scalability comments made by various authors. Sennhauser is quite optimistic about scaling up his blackboard system for text recognition [Senn94]. This system uses fairly regularly-structured, conceptually-simple knowledge sources, such as dictionaries and n-gram statistics; Sennhauser finds that the addition of these knowledge-source modules does not increase software complexity. Similarly, Wang and Srihari's blackboard framework isolates information about knowledge-source selection and utilization, so that changes to one knowledge-source do not cause side effects on other tools [WaSr89]. In contrast Vaxivière and Tombre are more cautionary in their discussion of scaling up a blackboard system for engineering drawing interpretation [VaTo94]. They use very high-level, irregularly-structured knowledge, including knowledge of disassembly and kinematics. Even their current, rather limited knowledge-base is somewhat difficult to manage, in maintaining consistency of rules, clarity, and an efficient search strategy. They conclude that their unstructured set of knowledge specialists is becoming unwieldy, and that the knowledge should follow a taxonomy, with hierarchical organization, perhaps as in a frame language.

A.2 Schema-based systems

Schema-based systems provide an alternative organization for diagram recognition. A schema class defines a prototypical drawing construct. During recognition a new schema instance is created to represent each drawing-construct that is found in the image. Two hierarchies are important in schema representations: the class hierarchy (to represent *specialization* relationships in knowledge organization) and the instance hierarchy (to represent *composition* of objects into more complex objects). Schema classes are organized into a tree-structured class hierarchy via the "subclass" relationship; this permits inheritance among schema-classes that describe related drawing constructs. Instantiated schemata are organized into an instance hierarchy via the "subpart" relationship; this provides the relationship between a schema and the smaller geometric entities that constitute it. Schemata can be used to represent both objects and relationships between objects, providing a uniformity of representation.

The Mapsee systems combine schema-based representations with constraint satisfaction, in order to interpret sketch maps [HaMa83] [MuMH88]. Schemata combine to form a *scene constraint graph*; constraints are propagated by a network consistency algorithm, to specialize the labels associated with schemata. Schema label sets have a specialization hierarchy, stating, for example, that both islands and mainland are landmasses, and both landmasses and waterbodies are geosystems. A constraint provides information such as "if a geosystem has a component river-system, road-system or mountain-range, then the geosystem-label is constrained to be a landmass". Mapsee-2 uses hypothesis trees to represent competing interpretations. Mapsee-3 instead uses labels (such as road, road/river, road/river/mountain) to represent competing interpretations; these labels are defined in a specialization hierarchy with multiple inheritance, so that specialization is not limited to a strict tree structure.

The Anon system [JoPr92] uses a three-layer structure (a strategy grammar, schemata, and an image analysis library) to recognize engineering drawings. At any given time, one schema is designated as the *controlling schema*. The controlling schema invokes the image analysis library, providing predictive information to guide the application of image analysis routines, and interpreting the results of the image analysis. The controlling schema encodes the image-analysis results as a token stream, which is passed up to the strategy grammar. The strategy grammar, an LR(1) grammar parsed by yacc, performs control actions (update the controlling schema, create a new schema, or designate a different schema to be the controlling one). Thus, spatial processing is directed by the controlling schema (where to look in the image, what to look for), whereas symbolic processing in managed by the strategy grammar (how to construct recognition hypotheses). The system is aimed at low and intermediate levels of diagram recognition: Anon analyzes an image to extract schemata representing drawing constructs. More global processes would be needed to integrate pieces of a drawing into a coherent whole. In contrast, the Mapsee system does perform global processing (via constraint propagation), but does not perform segmentation and symbol recognition (dots and line-segments are provided as input).

A.3 Syntactic Methods

Syntactic methods can describe the structure of a diagrammatic notation. Domain knowledge is represented by a grammar, which consists of a start symbol and a set of rewrite rules. Depending on the formalism, a rewrite rule replaces one substring by another, one subtree by another, or one subgraph by another [Fu82]. This section considers the use of grammars for diagram recognition. Non-grammar-based use of rewrite rules is also possible (Section A.5).

Anderson's grammar for the recognition of math notation provides an excellent case-study [Ande77]. The recursive nature of math notation is particularly well-suited for syntactic analysis. Anderson uses a set-based grammar (called a coordinate grammar). During top-down parsing, a production rule is given a set of symbols; the production rule applies spatial constraints to partition the symbol set into subsets, and assigns a syntactic goal to each subset. Anderson assumes the existence of a perfect symbol recognizer; recent work has eliminated this assumption [DiCM91]. Another syntactic approach to recognition of noisy math images uses stochastic grammars [Chou89].

Plex-grammars have been used to interpret dimensions in engineering drawings [CoTV93]. (A *plex* is an undirected graph; the nodes have named attachment points.) The parser mixes top-down and bottom-up processing. The processing of errors and uncertainty is delayed until after contextual information is obtained from higher-level analysis.

On a related note, a plex-grammar-based recognition system for three-dimensional objects is discussed in [LiFu89]. A 3D surface may be partially or totally occluded in the image plane. This makes it difficult to convert the input image into a three-dimensional plex to serve as input for the recognizer. A semantic-directed top-down backtrack recognizer is used, in which parsing directs the search for pattern primitives in the image.

A.4 Computational Vision Frameworks

Many of the problems we face in diagram recognition arise in general computational vision as well. Thus, frameworks formulated for computational vision are likely to contain ideas useful to us. Here we mention only one such framework. Joseph and Pridmore provide additional discussion of image understanding methods relevant to diagram recognition ([JoPr92], Section II).

Truvé describes a framework for high-level computational vision, based on multiple levels of interpretation, with feedback predictions [Truv90]. There are three phases to the process of going from one level of interpretation to the next: parsing, interpreting, and pruning (the *PIP paradigm*). The parsing phase uses local information to assign possible labels to image entities and groups of entities. The interpretation phase builds several interpretations, each assigning at most one label to a feature. The pruning phase uses global constraints to discard some interpretations. Parsing and pruning rules are expressed as multi-relational grammar rules (consisting of relations, constructors, and guards). The PIP paradigm applied to the sketch-map domain generates interpretations more efficiently than does Mapsee (Section A.2). So far, Truvé has applied this framework to two relatively simple domains: sketch maps and the blocks world.

A.5 Graph-rewriting

In our experience, graph-rewriting is a promising formalism for symbol-arrangement analysis. Graphs are a widely-used, flexible data structure, well-suited to the representation of document images. Graph-nodes represent entities in the physical image, or in the reconstructed logical representation. Graph-edges represent relationships among objects; these can range from physical relationships (e.g. adjacency or parallelism in the physical image), to logical relationships (e.g. a clef and a note in music notation, or matching brackets in math notation). Graph attributes, associated with nodes and edges, record non-structural information; initial attributes might record (x, y) image locations, whereas in the final graph, attributes record the meaning that has been extracted from the document image.

Graph rewriting can be used to transform an initial graph, derived from the original document image, into a final graph representing the logical structure and information content of the given document. Thus, the set of graph-rewriting rules encodes the *a priori* knowledge of the document type and its notational conventions.

Graph-rewriting rules can be organized in various ways: an unordered graph rewriting system, a graph grammar, an ordered graph-rewriting system (also called a programmed graph grammar), or an event-driven graph-rewriting system. These topics are discussed in the survey paper [BlFG95].

Ordered graph-rewriting has been used for various diagram-recognition applications, including circuit diagrams, music notation and math notation. In [Bunk82], ordered graph rewriting is proposed for the recognition of circuit diagrams. The high cost of parsing is avoided; instead ordered rule-application directly transforms an input graph (representing line-segment primitives in a circuit-diagram image) into an output graph (representing a circuit, with recognized components such as transistors and capacitors). Rule ordering is used, for example, to ensure that error-correcting rules are applied exhaustively (to close small gaps in lines and eliminate small overhanging lines) prior to the application of the component-recognition rules. This graph-rewriting approach is extended to music-notation recognition in [FaBl93]. The semantics of music notation depend both on nearby symbols (such as a sharp preceding a note) and on distant symbols (such as a distant key signature and a note). Thus, the necessary node-interactions cannot be predicted at the time of input-graph construction. Instead, a discrete input graph is used, and the necessary edges are built using Build, Weed, Incorporate phases of rewrite-rule application. (The Build phase creates edges between potentially-interacting nodes. The Weed phase deletes excess or conflicting associations. The Incorporate phase collapses the information from nodes and edges into the attributes of the remaining nodes.) This work is extended to address the uncertainty that may be associated with the identity of the notational primitives [Fahm95], and to perform recognition of mathematical notation [GrBl95]. We plan to investigate the use of graph rewriting in conjunction with a blackboard architecture. If data on the blackboard is stored as a graph, then selected knowledge sources could be coded as graph-rewrite rules.

References

[Ande77] R. Anderson, "Two Dimensional Mathematical Notation," in *Syntactic Pattern Recognition, Applications*, K. S. Fu editor, Springer 1977, pp. 147-177.

[BaIt94] H. Baird and D. Ittner, "Data Structures for Page Readers" *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 323-334.

[Belk94] A. Belkin, "Macintosh Notation Software: Present and Future," *Computer Music Journal*, Vol. 18, No. 1, pp. 53-69, Spring 1994.

[Bert83] J. Bertin, *Semiology of Graphics: Diagrams, Networks, and Maps*, University of Wisconsin Press, 1983.

[BlBa92] D. Blostein and H. Baird, "A Critical Survey of Music Image Analysis," in *Structured Document Image Analysis*, Eds. H. Baird, H. Bunke, and K. Yamamoto, Springer Verlag, 1992, pp. 405-434.

[BlFG95] D. Blostein, H. Fahmy, and A. Grbavec, "Practical Use of Graph Rewriting," Technical Report No. 95-373, Computing and Information Science, Queen's University, January, 1995.

[Bunk82] H. Bunke, "Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation," *IEEE Trans. Pattern Analysis and Machine Intelligence* 4(6), pp. 574-582, Nov. 1982.

[Chou89] P. Chou, "Recognition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar," *Proc. SPIE Visual Communications and Image Processing IV*, Philadelphia PA, pp. 852-863, Nov. 1989.

[CoTV93] S. Collin, K. Tombre, and P. Vaxiviere, "Don't Tell Mom I'm Doing Document Analysis; She Believes I'm in the Computer Vision Field," *Proc. Second Intl. Conf. on Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, pp. 619-622.

[DiCM91] Y. Dimitriadis, J. Coronado, and C. de la Maza, "A New Interactive Mathematical Editor, Using On-line Handwritten Symbol Recognition, and Error Detection-Correction with an Attribute Grammar," in *Proc. First Intl. Conf. on Document Analysis and Recognition*, Saint Malo, France, September 1991, pp. 242-250.

[FaBl93] H. Fahmy and D. Blostein, "A Graph Grammar Programming Style for Recognition of Music Notation," *Machine Vision and Applications*, Vol. 6, No. 2, pp. 83-99, 1993.

[Fahm95] H. Fahmy, “Reasoning in the Presence of Uncertainty via Graph Rewriting,” PhD Thesis, Computing and Information Science, Queen’s University, March 1995. (TR 95-382)

[Fu82] K. S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice Hall 1982.

[GrBl95] A. Grbavec and D. Blostein, “Mathematics Recognition Using Graph Rewriting,” *Third International Conference on Document Analysis and Recognition*, Montreal, Canada, August 1995.

[HaMa83] W. Havens and A. Mackworth, “Representing Knowledge of the Visual World,” *IEEE Computer*, October 1983, pp. 90-96.

[JaDB89] V. Jagannathan, R. Dodhiawala, L. Baum, Editors, *Blackboard Architectures and Applications*, Academic Press, 1989.

[JoPr92] S. Joseph and T. Pridmore, “Knowledge-Directed Interpretation of Mechanical Engineering Drawings,” *IEEE PAMI*, Vol. 14, No. 9, Sept. 1992, pp. 928-940.

[KaIn90a] H. Kato and S. Inokuchi, “The Recognition System of Printed Piano Music using Musical Knowledge and Constraints,” *Proc. IAPR Workshop on Syntactic and Structural Pattern Recognition.*, Murray Hill NJ, June 1990, pp. 231-248.

[KaIn90b] H. Kato and S. Inokuchi, “The Recognition Method for Roughly Hand-Drawn Logical Diagrams Based on Utilization of Multi-Layered Knowledge,” *Proc. 10th Intl. Conf. on Pattern Recognition*, Atlantic City NJ, June 1990, pp. 443-473.

[Knut79] D. Knuth, “Mathematical Typography,” *Bulletin of the American Mathematical Society*, Vol. 1, No. 2, March 1979.

[KoCh94] G. Kopec and P. Chou, “Document Image Decoding Using Markov Source Models,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 6, June 1994, pp. 602-617.

[LiFu89] W. Lin and K.S. Fu, “A Syntactic Approach to Three-Dimensional Object Recognition,” *IEEE Trans. Systems Man and Cybernetics*, Vol. 16, No. 3, May 1986, pp. 405-422.

[LBWR94] G. Lohse, K. Biolsi, N. Walker, H. Ruetter, “A Classification of Visual Representations,” *Communications of the ACM*, Vol. 37, No. 4, December 1994, pp. 36-49.

[MuMH88] I. Mulder, A. Mackworth, W. Havens, “Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach,” *IEEE Pattern Analysis and Machine Intelligence*, Vol. 10, No. 6, November 1988, pp. 866-879.

[NoBu93] G. Novak and W. Bulko, “Diagrams and Text as Computer Input,” *J. Visual Languages and Computing*, Vol. 4, 1993, pp. 161-175.

[OkMi91] M. Okamoto and B. Miao, “Recognition of Mathematical Expressions by Using the Layout Structure of Symbols,” in *Proc. First Intl. Conference on Document Analysis and Recognition*, Saint Malo, France, September 1991, pp. 242-250.

[Past94] B. Pasternak, “Processing Imprecise and Structural Distorted Line Drawings by and Adaptable Drawing Interpretation Kernel,” *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 349-363.

[Read79] G. Read, *Music Notation: A Manual of Modern Practice (Second Edition)*, Taplinger Publishing, New York, NY, 1979.

[Rous88] D. Roush, “Music Formatting Guidelines,” Technical Report OSU-3/88-TR10, Department of Computer and Information Science, The Ohio State University, 1988.

[Senn94] R. Sennhauser, “Integration of Contextual Knowledge Sources Into a Blackboard-based Text Recognition System,” *IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 211-228.

[Slob81] J. Sloboda, “The Uses of Space in Music Notation,” *Visual Language*, Vol. XV, No. 1, pp. 86-112, 1981.

[Srih93] S. Srihari, “From Pixels to Paragraphs: the Use of Contextual Models in Text Recognition,” *Proc. Second Intl. Conf. Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, pp. 416-423.

[Truv90] S. Truvé, “Image Interpretation Using Multi-Relational Grammars,” *Proc. Third International Conference on Computer Vision*, pp. 146-155, December 1990.

[Tuft83] E. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, 1983.

[VaTo94] P. Vaxivière and K. Tombre, “Knowledge Organization and Interpretation Process in Engineering Drawing Interpretation,” *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 313-321.

[WaSr89] C. Wang and S. Srihari, “A Framework for Object Recognition in a Visually Complex Environment and its Application to Locating Address Blocks on Mail Pieces,” *International Journal of Computer Vision*, Vol. 2, 1989, pp. 125-151.