

Towards Diagram Processing: A Diagrammatic Information System

Michael Anderson

From: AAAI-99 Proceedings. Copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Department of Computer Science
University of Hartford
West Hartford, CT 06117
anderson@hartford.edu

Abstract

We advocate the development of an agent capable of processing diagrammatic information directly in all its forms. In the same way that we will require intelligent agents to be conversant with natural language, we will expect them to be fluent with diagrammatic information and its processing. We present a methodology to this end, detail a *diagrammatic information system* that shows the merit of this line of research, and evaluate this system to motivate its future extensions.

Introduction

Of the set of behaviors that will be required of an artificially intelligent agent, a somewhat neglected member has been the ability to deal with diagrammatic information. Much attention has been paid to machine synthesis, recognition and understanding of natural language in both textual and audio forms. The understanding has been that such capabilities are required of an agent if it is expected to fully communicate with human beings and function in human environments. Much less attention has been given to machine understanding of diagrammatic information, an equally important mode of human communication. Effective capabilities in this mode will be crucial to an agent intended as a full partner in human discourse and activity. In the same way that we will require such agents to be conversant with natural language, we will expect them to be fluent with diagrammatic information and its processing. Ultimately, a machine with such capabilities will interact with a real world environment, rife with diagrammatic information, with a higher degree of autonomy than those without such capabilities.

The main thrust of diagrammatic reasoning research to date (from an artificial intelligence perspective) has been a search for computational efficiency gains through representations, and related inference mechanisms, that analogously model a problem domain. As this has been the aim of much of the seminal work in the field (e.g. Gelernter 1959; Larkin and Simon 1987), it is understandable that much effort has been expended in this direction. Although

it is arguable that some progress has been made through this line of research, we believe that the field's most important contribution will be the development of an agent that is capable of dealing directly with diagrammatic information in all its forms.

We envision a system that takes diagrams as input, processes these diagrams, abstracting information and drawing inferences from them alone and in concert with other forms of knowledge representation, and expresses this newly gained knowledge as output in the form of text, new diagrams, actions, etc. Although the approach taken by this system will not necessarily claim cognitive plausibility, the fact that human beings do these things as a matter of course will stand as proof by existence that such a system has been fashioned.

This *diagram processing system* will be comprised of a number of important components. It will require a means to input diagrams such as a vision component. It will require a way to internally represent diagrams. The diagrammatic representations so acquired will require storage, as will knowledge needed to deal with these representations, necessitating some storage management component. A processing component will be required that synthesizes and abstracts new knowledge from combinations of diagrammatic and other forms of knowledge representations. Various components will be required to use the new knowledge to produce desired output in a variety of situations.

Reflection on the design of these components raises a number of questions: What constitutes a diagram? In what form will diagrams be accepted as input by the system? How will diagrams be internally represented? How will knowledge be gleaned from diagrams? What is the nature and content of *a priori* knowledge that will be required? How will other forms of representation and inference be integrated with diagrammatic representations and inference? What is nature of the desired output? How will this output be produced? Etc. These are hard questions with a multiplicity of answers that in themselves generate more questions. They form the parameters of the problem. Our intent is to build a test bed in which various values for these parameters can be tested, compared and contrasted,

and ultimately forged into a single general purpose diagram processing system.

Following the methodology advocated in (Allen 1998), we 1) identify the particular intelligence we seek to study, 2) define a set of telescoping restricted tasks, 3) define evaluation criteria, 4) develop a model and system for our most restricted domain, and 5) evaluate this model and system for the next level of complexity.

Topic of Study

As we have stated, we are ultimately interested in developing an agent with full diagrammatic reasoning capabilities on par with human beings. If a picture is worth a thousand words, we would like to obviate the need for this text by being able to communicate with an agent directly via pictures or diagrams. An agent should be able to accept and understand such diagrammatic input from us and our environment as well as being able to produce such diagrams in its attempt to communicate diagrammatically representable notions to us.

Restricted Tasks Set

As this ultimate system is clearly beyond the grasp of current theory and technology, we refine it into a nested group of successively simpler tasks. Each nested level is a subset of all levels it is nested within and, further, a simplification of these containing levels. At the simplest level, we attempt to prove the merit of this line of research by developing a system that realizes the goal of this level. As we succeed at one level, we will attempt to build upon this success by applying what we have learned to the next less simplified level, expanding our solution to cover new problems it entails.

Given, as the outer most level, a system with *full diagrammatic reasoning capabilities*, we define as a simpler subset of this ultimate system a system that has *diagrammatic reasoning capabilities in some arbitrary task*. Examples of such systems are map understanding robots, scientific diagram abstracting agents, diagram generation systems, etc.

We choose, as the next level of simplification, to focus on *diagrammatic reasoning capabilities in one particular task* namely what we term *diagrammatic information systems*— systems that allow users to pose queries concerning diagrams, seeking responses that require the system to infer information from diagrams. As a final level of simplification we constrain any given instantiation of this diagrammatic information system to be knowledgeable about *particular* diagram types in a *particular* domain.

That said, we strive to develop a core of this diagrammatic information system that remains diagram type and domain independent, capable of accepting domain dependent diagrammatic and non-diagrammatic

knowledge. In this way, each body of knowledge produces a new instantiation of the diagrammatic information system knowledgeable in the particular domain and diagram types represented by this knowledge.

Evaluation Criteria

To evaluate systems at all levels of nesting, we compare its understanding of diagrammatic information with the understanding that a human being exhibits with the same information. In this manner, we evaluate an instantiation of a diagrammatic information system in a particular domain by the number of diagram types it can handle, the variety of queries it can respond to, and the quality of responses it gives to these queries. The level of success of such a system is measured by how it approaches, matches, or exceeds the performance of a human being reasonably capable within the chosen domain.

Model and System

As the first task we have set for ourselves is an implementation of a *diagrammatic information system*, we choose one possible set of values to the parameters of this problem.

We define a *diagram* to be a *tessellation* (tiling) of a planar area such that it is completely covered by atomic two dimensional regions or *tesserae* (tiles). Such a definition is broad enough to include arbitrarily small tesserae (points, at the limit), pixels, and, at the other end of the spectrum, domain-specific entities such as countries, regions, blocks, etc. Further, as this definition is not tied to any particular semantics, it is general enough to encompass all diagrams. Given the wide variety of semantic mappings employed by diagrams, a general definition that makes no semantic commitment is useful.

We sidestep a vision component by accepting bitmaps depicting diagrams as input to our system. As this is a likely output of a vision component, such a component can be appended later.

Our currently chosen approach gleans knowledge from diagrams by directly manipulating spatial representations of them. This approach is motivated by noting that, given diagrams directly input as bitmaps, any translation into another representation will require some form of direct manipulation of these bitmaps. In many cases, this translation is superfluous. Given this approach, we store input bitmaps directly with no further abstraction. This strategy not only allows us to manipulate spatial representations directly but, should the need arise, it will allow us to translate to any other representations as required.

We use, as a basis for this direct manipulation of diagrams, the theory of *inter-diagrammatic reasoning* (IDR) (Anderson and McCartney 1995, 1996). IDR



Figure 1: Vegetation in the United States

leverages the spatial and temporal coherence often exhibited by groups of related diagrams for computational purposes. Using concepts from set theory, image processing theory, color theory, and others, like diagrams are combined in ways that produce new like diagrams that infer information implicit in the original diagrams.

Knowledge required to process diagrams is likely to be both domain and diagram specific. Facts and rules pertinent to targeted domains are necessary as is information germane to processing diagram types represented. We represent this knowledge both diagrammatically and non-diagrammatically, as appropriate, constraining both domains and diagram types as necessary. We achieve integration of diagrammatic and non-diagrammatic knowledge and inferencing by providing an inter-lingua abstraction that furnishes a homogeneous interface to various modes of representation and inferencing, permitting inferences to be made with heterogeneously represented knowledge.

Our output is both diagrammatic and textual, meant for direct human consumption. Although we skirt other forms of output such as action or intermediate output intended for use by some other system, there is nothing in the nature of the processing that precludes use of its product in such ways.

Our first instantiation of a diagrammatic information system is informed about *cartograms* (maps representing information as grayscale or color shaded areas) of the United States.

An Example

As an example, consider the diagram in Figure 1. This is a cartogram that depicts in three levels of gray where each of the major vegetation types are situated in the United States. The darkest gray represents *forest*, medium gray represents *grassland*, and the lightest gray represents *desert*. Given this diagram as input to the system, as well as the semantics of the gray levels in this particular diagram, posing the



Figure 3: Location of grassland in the United States



Figure 2: Response to query: "Which states have grassland?"

query "Which states have grassland?" elicits the diagram in Figure 2 as a response from the system. In this diagrammatic response, each state in which grassland exists is represented by its shape in black positioned where the state lies within the United States. We use this example to examine the implementation of this instantiation of a diagrammatic information system in further detail.

Figure 1 is input to the system as a bitmap and is stored as such with no further manipulation. The system is supplied with the semantic mapping of the gray levels of the diagram to the vegetation types present. The input diagram is then parsed into three diagrams, each comprised of a single gray level. Each of these diagrams represents, then, the location of a particular vegetation type within the United States. Figure 3, for example, shows the diagram resulting from this parsing that represents the locations of grassland in the United States.

A priori diagrammatic knowledge required to respond to this query is comprised of a set of diagrams that represent the locations of each state within the United States. Figure 4 is an example of such a diagram which shows the location of the state of Nevada within the United States by marking its area on the map in black. There are fifty such state diagrams.

The response to the query "Which states have grassland?" is generated by comparing each of these state diagrams with the diagram representing grassland. When a state diagram intersects the grassland diagram (both diagrams without the United States outline), the semantics of the domain dictate that that state contains grassland. All such states are then accumulated on a single diagram (with the United States outline) and presented to the user as the response to the query.

In this manner, diagrammatic responses can be generated for a wide variety of queries concerning vegetation in the United States including "Which states do not have forest?", "How many states have desert?" (Simply return a count of the state diagrams that intersect the desert diagram.), "Does

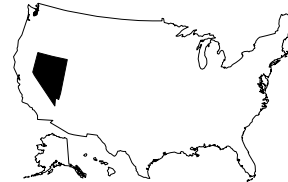


Figure 4: Location of Nevada in the United States

Rhode Island have desert?" (Simply return true if the state diagram for Rhode Island intersects the desert diagram.), "Which vegetation type covers the most states?", "Do any states have both grassland and desert?", "Which states have either desert or forest?", "Do more states have grassland than desert?", "Which states have forest but not grassland?", etc.

An overview of the formalism we use to generate responses to these queries, the theory of inter-diagrammatic reasoning, follows.

Inter-Diagrammatic Reasoning

The theory of inter-diagrammatic reasoning (Anderson and McCartney 1995, 1996; Anderson and Armen 1998) defines diagrams as tessellations. Tesserae take their values from an I, J, K valued subtractive CMY color scale. Intuitively, these CMY (Cyan, Magenta, Yellow) color scale values (denoted $V_{i,j,k}$) correspond to a discrete set of transparent color filters where i is the cyan contribution to a filter's color, j is the magenta contribution, and k is the yellow contribution. When overlaid, these filters combine to create other color filters from a minimum of WHITE ($V_{0,0,0}$) to a maximum of BLACK ($V_{I-1,J-1,K-1}$). In the current work, i, j , and k are always equal, providing grayscale values from WHITE to BLACK only. The following unary operators, binary operators, and functions provide a set of basic tools to facilitate the process of inter-diagrammatic reasoning.

Binary operators each take two diagrams, d_1 and d_2 , of equal dimension and tessellation and each return a new diagram where each tessera has a value v that is some function of the values of the two corresponding tesserae, $V_{i1,j1,k1}$ and $V_{i2,j2,k2}$, in the operands.

- **OR**, denoted $d_1 \vee d_2$, returns the *maximum* of each pair of tesserae where the maximum of two corresponding tesserae is defined as $V_{\max(i1,i2), \max(j1,j2), \max(k1,k2)}$.
- **AND**, denoted $d_1 \wedge d_2$, returns the *minimum* of each pair of tesserae where the minimum of two corresponding tesserae is defined as $V_{\min(i1,i2), \min(j1,j2), \min(k1,k2)}$.
- **OVERLAY**, denoted $d_1 + d_2$, returns the *sum* of each pair of tesserae where the sum of values of corresponding tesserae is defined as $V_{\min(i1+i2, I-1), \min(j1+j2, J-1), \min(k1+k2, K-1)}$.
- **PEEL**, denoted $d_1 - d_2$, returns the *difference* of each pair of tesserae the difference of values of corresponding tesserae is defined as $V_{\max(i1-i2, 0), \max(j1-j2, 0), \max(k1-k2, 0)}$.
- **NONNULL (NULL)**, denoted $\text{NONNULL}(d)$, ($\text{NULL}(d)$) is a one place Boolean function taking a single diagram that returns TRUE if d contains any non-WHITE (all WHITE) tesserae else it returns FALSE.
- **ACCUMULATE**, denoted $a(d, ds, o)$, is a three place function taking an initial diagram, d , a set of diagrams of

equal dimension and tessellation, ds , and the name of a binary diagrammatic operator, o , that returns a new diagram which is the accumulation of the results of successively applying o to d and each diagram in ds .

- **MAP**, denoted $\mu(f, ds)$, is a two place function taking a function f and a set of diagrams of equal dimension and tessellation, ds , that returns a new set of diagrams comprised of all diagrams resulting from application of f to each diagram in ds .
- **FILTER**, denoted $\phi(f, ds)$, is a two place function taking a Boolean function, f and a set of diagrams of equal dimension and tessellation, ds , that returns a new set of diagrams comprised of all diagrams in ds for which f returns TRUE.

Given these inter-diagrammatic operations, the vegetation and state maps as previously described, and a null diagram (denoted \emptyset) standing for a diagram with all tesserae WHITE-valued, the following more formally specifies the generation of a diagrammatic response to the query "Which states have grassland?":

$a(\emptyset, \phi(\lambda(x) \text{NONNULL}(\text{Grassland} \wedge x), \text{State}), +)$.

This 1) defines a lambda function that ANDs its parameter with the grassland diagram and returns true if the result is not null, 2) filters out diagrams from the set of state diagrams for which this lambda function does not return true (these are the state diagrams that do not intersect the grassland diagram), and 3) overlays the remaining state diagrams onto the null diagram giving the desired result. Figure 5 details this example.

Responses to all of the queries suggested previously can be generated via IDR operators. As in the example, those queries requiring a diagrammatic response produce an appropriate set of diagrams which are OVERLAYed together. Those queries requiring a numeric response produce an appropriate set of diagrams and return the cardinality of it. For instance, the number of states that have grassland can be returned by taking the cardinality of the set returned by the filtering operation instead of accumulating that set upon the null diagram as is done in the example. Those queries requiring a Boolean response return the value of the NONNULL function applied to an appropriately derived diagram. For instance, a response to the query "Are there any states that have grassland?" will derive a diagram as in the example and return the result of applying the NONNULL function to it. Responses to queries seeking negative information can be derived by using the NULL function to produce an appropriate set of diagrams. For instance, a response to the query "Which states do not have grassland?" can be generated by simply replacing the NONNULL function with the NULL function. Queries seeking responses to conjunctions or disjunctions need to use set intersection and set union (respectively) to produce the appropriate sets of diagrams. Responses to relational ($<, >, <=, >=, <>, =$) queries need to compare the

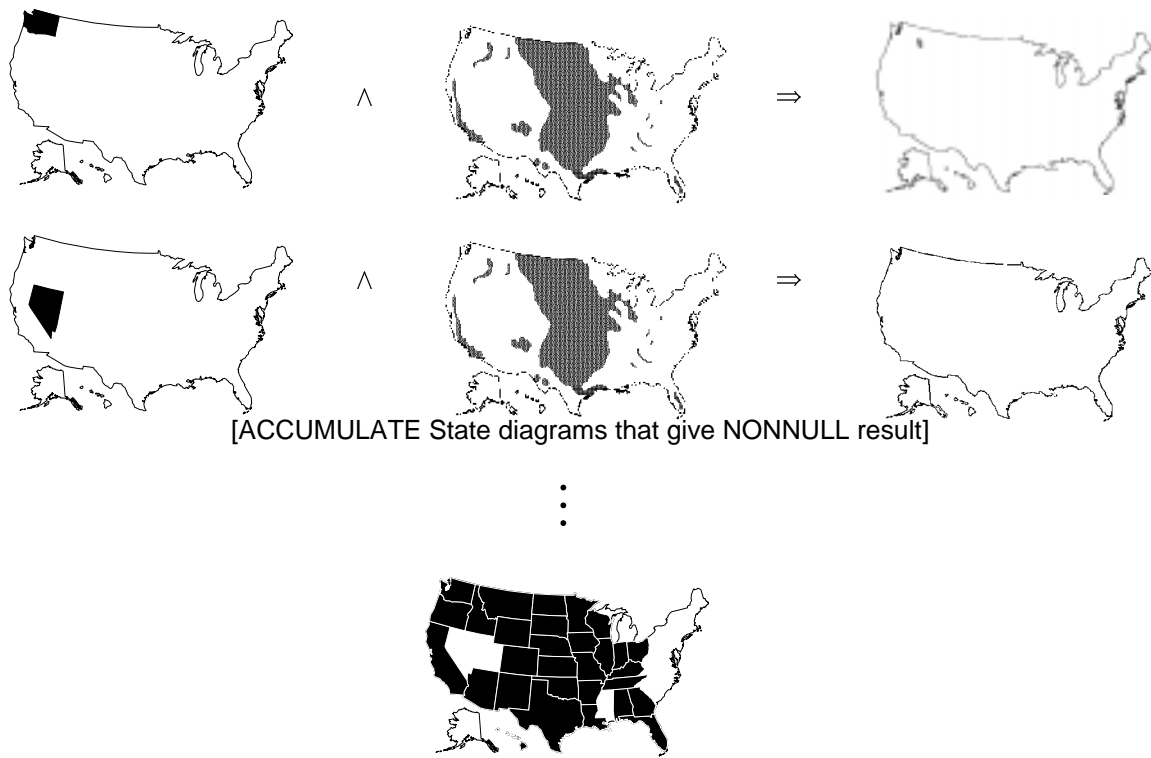


Figure 5: Generation of response to query "Which states have grassland?" via inter-diagrammatic reasoning operators

cardinality of each set of diagrams produced for each subquery involved.

Although IDR operators can produce responses to this wide variety of queries in this domain, it is by no means intuitive *how* they should be used to do so. In the following, we introduce a higher level query language that permits a user to query diagrams more intuitively, specifying *what* they wish to know more than *how* it should be generated.

Diagrammatic SQL

Diagrammatic SQL (DSQL) is an extension of Structured Query Language (SQL) (Date 1989) that supports querying of diagrammatic information. Just as SQL permits users to query information in a relational database, DSQL permits a user to query information in diagrams.

We have chosen to extend SQL for use as our query language for a number of reasons. As we will show, SQL has a remarkable fit to the uses we wish to make of it. It is a reasonably intuitive language that allows specification of what data you want without having to specify exactly how to get it. It is a well-developed prepackaged technology whose use allows us to focus on more pressing system issues. SQL's large installed base of users provides a ready and able audience for a fully developed version of the system. The availability of immediate and imbedded modes provide means to use the system for both direct human consumption and further machine processing. The

availability of natural language interfaces to SQL will allow the system to provide an even more intuitive interface for its users.

Besides providing a basis for a diagrammatic query language, a relational database that stores image data can be used by the system as a storage management component. Further, as relational databases already manage other types of data, use of one as a storage management component with a diagrammatic extension to its SQL gives the system a means to query both diagrammatic and non-diagrammatic data simultaneously. This provides a linkage between heterogeneous data allowing whole new classes of queries, for example, "What is the total population of states having desert?", "Which of the states having forest has the highest per capita income?", "What vegetation is contained by the state with the highest annual rainfall?", etc.

We have developed a grammar for a subset of DSQL that allows it to handle queries of the types previously discussed. Where SQL queries return relations, DSQL queries return sets of diagrams. These diagram sets can have their members **OVERLAY**ed upon a null diagram for diagrammatic results or counted to return numeric results. Further, these sets can be tested for emptiness to return Boolean results or used as operands in set operations such as union, intersection, and difference. Examples of DSQL syntax and semantics follow.

DSQL Example. Figure 6 shows an example data definition and sample queries in DSQL. Figure 6a uses

a. CREATE SCHEMA; CREATE TABLE US (State, Vegetation); INSERT INTO US (State) VALUES (Alabama, Arizona, ...); INSERT INTO US (Vegetation) VALUES (Grassland, Desert, Forest);	e. EXISTS ((SELECT State FROM US WHERE Vegetation = Grassland AND Vegetation IN State) INTERSECT (SELECT State FROM US WHERE Vegetation = Desert AND Vegetation IN State));	FROM US WHERE Vegetation = Forest AND Vegetation IN State) EXCEPT (SELECT State FROM US WHERE Vegetation = Grassland AND Vegetation IN State);
b. SELECT State FROM US WHERE Vegetation = Grassland AND Vegetation IN State;		
c. SELECT State FROM US WHERE Vegetation = Forest AND Vegetation NOT IN State;	f. (SELECT State FROM US WHERE Vegetation = Desert AND Vegetation IN State) UNION (SELECT State FROM US WHERE Vegetation = Forest AND Vegetation IN State));	
d. SELECT COUNT(State) FROM US WHERE Vegetation = Desert AND Vegetation IN State;	g. (SELECT State	

Figure 6: DSQL data definition and example queries

DSQL to define a schema for the diagrammatic information required by the examples presented previously. It creates a table named **US** that contains two diagram sets named **State** and **Vegetation**. In the current Lisp implementation, as no connection has yet been established to a relational database management system, a table is simply a list of related diagram sets. Each diagram set has inserted into it a number of diagrams appropriate to the set. In the current Lisp implementation, these are actually symbols that evaluate to diagrams.

Figure 6b is a DSQL query that represents the example query "Which states have grassland?". It has the same **SELECT FROM WHERE** clauses that SQL queries have and these share similar semantics with their SQL counterparts. Most often, the SQL **SELECT** clause specifies what attribute(s) will have values in the resulting relation. In DSQL, the **SELECT** clause specifies what diagram set(s) will have values returned from the query. The SQL **FROM** clause specifies which table(s) are involved in the query. In DSQL, the **FROM** clause specifies which list(s) of diagram sets are involved in the query. The SQL **WHERE** clause specifies which condition(s) have to be satisfied by values returned by the query. This is the same use that a **WHERE** clause is put to in DSQL.

The DSQL query in Figure 6b states that the set of diagrams from the diagram set **State** of the diagram set list **US** that conform to the constraints specified will be returned. The **WHERE** clause specifies 1) that the **Vegetation** diagram set of the diagram set list **US** is restricted to the **Grassland** diagram only and 2) that the

diagram in the **Vegetation** diagram set must intersect given **State** diagrams. In one context, the SQL **IN** Boolean operator returns true if and only if the value on the left hand side is a value in the attribute on the right hand side. In DSQL, **IN** is a Boolean operator that returns true if and only if the diagrams involved intersect. When sets of diagrams are involved, as in this and following examples, the semantics of a DSQL query dictate that this intersection be tested for each member of each set. In this case, the **Grassland** diagram will be tested for intersection with each member of the **State** diagram set, in turn, allowing the query to return only those states that contain grassland. As previously detailed, the response to this query is achieved by IDR operators as:

$$a(\emptyset, \phi(\lambda(x) \text{ NONNULL}(\text{Grassland} \wedge x), \text{State}), +).$$

Figure 6c is a DSQL query that seeks a response to the question "Which states do not have forest?". The semantics of this query is much like the previous example. In this example, though, the **Vegetation** diagram set is restricted to the **Forest** diagram and this diagram must *not* intersect with a state diagram for it to be included as part of the result. The response to this query is achieved by IDR operators as:

$$a(\emptyset, \phi(\lambda(x) \text{ NULL}(\text{Forest} \wedge x), \text{State}), +).$$

Figure 6d is a DSQL query that seeks a response to the question "How many states have desert?". This change in mode from a diagrammatic response to a numeric response is signaled by the application of the **COUNT** function to the diagram set in the **SELECT** clause. It is realized by the following IDR formulation where *cardinality* is a function returning the number of members in a set:

$cardinality(\phi(\lambda(x) \text{ NONNULL}(\text{Desert} \wedge x), \text{State})).$

Figure 6e is a DSQL query that asks "Are there any states that have both grassland and desert?". The fact that a Boolean response is required is signaled by the use of the EXISTS function. In SQL, the EXISTS function tests for an empty (single attributed) relation resulting from a subquery. In DSQL, it is used to test for an empty set of diagrams resulting from any query. To produce the set to be tested using IDR operations, the set of state diagrams that have grassland is intersected (\cap) with the set of state diagrams that have desert. If this resulting set is *not empty* ($\neg \text{empty}$), return true else return false. Following is the IDR realization of this query:

$$\neg \text{empty}(\phi(\lambda(x) \text{ NONNULL}(\text{Grassland} \wedge x), \text{State}) \\ \cap \\ \phi(\lambda(x) \text{ NONNULL}(\text{Desert} \wedge x), \text{State})).$$

Figure 6f is a DSQL query that seeks a diagrammatic response to the question "Which states have either desert or forest. This response is generated by taking the union (\cup) of the set of states that have desert and the set of states that have forest and, then, OVERLAYing them onto the null diagram. Expressed as IDR operations:

$$a(\emptyset, \phi(\lambda(x) \text{ NONNULL}(\text{Desert} \wedge x), \text{State}) \\ \cup \\ \phi(\lambda(x) \text{ NONNULL}(\text{Forest} \wedge x), \text{State}), +).$$

In a similar vein, Figure 6g is a DSQL query that seeks a diagrammatic response to the question "Which states have forest but not grassland?". This response is generated by taking the difference ($-$) of the set of states that have forest and the set of states that have grassland and, then, OVERLAYing them onto the null diagram. Expressed as IDR operations:

$$a(\emptyset, \phi(\lambda(x) \text{ NONNULL}(\text{Forest} \wedge x), \text{State}) \\ - \\ \phi(\lambda(x) \text{ NONNULL}(\text{Grassland} \wedge x), \text{State}), +).$$

Evaluation of Model and System

To reiterate, our current goal is to develop an instantiation of a diagrammatic information system knowledgeable about a particular diagram type (cartograms) in a particular domain (United States) that produces quality responses to the full range of queries that would be expected of a reasonably capable human being given the same knowledge.

A subset of a DSQL grammar required to handle the range of queries exemplified in this work has been developed, a rudimentary compiler that translates this range of DSQL queries into their IDR formulations has been constructed, and the IDR operations that produce the desired output have been realized in Common Lisp. The current instantiation of the diagrammatic information system responds to an interesting range of queries posed against cartograms of the United States. This range of queries can be characterized as those whose responses are

generated by various combinations of input diagrams and *a priori* diagrams. It is arguable that, within this range, the quality of responses equals or exceeds human capabilities with the same diagrammatic information. These are indicators of a promising line of research.

That said, there is much work yet to accomplish to realize fully the goal of even this relatively simple level. For example, not all queries that a human would be expected to answer in the example domain can be handled currently by the system. These include, for instance, queries that seek information about area or neighborhood relations. Further, only cartograms of the same size and orientation as the *a priori* diagrammatic knowledge can be handled by the system. Clearly, humans are capable of handling such variations. Noise and uncertainty concerns present in real world data have also been avoided. For instance, textual annotations on a cartogram, although helpful to a human, are noise to the current system. Ready recognition of these limitations is the product of a clearly delineated task and well-defined goal. These limitations, then, provide the focus for future work.

Queries that seek information about area provide an opportunity to integrate IDR with another theory of diagrammatic reasoning. Furnas' *BITPICT theory* (Furnas 1992) postulates a logic that deals with diagrams via BITPICT rule mappings that can be used to transform one diagram into another and, therefore, allow reasoning from diagrams to diagrams. A BITPICT rule is meant to convey that all instances in a diagram of the bit pattern on the left hand side of a rule are replaced with the bit pattern on the right hand side of that rule. As interesting as this theory is, it can be subsumed by IDR by using appropriate sets of diagrams representing the universal instantiation of BITPICT rules. With this theory irregular shapes can be normalized, allowing comparison of their relative areas. Queries such as "Does California have more desert than grassland?" can then be handled. Further, domains that seem less amenable to IDR techniques (for instance, line graphs) can be made more manageable by use of this theory (for instance, by shading the area under a curve).

Queries that seek information about neighborhoods provide an opportunity to integrate IDR with an image processing theory. *Mathematical morphology* (Serra 1982; Serra 1988) is a image processing technique based on shape that is used to simplify, enhance, extract or describe image data. Sets of pixels describe an *image object*. Information about the geometric structure of image objects is extracted via use of other objects called *structural elements*. Information pertaining to the size, spatial distribution, shape, connectivity, convexity, smoothness, and orientation can be obtained by transforming the original image object by various structural elements. As primitive mathematical morphological operators can be modeled by IDR operators, IDR subsumes this theory as well. One such primitive operator, *dilation*, can be

intuitively viewed as an operation that adds layers to the border of a two-dimensional object. Adding a sufficiently wide layer to a state diagram, for instance, allows this modified diagram to be tested for intersection with the other state diagrams. This can produce the set of state diagrams that neighbor the original state. In combination with area querying capabilities introduced previously, new classes of interesting queries can be handled. For example, "Which of the states surrounding Nevada have the greatest amount of forest?" is one such query that could then be handled.

Problems with real world data can be approached using geometrical transformations. Orientation and size of cartograms can be normalized by combinations of rotation and scaling operations. The search for the required combination and parameters of operations could be guided by the user. These operations, themselves, are implementable within the theory of IDR.

In addition to the above extensions, we are developing a full DSQL grammar, a complete interpreter and compiler to translate DSQL to IDR, and support for both immediate and imbedded modes of operation. We also are planning a relational database implementation with an exploration of the heterogeneous data inference that such an implementation will allow. We are also interested in investigating the extension to DSQL of natural language interfaces for SQL. Finally, when we have satisfactorily accomplished the stated goals for this level, we will then lift constraints and focus on the set of problems introduced by exploring new diagrams types in new domains.

Acknowledgments

We thank Dr. Chris Armen for his insightful comments and encouragement throughout the duration of this project. This research is supported by a grant from the National Science Foundation.

References

- Allen, J. 1998. AI Growing Up: The Changes and Opportunities. *AI Magazine* 19(4):13-23.
- Anderson, M. and Armen, C. 1998. Diagrammatic Reasoning and Color. In Proceedings of the AAAI Fall Symposium on Formalization of Reasoning with Visual and Diagrammatic Representations, Orlando, Florida. October.
- Anderson, M. and McCartney, R. 1995. Inter-diagrammatic Reasoning. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada. August.
- Anderson, M. and McCartney, R. 1996. Diagrammatic Reasoning and Cases. In Proceedings of the 13th National Conference on Artificial Intelligence, Portland, Oregon. August.

Date, C. 1989. *A Guide to the SQL Standard, Second Edition*. Addison-Wesley.

Feigenbaum, E. A. and Feldman, J., eds. 1963. *Computers and Thought*, McGraw-Hill.

Furnas, G. 1992. Reasoning with Diagrams Only. In (Narayanan 1992).

Gelernter, H. 1959. Realization of a Geometry Theorem Proving Machine. In Proceedings of an International Conference on Information Processing, 273-282. UNESCO House. (also in [Feigenbaum & Feldman 1963]).

Larkin, J. and Simon, H. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11, 65-99.

Narayanan, N. editor 1992. Working Notes of AAAI Spring Symposium on Reasoning with Diagrammatic Representations.

Serra, J. 1982. *Image analysis and Mathematical Morphology*, Vol. 1. Academic Press.

Serra, J. 1988. *Image analysis and Mathematical Morphology*, Vol. 2. Academic Press.