THE STRUCTURE AND FUNCTION OF DIAGRAMS IN ENVIRONMENTAL DESIGN: a Computational Inquiry

by

STEPHEN McTEE ERVIN

M.L.A. University of Massachusetts, 1981

Submitted to the Department of Urban Studies and Planning
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

September 1989

Signature of Author _____

Department of Urban Studies and Planning
June 20, 1989

Certified by _____

Aaron Fleisher
Thesis Supervisor
Professor, Department of Urban Studies and Planning

Accepted by _____

Langley Keyes
Chairman, PhD Committee
Department of Urban Studies and Planning

Abstract

Diagrams, especially but not only those encountered in urban design and planning, can be understood as having a relatively simple structure: a set of graphical elements and relations that refer to a corresponding set of elements and relations in another realm, typically a design proposal or an existing place, structure or system -- and a relatively complex function: to stimulate and enable the eye/brain visual cognitive system to make inferences that apply to the diagrammed realm, on the basis of the graphics and the system of reference. Capturing this structure and function in a computer program provides a valuable experimental environment for computer-aided design and design research.

I explore the proposition that diagrams can be distinguished from other kinds of drawings such as maps, plans, and sketches, on the basis of their origin and use, and that they can be produced in the course of design development by a process of transforming verbal propositional knowledge into graphics. I analyze several examples of diagrams observed in the literature of urban/environmental design, and show how they might be understood in these terms. On the basis of these examples I propose a limited lexicon of elements and relations that serve to form verbal propositions about spatial structure based on urban design criteria.

An information-processing model of designing locates diagrams in between symbolic propositions and pictorial drawings (plans and sketches), and distinguishes between more propositional 'A-diagrams' and more pictorial 'B-diagrams' on the basis of their content, and their origin. The model builds on the 'constraint model' that treats designing as the declaration of relations between elements, and proposes specific computational data structures for representing a design state as a set of elements and relations, and algorithmic procedures including constraint propagation and conflict resolution, for exploration of the consequences of variations in the design state. I argue that a computer program able to join in the process of designing with constraints would benefit from the ability to produce (and to read) diagrams, as an interface between designer and data structure.

I propose an algorithm for diagram construction with five essential steps: *filtering* a design state to select or transform a subset of its elements and relations that are to be diagrammed; *mapping* elements and relations from the filtered design state into graphic symbols and relations in a diagram, using or creating a legend, or key; *assigning defaults* (reasonable values) to diagrammatic variables in cases where the design state or the mapping is insufficient; *laying out* graphic symbols in a visual frame, to produce the graphic artifact; and *conflict resolution* , that may be required and may require human guidance, in cases where the other four operations are mutually incompatible.

A prototype program implemented in the framework of the constraint model (the "Constraint-Based Diagrammer", CBD) is used to test the hypothesis that a diagram can be specified as a set of relations between elements using the limited lexicon, and constructed by a process of default assignment, constraint propagation and conflict resolution. I describe two experiments, in which diagrams taken from the literature are partially reproduced by the CBD from a list of propositions. The results demonstrate that the CBD can produce simple diagrams in some circumstances, but that the propositional input and constraint propagation process together are insufficient to explain all the features of the observed diagrams. Exact replication fails in both cases because some necessary information in the original diagrams is 'transparent' to the reader, being embedded in global qualities of the graphics like symmetry and proportion, and so missed upon first reading. Replication also fails because of the difficulty of capturing all the salient, visible information in a limited vocabulary.

Introspection on this experience with the CBD suggests that the inferential content of graphic features of diagrams derives from both learned domain-specific reasoning and inherent capabilities and characteristics of human visual cognition. In the context of designing, these phenomena of visual cognition can be located with respect to three different kinds of inferences that proceed from diagrams -- "map-like deductions" that utilize literal translation from 2-d space to 2-d space (and so are best supported by 'B-diagrams'), "map-inspired analogies" that translate from 2-d spatial properties to analogous non-spatial properties, and "other visual analogies" that depend on associations with non-spatial visual properties. Each kind of inference is useful in designing, as it may lead to verification or modification of the design state being diagrammed. Since the inference or kind of inference desired is usually not known at the outset of diagramming, there is no "correct" or "best" diagram. This suggests that the value of a computer diagrammer rests not on its ability to replicate particular diagrams, but rather on its ability to produce a variety of diagrams, in the hopes of stimulating a variety of inferences, and the ability to control these variations might be a powerful ability in design exploration.

Some of the parameters of variation in diagrams, and their inferential import, are revealed by this research, but remain to be expanded upon by empirical studies of designers diagramming, psychological research into the effects of graphic variations, and computational research into procedures for implementing and controlling such variations. The first four steps of diagram construction -- filtering, mapping, assignment and layout -- provide a framework in which to locate the mechanisms of variation, and the CBD provides a vehicle with which to explore their effects. The enumeration and organization of a set of parameters that control diagrammatic variations and diagrammatic equivalence, along with an explanation of the inferential potency of each of the parameters alone and in combination, constitute a necessary first step toward a general theory of diagramming.

Results of exploration with the CBD indicate some of the requirements for other diagram-managing modules in a computer-aided diagramming environment, including a diagram-combiner, diagram-fitter, diagram-reader, and drawing-abstracter.

Thesis Supervisor: Aaron Fleisher

Professor, Urban Studies and Planning

Acknowledgements

I never met Kevin Lynch, but this work is dedicated to him, for the marginalia in his books are in large part responsible for my fascination with diagrams. I hope he would approve of my intent. Others more directly influenced the formation, formulation and completion of this thesis, and to them my debt is more immediate. My committee members each provided their own special brand of insight and skepticism, for which I'm grateful. Aaron Fleisher, who provided tea and fig brooklines early on, and a ready argument, chalkboard, and red pen throughout a rather long ordeal; Gary Hack, who got me into this in the first place, and generously supported my first few years of floundering; and Don Schön, who reflected in many useful ways on my early and more recent flounderings -- all provided the patience, impatience, encouragement and direction that thesis advisors are supposed to. I hope I may do as well some day.

My friends and colleagues in and around the CRL, too numerous to enumerate -- but including Joe, Leova, Rob, Kelly, Bizhan, Simon, and Joy -- made for a convivial environment and more than one great sailing trip along the way. James and Tom shared my enthusiasm for diagrams as windows into the design mind, and helped make N51 a special place for a while. All the members of the Design Research Seminar formed a receptive captive audience on more than one occasion, especially Bill Porter, Larry Bucciarelli, Avraham Wachman and Gabriela Goldschmidt. MIT's Department of Architecture and Project Athena, the Design Methods directorate of NSF, Apple Computer Co. and the good folks at Coral Software all provided material support that made this effort possible.

Victor and Laurie made sushi and skiing and macintoshing all the more delightful, and provided plenty of opportunities to procrastinate, as well as incentive to finish.

Puff gets special mention, and special gratitude, for blazing a trail; for introducing me to concepts and cultures directly responsible for, and highly visible in, the present work; for sharing keyboard, mouse, and other things; for RoadLab, CM-0 and CM2; and for carrying on.

The most special mention must be reserved for my family -- to Pat and Frank who prepared me to integrate art and science in my life and who have supported me every step of the way; to grandmother Ailee, one of the original inquiring minds; and Valerie, who gave me the best weekend of my life, and plenty of second-bests too; more love, support and companionship in more ways than can be recounted; respite from the grind and motivation to work, not always in equal amounts but usually at the right times; and a home that made writing tolerable, and not-writing rejuvenating. Thanks, all.

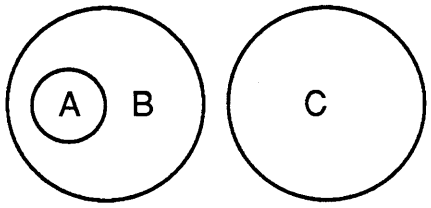
THE STRUCTURE AND FUNCTION OF DIAGRAMS
IN ENVIRONMENTAL DESIGN

TABLE OF CONTENTS

List of Captioned Figures

On the power of diagrams:

"If any sluggish imagination did not at once realise that from 'All A is some B', 'No B is any C', we could infer that 'No A is any C', he has only to trace the circles, and he sees it as clearly as any one sees the results of a physical experiment. And most imaginations, if truth be told, are sluggish enough to avail themselves now and then of such a help with advantage."



-- J. Venn, 1894 *Symbolic Logic,* (2nd ed.)

On the necessity of diagrams:

"On ne trouvera point de Figures dans cet ouvrage. Les méthodes que j'y expose ne demandent ni constructions, ni raisonnements géometriques ou mécaniques, mais seulement des opérations algébriques, assujetties a une marche reguliere et uniforme. ..."
"One will find no diagrams in this work. The methods that I present require neither constructions, nor geometric nor mechanical reasoning, but only algebraic operations applied in a regular and uniform fashion. " (approximate translation)

-- J. L. LaGrange, 1779*Mécanique Analytique (2ème Édition.)*

To all who, like me, are endowed with sluggish imaginations that enjoy reasoning more like Mister Venn's than Monsieur LaGrange's, this thesis is offered.

SME June 1989

Introduction

This is a story about diagrams. The setting for the story is the domain of urban design -- the moral of the story may well apply in other domains. The purpose of the story is two-fold. The principal purpose is to propose a way of understanding the structure of some diagrams, and their function in the process of urban designing; a secondary purpose is to describe an experiment in making a computer program that embodies some of that understanding, and to outline the functionality required of a system for computer-aided diagramming.

The impetus for the story is the simple observation that diagrams -- whatever they are, for now -- are ubiquitous and apparently powerful aids to human reasoning. In particular, in enterprises where the production of drawings is an essential part (such as architecture, landscape architecture, urban design and some engineering disciplines), diagrams seem to play an important role in "getting ideas on paper". Diagrams often are employed in preliminary, or conceptual, stages of designing; they are equally often employed as an adjunct to explanation or argument, where the mode is otherwise primarily verbal. Why? What is the special character of diagrams that makes them useful in these ways? What knowledge is embedded in diagrams, how is it embedded, and how is it used by designers?

The second reason for this story is that computers are used in design disciplines in the production of drawings, and in the communication of ideas in general. Most such uses only attempt to reproduce some desired end-product -- working drawings and specifications, say, or machine tool instructions -- which exploit computational speed, accuracy, and memory. As such, they assist in the production, but not the designing, of drawings; as such they are useful, but lacking. What would a computer program have to know, and how should it know it, in order to enter more actively into a process of designing and drawing?

I suggest that one answer to this last question will come from having some answers to the previous questions about diagrams. That is, a computer program that could engage in the production of diagrams could be involved in a process of designing -- or would at least be one step closer. Clearly, though, a full answer also requires some answers to prior questions like "What is designing?", "What constitutes design knowledge?" and "What role do drawings in general play in designing and design reasoning?" Each of these questions is too big to answer thoroughly or definitively, but I shall have to address each, at least tentatively, along the way.

These concerns will set the stage, and provide the framework, for the experiment described herein, which asks the question: "Can a computer program produce diagrams like those produced by urban designers, given information in a form similar to that which urban designers use?" The tentative reply is "Yes, but...", and the further detailed questions are then "What knowledge, in what form, must be given to the program?", "What principal problems and questions are exposed by its development?", and "What value would such a program have?"

My thesis is that such a program must have access to a knowledge base representing the propositional content in the form of elements and relations; translation processes that convert from propositions to graphic artifacts; and rules governing graphic form, that respond to both domain-specific constraints and conventions, and inherent human visual cognitive abilities and biases. The problems surrounding the development of propositions are principally related to the availability of an expressive and flexible grammar and vocabulary in which to express design intentions, in a way that does not compromise the designer's ability to express multiple and possibly contradictory meanings and to discover new ones. The problems to be overcome in capturing knowledge of visual cognition and gestalt perception in procedural form are many, but particularly include those of sorting out relatively universal perceptions from highly idiosyncratic ones, finding rules and procedures governing combinations and interactions of graphic properties, including context-relativity, and identifying parameters or axes over which the graphic form of diagrams can be

varied, and how the inferential significance of such variations might be measured. I suggest that these problems are ones which any theory of diagramming must address. Finally, I argue that the value of a diagramming program would lie in its ability to provide an interface between the powerful machinery of human visual cognition on the one hand, and the rigor of constraint based computing on the other; first as an experimental tool with which to pursue the research involved in exploring the problems just referred to, and eventually as a useful design tool, amplifying designers' abilities to express, explore and evaluate alternatives.

Research Objectives and Motivation

This research first aims to provide a theoretical base for describing the role of diagrams in designing. The essential pieces of the argument are two: first, a model of designing that locates the function of diagrams, and a knowledge representation scheme that captures their structure, that together constitute a theory about the distinct role(s) of diagrams; second, a computer program that embodies the knowledge representation scheme, and illustrates some of the operations central to the model, will serve to demonstrate the applicability of the theory to designing.

The quest for a theory is motivated both by a 'pure' desire for a framework in which to describe designing and design processes in the large -- knowledge for knowledge's sake -- and also by an 'applied' desire for guidance in the development of powerful and appropriate computer-aided design tools. Such tools to date remain principally concerned with production processes that follow from designing, and do not address the cognitive issues or the problems off representation associated with developing ideas up to the production stage.

Diagram 0.1 locates the enterprise of computer-sided diagramming with respect to three other common kinds of computational approaches to designing and design problems: so-called 'expert systems' combine domain knowledge and reasoning, but without graphics; most 'computer-graphics' applications make no use of reasoning, providing only relatively 'dumb', domain-specific drawing

tools; and the broad enterprises of computer vision and pattern recognition are generally indifferent to particular domain requirements and substantive knowledge.

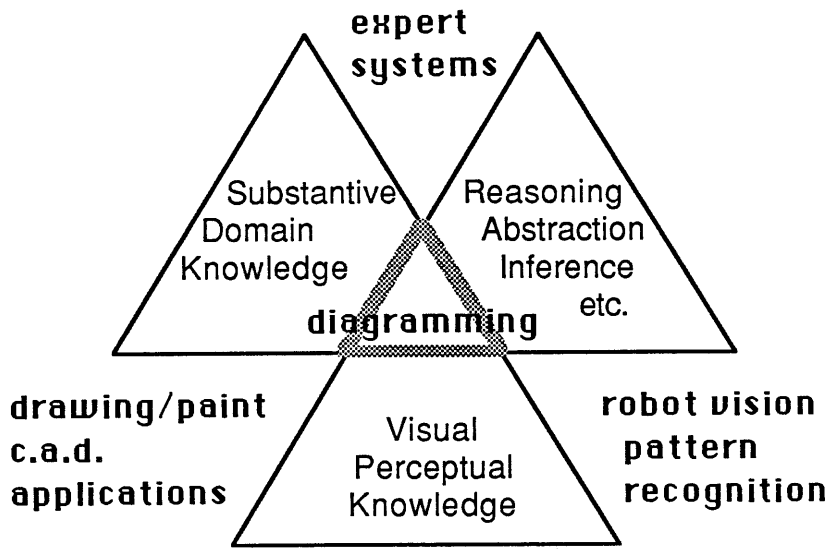


Figure 0.1 Computer-Aided Diagramming in Context

The diagram shows the activity of diagramming at the intersection of three realms: substantive domain knowledge, or expertise; reasoning mechanisms in general, including abstraction, induction, monotonic and non-monotonic inference, etc; and perceptual abilities and visual cognition. A computer-aided diagramming system must be prepared to amplify or exploit each of these abilities.

Assumptions

The central beliefs underlying this research are three: 1) that diagrams are a unique kind of graphics, distinct from maps, plans or sketches, that play a special and important role in environmental designing; 2) that the structure and function of diagrams must be understood with respect to their inferential potential, both as stimulus to and medium for reasoning; and 3) that a computer program can be built that contains a knowledge representation scheme that captures the structure of diagrams and exhibits behavior that plausibly reflects or supports some of the function of diagrams in designing.

The first two of these assumptions arise from a model of designing that locates diagrams as a bridge between concepts and graphics, that functions as analog representations of reality or design intentions, and proposes a purpose and a mechanism for the transitions between concepts, diagrams and other drawings. The model will be explained with reference to three particular questions: In the first place, how are diagrams different from drawings of other kinds (maps, plans and sketches, e.g.)? ; second, what significance does this distinction have, and how is it related to reasoning, in the processes of designing?; and third, how might these ideas be captured in a computer program?

The Domain: Urban/Environmental Design and Graphical Knowledge Representation

Environmental design provides a rich source of study for research in design theory and methods. The work of architects, landscape architects, urban designers -- designing places for people to live, work and play -- involves synthesis of a wide variety of concepts, considerations and constraints into a plan for physical and organizational development. The initial concepts may include such high-level and nebulous concerns as "community and privacy" or "social justice"; the end results may call for the placement of so many cubic yards of concrete in such-and-such places. How designers negotiate and navigate this path, from abstract, conceptual beginnings to concrete and mundane ends, is a general concern of this thesis. A particular reason to explore this question has to do with with the development of computer tools for designers; at present, computer-aided design tools are most useful at the (concrete) end -- this research aims to push them towards the (conceptual) beginning.

Much design (urban/environmental/architectural especially) is performed graphically. Designers produce graphics as end-products to direct the processes of construction and organization; as intermediate products to communicate with themselves and others; and often experimentally, for relaxation and inspiration. That graphics are powerful for communication is well-known, though

the source of the power is not well understood. The high bandwidth and acuity of the human visual system are contributory factors, but in addition there appear to be special cognitive processes at work, and some forms of reasoning unique to "visual thinking" [Arnheim].

In designing, graphics (drawings) often play the role of intermediary, representing and allowing us to reason about the thing(s) being designed. How are aspects of things being designed represented graphically, that allows us to reason about them? Often it is by simile, or literal transformation, as in the case of realistic renderings and scale drawings, which provide a simple surrogate to our visual field. Sometimes it is by convention, as in electrical schematics, where standard symbols substitute for real-world elements and connections. Sometimes, as in the case of the kinds of diagrams explored herein, the process of reasoning is a kind of analogy in which features of the drawing (diagram), come to stand for elements and concepts in a way that is neither completely literal nor conventional, though it may be partly either or both.

Since these kinds of diagrams are typically spatial representations of ideas about space, there is an element of literal association at work. And since designers work together, and have over time, there are some conventional elements at work also. But mostly what distinguishes these kinds of diagrams from pictorial drawings or schematics is the role played by reasoning (that is, knowledge-based inference) in their generation and interpretation. A computer program that can engage in designing with diagrams must have some knowledge base(s), and some rules for operating on that knowledge.

Related Work

This research is a fusion of three braided streams of inquiry; one into formal descriptions of spatial and environmental qualities; another into descriptive and computational models of reasoning, visual cognition in general, and designing in particular; and the last into interactive computer graphics and knowledge-based computer-aided design systems. Specific attention to diagrams in

any of these areas has been relatively rare, although environmental designers have seemingly always used diagrams, and recently cognitive psychologists have turned their attention to diagrams as knowledge representation devices [Larkin&Simon, Lindsay].

The first stream is represented by a small literature from the analytical side of environmental design, notably [Lynch] and [Lynch & Appleyard] who provided specific descriptive terminology for environments and environmental characteristics, combining words, graphics and design ideas. Both the five elements identified in *The Image of the City* and the graphic techniques developed in *The View from the Road* lend themselves to computational implementation, but none has been developed that I am aware of. This I attribute to the relatively small audience (urban environmental designers), and the relatively slow development of computer tools in the field, rather than to any shortcomings in the ideas themselves. Others have developed specialized descriptive systems for environmental analysis, including [Thiel] [Hillier & Leaman], [Higuchi]. All of them, though less well known than the two above, are candidates for algorithmic implementation and exploration. [Alexander 72] described and prescribed 'patterns' for the design of environments, packaging a vocabulary of simple terms together with aggregate qualities and behavioral results of the patterns -- in effect providing a 'knowledge based expert system' for the design of spaces ranging from the urban organizational scale to the level of paving materials. [Kasmar] reported on a lexicon of empirically derived environmental descriptors, based on subjects' descriptions of their surroundings, and proposed two fundamental categories of 'physical attributes' and 'esthetics and mood'. All of the above cited literature has provided a vocabulary for describing spatial environmental qualities, and a framework in which to pursue systematic designing, but not so far with the explicit intent of formulating computational (algorithmic) realizations of the descriptions. I'll concentrate in this thesis on physical attributes, understanding that they are connected in designers' minds, and in practice, with esthetics and mood in ways that are not amenable to analytic description or computation.

Perceptual/behavioral psychologists have also contributed to understanding the ways humans internalize and think about space [[Tuan]. [Hillier & Hanson] provide a metric for analyzing connectedness of enclosed spaces, and a theory about how this measure affects human behavior and organization. The distinction between *perceptual* and *conceptual* [Kaplan & Kaplan] is echoed in this present study. Following the work of Lynch and others, several authors have explored the phenomenon of 'mental maps' and their role in navigation and reasoning about environments (for example, [Kuipers]). [Retz-Schmidt] surveyed the ambiguities inherent in verbal descriptions of maps and directions arising from the relativity of prepositions to subject and object (for example, "to the left " may mean either to the left of the person giving the description, or the left of the person receiving the description, or to the left of some object that has a 'front'.) While the problem of resolving this ambiguity is substantial in interactive systems including robot navigation and 'speaking maps', for example, I 'll dodge the problem of ambiguity in relational terms by always using descriptions in a plane with a fixed origin and refernce system, and generally avoid directional descriptions, focussing on relations that are invariant under rotation or translation.

Even without the ambiguities of language, reasoning about space is a complex cognitive ability that calls upon all our mental abilities, including vision, kinesthetic coordination, memory and imagery. Studies of the first and last of these, vision and imagery, have concentrated on two approaches, loosely characterized as neuro-physiological and information-processing models. The neuro-physiological studies attempt to localize and describe the physical mechanisms of vision and imagery (for example, [Livingstone]); information-processing models seek to complement these by describing data structures (information) and algorithms (processing) that illustrate or plausibly reproduce (or less frequently, actually mimic) the proposed neural mechanisms, and incorporate them into a larger representational or computational model of mind in general ([Marr] , [Minsky], [Fodor], [Haugeland] , e.g.) The study of visual cognition, that spans the phenomena of vision and imagery, combines these two approaches [Pinker]; most studies have not particularly addressed the role of vision or imagery in designing in particular, but rather as a general mental phenomenon

(although the role of imagery in scientific thought has received specific attention [Block]). The apparent preference for a loosely defined simplicity ('good form') in perception was emphasized by the Gestalt school of psychologists [Koffka]; more recent researchers have proposed more detailed definitions of visual primitives [Attneave] and visual routines necessary for recognizing and processing them [Ullman]. I'll take the approach of building on, rather than attempting to explain, the human visual cognitive system, and join the ranks of those who seek to explain cognitive activities, like designing, as the behavior of systematically organized and interconnected 'symbol systems' [Simon].

Descriptive models of designing go back, some would say, to Vetruvius [Heath]; the chain of literature more directly relevant to the present study comes from the study of design theory and methods [Jones] [Archer] [Broadbent] and [Moore]; a milestone is marked by [Simon]; and current efforts continue vigorously, including [Cross] [Habraken] [Akin] [Schön 88] and others. The engineering community has also studied human design processes, especially in electrical and mechanical engineering design [Stauffer & Ullman]. Most of these models consist of sets of categories describing design 'phases', or sometimes types of design information, and showing relations between the phases and types of information. Many would now agree that no strictly chronological description of designing is valid, though the generalizations made by Archer and Broadbent, describing the linkage of 'inventory', 'analysis', 'synthesis' and 'evaluation' are considered reasonable and useful guidelines (for the billing and accounting of professional design time, for example.) The ideas that there is a distinctive mode of 'design thinking', and learned techniques for reasoning about graphics, and that there are different types of design information, and transformations of representation which may be loosely correlated to chronology, are central to the theory of designing and diagramming explored in this thesis.

[Dark], [Simon], [Akin], [Schön] and [Goldschmidt] all report studies of designer behavior by protocol analysis. In each case, some finding or conclusion is reported, as hypotheses or

explanations offered to interpret or make sense of the observations. These generalizations are useful as insights into the mental processes of designing, and the categories of knowledge unique to designing. As I explain below, some of the results inform this study, but the method is not adopted.

Approaches to computating in designing have mostly taken the form of particular tools intended to aid designers, often developed in an ad-hoc manner. Some efforts have emphasized implementation of a formalism that attempts to capture some aspect of the subject matter. [Alexander] described an approach to hierarchical structuring of environmental design criteria and an associated technique for decomposition and decoupling of design tasks; [Alexander & Manheim] and [Milne] described computational implementations of that model. The idea of 'blocks' of related information, that may be relatively independent is a central insight into cognition in general and the problems of managing complexity inherent in designing, and has a number of computaional implementations.

A large part of the difficualty in developing computer tools for design is related to the problem of unearthing what designers know -- most experts in any domain are not expert in describing their expertise! Some efforts at design computation have been directed at collecting and organizing specifc domain expertise. [Yessios] provides an example of capturing site-planning-specific knowledge of subdivision layout in a graphics production system. [Porter], [Negroponte & Groisser], and [Weinzapfel, et al.] described computer systems designed for exploration and elaboration of a comprehensive data base in urban and architectural design, and [Friedman] proposed a hypothetical program for decision making in an urban context. [Mitchell] has argued for the development of formalisms in describing architectural design problems; [Stiny, et al 80-88] have proposed one such formalism for describing and generating graphics that they claim captures essential design knowledge; [Fleisher] provides a critique of this so-called 'shape grammar' work. In this thesis I explore a general problem of knowledge representation and transformation with reference to specifc examples in the domain of urban design.

A significant formalism with related procedural methods is concerned with the role of 'constraints' in designing. The term is related to optimization theory in engineering, and most of the early work in constraint theory was concerned with engineering design problems, especially electrical engineering [Sutherland] [Freidman & Leondes] [Steele & Sussman]. Constraint satisfaction and propagation have become standard techniques in the arsenal of searching and problem-solving techniques employed by AI researchers involved with complex interacting systems [Winston]. Subsequent development of the approach applied constraint theory to problems in conceptual design in engineering [Navinchandra] [Serrano and Marks]. [Gross 85], [Ervin and Gross 86] , [Ervin 87] and [Gross et al 88] have explored the application of the theory to environmental and architectural design, and provide a base of computer software upon which the present development will build.

The computational implementation side of this research also builds on previous and ongoing work in the general field of automated reasoning and artificial intelligence. Research into automated reasoning has historically been concerned with applications in logic and theorem-proving [Hayes] [Wos]. Results from this research have been applied to a number of application domains, in the form of rule-based production systems, or so-called 'expert systems' [Hayes-Roth]; the use of expert systems in architecture and urban design has been limited [Gero and Coyne], [Gero]. This body of research has led to the understanding of the importance of the form of 'knowledge representation' employed by such systems [Brachman], and the roles of 'common-sense' and approximate reasoning in ordinary human reasoning.

A major approach to these issues in knowledge representation is based on the notion of 'frames' and 'defaults' [Minsky]; this along with theoretical considerations in computer science has given rise to the approach known as 'object-oriented' programming [Borning] [Beyers] [Agogino]. Study of the behavior of frame-based and object-oriented systems utilizing defaults for reasoning has led to several inquiries into the nature of 'non-monotonic' reasoning, which is a characterization of an

important distinction between formal, logical reasoning and flexible, common-sense, default-based reasoning [Ginsberg 87]. The present study uses an object-oriented knowledge representation scheme, with defaults, and recognizes the importance of the issues and problems raised by considerations of non-monotonic reasoning, as will be expanded upon in chapter five.

Computer hardware and software engineering research has provided some insights into diagrams as tools in system analysis and design: [Glinert and Tanimoto], among others, describe the use of diagrams and flowcharts in software development; [Liu and Wong] present an algorithm for the construction of a special type of node-and-arc diagram, as a tool for visual inspection of distributed computing systems; [Barnden] explored 'diagrammatic data structures' as an approach to parallel distributed processing in a computer science/ neural-network context. [Funt] described using diagrams in a computer-vision and reasoning system, making the point that the diagrams served as a model of a real world situation, from which inferences could be reached by a parallel-processing visual system -- his diagrams were technically 'projections' (elevations in all his examples), and the inferences were all deductions about geometric properties and motion in a plane.

Specific attention to diagrams in environmental design in any of this literature has been limited. Most sources treat diagrams indirectly or by example: [Herdeg 81] provides a compendium of examples of diagrammatic graphics, but only as a sourcebook of examples, and not from an analytical viewpoint; [Albarn & Smith] used diagrams to illustrate some points about analogical thinking and creative mental processes; [Graf] discusses diagrams as tools of architectural criticism; [Laseau] and [Crowe and Laseau] touch upon the role of diagrams as particular kinds of graphics used in architectural design; [Nelischer] illustrates a unique approach to diagram graphics in landscape design.

[Alexander 66] referred often to diagrams in his model of designing, and formulated the distinction between 'form diagrams', 'requirement diagrams' and 'constructive diagrams', but he used the term

to cover all kinds of graphics, models and other representations. [Milne] reported on a system of using diagrams derived by Alexander's decomposition technique in architectural and industrial design. In that work, the diagrams are referred to as 'form-tendency' diagrams, and the understanding was that the diagrams were, in a very literal sense, the foundations of a design solution. [Alexander and Manheim] presented a study of "using diagrams in the preliminary design of a roadway"; I'll argue in chapter two that they were using maps, not diagrams.

A formal treatment of modes of graphical representation and their information content, including this distinction between maps and diagrams, is found in [Goodman], I'll expand upon some of his definitions later. [Simon and Larkin] have explored the distinction between "diagrammatic" and "sentential" data structures, concluding that the diagrammatic data structures offer real efficiencies in algorithmic reasoning, primarily by reduced search time in comparison to linear sentential structures. [Goldschmidt] presents a study of the role of sketching in design exploration, and distinguishes diagrams from projections on the basis of their involvement in reasoning processes described as pictorial, or 'seeing as' (projections) and discursive, or 'seeing that'[1] (diagrams). I'll adopt a similar terminology in this thesis, distinguishing between pictorial and propositional representations, but disagree with Goldschmidt in minor ways (distinguishing between shape and form, for example.) [Lindsay] argues "from common-sense" that diagrams are primarily useful because of their role in inference, an argument that I'll repeat.

The closest works to the current study are two: the first was a proposal [David] for a diagram language for designing. This proposal was promising in outline, but failed to present a development of the ideas. The second related study is reported in [Montalvo 85], where the problem of diagram understanding is posed as the intersection of computer graphics and computer vision research. Subsequent research [Montalvo 86] has focussed on the development of a computer graphics system

[1] Wittgenstein originated this terminology, which has been used by Schön among others.

built on a set of geometric relations and primitives derived from [Bongard], which I list in chapter two.

The idea that complex environments could be described with a set of abstract, aggregate descriptors that have a correspondence to topological and geometric primitives [Lynch]; the idea central to 'object-oriented' and 'frame-based' computing that information can be encapsulated in relatively autonomous internally structured blocks which provide 'defaults' for missing values [Minsky]; the idea that constraint expressions can be used to model and compute the effects of relations between objects [Gross]; and the idea that visual processes may be described as at least partially inferential[Arnheim], although often 'nonmonotonic' and not necessarily rule-based, but possibly computable [Ginsberg]; are the central ideas on which this work is founded.

The present research is unique in its scope attempting to locate diagramming in a system of other design processes, its focus on diagrams as mechanisms of knowledge representation and inference, its exploration of the concept-to-diagram-to-drawing transformations, and the coupling of a default-based reasoning scheme with an object-oriented, constraint-based, interactive computer graphics system in the domain of environmental/urban design.

Research Question & Method

The first assumption stated above -- "that diagrams are a unique kind of graphics, distinct from maps, plans or sketches, that play a special and important role in environmental designing" -- has two parts. That diagrams are used by designers is empirically established; I display and describe a range of diagrams and other graphics from environmental design literature that provide the data to support this claim. That they are distinguishable from other graphics requires a theoretical basis for the distinction; I provide some guidelines for making the distinction and show how they help in characterizing the role of the graphics in design processes.

The second assumption "that the structure and function of diagrams must be understood with respect to their inferential potential -- as stimulus and medium for reasoning -- " requires an investigation into design thinking and visual cognition. I propose a role for graphics in design reasoning, and illustrate several examples of that reasoning taking place. I describe in detail some of the visual properties that enable the reading of and reasoning with diagrams, with reference to the examples already displayed.

The third assumption -- "that a computer program can be built that produces intelligent and intelligible diagrams" comes from an underlying 'information-processing' approach [Simon] [Akin], in which the driving questions are "What's the information?" and "What are the processes?", and leads directly to the particular research question: "What knowledge, in what form, must be given to a computer program that might produce diagrams?" I assume a context and a purpose for the production of the diagrams (based on the theory about the role of diagrams in designing) and describe the structure and behavior of such a computer program, based on an information-processing approach to designing.

I take this last question to be an epistemological and to some extent a cognitive one, but not a psychological one. That is, I make claims about the form, origin and intent of knowledge and its use in diagramming; but no claim about the actual mental process(es) of human designers. It is a prescriptive claim as much as a descriptive one: I argue not that designing necessarily is done some way; but that it might (and beneficially) be. These two considerations reflect on the nature and method of the research: I extrapolate from some graphical data (diagrams) and subjective design experience (introspection) to a set of proposals about what a diagrammer needs to know, and what it needs to be able to do, that plausibly explains the structure and function, and origin, of the particular diagrams, and of diagrams in general. I am not reporting on experimentation on designer subjects and their behavior, either for the purpose of making generalizations about design

reasoning or making inferences about mental processes, but I do make some suggestions about what to look for in such research.

The data presented are empirical -- diagrams from published sources, some along with accompanying commentary -- but not reproducible, in that the same designers next time might well produce different diagrams and different commentary. This irreproducibility is part and parcel of the designing process; designing seen as a path has no single determinable end-state, even in the presence of consistent and proscribed starting states[2]. Therefore the predictive power of my thesis is limited; it predicts not what particular diagram will be produced under controlled circumstances, but rather what features any diagram must have, and provides a terminology for describing diagrams and their use.

Like other research enterprises in artificial intelligence, this approach is not the traditional scientific one of formulating a theory to explain some observed set of facts, and then testing the theory experimentally; the facts remain too disparate, and the purpose and scope of a theory too vague. The computer program constructed is an experiment, however, and results of its use suggest a next round of experiments. Eventually, with enough experience collecting observations, the conditions may develop for a comprehensive theory and a rigorous terminology of diagramming.

The method of this research is to derive a data structure and algorithm from some empirical data (diagrams), and then to implement these in a computer program. The program is used to try to reproduce several of the specimen diagrams. The production is undertaken in a general sense; that is, the program is not tailored to the specific diagrams, but designed with data structures and a diagram-production algorithm in such a way that the particular diagrams are but examples.

---

[2] In this sense, it may be appropriate to consider designing a 'chaotic' process, highly sensitive to minor deviations in starting states and moves.

Predictably, the diagrams produced are different from the originals in several ways. These differences can be attributed to either deficiencies in the original input to the program , or the logic of the several parts of the diagram production process. Two examples of these reproductions and analysis of the differences provide some detailed evidence about the importance of visual properties in graphics, the difficulties in capturing this knowledge in procedural form, and the inferential potential of these visual properties.

The proposals I make about the structure and function of these diagrams are intended to provide a framework for understanding the diagrams and the diagramming/design process from an information-processing point of view (i.e., What's the information?, What are the processes?). These proposals are tested in the development of a computer program, and in investigating the form, quality and quantity of knowledge necessary to cultivate it in the direction of being a tolerably good diagrammer. The aim is modest, and highly dependent on partnership with a human designer. That is, there is no attempt to completely automate diagramming, or to replicate the abilities of a human designer/diagrammer. There are many parts of the phenomena that remain mysterious and well outside of the scope of this effort. The visual system and all its powers, and the design mind, with all of its, are to be exploited by the diagrammer, and not to be explained or replaced by it. In the description of a system for computer-aided diagramming, I identify the role and location of explicit (computable) knowledge versus the role and location of external 'homunculi' that are necessary partners in the designing and diagramming process.

Synopsis

In chapter one I observe that designers use diagrams, and ask the questions "Why? " and "How?".

A preliminary observation is that in general graphic products -- sketches, maps, plans and diagrams -- can be graded by the degree and kind of abstraction they embody, and hence what kind and quantity of information they contain. Sketches are intended to be evocative of 3-d shape and

outline without being detailed. Maps are not intended to be evocative, but true to 3-d position and size. Both sketches and maps rely on visual verisimilitude and direct perception, they are the most pictorial representations. Plans and sections are abstractions emphasizing dimension and material; they rely on icons, symbols and graphic conventions, and are less pictorial. All of them (sketches, maps, plans and sections) are directly concerned with physical form. Diagrams, by contrast, are abstractions emphasizing 'form-concepts' -- organizing principles and relations between physical elements that give rise to, but do not directly express, form; they are the least pictorial graphics, and in general are insensitive to graphic distortions, like rotation and scaling.

In chapter two I examine a handful of diagrams from urban and environmental design case-studies. These diagrams are both analytical and generative in their origin, and some are accompanied by texts of design reasoning or commentary. I analyze these diagrams in their role as reservoirs of design decisions, and as catalysts to design development. From these case examples, I argue that these diagrams can be understood as geometric, topological expressions of abstract design goals such as 'community, 'privacy, 'containment, 'access, 'barrier, 'connection, 'enclosure, etc., and propose a list of geometric and topological relationships found in the diagrams. These example diagrams are data from which two questions arise, (or a single question with two different emphases): "How come these diagrams?" , and "How come these diagrams?" That is, I am looking both for a general mechanism to describe the origin and purpose of diagrams, and for explanations for the specific features of individual examples.

In chapter three, I present a model of designing with diagrams. I argue that a central function of diagrams is to generate a visual, graphic representation of design goals, by matching graphic devices with design elements, concepts and relationships. These graphic representations (diagrams) then serve two purposes. First they enable the designer to make use of the complex, high-bandwidth human visual apparatus and the reasoning process called 'inferring by looking';

they serve both a a stimulus to, and a medium for, reasoning about the design. Second, diagrams provide a skeleton upon which more detailed, more particular drawings can develop.

I distinguish between A-diagrams, primary diagrams that derive directly from concepts and have little relation to particular form; and B-diagrams, which are either generated by abstracting from some more detailed drawing or created by developing an A-diagram. In short, A-diagrams are more abstract and less pictorially developed than B-diagrams. I present a schematic model of the relationship between concepts, diagrams and other, more pictorial graphics, generalized as 'drawings'. The model describes two processes of transformation between diagrams and drawings: Diagram⇒ Drawing, called "design development", and Drawing⇒ Diagram, called "creating a B-diagram". There are also two processes relating diagrams and concepts: Diagram⇒ Concept, a version of "inferring by looking", and Concept⇒ Diagram, called "creating an A-diagram". I identify this last process -- producing A-diagrams -- as a critical part of designing with diagrams, and choose it as a good candidate for exploration by computer, arguing that a computer program to produce diagrams is both possible and useful in the development of computer graphics and computational design tools. I discuss the transition from A-diagrams to B-diagrams, as the process of moving from diagrammatic to design development drawings -- a familiar process in environmental design.

Chapter four presents an algorithmic approach to the construction of diagrams. This is based on the motivation for producing diagrams that comes from the model of designing just presented and the vocabulary of concepts and terms found in chapter two. I discuss the issue of making graphical commitments to design decisions not yet committed to, and make some conclusions about the the need for, and role of, defaults in the diagramming process. The algorithm contains five essential steps, characterized as: 'declaration, 'filtering, 'mapping, 'layout and 'inferring. The first and the last of these are the prerogative of external intelligence (the human designer, e.g.), the middle three

are within the reach of a computer program. The program developed here is devoted to the layout stage, assuming a filtering and mapping done by the user.

In the first part of chapter five I review the constraint model of designing, and argue that the model is well-matched to the model of designing with diagrams, as it involves maintaining and exploring relations between elements. The model provides a foundation (in the form of the program called the constraint manager) for a computational implementation of the process of producing A-diagrams.

In the remainder of the chapter I detail the architecture of the program called the 'Constraint Based Diagrammer' (CBD), incorporating the constraint manager. The several different steps from the algorithm in chapter four, and the data-bases they require, are described. A small vocabulary of default diagrams and their constraint representations is displayed, along with a strategy for resolving conflicts that arise when constraints from several different default diagrams interact.

In chapter six I present two examples of diagrams produced by the CBD, reconstructing two of the original case examples presented in chapter two. I demonstrate that the constraint environment for constructing diagrams is a useful one for explorations of this sort. The reconstruction attempts highlight a number of things about the original diagrams that are not easily captured in a propositional representation, and must be explained by reference to the phenomena of visual cognition. Several problems that require user-intervention, including an example of a diagram that is beyond the scope of the simple CBD, are described.

In chapter seven, I explore in detail the results of these explorations and evaluate the knowledge base on which CBD rests. I propose a partitioning of this knowledge, into three general categories: knowledge about fundamental principles of visual cognition; diagramming rules that use the former

knowledge to produce legible and evocative graphics; and domain-specific, substantive knowledge that gets cast into diagramming rule and graphic conventions. I argue that in the context of designing, the phenomena of visual cognition can be located with respect to three different kinds of inferences that proceed from diagrams -- "map-like deductions" that utilize literal translation from 2-d space to 2-d space (and so are best supported by 'B-diagrams'), "map-inspired analogies" that translate from 2-d spatial properties to analogous non-spatial properties, and "other visual analogies" that depend on associations with non-spatial visual properties.

In chapter eight I summarize the findings of the previous chapters and conclude with a consideration of some questions for continued research and development.

Appendix 1 illustrates the architecture and knowledge requirements of a comprehensive diagramming system, including three companion programs -- the diagram-combiner, diagram-fitter, and diagram-reader, and details some questions about each of the three stages of the diagramming algorithm. Appendix 2 illustrates the use of the CBD to produce diagrams from outside of the domain of urban design, demonstrating that the approach of this thesis is a general one. Appendix 3 is a listing of the constraint manager representations of several of the diagrammer's primitive elements and relations.

Potential Benefits

The furtherance of systematic design depends upon the development of some basic theories about things designers do, including producing a variety of kinds of graphics, and reasoning with them; and also upon implementations of those theories in interesting and useful computer tools. In the beginning, of course, the computer program is disposable -- it is the possibility of a theory about the role of diagrams in designing, that relates their structure and function to other graphic artifacts and to design reasoning processes, that motivates this inquiry. The computer program presented here is a learning environment, an experiment, not a product or a 'design tool'. I do look forward to

computational design tools that take advantage of the power and peculiarities of diagrams, and along the way and in Appendix 1 I speculate about a designing environment composed of some such computer programs.

The value of such computer tools will be first as experimental devices to further probe the nature and role(s) of graphics in designing, as windows into the design mind, and eventually as useful tools for design development, amplifiers of the design mind. The connection to procedural, constraint-based descriptions of designs offers the promise of rule-checking and automated inference to assist designers in managing complexity in detail; the access to a library of default strategies and mechanisms of abstraction and aggregation in the diagrammer should offer assistance in managing complexity in the large, without being bogged down by details. Both abilities are largely missing from computer modelling programs today.

Understanding diagrams from an information-processing viewpoint offers the promise of integrating, if not reconciling, the competing symbol-based and imagery-based approaches to cognition, and to shed light on the connection between symbolic and pictorial information that are both essential to design thinking.

Chapter 1. What's in a Diagram?

"...scientists and philosophers have on the whole taken diagrams for granted..."
Nelson Goodman
Languages of Art, p. 171

The same can surely be said of designers, architectural, urban and environmental. "Taken for granted" does not mean dismissed, but relied upon, largely unexamined. Bubble diagrams are part of every design student's heritage, although their role in designing is unclear: design instructors' sentiments include both

"That design is poor -- you have merely built the diagram."

and

"A good site plan can be summarized in a palm-sized diagram."

These apocryphal comments are not outright contradictory, but they do reflect a certain ambivalence about diagrams -- their status is close to a necessary evil. Why? What are diagrams? How are they like and unlike other drawings — maps, plans, sketches? What's necessary, what's evil, about them? Why might it bad to build a diagram? What are the alternatives? How is a plan summarized into a diagram? What role(s) do diagrams play in designing?

In order to answer these and related questions, I need to look closely at diagrams, and at drawings, graphics and visual reasoning in general. I shall need a model of designing to refer to, and will consider the role(s) of inference, abstraction, and communication in designing. In this thesis I shall first attempt to distinguish diagrams from other graphics, with reference to their role in design inference, and then consider in detail what features of diagrams and diagramming need explaining.

Diagrams as graphics

The derivation of the word — Greek διαγραμμα, to mark out by lines — suggests that any line drawing is a diagram. Indeed that is the logical, if absurd, conclusion of an effort to define the

boundaries of the term. Like many useful labels, the criteria for application are fuzzy at the edges. Common usage and dictionary definitions might help:

> "A figure drawn in such a manner that the geometrical relations between the parts of the figure illustrate relations between other objects."
> Encyclopedia Britannica. 11th Ed.

> "A graphic design that explains rather than represents."
> Webster's International Dictionary, Second Edition

Evidently, some drawings are diagrams and others are not. Compare Figures 1.1 and 1.2, for example.

Figure 1.1 A diagram

Figure 1.2 Not a Diagram

But how are we to understand 'explains' as opposed to 'represents' ? Representations present the thing itself — explanations refer to some other thing, cause or principle. So while representational graphics point our attention at the explicit, surface characteristics of the manifest object (subject) of the graphics, explanatory graphics (diagrams) point our attention beyond the surface or into implicit characteristics, or relations. Now designing, whether of cities, buildings, or furniture, is a wholistic activity concerned equally with manifest surface characteristics, such as shape, color, texture, and less visible relations such as focus, proportion, structure. So it is no surprise that

designing typically involves both representational and explanatory graphics. But are they really distinct? And in any event, how are they related?

Consider the differences between figures 1.3 and 1.4 (next page) , two different drawings of the new-town Columbia, Maryland, conceived and built in the early 1960's, taken from a study of urban form [Chermayeff & Tzonis]. Figure 1.3, a simplified map, contains an explicit scale marker, implicit north arrow (up is north), irregular polygons indicating parcels of land ownership, a squiggly line representing a lake, and so on. It contains information about Columbia's spatial extent, shape and component parts. Figure 1.4, a diagram of schematic organization, illustrates several points about Columbia's structure and organization: it's located midway between two urban magnets (Washington and Baltimore), it has a hierarchical structure formed by a distinct center surrounded by several equal-sized sub-centers, the sub-centers are linked internally by a system of transportation that is qualitatively different from another system that surrounds them and links them to a larger, inter-urban system, and so on. The diagram shows the two magnets left and right; on the map, Baltimore is to the top, Washington at the bottom. The diagram shows the surrounding loop transportation system as rectangular; the map shows that it is in fact a an irregular polygon, and one suspects that it is fitted to the topography. The diagram shows the sub-centers arrayed on a regular grid; in the map they are scattered around inside the loop.

Now in fact, this diagram was produced *a-posteriori*, as an explanatory aid to discussion of urban form [Chermayeff & Tzonis]. But might not a similar diagram have hung on the wall when Columbia was first being designed? It really doesn't matter. The genesis of the diagram is one thing; its content and import are another. I'll consider both, and their relationship, in this thesis.

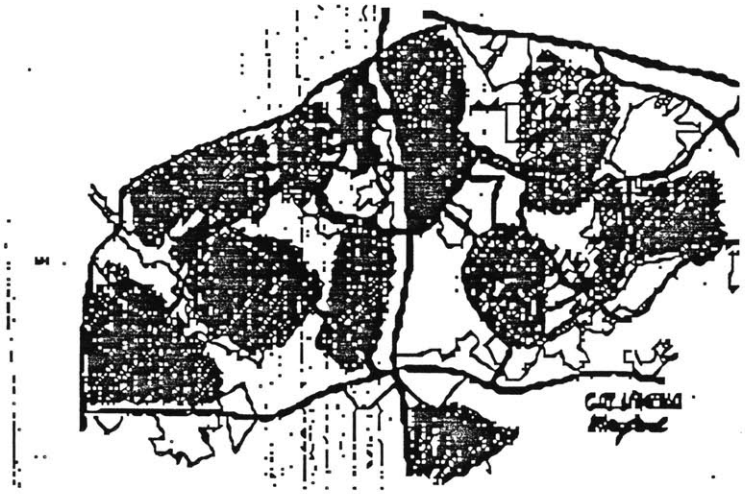


Figure 1.3 Map of Columbia New Town

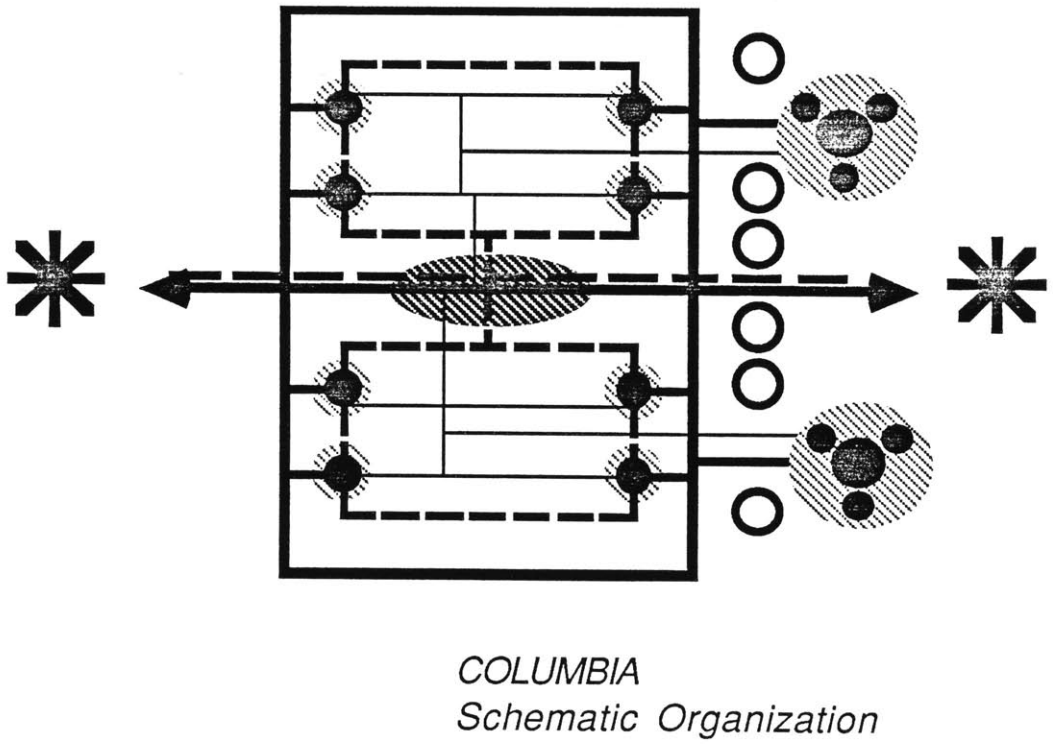


Figure 1.4 Diagram of Columbia New Town

Diagrams as generators

Considering the role of diagrams in designing, we must understand how diagrams are more than 'explanatory'. Explanation sounds as if it must necessarily come *a-posteriori.* Indeed, this is one way diagrams are used, especially in the context of communication between design partners (clients, designers, etc.) But diagrams serve another, possibly more important, and certainly more mysterious role:

> Since ... diagrams contain the kernel of the visual effect,... from them a more concrete design may be developed. They will therefore serve the function of the preliminary rough sketch in ordinary design procedure... The chosen scheme will guide the detailed design...
> [D. Appleyard/ J. Myer/ K.Lynch]
> The View from the Road, p. 25

In the mind of one or all of these distinguished designers, there is an 'ordinary design procedure' in which rough sketches guide detailed design. Diagrams, since they contain a 'kernel', can also serve that purpose. How does a rough sketch or a diagram guide detailed development? Of what does a design kernel consist, how does it germinate, and what nourishment does it need?

Diagrams have been described as "scaffolding -- a necessary, or useful construct for the development of a design, that disappears when the design is done"[1] . If that is the case, of what is a diagram made, and what does it support?

Whether conceived of as kernel or scaffolding, diagrams appear to have a generative power as well as explanatory. This is important for a theory of design, as it appears to localize at least some of the elusive generative, or creative, force in designing. It suggests that diagrams may come *a-priori,* and that their relationship to subsequent detailed design is causal in a way. If we are to understand the structure of diagrams, it must be with reference to this (at least) dual function. In

[1] This particularly evocative description comes from Avraham Wachman.

turn, I shall be able to design the functionality of a computer system with reference to this underlying structure.

Diagrams as symbols

In *Languages of Art*, the philosopher Nelson Goodman provides a theory that formally distinguishes classes of symbol schemes (which include pictures, musical scores, and written descriptions). A first distinction is between 'syntactically articulate' and 'syntactically dense' schemes — the former constitute proper notational systems, and include electrical wiring diagrams, and most verbal descriptions; the latter, not properly notational, include scale models, and most paintings and diagrams. Many pictures, like roadmaps, contain mixed dense and articulate schemes (pp. 170-173).

Among dense schemes, Goodman considers the distinction between the diagrammatic and the representational:

> ... While there is an at least theoretically sharp line between dense and articulate schemes, among dense schemes the difference between the representational and the diagrammatic is a matter of degree. ... one among a familiar category of familiar schemes may come to be regarded as purely representational if its constitutive aspects include those of all the other schemes; then the schemes that exclude as contingent some of the constitutive aspects of this representational scheme are regarded as diagrammatic.
> Languages of Art p. 230

The scale along which diagrammatic and representational schemes are separated by degrees is unspecified, but the notions of constitutive, or necessary and relevant, and contingent, or optional and irrelevant, aspects in a scheme are useful. This distinction crystallizes the sense that diagrams contain 'key' or essential aspects of a design being developed, and omit ("exclude as contingent") other aspects. What makes an aspect 'key', and what effect this has on the designer's inferential processes or the development of the diagram, are still open questions. This hypothesis that there is a scale from pictorial to diagrammatic is provocative, for it suggests a possible task for a computer

tool, and a requirement for a knowledge representation scheme: given any diagram/picture, it should be possible to procedurally produce another picture/diagram that is either more diagrammatic, or more pictorial. The particular mechanics of such a procedure will be related to the chosen knowledge representation scheme. Goodman gives another hint about these mechanics, in describing the process of 'attrition' from representational to diagrammatic schemes:

> ...Though pictorial and diagrammatic schemes are alike in not being articulate, some features that are constitutive in the pictorial scheme are dismissed as contingent in the diagrammatic scheme; the symbols in the pictorial scheme are relatively replete...While attrition from the representational to the diagrammatic is by restriction upon the constitutive syntactic aspects of the symbols, attrition from full to narrower pictorial exemplification is by restriction upon the constitutive aspects of what is symbolized.
> Languages of Art p. 235

This describes a specific, if obscure, mechanism for realizing the transition from representational to diagrammatic, and by implication, the reverse. Given a picture/diagram, restricting the 'constitutive syntactic aspects' of the symbols therein will produce a more diagrammatic result, and expanding those aspects produces a more pictorial result. The implementation of this mechanism then turns on the meanings of 'restrict' and 'expand' as applied to syntactic aspects. In chapter three I explore the restriction and expansion in more detail.

Diagrams and maps

Goodman's terminology helps to describe the difference between figures 1-3 and 1-4, the map and the diagram of Columbia. The formal relationship between the two drawings is as suggested by Goodman's distinction. In the diagram, the *shape* of the loop-road is contingent, its *topological connectivity* is constitutive. The *orientation* of the line between magnets is contingent; the position of Columbia *between* them is constitutive. The exact *number* and *size* of sub-centers is contingent; their *plurality* and approximate *equal-sized-ness* are constitutive.

This distinction between diagrams and maps is what Kevin Lynch had in mind when he wrote:

> Graphic diagrams, with words appended, are still the prime ways of conveying [descriptions of settlement patterns.] Mathematics is steadily becoming more important for doing so, particularly by means of topology, since many of the important spatial relations in settlements are nonmetrical. Insides and outsides, connectedness, gradients, grain, dominance, foci, enclaves and density are form concepts likely to be more critical than such geometrical analogies as square, triangular or round....
>
> GOOD CITY FORM, p. 357

Is the distinction between geometrical and topological[1] enough? How else might one characterize the distinction between shape, orientation, number and size on the one hand, and connectivity, between-ness, plurality and equal-sized-ness on the other? It might be characterized as absolute versus relative, which is partly right, but not entirely, for surely drawings can have relative size as well as absolute equal-sized-ness. The differences might be accounted for by resolution, or level of detail, which is again partly right, but not entirely. I will call the distinction *particularity* and *abstraction* , and mean to include all of the above nuances. In so doing I am using both these terms well within their common meanings, but their meanings and this usage are central to the following discussion.

Maps	Diagrams
(Particular)	(Abstract)
Geometrical	Topological
Absolute	Relative
High Resolution	Low Resolution
More Information	Less Information
Pictorial	Propositional

In light of these distinctions, it is useful to consider some other maps and diagrams in the literature. [Alexander and Manheim] in an early study of applying systematic design to highway layout,

---

[1]Like Lynch, I use the term topological in the limited but common sense of equivalence of planar figures under rubber-sheet distortions including scaling and rotation, emphasizing the properties of connection and adjacency, like its usage in electrical engineering [Sussman].

described "a system of constructing and overlaying diagrams". These diagrams were formulated as 'requirement diagrams' in Alexander's terms, that were added together in a combination of mechanical and intuitive techniques to produce 'form diagrams' and finally a proposal for a highway layout. Fig 1.5 shows two of the source diagrams of factors, leading to a final proposed layout:

Figure 1.5 Combination of Diagrams for Highway Layout

The technique for constructing the diagrams is described thus:

> ."... we construct a diagram, for each requirement, by assigning a utility to each point on a map of this terrain. Each diagram is a pattern of greys whose density varies over the complete range from black to white. The pattern is keyed to the base map of the terrain in such a way that a point marked black in the diagram for a particular requirement is a very good point for a highway location to pass through (from the point of view of that requirement.)
>
> Alexander & Manheim p.33...

Seen in the light of the distinctions in the table above, these figures are maps, not diagrams. The critical factors are two: in each of the figures, scale and orientation are constitutive -- it is only because they are at the same scale and orientation that they can be overlaid as they are. Secondly, each graphic mark is a metric description of some quality at a particular point in two-dimensional space -- cost of construction, noise impact, etc. It seems fair to say that the study and the technique presented by Alexander and Manheim was an early example of what has come to be known as the 'map overlay method' [McHarg , Tomlin] or a 'geographic information system' (GIS) [Burrough].

It is worth noting that Alexander and Manheim's technique differed from most GIS or map overlay techniques in the "intuitive component", an important part of the process they describe that defies explanation or algorithmic implementation:

> "Now we redraw this composite diagram, bringing out its principal pattern characteristics as strongly as we can...
> The eye, being what it is, we can always detect an underlying pattern in such a diagram, and we can bring this underlying structure out. This process is known as "levelling and sharpening" (from gestalt psychology)...
> ... certain configurations have "better" organization than others (in some sense not quite understood). "
>
> p. 111

This "better organization, ... not quite understood", represents the intuitive, non-algorithmic, gestalt judgement that Alexander previously referred to as "fit" [Alexander 1966], or in this study as "basic pattern properties". It is this element of abstraction, judgement made at some level other than geographic or metric, that gives a diagrammatic quality to these maps. As mentioned before, the distinction between kinds of graphics is bound to be fuzzy, and this is a good example. Nonetheless, it seems useful to try and be consistent and precise in the use terminology, and this critique is offered as an example of the distinction between maps and diagrams that I have in mind.

Alexander's use of the term 'diagram' was loose and seemed to cover all graphic and other kinds of representations, but one of his definitions points to a central quality of diagrams:

> "Any pattern which, by being abstracted from a real situation, conveys the physical influence of certain demands or forces is a diagram... "
> Notes on the Synthesis of Form p. 85

In this definition the emphasis on physical influences is too restrictive, and the looseness of "any pattern" is too vague, but the importance of abstraction is right on.

Diagrams as abstractions

> "...LeCorbusier was designing the *Ville Radieuse,* the early radiant city. It was as you know, a diagram. It doesn't belong to the real world... "
> Prof. Jacqueline Tyrrwhit,in Chermayeff & Tzonis, p. 193

What is abstraction and what does it have to do with designing? What do diagrams and abstractions have to do with the real world?

Almost all descriptions and models of designing, regardless of their differences in approach, intent, bias or origin, describe a progression in designing from more general to more specific [Cross] . The design brief, the design concept, the responsibility to named or unnamed constituents, the big question "Why?" -- all eventually find their way into specifications of material, placement, dimension, what, where, when and how. Ultimately, the product of designing belongs to the real world. Initially, however, it may start in some other, often idealized world. The garden cities of Ebenezer Howard gave rise to Columbia, Maryland; the *Ville-Radieuse* of LeCorbusier rose in multiple manifestations over the urban skyline.

Most models of designing dismiss any implied chronological sequence (starting with ideals, ending with concrete, e.g.), and accept that designing may have no necessary sequence, linear or otherwise. An early analysis of design processes [Archer 1965] presented a model of design development based on 'testing' design ideas using analogues. A continuum of kinds of analogues was identified, from "most abstract" to "most concrete":

> ... Words - Logic - Diagrams - Sketches - Drawings - Models...

along with the observation that designers should:

> "When testing *basic design ideas",* use most abstract model which will fully express it; when testing *a design embodiment* , use most concrete model which can be afforded." p. 80

The suggestion here is that "basic design ideas" are abstract ones, that take particular form in "design embodiments", and that diagrams, in the continuum, are at the threshold between the symbolic and the graphical. At this threshold, the tension between the abstract and the particular drives design. Serge Chermayeff, making the point that design spans two poles, practice informed by theory and theory informed by practice, observed:

> "...To say that you start with reality before you can usefully start abstraction is exactly the same as saying you must start with an abstraction before you can tackle reality. "
> ibid. p.30

Now, of course, saying the one is not literally the same as saying the other. So what might he mean? I think this designer was alluding to this circular, bipolar nature of designing. The tension between reality and abstraction drives designing, but doesn't depend on where you start. But abstraction is not fantasy, the opposite of reality. Where does the tension come from?

Again, we can start with dictionary definitions in an attempt to understand abstraction:

> abstraction:
> "the act or process of leaving out of consideration one or more qualities of a complex object so as to attend to others..."
> "...separates the parts or qualities of things for the sake of considering them in themselves, or in their relation to like parts or qualities in other things...thus, necessary to classification..."
> abstract:
> "considered apart from any application to a particular object; ... expressing a property, quality or attribute apart from any object or thing; ... opposite of concrete..."
> Webster's International Dictionary, Second Ed.

> Samuel Johnson defined the outcome of an abstraction as " a smaller quantity containing the virtue or power of a greater."
> Arnheim p.161

These definitions point to two related concepts: "classification" and "concrete".

Webster says abstraction is necessary to classification, but is not the same as classification. What's the difference? The difference is in the use, or purpose. We classify on the basis of abstractions. Classification in the taxonomic sense maybe called generalization, as it consists of identifying genera and species. As Rudolf Arnheim has pointed out, however, this is also a circular process:

> "In order to generalize one must first abstract, but in order to abstract usefully one must already know how to generalize."
>
> Henri Bergson, quoted by Arnheim p. 160

Arnheim, in his extended discussion of abstraction and its role in linking perception and thought, observes the importance of the distinction between "generative and central attributes and accidental or peripheral ones" to "productive thinking" [p. 174]; here "designing" may be substituted for "productive thinking". These are the constitutive and contingent attributes in Goodman's terminology. Note that Arnheim includes 'generative' with 'central', almost as though synonymous, although he does little to explore or explain this generative characteristic, although he does suggest:

> "In order to produce a sensible abstraction, a concept should be generative. It should be possible to develop from the concept a more complete image than that offered by the concept itself."
> p.174

Note also that Arnheim describes the developed image as "more complete", not "more concrete". In fact, he takes issue with the dictionary definition of concrete as the opposite of abstract (and suggests, for example, that the concept of "friendship" is just as concrete as any particular friend.) This observation, while counter-intuitive to the most common use of "concrete", does clarify the distinction between abstract and particular. Most urban/architectural design does in fact result in concrete being formed and placed, but we may accept that the abstract idea of 'civic plaza' is just as

concrete as any particular plaza, say Copley Square. I shall avoid the use of 'concrete', preferring the implications of 'particular', namely that a finer grain (particles) of description is taking place. Similarly, I shall prefer the term 'propositional' over 'abstract'. The sense is similar, but propositional will turn out to be more operative when I describe the process of designing and building a computer program. In general, then, the process of abstraction, when applied to some particulars, results in some propositions; when applied to a drawing, results in a diagram.

[Laseau] describes four approaches to abstraction: distillation, reduction, extraction, and comparison. Each of these can be considered in general terms, as aids to reasoning, and also as particular computational processes. Distillation is the process of selectively removing, or ignoring information according to some criteria; what is left is distilled, or 'essential'. Reduction not only selects relevant information, but groups and organizes it into fewer, more aggregate groups, to simplify the presentation and absorption of the information. Extraction in Laseau's terms is a technique of emphasis or highlighting, while leaving context in (not distilling) and leaving complexity in (not reducing). Comparison indirectly highlights relevant information by showing two or more systems side by side, with the same presentation; the perceived differences between the systems are then, apparently, the point of the presentation.

Computationally, the first two processes (distillation and reduction) are actually transformation of information, impacting a data structure; the last two (extraction and comparison) are really presentation (graphic) techniques.

Diagrams as approximations

A common interpretation of abstraction, especially in regard to diagrams, is approximation. In a way, this is appropriate -- diagrams often give approximate metric information, for example. Approximation is too limited a concept, however, and can give rise to problems in generating and

understanding diagrams. Approximation is essentially a metric concept, and diagrams contain more than metric information, and the information may be more properly considered 'tentative' rather than approximate. It may be appropriate in many cases to think of diagrams as 'first approximations', though, regardless of whether they arise from their generative purpose, or as abstractions of more detailed drawings.

The Place of Diagrams

Diagrams appear at the threshold between the graphical and the non-graphical. Though diagrams themselves are graphical artifacts, the information they contain is not essentially graphical; it is equally in some propositional (verbal/textual/logical/symbolic) form. Since diagrams are graphical, graphical inferences may be made from them, and these inferences may lead to a reformulation of the diagram, as part of an argument, or chain of reasoning. Likewise, since they are graphical, they may be further developed into more pictorial drawings. Seen the other way around, given a drawing, the transformation towards diagram is moving towards more abstract (less particular) information; detailed information may be aggregated (through 'reduction'), and graphical properties, such as shape, position, texture, size, dropped in favor of essentially non-graphical information (through 'distillation'). These two directions of design reasoning are taken up again in chapter three.

Diagrammatic Levels

If all figures lie somewhere on a scale between pictorial and diagrammatic, it should be possible to nudge any figure in one direction or the other, and change it to be more diagrammatic or more pictorial. Two problems arise upon contemplation of how such transformations might be described or achieved. First, the graphic image alone is insufficient to position the figure along the scale -- context is necessary. Second , no obvious quantum of motion along the scale suggests itself -- how shall we specify enough movement to produce a discernible change? Finally, there is no single

diagrammatic axis -- emphasis is needed along with context, and from any pictorial figure, a number of diagrams are possible. Nonetheless, the possibility of describing a specific transformation between more and less diagrammatic figures is tempting, as this would demonstrate a palpable distinction between such figures, which so far has been purely visual.

Supposing that we are willing to supply context and emphasis, we are still left with the analog problem of moving along a smooth transition. Unlike 'brightness', 'color' or 'contrast', 'diagrammaticity' does not lend itself to a smooth control between extremes. We can, however, imagine several distinct stops along the scale, or diagrammatic levels. These levels are suggested by the diagrams above: along the diagrammatic scale, the first four figures above might be ordered (most) 1.1, 1.4, 1.3, 1.2 (least.)

As 1.2 is a picture, or rendering, that is not diagrammatic at all, three levels will suffice to distinguish the other diagrams. At each level, some characteristics are included, others are excluded. From a higher (more diagrammatic) to a lower (more pictorial) level, more characteristics are introduced, and there is a specific relation between attributes on the higher level and their manifestation on the lower level. These levels should not be considered necessarily complete -- more could be added in between, above or below. They will suffice to illustrate the transitions essential to understanding diagrammatic reasoning.

The highest level, arbitrarily called Level 1, describes bubble diagrams, network diagrams and the most propositional extremes; the lowest, Level 3, is is the most pictorial, most like the maps and plans in ordinary use. The characteristic attributes of the three levels include:

Level 1: 'A Diagram', (propositional diagram)

-- only aggregate elements; distinctions between classes but not generally individuals; binary distinctions (such as between one or more than one, foreground/background, etc.); no relevant scale or constant transformation; differences in magnitude; connection and separation; pure shapes regular-sided polygons and circles.

Level 2: 'B Diagram', (pictorial or 'plan diagram')

-- more disaggregated objects; distinctions between classes and individuals of elements; variable scale and transformation; numbers up to six; shades of grey; differences in orders of magnitude; sense and polarity; variants of pure shapes.

Level 3: ' Projections, Maps and Plans'

-- distinction between all atomic elements; relatively precise 3-d locations and sizes; constant scale or transformation; arbitrarily complex shapes; arbitrary numbers of elements; large variations among shades of grey and color, etc.

Since all graphics embody some form and amount of abstraction, the last level might properly include all graphics, based on these loose definitions. To further distinguish diagrams requires a consideration not just of graphics features, but the role of the graphics in a reasoning process. The consideration of that role will reinforce the categories described here, showing that A- and B-diagrams are not just so classified on the basis of their appearance (structure), but also on the basis of their origin and purpose (function).

In the next chapter, I look in detail at some diagrams of urban form, in order to extract some particulars, and abstract some propositions, to form the basis of a lexicon of terms.

Chapter 2. Diagrams of Urban Form

In order to more fully describe the characteristics of diagrams, I examine some representative examples chosen from the domain of urban design, that lie somewhere in the middle of the range from more pictorial to more diagrammatic. The following analysis is intended to reveal structural similarities among the diagrams and to collect a lexicon of form-concepts from this domain.

Reston

Figure 2.1 is another diagram of a new town, from the same study of urban form as the previous map and diagram (figures 1.3 and 1.4.) Although the previous diagram, Columbia, and the present one, Reston, are similar in some ways -- an urban center located on a main axis surrounded by residential areas connected by loop roads -- a number of differences are visible between the diagrams.

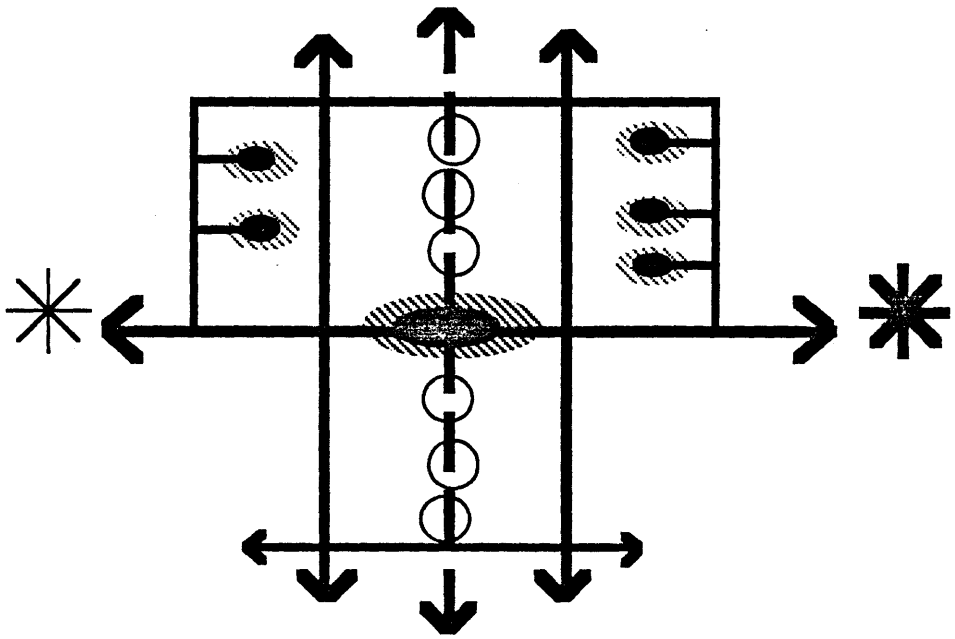


Figure 2.1 Reston New Town

Both show a settlement located on an axis between two urban 'magnets'; in Columbia, these external magnets are the same size and weight, in Reston one is noticeably smaller. That has no effect on the

structure of Reston itself, which is diagrammatically symmetrical around the axis perpendicular to the axis connecting the magnets. In the case of Columbia, the loop road surrounds both sides of the main axis, "inter-urban components" (circles) are located all outside of the loop, on one side of the city, and the inner residential settlements (hatched areas) are connected by pedestrian transportation (dashed lines). In Reston's case, the loop is all on one side of the main axis, inter-urban components are located inside of the loop along an axis, separated from residential areas by other main roads, and there is no pedestrian transportation system connecting the residential areas.

The elements of Reston are recognizable from an early planning report for the city [Crane] which describes the "principal organizational concepts" of the proposed city center:

> a. A north-south pedestrian spine which links a series of activity nodes.
> b. A major open space system which extends thorough the entire project.
> c. An internal transit system...operating on the pedestrian spine and connecting to future METRO rail.
> d. A series of multi-use activity nodes... which serve as the main activity focal points...
> p.4

In Reston, the 'spine' was the central organizing concept. In Columbia, by contrast, the idea of the developer, Rouse, was of a more generic 'center':

> "I ...visualize a series of small communities separated by topography, highways, public institutions, or greenbelts and united by a center that provided cultural, educational and recreational facilities for the many (say, 10 or 20) small towns around it."
> [Brooks 1974] p. 151

The concentric layout of Columbia derives from this vision, which in turn certainly derives in part from the original 'Garden Cities' of England, which were diagrammed as segments of different uses connected by concentric ring roads and 'greenbelts' [Howard].

"Spine", "nodes", "system", "connecting", "focal points", "separated", "center", "around" -- are some of the words used to describe the concepts in the designs being imagined, and diagrammed.

Ideal Communist City

Fig 2.2 is not any particular city, but a diagram of an idealized urban layout [Gutnov, Baburov et al.]. There is no map to contrast it with, but the diagrammatic elements can be clearly seen:

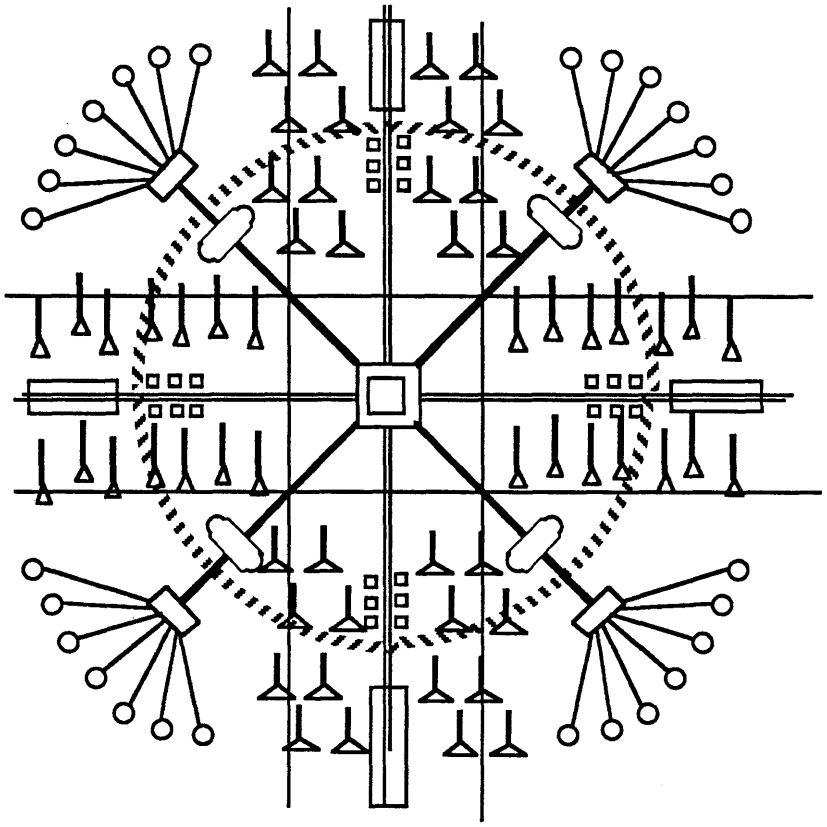


Figure 2.2 - Ideal Communist City (after Baburov, et al.)

Like Columbia and Reston, this diagram is organized by a 'center'. Three points from the text accompanying this diagrams are:

1. A pedestrian scale. The distance from any residence to the center of services including public transportation will not exceed 500 yards to 600 yards (seven minute walk.)
2. The school environment. Placing the school community at the edge of a residential area allows the school grounds to connect directly with park and forest lands.
3. Parks and recreational land. Spaces between the residential sectors will be green, linking the trees and landscaping of the center to forests and parks outside the developed area.

These points illustrate the 'high-level' concerns (short walks, access to nature) turned into formal and metric criteria in the diagram, and the essentially topological terminology in some of the arguments ("edge" "connect" "linking" "center" "outside")

Design for an Indian Village

A rather famous diagram in the literature is reproduced below. This diagram represents a schematic design for a village in India, the result of a systematic method considering the interrelated criteria of agriculture, religion, and domestic customs, and grouping related and isolating unrelated criteria[Alexander].

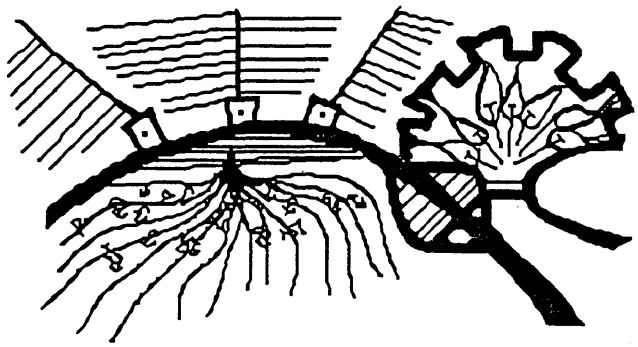


Figure 2.3 Design for an Indian Village (after Alexander)

The diagram is the result of combining a number of smaller diagrams, each of which responds to a particular set of criteria, and are put together according to principles laid out in the accompanying text. These principles of combination, like those used by Alexander and Manheim in the maps described in the previous chapter, are not as explicit as the accompanying analysis of design criteria. The diagram above is presented as a final design, with the claim that the details of a physical plan are completely contained in the diagram -- that it is, in fact, a plan. What is not explained is why the graphic features of the diagram are as they are; neither literally plan-like (although some elements are, such as the three 'antenna-like' structures, representing systems of parallel irrigation lines branching from a central feeder and a well, symbolized by a square with a dot in the middle); nor neutral geometric figures like those in the Communist City above.

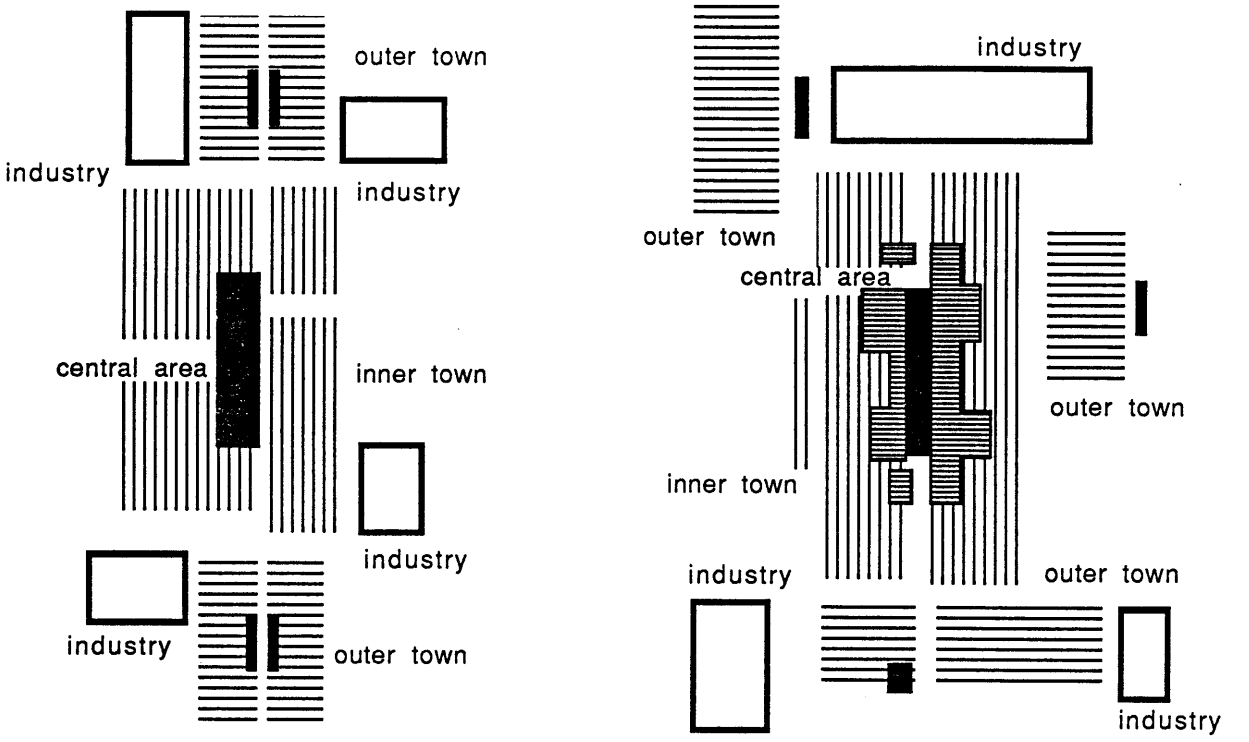
This diagram, according to the original text, follows from a verbal description of criteria, in a systematic way, but the criteria, (not reproduced here) are descriptive of the elements themselves ("cattle", "well", "fields", "doors", and so on) and the relationships between them are reduced to the simplest "interdependent" or "independent", and the dependency is not further elaborated verbally. Rather, the interdependencies are synthesized in the creation and combination of the diagrams -- once again, the system and the method of synthesis are left unstated. Consequently, the diagram shown above has graphic features that are seemingly explainable only as personal decisions by the diagrammer about what 'looks good'. (I'll return to this diagram in chapter seven, in a discussion of the inferential purpose of graphic representations.)

Hook New Town

Fig 2.4 is a pair of diagrams, taken from an urban design project -- the planning for a new town outside of London, undertaken by the London Town Council, for Hook, a new town of 100,000 people. These diagrams tell a story about urban design concepts, although the town was never actually built.

The Hook New Town study explicitly proceeded through a series of diagrams in the development of the city form. The first concept was based on a set of beliefs about the basic elements of a town (commercial area, residential areas, and industrial areas), and some convictions about proximity and layout (central commerce, peripheral industry, and in-between residential). These elements and relations were represented diagrammatically as shown in figure 2.4.

> "The initial concept [*first town diagram*] was recast into the form of a diagram which could provide the main framework for the master plan of the town. This is shown in [*second town diagram*] ." (Hook New Town study, London City Council p. 27).



First town diagram, showing form and disposition of main town functions

Second town diagram, showing initial idea as amended after consideration of local factors

Figure 2.4 Hook New Town Layout

Diagrammatic Inference

The above discussion has concentrated on the structure, or elements, of the diagrams. But what of their function? Two examples from the same literature are informative.

One simple diagram (figure 2.5) is found in the Hook New Town study, illustrating a decision about concentric development around a linear, rather than point, center.

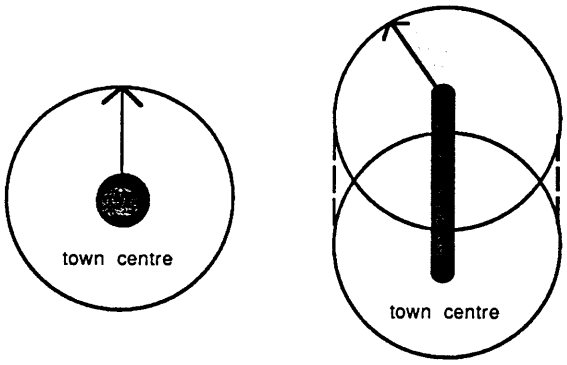


Fig. 2.5 Hook New Town Study Logic

The reasoning accompanying this diagram shows clearly the role of concepts in determining form:

> "The central area needs a form which will allow it to be accessible to the maximum number of pedestrians and vehicles and to which uses (such as central housing) augmenting its own attractiveness, can be easily attached. At the same time, its character and quality as the town's main regular gathering place and social centre will largely depend on its compactness and intimacy. For it to be efficient as a shopping centre (as well as for the reasons already mentioned) the shopping should be continuous, A linear form has therefore been adopted for the central area as best meeting these needs. A greater population can surround a linear central area within a given distance than is possible with a radial concept. Its overall length has been determined by the need to maintain shopping continuity. "
>
> Hook New Town study, London City Council p. 29

In this case, decisions about a topology and beginnings of a shape are based on concerns such as "accessibility" "attachability" "compactness" "intimacy" "efficiency" and "continuity"; these lead to a decision of "linearity" as well as overall length. Again, both formal and metric consequences derive from high-level concerns.

A second example of overt diagrammatic inference comes from the text describing the Ideal Communist City shown above (figure 2.2) The section of text entitled "Planning Residential Areas" starts by describing daily household needs and the provision of services, including schools, transportation and shopping and varying levels (local residential, city-wide, e.g.) A comparison is made between traditional English and Soviet practice, and then a proposal for a "new functional residential structure" is made, with the accompanying diagram (fig 2.6)

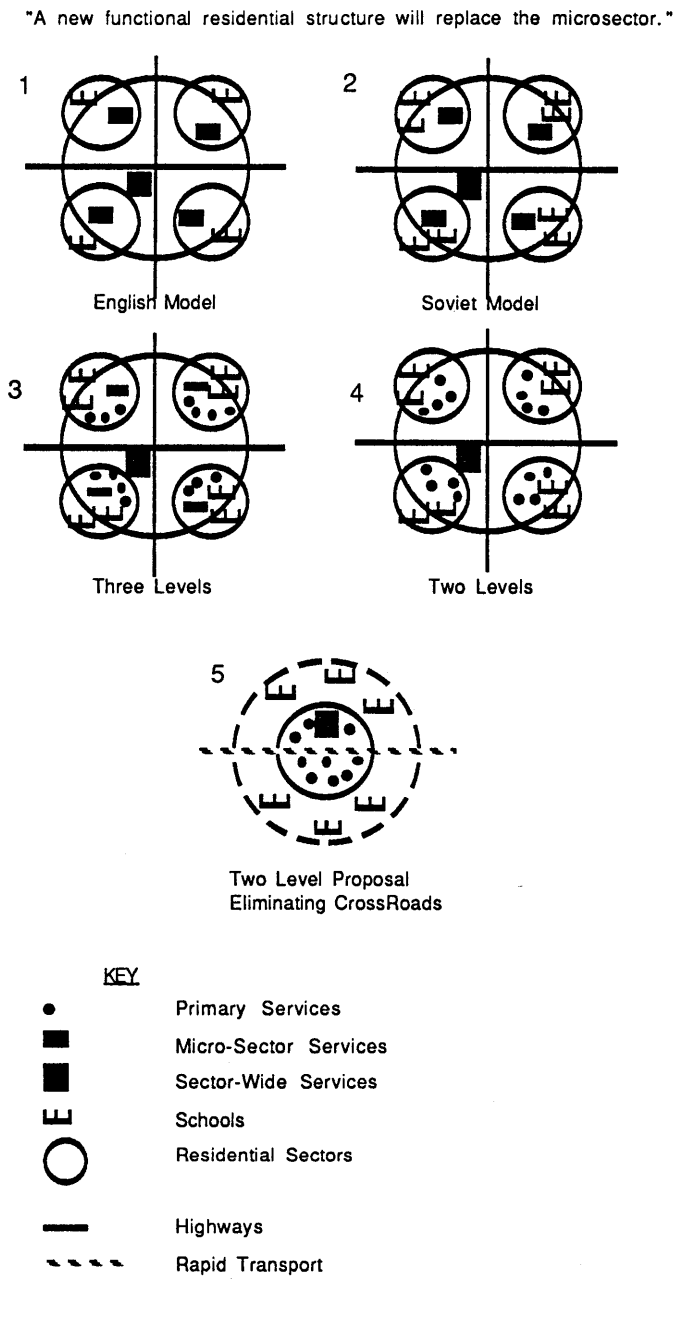


Fig 2.6 (after Gutnov, Baburov, et al.)

The argument accompanying these diagrams goes something like this: Traditionally, we provide local services within residential "microsectors", and more concentrated services for the sector at large. Both English and Soviet practice is to put schools at the outside edge of the sector, since

they require large areas of open space for playfields; in the typical Soviet town the density of housing is greater and so more schools are required. (This is the difference between ("sub-diagrams" 1 and 2 in the figure). But in this high density microsector, centralized local services are insufficient and not equally accessible to all; a better approach is to provide three levels of service, widely dispersed primary services, secondary services within each microsector, and central 'tertiary' services for the town (shown in sub-diagram 3.) But this approach leads to useless duplication of some micro-sector-wide services, and "too thin" (i.e.inefficient) distribution of services. So a two-level system is proposed, in which distributed local services provide the primary and secondary needs, and a central service center still provides for city-wide needs (sub-diagram 4.) Finally, this arrangement suffers from the serious disadvantage that "the system demands that residents cross large streets when they go to the shopping center". (p. 74-79). The solution then is to eliminate large streets crossing in the center, and replace them with rapid transport above or below grade that doesn't conflict with pedestrian flow. Now there is a dense shopping core, surrounded by residences and local services, with schools still at the periphery (sub-diagram 5.)

This last diagram is the central scheme for a proposal that will be developed more fully in the text(including arrangements of open space and parks, for example). What is important to note is the role of visual inference in supporting the argument. The critical inference in the argument is illlustrated in the transition from sub-diagram 4 to sub-diagram 5. In 4 it is obvious by inspection that to get to the central services from three of the four sectors requires crossing the highway, regardless of which sector the service center is located in. This can be demonstrated geometrically by drawing a line from anywhere in these sectors to the shopping center -- the line intersects the highway in at least one place. This intersection is unwanted -- the design goal then is to eliminate it or its cause. Of several possible approaches, one is to eliminate the highways altogether, which is the approach taken by the authors. (Others might include providing bridges and tunnels for pedestrians, e.g.).

A final example of diagrammatic inference comes from Lynch's Good City Form, in which one approach to describing cities is as " a story". In this view, the patterns of settlement tell a story of development over time. Without any legend or other reference, the little diagrams in figure 2.7 below accompany the text (rearranged here horizontally, and numbered).

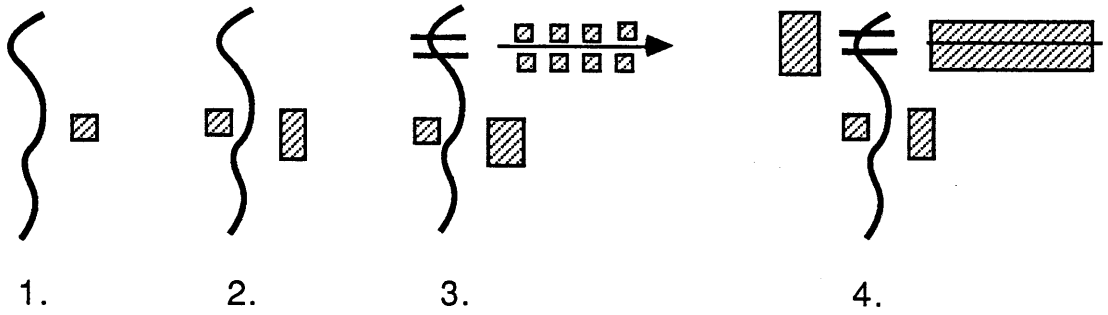


Figure 2.7 "The City as a Story" after Lynch, p. 328

The story is plain to read, for anyone familiar with the impact of transportation on urban history and form. Sub-diagram 1 shows a settlement by a river. In 2, the settlement grows, and another one develops on the other side of the river, unconnected. In 3, the original settlement has grown even more, but an important development is taking place up-river (or down-river, it doesn't matter.) A bridge is built across the river, and a road passes over the bridge. New developments appear along the path of the new road, rather than along the river as before. In 4, the original settlement has shrunk -- no longer growing, but in decline. The second settlement along the river hasn't grown at all; in the beginning, growth was concentrated at the first original settlement; since 3 neither settlement along the river has grown. The developments along the road have expanded and consolidated, and a new growth on both sides of the river is concentrated along the road, in a linear fashion.

This story is about transportation, communication, the impact of technology and change, the form of settlement patterns. All told without words, and with the simplest possible elements: a squiggly

line, two short parallel lines, an arrow, rectangles of various sizes. As before, these might be ambiguous without some sense of domain, but with the title is all that's needed. The word "city" provides us with concepts like settlement, growth, transportation, that we can try to attach to symbols in the diagrams; the word "Story" suggests a linear sequence and a flow of time, so that we know that the four figures are related sequentially (and are not just four alternatives, for example) and that one axis (in this case the horizontal one) represents time. The use of consistent graphic technique indicates family members: all rectangles, similarly shaded, represent settlements -- the critical information is in their position relative to one another or some linear element, and their relative size; linear elements represent transportation systems -- river and road. In the first case, hollow ovals would work just as well as shaded rectangles -- it's relative size and compactness versus linearity that are the constitutive attributes, shape and shading are contingent in this story.

Common cultural knowledge of history helps to decipher the story. Knowing that transportation once moved primarily along rivers, and only later along roads that were able to bridge the rivers, and that settlement often happened along transportation routes, and that roads are generally straighter than rivers, for example, helps. But even this knowledge is not necessary -- one might plausibly infer these things from reading the diagram, forming hypotheses, and then making associations. One such association might be "rectangles appear and grow faster along straight lines than along squiggly lines", or even "the growth of rectangles along straight lines inhibits the growth along squiggly lines". These are inferences that follow from an attempt to make sense of the images in a propositional form. Making the analogy to cities, roads and rivers then needs only the circumstance to trigger the associations. (Perhaps this could have been a microbiological story about the formation of tumors along tissues, and the straight versus squiggly distinction really stands for two kinds of tissue, rather than literally straight and squiggly things!) Whether or not such a story is read as a general principle, or an isolated event, depends on the circumstance in which the diagrammatic story is encountered.

These three fragments of diagrammatic reasoning illustrate several basic features: some characteristic that is apparent in the diagram is related to some characteristic in some other world (the design being considered, the real world events being portrayed), by a process of analogy. In designing, if this characteristic is desirable, then an analogous instantiation may be made in the design; if undesirable, some resolution is similarly made. For this purpose, the relationship in question must be revealed (apparent) in the diagram, and the diagrammatic medium (the 2-d plane) must support some reasoning about the situation. Many features from the original design may be left out of the diagram, for the purpose of emphasis, and the chosen graphic representation must be appropriate to the reasoning being undertaken.

All of the examples above were didactic in original intent -- they were made to prove or illustrate a point. Often in designing the point is not known beforehand; the diagram is produced to see if any points are proved or provable, or any consequences are visible that might alter or expand upon the design decisions in question. Sometimes the linkage of cause and effect is not so clear; the diagram is used as a stimulus to loose association, as a device to draw in relevant information from past or different experience. The structure of the diagram, and the requirements for its construction, in any case remain the same .

Design Diagrams

A short example will illustrate the use of a diagram in designing. This scenario illustrates a progression from diagram to schematic plan in the design of a hypothetical urban plaza.

The first diagram (figure 2.8a) describes an initial situation: a plaza, framed on two sides by buildings whose facades provide important axes, is surrounded by streets on three sides, and has a diagonal pedestrian path running through it. These elements, and their principal relations are the constitutive structure of the design exploration. The table following the diagrams is a symbolic (propositional) representation of the initial diagram.

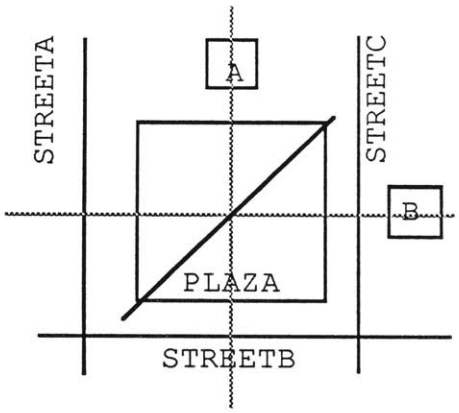


Figure 2.8a

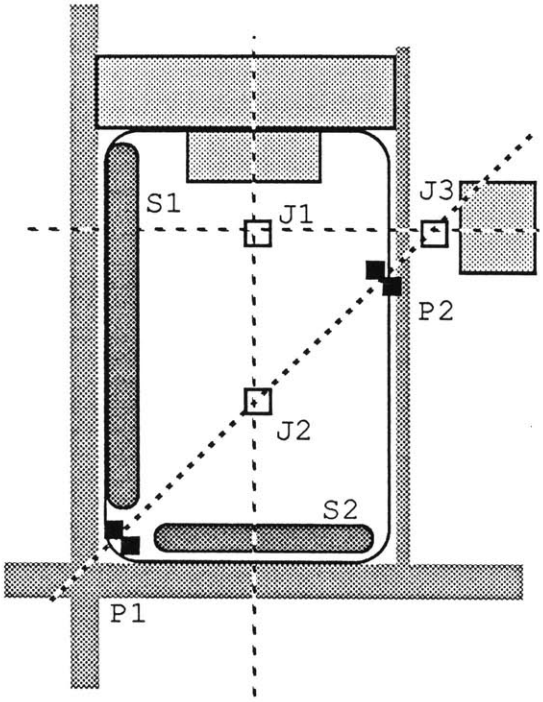


Figure 2.8b

ELEMENTS:

PLAZA, BLDGA, AXISA, BLDGB, AXISB, STREETA, STREETB, STREETC, PATHP

RELATIONS:

STREETA WEST-OF PLAZA
STREETB SOUTH-OF PLAZA
STREETC EAST-OF PLAZA
BLDGA NORTH-OF PLAZA
AXISA (N-S-AXIS BLDGA)
AXISB (E-W-AXIS BLDGB)
BLDGB EAST-OF STREETC
PATHP FROM (SW-CORNER PLAZA) (NE-CORNER PLAZA)

The second diagram (figure 2.8b) is a base map constructed on the previous diagram, and used for subsequent graphic refinement. The topological relations found in the initial diagram are still in force in the more detailed base map, but new (contingent) details -- size, shape, tone -- have been added. They may represent a particular site, or may be fictional[1]. In addition, a number of design "rules" have been applied to the base map, to create new elements and develop the design.

[1]In fact this scenario is based (very loosely) on the redesign of Copley Square, and the design proposal of John Whiteman and Michael Van Valkenburgh as reconstructed by Bill Porter.

Three rules are illustrated in this example:

1. "THE INTERSECTION OF A PATH AND A DISTRICT IS A PORTAL"
2. "THE INTERSECTION OF TWO PATHS IS A JUNCTION"
3. "A SCREEN IS REQUIRED BETWEEN A MAJOR PATH AND OPEN SPACE"

Rule 1 results in two portals (P1 P2), Rule 2 results in three junctions (J1 J2 J3), and Rule 3 results in two screens (S1 S2); they are now represented as symbols on the base map. Figure 2.10a shows the diagram after some graphical adjustment of symbols. S1 has been enlarged, S1 and S2 have both been moved. J1 has become a circle, and the designer has identified it as a FOCUS element. J2 has become a large rounded square, declared OPEN-SPACE, P1 a PORTAL rectangle. J3 and P2 have been grouped together into one element and collapsed, into a single PORTAL rectangle.

All these elements are still diagrammatic, represented by symbols, but the symbols have shape, position and size that serve as the backbone for the next level.

Finally, the symbols can be instantiated by physical elements (figure 2.9b). S1 and S2 both have become rows of trees. Some other rows of trees are added at this level, along STREETC; at the next higher level of abstraction, they too would be represented as SCREEN elements, since they were made by duplicating S2.

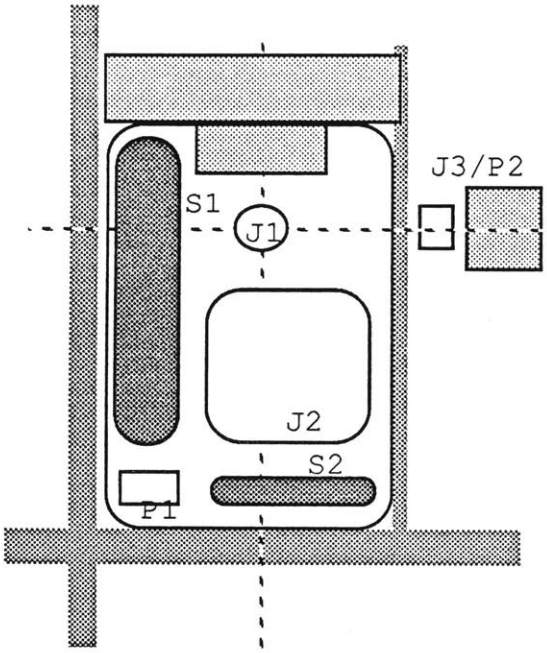


Figure 2.9a

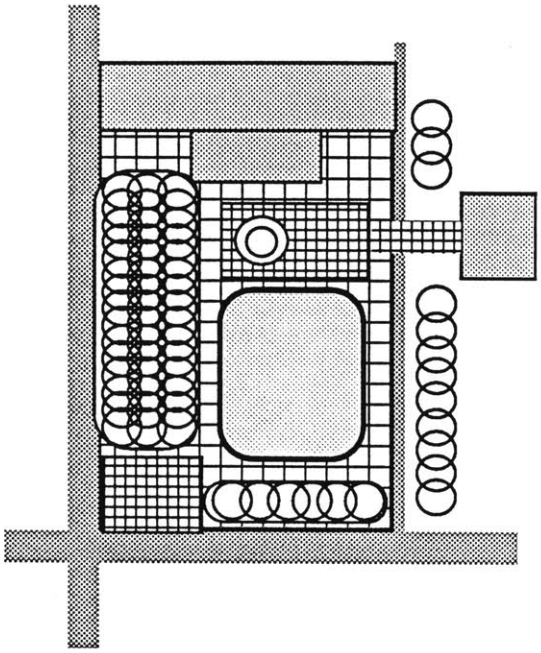


Figure 2.9b

J1 has become a circular fountain, combined together with J3/P2 as a higher level object describing the entry area previously represented by P2. J2 is an OPEN-SPACE area (district) within the plaza -- on closer inspection it is green lawn, surrounded by a low wall (edge), with other attributes. The paving pattern in the entire plaza is differentiated at the two entry areas, for example by a rule that specifies that the paving unit in entry areas is smaller than in the the plaza as a whole.

This short scenario has illustrated a progression from a diagram representing a high level of abstraction, in which all paths are the same size and objects are represented by simple icons (squares), to a very detailed level in which paving pattern and materials are specified. This progression is one view of designing, and illustrates a central function of diagrams, as reservoirs of design decisions, and catalysts to design development. Each sequential diagram above both captures a state of design development and suggests new development.

Structure

The structure common to all these diagrams can be summarized as follows: In each case, a small set of 'elements' are represented in relation to one another. The elements are typically abstract aggregates -- 'major road', 'loop road', 'city center', 'shopping center', 'catchment area'. The relations between them are sometimes represented by explicit graphic elements: lines, arrows, and other simple symbols; often the relations are suggested by graphic/geometric/topological features implicit in the figure: proximity, separation, sameness, relative size, e.g. I will call graphic parts of the figure 'diagrammatic elements' - lines, arrows, shaded areas, blobs, dots, etc, as distinct from the elements of urban design that are represented -- roads, city centers, etc -- that are design elements.

Some diagrammatic elements are found in all our examples: thick and thin lines, light and dark areas, for example, are inherent in the medium -- they are the medium. Their meanings, while context dependent, are literal: lines denote linear elements, different shades and colors express differences in some attribute, and so on. Other diagrammatic elements -- circles, triangles, asterisks, e.g. -- are found in only some of the examples. Their meaning is always arbitrary, though often conventional. Stars denote something special, circles something self-contained, and so on. A key is essential to making detailed sense of these diagrams, although a sense of the meaning may be conveyed without the key, through the more basic implicit elements and relationships.

Form-Concepts

In attempting to extract a lexicon of terms for expressing form-concepts, the question arises: "At what level of generality/specificity should these form concepts be described?" My thesis here is that a generic syntax and generic set of of topological elements and relations can usefully be used to describe the form of diagrams, and to partially describe their meaning (constitutive semantics); domain-specific information is used to extend and refine the meaning to enable domain-sensitive reasoning to occur. Thus, a line intersecting a region is simply a geometric/topological description

with certain logical consequences (the formation of two sub-regions, for example); interpreted as a street dividing a neighborhood it has strong implications for urban design (the creation of isolated sub-neighborhoods, and possible danger of pedestrian/automotive conflicts, for instance.) The former class of consequences (geometric/topological) can be rigorously specified and procedurally determined; the latter (urban design) consequences, being more provisional, may be heuristically determined but are never certain. This kind of logic, central to design reasoning, is contingent and 'non-monotonic': as new information is added (say, new elements in the diagram), the consequences and conclusions from the design point of view may change radically. Geometric/topological properties may change, but in a predictable way; the basic rules will not change as circumstances change, however.

Form-concepts are the semantics of diagrams. As we have seen, one function of diagrams is to give graphical representation to desired design qualities, and to begin to specify design elements and relations. In their diagrammatic state, qualities and properties are closely intermingled; a 'sense of enclosure' is represented by topological closure, e.g. In this state, we cannot say for certain whether the diagrammatic element will result in a final design *element*, or *relation*. A 'center of activity', for example, represented as a circle in a diagram, may become an actual kiosk (element), or it may be the space between several other objects, (a relation). Similarly, a line in a diagram may become an actual line in the pavement, or it may be specification of a linear relation between a number of elements. So in identifying form concepts, the distinction between elements and relations is somewhat artificial. At first blush, such a distinction would seem to be important for a computer implementation; however, we shall see that one approach (the constraint model) while it emphasizes the distinction, makes no a-priori commitment to the categorization of things as either elements or relations.

Design Attributes

Diagrams selectively portray design elements and their attributes; or they are produced at preliminary stages in the design process when only these elements and attributes are known. The totality of attributes that may be assigned during the course of designing is unknown and subject to constant invention, but some generalizations can be made. One such generalization is an ordering, along a continuum from most abstract to least, along the lines discussed above.

Most Abstract/Diagrammatic
- Existence;
- Type (Similarity) ;
- Grouping;
- Number (1-7);
- Relative Magnitude;
- Form (Structure);
- Topology;

More Concrete/Detailed
- Species (Differentiation);
- Shape;
- Number (7+);
- Absolute Magnitude;
- Rotation/Orientation;
- Detail Attributes

The purpose, and meaning, of such an ordering, is to predict the kinds of attributes that a diagram might contain, both the help in the reading and understanding of the diagram, and to help design the data structures for the computer diagrammer. This continuum doesn't necessarily correspond to chronological stages in designing, although it might. The terms above are suggestive categories, that help to distinguish more diagrammatic from less diagrammatic representations and inferences. The details within each category are sensitive to domain, circumstance and personality.

Lexicon

The lexicon of diagrammatic terms should refer first to the more diagrammatic concepts in the ordering above, and be developable toward the more particular. Furthermore, the elements and relations extracted from diagrams should not be arbitrary or merely intuitive -- the list should refer both to the text accompanying the diagrams, and to some theory about elements of urban form. Kevin Lynch, in 1960, provided such a theory, and described a simple set of five concepts that served to describe the mental models of urban residents:

LANDMARK
NODE
PATH
EDGE
DISTRICT

[THE IMAGE OF THE CITY 1960].

This set, empirically derived, seems to be at a good level of generality -- the concepts are simple and general -- and domain-specificity -- each of the terms has specific and evocative reference to urban form. As a basis for computer graphics, they are appealing, as they match with the graphical primitives points, lines, polygons. But they are perhaps too general to serve as the basis for urban design -- they fail to distinguish, for example, between major and minor, and different types of paths, edges and districts, etc.

This five-fold enumeration of urban parts may derive from a more fundamental duality in urban environmental design: the polarity between "public" and "private"; and the corresponding relationships of "connected/exposed" and "contained/protected". This tension is a driving force in spatial design, and designs at all scales attempt to achieve and integrate both "Community" and "Privacy" using the the devices of "containers" and "connectors" [Chermayeff]; many specific design terms can be related to these general goals and concepts.

Elements

A preliminary list of more fine-grained elements, observable in the diagrams presented, is below:

JUNCTION - A KIND OF NODE, OCCURRING WHENEVER TWO PATHS CROSS;
PORTAL - A KIND OF NODE, WHENEVER A PATH CROSSES A DISTRICT;
FOCUS - A KIND OF NODE, INTENDED TO BECOME A LANDMARK;
CLUSTER - A KIND OF NODE, CONSISTING OF SUB-NODES
AXIS - A KIND OF PATH, IMPLIED RATHER THAN NECESSARILY TRAVELLED ON;
SCREEN - A KIND OF EDGE, MEANT TO BE PENETRABLE, UNDER CONTROL;
BRIDGE - A KIND OF PATH, BETWEEN TWO OR MORE ELEMENTS
ISLAND - A KIND OF DISTRICT, SURROUNDED BY A LARGER DISTRICT
SEAM - A KIND OF EDGE, CONNECTING TWO DISTRICTS
BARRIER - A KIND OF EDGE, MEANT TO BE IMPENETRABLE, UNDER CONTROL;
LINK - AN ABSTRACTION INTENDED TO BECOME A PATH, BRIDGE OR PORTAL;
VISTA - A SPECIAL KIND OF AXIS, IMPLYING VISUAL ACCESS.
MAGNET - A SPECIAL KIND OF NODE OR DISTRICT, A SOURCE OR SINK
RING - A SPECIAL KIND OF PATH, CLOSED ON ITSELF
LOOP - A SPECIAL KIND OF PATH, A LOOP OFF ANOTHER PATH
ARTERY - A MAJOR PATH
CAPILLARY - A MINOR PATH
ZONE - A SPECIAL KIND OF DISTRICT, ALONG A PATH
CENTRED DISTRICT - A DISTRICT WITH A DISTINCT CENTER
DIFFUSE DISTRICT - A DISTRICT WITH NO DISTINCT CENTER
CONCENTRIC DISTRICT - A SPECIAL KIND OF CENTRED DISTRICT
INTERIOR - A SPECIAL KIND OF DISTRICT , INSIDE A LOOP
PERIMETER- A SPECIAL KIND OF EDGE, THE OUTSIDE OF A DISTRICT

Note that this list, while not arbitrary, is not and could not be exhaustive or universal. The definition and invention of elements -- things with names and properties, related to other elements, often but not necessarily hierarchically -- is an essential design activity. The list above identifies some of the things found in the diagrams, biased by the author's knowledge in the domain of urban/environmental design.

Relations

Relations and elements are intimately linked in diagrams. A 'center' element implies the 'surrounds' and 'connects' relations between the center and other elements; a 'loop' implies the 'inside' and 'outside' relations, and so on. Indeed, 'inside' and 'outside' may be serve elements in one argument, as in "The inside and the outside of the loop should be linked by bridges" and as relations

in another: "The pond should be entirely inside the loop". Relations also imply constraints on their constituent element; it is nonsense to say that a 'node' surrounds anything, for instance. In identifying relations, we implicitly identify the kinds of elements that enter into them; in the computer program, the identifications will have to be explicit, and default element types specified.

Following is a partial list of the relations found to relate the elements in the examples above:

EXISTENTIAL: EXISTS, IS-A, SAME-AS, DIFFERENT,
FORMAL: AXIAL, CLOSED, OPEN/CURVED, RADIAL,
TOPOLOGICAL: ALONG, AWAY-FROM, BESIDE, BETWEEN, CENTRED-ON, CENTRED-IN, CONNECTS, DIVIDES, INSIDE, INTERSECTS, LINKS, NEAR, OUTSIDE, SEPARATES, SURROUNDS, TOUCHES,
GEOMETRIC: ORTHOGONAL, PARALLEL, TANGENTIAL,
METRIC: LARGER, SMALLER, EQUAL

Clearly there are possible synonyms for many of these relations ('crosses' for 'intersects', 'bigger' for 'larger', etc.), and there are pairs of duals (e.g. (inside a b) == (outside b a)). For many relations, there are shades of meaning, and outright contradictory interpretations (a bridge may 'CONNECT' or 'SEPARATE' -- depending on the situation.) There are many terms which are replaced in the list by more general terms ('BESIDE' covers 'ABOVE', 'BELOW', 'LEFT-OF', etc.) Like the elements above, the list of relations cannot be exhaustive for all these reasons; nor can it be said to be universal or canonical. The intent is rather to have it be expressive and useful in the cases we are considering.

These relations are extracted from the sample diagrams presented here. In more exhaustive experimentation, a psychologist and A.I. researcher [Bongard] has extracted and proposed a list of attributes that are generally distinguishable by human observers. The list of primitive relationships, (ordered after [Montalvo 86b]) , includes:

Primitive Diagrammatic Relationships (after Bongard & Montalvo):

COUNT
EXISTS
SIZE
COLOR
CONCAVE/CONVEX
STRAIGHT/CURVED
NUMBER OF SIDES
ORIENTATION
RIGHT/LEFT
ABOVE/BELOW
NEAR
SQUIGLINESS
MAJOR/MINOR AXES
CONVEX HULL
CLOSED/OPEN
ORDERING
CLOCKWISE/COUNTERCLOCKWISE
ANGLE
NECK/LOBES
SPINE
SYMMETRY
INSIDE/OUTSIDE
CROSSING
X,Y SLOPE
EXTENSION OF LINE SEGMENT
OBJECTS CONNECTED WITH LINE SEGMENTS
DISTANCE BETWEEN OBJECTS
MIN/MAX/AVERAGE
DIRECTION
SHAPE SIMILARITY/DIFFERENCE
BRANCHING
LEVELS/DEPTH
from Montalvo , "DIAGRAM UNDERSTANDING COURSE NOTES" 1986

Some of these relations do appear to be deeply ingrained in the human visual apparatus, and hence universal; evidence suggests that the discrimination of shape, contrast and color are all functions of neurophysiological mechanisms between the eye and the brain [Livingstone]. Others, such as "ordering", or "levels/depth", seem more intentional and knowledge-based, hence more variable. Some of the more universal concepts seem to be important in diagrams, such as inside/outside; others, like left/right, are often irrelevant in diagrams and diagrammatic reasoning. Thus the list of primitives that will be needed for the computer program developed in the next sections will lie

somewhere between universal and domain-specific, including some relations from the Bongard list, and also some empirically derived concepts from the corpus of urban design diagrams, such as axes, barriers, bridges, etc. The most primitive terms (elements and relations), those expressed as simple constraints, are the most universal, coming from algebraic, geometric and topological domains; urban form concepts are packages of these fundamental elements and relations.

Conclusion

The use of atomistic terms and geometrical relationships may seem a stilted if not fatally flawed approach to describing or exploring urban design. [Lynch 1982] proposed a set of five dimensions along which to describe and evaluate urban form: "vitality, sense, fit, access, and control" (to which are added the 'meta-criteria' of "efficiency and justice"). These are highly evocative and metaphorical terms, but each relates to spaces, structures mental and physical, and connections between them, in unspecified (but richly illustrated) ways. These seem like much better touchstones for urban design than topology and algebra -- indeed they are. But whence derives the metric for access or control? These can be measured by a topological network description of spaces and connections [Hiller & Hanson]. "Vitality, sense and fit" are all judgements, not specifications, that derive from physical layout and groupings of elements and activities. Elsewhere, [Lynch and Hack] list spatial organizational strategies for achieving "congenial physical characteristics": "symmetry, order, repetition; continuity and closure; dominance rhythm, common scale or similarity of form". These descriptors are more nearly geometric, though still begging explicit tests and methods for achieving these characteristics. The seemingly reductionist approach revealed in the enumeration of the the list of relations above (and that will follow in a computer program), is not intended to displace or replace these high-level Lynchian criteria or descriptions, only to translate them into explicit (hence computable) meanings.

Summary

Relations in form concepts are more likely to be generic and domain-independent; elements are more often unique and domain-sensitive. Together they constitute a lexicon of form concepts that is suited to the description of diagrams of urban design, and also to the reasoning process in which the diagrams must be involved in designing. The grammar of diagrams ('diagrammar') combines the terms in the lexicon to produces statements in two distinct representations -- one in symbolic propositions and logic, one in graphical display. The interaction between these representations is the point of diagramming in design.

With the above information, and the skeleton of a lexicon, I can begin to formulate the purpose and characteristics of a computer-aided diagrammer. The purpose of such a program is primarily to connect inferential machinery (both human and automated) to graphics in the process of designing. The following chapters will describe the connection, and the designing, and the inferences as well as the graphics. The data structures and procedures that form the basis of the diagrammer will be my focus. First I must formulate a model of designing that supports and depends upon diagrammatic reasoning; the next chapter will describe such a model.

The model and the subsequent algorithm address the question "How come these diagrams?" The computer program once constructed serves as an experimental laboratory in which to explore the related question "How come these diagrams?" That is, I will propose both a general mechanism to describe the origin and purpose of diagrams in general, and then provide explanations for some of the specific features of the individual examples examined.

Chapter 3. Designing with Diagrams

In this chapter I propose a model framework in which diagramming fits into a larger process of designing. I propose to describe the structure of diagrams as a set of elements and relations, and their function as both medium and stimulus for visual inference, and as a skeleton for graphic development.

A Model of Designing

Designing encompasses many activities -- cognitive, communicative, reflective, active *et al.* --- and incorporates information at many scales. Definitions of design and models of designing have been many [Akin, Alexander, Eastman, Simon, Gross, among others], and I'm not proposing any radical new one here. I take the common and uncontroversial position that designing is a process of preparing for the production of an artifact (at whatever scale, in whatever medium, at whatever level of abstraction or aggregation) subject to some set of goals and constraints. For the present discussion, I assume that the production of drawings is included (though not essential) in that preparation, and that those drawings serve two purposes: ultimately to communicate with and direct agents in charge of the actual production of the artifact (builders, machine operators, machines); and in between to enable and support the designer's exploration of alternatives and consequences. In final working drawings, neither goals nor constraints may be apparent -- they will have been transformed into specifications on material properties. In intermediate design drawings, goals and constraints are more likely to be directly represented. In this model of designing, then, goals and constraints must be made explicit, represented graphically, and turned into specifications on material properties.

I'll use the general term 'concepts' for goals and constraints expressed as qualities -- such as privacy, access, community, coziness, monumentality -- to emphasize their essential nature as abstractions. In this discussion I assume that these concepts, to be made operational, are expressed verbally, as 'propositions'. The progression from concepts to specifications (usually in drawings) is designing. Other steps along the way of the progression include, for instance, the determination of appropriate goals, the resolution of conflicts between goals, the projection of future consequences of

the design, and so on. The path from concepts to specifications need not include diagrams, but often does; the final form of the specifications need not be drawings, but often are. I'm interested in the graphic phenomena, so this discussion is about these processes that do use diagrams and produce drawings. I'll use the general term 'drawings' to refer to all the more-pictorial graphics (non-diagrams) that designers produce.

A diagram of the relation between concepts and drawings, with diagrams in between, is shown in figure 3.1. The three nodes -- concepts, diagrams, and drawings -- contain different quantities and kinds of information, in different representations. Between these nodes are six arrows representing transformations on the information and forms of representation. This diagram illustrates a general model of designing, that specifies the locations and kinds of knowledge employed in designing, and the directions (though not the details) of operations on that knowledge.

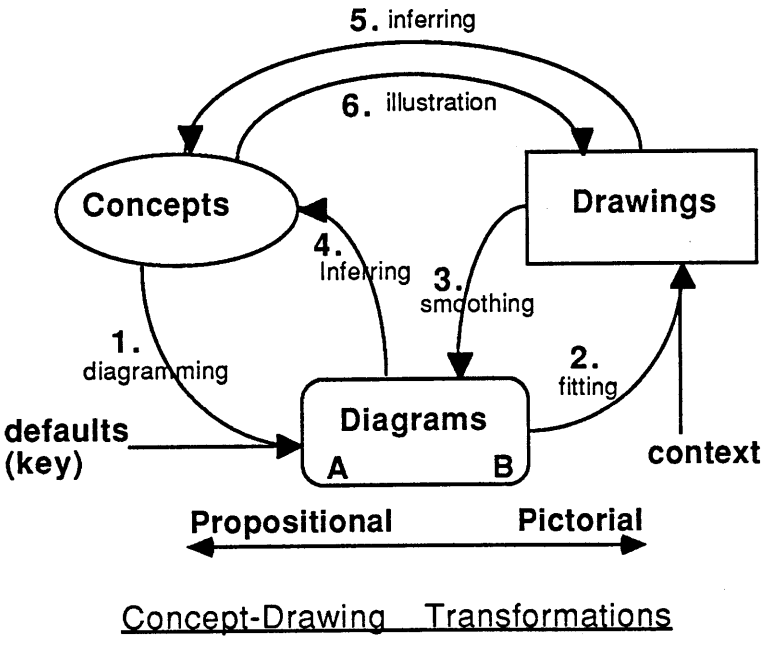


Fig 3.1 An Information Processing Model of Designing

Each of the transformations is a design operation that may be considered an inference, as it leads to new information that follows from previous information, but I only explicitly label those two

paths leading to concepts as "inferring". Operations involving diagrams are labelled 1 through 4 ("diagramming", "fitting", "smoothing", "inferring"). Operations 5 and 6 ("inferring","illustrating") are direct transformations between drawings and concepts that bypass diagrams altogether. Both of these operations use 'compiled' knowledge and are characteristic of expert or experienced designers.

Going from concept to drawing (illustrating) is the process of making preliminary sketches directly from the head, rather than based on some process of design development (which might include diagrams, for example). From the hands of an expert, such sketches are often pointed and poignant -- they provide direction and body to a design process. They may compress or completely pass over many steps of design development. From a novice, such sketches are often irrelevant and confusing, requiring considerable reworking or interpretation before they are made useful (or filed for future reference, or even abandoned altogether). For novices, compressing or passing over a careful design development process is not desirable, and diagrams play an especially important role in making the development process explicit.

"Inferring by looking" or going from graphics to propositions, paths 4 and 5, uses visual abilities that all humans have, though they are more highly developed and nourished in designers. All looking is knowledge-based [Arnheim], and appreciation, description and analysis of visual scenes is not a domain-specific skill, though again it is specially cultivated in designers and design critics and educators. Path 5, going from pictorial drawing to concept, is a skill of such experts. It is the skill of knowing how to look, and what to see, that is inculcated in the traditional process of learning design in the studio, classroom and atelier. Novices often don't know what to make of a drawing, whether sketch or diagram; one purpose of the design presentation and jury is to let novices see expert critics at work in their visual inference, as the posted drawings are mostly all the critics have to go on.

There is no reason to assume that operation 5 ("inferring by looking") is produced by the concatenation of operations 3 and 4 ("smoothing" and "inferring from diagrams") although the source and destination are the same. Inferring from diagrams is a case of the more general inferring by looking -- the differences may be only in the information source, and not in the processes, though it seems possible that the inferences from diagrammatic and more pictorial drawings are different. The experiments in chapter six explore in detail some of the inferences derived from diagrams. Operation 3, smoothing -- abstracting a drawing into a diagram for inferential purposes -- serves to reduce complexity, as much as anything; as observed earlier, this abstraction must itself include some reasoning. It is possible that in some (or many) designers, the operation smoothing is a concatenation of operations 1 and 5 -- that is, first directly inferring by looking, reducing the drawing to a set of concepts, and then diagramming those concepts as if from the propositional side. This concatenation may be so seamless that the result appears to have no intermediate steps or representations; it is part of the purpose of this thesis to indicate what those intermediate steps and processes might be, for a computer program implementation (a diagram reader.)

This model of designing doesn't specify a starting point or any sequence of operations; nor does it require diagrams at all. The decision about which route to take from concepts to drawings -- a question about the necessity and utility of diagrams -- is not resolvable by any theory of diagramming. It may well be possible for designers to live without diagrams, though it might not be comfortable. The source of the comfort may just be the high bandwidth of graphics in general, but there is a power as well as a comfort that accompanies diagrams more than other kinds of graphics. This power, I'll argue, derives from the special place of diagrams between graphics and concepts. (It seems possible that some concepts don't lend themselves to passing through the diagrammatic stage. This possibility will also be explored in more detail below.)

This model of designing recognizes the non-diagrammatic operations (5 and 6), but I won't consider them in any more detail here; this dissertation is about diagrams, and the first 4 operations.

Ultimately, I'll focus on operation 1 (diagramming), but that focus will occur in the context of considering the entire system of diagrammatic operations.

Diagrammatic Transformations

As diagrams stand in between pictorial and propositional poles, there are four directions of transformation to consider:

1. concept ⇒ diagram
2. diagram ⇒ drawing
3. drawing ⇒ diagram
4. diagram ⇒ concept

Each transformation represents a change in quantity and kind of information available in the chosen representations.

The first two transformations, 1. and 2., are processes of specification (particularization), leading away from the abstract. The second two transformations, 3. and 4., are processes of abstraction. They lead from the more particular (drawing or diagram) to the more abstract (diagram or concept). In both kinds of transformations, one of the operations takes place with graphical input and output (2. and 3., drawings and diagrams.) The other transformation (1. and 4., respectively), "breaks the graphical barrier" in one direction or the other, dealing with graphical data on one side of the input or output, and the concepts related to the drawing or diagram on the other side.

It seems unlikely that 3. is simply the reverse, or the logical inverse, of 2., or 1. of 4. Although the source and destination are reversed the processes are complex enough that no simple understanding of inverse (like looking up in a table in two different ways, e.g.) will suffice to describe the differences and similarities in the processes. Both pairs of processes represent one-

to-many and many-to-one transformations. Abstraction and particularization are flows in different directions, but also of different kinds.

Consider the processes of particularization:

To go from diagram to drawing requires some knowledge of the propositional content of the diagram -- completely graphical transformations will not suffice. 'Completely graphical transformations' are informally those that require no reasoning, only mechanical, graphic operations. These can be understood computationally as follows: Given a data structure holding a graphic image (not the data necessary to produce the graphics), modify the data structure in a systematic way, with reference only to that structure. So for example, tracing the outlines of figures, rotating and scaling, enhancing contrast, snapping all points to an arbitrary grid, changing patterns or densities, are all completely graphical transformations. These are the core of the enterprise of image processing, and can be be used to great advantage to improve the legibility of images and hence assist in the process of inferring by looking. They can change the appearance of a graphical representation, and they may change the meaning, as when an image is shrunken or enlarged beyond recognition. These transformations can not, however, change the meaning in any controlled way, because they have no reference to the propositions that comprise the essential meaning.

To develop a diagram further, into a drawing, requires reference to the meaning -- or semantics -- of the drawing. This development may well be rule-based, or based on pattern-matching operations of a sort, but these rules and pattern-matching principles can not be solely syntactic -- i.e., based on surficial structure only. Developing diagrams into drawings, the activity professionally described as 'design development', constitutes perhaps the majority of the time spent by designers at the drawing board. This process is an intriguing and important one for furthering the art of computer-aided design, and my contribution to that understanding will come not from addressing this process in detail, but rather specifying the structure of diagrams

that can guide design development, and describing an example of constrained graphic editing, an approach to design development and production of drawings that follows from computer-aided diagramming.

The other process of particularization, from concept to diagram, requires a commitment to a particular graphical system, or mapping. To construct a drawing to illustrate or represent a concept requires a major leap -- "crossing the graphical barrier". To make this leap a mapping between elements and relations in the concept to graphical elements and relations is required. Choosing one graphical system does not necessarily determine further graphical development, but may direct it. For example, choosing a rectangular or a circular icon to stand for a diagrammatic element is only syntactic (related to appearance), and has has no effect on the semantics (meaning) of the diagram. However, choosing a radial or a grid distribution system of elements, represented diagramatically by a circle or a rectangle, will produce different diagrams with different implications for further development.) In the diagram above I call a diagram generated from a concept, or proposition set, an "A Diagram" (*alpha* diagram).

Consider the processes of abstraction:

To go from drawing to diagram also requires a graphical key, and it requires some proposition set that is to be diagrammed -- there are many possible diagrams of any system, or drawing. The proposition set may be characterized as a filter, that serves to separate out relevant and irrelevant information in the process of simplifying the drawing. The role of the graphical key is to provide rules for the construction of the diagram. One way to describe this transformation is as a kind of pattern matching replacement rule: replace all squiggly lines by straight lines, round all bend angles to the nearest 45 degrees, make all lines equally wide, and so on (this set of rules might be used to generate a diagram of subway lines, for example.) These completely graphical rules will produce pictorial diagrams from drawings; I call a pictorial diagram constructed from a drawing, or intended to be treated as a drawing, a "B Diagram" (*beta* diagram).

To go from diagram to concept(s) or proposition(s) is inferring by looking, and requires a context in which the inference is being made. Without a context, some global, obvious inferences can be made from most diagrams ("There are three different kinds of things, two connected and one isolated...", e.g.). The properties that are context-free are the ones that make up a visual, geometric, topological dictionary of diagrammatic terms and relationships. In a particular design context, each of these terms and relations becomes connected to a larger set of propositions and properties that allow inferences to proceed -- this is the domain knowledge that supports inferring by looking and design reasoning in general. ("From an urban design point of view, an isolated community is not desirable, so one design goal will be to build linkages to connect the isolated one to the other two." Or, a possible alternative: "For the purposes of our sociological study, we want an isolated community, so choose that one.") Inferring from a diagram is different than inferring from a painting or a map. In a map, relations between elements are inferred from explicit information about position. In a diagram, inferences about position are made from explicit information about relations. In a painting or a sketch, the process of looking is more appropriately called 'appreciation' than 'inference'.

Diagrammatic Procedures

These four processes of transformation involving diagrams can be described as procedures, with the following arguments:

	Procedure Name	Arguments	
1.	Make-A-Diagram	(PropositionSet, Key)	[proposition ⇒ diagram]
2.	Make-B-Diagram	(Picture, Filter)	[picture ⇒ diagram]
3.	Make-Picture	(Diagram, Rules/Map)	[diagram ⇒ picture]
4.	Make-Propositions	(Diagram, Key)	[diagram ⇒ proposition]

In the arguments to these procedures, a PropositionSet is a formal, textual expression of a form-concept or other design goal -- the semantics of the diagram to be produced. A Key is a set of

associations, that describe graphical representations for elements and relations -- the surficial appearance of the diagram. Keys may have global or general components, and diagram- or problem- specific components.

The remainder of this discussion will be confined to an exploration and examination of the first procedure [Make-A-Diagram], and the question: "What knowledge, in what form, is required to produce a useful diagram -- that is, a spatial representation of a spatial arrangement that can enter into a designer's reasoning?" The knowledge will take the form of propositions, and rules for transformation. The criterion of 'usefulness' will rest directly upon two considerations: Is a diagram so produced visually, intuitively correct, and is the knowledge embedded in it of the right form for the purpose of inference for which the diagram is drawn -- usually either evaluation of a design, or exploration of design alternatives. Thus, for example, a stock illustration simply chosen out of a list for display, no matter how visually apt, has no embedded knowledge, so it will not serve; likewise, an arbitrarily complex knowledge base or description without an accompanying visual illustration is not a diagram, and will not serve.

Most diagramming is done intuitively and knowledge of diagram procedures is empirical, and most design education treats the subject implicitly, but some writers have explicitly dealt with the subject. Consider the following prescription for the construction of diagrams:

> "The basic process for building a diagram is:
> 1. Try to illustrate the basic identities and their relationships in a rough diagram.
> 2. Reduce the diagram to its simplest structure by applying rules of graphic grammar.
> 3. Modify the diagram to indicate a second level of information, using tones or heavy lines.
> 4. Add other levels of information as tag-ons to the basic diagram
> 5. If the diagram becomes too complicated break it into segments by grouping or placing a boundary around entities.
>
> Laseau p.53

This procedure for diagramming requires knowledge about: *basic identities, relationships, rough diagrams, rules of graphic grammar, levels of information,* and rules for identifying *too complicated diagrams,* and *segments and boundaries.*

"Basic identities" in this recipe refer to the principal elements of the system being diagrammed -- propositions of existence. Simply naming, or identifying, the elements is the necessary first step in the construction of a diagram. If no further relationships are expressed, still some information has been established, and some rule set maybe invoked to infer relations. In diagrams, even these propositions of existence result in some graphical display, and therefore of some implied relations, which may further (or hinder) inference. For example, the proposition "There are two kinds of ideas", diagrammed as two bubbles on the page, graphically suggests that the rest of the page is maybe considered: Is there space between the two bubbles, and if so, are there ideas between the two kinds? Is there a whole world of discourse represented by the empty page that these two kinds of ideas don't relate to? A different diagram -- perhaps a single line dividing the page into two domains -- would provoke different thoughts.

"Relationships" that are to be illustrated are the second, equally fundamental, part of a diagram. No relationships can exist without some elements, and often it is the relationships that come first ("enclosure" as a goal implies both the enclosing element(s) and the enclosed, e.g.). Some relationships have graphical or geometric analogues that have come to be common in our speech: Two overlapping ideas may be represented as two overlapping circles; a direct logical argument and a circuitous one may be diagrammatically contrasted by a straight line and a wiggly one. Some relationships and qualities are not directly diagrammable (monumentality, say, or coziness); these require a further development toward the pictorial before they can be diagrammed (a diagram of a cozy arrangement of chairs in a large room, for example.) Other relationships, like the ones identified in the lexicon above, are eminently diagrammable.

The recipe above refers to "Rules of Graphic Grammar", which are nowhere stated, but are commonly understood to control the graphic representations of diagrams. These are the implicit rules of syntax that diagrams obey. Some of the rules are: "Make each element distinct, and not overlapping with others unless the overlap relation is intended"; "Make no intersecting lines unless intersection is an intended property"; "Make figures regular and even unless their irregularity is important to the inference"; these are all specific cases of the general diagrammatic rule "Make no marks or suggest no differences that are not relevant to the inference intended." A common variant of this rule is "Equalities and equivalences are less significant than differences" - hence the tendency toward symmetry and regular polygons. These provide a neutral 'background' against which significant relationships -- topological, metric and otherwise -- will stand out.

Some diagrams use arbitrary tokens to stand for elements, in which case one rule for token-choosing is "Use tokens that are invariant with respect to rotation and scale, when possible, and use the index of possible readings of a token to indicate degrees of freedom in the element symbolized."

"Levels of Information" in the above recipe, prescribed as "tag-ons" to the basic diagram, suggest the importance of a hierarchical arrangement of relevant facts and propositions. The basic diagram stands without further elaboration, but further elaboration may be added until the diagrams become too complicated.

The test for "too-complicated diagrams" is also nowhere specified, but follows from the diagrammatic rules of grammar. Some rules of thumb: A diagram is too complicated if: " the eye must move to take it all in"; "the number of kinds of elements, or instances of any one kind of element, exceeds seven or so"; "line segments must intersect to fit them all in, when intersection is not intended"

In the event of a too-complicated diagram, the prescription is to identify segments and boundaries in the diagram. Some of these segments and boundaries arise from the graphical representation; more important ones are determined by the semantics, or underlying proposition set, and levels of abstraction represented in the diagram. For example, any set of subsidiary elements can be replaced by a simpler, aggregated object, as one way to reduce complexity. This is true both for the reasoner, and for the diagrammar.

The above procedure suggests something also about the kind of reasoning diagrams enter into. Identifying basic elements and relationships, keeping interactions simple, is what diagrams are all about. These two operations -- let me call them 'identifying' and 'simplifying' -- reveal the different nature of the two kinds of knowledge (and the two kinds of related reasoning) that enter into the construction of diagrams. The first 'identifying' is dependent primarily on substantive domain knowledge. That is to say. the nature of the argument, or the problem, being diagramed: what are the elements and relationships. This is not to say that the answer is only conventional or universal; idiosyncratic and unusual ones are commonplace in designing, and therefore in diagramming. Designers rely on conventional wisdom as they choose and require, but may also reject and/or elaborate it. So an urban designer may choose between the 'standard' alternatives of grid or star plan, but having decided on star may then imagine a comet, with nucleus and streaming tail, rather than a radially symmetric multi-pointed figure. This identification of elements (nucleus and tail) engenders a different diagram, and different reasoning (imagining a form/process that is dynamic rather than static, for example.)

The second operation 'simplifying' is primarily dependent on graphic and general categorical knowledge, rather than substantive domain knowledge. The simplifying is in the first place conceptual -- through the device of limiting numbers, applying hierarchical, linear or other ordering, making generalizations and abstractions, etc. Then the simplifying is graphical -- making a simple picture is partially, but not entirely, dependent on having a simple conceptual

structure. A simple picture of a complicated idea and a complicated picture of a simple idea are both possible. So this simplifying requires a knowledge of the medium, and of its possibilities, limitations and conventions. For example, diagramming is primarily a two-dimensional, often black-and-white (monochromatic) enterprise. This means that the variables available to the diagrammer are limited: size, shape, position, alignment, tone/density, pattern are the principal ones. In a color medium, chromatic variations (along a scale such as hue, saturation, value) are also available; in a dynamic medium variations in kinetic behavior (rate of flickering, rate of movement, e.g.) are available; in a three-dimensional medium an additional axis of location is available; and so on.

So one problem of simplifying is limiting the number of relationships and distinctions that are to be portrayed to a number that can be distinguished within the medium.

A second problem of simplification is limiting interactions within the medium. A common version of this in two-dimensional diagrams is preventing overlap and intersection. Since these features are easily distinguished by eye, they typically must be restricted in diagrams to intentional -- or consequential -- occurrences, rather than accidental. Untangling crossed lines, whose intersection is not consequential in the domain, is nonetheless visually significant as it simplifies the diagram. In the event the intersection is consequential in the domain -- as in wiring diagrams for electronic circuitry, or in multi-modal transportation systems, for example -- then the untangling is necessary on substantive as well as diagrammatic grounds, and solving the case for the diagram may indicate a framework for the solution in detail. In this latter case, the diagram is a more direct analogue of the problem (a "B diagram"), and the diagrammatic form may more closely resemble the final form.

Since part of simplifying is removing information, there may often be ambiguity about what is substantive and what is not in a particular diagram. This kind of ambiguity is often helpful in designing, as it leaves room for multiple possible interpretations and consideration of alternative

solutions. As a result of simplifying, and thereby omitting some information, the necessity of making "default" decisions may arise in the creation, and interpretation, of the diagram.

Elements

Diagrammatic elements are mutable, rather than absolute, with a changing relationship to design elements (that in the end will be fixed, absolute). The relationship between diagrammatic and design elements maybe literal, may include aggregation and abstraction, or may be null (there is no design element corresponding to a diagrammatic element). A "loop" in a diagram may be developed into a highway (one-to-one) , or a one-way road with an associated bike path and sidewalk (one to many), or may simply implied by the arrangement of other elements, say buildings (one-to-none). Elements may appear as *a-priori* specifications by the designer, or *a-posteriori*, either by the designer's declaration or by invocation of some rule for aggregation or abstraction.

Similarly perception of diagrammatic elements may change: one rectangle may be two L's, or four lines, or an elongated square, etc. To the designer, these multiple interpretations may exist simultaneously, or in sequence. A diagram may have several different descriptions of its elements.

Relations

Diagrams consist of elements with relations between them. I now need to show how a diagram like the ones we have looked at in chapter 2 can be described symbolically using a set of diagrammatic elements, graphic primitives and topological relations (e.g. "several small clusters inside of a major loop road" in Columbia). These relations will, in turn, specify a set of relations on some smaller constituent variables of the diagrammatic elements. For example, the diagram "A link connecting two nodes" explicitly mentions the relation "connects"; the connectivity relation implicitly invokes some "touching" relation, e.g. " Each end-point of the link touches one of the nodes", and the touching relation can be expressed geometrically and graphically in terms of equations on x,y coordinates of the endpoint of the link and the boundaries of the nodes. At the

same time, the "connecting" relation is invoked in the context of a particular design situation, where the detailed implementation of "connects" may vary: a road between town centers, a micro-wave radio link, or a series of uses and activities along a pedestrian path. These can all be diagrammed abstractly as a line between two nodes -- further refinement will render the line wider, or invisible, or full of twists and turns, etc.

Defaults in diagrams

"Default: absence or lack (of something needed). " Webster's Second International Dictionary.

"Defaults" enter into the generation and interpretation of diagrams in a fundamental way. When a diagram is generated from a drawing, for a particular purpose, the process involves applying a 'filter' to a large knowledge base, to extract only the relevant information, or features, which are then organized and represented in the diagram. In the process of visually representing the information, assumptions must be made for all non-relevant information. For example, if the argument of the diagram is not about location, it's important that the locations of the elements of the diagram -- which must have locations, since the diagram is a two-dimensional graphic entity -- be as neutral as possible. Thus several equal objects in a diagram will be shown evenly spaced and equal sized, since otherwise some assumptions about inequalities will arise. Rules for diagram construction in the absence of guiding information from the proposition set are used extensively, to express non-essential information, so that specific decisions can be made about the chosen information. These rules of diagram construction can be overridden for particular purposes, but serve to guide the process in most cases.

These assumptions, and the rules that guide them, are like the "defaults" and "default reasoning" discussed in the literature on commonsense reasoning in A.I. research. In the 'frames' description of reasoning [Minsky] for example, a reasoning system is presumed to possess organized, hierarchically ordered sets of related facts and observations (frames), garnered from experience, and a pattern-matching process, such that some set of facts (or observations) serves to provide access

to a related set that may reasonably be expected in the circumstance (in the original examples, a 'birthday party' leads to the expectation of cake and candles, the visible top half of a chair leads to the expectation of chair legs, e.g.) These expectations -- reasonable inferences based on regularities (though not certainties) garnered from experience-- have been described as defaults, and characterized as expressing the commonsense knowledge of 'usually' ("usually birthday parties entail cake and candles", and so on.) These expectations and regularities form the basis for the class-instance hierarchies (this particular birthday party is an instance of the class of parties in general), and inferences about particular individuals based on knowledge about their class has been called "default-based". There are two essential features to 'default knowledge'; it should provide appropriate and correct inferences in most cases, and it should be able to be overridden, or defeated, by subsequent information, without terminating or invalidating all reasoning.

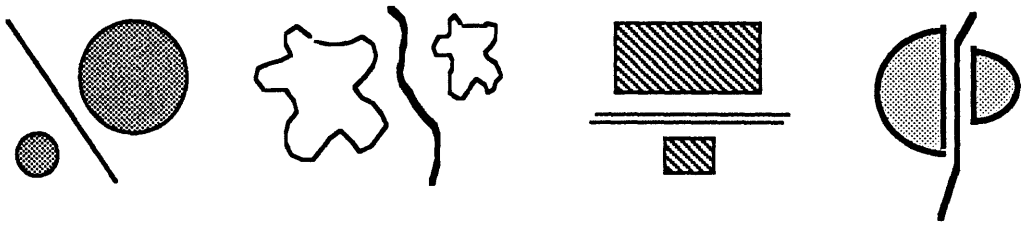
In a logic based inference system, any conclusion reached as a result of a set of propositions must still be valid after the introduction of new propositions -- this is the characteristic of 'monotonicity'. If a new fact ever contradicts or overrides a previous one, the result is said to be inconsistent and no inferences are justified, and no reasoning can proceed. Monotonic reasoning is useful for logical and geometric theorem-proving and reasoning, for example, but not in real-world decision making.

"Nonmonotonic reasoning" [Ginsberg] is more characteristic of commonsense reasoning, and of the reasoning that accompanies designing and diagramming, since it allows for conclusions and inferences that have been reached to be overridden or retracted on the basis of subsequent information. An urban designer might reason, for example, that since all towns have main roads, the new town in question will have one. Then commercial activities and housing areas may be located relative to the road (on it, or connected to it, for example.) If subsequent reasoning (insight, change of heart, instructions from clients, whatever) leads to the proposal that the design should be based on no roads, only pedestrian and rapid transport routes, the presumption about the existence of a main road would be retracted, but the locations of the shopping and residential areas

might remain. These locations, even though originally related to the road, may be seen to have other relations (between each other and to topography, for example) and so are not necessarily retracted when the road is retracted.

In diagramming, decisions are often made without complete information. Subsequent information later in the process of designing can modify those decisions or cause them to be retracted. The default knowledge in diagraming that permits the making of decisions in the absence of complete information, can be categorized as two kinds: domain- or problem-specific default decisions about the design and the elements of the design ('top-level' knowledge); and defaults for graphic representations, which may vary from domain to domain, as graphic keys do, but which generally don't reflect particular domain knowledge. The source of top-level defaults is domain expertise; the source of diagrammatic graphic defaults is in the human visual apparatus, and convention. The example problems and figures in [Bongard 69] provide a foundation for choosing diagrammatic icons and relationships on the basis of their ability to be distinguished by the human observer.

Default decisions are rampant in diagrams, and apply to many different attribute values. "Two similar settlements, one larger than the other, separated by a river" might be simply diagrammed by any of the following:



The input information (proposition list) doesn't specify size (only relative magnitude), shape (only similar kinds), amount of separation, etc. All of the above diagrams express the original statement, but on the basis of different assignments to the unspecified properties (size, shape, e.g..) The consequences of the default decisions made affect not only the graphical quality of the

diagram, but the inferential potential. The criteria for inferential potential are explored more fully in chapters six and seven.

Similarly, the interpretation of diagrams relies on defaults. In the process of omitting some information, to emphasize some other, default assumptions must be made to make up for the lost (omitted) information. These defaults are context dependent, and operate on both explicit and implicit features of the diagram. Thus by default, a wide double line on a map represents a highway, even if it's not labelled as such; on a structural plan it represents a wall. Defaults such as these are often conventional, and derive from literal analogy (linearity of highways and walls).

The diagram interpreter (the human observer, designer) may well be influenced by changes in defaults that affect appearance, so the choice of diagrammatic defaults and decisions about them are not trivial or irrelevant to the designer. Choosing a good diagram is important especially since diagrams are often produced near the beginning of the process, and misinterpretations may have far-reaching implications.

One role of diagrams is to force the appearance and confrontation of issues. Having made one or more design decisions, others are often implied or suggested by the initial decisions. Making a diagram is one way of surfacing these implications, and of revealing potential conflicts or opportunities. The starting decisions may require other default decisions in the diagramming -- inspecting these defaults serves to expand the designer's consideration of issues.

Summary

It's inferential purposes, more than any overt graphic properties, that distinguish diagrams from maps, plans, or sketches. All of these graphics embody abstractions of a sort, and both enable us to reason in one domain (2-d plane) and transfer our reasoning to another domain (a multi-dimensional

real or proposed world). They are diagrams to the extent that we suspend our expectation of literal representation and allow abstractions of many sorts, and to the extent that we not only transfer our reasoning about the graphics into another domain, but also translate the results of the reasoning into different terms. Thus if a sketch "looks good", we expect the sculpture, or house, or plaza, will look good; if a diagram looks good, we have no such expectation about the house, or plaza, or new town. We translate distance on a map into distance on the ground; distance on a diagram need have no such implication. The enabling of not-necessarily-spatial reasoning in a spatial medium is what diagrams especially contribute to designing.

In the light of their inferential purpose, the structure of diagrams can be described as consisting of elements and relations, that stand in some relation to elements and relations in another source state (design, for example). The function of diagrams is at least four-fold, to:

a.) identify elements at various levels of aggregation and abstraction;
b.) reveal relationships " " ";
c.) support inferences about the subject (design) being diagrammed;
d.) provide a graphic skeleton for modification and refinement.

Now I can consider an algorithmic approach to constructing and using diagrams. I'll describe the steps necessary to produce a diagram in a skeletal algorithm, and then present a prototypical program that partially implements the algorithm -- enough to explore its behavior and reach some generalizations and conclusions about its requisite knowledge base.

Chapter 4. Construction of Diagrams

Given the role of diagrams -- to provide fuel for the process of visual inference by identifying and revealing relationships between design elements at various levels of abstraction -- I can now give a detailed account of the necessary steps for the construction of diagrams. This account will rest on a set of simple definitions, and will guide the development of a computer program.

Definitions

A variable is a simple pair: a name associated with a symbolic value, often but not necessarily numeric -- for example (WIDTH = 60) denotes a variable named "WIDTH", with value 60. A design element is a compound structure: like a variable, it has a name, but its value is not atomic, it is a set of variables. (If the set of variables numbers only one, the element is just a variable; a single variable may be an element.) For example: the element STREET might contain the variables: { (WIDTH = 60), (SURFACE = ASPHALT), (CAPACITY = 2000), (SIDEWALKS = NO), (NAME = MAIN) }. A variable in an element may also be called an "attribute", "slot", or "field". In this text[1], an attribute within an element is indicated by a list: (ELEMENT ATTRIBUTE). A relation is an n-ary predicate on one or more elements or variables, written as an S-expression[2], for example: (CROSSES MAIN-ST BUSINESS-DISTRICT), which expresses the relation "Main St. crosses the Business District" or (GREATER-THAN (STREET1 WIDTH) (STREET2 WIDTH)), which says that the value of the variable WIDTH in the object STREET1 must be greater than the value of WIDTH in the object STREET2. A design state, for these purposes, is a simple set of design objects, relationships, and variable values, however partial or complete. A diagram state is just like a design state -- a set of elements, relations and variables --determined by an explicit (but not necessarily one-to-one) translation from some of the elements, variables and relations in a design state. This translation -- a "mapping" -- takes *source* design elements and relations to *destination* diagram elements and relations, or vice versa. This mapping is not the only possible relationship

---

[1] and in the constraint manager CM2 on which the diagrammer is built -- but contrary to usual LISP syntax!

[2] The usual way to read these "prefix" expressions in pseudo-English is to take the first element of the list, and read it like a verb acting on the second and subsequent elements. In these expressions, as in LISP, the first element in the list is a procedure and the subsequent elements are the arguments to the procedure.

between the diagram state and the design state, but it is explicit and must account for each element in the diagram state. Note that the mapping may be only to some of the design terms (more on this filtering below.) A diagram is just a graphic artifact, constructed from a diagram state . A diagram is a composite graphic entity consisting of discrete graphic symbols -- graphic entities composed of points, lines and filled areas, with attributes of position, size, shape, color and lineweight -- such that these sysmbols and their attributes are completely specified by the diagram state. Note that there is a one-to-one relationship between the elements in a design state and the symbols in the resulting diagram. I'll use the term diagram-element, or d-element, to refer either to the symbol or its corresponding data structure.

Then the question "What knowledge, in what form, must be given to a computer program to enable it to produce diagrams?" becomes: "Given a design state as defined above, what knowledge is necessary to produce a diagram state, such that the resulting diagram stands in the desired relation to the design state, and serves the required inferential purposes ?" Taking this question apart, it's clear that at least the following are required: a source design state to be diagrammed; a desired mapping relationship from design state to diagram state; a graphics-producing component that is responsible for producing a graphic artifact from a diagram state; and an end-result viewer and inferrer.

Graphics

In attempting to isolate diagramming knowledge and its structure, its useful to separate out the graphics-producing component, and treat it as being as simple as possible; it should be specified in a small set of operations, and have a limited and completely prescribed repertoire. The operations it will be endowed with for this discussion (and the following program) are basic: draw a point, draw a line, draw a rectangle, a rounded rectangle, or an oval; with in general five possible

attributes on each: position, size, linewidth, color[3] and fill (tone is either black or gray, and fill applies only to rectangles and polygons.) Then any essential diagramming knowledge is pushed away from the drawing component, and must then reside in the mapping relationship between source design parts and destination diagrammatic parts.

The arrangement of symbols is implicit in the list of diagram relationships that are part of the diagram state; it is explicit when all the attributes (position, size, color at least) of the diagrammatic elements have values. That is, the variables and values specifying graphic attributes in a diagram state are necessary and sufficient to produce a diagram; the production of the graphic artifact is a one-to-one traversal of the list of d-elements and conversion of their attributes into the five recognized drawing commands. It is useful to have the list of diagrammatic relationships around, as part of an evolving design process, so they are not discarded; indeed, since the d-elements contain only static graphics attributes, it is the accompanying relationships that are the repository of essential knowledge that distinguishes the diagrams from any other graphic artifacts.

That leaves the translation between a source design state and destination diagram state as the location of critical diagramming knowledge. This translation is between two similar data structures: each a list of elements with component variable values and a list of relations. The translation consists of creating elements, variables and relations on the destination side, and filling in variable values with some fixed relationship to the source side. This fixed relationship can be largely captured in a 'legend', that specifies part of the translation. The legend is a table, relating parts in the design state to parts in the diagram state. A legend is a data structure, often

---

[3]Diagrams using color are obviously useful, but limitation to black, white and gray serves to simplify the process without any loss of generality. Similarly, continuously-variable tone graphics are a possibility, but can also be ignored for now. Structurally these variants are no different from the black/white line drawings discussed here, although they require a different set of primitive graphic abilities.

represented as a simple diagram using the horizontal alignment relationship (or the 'arrow' element, or both) to stand for the "stands for" relationship.

Diagrammatic Steps

The translation from design to diagram can be divided into several steps: first a declaration of design state variables, values, elements and relations; then a filtering, in which design parts to be translated are selected; then a mapping, in which elements and relations in the target diagram are created to match elements and relations in the source filtered design state, and selected variable values are mapped into the target state. Some (possibly all) variable values in the diagram state may not be specified by the mapping; two more processes follow that must fill in all remaining variable values (and result in a diagram state that is drawable, as specified above.) There is first a default assignment phase, in which all values (except position) that are not specified get assignments based on some set of diagrammatic rules; then a layout stage fills in all remaining values --primarily positions -- to make a drawable diagram.

Each of the last two steps is controlled by the diagrammatic relationships, that serve to provide default assignments and layout rules. The layout stage may require a conflict resolution procedure, in the event that assignments and layout interact, as they often do. Finally, the diagram is viewed and interpreted by an inferrer (usually a person.) The relationship between these steps and representations is shown in figure 4.1 (a diagram).

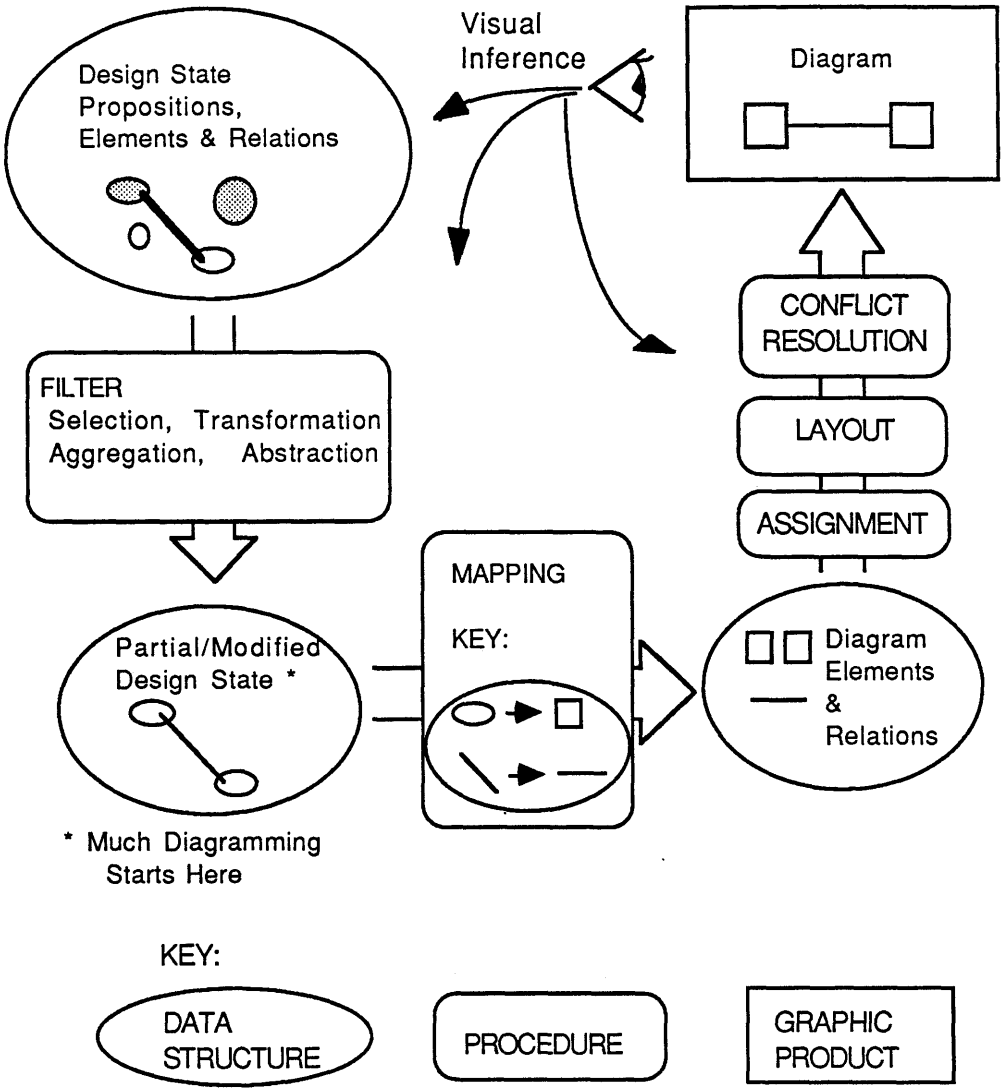


Figure 4.1 Diagram of Diagramming Algorithm

In the diagram shown, there is a design state in the upper left corner that is transformed by a filter into a partial design state in the lower left corner. This describes one kind of diagramming, in which the design state is relatively complete and must be simplified for the diagram -- what I characterized in the last chapter as producing a B-diagram from a drawing, for example. Another kind of diagramming, the production of A-diagrams from a simple set of propositions, starts with the partial design state. In ths case, the filtering is not *a-posteriori*, applied to a design state, but is *a-priori*, built into the process of developing design statements -- the designer starts with

partial and incomplete states. This kind of diagramming is the kind the majority of this thesis is devoted to.

Each step in this process must be informed by some knowledge about how the diagram will be read and interpreted by the human designer/inferrer; that knowledge, in turn, would form the basis for a program that could interpret (read) diagrams. For each operation on the production side, there is a mirror operation on the interpretation side (although there are several one-to-many or many-to-one relationships, so that the creation of the diagram-reader doesn't automatically follow from the diagram-producer.) The remainder of this chapter is devoted to the production of diagrams, but some notes along the way describe the reading of diagrams and anticipate a partner program, the diagram reader. The conclusion in chapter eight more directly addresses the reading of diagrams.

Restating the above process, the major steps in the production of diagrams (declaration, filtering, mapping, default-assignment, layout, inference) fall into three groups:
1.) initial declaration of elements and relations in the design state, and specification of the filtering -- this must come from the human designer, and results in a partial design state;
2.) choosing graphical symbols to be used as elements in the diagram, and the mapping part of the translation -- this results in a partial diagram state, with some but not all variable values filled in; and 3.) default assignments and layout of the diagram, resulting in a completely filled in diagram state, and therefore a drawable diagram.

Pre-Diagrammatic Design Declarations

Declaration of elements and relations in the source design state takes the form of 'verbal' -- that is, symbolic -- propositions asserting the names of elements and specifying their relations. These propositions need not be final or fixed -- they may be tentative and contingent, but are entirely necessary (you have to start somewhere; you have to know something.) Some design states are more

diagrammable than others, and more fruitfully diagrammable: The proposition "The playground must be beautiful" names an element and a relation, but is not particularly diagrammable in a useful way: "The playground contains an active area, a rest area and a secure area" is much more so. Partly this is because the first example is unary (there are no other elements, so no inferences proceed from the diagram that don't proceed anyway; if the conclusion is "Therefore it must have a water feature and flowers in it" that conclusion can be reached without the diagram, and the diagram doesn't help) and partly because the relation "beautiful" has no obvious translation into symbols or arrangement. In the second case, since there are more than several elements, and because the relationship 'contains' has a direct two-dimensional graphic analogue, there are inferences that proceed from a diagram. These may include "Therefore either all the activities are mutually adjacent, or there is some space in between them that needs a name and possibly some treatment" and "There is some spatial relationship between the active and the restful area; what is it?", for example.

The following discussion expands on the design declaration that is preliminary to diagramming. This declaration of a design state, and of an appropriate filtering of that state, are up to the designer and outside of the computer diagrammer. The following discussion will introduce the distinction between Properties and Qualities, which although not a hard and fast one, is useful to understand some of the role of diagraming [Bucciarelli and Schön].

The Declaration of Intent. This design step is the first act of will of the designer, in which some particular intent or design goal is declared. The declaration may be vague, general, grand, trivial, specific, etc, but it guides the development of some one or more design artifact(s), through the following steps. An example might be "Use brick", or "Create a restful public place". The former moves directly to a Statement of Properties; the latter to a Statement of Qualities.

Statement of Qualities. This step may refine a Declaration of Intent, or may come from out of the blue; the purpose is to articulate specific desired qualities in designer-chosen terms. Qualities (as distinct from properties, described below) are understood as subjectively stated, qualitative adjectives describing ends, effects, characteristics of and "felt responses" to the desired design artifact or its graphic representation. Examples of qualities are "profound" "vivid" "private" "secure" "comfortable" "monumental" "colloquial" "community-fostering" "beautiful" etc.; any number of phrases beginning with "a sense of..." or "reminiscent of ..."; and any number of terms from the properties list below, used metaphorically or imprecisely as "continuous" "serpentine" "bright" "dark" "protected" etc.

Statement of Elements. This design step may be partly incorporated into the above, and like the others it may be returned to over and over again. The result is the identification of 'things' that the design is concerned with, and they may be specific, generic, atomic, aggregate, nebulous, well-defined, etc. "A park" "a brick wall" "a composition of rectangular spaces" "a new community" "a prototypical facade" "an underpass" "a minimum dimension" "a cozy nook" are all examples. Each of these elements will, hopefully, join to produce the desired intents and qualities; their description may contain qualitative modifiers.

Statement of Properties. This step may refine a Declaration of Intent, or may come from out of the blue; the purpose is to articulate specific desired properties in designer-chosen terms. Properties are understood as objectively stated, measurable characteristics that derive from the physical or geometric characteristics of the design artifact, its elements, or its graphic representation. An essential feature of properties as distinct from qualities is that they are accompanied by an explicit or implicit metric and test that can be objectively applied and generally agreed upon (as distinct from qualities that may be idiosyncratic, immeasurable, undefined, etc.) Examples of properties are "made of brick" "circular" "rectangular" "symmetrical" "axial" "radial" "one meter long" "clustered" "linear" "enclosed" "protected" "14th-century Italian" "warm" "cool", etc. Each of

these terms may of course be applied with any degree of precision, or amount of approximation or interpretation -- it is just that these degrees, amounts and criteria, if not conventional, can be articulated.

The step of declaring properties need not follow from intents or qualities, although it often does. This step may be the starting point, with no antecedent or justification; it is a critical step for the construction of a diagram, along with the statement of elements. These properties may be understood as the first level of statement of relations between elements; each of the properties applies to some one or collection of elements.

Translation to Specific Terms (Elements/Relations/Variables) -- quantitative refinement of relations/properties into Elements, Relations and Variables. This step may be partially rule-based, but requires invention on the part of the designer. What comprises an element or a relation is not absolute or determinable *a priori*; this statement of elements, relations and variables is a value-laden and implication-rich step. Some qualities and goals may not immediately be expressible in constraint terms -- this deficiency may lie either with the expressiveness of the available terms or the inventiveness of the translator. It may be that some qualities, goals, elements, etc do not lend themselves to this quantitative/algorithmic translation; that question is an open one, but need not concern us here (except that this means these goals, elements are not diagrammable!)

The above steps are generic design decision processes, that generate or collect design knowledge related to the design at hand, and state them in specific terms. The steps that follow are specifically diagrammatic decisions -- these same steps could also lead directly to a plan or other drawing; the difference is only a matter of degree, and of certain levels of commitment and specificity, as detailed below.

Filtering

Diagrams typically do not represent design states completely, but rather views on them; for a site plan, for example, a circulation diagram, an open space diagram, a building massing diagram. So a diagram proceeds from a filtering; removing some set of elements, and some relations and variables, from consideration. This must be accompanied by grouping; some set of elements lose their individuality and become represented by a single element (several houses become aggregated as a cluster, e.g.) The definition of this filtering is conventional for some standard diagrams, like circulation diagrams, or wiring diagrams in electronics; but the filtering and grouping may also be invented on the spot, for the sake of an argument. This filtering and grouping for the sake of simplicity is often accomplished by the abstraction previously discussed.

A crucial part of the filtering step may involve regrouping elements and relations the design state, if the elements and relations desired for the filtering are not directly represented, but must be inferred in the source design state. For example, only walls and doors may be stored in a design state, but to produce a circulation diagram, nodes and connectors are required. These latter can be inferred from the former (though not necessarily easily or cleanly!), and must be, before the diagram can proceed. This regrouping of elements in the design, or invention of new elements, may be considered an enhancement of the design state, and permanently recorded, or it may be ephemeral, only for the purposes of the diagram, and not permanently recorded. This judgement depends on the designer and the capabilities of the constraint manager.

Filtering produces, or may be guided by , a 'legend', or key. I observed earlier that each diagram is typically accompanied by a legend, providing a key to symbols. The filtering stage, in selecting what elements from the source will be represented, specifies one side (the left-hand side) of the legend; the right-hand side of the table is determined by the destination elements and relations, the built-in capabilities of the diagramming medium.

Diagrammatic Steps -- Mapping

The Mapping stage includes the actual translation, and the assignment of some initial values in the diagram state (the rest of the values get filled in by the Layout phase.)

Mapping between Design State and Diagram State. This step seeks to cast the elements and relations in the source design data-base into the diagram data base. Each object, relation and variable value potentially has a corresponding diagrammatic-element, -relation, or -attribute. The choice of target elements (d-elements, -relations and -attributes) is pre-determined, by the built in library of the diagrammer (but is also subject to modification and invention.) This mapping may be governed by 'style' preferences, as indicated below. The details of this mapping also determine whether the resulting graphic is diagrammatic or more detailed (plan-like).

In this mapping, design elements need not go to d-elements; they may become d-relations, and vice versa (relations become d-elements). As an example, a relation like 'implies' or 'connects to' in the design state may become an element in the diagram -- an arrow, related to another element by an alignment relation. Just as the initial decision in the design state as to what is an element and what is a relation is open to the designer, so is the decision about whither to map elements and relationships.

Elements $\Rightarrow$ d-elements. This step requires casting relevant elements as diagrammatic-elements signified by symbols (which may be atomic or compound). The choice of available symbols is built into the diagrammer's graphic capabilities. This conversion may be governed by visual analogies (linear things get linear symbols, spread-out things get blob symbols, e.g.); by less-direct analogies (expensive things get large symbols, larger things get darker symbols, etc.); or arbitrary assignment (expensive things get linear symbols, larger things get square symbols, etc.) In the last case, the diagram may be obscure and difficult to read, depending on the relationships portrayed.

Relations ⇒ d-relations or d-elements. As with the previous conversion, this conversion may be governed by visual analogies ("surrounds" becomes symbolized by the relation of concentric, "shares some attribute" becomes overlapping, e..g)); by less-direct analogies ("larger-than" becomes "left-of"); or arbitrary assignment ("smaller than" becomes "larger than", e.g.) In the last case, again, the diagram may be obscure and difficult to read, depending on the relationships portrayed.

In this translation, relations may become symbolized by elements. For example, "communicates with" may become symbolized by a connecting line, a d-element that corresponds to no object in the source design state (unless there is a telephone wire implementing the "communicates with" relation) .

Variable ⇒ d-attributes. This mapping is related to the previous decisions. For example, if the mapping was from "larger than" to "left-of", then the variable ⇒ attribute mapping must be "size ⇒ x-coordinate" where the mapping is not equality, but requires a mapping function f

f: x-coordinate ⇐ f(size).

Assignment of Diagrammatic Defaults. This step is critical in the distinction between diagrammatic and other graphics. Default values may be applied instead of mapping from variable values; for example, all diagrammatic-elements may become the same size (diameter), ignoring any size attributes in the design state. This assignment is significant in the implications of the diagram, for it suggests (or perhaps stronger, dictates) that the relative size of elements (or the corresponding attribute of design objects) is not relevant to the argument of the diagram. The default values assigned are largely knowledge based, not arbitrary, and may be conventional or unique.

The term "default" might suggest a trivial, or insignificant decision, but that's not the case. Diagrammatic defaults are knowledge-based, and their choice and assignment is high art, rather

than blind allocation. The decision is "default" in the beginning, in the sense that it is a decision made in the absence of any other compelling information; but as soon as it is made, it is implication-rich and may be significant. For example, if a mapping is made back from the diagram to the design state, instantiating the diagrammatic default in the design, then the design is subject to the criticism of "building a decorated diagram" [Herdeg], or "implementing the bubble diagram", considered a fault because there is no subtlety or refinement in the resulting design state .

Default assignments are made exactly so that the diagram state, and by analogy and inference, the design state, becomes visible, and so provides fuel for visual inference. One function of diagramming is to test the implications of a single decision or set of decisions, to see what consequences are triggered; these consequences become visible in part in the defaults. This visual inference adjusts for and relies on ambiguities in the default assignments, as these at once provide neutrality and implication.

Layout Algorithms

The final step, layout, is the step of actually producing graphics. Attributes and relations interact, and these interactions must be controlled. The problem, simply stated, is to place all the selected symbols inside an arbitrary frame to produce the diagram. Along with placing the symbols -- specifying the position values -- other attributes may be modified, notably size and shape, as the layout proceeds. These dynamic modifications are in response to scale and layout requirements imposed by the frame, and to basic rules of graphic legibility. The layout process may need to explicitly refer to a conflict resolution procedure, in which conflicts resulting from default assignments and layout procedures are resolved, possibly by recourse to human intervention. The degree to which conflict resolution may be automatic, or must depend on human intervention, is a function of the adequacy and adaptability of the default assignment and layout procedures. Two general algorithmic approaches are described below.

A major rule of graphic legibility is to forbid overlap and intersection of elements, except as explicitly intended by the relations. A second rule relates to white space and proportion in the diagram. Although diagrams don't usually make of use of background/foreground ambiguity -- the foreground is usually intentionally clear -- the legibility and style of the diagram depends on the spacing between elements, especially when there are inclusion and surrounding relations. I'll return to questions of legibility, and the factors and sources of knowledge that determine it, in chapter seven.

There are two basic approaches to the layout phase (corresponding to established approaches to constructing adjacency graphs such as floor-plan layouts), sometimes called 'permutational' and 'additive' [Steadman & March] . The first is an iterative generate-and-test approach: produce an initial complete layout by some method, then test the layout for compliance with the rules, and attempt to improve the layout by permutation of elements if any conflicts are noted. The second is an incremental approach: place elements one by one, at each step doing the best job possible and adjusting the layout as required; at completion, assume the layout is the best and stop (no recourse to the generate & test approach above). These two strategies are different in several regards. The first approach must have a method for generating a starting diagram, without particular regard for its adequacy, and a method for devising strategies to resolve conflicts when they are detected.

The second approach seeks to prevent conflicts from ever arising, by step-by-step construction, adjusting the diagram as necessary in the process. This adjustment can be seen as the same as the conflict resolution in the previous approach, but since it deals with a single element at a time, it seemingly has more hope of isolating and solving problems. At each step in the process the task is to place one element (symbol) so that its position, size and other attributes satisfy all specified relations, don't interfere with any pre-existing ones, and don't suggest the contrary of any specified relations. This is a 'hill-climbing' process that suffers from the usual problem of local high points -- in seeking to satisfy all the local and current considerations, the process can not 'move backwards'

to a better starting position and so may arrive at a local maximum which is less good than another 'global' alternative. In a computer-aided design approach, this is the logical place to allow human intervention -- when the process seems to be stuck, it may be beneficial to let human visual cognition intercede. Detecting the 'stuck' state is slightly simpler in the incremental approach: whenever a lack of suitable rules or any irreconcilable conflicts make it impossible to proceed, the program can detect the impasse. In the above iterative approach, a more complex logic is required, because of the possibility of being stuck in a circular sequence. The detection of cycling is more complicated, requiring a complete history to be maintained, and checking the history at every step, looking for repetitive cycles of arbitrary length.

[Steadman and March] point out that the incremental or additive process requires two subprocesses: a 'framework' for layout which, among other things, guides the placement of elements in the absence of any relevant relations; and a strategy for determining the order in which elements are placed. An example of the first is a grid, which applies local constraints to element positions, and can be applied locally; this might be accompanied by a goal of rectangular, linear, or circular overall form, for example. This latter goal can not be satisfied by only local decisions, but a guiding algorithm that takes account of all other elements may be sufficient to roughly achieve the goal.

As for a strategy for placing elements, three possibilities are obvious. The first is to take chronologically the assertions provide by the designer, and incrementally improve the diagram as the relations are encountered. The second strategy seeks to minimize possible conflicts and dead-ends, by rank ordering the elements according to their degree of interconnectedness, and then placing them in decreasing order; that is, place the most interconnected element first, and so on. The metric for interconnectedness is not simple, as it must account for chains of relatedness as well as for direct relatedness. This approach is more satisfactory for adjacency graphs than for other kinds of diagrams, since there are topological constraints on the numbers and forms of adjacency graphs that

are planar, and so the 'diagrammability' of these can be algorithmically determined. Where the constraints on placement and intersection are not so rigorous, the determination of a satisfactory solution may not be algorithmically computable, and require human cleverness and/or luck. The third strategy also seeks to minimize conflicts, but based instead on size. Where diagrammatic size is fixed or significant (instead of being variable or insignificant) then rank-order the elements by size, and place the largest first, with respect to some overall strategy such as "center in the frame."

The iterative approach to layout has the benefit of allowing for 'gestalt' effects in the interpretation of the diagram, but also the attendant difficulty of devising the program to do the interpretation. There is also the problem of determining a starting position, even an arbitrary one. In this approach, improvements and corrections can be made on the basis of the whole diagram, but the process may suffer from combinatorial explosion in the detection of conflicts and devising resolution strategies, since there may be more than one conflict detected at each test.

The incremental approach has the disadvantage that many small decision are made in isolation, but the attendant advantage that these are relatively easy to make. Furthermore, the incremental approach at least has the potential of producing an initial diagram for the iterative approach to go to work on. For this reason, and because of the difficulties of the diagram interpretation required in the iterative approach, the Diagrammer presented in the next chapter takes the incremental approach.

In the incremental approach, the question remains where to start. One possibility is in chronological order, as propositions arrive, but a more fruitful approach starts with an analysis of the complexity of the diagram, to indicate a strategic starting point.

The analysis must first identify the 'cardinality' of the diagram to be produced, that is, the number of different objects or variables to be represented (this may be the number of different kinds

of design objects represented in the diagram, for example). For the diagramming program, values for the available d-elements must be listed in a table beforehand. In practice, designers often invent these values as needed, to augment or replace a set of conventional ones.

In addition, an analysis must be made to determine a value I'll call the "diagrammatic degrees of freedom", (or d-df) of the diagram: this is a value indicating the maximum number of axes or mutually exclusive attributes that any one element must have. This could be the maximum number of different categories any one element belongs to, that must be represented by some graphical attribute. For each such attribute, a spectrum of values will be required. For example, 'shape' is an attribute; 'round', 'square', 'triangular', etc are values.

The first count -- d-cardinality -- is an indicator of the "magnitude" of the diagram, the second number -- d-df -- is an indicator of complexity. For example, a diagram showing seven different elements with no overlap between any of the elements might simply use seven different colors, and only one shape, to indicate the categories -- it has cardinality seven, but d-df of one. On the other hand, a diagram with only one element, which belongs to seven different categories cannot simply use seven different colors, but must have seven distinguishable attributes -- color, size, lineweight, style, tone, shape and position, for example. Thus a red, large, bold, wiggly, dotted square in the first quadrant is distinguished from a green, small, light, straight, dashed triangle in the third quadrant; this example has only cardinality of two, but seven degrees of freedom. As suggested by this example, degrees of freedom above about seven become very difficult to diagram, because the possible attributes are increasingly different to invent or to distinguish by eye.

The following 'pseudo-code' describes a skeletal algorithm, in steps grouped into named modules, for producing a diagram from a design state.

Diagramming Algorithm

DECLARATIONS:

state goals in personal terms: G_i

foreach G, translate to desired or required qualities: Q_i

state elements in personal and conventional terms: E_i

foreach Q, translate to desired or required properties: P_i

foreach pair E_i, P_j,

translate to Relations R_i Elements Ei and Variables V_i

assign 'fixed' values where known

FILTERING:

eliminate some elements and relations, leaving a set of E R and V

MAPPING:

foreach R, assign diagrammatic-relation from list

foreach E, assign diagrammatic-element from list

foreach V, assign diagrammatic-attribute from list

ORDERING:

if size-is-relevant

rank order all diagrammatic-elements by size

if adjacency-is-relevant

rank order all diagrammatic-elements by interconnectedness

DEFAULT-ASSIGNMENT-1:

foreach V that is unassigned, (except position)

assign diagrammatic-default for all diagrammatic-attributes

propagate assignment, check relations, resolve conflicts

LAYOUT:

foreach diagrammatic-element, in order

DEFAULT-ASSIGNMENT-2:

if position is unassigned

assign default position

propagate position, resolve conflicts (see detail)

GLOBAL CHECKING:

if possible ;; requires diagram-reading, global inference

check all properties P for compliance/presence

resolve conflicts (see detail) if possible...

Detail of Layout & Conflict Resolution Algorithm

The following algorithm is a detailed description of the step called "resolve conflicts" above.

LAYOUT AND CONFLICT-RESOLUTION:

```
assign default position
    case:   no elements placed
            place in arbitrary starting position (center frame e.g.)
        element related by d-relation
            apply d-relation algorithm to place
        some elements already placed
            place in rule-based next  position (offset by dy,dx e.g.)
propagate-position, resolve conflicts
    propagate through chain of d-relations to all new-values
    foreach new-value
        case:  previously unassigned
               assign and propagate (recur)
           previously assigned default
               check for circularity *
               if no circularity
                   find previous justification
                   look for simultaneous solution **
                       if found
                           assign and propagate
               else
                   check consistency of new-value
                   with previous justification ***
                   if consistent
                       assign and propagate
                   else
                       try conflict-resolution procedure ****
                       if conflict remains
                           give up -- ask for user help! *****
```

Notes on Detail Algorithm

The following notes refer to the starred lines in the conflict-resolution algorithm detail.

* check-for-circularity keeps a global history list for each chain of propagation. If this value-justification appears on the list, then circularity is detected -- give up.

** check-for-simultaneous-solution looks for algorithmic resolution of interacting relations. This maybe rule-based, table lookup, etc. If none found, try incremental solution, then give up if conflicts remain. This step may also look for known impossibilities or logical contradictions, and report them if found, by jumping to ask-user-for-help.

*** check-consistency for any value applies predicate form of each relation in turn, to see if satisfied.

**** apply-conflict-resolution procedure looks for optional resolution procedure attached to each relation and tries it (for example, relaxation, iterative approximation, etc.)

***** ask-for-user-help is the step of last resort, appealing to the designer eye-brain for a solution to the multiple relationships. This step may cause a restatement of the relations at the Element/Relation-level, and cause a new diagramming to begin (recycle)

Managing Defaults

In the operation of this skeletal algorithm, especially in the mapping and layout stages, initial or tentative values are assigned that are necessary for producing graphics, but are not directly derived from the design state, and may be changed by subsequent diagramming steps, as new propositions are added. These values I have called defaults, since they are values based largely on convention and convenience, represent no particular commitment to particular design circumstances, and may be defeated by subsequent information. Many of the decisions and assumptions made by the diagrammer in layout share these features of defaults, though their derivation is different.

For example, when the diagrammer places an symbol in the center of the frame of an empty diagram, this decision is not because "usually symbols are in the middle of the frame" (though in the case of a single symbol, that may be true), nor is it because the symbol "is an instance of the class of things that generally appear in the center of a frame". Rather, this decision is based on diagrammatic knowledge about visual perception -- the visual system preference for symmetrical or 'balanced' figures, for example -- and about the inferential purpose of the diagram -- the fact that some decision must be made, and the decision should be as value-free, or neutral, as possible. Were the symbol or group of symbols to be placed in the lower-right-hand corner, for example, the visual tension so induced would lead to the question "Why?" and "What might be the significance?"; and since there is no significance intended, this reasoning would be inappropriate. When another symbol is added to the diagram, the position of the original one might change (the two are horizontally aligned around the center, or this one is inside that one, etc.) The position decision is defeasible, in the light of new knowledge.

It might seem that this modification of values is a process of successive refinement, moving from approximate to precise values. In some cases, this progression is characteristic of designing in the move from diagrams to more detailed drawings. (At first the fountain is just "inside the plaza", then "in the northern half", then "at coordinates x,y".) But this process is distinct from (and not even parallel to) the reasoning that the diagrammer must go through: confronted with the assertion that the "fountain is inside the plaza", the position of the fountain for the purposes of producing graphics can not be an approximation or an interval, but must be a value that is precise, but unfirm. [4]

[4]The possibility of graphically representing approximations, ranges or interval values, is an interesting one that is outside the present scope.

In general, in a diagram in evolution, relatively few values are firm (those that are result from constant such as a minimum size constraint, for example.) Then the question becomes which variable value to change in the satisfaction of the new predicate -- that is, which value fixes to relax in order to satisfy all predicates. In the program described in the following chapter, this operation is called the 'Conflict Resolution' procedure, and I'll expand on the conflict resolution strategy there.

Visual Inference

The final step after the production of the diagram is the critical "Visual Inference" step in which the designer looks, infers, and recycles into the algorithm, usually into the declarations steps, but possibly also to override some default value assignments with fixes, either to refine some value in the design, or to modify the form or appearance of the diagram. This interaction is generally motivated by one of two reasons: adjusting the diagram's graphic features *per se*, for visual clarity and regularity (what was referred to in chapter three as the "Rules of Graphic Grammar"), or adding, removing or modifying features in response to a substantive design decision made on the basis of an inference from the diagram. Examples of both of these kinds of responses, and interactions, will be examined in chapter six.

Computer Implementation

This chapter described an algorithmic approach to producing diagrams, in the context of a process of designing. This next chapter introduces the terms of the 'constraint model' and describes in detail the Constraint-Based Diagrammer, that implements the Layout step of this general algorithm.

Chapter 5. Diagrams as Constraints and The Constraint-Based Diagrammer

In the last chapter I presented an approach to producing diagrams. Now a computer system for producing diagrams -- the Constraint Based Diagrammer, or CBD -- is described in more detail, using the terms and principles of 'constraint-management'. A simple lexicon of elements and relations is detailed.

Following the principles of diagramming described in chapter three, and the approach to diagram production described in the last chapter, I can describe the details of the computer program called the Constraint-Based Diagrammer (CBD). The diagrammer takes a list of propositions as input, and produces graphics -- a diagram -- as output. A hierarchy of element types and a library of procedural definitions of relations must be predefined by the user, and the proposition list must be expressed using these predefined elements and relations. First the constraint model of designing and the constraint manager computer program are reviewed.

The Constraint Manager

As both designs and diagrams are characterized by sets of elements and relations, a data structure and an approach to computing that manages relations between elements is required for the diagrammer program. The constraint model [Gross 85, Gross, Ervin Anderson & Fleisher] provides a formalism and just such a data structure and computational approach.

The term 'constraint' in the constraint model has both a commonplace and technical usage -- I used the term in its common sense several times in the last chapter. In technical usage the term is perhaps most often applied to bounding conditions in optimization theory. Those constraints usually take the form of equations (linear or non-) on one or more variables, that express some condition on the permissible values for the variables. More generally, "Constraint Satisfaction Problems" (CSP's) are concerned with finding a consistent set of values (out of a possibly indefinite range) to assign to a set of variables controlled by some conditions, or constraints. Common usage of the term has a negative connotation, and designers in particular commonly distinguish negative

'constraints' from positive 'opportunities' in design development, but in the constraint model, and here, the neutral technical meaning of any explicit condition on variable values is intended. In general, I shall use the terms 'constraints' or 'relations' and 'variables' or 'elements' interchangeably.

The constraint model describes designing as a process of expressing and exploring constraints and trying to achieve objectives. The process is composed of several operations: describing and structuring design elements (variables); expressing constraints and objectives; exploring fixes (settings of values to variables) and their consequences; identifying and resolving conflicts between constraints; comparing alternative sets of variable values and relations. No fixed or linear sequence is implied: the process cycles, refines, branches, and backtracks[1].

The theory is about knowledge and reasoning— designing is seen as exploring a space of alternatives by applying constraints, introducing new design variables, trying fixes, restructuring collections of variables and constraints, and occasionally optimizing some sub-problems. The theory does not argue that designers work this way nor that they should. It does argue a specific way designing can be done, and provides a basis for a computer program to record variables, constraints and fixes, and maintain consistency and report conflicts among fixes. Constraints, variables and values are the repositories of knowledge in this model; the dynamic process of designing is described as exploration, or search (where the search is complicated by the fact that the object of the search is partly or completely unspecified).

Designing is rarely a matter of optimization; there is rarely a singular solution or a single objective function to optimize. [2] Nonetheless, a design does usually represent a choice that keeps some values

---

[1] The words in this sentence, and several surrounding it, originally appeared in [Ervin and Gross] and were originally written by Mark Gross.

[2]One model has described designing as choosing between 'Pareto optimal' alternatives [Navinchandra 1987], meaning alternatives that are equally acceptable, or with equivalent scores

within prescribed bounds, and respects some relationships between elements of the design. In optimization problems, the constraints are taken to be given and fixed; in designing, discovering and revising constraints is a major part of the job, called the 'Declaration' stage in the last chapter.

CM2-- Structure and Logic

The model of designing by exploring constraints is implemented in a computer program called CM2. The details of its operation are described in [Ervin & Gross]; an overview will suffice here. The Constraint Manager is interactive, including as an essential component a human designer. The operation of the program is a dialog: the designer makes a statement (or asks a question) in the language of constraints and variables, the constraint manager records the statement, calculates its consequences if any, and if the consequence is a conflict of any sort informs the designer, who must then resolve the conflict by retracting some previous statement, or adding some new statement that removes the conflict. The process continues indefinitely so long as the designer has statements to make, or questions to ask. After each statement, the constraint manager is said to contain a 'constrained state' that describes the design. So long as each conflict has been resolved as it came up, the constrained state will always be consistent. Some conflicts may not be resolved immediately; an inconsistent design state is both possible and common during designing, but not as a final state.

The statements made by the designer are generally of two sorts: stating a constraint between two or more variables ("registering a constraint"); or assigning a value ("making a fix"). In either event, the constraint manager considers all other constraints and fixes, and if the new information enables the calculation of any values, they are calculated by a process of inference ("propagating the constraints"). If a newly fixed or calculated value of a variable is different than its existing value,

---

on some scoring system. This is not much more help than saying that designing is choosing from among alternatives, which is not a bad characterization - but then why bother to make the choices 'Pareto optimal'?

then a conflict exists and must be resolved. This dialog, and its logic, will serve as the basis for the constraint-based diagrammer.

CBD Elements – Types and Attributes

The diagrammer starts out with a base of three primitive graphic object 'types': points, lines and polygons. Each of these primitive types constitutes a class of objects; that is, particular objects can be created that are instances of the type (Line1, Line2, Line3, are all lines, e.g.), each instance having all of the characteristics of the primitive type, and possibly particular local attributes not shared by all (Line1 is dashed, Line2 dotted, say). Each type definition describes common characteristics of all its instances: the generic procedure to draw a line, for example, is part of the definition of the line type; individual lines simply refer to this generic procedure, providing only the information that is unique, or local, to the instance (the x and y coordinates of its endpoints, for example.) This relationship between individuals, classes and attributes, called 'object oriented inheritance' in computer terminology, is an intuitively attractive and computationally powerful way to organize knowledge about the world[3].

Each primitive type has a predefined set of attributes that describe its graphic appearance : size, linewidth, color (tone in a black and white implementation), and fill (for polygons only). Points and lines are primitive graphic objects with no predefined variations, but there are three kinds of polygons that are generally interchangeable: rectangles, ovals and rounded rectangles. Thus, the drawing part of the diagrammer recognizes only five commands (procedures with arguments):

```
(DRAW-POINT POSITION LINE-WIDTH TONE)
(DRAW-LINE START-PT END-PT LINEWIDTH TONE)
(DRAW-RECTANGLE UPPERLEFT-PT LOWERRIGHT-PT LINEWIDTH TONE FILL)
(DRAW-ROUNDED-RECTANGLE UPPERLEFT-PT LOWERRIGHT-PT LINEWIDTH TONE FILL)
```

[3]This facility, the management of names and types of objects, and the assignment of values and constraints on the basis of object type, are handled by the 'Object-LISP' inheritance mechanism of the Allegro Common LISP interpreter.

(DRAW-OVAL CENTER-PT RADIUS LINEWIDTH TONE FILL)

These graphics objects constitute the vocabulary of elements that can be used as the destination side of the mapping in the diagramming algorithm. Relations on the destination side are those that can be defined in terms of the attributes of these elements, as described below.

The diagrammer also recognizes elements and classes of elements in the design state, on the source side of the mapping. These elements and classes are domain-dependent, and not built in. They must be defined by the user, and may be changed in the process of designing. For the purpose of the examples in following chapters, a simple lexicon of design elements will be used, like Lynch's five-fold classes: nodes, landmarks, paths, edges, districts. Again, individual nodes and landmarks may be created that inherit all the attributes of nodes and landmarks in general. The base of elements can be extended by the user to include new kinds of objects related to the primitives: a fountain as a kind of landmark, an urban plaza a kind of district, a subway stop a kind of node, for example. This hierarchical nesting may be arbitrarily deep: two different kinds of fountains (modern and classical), three different kinds of modern fountains, and so on.

The mapping between source state and diagram state then specifies a relationship between each of the source elements and relations, and diagrammatic elements and relations. For simplicity and economy, the mapping may only be from class to class: the mapping from design elements of the class "path" to diagrammatic elements of class "line" is an obvious one, as are "node" to "point" and "district" to "polygon". Individual source elements get mapped as their class is mapped: if P1 is an instance of a path, and a path is mapped to a line, then P1 will be mapped to a line in the diagram. The source element P1 will have all the attributes of a "path", such as type of paving, frequency of use, width, etc; some or all of these attributes will be mapped to corresponding attributes of lines; linetype and linewidth, for example. The list of attributes available to the diagrammer for each graphic type provides a limitation on the mapping. For example, only three

attributes of path-like elements can be mapped to the corresponding attributes of linetype, color and linewidth.

Relations

Relations between elements are also built-in to the diagrammer, in the form of an extendable library of LISP procedures. Relations in the CBD look like ordinary LISP procedures in the form of s-expressions like:

(RELATION ELEMENT1 ELEMENT2 ... ELEMENTn)

where the RELATION must be defined as a procedure that does four things:

1.) For each ELEMENT, if the element already exists in the diagram, check its type to make sure that it is legal for the relation, and signal an error if not. Each relation, when defined, specifies a list of legal types (point, line, polygon) for each element in its argument list. For example, the relation CONNECTS requires a linear element in the first position (the connector); the relation SURROUNDS requires a polygon. This requirement forces an explicit[4] vocabulary of diagrammatic relations, and makes application of the relations possible.

2.) If any ELEMENT does not already exist, create it according to a default type. In the list of legal types for each element, the first (or only) element in the list is taken to be the preferred default type. For example, the relation SEPARATES will accept either linear or polygon elements as the separating element, but the linear type is preferred, and used as the default type. Again, this requirement forces limitations on the vocabulary of relations available to the diagrammer, but helps to establish predictable behavior. The specification of legal and default types, just like the specification of relations in general, is a knowledge-based enterprise undertaken by the designer (in the case of the CBD, the author.)

3.) Register with the constraint manager a predicate that tests for the existence, or truth, of the relation, by a set of equality or inequality constraints. This predicate is called into play if all the

[4] some might say stilted!

elements exist and have attribute values assigned; if the predicate is satisfied, no further action is necessary.

4.) Register with the constraint manager a generator procedure, to be used if the predicate is not satisfied, that generates a set of default values for all the unassigned attributes, designed to satisfy the predicate. This procedure takes the form of a set of equality or assignment constraints between the variable values of the elements in the relation.

Predicate-Generator Relations

A relation expressed as a constraint is meant to be satisfied, or enforced, in the design or diagram. Some such constraints can easily be written as equalities: If a "Street must be three times the width of a car", a constraint might appear as: (= (STREET WIDTH) (* 3 (CAR WIDTH))) . If (CAR WIDTH) is known, then (STREET WIDTH) can be calculated, and vice versa; if both are known, then the truth of the relation can be established by an equality check. In this case, the constraint is both a predicate (the equality can be checked if both values are known), and a generator (if either value is known, the other can be generated, calculated by solving in the equation). Finished designs are full of such relations [Gross], but diagrams contain far fewer.

Diagrammatic relations are more likely to be inequalities and approximations: "A street must be at least as wide as two cars", which can be expressed by the inequality constraint: (> (STREET WIDTH)(* 2 (STREET WIDTH))). Again, if both values are known, the truth of the relation (the 'greater-than' predicate) can be checked. However, if one or the other, or both, values are unknown, the generator is not as simple as in the case of equality. The constraint doesn't specify how much wider the street is to be than two car (that's a detail for later on; for the purpose of the diagram, it just must be large enough for two cars to pass.) The generator must be able to produce some acceptable value in this case. One approach is to replace the inequality by an equality constraint containing a constant; for example, "greater-than" becomes "equals times some constant" or "equal plus some constant": (> X Y) becomes (= X (* Y k)), or (= X (+ Y k))

where k is some constant. The value of the constant is arbitrary, but its choice will affect the overall appearance of the resulting diagram -- if large constants are used, the resulting diagram is likely to be spread out and sparse, for example.

Inequalities may also be aggregated to define topological relationships in the lexicon, like "A must be connected to Area B which is inside Area C". In expressions like this, the ambiguity is not just in metric inequalities, but in other attributes. Connection, for example, might be visual, auditory, hydrological, physical, by telecommunications, etc. The process of designing, as I have said, is partly the expression of sentiment like "connected" followed by specification of the type and details of the connection, and diagraming often occurs after the sentiment and before the specification. In a diagram, "connected" might become represented by "adjacent, or sharing endpoints of a line" (where the line represents any of the various kinds of connections.) So the test for connected, the predicate, is to determine whether two areas are adjacent or share the endpoints of a line; this is not so simple a test as the equality test in the previous example. And the test for "A inside B", for example, includes "A smaller than B", and "union of boundaries of A and B same as boundary of A and not same as boundary of B." These relations must be built up out of primitive algebraic constraints, and their associated generator procedures must embody some relevant domain knowledge. So for example, in the CBD, the default generator procedure for CONNECTED produces a line between the centerpoints of the two connected elements. Note that in this case, a diagrammatic element -- the line -- is created to represent a source relation (CONNECTED), and the constraints in the constraint manager are equalities on the x- and y- coordinates of the respective line endpoints and element centerpoints.

Defaults

The values returned by generator procedures, in general, have the status of defaults. They are acceptable, but are only tentative and may be overridden. The choice of defaults is dependent upon a hierarchy of rules that determine the level and appropriate default decision. These defaults,

and the rules that govern them, can be divided into three levels. At the top-level, are domain-specific defaults like, for example: "Where a screen is desired, use a thick linear element". This is an example of domain expertise, in urban design -- the choice to use a single non-linear object, for example, might be more appropriate in a sculpture garden. This default can be encoded in a mapping, that maps 'screens' to 'lines' .

The next level of defaults governs details of graphics appearance: for example, that screens are to be represented by a single wide line in a gray tone, or that the line should be vertical instead of horizontal if there is a choice. These defaults too are modifiable by the designer at will, but are decisions based more on immediate graphics feedback and finetuning than on domain knowledge.

Finally, there are defaults that are generally stable, not usually changed by the designer, for example that a single line will be one-pixel wide, a single element will be 20 pixels in diameter. This level of default is a graphic device that depends on the current graphics environment (size of window, extent of diagram etc.) All of these levels must be built into the generator procedure that produces the default value.

Defaults enable drawing of incompletely specified objects. Some example defaults in CBD are: "All objects are represented by a circle" "All dimensions are at least 20 pixels" "All colors are black". These defaults can be overridden at any time, by the designer or by another constraint in the system.

Vocabulary of Elements, Relations and Diagrams

The diagrammer contains a primitive set of graphic capabilities -- points, lines and polygons -- as described in the last chapter, and a set of attributes that may be attached to these primitive elements. The choice of these elements and attributes is media-dependent; the ones chosen here are

a result of computer capabilities[5]. These provide the destination side of the mapping operation. The source side of the mapping, and of the filtering, are user-defined: whatever elements and relations are chosen for the domain in question. Some of these urban-design-specific elements and relations were presented at the end of chapter two, along with a generic list of relations derived from [Bongard]. Of the latter, only some are implemented here, the others representing non-trivial problems of algorithmic expression (the general detection, or generation, of symmetry, branching, or squiggliness, for example). Many of the formal relations like axial, gridded, radial, etc, are in this latter class.

The Lexicon as Constraints

The following section describes briefly the diagrammatic relationships that make up the language of the simple CBD.

EXISTENTIAL:

(EXISTS ELEMENT-A)
"create a new object in the data base, name it ELEMENT-A"

(IS-A ELEMENT-B ELEMENT-B)
"ELEMENT-B is an instance, with all the attributes, of ELEMENT-A. Changes in ELEMENT-A are reflected in ELEMENT-B, though not vice-versa"

(SAME-AS ELEMENT-A ELEMENT-B)
"ELEMENT-B has all the attributes of ELEMENT-A and vice-versa"

(DIFFERENT ELEMENT-A ELEMENT-B)
"For any attribute in ELEMENT-A, ELEMENT-B should have ad different value for that attribute and vice-versa"

TOPOLOGICAL:

(ALONG ELEMENT-A ELEMENT-B)
"ELEMENT-B must be type "LINEAR" or "AREA"; the distance from ELEMENT-A to ELEMENT-B must be less than DELTA. By default DELTA is [ELEMENT-B SIZE], and DISTANCE = 0."

(AWAY-FROM ELEMENT-A ELEMENT-B)
"the distance from ELEMENT-A to ELEMENT-B must be greater than DELTA. By default DELTA is (MAX [ELEMENT-A SIZE] [ELEMENT-B SIZE]), and DISTANCE = DELTA.

---

[5] in particular, the QuickDraw capabilities of the Macintosh.

(BESIDE ELEMENT-A ELEMENT-B)
"the distance from ELEMENT-A to ELEMENT-B must be less than DELTA. By default DELTA is (MIN [ELEMENT-A SIZE] [ELEMENT-B SIZE]), and DISTANCE = DELTA/2.

(BETWEEN ELEMENT-A ELEMENT-B ELEMENT-C)
"ELEMENT-C is located on the axis from [ELEMENT-A CENTER-PT] to [ELEMENT-B CENTER-PT]. By default (DISTANCE [ELEMENT-A] [ELEMENT-C]) = (DISTANCE [ELEMENT-B] [ELEMENT-C]).

(CENTRED-ON ELEMENT-A ELEMENT-B)
"ELEMENT-B is type "LINEAR" and
[ELEMENT-A CENTER-PT] = [ELEMENT-B MID-PT]"

(CENTRED-IN ELEMENT-A ELEMENT-B)
"ELEMENT-B is type "AREA" and
[ELEMENT-A CENTER-PT] = [ELEMENT-B MID-PT]"

(CONNECTS ELEMENT-A ELEMENT-B ELEMENT-C)
"ELEMENT-A is type "LINEAR" and
[ELEMENT-A START-PT] = [ELEMENT-B CENTER-PT] and
[ELEMENT-A END-PT] = [ELEMENT-C CENTER-PT]"

(DIVIDES ELEMENT-A ELEMENT-B)
"ELEMENT-A is type "LINEAR" and ELEMENT-A is type "AREA" and
(INTERSECTION [ELEMENT-A] [ELEMENT-B]) is not NULL."

(INSIDE ELEMENT -A ELEMENT-B)
"ELEMENT-B is type "AREA" and [ELEMENT-A SIZE) < [ELEMENT-B SIZE] and
(DISTANCE [ELEMENT-A] [ELEMENT-B]) < ([ELEMENT-A SIZE] / 2)."

(OUTSIDE ELEMENT-A ELEMENT-B)
"ELEMENT-B is type "AREA" and
(DISTANCE [ELEMENT-A] [ELEMENT-B]) > ([ELEMENT-B SIZE] / 2)."

(SEPARATES ELEMENT-A ELEMENT-B ELEMENT-C)
== (BETWEEN A B C)

(SURROUNDS ELEMENT-A ELEMENT-B)
== (CENTRED-IN B A)

(LINKS ELEMENT-A ELEMENT-B ELEMENT-C)
;; LIKE (CONNECTS ..) BUT DISTINCT, C IS LARGER AND DOMINANT

GEOMETRIC:

(INTERSECTS ELEMENT-A ELEMENT-B)
"(INTERSECTION [ELEMENT-A] [ELEMENT-B]) is not NULL. BY DEFAULT "

(TOUCHES ELEMENT-A ELEMENT-B)
"(INTERSECTION [ELEMENT-A] [ELEMENT-B]) is not NULL. BY DEFAULT "

(ORTHOGONAL ELEMENT-A ELEMENT-B)

```
        "((INTERIOR-ANGLE [ELEMENT-A AXIS] [ELEMENT-B AXIS]) = 90) "

(PARALLEL                ELEMENT-A       ELEMENT-B)
        "([[ELEMENT-A AXIS] ANGLE] = [[ELEMENT-B AXIS] ANGLE]) "
```

METRIC:

```
(LARGER                  ELEMENT-A       ELEMENT-B)
        "([ELEMENT-A SIZE] > [ELEMENT-B SIZE]) "

(SMALLER                 ELEMENT-A       ELEMENT-B)
        "([ELEMENT-A SIZE] < [ELEMENT-B SIZE]) "

(EQUAL-SIZE              ELEMENT-A       ELEMENT-B)
        "([ELEMENT-A SIZE] = [ELEMENT-B SIZE]) "
```

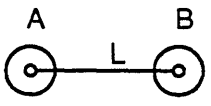
Construction of Diagrams

The operation of the CBD program may be simply described: a list of propositions constitutes the input; a diagrammatic data base and graphic illustration are the output. The essential work of the program is to implement the layout phase of the algorithm presented in the last chapter.

CBD uses a very simple layout algorithm: each new element is placed in the lower right hand corner, outside of the bounding box of all already-placed elements -- this guarantees that the new symbol will never overlap any existing symbols -- and then the position is adjusted if necessary as guided by the relationships.

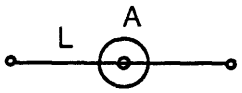
Even a single proposition from the lexicon will result in a diagram: the proposition (EXISTS THING) (which asserts only that an element called THING exists), will result in the default diagram:

The default type is a polygon (circle), the size is 20 pixels, the color is black, and the diagrammatic element has a center point. Most relations in the lexicon include two or more elements, and a position relation between them is specified by the constraint expression . Figure 5.1 shows nine 'primitive' diagram relations, with their corresponding default diagrams and constraint expressions.



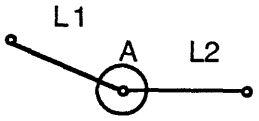
(connects L A B):

```
(poly A)
(poly B)
(line L)
(same-pt
     (L startpt)
     (A centerpt))
(same-pt
     (L endpt)
     (B centerpt))
```



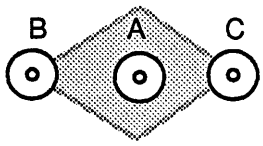
(centred-on A L):

```
(poly A)
(line L)
(same-pt
     (L centerpt)
     (A centerpt))
```



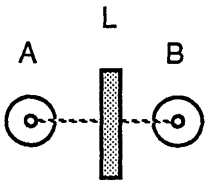
(joins A L1 L2):

```
(poly A)
(line L1)
(line L2)
(same-pt
     (A centerpt)
     (L1 endpt))
(same-pt
     (A centerpt)
     (L2 endpt))
```



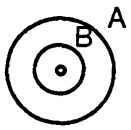
(between A B C):

```
(poly A)
(poly B)
(poly C)
(line L
     (B centerpt)
     (C centerpt))
(same-pt
     (A centerpt)
     (L centerpt))
```



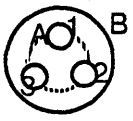
(separates L A B):

```
(fat-line L)
(poly A)
(poly B)
(between L A B)
(<-
     (L LENGTH)
     (sep-proc A B))
```



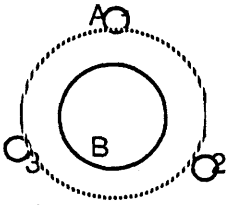
(surrounds A B):

```
(poly A)
(poly B)
(same-pt
     (A centerpt)
     (B centerpt))
(= (A DX)
     (* 2 (B DX)))
```



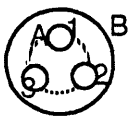
(inside A1...An B):

```
(poly A1 ... An)
(poly B)
(<-
     (An centerpt)
     (insideproc B n))
```



(outside A1...An B):

```
(poly A1 ... An)
(poly B)
(<-
     (An centerpt)
     (outsideproc B n))
```



(located-on A1...An B):

```
(poly A1 ... An)
(poly B)
(<-
     (A centerpt)
     (locateproc B n))
```

Figure 5.1 Nine Primitive Relations in the Lexicon

Description of Layout

The general diagram layout rule is stated simply: for each object, assign an icon according to a table (the destination side of the table contains the standard elements: points, lines, circles and rectangles); place the icon in an unused place in the diagram (just outside the lower right corner of the bounding box of all other objects, by default); proceed to adjust the position and, if indicated, size and other attributes of the icon in response to each of the constraints placed upon it by the relations.

In this process, it is likely that more than one relation will be imposed upon any one element or set of elements. In this case, the diagrammer must respond in a way that satisfies all the relations. Two different approaches to this situation, as mentioned earlier, might be characterized as the 'combinatorial' or the 'sequential'. It is plausible -- in fact likely -- that experts of all sorts, including diagrammers and designers, carry in their memory numerous standard and unique combinations (much like a good chess player recognizes certain positions, end-games, and notable precedents.) This knowledge is used to augment and override the reasoning from first principles ability that they also must have. Indeed, it has been suggested that this is a large part of what distinguishes a good but inexperienced novice from an expert. So our diagrammer might have a large repertoire of combinations in memory, that serve as a basis for modification, and obviate the need to construct a diagram step by step given a list of propositions.

For our purposes, no basis exists for encoding such a repertoire. Two possibilities suggest themselves: 1) collect multiple examples of diagramming conventions, and organize them into a library of standards, much as a 'knowledge engineer' seeks to brain-dump experts into an expert system shell; or 2), let the diagrammer learn by experience. Neither approach will be taken here, though both are reasonable future activities. In the first place, the basis on which to gather and characterize diagramming combinations is not yet clear (and is a time consuming operation); in the second place,

programs for learning by experience and example are still much under development, and are not the thrust of this research.

Instead, the simple-minded approach of sequential construction is taken in the CBD. The input to the program is a list of propositions like:

```
(relation1  var1  var2  ...  varn)
(relation2  var1  var2  ...  varn)
(relation3  var1  var2  ...  varn)
(relation4  var1  var2  ...  varn)
.
.
.
(relationn  var1  var2  ...  varn)
```

(Note that this list is illustrative -- in practice, each relationship does not relate all the same elements or variables.)

The program operates sequentially, so that after each proposition in the list a partial diagram is produced; at the end of the list, a final diagram is displayed. The layout procedure responds to each new relation by creating any elements referenced in the relation that don't already exist; assigning default values to all attributes in the new elements; then checking the predicate part of the relation and invoking the conflict resolution or constraint satisfaction procedures as necessary.

Conflict Resolution procedure and precedence of defaults

In the development of a diagram, some default values already assigned may require modifications for several reasons. In one case, default decisions are simply arbitrary values chosen from within a range; subsequent information renders the particular value chosen inappropriate, but doesn't modify the bounds of the range. In this case, another arbitrary choice may be made. For example, "element A outside element B" constitutes a constraint with a lower bound (minimum distance from center to center, for example), from which a default position must be chosen. If element A is then

constrained to be "near element C", which is already placed, the position of A must be reconsidered, but the bounds imposed by "outside" and "near" are not changed.

In another condition, new information refines the limits on such a constraint. If element A is outside B, that sets a lower limit; if later on A is "near B", that sets an upper limit, and again the default position may need to be reconsidered.

So the procedure for selecting a default value to satisfy a range, or inequality constraint, must be able to consider all the relevant constraints contributing to the bounds. In the case of both an upper and a lower limit, the procedure may reasonably take an average, although this often may require some further arbitrary commitment. If A is both "outside B" and "near B", even if these result in an average distance from center to center being chosen, A still may be located anywhere in a 360 degree range around B, and some decision still has to be made. Some heuristic such as "choose zero degrees or some minimum increment from the last placed such element" is required. In the case of only one bound (just "A outside B", e.g.), or in which one bound is implicitly infinite (as in this case), the decision is harder, since there is no meaningful average between upper and lower. In this event, some heuristic must be available to guide the default assignment, such as "add some minimum increment to a lower bound" (or subtract from an upper bound).

Another possibility for interactions of defaults is that two distinct constraints dictate different default values for a single variable. For example, if "A outside B", then some heuristic like the ones just mentioned could be used to assign a default position to A; if also "A outside C", then the same procedure would attempt to place A relative to C. Which should rule? A partial solution to this problem is to have each relation like "outside" "connects", etc. consist of two parts, a predicate (test) and a default solution (generator). The generator is only applied if the predicate fails, or if no value has yet been assigned. Then the generator must be governed by some rules about which values can be changed, and which are firm.

Another possibility causing a conflict is an actual contradiction or impossibility in the source design state, which may be flagrant or subtle. The conflict may arise from a topological impossibility (e.g. "five mutually adjacent regions"), or a logical absurdity ("a between b and c" .... "b between a and c") in the design state. (Of course, if a constraint manager has been at work on the consistency of this design state, such a conflict should never occur.) In this case, there can be no useful resolution of the conflict in the diagramming -- rather, encountering the conflict points the way back to the source, where the designer must resolve the problem by modifying the design state.

A more likely source of conflict is that one or more of the defaults assigned in the mapping process are no longer appropriate (for example, A and B both have diameter 20 by default, but then "A inside B" requires A smaller or B larger). In this case, there must be a mechanism for undoing or reassigning the conflicting default assignments.

The mechanism in CBD is to look for some variable value that might be changed (retracted and re-fixed) in order to resolve the conflict. Any variable value that was assigned by default is a candidate for retraction, with some limitations. One such limitation is any global constraint, such as one requiring a minimum size on all elements. In the last example above "A inside B", since both diameters are 20 by default, either one should be a candidate for undoing; but since changing A would require making its diameter less than 20, some alternative is preferred. Increasing the diameter of B is acceptable, and in this case is the only alternative. Were there several alternatives, the strategy would be to consider any rules, heuristics or constraints that would prefer one alternative to all the others, and choose it if found; or that would eliminate any of the candidates, and then choose arbitrarily among the remaining. One such heuristic would be to look ahead and see if the proposed change would result in any new conflict(s), and to prefer a solution that generates no new conflicts.

Some kind of look-ahead is a familiar strategy in programs such as chess- or checker-playing, and not too difficult to implement, but has the disadvantage of combinatorial explosion if the look-ahead is more than one or two steps or the variables (elements and relations in the diagram, e.g.) at all numerous. Were the problem one of critical decision-making (to avert disaster), or even a competitive goal (to win), a look-ahead strategy would be essential. But diagramming is neither critical nor competitive; the problem is only to make a decision, not the decision. So a strategy of choosing arbitrarily among alternatives, or of producing several variants, is acceptable. The CBD program only looks ahead one step, and only for a violation of the minimum size constraint. This is a simple-minded strategy, and produces simple minded results and occasional avoidable conflicts, but also works much of the time.

In the event of no suitable alternatives to over-ride , the diagrammer has only one option: to appeal to the user to either undo some fixed value or constraint, or to provide guidance in the resolution. As previously mentioned, there are usually few fixes in diagrams, and so the first situation rarely arises. Later on in the process of design development, as default values get modified by designer preference, this kind of conflict becomes more common and the user-choice of where and how to resolve conflicts becomes more important.

User Guidance

There are several situations in which the CBD needs user guidance in resolving a difficulty. The first, and simplest, is when there are several apparently equally acceptable alternatives to choose from in the course of the layout or conflict resolution. As a simple example, consider the first two steps of the Columbia diagram, simplified as "A is between B and C" and "A surrounds D". A,B, and C are all elements that get created the same (default) size, and the first relation causes them to be arranged in a straight line. When the second relation is encountered, the element D is created. The "surrounds" relation knows that the surrounding element must be the larger, so D is

made (some default amount) smaller than A. Then the "minimum size" constraint is applied, and D must be made larger, so A must be made correspondingly larger -- now A is so large as to intersect B and C, that it's supposed to be between. The "between" relation specifies that the element must be entirely between (not touching or overlapping) the others, which effectively means that the distance between B and C must be increased. This end can be accomplished by three means: moving B, moving C, or moving both B and C. Furthermore, "moving" can constitute movement in the x-axis, or y-axis, by either positive or negative amount: thus there are (three times four) twelve alternative fixes that will accomplish the goal and satisfy the constraints.

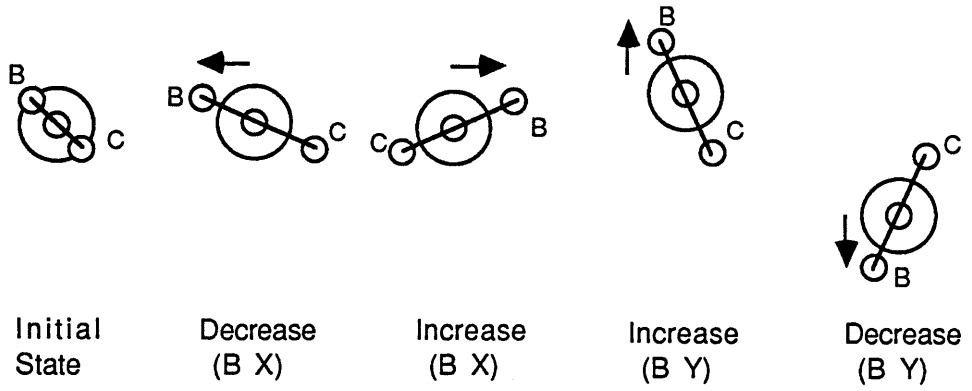


Figure 5.2 Four of the twelve possible alternative fixes

Since all these choices are translations or rotations in the plane, the net effect is similar for any of them, and with a corresponding shift in origin or surrounding frame, there is no difference at all between sets of three (moving the x-coordinate of B, or C, or both, for example, all result in congruent diagrams.) There is a difference in choosing the x- or y-coordinates, however: the difference between a horizontal or a diagonal line resulting. Previous discussion indicated that the diagrammer might prefer to maintain a horizontal line, *ceterus paribus*. But even this consideration doesn't much limit the choices: if the preference for a horizontal line is encoded as a constraint on the slope of the line, then one y-coordinate may be changed, and the other will be adjusted by the propagation of this horizontal constraint; so only one choice (the y-coordinate set by propagation) is eliminated from consideration.

In this and similar cases, a simple strategy is reasonable: arbitrarily choose one of the several competing choices. As described in chapter five, this approach may be made slightly smarter by 'looking ahead' one (or more) move(s), to see if the choice results in some violation of another constraint, and to choose another alternative if so. The approach might also be somewhat more efficient, if not smarter, by considering the number of subsequent relations triggered by each choice, and choose the one with the fewest consequences. (In the example above, if one of the end elements B or C had some set of elements or relations attached to it, then it would save some effort to move the other element.)

Alternatively, of course, the diagrammer can present the set of choices to the user, and let him decide. An advantage of this interactive approach is that the fact of being confronted with the choices may trigger in the designer an insight that results in a new constraint or proposition being added to the design data base. A disadvantage is that there may be many similar choices, and coming to an initial diagram will be slowed down by these possibly needless interactions. In all of the examples previously presented, the simple strategy of choosing the first from a set of alternatives was adopted.

The other extreme, another source of conflict, is rarely encountered in the initial diagramming stage, but as development proceeds, and after some user interventions, becomes more likely: the absence of any "soft" values to over-ride, or put another way, too many 'fixes'. Thus in the last example, if the designer had already moved the positions of B and C for some preference, then in order to increase the distance between them would require explicit permission from the designer (relaxation of one or the other 'fixes', or constraints).

A different source of conflict results when several interacting relations each have generator procedures that are each individually fine, but are mutually incompatible. The simplest example of this problem illustrates a situation that could be avoided by use of the approach of assigning

'interval' values. Assume the propositions "A above B" and " A below C". In the first place, if in addition "B above C", then this state is logically absurd; no arrangement satisfies all three. This is a conflict that cannot be detected by diagramming, although it can be detected by the impossibility of diagramming. Assume though that "B below C", but the distance between B and C is less than the minimum increment used by the "above" or "below" relation's generators. Then in response to the first relation, A will be placed the minimum increment above B; then the second relation will not be satisfied. If A is moved to the minimum increment below C, then the first relation will not be satisfied, and so on. The simple solution is to split the difference, and place A equally above B and below C. This amount cannot be foreseen by the generators, however, and must be computed on the spot. This requires intelligence that recognizes a range, or interval, on possible values for A (or its y-coordinate); and generator procedures that explicitly install interval values when appropriate. The present CBD doesn't utilize intervals, and so a problem of this type results in an endless loop.

Another situation that is troublesome is when there are really several generators needed, or one generator that needs to be variable behavior depending on circumstances. A simple example of this requirement is generators that must be able to respond to differences in geometry. A default procedure for locating one element "at the center" of another need not vary with geometry; but to place "at the periphery" or "inside" may well require different detailed procedures for rectangular or round, compact or elongated elements, for example. In the CBD, most generators assume elements represented by circles, and are subject to failure under other circumstances.

Relations like "peripheral" or "inside", for instance, pose another problem that may result in conflicts: they need to represent not numeric intervals, or ranges along one axis, but arbitrary selection of a point (or points) in a bounded area of 2-d space. The generators for such relations may contain knowledge that works well in isolation, but not in combination. For instance, the relation (along river house1 house2 house3) may locate multiple elements sequentially along a linear

element; (outside town-center house1 house2 house3) may locate elements outside a polygon -- see figures below:

Figure 5.3a
(along river house1 house2 house3)

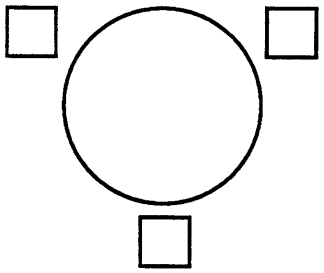


Figure 5.3b
(outside town-center house1 house2 house3)

But given the two specifications together, what should result? The CBD reports an insoluble conflict on this combination, since it can't adjust the "along" generator to avoid the inside of the center. However, any intelligent diagrammer (including the author) can suggest solutions, like the one shown below:

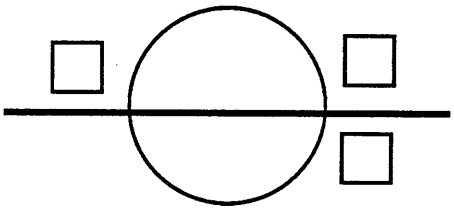


Figure 5.3c
(along river house1 house2 house3)
(outside town-center house1 house2 house3)

One way to describe the particular solution found is the insertion of a "gap" into the available solution space for the "along" relation -- the area inside the center circle is omitted. This might be described mathematically as a set of inequalities: given the line L with slope m and y-intercept b, the possible solutions for the "along" relation include

(x,y) : $y = mx + b$, $x<lx$, $x> rx$,

where lx and rx are the left and right edge, respectively, of the circle.

Alternatively, it is a modification of the "outside" relation, such that not only are the points outside the circle, they are "along" the line. The solution set from this point of view might be stated:

(x,y) | (sqrt (x^2 + y^2)) > r^2, and dist (L, (x,y)) < delta, where r is the radius of the circle, and delta is some threshold distance that describes "along".

However, these algebraic formulations constitute predicates, not generators. The problem remains of picking points that satisfy the criteria. This exactly describes, in microcosm, the general design problem of proposing a solution that satisfies some set of constraints. The approach taken to this problem in the CBD has three steps: First use the constraints in the most general terms to choose a 'default' generator procedure: the generator procedure for "along" has the form:

```
to place n objects along line L:
        let     dx = (x-distance L ) / n
                dy= (y-distance L) / n
                x0 = x-coordinate of start-pt of L
                y0 = y-coordinate of start-pt of L
        for i = 1 to n
          let   x= x0 + n*dx
                y= y0 + n*dy
                delta = max diameter of object i
                        fix the x,y coordinates of object i at (x+delta,  y+delta)
```

The "outside" relation has a similar generator. The next step is to check all predicates for satisfaction. In the present system, iff any predicate is unsatisfied, the diagrammer simply reports a conflict and requests user guidance to resolve it. This outside advice generally takes the form of a "correction" procedure. For instance, the "along" correction procedure is effectively "move the offending object closer to the line"; for the "outside" correction is effectively "move the offending object away from the center of the circle". These correction procedures combine, in this case, to produce the proposed solution shown above.

Such correction procedures are not presently included in the CBD lexicon, but they certainly could be, as extensions of the 'satisfier' procedures already attached to predicates. It seems likely that a more comprehensive diagrammer will have to have such abilities.

Given the observations above, the simple sequential approach to layout of interacting relationships is bound to be unsatisfactory in various ways -- it is the discovery of those ways and their reasons that provide the data upon which to base the research required to develop a general theory of diagraming and a more robust diagrammer. Some such examples are presented in the next chapter.

Chapter 6. Examples of Implementation & Exploration

This chapter illustrates the performance of the CBD in the production of diagrams from proposition lists. The diagrams so produced are analyzed with respect to their graphic qualities and inferential potential, especially by contrast and comparison to empirically observed diagrams. I examine a diagram that's not within the present scope of the CBD, and conclude with an analysis of the successes and failures of the experiments.

Exploration of Diagrams

This section illustrates the CBD producing diagrams. The problems of trying to construct the Columbia diagram (figure 2.1), and the Ideal Communist City (Figure 2.3) provide examples in which simple relations between individual parts combine to produce more complex wholes. The idea is to compare the results produced by a simple-minded diagrammer with the source diagrams produced by more complex diagrammers (people), to tease out the kinds of knowledge employed by the latter. Producing a replica is a useful enterprise in that it suggests the kinds of attributes, relations and variable values required in the diagram-state data structure to capture the appearance of the source diagrams, and so in the Columbia example I try to replicate the original appearance. After the initial diagram is produced by the CBD, each step required to replicate the appearance is related to some diagrammatic or domain-dependent knowledge. Replication is not the goal of a system like the CBD -- if the diagram already exists to replicate, there's no need for a program to do it -- but provides a test of the applicability and limitations of the diagram generation procedures.

Building Columbia

The Columbia diagram, as described earlier, came from a study of "New Towns", and was part of a display meant to enable a discussion of different examples [Chermayeff & Tzonis]. The input from which the diagram was generated is not recorded in the study (presumably it was a set of maps, or some of the design documents). For the purposes of the CBD, a proposition list must be presented as input. In this case, I prepare the proposition list by 'reading' the Columbia diagram, by trying to verbalize what seem to be the salient features of the diagram (supported by reading the

accompanying text, which is in part a discussion of the relationship of residences, places of work, and transportation systems). One possible reading[1] of the Columbia diagram results in the proposition list shown in table 6.1. These propositions are not exhaustive, nor are they the only possible reading, but they account for each of the elements in the diagram, and relate them one to another, in ways that are plausible given the context of urban design in which Columbia evolved.

1. *"a new town is located on the axis between two external metro magnets"*
2. *"the new town has a commercial center , located on the main axis"*
3. *"a main loop road surrounds the commercial center"*
4. *"several residential centers surround the commercial center, inside the loop road "*
5. *"a minor transit loop connects the residential centers"*
6. *"automotive links connect the residential centers and the loop road"*
7. *"an industrial center is located outside the ring road'*
8. *"an automotive link connects the ring road and the industrial center"*
9. *"pedestrian paths connect between the residential centers"*
10. *"pedestrian paths connect the pedestrian path network to the industrial center"*
11. *"several 'intra-urban' elements are located outside of the loop road"*

Table 6.1 Propositions for Columbia in Sentence Form

```
(inbetween looprd washdc balto)
(centred_in townctr looprd)
(ringbetween transloop townctr looprd)
(located-on resarea1 transloop)        (located-on resarea2 transloop)
(located-on resarea3 transloop)        (located-on resarea4 transloop)

(links l1 resarea1 looprd)      (links l3 resarea3 looprd)
(links l2 resarea2 looprd)      (links l4 resarea4 looprd)
(links l5 townctr transloop)

(outside ind-center looprd)
(links cl1 ind-center looprd)

(outside intra-urb1 looprd)     (outside intra-urb2 looprd)
(outside intra-urb3 looprd)     (outside intra-urb4 looprd)

(all-connected resarea1 resarea2 resarea3 resarea4)
```

Table 6.2 Propositions for Columbia in LISP form

[1]Note that for these purposes of diagram construction, I do not here analyze this process of 'reading', but take it as an ability of the eye/brain, informed by some domain knowledge. In the subsequent analysis, and in the next chapter, I examine this 'reading' of diagrams in more detail.

The propositions in table 6.1 have been translated (by the author, not by a computer program) into the relations in Table 6.2, that form the input to the computer diagrammer. A diagram is produced after each subsequent proposition, as described in chapter five. Figure 6.1 show the results of starting to apply the diagramming algorithm to the proposition list, one proposition at a time, using the primitive diagrams. In this example, to most fully expose the role of graphic properties in the production and interpretation of the diagrams, the simplest possible key has been assumed: all symbols are circles, all linewidths are 1, all colors are black, no symbols are filled-in. (Labels are added in these diagrams for legibility, not produced by CBD).

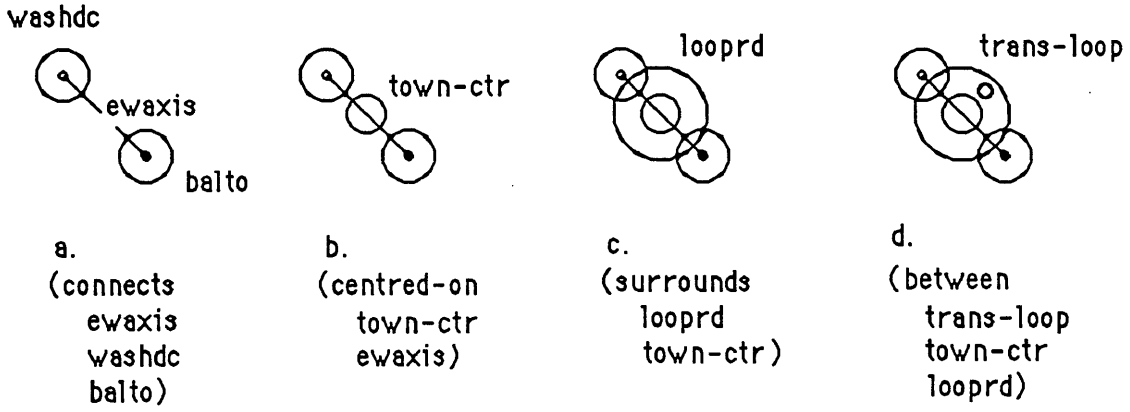


Figure 6.1 a-d The first four steps of diagramming Columbia

At figure d., note that the last element introduced, the trans-loop, is a small circle 10 pixels in diameter as a result of the relationship that assigns the size of the "between" element as a function of the distance between the two enclosing elements (town-ctr and loop-rd). For the purposes of legibility, a global constraint is applied to all elements: that they must be some minimum size (20 pixels in this case.) The size of trans-loop triggers a conflict in this global constraint (since 10 < 20), and a conflict resolution mechanism is invoked. The simplest approach is to try to set the size to the minimum, and propagate that change through all the other elements, maintaining all the relations entered so far. Just increasing the size of this element isn't sufficient, since it will overlap both the inner and the outer (town-ctr and loop-rd), so the size of the outer loop-rd must be increased also . Since all sizes in the diagram so far are defaults, and all arise from the original

(default) length of ewaxis, this propagation meets no resistance, and results in the expanding of the overall size of the diagram (first changing the diameter of the looprd element, and therefore of the distance between the washdc and balto elements.)

This strategy of preventing symbols from ever being less than some minimum diameter is sufficient, with the relations so far described in table E, to produce the default diagram shown in figure 6.2. (labels added for reference):

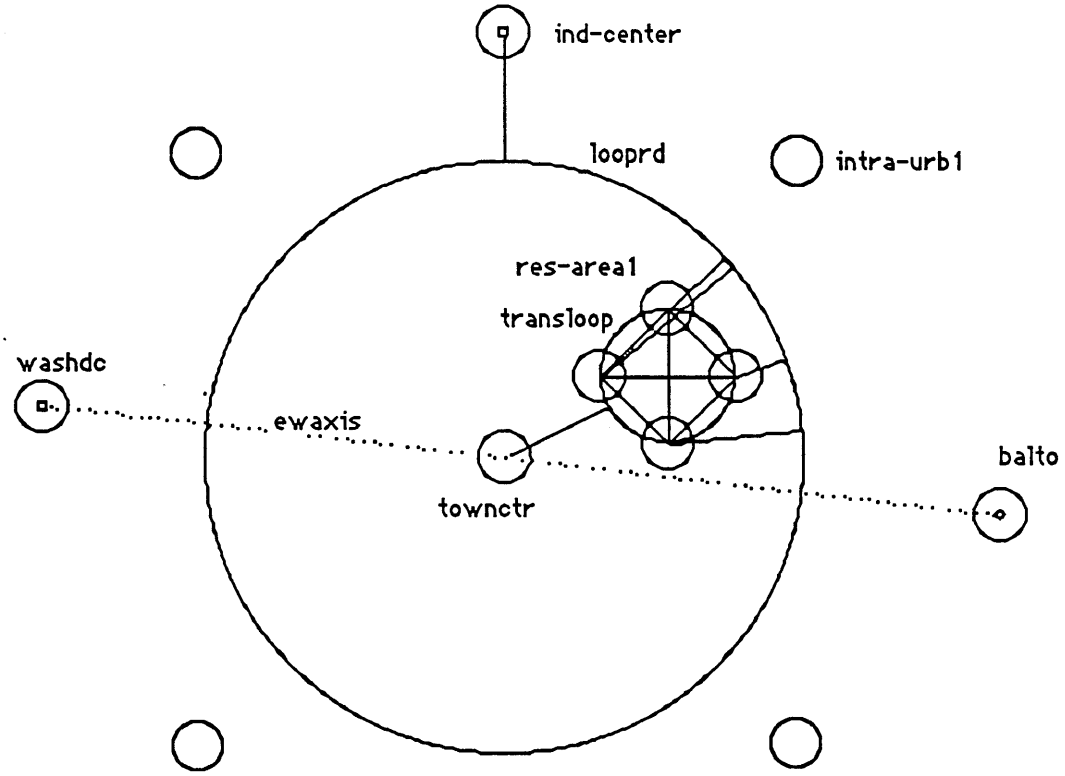
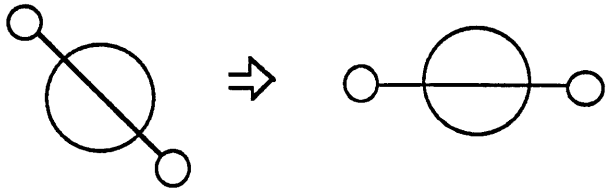


Fig. 6.2. Default Diagram of Columbia

This is a diagram of the propositions in Table E, using the basic relations and assumptions built into the diagrammer. Note that this diagram, for simplicity sake, only covers the 'top h ..' of the original diagram, since the original is symmetrical, the bottom half is just a repetition of the top. While this partial diagram is topologically similar to the original, there are obvious differences: there are no variations in shapes (no rectangles, or ellipses, or asterisk shapes, e.g.), no variations

in linewidth, no filled symbols, the central axis isn't horizontal, to name just the most obvious. How significant are these differences, for the inferential content of the diagram, and whence do they derive? To explore this question, I'll take the approach of applying modifications, one at a time, to the default diagram, and consider the source and consequences of each of the modifications.

The first step is a rotation, to make the main axis line horizontal. The default strategy in layout adds new elements at the lower right corner of the diagram, by assigning x- and y-coordinates of the center of the new element equal to the lower right bounds of the existing diagram plus some small increment (equal to the minimum size, 20 pixels) tends to produce diagonal lines in the absence of other information, as shown in the first step in diagram 6.1a. This diagonality reads as an imbalance -- by analogy to a set of balance scales, one side appears "heavier" than the other. Why should we seek analogy to a set of balance scales, in effect inferring a horizontal line where this is none? This might represent a common visual bias, deriving from our experience of the constant horizon in our normal visual field (perhaps dwellers in steep mountainous areas wouldn't share this bias?). It might also be cultural bias from a written language that reads horizontally left to right (perhaps Japanese diagrammers wouldn't share this bias?) This change, making the ewaxis horizontal, is primarily one based on visual comfort or convention; the original propositional input, that the axis connects its two endpoints, is not affected by the orientation of the line, nor is the relation that locates the looprd "inbetween" the two endpoints (magnets). From the propositional point of view, the two diagrams below are equivalent. The horizontal version is preferred for the reasons just given, and also because the horizontal line is more neutral than the diagonal one -- the diagonal carries an implication of significance (instability, or unequal weight) that the horizontal doesn't, a point I'll return to in chapter seven.



This change can be applied by the addition of a relation in the diagram state: the slope of the axis ewaxis equals zero, or equivalently, the y-coordinates of the centers of washdc and balto are equal: (= (EWAXIS SLOPE) 0) or (= (WASHDC (CENTERPT Y)) (BALTO (CENTERPT Y))). Which form the relation is expressed in has no visual significance; the resulting diagram is the same. The difference is a matter of emphasis -- whether we are concerned with attributes of the centerpoints of the external magnets, or of the connecting axis. Since the intent of this modification is to produce a horizontal line, the emphasis on the axis itself is preferable.

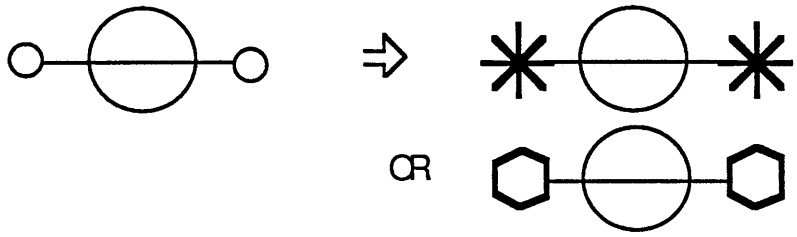
It's important to note that this is not a condition that derives from any stated, inferred, or intended relation between the source (design) elements; the condition is local to the diagram, and derives from diagrammatic rules, rather than knowledge about the domain or the state being diagrammed. This is the distinction between "How to diagram" and "What is diagrammed". This modification, like the ones that follow, is an adjustment to the diagram state that does not, and is not intended to, reflect back into the design state. The source of the adjustment is an inference, as it derives from looking, but is not an inference about the design -- it's a judgement of the diagram.

Another step in changing the appearance is to modify the symbol for magnet (washdc and balto), changing it from a generic circle to an asterisk (star) shape, bolder than the rest of the lines. A star-shape is a more appropriate symbol for several reasons: since two lines crossing provide a suggestion of a place, or node ("an X marks the spot"), multiple lines crossing reinforce the implication ("a star marks a special, notable spot.")[2] Furthermore, towns and settlements historically can be described as star-shaped, with centers defined by crossroads, and increased density at the center. The lineweight of the symbols is significant; single-thickness lines look too thin in an asterisk symbol, perhaps because there is not enough weight at the center, or hub. Since

[2]The literary conventions of asterisk, dagger, cross, etc. for footnotes presumably derive from this same association, to mark an important point.

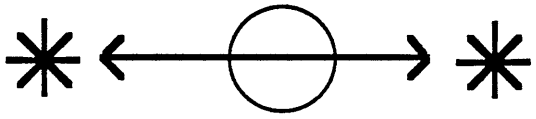
the magnets in this diagram are also "anchors" or "pins", they need the additional weight to "hold" the diagram between them.

These metaphorical terms -- anchor, pin, weight , hold -- are suggestive of knowledge that is hybrid between diagrammatic and domain; the judgment of proportion in graphical weight is essentially a visual one, like the preference for horizontal lines above, whereas the knowledge that external metropolitan magnets are centers of population and activity that hold the new town between them, is domain-knowledge that transcends the simple proposition "the new town is located between magnet-a and magnet-b". In fact of course, both Washington and Baltimore are larger in most metrics than the proposed new town, but this is a diagram of the new town structure, and not a map or a plan. So the external metropolitan magnets must be illustrated as significant entities that help define the structure of the central subject, but don't dominate it.



The particular choice of asterisk shapes is appropriate, and conventional, but not necessary. A larger repertoire of standard symbols (triangles, hexagons, asterisks, crosses, etc., for example) along with a mapping strategy (key) that assigned different symbols to different kinds of things, are the essential parts of this modification.

Another aspect of visual metaphor can be seen at work in the original diagram: the central axis is shown as a two-headed arrow, pointing at but not connected to the external metro magnets.



The fact that they are located at a distance away is suggested both by the arrows pointing to but not connected ("the magnet is in that direction, with a separation"), and by the size of the magnets.

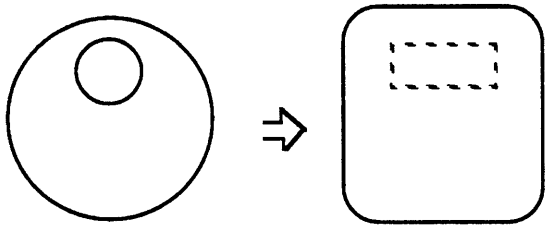
We know (from cultural knowledge outside of the diagram or the list of propositions) that Washington is bigger than the proposed new town; therefore, for Washington to appear that small size, it must be far away, by analogy to the common experience of apparent size diminishing with distance in perspective. Even though there's no hint of normal perspective cues in this two-dimensional diagram, the suggestion is there and helps our eye/brain make sense of the symbols and their attributes.

This modification could be taken care of by two additions to the lexicon: lines with arrow heads as special kinds of connectors; and a special form of the "connects" relation, perhaps called "influences", in which the connector doesn't touch the connected elements, but includes a gap of the standard minimum size. At the more pictorial, map-like extreme, in the light of the domain knowledge about actual distance between the magnets, this modification reflects the convention of using a broken line to indicate distances "not to scale".

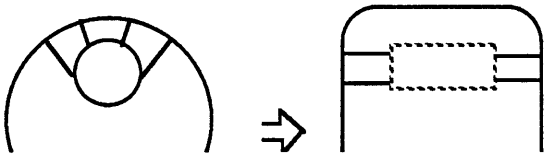
Two additional modifications are more literally analogous, based on domain-knowledge. The town center, central circle, is located along the main axis road, and since the road is a linear feature and development typically occurs along a linear system, the center will be elongated along the axis. The symbol therefore is made more oval. Both the town-ctr and the res-area's are filled in with a tone, to emphasize their character as regions, rather than than just boundaries, and to distinguish them from the loop road systems.

To additionally distinguish road-systems from center or residential areas, the two roads (loop-rd and trans-loop) are changed from circles to rectangles. The rectangles get rounded corners, because in fact road-systems tend not to have sharp right-angle corners. The trans-loop (public transit) is changed to a dashed line, to distinguish it from the solid line loop-rd (automotive). Not only is

there a categorical distinction between solid and dashed lines, while retaining a family resemblance (linear), there is an element of literal association behind this choice. Public transit lines may often be above- or below-grade, thus having less impact on the ground and on surface transportation[3].



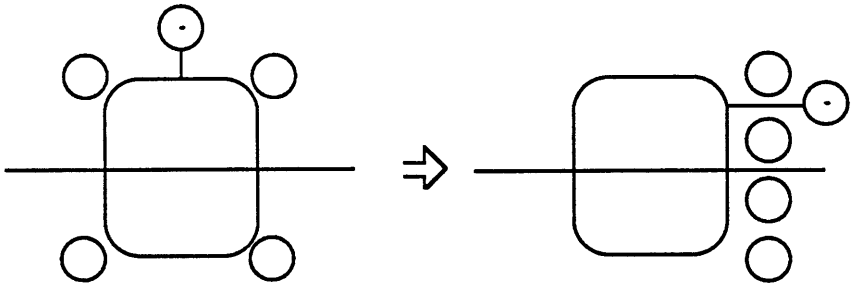
In the default diagram, the connecting link roads radially connect the res-areas to the loop-rd as a result of the default generator for connects, that connects two polygons between their centerpoints. With just one connector this is a reasonable strategy, but with several present, the result is a "corona", or "explosion", that seems to have "motion", inappropriate for connector roads. The modification here is to adjust these connectors to be horizontal. This regularity, helps to make the diagram visually "restful", just like the horizontal main axis above. Furthermore, the horizontal regularity emphasizes the relationship of the link-rd between the res-centers and the loop-rd, rather than the arbitrary relationship to the center of the diagram.



When the outlying industry-ctr is added to the diagram (outside looprd), by default it gets positioned at the top center of the diagram (this default is built into the outside relation.) Similarly, four "intra-urban components" are located outside the loop road, evenly distributed around the perimeter. To match the original diagram, both the ind-center and the intra-urb's must be moved to the side.

---

[3] The drafting convention of showing lines representing edges outside of the working plane as dashed instead of solid comes from the same reasoning.



One reason for, and inference from these moves, is that the industry and intra-urban components are not symmetrically distributed around both sides of the loop-rd; they are concentrated on one side only, possibly in some relation to the metro-magnet on that side. Which particular side they appear on is irrelevant in the diagram; in design development, a choice will have to be made. The diagrammatic decision is to have these elements on only one side rather than both; the design problem is to choose which side.

To accommodate this change, another lexicon element needs to be added "to-one-side"; it would either need to always choose one side, or have a random-choice operation to pick a side.

After all this, the lineweights of all the lines are increased by one increment -- as with the asterisks above, the single lines look too "thin", and need added weight. Diagrams typically contain fat lines, rather than thin, to help distinguish them from detailed working drawings. In the latter, thin lines are used to help specify exact dimensions and locations. In diagrams, the presence of fat lines helps reinforce the idea of approximate dimensions and locations. The intersection of two thin lines, as in a telescope crosshair, precisely indicates a point; the intersection of two fat lines indicates a region. The fatness of the lines is not a literal scaling, as it might be in a map or a plan, but an abstraction that helps to reinforce the diagrammatic nature of the positions and sizes indicated. Pedestrian links between the centers, however, are kept at the single line weight, to distinguish them from the heavier automotive links. In this diagram the thinnest line, the representation of a pedestrian path, is an indicator of the range of scales (the pedestrian is the smallest unit considered).

Finally, all the elements and relations from the original trans-loop through these last pedestrian links, are duplicated around the main axis, so that the diagram is symmetrical around the main axis. The resulting diagram is shown in figure 6.3 (compare to original default diagram in 6.2).

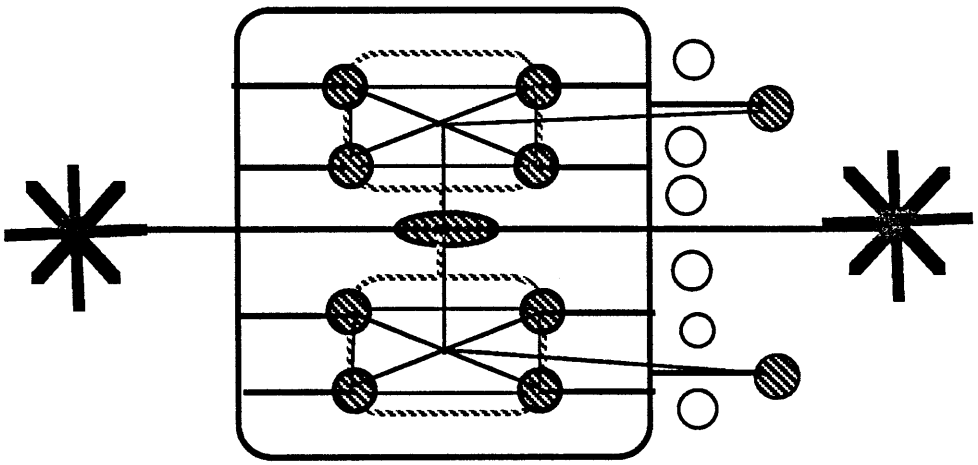


Figure 6.3 Complete Diagram of Columbia, with modifications

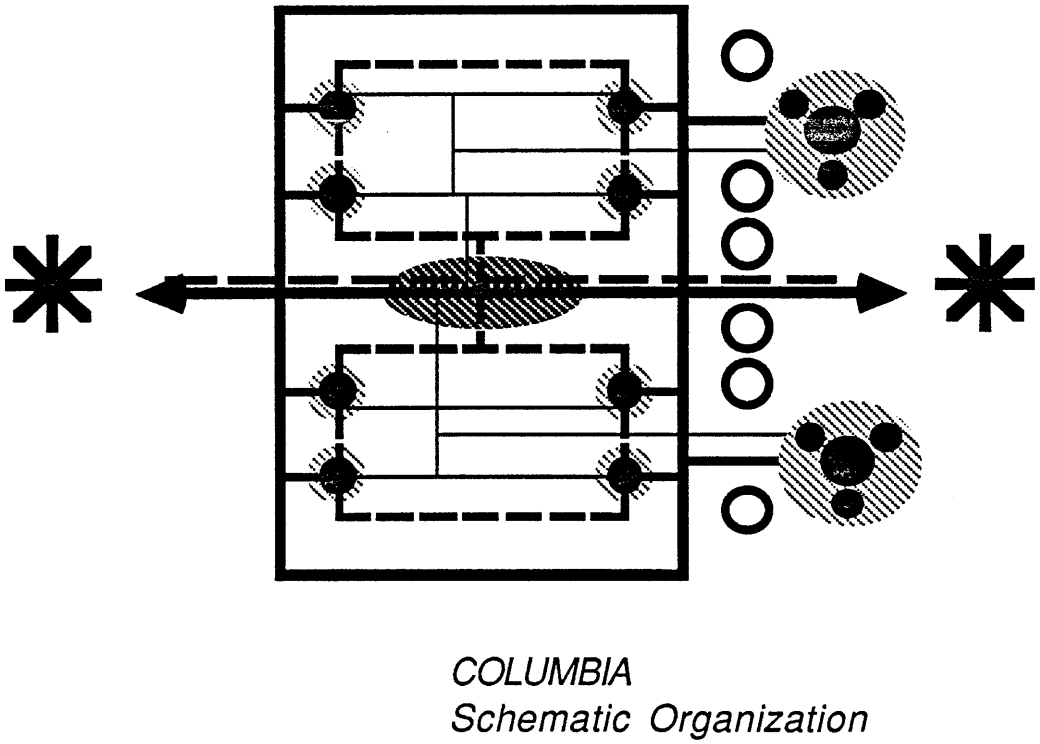
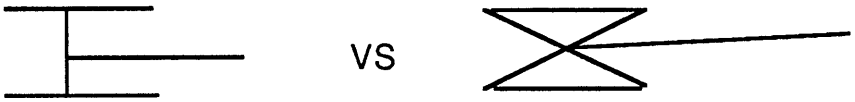


Figure 6.4 Original Columbia Diagram

Note that the result in figure 6.3 is still not identical to the source diagram (in figure 6.4) Some of the differences are omissions from the proposition list; the parallel transit line (dashed line along the main axis) in the original, for example, could be added by a new-relation "parallel-to" and the proposition: (parallel-to transit-axis ewaxis). This omission was made for simplicity, and could (should) be added. Some differences are cosmetic -- the actual proportion of the main rectangular element, or the style of shading, or the representation of the ind-center as a dense shaded figure with three nuclei inside it. Other differences are not just cosmetic, or related to surficial appearance; they are improvements, making the diagram more informative. Two significant examples of these are the rounded corners on the rectangular loop roads, and the interconnected network of pedestrian paths. The former is better because of the relation to the reality constraint of rounded road corners; the latter is better because it more truly represents the idea that the res-centers are all interconnected by pedestrian paths.



This arrangement of pedestrian paths perhaps falsely suggests that there is a central junction of all the paths between the residential areas, when this need not be so; but this suggestion is exactly of the kind that diagrams are produced to invoke. That is, the junction is noticeable in the diagram, and brings to the fore the question of how the pedestrian paths interact with each other: Is there in fact a central junction? Would this be a good place to put a map, a kiosk, a bulletin board? Does this junction take the form of a "rotary", with an open space (park, playfield, pond) inside of it? The diagram in fig. g raises a different set of suggestions: Are the centers shown as connected the only ones connected, and if so why? Are there multiple intersections and junction in this grid, and what treatment should they receive -- signs, kiosks, lighting, etc.? Both of the diagrams are informative and suggestive; I argue that figure f. is more provocative in its suggestions. This discussion emphasize the fact that there is no "correct" diagram, though there may be better and worse ones.

This example of the pedestrian paths suggests two more things about the diagrammer. The lexicon of elements needs to have a hierarchical set of linear elements -- connectors, influences, paths, main roads, transit lines, e.g. -- each with its own graphics properties. This is required in the vocabulary to express the elements being dealt with in the diagram. Second, there may be a computable inference in the example just considered. The individual paths combined to create a 'network', an element for consideration that wasn't present in the original input. This element has its own features (a shape and a center for example), and leads to its own inferences, as shown above. A component of the diagrammer responsible for detecting intersections and connections might detect (infer) this element, and even raise some of the questions. This possibility, so far reserved for the human observer, is taken up again in the conclusion in chapter eight.

In the above steps in the reconstruction of the diagram, we can distinguish three different kinds of necessary knowledge. The first is the knowledge of the domain we are diagramming in -- urban design in this case. For example, knowing that there is a fundamental distinction between "containers" to be represented as filled blobs (circles), and "connectors", to be represented as lines and loops, and that there are different kinds of connectors (pedestrian, transit, automotive), is essential. This knowledge is contained both in the declarations, in the filtering (this filtering is for containers and connectors), and in the chosen icons.

A second kind of knowledge is contained in the right side of the legend, that specifies the graphic devices available for the diagram -- solid lines, dashed lines, different fill patterns, and so on. The necessary knowledge here is of the capabilities and limitations of the graphic medium; what are primitives, what are available attributes. This is the same kind of knowledge that suggests a default strategy of placing new elements at the lower right corner of existing elements; a strategy based on the available resources of the diagrammer. Another approach would just be to place new

elements in a horizontal or vertical line, or on successive grid locations. Were a "find-space" algorithm available that could locate an empty space in the diagram, large enough for the new element, that algorithm could be substituted. There is nothing absolute or compelling about any of these -- they are artifacts of the chosen output medium. These are examples of diagrammatic attributes and rules.

The final kind of knowledge informs the above diagrammatic knowledge, but is distinct: knowledge about visual perception, that specifies a minimum size for all elements, so they are large enough to be seen and distinguished, for example. The observation that horizontal lines are "restful", or that asterisks made with heavy lines have more "weight", are also examples. They may be instantiated as diagrammatic rules, as they are discovered and verified, or they may be counted upon to arise as needed in the interaction between human and diagram-machine. Some of these rules are likely to be general and nearly universal (like horizontal lines and vertical symmetry, e.g.), others will depend very much on interactions in context.

For example, the reading of rectangular elements as buildings or structures and of round or curvilinear elements as natural features is reasonable and predictable in a diagram of urban form or development, based partly on our knowledge of this generalization in our culture. But in the diagram of Columbia, we read the rectangle as a road system, because of its scale, and the circles as developments, clusters of structures, because of their relation in the context of the road system. The large rectangle representing the loop road has an overall characteristic of "connected loop", setting up understanding of it as an enclosing and connecting device; but locally, any small piece of it is just like a linear element, as the rectangle is composed of lines. Making connections along the length (especially in the center of) any side is just like connecting to a line like the central axis; connections made right at the corners would have a different connotation, since they wouldn't be simple 'tee' junctions, and would emphasize the angles rather than the continuity. Thus the

reading of the loop road takes place at two different scales; at the global scale as a connected loop, at the local scale as a linear element. The distinction between these two scales is helped by the actual magnitude of the element, so that at the local scale the global structure, and the right-angle corners, can be ignored in visual inspection. Our visual system is capable of varying scales of attention, and the diagram supports these varying scales.

The Ideal Communist City

This example comes from a proposal for city form based on a set of socialist ideals from the late 1960's. The booklet "The Ideal Communist City" is full of discussion and argument about city planning, discussed previously in chapter 2. This examination takes two forms. The first is an attempt at construction of one of the diagrams, from a set of stated propositions that are extracted directly from the text. The characteristics of the initial diagram, like the Columbia diagram, are examined. Second, I examine a human-made diagram produced from a verbal description taken from the text, and use this diagram to discuss two related issues for the diagrammer: the translation from text to simple relational propositions, and the domain knowledge implicit in the program's vocabulary.

Propositional Content

The essential text in the chapter entitled "Structure of the Urban Environment" follows:

> "Our plans for NUS (New Urban Structure) illustrate no more than our general conception of its spatial arrangement...
>
> The fundamental principles governing the NUS:
> 1. Equal mobility for all. Residential Sectors are at equal walking distance from the center and the forests and parks surrounding them.
> 2. Pedestrian scale. No home is so remote from the center or from the park area that it cannot be reached by a reasonably short walk.
> 3. Elimination of danger from vehicular traffic. Rapid public transportation operates outside the pedestrian area yet is linked centrally with NUS. Its circuits carry people from home to work and from home to home.
> 4. Green belts. Every sector is surrounded on at least two sides by open land.
>
> ... Sectors are planned like spokes around the NUS center, which focusses attention on the large community park and the sociocultural center. Residential sectors are linked to each other ... by a system of public transportation. Motorized traffic circulates on a peripheral

> highway, except for a through road that takes it into the center. Roads branching from this one provide access to, but do not traverse, residential educational, or recreational areas.
>
> ... Placing the school community at the edge of a residential area allows the school grounds to connect directly with park and forest lands. ... Spaces between the residential sectors will be green, linking the trees and landscaping of the center to forests and parks outside the developed area... The residential sector is densely built-up ... surrounded by a green belt ..
>
> ... Our diagrams should be sufficient ... to show that we can really create an urban environment that meets individual and social needs..."
> pp 117-119

These statements consist of both statements of high level goals ("equal access" "green space" "elimination of danger") and proposed means of achieving these goals ("links, connects, do not traverse, edge, center", etc.). These means are 'diagrammatic' in that they include relations from the lexicon of the CBD, and specify approximate dimensions and topological and formal solutions, rather than particular dimensions, shapes or layouts.

One possible straightforward translation of these criteria into relational propositions is shown in table 6-3. This table assumes the same relations and categories as previously, in the Columbia example, with the exception of the new relation 'on-edge-of' (described below):

```
;; the highest level object is the developed area  NUS[4]
(surrounds  forest NUS)
;; the  NUS is surrounded by forest
(surrounds NUS  transit-loop)  (surrounds transit-loop cent-dist)
;; the transit loop is inside the NUS, surrounding the center
(contains cent-dist comm-ctr park)
;; the central district contains a community center and a park
(contains NUS   res-sector1 res-sector2 res-sector3 res-sector4 )
;; this just expands into: (inside NUS res-sector1), (inside NUS res-sector2) , etc.
;; arbitrarily include 4 residential areas -- "several" could be more or less
(between greensp1 res-sector1 res-sector2) (between greensp2 res-sector2 res-sector3)
(between greensp3 res-sector3 res-sector4) (between greensp4 res-sector4 res-sector1)
(on-edge-of  school res-sector1 forest)
;;  a school is between the residential area and  the forest -- just do one for this example
```

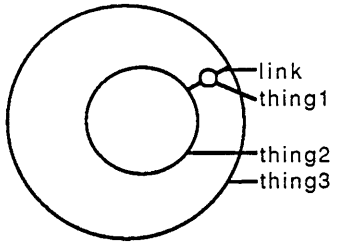
Table 6-3. List of Propositions for the Ideal Communist City in LISP form

---

[4] text preceded by semi-colons (;;) are taken to be comments by the LISP program, and are ignored.

The simple assertion "the school is at the edge of the residential sector" -- in particular, the outside edge, or edge nearest the forest, is not exactly captured by any of the relations in the CBD's simple vocabulary. So the new relation on-edge-of must be invented, defined as:

```
(defrelation
        on-edge-of (thing1 thing2 thing3)
              (centred-on          thing1
                           thing2-thing3-link))
```



that is, the "on-edge" thing1 is centred on the link between the two other things; a link is just the shortest line connecting the edges of the objects (see diagram above). This doesn't exactly capture the full meaning, but it may serve for the present purpose. With these propositions, the CBD produces an initial diagram of the NUS, shown below:

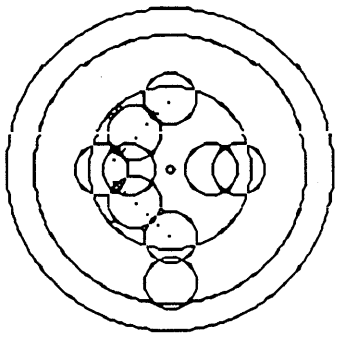


Fig. 6.5 Default Communist City Diagram

The sequence of construction is similar to that of Columbia; elements are added one at a time and position and layout are adjusted after each step. The principal problem illustrated here is the problem of overlapping elements. Although the CityCtr is said to be the center of the NUS, and the NUS contains the residential elements, there is no understanding in the simple-minded diagrammer that these separate elements shouldn't overlap (or at least, that the residential sectors should appear inside the CityCtr. They might indeed overlap, as an indication of the connection between elements. In fact it's not so much the overlap problem as the relative size of the CityCtr and the NUS. See the adjusted version below.)

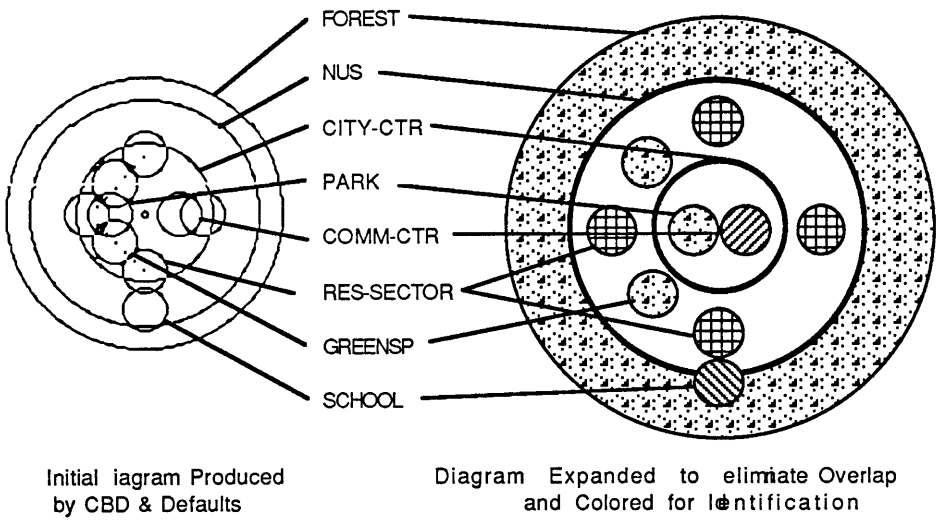


Figure 6.6 Modified Ideal Communist City
a. Initial Diagram b. Expanded & Colored

The initial diagram, figure 6.6a., and even figure 6.6b., expanded to eliminate overlap and colored with tones to identify elements, doesn't even superficially resemble the source diagram (Figure 2.3) That original diagram is made up of unique symbols, or icons (not just circles and lines), and also contains many structural/geometric/topological regularities not specified by the proposition list. For example, the arrangement of "school and sports area" as a hub with radiating spokes terminating in objects (activities?) of some sort is nowhere mentioned in the text; nor is the arrangement of residential sectors as rectangles.

The first of these is an example of filling-in information in the process of producing the diagram -- presumably the author/diagrammer had more ideas about schools than appears in the text. The second example is an illustration of exactly the kind of 'default' decision the diagrammer must make -- the text only says the residential sectors exist and are densely built up, not how they are arranged. The rectangular decision is related to the top-level decision to make a four-square symmetrical diagram, although even that doesn't preclude wedge-shaped or other arrangements. It's important to understand that the diagram is responsible for indicating the adjacency of the

residential areas to the schools, public transportation and green areas, but not to their shape or internal composition. The former are the (constitutive) diagrammatic constants; the latter are (contingent) variables that must be filled in.

Note also that in the original diagram no 'green space' is explicitly shown; in the CBD output, two circles represent the green spaces between the residential sectors. In the original, the 'green space' is taken to be the entire background, and all spaces between other uses. This is an example of domain knowledge: an attitude that green space is not 'something', like a house or a park, but is really understood as an attribute of background, all space that permeates the new city. (Note that this attitude about green space is contrary to the understanding of landscape architects and others who appreciate that green space is something to be designed and identified in its own right.) This attitude to the domain explains why the surrounding forest and the "large community park' don't appear in the original diagram, which is primarily concerned with structural elements (highways, transit, housing, schools). Seen in this light, it is a function of the 'filtering' part of the diagramming, in which those elements (forest, park, green space) are omitted.

It's not so much the resemblance of the diagrams that's at issue, its the inferential power. Do the two diagrams contain information that enable the viewer to reason about the proposal, to test the authors' assertion that their "...diagrams should be sufficient ... to show that we can really create an urban environment that meets individual and social needs"? This depends on what information we have been given about "individual and social needs", and what reasoning about them is enabled by the diagrams. In the text in question, considerable discussion of these needs precedes the diagrams, and the diagram is constructed from a translation of these needs into specific formal concepts, as described above. Then the diagram is presented by the authors as a kind of 'proof', that there is at least one physical manifestation of the principles they have propounded. That the environment so created will meet any needs is an inference that follows indirectly: "Since the

principles described are designed to meet [individual and social] needs, and since this diagram follows from these principles, then the meeting of needs will follow from the diagram (and any particular implementation of the diagram)." Without the supporting arguments and text, the diagram would not at all serve to show that any social and individual needs could be met; the diagram might show that certain uses could indeed be located adjacent to or contained in others. The diagram enables the viewer to reason about the question, but isn't itself a proof.

A final example of the NUS, for comparison, was produced by a moderately experienced landscape architect, given just the text quoted above and asked to "diagram the structure of the NUS." In this diagram, a facsimile reproduced below, the main elements appear labelled: forest, park, green space, residential sector, school, community center, public transportation, and highway. This diagram is most similar to the first, in being concentric and radially symmetrical, with a prominent center. Unlike the previous examples, this diagram indicates the peripheral highway and road system. The authors of the original study had a clear bias against highway automotive circulation, and in favor of public rapid transport, and so the highway is barely visible in their diagrams; the designer who drew this last diagram comes from a culture in which automotive transportation is dominant. The structure of the road system is taken literally from the text: "a peripheral highway, with a through road that takes it into the center, and roads branching from this one providing access to, but not traversing, residential areas".

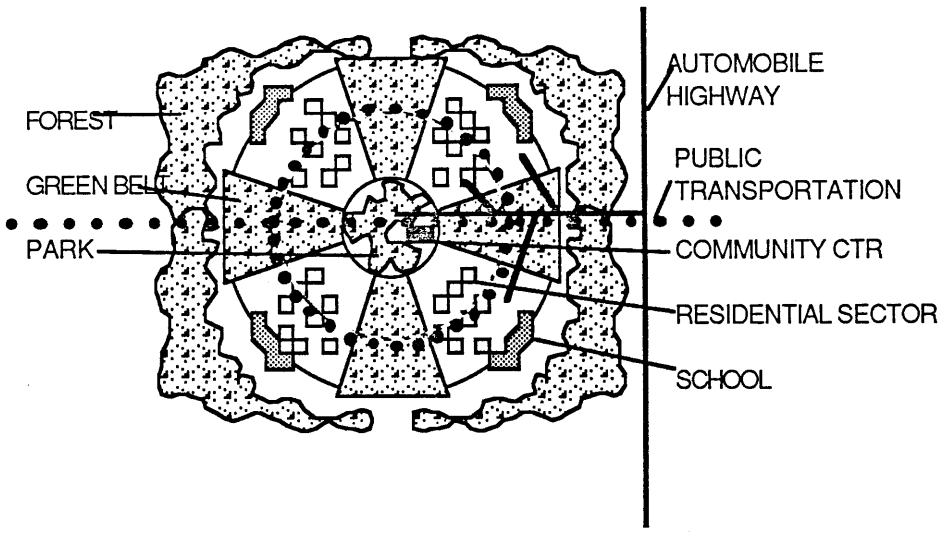


Figure 6.7 Alternative Ideal Communist City Diagram by Landscape Architect

What knowledge and what procedures would be necessary for the CBD to produce diagrams like this last one? Again, an approach to answering this question is to consider the transformations required to modify the initial diagram (figure 6.7), thereby revealing the attributes involved, and some possible reasons for them. I'll take the last diagram as the example, and show how the differences can be understood as the difference between A-diagram and B-diagram.

Starting with the 'expanded and colored' version of the CBD's initial NUS diagram (fig 6.6), there are three obvious modifications required right away. The 'green belts' between residential units become wedge-shaped, extending into the central park and out into the surrounding forest. This could be considered just changing a symbol (trapezoid for circle), or refining the shape of the generic polygon (circle to trapezoid), but the significance of the wedge-like trapezoid is more than simple shape. The textual input described the green space not just between the residential centers, but 'linking' the center and the forest. So the green spaces take on the character of linear elements, but they are 'spaces', not 'paths', and so they are larger and wider than a linear element. Since they radiate, there is more available space at the periphery than in the center, so the green spaces widen out nearer the periphery to maintain a proportional share of the circumference. Metaphorically, this designer may have been thinking of the green spaces as 'invading' from the

forest, driving a 'wedge' of green into densely built up residential areas, adding reason for this shape.

Another change involves giving the irregular outline to the surrounding forest, changing it from a perfect circle. This change is mimetic, reflecting the rough-edged, unkempt nature of the forest, and distinguishing it from the built elements in the diagram. This distinction is further made by contrast to the regular, smooth edged shape of the green spaces (trapezoids), which are suggested as more man-made or managed than the forest. The park in the center is also an irregular blob, not a perfect circle; again this change can be considered as a matter of contrast, the injection of a natural element into the densely built up central area. The residential areas (circles) are expanded and represented as clusters of rectangles, indicating the nature of the district as containing individual structures. The school representations are also given irregular outlines, distinguishing them from the residential rectangles.

All of these last modifications are in the nature of 'fitting' -- disaggregating and detailing the elements in the diagram, but not so far as to indicate actual plan-like details, only symbolic, suggestive diagrammatic details. Residential districts contain individual structures, though not the number, size or shape shown; school buildings are larger and more complicated than residential structures, though not the size or shape shown; and so on. In this light, the human-produced diagrams of the NUS are more B-diagrams; the one produced by the CBD is an A-diagram.

Multi-Centred Net Example

A final example explores a diagram from the same urban design study that contained the Columbia example, but this example is beyond the capabilities of the CBD. The diagram, reproduced in figure 6.8 below, accompanies a description of an idealized metropolitan form by Kevin Lynch, described as the "Multi-centred Net".

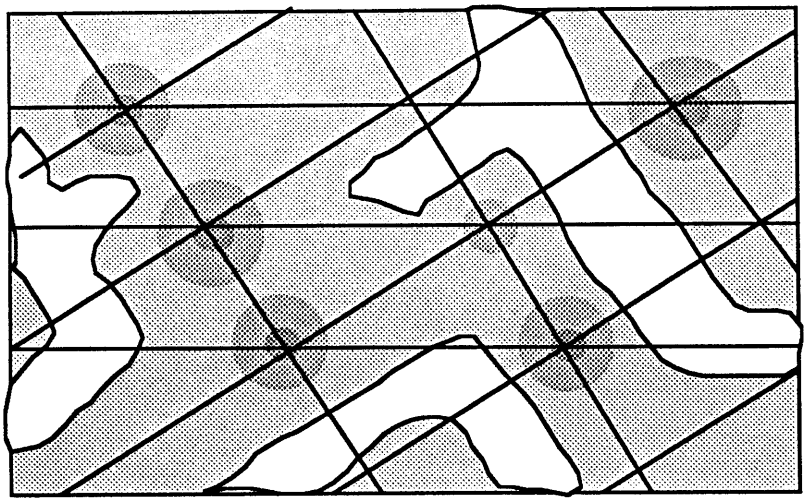


Figure 6.8 Multi-Centered Net (after Lynch)

The text contains a concise, and relatively vivid, verbal description of the pattern:

> Kevin Lynch
> Pattern of the Metropolis: The MultiCentered Net
>
> 1) "... a triangular grid pattern that grows at the edges and becomes more specialized in the interior"
> 2) "... densities [with] wide range and fine grain, with intensive peaks at junctions in the circulation system and with linear concentrations along major channels, but with extensive regions of low density inside the grid"
> 3) "... through the interstices of the network, belts and tongues of open land [forming] another kind of grid"
>
> Chermayeff p. 130

This description is full of terms like the ones in previous diagrams: grid pattern, network, channel, linear concentration, along, inside, regions, etc. But the lexicon of the CBD as it stands is adequate to describe or produce the diagram. A principal reason for this is that the multi-centered net is above all a description of the inter-relationships between members of sets: a set of circulation channels organized as a triangular net, a set of settlements with constraints of a statistical nature on their densities and their locations, and a set of open spaces distributed with respect to the circulation system. In the Columbia example, there were implicit set descriptions, that were finessed by substituting particular numbers (four, for example) and explicitly identifying the members of the set (res-area1, res-area2, etc.). Since the number of elements in the sets were few in the original example this was reasonable. But in this example, the sets are large, and a more

general mechanism for describing them is required. Along with this requirement is the requirement for relationships -- predicates and generators -- that are statistical or probabilistic in nature, describing for example a high standard deviation around a mean ("wide-range" and "fine grain"), or that describe correlations ("high densities located at nodes", e.g.)

The CBD as it stands is not up to these demands. The following imaginary description of the multi-centred net in relational terms is an example of "wishful thinking" that constitutes a requirement for the next version of the diagrammer.

```
(is-a-triangular-grid    "circ-system")
(for-some (set-members (circ-system link))
        (attribute "major"))

(set-of  settlements  "centr" (attribute density))
(wide  (range (set "centr" density)))
(fine  (grain (set "centr" density)))
(many  (set-members "centr" c (low (c density))))

(set-of  open-land "opensp")
(for-some (o  (set-members "opensp"))
        (or (type-of o "belt")
              (type-of o "tongue")))

(for-all   ( c  (set-member "centr"  (high (c density)))
        (or  (located-on   c  (select  node circ-system))
                    (located-along  c (select (link "major") circ-system))))

(for-all   (  c (set-member "centr" (low (c density)))
        (located-inside  c  circ-system)

(for-all  ( o  (set-member "opensp")
        (through-interstices  o  circ-system)))
```

Table 6.3 multi-centred-net example

This listing of relations should generate three different kinds of elements with attributes and relation son them: a circ-system, that is a triangular network of links, with some links having the attribute "major"; a set of centr's, nodes that have attribute density, such that the range of densities is wide, the grain is fine, and there are many members with low density; and a set of

opensp elements that are either "tongues "or "belts" (elongated polygons). Then the for-all statements describe the layout in more detail: high-density centers are either located at junctions in the circ-system, or along major channels; low-density centers are located inside the network (not on grid lines), and the open-sp elements are interconnected and distributed through the interstices of the net.

This description requires pre-defined types such as belt and tongue, that describe form "linear and peninsular, e.g.), and special procedures such as "through-interstices" that might be defined as "crossing links, but not nodes, of a network". Also required are translations of qualitative modifiers "wide", "fine", "many" "high" "low"; in each case some quantitative value must be chosen that is relative to the scale of the diagram and that is in scale with other such values (a wide range may be 0 - 1000, or 0 -10, e.g., which determines the values for "high" and "low"). The relations for-all and for-some are procedures that have both predicate and generator forms that check for or assign some attribute to members of a set.

Then the production of the diagram would require a mapping that produces icons for settlements such that the size of the icon is proportional to the density of the settlement, and always less than the spacing between nodes of the network, for example.

Summary

The two experiments with the CBD show that it is possible to attribute some of the structure of the urban design diagrams to their propositional content, and to produce diagrams from input made out of a (possibly very large) lexicon of elements and relations with expressed as data structures and algorithms, but that the propositional input cannot explain all of the apparent features of the diagrams, and the algorithms cannot anticipate all the interactions that arise between relations. Graphical features may (did) need to be added, and interactions of relations resolved, by human

intelligence as a result of visual inference. Some inferences relate just to the diagram as a graphic artifact, and what might be called visual comfort or preference, and result in changes to the diagram state; others are inferences about the design state referred to, and result in modifications to the propositional input.

The knowledge that seems to govern both kinds of inferences in the interpretation of diagrams, and therefore should go into their production, arises from three distinct sources: global, or general knowledge about the human visual system and visual cognition; domain-specific, or substantive knowledge about the domain in which the diagrammatic inference occurs; and combinations of these kinds of knowledge in diagrammatic conventions that govern the specific production of graphics. Some such knowledge is explicit: knowing that roads and residential districts are important in town planning helps us attach meaning to lines and circles. Other knowledge is almost entirely implicit: that we can distinguish edges and enclosure at all is what gives lines and circles their visual utility. Diagramming knowledge allows us to separate the contingent from the constitutive in reading a diagram, and to make inferences in the diagram, or in the diagrammed domain. Sometimes horizontal alignment in a diagram stands for some significant relation in a source state; sometimes it's just an artifact of a default layout procedure.

The modifications that were made by hand to the Columbia diagram, or envisioned in the case of the Ideal Communist City, suggest that the data structure provided by the CBD is sufficient to express the overt graphic properties of the sample diagrams considered, and that the lexicon of elements and relations needs to be extensive and user-modifiable to capture the meanings intended in designing. The example of the Multi-Centred Net recommends that the lexicon needs a fundamental ability to handle expressions involving sets of elements, and relations between sets.

That the data structure of the CBD, or some future CBD, is sufficient to capture the properties that govern the appearance of some diagrams is encouraging. But capturing the structure is only one half

of the problem of diagrams; understanding their function is the other. That function has been shown to be more subtle, more complicated, and less likely to be susceptible to algorithmic implementation, depending as it does on deep visual knowledge. The next, penultimate, chapter addresses the interaction of visual knowledge and design reasoning in the context of diagrammatic inference.

Chapter 7. Visual Knowledge and Diagrammatic Inference

The analysis of the preceding experiments with the CBD uncovered a number of isolated bits of visual knowledge that would seem to be important to the diagrammer. In this chapter I organize these bits by locating features of visual cognition relative to the principal diagrammatic purpose of encouraging and enabling inference.

Experiments with CBD showed that the process of translation from a proposition list to graphics must be guided by knowledge of visual cognitive phenomena, including gestalt perception and analogical inference, and diagrammatic techniques, including methods for combining relations, that contribute to the utility of the diagrams in reasoning processes. The analysis of the experiments in the preceding chapters uncovered a number of particular bits of such knowledge, each essential to a particular problem or diagram. It remains to try to locate these bits in a larger framework, and to relate visual knowledge to diagrammatic inference.

Inferring by Looking

Diagrams are for looking. One purpose is to communicate existing, or developed, knowledge -- working drawings for contractors, renderings for clients, diagrams for public meetings. These graphics are not meant to be ambiguous or have multiple meanings (except perhaps in the case of renderings which are meant to be evocative). Another common purpose of drawings, of all sorts, is for problem-solving, in which a particular question or set of questions is to be answered. Many scientific and engineering diagrams, including bar charts and line graphs, are used this way, as analog devices to pose and solve questions (about the relationship of one variable to another, for example). Finally, there are drawings, including diagrams, used in design exploration, development and change that serve as fuel for looking, but often, in contrast to the problem solving purpose, with purpose unknown. Surely designing requires communication and includes some problem-solving episodes, used to test particular questions or propositions, but in the large is not guided by a predetermined goal. Exploratory details, outline sketches, preliminary diagrams, are all made to be used and discarded.

The overriding problem facing any diagrammer (machine or human) is how best to exploit the designer's visual perceptual abilities and encourage inferences by control over map-like features of the diagram and the combination of graphic properties that give rise to other visual inferences. Some inferences -- the ones that arise automatically in visual perception -- are unavoidable, largely shared by most viewers, and need not be encouraged, though they can be highlighted. The combination of these with domain knowledge give rise to diagrammatic inferences, that are more variable and susceptible to encouragement. The first level of control required by the diagrammer is over basic features that influence legibility and scale of the diagram, and that can be partially algorithmically controlled by defaults; the second level of control is over the more variable features with inferential content, that are less likely to be captured algorithmically and therefore demand interaction with a viewer. Each of these levels of control was illustrated in the attempts to bring the CBD's default diagrams to match the original diagrams in the preceding chapter.

Mis-Matches

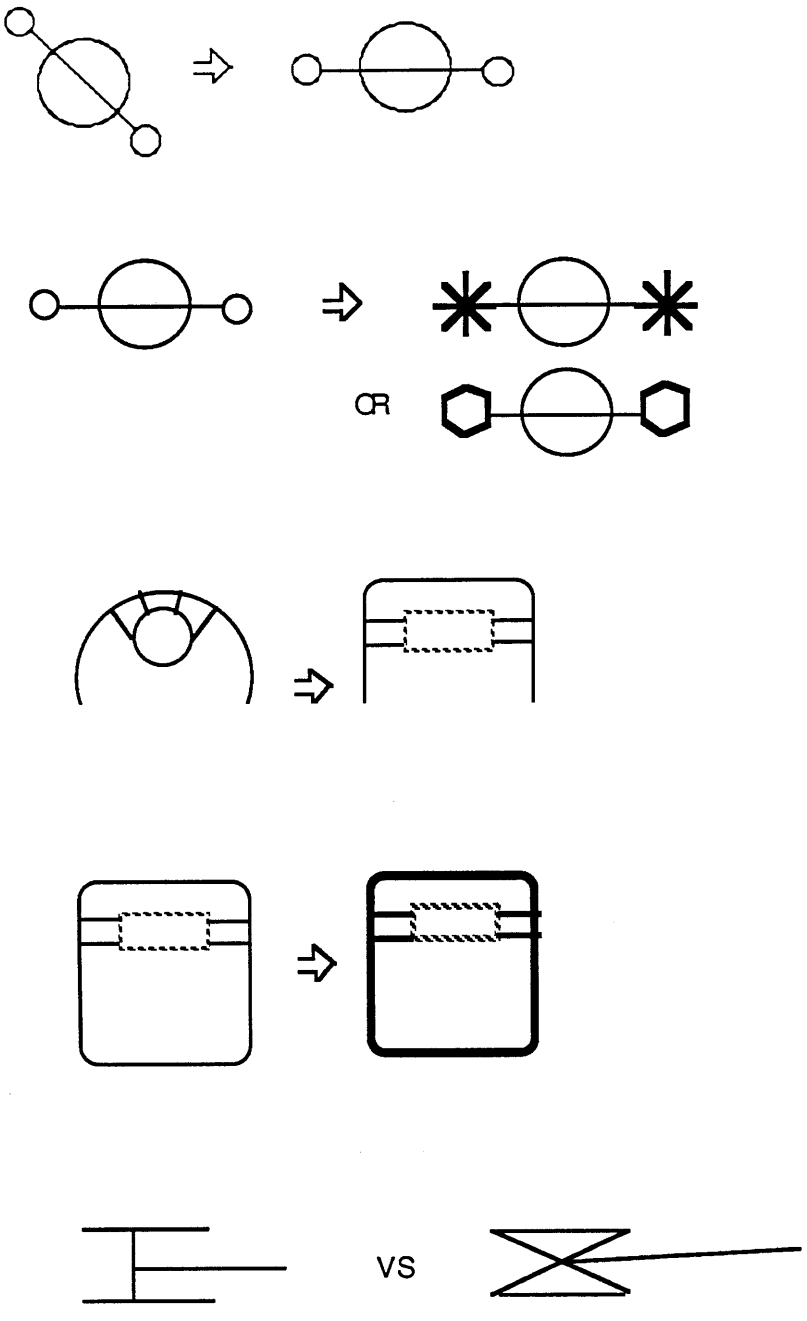
The example of Columbia was the most successful replication at last, but exposed a number of failures, or mis-matches between the source diagram and the default diagram produced by the CBD, five of which are collected in Figure 7.1 on the following page. For each one, I identify the source of the mismatch in the text at the right, and the location in the algorithm or in the CBD system where the mis-match could be accommodated.

The first mis-match discovered was the diagonal main axis in the CBD's diagram, produced by the default layout procedure for locating new objects, including line endpoints, thatplaced them in the lower right hand corner of the frame. I observed that a horizontal line could be produced by a different default strategy, or by a global constraint requiring horizontal symmetry. Such a constraint could be implicit in the default or layout procedures, or might be included in the original

proposition list. It was not included in the test case, because it was not 'read' as a salient feature, suggesting that the reader (author) didi not consciously attend to the horizontality, thereby implicitly expecting it to be included as a standard feature of layout. This horizontal symmetry was 'transparent' in the first reading, as it had nothing to do with the substantive domain knowledge or state referred to by the diagram.

This is an example of perceptual knowledge, or bias, at work, as it has to do with superficial aspects of the graphics and how they are perceived. Note that not all diagonal lines are read as unbalanced, or inappropriate, and a universal rule preventing them would be no improvement.

The next example is the circle produced as a symbol for the external metro magnets at either side of the diagram, instead of the asterisk symbols used in the original. In this case, the mis-match is just a result of an experimental attemt to see what was the minimum set of defaults that might work -- all elements were represented by circles initially. I observed that at the least the mapping algorithm might invent different symbols for elements with different names, presuming that a different name in the proposition list implied a substantive differrence between elements. A more adequate system would use a library of conventional terms and defaults, supplied as a key, before the diagramming started. This key would contain the symbols to be used, associated with the classes of elements, much as the original source document contained a key showing all symbols at the beginning. The question of why an asterisk is a particularly appropriate symbol (an icon for the intersection of streets in a city, suggesting density at its core) is not one for this thesis; the CBD can reasonably expect to be supplied with symbols, and must make provisions for replacing and modifying them upon inspection, but is not designed to be a device to invent appropriate icons (such an algorithm would surely be a difficult and intriguing one!)



Global Preference for Horizontal Layout & Symmetry
- in LAYOUT procedures and DEFAULT ASSIGNMENT

Requirement for Variety of Symbols & Iconic Shapes
- in Proposition List, LIBRARY & MAPPING procedures and DEFAULT ASSIGNMENT

Preference for Horizontal Similarity
- in LAYOUT procedures and DEFAULT ASSIGNMENT

Differentiation of Elements by Linestyle - Iconic Shapes
- in MAPPING procedures and DEFAULT ASSIGNMENT

Simpler Iconic Representation and Inference Rule
- in Proposition List, MAPPING & LAYOUT procedures and DEFAULT ASSIGNMENT

Figure 7.1 Mis-Matches in Columbia Diagram

The third example is the case of adjusting 'connecting links' from being radial to being horizontal, and changing the representation of the link road from circular to rectangular with rounded corners. I suggested thatthe first decision was essentially stylistic, neither literally representing the road layout nor based on any convention, but motivated by a visual judgement -- a sense of 'visual harmony -- making the lines parallel with the main axis. Such modifications are impossible to predict, but must be managed by interaction with the CBD, and provision for controls on graphic and other attributes of the elements in the diagram. Making the loop road rectangular is anothe stylistic decision, perhaps motivated by the rectangular frame of the page; the rounded corners I argued were a literal representation of the reality of road layout, and hence superior to the sharp right angle corners in the original.

The fourth example of mis-match/modification is the bold lines introduced for two purposes. One purpose was to distinguish two different types of road, "major " and "minor". This purpose could be accomodated by a lerger vocabulary of symbols and attributes, and the mapping of modifiers like major and minor into corresponding attributes. The choice of corresponding attributes may be arbitrary, or as in the case of bold lines, based on literal or metaphorical resemblance. The second motivation for the vold lines was the observation that diagrams tend to favor thick lines, to add to the sense of approximation, rather than "hairline" precision. This knowledge is global, about diagramming in general, and should be built into the basic defaults.

The last example above is the difference in the connecting pedestrian paths; 't-shaped' in the original, 'star-shaped' in the CBD as a result of the proposition "All the residential centers are inter-connected". This proposition resulted from the reading of the original; the author abstracted from the lines shown to a proposition, and the CBD implemented that proposition in a

starightforward way. I argued above that this arrangement is better than the original, more truly representing the proposition, but of course it more truly represents only the proposition extracted by the reader. What the original intent of the pedestrian paths was as shown, and whether they in fact had a more particular and less abstract reference, cannot be known. What is most important about this example is that the inference desired -- "connected" or "communicating" residential centers -- is actively encouraged by the diagrammatic layout.

The examples above illustrate the way in which 'high-level' inferences about the source or design domain are influenced and channeled by basic graphic features of the diagrams and of the human visual cognitive apparatus.

Visual Cognition

All purposes of diagrams (communications, problem solving and design development) take advantage of our human eye-brain perceptual and cognitive abilities, more than our ability to process logical productions. Many researchers and theorists describe the acts of looking, and seeing, as knowledge-based reasoning processes [Arnheim, Block, Marr, Pylyshyn] built on top of a base of purely physiological, sensorial abilities, like the perception of motion, colors, light and dark, edges and interiors [Livingstone]. The diagramming processes uses these abilities, but depend largely on the simplest and most vivid of them, such as the perception of global features before details [Navon] [Block].

Some of these abilities are described by the gestalt psychologists who characterized the process of seeing 'wholes' rather than parts, guided by the principles of "Pragnanz" (qualities including simplicity, regularity, stability, and "minimum energy"). "Wertheimer's Laws of Grouping" [Koffka] state that the human perceptual system can detect particularly well, and is especially prone to inferring, three fundamental characteristics in the input field: proximity, symmetry,

and continuity. These groupings seem to take place not only in visual but also in other perceptual modes including auditory [Bamberger and Schön].

These characteristics suggest some of the principal relations that will be detected in graphics, and that should be exploited by the diagrammer. Clusters and linear (including curvilinear) arrangements are fundamental relations that organize elements. In urban design diagrams these most often are taken as analogues to actual plan layout, or principles of layout, but in other kinds of diagrams these relations may be an analogue for some non-spatial attribute among the elements (a linear relationship indicating a common attribute of some sort, e. g.) The principle of continuity is one way of detecting relations that transcend proximity, as in linear or circular arrangements of elements for example; local continuity is a means of disambiguating line intersections and identifying separate elements out of an array-like representation [Block]. Conversely, discontinuity in the form of 'indentations' in outlines has been proposed as a fundamental percept that results in segmentation of images and hence the identification of elements [Richards]. These principles suggest criteria for choosing symbols and relations in the mapping and layout operations, and may also provide a basis for the development of algorithms to "read" diagrams.

Recent developments in information-processing approaches to vision have attempted to make specific and algorithmic definitions of these gestalt principles. For example, coding theory [Restle] and the "minimum information theory" proposals [Leeuwenberg] have been devised to both suggest a way to symbolically encode information from the visual field and to explain the preference in the perceptual system for 'simplicity'. These approaches treat visual information as retinal arrays co..verted to strings of symbols, and so are appropriate to computational approaches. The theory states that when given two possible encodings of a visual scene, the most concise one will be preferred, and also suggests a principle for diagram-reading. Contour tracing [Ullman], has been proposed as one fundamental capability of the eye-brain as a mechanism for detecting

clusters, and making inside/outside, foreground/background judgements. Another suggestion in the information-processing vein is [Attneave] that representation generated by the visual system maximizes certain properties of the things represented: equal-length, co-planarity, parallelism, collinearity, overlap/coincidence, connection, probability/familiarity; and that these regularities account for a variety of empirically demonstrated visual effects and illusions. These regularities, many of which are found in diagrams, provide a basic outline of the properties that the mapping and layout strategies should attempt to maximize.

A central organizing principle for diagrams that derives from visual perception is the importance of vertical symmetry: the eye 'prefers' vertical symmetry and detects very minor deviations from it [Rock]. Thus, a significant distinction in a design state may be mapped so that the relation causes the diagram to become vertically asymmetric; this asymmetry will be read distinctly. Figure B, below, for example, seems to represent "A is bigger than B" more clearly than figure A, in which the relation of alignment, (or "on top of") seems more evident. In each case, it is the relation relative to the vertical axis of symmetry that is most prominent.

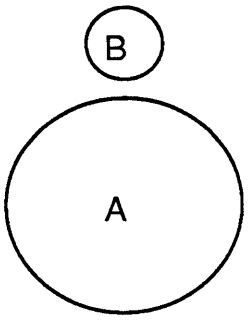


Figure A

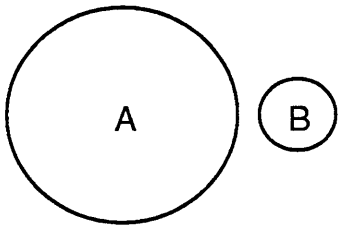


Figure B

The preference for vertical symmetry may be very deeply seated in our being, deriving from the fact that most animals and all faces we know are vertically symmetric.

In addition, it is one manifestation of a general principle in diagramming, that I call the "principle of minimum difference". This principle simply stated is: differences are more inferentially potent than similarities. Generally, any attribute that is common and consistent (equal-valued) between all elements in a diagram is not initially a part of its significance -- primarily deviations from that regularity are significant. Thus if a diagram is asymmetric, that fact is presumed to have significance; if it's symmetric, it's not specially noticeable. Similarly, if all lines in a diagram are medium-fat, except one that's thin and one that's extra-fat, it's the thinness and the extra-fatness that have significance.

The principle governing interpretation translates into a principle for diagram production: make no distinction, by graphic mark, property or relation, that doesn't have a correspondence or significance in the source domain. This is a conservative principle, that at its extreme produces simple diagrams like the ones produced by the CBD in the last chapter. On these meager frameworks more properties can be added in design development.

A corollary of this principle is "Regularity is more neutral than randomness. " When placing new elements in a diagram, absent any other relationships governing their position, there should be a regular principle governing their default locations (like the one the CBD uses, putting elements in the lower right corner.)

Grids are one example of regularity in diagrammatic representation. Diagrams often align elements on grids and show line segments at equal or modular lengths, when representing things that are in fact not located on grids, or are not equal length. This smoothing is appropriate when the spacing or the lengths are not salient to the argument, as when showing connections, or family memberships. The sign over the subway door, that lists the stops in order, shown evenly-spaced along a straight line, is a diagram, and not a map: distances between stops, and bends in the track,

have been filtered out, leaving only sequence and name as the salient characteristics. For this diagrammatic purpose a horizontal, vertical or diagonal line is appropriate, depending on the available wall-space or medium; a circular or compact rectangular form would not be.

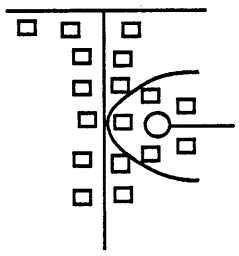
Legibility

Legibility of diagrams is a more of a judgement than a strict measurement. This criterion, basic as it is, is elusive, since the meaning of 'decipherable' is not specified, can only be illustrated at its extremes, and usually cannot be separated from interpretation. The legibility of text, or of a set of blueprints, can be defined as the extent to which each letter (or symbol, in general) can be identified by the reader as the one intended by the writer; legibility is thus distinguished from interpretation, which has to do with the meaning intended by the writer and received by the reader. Since there is no fixed convention governing what constitutes a symbol or its interpretation in a diagram, this measure cannot be generally applied. To the extent that an 'alphabet' of diagrammatic elements can be pre-specified, as by a key, then legibility can be defined as the extent to which each mark in the diagram can be unambiguously identified as a member of the alphabet. Although even this begs the question of what constitutes a mark, diagrams are, in general, drawings in which discrete marks are identifiable, and questions of legibility in this sense can be resolved. How discrete marks are grouped together to become symbols and graphic relationships, is a matter of interpretation and not legibility.

The production of diagrams must begin by obeisance to rules of basic legibility, however, and the diagrammer's procedures must reflect knowledge about inherent neurophysiological perceptual abilities, that influence superficial graphical qualities of the diagram -- lineweights, overlaps and intersections, variations in tone or angular measure, for example. As a minimum standard of legibility, a diagram must be neither so extensive nor so dense as to be indecipherable. The next concerns for legibility are to distinguish individual elements, and to highlight relationships and

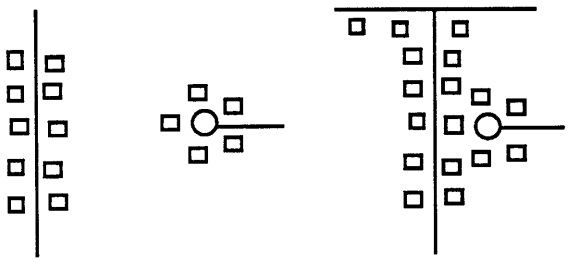
groupings. These ends are served primarily by two techniques: graphical highlighting in mapping and value assignment, as through variations in color, texture, intensity; and by layout, which controls the placing and spacing of elements in the diagram. The simplest examples of rules pertaining to legibility are intersection/overlap and minimum and maximum size rules; clearly, elements overlapping or coincident, too small to be seen, or too far apart to be related, will produce an illegible diagram. Of these, the first two (coincidence and minimum size) are easily judged and have relatively constant parameters; the latter (appropriate proximity or distance) is a gestalt effect related to the matter of scale.

The legibility of elements is more easily covered in this way than the legibility of relations. Since relations are often implicitly represented by diagrammatic arrangement, rather than explicitly represented by diagrammatic elements, their legibility is necessarily a matter of interpretation. To the extent that there are conventions, and more or less obvious or universal relations, like linear arrangements or clusters, they too are governed by fundamental visual visual abilities. Relations are more subject to interaction effects, and so may become obliterated or distorted in context. One method of clarifying relationships is to map them to explicit diagram elements, such as connecting or enclosing lines, or arrows. Consider as an example the following diagram, illustrating "social effects" of paths and proximity in residential layout [Lynch and Hack, p. 202]:



The phenomenon being illustrated is that one house is incorporated into two different groups, because of sharing access to two roads, one linear, one cul-de-sac. The accompanying text mentions "friendships are made along the street " and "a cul-de-sac will focus a neighborhood group", but

without specific reference to the marginal diagram. The noteworthy feature of this diagram is the introduction of the curved c-shaped line that surrounds the cluster of houses around the cul-de-sac, and emphasizes the fact that the one house along the street is captured by the group. The capturing line is necessary because the cul-de-sac cluster is not as strong as the linear one when they are combined, even though each is clearly perceived in isolation:



In the first two diagrams above, the two original relations/elements (linear street and cul-de-sac) are shown separately, and each is clearly perceived as a group. In the diagram at the right, like the original except without the curved line, the conflict between the linear street and the cul-de-sac is not so clear as in the original; there is a merging rather than a conflict, and a muddying of the intended effect (for example the appearance of the the symmetrical groups of three on either side of the cul-de-sac, merging into the street neighborhood.) The c-shaped line in the original is a diagrammatic element explicitly representing a relation, giving added emphasis to the cul-de-sac grouping, and to the "scooped out" or "captured" effect on the one house along the linear street.

Scale

Scale is a feature that governs both legibility and interpretation. Diagrams are intended to be to a large extent independent of scale. A diagram twice or half as large serves for almost all purposes, and the relative sizes of constituent elements are usually approximations permitting of significant variation without affecting the meaning or inferential potential of the diagram. Nonetheless, for legibility and interpretation, scale is an issue. Elements must be so sized that they can be distinguished, and still fit together (as in a puzzle). Elements too far apart or too small will not be read as belonging to a group, or a single diagram; elements to large or too close together will merge

indistinguishably together. "How far apart is too far apart, or too close together?" is not a question that can be answered with any precision, and is typical of the many questions that must be addressed by diagrammatic defaults. The circumstances and the interaction of other features -- color, intensity, shape, for example -- will influence the answer, and these answers will vary from person to person. The diagram layout procedure may be able to use reasonable defaults for many instances, but will always depend on interaction with a viewer or viewers as part of the production of diagrams in which these effects appear. The possibility of psychological research to establish upper and lower bounds on such values should be considered for future research.

Any graphic exists within a frame (a page, or portion of a page, or computer screen, etc.) This frame provides the first source of structure to a diagram; the area and proportions of the available space will control the placement of elements within it. In common practice the frame provides real guidance; I have suggested that the rectangular structure of the Columbia diagram, in particular the rectangular form of the element symbolizing the main loop road, derives simply from the standard rectangular format of the paper on which it was produced. An equivalent version of the diagram, showing a circular or elliptical loop road, can be read just as clearly.

Legibility is only a test that assures that something can be read -- not what is read. The extent to which the original propositional input is manifest in the diagram, or to which emergent elements and relations are inferred from the diagram, is another matter altogether. Given a minimum measure of legibility, the value of a diagram in designing then derives from its use in "diagrammatic inference".

Shape

Shape is an attribute of great concern to some designers, and an almost irrelevant one to diagrams. A large part of the process of design development is the process of giving shape to formal

organization. My thesis here is that organizational and topological criteria often come prior to shape, and that a final shape results from a fitting of the diagram to particular circumstances, such as site or culture. Sculptors and architects may make shape their sole concern, but at the risk of being charged with mannerism. Urban/environmental designers are less likely to be concerned with shape (as in outline, or contour), and more concerned with form (as in structure, organization). And yet, shape discrimination is one thing our eye is quite good at, detecting very minor variations; and whether or not we have names for them, or assign meaning to them, diagrams are full of shapes.

The uses and significance of shapes in diagrams are primarily four: as indicators of family or categorical resemblance (all rectangles or all triangles share some property, e. g.); as indicators of categorical or family differences (rectangles represent one category, triangles another, hexagons a third, and so on); as analogical indicators of some other salient property (rectangular shapes stand for man-made things, curvilinear for natural things, e. g.); and as purely stylistic devices in diagrams (choosing to produce a rectangular, triangular, or circular diagram of the same organizational concept, e. g. , simply because one has to choose some shape.)

The choice of shapes for the diagrammer is virtually unlimited within the bounds set out above. The characteristics identified in [Bongard] provide guidance in the selection of shapes and configurations that will be easily distinguished by eye; the matching of these shapes and configurations to intended relations and attributes is a matter of knowledge about the domain and the characteristics being portrayed (closure, compactness, continuity, e.g.)

The common use of circles, or blobs, as the default shape in diagrams is partly a response to the diagrammatic desire to avoid shape. In our experience, things that have no fixed shape, or that change shape readily (like water balloons or amoebas) tend to be rounded. As Arnheim observes:

"Roundness is chosen spontaneously and universally to represent something that has no shape, no definite shape, or all shapes. " [*Visual Thinking* p. 280] This may be because the most salient feature of a circle is its center point, and it can be interpreted as a region of probability around the center point. The ubiquitous bubble-diagram of designers derives from the same shape-avoiding tendency; the oft-cited danger of "simply building the diagram", or of just decorating it [Herdeg] is the danger of not proceeding far enough along the design process to give shape to diagrammatic form.

Style

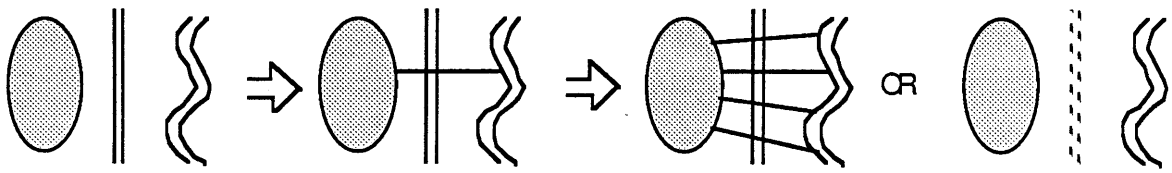
Stylistic differences between diagrams are, like font changes in text, not essential to inference. Graphical devices used for the purposes of legibility -- shape, texture, intensity -- may result in stylistic differences between diagrams that are equivalent in legibility and content. The decision to use shades of gray or black and white, lines or tones, solid lines or dashed, for example, are often questions of style rather than substance. How can these be distinguished?

According to the principle of minimum difference, any attribute that is common and consistent (equal-valued) between all elements cannot enter into the significance of a diagram, and can therefore be considered a stylistic attribute. Masking a solid line diagram with a half-tone filter, for example, will not effect its legibility (unless of course the original contains detail that is lost in the mechanical process.) A diagram mixed between solids and tones, on the other hand, will lose a potentially significant distinction if the half-tone filter is applied, obliterating the distinction. In this case, the decision to use a half tone filter is a style decision, but the distinction between two classes is a substantial one. There can be no theoretical determination *a-priori* whether a graphic distinction is substantive or not. The question is one of inference.

Diagrammatic Inference

In the context of designing, the phenomena of visual cognition must be understood with respect to three different kinds of inferences that proceed from diagrams -- map-like deductions that utilize literal translation from 2-d space to 2-d space (and so are best supported by 'B-diagrams'), map-inspired analogies that translate from 2-d spatial properties to analogous non-spatial properties, and other visual analogies that depend on associations with non-spatial visual properties.

I described diagrams before as being both catalyst and medium for inference. As catalyst, they can be viewed as the left-hand, or conditional side of logical productions (and often also as the right-hand, or consequence side.) The process of diagram construction is a sequential operation that can be understood as an inference -- a diagram is one logical consequence of the propositions. The process of inferring from the diagram uses the diagram as a condition: Since this diagram ("the waterfront is separated from the downtown by the expressway", e.g.) therefore that one ("A link between them" or "A series of pedestrian links crossing over or under the expressway", or "Remove the expressway!").



As medium, diagram graphics are a replacement for symbolic logical productions, in which the intrinsic properties of two-dimensional space and black and white line graphics are both the premises and the inference engine ("Straight lines drawn between the downtown and the waterfront intersect the expressway").

In the example just cited, both implicit and explicit knowledge are at work. A line between the downtown and the waterfront is not drawn in the first diagram, but can be inferred (unconsciously, because any two elements imply a line between them; consciously, because of a desire to connect

downtown with waterfront.) The line once imagined, or drawn, leads to more inferences. Since the line explicitly connects at one point to the downtown, and since the downtown element is an abstraction, implicitly containing multiple constituent elements, multiple possible connection points are inferred, hence multiple connecting lines. This happens because the downtown element is an aggregate structure, that can be expanded to its constituent elements. The expansion is only implied, and drawn in a limited way (the endpoints of the connecting lines), in the third diagram.

The intersection inference is an obvious one, a deduction that logically follows, and is inevitably perceived. The implication of the intersection (conflict, interference) is domain dependent, as are the conclusions (overpasses, underpasses) that follow.

The last alternative conclusion suggested above ("Remove the expressway") is a result of the detection of intersection and the application of a rule (to remove the conflict or interference, remove one of the intersecting elements). This diagrammatic inference, once made, connects again to domain-specific reasoning (desirability of cars in city, political considerations, and so on.) This is not a conclusion that depends on the diagram; it could presumably be produced by a logical production with a set of appropriate rules, but the diagrammatic medium encourages it.

Map-like deductions

The initial inferences above, the perception of a line, and its intersection with others, are examples of the most common inferences from diagrams: those relating to position, size, shape, proximity and arrangement in the 2-d plane. I refer to these as the map-like deductions: that element A is between C and D, that it is in the lower left corner, and so on. These are deductions because they can be completely specified and inferred from the rules of geometry, given the starting premises (the coordinates of the elements on the screen, e.g.) Like other logical deductions, there may be an indefinitely large number of them that follow from any set of premises (diagram), but

any one posed as a question, or predicate, can be unambiguously answered. One class of these deductions is the identification of emergent elements: from two elements, a connecting line is inferred; from two crossing lines, an angle is created, for example. The visual system detects these things, and much reasoning can proceed from them. [Funt] described a system in which 'diagrams' (really elevation projections) were used to detect potential collisions between objects in a robot's world, by translating and rotating elements of the diagram and detecting collisions in the diagram. [Larkin and Simon] analyzed the problem-solving involved in a geometric proof, and attributed much of the efficiency of solving the problem with a diagram to the map-like deductions that follow from constructing and viewing the diagram of the given elements. In their analysis, the inferences arrived at by observing the diagram were critical to simplifying the problem for the logical pattern matching machinery. [Lindsay] describes these kinds of deductions as arising from the "natural constraints" of the medium (line graphics in a 2-d plane).

These deductions are the raw-material for other diagrammatic inferences, and are not the end of line. Design inference may proceed from them through analogy.

Map-inspired analogies

The map-like deductions just described have an important feature -- the result, or conclusion of the deduction is a proposition, or element, in the same domain as the input. That is to say, lines, angles, shapes, distances, proximities are both the premises and the consequences of the deductions. From these map-like features, however, analogical associations and inferences may proceed: a square tilted 45 degrees looks "uneasy", or "like a baseball diamond". These results come from the map-like features, but result in a new domain, from which new reasoning flows ("If it's an uneasy situation, then tension-reducing steps are appropriate"; "A baseball diamond implies a place of energy and conflict", e.g.)

These analogies often follow from common metaphors that are inherent in our language and reasoning, for example "Up is More" [Lakoff & Johnson]. They may also be mapped explicitly or transformed through a key: since element A is above element B, it is more expensive (the key says "Up is more expensive"). These inferences have the effect of selectively modifying or overriding the deductions from which they derive. They may follow from learned experience in the domain, and the process of generalizing principles.

For example, "Seen as a diagram, design X consists of two separate parts connected by a third". Now, by analogy, all other designs with two parts are associated, and may be recalled, and lessons learned from them may be applied to the instant case. The connecting third element may be seen by analogy as a bridge, and lessons learned or rules about bridges can be applied: the element might constitute a bottleneck in flow between the other two parts, for example, or the importance of connections between the bridge and the elements bridged reinforced.

The determination of relevant analogues is a process that requires both syntax (structure) and semantics (meaning). Some approaches to finding analogies might be based on structural pattern matching [Gerstner], or syntax alone . [Holland, Holyoak, Nisbett and Thagard], in their research on induction and concept formation in scientific discovery, insist that syntax alone is insufficient, and propose a mechanism to allow the semantics and context of the current domain to modify analogical reasoning and suggest appropriate analogues, as well as what to do with them. A diagramming system that sought to replace or augment the human analogical reasoning would have to incorporate such techniques (the CBD makes no such claim.)

A benefit of the abstract nature of diagrams, and analogical reasoning, is the enabling of the process of 'seeing in different ways', or 'reinterpreting' design elements and relations. A change from one viewpoint, or interpretation, to another, can be understood as context switching. The

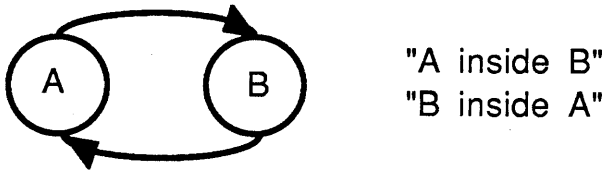
particular meaning, and viewpoint, as well as all the surrounding knowledge that supports or correlates that viewpoint, or interpretation, can be packaged together as a 'context'. Contexts, then, contain design intent, as well as design states. Switching contexts is a way of achieving multiple meanings of diagrams. Seen this way, diagram X suggests proposition P; seen another way, it suggests proposition Q.

[Schön] has described the ability and the tendency of the designer to "see as", in addition to simply "see", when interpreting design drawings. An important source of this metaphorical reasoning is selective perception of features or groups, as in seeing a rectangle either as "two L's joined", or "four line segments joined", or "a flattened square", "an open door"etc. This kind of perception maybe called "dynamic" instead of "static", as it seeks to link the perceived object into a chain of cause-and-effect and atmosphere, in which other causes, other effects, may be imagined. These chains may be captured by sequences of diagrams, that may share a common beginning or ending. These kinds of analogies, followed by inferences, are what I called above "map-inspired and other visual analogies". They are triggered by graphic features, but then other associations are made, and provide design development by referring back into the design state with new information. It is in this way that visual inference supports various "ways of seeing".

In a study of the role of sketching in design exploration, [Goldschmidt] distinguished diagrams from projections on the basis of their involvement in reasoning processes described as pictorial, or 'seeing as' (projections) and discursive, or 'seeing that' (diagrams). This is the distinction I have made between problem-solving with map-like deductions (seeing-that), and visual analogies (seeing-as). Unlike Goldschmidt, I argue that 'seeing-as' analogies may proceed from diagrams, and 'seeing-that' deductions from projects (maps and plans.)

Diagrammatic Experiments

When designing, diagrams can be considered experiments. In some cases a hypothesis is being tested; in others an exploration is being made, with no predetermined question. Since the inference or kind of inference desired is not known, there is no "correct" or "best" diagram. A diagram by itself provides fuel for visual inference about visual, formal, topological properties; in the context of an argument (which may be derived from the diagram), the reasoning can be applied to the domain through analogy. A diagram is correct or incorrect only in respect to some source state being referred to; it is good or bad with respect to inferential potential. What would it mean to produce a correct, but a bad, diagram? A correct diagram is one that identifies elements and relationships is a fixed manner; a bad diagram is one that doesn't help in inference. For example, if the relations being symbolized are not well chosen, then a reasonable appearing diagram might be produced from an unreasonable or impossible set of propositions. Mapping the "inside of" relation to a connecting arrow, for example, it's possible to diagram "A inside B" and "B inside A" as two elements with arrows between them -- a reasonable diagram of an impossible state.



A reasonable diagram of an unreasonable set of propositions

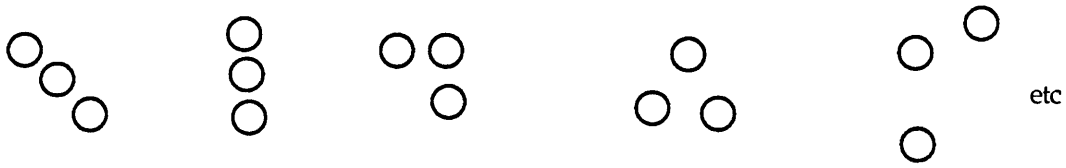
Conversely, using the "inside of" relation for the "visible-from" relation, it's impossible to diagram the reasonable set of propositions "A visible-from B" and "B visible-from A". One criterion for analogous relations used in diagrams, then, is that both are equally and similarly commutative and transitive and share similar inverses.

When designing, we don't know *a priori* what inferences are warranted, so there are no 'correct' inferences, and we can't tell, on their own, whether or not inferences are useful or productive; it's in combination, and in retrospect, that they are so judged. When illustrating a given source state, the minimum test is that all of the salient propositions should be visually obvious, and the point of

the argument should be inferrable, in the diagram; other inferences are undesirable to the extent that they contradict or complicate the argument being presented. There will always be visual inferences that are unintended or irrelevant. In designing, these unintended relations are often the point of the diagramming.

Emergent Relations

A major consideration for the diagrammer is that the resulting diagrams are continually subject to the problem of unintended consequences: attributes and relations that appear during layout. These are ubiquitous and unavoidable; any arrangement of elements likely has more than one possible reading. For example: given just the propositions "A exists", "B exists" and "C exists" and a mapping that specifies "elements $\Longrightarrow$ circles", the CBD is bound to produce either a straight line or a triangular relation, as shown below:



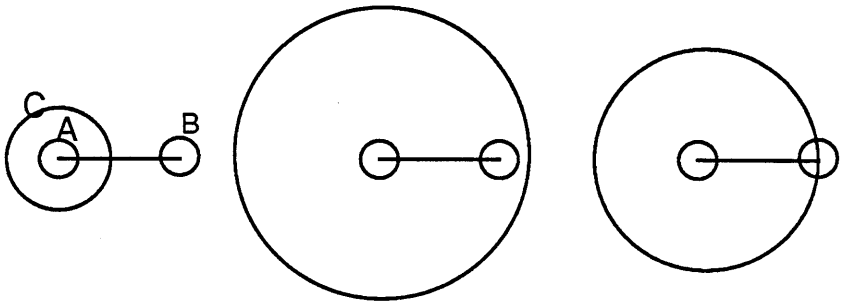
Any of the above diagrams potentially leads to some additional inference; each arrangement can be read as a relation connecting the three elements. One of the elements appears either "in between", "above" or "below" the others, or to be an "end" or a "corner" or an "outlier", for example.

This highlights two of the of the principal problems in diagraming: the appearance of unintentional but unavoidable relations, and the uncertain distinction between relations among elements versus attributes of a single element. Each relation potentially has implications on individual elements' attributes, and vice-versa (e.g. "straight line" relation among elements results in "central" and "end" attributes of elements). This complication is not fatal, but points to a task that is central in designing in the constraint model, namely identifying (and discovering) what's important. That is, the choice must be made how to state the design constraints on each

element, and group of elements, with the understanding that each statement will have impacts on the whole.

Ambiguous Relations and Interactions

To say "A smaller than B" and "B smaller than C" implies "An ordering on the three elements" and "A smaller than C". Neither, or both, of these implications may be important to the design(er). It is not appropriate to expect the designer, or the diagrammer, to seek all possible relations and implications, for the list may be very large. It might be an appropriate role for the diagrammer to suggest several alternatives; this is often the role of diagramming in designing. Suppose, for example, "A is linked to B", and "C surrounds A". What is the relation of B to C -- inside, outside, intersecting?



In the example shown above, three possibilities are suggested. If there is only one correct possibility -- and it is known -- then the result of the diagramming exercise is to refine the propositional input by correcting it, to include the additional information that makes the diagrammatic solution unambiguous. If the question arises in a design situation, and the answer is unknown, then the three possibilities are to be considered -- the diagrammer should produce all three, and offer them for consideration. This is an important function of the diagrammer in the design dialog: to produce variants and so allow assumptions to be checked.

It's important to observe that it is the function of the diagrammer to raise, and not to answer, questions. When a human designer makes a diagram concisely, apparently in one smooth flow, she

is asking and answering these questions in the process of visual/motor feedback. In fact, many false starts are made, and partial diagrams crumpled up and discarded, in exactly this process[1]. Perhaps many diagrams never get completed in the process of designing, for just starting them raises these questions and cause a backtracking or modification on the spot. Any effort to have the diagrammer anticipate each of these interactions could bog it down completely, either in internal decision making, or constant checking/questioning the designer.

There is related to this approach a potential combinatorial problem, for the question may well be asked "What is the relation between each element and every other element and combination of elements?" as well as "What is the relation between each relation and every other relation or combination of relations?" These kinds of questions are essential to the search-process of designing, and to some extent of problem-solving. In designing, exploration of the combinatorial possibilities is desirable, but in problem solving some means of limiting the combinatorial possibilities is desired. (The so-called "A-Star" search process [Winston] is one algorithmic approach to searching that attempts to evaluate the resources required versus the potential rewards of exploring any path.) The common (but not simple!) mechanism of "focussing attention" is one means of limiting search in visual inspection. Knowing what to look for -- in general terms, or in terms of how to look -- is a design skill that involves transformation of design knowledge in the source domain into diagramming knowledge.

Transformations of Design Knowledge

Diagrams, as I have placed them, are the result of a transformation from one representation to another; in one case (production of A-diagram) from linear, propositional form to 2-dimensional graphic form, in another case (production of B-diagrams) from complex 3-dimensional detail to 2-d graphics. Chapter six illustrated some of the subtle and not-so-subtle procedural problems of the

[1]So I claim from personal experience. Research documenting this phenomenon in both educational and practicing studios would be interesting.

first kind of transformation. I am concerned here with the advantages and other effects of the transformations between dimensions. The question of forms of representation, and their inferential and computational advantages and disadvantages, has received considerable attention from researchers in problem-solving [Simon and Newell], [Lindsay],[Larkin and Simon], vision [Richards], and cognitive science generally [Pylyshyn, Dennet]. One important theme in this research is the question of 'chunking', addressing the question of the size and structure of chunks of information used in memory and in problem-solving [Newell], and another is a debate about the respective roles of 'propositional' and 'pictorial' representations in mental processes such as memory, association and imagery [Pylyshyn, Kosslyn, Pinker, Lindsay]. The question of which representation most truly models the neural or cognitive machinery is not the aspect that concerns this thesis. The concern here is the inferential utility and computational requirements of the representations and transformations.

I've argued that the pictorial diagrammatic representation enables the designer to uncover two-dimensional analogs -- through map-like features -- and so is helpful in the design process concerned with spatial organization. Both [Larkin and Simon] (henceforth "L&S") and [Lindsay] described this feature of diagrammatic inference, and attributed a major inferential power to it: first, in the elucidation in the diagram of all the consequences of a set of original premises, what Lindsay called the "non-proof-procedural inference", and second, as a result, efficiency in problem-solving due to the physical adjacency of relevant and related facts, what L&S called the "spatial indexing" of information. The first phenomenon, the elucidation of consequences, appears "automatic" in contrast to proof procedures that are sequential and deterministic, but this just points out the sub-conscious level of perception that performs the reasoning for us in identifying features in the diagram. The second phenomenon, physical adjacency of related facts in diagrams, was true for L&S's chosen examples, but isn't always, as in the case of groupings made on the basis

of other graphic features than adjacency (color, shape, e.g.) that are equally responsible for the inferences that proceed from diagrams.

Both Lindsay and L&S were primarily concerned with the role of diagrams as surrogates for deductive inferences from a proposition set, and both asserted that the principal advantage of the diagrammatic representation was a matter of efficiency over the associated symbolic proof-procedures, not a matter of necessity. In particular, L&S only used visual inferences to augment the proposition set that their production system used to prove or answer the problem given it.

In both cases, the examples given were problems whose nature and whose solution depended solely on the constraints of two-dimensional geometry (that two intersecting lines would "produce" an angle, that a line crossing two parallel lines would produce two equal angles, for example) and not on any other overt graphic properties (lineweights, choice of symbols, e.g.). In the geometric diagrams, graphic lines represent truly abstract entities, absolutely straight with zero thickness and infinite extent, for example. The properties of the diagram are used to make these things visible, but the results are translated back into a non-graphic world of geometric entities and relations, and not in the direction of more graphics. Larkin and Simon also presented a problem of pulleys and ropes, but again the solution depends not on the thickness of type of ropes, or the size or shape of pulleys: it is entirely a network problem, where connections and transfers of forces are the variables. They claim in their analysis that "most people" draw a diagram with pulleys and vertical ropes, but the problem solving procedure they describe would work just as well on a electric circuit diagram of the system (much as thermal engineers use electrical circuit analogies to analyze thermal performance of structures.)

This problem-solving mode of diagrammatic inference is significantly different from the design use in this way: while the diagram serves as medium of inference, the results of the inference are

translated out of the two-dimensional realm (for example, into an assertion of "TRUE" or "FALSE", or the calculation of a value, or a new proposition) and as a result the choice of graphic representations is relatively arbitrary and simplified. For the diagrammer in this case there are no analogical or metaphorical qualities to consider, and there is no danger of "building the diagram".

In the problem-solving mode, especially of geometric or network problems, the assignment of defaults is trivial, since the criteria for "good" are relatively simple. The CBD is capable of producing a good diagram for solving the pulley problem analyzed by L&S with a meager set of defaults and no user intervention, since the most important criterion in this problem is to show connectivity. The choice of straight line segments and circles as the universal defaults is a happy coincidence in this problem, but dotted lines and rectangles would serve the purpose just as well. This observation is true of other problem solving diagrams, bar charts, scatterplots, line graphs and pie charts for example, in which stylistic decisions (color, shading, texture, lineweight, e.g.) are ignored in the inferences (comparisons of area, or height, or angular measure, or detection of clusters or outliers) that the eye performs so well (see Appendix 1 for the application of the CBD to other kinds of diagrams.) Our inference-making machinery is robust and insensitive to such variations, although affected by physical details such as minimum resolution and perception of regularities like right angles: if the comparison shown in a chart is 91 degrees against 89 degrees, pie slices would not be the best choice of diagrammatic representation; bars next to each other (at sufficient scale) would be more distinguishable.

The designing diagrammer, on the other hand, is not just testing the truth of a hypothesis, or calculating a value, but heading for a final graphic representation on the one hand, and fishing for associations and unpredicted observations on the other. The fact of heading for a graphic product adds significance to each overt graphic property and default, that the problem solver doesn't have

to reckon with. The fishing for associations adds significance to the diagram's graphic properties, since the result will be part of an ongoing process of visual stimulation and evaluation. The qualifications for a good diagram of urban form are not simply connectivity, adjacency, and type distinctions, as the examples in chapter six demonstrated, but include on the one hand sufficient literal properties to serve as a graphic skeleton, and on the other hand the ill-defined qualities of gestalt 'good form'.

Transformations of Inference Rules

The process of diagramming is not just a transformation of propositions into a graphic artifact, but also the transformation of domain knowledge in the form of rules and preferences in the source realm, into rules of inference applicable in the 2-dimensional realm. L&S observed that part of the efficiency in solving the pulley problem with a diagram resulted from the packaging, or compiling, of first principles (in their case Newton's first law, and the assertion that the system is in equilibrium) into diagram-specific rules (that tension must be equal on both sides of a rope passing over a pulley, so that the two lines connecting into a pulley object have the same value). They observed that a part of expertise that comes with learning may be this ability to cast first principles into operative rules specific to circumstances.

In the urban design domain, the packaging of domain knowledge and principles into diagrammatic rules of inference is equally important. Recall the design for an urban plaza from chapter two (figs 2-8 and 2-9), that showed a progression from a simple diagram of a few elements into a schematic plan, based on some explicit rules of inference and some inexplicable design moves. The three explicit design rules stated were:

1.) "IF A PATH INTERSECTS A DISTRICT THEN A PORTAL RESULTS",
2.) "IF TWO PATHS INTERSECT THEN A JUNCTION RESULTS",
3.) "IF A MAJOR PATH AND AN OPEN SPACE ARE ADJACENT, THEN A SCREEN IS REQUIRED BETWEEN THEM"

Each of these rules contains design knowledge, that derives from knowledge of perception, behavior, kinesthetics, physics and symbolism, at least. Each may be simply memorized as dogma by students and practitioners, or may arise after years of experience and be stored embedded in a network of examples and memories (memorable paths, portals and screens, for example.) Such rules may be derived from first principles, or may be empirically observed, or may be arbitrarily asserted, as a design preference or belief. Rule 3, for example, may be one specific instance of a set of principles about the interaction of community and privacy that are considered fundamental to urban spaces; or it may be an experimental attitude. These 'rules' are not universal, nor inviolate, but they provide stimulus to design development, in the form of experimental or conventional moves that can be explored. In the example scenario in chapter two, the rules were applied and the results modified, expanded, aggregated, and ignored at the designer's discretion.

For the purposes of designing with the diagrams, each of the rules is stated so that it can be applied and interpreted in the two-dimensional medium. A path, or a portal, or a district, in its full-blown version is a multi-dimensional combination of temporal and perceptual experience and spatial structure; for diagrammatic purposes each is reduced to lines and icons. The simple intersection of line segments can be detected in the diagram unambiguously, and the existence of the intersection, and the resulting portal or junction, can be asserted. The appropriateness, and the details of the result are not specified, but they are now open to exploration. The packaging of complex elements into line segments and icons must be accompanied by the packaging of design knowledge into rules of inference applicable in the diagram, that lead on the one hand to two-dimensional spatial diagrammatic decisions (locus, orientation) and on the other hand point back into the richer source realm for associations and detailed development.

Rule 3, for example, collapses the complications of "screen" and "between" in the real world of three-dimensional space and multi-sensorial screens into the simple element "thick-line" or

"shaded rectangle" and the relation "located on the centerpoint of and perpendicular to the connecting axis". These defaults may lead to further literal graphical development (a hedge or a fence, so located and oriented), or they may be modified in detail (a change in elevation between two spaces), or they may be completely transformed or ignored. In the transformation back into the source realm, the graphical attributes may be transformed into other attributes, elements or relations (thickness may become acoustical opacity, or density of foliage, for example).

The diagrammer, then, must know about the rules that are to be translated to be used in the diagram, and match the chosen symbols and graphic relations to those translated rules. This knowledge may take the form of general principles about conventional inferences (e.g. "Line intersections and pattern densities are generally important") that affect the global mapping and layout strategies, or may be embedded in the definitions of individual lexicon elements. This latter approach suggests that the lexicon needs to be rewritten or modified for different purposes, not just for different content -- a possibility that seems likely and would help explain the observed variation of diagrams in the design world. [Simon] once suggested that the observed differences in 'style' among architectural designers might be attributable in part to the differing processes they used; similarly, a difference in style between diagrams might be explained by differences in the transformations of inference rules that accompany their production.

Making and Using Diagrams

Designers mostly diagram intuitively. While there are conventions they are often distorted or ignored. Design students and novices, perhaps too concerned with "finished" products, tend not to diagram, or not to diagram enough. Design critics and instructors, intent on conveying principles and systematics, use diagrams often, as pedagogic devices ("If we diagram what you've done here, then you can see the cause of your problems", e.g.) Principles of design -- avoid diagonal circulation through spaces, create nooks out of the main flow, terminate axial views, separate pedestrian and

automotive paths, create clusters rather than isolated instances, respond to principal axes, etc. -- are conveyed in diagrams and illustrated by examples. A conventional learned language of diagramming is valuable in this usage. Standard concepts in the domain (path, axis, spaces, flows, clusters, etc.) need to be identifiable without labels, to facilitate the dialog that is part of design communication and education.

On the other hand, idiosyncracy and customization is valuable to designers in conversation with themselves and their media. Freedom from convention helps invention, at the same time as convention provides a basis for invention. The standardization of representations captures some knowledge, transforms some rules of inference, but can't capture individual insight. [Saund] has argued in the context of shape representation and recognition that shape descriptions built out of universal primitives are inherently less effective in any particular domain than the development of special primitives that express the relevant and important knowledge in the domain. His examples were concerned with the inference of recognition, but the principle remains when concerned with other inferences and analogies.

Specialized representations, that correspond to personal insights about the structure of elements, the behavior of relations, and the possibilities of inference, can make diagrams more useful to the designer and the particular instance of designing. Examples of this principle can be found in Alexander's diagrams for a New Indian Village. As I observed in chapter two, the appearance of these diagrams is idiosyncratic and apparently inexplicable, except as personal preference, or as I observed there, 'looking good' to the designer. Now I can amend that evaluation, with the observation that the criteria for 'looking good' are not completely aesthetic, but contain elements

of intention and projected use that capture the designer's knowledge. That knowledge is both universal and personal, so diagrammatic representations must also be both[2].

A conventional diagramming language has its value in communications and problem-solving, but must be extended (and transcended) for designing (a corollary of the observation that poetry may abandon standard syntax.) One challenge in this extension is to find combinations of graphic properties, including shape, texture, and relations, that are evocative for the designer but not merely literal plan and material translations. Alexander's examples stand out again, as his diagrams contain elements that are suggestively plan-like, but that are sufficiently idiosyncratic that they are highly personal, and one presumes evocative.

The choice of these techniques of representations, to be more than mannerist, must be governed by the questions "What are devices likely to inspire inferences?" and "Which are basic to most reasoning and hence unchanging, which are unique and modifiable?" Alexander's diagram for example, uses the universal relation 'attached' and its simple graphic manifestation, but highly variable symbols for the elements and activities so related. I have proposed in the lexicon of the CBD that such topological relations as "attached", "centred", "outside", etc., and a set of graphic properties like thickness, radius, tone, etc. are basic foundations on which personal diagrammatic styles are built. The inferences I have called map-like deductions are the most likely to be universal, the associations that each designer brings are surely the most variable. The problem-solving and communications oriented diagrams, like bar-graphs and pie-charts depend on those most fundamental devices. The design diagrams like Alexander's New Indian Village or the Ideal

---

[2]Some of the criticism Alexander's later work, i.e. the patterns of the Pattern Language, has been directed at the confusion of personal for universal, and Alexander's implicit claim, or his readers' inference, that some highly personal interpretations are universal.

Communist City make more use of personalized devices for inferences including evaluation and further development.

The likelihood, or ability, of graphic devices to support or inspire inferences is a matter not particularly susceptible to prediction by theory, but more likely dependent on circumstance and preference. Certainly the purposes of geometric hypothesis testing, whether for simple adjacencies or more subtle questions of visibility or sequence, are more sensitive to layout decisions and regular lineweights than to choices of symbols or colors and tones. Open-ended explorations and wholistic evaluations of 'fit', by definition, may be sensitive to any variation along any parameter.

Using Diagrams in Search

I said in chapter three that designing with constraints is a kind of search, in which the specification of the searched-for solution is part of the search. Search is a confusing term in this context, for searching usually presupposes knowledge of what is being searched for. Based on knowledge of the object of the search, one decides possible/probable places to look, and continues until the specified object is found. In designing, as we have said, the desired end-result is rarely predefined. Rather, some shadowy outline or incomplete set of specifications starts the process, and by the time the designing is 'complete', a whole set of interconnected elements has been defined. In this view of designing as searching, discovery and invention are one and the same. We speak of having 'found a solution', highlighting an essential ambiguity in this search: was the design always 'out there', waiting to be found, or did it come into being only at the moment of discovery?

Lacking a way to resolve the metaphysical question, I prefer to use the terms 'search space' and 'design state' as equivalents, recognizing that any design specification, no matter how abstract or how how detailed, is at once both a 'final' state and a starting point. Since designing is a journey,

passing through many places, or states, if we record the path of the journey we can always see it in retrospect as a search leading to the final state, understanding that the state was not predefined, and that the stopping is usually not absolute. The answer to the question "Is the search over?" is always provisional. There is a list of constraints and goals that must be satisfied, but it is in the nature of designing that any new state may trigger new goals and constraints, either according to some rule-based process , or according to the designer's insight. In this way, the decision that a design is complete or final is more like Simon's 'satisficing' than a matter of true finality. A design, or design solution, is complete when it is good enough, not necessarily inevitable or unchangeable. Indeed, most designers recognize that designs continue to evolve while the complex processes of construction and occupation or utilization proceed.

In architectural design, more so than urban design, the final design state is usually quite specific, specifying details and dimensions with precision. Such a design state represents a single 'point' in a large multi-dimensional space. An urban design state is more likely to be an 'envelope', or multi-dimensional region, specifying the outer bounds and approximate locations of many dimensions, but allowing for latitude in final design ("A complex of mixed-use buildings, no more than six stories high, set back at least 100 feet from the major thoroughfare"). For the purposes, and within the socially agreed-upon scope, of the urban designer, this specification is the appropriate level of detail. While for the architect or electrical engineer, working drawings and precise dimensions are necessary, for the urban designer a diagram is often good enough. "A two-way spine with low-density uses along it, and contrasting high-density uses at either end" may be a final conclusion for the urban designer, and a starting point for the architect or site planner.

Any design representation, in this view, describes a space of design possibilities, (a 'feasible region', in the language of optimization.) The dimensions of that space can be defined by a set of constraints, specifying both the dimensions, and possible or prescribed values along those

dimensions. The precise form (verbal, visual, or algebraic, e.g.) or level of specificity, aggregation, abstraction, or confidence of constraints is not prescribed by the model. Nor is any particular method of refinement of specifications, or search through spaces, prescribed, nor is any method for choosing between competing alternatives or resolving conflicting constraints. These are open to the designer's preference and prerogative, and are areas for research and inquiry. The constraint model does provide a formalism for describing search spaces and design states, and this formalism provides one way of understanding both the structure and function of diagrams. Diagrams are concise and powerful representations of constraints, and serve as markers of a search path. Understanding the inferential purposes of diagrams and their information-content sheds light on a strategy of searching design spaces.

In this view of designing as searching, looking back and looking forward are important operations. Computationally, storing partial and previous states , and characterizing future states are necessary functions. Previous states must be stored so that they can be returned to; partial states so that they can be subsequently combined; and future states characterized somehow so that intelligent decisions can be made about pursuing them, that is, proceeding further along the present branch of the tree of search spaces, based on available resources and possible outcomes. This latter difficulty is an important theme in AI research about search strategies [Schank].

Diagrams are a concise representation of a design state, or search space, so they are an ideal representation of important branching points. The designer's sentiment "I like the general idea here, but the details need work" can be captured by storing a diagram and the specifications/modifications to it separately. Backtracking by the designer can then be to the previous (or some preceding) diagram.

In either role, catalyst or medium, diagrammatic inference need not always lead forward. The diagram may be seen as the result, or right hand side; then though induction (or abduction) a plausible left hand side is inferred: "What pre-condition could result in this state?", where the precondition becomes the design goal. In this mode of reasoning, the diagram serves as stimulus to backward chaining, like some so-called expert systems or the Prolog computer language. This difference in function of diagrams doesn't affect their structure.

Saving partial design states is also a diagrammatic problem. Partiality suggests either some fragment of a design state, which requires delineation of the boundaries of the fragment, and some strategy for cutting at the boundary, or some aspect of a design state, which requires articulation of the aspect. Diagrams, by representing elements and combinations or groups of elements, are one way to identify fragments; by incorporating a key or set of relationships, offer a way to identify aspects.

Since diagrams by their nature contain the seeds of future design states, they offer a way of characterizing and eliminating possible future states. The acceptance or rejection of a diagram is not just of the diagram itself, but also of all its possible instantiations. In the process of searching, one function of diagrams and graphics in general is to serve as a storage repository of design decisions to augment our short- and long-term memory. The role of diagrams in design development is taken up in Appendix 1, in consideration of the processes of design-combining and design-fitting.

In the next chapter I summarize the conclusions of this chapter, and present a set of questions and directions that are raised by contemplating continued research about diagrams and computer-aided diagramming.

Chapter 8. Review, Conclusions, and Future Directions

In this final chapter I summarize the argument of the thesis, and outline some directions for the continued research necessary for the development of more robust computer-aided diagramming /design systems and the beginnings of a comprehensive theory of diagramming. Appendix 1 presents an organizational diagram of a more comprehensive computer-aided diagramming system along with a consideration of three important components of such a system: the diagram-combiner, -fitter, and -reader. Appendix 2 illustrates the applicability of the CBD to some diagrams from outside of the domain of urban design. Appendix 3 lists the constraint language implementation of some selected elements of the lexicon.

At the beginning of this thesis I posed several large questions including "How are diagrams different from other kinds of drawings (maps, plans, and sketches)?" and "What significance does the distinction have, and how is it related to reasoning, in the processes of designing?" I argued that the inferential purpose of diagrams is what sets them apart from other graphics, and on the basis of that assumption I proposed a theory about making diagrams. The computer program CBD and the diagram replications and productions of the last chapters were experiments about that theory. In this chapter I review the theory and evaluate its implementation, consider the lessons learned from the experiments, and speculate about future developments in both theory and implementation.

Review of the Research

The effort to understand diagrams in general was applied in the domain of urban design for several reasons. As I have suggested, the domain is one in which diagrammatic decisions, and hence diagrams, are central and ubiquitous, and in which conventions are not as strongly established as in engineering disciplines, for example. The example diagrams displayed in chapters 1 and 2 were a biased sampling, not a thorough or a random sampling, as I was looking for diagrams that might be amenable to algorithmic production. Furthermore in urban design a useful condition prevails: the diagrams are most often two-dimensional spatial representations of two- or three-dimensional spatial layouts, so that the literal analogies that derive from map-like deductions are foremost in their interpretation. Urban design is concerned with social, perceptual and aesthetic consequences

that arise from physical arrangements, and in the examples (Columbia, Reston, Ideal Communist City) I was able to connect such high-level goals with particular geometric and topological arrangements, thus making a strong case for the relevance of the CBD (or something CBD-like) to design thinking in that domain. It is possible that some or many other interpretations of the structure and function of diagrams are possible, that may even be orthogonal to the interpretation I have given. This thesis is no less complete for that possibility, but the possibility deserves to be studied in the context of a continuing and enlarged understanding of graphic representations and their inferential import in general. The role of 'other visual analogies' in productive thinking, in urban design as well other fields, is a not-well-researched topic that may inform a better understanding of diagrams, and vice versa. Rather than any regularities in these analogical and metaphorical inferences, I would expect such research to uncover a great diversity, and in the best case to identify broad categories of such inferences and their function in design development.

The lexicon developed in chapter 2 and partly implemented in chapter 5 is an incomplete but extendable set of elements and relations. The purpose of the illustrative set is to demonstrate the value of using some such set, not necessarily the one presented. Declaring and elaborating upon a set of terms is both a continuing and episodic enterprise that is responsible for the evolution of cultural conventions and universals and the development of a personal style in any design domain. The adequacy of the object oriented descriptions, constraint propagation and conflict resolution mechanisms for the purposes of urban design is not fully established, but the results are encouraging, and I anticipate the development of both standard and highly personal vocabularies above and beyond geometric primitives will be central a central feature of computer systems for environmental design.

An Implementation of a Theory

Chapters three and four exposed a theory about the structure and function of diagrams. The theory locates diagrams with respect to propositions on one hand and drawings on the other, and

diagramming as an activity within the context of designing argumentation, inference-making that may be either conclusive or tentative. The theory states that diagramming is a transformation of representation that does not maintain 'equivalence', but is undertaken intentionally to exploit the inequalities of representations, and the characteristics of two-dimensional representations that have inferential differences, and in many cases advantages, over linear propositional representations.

The theory anticipates transformations in four directions, to and from diagrams and both propositions and drawings, but this thesis only addressed one of the directions in detail. The theory holds that the transformation from propositions to diagrams consists of several distinct steps, giving operational definitions to different parts of the design argument: filtering, mapping, assignment and layout. The theory construes such terms as 'emphasis', 'essential', and 'abstraction', that might be used to informally explain features of diagrams, in terms of the four proposed operations. Emphasis, for example is first achieved by the selection of elements to diagram ("This diagram emphasizes pedestrian pathways and conflicts", e.g.) in the filtering step; emphasis is also achieved by manipulation of graphic properties, such as bold lineweights or variations in color, in the mapping and assignment steps. The determination of 'essential' features of an argument is outside of the theory of diagramming, but once the determination is made, the algorithm says what to do about it in a diagram. The theory does not yet give an explicit description of the "diagrammatic grammar" referred to in chapter three [Lasseau], but the details in the last chapter indicate at least some of the requirements and characteristics of such a grammar.

The computer program described in chapter five is an implementation, and interpretation, of the theory. Tests such as those in chapter six help to demonstrate that the theory is a good theory of diagramming, and then go further, directing exploration towards the requirements for extending it into a theory of good diagramming, and developing the parameters and measures of that goodness.

The CBD is only a partial implementation of the theory: filtering and mapping are largely left out of the machine, and up to the user. Testing the powers and limitations of a constraint-based approach to assignment and layout, and the role of defaults and interactions in those processes, were the principal purpose of the CBD. As I suggest below and more in Appendix 1, much research remains to be done to explore the steps of filtering and mapping, the former in the context of a more complete constraint-based design system, the latter with respect to both conventional meanings of symbols, and innate psychological responses to visual cues.

The CBD, limited as it is in scope and expressiveness, demonstrates that a constraint-based approach is capable of achieving two dimensional layouts, but reveals an important consideration in that regard. A common claim for constraint satisfaction techniques is that they can maintain global consistency through only local operations [Winston]. Each constraint only operates on the variables directly referenced by it, and each object only 'knows' about the constraints in which it participates. Only the propagation procedure, or the conflict resolution mechanism, connects and combines constraints. This is useful as it's computationally efficient, avoids redundancy, and potentially renders the approach amenable to parallel processing. However, as in the case of diagrams, global side-effects other than consistency may be incurred in the process.

It is the nature of visual perception that it detects global properties (size, scale, outline, symmetry, grain, pattern, etc.) that may defy local specification or detection. Some of the 'mismatches' and anomalies observed in chapter six were of this sort. In the thought experiment related to the last example in chapter six (the MultiCentred Net) I considered the necessity of generators and predicates able to work with global properties like density. The nature and details of such procedures are open questions, but their necessity in a robust diagrammer is obvious.

Conclusions of the Research

If the lexicon, diagramming algorithm and computer program described in chapters 2 though 5 are viewed as a hypothesis (one with several distinct parts), then one of the results of the experiments in chapter 6 should be to verify, refute or extend the theory. I consider the results mixed; though the theory is not refuted -- the CBD is capable of producing some kinds of diagrams -- neither is it fully verified -- the CBD often makes rather poor diagrams -- and all the extensions that are required of the theory are not obvious.

It is clear that translations between propositional and graphic representations is not a translation of equivalence; that the inequivalence can be exploited in design exploration and problem-solving tasks; and that the inequivalence is only partly controllable, that is, that the interpretation of graphics into propositions is a 'many-to-many' translation. The translation from graphics -- the reading of diagrams -- is guided by fundamental neurophysiological constants, learned conventions, domain knowledge, and highly variable personal interpretations in mixed degree; consequently, diagram construction must also be so guided. The first three may be amenable to encoding into a knowledge base in the form of a library of rules and symbols, the latter must be accommodated by a provision for customization and user interaction. In any event the basis for, and the interactions of, inferences from graphics are complex and ill-understood. The present research has only scratched the surface of the knowledge that goes into apparently simple diagramming decisions.

In the experiments and discussion in the last two chapters, I've focussed on the inferences that follow from the graphic properties of diagrams, in an effort to explain the structural properties of diagrams, and their function. It's these inferential purposes, more than any overt graphic properties, that distinguish diagrams from maps, plans, or sketches. All of these graphics embody abstractions of a sort, and both enable us to reason in one domain (2-d plane) and transfer our reasoning to another domain (a multi-dimensional real or proposed world). They are diagrams to the extent that we suspend our expectation of literal representation and allow abstractions of

many sorts, and to the extent that we not only transfer our reasoning about the graphics into another domain, but also translate the results of the reasoning into different terms. Thus if a sketch "looks good", we expect the sculpture, or house, or plaza, will look good; if a diagram looks good, we have no such expectation about the house, or plaza, or new town. We translate distance on a map into distance on the ground; distance on a diagram need have no such implication. The enabling of not-necessarily-spatial reasoning in a spatial medium is what diagrams especially contribute to designing.

The first set of conclusions from this research are related to the making of diagrams. The second set, less clearly established but clearly indicated, relate to the reasoning.

Several major considerations emerged in the computational implementation of diagramming in the CBD: the task of finding or inventing words and algorithms to express design intentions in the form of elements and relations; the systematic translation of elements, relations, and rules of inference into two-dimensional graphics; the problem of assigning default values to diagrammatic variables while respecting the previous requirement; the requirement to allow for gestalt perception in the interpretation of diagrams produced sequentially; and the essential role of the dialog between designer and diagram that must be supported by a computer diagrammer. The nature of the dialog has not been explored in any detail in this thesis, but raises a whole host of related questions about user interface, system efficiency and responsiveness that re matters for continued research and development. The following paragraphs briefly address each of the other considerations.

The knowledge representation problem embedded in diagrams is both a choice between, and problem of combination of, words and graphics. Designers can presumably design, and make diagrams, without the explicit verbal/symbolic propositions I've taken for granted here. If such graphics are to function as anything more than patterns, however, I maintain that the propositional content must be there, even if not articulated. For personal designing, such a

proposition-less mode is perfectly fine. But in the realms of public debate, design education, and computer-aided design and inference, the explicit words are important and powerful. The links between the geometric/topological primitives I've explored, and the kind of high-level design goals and intentions that motivate designers, are not fixed and absolute, but are part of the design process. Some of the inferences reached by using diagrams are purely visual -- decisions about proportion, or grain, for example -- and not directly translatable to words and relations. Others -- like observations about intersection or containment, e. g, -- translate into propositions and relations that may trigger others through analogy and association.

The question of default-assignments and conflict resolution is not one to which there are 'correct answers', but to which responses are guided by convention, utility, and experimentation. The requirement for default values is like the requirement for algorithmic expressions in general; it requires both an articulation and a translation of design knowledge. The details of default assignments, and their interaction, are responsible both for substantive and stylistic results. The decision whether to put knowledge into the diagrammer's defaults, or into a particular lexicon entry, or into a proposition list for a particular diagram, is not settled, and cannot be. The CBD provides an experimental laboratory in which to explore the effects of various default decisions and approaches, which are bound to vary with different designers at different tasks.

The phenomena of gestalt perception, including the context sensitivity of graphic symbols, don't constitute a problem the diagrammer must overcome or avoid, but a solution, that the program can rely on. The neurophysiological and information-processing explanations of vision, including gestalt perception, are an intellectual puzzle, and their exploration will doubtless enhance future diagrammers (and vice versa), but they aren't a barrier to progress. The powers of the eye/brain are to exploited, not simulated or replaced. So the diagrammer must have a repertoire of available devices, graphic styles and symbols, relational definitions to help the designer make and refine diagrams, but it shouldn't be expected to be capable of vision (at least in the near

future.) Nor should we expect that the knowledge that guides diagramming decisions will be readily explicated. The present research has provided one vehicle for pursuing the questions, and suggested some of the directions that future research must address.

The interactions that give rise to emergent elements and relations may either illuminate or obscure the inferential content of the diagrams. Inadvertent and emergent relationships are a large part of the purpose of diagrams used in the design inference mode; they are to be embraced, not rejected. Used as communicators of existing (or proposed) design states, as in public hearings or navigational aids, for example, inadvertent relations are undesirable. The diagrammer thus needs to know the distinction between the two modes. In either mode, it might well ask questions or produce distinct variants when ambiguities or potential conflicts can be detected or predicted.

This phenomenon of emergent elements and relations vividly illustrates the 'non-monotonic' nature of diagrammatic inference, as the introduction of any new element, or the combination of any set, may produce inferences that supersede or conflict with earlier ones. It is the exploration of these interactions that makes diagramming powerful in design, and not just mechanical -- it is also the reason why an interactive system is necessary for diagramming. If the conclusions of a diagram are monotonic and hence predictable, an 'expert system' or theorem proving approach could substitute.

A question arising from the experiments described in chapter six is the issue of the reliability, or validity, of the interpretations made by the author from the original diagrams, that gave rise to the proposition lists from which the CBD constructed its diagrams. These propositions were reasonable, for the diagrams being considered, but not exhaustive. In fact, many of the 'mismatches' between the original and the reproduced diagrams were the result of missing or imperfect propositions in the input. Some of these omissions were justified on the grounds of simplicity; others are more revealing, in the sense that their omission tells us something about the reading -- and the reader -- of the original diagrams. A number of these omissions are of features

that are so constant in diagrams that they are expected (by the reader) to be defaults, and not to require explicit statement. For example, in Columbia the 'imbalance' that resulted from the diagonal line of the principal axis was the result of a (bad, in this case) default procedure for laying out lines. No explicit mention of 'balance' or 'symmetry' or 'horizontal organization' was made in the proposition set. These features did not appear in the initial reading of the source diagram because they are descriptions of the diagram, not of the referent design, and so they are 'transparent', a part of the medium.

Some of the modifications to the default diagrams were of features that were expected to be encapsulated in the definitions of elements. The use of a bold asterisk symbol for the external metropolitan magnets in the Columbia diagram, for example, should arise from the set of attributes and defaults associated with the generic element "metro-magnet" in the designer's lexicon, modified by the common cultural knowledge that allows us to reason about Washington and Baltimore in context. The assignment of properties to these elements is transparent in the reading of the source diagram, and so was 'missed' in the specification of the proposition list. This transparency of some diagrammatic features, and the dependence on context-modified defaults to produce sensible diagrams, is related to the emergent relations discussed above. What is visible in a diagram is never fixed or or finite. This suggests that any future diagram-reading program will have the requirement of producing multiple alternate readings, and that a computer-assisted diagramming system must enable the ready change or modification of propositional descriptions attached to graphics.

Some of the parameters of variation in diagrams, and their inferential import, are the factors identified in the previous chapter: shape and families of shapes, scale, linewidths and linetypes, axes of symmetry, constants used as defaults in inequality relations and in default generation. Such an enumeration and organization of a set of parameters that control diagrammatic variations and diagrammatic equivalence, along with an explanation of the inferential potency of each of the

parameters alone and in combination, will constitute a large step toward a general theory of diagramming.

Benefits, Applications and Future Directions of the Research

Raising miscellaneous observations to the status of systematic principle is considered justification unto itself [Fleisher], without regard for application or utility of the principle; this is the basic drive of scientific inquiry. In the introduction I attributed mixed motives to this thesis, both the pure and the applied. The contributions of this research are on both levels; providing a theoretical framework in which to explore diagrams and diagramming, and also some material guidance in the development of procedural approaches to that exploration. I'll consider first the theoretical aspects, and then address some possible utility below.

Toward a better theory

A theory is to explain; if the explanation is powerful it will serve to predict. In the realm of designing, theories have little sway, for the notion of predictability is antithetical to the culture of design. Most designers admit to "rules of thumb", but hasten to add that they break them as often as follow them. Such 'theories' as exist mostly take the form of either broad guiding sentiments ("seek a balance between community and privacy"), or collections of miscellaneous accumulated wisdom and prejudice. More tangible 'methods' enjoy slightly more credibility, but are still suspect [Alexander 71]. These theories and methods are about the design domain(s), but there are also theories about the domain of designing. The theory about diagraming I have proposed is in this latter class. What good is such a theory, and what are the requirements for a good theory?

The benefit of a theory of diagramming is first as a piece of a large intellectual puzzle: "What is that designers do when they design?" Diagramming is a part of designing, and the theory should say *what* part, and how it's connected to other parts. A derivative benefit is placing diagramming

methods and tools in a logical place and relation to a repertoire of other design tools and methods. It would be a mistake to claim that 'better' diagraming will follow from a theory, as much a mistake as it would be to argue that a knowledge of tonal or atonal theory produces music superior to improvisation by ear. A knowledge of theory, if the theory is good, does enable the considered generation of variants and assists learning by assimilating individual examples into a framework, but it does not guarantee a good product or practice. A good diagramming theory, and eventually a theory of good diagramming, may offer benefits to design practice and learning, but its first contribution is to design theory.

A complete theory of diagramming will have to explain the structure and function of diagrams as graphical artifacts and repositories of information, and also the activities of human diagrammers, sorting those activities, organizing them with respect to a purpose, and showing the necessary connections between the activities. Since there are observable variations in diagrams produced by designers, the theory will have to explain the variation, as well as provide parameters along which to measure the variations; in so doing it might suggest correlations between parameters. Such a theory will not 'predict' diagrams, in the sense of predicting the particular diagram produced under particular circumstances, but it should provide a terminology with which to describe the diagrams, and the circumstances. An incomplete theory (but the beginning of a complete theory) may address either diagrams themselves or the behavior of diagrammers, separately. I have approached the former: a theory about diagrams, rather than about diagrammers. I have offered what I think is a good theory of diagrams; but it is not yet a theory of good diagrams. In the best event, the theory will prove extendable, to provide a basis on which to predict the inferential power of diagrams, that is, a measure of better and worse diagrams, and so become a good theory of good diagramming.

A metric of quality between instances implies and requires a metric of variation between instances, and so a theory of variations between diagrams is a start toward a theory of the goodness of

diagrams. I explored some of the parameters of variation in the sample diagrams in chapter six, and some clues to the sources and impacts of the variation in chapter seven. Some of the variations I have characterized as 'inferentially potent', others as 'stylistic', like the change of fonts in text. The role of a theory of good diagrams is to explain variations in terms of inferential potency. If the diagramming steps proposed in this thesis -- filtering, mapping, assignment and layout -- have validity, then one requirement of the theory is to describe and relate the inferentially potent variations possible within each step. The form of research necessary to clearly isolate parameters of diagrammatic variation, and to measure such differences, is not clear. The need to identify and eliminate the effect of personal preference, and the limitations of media, are initial obstacles to such research. A possible result of a systematic treatment of variation would be a typology of kinds of diagrams and diagramming.

Each of the four diagramming steps places its own demands on a comprehensive theoretical treatment, (and I outline some of these as research questions in Appendix 1), but filtering in particular raises the question: "How is 'relevance' to an argument specified or measured?" An approach to that question would provide a basis for a theoretical explanation of what kind of diagram would be best suited in various circumstances. The theoretical treatment of filtering must incorporate an explanation of the selection and transformation of rules of inference, along with elements and relations.

An additional task for a theory of diagramming is to explain the process of combining two or more diagrams, one technique of design synthesis and development. Combining diagrams, like diagramming in general, is usually done intuitively, without the benefit of a systematic approach (consider [Alexander] or [Alexander and Manheim]). In the framework of a general theory, the operations of combination need to be named and their effects specified. This includes explaining the mechanism whereby elements in one diagram refer or correspond to non-identical elements in another diagram, so that the two may combined. (Some problems of diagram combination, and the

related problem of fitting a diagram to a map, are outlined in Appendix 1.) It may be that combinations are subject to variation just as other diagram attributes are, and the parameters of variation may provide a key to diagram combination.

A comprehensive theory of diagramming will have not only to address particular details of diagrams such as those I have suggested, but also to accommodate a larger theoretical framework that deals with the roles of representation, inference, recognition, search, planning and learning in design thinking. The differences and respective advantages of symbolic versus graphical representations have been studied in the context of problem-solving; the differences between different graphic representations are far less well understood. Diagrams offer both a medium for selectively controlling graphic characteristics and a contrast to pictorial sketches, features which can facilitate research into representations and their use. The uses of representations in this sense is for inference, and the switching of representations is itself an inference of a sort -- the construction of a diagram forces some implicit information in a proposition set to become explicit. When how and why the switching takes place in design process is a subject for more focussed research.

Recognition is a special form of inference, that is the basis for chains further inference and analogical reasoning. The algorithmic technique of "pattern-directed inference" may be a metaphor, if not the literal mechanism, for some design reasoning. The role of diagrams as keys by which design ideas, images and exemplars are stored and retrieved is another area for further research. Diagrams might prove to be a useful alternative to lists of keywords, which are often unsatisfactory keys in multi-dimensional indexing schemes.

This use of diagrams as indices is a part of the role of diagrams in search, a subject I discussed at the end of the last chapter. A diagram of a universe or state being searched, as well as of individual paths, may prove to be valuable tools for thinking about thinking, in the process of

designing, in the same way that computer system designers and operators use network diagrams to monitor the state of complex interconnected machine processes. Planning as part of designing may take the form of designing the design process in which competing demands and limited resources must be combined into a planned approach. Designing such abstract systems as design processes or organizational systems is the ideal application for diagrams above all other representations. Finally, learning to design consists, at least in part, of synthesizing and organizing all of the above. Diagramming demands a judgement of what's important at what time, in what context, which is why design critics and instructors use diagrams, albeit intuitively and informally. The benefits of explicit and focussed use of diagrams in design learning, would include providing a rationale for selective ignorance and emphasis, surfacing important issues in their context, and an opportunity to explore the advantages and disadvantages of "building the diagram".

The theory proposed in this thesis is based on introspection and analysis of diagrams, but not on observation of diagrammers at work. I would expect that such empirical observations, and accompanying protocol analysis, might have two purposes in this regard: to verify, refute or modify the steps and connections presented in this thesis, and to provide additional detailed experiences and examples on which to develop the other half of a complete theory of diagramming, addressing the behavior of diagrammers at work, and locating that behavior within a larger description of designers' behavior.

Potential Utility

The conceptual advantages of a theory of diagramming will be accompanied by tangible benefits in the development of computer-aided design systems. Computational designing demands algorithmic representations of sufficient generality to span more than one design episode, and an expressive vocabulary of elements and relations that permits of variation -- both of which are provided in any approach to diagramming. The implementation of systems utilizing diagrams as basic input/output

communicating with constraint managers and 2- and 3-dimensional drawing and rendering systems, (such a system is outlined in appendix 1) is a distinct possibility with practical value.

The practical value of having a CBD-like system would be in two main areas: design research and education, and design practice. In the first case, computer systems for describing and exploring design ideas, such as the vocabulary of elements and relations required by the CBD, provide a laboratory in which to inquire about fundamental issues of knowledge representation. In the second case, while the CBD itself is nowhere near being a production tool, such 'conceptual' font-ends to drawing and rendering systems will be a major advance in the application of computing to environmental design.

Consideration of the role of diagrammatic inference in designing suggests that the value of a computer diagrammer rests not on its ability to replicate particular diagrams, but rather on its ability to produce a variety of diagrams, in the hopes of stimulating a variety of inferences. A replication, such as the diagram of Columbia in chapter six, is evidence of the diagrammer's utility, and exposes parameters that must be controlled, and hence may be varied. The ability to systematically control these variations would be a powerful aid in design exploration.

This ability will depend both on the designer-user knowing when to request variations, and on the diagrammer knowing how to produce them. In the open-ended searching that characterizes designing, the occasions when a diagram, or a different diagram, might be requested or produced are impossible to predict. The variations should be available by specific request ("Show me this diagram replacing the liner arrangement with a concentric arrangement"), or general request ("Show me another variant"). A theory relating the form and degree of variation to the argument being pursued, or its likely consequences, is required to direct this generation of variants, otherwise the variants can be no better than random.

One important form of variant generation and testing is the systematic substitution of opposites and known alternatives for relations. This is a technique used by design instructors ("Suppose we turned this design inside out...") and by some designers. Since diagrams are more inferentially potent when two or more are side by side, this mechanism might be a poweful one for design exploration and education.

The fact that many features of diagrams are transparent upon first reading, complicated the problem of interpreting diagrams to produce a proposition list in the examples in chapter six. This phenomenon also affects how diagrams are used in design exploration. A major benefit -- and pitfall -- of computational approaches in general is that they require articulation of knowledge. Since it is often difficult to articulate design knowledge, it maybe difficult to productively use a computer-based diagrammer in the manner proposed here. It's possible (even likely) that designers confronted with a diagram are able to make modifications and extensions, in the diagrammatic medium, without being able to articulate the reason for or nature of the modifications. To the extent that diagrams thus completely substitute for propositional reasoning they lose the advantage of moving between representations, and their reason for being. The use and development of CBD-like systems will clarify the distinction between articulable and potentially algorithmic knowledge on the one hand, and implicit, tacit knowledge on the other.

Toward a better Diagrammer

The next steps in that direction involve improvements in all aspects of the CBD. The development of a vocabulary of elements and relations, and an understanding of their use, will come from use of something like the CBD, in both research, teaching and practice settings. The first two settings are not inherently difficult to achieve, although the disciplined development of vocabulary requires some significant insight into the problem domain, and is not altogether separable from other improvements in the diagrammer. Such a vocabulary will require, for example, the development of more techniques of expressing complex and abstract constraints, including

constraints on sets and more explicit handling of inequalities, and some mechanism(s) for controlling interactions between constraints, either in the form of synthesis and combination, or branching. These kinds of improvements requiring substantial coding and overall reorganization of the system of the CBD, and are difficult to the extent that their requirements remain ill-defined, as they do.

Such coding enterprises are in the form of experimenting, and not just building experimental apparatus, since the questions to which answers are sought are not well established. Improved mechanisms for layout, including "find-space", "remove-intersections", and "make-symmetrical" algorithms, for example, are computational issues that can be substituted for by eye, but which will offer the diagrammer more opportunities to generate variants procedurally. They are difficult in detail, and represent challenges to algorithmic development, but are not major obstacles to the further understanding of diagramming. Making layout decisions interactive with mapping and assignment decisions -- adjusting linewidth, size or shape in response to context, for example -- is not inherently difficult as an algorithmic or procedural task, but establishing criteria by which to govern such interactions and adjustments is a major enterprise, fundamental to further understanding of diagramming, and also dependent on experimentation.

The research that will be required to uncover and organize the details of our visual cognition system that enable us to use diagrams at all, need not be computational, but there are several benefits to the computational approach. Explorations of vision and visual properties that are more anecdotal or empirical than procedural, such as the gestalt psychologists, do not provide explanations in terms of other well-understood principles or mechanisms. They provide impetus for the development of such explanations, but not guidance. [Arnheim], [Goodman], [Laseau], [Alexander and Manheim] all describe features of visual thinking that contribute to the making or understanding of diagrams, but in ways that are neither rigorously testable or implementable, no matter how plausible or convincing. Embedding an explanation of the inferential import of

graphics such as diagrams in a computational system demands a specific theory at each step, and provides a controlled environment for experimentation. The modern proponents of cognitive psychology seek just such specificity and control in their attempts to implement "models of the mind". Development of a deeper understanding of diagramming and diagrammatic inference within the context of computation will benefit from, and contribute to, that ongoing research. (See the end of Appendix 1 for a consideration of some objections to computational models of cognitive tasks.)

Summary

In the first two chapters of this thesis I demonstrated diagrams from the domain of urban design, and showed that they are linked to knowledge-based propositional reasoning about the design domain, and depend on visual cognition that depends on, but is different from both the appreciation of sketches and renderings that leads to aesthetic judgements, and the literal interpretation of maps that leads to geometric deductions. I proposed that diagramming plays a central role in some design reasoning, bridging the gap between propositional and graphic representations, and providing an initial graphic artifact to spur design development. In chapter three I described four pathways between representations of design information, with symbolic propositions at one end, detailed design drawings at the other end, and diagrams in between.

In chapter four I proposed that the procedural requirements for diagramming could be described as filtering, mapping, default assignment and layout; each process removing, transforming or adding information. In the last three chapters I explored the behavior of a computer program that produces graphics from verbal representations, focussing on the last two of the three steps just described, by attempting to replicate some sample diagrams from urban design literature. In the mapping and layout steps I identified the need to invent specific quantitative default values for inexact qualitative specifications, and the need to adjust and propagate constraints in the face of

incremental new information. The CBD was designed to explore one particular pathway out of the four paths through diagrams, and the requirements for mapping and layout.

Two experiments in producing diagrams from a list of propositions demonstrated that the CBD could produce simple diagrams in some circumstances, but that the propositional input, default assignment and constraint propagation process together were insufficient to explain all the features of the observed diagrams. This observation leads to questioning the process of reading diagrams and deriving propositions from them, and the sources of knowledge that underlie the reading and making of diagrams. I concluded that the analysis of the mis-matches points to partial answers to both the general question "How come these diagrams?", and to the more specific question "How come these diagrams?"

The evidence suggests that the inferential content of the graphic features of diagrams derives from both inherent capabilities and characteristics of human visual cognition, diagramming conventions, and learned domain-specific reasoning, all of which maybe culture-dependent. I argued that in the context of designing, these phenomena of visual cognition can be located with respect to three different kinds of inferences that proceed from diagrams -- "map-like deductions" that utilize literal translation from 2-d space to 2-d space (and so are best supported by 'B-diagrams'), "map-inspired analogies" that translate from 2-d spatial properties to analogous non-spatial properties, and "other visual analogies" that depend on associations with non-spatial visual properties. These categories suggest the forms of knowledge that must be embodied in any diagrammer, the kinds of 'inference engines' required for an intelligent diagrammer, and the types of research required to further elucidate the knowledge.

I suggested that the observed mismatches resulted from two principal sources: first, from the failure to include certain propositions in the original reading of the source diagrams, those features being transparent in the sense that they were global properties of the diagrams, and hence not

salient to the specifics of the design argument. The very purpose they served, to tie together the elements of the diagram, made them 'invisible' on first reading. These kinds of qualities and the relations from which they derive are the ones that must be supported by the default values and strategies in the diagram mapping and layout. Second, regardless of the ability to produce an exhaustive list of propositions, the relations capable of being read off any graphic are virtually unlimited, and so it is no surprise that some 'unanticipated' relations emerge from the combinations of elements and relations in the diagram. This fact points to the need for an interactive, and customizable, diagraming system. I argued that neither of these sources of mis-match is a defect in such a system, but are unavoidable, and point to the complexity of the perceptual knowledge.

I concluded that the value of a computer diagrammer would rest not on its ability to replicate particular diagrams, but rather on its ability to produce a variety of diagrams, in the hopes of stimulating a variety of inferences, and the ability to control these variations might be a powerful ability in design exploration. Some of the parameters of variation in diagrams, and their inferential import, were revealed by this research, but remain to be expanded upon by empirical studies of designers diagramming, psychological research into the effects of graphic variations, and computational research into procedures for implementing and controlling such variations.

The value of diagramming in design, I argued, is as a transformation of knowledge representation that uses visual and spatial-reasoning abilities to augment other reasoning. The value of a comprehensive theory of diagramming would be to collect and organize the principles of visual reasoning with respect to a larger theoretical treatment of cognition in general. The benefits of such a theoretical framework would be at least two-fold: in design education to provide a terminology and a logic for the progression of design concepts into design artifacts; and in design practice to provide a basis for more intelligent and responsive computer-aided designing systems, that would incorporate diagramming as a basic mode of interaction and communication (such as the outline described in appendix 1.)

Finale

I said at the beginning that a computer program is disposable, more of a means than an end. The important findings here are about diagrams and their use, not about computational techniques. The graphics repertoire of the CBD is simple, its data structures conventional, and its weaknesses numerous. Nonetheless, the experience gained in the development and use of the CBD has proved fruitful and provided insights into the problems facing diagrammers, human and machine.

The general import of this research has been to raise the status of diagrams from "taken for granted" to "subject to scrutiny and computation". I have laid out the justification for, and some of the requirements of a general theory of diagramming, and its computational implications. Among the specific contributions made by this thesis are:

-- locating diagrams in relation to other graphic products (maps, plans and sketches) and describing them as expressions of constraints; identifying the role of diagrams in design inference, as both graphical products subject to graphical operations and refinement (the process of fitting), and expressions of propositional content leading to new propositional expressions (the process of abstraction);

-- associating specific computational procedures and requirements with the general purpose and features of diagrams: *filtering* for simplifying and concentrating propositional content; *mapping* for making analogical links between design elements and relations and diagrammatic elements and relations; and *layout*, for producing particular graphics with particular stylistic characteristics; and identifying some of the inferential import and computational problems of each of these steps;

-- identifying three different sources of knowledge required for making and reading diagrams : knowledge of *visual cognition constants*, that provide some regularities that structure perception and upper and lower bounds on graphic properties that are distinguishable; *diagrammatic conventions*, that provide minimum standards of legibility and some standards of

symbolism, based on the previous knowledge; and *substantive domain knowledge,* that modifes and extends the conventional diagramming knowledge, and provides a context for disambiguating symbols and their references in diagrams;

-- identifying three different kinds of inferences that proceed from diagrams : *map-like deductions, map-inspired analogies,* and *other visual analogies* -- giving examples and and describing the role of each in design reasoning;

-- describing a constraint-based computer program that demonstrates the possibility of producing some diagrams with a limited lexicon and constraint-propagation approach, and provides a research vehicle for exploring the parameters that control the appearance and inferential potential of diagrams;

-- describing in outline the features of a multi-component computer-aided diagramming environment (in Appendix 1), and identifying some specific research questions required for its development.

It is my belief that this thesis contributes to the beginnings of a comprehensive theory of diagrams and diagrammatic inference, and provides some direction to the continued research necessary for the elucidation of diagramming knowledge, and the development of more robust computer-aided diagramming and designing environments.

Appendix 1. Constraint-based Diagramming/Design System

The body of this thesis focussed on a limited case of the diagram transformation, production of A-diagrams, from a proposition list, in the form of the CBD. A more comprehensive Constraint-Based Diagramming System should be at least in outline prepared to deal with all four transformations involving diagrams -- from and to propositions, and from and to more detailed drawings. With the understanding that each of these transformations will entail problems, similar to the problems in this one transformation as well as possibly novel ones, I outline below the components and benefits of a more comprehensive computer-aided diagramming environment.

In imagining a better version of the diagrammer, I am imagining a more complete computer-aided design environment, shown as a diagram in figure 9.1:

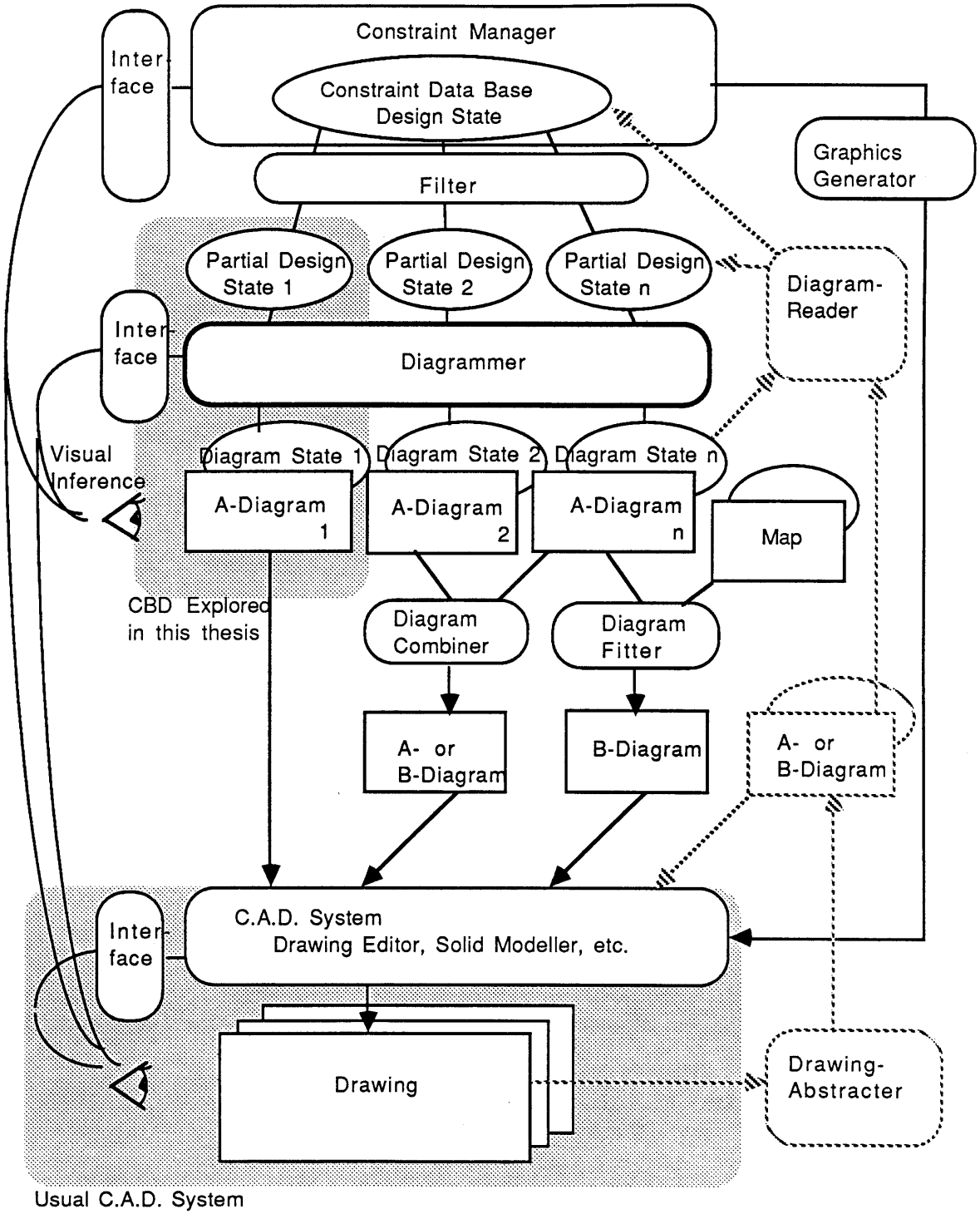


Figure 9.1 Comprehensive Computer-Aided Diagramming Environment

The diagram shows, for perspective, one shaded rectangle behind the small portion of the whole picture that was addressed by the CBD in this thesis, and another behind the small portion

addressed by most usual CAD systems. This environment contains a constraint manager and propositional data base, a set of procedures for filtering constrained design states, a diagrammer producing diagrams in the center, and a CAD graphic editor to handle output from the diagrammer and to produce and modify drawings in general. In addition to the central diagramming procedure, two other diagram-handling procedures are proposed: a diagram-combiner, that takes two diagrams as input, and produces a new diagram; and a diagram-fitter, that takes a diagram and a map as input, and produces a modified diagram 'fitted' to the map (described in more detail below). Like the constraint explorer described in [Gross], this environment contains a mechanism for recording and storing history during the process of designing. In this case partial designs are stored as diagrams, along with a key.

The entire system is shown as an interactive one, in which the designer user (at left) is responsible for input to the constraint manager, the diagrammer, and the graphic editor, and for viewing the output, diagrams or drawings. All of the flow of information is shown going from top to bottom (concepts to graphics), except the flow through visual inference, shown at the left, and the dotted lines in the proposed path at the right. This path includes two more proposed procedures: a drawing-abstracter, responsible for producing diagrams from drawings, through a combination of processes of smoothing and abstraction; and a diagram-reader, responsible for inferring propositional content from a diagram data structure. The implementations of these latter two procedures are less immediately imaginable than the combiner and fitter, so they are shown dotted.

Implications for Designing

The system envisioned is one in which the designer is a writer (of text, and code) as much as (or more than) a drawer of drawings. He is responsible for producing a lexicon of elements and relations, starting with a basic set like the points, lines, polygons and topological primitives in the CBD, and a set of filters, for which there may also be standard templates. These knowledge-

based writing/coding activities come before any specific designing, and allow the diagramming system to accumulate a customized and personalized vocabulary. These activities are designing; they also constitute a large part of design learning, and one benefit of computer-aided diagramming in education is in forcing this explicit representation of knowledge (including encountering and pushing the limitations of the representation.)

Specific designing tasks proceed by a dialog with the constraint manager, in which variables, fixes and constraints are entered in the usual way. The designer may request a diagram at any time, by specifying a filter and a key. The diagram produced by the diagrammer is visible as graphics, and also available as a data structure, to be saved and further processed by other procedures, such as the combiner or fitter. Diagram output may go directly into the graphics editor, for development into a drawing, and in order to have this development saved and modifiable, the development is reflected in the constrained state as well. I imagine that a graphical interface will be provided to the constraint manager, so that this operation is not too cumbersome. The inferences that arise from the diagrams, from the designer, result in either problem-specific amendations to the proposition list (constraints), or diagram specific adjustments to the appearance. In the latter case, they may be implemented as more general adjustments to the lexicon or relations, if they can be captured ion a general way by the designer. One application of some AI research in learning by example [Winston] might be to enable the diagrammer to try to generalize from specific modifications to general principles.

Making propositions into partial design states is equivalent to the act of drawing a diagram directly, so a graphical input option will be provided to the diagrammer. This takes the form of a simple interface in which the verbal expression of propositions is replaced by iconic manipulation. The features of this interface will also be modifiable by the designer, built upon a set of standard abilities for naming and relating elements. Reading the propositional content of a diagram thus entered may be simple, if the graphic interface is constructed out of primitives that substitute for

their propositional content, or may be complicated to intractable if the graphic input is quite general. The options for diagram-reading of this sort are discussed below; the enterprise may benefit from developments in so-called 'graphical programming' which use graphic interfaces for expressing algorithms in computer language.

The procedural implementation of any revised diagrammer will have to handle the same transformations as the CBD, in the form of filtering, mapping, assignment and layout operations. These operations will require the kinds of improvements suggested in the previous chapters, and doubtless others, some of which are anticipated in the following list of considerations.

Filtering

The Filtering step is responsible for selecting elements from a source constrained state to be diagrammed; in addition, it may be necessary to translate changes in the filtered state back into changes in the design state. In addition, prior to selection the design state may have to be transformed if the filtering requires elements that are not explicit in the initial state. This transformation can include aggregation and abstraction of existing elements and relations into new elements and relations, or the rule-based generation of new elements. Each domain will have certain conventional filters, that can be stored and retrieved by name; these too may be combined for particular purposes. Since each filter is associated with a key, stored filters may be referenced by their key. The transformation from drawing to diagram is likewise performed by filtering, although in the case of B-diagrams the filtering may be on the graphic elements rather than the propositional data base -- this is the problem of 'graphic smoothing'.

Research questions that are raised by Filtering include:

What methods might be used to encode and store filters -- combinations of rule-based transformation and selection -- that are sufficiently general in their syntax and rules of reference that they can be applied unambiguously to arbitrary constrained states?

Similarly, what techniques are available to encode and store rules of of aggregation and disaggregation, abstraction and specification, so that filtering and fitting can be performed automatically on either side of diagrams? Filtering and fitting can be described as procedures that take a constraint state as one argument, and one or more other arguments indicating the circumstances and nature of the filtering or fitting, and return another, transformed design state. How do arguments (in the discursive, not computational, sense) generate filters? That is, when diagrams are used as communication rather than exploration, are there conventions and shorthand for specifying the filter to be applied to generate the diagram? Even when used for design exploration, there must be a 'relevance' filter applied. How is relevance with respect to an argument, or an inference, judged, described and filtered for? The "Information Lens"program [ref] , that extracts "useful" information from a stream of data might provide insights into these questions.

Another set of filters is composed of those that perform graphic transformations -- snapping to a grid, smoothing curves, imposing modular lengths and regular shapes, for example. What algorithms exist for these operations, what requirements do they pose for the data structure representing a drawing, and how can these filters be used, if at all, in designing?

Mapping

The filtering and the mapping communicate through a key, a data structure that is itself a diagram, indicating the mapped relation between elements. The mapping procedure must know the available repertoire of the graphics manager, to establish the possible elements of the right hand side of the key, and must have access to domain knowledge that indicates the salient characteristics of the elements being mapped, and visual/perceptual knowledge so that appropriate analogical characteristics can be chosen in the diagrammatic representation. Mapping is further complicated by the four-way combinations: elementsfi elements, elementsfi relations, relationsfi relations and relationsfi elements. In the CBD2, the mapping process must be

able to be run backwards in the process of 'reading diagrams', inferring design elements and relations from diagrammatic ones.

Research questions that are raised in Mapping:

What set of elements and relations are necessary and sufficient on the destination side of the mapping, and how does this vary with domain? The set of distinctions identified by [Bongard] are candidates; research is required to determine the common and domain-specific interpretations of each of them. Are there possible rules or techniques for inventing symbols, that would rely on both domain and perceptual knowledge?

What conditions, and combinations, determine whether elements or relations will be read? What rules are there guiding the cross-mapping from elements to relations, and relations to elements?

The diagrammatic elements are relations will be formed by specific graphic attributes. In choosing values for these attributes, some quantitative values must be assigned for qualitative expressions -- what distance is 'far', or 'near', what size is 'big', how much smaller is 'smaller', and so on. Any general rule (like "two times bigger") is bound to fail sometimes, or perhaps even often, in conveying the intended relation. It seems likely that the best approach will be to let the human viewer adjust these decisions, but research may discover generally better starting positions. Are there some general rules that work often? What conditions affect the utility of these assignments?

Layout

The layout procedure is the most complicated control problem, and uses information provided by the mapping, but may override it. The requirements on the layout procedure include making decisions in the face of multiple choices; adjusting decisions from previous steps in response to new information; avoiding certain notable conditions (like intersection, overlap, coincidence) in the graphics, and combining requirements from multiple interacting relations. It is possible that pre-compiling layout approaches to common combinations is an efficient approach, and serves to create particular 'styles' of diagramming.

The layout procedure should be able to detect translation- and rotation-congruent moves, so as to minimize search and reduce unnecessary user guidance. The layout procedure needs to be able to accumulate relations that result in interval values, and choose suitable values from those intervals (averages, for instance). The layout needs to be shape-sensitive, to adjust layout strategy to shapes and available open-space in the evolving diagram.

The converse of layout is the inference of actual location and other attributes of design elements from diagram elements, the process called fitting. It is likely that some aspects of fitting can be captured in rules that describe prototypical responses to environmental conditions, or rules for generation of variants.

Research questions raised by Layout considerations include:

Some topological configurations are known to be impossible in the 2-dimensional plane. How can these be recorded and recognized? Should the layout procedure adopt a different mapping that would make the relations possible in the plane, or communicate with the user through some conflict resolution means?

A number of 'utility' procedures would be useful to the layout procedure, such as uncrossing lines, finding available open space, detecting and producing bi-lateral symmetry. How can these be incorporated and called upon as necessary in the layout operation?

Many diagrammatic values are defaults representing choices from within a range. How can ranges, or intervals, be explicitly represented, as by tones or arrows? What are the inferential advantages and disadvantages of such representations? What are the graphic procedures for producing them, and limitations on their use?

Some layout decisions are essentially 'stylistic'. Rounded corners on rectangles, dashed and dotted line patterns, the choice of grid lines, are all graphic attributes that don't derive from the propositional import of the diagram, but do affect appearance and therefore potentially affect

inference. How can these kinds of decisions be characterized? Is there a concise way to represent a set of style decisions, as in a key? Would this be a benefit to a designer in producing customized diagrams?

When partial diagrams are saved and restored, the problem arises of combining and resolving them. This problem is similar to the problems of reference raised by storing and recovering filters; a syntax and means of reference must be imposed that allows relations and elements to be described in the abstract.

Diagram Combination & Fitting

The questions above are all related to the production of diagrams from propositions. The other side of the diagramming cycle -- that develops diagrams into drawings, or moves them along the continuum from A-diagrams to B-diagrams, maps and plans -- is perhaps also attainable in a prototype form.

Interval Solutions

In chapter five I illustrated the problem of combining two relations -- a set of elements both outside a circle, and along a line -- and showed that while the simple CBD could not handle the interaction, solutions seem immediately obvious to the human (recall Figure 5.3) What intelligence, and knowledge, permits these solutions to be generated?

In many cases, these solution sets can be described as the result of the intersection of planar sets, much as complex 3-dimensional shapes are so described in some approaches to computational geometry [Woodbury]. The result of these intersections can be represented by shaded diagrams like the one below,

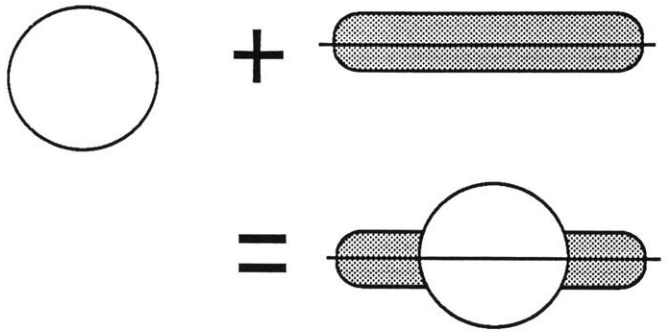
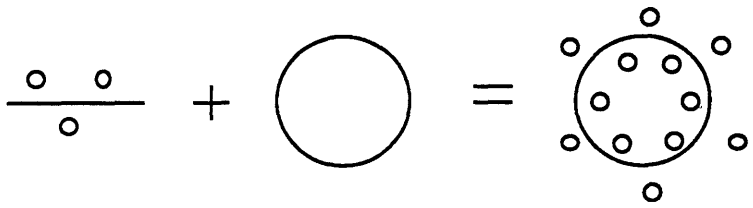


Figure 9.2 Diagram Addition
(along river house1 house2 house3) +
(outside town-center house1 house2 house3)

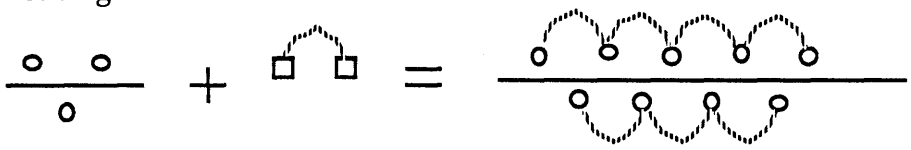
although such diagrams, that explicitly show the bounded solution space, are not as common in design as those that show an instance of a solution within the space. This is explained by the desire to use the diagram to develop toward a particular solution; a diagram of an instance of the possible solutions, like figure 9.2 above, evokes both the bounded space and the details accompanying any solution, and so is more powerful for inference. Diagrams like the one above are more likely to be encountered in pedagogic or communications circumstances, as they convey the constraints and the solution space clearly, without suggesting a particular solution within the space. The logical and graphic methods required to derive and produce diagrams like these last ones are an area for further research and development.

Diagram Combination

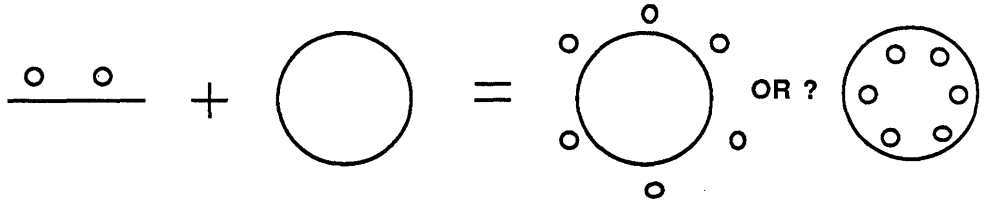
The possibility of rules and procedures for combination of diagrams requires an ability to resolve references in two separate domains. With two diagrams at equivalent levels of abstraction, the principal question is how to correlate elements and relations in the two diagrams, and what do do if there is no correlation. For example, one diagram may indicate the relation between dominant and subservient elements (houses are located directly on a road). If the second diagram contains a road-like element, then it may be taken as the dominant element, and houses located in the same relation to it.



If the second diagram contains house-like elements, the relations and attributes on them can be applied to the first diagram.



Such combinations, though, are not without ambiguity. Even with access to the original propositional content, the combination of diagrams raises the same questions of interpretation that the layout problem has, and must make default decisions for some or all relations.



The constraint-based, object oriented representation of diagrams and elements provides a basis for making these combinations, but also presents difficulties. How are equivalent objects recognized, if not explicitly identified? How are constraints in one object translated to constraints in another object? These questions demand an appropriate syntax in constraint expressions, and rules and conventions governing their translation.

It is possible that a substantial amount of concept formation and design development can be described as the combination of diagrams, and therefore that an algorithmic approach to combinations would provide an interesting means of design exploration. Some work in GIS has defined an 'algebra' of combinations for maps, and shown how that algebra can be used for the analysis of planning problems at appropriate scales [Tomlin]. An intriguing question is whether an analogous formalism can be discovered or invented for diagrams, realizing a suggestion first made by [Alexander]. If so, it will have to address all of the issues described above; if even a limited

case can be devised it will be a strong step towards a more powerful computer-aided diagramming environment.

Diagram-Fitting

The diagramming algorithm presented in chapter four assumed the one-way flow from propositions to diagram that the CBD explored, but each of the four steps -- filtering, mapping, assignment, layout -- can be considered as part of the two-way flow in the larger diagramming picture. The complement to filtering, which aggregates, abstracts and removes information, is the 'fitting' process, in which a diagram is developed into a more detailed drawing, and information is disaggregated, refined and increased. In this fitting step there is also a mapping in which some constraints and attributes introduced in the diagramming process are discarded or replaced with detailed design values, attributes and constraints. This mapping is the complement of the diagrammatic mapping, and incorporates the complement to the layout step, in which position of diagrammatic elements is discarded or translated into positions or other attributes of design elements.

This process called 'fitting' or design development, can take two obvious forms. One is the addition of detail and complication to a diagram according to some rules or conventions, but absent any real intended application or site -- that is, the detailing of an idealized imaginary design. The other form of fitting is the development of a diagram in response to some particular circumstance, or site. I argued that the first example diagram of the Ideal Communist City was of the former type -- a diagram that had been arbitrarily expanded without reference to a site -- and the second example was of the latter type -- a diagram fitted to a site (albeit an imaginary one).

The first form of design development will require a library of domain-specific transformations that refer to the formal and geometric properties of the artifact being designed. For example, roads are typically straight with circular bends; walls and structures are typically straight with angular

bends. The library or the diagram fitting procedures will also need to know about exceptions to these rules: intersecting roads cross at right angles, some buildings may have curvilinear walls. Developing a diagram in the absence of a site or circumstance is a rather empty gesture, resembling an academic exercise. The results will typically be a (possibly large) tree of possibilities; the most easily imaginable application will be as a pedagogic or design exploration device to present multiple alternatives, or test the impacts of certain rules or combinations of rules on development.

The second form of design development, in response to a particular site, is more intriguing as a component of a computer-aided design environment, and raises different research questions. The possibility of placing the same diagram in different settings, and observing different outcomes, is potentially a powerful tool for design pedagogy and exploration. The requirements for such a computer program include the encoding of sites and circumstances, for which there are some standard techniques (as embodied in 2-d and 3-d CAD and GIS systems, for example), and of rules and procedures for 'fitting'. The knowledge in this case must include details about how roads and buildings adjust to topography and other environmental factors, for example. A syntax is required for expressing such rules in a form shared by the constraint representation and filters described above.

Diagram- reading

In the larger picture of diagramming, transformation from diagrams is as important as production of diagrams. A comprehensive computer-aided diagramming system will require a component responsible for inferring propositions from the diagram (an activity which has been reserved for the human so far). Two different approaches to a diagram-reader might be taken. One, more likely to benefit from research on vision and pattern recognition, would take a complete data structure representing an image of a diagram (as a bit-map, say), and a key, as input, and attempt to read the diagram by first extracting the elements and relations, or multiple possible sets of elements and relations, and passing these through the key, to return a design state possibly

represented by the diagram. A second approach, more likely to be of use in a dynamic computer-aided design environment, would take input a step at a time, in sequence -- it would provide an electronic sketchpad on which the designer could work. Its inferences would come at each step, attempting to construct or modify a design-state in response to each partial diagram.

This latter approach offers more promise for several reasons. One utilitarian reason is that such a machine could serve a designer in a useful way, as it would serve as a kind of amplifier -- by making a diagram and a key, both relatively lean data structures, a more complex and full data structure would be automatically constructed. Modifications made to the diagram could be propagated through to more complex changes in the corresponding design state. A second reason to prefer this sequential diagram reader is that information about sequence would significantly ease the problem of detection of elements and relations. If the primitives used by the designer were elements and relations, some of the ambiguity in detecting them could be eliminated, and more inference power could be devoted to detecting interrelations and consequences in the design state.

The development of a diagram-reader, even a prototype, will require an extensive set of experiments like the ones presented here, to identify the issues and problems involved. Such a development will justify the emphasis placed in this inquiry on diagrammatic inference and the operations behind it.

Objections to A Computational Model

Even if the theoretical analysis of diagrams presented here is accepted, one might still object to the general idea or the particular strategies of involving computers in the process. Proposing, or making, computer systems that do things once or usually reserved for humans is subject to several attacks: that the systems perform so poorly that they are not worth pursuing; that the very idea is offensive or bound to fail; and that the proposals and research and development efforts are flawed

by deceptive or impossible claims. [Dennet] has commented on this last kind of objection, and listed four chief dangers that beset system designers in artificial intelligence research in general:

> (1) designing a system with component subsystems whose stipulated capacities are *miraculous* given the constraints one is accepting (e.g. positing a subsystem whose duties would require it to be more "intelligent" than the supersystem of which it is to be a part) ...
>
> (2) mistaking conditional necessities of one's particular solution for completely general constraints...
>
> (3) restricting oneself artificially to the design of a subsystem... that is systematically incapable of being grafted onto other subsystems of a whole cognitive creature...
>
> (4) restricting the performance of one's system to an artificially small part of the "natural" domain of that system and providing no efficient or plausible way for the system to be enlarged.
> [*Brainstorms, pp. 112-113*]

These dangers were cited in reference to claims or cognitive or psychological plausibility. I have explicitly disavowed any claims that the structure or behavior of the diagrammer here described mirrors any human cognitive structure or function; nonetheless these objections to intelligent systems in general may be applicable to the present work, and must be considered.

(1) "...designing a system with component subsystems whose stipulated capacities are *miraculous* ..." This danger is that of hiding intelligence in some part of the system that is inexplicable -- it is the 'homunculus' charge. The safeguard against this is a hierarchical and modular structure of subsystems in which each subsystem is progressively less powerful and operates on less and simpler information. In the diagrammer, the executive control of the system is provided by the human designer -- outside of the system and at the top of the hierarchy. Each of the component subsystems works in isolation -- assigning defaults, resolving conflicts, making layout decisions -- without any particular global or "miraculous" intelligence. Indeed, this is why the diagrammer may become stuck and require assistance from outside. Relying on outside expertise, in the form of a human designer, is necessary and judicious; the purpose of the diagrammer is to interact, rather than to stand alone.

(2) " ...mistaking conditional necessities for completely general constraints..." This danger might be that of assuming, for example, that the human diagrammer depends on the same graphics primitives, and has the same graphics abilities, as the Macintosh QuickDraw routines that the CBD depends on. Now while the QuickDraw routines were devised on the basis of some experience with graphics requirements, there is no claim that they cover all the requirements the human designer; obviously, humans create diagrams all the time outside of the range of examples I have illustrated and that are capable of being produced by the CBD. Likewise, I make no claim that designers structure their knowledge into constraint terms for the purpose of designing or diagramming, much less that they do so in the terms and relations I've used here; I do claim that it is both possible and beneficial in many cases to do so.

(3) "...restricting oneself artificially to the design of a subsystem... systematically incapable of being grafted onto other subsystems... " This danger is the danger of producing a system incompatible or incongruent with any other system or systems. While Dennett's criterion was for a "whole cognitive creature", in the context of psychology and large claims for intelligent systems, I take the criterion to be for a "whole designing system". I have proposed above what some of the elements of such a system might be, and since the CBD shares a common data structure with the generic Constraint Model, and produces graphic artifacts that can be manipulated by other computer graphics systems, I propose that the CBD is not an isolated subsystem, but indeed lends itself well to integration. I described above the place of the CBD within a larger system of designing with diagrams.

(4) "...restricting performance to an artificially small domain ... providing no efficient or plausible way for the system to be enlarged..." The two parts of this danger are an artificially small domain, and no plausible way for the system to be enlarged. On the first count I may be guilty; it is possible that the choice of 'producing A-diagrams' from propositions is too limited a part of the role of

designing with diagrams. But as limited as it is, it is a big problem; and it is plausibly connected to a larger system which encompasses both reading diagrams and refining diagrams into detailed drawings. As to the latter charge, as with the previous danger, resting the CBD on the basis of the constraint Management provides a mechanism for extensibility and plausible enlargement.

Appendix 2. Non-Urban Design Diagrams

The emphasis of this dissertation has been on diagrams taken from the domain of urban design. Along the way, I made some general claims and observations about diagrams, and presented an approach to diagramming -- in the form of data structures and a skeletal algorithm -- that is not specific to any domain. The structure of diagrams as elements and relations, and their function as medium and stimulus for visual reasoning, are common to all diagrams. I'll demonstrate here three different examples of diagrams from other domains, and show how they can be explained by the terms and approach of this thesis, and produced by the CBD.

Physics: A Pulley Problem

In "Why a Diagram is (Sometimes) Worth Ten Thousand Words", [Larkin and Simon] explored the power of diagrams as problem-solving aides, and concluded that "diagrammatic" data structures, compared to "sentential", offer significant advantages for problem solving of certain types (I described their analysis in chapter seven). They illustrated their paper with two detailed examples of how a production system might converge on the solution of a problem, given an initial data structure (knowledge base) and rules for inference (rule-base); and demonstrated that construction and visual inspection of a diagram, with the attendant perceptual enhancement, offers advantages to the human problem solver and efficiencies over the production system. The first example in their paper was a problem of statics involving pulleys and ropes, where the goal was to find the relationship between two weights in a pulley/rope system in equilibrium. They showed that the production system could handle the problem with a small set of rules about ropes and pulleys, and that the human problem solver could solve the problem by constructing a diagram and applying versions of those same rules, using the diagram as a kind of analog computer in which relevant facts and rules were brought into play by visually following along the paths of ropes.

Attached are two figures showing the output of the CBD given the same propositional description of the problem as in the original paper. Table 10.1 shows the list of assertions that make up the sentential data structure for the pulley problem. Figures 10.1 and 10.2 show two resulting diagrams, the first one is a correct network diagram of the problem, and serves to solve the problem of determining the weight of W2 given the weight of W1 in equilibrium. The second diagram, more closely approximating that shown in the original text, was produced by adding two important pieces of knowledge: the constraint that all "ropes" are vertical, and the necessary information indicating the left-end and right end of the ropes and the left- and right- sides of pulleys. These two additions are sufficient to produce the diagram (through the mechanism described below) quite similar in appearance to the diagram shown in the original paper, described as "similar to the diagram drawn by most people confronted with the problem."

The process of the construction of the diagram depends on two parts: a library (data base) of generic objects with their built in properties; and a set of generic procedures for diagraming. Each of these is then complemented by special objects, constraints and customized procedures for the problem at hand. The custom objects "up-pulleys", "down pulleys", "1-pt-weights"and "2-pt-weights" with rope connection points were added to the CBD for this problem. The graphic object 'triangle" with a procedure for drawing it were also added, to graphically distinguish weights (trapezoids could also have been used, as in the original.) No special rules of diagramming are necessary, although as illustrated below, the knowledge that ropes under tension are vertical is useful. This is expressed as a constraint (dx=0) on the generic class "rope", simply a specialization of "line".

For example, the assertion "(rope RX)" first creates an object named RX, of type "rope", and draws a line of a standard unit length, with a slope of -1, since the two endpoints by

default are placed successively in the lower right corner. Which point is chosen is an arbitrary choice at this point. Now the assertion "(Connects RX PA0 PB1)" which says Rope X connects the 0 point (top) on PulleyA with the 1 point (left) of PulleyB , first produces two pulleys and their icons -- by default circles of a standard size -- and then tries to enforce the connection constraint that the endpoints of the rope are coincident with the specified connection points of the pulleys. The choice of which (rope or pulley) to move is still arbitrary, so long as all constraints are observed. The previous assertion also implicitly (and in our program, explicitly) states that "Pulley B is ABOVE Pulley A". This constraint is observed by adjusting the Y-coordinate of the center of one of the pulleys to be some arbitrary standard offset from the other pulley's centerpoint. This process is repeated until all ropes and pulleys are created and drawn. The result is shown in figure X, where some of the progression to the lower right corner is still observable, tempered by the "ABOVE" constraints. This diagram is "correct" in its representation of the connection of ropes, pulleys and weights, even though some of the ropes are crossed and the diagram doesn't resemble the one shown in the original paper. It is suitable for tracing out the solution of the problem, although some ambiguities arise when lines cross, and additional 'perceptual' abilities are called into play (for instance, being able to project the straight trajectory of a line segment.)

```
;; definitions
(is-rope rp)
(is-rope rq)
(is-rope rs)
(is-rope rt)
(is-rope rx)
(is-rope ry)
(is-rope rz)
(is-up-pulley pa)
(is-up-pulley pb)
(is-dn-pulley pc)
(is-weight w1)
(is-weight w2)

;; connections
(connects rp (w1 cp0) (pa cp1))
(connects rs (pc cp0) (w2 cp2))
(connects rx (pa cp0) (pb cp1))
(connects rq (pa cp2) (w2 cp1))
(connects rt (pb cp0) hanger1)
(connects rz (pc cp2) hanger2)
(connects ry (pb cp2) (pc cp1))
```

Table 10.1. Proposition List

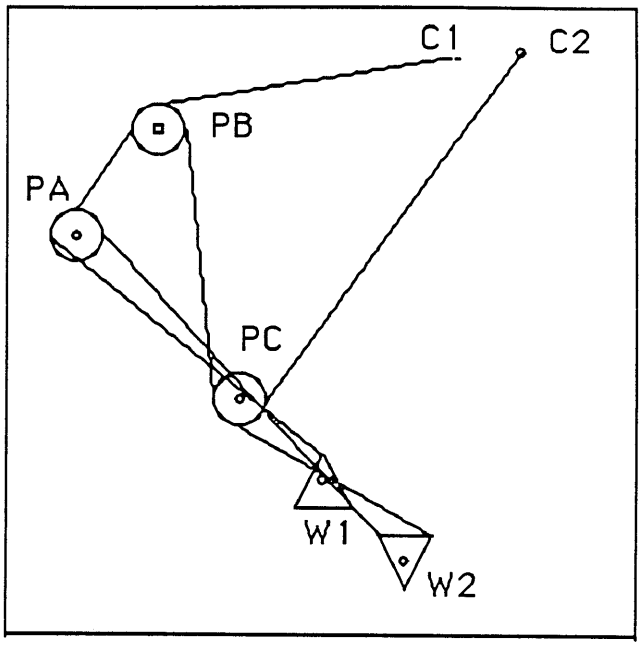


Fig. 10.1 Initial Diagram

Figure 10.2 shows the result of the same process with an additional piece of knowledge: all ropes have a "(DX= 0)" constraint (they must be vertical). This has the usual effect of eliminating crossed ropes, although the possibility of coincident ropes remains (A troublesome possibility -- see below.) In the connection of the ropes and pulleys, the endpoints of each rope are constrained to remain vertical, so that the pulleys or the rope must move horizontally to become lined up. The final result of diagramming given the vertical ropes constraint is very close to the diagram given in the original paper, in which all ropes are vertical and none are crossed.

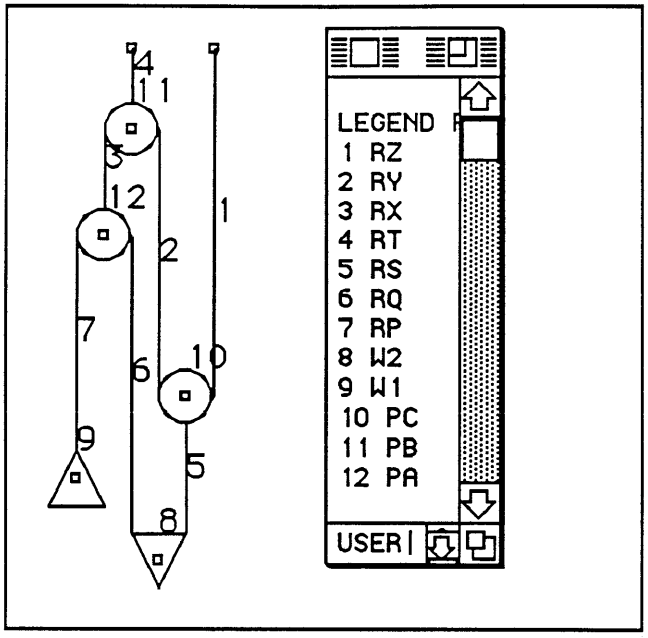


Fig. 10.2 Adjusted Diagram of Pulley Problem

Several other pieces of real-world knowledge are implicit or explicit to various degrees in the program and the knowledge base for producing the diagram. For example, where the original problem had two ropes RT and RZ connected to C (the ceiling), in our example it helps to specify two separate positions on the ceiling Hanger1 and Hanger2, that are constrained to be horizontal (and they could have a line between them for verisimilitude.) Also, making connection points on the weight the same distance apart as the connection points on pulleys helps to maintain the "vertical ropes" constraint -- the program is hopelessly stuck in conflict without this nicety. In fact, common-sense knowledge like "distances must be positive, therefore pulley-sizes must be positive" constitutes an implicit constraint on the system. Without such a constraint, the system will attempt to satisfy the verticality and connection constraints by giving weights and pulleys negative sizes (x-dimensions), in order to make ropes and pulleys line up[1].

[1]In fact, the LISP in which the system is implemented has full complex-number abilities, and the system has been observed to place ropes, pulleys and other objects in complex space in order to satisfy constraints! This of course causes problems for the graphics output module.

The diagram constructed by CBD is quite similar to that shown in the original article, and supports the claim that this method of producing diagrams is generic. In addition, since the diagram is related to a constrained-data structure, the addition of a simple constraint for the rules of physics allows the quantitative problem to be solved. By the use of the constraint "Tension in Rope0 = Tension in Rope1 + Tension in Rope2" in pulleys, a propagation network set up is such that the solution of the problem, by setting W1=1 , follows exactly the chain of logic as described by Larkin and Simon in their diagrammatic solution, to arrive at W2=5.

Line Graphs, Bar Charts, Pie Charts

Diagrams are ubiquitous in scientific, mathematical, economic and other arguments, especially in the form of the scatter-graph, line graph, bar chart and pie chart. Each of these constitute a mapping of usually non-spatial attributes and relations onto 2-dimensional space, frequently with a third attribute mapped onto a graphic attribute like shape or color. For example, a scientific chart showing a set of measurements, say time and temperature, on two samples, might map temperature to the y-coordinate, time to the x-coordinate, and sample identity to shape (say circles and squares). In addition, two diagrammatic elements are created, the labeled axis-lines that are related to the coordinate space of the diagram. In the result, an example shown in figure 10.3, is used to reveal the form of the correlation between time and temperature in the two samples, by analogy to the form of the clusters of symbols in the graphic output. In the example shown, sample 1 shows a linear correlation between time and temperature, sample 2 shows no obvious relationship (though some non-linear regularity might be found).

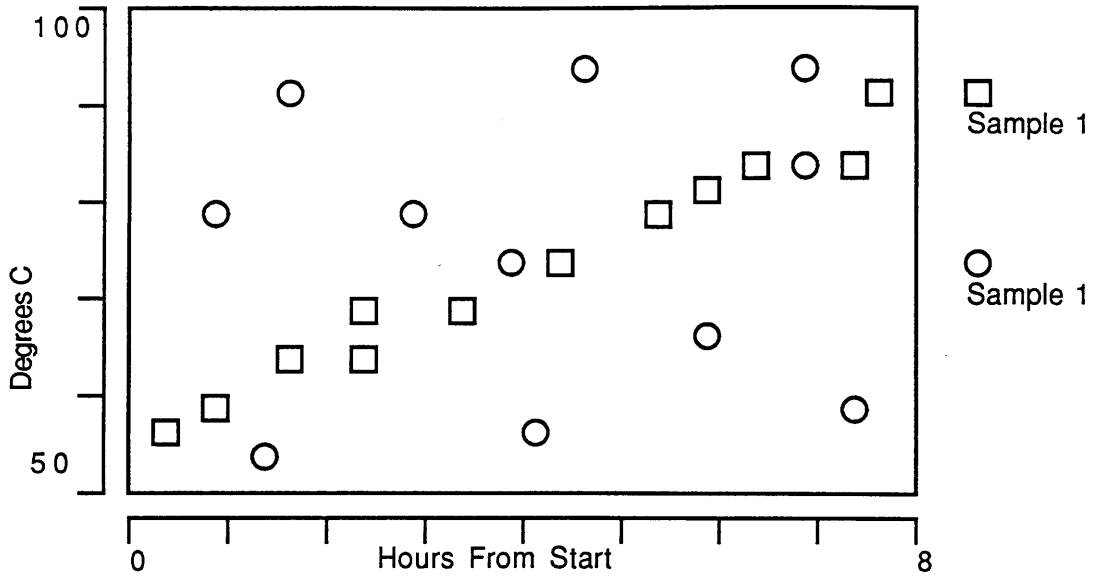


Figure 10.3 Simple Scatter Plot

This example can be produced by the CBD from a set of observations, a mapping from time and temperature to x- and y-coordinate, and the legend shown (the labels on the axes are conventional, and included in the mapping of the key.)

Organizational Diagram

Diagrams showing aggregations and connections are common organizational devices. The following diagram is taken from a paper describing a computer system for idea generation in design.

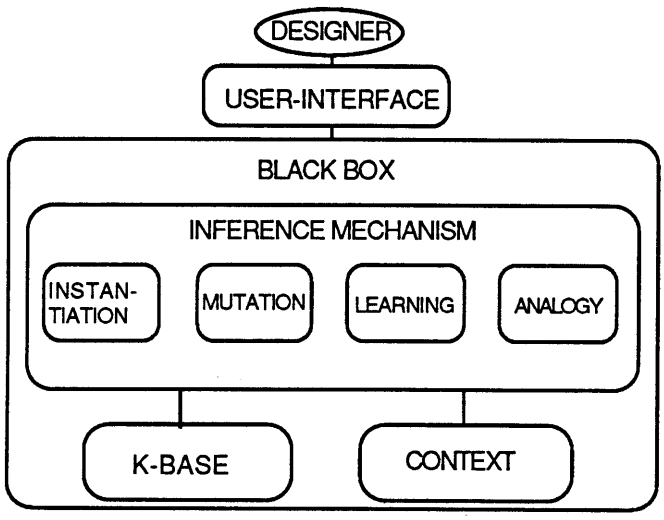
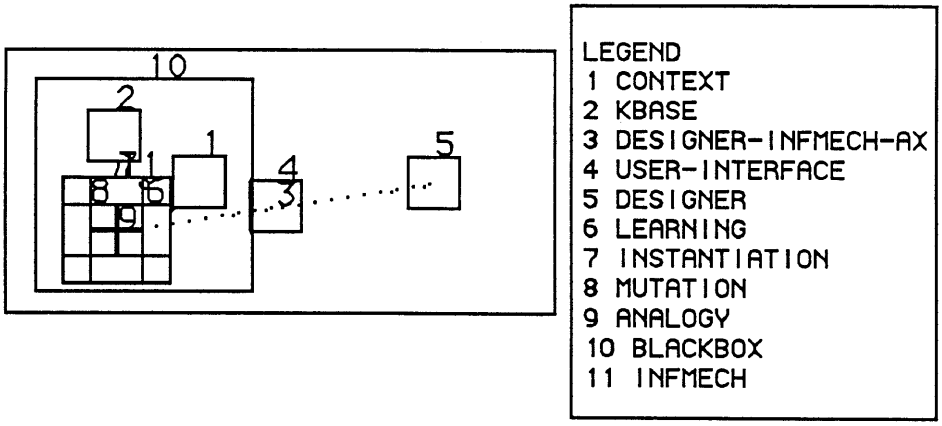


Figure 10.4 Organizational Diagram

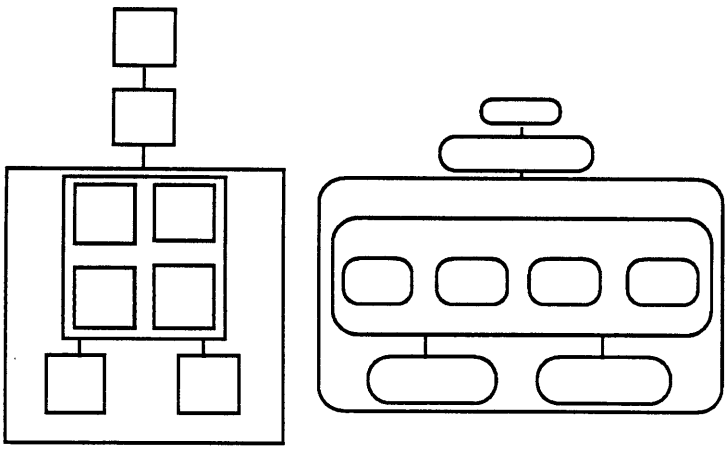
The diagram consists of a variety of parts, each with a connection or inclusion relationship to the others. The layout reveals vertical symmetry and the mapping uses rounded rectangles. A list of propositions that describe this diagram follows:

```
(contains blackbox
          inference-mechanism
         kbase
          context)

(contains  inference-mechanism
          analogy
          mutation
           instantiation
          learning)

(outside designer blackbox)
(between  user-interface
        designer
         inference-mechanism)

(connected kbase inference-mechanism)
(connected  context  inference-mechanism)
```

The relations are all in the lexicon of the CBD, and produce the initial diagram shown below:



As in the examples in chapter six, this differences between this diagram and the source can be explained with several simple moves. The first move is to separate all the elements so they are legible (in the layout procedure), and arrange elements labelled 5 and 4 (designer, and user interface) vertically aligned on the central axis (layout). This step is shown at the left below. Next elements 6,7,8,9 have to be aligned horizontally (layout), and element 1,2 have to be aligned horizontally and located underneath 6,7,8,9 (layout, and an additional proposition in the input, locating 6,7,8,9 "between" 1,2 and 4). Finally all the squares are given rounded corners (mapping) and resized to have different proportions (designer preference). The result of these moves is shown at the right below:



The point is not that particular diagrams can be produced by the CBD with some user intervention, for this can be done by any CAD system under user control. The point is rather to locate the knowledge and give names to the controls, that enable this production of diagrams. The terms and procedures of the the algorithm in chapter four, and the primitives in the lexicon of elements and relations, are seen to be sufficiently general to produce many different kinds of diagrams.

Appendix 3. Selected Listing of Constraint Definitions of Diagram Lexicon

```
;;  LISP definitions of selected DIAGRAM LEXICON relations:
;;    CONNECTS, SCREENED, SURROUNDS, OUTSIDE
; ;
;; each defintion has two parts:
;; the macro DEFRELATION installs the relation in the vocabulary,
;; states the acceptable 'types' of elements : points, lines or polys
;; gives a predicate form for CM-INFER,
;; and gives  the name of a generator, usually  MAKE-RELATION
;; the generator follows in a normal DEFUN

;;      **********************          CONNECTS      ******************************

(defrelation CONNECTS (A B C)
  ()     ;; no equivalent relations
  ((dline dpt dpt)  ;; acceptable types
   (dline dpoly dpoly)
   (dline dline dpoly)
   (dline dpoly dline)
   (dline dline dline))
  (ask the-diagram (cm-infer '(connects ^a ^b ^c))) ;; predicate  form
  (make-connector a b c) ;; default generator form
)

(defun make-connector (l p1 p2)
    (let* ((rname1 (gentemp "CONN-REL"))  ;; gentemp some relation names
          (rname2 (gentemp "CONN-REL"))
          (rname3 (gentemp "CONN-REL"))
          (rname4 (gentemp "CONN-REL"))
          (lname   (ask l obj-name))
          (p1name  (ask p1 obj-name))
          (p2name  (ask  p2 obj-name))
          (cause (list 'CONNECTS lname p1name p2name))) ;; make the 'cause' description
      (if (and (eq (dtype-of p1) 'DPT)
               (eq (dtype-of p2) 'DPT))
        (ask the-diagram   ;; if both elements are points,  connect them with a line
          (cm-command `(SAME-PT (,lname startpt) ,p1name))
          (cm-command `(SAME-PT (,lname endpt) ,p2name )))
        (ask the-diagram    ;; otherwise,  connect their centerpoints
          (cm-command `(SAME-PT (,lname startpt) (,p1name centerpt)))
          (cm-command `(SAME-PT (,lname endpt) (,p2name centerpt)))))
T)) ;; RETURN
```

```
;;    ********************        SCREENED-FROM    **************************
;; generates a new LINE element screening A from B

(defrelation screened-from (A B)
  () ;; no equivalents
  ((dpoly dpoly)) ;; acceptable types
   (ask the-diagram (cm-infer '(screened-from ^a ^b))) ;; predicate form
   (multiple-value-bind (lobj lname)
                  (ask the-diagram (new-dline))
        (cm-command `(screens ,lname ^a ^b ))) ;; default generator form
)

;; screens
(defun make-screen (l p1 p2) ;; l will be on axis
    (let* ((rname1 (gentemp "SCREEN-REL"))  ;; gentemp relation names
        (rname2 (gentemp "SCREEN-REL"))
        (lname   (ask l obj-name))
        (p1name  (ask p1 obj-name))
        (p2name  (ask  p2 obj-name))
          (axname (intern (format nil "~A-~A-AXIS"
                  p1name p2name)))
       (cause (list 'SCREENS LNAME p1name p2name))) ;; cause
    (cm-command `(IS-AN-AXIS ,AXNAME)) ;; create an axis
    (cm-command `(CONNECTS ,axname ,p1name ,p2name))  ;; connecting the two elements
    (cm-command `(SAME-PT (,lname centerpt) (,axname centerpt)))  ; screen is centred on
     (ask the-diagram
          (ask l (busy-retract '(startpt x))) ;; avoid circularity by overriding default centerpt
          (ask l (busy-retract '(startpt y)))
         (ask l (busy-fix 'linewidth  3 'default))
          (ask l (busy-fix 'pen *gray* 'default))
         (busy-add-relation rname1 `(<- (,lname vx)
                                  (compute-buffer-dx  ;; a procedure that finds the dx
                                         ,axname
                                         ,p1name
                                          ,p2name  ;; dummy vars
                                         (,axname length)
                                            (,p1name (bbox right)) (,p1name (bbox left))
                                           (,p1name (bbox top))  (,p1name (bbox bottom))
                                            (,p2name (bbox left))  (,p2name (bbox right))
                                            (,p2name (bbox top))  (,p2name (bbox bottom)))))
         (busy-add-relation rname2 `(<- (,lname vy)
                                  (compute-buffer-dy
                                         ,axname
                                         ,p1name
                                          ,p2name ;; dummy vars
                                         (,axname length)
                                            (,p1name (bbox right))  (,p1name (bbox left))
                                           (,p1name (bbox top))  (,p1name (bbox bottom))
                                            (,p2name (bbox left))  (,p2name (bbox right))
                                            (,p2name (bbox top))  (,p2name (bbox bottom)))))
    )
T)) ;; return value
```

```
;;    **********************        SURROUNDS    ****************************

(defrelation SURROUNDS (p1 p2)
  ()
  ((dpoly dpoly) (dpoly dline))
  (ask the-diagram (cm-infer '(surrounds ^p ^l)))
  (make-surrounding p1 p2)
)

(defun make-surrounding (p1 p2)
    (let* ((rname1 (gentemp "SURROUND-REL"))
        (rname3 (gentemp "SURROUND-PRED"))
         (pname1 (ask p1 obj-name))
         (pname2 (ask p2 obj-name))
         (cause (list 'DEFAULT 'SURROUNDS pname1 pname2 ))) ;; so names wont be objects
      (ask the-diagram
        (cm-command `(SAME-PT (,pname1 centerpt)
                              (,pname2 centerpt)))
;; and now add relations
         (busy-add-relation rname1 `(= (,pname1 dx)
                                       (+ (,pname2 dx)
                                          ,*default-x-inc*))
                            cause)

         (busy-add-relation rname3 `(? (POLY-INSIDE-P
                                        ,PNAME2 ,PNAME1 ;; THE REAL VARS
                                        (,PNAME1 (BBOX TOP))
                                        (,PNAME1 (BBOX BOTTOM))
                                        (,PNAME1 (BBOX LEFT))
                                        (,PNAME1 (BBOX RIGHT))
                                        (,PNAME2 (BBOX LEFT))
                                        (,PNAME2 (BBOX RIGHT))
                                        (,PNAME2 (BBOX TOP))
                                        (,PNAME2 (BBOX BOTTOM))))
                            `(,rname1)) ;; resolvers
)
T)) ;; return from surrounding
```

```
;;    ****************    OUTSIDE    ****************
(defun make-outside (p1 p2)
    (let* ((n (ask p2 (inc-outsidecount)))
        (rname0 (gentemp "OUTSIDE-REL"))
        (rname1 (gentemp "OUTSIDE-REL"))
        (rname2 (gentemp "OUTSIDE-REL"))
        (rname3 (gentemp "OUTSIDE-TEST"))
       (pname1   (ask p1 obj-name))
       (pname2  (ask p2 obj-name))
       (cause (list 'OUTSIDE pname1 pname2 ))) ;; so names wont be objects
     (ask the-diagram
       (ask p1 (busy-fix '(centerpt x) unassigned 'default))
       (ask p1 (busy-fix '(centerpt y) unassigned 'default))
      (ask p1 (busy-fix 'outsideno n '(PROPAGATED NIL))) ;; so p2 controls
       (busy-add-relation rname0
                          `(= (,pname1 dx)
                              (* (,pname2 dx) .25))
                          (cons 'default cause))
       (busy-add-relation rname1 `(<- (,pname1 (centerpt x))
                                  (nth-outside-xpos  ;; a generator procedure returns a good position
                                  (,pname2 dx) (,pname2 (centerpt x)) ,n
                                 (,pname2 outsidecount)
                                 (,pname2 START-ANGLE)))
                        (cons 'default cause))
       (busy-add-relation rname2 `(<- (,pname1 (centerpt y))
                                  (nth-outside-ypos
                                  (,pname2 dy) (,pname2 (centerpt y)) ,n
                                 (,pname2 outsidecount)
                                 (,pname2 START-ANGLE)))
                      (cons 'default cause))
      (busy-add-relation rname3 `(? (POLY-OUTSIDE-P
                                  ,PNAME1 ,PNAME2 ;; THE REAL VARS
                                  (,PNAME1 (BBOX TOP))  (,PNAME1 (BBOX BOTTOM))
                                  (,PNAME1 (BBOX LEFT))  (,PNAME1 (BBOX RIGHT))
                                  (,PNAME2 (BBOX LEFT))  (,PNAME2 (BBOX RIGHT))
                                  (,PNAME2 (BBOX TOP))  (,PNAME2 (BBOX BOTTOM))))
                     ` (,rname1 ,rname2)))
T))


;; required auxiliary procedures for OUTSIDE

(defun inc-outsidecount () ;; increment the internal outside counter
... )

(defun nth-outside-xpos (dx cpx n count &OPTIONAL (START-ANGLE 0))  ;; generator
;; return the x-coord of a good place to put something, also update max count
...)

(defun nth-outside-ypos (dx cpy n count &OPTIONAL (START-ANGLE 0)) ;; generator
;; return the y-coord of a good place to put something, also update max count
 ...)
```

References

Abel, C. 1982 "The role of metaphor in changing architectural concepts" in Evans, Barrie, J.A.Powell, and R. Talbot, (Eds) *Changing Design.* New York: John Wiley & Sons

Abelson, H., J. Sussman, J. Sussman 1985 *Structure and Interpretation of Computer Programs.* Cambridge, Mass: MIT Press

Agogino A M,1987, "AI in Computer-Aided Design: Qualitative Reasoning and Symbolic Computation", *Proc. NSF Oakland Conference on the Design Process* 189-214.

Agogino A M,1988, "Object Oriented Data Structures for Designing by Features", *Proc. NSF Grantee Workshop on Design Theory and Methodology*

Akin, Ö. 1986 *Psychology of Architectural Design.* London: Pion Limited

Albarn, Keith and J. M. Smith, 1979 *Diagram; The Instrument of Thought.* London, Thames and Hudson

Alexander, C. 1966 *Notes on the Synthesis of Form.* Cambridge: Harvard University Press.

Alexander, C. 1971 "The State of the Art in Design Methods", in Cross, N. (Ed.) 1984 *Developments in Design Methodology.* New York: John Wiley & Sons

Alexander, C., and M. Manheim,1962 *The Use of Diagrams in Highway Route Location: An Experiment* Research Report R62-3, Civil Engineering Systems Laboratory, MIT

Alexander, C., and M. Manheim,1962 *HIDECS2: A computer program for the hierarchical decomposition of a set which has an associated linear graph,* Research Report R62-2, Civil Engineering Systems Laboratory, MIT

Alexander,C. and B. Poyner, 1966 *The Atoms of Environmental Structure,* Center for Planning and Development Research, University of California, Berkeley

Alexander, C., S. Ishakawa, and M. Silverstein,1975. *A Pattern Language: Towns, Buildings, Construction.* New York: Oxford University Press.

Alexander, C. , H. Neis, A. Anninou, and I. King, 1987. *A New Theory of Urban Design.* New York: Oxford University press

Appleyard, D., K. Lynch, and J. Meyer, 1961. *The View from the Road.* Cambridge, Mass: MIT Press.

Appleyard, D., 1970. "Styles and Methods of Structuring a City", *Environment and Behavior* Vol 2 No. 1 June

Archer, L.B. ,1965. "Systematic Method for Designers", in Cross, N. (Ed.) 1984 *Developments in Design Methodology.* New York: John Wiley & Sons

Archer, L.B. 1968 "The structure of the design process" in Broadbent, G. and A.Ward (Eds.) 1969 *Design Methods in Architecture* London: Lund Humphries Publishers Limited

Arnheim, R. 1969 *Visual Thinking.* Berkeley: University of Califonia Press

Attneave, F. 1985, "Pragnanz and Soap Bibble Systems: A Theoretical Exploration" in Beck, J. 1985 (ed.) *Organization and Representation in Perception,* Hillsdale, N.J.: Erlbaum

Baglivo, J. and J. Graver, 1983 *Incidence and Symmetry in Design and Architecture,* Cambridge: The Cambridge University Press

Bailey, J. (Ed.) 1973 *New Towns in America: The Design and Development Process* New York: John WIley & Sons.

Bamberger, J. and D. Schön, 1978 "The Figural<->Formal Transaction", unpublished manuscript

Barnden, J.A. 1982 "A Continuum of Diagrammatic Data Structures in Human Cognition", Computer Science Department, Indiana University, Technical Report # 131

Beck, J. 1985 (Ed.)*Organization and Representation in Perception,* Hillsdale, N.J.: Lawrence Erlbaum Associates

Bell, A., L. Conway, M. Stefik, C. Tong,1981 "Toward a characterization of Abstraction", Stanford/PARC KB-VLSI Memo 81-2

Beyers, Gary, 1980? *Object Lisp* unpublished manuscript

Block, N. (Ed.) 1981 *Imagery* , Cambridge, Mass.: The MIT Press

Bobrow, D. 1981 "Sorting out the Abstraction Trees", Stanford/PARC KB-VLSI Memo 81-5

Bobrow, D. (Ed.) 1985 *Qualitative Reasoning about Physical Systems,* Cambridge, Mass.: The MIT Press

Bongard, M. M. 1970 *Pattern Recognition,* New York: Spartan Books

Borning, A. H. 1981. "Programming Language Aspects of ThingLab" in *ACM Trans. on Prog. Languages and Systems* vol 3 no 4 (Oct) pp 353-387.

Brachman, R. and H.Levesque (Eds.) 1985 *Readings in Knowledge Representation.* Los Altos, California: Morgan Kaufman Publishers

Breckenfeld, G. 1971 *Columbia and the New Cities.* New York: Ives-Washburn Inc.

Bregman, A.S. 1977 "Peception and Behavior as Compositions of Ideals", *Cognitive Psychology* 9, pp 250 - 292

Broadbent, G. and A.Ward (Eds.) 1969 *Design Methods in Architecture* London: Lund Humphries Publishers Limited

Broadbent, G. 1973 *Design in Architecture.* New York: Wiley

Brotchie, J.F., R. Sharpe and B. Marksjo, 1986 "Introducing Intelligence and Knowledge into CAD", Proceedings of the First International Conference on the Applications of AI to Engineering Problems

Bucciarelli, L. and D. Schön, 1987, "Describing and Modelling Designing and Design Knowledge in Architecture and Engineering", NSF Grant Proposal - unpublished

Chermayeff, S. and A. Tzonis, 1967 *Advanced Studies in Urban Environments: Toward an Urban Model.* Yale University, Institute for Applied Technology

Chouraqui, Eugene, 1984, "Computational Models of Reasoning", in Torrance, S. (ed.) *The Mind and the Machine: philosophical aspects of artificial intelligence. pp 145-155* Chichester: Ellis-Horwood Ltd

Churchman, C.W. 1971 *The Design of Inquiring Systems : basic concepts of systems & organization.* New York: Basic Books

Coles, L. S. 1972 "Syntax Directed Interpretation of Natural Language", in Simon and Siklossy (Eds.), *Representation and Meaning: Experiments with Information processing Systems,* New Jersey: Prentice-Hall, pp. 211 - 278

Crane, D. and Partners. 196? *Reston Town Center: Concept Plan and Program*

Cross, N. 1977 *the automated architect* , London: Pion Limited

Cross, N. (Ed.) 1984 *Developments in Design Methodology.* New York: John Wiley & Sons

Crowe, N. and P. Laseau 1984 *Visual Notes for Architects and Designers.* New York: Van Nostrand Reinhold Co.

David, R.E., 1971 "Proposal for a Diagrammatic Language for Design", in Kennedy, M. (Ed.) *Proceedings of the Kentucky Workshop in Computer Applications to Environmnetal Design*

de Kleer, J. 1986 "An Assumption-based TMS", *Artificial Intelligence,* Vol. 28, No. 1

Dennett, D. C. 1981 *Brainstorms: Philosophical Essays on Mind and Psychology.* Cambridge, Mass: MIT Press

Dickey, J.W. and G.G.Roy 1987 "C/SI: An Idea Generating Sytstem for Urban Planning" in Alexander, E. (Ed.) *Proc. 1987 Conference on Planning and Design in Urban and Regional Planning* 125-130

Doyle, J. 1982 "Expert Systems Without Computers, or Theory and Trust in Artificial Intelligence", *The AI Magazine Summer* 59-63

Dreyfus, H. 1984 "From Micro-Worlds to Knowledge Rewpresentation: AI at an Impasse", in Haugeland, J. (Ed.) *Mind Design: Philosophy, Psychology, Artificial Intelligence.* Cambridge, Mass.: The MIT Press pp 161-204

Eastman, C. 1968 *Explorations of the cognitive processes in design.* PhD Dissertation, Carnegie-Mellon University

Eastman, C. (Ed.) 1975 *Spatial Synthesis in Computer Aided Building Design.* New York: John Wiley & Sons

Eastman, C. 1978 "Representation of Design problems and maintenance of their structure" in *Artificial Intelligence and Pattern Recognition in Computer Aided Design,* pp 335-366. Ed. J. C. LaTombe. New York: North-Holland

Ervin, S. 1987, "Levels of Abstraction in Environmental Design: A Computational Approach", *Proceedings of the 1987 Conference on Planning and Design in Urban and Regional Planning* 91-96.

Ervin, S. and M. Gross,1987. "RoadLab - A Constraint-based Laboratory for Road Design" in *AI and Engineering: Planning and Design* Adey, R.A and D. Sriram (Eds.), Computational Mechanics Press

Evans, Barrie, J.A.Powell, and R. Talbot, (Eds) 1982 *Changing Design.* New York: John Wiley & Sons

Evans, T.G. 1968 "A Program for the Solution of A Class of Geometric-Analogy Intelligence-Test Questions" in Minsky, M. (Ed.) *Semantic Information Processing,* Cambridge, Mass: MIT Press pp 271-353

Fahlman, S.E. 1979 *NETL: A System for Representing and Using Real-world Knowledge.* Cambridge: MIT Press

Feigenbaum, E.A. and J. Feldman,(Eds.) 1963 *Computers and Thought.* New York: McGraw-Hill.

Fleck, M. 1988 *Boundaries and Topological Algorithms,* MIT Phd Thesis, 1988

Fleisher, A., W. Porter & K. Lloyd, 1969 "DISCOURSE: Computer Assisted City Design" in Milne, M. (Ed.) *Computer Graphics in Architecture and Design.* New Haven: Yale School of Art and Architecture

Fleisher, A. 1989 "Grammatical Architecture?", Final Report to NSF #DMC86-11375, MIT DTM Group

Fleisher, A. , S. Ervin & M. Gross 1989 "CM2: A contraint-based environment for design exploration", Final Report to NSF #DMC86-11375, MIT DTM Group

Friedman, G. J. & G. Leondes, 1969, "Constraint Theory, Parts I, II, III" *IEEE Transactions on Systems Science & Cybernetics* Vol SSSC-5 No 1, 2, 3.

Friedman, Y. 1980 *Toward a Scientific Architecture.* Cambridge, MA: MIT Press

Funt, Brian. 1980 "Problem Solving with Diagrammatic Representations", *Artificial Intelligence,* vol. 13, no. 3, 1980

Gentner, D. 1983 "Structure Mapping: A Theoretical Framework for Analogy", *Cognitive Science* Vol 7 pp. 155-170

Gill, H. 1982 "A descriptive and operational model for design" in Langdon, R. and P. Purcell (Eds.), *Design Theory and Practice: Proceedings of an international conference on design.* London: The Design Council

Ginsberg, M.L. (Ed.) 1987, *Readings in Nonmonotonic Reasoning.* Los Altos: Morgan Kaufman Publishers

Glinert, E. and S. Tanimoto, 1984 "PICT: An Interactive Graphical Programming Environment, *Transactions IEEE,* November 1984

Goldschmidt, G. 1989 "Architectural Sketching: Seeing As and Seeing That", NSF Final Report # DMC-8611357

Goodman, N. 1976 *Languages of Art.* Indiannapolis: Hacket Publishing

Graf, Douglas, 1985 "Diagrams", *Perspecta 22, Yale Architectural Journal*

Gross, M. and A. Fleisher, 1984, "Design as the Exploration of Constraints", *Design Studies* Vol. 3, No. 5.

Gross, M. , 1985, *Design as the Exploration of Constraints.* PhD Dissertation, MIT Department of Architecture

Gross, M., S. Ervin, J. Anderson, and A. Fleisher, 1987, "Designing With Constraints", in *The Computability of Design* Ed. Y. Kalay, New York: Wiley and Sons

Gross, M., S. Ervin, J. Anderson, and A. Fleisher, 1988, "Constraints: Knowledge Representation in Design", *Design Studies,* July 1988

Gutnov, A. , A. Baburov, et al., 1968*The Ideal Communist City.* New York: George Braziller

Habraken, N. J. 1983 *Transformations of The Site.* Cambridge, Mass.: Awater Press

Habraken, N.J. 1985 *The Appearance of the Form.* Cambridge, Mass.: Awater Press,

Halprin, L. 1969 *The R.S.V.P. Cycles; Creativity in the Human Environment.* New York: George Braziller

Haugeland, J. (Ed.) 1984 *Mind Design: Philosophy, Psychology, Artificial Intelligence.* Cambridge, Mass.: The MIT Press

Hawkes, D. (Ed.) 1975 *Models and Systems in Architecture and Building.* Lancaster: The Construction Press, Ltd.

Hayes, P.J. 1978 "In Defense of Logic", *Theorem -Proving 3* 559-565

Hayes-Roth, F. 1985 "Rule-Based Systems", *Communications ACM* Vol 28 No. 9 September 921-932

Heath, T. 1987 *Method in Architecture.* New York: John Wiley & Sons.

Herdeg, K. 1983 *The Decorated Diagram: Harvard Architecture and the Failure of the Bauhaus Legacy.* Cambridge, Mass: MIT Press.

Herdeg, W. (Ed.) 1981 *Graphis/Diagrams: The Graphic Visualization of Abstract Data* 4th ed. Zurich: Graphis Press Corp.

Higuchi, T. 1983 *The Visual and Spatial Structure of Landscapes,* Cambridge, Mass.: The MIT Press

Hillier, W. and A. Leaman, 1975 " "The Architecture of Architecture", in Hawkes, D (Ed.), *Models and Systems in Architecture and Building.* Lancaster: The Construction Press, Ltd.

Hillier, W. and J. Hanson, 1984 *The Social Logic of Space.* Cambridge: The Cambridge University Press

Holland, J.H., K Holyoak, R. Nisbett, P. Thagard 1986 *Induction: Prcoesses of Inference, Learning and Discovery.* Cambridge, Mass: MIT Press.

Indurkhya, B. 1987 "Approximate Semantic Transferance: A computational theory of metaphors and analogies" *Cognitive Science* Vol 11, No. 4 pp. 445-480

Jackendoff, R. 1987 "On Beyond Zebra: the relation of linguistic and visual information" *Cognition* vol. 26 no. 2, July 1987 pp. 89-114

Jaffar, J. and J.L. Lassez, 1987, "Constraint Logic Programming", *Proc 14th ACM Symposium Principles of Programming Languages* pp 111-119.

Jones, J. C. and D.G. Thornley, 1963, (Eds) *Proceedings of the Conference on Systematic and Intuitive Methods in Engineering, Industrial Design, Architecture and Communications, London, 1962.* New York, Macmillan Co.

Jones, J. C. 1970 *Design Methods.* London: Wiley-Interscience

Jones, J.C. 1969 "The State of the Art in Design Methods" in Broadbent, G. and A.Ward (Eds.) 1969 *Design Methods in Architecture* London: Lund Humphries Publishers Limited

Kalay, Y. 1987 *Computability of Design.* New York: Wiley & Sons

Kasmar, J. 1970 "The Development of a Usable Lexicon of Environmental Descriptors", *Environment and Behavior,* Vol. 2. No. 2, September

Koffka, K. 1935 *Principles of Gestalt Psychology.* New York: Harcourt Brace Jovanovich

Kosslyn, S. M and J. R. Pomerantz, 1977 "Imagery, propositions and the Form of Internal Representations" *Cognitive Psychology* 9, pp 52-76

Langdon, R. and P. Purcell, 1982 *Design Theory and Practice: Proceedings of an international conference on design.* London: The Design Council

Langley, P., H. Simon, G. Bradshaw, J. Zytkow, 1987 *Scientific Discovery: Computational Explorations of the Creative Processes.* Cambridge, Mass: MIT Press.

Larkin, J and H. Simon, 1987 "Why a diagram is (sometimes) worth 10,000 words", *Cognitive Science* Vol 11, pp 65-99

Laseau, P. 1980 *Graphic Thinking for Architects and Designers.* New York: Van Nostrand Reinhold Co.

LeCorbusier, 1943 *The Athens Charter.* (1973 translation) New York : Grossman Publishers

Leeuwenberg, V. 1985 , "Metrical Patterns and Structural Information Theory", in Beck, J. 1985 (Ed.) *Organization and Representation in Perception,* Hillsdale, N.J.: Lawrence Erlbaum Associates

Leler, W. 1987 *Constraint Language Programming.* Boston: Addison Wesley.

Lenat, D.B. 1984 "Computer Software for Intelligent Systems",*Scientific American*, Vol 251, No. 3 September

Lindsay, R. K. 1988 "Images and inference" *Cognition* vol. 29 no. 3, August 1988 pp. 229 - 250

Liu, K.C and K.S. Wong. 1988. "Automatic Generation of Process Flow Diagrams", *Computers and Graphics* Vol. 12, No3/4, pp 525-539

Livingstone, M.S. 1978 "Art, Illusion and the Visual System", *Scientific American* June 1978

London City Council, 1963, *Hook New Town Study*

Lynch, K. 1982 *Good City Form.* Cambridge, Mass: MIT Press.

Lynch, K. 1960 *The Image of the City.* Cambridge, Mass: MIT Press.

Lynch, K. and G. Hack. 1983 *Site Planning (3rd ed.)* Cambridge, Mass: MIT Press.

M'Pherson, P.K. "Thinking with Pictures -- graphics as aids to complex system design and policy analysis" in Langdon, R. and G. Mallen *Design and Information Technology: Proceedings of an international conference on design.* London: The Design Council pp 26-29

Manheim, M.L. 1966 "Problem Solving Processes in Planning and Design", *Design Quarterly* 66/67 December 31-40

Manheim, M.L. 1966 "Role of the Computer in the Design Process", *Building Research* 3 March/April No. 213-17

March, L. and P. Steadman, 1971 *The Geometry of Environment.* Cambridge, Mass.: The MIT Press

Marr, D. 1982 *Vision*, San Farncisco: Freeman

Maxwell, J.C. 1910, "Diagrams" in *The Encyclopedia Britannica,* Vol. 4, 11th Ed. 1910 pp. 146-149

McCall, R.J. 1986 "Issue-Serve Systems: A Descriptive Theory of Design", in *Design Theories and Methods* Vol. 20, No. 3

McDermott, D. 1981 "Artificial Intelligence Meets Natural Stupidity", in Haugeland, J. (Ed.) *Mind Design,* Montgomery Vermont: Bradford Books Publishers Inc. pp 143-160.

Miller, G.A. "The Magic Number Seven Plus or Minus Two: Some Limits on Our Capacity for Information Processing", *Psychological Review,* Vol. 63, No. 2, Mar/Apr. 1956 pp. 11-20.

Milne, M. 1966 *Computer-Aided Design: An experiment with a method...,* University of Oregon Department of Architecture, Eugene: University of Oregon Press

Milne, M. (Ed.) 1969 *Computer Graphics in Architecture and Design.* New Haven: Yale School of Art and Architecture

Minsky, M. 1975 "A Framework for Representing Knowledge", in Winston, P. (Ed.) *The Psychology of Computer Vision,* McGraw-Hill

Mitchell, W. J. 1970 "Computer Aided Spatial Synthesis" *5th Annual Urban Symposium, Papers on the Application of Computers to the Problems of Urban Sociaety.* New York: Association for Computing Machinery, 101-121.

Mitchell, W.J. 1986 "Formal Representations: a foundation for computer aided architectural design", Environment and Planning B, Volume 13(2)

Montalvo, F. 1985 "Diagram Understanding: the Intersection of Computer Vision and Graphics", *MIT AI Lab Memo No. 873.*

Montalvo, F. 1986 "Diagram Understanding: Course Notes", MIT Media Lab, unpublished

Moore, G. (Ed.) 1970 *Emerging methods in Environmental Design and Planning.* Cambridge, Mass.: The MIT Press

Navinchandra, D. and Marks, D. 1987 "Design Exploration through Constraint Relaxation" in *Expert Systems in Computer Aided Design,* Ed. J. Gero. North Holland

Navon D. 1977 "Forest before Trees: The Precedence of Global Features in Visual Perception", *Cognitive Psychology* 9, pp 353-383

Negroponte, N. 1970 *The Architecture Machine: Toward A More Human Environment.* Cambridge, Mass.: The MIT Press

Negroponte, N. and Groisser, L. 1970 " URBAN5 - A Machine that Discusses Urban Design." in Moore, G. (Ed.) *Emerging methods in Environmnetal Design And Planning.* Cambridge, Mass.: The MIT Press

Negroponte, N. 1975 *Soft Architecture Machines..* Cambridge, Mass.: The MIT Press

Negroponte, N. (Ed.) 1975 *Reflections on Computer Aids to Design and Architecture* . New York : Petrocelli/Charter

Nelischer, M. and D. Hinde 1985 "A Graphic Language for Designers", *Landscape Architecture,* July/August. pp. 60-63

Newell, A. and H. Simon 1972 *Human Problem Solving.* New Jersey: Prentice Hall

Newell, A. 1979 "Reasoning, Problem-Solving and Decision Processes: The Problem Space as a Fundamental Category", Department of Computer Science, Carnegie Mellon University, CS - 79 - 133

Oxman, R. and J. Gero, 1987 "Using an expert system for design diagnosis and design synthesis", *Expert Systems* February Vol. 4, No. 1

Pinker, S. (Ed.) 1982 *Visual Cognition,* Cambridge Ma: MIT Press

Pollock, J. 1987 "Defeasible Reasoning", *Cognitive Science* Vol 11, No. 4 pp. 481 -518

Porter, W. L. 1974 "DISCOURSE: ", PhD Dissertation, Department of Architecture, MIT

Porter, W. L. 1988 "Reconstructing the Logic of Architectural Design: Two Thought Experiments", *Design Studies* July 1988.

Porter, W. L. and W. McMains. 1975 "Notes on City Design Thinking", in N. Negroponte (Ed.) *Reflections on Computer Aids to Design and Architecture* pp 37-49.

Preziosi, D. 1979 *The Semiotics of the Built Environment: An Introduction to Architectonic Analysis.* Bloomington: Indianna University Press

Pylyshyn, Z. 1984 *Computation and Cognition: Towards a Foundation for Cognitive Science.* Cambridge, MA: MIT Press

Reiter, R. and G. Criscuolo. 1981 "On Interacting Defaults", Proc. 7th IJCAI, pp 270-276

Retz-Schmidt, G. 1988 "Various Views on Spatial Prepositions" , *A.I. Magazine* Summer pp. 95 -105

Robins, E. 1988 "Drawing and The Social Production of Architecture" in P. Knox (Ed.) *The Design Professions and the Built Environment.* New York: Nichols Publishing Co.

Rowe, P. 1987 *Design Thinking.* Cambridge, Mass.: The MIT Press

Saund, E. 1988 *The Role of Knowledge in Visual Shape Representation* , MIT Phd Thesis, 1988

Schön, D. 1983 *The Reflective Practicioner; how professionals think in action.* New York: Basic Books

Schön, D. and P. Purcell, (Eds.) 1984 *"Design Studies" (Vol 5/3)* . Guildford: Butterworth & Co.

Schön, D. 1988 "Designing: Rules, Types and Worlds" *Design Studies* July 1988. Guildford: Butterworth & Co.

Serrano, D. and Gossard, D. 1987 "Constraint Management in Conceptual Design" in *AI and Engineering: Planning and Design* Ed. R.A. Adey and D. Sriram. Computational Mechanics Press.

Simon, H. 1969 *The Sciences of the Artificial.* Cambridge: MIT Press.

Sowa, J. 1984 *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley.

Spillers, W. (Ed.), 1975 *Basic Questions of Design Theory* . New York: North Holland

Stauffer, L.A. and D.G. Ullman, 1988 "A comparison of the results of empirical studies into the mechanical design process", *Design Studies,* Vol. 9. No. 2 April 1988 pp 107-114

Steele, G. J. and G. Sussman, 1979, "CONSTRAINTS - A Language for Expressing Almost-Hierarchical Descriptions", *Artificial Intelligence* 14:1-39

Stiny, G. and J. Gips 1978 *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts.* Los Angeles: University of California Press

Stiny, G. and L. March 1981, "Design machines", *Environment and Planning B,* volume 8, pp 245-255

Stiny, G. 1985 , "Computing with Form and Meaning in Architecture", Journal of Architectural Education Fall 1985

Sussman, G.J. and G.L. Steele, 1980 "Constraints: A language for elxpressing almost-hierarchical descriptions", *Artificial Intelligence,* 14 , 1-39

Sutherland, I. 1963, "Sketchpad - a Graphical Man-Machine Interface", M.I.T. Ph.D. Dissertation

Thiel, P. 1961, "A Sequence-Experience Notation" in *Town Planning Review,* April 1961

Torrance, S. (Ed.) *The Mind and the Machine: philosophical aspects of artificial intelligence.* Chichester: Ellis-Horwood Ltd

Touretsky, D.S. 1987 "Implicit Ordering of Defaults in Inheritance Systems" in Ginsberg, M.L. (Ed.) *Readings in Nonmonotonic Reasoning.* Los Altos: Morgan Kaufman Publishers pp. 106-109

Tuan, Y. 1974 *Topophilia: A study of Environmental Perception, Attitudes, and Values.* New Jersey, Prentice-Hall.

Ullman, D. , S. Wood & D. Craig, 1989 "The Importance of Drawing in the Mechanical Design Process", in *Proc. NSF Engineering Design Research Conference*

Ullman, S. 1982 "Visual Routines", in Pinker, S. (Ed.) *Visual Cognition,* Cambridge Ma: MIT Press pp 97-156

Venn, J. 1971 *Symbolic Logic (2nd Ed.)* Bronx, NY: Chelsea Publishing Co.

Vlietstra, J. and R. F. Wielinga, 1973 *Computer Aided Design: Proceedings of the IFIP Working Conference on Principles of Computer Aided Design.* Amsterdam: North-Holland Publishing

Weinzapfel, G. , T. Johnson and S. Handel 1975. "IMAGE: Computer Assistant for Architectural Design" in *Spatial Synthesis in Computer-Aided Building Design* Ed. C. Eastman pp61-68 New York: Wiley

Woodbury, R. 1988. "The Knowledge Based Representation and Manipulation of Geometry", PhD Dissertation, Carnegie Mellon University

Winston, P. 1984 *Artificial Intelligence.* Reading: Addison-Wesley

Wittgenstein, L. 1922 *Tractatus Logico-Philosophicus* (Pears & McGuiness trans.) New York: The Humanities Press

Wos, L. 1988 *Automated Reasoning: 33 Basic Research Problems.* New Jersey: Prentice Hall

Wurman, R.S. 1974 *Cities: Comparisons of Form and Scale.* Philadelphia: Joshua Press

Yessios, C. 1975 "Formal Languages for Site Planning", in Eastman, C. (Ed.) *Spatial Synthesis in Computer Aided Building Design.* New York: John Wiley & Sons

Zeigler, B., and R. Ada, 1984, "Abstraction in Methodology: A Framework for Computer Support", *Information Processing and Management* 20:63-79.

Zimmer, J. 1985 *Abstraction for Programmers.* New York: McGraw Hill.

Biographical Note

The author holds a Master's degree in Landscape Architecture, from the University of Massachusetts at Amherst, and has taught design and computing in the Landscape Architecture program at UMASS, in the School of Architure and Planning at MIT, and at Harvard University GSD.

Two of the examples in this thesis, and many of the ideas, have separately appeared in:

Ervin, S. 1987 "Levels of Abstraction in Urban Planning and Design", *Proceedings of the 1987 ASME Conference on Planning and Design in Urban and Regional Planning* 91-96.

Ervin, S. 1989 "Computer-Aided Diagramming: A Role for Computing in Design Education", *Proceedings of the 1989 Harvard GSD CAAD Futures Conference*

Ervin,S. and J. Anderson 1988 "A Constraint-Based Graphical Editor", proposal to the Apple/ICEC fellowship competition.

Fleisher, A. , S. Ervin & M. Gross 1989 "CM2: A contraint-based environment for design exploration", Final Report to NSF #DMC86-11375, MIT DTM Group

Gross, M., S. Ervin, J. Anderson, and A. Fleisher 1987, "Designing With Constraints", in *The Computability of Design* Ed. Y. Kalay, New York: Wiley and Sons

Gross, M., S. Ervin, J. Anderson, and A. Fleisher 1988, "Constraints: Knowledge Representation in Design", *Design Studies*, July 1988

Apple Computer Co., Coral Software Co., MIT's Project Athena, the School of Architecture and Planning at MIT and NSF Grant #DMC86-11375 all contributed to the material support of the author and the development of the ideas in this thesis.