9
Top-Down Knowledge-Based Design

William J. Mitchell
Harvard University Graduate School of Design

Robin S. Liggett
Graduate School of Architecture and Urban Planning
University of California, Los Angeles

Milton Tan
Harvard University Graduate School of Design

Traditional computer drafting systems and three-dimensional geometric modeling systems work in bottom-up fashion. They provide a range of graphic primitives, such as vectors, arcs, and splines, together with operators for inserting, deleting, combining, and transforming instances of these. Thus they are conceptually very similar to word processors, with the difference that they operate on two-dimensional or three-dimensional patterns of graphic primitives rather than one-dimensional strings of characters.

This sort of system is effective for input and editing of drawings or models that represent existing designs, but provides little more help than a pencil when you want to construct from scratch a drawing of some complex object such as a human figure, an automobile, or a classical column: you must depend on your own knowledge of what the pieces are and how to shape them and put them together. If you already know how to draw something then a computer drafting system will help you to do so efficiently, but if you do not know how to begin, or how to develop and refine the drawing, then the efficiency that you gain is of little practical consequence. And accelerated performance, flashier color graphics, or futuristic three-dimensonal modes of interaction will not help with this problem at all.

By contrast, experienced expert graphic artists and designers usually work in top-down fashion-beginning with a very schematic sketch of the whole object, then refining this, in step-by-step fashion, till the requisite level of precision and completeness is reached. For example, a figure drawing might begin as a "stick figure" schema showing lengths and angles of limbs, then be developed to show the general blocking of masses, and finally be resolved down to the finest details of contour and surface. Similarly, an architectural drawing might begin as a parti showing just a

skeleton of construction lines, then be developed into a single-line floor plan, then a plan showing accurate wall thicknesses and openings, and finally a fully developed and detailed drawing.

"How to" drawing manuals often demonstrate this top-down stepwise refinement process in very direct and explicit fashion. For example, the famous sketchbooks of Villard de Honnecourt, Albrecht Dürer, and Leonardo da Vinci all show examples of schemata for various types of objects, together with demonstrations of how to elaborate these into fully-developed drawings (figure 1). Dürer's *Four Books of Human Proportion (1528)* sets out the method, as applied to human faces and figures, in systematic, textbook fashion. Similarly, J-N-L. Durand's architectural design method, as illustrated in the plates in his *Précis (1802)* and *Partie Graphique (1821),* proceeds from a highly schematic parti through a sequence of refinement steps (figure *2).* In our own time, the books of tips for amateur artists that you can find in artists' supply stores are usually organized in much the same way.

Analogous methods are widely used in computer programming and in the development of text documents. Top-down stepwise refinement is a standard technique of structured programming, and is encouraged by the structure of languages like Pascal. The concept of a word processor has been elaborated into that of an outline processor, which supports refinement of a schematic set of headings into subheadings, and finally complete and finished text.

The idea of a computer-aided design system that supports stepwise refinement of a schematic idea into a complete and detailed design is, then, a plausible and

Figure 1 A sketch by Albrecht Dürer showing elaboration of stick-figure schemata

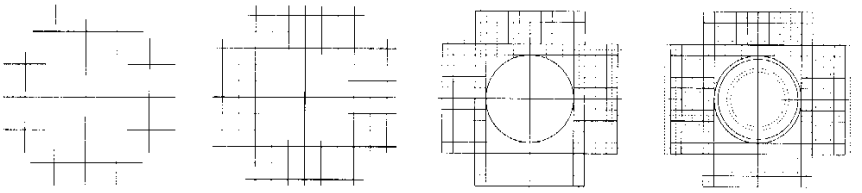

Figure 2 Durand's method for refinment of plan schemata

attractive one. Furthermore, if such a system provides a library of schemata for various types of artifacts, together with rules for refinement of schemata, it can effectively deliver knowledge of how to design these artifacts. The system becomes not merely an efficient design medium, but a design consultant that supplies at each step in the design process appropriate suggestions about what to do next. Schemata and development rules can evolve overtime as they are tested in practice-much as E. H. Gombrich suggested in Art *and Illusion (1969).*

This paper describes a prototype top-down, knowledge-based design system, and illustrates its application. The system is called *Topdown.* Versions have been developed in Lightspeed Pascal for the Macintosh and in Microsoft Pascal under Microsoft Windows for the IBM P5/2, and these have been extensively used in teaching at the Harvard Graduate School of Design and the UCLA Graduate School of Architecture and Urban Planning.

Topdown can be described at three levels. From the user's viewpoint it is a highly interactive graphics system controlled by mouse operations. From the programmer's viewpoint it is an environment for encoding knowledge of how to draw or design things. At the substructure level it is a piece of software, coded in Pascal, which maintains a data structure and provides a graphics interface. We will consider these in turn.

The User Level

A designer working at the user level sees the screen shown in figure 3. There are two graphic windows-the peek window and the poke window. The peek window always shows the current state of the design, depicted as a two-dimensional composition of lines and color-filled polygons. There are the usual pan, zoom, and other basic display functions. The peek window is for viewing only: the user cannot directly select or operate upon graphic objects displayed within it.

All design interaction is accomplished via the poke window. The designer only needs to know two basic moves: substitution of a more detailed representation for a less detailed one, and parametric variation of dimensions, angles, colors, and so on. Substitution is accomplished by clicking in the poke window on the part which is to be replaced. This brings up a dialog box showing the substitution options (figure 4). When an option is selected by clicking in the dialog box it is displayed in situ in the

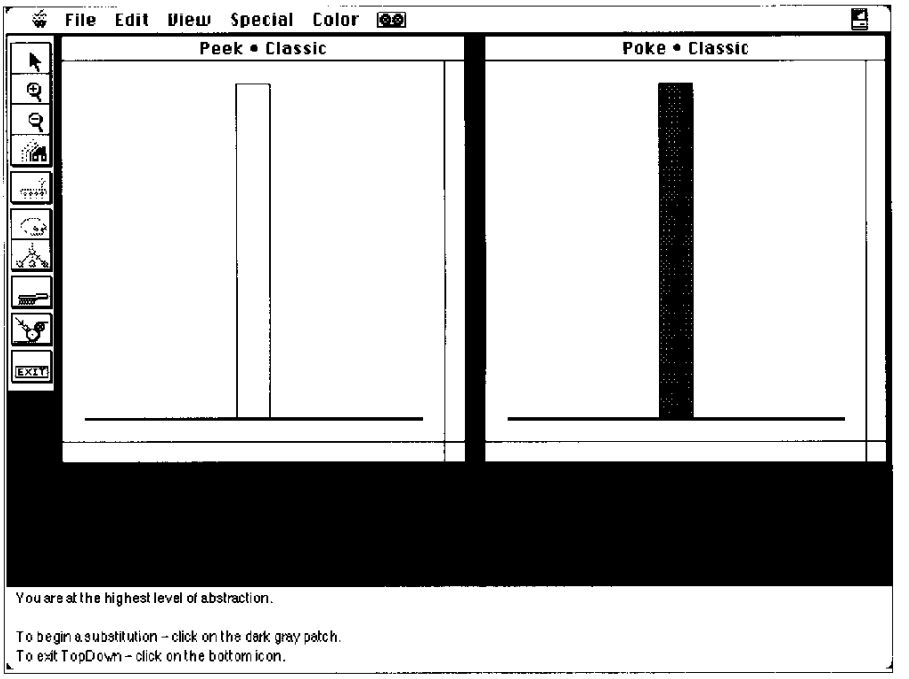

Figure 3 The Topdown screen with peek, poke, and message windows

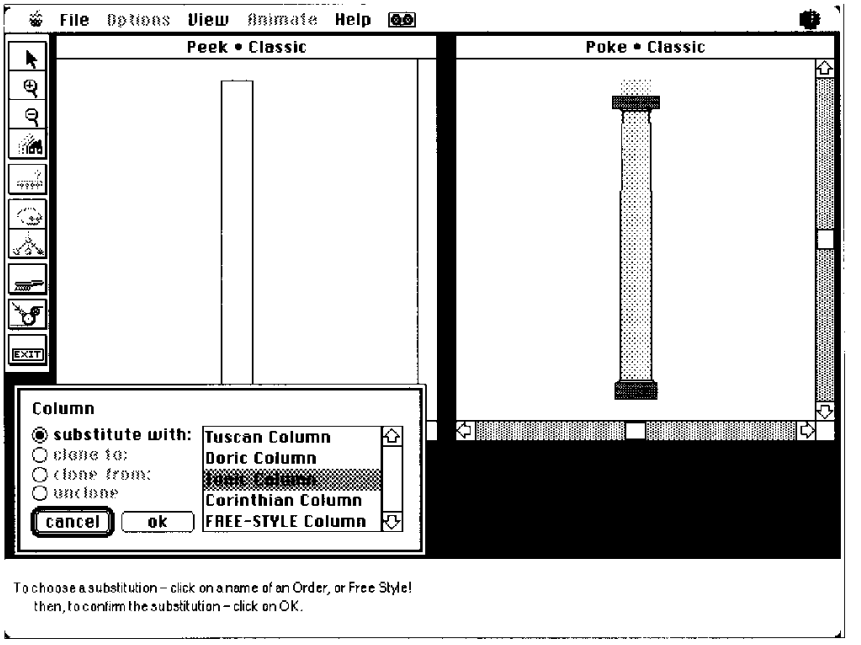

Figure 4 The substitution dialog box

poke window. Then, when ok is clicked, the selected option is substituted in the current version of the design that is displayed in the peek window, and the dialog box disappears.

Parametric variation of dimensions is accomplished by clicking on a dimension line in the poke window to bring up a control bar with a slide bar (figure 5). Dimensions are then varied, in real time, by means the slide bar. The control bar displays the name and current value of the variable that is being manipulated. When ok is clicked the current value of the variable is entered in the data structure and the control bar disappears.

The peek window always shows the design with all the detail that has so-far been established, but the user can set the poke window at any level of abstraction. Thus by setting the poke window at a high level of abstraction the user can make global decisions-to readjust overall dimensions of an object, for example-and immediately see the effects of these propagated to the details shown in the peek window. Conversely, by setting the poke window at a low level of abstraction, the user can work on details within the framework of an established design. This freedom to move freely between and work at different levels of abstraction is not provided either by physical media or by traditional CAD systems.

At all times a message window is open at the bottom of the screen. This provides

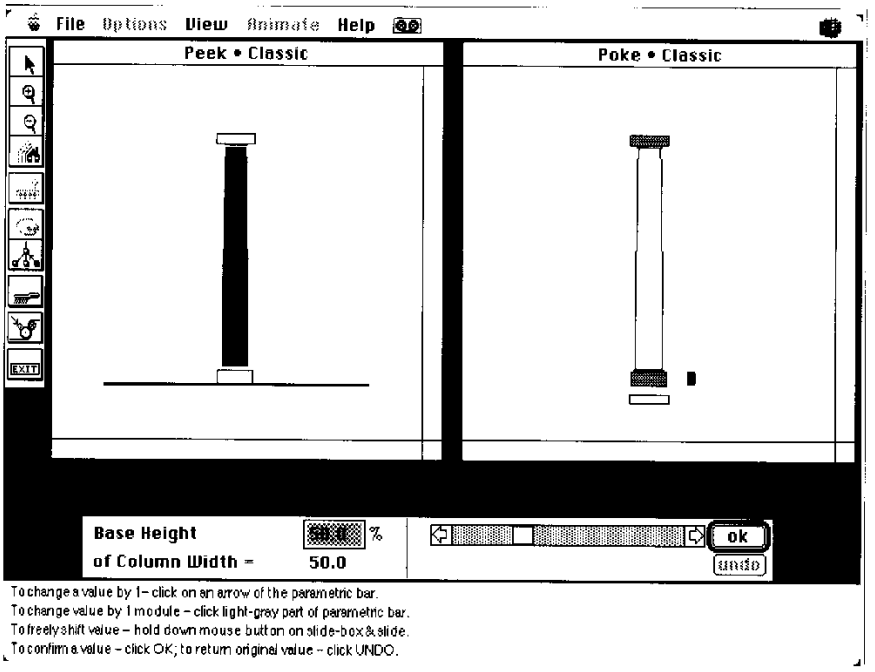

Figure 5 The control bar for parametric variation

a continuous stream of context-sensitive advice and suggestions, critical comments, and results of relevant analyses (such as area and cost calculations). It can, of course, be ignored-but some appropriate advice is always there when the user happens to give it a glance.

Essentially, then, the user sees the current state of the design in the peek window, variables and options in the poke window, and advice and criticism in the message window.

The Programmer Level

The programmer's task, in constructing a *Topdown* knowledge base, is to capture and encode knowledge of the shapes and sizes of elements and subsystems, and of how to put them together. In other words, the programmer must encode the vocabulary and syntactic rules of a parametric shape grammar (Stiny, 1980) for the type of artifact that is to be designed. The user is then able to explore designs in the language specified by that grammar.

The first step in constructing a knowledge base is to specify the vocabulary of parameterized shapes from which designs are to be assembled. Secondly, rules specifying which shapes may be substituted for a given shape must be defined. Finally, the positions and sizes of substituted shapes must be related to the positions and sizes of their predecessors by specifying how parameter values will be inherited by substituted shapes. If a detailed depiction of a window is to be substituted for a simple rectangle, for example, the window's width and height should be inherited from that rectangle.

This sort of programming might, in principle, be carried out in many different ways. In current versions of Topdown, which are used for teaching programming, the programmer sees an extension of Pascal. Shapes are represented by parameterized procedures (using the approach described by Mitchell, Liggett, and Kvan 1987), and there are facilities for specifying substitution possibilities and paths of inheritance of parameter values. A more sophisticated production version would take greater advantage of techniques of object-oriented programming, and would provide a point-and-click graphic method for specifying shapes and shape substitution rules.

The Substructure Level

Knowledge bases for design of different types of artifacts can be plugged into the Topdown shell-a piece of software that provides the interface between the user and the rules encoded by the programmer. The shell presents shape substitution rules as selectable shapes in the poke window together with substitution options in the substitution dialogue box, and it presents dimensioning variables as dimension lines in the poke window which can be selected to invoke the slide bar that is used to assign a value. It also records the user's design decisions in a data structure, and

displays the current state of the design in the peek window. Finally, it provides file import, export, and print utilities.

In order to present shape substitution possibilities and perform substitution operations the shell must have some kind of structural description of a developing design as a collection of subshapes of various types in various relations to each other. Current versions of *Topdown* organize designs as strict hierarchies of subshapes within subshapes, and require the programmer to predefine this hierarchical structure. Thus they do not capture the full power and generality of the shape grammar formalism (Stiny, 1982). We expect that more sophisticated future versions will incorporate unrestricted subshape recognition capabilities, and thus allow structural descriptions to be created dynamically during a design process. Some parallel research is exploring ways to accomplish this (Nagakura 1989, Tan 1989).

The shell is designed to provide real-time performance: substitutions are performed instantly, and movement of the slide bar results in simultaneous resizing of affected elements in the peek and poke windows. This provides a qualitatively different experience from that of systems which exhibit perceptibly delayed response, and we believe that the tactile character and immediate feedback provided by a real-time system are important both to comprehension and to fluidity and spontaneity in design exploration. Designers come to think of drawings not as basically static objects that can be edited in discrete steps, but as dynamic mechanisms that can be adjusted fluently to the particular state that is desired.

In order to accomplish real-time performance on the chosen hardware platforms it was necessary to accept certain limitations. *Topdown is* two-dimensional, it can only handle designs of limited size and complexity, and at the programmer level it does not take advantage of high-level programming constructs that would impose an unacceptable computational overhead. But these are all limitations that can be eliminated as faster platforms become available at acceptable cost.

The shell software is written in Pascal, making use of the windowing and graphics functions of the various environments for which it is implemented. Thus it varies from implementation to implementation, but this variation is completely invisible both to the programmer and to the user. Knowledge bases can be transferred between implementations, and screen displays and user interaction are identical in different implementations.

An Example

A knowledge base that tells how to put together classical columns provides a particularly clear illustration of the operation of *Topdown,* since the rules for classical columns are well known, and since the traditions of classical architecture assign them a well-defined hierarchical structure (Summerson 1966, Tzonis and Lefaivre 1986, Onians 1988). Essentially, this knowledge base encodes a parallel of the orders.

At the highest level of abstraction a column is represented as a narrow vertical rectangle (figure 6). This is what the user first sees in the poke window. Clicking on it brings up a dialogue box which presents the options of developing it into a Tuscan, Doric, Ionic, Corinthian, or freestyle column. Let us assume, for the moment, that the Ionic order is selected. The poke window now shows a structure of base, shaft, and capital in correct Ionic proportions. Each of these parts can be selected for further development. When this is done, correct Ionic details are presented and can be substituted to complete the design as shown.

The part vocabulary of this knowledge base includes several types of capitals (figure 7), several types of bases (figure 8), and two types of shafts. Furthermore, some of these have alternative details (figure 9). But the substitution rules do not allow combinations that violate the integrity of a chosen order. Only if the freestyle type has been chosen will free combination of parts from different orders be allowed.

As parts are selected, dimensions and proportions are also assigned. First, overall height and diameter, and the relative sizes of base, shaft, and capital are established. Then, at the next level of detail, the internal proportions of the chosen base type, the chosen shaft type, and the chosen capital type can be adjusted. At the next level of detail the proportions of small mouldings and fillets are manipulated. Whenever the user returns to a higher level of abstraction, and alters dimensions at that level, the proportions that have been established for lower-level details are preserved as parts are resized.

The knowledge base incorporates proportioning rules. Whenever a part is first presented in the poke window its dimensions and proportions default to correct values for the particular order that has been selected to control the design. The user can specify variation from these default values (by moving the scroll bar), but only over a sanctioned range. If the freestyle type has been selected, however, proportioning rules do not apply and dimensions can be varied over wide ranges to produce bizarre results.

A comparison of figures 10 and 11 illustrates the effect of the knowledge that is provided by this knowledge base. Figure 10 shows correctly combined and proportioned classical columns that were produced in a few seconds by choosing, at a high level of abstraction, to follow the rules of a particular order. Figure ii shows a few of the many strange mutants that can be produced by selecting the freestyle option which allows recombination of parts from different orders and unfettered variation of dimensions and proportions.

The difference in time required to design a classical column using this knowledge base rather than a traditional computer drafting system (even for an architect thoroughly familiar with the classical orders) is one of seconds compared to hours. This multiple order of magnitude speed-up results from a combination of real-time performance with use of high-level, specialized, context-specific, design operations rather than low-level, general-purpose, drawing-editing operations.

Figure 6 Steps in elaboration of an Ionic column

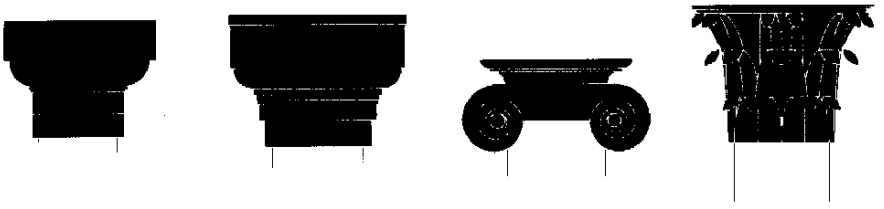

Figure 7 A vocabulary of capitals

Figure 8 A vocabulary of bases

Figure 9 Alternative ways to detail a Doric capital

Figure 10 Correctly configured and proportioned columns:
Tuscan, Doric, Ionic, Corinthian, and Composite

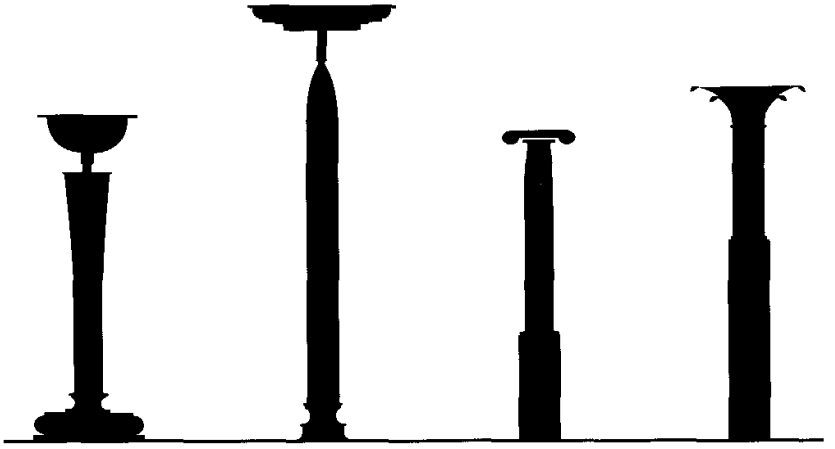

Figure 11 A variety of freestyle columns

Integration of Topdown with Traditional CAD Systems

Traditional CAD systems are content-free, general-purpose tools, whereas a particular *Topdown* knowledge base is a content-rich, special-purpose tool. Thus the two perform complementary roles, and they are best used in combination. They can be integrated effectively by treating *Topdown,* with a library of knowledge bases, as an accessory that can be invoked as needed from within a CAD system. A shape in the CAD drawing or model can then be specified as the starting shape to be developed into a detailed element or subsystem design with Topdown. When the detailed design has been developed it can be exported back to the CAD database and there substituted for the original shape. Subsequently, this detailed design can be modified as required using standard CAD system editing tools. In the current implementations this capability is approximated by providing for export of designs in standard formats such as DXF.

A library of *Topdown* knowledge bases might include tools for designing many different types of architectural subsystems, elements, and details. Our students have, for example, experimented with knowledge bases for producing schematic apartment plans, detailed layouts of kitchens and bathrooms, door and window designs, fenestration of elevations, sectional window details, trees, and human figures. When libraries of knowledge bases become large it is necessary to provide tools for finding the ones that are relevant to particular design contexts. Hypermedia techniques can he useful here. We have, for example, experimented successfully with launching *Topdown* from *Hypercard,* and organizing *Topdown* knowledge base libraries in *Hypercard* stacks.

A *Topdown* knowledge base does not have to provide flawless solutions under all conditions in order for it to be useful: it merely needs to provide a good starting point for further exploration. Nor does it need to be universal: it might provide a highly personal and idiosyncratic set of suggestions about how to develop a particular element or subsystem. The point of a *Topdown* knowledge base library is not to replace the skill and judgement of the designer, but to speed the process of design exploration by providing a wide variety of specialized tools for swift development, in alternative ways, of aspects of a design.

Conclusion

The *Topdown* prototypes have demonstrated the concept of top-down knowledgebased design, and they have shown its potential usefulness. They illustrate one way in which the knowledge captured and encoded in a shape grammar can be delivered effectively to a designer at a CAD workstation.

The prototypes have many limitations. They are restricted to two-dimensional design, they can only maintain real-time performance with designs of very limited complexity, they rely on an overly simple and rigid method of structural description, and they are relatively cumbersome to program. But none of these limitations

seem fundamental: we expect them to be overcome as increasingly powerful hardware platforms become available, as more sophisticated environments for graphic, object-oriented programming emerge, and as current research on the structural description of designs comes to fruition. As the relevant technologies mature, top-down knowledge-based design systems should become important practical tools.

Traditional CAD systems are approaching a productivity plateau that is a consequence of their fundamental character: further speeding of operations and addition of features will make only a marginal difference. Use of top-down knowledgebased techniques seems one promising way to escape the limitations of traditional CAD and to carry design productivity to entirely new levels.

Notes

Durand, Jean-Nicolas-Louis. 1802. Précis des lecons d'architecture. Paris: Ecole Polytechnique.

Durand, Jean-Nicolas-Louis. 1821. Partie graphique des cours d'architecture. Paris: Ecole Polytechnique.

Durer, Albrecht. 115281.1970. Vier Bucher von menschlicher Proportion. (Four Books of Human Proportion). London: C. M. Wagner.

Gombrich, Ernst H. 1969. Art and Illusion: A Study in the Psychology *of* Pictorial Representation. Princeton: Princeton University Press.

Mitchell, William J., Robin S. Liggett, and Thomas Kvan. 1987. The Art *of* Computer Graphics Programming. New York: Van Nostrand Reinhold.

Nagakura, Takehiko. 1989, "Shape Recognition and Transformation-A Script-Based Approach." This volume.

Onians,John. 1988. Bearers *of* Meaning: The Classical Orders in Antiquity, the Middle Ages, and the

Renaissance. Princeton: Princeton University *Press.*

Stiny,George. 1980. "Introduction to Shape and Shape Grammars." Environment and Planning B 7: 343-51.

Stiny, George. 1982. "Spatial Relations and Grammars." Environment and Planning B 9:113-114.

Summerson, John. 1966. The Classical Language *of* Architecture, Cambridge, Mass.: MIT Press.

Tan, Milton. 1989. "Saying What it Is by What it Is Like." This volume.

Tzonis, Alexander, and Liane Lefaivre. 1986. Classical Architecture: The Poetics *of* Order. Cambridge, Mass.: MIT Press.