Chapter 13

Representing the structure of design problems

Brian S. Logan

13.1 Introduction

In recent years several experimental CAD systems have emerged which focus specifically on the structure of design problems rather than on solution generation or appraisal (Sussman and Steele, 1980; McCallum, 1982). However, the development of these systems has been hampered by the lack of an adequate theoretical basis. There is little or no argument as to what the statements comprising these models actually mean, or on the types of operations that should be provided. This chapter describes an attempt to develop a semantically adequate basis for a model of the structure of design problems and presents a representation of this model in formal logic.

In general, conventional CAD systems adopt a broadly Cartesian view of design; problems are broken down into fragments, each of which is solved separately before being synthesized into an overall solution (Asimow, 1962; Alexander, 1964). Subproblems are implicitly assumed to be largely independent of each other, and the various criteria are effectively considered in isolation. Models have been developed for those subproblems for which formulations are readily available. However, their limited range and fixed level of abstraction makes it difficult to relate them to models of other criteria, and results in a limited problem definition which must be augmented by the designer’s knowledge and experience.

Several authors have argued that these assumptions are invalid (Broadbent 1973; Lawson, 1980). It is a characteristic of design problems that their definition in terms of criteria is often incomplete. It is rare for a design problem to be comprehensively stated in such a manner as to allow the logical derivation of a solution, and design problems are not decomposable in the sense defined above. Design problems often contain many illdefined and conflicting objectives. Very rarely does any part of a designed thing serve only one purpose, and it is frequently necessary to devise an integrated solution to a whole cluster of requirements. Design decisions may have results other than those intended and highlight previously unrecognized criteria and relationships. In many cases the stated objectives are in direct conflict with one another. Rarely can the designer simply optimize one requirement without suffering some losses elsewhere. Different trade-offs between the criteria will result in a whole range of acceptable solutions, each likely to prove more or less satisfactory in different ways and to different clients and users. It is the very interrelatedness of all these factors which is the essence of design problems rather than the isolated factors themselves, and it is the structuring of these relation-
ships which forms the basis of the design activity (Simon, 1969). This set of relationships forms what we shall call the 'structure' of the design problem.

This structure is implicit in the problem model describing the current solution. Design problems have no inherent structure; the problem acquires a structure as solutions are proposed and problems are reduced to subproblems. The developing solution forms a representation of the relationships defining the structure of the problems as modelled by the relations linking solutions and criteria. In a very real sense the relationships between criteria can be seen as a function of the approach to design embodied in the proposed solution rather than as inherent in the problem itself. The evolving problem model reflects the designer's conceptualization of the functional dependencies within the problem, and it is these dependencies which form the basis of relationships between criteria.

Relationships between criteria thus do not exist in any absolute sense but rather depend on the functional decomposition strategy adopted during design, and are often influenced by the degree of flexibility required of the solution. For example, in a load-bearing structure the walls both organize space and carry structural loads, whereas in a framed structure these (sub)systems are functionally and conceptually distinct; the position of the walls is not dictated by structural requirements nor the structure by the spatial organization. The difference is in the way the systems are conceptualized rather than the underlying physical relationships.

In general it is likely that there will be more than one decomposition of an element, and a designer will employ many such representations, each embodying a different view of the relationships between criteria (Freeman and Newell, 1971). None of these representations can be regarded as any more 'real' than any other; rather, they are simply more or less useful for some purpose. Typically there will be many internally consistent but mutually inconsistent 'variant' models - distinct conceptualizations of the relationships within the problem - including as a special case the alternative solutions to the problem. To simplify the discussion we limit consideration to the representation of a single variant, i.e. a single distinct conceptualization of the relationships within the problem.

### 13.2 Models of problem structure

A design problem can be considered formally as a series of relationships between 'constraints', where the term constraint is interpreted broadly to include all the elements which enter into the problem definition (Sussman and Steele, 1980). As used here, the term refers to any arbitrary statement or proposition describing the problem at some level of abstraction. This set of propositions comprises the 'problem definition' and can be loosely interpreted as a set of goals or criteria which have to be achieved to ensure the satisfactory performance of the designed object.

The problem definition consists of some non-empty set of (possibly abstract) objects or problem elements which are in some sense considered to be elemental, and a series of statements about these objects. The statements describing an object are commonly termed 'properties' or
'attributes', while those expressing a relationship between two or more objects are termed 'relations' or 'relational expression'. Object are defined by their properties and their relationship to other objects. Two objects are equivalent if their descriptions are the same (i.e. all the properties true of one are true of the other). A set of propositions is said to be inconsistent if it contains more than one distinct description of some object; otherwise it is said to be consistent.

The relationships between constraints express logical dependencies between the elements comprising the problem definition, in effect defining the 'higher-level' constraints in terms of the 'lower-level' constraints on which they depend. A constraint defined by such a relation is in a sense contingent on its constituent parts in that its satisfaction is dependent on the attainment of the lower-level constraints which form its definition. This relation is one of entailment rather than logical equivalence; in general, there will be many ways of satisfying a given constraint, and the existence of a relation does not necessarily imply any lower-level concept.

If we view statements themselves as objects, we can extend this idea to statements about statements, statements about these statements and so on to any level of abstraction. The resulting structure comprises a series of statements which can be thought of as occurring at some 'level'. A partial ordering over these statements can be defined in the sense of Russell (1956). Objects are said to be of order 0, statements about objects are said to be of order 1, statements about these statements are of order 2, etc.' The set of statements which mention an object can be said to form a 'description' of that object. An object at any level is defined in terms of the assertions made about it at higher levels. An object of order \( n \) is defined in terms of its description at \( n+1 \). The structure of the model is thus entirely extensional. There is no 'correct' set of statements which completely or absolutely define any object or relation. Concepts have no a priori meanings and are defined only by the relationships in which they take part. Any statement about an object is as much a part of its description as any other. There is no simple sense in which any one statement or group of statements contains a definition of an object or a complete description in terms of its structure. Rather any arbitrary collection of statements can be seen as a description of a 'problem state', i.e. a representation of a specific solution at some level of detail. More precisely, we define the problem state to be the current problem model, where the problem model is seen simply as a collection of statements or declarative sentences about the problem that the designer has chosen to state.

The set of constraints and relationships between constraints which define the model can be viewed as an abstract structure (or theory) describing the interrelationships between the elements of the design problem. This abstract structure defines an 'epistemological' representation of a single problem state and forms a basis for organizing the various combinations of relationships which together define the structure of the problem. It is not suggested that this structure is the necessary or inherent structure of design information. Rather the theory is an heuristic to be judged by its sufficiency in representing different kinds of relations and the clarity and power of its conceptual organization.
13.3 Relationships in design

It is largely through the existence of these functional dependencies that any cognitive problem-solving activity can take place. They are, and must be, used by the problem solver to structure the problem in terms in which he can solve it, and act as a kind of plan for finding a route through problem material that would otherwise appear undifferentiated and amorphous (Hillier et al., 1972).

In a broad sense all of these structures can be viewed as 'production rules' or 'problem transformations', a concept well developed in artificial intelligence and cognitive psychology. A rule in the context of design is any problem transformation linking the solution and criteria spaces; that is, some relation which reduces the size of the solution space by mapping a problem expressed in terms of abstract requirements into some solution or class of solutions which satisfies these requirements (Hillier et al., 1972; Akin, 1978). Design can be seen as a sequence of actions which advance the problem state from one state to the next. In effect the boundary between the criteria and solution spaces moves to include as criteria solutions to previous problems, as the problems represented by the criteria are reduced to a series of simpler subproblems whose solution is known, or which are at least easier to solve.

The use of these relations as production rules is not logically determinate and often involves a search for a solution. In general, the change of state which results from the generation of an hypothesis cannot be guaranteed to be consistent with the current problem model. As the design develops it provides an increasingly detailed context against which to test the hypotheses, and the evaluation of a proposal can result in the discovery of previously unrecognized relationships and criteria. Solutions to particular subproblems are apt to be disturbed or undone at a later stage, when new aspects are attended to and the considerations leading to the original solution are forgotten or not noticed. Such unwanted side effects accompany all complex design processes. There is no meaningful distinction between analysis and synthesis in this process; problems and solutions are seen as emerging together rather than one logically following on the other. The problem is explored through a series of attempts to create solutions and understand their implications in terms of other criteria. Designers analyse their problems through the generation of solutions rather than setting out specifically to study the problem (see, for example Eastman, 1970; Lawson, 1979; Darke, 1979).

The fundamental objective thus becomes one of understanding the problem structure, with a major part of the effort in design directed as structuring problems and only a fraction of it solving problems once they are structured (Simon, 1969). Design can be viewed more generally as a process for gathering information about problem structure that will ultimately be valuable in discovering a solution. McCallum (1982) argues that an understanding of these relationships is fundamental to a design concept and necessary to the creation of new designs. He states 'one of the most challenging problems is to be aware of and understand the influence of one particular characteristic of the system on all the others ... If these
influences are well understood it becomes easier to improve designs.

However, if models of design problems are viewed as simple 'black-box' algorithms transforming an input into an output, no real understanding of the complex relationships they represent can be achieved. Architects, for example, do reasonably well in what might be termed 'traditional' sub-problems, such as the organization of space, where the relationships, although informal, are fairly well developed. For example, in developing a design for a house an architect may evaluate its planning performance on the basis of the relationship between the kitchen and dining areas or between the kitchen and bathroom and their common services. In either case the 'cause' of a failure to achieve satisfactory performance is clearly evident; the relationships are inherent in the evaluation in the form of 'codes' linking solutions and performance. However, when the causal relationships are not explicit in the evaluation, or are not one to one, or are expressed in terms of non-designer-manipulable concepts (as, for example, in complex quantitative systems such as plant/fabric interaction or systems of legislative constraints), the structure of the problem may never be adequately understood. The design fails on some 'new' criterion, with no clear understanding of the reasons for failure. Hillier (1972) has argued that making more information available to the designer without a corresponding understanding of the relationships or 'codes' which make it intelligible, far from helping the designer to escape his preconceptions, serves merely to make design more difficult, forcing a retreat either into the most basic form of problem restructuring - the adaption of previous solutions - or a blind search of the solution space.

It is convenient to formalize these design operations in considering their effect on the problem model. The framework we adopt is essentially that of March (1976), and is based on the concept of syllogistic inference. Logical inference leads from premisses - statements assumed or believed, for whatever reason - to conclusions which can be shown on purely logical grounds to be true if the premisses are true. The piece of reasoning represented by the argument is termed a 'syllogism', and represents a relation which holds between three propositions. To define a syllogism precisely we introduce three terms P1, P2, and P3, which we shall call the minor term, the middle term and the major term, respectively. Then a syllogism is a sequence of three propositions such that the conclusion contains the minor and major terms and the second premiss contains the major and middle terms and the second premiss contains the minor and middle terms. The first premiss, since it contains the major term, is the major premiss, and the second premiss, since it contains the minor term, is the minor premiss. The middle term appears in both premisses but not the conclusion.

A major premiss, or 'rule', as its name implies must be of the utmost generality and must hold without exception. The minor premiss, or 'case', is a more particular proposition under the rule. A case often refers to an actual thing or event but not necessarily so. It may be a general proposition, only less general than the rule and therefore particular in relation to the rule that subsumes it. The 'result' is the logical conclusion of the argument. In order that there should be a syllogistic conclusion there must be two premisses containing a middle term distributed in one but not in the
other. The conclusion is drawn by compounding the two premisses in such a way that the middle term may be deleted. By permuting the possible binary operations on these three terms Peirce (1931-1958) defines three modes of reasoning: deduction, abduction and induction.

Deduction, or analytic reasoning, is concerned with the production of effects, and can be considered the derivation of a result from a rule and a case: for example, from A and A \(\rightarrow\) B infer B. Somewhat unconventionally, Peirce identifies two forms of synthetic reasoning. Abduction or abductive inference can be considered the derivation of a case from a rule and a result - the generation of a logically consistent hypothesis, which, if it were true, would explain an observation: for example, from B and A \(\rightarrow\) B infer A. Induction can be considered the derivation of a rule from a case and a result - the formation of rules from instances or examples which may be specific to the current problem: for example, from A and B infer A \(\rightarrow\) B.

It can be shown that this system is complete. Within the context of the model presented above the three forms of inference proposed by Peirce are capable of generating all the consistent extensions of the current problem state. Together with the model they form a coherent framework for a descriptive model of the design process as expressed in the rules, cases and results comprising the current problem model. The three modes of inference are not directly represented; rather the effects of their application can be inferred from the statements comprising the model. Each of these three forms of reasoning can be mapped onto a corresponding logical inference system. However, we will concentrate on the deductive model in the remainder of this chapter.

13.4 The semantics of problem structure

Together the model and its operators form the framework for a 'theory of representation'. They provide an extensional definition of the 'meaning' of the statements comprising a model. The model appears as a (necessarily finite) set of elements subdivided into domains on which relations are defined. We can formalize the meaning of these statements by interpreting them as the axioms of an axiomatic theory. 4 To simplify the discussion, we limit consideration to a single variant model of order 1, i.e. a model defined in terms of variables of order 1 and the relations defined over these variables. We shall argue that even this severely limited model is capable of expressing a significant class of relationships.

The term 'axiomatic theory' (as used here) refers to two sets of statements, one of which is a distinguished subset of the other (Stoll, 1961; Schoenfield, 1967). The entire set of statements define the subject matter of the theory. The members of the distinguished subset are called 'provable statements' or 'theorems', and are defined to be those statements of the theory which are deducible by logic alone from certain initially chosen statements called 'axioms'. The basic concepts of an axiomatic theory are called its 'primitive terms' and are taken as undefined. Collectively, the primitive terms serve as a basis for imposing a certain structure on the
subject matter of the theory (which is the object of study of the theory). The
structure itself is given in the non-logical (or proper) axioms which are the
assumptions made about the theory being formalized and its basic concepts
(including, possibly, the existence of interrelations among them).

A description of a problem state can be regarded as a series of elements
linked together by relations and functions which express the logical depen-
dencies implicit in their extensional definition. By defining the statements
comprising the problem model to be the proper axioms of an axiomatic
theory, the consequences of these statements become derivable from the
axioms as theorems. More precisely an axiomatic theory, T, is defined by
specifying (Nicholas and Gallaire, 1978):

(1) Its set of constraints (respectively, predicate names) is the set of
individuals (respectively, relations) comprising the model.
(2) Its set of proper axioms is the set of statements which comprise the
problem model (i.e. the assertions and relations), such that if a
statement is known to be true then the statement is an axiom, and if it is
known to be false then the negation of the statement is an axiom.

The basic concepts, axioms and theorems of an axiomatic theory together
comprise an 'axiom system'.

It is convenient to restrict our attention to theories expressed in clausal
form. These have an especially simple syntax but retain all the expressive
power of the full predicate logic. We present an informal definition of the
syntax of clausal form and indicate its correspondence with English
(Kowalski, 1979).

A clause is an expression of the form

\[ A_1, \ldots ,A_n \leq B_1, \ldots ,B_m \]

where \( A_1, \ldots ,A_n,B_1, \ldots ,B_m \) are atomic formulas, \( n \geq 0, m \geq 0 \). The
atomic formulas \( B_1, \ldots ,B_m \) are termed the 'joint conditions' of a clause
and \( A_1, \ldots ,A_n \) are the alternative conclusions. If the clause contains
the variables \( x_1, \ldots ,x_k \) then we interpret it as stating that for all \( x_1, \ldots ,x_k \)

\( A_1 \) or \( \ldots \) or \( A_n \) if \( B_1 \) and \( \ldots \) and \( B_m \)

If \( m = 0 \) we interpret it as stating unconditionally that for all \( x_1, \ldots ,x_k \)

\( A_1 \) or \( \ldots \) or \( A_n \)

If \( n = 0 \) we interpret it as stating unconditionally that for all \( x_1, \ldots ,x_k \) it is
not the case that

\( B_1 \) and \( \ldots \) and \( B_m \)

If \( m = n = 0 \) we write it as \( \emptyset \) (the empty clause) and interpret it as a sentence
which is always false. The set of clauses \( C_1, \ldots ,C_n \) is interpreted as the
conjunction \( C_1 \) and \( C_n \).

An atom (or atomic formula) is an expression of the form

\[ P(t_1, \ldots ,t_m) \]

where \( P \) is an \( m \)-place predicate symbol, \( t_1, \ldots ,t_m \) are terms and \( m \geq 1 \).
We interpret the atom as asserting that the relation called \( P \) holds among
the individuals $t_1, \ldots, t_m$. A term is a variable, a constant symbol or an expression of the form

$$f(t_1, \ldots, t_m)$$

where $f$ is an $m$-place function symbol, $t_1, \ldots, t_m$ are terms and $m \geq 1$. The sets of predicate symbols, function symbols, constant symbols and variables are any mutually disjoint sets.

The basic elements of the model are the primitive statements or concepts. Concepts are represented by the atomic formulas $P(x)$, $Q(f(x))$, etc. For example

- $\text{Red (block 1)}$ : 'block1' is red
- $\text{At (block1, loc1)}$ : 'block1' is at location 'loc1'

Atomic formulas denote the attributes of some 'object' (not necessarily having physical existence) such as size, colour, cost, physical properties, etc. Objects are defined entirely in terms of their attributes and have no other existence within the model. The resulting representation could be termed attribute oriented in allowing the description of an object in terms of some arbitrary set of its properties.

Concepts are 'defined' recursively in terms of other (simpler) concepts. The clause $P(x) \leftarrow U(x)$, $V(x)$ can be interpreted as $P$ is defined if $U$ and $V$ are defined, or if $U$ and $V$ are provable, $P$ is provable. For example,

$$\text{Volume (x,f(u,v,w))} \leftarrow \text{Length (x,u)}, \text{Breadth (x,v)}, \text{Height (x,w)}$$

states that $x$ has volume $f(u,v,w)$ if $x$ has a length $u$, a breadth $v$ and a height $w$. Several alternative definitions are possible for each concept; for example,

- $\text{Block (x) } \leftarrow \text{Cube (x)}$
- $\text{Block (x) } \leftarrow \text{Pyramid (x)}$

are required to express the disjunctive definition $\text{Block (x) } \leftarrow \text{Cube (x) v Pyramid (x)}$. In general, we limit our definitions to clauses, containing at most one conclusion. Such clauses are called 'Horn clauses', and it can be shown that any problem expressible in a standard form can be re-expressed in terms of Horn clauses. A model of this form is deterministic in the sense that no set of conditions ever implies a disjunctive set of conclusions. In addition, in accordance with our definition of a variant model we require that a theory is acyclic, i.e. that it contains no circular definitions.

A clause is interpreted as making an assertion about the state of the world. An expression cannot usually be said to definitely correspond to anything in the actual world and its meaning is fixed only with respect to a possible world. An expression means what it claims about a possible world. Two expressions which are satisfied by the same possible world have the same meaning. If we assume that it is possible to state some criteria by which we can judge whether a suggested possible world satisfies an expression, or whether, on the contrary, it is a counter-example to the claim made by the expression, then these criteria can be used as an account of meaning. The expression 'block1', for example, is intended to denote a particular block in the 'real' world. In order to achieve this identification we would have to assert enough axioms containing the expression 'block1'
to ensure that in any possible world satisfying them, the denotation of 'block1' corresponded to the particular block in question in the actual world. These axioms will contain other names and relation symbols, and we cannot in general say conclusively that any of these is defined in terms of some particular subset of the others. The entire web of logically connected assertions is presumably tied down to the actual world by some of them having an interpretation as observations.

The problem with this approach to meaning is, of course, to specify what we mean by a possible world in such a way that we can state the meaning criteria or 'truth conditions'. First-order logic makes only very elementary assumptions. A logically possible world is a set of individuals (constants) and a set of relations between then (predicates). However, we shall argue that this relatively simple view of the world as a set of (possibly abstract) objects and the relationships between them is adequate to represent many of the relationships of interest in design.

Any meaning that might be associated with a logical formula is thus relative to the collection of sentences which express all the relevant assumptions. If T1-3 below express all the relevant assumptions about a world

T1  Red(block1)
T2  Cube(x) <- Red(x)
T3  Pyramid(x) <- Blue(x)

then there is nothing to rule out the interpretation in which the assertion

T*  Pyramid(block1)

holds. Such a possibility is consistent with the stated assumptions T1-3, which alone determine any meaning that might be associated with the symbols 'Red', 'block1', 'Pyramid', etc. To rule out the possibility, T* requires some additional assumption such as

T4  - (Cube(x) Pyramid(x))

T* is consistent with Ti-3 but inconsistent with T1-4.

Given a set of formulas which express all the assumptions concerning a problem domain, to understand an individual statement or formula it is necessary to determine what is logically implied by the assumptions. The meaning of a predicate symbol, such as 'Pyramid', can be identified with the collection of all sentences containing the predicate symbol and implied by the assumptions. This includes the denial

Pyramid(block1) [and hence - Blue(block1)]

but the meaning of T1-3 does not.

The statements comprising the model literally mean nothing unless their meaning is specified by axioms. The model-theoretic account of meaning makes this absolutely precise; as one cojoins assertions so the set of interpretations for the set of symbols occurring in them is reduced and the set of possible inferences from them enlarged. Their meaning is progressively tightened as more facts involving them become inferable. In a sense, logical meaning justifies inferences. The meaning of an assertion is the theorems that can be derived from it.
It follows that it is unnecessary to talk of semantics at all. All talk of semantics can be re-expressed in terms of logical implication. To define the semantics of the clausal form of logic, therefore, it suffices to define the notion of logical implication. For any logical consequence $B$ of a set of clauses $A_1, \ldots, A_n$, it is possible to construct a proof of inference steps based on purely syntactic criteria (Schoenfield, 1967). A proof that some statement $B$ is a logical consequence of a given set of clauses $A_1, \ldots, A_n$, is a demonstration that $B$ logically follows from $A_1, \ldots, A_n$. More precisely, a proof is a finite sequence $S_1, S_2, \ldots, S_k$ of statements of the theory such that each $S_i$ is either an axiom or comes from one or more of the preceding $S_j, \ldots S_i$ by a logical rule of inference. A theorem is a statement which is the last statement of some proof, i.e. $S_k = B$. To demonstrate that any given formula is a theorem we need only show that it has a proof. The truth of any statement can therefore be reduced to a concept of proof based on the superficial syntax of the formulas and independent of their meaning.

Theorem proving in the model is equivalent to evaluation in a procedural system:

1. The proof of $P(x)$ can be viewed as the derivation of the current value of $x$ from the current set of assertions and rules comprising the model. The maximal set of dependencies of $P(x)$ can be obtained from the proof tree.
2. Repeated application of bottom-up inference to the unit clauses will generated the maximal set of consequences for each assertion giving some indication of the likely effects of modifying the associated design variable.
3. If the base assertions are unsubstantiated the proof generates a composite function expressing the dependency of the goal on the assertions allowing quantification of dependencies to be carried out symbolically rather than numerically. The partial derivative of the composite function (assuming it is differentiable) predicts approximately the change in the goal value for a unit change in the value of the assertion.

We can extend this concept to view abductive and inductive inference as hypothesis generation and rule formation, respectively. In general, these operations are non-deterministic and it is necessary to abandon the concept of 'truth' embodied in classical logics in favour of the weaker criterion of consistency formalized in non-monotonic logics (McDermott and Doyle, 1980; Reiter, 1980). However, these three forms of logical inference can be shown to be sound relative to the power of the theorem prover, and together they are capable of generating all the (logically) consistent extensions of the problem model (Melzer, 1970; Winograd, 1980).

The order in which these operations are performed can be viewed as a model of some higher-order organization or strategy for the design process itself. Such meta-codes or meta-rules are responsible for the overall control of the design process; they control which problem to consider and when, which criteria are the most important and when (and if) to perform each of the basic object-level operations. In a sense they form a system which takes the first system as its object. In the next section we discuss how these processes can themselves be represented within the framework discussed above.
13.5 Meta-level systems

It can be shown that this heuristic and control information can be viewed as a particular case of the higher-level descriptions discussed above. In general, these concepts cannot be represented within the framework of a first-order language, and generalized concepts defined using predicate variables such as 'all solutions to this problem' or 'all walls of this building' can more naturally be represented as an axiomatization of a meta-level.

The prefix 'meta' denotes a language whose subject matter is the representation of some theory as distinct from the theory itself. The statements of the meta-language take those of the object-language as their objects and allow us to make assertions about these statements. Formalizing the meta-level of an object system greatly extends its power in allowing us to reason explicitly about things which cannot be represented explicitly at the object level. Control of inference can itself be formalized as a higher-level object to provide an explicit representation of control strategies and heuristics such as alternative proof procedures, loop detection and tail-recursion optimization. Modifying the interpreter becomes simply a matter of changing its definition.

Rather than implementing a meta-level directly, it would be better to implement the means of implementing a meta-level. One of the more elegant ways to do this which avoids extending the system to cope directly with second- or higher-order logics is to provide the system with a representation of itself; in effect a self-referential or self-reflexive system (see for example Weyhrauch, 1980). The combination of object language and meta-language is a special case of a more general construction. Given any two languages (i.e. systems of logic with their associated proof procedures) it may be possible to simulate the proof procedure of one language L1 within the other L2. The simulation is accomplished by defining in L2 the binary relationship Pr which holds when a conclusion can be derived from assumptions in L1. Provided the definition of Pr correctly represents the provability relation of L1, simulation by means of Pr in L2 is equivalent to direct execution of the proof procedure of L1. In general, a simple unsophisticated problem solver can improve itself by using simulation to behave like a more sophisticated one (Kowalski, 1979). In the case in which the object language and meta-language are identical, the single language augmented by the definition Pr of its own provability relation is an amalgamation of an object language with its meta-language.

This level of expression is required in the representation of arbitrarily complex descriptions and relationship, such as those involving simultaneous interdependency, which generally involve recursion through metalevels or reasoning across variant models.

13.6 Conclusions

The only knowledge that is explicitly represented in conventional models of design problems is a description of the 'solution' in terms of the current state of primitive elements or concepts (Eastman, 1978). Abstract objects, and in particular the relations between them, are dealt with only in the
designer's head or on paper used temporarily during the early stages of the design process. Knowledge concerning links to other concepts remains implicit in their definition, and they rely on the designer's knowledge and experience of the systems being modelled. The model developed above focuses primarily on the relations between the information describing the design proposal, and allows both the explicit representation of both abstract objects and their functional dependencies.

It is argued that these models provide a useful conceptual framework both for organizing design knowledge and describing the design activity. They reflect the designer's conceptualization of the problem as expressed in the functional dependencies between concepts. Like drawings, they allow the development of unique models of problem structure able to support a wide variety of interaction, and provide a basis for building essentially personal or idiosyncratic models of arbitrary design concepts. The formalization of such a first-order model as the basis of a series of meta-levels could form a theoretical framework for the investigation of a wide range of design issues (for example, abstraction and generalisation, induction and rule formation, hypothesis formation and reasoning by analogy) and provide a tool for further design research.

Notes

1. This ordering of relations can be formalized within the theory of classes (see, for example, Quine, 1963; Sappes, 1957).
2. See, for example, Newell, et al.'s (1963) table of connections and Minsky' s (1963) 'heuristic connections'.
3. Simon (1973) argues that this methodology of 'analysis though synthesis' is inherent in the limitations of the cognitive processes underlying the design activity.
4. We do not attempt to present a detailed introduction to first-order logic here. See, for example, Church (1956).

References


MCDERMOTT, D. and DOYLE, J. (1980). Non-Monotonic Logic I. Artificial Intelligence 13(1, 2), 41-72


