Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2006), *20*, 63–82. Printed in the USA.

DOI: 10.1017/S0890060406060057

KEYNOTE ARTICLE

Whither design space?

ROBERT F. WOODBURY[1] AND ANDREW L. BURROW[2]

[1]School of Interactive Arts and Technology, Simon Fraser University, Vancouver, British Columbia, Canada
[2]Spatial Information Architecture Laboratory, RMIT University, Melbourne, Australia

(RECEIVED March 16, 2005; ACCEPTED December 13, 2005)

Abstract

Design space exploration is a long-standing focus in computational design research. Its three main threads are accounts of designer action, development of strategies for amplification of designer action in exploration, and discovery of computational structures to support exploration. Chief among such structures is the design space, which is the network structure of related designs that are visited in an exploration process. There is relatively little research on design spaces to date. This paper sketches a partial account of the structure of both design spaces and research to develop them. It focuses largely on the implications of designers acting as explorers.

Keywords: Design Space Exploration; Knowledge Representation; Search; State Space; Typed Feature Structures

1. INTRODUCTION

Design space exploration is the idea that computers can usefully depict design as the act of exploring alternatives. This involves representing many *designs*, arraying these represented designs in a network structure termed the *design space*, and exploring this space by traversing *paths* in the network to visit both previously represented designs and to find sites for new insertions into the network. This conceptual complex arose by elaboration of initial ideas in design computing, and as a consequence of researching descriptions of design, engineering prototype systems, performing thought experiments, exploring suitable mathematical abstractions, and engaging in considerable scholarly debate. Today, it rests on three premises, each of which is a distinct area of research.

The first premise is that exploration is a *compelling* model for designer action, namely, that it accords with observed designer actions and suggests approaches to the problem of supporting designers. Numerous studies have shown the utility of modeling designers as information processing systems that search to satisfy changing goals in a strongly constrained problem space. Furthermore, humans are endowed with specific and limited cognitive structures that constrain their behavior as searchers and representors of problem spaces. Thus, it is possible to argue that exploration is a compelling model for designer action by demonstrating a correspondence with the actions of designers and by suggesting means to work around human cognitive limits.

The second premise is that exploration is an *effective* basis for computer support, namely, that designers benefit from tools that amplify their abilities to represent goals and problems spaces, and to search for designs. Existing design space explorers demonstrate the case. Researchers have created a modest number of computer programs that attend to such amplification tasks, and one of these, KIRTS/GENESIS (Heisserman et al., 2000), has been used in industrial application. However, there are difficulties in demonstrating the effectiveness of a cognitive model by building tools. In particular, sociological factors determine that tools evolve and cannot be considered independently of the context in which they are used. Unable to simply build and test design space explorers, it is possible to argue the effectiveness of exploration by demonstrating a lineage from existing activities and by producing evidence of opportunities for support.

The third premise is that computational support for exploration is *feasible*, namely, that there are tractable representations and algorithms that provide suitable amplification for design explorers. A large body of literature provides

Reprint requests to: Robert F. Woodbury, School of Interactive Arts and Technology, Simon Fraser University, Surrey Campus, 14th Floor, Central City Tower, 13450 102nd Avenue, Surrey, BC V3T 5X3, Canada. E-mail: rw@sfu.ca

support for this premise. Numerous formalisms exist for representing designs, for acting on designs to produce other designs, and for recording such action. A large secondary literature uses such mechanisms to produce relatively compact generative descriptions of corpora of existing and conjectured design work.

Research in design space exploration takes on several distinct patterns. Typically, it addresses representation, search algorithms, task description, or interaction design. Relatively little work focuses on the design space itself. Instead, most research focuses on design states and on making action explicit. Little work exists that depends upon computational access to the design space (Chien & Flemming, 1997; Chien, 1998; Woodbury et al., 1999, 2000*a*; Woodbury & Burrow, 2001). Yet, it would appear that the design space itself is where the largest gains are to be made. Designers typically consider a very small number of alternatives in their work, and this is explained by cognitive limits.

Why is there a paucity of research on the design space? Conjecture is easy, and the debate is open. Clearly, prior work on representation, including generative mechanisms, is a prerequisite for serious thought about design spaces. It may be that the current state of our knowledge of representation and generation is inadequate to the task. Furthermore, any useful computational account of a design space requires both mathematical sophistication and deep knowledge of the design domain. There may be easier pickings elsewhere, or it may be that design space exploration has limited utility. There is probably truth in all of these conjectures.

In recent years, a group of researchers, initially at Adelaide University, but now distributed across Australia, Canada, and Taiwan, has developed a representation for design spaces that is computable, and thus formal, and appears to have sufficient informal expressiveness to capture complex aspects of designs. The representation builds on a formalism from computational linguistics, Carpenter's (1992) *typed feature structures*, and extends it to be an account of the design space. To researchers in conventional design grammars, the formalism appears strange as it substitutes types for rules, a unification-based resolution procedure for generating designs, and a set of composable navigation operators for complex moves in design space. It does posit an account of the indefinite parts that are declared essential in the shape grammar literature (Chang, 1999). It is a highly regular system as a consequence of aiming at disciplined generation, navigation, and reuse. It has certain striking limits that are a consequence of its attainment of efficient algorithms for key computations.

This paper explores some of the implications of having such a design space representation. It does this in reference to each of the three premise areas. In particular, it considers the correspondence between designer action and exploration, the opportunities for amplification in design activities, and the feasibility of symbolic representation and computation of design spaces. Its main focus is on the nature of designer action, although it also presents brief sketches of the arguments for amplification and computation.

2. DESIGNER ACTION AND THE DESIGN SPACE

The metaphor of designers as finite explorers entails both their abilities and the structure of their exploration. In this section, we take a brief look at each, focusing on the two key issues of design representation and the situation of the designer in the design space. We find that good design representations are invariably both partial and intentional; and that accessibility is a key to addressing the incomprehensibly large space of possibilities that comprise all realistic design spaces.

The empirical work on designer action provides both a starting point and inspiration to design space exploration. Akın (2001), a canonical senior scholar in the field, summarizes his past work by describing invariants of designer behavior and, in comparison with the larger class of designers, variants seen in architectural designers. His constructs are those of exploration, as seen in the finding that expert designers display a typical strategy of *breadth first, depth next* in solving problems. This is in comparison to novices, who typically display less breadth of exploration. What remains unclear is to what extent such strategies are conditioned by the external memory aids available to designers.

In interpreting this empirical work, the question is whether the results generalize across media. Akın's studies (and others) were conducted using conventional external media such as sketch paper and pencil. Such media do not, of themselves, support the creation, recall, and reuse of alternative designs; computer support has the potential to provide rapid access to both a breadth of alternatives and depth of exploration. In the face of these quantitative differences in speed of recall and reuse, does the earlier framework remain applicable? We are aware of one piece of evidence supporting this conjecture. The KIRTS/GENESIS system (Heisserman et al., 2000) is, *inter alia*, a design completion system. With it, a designer of an airplane hydraulic system indicates an approximate physical path for a hydraulic line, and the system completes the design to code conformance and to a representation suitable for automatic fabrication by a CNC machine. The effect is to shift the relative cost of depth versus breadth of exploration. In terms of designer effort, depth becomes cheaper than breadth. The resulting style of work might be described as *depth first, breadth next*. In this example, the framework remains applicable to the description of a procedure that has been transformed by quantitative differences in the ability to evaluate alternatives.

Most generative system work has been built using the empirical work as a sort of rough design guide. The preceding example demonstrates the impact of media on the strategies of the designer. To our knowledge, there are no detailed analyses of designer action in the face of more capable media. Instead, results emerge from the practice of design-

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

ing and building design space explorers. From these experiences, we know of several traps for the unwary.

2.1. Representation and meaning

A principal danger in research that is lead by software development is programmatic thinking. By *programmatic*, we mean the overly literal attention to representing and computing directly over the concrete entities in the problem at hand. To be fair to the field, almost all serious expert systems for design were built on two or three levels: a lower abstract inference engine, sometimes a midlevel representation of tasks and methods, and an upper representation of problem particulars. For example, R1 (McDermott, 1982) was built on top of OPS (Forgy, 1981), LOOS (Flemming, 1990) originally in OPS, and GENESIS (Heisserman, 1994) on a hybrid logic/solid-modeling substrate. Newell's *knowledge level* (1981) is perhaps the definitive argument for midlevel representation. Others were working in parallel; for example, Chandrasekaran and Johnson (1993) coined the term *generic task* to identify a representation level in which task types intervened between the domain and lower levels of implementation. Special attention was given to design tasks (Chandrasekaran, 1990). Their formulation grew out of work such as Brown's thesis (1984) and the subsequent book (Brown & Chandrasekaran, 1989). The first trap opens when the specific domain representation is taken too seriously. To put criticism close to home, the early thinking of researchers in the SEED-Config project (Akın et al., 1997) confused the knowledge level, involving such things as functional units, design units, and technologies, with underlying computations that could support it. The result of falling into this trap is to miss two crucial points: good representations are, in a deep sense, both *intentional* and *partial*, and good accounts of designing are given in terms of the most general constructs possible.

Consider how the term intentional, drawn from philosophy of mind, relates to designs. If we are committed to realizing designs as physical artifacts, then ultimately we are committed to their intentionality. That is, we are committed to the notion that designs are *about* other objects. A representation is intentional because it is a statement about another object. It need not be a complete statement, nor need it be the only statement. The latter is crucial: you can have multiple representations that are indistinguishable in terms of the information they convey without implying that their referents are the same or are distinct.

This level of indirection, the "aboutness" of designs, becomes a second trap when we conflate the properties of a design with the properties of its subject. This trap leads to overly literal interpretations of representation. This conflation has facets.

The first facet is that a designed object embodies phenomena beyond the reach of any particular representation. Thus, we do not find air flow around buildings accounted for in a typical building model, nor do we find explicit reference to nonstructural cladding in a structural design system, and especially seldom do we find represented the ineffable sense of space and light that is the hallmark of much fine architecture. Therefore, those aspects of its referent that it captures necessarily limit what we can do with a representation. Yet, as designers, we must constantly make decisions in just such explicitly unrepresented realms. Therefore, it is important that we are able to distinguish the unrepresented properties expressed by an instantiation of a design and the represented properties without confusion.

A second facet is that we use representations in distinct ways, which we label here *strong* and *weak representation*. Strong representation occurs when we imbue the representation with the ability to make algorithmic inference about its referent. A ubiquitous example is the now near-universal use of three-dimensional (3-D) modeling in architectural schools. The chief inference here is, of course, visualization. Students, and then designers, have found that the effort involved in building a 3-D model is more than offset by the ability to multiply render the resulting design. Another example, more prosaic, but perhaps even more ubiquitous, is the spreadsheet. With it the quick quantitative model of a budget, an architectural brief or a plan of work has become an essential part of much professional practice. Cognitive accounts of designing tend to focus on the strong aspects of a representation, as these are precisely the ones amenable to the constructive nature of the theory building at hand with its requirement for computationally executable theories. Weak representation happens when we use a representation as a reminder, hint, or framer of new insight. Situationist accounts of design focus mostly on weak aspects of representation, largely and precisely because the cognitivists have mostly ignored such.

A third facet is that, to use representations computationally, we must imbue them with exogenous properties, that is, ones that are manifestly not in the phenomena they purport to represent. We have to provide a set of operators by which we can change representations. We have to do this to use representations: the fact that designers work by search in a problem space means that they have to be able to alter representations and most such alterations do not follow the logic of physical objects represented. Good design representations are inherently about supporting change precisely to help designers move from idea to idea in their search.

Consider how the term *partial* relates to designs. Having a design representation does not mean you know all about a particular design. If you have a sketch in front of you, this is abundantly clear. It seems to be a lesson often lost on designers of computer-aided design (CAD) systems. Much hay has been made over the years through criticism that CAD systems force one to overcommit to particulars in comparison to sketches, which are claimed to be contingent in every respect. We argue that such criticism is sustained, and sustainable, because design representations are not thoroughly partial at every level. Partiality swims with intentionality. A design is *about* an artifact. That is no commitment

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

to being *all about* the artifact, and especially no commitment to being *as much about* the artifact as it might some day become. Designs are inherently partial: we add to them and subtract from them as a routine part of doing design. Furthermore, aboutness harbors information that may itself be partial: representations may refer to the same or distinct objects. For instance, we might have two identical representations of a column in a building, each playing a role in some part of a structural system, yet these two representations might be identified as referring to a single structure in a future version of the representation. Such function sharing is one of the oft-cited sources of design innovation (Ulrich & Seering, 1992).

A good representation to some degree must avoid these traps. It must not over commit to a specific and limited set of represented properties and feasible inferences. It should display some aspects of strong representation; otherwise, a significant part of the benefit of using a computer is lost. It must remain open to the weak uses of representation, or designers will resolutely suborn it to such. It must afford a disciplined notion of change; otherwise, it is not a design representation and certainly not a computable one. It must achieve a deep notion of partiality in how it represents designs.

2.2. Vastness

The domain of design imposes further conditions on representation that stem from the nature of the collections of design alternatives considered. In particular, design spaces are *vast* in the sense meant by Dennett (1995), namely, hyperastronomical in extent. For example, there are a finite but incomprehensibly large number of possible games of chess. Although almost all conceivable classes of designs are also vast, they remain a *vanishingly* small part of the entire design space, so that randomly selecting a worthwhile design is an event with a vanishingly small probability. Yet, without an oracle, we must begin at just such a random place and proceed to search.

In the analogous context of Darwin's theory of evolution, Dennett (1995) argues that expressiveness does not give an adequate account of possibility—the grade of possibility that matters is not whether a suitable artifact is expressible as a design, but whether a suitable design is accessible with reasonable effort to an explorer positioned somewhere in the design space. This line of reasoning can be expected to apply to other vast design spaces. Therefore, the explored fraction of a design space matters precisely because it decides the cost of accessing the unexplored designs.

Figure 1 demonstrates two dimensions of accessibility. Distance in the figure is a measure of derivational effort. Arcs link designs in the explored part of the space to designs not yet found. Following an arc expands the explored part of the space. First, effort counts. Design *b* will be more accessible than design *a*. Second, connectivity counts. Designs *c* and *d* gain an advantage over designs *a* and *b* through their connection to a larger number of already discovered designs.

Vastness has other implications to the implied mechanism of search. Effort counts in other ways than making

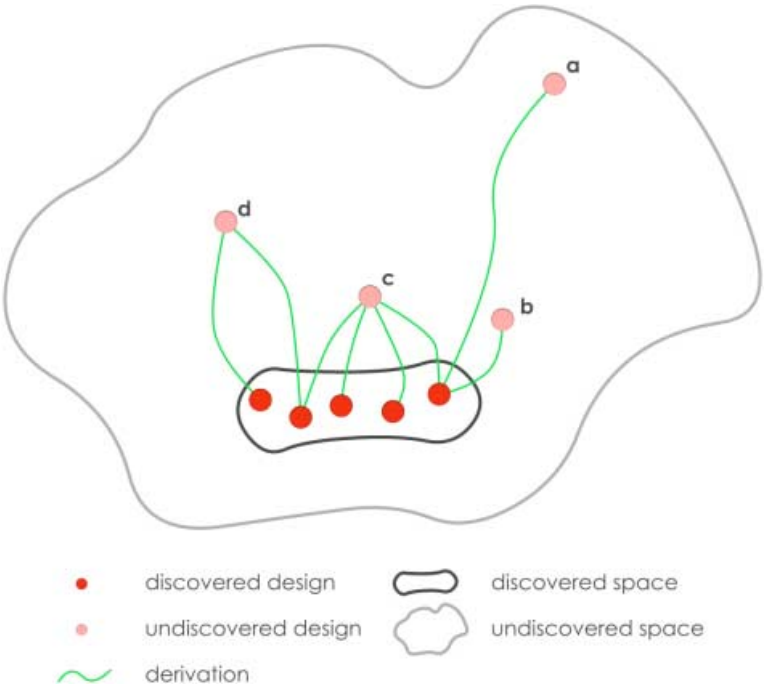

Fig. 1. The dimensions of design space accessibility. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

designs accessible along short paths. Design is time pressured; the fact of vastness of the space and finite human life guarantees this. The effort required to formulate a proposal is large relative to the time available for the entire process. Designs are precious productions and designers do return to them, to use them directly or to infer from them analogous structures and moves that can be applied to other designs. In terms of a design space, designers pull paths of actuality from existing designs and apply them to others in the space. In the time pressured world of designing, even partial and inconsistent designs find value as launching points into unexplored parts of the space. Designs themselves are important objects, irrespective of what they represent. That they are placed in the design space captures a position from which further exploration can be made. This is a corollary of the fact that accessibility is the measure of possibility: designs without physical interpretation or with poor qualities may be the basis for other realizable designs. Although this matter is also related to the expressiveness of the formalism, particularly whether it allows the description of abstract forms, the role of unsound designs derives from the importance of accessibility in the design space. Designs without satisfactory interpretations draw their utility from the designs that they make accessible either by forming a link in a chain of explored designs, or by providing a chain of exploration that may be reused by analogy.

2.3. Expressing intent

The primary design action of search in a space deeply colors the accounts of design intent that will be feasible. We can equally dismiss *a posteriori* rationalizations of design intent as self-serving, and atomistic efforts to include intent as overly strong representation. Intent is contextual, and the context is the thread of design space along which it comes into play. Design spaces enlarge the context for design decisions beyond the brief initiating the exploration. Just as design space accessibility stands for possibility, the correct context for interpreting the intent of design decisions is the position in the design space. When asking the *why* question concerning the *telos* of a feature of a design, or of the resulting artifact, the most satisfactory explanation will include the design space path and the alternatives forsaken. As the brief changes in the process of exploration, designs create the context for further decisions on them and the design space path records alternative paths considered. Thus, a move in the design space can be interpreted as seeking a new solution to a particular problem arising in some previous design, where in general the representation of the problem and its solution may remain implicit in both the antecedent and consequent designs. Put another way, the co-option of earlier features for new roles is a part of design that can only be understood with respect to the design space path. In analogy to Gould's (1990) claims about the necessity of history in any specific evolutionary account, design intention finds its most powerful explanations through a contingent history of process. Figure 2 shows a specific example. Justifications for residential air conditioning outside of the tropics come not solely from a desire for human comfort but also from the design decisions proposing a building configuration producing uncomfortable thermal environments. Figure 2 presents a hypothetical case of a design process for an actual building in a Mediterranean climate, namely, a house in Adelaide, Australia. The house begins as a functional enclosure for living. If the design was concluded at this stage, air conditioning would be a necessity because much vertical house surface is exposed to the sun. The house mass is extended to shape the ceiling/roof and to

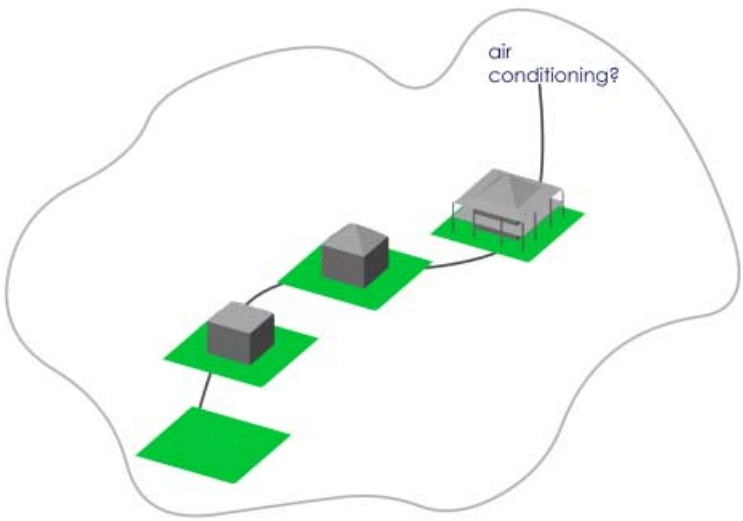

Fig. 2. A path developing a house design in a design space. Any decision to air condition, or indeed to recognize that air conditioning is important, is tempered by the decisions already taken along the path. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

provide an elevated balcony around the upper floor. For reasons not related to thermal comfort, design choices are made that the upper floor is lightweight wood, and the lower floor of masonry construction. Suddenly the need for air conditioning disappears as the heavy part of the construction is now shielded from absorbing solar radiation and acts as a heat sink during even protracted periods of hot weather.

We see from both professional (Wright, 1945) and scholarly (Bruton, 1997) literatures that derivation is a productive form of explanation. It is a recurrent theme in the generative design literature (Flemming, 1987*a*, 1987*b*). It is a typical device in papers reporting grammars for design corpora (Stiny & Mitchell, 1978, 1980; Chiou & Krishnamurti, 1996). The grammar rules are introduced, then comes the application of the rules to produce one or more designs in the space. This part of the exposition helps the reader knit together an understanding of the rules in joint action.

Designer action changes with experience. This is most commonly called learning. The process has explicit aspects: in mature learners at least, learning itself becomes an object of cognate action (Schön, 1983). Under the standard cognitive claims about the production-like quality of cognition, it would hardly be surprising if such a production-like quality surfaced in first-person accounts of design learning. It does (Wright, 1945; Archea, 1987; Woodbury, 1993). In summary, designers use production-like encodings of extant and newly learned actions. They use such encodings both internally and in communication with colleagues.

2.4. Summary of designer action

We claim that the above features of design representations are invariants of the exploration view and are necessary properties of a useful design space representation. In summary, representations must avoid the programmatic pitfall, must be intentional and partial, are inevitably limited in scope, carry both strong and weak representational qualia, and have properties exogenous to the artifact represented. That representations are used in search in a vast space makes accessibility the primary criterion of utility, drafts unsound designs into a potentially crucial role, and points to contingent historical accounts of design history as the best record of design intent.

Simon (1980) long argued that the limits of cognitive capability would seldom surface in observing problem solving action. Humans work within their limits. Entirely absent from unaided human problem solving are strategies that require the storage of even modest numbers of problem states. Indeed, the absence of such strategies forms a profound falsification test for cognitive theories of problem solving. Discovery of a single instance of such a strategy would deeply challenge the entire edifice. Yet, design space exploration is precisely about supporting humans to do design work such that the overall system engages in patterns of exploration that are beyond unaided human capability.

We know that external memory, of almost any form, is a significant amplifier of action. What is not, and cannot be, clear from accounts of designer action alone is the utility of providing computational means for designers to surmount these biological limits. Can designer action be so amplified?

Here, surely, is a case where the absence of evidence should not be taken as evidence of absence. Given the state of design space exploration work today, we have little evidence for amplification of designer action through supporting exploration. We also have little ground for arguing against such amplification. Having almost no systems that engage designers with multiple states, we can make only shallow and tentative inferences about their utility.

In closing our discussion about the domain that design space exploration intends to support, we would distance ourselves from some current debates in design research. Design space exploration takes a philosophical position that is neither cognitivist nor situated. As Bredo (1994) points out with equal relevance to both cognitivist and situationist ideologues:

> Yet difficulties with the approach would vanish if it were seen as simply an attempt to model human capabilities on the computer so as to better understand them. If the model were a tool, useful for solving certain problems, rather than "the way the world is" (Goodman, 1972), then it would be unobjectionable. All of the dualisms I outlined would then disappear along with the attempt to force human cognition into a given descriptive language. Computational models would simply be tools for working out the implications of certain formal theories.[1]

3. ACTION AMPLIFICATION

Simon notes repeatedly that we should expect to see primarily the properties of the task environment when we observe designer action. Except in extreme situations we do not see the situations in which cognitive limits are reached as problem solvers simply avoid such situations. Yet, it is precisely at such places where human cognition limits action that we should seek to amplify human ability. In addition, we know they exist. Because, with a few exceptions (Newell & Simon, 1972, p. 826), we do not have positive empirical accounts of designer limitations, we need to resort to argument and anecdote to build a catalog of opportunity. Presuming that humans find amplification of their abilities useful, we should expect to find opportunity at the boundaries of systems and we sketch examples of these below. The first refers to informal qualities of representations that are found, in some way, in almost every commercial CAD system. It appears here more as an existence proof for amplification than as the focus of design space explorers. The

[1] Either "cognitivist" or "situationist" could be substituted for "Computational" here with equal relevance.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

second, *codification*, is the strategy behind both scripting languages for CAD systems and the early rule-based generative design systems. The third looks to the record of past work, the *explicit space*, as a source of useful resources for work. The fourth, *implication*, points to the generative mechanism as helping a designer project potential implications of design moves. The fifth is *speed*, less of computation than of computationally supported processes. *Backup, recall*, and *replay* complete the suite of amplification strategies.

3.1. Representational prowess

External memory use is near ubiquitous for designers, and representation of designs is a primary function for external memory. A good account of the history of commercial design computing can be built on events introducing new representational tricks. One such trick was solid modeling, the representation of the geometry of 3-D solid objects. At its inception, solid modeling was seen as a key support technology for a larger enterprise of supporting design with computers.[2] This happened, but as things played out, boundary representation solid modeling also gained a life of its own and is now, in and of itself, a key design medium. Why did this occur? It turned out, that in analogy to Dennett's terminology (Dennett, 1995), solid modeling was *just enough* to give designers a playground and not *too much* to overburden them with specifics. The typical operations of abstract, primitive parametric solids, boundary tweaking, and Boolean operators allowed designers to make virtually any form they could conceive. There is no meaning captured beyond solidity; designers are free to make their own associations between the forms represented and the thing being designed. Rendering allowed designers to play with visual effect far beyond simple realistic veridicality with the world. Solid modeling with computers is the analog to the model-making shop still to be found in many design schools.

3.2. Codification

Numerous authors, both designers reflecting on their work and scholars observing designers, note processes of codification of design moves. These vary in temporal extent from within a single design episode (Akın, 1986) through individual projects (Archea, 1987; Flemming, 1987*a*) to constancies and changes through the development of a life's corpus (Knight, 1981). Designers preferentially use past successful moves, their own or others, in future projects. One of the long promises of grammars is the ability to explicitly encode such moves.

Such encoding coheres with the ability of computers to be active with data. Most professional tools come with a programming language that can be used to extend the tool to automate tasks. Apparently, explicit codification of action is important. Very few such languages support the production-like character that we noted above as a feature of much designer communication on design process. A partial exception is the venerable AutoLISP, recently reborn as Visual Lisp, which at least has structures that make production-like expression relatively easy. Designers and firms do use these codification tools, although seldom to the degree that academic purists might prefer.

The main strategy for amplification in early grammar formalisms and interpreters was the codification of design moves as grammar rules. Those that went from formalism to implementation, and especially to those used by other than their authors (Akın et al., 1997; Flemming, 1978, 1987*a*, 1987*b*, 1989; Heisserman, 1994; Heisserman et al., 2000) made this codification active, and thus, at some level, dealt with design space. In every case except for SEED-Config (Akın et al., 1997) and SEED-LOOS (Akın et al., 1997), the strategy was minimal: the design space was represented either through a subset of the language of designs generated by a grammar or primarily through strands of derivation taken one at a time. Carlson's (1993) thesis presented a unique exception. He presented a formalism called *grammatical programming* and implementation (GRAMMATICA) for embedding grammars in a nondeterministic functional programming environment. Grammatical programming provides control mechanisms richer than grammar, subroutine-like mechanisms for dealing with complexity, and the ability to use transformations other than rewrite rules. Further, grammatical programming can apply to spaces of parametric, constrained designs. Carlson provided an ability to recursively enumerate languages of designs specified by statements in a control algebra over transformations. Figure 3 reproduces one of its outputs, a drawing of an early Gothic tracery.

Codification is, at least in part, a successful amplification strategy. But just how does it work for the designer? What assurances and efficacies does it provide?

Authors such as Flemming (1987*a*), Archea (1987), and Bruton (1997) describe a mode of work with rule systems, either used formally with an interpreter or informally as a metaphorical guide to action, in which the designer acts as author of the rule system. The account goes that the designers work in circles by making rules, using them, observing their actions, modifying the rules and reentering the circle by using them again. Bruton (1997) captures this process especially well in the title of his thesis: *A Contingent Sense of Grammar*. The measure of rule utility is what it generates now and perhaps in future contexts. In general, we cannot assure formal properties in the utterances of a grammar. We might like to think that generative design holds forth the promise of operating within a distinct space of possibilities, for example, the space of well-formed Gaudian surfaces, but we cannot guarantee this space. The space is implicit in the given rules, which induce appropriate

[2] At a first approximation, we can make this claim without citation. One of the authors (Woodbury) was a participant in this process.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

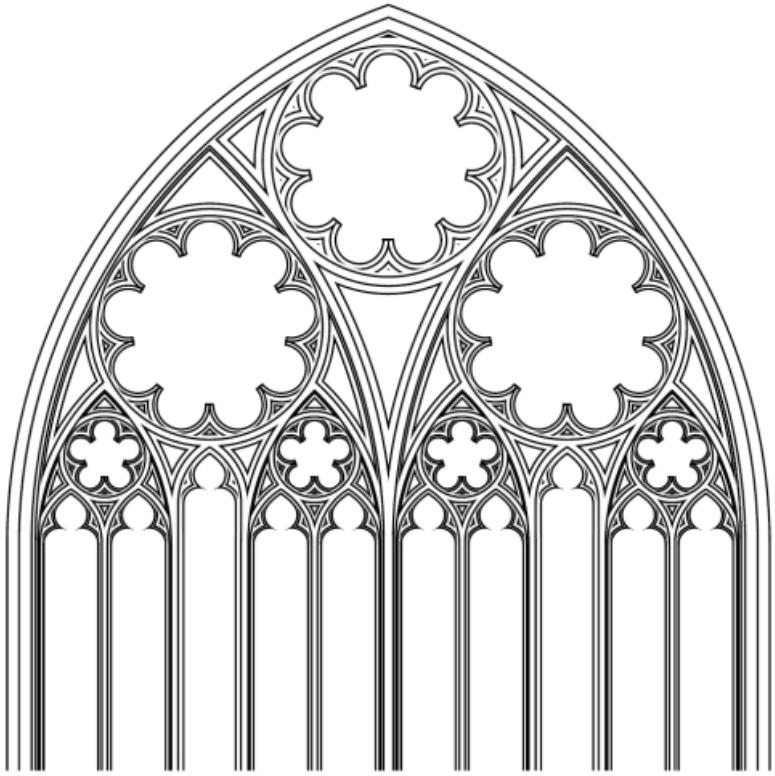

Fig. 3. A Gothic tracery generated as the output of a GRAMMATICA program.

designs or designs that can easily be ruled out. However, where do rules arise? When we presume that a generative system is restricted to a well-formed subset of the possible Gaudian surfaces, we do so on the basis of past moves in a freer design space. We have, in general, no device for ensuring either that we capture all well-formed Gaudian surfaces or that we do not generate ill-formed surfaces along with the good ones. All we can do is temper our expectations with experience in the more open context in which we, or others, built the rules. In this more fluid context, ultimately the unrestricted design space, a path records the co-option of design features and their assignments to roles. Rules ossify such experience. We have confidence in them to the extent that we can reinterpret the resulting designs ourselves, can follow the logic of the design space path or have enumerated enough outcomes to feel confident in future explorations. Each of these sources of surety is an illusion. Reinterpretable designs arise in weak representations that defer explicit commitments to the viewer and thus surrender potential ground for computational assistance. Understanding design space paths privileges the easily explainable over the creatively explored. Confidence in numbers is a simple recapitulation of the general logical error of induction of general laws from specific facts.

In the cases of both single-fronted cottages (Woodbury et al., 1999), the historically dominant cheap housing in Adelaide, and Queen Anne houses (Flemming, 1987*a*, 1987*b*), we are permissive in the interpretation of features and create logically coherent design space paths. Each single-fronted cottage consists of a hall the length of one wall and a parallel collection of nearly identical rooms. These are generated by rules that attend to the geometric pattern alone. The assignment of roles to rooms is deferred to a later stage of rule application. The rules are constructed to apply in stages and this orderly filling in of design detail aids both rule creation and comprehension of rule application. The same overall structure is mirrored in Flemming's Queen Anne house grammar (Flemming, 1987*a*, 1987*b*), which has a similar deferral of functional assignment and goes through successive stages of completion with different rules acting at each stage.

The accounts generated in exploration are thus necessarily both constructive and contingent. It is not clear that you can provide a computationally tractable and humanly understandable mechanism that mimics set theoretical selection, that is, selection based on the interpretations of designs. In a constructive world, be it evolution or design, accessibility is the only sound concept for discovery. Accessibility derives not only from the operations at hand in the design representation, but also from the reuse of recurring patterns in the threads of actuality that trace out the explicit design space: the space of visited designs. Reuse of threads is the intuition behind, for instance, case-based reasoning. However, from a design space explorer perspective, the case lies not in the individual designs but rather in the design space paths embedded in an explicit design space.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

3.3. The explicit space

Design space paths are embedded in both implicit and explicit design spaces. The former is the graph of all possibilities engendered by the codification system used. The latter is that subgraph that a designer, or a design organization, has previously visited. Ground in the explicit space is gained through designer work and it expands over time as a library of potentially recoverable, reusable and readaptable work. At least that is the theory. In practice, the storage and recall of past design states was, in 2005, hardly supported in any commercial system and hardly touched as an issue in research.

There are several reasons why the explicit space is important. Designers *recall* prior work and adapt it in new contexts. Design firms that invest in developing libraries of standard details are using a recall strategy. Designers may wish to *replay* paths previously discovered in a space, but in new contexts thus producing new results. This is the strategy behind systems, such as ArchiCAD, that provide procedural languages to write programs for adapting details to new contexts. Recall and replay are themselves important strategies worthy of their own sections in Sections 3.7 and 3.8, respectively. A third reason to be interested in the explicit space is that it contains alternative solutions to the design problem at hand. Alternatives are important because designs are inevitably evaluated along multiple criteria, only some of which are, or can be, captured in a given symbolic representation. They are also important because of the primacy of accessibility over possibility in making new discoveries in the implicit space. Last, they are important because the thread of actuality along which each develops is the best available carrier for explanations of design intent. Narratives of design intent provided by replay of thread, and embellished by commentary, themselves provide a basis for comparison and reflection on design possibilities.

Designers create alternatives as they work. According to Akın (2001), a main distinguishing mark of expert designers compared with nonexperts is the creation of comparatively more alternative problem formulations. Why alternatives? One part of the answer is revelation. Alternatives reveal things you have not considered, and thus suggest future avenues of exploration. Different alternatives reveal different avenues of exploration; they make new parts of design space accessible to future exploration. Further, different alternatives reveal not only oversights but also intangibles that cannot be considered formally in a symbolic representation. Another part of the answer is comparison. As a practical matter implied by finite capabilities and resources, including time, being available for any given design process, evaluation against criteria is an act of satisficing (Simon, 1955, 1956) according to which designs are accepted on the basis of their satisfying rather than optimizing performance against criteria. Important ground is gained though by satisficing against what is, or can be, known about the exploration so far. It is, rhetorically at least, better to claim that a design both satisfies criteria and is the best among those considered, than to simply make the claim that it satisfies.

As of 2005, available systems were resolutely based in a metaphor of the single state. One uses such a system to work on "a design" that one alters over time. The current design (one point and one point only in explicit space) is all that the system provides a user. Users of systems learn manual strategies of making versions, either through file copies or sometimes elaborate manual patterns of work with such structures as *layers* or *groups*, but systems themselves provide little or no systematic support for multiple states. Even such signal works as Chien's (1998) thesis builds design space navigation on a metaphor of a single explorer who is "located" at precisely one state at a time.

In design space, the primary view is of the space, in which states participate as members. Displays and interactions involving multiple states are in the set of elements from which interfaces to design space explorers will be built. There are facets to this broad claim.

A design space provides access to multiple points in explicit space. Interdesign comparison, multidesign interaction, and marking important places in the space are likely elements of interactions.

Interdesign comparisons potentially allow a designer to grasp possibilities within a region or on a front of explicit space. The SEED-Config (Woodbury & Chang, 1995; Akın et al., 1997) and discoverForm (Carlson & Woodbury, 1994) interfaces each provide negative examples of why interdesign comparison is important. SEED-Config provided separate windows in which alternatives could be visited, a window for a current design node and a layout of the nodes in its design space. One of these windows is shown in Figure 4. Clicking on a node produced a view of the design at that node in the current window. A combination of a specific key and a click opened a new window on the design of the clicked node. Designs were not well juxtaposed; the distance both in graphic space and interaction work between displays of designs made comparisons cumbersome.

The SEED-Config authors knew this well. When they prepared explanations of the system for publication (Akın et al., 1997) they crafted both abstracted examples of the interface using its metaphor of isolated views (Fig. 5) to explain development along a path, but also included a view of a set of designs from design space (Fig. 6). We interpret this inclusion as implicit admission of the need for better interdesign comparisons.

DiscoverForm is another negative example. Its main goal was immediacy in the interface. This is reflected in its declared primary principle of interaction, noted by Carlson and Woodbury (1994, p. 126) as "All changes to the motifs and clones are reflected immediately in the patterns that they generate."

Again we see an implicit admission that interdesign comparisons are important. Figure 7, which reproduces figure 2 of Woodbury (1993), shows two paths of exploration from

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

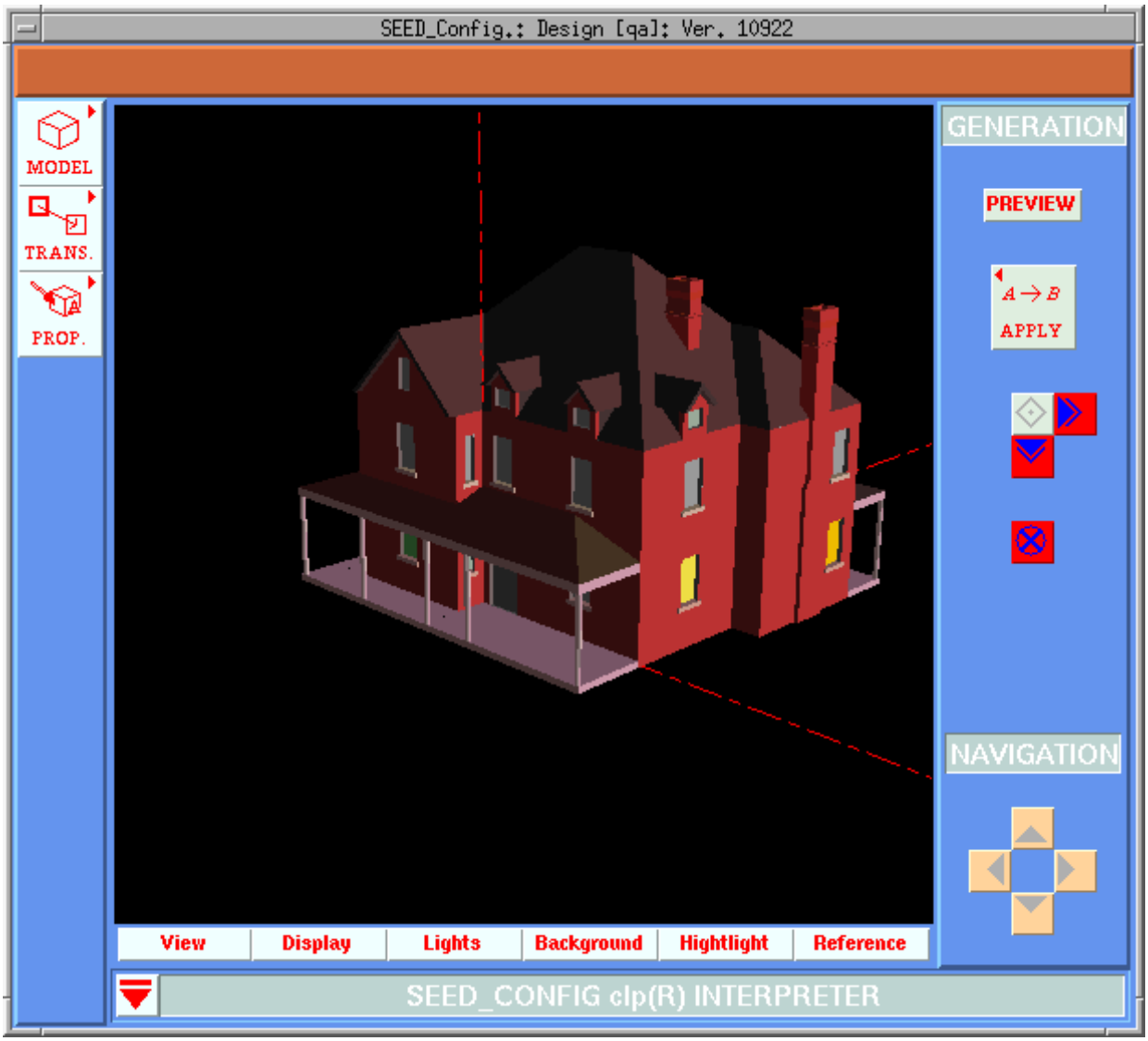

Fig. 4. A design node window in SEED-Config. Such windows could be opened on any state in a design space. Once opened, they were not visually linked to the design space in any way. [A color version of this figure can be viewed online at www.journals.cambridge.org]

an initial, symmetric tree. Carlson and Woodbury's (1994) figure 18 (not reproduced here) shows an exploration of a binary tree as a cycle through a rotation. Both of these diagrams were manually prepared and both illustrate the importance of making connections among states. Neither could be accessed as a whole in the discoverForm interface, and it is granted that such was not the goal of discoverForm.

Comparisons can be symbolic as well as visual. Woodbury et al. (2000*b*) argue for comparing representations based on information specificity. This yields a variety of comparison operators including equivalence, inclusion, commonality, difference, and consistency. In turn, this calls for a representation in which computation can play a role larger than state visualization in preparing comparisons of states for designer interaction.

Multidesign interaction labels the idea that, by making changes to an abstract state, a designer can effect changes to several more concrete states. Alternatively and equivalently in information terms, if a set of states share a feature or parameter, changing that feature logically changes all of the dependent states. To our knowledge, this strategy has received no explicit attention. The SEED-Layout system did have an ability to update prior states as a side effect of the finalization process in which a layout was dimensioned according to current constraints (U. Flemming, personal communication, 2003).

Marking prior places, or *landmarks*, in an exploration is a recurrent theme in research on navigation in complex information spaces. Chien (1998, p. 32) lists it as the first of four principles for designing navigation support in generative systems. Strong (1998, chap. 3) makes a similar argument in his thought experiments on cyberspace navigation. These are just two examples from many. Interestingly, they both use the same root examples of wayfinding in architec-

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

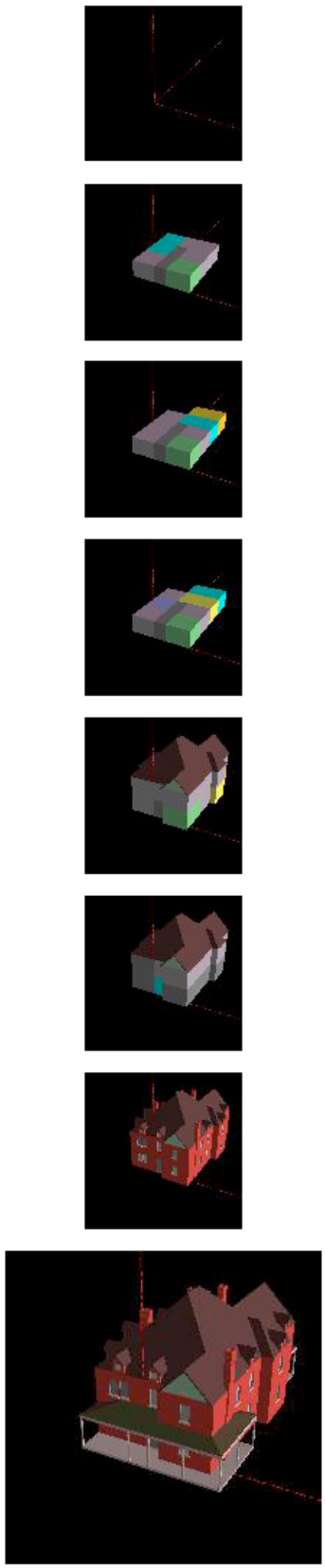

Fig. 5. The presentation of SEED-Config (adapted from Akın et al., 1997). An abstracted view of the separate windows provided by the interface explains development of a design along a path. [A color version of this figure can be viewed online at www.journals.cambridge.org]

tural and urban spaces (Lynch, 1960; Passini, 1984). In the current argument, the privileged place of accessibility over possibility in a vast design space provides an analytic argument for landmarks. Exploration work against the implicit space is a precious commodity. Such work itself can produce a set of states sufficiently large that the principle of accessibility over possibility becomes recursively relevant. By marking states with special exogenous meaning, users add a grade of accessibility to their explorations. It is interesting to note that this strategy has been found lacking in bookmark systems in Web browsers. Bookmarks quickly proliferate and themselves need organization. In these the recursion is carried a step further: bookmarks can themselves be classified into simple recursive labeled containers.

3.4. Implication and "implicature"

Construction of design space paths is a main task for design space exploration. At any state along a path, or in design space in general, exploration is largely conditioned by knowledge about both the present state and states reachable from that state by path traversal. Thus, an important property of a representation, be it of a state or a space of states, is what can be reliably inferred from it. Inference has long been a principal strategy for constructing design support tools. Arguably, it was the motivator for projects as early as Sketchpad (Sutherland, 1963), in which the designer was to build a geometric framework of which implications could subsequently be explored.

Implication as a strategy has exemplars in such common tools as renderers, animation packages, and spreadsheets. In renderers, a model is created and subsequently explored via renderings taken with different viewpoints, camera settings, lighting configurations, and material choices. Animation systems support a simulacrum of end-user experience of a design through automation of camera models over time. Spreadsheets implement a small step toward design space. In a spreadsheet, the structure is a cascade of equations that allows users to quickly see the implications of changing the independent variables at the top of the cascade. Designers such as Jay Parrish (personal communication, 2003) further use spreadsheets by developing codified, yet informal, models by which values in a spreadsheet are taken as parameters by which a design state, for example, a stadium is defined. Spreadsheets are an exemplar of the long-standing strategy of design parameterization, which, in 2005, is in revival in architecture through adaptation of such systems as CATIA (2003), and development within the AEC community of new systems, for example, Generative Components (Aish, 2002, 2004; Aish & Woodbury, 2005).

Deferral is the flip side of implication. Under some circumstances, it is worth designer effort to build a representation through which decisions can be deferred. Deferral only works when a representation supports some capability for implication. A typical use of parametric systems is to develop an example design that can then be adapted to fit a

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

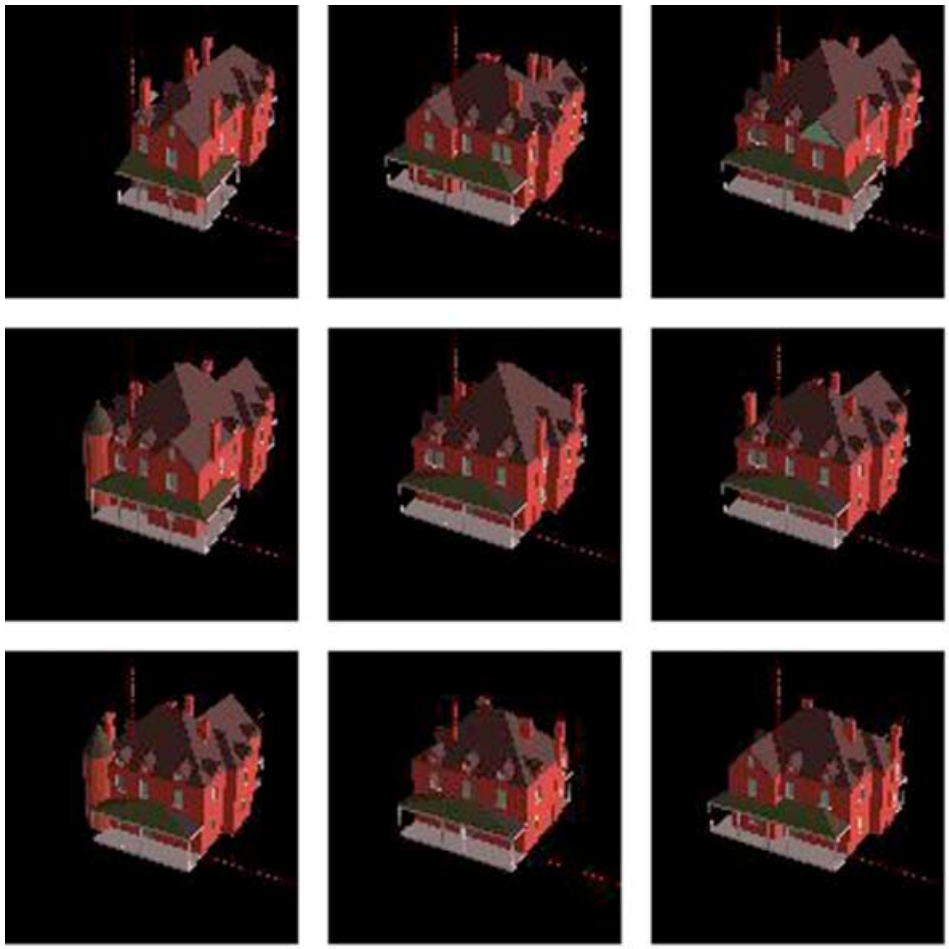

Fig. 6. Presentation of a set of designs produced by SEED-Config as a coherent whole (reprinted with permission from Akın et al., 1997). The interface to SEED-Config had no analogous view. [A color version of this figure can be viewed online at www.journals.cambridge.org]

particular geometric context. Such adaptation stands in contrast to the extensive rework that must be done to modify a complex design that is represented in a strictly nonparametric modeler, as demonstrated in Figure 8.

The resulting ability to make rapid changes along a limited range of variation is the primary argument for parameterization. Deferral strategies can also be found in the design and use of conventional CAD systems. For instance, assignment of logical material types throughout a design allows for rapid refinement of material properties when rendering from the design, In practice, rendering is an iterative process of changing material properties, lighting, viewpoints, and rendering settings. Figure 9 demonstrates the effect of this type of deferral. The appearance of a design can be quickly changed simply by reassignment of material properties, not in the design, but in a table of materials. Deferred decisions may be taken either automatically, for example, in parametric design systems, or manually, for example, in user choices within rendering systems.

For very good computational reasons, most automatic implementations of a deferral strategy involve situations in which the deferred decisions are linked to the design through deterministic, if not continuous, functions. Nondeterministic choice is much harder to support and there are few exemplars (but see Harada et al., 1995). Paths in design space inevitably capture nondeterministic and discrete changes; this is how the structure of a design develops. Nondeterminism creates branch points; discrete change can greatly alter what can be inferred from a design.

Implication has different meanings for design states and exploration paths. The implicature of a design state is what can be inferred about that state from its representation. A representation of a state may allow certain decisions to be deferred, that is, decided at some later time. In contrast, the implicature of a state along an exploration path is the other states and their implicatures that can be accessed by exploration from the given state. What can be implied along an exploration path is thus affected by the properties of designs that are preserved along the path. Unfortunately, what can be inferred along a path is limited in most design space explorers.

Grammars and their variants are perhaps the most common strategy for implementing generative systems, at least

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

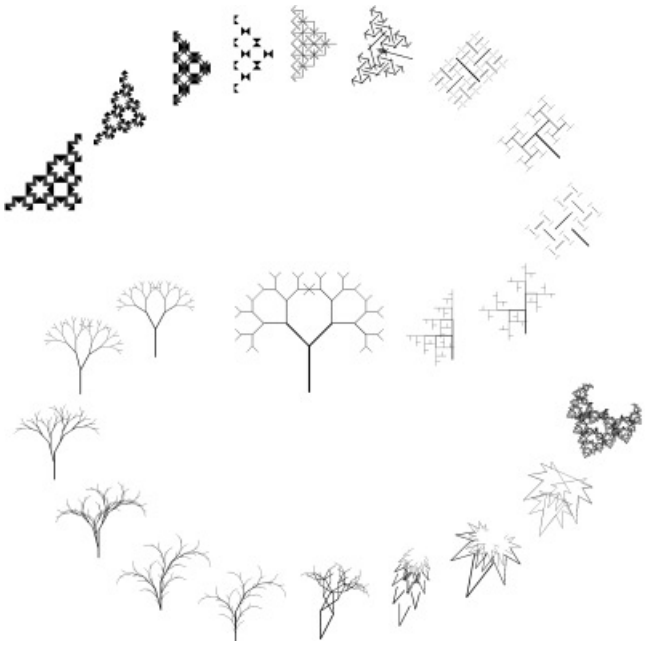

Fig. 7. An exploration in discoverForm (Woodbury, 1993) showing two paths of development. Changes to both cloning rule and motif are used freely throughout.

in the building domain. A grammar comprises a representation, a collection of rules, and an initial state. Used as a design space explorer, its sentential forms stand for the design space and its language to designs that are understood, in some sense, to be complete. Rules are typically defined as a left-hand side and a right-hand side. A *rule match* occurs by finding the left-hand side in a state. A *rule application* removes the left-hand side and adds the right-hand side under conditions determined by the rule match. A rule can thus map between arbitrary places in design space. In an extreme case, an entire design could be erased by its removal in the first part of a rule application and an arbitrary design put in its place.

In certain domains stronger guarantees on design space exploration are possible. Flemming (1987*b*, 1989) implemented *rectangular layouts* in such a way that the spatial relations between rectangles in a layout are invariant along an exploration path.

3.5. Speed

The finiteness of human existence puts intellectual curiosity at odds with time. In a rushed life the speed at which we can do things is important, as are the moments of silent reflection bought by speedy action. Standard accounts of human–computer interaction, for example, Card et al. (1983), place speed and accuracy as primary criteria against which to evaluate interactions with systems. When exploring design spaces, one measure of speed is against the unfolding of paths in design space. The focus here is not the time taken to compute a particular path, but rather the time taken to consider and make the choices along the path. That is not to say that efficiency in computing paths is not important, indeed it is a primary concern in developing a design space explorer representation. For example, Heisserman (1991) went to great lengths in his thesis to ensure that the rule matching and invocation mechanisms were efficient as computations. Rather, once you have an efficient mechanism, making choices will inevitably dominate the speed at which exploration can occur. There are at least two dimensions of choice. The first is simple consideration of the choices available at a given point in a path. There may be many such choices and it is common that authors of generative mechanisms pay considerable attention to reducing the choices available at a given time; for example, this is one of the primary practical uses of labels in shape grammars. The other dimension is one of anticipation. Exploration is contingent: we cannot know the range of possibilities that lie in front of a given partial path. That said, indications are important. For example, discoverForm employs a limited domain comprising a one-rule set grammar in which a path is uniquely determined by choice of motif, branching factor, and affine transformation (Carlson & Woodbury, 1994), but trades on providing rapid anticipation of path choice.

In comparison, the SEED-Config interface was dismal in looking forward to likely effects along paths. The user was

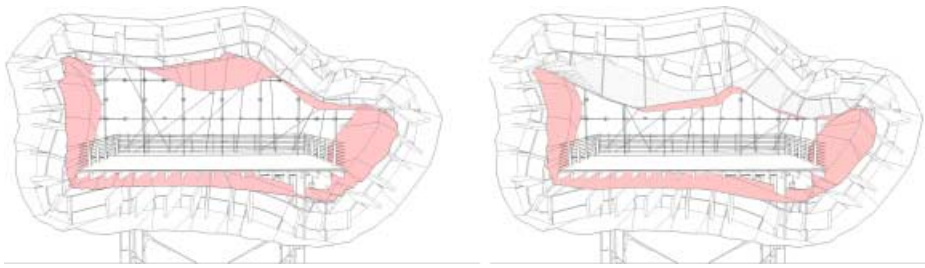

Fig. 8. A model of a theater created using a conventional solid modeler (form•Z). Considerable manual work is required to change the model from its original state on the left to a variant state on the right. This is despite the model being specifically crafted to take advantage of the operations provided by the modeling system. Reprinted with permission of Tristan d'Estree Sterk, The Office for Robotic Architectural Media (www.ofram.com), 2003. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

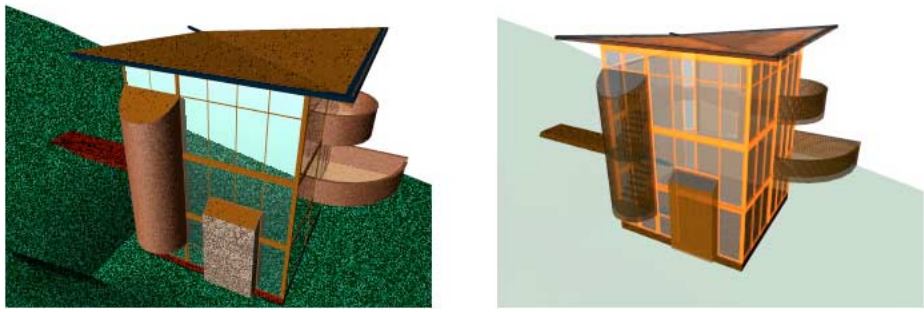

Fig. 9. A model of a house by Australian architect Max Pritchard. The model was created as part of a suite of teaching exercises (Woodbury et al., 2001; Adelaide University, 2002) aimed at learning the tools provided by CAD systems. It is configured so that changes to the properties of solids in the model can be easily made and propagated throughout the model (by permission). [A color version of this figure can be viewed online at www.journals.cambridge.org]

resolutely pinned down to local choices of rule application and had to manually explore from a particular point to gain some insight on likely future productions.

Harada and colleagues (Harada, 1997; Harada et al. 1995) showed an alternative approach. In essence, she put a grammar interpreter inside a physically based modeling interaction loop. When, in the midst of a user interaction, a parametric design reached a limit state along which no change was possible, the grammar interpreter performed a limited exploration and made a representational change using a principle of least disturbance of the interface. All this happened quickly within one interaction cycle. The resulting system was responsive to user input and resulted in a user strategy of dragging objects to explore possibilities. Harada did not record the space of possibilities explored, so did not provide explicit assistance in forming an impression of what might be possible from a particular point in design space.

3.6. Backup

Newell and Simon (1972, p. 826) note that humans use very limited backup strategies, and this is directly due to cognitive limits. In particular, people do not use search and scan strategies in which they consider a large number of alternatives and proceed from the prior alternatives that seem most promising at a given time. To some extent external media, teams, and assiduous librarianship can overcome these limitations, but such occurrences are relatively rare probably due to the time and expense involved. Synthetic problem solvers, on the other hand, are often designed with large fast memory structures, which make possible relatively profligate use of strategies employing backup. Almost all commercial CAD systems provide an *undo* command, essentially a backup along a single thread of development. One has only to work with and without such a limited form of backup to realize the degree to which even such a modest feature enhances work. In essence, having backup reduces the costs of recovery from a mistake to zero. The resulting sense of play is palpable. What is not clear is how to structure interaction with a large number of backup states. Yet, the number of potential backup states is not limited to those available along a single derivation thread. Woodbury et al. (2000*a*) describe *hysterical undo* as an operation that permits backup along paths not previously taken to a particular state. Chang and Woodbury (2003) demonstrate that user choices of Boolean operators in a conventional solid modeling system provide the basis for a form of backup that picks an expression of operators apart in a variety of ways. Both point to richer amplification strategies based on generalized backup mechanisms.

3.7. Recall

Backup implies proximity, either in derivational distance or in terms of design episode. We backup to prior states in a process. Other states may be every bit as interesting in a present context: but they may be unrelated by time, task, derivation, or author. *Recall* is the metaphor for such distant access.

Backup and recall comprise a necessary substrate on which exploration action grows. Normal design, and this in the sense of a large majority of design work, proceeds from the known to the unknown. Designers work on and through prior work. They make steps by modifying and adapting prior work until it becomes new and fresh. Although rarely employing carefully recorded sketches, designers sometimes make leaps from what they know to a new synthesis (Wright, 1945). Sometimes, and such moments are often recounted as memorable events, they combine two or more knowns into a hitherto unknown. In such a model recall is that action that draws precedent into play. The precedent may be the work of anyone: the designer, a colleague, another distant design professional, or it may be a vernacular production. It might even be a natural phenomenon. That precedent is valuable is easily seen in the regular and prolific publication of professional journals and picture books on architecture. That it is used can be seen by the presence of such publications, numerous photographs, and found objects that tend to collect in design offices. Devices for organizing such precedents are widely known and sometimes used: the office catalog, library, and CAD symbol collection are all examples.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

In the context of design space exploration then, an important amplification might be the ability to recall what you or someone else has done before. Such recall is, of course, one of the aspirations of the complex of work labeled case-based design. As we have argued elsewhere, and without disparaging the prior work on cases (Woodbury et al., 1999), considering the design space seriously leads to different conclusions on formalizing and implementing the functionality of cases. We will return to this issue in the discussion of representation and computation below.

Recall must operate on both wholes and parts, with the parts being more important than the wholes in practice. The chance of reusing an entire office design is significantly less than that of reusing a window detail.

A corollary of recall is knowing when you have been in a particular area of design space before. This is a generalization of the duplicate detection problem in generative systems in which it is important either to recognize when you reach a prior design or to avoid ever doing it at all. Knowing when you are in the neighborhood of things you tried before could save a lot of work.

What is not clear is how recall can be made to work. This can be seen in difficulties encountered in recall in a much, much smaller space than the ones under consideration here: the World Wide Web. A major current issue here is searching. It is bluntly clear that current search tools are inadequate to the task on the Web at large; further, the available ways of storing and recalling even the relatively small number of bookmarks generated by a single person are hardly better. There is some light, though, in such systems as CZWeb (Fisher et al., 1997), which provided a way of organizing and communicating bookmarks such that they related visually to the logical structure of the browsed Web sites and to a user's choices about appropriate visual layout.

3.8. Replay

Once you have recalled, how do you use the recalled thing in a new context? In a professional context, the question might be "How do we use this detail we have found in our library?"

To this question, conventional CAD systems have a definitive answer: *copy and paste*. The standard CAD system representation is an extreme of declarativeness: it simply describes a set of geometric and attribute objects. These generally have an independent existence and can be simply put into a new context. The user has the entire responsibility of properly relating the inserted objects to what is already in the design, and this responsibility is typically not onerous. Most CAD systems, and most styles of using them, veer to the weak end of a weak to strong representational spectrum. Relatively few properties of the designed object are maintained formally in the representation. Examples include dimensions in working drawings, and polygon integrity and material properties in models used for rendering. *Copy and paste* works: it has become a near-universal user interface device across a wide range of system types. However, it is fragile; it is challenged by the slightest hint of interobject relations. Consider spreadsheets, an exemplar of which is Excel. The underlying abstract data structure of most spreadsheets is a unidirectional constraint network in which links between nodes model the propagation of the value of a source node into the equation that, in its turn, computes the value of a sink node. Using *copy and paste* is tricky in spreadsheets, as all users of these systems quickly discover. The usual metaphor is that, within the formula of a cell, a reference to another cell is made either in absolute or relative terms, or a mix of the two. When doing *copy and paste*, the user must be aware of, and must control, which reference style he or she has used. Users of spreadsheets develop an idiom of use, which might be labeled *copy paste and patchReferences*,[3] in which users first try to *copy and paste* and, if it fails to produce a sensible result, either patch the reference style in the copied cells and *copy and paste* again, or patch the copied equation itself.

In contrast, most of the representations used in generative design systems and demonstrations have a stronger character and this seems to be required by the codification of designer experience in the design generator. For example, shape grammars for design corpora universally make profligate use of labels to carry intentional information (Stiny & Mitchell, 1978, 1980; Chiou & Krishnamurti, 1996). This style of programming seems idiomatic: grammars in systems such as GENESIS (Heisserman 1991, 1994) and Grammatica (Carlson 1993) make extensive use of those systems' analogues of labels. The information states of these labels are typically fragile; they relate to specific places in a generation process.

The natural analogue to *copy and paste* in generative systems is *copy and apply path* where a *path* is a sequence of design states related by rule application or other moves to a *start state* in a design space. Copying a path copies not the design states, but the moves that generate the state sequence from the start state. Applying a thus-copied path is to reexecute the rule applications from a different start state (Fig. 10).

Path reuse has been a *bête noir* of generative systems. To our knowledge, no grammar-based generative system has implemented it, despite considerable discussion of it among developers. Yet, is it a key amplification point for design systems.

4. REPRESENTATIONS AND COMPUTATION OF DESIGN SPACE

A core feature of system designs for design space exploration is explicit representation of the space of discovered

[3]This claim is based on anecdote, not empirical study. The basis of the claim is that one of the authors (Woodbury) uses spreadsheets extensively and has taught students how to use them.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

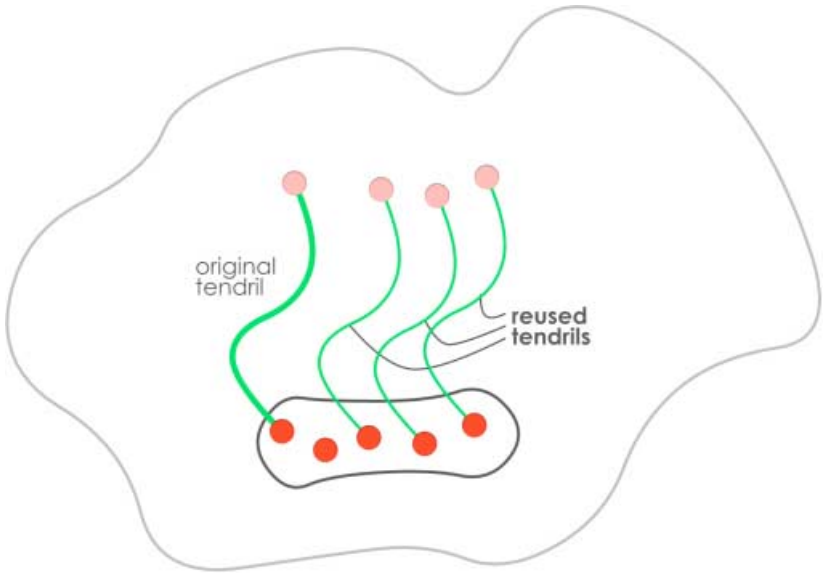

Fig. 10. Path reuse can either be through the recognition of homologous pattern in the ancestry or analogous pattern in design form. In both cases the ability to reuse known paths is crucial in enabling design space exploration. [A color version of this figure can be viewed online at www.journals.cambridge.org]

designs, namely, the explicit design space. It necessarily inherits the properties of the task environment (designer action by search in a problem space) that it must serve. As we argued above, these include intentionality, partiality, vastness, and the attendant primacy of accessibility over possibility. Further, a design space representation would seek to support one or more of the strategies outlined in Section 3.

What are the implications for design space representation? All such must arise against the constraints that a useful representation must be simultaneously understandable to its users and computable by a machine. Understandability may take many forms, but a chief strategy is to present to the user in terms of the tasks being undertaken. In the case of design space exploration, this is in terms of all the earlier-mentioned elements of its task environment and some, or all, of the amplification strategies. Computability means that we must carefully attend to using only algorithms and representations that work within the finite resources available to computation.

We call out some principal constraints from both understandability and computation. Work by search in a problem space immediately implies that some of the possible states will have been visited and some not. The visited ones are accessible by recall and subject to interstate comparisons. The unvisited states are possible and may become accessible through future acts of exploration. We distinguish these two classes of states by the designation of two subsets of design space. The *implicit space* is that made possible by the generative mechanism. The *explicit space* comprises those states that have been visited, in the current or an available past exploration episode.

The vastness of design space is a permanent constraint. Examples of vast spaces are the game tree of chess or the collection of all possible 500-page ASCII manuscripts. The latter has more states than exist protons in the universe. The fact is more often appreciated than the converse: that these spaces are, in fact, finite, are composed of finite objects, and are reachable by finite computations. The upshot of vast spaces, composed of finite elements, is that there are clear shortest paths in the design space, and that such paths are generally tractable. For example, a book in the library of Babel (Borges, 1962) is reachable in a year to anyone prepared to type a couple of pages a day. However, to consider the contents of the same library is to boggle the mind. There is no class of English literature, or literature in any language, for example, biographies of the reader, that it does not house in vast quantities. Therefore, we may be reasonably extravagant with algorithms that process paths in the design space, because they deal with tractable numbers of elements, although we must strenuously avoid algorithms that attempt brute force work against anything but a vanishingly small fraction of the design space. Design space exploration thus inevitably mixes human intervention with machine generation; it is a *mixed-initiative enterprise*.

The term mixed initiative has a specific meaning in the wider literature, which we adopt without change here. It refers to the understanding of interaction with a system in terms of the communication that would occur were the system a colleague. Interaction changes from issuance of commands and reception of results to a multilevel system of *dialogue*, in which agents communicate within a domain to coordinate tasks and achieve goals (Allen, 1999; Datta, 2004). An implication for design space explorers is that the opportunities for external access to any generative mechanism need to be fine grained. This is not a claim that all interaction must be fine grained, just that it can be. In a vast

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

space, task control becomes crucial: search is intractable, but movement along specific paths is efficient. Conversely, given a mechanism that can actually do much of the hard work of developing designs along given paths, the mechanism needs, at times, to retain initiative.

In the mixed-initiative enterprise of design space exploration, there are at least two important sources of clarity to exploit. The first is navigation, and the second is recombination.

Navigation is cognate movement through a space. It involves both moving along and understanding your path. Navigation in implicit and explicit spaces is subject to the same constraint: it can only be done in reference to what is already known. This constraint manifests differently in each space. In implicit space, a navigator stands at the frontier of the explicit and must base action on some location on that frontier. She has in hand a set of operators, and some necessarily fallacious, but heuristically useful knowledge about what those operators might do. In explicit space, a designer has a richer context of reference. She can move from known state to known state along explicit paths that express some measure of relatedness. She can search through indexes of design state properties.

A flaw in standard rule-based accounts of design space exploration in implicit space is that the usual formulation of rules cast navigation solely in terms of derivation, thus putting the landscape of the explicit space forever beyond the sight of the navigator. Applying rules of the form $\alpha \rightarrow \beta$ involve removing a transformed version of α from the design and substituting for it an identically transformed version of β. This means that the granule of movement specified in a rule can go from one point in a design space to another arbitrary point irrespective of any underlying design space order. Rules of the form *match and apply* create a similar screen between derivation and design space structure to the extent that the apply part of the rules does not create an explicit account of its movement through the structure. Recasting the devices of navigation from rules to operations that make explicit reference to underlying structure permits us navigators to know more about the paths we have taken (Woodbury et al., 2000*a*).

Recombination generalizes the notion of *copy and apply path* argued above as a natural analogue to *copy and paste* in more conventional systems. Recombination permits the application of past navigation to future explorations. For example, we may splice together paths, or reorder the steps along a path. A given design space representation will make a variety of such operations feasible, but we require certain regularities in the representation of design states that allow us to cheaply and robustly reapply design transformations, and to recognize some monotonically increasing properties along the paths.

Against the vastness of the design space, it is clear that we cannot hope to exhaustively optimize. Even a vanishingly small subclass of the design space is, in general, also vast, so that design problems are examples of what Simon (1955, 1956, 1980) calls "satisficing." For example, you can only expect to arrange a vanishingly small fraction of the competent autobiographies contained in the library of Babel, and those that you will reach will be by refinement of your original attempt. Therefore, it is crucial that the organization of the explored space makes possible the reuse of drafts.

If, as we argued above, the best accounts of intent will necessarily include the decision history of design process, then narrative becomes an important part of a design system. The bet here is that telling stories of design decisions, including the decisions foregone, begets new understanding. Efficient navigation and recombination reinvigorate such narrative. To explain the *telos* of a design, simply replay an account of its creation. If the navigation mechanism embodies structure other than derivation, for example, plausible explanations other than those followed by the designer might be available and such explanations could just be interesting. Navigation and recombination reify story telling as a design tool.

Explicit space is necessarily a vanishingly small part of design space, but that does not mean that it is small in computational terms. Over design episodes on a project and multiple projects it is likely to grow to very large size. This puts a premium on methods of recall that are both efficient in computational resources and effective to the designer. Similarity is the usual term used for such recall, and is the stock in trade of the large case-based reasoning literature. Again, we call on a necessity for regularities in design space to both formalize and make efficient notions of similarity-based recall. An insight is to use the structure of the design space itself as the recall mechanism. To recall a past design, put an approximation of part of it into a design space. The designs proximal to it will be similar by the measures implied by design space structure.

The calls for navigation richer than derivation, robust recombination, recapitulation through replay and content-based recall all imply a primary space structuring mechanism other than rewrite rules, although rewrite rules could be built on top of such a mechanism. We are aware of at least one such computationally efficient mechanism, and this is based on a formalism known as *typed feature structures*. The essential information structuring relation here is *subsumption*, that is, a relation of information specificity. One state subsumes another if it contains strictly less information than the other. With some restrictions on the kinds of information carried, notably functional values for features, subsumption can be computationally efficient. Further, the formalism provides a means of codification and generation involving *type hierarchies* as an analogue to rewrite rules. It turns out the consequent style of expressing design information is familiar in computational design circles. It is essentially that of *constraint languages* in which users build descriptions of collections of objects as networks in which the arcs express either constraints or equality among variables depending on the view taken. In constraint languages,

the utterances are used directly as design descriptions. In typed feature structures, they effectively become the rules that control generation through a *resolution* process, specifically called π-resolution.

Woodbury et al. (1999) introduced the typed feature structures representation to design space exploration, argued its relevant properties, and showed how it could be used to model simple housing designs.

Burrow and Woodbury (1999) gave precise definition to the typed feature structures representation as adapted from Carpenter (1992), and showed how to make the π-resolution incremental. An incremental algorithm in which each stage of the resolution process is open to user interaction is a crucial property for a mixed-initiative design space explorer.

Woodbury et al. (2000*a*) introduced a set of fundamental operators over typed feature structure subsumption networks and showed how these could be used to generate explicit design spaces and navigate design spaces in general. In his PhD thesis, Chang (1999) showed how the typed feature structures representation could be extended to include objects, such as solids, that have indefinite subparts. He showed simple examples of generation using the extended representation.

Datta's (2004) thesis constructs a mixed-initiative model for interacting with subsumption-based representations in general and the extended typed feature structures representation in particular.

In his forthcoming thesis, Burrow (2005) has shown that efficient design spaces can be organized around typed feature structures and π-resolution. For the first time, a formalism simultaneously captures generation, navigation, recombination, recapitulation and recall.

The typed feature structure representation provides a computationally efficient substrate for design space exploration. In order to do this, it imposes rigorous constraints on the properties of a representation; it introduces significant properties that are exogenous to the task of simply representing a design. The representation supports the properties we argued for in the discussion of the design space exploration domain in Section 2, namely abstraction above the program, partiality, intentionality, and the ability to provide both strong and weak representations. Its form of codification is more declarative than conventional rules: a user describes structure rather than operations. Its algorithms are incremental at a fine-grained level of the generation process, thus allowing for mixed-initiative interaction at almost any stage of the generation and navigation process.

The representation is young. In front of it lie the sorts of effective, medium-scale demonstrations achieved by Heisserman (1991), Woodbury et al. (1992), Carlson (1993), Carlson and Woodbury (1994), Akın et al. (1997), and Harada (1997). Far in front of it lies serious industrial application (Heisserman et al., 2000). It is though, the first representation that gives system designers a meaningful handle on the primary object of research in the field, the design space.

Without knowing the periodicity and lifetime of patterns in the design space, it is better to equip the designer with an explorer that can be reshaped and paths that can be reused as patterns emerge in the *work* of constructing threads of actuality. This is another reason why design space explorers are mixed-initiative systems, not viewers of ready-made designs for a human. They are systems requiring reflective thought from designers.

5. ANALOGIES WITH EVOLUTIONARY THEORY: HOMOLOGIES, ANALOGIES, AND TAXONOMY

We might construct a generative design system by codifying the past successful moves in the design space. However, will this help us find new designs for say, fire stations? We may find surprising results, fire station designs hidden in the rules, but this is limited, because the rules have ossified past decisions. Although the rules work, they operate within the scope of the experience from which they were drawn; once outside this scope, the likelihood of striking a productive region in the underlying design space is minute. A borrowed move can be interpreted as seeking a solution to the problem that arose in an earlier design, but because the *telos* of these moves is bound up in the earlier path, reuse under matching is not guaranteed to be productive. These observations, based on ideas of path reuse, show both coherence and difference with the most significant design space of which we have even a partial record. This is the design space of life.

This section is almost purely speculative. In it we briefly discuss a few ideas from evolutionary theory that might be helpful in thinking through potential design space formalisms.

Homology and *analogy* are terms describing two modes of feature sharing. In the case of homology, features are shared by common inheritance. An example is the anatomy shared by reptiles and mammals, which is evidence of shared ancestry. This exemplifies our equation between accessibility and possibility. In a design space explorer, homology is important to recognize: when a branched path encounters a problem that is solved by transforming a shared feature on one branch, it is possible that another branch can transform the feature in a similar manner. Note that this is very unlike evolutionary theory, in which homologous structures endure along a thread of actuality. Here we posit that we can introduce homology structures because the new structures can be built on homologous foundations. Therefore, homology is a useful trait to search for, and one that can be handled by backtracking along paths. However, analogy also occurs outside of homology where the designer is able to synthesize a relation between the two designs. In evolutionary terms such is an instance of convergence, in which similar features evolve over time from nonhomologous parts of the tree.

https://doi.org/10.1017/S0890060406060057 Published online by Cambridge University Press

The activity of synthesizing relations across the design space is an act of *taxonomy* that seeks to uncover notions of purpose in the arrangement of features. It need not be sound or complete, because the notions of essential purpose have been disposed of in place of a *derived purpose* occurring within the context of design space paths. In the activity of design, making the past accessible by synthesis and critique is a discipline intended to hone the reuse of path fragments. It serves to keep alive the visited filaments in the design space and therefore to maintain accessibility.

Whether by homology or analogy, a goal for design space explorers must be the robust reuse of paths of exploration.

6. SUMMARY

To take the metaphor of design space exploration seriously is to admit the design space as the primary object of research. Possible structures for design space are conditioned by models of exploration behavior by designers, by choices of strategies for amplifying designer action, and by the limits imposed by both computation itself and our knowledge of it. Formalisms for design space exploration must simultaneously accord with designer action, implement a useful amplification strategy, and be both formalizable and computationally tractable.

ACKNOWLEDGMENTS

This work was partially supported through the Australian Research Council Large Grants Scheme, the Australian Research Council Small Grants Scheme, the Canadian National Sciences and Engineering Research Council Research Grants Program, the Advanced Systems Institute of British Columbia Provincial Research Fellowships Program, the University of Adelaide, and Simon Fraser University.

REFERENCES

Adelaide University. (2002). *The Architecture Games*. Accessed at www.arch.adelaide.edu.au/games on December 2, 2005.

Aish, R. (2002). *Smart Geometry*. Accessed at www.smartgeometry.com/bd.htm on December 2, 2005.

Aish, R. (2004). *Bentley's Generative Components. A design tool for exploratory architecture*. Accessed at www.cmu.edu/architecture/graduate/G-CAD/pdf/GenerativeComponents.pdf on December 2, 2005.

Aish, R., & Woodbury, R. (2005). Multi-level interaction in parametric design. In *Proc. SmartGraphics, 5th Int. Symp., SG2005, Lecture Notes in Computer Science 3638* (Butz, A., Fisher, B., Krüger, A. & Oliver, P., Eds.), pp. 151–162. Berlin: Springer.

Akın, Ö. (1986). *The Psychology of Architectural Design*. London: Pion.

Akın, Ö. (2001). Variants of design cognition. In *Design Knowing and Learning: Cognition in Design Education* (Eastman, C., Newstetter, W., & McCracken, M., Eds.), pp. 105–124. New York: Elsevier.

Akın, Ö., Aygen, Z., Chang, T.-W., Chien, S.-F., Choi, B., Donia, M., Fenves, S.J., Flemming, U., Garrett, J.H., Gomez, N., Kiliccote, H., Rivard, H., Sen, R., Snyder, J., Tsai, W.-J., Woodbury, R., & Zhang, Y. (1997). SEED: a software environment to support the early phases of building design. *The International Journal of Design Computing*. Accessed at www.arch.usyd.edu.au/kcdc/journal/vol1/papers/flemming/index.html on December 2, 2005.

Allen, J. (1999). Mixed initiative interaction. *Proceedings of the IEEE Intelligent Systems 12(6)*, 14–23.

Archea, J. (1987). Puzzle-making: what architects do when no one is looking. In *Computability of Design, Principles of Computer-Aided Design* (Kalay, Y., Ed.), pp. 37–52. New York: Wiley.

Borges, J. (1962). The Library of Babel. In *Labyrinths: Selected Stories and Other Writings*, pp. 51–58. New York: New Directions.

Bredo, R. (1994). *Cognitivism, Situated Cognition and Deweyian Pragmatism*. Accessed at www.ed.uiuc.edu/EPS/PES-yearbook/94_docs/BREDO.HTM on December 2, 2005.

Brown, D.C. (1984). *Expert systems for design problem-solving using design refinement with plan selection and redesign*. PhD Thesis. Ohio State University, Department of Computer and Information Science.

Brown, D.C., & Chandrasekaran, B. (1989). *Design Problem Solving: Knowledge Structures and Control Strategies. Research Notes in Artificial Intelligence*. London: Pitman.

Bruton, D. (1997). *A contingent sense of grammar*. PhD Thesis. University of Adelaide.

Burrow, A. (2005). *Typed feature structures and design space exploration*. PhD Thesis. University of Adelaide.

Burrow, A., & Woodbury, R. (1999). π-Resolution in design space exploration. In *Computers in Building: Proc. CAAD Futures '99 Conf.* (Augenbroe, G., & Eastman, C., Eds.), pp. 291–308. Dordrecht: Kluwer Academic.

Carlson, C. (1993). *Grammatical programming: an algebraic approach to the description of design spaces*. PhD Thesis. Carnegie Mellon University, Department of Architecture.

Carlson, C., & Woodbury, R. (1994). Hands-on exploration of recursive patterns. *Languages of Design 2(2)*, 121–142.

Carpenter, B. (1992). *The Logic of Typed Feature Structures with Applications to Unification Grammars, Logic Programs and Constraint Resolution. Cambridge Tracts in Theoretical Computer Science*. New York: Cambridge University Press.

Catia. (2003). *The CATIA system*. Accessed at www.3ds.com/products-solutions/brands/CATIA/ on December 2, 2005.

Chandrasekaran, B. (1990). Design problem solving: a task analysis. *AI Magazine Winter*, 59–71.

Chandrasekaran, B., & Johnson, T.R. (1993). Generic tasks and task structures: history, critique and new directions. In *Second Generation Expert Systems* (David, J.-M., Krivine, J.-P., & Simmons, R., Eds.), pp. 232–272. Berlin: Springer–Verlag.

Chang, T.W. (1999). *Geometric typed feature structures: toward design space exploration*. PhD Thesis. University of Adelaide.

Chang, W., & Woodbury, R.F. (2003). *ACADIA 2003*, pp. 19–27, Indianapolis, IN, October.

Chien, S.-F. (1998). *Supporting information navigation in generative design systems*. PhD Thesis. Carnegie Mellon University, Department of Architecture.

Chien, S.-F., & Flemming, U. (1997). Information navigation in generative design systems. *Proc. Second Conf. Computer Aided Architectural Design Research in Asia, CAADRIA 97* (Yu-Tung Liu, J.-Y. Tsou, J.-H.H., Eds.), pp. 355–366. Hsinchu, Taiwan: National Chia Tung University.

Chiou, S.C., & Krishnamurti, R. (1996). Example Taiwanese traditional houses. *Environment and Planning B: Planning and Design 23(1)*, 91–116.

Datta, S. (2004). *Unfolding design spaces interactively*. PhD Thesis. University of Adelaide.

Dennett, D.C. (1995). *Darwin's Dangerous Idea: Evolution and the Meanings of Life*. New York: Simon & Schuster.

Fisher, B., Agelidis, G., Dill, J., Tan, P., Collaud, G., & Jones, C. (1997). CZWeb: fish-eye views for visualizing the World-Wide Web. *Proc. Seventh Int. Conf. Human–Computer Interaction (HCI Int. '97)*, pp. 719–722.

Flemming, U. (1978). Wall representations of rectangular dissections and their use in automated space allocation. *Environment and Planning B: Planning and Design 5(2)*, 215–232.

Flemming, U. (1987*a*). The role of shape grammars in the analysis and creation of designs. In *Computability of Design, Principles of Computer-Aided Design* (Kalay, Y., Ed.), pp. 245–272. New York: Wiley–Interscience.

Flemming, U. (1987*b*). More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design 14(3)*, 323–350.

Flemming, U. (1989). More on the representation and generation of loosely packed arrangements of rectangles. *Environment and Planning B: Planning and Design 16(3)*, 327–359.

Flemming, U. (1990). Knowledge representation and acquisition in the LOOS system. *Building and Environment 25(3)*, 209–219.
Flemming, U. (2003). Personal communication.
Forgy, C. (1981). *OPS5 User's Manual*. Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science.
Gould, S.J. (1990). *Wonderful Life: The Burgess Shale and the Nature of History*. New York: Norton.
Harada, M. (1997). *Discrete/continuous design exploration by direct manipulation*. PhD Thesis. Carnegie Mellon University, Department of Architecture.
Harada, M., Witkin, A.P., & Baraff, D. (1995). Interactive physically-based manipulation of discrete/continuous models. *Proc. SIGGRAPH 1995*, pp. 199–208.
Heisserman, J. (1991). *Generative geometric design and boundary solid grammars*. PhD Thesis. Carnegie Mellon University, Department of Architecture.
Heisserman, J. (1994). Generative geometric design. *IEEE Computer Graphics and Applications 14(2)*, 37–45.
Heisserman, J., Callahan, S., & Mattikalli, R. (2000). A design representation to support automated design generation. In *Artificial Intelligence in Design 00* (Gero, J., Ed.), pp. 545–566. Dordrecht: Kluwer Academic.
Knight, T. (1981). Languages of designs: from known to new. *Environment and Planning B: Planning and Design 8(2)*, 213–238.
Lynch, K. (1960). *The Image of the City*. Cambridge, MA: MIT Press.
McDermott, J. (1982). R1: a rule-based configurer of computer systems. *Artificial Intelligence 19(1)*, 39–88.
Newell, A. The knowledge level. (1981). *AI Magazine Summer*, 1–19.
Newell, A., & Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice–Hall.
Parrish, J. (2003). Personal communication.
Passini, R. (1984). *Wayfinding in Architecture*. New York: Van Nostrand Reinhold.
Schön, D. (1983). *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books.
Simon, H.A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics 69(1)*, 99–118.
Simon, H.A. (1956). Rational choice and the structure of the environment. *Psychological Review 63(1)*, 129–138.
Simon, H.A. (1980). *The Sciences of the Artificial*, 2nd ed. Cambridge, MA: MIT Press.
Sterk, T.d. (2003). Personal communication, Office for Robotic Architectural Media (www.ofa.com).
Stiny, G., & Mitchell, W.J. (1978). The Palladian grammar. *Environment and Planning B 5*, 5–18.
Stiny, G., & Mitchell, W.J. (1980). The grammar of paradise: on the generation of Mughul gardens. *Environment and Planning B: Planning and Design, 7(2)*, 206–226.
Strong, J. (1998). *Cognitive architecture, bridging the gap between real and virtual environmental design*. Honours Thesis. University of Adelaide, School of Architecture, Landscape Architecture and Urban Design.
Sutherland, I.E. (1963). *Sketchpad: A Man–Machine Graphical Communication System.* Cambridge, MA: MIT Lincoln Lab.
Ulrich, K.T., & Seering, W.P. (1992). Function sharing in mechanical design. In *Artificial Intelligence in Engineering Design* (Tong, C., & Sriram, D., Eds.), vol. 2, pp. 185–213. New York: Academic Press.
Woodbury, R., & Burrow, A. (2001). Design spaces: the forgotten artifact. *Proc. Third Int. Conf. Mathematics and Design* (Burry, M., Datta, S., Dawson, A., & Rollo, J., Eds.), pp. 56–62, Geelong, Victoria, Australia.
Woodbury, R., Datta, S., & Burrow, A. (2000*a*). Erasure in design space exploration. In *Artificial Intelligence in Design 2000*, pp. 521–544. Dordrecht: Kluwer.
Woodbury, R.F. (1993). Grammatical hermeneutics. *Architectural Science Review 36(2)*, 53–64.
Woodbury, R.F., & Chang, T.-W. (1995). Massing and enclosure design with SEED-Config. *ASCE Journal of Architectural Engineering 1(4)*, 170–178.
Woodbury, R.F., Burrow, A.L., Datta, S., & Chang, T.-W. (1999). Typed feature structures in design space exploration. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 13(4)*, 287–302.
Woodbury, R.F., Burrow, A.L., Drogemuller, R., & Datta, S. (2000*b*). Code checking by representation comparison. *Proc. Fifth Conf. Computer Aided Architectural Design Research in Asia (CAADRIA2000)*, May 18–19, 2000, pp. 235–244. National University of Singapore.
Woodbury, R.F., Radford, A.D., Taplin, P.N., & Coppins, S.A. (1992). Tartan worlds: a generative symbol grammar system. *ACADIA 92* (Noble, D. & Kensek, K., Eds.), pp. 211–220, Charleston, SC.
Woodbury, R.F., Shannon, S.J., & Sterk, T.D. (2001). What works in a design game? Supported by Student Reactions to Being Made to Play. *Proc. Sixth Conf. Computer Aided Architectural Design Research in Asia (CAADRIA '2001)*, April 2001, pp. 411–420. Sydney, Australia.
Wright, F.L. (1945). *An Autobiography*. New York: Hyperion Press.

Robert F. Woodbury holds a Bachelor of Architecture from Carleton University and a Master of Science and PhD from Carnegie Mellon University (CMU). He has served in the Architecture and Engineering Design Research Center at CMU; in the Architecture, Landscape Architecture, and Urban Design Department at Adelaide University; and in the School of Interactive Arts and Technology at Simon Fraser University. Dr. Woodbury's research is in computational design, design collaboration, and design learning. His current work in generative systems is on subsumption-based design space explorers, an alternative to the long-standing and dominant rule mechanism. He has over 100 technical publications.

Andrew L. Burrow is a Research Associate in the Spatial Information Architecture Lab at RMIT University. His research interests cluster around the following activities: ontology-based design collaboration; transdisciplinary design communication; applying the theory of partially ordered sets to information structures in design space exploration; analyzing representations for incremental updates to bring direct manipulation interfaces to sophisticated information systems; and exploring collaboration in tools that do not prescribe a single approach to metadata.