Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2011), *25*, 57–75.

doi:10.1017/S0890060410000478

Discovering implicit constraints in design

MADAN MOHAN DABBEERU AND AMITABHA MUKERJEE
Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India

(RECEIVED September 22, 2009; ACCEPTED August 23, 2010)

Abstract

Designers who are experts in a given design domain are well known to be able to Immediately focus on "good designs," suggesting that they may have learned additional constraints while exploring the design space based on some functional aspects. These constraints, which are often implicit, result in a redefinition of the design space, and may be crucial for discovering chunks or interrelations among the design variables. Here we propose a machine-learning approach for discovering such constraints in supervised design tasks. We develop models for specifying design function in situations where the design has a given structure or embodiment, in terms of a set of performance metrics that evaluate a given design. The functionally feasible regions, which are those parts of the design space that demonstrate high levels of performance, can now be learned using any general purpose function approximator. We demonstrate this process using examples from the design of simple locking mechanisms, and as in human experience, we show that the quality of the constraints learned improves with greater exposure in the design space. Next, we consider changing the embodiment and suggest that similar embodiments may have similar abstractions. To explore convergence, we also investigate the variability in time and error rates where the experiential patterns are significantly different. In the process, we also consider the situation where certain functionally feasible regions may encode lower dimensional manifolds and how this may relate to cognitive chunking.

Keywords: Chunking; Design Change; Design Space Shrinking; Functional Emergence; Implicit Constraints

1. INTRODUCTION

It is well known that a designer who is an expert in a particular design domain is "confident of immediately choosing a good [design] based on experience" (Gross, 1986) or that they "intuitively know that [their] interpretation of the problem and solution . . . is the correct one" (Lloyd & Scott, 1994). This rapid convergence to good solutions is a common hallmark of expertise across a large number of domains ranging from chess, medicine, computer programming, and bridge to physics (Ericsson & Lehmann, 1996). Cognitive models of expertise suggest that these constraints among the parameters of the task may result in a set of chunks, which provide a more compact description of the problem (Chase & Simon, 1973; Gobet et al., 2001). For example, experienced padlock designers may discover that to balance strength evenly, the U-bolt diameter must increase roughly in proportion with body size. Thus, these sets of parameters can be combined in a single chunk, reducing the dimensionality of the design space. Although this reduces search, it is more important that it also enables a shorter description of the design problem and a restructuring of the design representation (Campbell et al., 2003; Bor & Owen, 2007).

In a cognitive sense, from chess players to human designers, the chunking process is more often implicit than explicit. Expert chess players often do not see the poorer moves (Gobet & Wood, 1999), so that only the better options appear to be coded. Human designers in think-aloud sessions may use terms like "looks right" or they may refer to past experience merely by saying that it "worked before," although the design is obviously for a novel task (Ahmed et al., 2003). It has been suggested that these experiences may also guide the formation of design knowledge and are related to schema emergence (Oxman, 2002; Janssen, 2006). At a later stage, some of this knowledge may become explicit; but even in the early stages, it may help the designer make quick conjectures that she moves on to verify (Cross, 2004).

Constructing computational models for these cognitive processes remains an important challenge for computational design. In this paper, we suggest that these patterns may be discovered using machine-learning techniques focusing on the fact that "good" designs often lie along a limited range in the design space, which we call the functionally feasible re-

Reprint requests to: Madan Mohan Dabbeeru, University of Maryland, College Park, 18-F Crescent Road, Greenbelt, MD 20770, USA. E-mail: madan.dabbeeru@gmail.com

gion (FFR). As with chunking, the constraints underlying the FFR may be implicit; that is, the designer may find it difficult to articulate them or provide reasons for them (Schon, 1983).

Discovering and characterizing FFRs (the regions in the design space that correspond to good designs), is a critical first step in the discovery of chunks. This paper explores this process, and investigates how the patterns underlying the FFRs may be discovered, how these may be similar across similar design experiences, and how the process of learning these patterns may converge. After discovering the FFRs, it may be seen that these "good solutions" actually occupy a lower dimensional subspace, in which case they may be good candidates for chunks. In the language of machine learning, these chunks lie along a m-dimensional manifold that is a subspace of the original N-dimensional design space, where $m \ll N$; hence, any point in this m-dimensional space can be described using m parameters, as opposed to the original N.

Sometimes, when the performance criteria lie over a narrow band, the FFR may be easily seen to yield a chunk. For example, the padlock latch–bolt design fragment dealt with in Section 3.1, has a two-dimensional design space w, t. One may consider two performance metrics: strength of the lock π_{str} measured as a maximal force and the ease of insertion π_{ease} measured in terms of clearance. If the ease of insertion has a tight range of acceptability, the "good designs" for the latch–bolt fragment are seen to lie along a one-dimensional line (a manifold) with invariant $w - t$ (Fig. 1). This line can be discovered through design exploration (Fig. 1b), where the gray region represents the FFR generalized from a set of feasible design instances (pluses) and black is the infeasible region generalized from failed instances (open boxes). Here the FFR yields a lower dimensional manifold that is easy to discover, but more generally, it may be nonlinear or high dimensional and is a difficult problem on its own. However, the first step in the process is clearly to discover the pattern inherent in the good designs, similar to what Oxman has called a "perceptual act" operating on a visual memory of designs in his model of design conceptualization (Fig. 2). In this work, we consider the process of discovering such FFRs and their implications, and do not consider the manifold learning step required for discovering chunks per se.

The discovery of underlying patterns while executing designs repeatedly is of value not only because one can quickly find the good designs, but as an important cognitive discovery at the core of design expertise. Quite often similar patterns may hold in other design situations with similar characteristics. Sometimes the implicit pattern discovered may draw the designer's attention to those aspects of behavior, which may help formulate an explicit awareness of certain interrelations, what has been called "situated invention" (Suwa et al., 2000).

However, this process of discovering implicit patterns depends on the particular experimental history of the designer. Differences in evaluating functions (e.g., subjectivity, as in aesthetics) or differential explorations of the design space (experience in different classes of products) may lead to differences in learned patterns, which possibly constitutes one of primary factors behind differentiations in design style.

The pattern underlying good functional performance is usually discovered initially in well-defined problems spaces, what Janssen (2006) calls niche environments. Here we also limit ourselves to "familiar" classes of designs, similar to an apprenticeship situation. The set of design variables, the workings of the design, and ultimately a set of function evaluation metrics are available to the learning system. However, these evaluation functions are often complex and may require intermediate computations [e.g., configuration space (C-space) computations for mechanical devices as in the examples here]. Hence, identifying the subspace of good designs, the key task in functional emergence, remains a difficult problem, and this is what we focus on here.

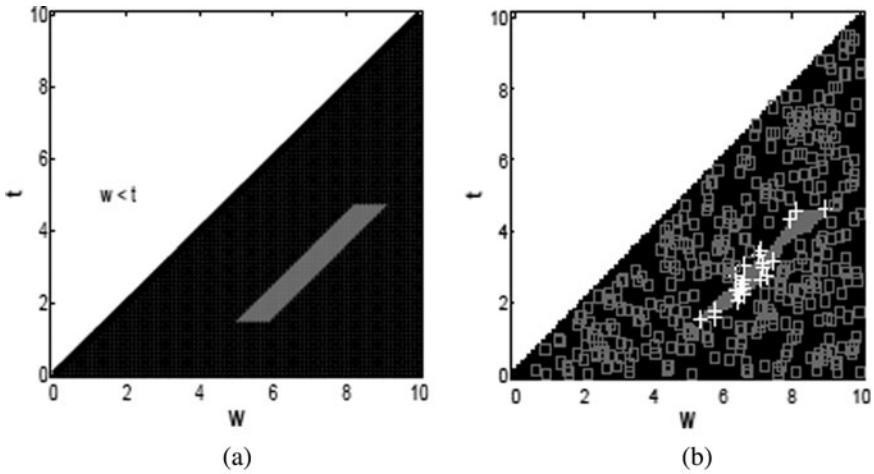

Fig. 1. Functionally feasible regions (FFRs) for a padlock design fragment. (a) 100 N $< \pi_{str} <$ 1000 N and 3.5 mm $< \pi_{ease} <$ 4.5 mm. (b) Learning with 400 data points. The acceptable performance criteria have tight bounds on the ease of insertion. The functionally feasible region is discovered as a thin elongated region in the design space. It may be viewed as a one-dimensional manifold in the two-dimensional design space (w, t). This may lead to the discovery of a chunk relating w and t, say, $w - t$.

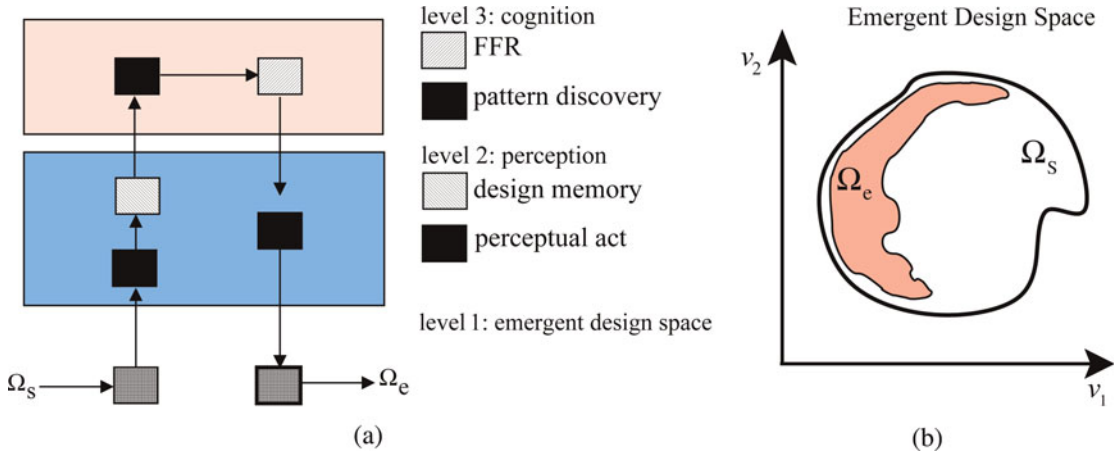

Fig. 2. Learning constraints through exploration. (a) Looking at designs one has evaluated is a recognition, resulting in an abstract pattern (based on Schon, 1983; Oxman, 2002). The first step in this abstraction process is learning the functionally feasible region. (b) Through design explorations, the initial design space Ω_s, bounded by a set of specification constraints f_s, shrinks to an emergent design space Ω_e, with the emergent functional constraint f_e. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

Another key question for the emergent patterns or FFRs is whether the constraints that are learned for one design generalize to other similar design domains. In order to investigate this question, we consider two locking mechanisms similar to our main padlock example. We first consider a change in a solution mechanism (embodiment), from a translational lock (padlock) to a rotational lock (slotted wheel), and show that for the latch–groove design fragment, the basic pattern of FFRs remains similar (Section 4.1). Next, we consider a different design space, where an extra degree of freedom (DOF) is added to the design (Section 4.2). We also find that for small values of this new parameter, the behavior remains similar to the earlier patterns. These initial explorations suggest that there is a possibility that the patterns underlying these good designs may apply to families of functional similarity, as opposed to the single embodiments on which they are learned. Although these similarities are apparent, they indicate the power of methods that can be deployed based on FFRs. However, the construction of computational formalisms for discovering similarities is beyond the scope of this work.

Because the learning of patterns is based on randomly drawn instances from the design space, a question may arise about the stability of the learned pattern. Does the learned pattern always converge to the same pattern? This question is explored empirically in Section 5. Next, we correlate our search for patterns in the function space to the models of cognitive design suggested by Schon, Oxman, and others.

1.1. Design space exploration, sketching, and emergence

The human designer often uses sketching as the mechanism of choice for his preliminary exploration of the design space, which results in patterns of functional effectiveness. During this process, various design choices are quickly evaluated for functional feasibility, often using additional visual constructs that operate on the sketches themselves (Goel, 1995). An early insight into this process by Schon (1983) suggests that the designer shapes a situation, which then talks back. By observing and evaluating this situation, often through an external representation, the designer forms new appreciations that further guide his decisions. This process has been investigated qualitatively by Oxman (2002), who equates perceptual exploration with recognition or reformulation of the cognitive models relevant to the design (Fig. 2). This is a key insight into the design process, but it is expressed rather imprecisely. It is our goal in this work to suggest a computable model for the first phases of this insight-generation process.

Recently, Janssen (2006) proposed that computer-aided design (CAD) models may be able to learn designer preconceptions through evolutionary design ideas. Our model goes further and proposes a general machine learning model that actually generates the first preconceptions: the functionally emergent constraints on the design space. However, the terms emergent and function have been used in many senses in the design literature, and it is necessary to clarify the sense in which we are using these next.

1.1.1. Three types of emergence

The term emergence has various uses in the design literature, usually involving some form of the exploration–abstraction process. We observe (at least) the three following usages:

1. *Creative emergence:* Adding new dimensions of exploration, or expanding the design space. This is the result of observations of certain possible configurations that were not present in the original specification. A typical example from Soufi and Edmonds (1996) is how a triangular construction may be suggested while exploring a space defined by the intersection of squares (Fig. 3). Representing the new shapes require additional design variables to be introduced (Cagan & Agogino, 1991).

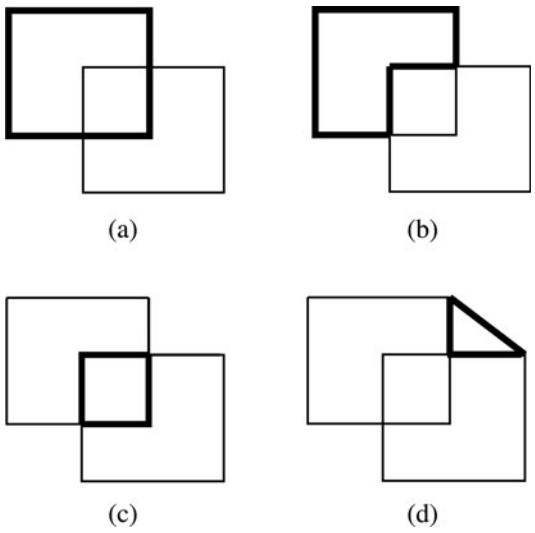

Fig. 3. Creative emergence (novel shapes). (a–c) The initial design space is based on Boolean operations on two squares. However, while interacting with these shapes, some other designs such as (d) may suggest themselves (Soufi & Edmonds, 1996), which are representions that would require additional variables in the design space. Thus, creative emergence expands the design space.

2. *Functional emergence:* A shrinking or restructuring of the design space based on a characterization of the space of good designs may be an emergent pattern (Fig. 2b) or an enabling prejudice or preconception (Janssen, 2006). The functional constraints discovered may be implicit ("similar stress worked for other turbine blades"; Ahmed et al., 2003) or explicit ("A ticket office should be close to an entrance"; Suwa & Tversky, 1997). Where the functional constraint reflects a significant dimensionality reduction, the resulting abstraction may get codified as chunks.
3. *Symbol emergence:* Some patterns arise repeatedly and come into conscious awareness (are *reified*). Such patterns may eventually acquire a label in linguistic convention, in which case they become a *symbol* (Dabbeeru & Mukerjee, 2010).

In this paper, we limit ourselves to the discovery of the primary form of functional emergence, the FFRs. Functional emergence in this view involves the designer realizing that most designs in the initial design specs are poor designs and that the designs with superior performance lie in a restricted region of this space. Thus, functional emergence results in shrinking the design space to that region where high-performing solutions are seen to lie. Note that our investigation operates in an apprenticeship mode, and we do not consider changes in the function space (i.e., the performance metrics do not change) and we also do not enlarge the given design specs. Thus, the aim is less ambitious than attempts such as Poon and Maher (1996), who consider the simultaneous evolution of both design space and the function space, requiring a mechanism for abstract diverse functional specifications into a compact genotype, and measures for evaluating the suitability of the function metrics (selecting a problem space). Nor do we attempt to discover similarities with known designs (cases) in a design database (e.g., Yaner & Goel, 2008), in which one assumes that certain symbolic mappings are already known that enable segmenting vector diagrams of a design into constituent parts and determining the results of compositions among these parts. As a simulation of a trainee being told what to do by a master designer, our goals are more modest; we merely wish to attempt to model the space of well-performing designs, which may enable us to learn some salient aspects of the design space such as how different variables may have latent interrelations in order to arrive at good designs. Our work proceeds in two steps. During the initial learning process, it is perhaps appropriate to consider familiar design spaces, where the function space is stable and well understood. In the second phase (Section 4) we consider how these patterns are initially learned in a fixed setting and how they may generalize to other, related task domains. The objective here is to discover how good functional performance may impose additional (emergent) constraints on the design space. Before we proceed, however, it is necessary for us to formally specify what we mean by "function."

1.2. Function, performance, and embodiment

The term function is used in many senses by designers (Gero, 1990; Wolter & Chandrasekaran, 1991; Faltings, 1992; Umeda & Tomiyama, 1997; Suwa et al., 2000). It is clearly related to the *user intent*, often inchoate, and the designer's initial task is to understand its contours. Large problems may be decomposed into some subtasks. At an early stage, several solution mechanisms may be proposed, one of which is adopted, what we call its embodiment. Once this is available, the user may provide additional constraints on what it is that is desired. Design is an ill-posed problem because the criteria are not defined at the outset; these depend on the prejudices brought to bear by the designer and on the solutions as they are explored and are thus a part of the dynamics of design. As the design progresses further, additional constraints become clear. Thus, function is a role of the context in the design process, what has been referred to as "situatedness" (Suwa et al., 2000; Gero & Kannengiesser, 2004): at each stage of design, the meaning of function may be different.

We must distinguish two senses that are often conflated in discussions of computable models of function. The first views a set of functional needs as independent of embodiment [e.g., "provide illumination," what Gero (1990) calls "function"]. The second considers how these functions are evaluated or compared [e.g., "the light intensity variable in user's room must be above a certain lumens value" (Chandrasekaran, 2005), similar to what Gero calls expected behavior or B_e]. In order to distinguish these more clearly, we use the term *performative behaviors* for the former, which is the set of user needs. These are the subset of any object's behavior that is of

interest to the user (e.g., torque, illumination), and they are general characteristics without considering comparison or evaluation. When we say that two design classes have the same function, we are interpreting function as performative behavior.

The second usage of function is evaluative; given a design instance, this helps determine if it is satisfactory. This interpretation is captured in what we call the *performance metric*, whose construction is dependent on the type of embodiment. Every performative behavior required of the design, is related to a performative metric π_i, which may be computed for any design in the particular embodiment class. Different user needs may result in different levels of satisfactory performance, defined as a Boolean function on the set of performance metrics, $g(\boldsymbol{\pi_1}, \boldsymbol{\pi_2}, \ldots)$, which must be true for the design to be satisfactory. This acceptability condition (e.g., a range of acceptable strengths/light lumens) may also be negotiable, but here in this apprenticeship situation, we take it as given. We note that, although we use the term metric and in the examples below this function is always quantifiable, the process we propose would also work if one had a relative measure or comparison that can order any pair of design instances.

Let us now also define what it means for the design problem to be familiar or well understood. First, we assume that we know the solution mechanism to be used, which we call its *embodiment*. Second, all designs in this familiar class have the same set of independent design variables, constituting the design variable vector $\boldsymbol{\upsilon}$. Third, and most significantly, the designs in this class have the same set of performance metrics $\pi_i(\boldsymbol{\upsilon})$. This is because designs in the embodiment class, or the embodiment part family (EPF), instantiate the function in a similar manner. For example, for the domain of locking devices, the padlock embodiment uses a latch sliding into a slot on a U-bolt. For all padlock designs, the performance metrics (e.g., evaluation of strength) would involve the same function that is ultimately linked to its independent design variables. As we will see in the examples below, this may involve some complex operations, for example, the construction of C-spaces for mechanical assemblies, which may be possible only for the class of designs that share the familiar embodiment. To take another example, if the problem is to provide light and air in an architectural space, the embodiment chosen may be a window, and one may define the amount of light or air in terms of the dimensions of the window. Although the performance metrics are same, clearly different designs may have differing acceptability criteria.

Next we define the notion of "design class" based on the above notion of function.

1.3. Design process

In its broadest generality, design deals with all possible artifacts (Fig. 4, top level), at which point, "function" is at its most vague. Next, we may consider designs that use similar principles to serve similar functional needs, which constitute the *phenomenological design domain* (PDD), for example, arranging for light and air in an architectural space or restricting access to some interior space. There are many ways in which these design goals may be specified and met. Within a PDD, the designs that are evaluated in terms of the same performative behaviors are what we call the *functional part family* (FPF). Thus, single-panel windows, multiple-panel windows, possibly even rolling shutters, if evaluated using similar performative behaviors, may belong to the same FPF. However, some other structures, like fixed windows (which do not have "letting in air" as a performative behavior) would constitute a different FPF. Within a specific functional class FPF_1 the designs that meet a set of functions using the same physical structures, so that the design variables map to the performance metrics in the same way, constitute the EPF (embodiment class).

The EPF is the niche class that we consider throughout this paper. We also demonstrate how some of the patterns learned on one EPF may be similar to those in other EPFs in the same functional class (Section 4). However, we present a computational model for obtaining the FFRs only within an EPF, and do not provide a process for the more difficult problem of generalizing across EPFs.

For our example of locking devices, the phenomenological level is based on some physical mechanism for restricting access. Of these, some may share the same set of performative behaviors (FPF). In Figure 4, FPF_1 and FPF_2 both involve keys moving a latch in and out, except that in the latter the object that will be constrained by the latch is external to the design object. For FPF_1 (padlocks, rotating barrel locks, etc.), shared performance metrics may involve the maximum force it can resist (strength), weight, ease of use, and so forth. In contrast, class FPF_2, which is intended to be fixed to something like a door frame, the performative behavior may consider volume instead of (or in addition to) weight; thus, the set of performative behaviors is different. The class of padlocks, which share the same design structures, constitute an EPF within FPF_1.

Each EPF is associated with a design space Ω, characterized by a n-tuple design vector $\boldsymbol{\upsilon} = (x_1, x_2, \ldots, x_n) \in \Omega$, where x_i are the independent design variables or driving variables for the design. Any other variables needed for specifying the final structure (dependent variables) are defined in terms of these driving variables. At lower levels in the hierarchy there are fewer DOFs (i.e., number of design variables go down), but the function becomes more crisply specified. At the bottom of the hierarchy are specific design instances, each of which is completely specified (0 DOF).

In this paper, we are given an EPF and will try to discover patterns of functional feasibility in its design space. The range of designs is initially given through a Boolean design specification function $f_s(\boldsymbol{\upsilon})$, and in Section 2 we will see how tighter constraints on this emerge after evaluating many designs for their performance. Section 3 demonstrates this approach as applied to our canonical example, the padlock. To investigate how learning one FFR for a specific embodiment may help us understand other similar designs, we consider two other designs from the same FPF in Section 4. Finally, we consider the question of convergence for the FFR in Section 5.

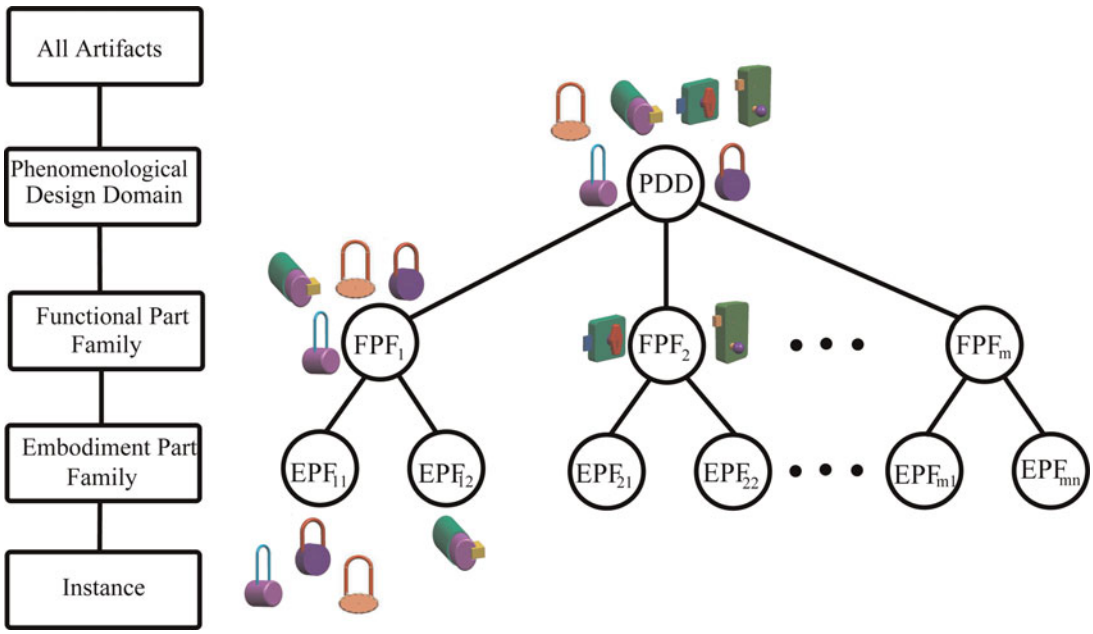

Fig. 4. Hierarchy in design. Starting with all possible artifacts, the designs that share some principles of operation constitute the phenomenological design domain (PDD). Within these, those that have the same set of shared user needs constitute a functional part family (FPF). Among these, designs with the same embodiment constitute the embodiment part family (EPF). [A color version of this figure can be viewed online at journals.cambridge.org/aie]

2. IMPLICIT CONSTRAINT DISCOVERY

Since the origins of CAD, for example, in the very early ideas of Ivan Sutherland (1963), there has been considerable emphasis on the process of discovering constraints in the design space, often through computational imagery that simulates the process of sketching (Gross et al., 1988). The idea of constraints on design parameters, originating in the work of Gossard (Lin et al., 1981), revolutionized CAD by introducing the notion of parametric modeling.

However, not much computational work has focused on the task of discovering implicit constraints by exploring function in the design space. This is partly because of the difficulty in understanding function. In this work, we only consider design problems that are well understood, so that function is definable and may be expressed in quantifiable terms. We operate at the embodiment level and take functions to be evaluatable in terms of performance metrics. For our example class of locking devices (Fig. 5), possible performative behaviors may be strength, weight, cost, and robustness against jamming. For each such behavior of interest, we assume that a performance metric is available, and we show how exploration in this multifunction space results in identification of feasible subspaces (FFRs) in the design space.

2.1. Design space shrinking

Any general purpose function approximation algorithm can be used to learn the implicit patterns that constrain the space of "good designs." Here we use multlayer perceptrons as our vehicle for learning the FFRs. In practical systems, the resulting constraints may be of direct value to novice designers, especially in situations involving many choices (Gross, 1986). This would constitute an important step if we are to enable computers to have greater access to the range of creative improvements possible for the human designer (Janssen, 2006). In a computational sense, the process of design, involving computationally expensive analysis of aspects such as strength, flow, or motion, can then be limited to a much smaller range. It is more important, however, that it may be possible to discover abstractions in the design space leading to compact representations.

Let Ω_s be the initial design space defined by the set of design specifications $f_s(\boldsymbol{\upsilon})$. Let $f_e(\boldsymbol{\upsilon})$ be an emergent constraint that is learned through exploration, based on a function characterized by a set of performance metrics $\pi_i(\boldsymbol{\upsilon})$. Whenever f_e is true, f_s must also be true; otherwise, these designs $\boldsymbol{\upsilon}$ would not have been explored at all. Now we provide the following simple result, which motivates this work.

DEFINITION 1 (specification constraints): The specification constraints $f_s(\ )$ constrain the initial design space Ω_s, which is defined as $\Omega_s = \{\boldsymbol{\upsilon} | f_s(\boldsymbol{\upsilon})\}$. ■

DEFINITION 2 (functional constraints): Acceptable levels of performance are defined as a Boolean function of the performance metrics $g(\pi_{\boldsymbol{\upsilon}})$, for example, $g(\pi_i(\boldsymbol{\upsilon})) = \bigwedge_i(\pi_i(\boldsymbol{\upsilon}) - \min_i) > 0$. This *ideal functional constraint* $g(\ )$ is modeled by the *emergent functional constraint* $f_e(\boldsymbol{\upsilon}) \approx g(\pi_i(\boldsymbol{\upsilon}))$, which is an approximation to $g(\ )$ learned over a set of experienced design instances. ■

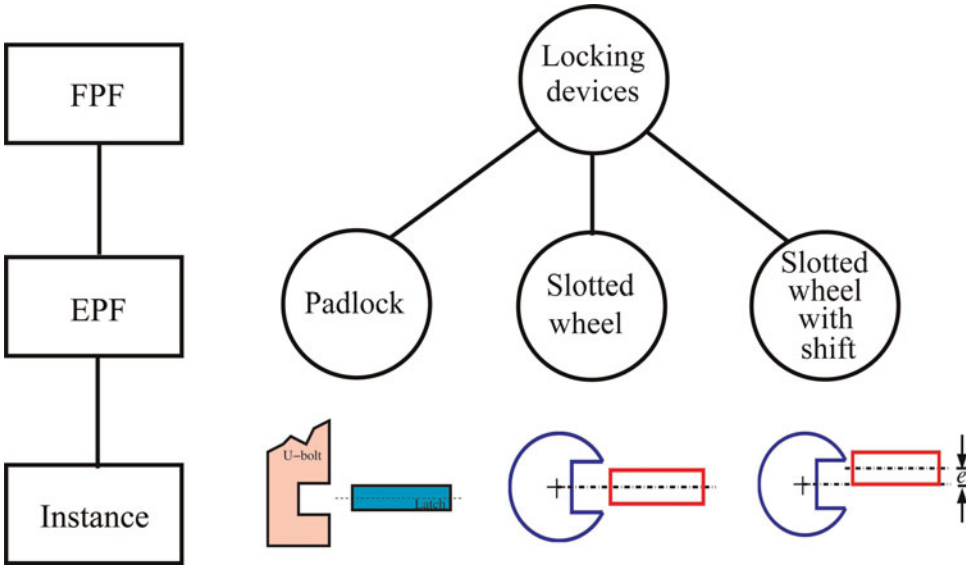

Fig. 5. Design change. Under the same functional part family (FPF), we consider design change at two levels: an embodiment change, padlock to slotted wheel mechanism, and a design space expansion, adding an additional variable (e) to the slotted wheel mechanism. Our objective is to see if the some aspects of the knowledge captured in the functionally feasible regions are similar in these new embodiments. EPF, embodiment part family. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

The functional constraint $g(\ )$ is trivial if it holds for all $\mathbf{v} \in \Omega_s$. For any nontrivial functional constraint $g(\pi_i(\mathbf{v}))$, there exist some design instances $\mathbf{v} \in \Omega_s$, such that $\neg g(\pi_i(\mathbf{v}))$.

As experience increases (as more design instances are explored), $f_e(\ )$ becomes increasingly precise in its approximation to $g(\ )$. In Section 5 we consider some measures for the accuracy of this approximation.

DEFINITION 3 (FFRs): The ideal FFR is the constrained design space $\Omega_i = \{\mathbf{v}|g(\pi_i(\mathbf{v}))\}$. This *ideal FFR* Ω_i is approximated by the *emergent FFR* $\Omega_e = \{\mathbf{v}|f_e(\mathbf{v})\}$. ■

THEOREM 1 (*design space shrinking*): *A nontrivial functional constraint* $g(\pi_i(\mathbf{v}))$ *narrows the design space from* Ω_s *to* Ω_i, *where* $\Omega_i \subset \Omega_s$. ■

Proof: This follows from $(\exists \mathbf{v})\{f_s(\mathbf{v}) \wedge \neg f_e(\mathbf{v})\}$. ■

Because Ω_e is an approximation of Ω_i, it is also expected to be narrower than Ω_s.

If one can obtain a measure for the cardinality of the design space, then one may also define the effectiveness of an emergent constraint in terms of the degree of shrinking $\|\Omega_e\|/\|\Omega_s\|$. If this shrinking factor is observed to be high, the designer's conscious attention is drawn to it, and she may explore the reasons for this behavior. This is possibly one aspect of "interestingness" of the emergent relation.

Note that the set of performance metrics π_i constitutes a mapping from the design space to the performative behaviors. The acceptable set of $\mathbf{v}$, determined by $g(\pi_i(\mathbf{v}))$ is specified based on user preferences, as encountered in the past (these are given to us in the apprenticeship situation). The functions $\pi_i(\mathbf{v})$ may be quite complex, for example, in products involving mechanical assemblies, the computation may involve relative motion of the subparts; these behaviors may be captured using C-space. The learned emergent constraint $f_e(\mathbf{v})$ provides a more easily computed approximation.

We now demonstrate the process of computing the emergent design space (or the FFR) for several classes of locking devices.

2.2. EPF: Locking devices

Let us consider the U-bolt and latch design fragment of a padlock (Fig. 6). The design variables (i.e., the set of independent variables that define a design instance) constitute the design vector $\mathbf{v}$, $\mathbf{v} = \{w, t\}$. The dependent variables are the supporting length (l_1) and the width of the latch (b). These are determined from the independent variables w, t as $l_1 = 6$ mm and $b = 8$ mm space of two-dimensional design vectors, as constrained by the design specifications, called specification constraints $f_s(\mathbf{v})$, is the initial design space $\Omega_s = \{\mathbf{v}|f_s(\mathbf{v})\}$.

Here the set of design variables, these constraints, and the other design parameters (b, l_1), are given to the system. The design space is bounded by defining a set of specification constraints $f_s(\mathbf{v})$ that must be true (here f_s is taken to be Boolean; one may consider these to be defined based on algebraic functions, say, of the form $\pi(\ ) >$ min). Given this initial specification of the design space, and the performative metrics, our task is to discover the emergent patterns in the design space that determine good functional performance.

2.2.1. C-space

In designs such as padlocks that involve mechanical motion, relating functions to structure often involves abstractions on the relative motion of the subparts. C-space is a well-known

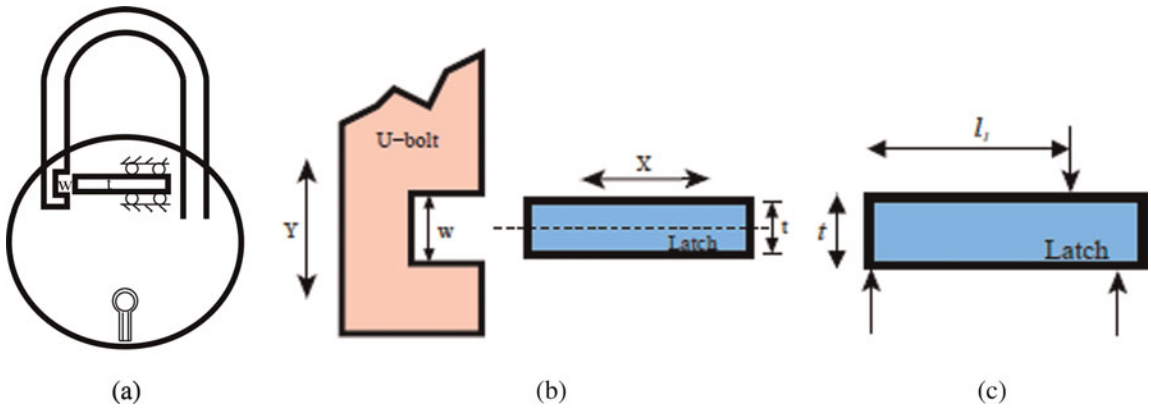

Fig. 6. The latch–bolt design fragment. (a, b) Padlock with U-bolt–latch mechanism. (c) Design variables exerting forces on latch: w, t; design parameter variables: l_1, supporting length; b, breadth of latch (into the page). [A color version of this figure can be viewed online at journals.cambridge.org/aie]

approach for modeling such abstractions (Faltings, 1992). However, computing the C-space for general motions remains an intractable problem (Ji & Xiao, 2001). Further, given a C-space, obtaining successful abstractions on it; that is, segmenting the free space into behaviorally significant regions; for example, using topologically different contact types (Mukerjee & Bhatia, 1995), remains a considerable challenge. Here, we assume that similar designs have been explored already, so that some understanding of the C-space and its abstractions are available for the EPF under consideration, such as the three locking device embodiments considered below. Thus, a performance metric can be computed for any design instance in each EPF.

3. DISCOVERING PATTERNS OF FUNCTIONAL FEASIBILITY

We adopt a supervised learning approach toward discovering FFRs in the design space. The training set is the set of designs explored in the design space: each visited design instance is evaluated with the given performance metrics. Given a set of acceptability criterion, design instances are categorized as feasible or infeasible, and the learning system attempts to construct a hypothesis for these accept/fail evaluations. We have tested the system with several well-known pattern learning algorithms (Bishop, 2006): multilayer perceptron, radial basis functions, and support vector machines. Within the variability of the process, each of these converges to similar output, especially where the training set is large. Here we present the results from learning with the simplest multilayer perceptron structure (essentially a steepest gradient learner), where the input will be the design vector $\boldsymbol{\upsilon}$, the oracle is the acceptability condition $g(\pi_i)$, and the output is whether the design instance is feasible. We use a single hidden layer with 50 neurons with a tan–sigmoid transfer function. The weights and bias values in the backpropagation training are updated according to the Levenberg–Marquardt optimization technique. The commercial package Matlab has been used for this.

This process differs from the human designer's function emergence process in several significant aspects. For humans, the process of arriving at the constraint operates at a more abstract level than the design instance; each sketch, or imaged design, is ambiguous and applies to a region in the design space rather than a single design (Oxman, 2002). Thus, in humans this search rules out large subclasses of design at each step; in the machine learning process adopted here, the machine can only evaluate designs at the design instance level, so many more evaluation sequences are required. However, it is posited that the human is able to operate at a more abstract level because she already has many "preconceptions," possibly from exploring similar domains. It is these very preconceptions that the machine is trying to learn at this point.

We demonstrate this process next with the example of the padlock latch–bolt design fragment.

3.1. Padlock latch–bolt fragment example: Explicit constraints

For discovering new functional constraints, we consider two performative behaviors: ease of insertion and strength. We consider how performance metrics for these may be specified for several embodiments for locking devices, which are all in the same FPF because they share the same set of performative behaviors.

Considering the latch–bolt fragment of a padlock, we observe that ease of insertion will decrease as the clearance $w - t$ is reduced. Hence, we may define the performance metric for ease of insertion (π_{ease}) as the clearance $w - t$. For the region of the design space where $w < t$, the latch cannot enter into the slot, and the design is infeasible. Further, very low values of $w - t$ may also make it hard to insert. Acceptability may then be defined as a minimum acceptable level of this performance metric.

The strength of the lock depends on a number of factors, such as the tensile strength of the U-bolt, the bending strength of the latch, the groove on the U-bolt, the support for the latch

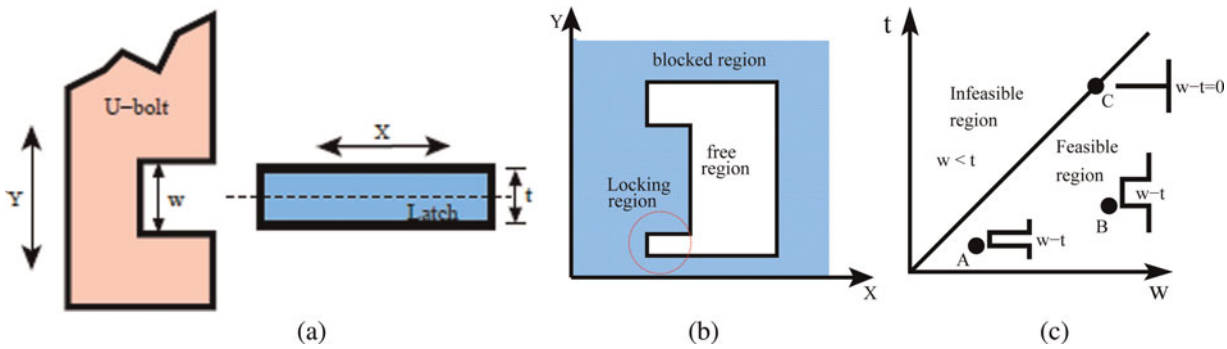

Fig. 7. The padlock example. (a) U-bolt moving vertically and latch horizontally and (b) the configuration space (X, Y) for the latch–bolt design fragment; the locking region is to the bottom left. (c) A, B, and C are three design instances in the w, t design space; their C-space locking regions are shown. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

inside the lock, and so forth. Here we consider the latch as supporting beam with roller support as shown in Figure 7a. When the lock is hammered, an impact force is applied near the left end of the latch shown as an upward arrow in Figure 7c. Although the effect of this impact loading is more difficult to model, a reasonable simplifying assumption is that a lock that is strong in normal loading would also be strong in impact loading. Thus, we seek to compute the maximum force F, by which the latch can withstand. The maximal bending $\sigma = 6Fl_1/bt^2$, and setting this equal to the yield strength gives us the maximal sustainable F:

$$\pi_{\text{str}} = \sigma_Y bt^2/6l_1,$$

where σ_Y is the yield strength of the material (incorporating a suitable factor of safety, etc.) and l_1 and b are as defined above.

3.1.1. Discovery of functional feasible regions for padlock

By applying different standards of acceptability for the performance metrics to different instantiations in the design space, one can obtain different bounds to the FFRs. The performance metric for ease of insertion is negative for the region above the $w = t$ line, and this region is permanently infeasible. For increasing minimal acceptability levels of π_{ease}, the feasibility region boundary shifts more to the right. Similarly, for any given level of π_{str}, the material and other dimensions remaining the same, the strength increases proportionally to t^2. Increasing acceptability levels for the combination of these two performance metrics would result in the t-boundary shifting upward and the w-boundary moving to the right. Figure 8 shows the ideal FFRs in the padlock design space for single or combined performance metrics.

The ideal FFR is the region defined by the acceptability condition $g(\,)$ defined on the performance metrics. Although the functions given here are easy to compute, design evaluation in practices is often a computationally expensive task. In order to abstract patterns on these, it is therefore necessary to evolve a separate abstraction mechanism, that trains on the set of designs already explored (the design memory), to learn a classifier that matches the underlying, ideal FFR as best as it can. These learned FFRs, for differing levels of design exposure, are shown in Figure 9, corresponding to the ideal FFRs of Figure 8b and 8c. With very few instances, the system has

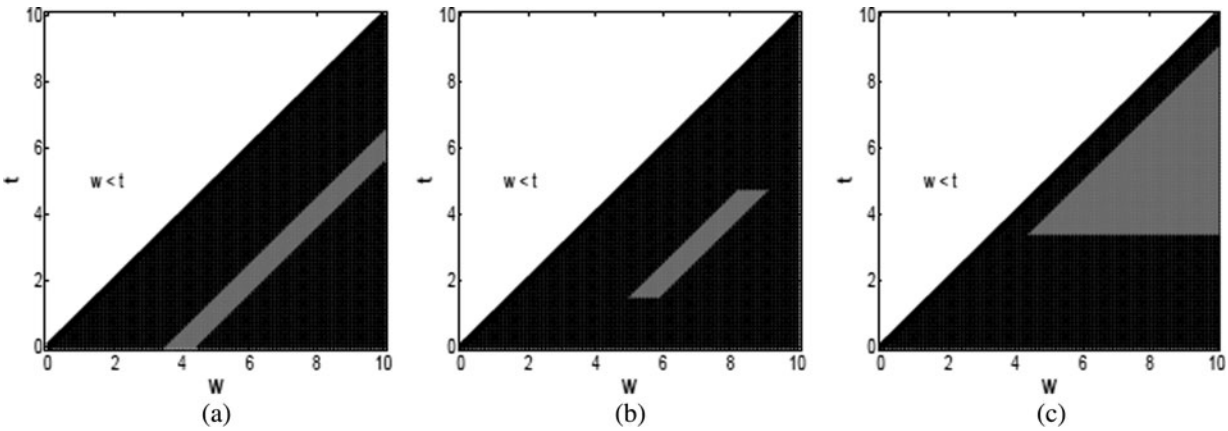

Fig. 8. Ideal functionally feasible regions (FFRs) for latch–bolt fragment. (a) Ease of insertion metric π_{ease} along a narrow range of 3.5 mm $< \pi_{\text{ease}} <$ 4.5 mm. (b) Combining both π_{ease} and π_{str} metrics at 3.5 mm $< \pi_{\text{ease}} >$ 4.5 mm and 100 N $< \pi_{\text{str}} <$ 1000 N. (c) The only lower bounds for π_{ease} and π_{str} at $\pi_{\text{str}} >$ 500 N and $\pi_{\text{ease}} >$ 1 mm.

very limited experience and the feasible regions learned at this stage are ill defined. The learned pattern becomes clear with more training points. In addition, we find that the process takes longer to converge for narrow bands (as in Fig. 9c).

For the padlock, the evaluation computation involves two simple closed-form equations; clearly, there are no limits to the complexity of the performance metrics. For many problems, elaborate iterative computations such as finite element method or finite difference computations over a mesh may be needed. However, designers in such cases often invoke "similar" tasks they have explored earlier, for example, "a similar stress worked for other turbine blades" (Ahmed et al., 2003). Thus, at least for some situations, there appears to be a possibility of applying some of the learned abstractions to different but related embodiments. In the following section we consider the *slotted wheel mechanism* and a variant with a vertical shift in the latch axis and explore if there may be some pattern similarity in these cases. However, defining a computational procedure for identifying such similarities is beyond the scope of the present work; we merely indicate the possibility of transferring such mechanisms.

4. DESIGN CHANGE

In the previous section we have discovered some function-based constraints on the design space in terms of the FFRs. This was done for a specific embodiment (the latch–bolt of padlocks), based on strength and ease of insertion as the performative metrics. Next, we ask if it is possible that these patterns, learned for one embodiment, may inform our perception of such emergent constraints in related design problems. We explore this question in the context of two other locking devices that have the same performative behaviors, that is, they are in the same FPF. For this, we consider two types of design change: embodiment change (Section 4.1) and design space expansion (Section 4.2). In both situations, a latch is inserted into a slot on a rotating barrel as opposed to a translating U-bolt.

4.1. Design change 1: Slotted wheel mechanism

As a first step we consider a slotted wheel mechanism, with a latch that enters a slot on a rotating barrel, thus preventing fur-

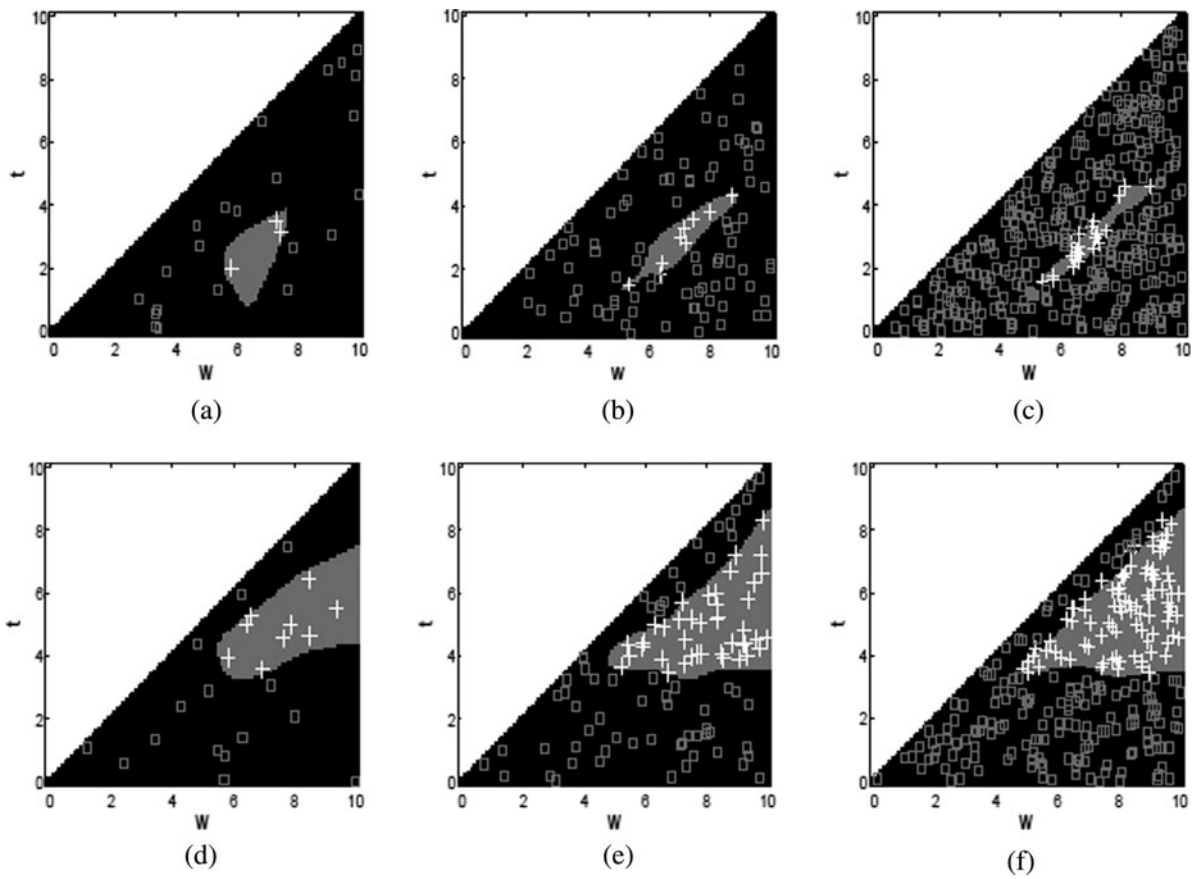

Fig. 9. Learning functionally feasible regions (FFRs): dependence design experience. Implicit constraints on "good designs" are learned for (a) 25, (b) 100, and (c) 1000 design instances under the functional specifications 4.5 mm $> \pi_{ease} >$ 3.5 mm and 1000 N $> \pi_{str} >$ 100 N (Fig. 8b). The decision surface learned after sufficient trials matches the ideal FFR very well, but convergence is slower for more complex patterns. Similarly, other constraints are learned for (d) 25, (e) 100, and (f) 400 design instances for acceptability: $\pi_{str} >$ 500 N and $\pi_{ease} >$ 1 mm (ideal FFR, Fig. 8c).

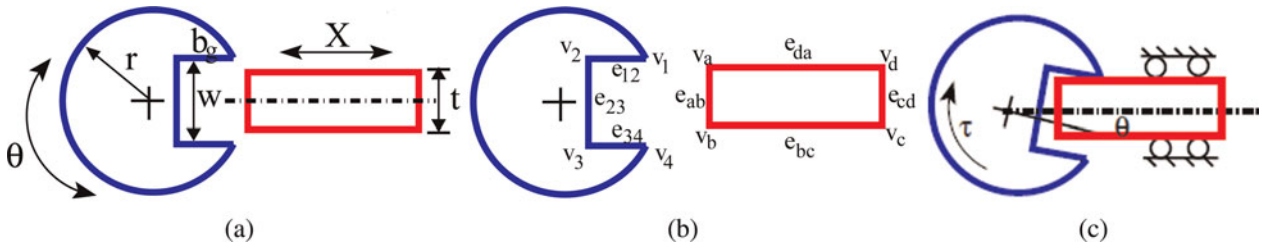

Fig. 10. The slotted wheel mechanism. (a) Design variables: degrees of freedom θ, radius of the slotted wheel r, depth of the groove b_g, width w, horizontal movement X, and thickness of latch t. (b) Contact states can be described with sets of vertices and edges of the contacting bodies. (c) The contact state here is the contact between the vertex v_1 of the wheel and the edge of the latch e_{da}. The strength π_{str} is derived from the maximum torque τ that it can withstand. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

ther rotation. The design variables for this mechanism are similar to the latch–bolt: slot width w and latch thickness t. The other design parameters are the radius of the slotted wheel (r) and the depth of the groove (b_g). The performative behaviors remain the same: strength and ease of insertion. However, the kinematics is quite different (Fig. 10), and it belongs to a different EPF Figure 4. The latch moves horizontally (X) as in the padlock but the slotted wheel rotates (DOF θ). These two motion variables define the C-space for this embodiment. In order to estimate the performance metrics it is necessary to understand the motion space θ, X of these interacting objects. For this we compute the C-space of the locking region (Fig. 11b) based on the contact state graph as shown in Figure 11a. Here each node represents a contact state between geometric elements (Fig. 10b). For example, CS02 involves contacts between v_4, e_{bc} and e_{23}, v_b. A contact state constrains the possible states of the contact pair, removing 1 or 2 DOF. Thus, each CS is represented as a line (–1 DOF) or point (–2 DOF) in the θ, X C-space (Fig. 11b). Note that, as in the padlock, the locking region in the C-space is an indentation into the obstacle region, except that the deep end of the indentation is flat for the padlock (Fig. 7b), and is a shallow groove in the slotted wheel (Fig. 11b).

The strength performance metric for the slotted wheel is derived from the maximal torque it can withstand, as opposed to maximal force. The maximum torque τ the latch can support is given as as $\pi_{str} = Fr$ (Fig. 10c), where F is the maximal force on the latch and r its moment arm. The maximal sustainable F occurs when the bending stress $6Fl_1/bt^2$ equals the yield strength π_{str}.

The ease of insertion is again determined by the clearance, $\pi_{ease} = w - t$. Based on these two performance metrics, the ideal FFRs are shown in Figure 12.

Given different ranges of acceptability for the performance metrics, we again obtain different FFRs. We observe that the shape of the FFRs is similar to that for the padlock for similar combinations of the performative behaviors, although the embodiment was quite different; compare Figure 12a to Figure 8c for π_{str} and Figure 12b to Figure 8a. The similarities may be because the C-spaces of the two mechanisms remain similar, although the embodiments are quite different.

Next, we investigate a second kind of design change. A new dimension of variability in the design is introduced, resulting in an additional design variable in the new embodiment. If this new mechanism shares the same performative behaviors, it will come under the same FPF as the padlock and the slotted wheel. This is considered in the next section.

4.2. Design change 2: Slotted wheel with vertical shift

Consider that the latch of the slotted wheel has an additional DOF e representing an upward shift in the latch axis

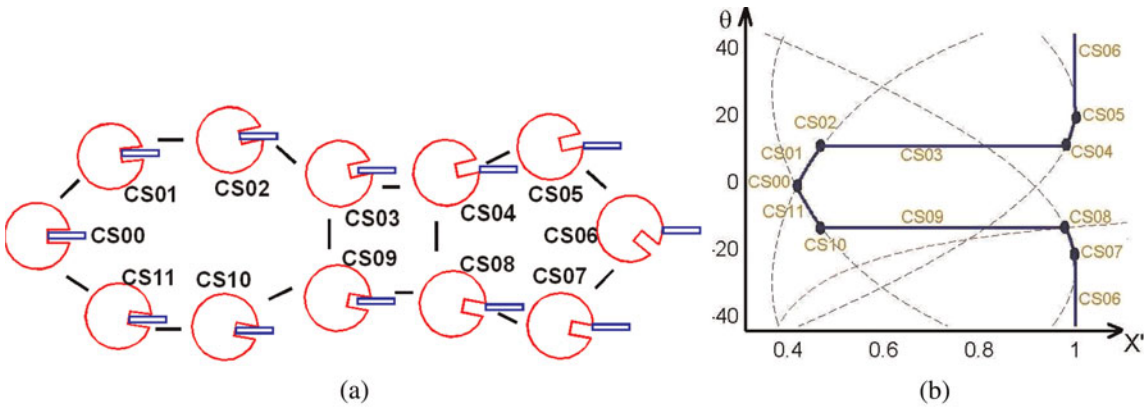

Fig. 11. (a) The contact state graph for computing (b) the configuration space. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

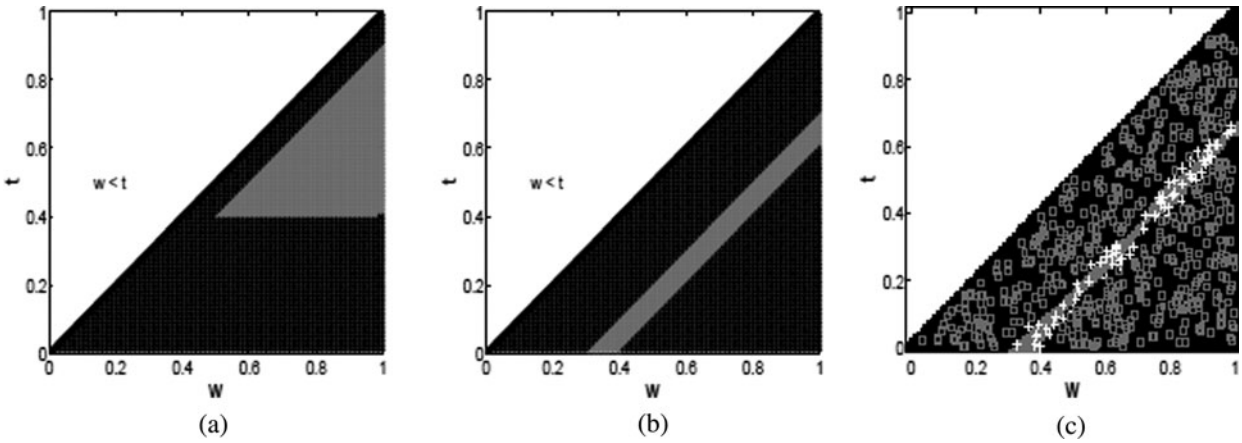

Fig. 12. Ideal functionally feasible regions (FFRs) for the slotted wheel with various functional constraints: (a) $\pi_{str} > 6$ N m and $\pi_{ease} > 0.1$ mm, (b) $\pi_{ease} > 0.3$ mm and $\pi_{ease} > 0.4$ mm, and (c) learning with 600 instances.

(Fig. 13a). This results in a design space w, t, e for this altered slotted wheel mechanism. Now the translation axis of the latch is above the slotted wheel center. This may arise as a result of some constraint in the installation, or as a result of manufacturing inaccuracies. This shift will then affect the quality of the penetration, and hence the strength. However, the performative behaviors considered remain strength and ease of locking, so that this constitutes a different EPF in the same FPF.

With the addition of vertical shift e in the latch axis, the motion behavior changes considerably. In the contact state graph (Fig. 11a), a new contact state CS04a $= \langle e_{12}, v_a \rangle$ is observed (Fig. 13b); this causes the mechanism to experience new set of behaviors at certain e values. The corresponding C-space is shown in Figure 14 with contact state constraints (for a detailed analysis, see Dabbeeru & Mukerjee, 2008). Now we consider the mechanism for a fixed $e = 0.3$ for computing the performative behaviors at a contact state CS04a.

As before, we consider the same performative behaviors π_{str} and π_{ease}. For computing the performance metrics π_{str} of the lock we consider the torque τ, it can withstand that can be determined by the contact force F and its moment arm; this contact force is in turn limited by the latch strength. The maximum force that can be supported by the latch is still given by $F = \sigma_Y bt^2/6l$. The torque corresponding to this occurs when the horizontal arm of the contact force moment is given by $d = (2e\cos\theta + t\cos\theta - w)/2\sin\theta$, and θ is the angle at which this penetration is achieved. The corresponding maximal torque is then given by $\tau < -F\cos\theta\, d - F\sin\theta\,(e + (t/2))$.

The presence of shift e affects the penetration depth and hence the FFRs (Fig. 15a) would emerge differently for the same range of π_{str}, and similar for π_{ease} (Fig. 15b), for the values considered in unshifted slotted mechanism. The learning pattern for Figure 15a is shown in Figure 15c.

4.3. Discussion

As humans looking at the FFRs emerging in the three locking devices (Fig. 16a–c) we observe that the FFR for the padlock is quite similar to the slotted wheel mechanism although the performance metrics are different. At a second level of design change with an additional design variable, for the same acceptability range for the performative behaviors the emergent pattern is also quite similar though the lower boundary of the FFR is now angled up because of the e shift.

To comprehend why this similarity arises although the embodiments in these three instances are quite different, we observe that the physical principles by which these locking de-

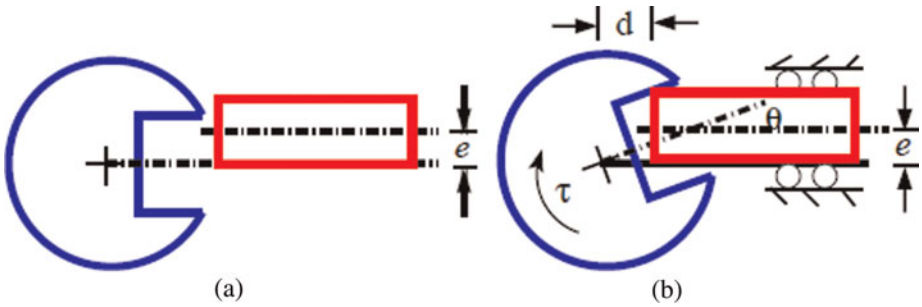

Fig. 13. The vertical shift in the latch. (a) The slotted wheel mechanism with shift e. (b) In addition to the contact formations of Figure 11a, a new contact state $\langle e_{12}, v_a \rangle$ is seen for upward shift e in the latch. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

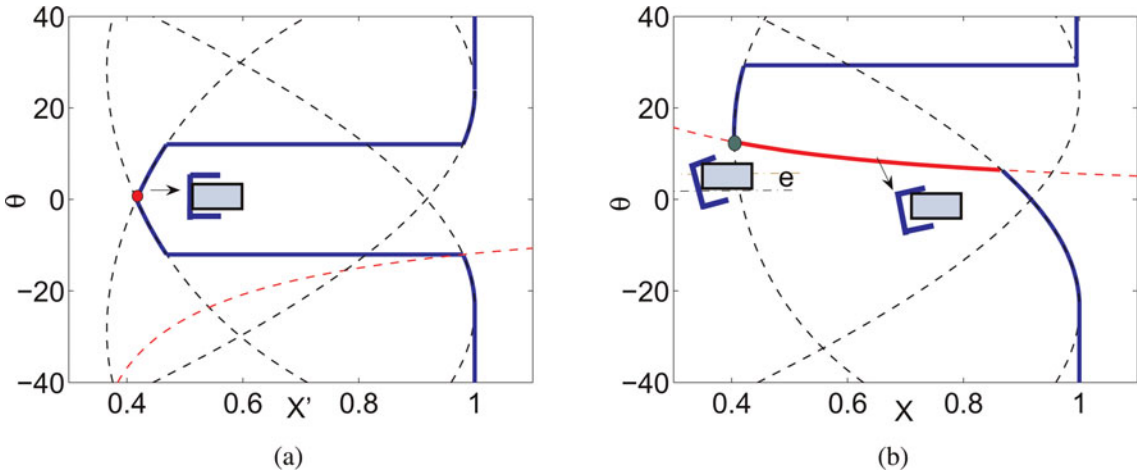

Fig. 14. The locking region configuration space (a) without axial shift or (b) when there is a upward shift in latch axis by shift *e*. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

vices work are similar, in that a latch is inserted into a slot preventing motion of the bolt or barrel containing the slot. This implies that in the locking region of the C-space, in each of these, there is a constraint on the motion of the bolt/barrel (Y or θ) when the latch position (X) is in a certain region. This can be seen clearly in the C-spaces in the bottom row of Figure 16, where we observe that for certain values of X, the Y, θ values are constrained and may not increase, thus locking the device. In contrast, once the latch is moved to the right (X increases beyond the locking limit), the Y, θ motion is unfettered in the free space of the C-space. Thus, the lock can be opened in these states. This type of similarity in the underlying principles incorporating function often holds across many embodiment differences, and in such situations one may expect to see certain similarities in the emergent functional patterns.

It is possible that the designer's intuitive feel for the degree to which such transfers are permissible is part of what Janssen (2006) has called design stance, which involve a broader and less specific type of preconception than Functional emergence being considered here. Eventually, it may be possible that CAD systems would be able to discover such higher level regularities, but clearly it would take much more experience and would require as a prerequisite the type of functional pattern abstraction being presented in this initial exploration.

5. VARIABILITY OF EXPERIENCE: ARE IMPLICIT CONSTRAINTS STABLE?

One of the questions that arise is regarding the nature of the convergence of the learned classifier function $f_e(\boldsymbol{\upsilon})$. We know that if f_e is a good characterization of $g(\pi_i)$ and if it is well learned, then it will narrow the design space. However, there are two questions this raises:

1. *Degree of convergence:* What does it mean to be well learned, that is, when do we know that the learning process has converged?

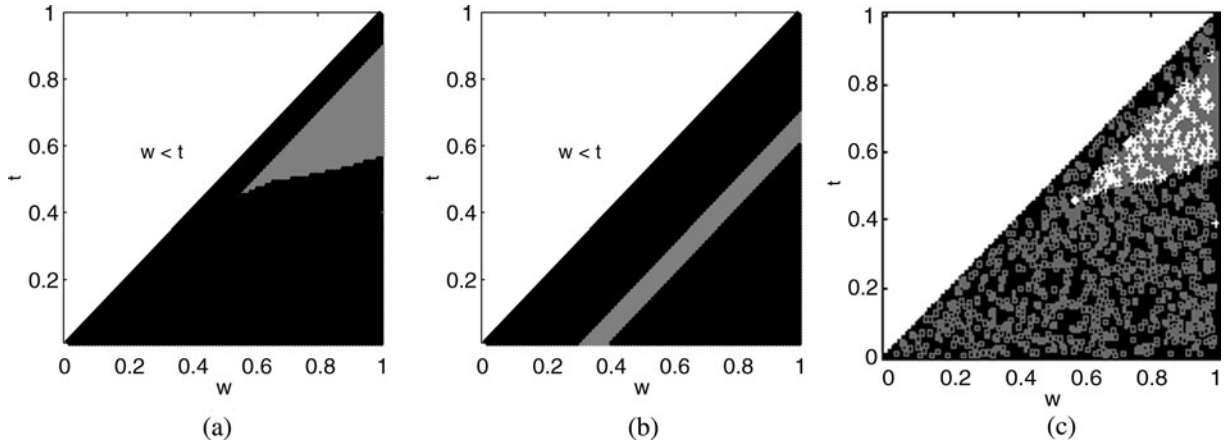

Fig. 15. Evolving constraints for slotted wheel mechanism with shift. (a,b) Different functionally feasible regions (FFRs) in the w, t design subspace for differing constraints on π_{str} and π_{ease} and (c) the learned FFR for (a).

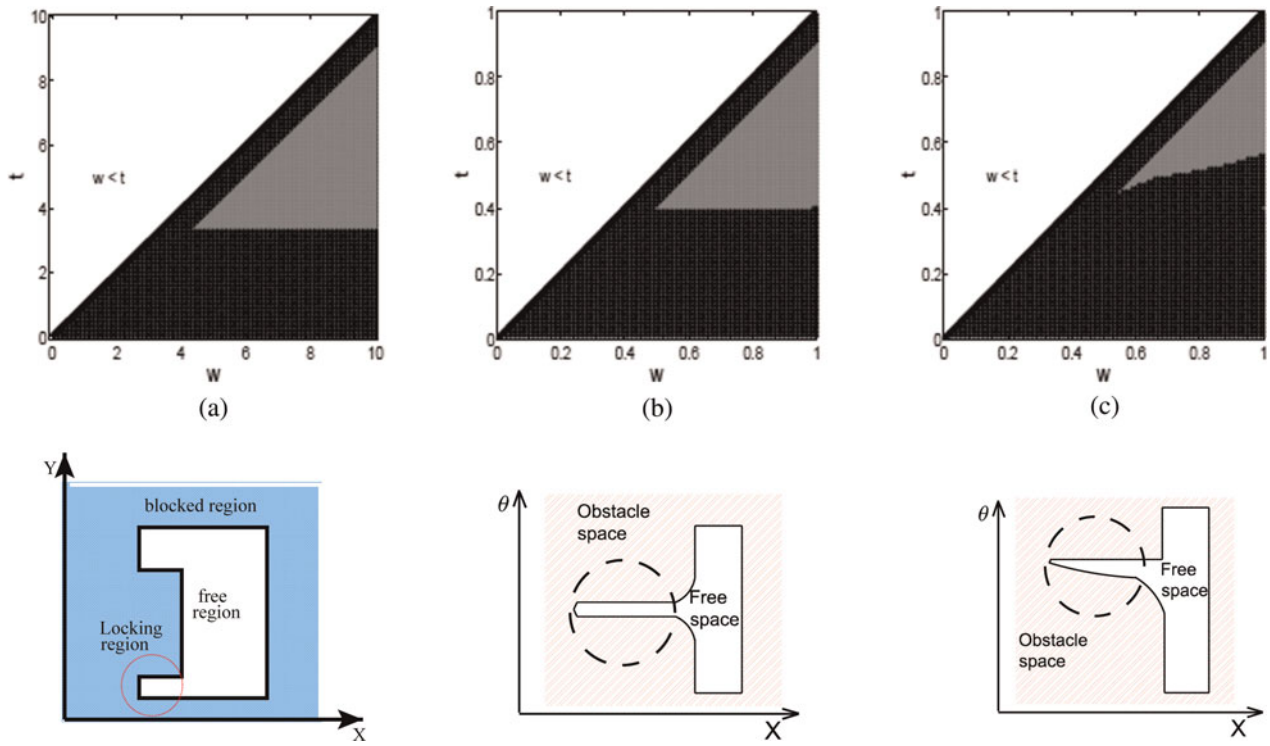

Fig. 16. The functional similarity in functional part families. (a,b) Different functionally feasible regions (FFRs) in the w, t design subspace for differing constraints on π_{str} and π_{ease} at (a) $\pi_{str} > 500$ N and $\pi_{ease} > 0.1$ mm, (b) $\pi_{str} > 6$ N m and $\pi_{ease} > 0.1$ mm, and (c) the learned FFR for (a) at $\pi_{str} > 6$ N m and $\pi_{ease} > 0.1$ mm. There are certain similarities in the emergent functional patterns for three different embodiments in the w, t design subspace for differing constraints on π_{str} and π_{ease}. The bottom row contains C-space mappings for the locking regions. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

2. *Stability of converged pattern:* Will two different learning experiences that are exposed to different trajectories of design instances end up in the same emergent space Ω_e or different ones?

Both questions are classic problems in computational learning theory. We are given a training sample $\mathcal{D}$, that is, a set of pairs $\mathbf{v}$, y, where y is a bit indicating whether the design $\mathbf{v}$ is a feasible design. The sample is drawn from a distribution in the design space, and y is the noise free result, that is, $g(\pi_i(\mathbf{v}))$. The function $f_e(\mathbf{v})$ is said to be a good generalization of g if its expected error rate on data drawn from the same distribution is less than ε with probability $1 - \delta$ [Bishop, 2006]. Constructing such a theoretical analysis requires one to model the distribution of designs as they are taken up for exploration; even then, the results present a worst-case analysis that is often much poorer than actual experience.

In this work we therefore restrict ourselves to an empirical investigation of these questions with data from the most complex of the three locking devices: the slotted wheel mechanism with axial shift. The first question, whether a learning process has converged or not, can be determined by estimating the error rate. This is done using a dense sampling over Ω_s at several iterations during the learning process, and seeing if a large fraction of samples have changed sign from an earlier learned pattern. The function is said to have stabilized if the fraction of samples changing signs between iterations is less than some parameter ε.

In order to study convergence, we consider the error in the learned function f_e as the misclassified fraction, that is, those design instances that are actually infeasible but show up as accepted in the learned function, which are false positives (FPs), and those that are feasible but are rejected, which are false negatives (FN). True positives (TP) are accepted design instances and true negative (TN) are the ones rejected by the learned function. Now we define error as the false results (FP + FN) over the whole region Ω_s (FP + TP + FN + TN). As we see in figures such as Figure 9a or 9d, patterns learned based on a small sample of the design space tend to have large areas that do not match the underlying functional constraint.

As with human designers, we find that our learned function better reflects the underlying patterns of feasible designs as the training set increases (Fig. 17 for the slotted wheel with axial shift; see also Fig. 9 for the padlock). Next, we consider variations in the learned pattern given a sparse training set (50 samples, Fig. 18a–c), and for a dense training set (500 samples, Fig. 19a–c). The patterns learned after greater exposure are clearly more uniform and have a smaller error rate.

We now consider a statistical analysis of the error rate over a large set of training runs (Fig. 20). All the classifiers here are attempting to learn the FFRs (feasible region constraint) for the slotted wheel, for the performance constraint $\pi_{str} >$

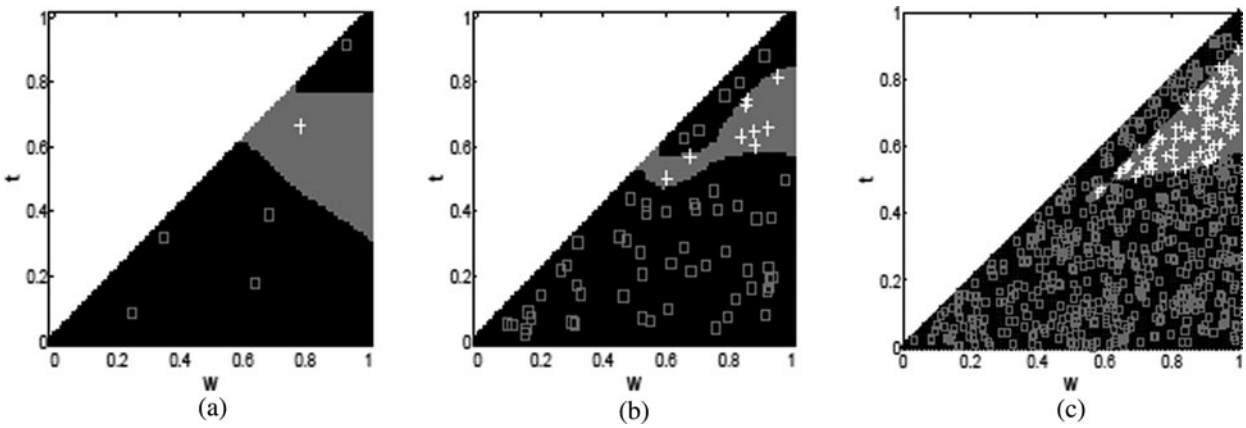

Fig. 17. The quality of implicit constraints learned improves with experience. With increasing experiences of (a) 6 (error = 0.1570), (b) 60 (error = 0.0689), and (c) 600 (error = 0.0145) instances, the functionally feasible region (FFR) converges more toward the ideal FFR in Figure 15a.

6 N m. Sample sizes for training are chosen in multiples of two, from 2^3 to 2^{10}. The mean error and standard deviation are computed from 25 trials with each sample size. We observe that even with a meager eight training samples, the error (percentage of test cases classified wrongly) is only 13.2%. Although the mean error keeps decreasing, the standard deviation remains quite high at about 2.2% up to $N = 64$. Eventually, the standard deviation becomes negligible as the training size approaches 1000 samples and the error percentage, which is falling asymptotically, goes down to about 1.6%.

Thus, this analysis on these simple initial case studies show that as expected, the learned function tends to converge closer to the target function as training size increases. A significant aspect of this analysis is that, as variance decreases, the system acquires greater confidence in the validity of the learned function; this replicates human experience. In the early stages, the designer is unsure of her abstractions but becomes increasingly confident after exploring a large number of design instances. This can also provide a measure for what constitutes an "adequate" sample size. Clearly, for more complex design spaces with higher dimensionality, this would still be a very large number. Finding ways to reduce the dimensionality and therefore the adequate sample size would be an important challenge.

5.1. Differences with human sketching

An important difference between human and machine learning of these implicit constraints is that the program requires hundreds of sample points to be explored even in this relatively simple design space, whereas designers typically go through only a handful of sketches even in more complex situations. This may be explained by three factors. Each sketch is not a fully defined design instance, so it leaves many aspects undefined. Thus, a sketch represents a constrained region of the design space, and not just a single design. Rejecting or refining a sketch is equivalent to exploring hundreds of design instances in our paradigm, and the human designer

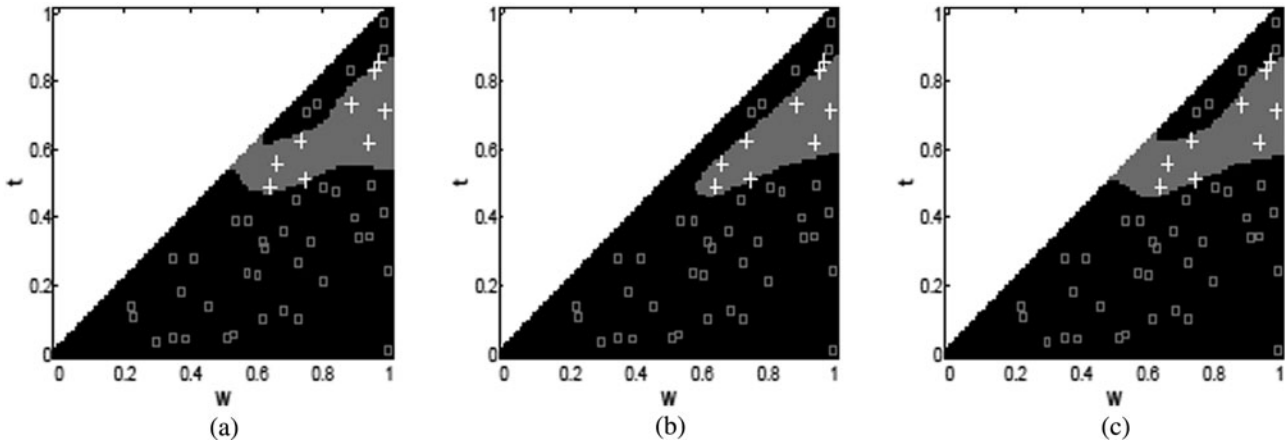

Fig. 18. Learning based on 50 design instances. With limited exposure, the functionally feasible regions (FFRs) exhibit wide variation: (a) error = 0.0592, (b) error = 0.0608, and (c) error = 0.0586.

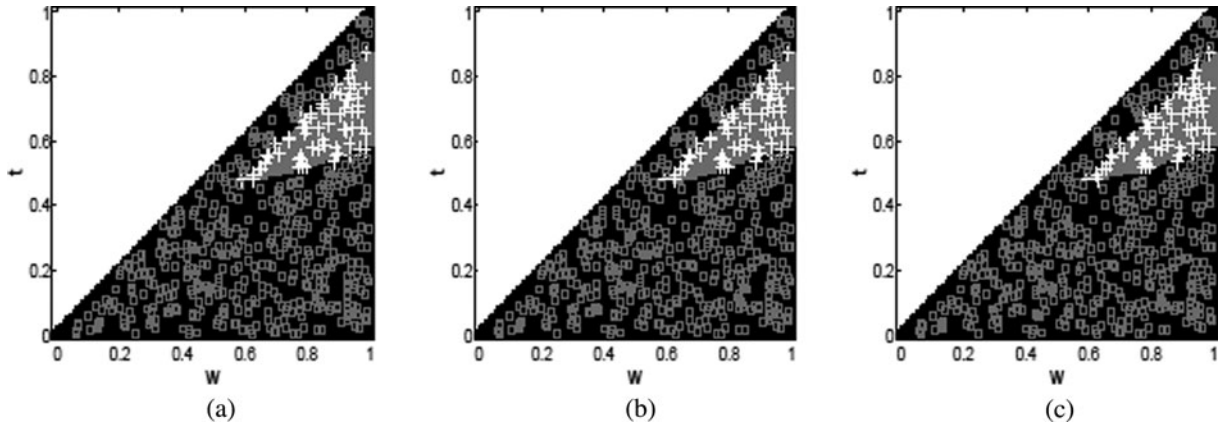

Fig. 19. Learning after 500 instances. The error is almost constant with a higher number of different experiences: (a) error = 0.0254, (b) error = 0.0248, and (c) error = 0.0251.

rules out large swathes of the design space in each evaluation, leading to faster convergence. Of more importance, the human designer is not choosing the sketches randomly; models that are quickly seen to be unlikely to serve the performance requirements are not sketched at all. It may be argued that the progression of sketches already incorporates some idea of the functional constraints as has been abstracted by the early experience. As seen above, even a few design explorations reveal a good bit about the structure of the space of good designs; hence, using this pattern to drive new sketches, instead of choosing randomly all of the time, makes for much faster convergence. Finally, the human designer starts with a already available store of domain knowledge, defined in terms of similar functional structures as seen earlier, whereas the computational process identified here is (at least for now) blind to such interrelations.

6. CONCLUSION: SCALABILITY AND FUTURE WORK

In this paper, we have presented a computational process that attempts to imitate an expert designer's ability to quickly come up with good designs via functional emergence, which is modeled here in terms of FFRs. This is suggested as a first step in the cognitive process whereby a designer redefines the design problem in terms of new chunks and eventually symbols, which are emergent properties of the design problem that are discovered during exploration. Discovering these regions of functional feasibility, the FFRs, is itself an early form of functional emergence, because the patterns revealed are novel and may result in considerable narrowing or information compression. We show how computational algorithms can identify such FFR regions in an apprenticeship mode, where well-defined problems are presented to it. We demonstrate this process on a padlock embodiment. By considering two other embodiments in the same functional class, we present empirical evidence that such patterns of functional emergence may be similar across some set of designs in a functional class, suggesting that one may also be able to learn higher level patterns of functional emergence beyond the embodiment level. Such a model may be thought of as a key step in the shift from novice designer behavior, which considers case-based similarity with a specific design problem, to expert behavior that schematizes for a class of designs (Linsey et al., 2008). However, such schematization requires the system to discover the kind of abstract representations for each embodiment class that we are trying to develop here; thus, the computational procedure for discovering such schemata for larger classes of designs remains work that can only be done after this step is well entrenched.

Obtaining the regions of functional feasibility is only the first step in a long process of abstraction in design. Sometimes, the feasible regions constitute thin parts of the design

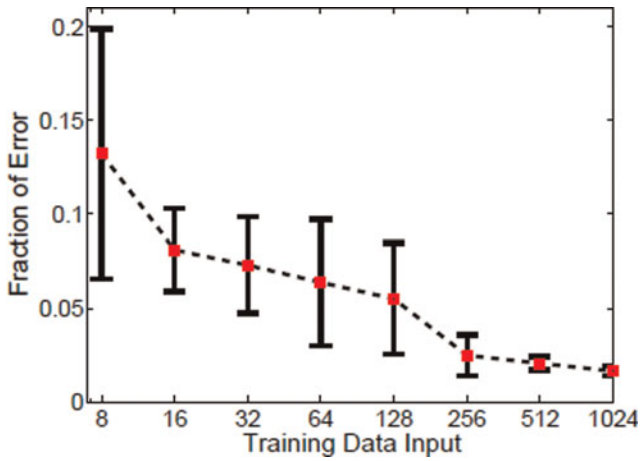

Fig. 20. The function quality with increasing exposure. The error is the percentage of the false region in Ω_e to the total region Ω_i in the design subspace w, t. The mean error of the learned constraint function, as well as its standard deviation (σ), decreases significantly as the number of training instances increase from 16 to 1024. These data are based on 25 independent runs for each sample set for the slotted wheel mechanism with $e = 0.3r$. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

space; in such situations they may readily yield lower dimensional manifolds, an example of which is presented for a simple linear chunk. In other work, we have explored nonlinear dimensionality reduction, especially as they may relate to multiple-objective optimization of designs (Mukerjee & Dabbeeru, 2009). The existence of such lower dimensional manifolds indicates the existence of tight interrelations between the design variables that must hold in order to meet some functional requirement. If these latent interrelations form a stable pattern, they are likely to represent chunks. Chunking is often suggested as a key step in the process of developing human expertise, because they significantly reduce the dimensionality and hence the complexity of the design search. In our current work, we are considering various models for discovering chunks as low-dimensional manifolds in the design space.

The examples demonstrated here are rather simple, and an important question is that of scalability as we consider increasingly complex designs. Even in more complex cases, for example, involving many more design variables and performantive behaviors, such a process would still be applicable as long as the function is quantifiable in terms of performance metrics. However, learning adequately accurate functions is likely to require a much larger training data in more complex spaces, and the algorithmic complexity of this training process remains an important consideration for future. Nonetheless, even as presented here, the system is possibly a great help to novice designers, who may observe in the resulting patterns some explanation in terms of the underlying parameters, which would add to her design knowledge in this domain.

6.1. Learning "commonsense" knowledge

One of the aspects in this type of machine-learning process differs from human design exploration lies in the fact that humans bring to bear a great degree of broad prior knowledge to the design task. This helps the human avoid unnecessary evaluation of thousands of unlikely design instances, where each evaluation may involve extensive computations. Without a similar capability, machine abstraction would clearly have difficulty in scaling up to larger problems. This much richer conceptual base, sometimes known as commonsense knowledge (e.g., that a fat peg cannot go into a thin hole), may help select more likely candidate designs. In this sense, the machine may be thought of as an infant first experiencing such tasks (the peg-in-hole constraint is one of the early constraints learned by humans by age 5 months; Spelke & Hespos, 2002). The approach posited here makes only a small beginning, but it is supposed that in the long run, the functional constraints learned hereby would be generalized to much broader design systems.

As a simple example of how this may be working, let us consider the peg-in-hole task a bit further. Consider an infant playing with inserting sticks into holes, and a padlock designer in the conceptual design stage, thinking about how the latch enters in a slot in the U-bolt. Both are insertion tasks, but compared to the infant, the designer has extremely sophisticated abstractions of how objects fit into one another, the constraints these impose on the relative motion of the parts, and so forth. We propose that such learning may occur easily using the paradigms presented here. As a side effect of exploring the w, t space demonstrated here, the system can also learn the constraint, at least implicitly, that $w < t$. This is shown in Figure 21 for 10, 50, and 200 instances. At this point, the system is in no position to make any generalizations from this behavior; after observing similar constraints in a wide range of other mechanisms, it is possible that the system may be able to generate a general constraint for insertion situations from a large number of *ab initio* explorations such as this one. As opposed to the human-defined logical formalisms, such generalizations, when learned, would be grounded and be able to "capture product semantics" (Brunetti & Golob, 2000) in a far more flexible manner than the brittleness of present knowledge-based systems. This work thus proposes that, instead of attempting to preprogram such domain knowledge, it may be better to acquire this knowledge as patterns within the design space.

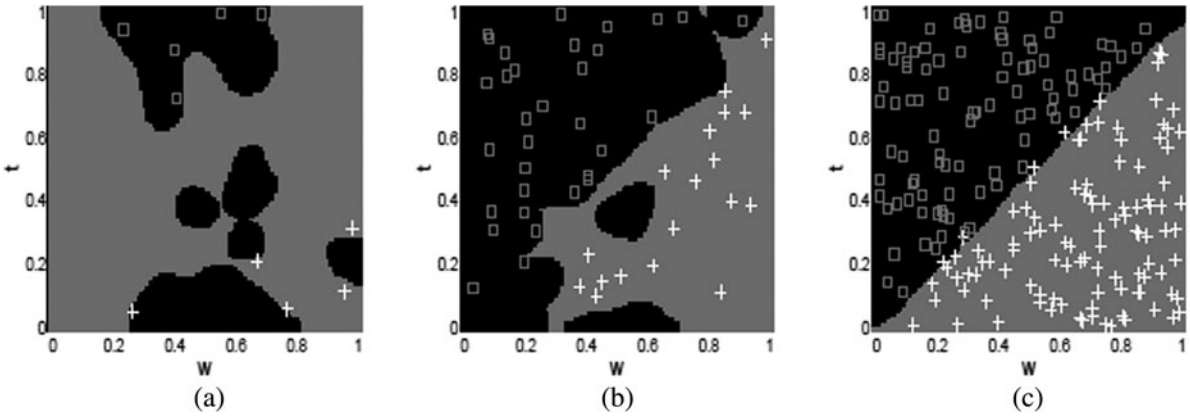

Fig. 21. Learning through experience that the latch must be smaller than the slot ($w > t$). The quality of the learned pattern improves with experience: results after experiencing (a) 10, (b) 50, and (c) 200 instances. Crosses indicate feasible trained data, and boxes indicate infeasible trained data.

6.2. Learning symbols

Thus far, we have considered only implicit knowledge, which is knowledge that is understood as an unstable pattern that is difficult to relate to newer experiences. In human usage, symbols, which are grounded in experience and have a reference label, serve the needs for indexing the knowledge structure so that new experiences can associated with it and modify it. At the same time, social conventions of symbol usage also tend to stabilize the model; without it, newer experiences would continue to shift the symbol, and there would be no stability. The other aspect of symbols is that they are conscious or explicit; they are not the implicit functions we have been learning throughout this paper. We suggest that learning such symbols may not be a very difficult step based on what has been demonstrated here with FFRs.

Commonsense knowledge of the type shown above, after being observed in a large number of instances, may become *reified* or come into conscious awareness. At this point, if one talks to human designers, one may learn a preexisting term for the concept, or one may invent such a term oneself. Thus, the concept comes to have both a grounded model, say, a low-dimensional chunk defined on the variable space, and a label by which it can be indexed in memory. Then new experiences can attach to this meaning-label pair, which is traditionally called a symbol. Thus, the process of symbol emergence involves exploration of the design space, discovery of low-dimensional chunks, and then interaction with (human) language users to discover preexisting labels, if any. We feel this process is an important research direction for computational models of design, because it opens up a bridge from the creative design models proposed by Schon (1983) to a more structured knowledge system which can be usefully deployed in design. By developing computational models for the reflective practice of design, we feel the proposed approach opens many avenues for discovery of abstractions in the design process.

ACKNOWLEDGMENTS

This research was supported by the Research I Foundation, Indian Institute of Technology Kanpur (IIT Kanpur). We thank the anonymous reviewers for their valuable recommendations and comments.

REFERENCES

Ahmed, S., Wallace, K.M., & Blessing, L.T. (2003). Understanding the differences between how novice and experienced designers approach design tasks. *Research in Engineering Design 14(1)*, 1–11.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.

Bor, D., & Owen, A. (2007). Cognitive training: neural correlates of expert skills. *Current Biology 17(3)*, 95–97.

Brunetti, G., & Golob, B. (2000). A feature-based approach towards an integrated product model including conceptual design information. *Computer-Aided Design 32(14)*, 877–887.

Cagan, J., & Agogino, A. (1991). Dimensional Variable Expansion—a formal approach to innovative design. *Research in Engineering Design 3(2)*, 75–85.

Campbell, M.I., Cagan, J., & Kotovsky, K. (2003). The A-design approach to managing automated design synthesis. *Research in Engineering Design 14(1)*, 12–24.

Chandrasekaran, B. (2005). Representing function: relating functional representation and functional modeling research streams. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 19(2)*, 65–74.

Chase, W., & Simon, H. (1973). The minds eye in chess. In *Visual Information Processing* (Chase, W.G., Ed.), pp. 215–281. New York: Academic Press.

Cross, N. (2004). *Engineering Design Methods: Strategies for Product Design*. Chichester: Wiley.

Dabbeeru, M.M., & Mukerjee, A. (2008). Negotiating design specifications: evolving functional constraints in mechanical assembly design. *Proc. 2008 ACM Symp. Solid and Physical Modeling, SPM '08*, pp. 333–338. New York: ACM.

Dabbeeru, M.M., & Mukerjee, A. (2010). Learning concepts and language for a baby designer. *Proc. 4th Int. Conf. Design Computing and Cognition*. Stuttgart: Springer.

Ericsson, K.A., & Lehmann, A.C. (1996). Expert and exceptional performance: evidence on maximal adaptations on task constraints. *Annual Review of Psychology 47*, 273–305.

Faltings, B. (1992). A symbolic approach to qualitative kinematics. *Artificial Intelligence 56(2–3)*, 139–170.

Gero, J.S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine 11*, 26–36. Accessed at http://people.arch.usyd.edu.au/john/publications/ger-prototypes/ger-aimag.html

Gero, J.S., & Kannengiesser, U. (2004). The situated function–behaviour–structure framework. *Design Studies 4(25)*, 373–391.

Gobet, F., Lane, P.C.R., Croker, S., Cheng, P.C.H., Jones, G., Oliver, I., & Pine, J.M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences 5(6)*, 236–243.

Gobet, F., & Wood, D. (1999). Expertise, models of learning and computer-based tutoring. *Computers & Education 33(2–3)*, 189–207.

Goel, V. (1995). *Sketches of Thought*. Cambridge, MA: MIT Press.

Gross, M., Ervin, S.M., Anderson, J.A., & Fleisher, A. (1988). Constraints: knowledge representation in design. *Design Studies 9(3)*, 133–143.

Gross, M.D. (1986). *Design as exploring constraints*. PhD Thesis. Massachusetts Institute of Technology, Department of Architecture.

Janssen, P.H. (2006). The role of preconceptions in design: some implications for the development of computational design tools. *Proc. 2006 Design Computing and Cognition, DCC'06* (Gero, J.S., Ed.), pp. 365–383. New York: Springer–Verlag.

Ji, X., & Xiao, J. (2001). Planning motion compliant to complex contact states. *International Journal of Robotics Research 20(6)*, 446–465.

Lin, V.C., Gossard, D.C., & Light, R.A. (1981). Variational geometry in computer aided design. *Computer Graphics 15(3)*, 171–177.

Linsey, J., Wood, K., & Markman, A. (2008). Modality and representation in analogy. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22(2)*, 85–100.

Lloyd, P., & Scott, P. (1994). Discovering the design problem. *Design Studies 15*, 125–140.

Mukerjee, A., & Bhatia, P. (1995). A qualitative discretization for two-body contacts. *Proc. 14th IJCAI Conf.*, pp. 915–921, Montreal.

Mukerjee, A., & Dabbeeru, M.M. (2009). The birth of symbols in design. *Proc. ASME 2009 Design Engineering Technical Conf., DETC'09*.

Oxman, R. (2002). The thinking eye: visual re-cognition in design emergence. *Design Studies 23(2)*, 135–164.

Poon, J., & Maher, M.L. (1996). *Emergent Behaviour in Co-Evolutionary Design*, pp. 703–722. Dordrecht: Kluwer Academic.

Schon, D.A. (1983). *The Reflective Practitioner*. New York: Basic Books.

Soufi, B., & Edmonds, E. (1996). The cognitive basis of emergence: implications for design support. *Design Studies 17*, 451–463.

Spelke, E., & Hespos, S. (2002). Conceptual development in infancy: the case of containment. In *Representation, Memory, and Development: Essays in Honor of Jean Mandler* (Stein, N.L., Bauer, P.J., & Rabinowitz, M., Eds.), pp. 223–246. Mahwah, NJ: Erlbaum.

Sutherland, I.E. (1963). Sketchpad—a graphical man–machine interface. PhD Thesis. University of Cambridge.

Suwa, M., Gero, J., & Purcell, T. (2000). Unexpected discoveries and S-invention of design requirements: important vehicles for a design process. *Design Studies 21(6)*, 539–567.

Suwa, M., & Tversky, B. (1997). What do architects and students perceive in their design sketches? A protocol analysis. *Design Studies 18*, 385–403.

Umeda, Y., & Tomiyama, T. (1997). Functional reasoning in design. *IEEE Intelligent Systems and Their Applications 12(2)*, 42–48.

Wolter, J., & Chandrasekaran, P. (1991). A concept for a constraint-based representation of functional and geometric design knowledge. *Proc. 1st ACM Symp. Solid Modeling Foundations and CAD/CAM Applications Conf.*, pp. 409–418.

Yaner, P., & Goel, A. (2008). Analogical recognition of shape and structure in design drawings. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22(2)*, 117–128.

Madan Mohan Dabbeeru is a Postdoctoral Researcher at the University of Maryland, College Park. He received his MS degree in mechanical engineering with a specialization in robotics from Birla Institute of Technology and Science, Pilani, India, and his PhD in mechanical engineering from IIT Kanpur in 2009. During his doctoral work at IIT Kanpur, he studied artificial intelligence and cognitive science in design and proposed an infant designer enterprise to discover symbolic knowledge about the design via explorations in the design space. Dr. Dabbeeru's research interests are machine learning in design and robotics.

Amitabha Mukerjee is a Professor of computer science and engineering at IIT Kanpur. A graduate of IIT Kharagpur, he attained his PhD at the University of Rochester in 1986 and was an Assistant Professor at Texas A&M University before joining IIT Kanpur in 1992. Dr. Mukerjee works in cognitive models of space, perception, and language and is particularly interested in the developmental process whereby sensorimotor similarities result in perceptual schemas.

APPENDIX A: GLOSSARY

Design space Ω	The n-dimensional space in which designs are represented by points, where n is the number of independent variables
Design variables $\boldsymbol{\upsilon}$	The set of independent variables that define the design, sometimes called driving variables
EPF	Embodiment part family: a set of members, in which each member will share the same set of performance measures π and the mapping from design variables to performance measures is the same and hence have the same performance metrics
FPF	Functional part family: a set of members, in which each member will share the same set of performance behaviors but the specific performance measure π_i may be different because the mapping from design variables to performance measures can be different
Performative behaviors	The subset of behavior space that is of interest to the user
PDD	Phenomenological design domain: has a set of members, in which each member will share some set of performance behaviors
Specification constraints $f_s(\boldsymbol{\upsilon})$	The specification constraints $f_s(\ )$ constrain the unbounded design space Ω