Discovering Implicit Constraints in Design

Madan Mohan Dabbeeru and Amitabha Mukerjee
Indian Institute of Technology Kanpur, India

Abstract. In familiar design domains, expert designers are able to quickly focus on "good designs", based on constraints they have learned while exploring the design space. This ability to learn novel constraints is a key aspect in which design differs from traditional optimization; the constraints on the search are constantly re-defined based on the search experience itself. Moreover, such constraints are often implicit, i.e. the designer may find it difficult to articulate these constraints and provide reasons for them. Here, we ask if computer-aided-design systems can discover such implicit constraints in well-understood design situations, where the function can be articulated clearly enough to be quantified in terms of *performance metrics*. By considering function across a large number of design instances, patterns of functional feasibility may be learned as a byproduct of evaluating different designs. We show how patterns of functional infeasibility result in novel constraints that rule out certain regions of the design space. We demonstrate this process using examples from the design of simple locking mechanisms, and as in human experience, we show that the nature of the constraints learned depends on the extent of exposure in the design space, and may be widely variable in early stages. We also show how the process of design change, when the design space is modified, e.g. by adding new design variables, can build on patterns learned on past designs. In conclusion, we discuss the ramifications of this process on chunking and representational change, and also on design creativity.

"The expert designer has explored extensively in previous sessions and no longer needs to try out many different alternatives. The expert is confident of immediately choosing a good one, based on experience. [16]"

1 Implicit Constraints in Expert Design

Repeated studies have observed how experienced designers come up immediately with designs that are superior in most respects to those produced by novice designers or CAD systems [16, 19, 5]. Clearly, this proficiency is related to some patterns learned in past experience.

This paper builds on the well-known cognitive claim that in part, the expert's ability to immediately come up with such good designs lies in her use of additional constraints, which are often implicit in the sense that the designer herself may not be able to articulate them coherently. In think-aloud sessions, designers may use terms like "looks right" without giving more detailed justifications other expressions such as "worked before". This phenomenon has been called

"intuition based upon previous experience" [1], contrasting it with situations where the designer refers to past designs in a deliberate manner.

The implicit nature of such domain-specific constraints appears to be common in expert behaviour across many domains, ranging from chess, medicine, computer programming, bridge, physics, etc. [8]. In chess, grand-masters do not evaluate more positions than far weaker players, but the positions they do evaluate tend to be the better ones [15]. It has been noted that their constraints are also implicit; they are rarely revealed in direct introspection, but when the expert is questioned as to why a certain line was not considered, they will often say that it simply did not strike them [15]. Later that line almost invariably turns out to be flawed, thus validating the power of the intuitive constraint that was used. While analogies of design processes with domains such as chess have been contested since design is a much less well-defined problem [1], the implicit nature of these intuitions across so many domains cannot be ignored. Among expert human designers, similar questioning occasionally reveals domain-specific biases, but the presence of such constraints may be more widespread than appears in introspective testimony. Indeed, some researchers feel that design is essentially a process of exploring constraints [28]. Whatever be the nature of this knowledge, it is clear that somehow the expert is able to convert her experience in familiar design domains into a constraint that narrows down the design to a more fruitful region of the search space. On the other hand, the novice designer (as well as the CAD system) gives equal importance to all designs that satisfy the design specs. Clearly, the ability to learn such patterns would confer a significant search advantage for CAD systems.

The discovery of underlying patterns in the design space may be one of the earliest steps in a long process of cognitive discoveries that forms the core of design expertise. Sometimes the implicit pattern discovered may draw the designer's attention to those aspects of behaviour, which may help formulate an explicit awareness of certain interrelations, what has been called "situated invention" [33]. In the long run, this may be the first step towards generalizing over regions in the design space, a process called chunking, that results in a restructuring of the design representation itself [36]. Differences in evaluating functions (e.g. 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 the primary factors behind differentiations in design style.

1.1 Design Space Exploration, Sketching, and Emergence

The human designer discovers patterns of functional effectiveness while exploring different parts of the design space, often using sketching as the mechanism of choice for this preliminary exploration. During this process, various design choices are quickly evaluated for functional feasibility, often using additional visual constructs that operate on the sketches themselves [13]. In the language of [14], "one reads off the sketch more information than was invested in its making." Thus the sketch "triggers" other images in the mind, leading to a wider

exploration of the visual and spatial ramifications of the particular part of the design space, cycling between "seeing as" and "seeing that", where the former may potentially lead to a restructuring of the design space [34]. Where sketches are explicitly unavailable, the designer may conduct a similar exploration of the design space using mental imagery, often revealed through gestures or other modalities [2].

A key aspect of sketch-based exploration is the emergence of new ideas, closely related with the idea of implicit constraints. Emergence occurs when "interesting properties" arise that were not part of the initial design goals [4]. These interesting functions often arise from the complex interaction involving relatively simple low-level mechanisms. An alternate view of emergence is that aspects that were implicit in the design, are thought to be made explicit[31]. In either view, design is seen a dynamic process, where both the solution and the search space evolve dynamically in the course of the design with the evolving implicit constraints. As a result of these constraints, the process of design is far from open, and the next sketch is extremely biased by the designer's evaluation of the present sketch, so much so that design becomes more a process of *generation* rather than exploration [7]. Computational paradigms have attempted to capture this by simultaneously evolving both the solutions and also the optimization criteria [27, 30]. As part of this process, new abstractions are formed of what constitutes the feasible zone for a design in a design space. However, exactly how these specifications may be computationally reformulated in terms of function has not been worked out in detail.

2 Discovering Implicit Constraints in Machine Design

Since the origins of Computer-Aided design, e.g. in the very early ideas of Ivan Sutherland [32], 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[28]. Indeed, the idea of constraints on design parameters, originating in the work of Gossard [26], 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. Partly this is because of a difficulty of 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. Part of the design quest is also this clarity in defining the function; but as of now, most CAD systems leave the specification of function for the human designer. Recently, Janssen [19] has proposed CAD models that incorporate designer preconceptions as similar embodiments and functions; in this paper, we use the term *Functionally Feasible Regions* FFRs to denote these "preconceived" functional constraints. While Janssen characterizes these functions, we present a mechanism for learning such FFRs within a computational framework.

One of the assumptions of this work is that function is well-characterized for the class of designs under consideration. For instance, in the case of locks, an important function may be strength, along with weight, cost, and robustness against jamming. We show how exploration in the multi-function space results in identification of different levels of feasibility in different parts of the design space. Some of the constraints learned for our second example design, the slotted wheel mechanism (rotating barrel lock), some of the constraints "discovered" by the system are quite enlightening.

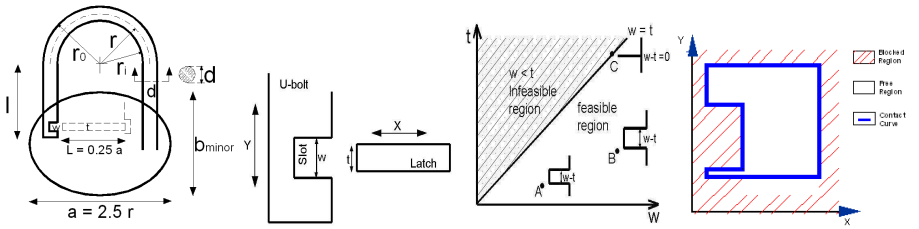
In human sketching, functional evaluation uses visual mechanisms that are made available while exploring the imagery of the sketch itself[34]. In the case of computational models, such processes can be simulated using computable functions applied to various fully instantiated 3D models. In order to address this question computationally, we need a formal notion of design space.

2.1 Design Space

Following the design optimization tradition, the Design Space Ω as the space defined by a finite set of *design variables* v_i, i.e. the set of independent parameters that define a design instance. These design variables are traditionally expressed as the design vector $\underline{v}$, $v = \{v_1, v_2, v_3 ... v_n\} \in R^n$. Fixing all the parameters results in a unique design, which may then be evaluated for different functions. Typically the design space is constrained by certain user-specified criteria, often called *design specifications*.

Consider the padlock shown in Fig. 1. In our model, the padlock design space consists of five design variables: b_{minor}, height of the elliptical main body; r: U-bolt curvature; l: length of U-bolt; t: thickness of latch, and w: width of the slot opening in the U-bolt. A number of other structural dimensions related to shape and function are determined from these design variables alone, e.g. the width (major axis) of the elliptical body, a is specified as $2.5r$ and there are few constraints $r_0 > r_i$ to define valid geometry. The design vector $\underline{v}$ is the vector of these five design parameters. The Design Space (Ω) is the space of the five-dimensional design vectors.

Given a set of values for the the design variables, these uniquely define a larger set of *structural variables*, which are dependent parameters. For the padlock example, given the five design parameters specified above, other structural variables such as r_i can be derived. Given the full set of structural variables, the final design is uniquely specified. The design space is bounded by defining a set of specification constraints $f_s(\underline{v})$ which must be true (here f_s is taken to be boolean; one may consider these to be algebraic functions, of the form $f() < 0$, say). These constraints typically define one or more continuous regions in the space of the design variables, and it is assumed here that the design is not over constrained (no valid designs are possible) or under constrained (design space is unbounded). Thus the set of design variables, together with these constraints, along with the mappings from the design variables to the structural variables, define the design space Ω.



(a) (b) (c) (d)

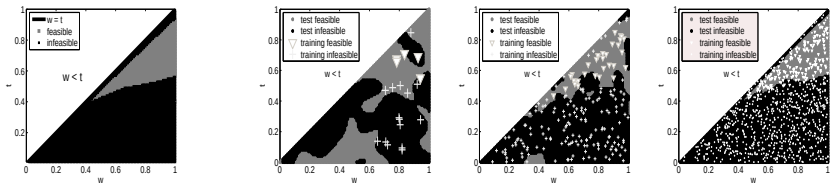
Fig. 1. *Functional Constraints: Padlock Example*: (a) Padlock design vector $\underline{v} = \{b_{minor}, r, l, t, w\}$. (b) Design fragment U-bolt and latch moving vertically and horizontally respectively. (c) The (w, t)-slice in the design space. Here $w < t$ is clearly infeasible, since the latch collides with the slot. If $w - t$ is too small, ease of operation may be hindered by small misalignments; if it is too large, it may leave too much play and weaken the lock. Parts of the configuration space for design instances C, B, and A are shown in the figure (d)Configuration Space for latch-bolt design fragment - relates horizontal motion of latch (X) with vertical motion of U-bolt (Y).

2.2 Computational Discovery of Functional Patterns

How might computational processes come up with implicit functional constraints? We describe a simple approach for learning the patterns underlying the functionally feasible regions (*FFR*s) based on the evaluation of candidate designs. The discovery of such new constraints may also be called *Schemata* or patterns in the design space that underlie expert designer knowledge [25]. Any general purpose function approximators algorithm can be used; here we use multi-layer perceptrons as our vehicle for learning the *FFR*s. In practical systems, the resulting constraints may be of direct value to novice designers, especially in situations involving many choices [16]. Clearly this would constitute an important step if we are to enable computers greater access to the range of creative improvements possible for the human designer [19]. Computationally, the process of exploring design, involving computationally expensive analysis of aspects such as strength, flow, or motion, can then be limited to a much smaller range. More importantly however, it may be possible to determine a tradeoff between the constraints given and the functions that are specified, a key requirement in Creative design [5].

In geometric design, the specification constraints $f_s(\underline{v})$ typically deal with the relationship between design variables in the form of algebraic (non-linear) equations. Some of these constraints ensure that the design is geometrically sound (e.g. the geometry has no singularities, or that the mapping functions to the structural variables are not violated). Based on the specific constraint values, a geometric object may be well constrained, under constrained and over constrained [18].

Here we are interested in an additional class of constraints, which we may term as *Functional constraints*, that specify the relationship between elements of function. Consider for example, the function of a padlock Fig. 1, where the latch of thickness t has to enter the slot with a clearance w. Here, the constraint



$\tau > 6$ for actual functional constraint $\tau > 6$ learning with 20 $\tau > 6$ learning with 200 $\tau > 6$ learning with 450

Fig. 2. *Learning through experience for slotted-wheel.* The implicit constraint on the w, t subspace of the Design Space is learned under the functional specification that the breaking-torque τ must be greater than 6k-Nm. The exact FFR is shown in the leftmost figure. The implicit constraint learned by a multi-layer perceptron based on 20, 200, and 450 samples are shown next. The decision surface learned with fewer trials are extremely variable, but it stabilizes for larger samples. By 450 instances, the learned model is quite stable

$w > t$ defines a partition in the design space which may not have been explicitly specified as part of the formal design specifications $f_s(\underline{v})$. However, even novice designers will discover this constraint immediately upon exploring even a single design; thus this constraint will become explicit very soon. This may be treated as a trivial instance of an emergent constraint. On the other hand, there may be other constraints arising from functional considerations, e.g. the strength of the lock against hammering, which impose other patterns on the design space which are far from obvious. Quite often, even with experienced designers, such patterns may be internalized in an inarticulate manner, and will thus remain as implicit constraints.

As an example of such an implicit constraint that may be learned by our system, consider the functional specification that a lock (not the padlock, but one with a rotating barrel, say) must have a strength that can withstand a given torque. For such a specification, it turns out that the thickness t cannot be too low, for then the latch will fail too easily. Similarly, if the gap between the slot-width and the latch, $w - t$, is very low, the latch may not enter deep enough into the slot, resulting also in failure. The actual and learned FFRs, or implicit constraints, for resisting a torque of 6 kilo-newton-meters is shown in Fig. 2.

2.3 Design Space Shrinking

Since the patterns for the feasible functional regions holds only over designs with a high functional performance, it results in a narrowing of the design space. Let Ω be the initial design space defined by the set of design specifications $f_s(\underline{v})$. Let $f_e(\underline{v})$ be an emergent constraint that is learned through exploration, based on a function characterized by a set of performance metrics $\pi(\underline{v})$. Now, whenever f_e is true, f_s must also be true, otherwise these designs $\underline{v}$ would not have been explored at all. This is the basis of the following simple result, which motivates this work.

Definition 1 (Specification constraints). . *The specification constraints* $f_s()$ *constrain the Design Space* Ω*, which is defined as:* $\Omega = \{\underline{v} | f_s(\underline{v})\}$.

Definition 2 (Emergent constraint). *An emergent constraint* $f_e(\underline{v})$ *is true for those design vectors where some functional metric* π *is higher than elsewhere in the design space.*

Lemma 1. *There are some designs* $\underline{v}$ *in* Ω *s.t.* $\{f_s(\underline{v}) \wedge \sim f_e(\underline{v})\}$*; i.e., for these designs, the value of the performance metric* π *is lower than that acceptable by* $f_e()$. □

Theorem 1 (Design Space Shrinking Theorem).
If $f_e(\underline{v})$ *is an emergent pattern corresponding to some functionally feasible regions, then applying this constraint narrows the design space from* Ω *to* Ω_E*, where* $\Omega_E \subset \Omega$.

Proof. This follows from the fact that $(\exists \underline{v})\{f_s(\underline{v}) \wedge \sim f_e(\underline{v})\}$.

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 $\frac{\|\Omega_E\|}{\|\Omega_S\|}$. 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 behaviour. This is possibly the reason behind what [4] has called "interestingness" of the emergent relation.

Clearly, the function evaluation π is a key aspect of any computational process for discovering implicit constraints. In the next section, we develop on our model of function in terms of a quantifiable performance metric for the examples of a simple padlock and a rotating barrel padlock. Last, we show how changing a design space, e.g. by introducing an additional design variable for the rotating barrel, causes some of these implicit functions to change.

3 Function as Performance Metric

Modeling function and how it relates to structure is a complex question that has been addressed in many ways [37, 10, 11, 9]. Gero has proposed Function-Behavior-Structure (FBS) framework, [10] which relates the function of a design object to its behavior and its structural descriptions. In Umeda's framework Function-Behavior-State [35], "state" refers to the physical structure of a design at a particular physical state during its behavioral process. The state of a physical structure is not addressed in Gero's framework but later introduced "situatedness"[12, 11] as an extension to his original FBS framework. In mechanical assembly design the function of mechanical object is dependent on the way that motion and forces are transmitted through the contact between pairs [9].

Here we represent function as *performance metric* [6] defining on set of intended behaviours. For the padlock example, the set of behaviors can include strength of the lock, (in terms of the maximum force it is expected to sustain),

ease of locking (resistance to jamming), and other aspects such as cost, ease of assembly, and also subjective aspects such as aesthetics. In mechanical assemblies, relating functions to structure often involves relative motion of the subparts - these may be captured using Configuration Space or *C-space*[24, 22]. However, computing the C-space for general motions remains an intractable problem [20]. Further, given a C-space, obtaining successful abstractions on it - i.e. segmenting the free-space into behaviorally significant regions - e.g., using topologically different contact types[29], 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, so that a performance metric can be defined for a given design instance. This is clearly true only for well-explored design problems, and implicit constraints can only be learned in such spaces. Design as search then involves determining the motion (behavior) as determined through kinematically constrained geometry interactions [23]. These motion constraints among the motion variables are in turn related to the design variables[21].

In the design search process, a set of structural variables γ define a specific structure in the structure space Γ. There may be several constraints between the structural variables in order to obtain a consistent design; these are reflected in the mappings that obtain the structural variables from the design variables, and the constraints on the design variables themselves. Together these variables and their constraints define the design space Ω.

In the padlock example, of Fig. 1, if we assume the width of the slot w and the thickness of the latch t to be part of the design vector $\underline{v}$, then for the part of the design space where $w < t$, the latch cannot enter into the slot, and thus though the shape is geometrically valid it is functionally infeasible. If we define a performance metric for ease of assembly that is dependent on the clearance $w - t$, then for different minimum acceptable levels of this performance metric, different regions in the design subspace w, t will become infeasible. Similarly, strength considerations, as well as other performance issues relating to the manufacturability, assembliability, etc. may further restrict the "good" functional zones or schemata known to the expert designer. It is our purpose to construct computational algorithms for exploring such patterns in the design space.

We consider now some specific function metrics for a) the padlock example introduced in Fig. 1, (b) a Slotted wheel mechanism, and (c) Slotted wheel with latch vertical shift.

4 Discovering Patterns of Functional Feasibility: Multi-Layer Perceptron

In this work, we use a simple multi-layer perceptron (MLP Neural Network) as our device for discovering functionally feasible regions (FFRs) in the design space. Neural networks are general-purpose function approximators that generalize statistical models from incomplete and uncertain behaviour information, and can be designed to adapt to the designer's requirements. Artificial Neu-

ral Networks (ANN) are computing systems made up of a number of simple, highly interconnected processing elements, which processes information by their dynamic state response to external inputs [3]. These ANNs map from a set of given input patterns to an associated set of known output values.

The MLP network (also called back-propagation) map is trained using a set of given input patterns for which the associated output values are known. Weighted sums of the inputs are propagated through one or more hidden layers [17], and the errors at the output are used to adjust the weights. In our case, design vector $\underline{v}$ is the input pattern and the known output is performance metric π. During the training process, the system uses the evaluations it has made of π on some of the points in the design space to clamp the output, and this is then used to adjust the weights of the internal connections to minimize the errors between the network input and the target output. ANNs, like many other machine learning systems, can thus be thought of as general purpose function approximation tools. In the figures of the results shown, the sample designs on which evaluation was performed are shown with a "+" (infeasible) and "$\bigtriangledown$" (feasible), and we have used different colors for feasible designs (π greater than some minimal π_0) from infeasible ones.

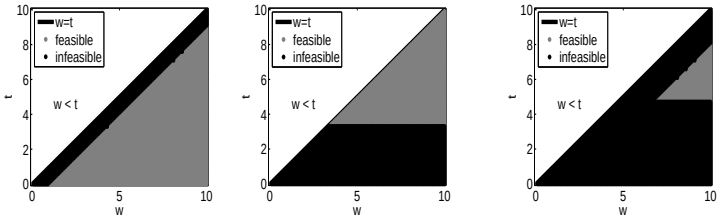
How the sequence of designs are selected for exploration is a critical issue in design, but this is not evaluated in this work; we use a random selection of a given number of designs as the training set for the ANN. The network architecture used has a single hidden layer and 50 neurons with single output parameter (π_{str}).

4.1 Example: FFRs for Padlock

To learn the FFRs for the padlock example, we sample many points $\underline{v}$ in Ω, each of which corresponds to an actual design. For each design instance, we evaluate the performance metrics. The performance metrics we consider here are $\pi_{str} = \frac{\sigma_Y \, b_l \, t^2}{6 \, l}$ and $\pi_{ease} = w - t$ [6] and see if the design is feasible or not. Earlier, in Fig. 2, we have presented such learned implicit constraints for a slotted wheel mechanism.

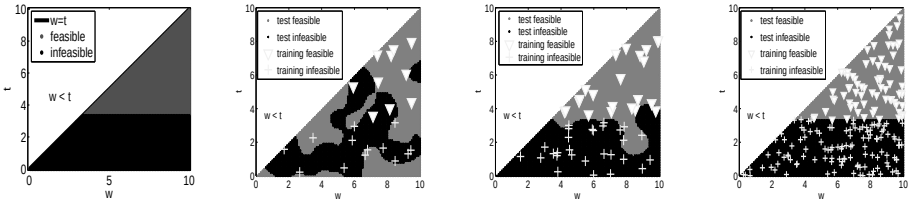
By applying different standards of acceptability for the performance metrics to different instantiations in the design space, one can obtain different bounds to the FFRs. The performance measure for ease of locking is negative for the region above the $w = t$ line, and this region thus becomes permanently infeasible. For any positive value of π_{ease}, the line shifts more to the right. Similarly, for any given level of π_{str}, the material and other dimensions remaining the same, the strength rises proportionally to t^2. Combining these results in the functionally feasible space shifting to the right and up. At the same time, since high w and high t also imply higher values for other dimensions, cost and weight considerations are likely to squeeze the feasible region more to the bottom and left. This would eventually produce different viable zones for different levels of padlock function.

While for the padlock, this analysis is quite straightforward, clearly there are no limits to the complexity of the performance metrics. Even mechanisms



(a) $\pi_{ease} > 1mm$ (b)$\pi_{str} > 500N$ (c) $\pi_{str} > 1000N$ & $\pi_{ease} > 2mm$

Fig. 3. *FFRs (Functionally Feasible Regions) in padlock design space.* Even in the region $w > t$, some regions close to the boundary may be ruled out at certain levels of the Ease of Locking metric π_{ease} (a), while low values of latch thickness t are likely to be ruled out by strength considerations in π_{str} (b). Combining π_{ease} and π_{str} metrics superposes these constraints; (c) shows the resulting FFR for double the metrics of (a) and (b).



(a) actual *FFR* (b) learned with 20 instances (c) learned with 80 instances (d) learned with 200 instances

Fig. 4. *Learning through experience for padlock.* The implicit constraint on the w, t subspace of the Design Space is learned under the functional specification $\pi_{str} > 500N$. The exact functional constraint is shown in the leftmost figure. An implicit constraint learned by the system based on exploring 20 design instances is shown next. The decision surface learned after 80 trials seems more convoluted, but this is because the sample space is an inadequate model for the actual pattern. By 200 instances, the learned model is quite good.

exhibiting only slightly more complex motion behaviors will evolve in less obvious ways. In the following section we explain the *slotted wheel mechanism* and also a variant with a vertical shift in the latch axis which in not uncommon in the actual working mechanisms.

5 Slotted Wheel Mechanism

The constraints discovered in the padlock example seem extremely straightforward, of the kind that even a novice designer may discover within a few trials, along with the causal process underlying this constraint. Thus for the human therefore these FFRs become explicit, leading to a process of chunking and re-

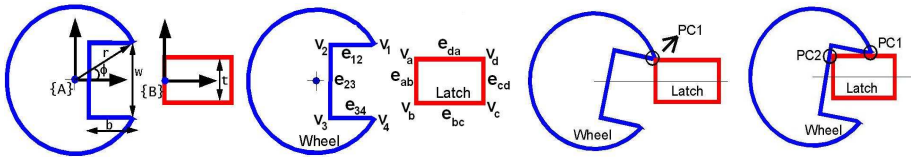


Fig. 5. (a) Design Parameters for Slotted-Wheel mechanism: radius r, width w, breadth b, thickness of the latch t and $\phi = \sin^{-1}(\frac{w}{2r})$. ϕ is a dependent parameter on design parameters. (b) Its geometric elements' representation (c) $CF =< PC_1 >$ is a contact formation having single principal contact (d) $CF =< PC_1, PC_2 >$ is a contact formation which has two principal contacts.

structuring the design space. For a machine, the constraint would remain implicit unless it has access to further domain knowledge.

Even with this very simple methodology however, we may encounter patterns of constraints that are not very obvious at all. Let us now consider a mechanism only a little more complex than the padlock - a lock mechanism with a rotational slotted wheel and a translational latch (Fig.5(a)). The rotation of the wheel will be locked when the latch assembles into the slot of the wheel.

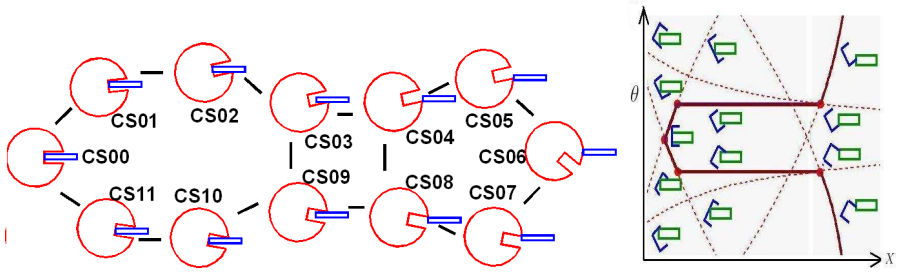


Fig. 6. *Contact State Graph: (a) Each node represents a contact state and the edge represents the transition of neighboring contact states (b) C-space with different contact states*

Each contact curve in Fig. 6(b) satisfies a contact equation $f(x, \theta, \underline{v}) = 0$ which represents the function of each contact and it is result of solving contact constraints among the touching geometric elements. The shape of the curves depends upon the shape of the contacting geometric elements and type of contact. If the functional specification is to reach the contact state CS00, (PC $= < e_{23}, e_{ab} >$) then it is a point in the contact space at $\theta = 0$.

A key aspect of design is reformulating the problem itself. Part of this process may be changing the design variables, e.g. by adding a new variable. Here, for the slotted wheel, we shall now explore how such a design change may add a

variable for a shift in the latch so that it is no longer along the axis of the rotating wheel.

6 Emergence and Design Change

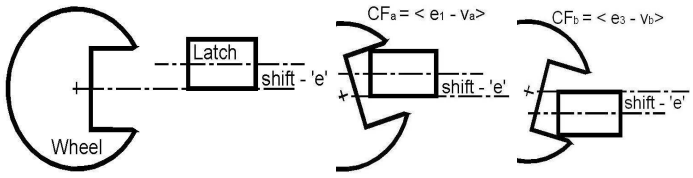
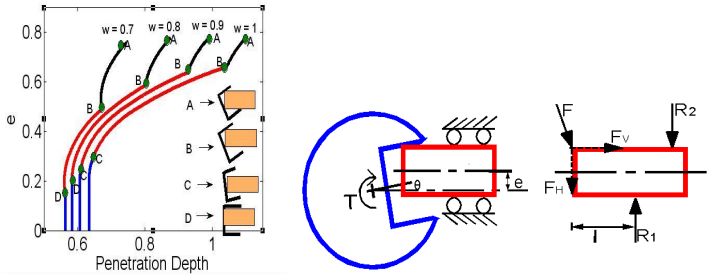


Fig. 7. *Vertical Shift in Latch* (a) Note that in addition to the contact formations of Fig. 6(a) two new contact formations are observed for shifts that are positive (CF_a,Fig.(b)), or negative (CF_b, Fig.(c)).

We next consider incorporating a new design variable as a design change. Based the designer's exploration of the behaviour of the slotted wheel lock, it becomes clear that if the latch were to be shifted so that the rotation axis of the barrel is below the latch, e.g. as a result of some constraint in the installation, then it would affect the quality of the penetration, and hence the strength, profoundly. Exploring this possibility necessitates adding this shift as a new design variable. The shift may also arise as a result of manufacturing inaccuracies, and the same analysis is relevant to the modeling of tolerances.Consider the slotted-wheel lock mechanism where the latch translational axis is displaced from the slotted wheel center by a distance e(Fig. 7a). In the new design space Ω, the design vector $\underline{v}$ will be $\{r, w, t, b, e\}$. With this design change we find out the feasible and infeasible regions in the design sub space (w,t) based on the functional constraints.The mechanism is in the locked state when the latch is inserted into the slot of the rotating wheel. As e becomes more than $\frac{w-t}{2}$, the degree of penetration varies considerably, and this affects its function. Here, the strength of the lock may be measured in terms of the maximum torque that the barrel can withstand. The contact state graph(6)(a) also changes; two new contact nodes(7(b),(c)), emerge with the addition of vertical shift e; and this causes the mechanism to experience new set of behaviors at certain e values.

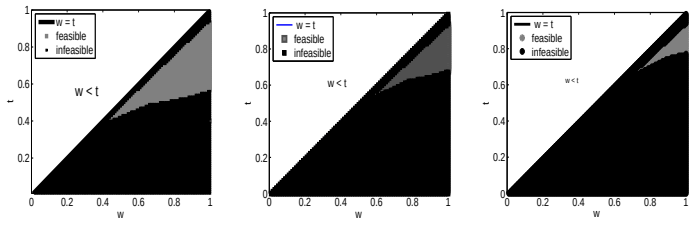
6.1 Performance Metrics

Maximum Penetration The penetration depth (PD) varies with the change in the vertical shift (e) values. The maximum penetration in the different ranges of e is shown 8(a).



(a) (b) (c)

Fig. 8. (a) Maximum penetration depth for different ranges of axial shifts. **A,B,C** and **D** are the different contact states at which maximum penetration is possible for a design variable $\frac{t}{r} : 0.4, \frac{b}{r} : 0.5$.(b) A typical contact configuration; here the maximum contact force is determined given the maximum torque that the slotted wheel axis is expected to bear. This maximum torque τ_{max} is equal to the contact force times a moment arm $d(X, \theta)$ (Eq.1)that depends on the penetration depth of the latch.(c) Free body diagram of the mechanism.



(d)$\tau > 6$ (e)$\tau > 10$ (f) $\tau > 15$

Fig. 9. *Evolving constraints based on performance measures* π_{str}. These three figures present various feasible regions (FFRs) in the w, t design subspace, for differing constraints on π_{str} or the maximum torque(τ_{max}). (Slotted wheel mechanism, axial shift $\frac{e}{r} = 0.3$ and $\frac{b}{r} = 0.5$)

Strength: Maximum Torque If we are to evaluate this lock based on its strength, an obvious measure would be the maximum torque τ_{max} it can support. This torque will vary in different contact states, and we may compute it for the maximum penetration, as discussed in section 6.1. Let us consider a contact state **B** as shown in Fig. 8(a). At this contact state the latch in the mechanism is considered as a simple supported, for which the free body diagram is shown in Fig. 8(b). Also, Eq. 1 gives the relation for the maximum strength of the latch which can withstand against the desired torque τ, where $w_1 = \frac{w}{r}$, $t_1 = \frac{t}{r}$, and $e_1 = \frac{e}{r}$.

Here we obtain the following constraints based on Fig. 8(b). The maximum torque τ_{max} that can be supported is determined by the contact force F and its moment arm; this contact force is in turn limited by the latch strength.

$$\begin{aligned} F &= \frac{\sigma_Y b_{latch}(t_1)^2}{6l} \\ \tau_{max} &< -F\cos\theta\ d(X,\theta) - F\sin\theta\ (e_1 + \frac{t_1}{2}) \\ d(X,\theta) &= \frac{2e_1\cos\theta + t_1\cos\theta - w_1}{2\sin\theta} \end{aligned} \tag{1}$$

where $d(X,\theta)$ is the moment arm for the vertical component of the contact force F.

The performance metric π_{str} for the slotted wheel may be simply set to be τ_{max} as given in the equations above. Now, asserting different levels of acceptability for π_{str} results in constraints involving different parts of the design space, as seen in Fig. 9. Designs with thin latches (low values of t)are clearly rejected by the requirement for strength, but owing to the penetration depth being related to theta, when e is a significant compared to t, high strength also requires higher slot width w's.

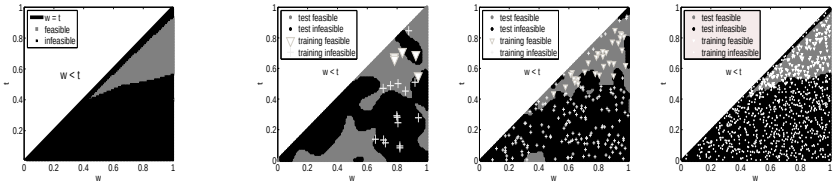
7 Design Experience and the Quality of Implicit Constraints

Although we are able to learn some types of constraints from the function evaluations across the design space, there are many questions that remain. For one, what is the nature of the convergence of the learning function? Are the constraints learned indeed reasonable? How do these constraints depend on the design experience? How can the next design exploration be controlled so as to maximize the effectiveness of the learning?

Human designers clearly improve in the quality of their immediate design decisions, as they gain experience. In our case, we can see a similar phenomenon that can be understood in terms of learning machines. Any function generalizer that learns from a set of training data is sensitive to the size and distribution of its data. Clearly, as more samples from the design space become available as training data, the quality of the learned function will improve.

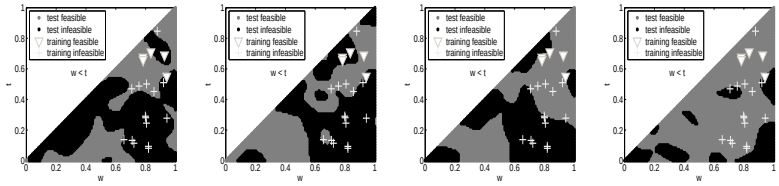
We next explore how the quality of learning changes depending on the degree of exploration in the design space. We consider a learning system that learns from a small (20), medium (200) and large (450) number of samples (Figure 10) for the slotted wheel with $\underline{v} = \{\frac{b}{r} = 0.5, \frac{e}{r} = 0.3\}$. Here we attempt to learn the FFRs (feasible region constraint) for the performance constraint $\tau_{max} > 6$.

Given that we can compute the exact FFRs that obtain in the underlying design space, we can then compute the errors in each case in terms of the percentage of false responses in the resulting implicit constraint. We define error



$\tau > 6$ for actual functional constraint | $\tau > 6$ learning with 20 error = 0.812 | $\tau > 6$ learning with 200 error = 0.2312 | $\tau > 6$ learning with 450 error = 0.0921

Fig. 10. *Quality of Implicit Constraints learned improves with experience.* The design space shows the learning after experiencing 20,200, and 450 design instances. The actual functional constraint is shown in the leftmost figure (a).



error = 0.4602 | error =0.5450 | error = 0.6717 | error = 1.2402

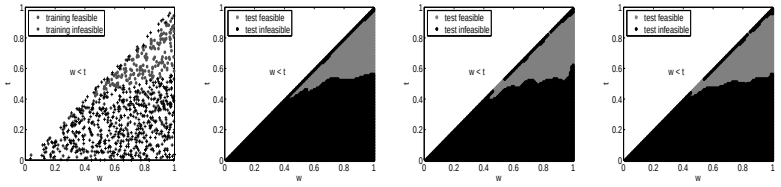
Fig. 11. *Learning through different experiences for the same design instances.* The design space shows the learning with 20 design instances for the torque $\tau > 6$.

as the false fraction - i.e. those design instances that are actually infeasible but show up as accepted in the learned function (False Positives, *FP*), and those that are feasible but are rejected ((False Negatives, *FN*). FP can be visualized as the part of the reject area in Fig 10a that is marked as reject by the learned function, and FN is the part of the accept area marked as reject. As expected, large swathes of the design space are marked falsely with 20 samples, and the accuracy improves or error decreases considerably with 400.

However, it is observed that the learning does not improve monotonically - i.e., occasionally the implicit constraint learned after evaluating 50 design instances may be poorer than the pattern with only 10 instances. This is because of the wide variability in the early stages of learning. This leads us to issues of convergence in learning the function.

7.1 Convergence: Variability of Learned Function

Statistical methods are commonly used in the development of empirical relationship between various interacting factors. Since it is extremely difficult to model the convergence of neural networks, we explore the issue of convergence empirically. We consider learning from the identical training data, but with random variations in the network weights.



error = 0.0301 error =0.0421 error = 0.0487 error = 0.0503

Fig. 12. *Learning through different experiences for the same design instances.* The design space shows the learning with 450 design instances for the torque $\tau > 6$. The sample training input is shown in the leftmost figure.

Fig. 11 shows the implicit constraints, learned on the w, t subspace of the Design Space under the same functional specification $\tau > 6$ considered for the 20 design instances with the same sample set as input. Here we report the w, t subspace, holding the other parameters as $\{\frac{b}{r} = 0.5, \frac{e}{r} = 0.3\}$. It is observed that the learned pattern varies remarkably for a sample size of 20, but is markedly less when the training set is 400 (Fig. 12). This indicates that a high degree of exploration in the design space is critical to learning adequate constraint functions. We report the error on each learned constraint pattern in terms of the fraction of total false reportages.

Data points	Mean	Standard Deviation
20	0.9210	0.2147
100	0.3213	0.1200
250	0.1949	0.0162
400	0.1189	0.00110

Table 1. *Function quality with increasing exposure.* The mean error of the learned constraint function, as well as its variance, decreases significantly as the number of training instances increase from 20 to 400. This data is based on 25 independent runs for each sample set, for the slotted wheel mechanism with $e = 0.3r$.

Variability is measured in terms of standard deviation on the error. Table 1 shows the mean and standard deviations of the total different 25 trials conducted on same set of design instances for different data set. With the increase in the training input data the mean error and variation is reduced. This replicates human experience; in the early stages, the designer is beginning to form a function but this its quality varies considerably. Also, as the function improves, the system is itself able to measure the error rate by considering how subsequent design explorations fare with this function; thus the rate of convergence, and the sample size needed for "adequate" exposure can itself be determined empirically.

Clearly, for more complex design spaces with higher dimensionality, this may be a large number, and finding ways to reduce the dimensionality and therefore the adequate sample size, would be an important challenge.

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. One aspect of the human experience is that each sketch represents a large-ish region in design space, and not just a single design. More importantly, the human designer is using much more domain knowledge in terms of relations between the underlying parameters and their effect on the function, whereas the computational process identified here is blind to such functional inter-relations.

8 Conclusion and Future Work

In this paper, we have presented a computational process based on theories of emergence to imitate an expert designer's ability to learn implicit functions that indicate regions of high feasibility in the design space. We show how computational algorithms can also identify such regions, and also present results, using the simple mechanism of a lock, on how the convergence of such algorithms may be evaluated by using the quality of the functions even as exploration proceeds.

We have also explored the nature of design change by introducing a new design variable $e(\text{shift})$ into the design space, and show how even this small change results in unpredictable changes in behaviour, and how these in turn affect the functional requirements. While the examples involve very simple designs, there are no theoretical limits on the complexity of the design spaces on which such functions can be built. However, in each situation, we require that the function be well understood and quantifiable in terms of performance metrics - and at this point, the human user is required to specify these. Also, 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. Even as presented here, the system is possibly a great help to novice designers, who may observer in the resulting patterns some explanation in terms of the underlying parameters, which would add to her design knowledge in this domain.

Discovering such high-feasibility regions and defining implicit constraints based on them is important for abstracting other patterns of behaviour, and in particular, for seeding the chunking process, whereby the representation of the design itself changes, typically accompanied by dimensionality reduction. Another area of exploration in the future is the possibility of starting with a simple physics knowledge-base and developing representations that capture such functional relations among the parameters based on multiple design experiences for different classes of objects.

Acknowledgements

This research has been supported by project "Research I Foundation", I I T Kanpur, India. We would like to thank the anonymous reviewers for their valuable recommendations and comments.

References

1. Ahmed, S., Wallace, K. M., and Blessing, L. T. Understanding the differences between how novice and experienced designers approach design tasks. *Research in Engineering Design 14*, 1 (February 2003), 1–11.
2. Athavankar, U. A. Mental imagery as a design tool. In *Thirteenth European Meeting on Cybernetics and Systems Research, EMCSR 96* (1996), pp. 1–10.
3. Caudill, M. Neural networks primer. *AI Expert Part I* (December 1987), 46–52.
4. Chalmers, D. J. Strong and weak emergence, 2006.
5. Cross, N. Expertise in design:an overview. *Design Studies 25*, 5 (September 2004), 427–441.
6. Dabbeeru, M. M., and Mukerjee, A. Functional design for part families of mechanical assemblies. G.Hu, Ed., no. ISBN 978-1-880843-65-9, ISCA, ISCA, pp. 263–268.
7. Eckert, C., Kelly, I., and Stacey, M. Interactive generative systems for conceptual design: An empirical perspective. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13* (1999), 303–320.
8. Ericsson, K. A., and Lehmann, A. C. Expert and exceptional performance: evidence on maximal adaptations on task constraints. *Annual Review of Psychology 47* (1996), 273–305.
9. Faltings, B. A symbolic approach to qualitative kinematics. *Artificial Intelligence 56*, 2-3 (1992), 139–170.
10. Gero, J. S. Design prototypes: A knowledge representation scheme for design. *AI Magazine 4*, 11 (winter 1990), 26–36.
11. Gero, J. S., and Kannengiesser, U. The situated function-behaviour-structure framework. *Design Studies 4*, 25 (2004), 373–391.
12. Gero, J. S., and Reffat, R. M. Multiple representations as a platform for situated learning systems in designing. *Knowledge-Based Systems 14* (2001), 337–351.
13. Goel, V. *Sketches of thought*. MIT Press, Cambridge18 Goodman, N Languages of art Bobbs-Merrill, Indianapolis, 1995.
14. Goldschmidt, G. The backtalk of self-generated sketches. *Design Issues 19*, 1 (Winter 2003), 72 – 88.
15. Groot, A. D. *Thought and Choise in Chess*. 1946/1978.
16. Gross, M. D. *Design as Exploring Constraints*. PhD thesis, Massachusetts Institute of Technology, February 1986.
17. Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.
18. Hoffmann, C. M., and Joan-Arinyo, R. A brief on constraint solving. *Computer-Aided Design and Applications 2*, 5 (June 2005).
19. Janssen, P. H. The role of preconceptions in design: Some implications for the development of computational design tools. In *Design Computing and Cognition '06* (2006), J. S. Gero, Ed., Springer Verlag, pp. 365–383.

20. Ji, X., and Xiao, J. Planning motion compliant to complex contact states. *International Journal of Robotics Research 20*, 6 (June 2001), 446–465.
21. Joskowicz, L., and Addanki, S. From kinematics to shape : An approach to innovative design. In *Proceedings of the AAAI* (1988), St.Paul.
22. Joskowicz, L., and Sacks, E. Computational kinematics. *Artificial Intelligence 51*, 1-3 (1991), 381–416.
23. Joskowicz, L., and Sacks, E. Configuration space computation for mechanism design. In *Robotics and Automation* (San Diego,CA, USA, May 1994), . I. I. C. o. Proceedings., Ed., vol. 2, pp. 1080–1087.
24. Latombe, J. C. *Robot Motion Planning*, first ed. Kluwer Academic Publishers,Boston, 1991.
25. Lawson, B. Schemata, gambits and precedent: some factors in design expertise. *Design Studies: Expertise in Design 25*, 5 (September 2004), 443–457.
26. Lin, V. C., Gossard, D. C., and Light, R. A. Variational geometry in computer aided design. *Computer Graphics 15*, 3 (1981).
27. Maher, M. L., and Tang, H.-H. Co-evolution as a computational and cognitive model of design. *Research in Engineering Design 14*, 1 (February 2003), 47–63.
28. Mark D. Gross, Stephen M. Ervin, J. A. A., and Fleisher, A. Constraints: Knowledge representation in design. *Design Studies 9*, 3 (July 1988), 133–143.
29. Mukerjee, A., and Bhatia. A qualitative discretization for two-body contacts. In *Proc. of the 14th IJCAI* (Montreal, 1995), vol. 1, pp. 915–921.
30. Poon, J., and Maher, M. L. Emergent behaviour in co-evolutionary design. Kluwer Academia,Dordrecht, pp. 703–722.
31. Saunders, R., and Gero, J. S. The importance of being emergent. In *Proceedings of Artificial Intelligence in Design* (2000).
32. Sutherland. *I 'Sketchpad - a graphical man-machine interface'*. PhD thesis, 1963.
33. Suwa, M., Gero, J., and Purcell, T. Unexpected discoveries and s-invention of design requirements: important vehicles for a design process. *Design Studies 21*, 6 (November 2000), 539–567.
34. Suwa, M., and Tversky, B. What do architects and students perceive in their design sketches ? a protocol analysis. *Design Studies 18* (1997), 385–403.
35. Umeda, Y., and Tomiyama, T. Functional reasoning in design. *IEEE Intelligent Systems and Their Applications 12* (March 1997).
36. Verstijnen, I., van Leeuwen, C., Goldschmidt, G., Hamel, R., and Hennessey, J. Sketching and creative discovery. *Design Studies 19*, 4 (October 1998), 519–546.
37. Wolter, J., and Chandrasekaran, P. A concept for a constraint-based representation of functional and geometric design knowledge. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications* (1991), pp. 409–418.