Computer-Aided Design 44 (2012) 901–915

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Grounded discovery of symbols as concept–language pairs

Amitabha Mukerjee [a], Madan Mohan Dabbeeru [b,*]

[a] Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India

[b] Center for Robotics, Indian Institute of Technology Kanpur, Uttarpradesh-208016, India

ARTICLE INFO

Keywords:
Cognition in design
Design symbols
Chunking
Language learning

ABSTRACT

In human designer usage, symbols have a rich semantics, *grounded* on experience, which permits flexible usage — e.g. design ideation is improved by meanings triggered by contrastive words. In computational usage however, symbols are syntactic tokens whose semantics is mostly left to the implementation, resulting in brittle failures in many knowledge-based systems. Here we ask if one may define symbols in computational design as {label,meaning} pairs, as opposed to merely the label. We consider three questions that must be answered to bootstrap a symbol learning process: (a) which concepts are most relevant in a given domain, (b) how to define the semantics of such symbols, and (c) how to learn labels for these so as to form a grounded symbol. We propose that relevant symbols may be discovered by learning patterns of functional viability. The stable patterns are information-conserving codes, also called *chunks* in cognitive science, which relate to the process of acquiring expertise in humans. Regions of a design space that contain functionally superior designs can be mapped to a lower-dimensional manifold; the inter-relations of the design variables discovered thus constitute the chunks. Using these as the initial semantics for symbols, we show how the system can acquire labels for them by communicating with human designers. We demonstrate the first steps in this process in our *baby designer* approach, by learning two early grounded symbols, TIGHT and LOOSE.

1. Symbols in design

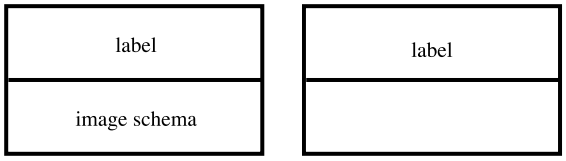
The knowledge that designers have is private, and not easy to access. What is available externally are what designers say about their design, or other depictions such as sketches. These external models of design use *symbols*, e.g. a word such as "tolerance" or a sign such as  . The word symbol means "something that stands for or denotes something else" (OED, sense 2a). However, in formal use, it is merely a token used in an algebraic expression, and what is "stands for" is not part of the model. In constructing formal models of design, symbols are often treated as logical predicates; thus, one symbol is defined in terms of some other symbols and so on. On the other hand, a designer's mental concepts of symbols are grounded in experience, and encode context-dependent variations and permit flexible usage.

This work proposes a mechanism for defining symbols as cognitive entities, determined by experience, as opposed to defining them manually. In this view, symbols constitute a tight coupling between a linguistic or graphical or gestural *label*, and an associated concept or idea, the semantics—referred to in cognitive science as its *image schema* [1]. The label is sometimes called the *phonological pole* of the symbol, and the image schema its *semantic pole* [2]. These label–meaning pairs follow an agreed convention within a group, so that others may recognize the image schema when exposed to the label. Thus, the symbol evokes a rich suite of thoughts— that it is an electrical device, that its current varies linearly with voltage, that it gets hot over use, and so on. The more experienced a designer, the more richly nuanced this semantics.

What lends flexibility to symbol usage among humans is the availability of this rich semantics base; without a semantics, or even with a sharply defined (boolean predicate) notion of semantics, symbol usage becomes very unstable. As a recent example, we may consider a design repository proposal [3] for suggesting similar design solutions during the ideation stage. Users may search the repository by typing in keywords. However, the design repository fails to produce any results when the user searches with the term "tank" instead of a standard term "reservoir" [4]. One solution suggested for this problem is to arrive at a "standardized vocabulary" of design [3,5]. This proposes to create an enormous inventory of symbols, organized in a relational graph (called an ontology or a taxonomy) to fit all kinds of design problems. However, this is unlikely to be effective, since very few symbols mean exactly the same in all contexts.

* Corresponding author. Tel.: +91 988 966 7671.
E-mail addresses: amit@cse.iitk.ac.in (A. Mukerjee), madan.dabbeeru@gmail.com (M.M. Dabbeeru).
URLs: http://www.cse.iitk.ac.in/users/amit (A. Mukerjee), http://terpconnect.umd.edu/~mmd (M.M. Dabbeeru).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.06.004



(a) Human symbols. (b) Formal symbols.

Fig. 1. (a) Human symbols are a tight coupling between a label and a meaning or "image schema". (b) Formal computer symbols are just the term or label. How such symbols map to a design domain is not part of the theory.

Deprived of a mechanism for representing the semantic–pragmatic context, any attempts to convert user input into a standardized set of synonymous terms is unlikely to be effective. Also, while expanding the system, different humans working on it would define overlapping definitions and hidden conflicts.

Instead, we argue for a grounded symbol learning system. The mechanism proposed works in three steps: (a) determining which aspects of a system deserve symbol-hood, (b) propose a mechanism for representing the semantics of these proto-concepts, and (c) learn labels for these based on interacting with other users.

A close coupling between the semantics and the term is also reflected in a number of processes within design. For example, a number of studies on concept generation have demonstrated a role for verbal cues — e.g. contrastive words [6–8]. Clearly, the human creative processes are working on the semantics level to trigger a complex web of inter-related concepts, enabling designers to come up with divergent ideas. These types of interactions have led to an argument for a semantic relation between design concepts and the language of design [9]. This cognitive view of symbols is close to our approach, and differ substantially from formal computational theory which has been adapted for use in knowledge-based design [5,10]. Formal symbols are merely syntactic place-holders, to be operated upon based on rules of formal grammar. Where an implementation or a demonstration has to be produced, the symbols are related to the variables of design using a sharply defined (true–false) function. The structure of this function is not a part of the theory but is defined by the programmer at demo time. The result is that quite often, the system works for the demo problems, but when given a new problem, the rules turn out to require extensive re-working, a problem known as *brittleness* in the Artificial Intelligence literature [11].

Thus, any attempt to understand the language of design, whether from a theory of design perspective, or from a computational (implementational) viewpoint, must carefully consider, from the very early stages, the semantics of its symbols. This is our main objective — to try to develop design symbols in terms of label–meaning associations, as in human symbols (Fig. 1).

There are three main difficulties in doing this: (a) the semantics (or image schema) for a symbol, even in the simplest situations, is often not known even to the user of the symbol; (b) what meaning a symbol has is crucially dependent on context (as in the "tank" example above) and (c) meaning is *dynamic*—i.e. the semantic structures undergo continuous change, small and large, with experience. This last point highlights the utility (and difficulty) of symbols—unlike static formal systems, every time a symbol is used or observed in a phrase, the image schema must also be updated, in the sense that some associations are strengthened or weakened or created or deleted.

Given these difficulties and the novel nature of this enterprise in the design context, our objective in this initial work is very limited. We attempt to discover label–meaning pairs for only the very *initial* symbols as first encountered by a system, and not to the full dynamic sense of "symbol" as understood in design.

Here the initial image schema is represented in terms of variables from a problem domain. In this initial stage, we consider the process whereby an image schema is discovered, and a label for it learned by associating the new concept with units from a language. During the initial stages being considered here, the image schema emerges independently of the label, based on functional aspects of the task domain. Thus, meaning emerges first and language is mapped to it later, as is being increasingly asserted in infant cognition [12].

We note several differences from other attempts to model domain knowledge in CAD systems. First, our set of symbols (the *lexicon*) is not pre-determined but are *discovered*. Discovery implies that the lexicon is more likely to cover the kinds of distinctions needed in practice. A concept TOLERANCE, may emerge because we find that in order to maintain TIGHT fits, we need to hold each part to a narrow range of dimensions. Symbols are discovered when such patterns are repeatedly observed to be relevant to function. For example, an engineer may find it important to distinguish fits with very small clearances that allows guided motion (say, RUNNING FIT) vs. fits that can transmit high torques without slipping (PRESS FIT) — concepts that emerge in the domain of mechanical assembly. For an electrical engineer designing a power socket, similar dimensional variations have a different symbolic expression. Thus, function plays a key role in determining the lexicon of symbols.

We call our algorithm that follows this process and discovers symbols a *Baby Designer*. The name reflects the very elementary level of learning achieved by the system. The attempt to discover potential symbols starts by first identifying regions of the design space containing many "good" solutions; good solutions are determined by how well the design is doing on a set of performance measures — the *performance metrics*.[1] The regions with good designs will be paid more attention (they will be *salient*)—We call such regions the Functionally Feasible Regions (FFRs).

If there is a pattern – an inter-relation between the variables – that holds over the FFR, then it is likely to become encoded as an image schema. These inter-relations, known as *chunks* in the cognitive literature, constitute information-preserving abstractions on the design space — i.e. they describe compactly many constraints that must hold between variables. For example, when a beam width increases, its thickness is likely to increase as well, for otherwise the strength properties would be unbalanced. The two variables of width and thickness may be reduced to a one-dimensional chunk (a manifold) for the space of successful designs. These inter-relations are thought to result in structures in long term memory called chunks, which are characteristic of the mental models of experts across many domains from chess to physics to fire-fighting [13]. The more information-compacting the chunk, the more it relates "to underlying principles, rather than focusing on the surface features of problems" [14].

In this work, we propose that the initial image schemas for symbols are these chunks which compactly capture the functionally relevant aspects of a good design. For example, a schema that recognizes TIGHT fits may emerge from multiple insertion experiences, as that class of fits where the inserted object is more likely to wedge into the hole. It is only later that such a model of meaning may get attached to a label such as "tight".

In this work, we shall use the notation of small capitals for the image schema or semantics (e.g. TIGHT), the string in quotes for its label ("tight"), and square brackets for the symbol as a whole, [TIGHT].

[1] Note that the "performance metric" does not need to be a mathematical metric. At the very least, we require transitivity — i.e. if A is preferred over B and B over C then A should be preferred over C. Thus, any stable ranking would suffice. Given such a measure, the approach for determining regions of good functionality is outlined in Section 5.