HETEROGENEOUS INFERENCE IN DESIGN

Varol Akman

BILKENT UNIVERSITY

Department of Computer Engineering
and
Information Science

Technical Report BU-CEIS-93-14

Heterogeneous Inference in Design

Varol Akman *

Abstract

For those of us involved in the attempt to construct formal models and environments in which the *world of design* can be subjected to scientific experimentation, the *raison d'être* of logic has been rather well-understood. The aim of this position paper is not really to challenge this view but rather to complement and extend it. Specifically, we discuss why *heterogeneous inference*—inference that proceeds from information represented in *more* than one form—is crucial.

In order to make adept, temporal comments,
an architecture machine must have a certain
basic understanding of qualities. Though at first
primitive, this qualitative appreciation itself
would evolve within a value system that is very
personal, between a man and a machine.
NICHOLAS NEGROPONTE (1970)

1 Motivation

In 1963, John McCarthy made an interesting remark. He expected that *the relationship between computation and mathematical logic will be as fruitful in the next century as that between physics and analysis in the last.* What makes the above observation a truly visionary one is that today McCarthy's expectation is largely fulfilled and the next century is still ten years ahead. Logic programming is now a household name and there is hardly an area of computing where logical formalisms and theories are not dominant.

Encouraged by this remarkable development, a main motivation for our work has been to investigate the use of mathematical logic in design. It should immediately be confessed that doing so is swimming in dangerous waters because design is ordinarily associated with mysterious, at best intricate, mechanisms which involve a variety of high-level cognitive processes [18, 23]. In other words, it is commonly thought to be difficult, if not impossible, to pin down precisely the formal aspects of design. Yet,

*Department of Computer Engineering and Information Sciences, Bilkent University, Bilkent, 06533 Ankara, Turkey.

our *approach* to deal with this puzzling nature of design is decidedly formal. We think that only through a theoretical framework can we arrive at a scientific theory of design.

If logic is one important facet of this paper, the other is naive physics. We believe that the emerging theories of naive physics can be used to obtain good representations of design artifacts and to reason about device behavior. It is noted that, following [12], *we usually do not distinguish among the areas of naive physics, common sense theory, qualitative simulation, qualitative reasoning, and qualitative physics.*

It has long been argued that a general theory of *devices* is essential. To quote Minsky [16]:

> The classical idea of a simple machine—lever, wheel, inclined plane, etc.—does not capture the spirit of what is involved in today's machines because it doesn't help understand anything except the transmission of force. We cannot explain in those terms even some parts of clockwork, such as the ratchet (an information-storage device) or the spring (an energy-storage device).

It is our belief that naive physics is a fine candidate to provide this general theory.

1.1 The Need for Intelligent CAD

CAD systems capable of giving intelligent support to designers will be important in the future. As technologies advance, only those systems which incorporate advanced reasoning capabilities will be able to deal with the complexity arising from the management of large quantities of design data and design experience. Intelligent CAD systems will be sophisticated *design environments* to support a designer's intellectual activities with integrated design knowledge. This certainly requires that some, not necessarily perfect, concept of design is already available and this is precisely the aim of this paper, viz., to present a view of design as particularly amenable to automation. Understanding the nature of design processes and obtaining good representations of design artifacts in an *evolutionary* design framework are, in our view, the most urgent goals of research in intelligent CAD. The progress towards success will be measured by how much these efforts will be different from pragmatic yet obviously significant issues such as geometric reasoning, geometric features, solid modeling, finite elements, optimization, etc. which are aimed at *realizations* of CAD systems.

Remark. Since design activities and philosophy for CAD systems are dependent on the target area, these problems can be discussed only vis-à-vis a particular field. We take machine design as the target area.

1.2 The Nature of Design

Let us start with the following definitions which are taken from *The New Merriam-Webster Pocket Dictionary*, 49th printing (October 1972):

> **de·sign** \di-'zīn\ *vb* **1 :** to conceive and plan out in the mind; *also* **:** DEVOTE, CONSIGN **2 :** INTEND **3 :** to devise for a specific function or end **4 :** to make a pattern or sketch of **5 :** to conceive and draw the plans for <∼ an airplane>—**de·sign·er** *n*
>
> **design** *n* **1 :** a mental project or scheme **:** PLAN **2 :** a particular purpose **:** deliberate planning **3 :** a secret project or scheme **:** PLOT **4** *pl***:** aggressive or evil intent—used with on or against **5 :** a preliminary sketch or plan **:** DELINEATION **6 :** an underlying scheme that governs functioning, developing, or unfolding **:** MOTIF **7 :** the arrangement of elements that make up a structure or a work of art **8 :** a decorative pattern

In the light of so many meanings attached to the word *design* (even by a pocket-size dictionary), it is probably redundant to say that the present paper will not cover all aspects of design. Rather, it will explain a certain, hopefully original, outlook which sees design as an intellectual activity and thus tries to embody AI techniques to automate it. Before we get into that, we want to cite a couple of observations regarding the nature of design. The following is due to Schön [21]:

> Designers are usually unable to say what they know, to put their special skills and understanding into words. On the rare occasions when they try to do so, their descriptions tend to be partial and mistaken: myths rather than accurate accounts of practice. Yet their actual designing seems to reveal a great deal of intelligence. How, then, if we reserve 'knowledge' for what can be made explicit, are we to explain what designers 'know'? And if, on the contrary, we recognize designers' *tacit* knowledge, what shall we say about the ways in which they *hold* it, or get access to it when they need it?

The next quote is due to Newell and Simon [20]:

> *Designing a Machine.* Take as U the set of all possible parameter values for a machine design; take as G the subset of parameter values that: (1) satisfy the design specifications, and (2) meet certain criteria of cost minimization. For cases of practical interest, the set U will be immense and hence will have to be explored in somewhat systematic fashion. In early stages of the search, for example, particular design variables may be bounded, or even fixed, prior to establishing limits on the other variables. If the priorities are fairly definite, then the search tree will have hierarchic properties—the branchings at different stages referring to different classes of design variables.

1.3 Heterogeneous Inference

Especially in the early stages of a design process *heterogeneous inference*—inference that proceeds from information represented in *more* than one form—is crucial. This idea is in the precise spirit of Barwise and Perry's *situation semantics* which sees valid inference not as a relation between sentences that preserves truth but as a *situated activity* whose aim is the extraction of information from a situation, information relevant to the person who extracts it [4, 5]. Thus, the key insight of our paper is that design is a situated activity—an activity which is carried out by an intelligent agent (designer) in a rich environment that can be utilized in many ways [1, 2].

Design programs assist a designer in specifying an *artifact*, e.g., a house, a machine, or an electronic circuit. The assistance can range from mere registration of the design results to analyzing the proper functioning of the designed object, maybe through simulation. More advanced forms of assistance include problem-solving activities such as optimization, routing, and even suggesting a solution based on the given specifications. The latter activities become more dominant when the assistance could be extended to the earlier and more difficult phases of design.

The design process can be defined as *transforming* a set of specifications into a set of attributed objects which together perform as required by the specifications. The process can be structured in terms of stages (e.g., analysis, synthesis, evaluation, and so on) by decomposition into subprocesses for parts of the partitioned design. Moreover, there may be assorted forms of backtracking, iteration, detailing, etc.

In some sense, the design process can be characterized by the way it will interact with the design object. An important goal is to find the appropriate means for describing the design process and to define the semantics in terms of the design transactions. Following this, a coherent solution for integrating the various design activities as well as user interface issues can be researched.

Much of a designer's activities consist of manipulations of the design object to add new information and changing and inspecting. In advanced systems, the *context* in which these take place may vary in time. Even the purpose of such activities may initially be left unspecified. This could, in some cases, influence the way in which the corresponding transactions are visualized. A design artifact must be properly represented to identify the status of the artifact information—proposed, decided, changeable, etc. Much of the interaction between process representations and artifact representations is dependent on the status information. In particular, the status must be allowed to be incomplete, inaccurate, or even inconsistent (in a given intermediate situation).

2 The Practice of CAD

We shall, as we noted earlier, be concerned with mechanical CAD. We'll take mechanical part design (machine design) as our domain of discussion for it probably has the strongest industrial appeal. As a well-established integral part of Computer-

Integrated Manufacturing (CIM), mechanical CAD is the backbone of today's highly industrialized world. It helps engineers develop products ranging from the simple and ordinary (e.g., chairs, bicycles) to the complex and sophisticated (e.g., cars, aircrafts). It multiplies the productivity many times and renders, using CIM techniques [26], robust products.

However, the practice of CAD in the industry is not without problems. It is commonly accepted that current CAD systems are large software systems which are difficult to master and inflexible to adapt to growing needs. They can only deal with limited domains and occlude attempts to integration. They do not cleanly support a crucial ingredient of design, viz. interaction. What is worse is that they lack a distinguishing characteristic of human designers: they have little or no *intelligence*.

Efforts to provide a wider perspective of CAD are now underway at several companies, universities, and research institutes. Since design is a highly mental activity, researchers have long felt the need for making CAD systems more intelligent. It should, on the other hand, be remarked that design is not *basically* a mental activity. Ask any painter, musician, architect, designer of mechanisms, or software engineer. Design is basically a physically creative activity. It is only in recent times (since the industrial revolution) that certain kinds—such as mechanical design—have become so abstracted from the associated physically creative (manufacturing) aspects that it can now *appear* to be a purely mental activity. This, however, is only an appearance, not a real property.

Our work is directed towards contributing to the theory of intelligent CAD systems. Our research regards AI techniques and knowledge engineering tools as fundamental to a design system which is hoped to be more substantial than expert systems. This paper gives a brief yet quite complete overview of our philosophy.

2.1 User Interface Issues

There are several useful ways of looking at intelligent CAD user interface architecture. The following dichotomies are quite common:

- CAD *vs.* Automated Design (AD).
- Designer's Apprentice (Assistant) *vs.* Autonomous Design System.
- Glass Box *vs.* Black Box.

The boundary between CAD and AD is indeed hard to delineate. We cannot object to the view that the ultimate aim of the computerization of design is to arrive at completely automatic design systems which can compete with and even surpass the best human designers. However, the interactive nature of design will probably dictate that, for a long time to come, CAD as man-machine cooperation must dominate. The same holds true for *apprentice vs. autonomous systems*. An apprentice system has less hard-wired knowledge than an autonomous system but knows better how to interact and has a generic model of design. An autonomous

system is very powerful for narrow domains. Besides, in such domains there may not be a need for a lot of interaction anyway. It is relatively easy to extend an apprentice by teaching it new skills. It is unwieldy to extend an autonomous system since its very constitution warrants myopia.

A more natural look at these dichotomies is via the metaphor *glass box vs. black box*. If a CAD system has a glass box structure then the user can, at any time, look through it to see partial results and processes. On the other hand, a black box system resembles to a batch processing environment; one submits the tasks to be executed and the system reports back with the results (or failure).

The seasoned researchers of CAD may remember those times when ideas such as *general CAD* have become fashion and then have been silently abandoned. Today, demands for integration suggest that we may want to reconsider that sweeping panorama of design. The view which regards design as a large collection of intelligent tools is different from the view which regards a design system as a framework. The intelligent tool approach assumes that if you have a cooperating set of experts which can communicate with each other then you can solve many problems. The framework approach regards the *shell* of the design systems as their biggest advantage; the domain specific issues can be dealt with separately, using the facilities provided by the shell.

2.2 Current Efforts

Mechanical CAD has evolved rapidly. Sutherland's revolutionary SKETCHPAD [24] in the Sixties generated much enthusiasm for using interactive graphics in engineering. (SKETCHPAD allows geometric shapes to be represented and various conditions to be specified over these shapes in terms of constraints, to which they then obey.) This in turn motivated a stormy decade when turn-key 2D drafting systems gradually replaced the drawing boards in the professional environment. Finally, the dust settled with their general acceptance by the industry; 3D modeling systems became available and they were also widely accepted by the industry as indispensable tools in product development.

Nevertheless, using an analogy due to Bobrow *et al.* [6], it is not unjust to claim that all these systems follow the *low road* approach. They regard design from a singular viewpoint, e.g., as a mainly geometric activity. Thus, despite their popularity, there are many problems with the existing CAD systems. As we'll shortly see, even the recent research cannot escape various inherited pitfalls.

As with any other discipline, a critique of other approaches to CAD systems presupposes a starting point, i.e., a vision of design. Briefly, we see design as an intellectual activity performed by human designers. We think that the essential thing in a designer is that *he builds us his world*. Thus, we believe that design systems should provide a framework where *designers can exercise their faculties at large*. With this view, we support, more or less, the idea of apprentice—as opposed to autonomous—CAD systems. We also carefully distinguish our view from other common views about the nature of design such as:

- Design is a routine process.
- Design is an inventive (innovative) process.
- Design is a problem-solving process.
- Design is a decision-making process.
- Design is an optimization process.

The first view above treats design as a rather straightforward activity where the designer selects from a previously known set of well-understood alternatives. A recent example is the AIR-CYL system [7]. Clearly, this view is ingenuous and does not reflect the intricate nature of design. The second view embraces the exciting ideas of AI aimed at creating novel devices by using knowledge of naive physics, qualitative reasoning, planning, analogical reasoning, brainstorming, and discovery heuristics. It is quite early to predict whether this can be achieved in domains more involved than the usual micro worlds of AI; thus EDISON [11] is only a toy system. The remaining views in the above list underplay the holistic nature of design. They are mostly implemented as expert systems which solve specific problems of a specific design process. An example of this *middle road* approach is the PRIDE [17] system which nevertheless is an interesting system with useful ideas behind it. An annoying and often cited problem with expert systems is that they cannot deliver genuinely expert performance since they have no *underlying mechanism* to understand what is going on. This problem manifests itself when a particular expert system in unable to solve a simple problem in spite of its proven expertise with difficult problems. This discrepancy contributed to the emergence of terms like *deep* and *shallow* (although there are several drawbacks to such usage) in AI.

High road systems are deep systems and our research is aiming at them. The knowledge of such systems is expected to represent the principles and theories underlying *design*. This may require that we try to demystify several aspects of design by way of formal, mathematical methods. (N.B. Such a formalization may not say much about what goes on during the design process, and serves as a rather simple *post hoc* rationalization of it.) It should be noted that we are not claiming, by positing the existence of such theoretical frameworks, that we know all the problem-solving components of general design and can offer a comprehensive model of it. Nor do we deny that there are many domain-specific sides to design. For example, VLSI design is mostly 2D while mechanical design is inherently 3D. We hope to incorporate the similarities in design, leaving the application-dependent issues to further consideration as side requirements. We believe that only through a clean formalization can one arrive at testable conjectures of design and build computer models of it. We shall shortly see that we value logic as the principal tool in this formalization. The reader is also referred to Coyne's treatise [8] for an interesting study of logic models of design. It should be remarked that our approach doesn't carry much similarity to Coyne's. While we consider it as a contribution to the

literature, we should nevertheless state that [8] demonstrates the applicability of *logic programming* to design, specifically architectural design such as spatial layout.

2.3 Key Problems

Problem: CAD systems support few design processes and models.

Producing final drawings is where current CAD systems tend to excel. This clearly depends upon what is meant by final drawings, but in the usual sense of them being what is handed to a production engineer, CAD systems are not that good even at producing final drawings—they cannot yet handle much of the details involved, such as tolerances, finishes, materials, etc. On the other hand, they are virtually powerless with respect to initial sketching. There are systems that can accept rough drawings but there is no system which can handle crude information during the design process.

Integration of models is essential since mechanical design deals with complicated gadgets. A design object must be viewed from various angles using different models. A good example is a wristwatch which can be viewed as an intricate assembly of gears, as a simple device with two hands rotating about a pivot, or as an abstract machine pointing to numbers denoting the time. More complicated examples follow when we consider the kinematic, dynamic, and control-theoretic models of a robot manipulator. In general, the present trend is to integrate the CAD systems around models concerning products. We suggest that they should be process-oriented and the so-called conceptual design stage should be supported by tools which contribute to the integration of CAD systems. We mean by integration:

- An integration of subsystems (i.e., auxiliary programs).
- An integration of design models and views based on an integrated model description scheme.
- An integration of design processes and automation of very early design stages.

Problem: CAD systems do not support error-checking.

Current CAD systems are not fully able to recognize inconsistencies in their input data. To worsen the situation, final outputs of conventional systems are so impressive that many errors go unnoticed for they exceed the mental capacity of designers. A remedy is to provide continuous error checks and to make sure that only the correct commands are accepted. Unfortunately, semantic error checks are difficult.

Mechanical engineering data exchange may cause deterioration of meaning. When we have a 3D solid modeling system based on say, boundary representation (Brep), we cannot easily exchange data with another solid modeling system based on say, constructive solid geometry (CSG).

Problem: Data entry is problematic.

This has to do with the lack of task domain *terminology* in the system. Because a conventional CAD system has no commonsense knowledge of machine design and cannot follow the designer's intentions, one is likely to enter a good deal of information to state simple requests like *Here I need a hole to insert the shaft I just created.*

When one inputs raw data manually, errors and misunderstandings during man-machine communication are inevitable regardless of the input devices. The ultimate solution is that systems must accept substantially reduced yet comprehensive data instead of raw data. For example, a CAD system should accept commands like *I would like to generate an object with such and such properties* or *I have supplied the minimum requirements, so proceed as you think fit.* (Here we are using natural language just to write our commands concisely and naturally; normally, the user enters these commands in some formalism other than natural language.)

Problem: Temporality, ambiguity, and inconsistency are not allowed.

In design, instead of sticking to one particular idea we may want to experiment with several ideas. This brings a time dimension to design. We may, during design, purge things we have previously built or introduce things that we have not considered before. We may require that the system temporarily forgets a particular facet of a design object since we are not concerned with it at the moment.

We also frequently want to separate the *structure* of a design object from the values of its attributes so that we can first decide about its shape during the conceptual design stage. For example, it is more important to recognize first the topology of a part if it is going to be inserted in another. Similarly, we may sometimes acknowledge the existence of a point rather than specify its exact location. A similar problem has been studied in database theory where it is known as *null values*. Simply stated, the fact that an entity has attributes is different than the notion that an attribute has a value [15].

Problem: Symbolic and numerical computing are not coupled.

Mechanical engineering systems normally use complex numerical and optimization procedures during design. However in many cases, insight into the problem-solving process is not present. Insight is also needed to interpret the outcome of some computation. As Richard Hamming declared: *The purpose of computing is insight, not numbers.* Traditionally, mechanical design systems contain a good deal of numerical knowledge (e.g., bulky libraries of numerical code) but nothing else. Users are left alone in analyzing the results of long, confusing computations. Recent research in *coupled systems* is directed towards integrating the explanation and the problem-solving abilities of expert systems with the precision of numerical computing [14].

3 A Theory of CAD?

A well-founded design theory may serve as a basis for specification and implementation of intelligent CAD systems. To be useful for this purpose, a design theory will have to satisfy the following:

- It is *realistic* in the sense that it has a close relation to design practice; it describes design processes as they are in practice or as users would like (or are accustomed) to do these.
- It is founded on a logical basis so that there are guarantees that a system, developed according to the design theory, will take *sound* steps.

These requirements for a design theory can be detailed as follows:

- *Phenomenological part:* Here descriptions are given as they are in practice. Also, identification of a number of design types (such as routine design *vs.* more creative types of design), design strategies, and design styles can be made.
- *Foundational part:* Here logical foundations are given for the conceptual descriptions in the phenomenological part. Thus, we allow for logical representations of (i) incomplete descriptions of objects, (ii) patterns of reasoning involved in design, and (iii) a multiworld mechanism (which is described in the sequel).

There are three ontological aspects of design: processes, models, and activities. This implies that we need theories corresponding to each aspect. A theory of CAD is then an aggregate of the following three theories:

- *Theory of design:* A theory which describes the design processes and activities.
- *Theory of design objects:* A theory which deals with the models of design objects. For our purposes, this should be a theory of machines; in VLSI design it would be a theory of VLSI, and so on.
- *Theory of knowledge:* A metalevel theory to describe our knowledge about design.

3.1 Design Theory and Logic

Usually, a design process is regarded as a mapping from the *function space* onto the *attribute space* [25]. Both spaces are defined on an entity set. A design process is an evolution process about a metamodel. A *metamodel* is a set of attributive descriptions of a design entity. During design, new attribute descriptions will be added (or existing ones will be modified) and the metamodel will converge to the

design solution. In other words, design specifications will initially be presented in functional terms and the design will be completed when all relevant attributes of the design object are determined so as to be able to manufacture it.

A simplified view of design is then as follows. A sequence of metamodels are generated from an initial specification and are detailed. If they cannot pass a feasibility check, then a compromise is made or backtracking is applied. Note that the models derived from metamodels are in the mean time evaluated for consistency. We thus have:

- *Specification:* $s = T^0 \rightarrow \cdots \rightarrow T^n \rightarrow \cdots \rightarrow T^N = g$
- *Metamodel:* $M^0 \rightarrow \cdots \rightarrow M^n \rightarrow \cdots \rightarrow M^N$
- *Propositions:* $q_0 \rightarrow \cdots \rightarrow q_n \rightarrow \cdots \rightarrow q_N$

In this scheme, s is the original design specification and g is the design goal. Each design step has an associated set of propositions which are denoted by q_n above. Two central concerns here are (i) how to choose q_n (i.e., how to proceed with design), and (ii) what if we discover $\neg q_n$ (i.e., how to deal with contradictions). We define a few other things:

$$q_n = p_0 \wedge p_1 \wedge \cdots \wedge p_k$$

where each p_i is an atomic fact concerning the metamodel,

$$q_n \overset{C_n}{\vdash} m^n$$

where $\vdash$ is the syntactic turnstile as usual in proof theory, m^n is a model and C_n is the *control knowledge*, and

$$q^n, m^n \overset{D_n}{\vdash} r$$

where D_n is the *detailing knowledge* and r is a proposition which should be added to q_n to arrive at the next description, q_{n+1}. Knowledge appearing above the syntactic turnstile is used in the derivations as metaknowledge. With this notation, the following partial classification of design can be made:

- *Invention:* Given s, find q_n, C_n, D_n, and g.
- *New product development:* Given s and C_n, find q_n, D_n, and g.
- *Routine design:* Given s, C_n, and D_n, find q_n and g.
- *Parametric design:* Given s, q_n, C_n, and D_n, find g.

Let $t\alpha p$ denote that p holds after time point t and $t\beta p$ denote that p holds before time point t; $[t_1, t_2]$ denotes a time interval. Then:

$$t\alpha\neg p = \neg(t\alpha p)$$

$$t\alpha(p \vee q) = t\alpha p \vee t\alpha q$$

$$t\alpha(p \wedge q) = t\alpha p \wedge t\alpha q$$

$$[t_1, t_2]\alpha p \stackrel{\text{def}}{=} t_1\alpha p \wedge t_2\beta p \wedge t_1 < t_2$$

Using this temporal logic, we can describe inference control for production rule systems in a more explicit way. For example, in Prolog the order of rules matter. In general, this knowledge is embedded in the interpreter of the language. By disclosing this control we may introduce suppler control. For example, the detailing knowledge D_n introduced above may be a set of rules of the following sort:

$$t_1\alpha q_1 \wedge t_2\alpha q_2 \wedge t_1 < t_2 \supset t_2\alpha q_3$$

$$t_1\alpha q_1 \wedge t_2\alpha q_2 \wedge t_1 > t_2 \supset t_2\alpha q_4$$

These two formulas should be read as follows. If q_1 holds after t_1, q_2 holds after t_2, and t_1 is earlier than t_2, then a new property, q_3, holds after t_2. Otherwise (i.e., t_1 is later than t_2) another property, q_4, holds. Applications of temporal logic to design process representation should now be manifest.

Intuitionistic logic can be blended with temporal logic. Let $t_p\alpha p \equiv true$. Now, introducing a logical symbol *unknown*, we can formalize intuitionism in terms of temporality:

$$t_p\beta(p \vee \neg p) \equiv unknown$$

$$t_p\alpha(p \vee \neg p) \equiv true$$

Here, p is a property that we are trying to prove about the current design object or design process.

3.2 A Multiworld Mechanism

During the course of design, it is of vital importance that a system allows the designer to represent the design object in various ways. These various descriptions are called *models* of the design object. Four such models are:

- Functional specification of the design object in terms of the constraints that should be met.
- A geometrical model which creates a visual representation of the design object (something that current CAD systems know rather well).

- A finite elements model enabling strength predictions to be made on complex mechanical constructions.
- A cost analysis model to calculate the manufacturing costs.

Models are able to communicate with each other by means of a metamodel. A metamodel is a central description of the design object to which all models refer and depend on. All changes that occur in a certain model are propagated through the metamodel to all other models (which are updated appropriately).

This is achieved by a special construct, the *multiworld mechanism*. A model is created by a call to a design *scenario* which in turn opens a world. A *world* is a part of the design object description together with some information that belongs to the particular model that is created. The multiworld mechanism lets the designer open several worlds simultaneously; in other words, several models may be active at the same time. For example, the designer may input new constraints and examine the results of these in a CSG model and a finite elements model.

Moreover, the multiworld mechanism allows the designer to create multiple descriptions of the design object in parallel. Here, we make a distinction between *dependent vs. independent worlds*. Dependent worlds result in a unique design object description; they reflect the same metamodel. Independent worlds, however, can result in different design object descriptions; they do not reflect the same metamodel. In a nutshell, dependent worlds are used to create multiple views of the design object while independent worlds give rise to alternative design solutions.

Scenarios specify how many worlds exist simultaneously and how they relate to each other. Special care is taken for constructs which close or update a world, thereby transferring some of its properties to the metamodel. This control mechanism also checks the validity of the worlds with respect to the metamodel and propagates changes in the metamodel to all applicable worlds.

4 Naive Physics and Machine Design

In machine design, it is not yet precisely known in which symbolism one should describe the functions of machines. There is, however, a view that functions can be represented in terms of the physical phenomena that the machine exhibits [3]. Regarding this point of view, the representation of functions can be reduced to the representation of physical phenomena and qualitative reasoning about them. Minsky explains this [16]:

> When a particular machine is described to us, we do not first ask questions about its material construction. Given an engineering drawing, a circuit diagram, a patent description—something must first convince us that we understand how it works *in principle*. That is, we must see how it is 'supposed' to work. We inquire only later whether this member will stand the stress, or whether that oscillator is stable under load, etc. But the *idea* of a machine centers around some *abstract* model or process.

In qualitative physics, the reduction of information is arrived by creating an abstract layer which may, strictly speaking, be incorrect but sufficiently correct for the problem at hand. Naive physics observes that people have a different kind of knowledge about the physical world. This knowledge can best to described as *common sense* and is attained after years of interaction with the world. Accordingly, naive physics ideas are useful in machine design and we want to codify them. To this end, we follow the manifesto of [13] and hope to capture naive physics in logic. Additionally, we use as a mathematical tool *qualitative reasoning*; this is explained below.

Consider the following concepts underlying change in physical systems: state, cause, equilibrium, oscillation, force, feedback, etc. Naive physics regards these concepts in a simple qualitative way. It maps continuous variables to discrete variables taking only a small number of values (e.g. $+$, $-$, and 0). Accordingly, differential equations are mapped to qualitative differential equations (also known as *confluences*). Consider a pressure regulator (cf. Figure 1 in [10] to make sense of the discussion to follow). The confluence

$$\partial P + \partial A - \partial Q = 0$$

describes this device in qualitative terms. Here P is the pressure across the value, A is the area available for flow, and Q is the flow through the valve; ∂ means change. Let us look at the way the regulator works. An $\uparrow$ in pressure at $a \Rightarrow$ an $\uparrow$ in pressure at $b \Rightarrow$ an $\uparrow$ in flow through $b \Rightarrow$ an $\uparrow$ in pressure at $d \Rightarrow$ diaphragm at e pushes downward $\Rightarrow$ valve at b tightens $\Rightarrow$ an $\uparrow$ in area for flow $\Rightarrow$ a $\downarrow$ in flow through b. Thus the regulator maintains constant pressure at c in spite of fluctuations at the supply side a. Note that if the valve is closed (i.e. $\partial A = \partial Q = 0$) then it is predicted that P is constant. The correct confluence for this state should instead read $\partial Q = 0$; we simply do not say anything about P.

We have to admit that when one speaks of machine design, *mechanism* (interconnected links and joints) design should also be considered. Mechanism design represents the foundation of a large percentage of the entire field of machine design. Also, it is probably more complex than other types of mechanical design (i.e. static components). There is not much useful work (yet) regarding the qualitative analysis of mechanisms.

The use of qualitative reasoning facilitates the analysis of the working of physical devices. This may not be without a price. Qualitative techniques may cause ambiguities. Assume that a certain quantity M varies with $\frac{N_1}{N_2}$, i.e., $M \propto \frac{N_1}{N_2}$. If N_1 $\uparrow$ while N_2 remains constant then M also $\uparrow$. However, we cannot reason without a finer knowledge of the individual changes when we are told that both N_1 and N_2 $\uparrow$ (or $\downarrow$). The techniques of *order of magnitude reasoning* are designed to handle precisely this kind of problem without requiring knowledge of the numerical values involved.

In the pressure regulator example, we are employing a powerful principle in our reasoning, viz. the *mechanistic* world view. This asserts that every physical situation is regarded as a device made up of individual components, each contributing to

the overall behavior. Nevertheless, the laws of the substructures may not presume the functioning of the whole—the principle of *no-function-in-structure* [10]. Additionally, assumptions that are specific to a particular device should be distinguished from the *class-wide assumptions* which are common to the entire class of devices. A simplistic view of modeling devices comprises three kinds of constituents: materials (e.g. oil, air), components (e.g. shafts, wheels), and channels (e.g. water pipes, electric cables).

After modeling a device we can reason about it. In *envisionment* we start with a structural description to determine all possible behavioral sequences. Thus, in envisioning, we momentarily forget about the real values of the problem variables and try to see all possible outcomes of some action [9].

Naive physics concepts are required for design because a design object will have a physical existence and accordingly, obey natural laws. If we want to create designs corresponding to physically realizable design objects then we will have to refer to naive physics notions such as solids, space, motion, etc. Furthermore, if we want to reason about a design object in its destined environment (think of our pressure regulator to be installed in a nuclear reactor) we will need naive physics procedures such as envisioning, simulation, and diagnostics.

Caveat. Naive physics might be sufficiently correct for use in describing (to some level of accuracy) some physical phenomena, but this is a different matter from using naive physics to design real world objects, where it is much harder to know *a priori* what level of correctness is required. Design usually assumes a high accuracy is obligatory and that the best representations of natural laws are requisite. That is why we use Newtonian mechanics for example, rather than something like Aristotelian mechanics, which might seem more intuitive to a naive physicist. It is one thing to use naive physics notions to predict what might happen in some physical system, it is quite another to use them to determine how to get something to happen in a physical system.

5 Acknowledgments

We thank Paul ten Hagen (Centrum voor Wiskunde en Informatica, Amsterdam) and Tetsuo Tomiyama (University of Tokyo) for their moral and intellectual support.

References

[1] V. Akman, P. ten Hagen, J. Rogier, and P. Veerkamp, Knowledge engineering in design, *Knowledge-Based Systems* **1**(2), pp. 67–77 (1988).

[2] V. Akman, P. ten Hagen, and T. Tomiyama, A fundamental and theoretical framework for an intelligent CAD system, *Computer-Aided Design* **22**(6), pp. 352–367 (1990).

[3] V. Akman and P. ten Hagen, The power of physical representations, *Intelligent CAD Systems II: Implementational Issues*, V. Akman, P. ten Hagen, and P. Veerkamp (eds.), Springer-Verlag, Berlin, pp. 170–194 (1989).

[4] J. Barwise, *The Situation in Logic*, CSLI Lecture Notes **17**, Center for the Study of Language and Information, Stanford, Calif. (1989).

[5] J. Barwise and J. Perry, *Situations and Attitudes*, MIT Press, Cambridge, Mass. (1986).

[6] D. Bobrow, S. Mittal, and M. Stefik, Expert systems: Perils and promise, *Communications of the ACM* **2**(9), pp. 880–894 (1986).

[7] D. Brown and B. Chandrasekaran, Knowledge and control for a mechanical design expert system, *IEEE Computer* **19**(7), pp. 92–100 (1986).

[8] R. Coyne, *Logic Models of Design*, Pitman, London (1988).

[9] J. de Kleer, Qualitative and quantitative knowledge in classical mechanics, technical report AI–TR–352, Massachusetts Institute of Technology, Cambridge, Mass. (1975).

[10] J. de Kleer and J. Brown, A qualitative physics based on confluences, *Artificial Intelligence* **24**, pp. 7–83 (1984).

[11] M. Dyer, M. Flowers, and J. Hodges, EDISON: An engineering design invention system operating naively, *Artificial Intelligence in Engineering* **1**(1), pp. 36–44 (1986).

[12] P. Fishwick and B. Zeigler, Qualitative physics: Towards the automation of systems problem solving, *Proceedings of the AI, Simulation, and Planning Workshop in High Autonomy Systems*, IEEE Computer Society Press, pp. 117–134 (1990).

[13] P. Hayes, The second naive physics manifesto, *Formal Theories of the Commonsense World*, J. Hobbs and R. Moore (eds.), Ablex, Norwood, New Jersey, pp. 1–36 (1985).

[14] J. Kowalik (ed.), *Coupling Symbolic and Numerical Computing in Expert Systems*, Elsevier, Amsterdam (1986).

[15] H. Levesque, The logic of incomplete knowledge bases, *On Conceptual Modeling*, M. Brodie, J. Mylopoulos, and J. Schmidt (eds.), Springer-Verlag, New York, pp. 165–186 (1984).

[16] M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, N. J., p. 7 (1967).

[17] S. Mittal, C. Dym, and M. Morjaria, PRIDE: An expert system for the design of paper handling systems, *IEEE Computer* **19**(7), pp. 102–114 (1986).

[18] J. Mostow, Towards better models of the design process, *AI Magazine* **6**(1), pp. 44–57 (1985).

[19] N. Negroponte, *The Architecture Machine (Toward a More Human Environment)*, MIT Press, Cambridge, Mass. (1970).

[20] A. Newell and H. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, N. J. (1972).

[21] D. A. Schön, Designing: Rules, types, and worlds, *Design Studies* **9**(3), pp. 181–190 (1988).

[22] H. Simon, The science of design: Creating the artificial, *The Sciences of the Artificial*, MIT Press, Cambridge, Mass., pp. 55–83 (1979).

[23] H. Simon, The structure of ill-structured problems, *Models of Discovery (and Other Topics in the Methods of Science)*, D. Reidel, Dordrecht, pp. 304–325 (1977).

[24] I. Sutherland, SKETCHPAD: A man-machine graphical communication system, *Proceedings of the AFIPS Spring Joint Computer Conference*, Spartan Books, Baltimore, Md., pp. 329–346 (1963).

[25] T. Tomiyama and H. Yoshikawa, Extended general design theory, technical report CS–R8604, Center for Mathematics and Computer Science, Amsterdam (1986).

[26] R. Yeomans, A. Choudry, and P. ten Hagen, *Design Rules for a CIM System*, North-Holland, Amsterdam (1985).