Massachusetts Institute Of Technology
Artificial Intelligence Laboratory

A.I.Memo No. 1062 January, 1989

Quantitative Inference in a Mechanical Design "Compiler"

Allen C. Ward
Warren P. Seering

ABSTRACT

This paper presents the ideas underlying a computer program that takes as input a schematic of a mechanical or hydraulic power transmission system, plus specifications and a utility function, and returns catalog numbers from predefined catalogs for the optimal selection of components implementing the design. Unlike programs for designing single components or systems, this program provides the designer with a high level "language" in which to compose new designs. It then performs some of the detailed design process for him.

The process of "compilation", or transformation from a high to a low level description, is based on a formalization of quantitative inferences about hierarchically organized sets of artifacts and operating conditions. This allows design compilation without the exhaustive enumeration of alternatives. The paper introduces the formalism, illustrating its use with examples. It then outlines some differences from previous work, and summarizes early tests and conclusions.

This report describes work done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Funding for the work was provided in part by the Industrial Technology Institute of Ann Arbor, Michigan, and in part by the Office of Naval Research under University Research Initiative contract N00014-86-K-0685.

1 Introduction

> ... in practice, failure is still far less frequently the result of bad working principles than of poor detail design. *Pahl and Beitz[1]*

In this paper we introduce the theory underlying a computer program that selects standard components from catalogs in order to implement a wide variety of mechanical designs. The user of the program forms a schematic by combining such elements as those in Fig. 1. Given the schematic, specifications, and a utility function, the program returns the optimal catalog numbers.

We can view the schematics and the specifications as a description in a "high level (source) language", and the catalog numbers as a description in a "low-level (target) language". Then, by analogy with computer language compilers, we can call our program a "mechanical design compiler". Like computer language compilers, such programs should improve designer productivity, prevent errors, and allow the exploration of more alternatives in greater depth.

1.1 Observations on Component Selection

Suppose that we wanted to design a power train for an ice-cream stirrer (Figure 2). We will call this the Toscanini's problem, after a local eatery. Given a range of acceptable stirring speeds, the torques required, and a catalog, we might use the transmission input-output equations $RPM_i = (ratio)(RPM_o)$ and $torque_o = (ratio)(torque_i)$ to systematically eliminate those transmissions unable to provide the required speed with the available motors. Then we might eliminate those motors unable to provide the required torque through any of the remaining transmissions.

We have several observations to make. First, note that in this example we think about *sets* of artifacts (e.g. all Dayton 3-phase motors), rather than particular artifacts (the motor that fell off the loading dock yesterday). Because of manufacturing variation, even a single catalog number designates a set of different physical artifacts, which may or may not be interchangeable in

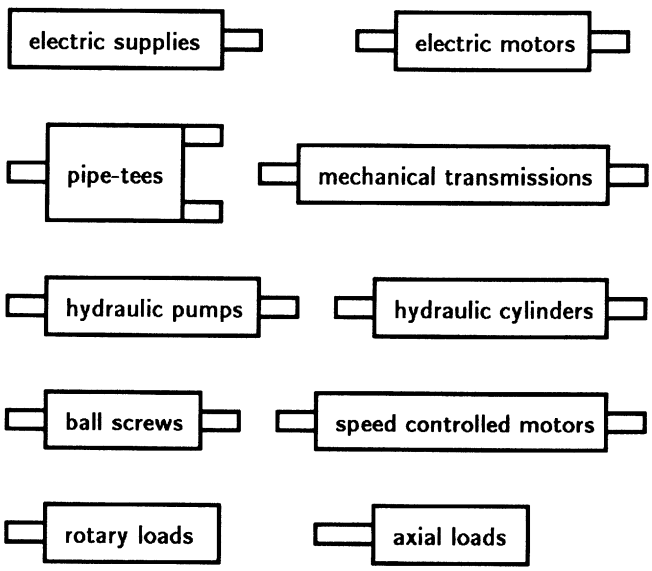

Figure 1: Schematic elements

a particular design. We must also consider sets of operating conditions; for example, the ice cream maker may be nearly empty, or full of cold Double Dutch Chocolate. We cannot always assume that the maximum load is the only one that matters—some electric motors over-heat unless operated at nearly full load.

Second, torque is a *quantitative* property, normally expressed in terms of real numbers. Our reasoning about torque is therefore also quantitative. However, while the torque at a particular operating condition is normally represented by a real number, the torques required by the stirrer under all ice-cream viscosities and fill levels correspond to an interval of real numbers (say those from 10 to 40 newton-meters.)

Third, the artifact sets are organized, for example by horsepower and motor speed. We can eliminate large sets simultaneously (e.g. all the motors of less than 1 horsepower).

Finally, the only *mathematical* expressions used in the example were algebraic equations. Most designers would attack the problem by substituting single values (say for the largest output torque) into the equations, then comparing the results with other single values from catalogs. If asked to justify using calculations on single values to draw conclusions about sets, they would provide intuitive arguments, in English and specific to the particular problem being considered.

We might write these intuitions into an "expert system", and that program might work well in a sufficiently narrow domain. But a compiler should give correct results on every design which can be composed from the schematic elements. It therefore needs a general and precise theory, which can be closely examined and confidently applied to diverse design problems.

1.2 Preview

This paper introduces a theory for quantitative inference about sets of artifacts and operating conditions. The theory provides the basis for a mechanical design compiler which operates by eliminating unsatisfactory alternatives from catalog sets of artifacts.

We will begin with a brief overview of the compiler. We then introduce some operations on real number intervals. From intervals, we build up a language of "labeled-intervals", or "specifications". Then, we illustrate the use of formal operations on this language to perform quantitative inferences in the solution of the Toscanini's problem.

2 A design compiler

A user of our compiler creates a design schematic by pointing in sequence at displayed icons. Each icon represents a computational "object", which normally includes a list of catalog numbers. Thus, it also represents the set of real artifacts purchasable by ordering from the catalog. Associated

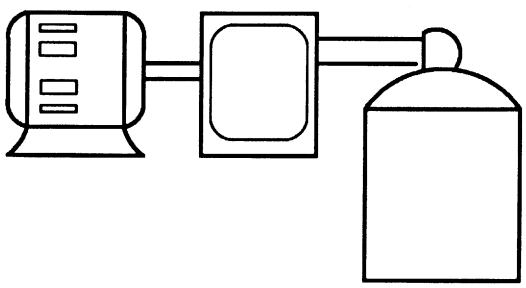

Figure 2: The Toscanini's Problem

with each catalog number are specifications in the labeled interval language. Other specifications automatically abstracted from these, along with equations built into the schematic object, describe the whole set of artifacts represented by the icon.

The schematic assembly process establishes an identity between corresponding variables for connected components. For example, in the Toscanini's problem the output torque of the transmission is identified with the input torque to the stirrer.

Having assembled a schematic, the user supplies specifications in the labeled interval language for the most convenient objects, usually loads. The objects pass each other these specifications, the specifications abstracted from the artifact sets, and new specifications derived from these by using equations. The objects eliminate from consideration incompatible artifacts (by deleting numbers or groups of numbers from the catalog listing), and abstract new descriptions for the resulting subsets. In the Toscanini's problem, the user might specify the range of torques required at the stirrer input shaft. This information, propagated through equations in the the transmission object, would eliminate those motors unable to supply enough torque to drive the load through *any* of the transmissions under consideration.

Since the information reaching the motor object is about *all* the possible combinations of transmission and load, the compiler does not explicitly enumerate the alternative combinations of motor and transmission. This approach may be contrasted with one in which alternatives are generated, evaluated, and discarded or modified. If we think of the design process as searching a space of artifacts, our approach works by eliminating volumes of the space, while the other evaluates designs at points in the space. At any time during our program's operation, the schematic represents the volume of the artifact space which has not been eliminated.

This approach has several advantages.

- Manufacturing tolerances and operating condition variations are represented explicitly.
- The program need not examine each alternative individually.
- Elimination inferences, unlike choice inferences, can be confidently made from partial information. For example, our program does not yet contain a representation of geometry, but it can still safely eliminate motors providing insufficient torque. It could *not* safely choose a motor—it might not be suitable geometrically.
- The inference system has been designed to produce only statements which are true of *each* of the objects being considered at the present stage of compilation. The sets of artifacts considered at later stages will be subsets of this set, so the statement will still be true of each artifact. Therefore, statements never need to be withdrawn.
- The meaning of design representations is often left intuitive; designs are sometimes said to stand for an "archetype", or a "partially defined object". In contrast, at each stage of the compilation process our representation stands for a well-defined set of physical objects. We can therefore evaluate operations by using physical reasoning about the objects represented before and after a formal operation.

This set-based approach, however, has one significant disadvantage: conventional, single-valued or even "constraint propagating" systems of mathematical inference are inadequate to deal explicitly with sets of artifacts and operating conditions. We now begin building appropriate inference tools based on relationships between variables and intervals of real number values.

3 Some Operations on Intervals

We need to work with sets of values, for example the torque required to drive an ice cream stirrer under all load conditions. We might write $0 \leq torque \leq 10$ (in our favorite units), or $torque \in [0\ 10]$ but will instead write $\langle torque\ 0\ 10\rangle$; for now, the reader can assume these statements mean the same thing.

Using this notation, we will present eight operations on intervals. Because we are trying to convey a general understanding we will present the operations using examples, and claim without proof that under appropriate circumstances the operations are both well defined and computable. For more detail, see [2].

The first five operations used by our design compiler are straightforward, and are illustrated in the following examples.

- Intersection: $\cap(\langle x\ 1\ 4\rangle,\ \langle x\ 2\ 6\rangle) \longrightarrow \langle x\ 2\ 4\rangle$.
- Not-intersection: $\not\cap(\langle x\ 1\ 4\rangle, \langle x\ 2\ 6\rangle) \longrightarrow$ FALSE.
- Filled-union: $\cup(\langle x\ 1\ 4\rangle, \langle x\ 8\ 10\rangle) \longrightarrow \langle x\ 1\ 10\rangle$.
- Subset: $\subseteq(\langle x\ 10\ 12\rangle,\ \langle x\ 10\ 14\rangle) \longrightarrow$ TRUE.
- Not-subset: $\not\subseteq(\langle x\ 10\ 12\rangle,\ \langle x\ 10\ 14\rangle) \longrightarrow$ FALSE.

We will call the sixth operation RANGE. RANGE takes an implicit equation in three variables and a pair of intervals in two of the variables, and returns the compatible interval in the third variable. More precisely, suppose that $g(x, y, z) = 0$ is the implicit equation, and X and Y are intervals in x and y respectively. Then $\text{RANGE}(g, X, Y) \longrightarrow Z$, where Z is the minimal interval such that for every assignment of $x \in X$ and $y \in Y$, there is an assignment of $z \in Z$ which satisfies g.

Let us do an example. Suppose that in the Toscanini's Problem, we had available transmission ratios only from 2 to 4, and we knew that output torques above 8 would damage the stirrer. Figure 3 represents the transmission equation, $t_i - \frac{t_o}{ratio} = 0$, by showing lines of constant output torque. From the figure we see that regardless of our choice of transmissions, any motor providing input torque above 4 will induce output torque above 8. We might reasonably conclude that regardless of our choice of transmission we should not use any motor producing running torques above 4.

The RANGE operation produces the appropriate interval:

$$\text{RANGE}(t_i - \frac{t_o}{ratio} = 0,\ \langle t_o\ 0\ 8\rangle,\ \langle ratio\ 2\ 4\rangle) \longrightarrow \langle t_i\ 0\ 4\rangle.$$

The RANGE operation is equivalent to what is usually called the constraint propagation of inequalities, and has been well explored[3]. However, it is not the only operation of interest. Suppose that instead of saying that the stirrer will be damaged by torques above 8, we say that torques ranging from 0 to 8 may be required to drive it. We should conclude that we need motors able to provide torques ranging ranging at least from 0 to 2; see Figure 4. We will call the operation producing this interval DOMAIN; it can be defined as an inverse of RANGE. For example,

$$\text{DOMAIN}(t_i - \frac{t_o}{ratio} = 0,\ \langle t_o\ 0\ 8\rangle,\ \langle ratio\ 2\ 4\rangle) \longrightarrow \langle t_i\ 0\ 2\rangle$$

precisely because

$$\text{RANGE}(t_i - \frac{t_o}{ratio} = 0,\ \langle t_i\ 0\ 2\rangle,\ \langle ratio\ 2\ 4\rangle) \longrightarrow \langle t_o\ 0\ 8\rangle.$$

Finally, we define the eighth operation, SUFFICIENT-POINTS, as another sort of inverse to RANGE. Suppose in the Toscanini's problem we knew that we had available only motor torques

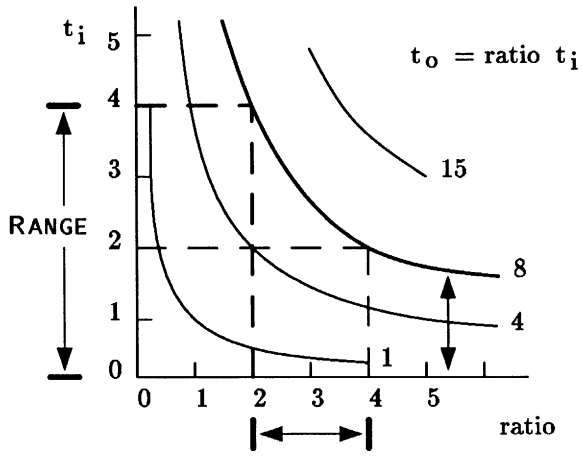

Figure 3: An illustration of the RANGE operation

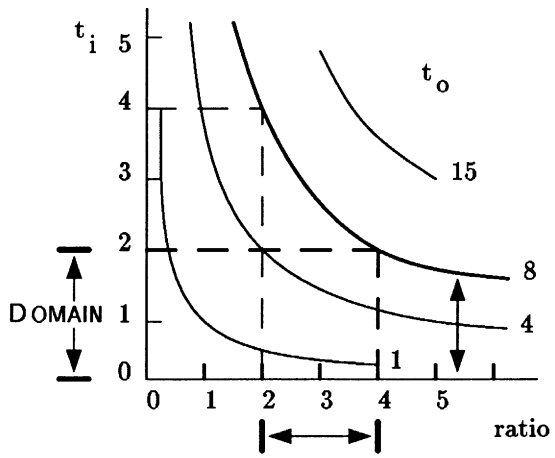

Figure 4: An illustration of the DOMAIN operation

up to 2, and we needed stirrer torques up to 8. Looking at Figure 5 we would conclude that any transmission ratio of 4 or above would do. That is,

$$\text{SUFPT}(t_i - \frac{t_o}{ratio} = 0,\ \langle t_o\ 0\ 8\rangle,\ \langle t_i\ 0\ 2\rangle) \longrightarrow \langle ratio\ 4\ \infty\rangle$$

because for all ratios in [4 ∞], the RANGE of the ratio and the input torque includes the output torque. For example, a ratio of 5 would give the output torque interval 0 to 10, which includes the desired interval, 0 to 8.

$$\text{RANGE}(t_i - \frac{t_o}{ratio} = 0,\ \langle t_i\ 0\ 2\rangle,\ \langle ratio\ 5\ 5\rangle) \longrightarrow \langle t_i\ 0\ 10\rangle.$$

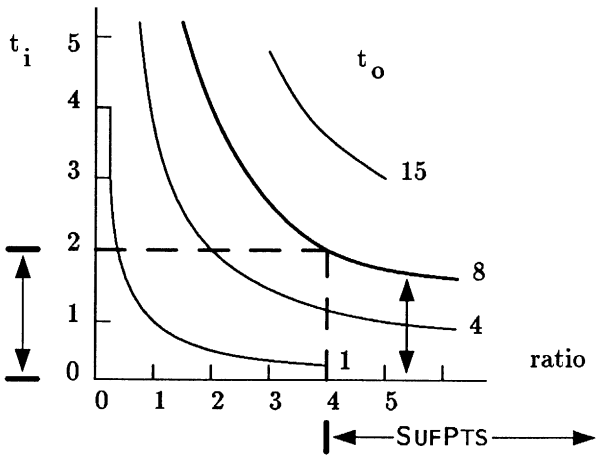

Figure 5: An illustration of the SUFFICIENT-POINTS operation

All of the operations presented are sometimes useful in design, but when should we use each one? In these examples we used our experience as designers to decide which operation would produce the desired interval. In a formal system, we need to build the information guiding those decisions into the specifications themselves. We will call these augmented interval statements **labeled intervals**.

4 The labeled interval specification language

We will return to the examples of the previous section, but first introduce the language of labeled intervals using an even simpler design problem—selecting one of a set of motors to be connected directly to a load (Figure 6).

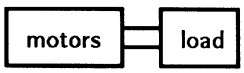

Figure 6: A very simple power train

4.1 Limits and operating regions

Suppose that we know that each of some set of motors can produce torques throughout the interval 0 to 20, but that damage may result to the load if the torque goes above 10. We want to eliminate these motors from consideration. Given only the intervals ($\langle t\ 0\ 20\rangle$, $\langle t\ 0\ 10\rangle$) a program would not have enough information to specify what operation to use. For example, if the larger interval applied to the load and the smaller to the motors, we would not eliminate the motors. We can attach the information required using the following labels.

The **Limits** label, symbolized by $\overset{only}{[\ \ ]}$, indicates that values of the variable will or must be drawn *only* from the interval. Thus, $\langle \overset{only}{[\ \ ]}\ t\ 0\ 10\rangle$ means that the torque must not reverse or go above 10. Similarly, the tolerance on a bearing inner diameter can be expressed as $\langle \overset{only}{[\ \ ]}\ d_i\ 2.99\ 3.01\rangle$.

The **Operating-Region** label, symbolized by $\overset{every}{\leftrightarrow}$, indicates that the variable will or must assume *every* value in the interval; $\langle \overset{every}{\leftrightarrow} t\ 0\ 20 \rangle$ indicates that the motor torque can at least range from 0 to 20 (and perhaps beyond.)

We will later define a rule which eliminates these motors because, for the variable t, the operating-region interval is not a subset of the limit interval.

4.2 Required, Assured, and No-stronger labels

Suppose that in our motor-load example we want the load speed to be regulated to between 1750 and 1800 rpm. We introduce a **Required** (**R**) interval label, meaning that the statement must be true for proper function. For the load, we can write $\langle \mathbf{R} \overset{only}{[\ ]}\ RPM1750\ 1800 \rangle$.

Suppose further that some catalog number designates a set of high-slip motors, capable of regulating the speed only well enough to keep it between 1725 and 1800. We introduce the **No-stronger-possible** (**N**) label, and write for the high slip motors $\langle \mathbf{N} \overset{only}{[\ ]}\ RPM1725\ 1800 \rangle$. By this we mean that we cannot specify any subset of these motors which guarantees stronger limits. (Because of manufacturing variation, some of them probably do guarantee better speed regulation, but we cannot, within the framework given, select these.) We will define a rule which eliminates these motors because the No-stronger-possible limit interval for *RPM*is not a subset of the Required limit interval.

The final label in this class is **Assured** (**A**), indicating that we are sure a particular statement will be true for all the artifacts represented (under appropriate conditions). Thus for our high slip motors, we have also $\langle \mathbf{A} \overset{only}{[\ ]}\ RPM1725\ 1800 \rangle$.

We have illustrated the Assured, Required, and No-stronger-possible labels only in conjunction with the Limit ($\overset{only}{[\ ]}$) label, but they can be defined comparably in conjunction with the Operating-Range ($\overset{every}{\leftrightarrow}$) label.

Labeled interval descriptions are models of artifact sets, and we can choose the level of abstraction of the model. For example, there is a torque curve for each motor type, which would allow more accurate prediction of the speed regulation based on the possible torques. If we chose to include the torque curve in our describing equations, we would apply labeled interval specifications to the equation's coefficients.

In addition to the labels defined above, we designate each quantity as either a **state variable** or a **parameter**. Parameters, such as gear ratio, are fixed at manufacture, while state variables like torque may vary during operation.

Each labeled interval pertains only to a specified set of operating conditions such as start-up or normal operating conditions. We will assume "normal operating conditions" throughout this paper.

5 Operations on Labeled Intervals

The key activities of our compiler can be specified by three groups of formal operations on labeled intervals: elimination, abstraction, and specification propagation.

5.1 Elimination

These operations eliminate artifact sets whose labeled interval specifications conflict with the specifications imposed by the user or by other parts of the design.

We represent these operations using patterns. Suppose that for our motor-load power train we have the same speed regulation requirement as above: $\langle \mathbf{R} \overset{only}{[\ ]}\ RPM1750\ 1800 \rangle$. We want to eliminate motors with weaker speed regulation, say $\langle \mathbf{N} \overset{only}{[\ ]}\ RPM1725\ 1800 \rangle$. These two specifications

match the pattern

$$\langle \mathbf{N} \overset{only}{[\ ]} X\ x_l\ x_u \rangle \not\subseteq \langle \mathbf{R} \overset{only}{[\ ]} X\ x_l\ x_u \rangle \longrightarrow \text{eliminate},$$

with X taken to be the RPM and x_l and x_u the lower and upper bounds of the corresponding intervals.

Since the No-stronger-possible specification is not a subset of the Required specification, the program removes the relevant catalog numbers from the associated list.

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} x \ldots \rangle \not\subseteq \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} x \ldots \rangle$
$\langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} x \ldots \rangle \not\cap \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} x \ldots \rangle$
$\langle \mathbf{N} \overset{only}{[\ ]} x \ldots \rangle \not\subseteq \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} x \ldots \rangle$
$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} x \ldots \rangle \not\subseteq \langle \mathbf{N} \overset{every}{\leftrightarrow} x \ldots \rangle$

Table 1: Elimination patterns

All our elimination patterns are shown in Table 1 (with the arrow and the word "eliminate" omitted for brevity). When the list "(**R A**)" appears in a pattern, it can be matched against either a Required or an Assured statement.

5.2 Abstraction

In the Toscanini's problem, we want to evaluate motor alternatives with respect to the set of all transmissions under consideration. Therefore, we need a set of specifications which describe all the transmissions. The program **abstracts** these specifications from the previously encoded descriptions of the individual "catalog number" subsets.

The program uses either the *intersection* or the *filled-union* operation to combine the intervals associated with a given variable and pair of labels in each subset. For assured limits it uses the filled-union operation, so for example it combines $\langle \mathbf{A} \overset{only}{[\ ]}\ RPM1150\ 1200 \rangle$ and $\langle \mathbf{A} \overset{only}{[\ ]}\ RPM1750\ 1800 \rangle$ to form $\langle \mathbf{A} \overset{only}{[\ ]}\ RPM1150\ 1800 \rangle$. There are six types of labeled interval defined by combining the two label sets (Assured, Required, No-stronger-possible) and (Limit, Operating-Region). Table 2 shows the operation appropriate for combining each type of labeled interval.

interval type	operation
$\langle \mathbf{A} \overset{every}{\leftrightarrow} \rangle$	$\cap$
$\langle \mathbf{A} \overset{only}{[\ ]} \rangle$	$\cup$
$\langle \mathbf{R} \overset{every}{\leftrightarrow} \rangle$	$\cap$
$\langle \mathbf{R} \overset{only}{[\ ]} \rangle$	$\cup$
$\langle \mathbf{N} \overset{every}{\leftrightarrow} \rangle$	$\cup$
$\langle \mathbf{N} \overset{only}{[\ ]} \rangle$	$\cap$

Table 2: Abstraction operations

5.3 Propagating Labeled Intervals Using Equations

We turn now to a more complex question: how can we propagate labeled intervals through equations, so that, for example, the torque requirements for the ice cream stirrers can be converted

into torque requirements for the motors? We introduce two operations on labeled intervals and equations.

The first is represented by the following pattern:

$$\langle(\mathbf{R\ A})\ [\ ]^{only}\ v_1\rangle\ \&\ \langle(\mathbf{R\ A})\ [\ ]^{only}\ v_2\rangle\ \&\ g(v_1,v_2,v_3)=0$$
$$\longrightarrow\langle(\mathbf{RubA})\ [\ ]^{only}\ v_3\ \textsc{Range}\rangle.$$

The labeled interval patterns to the left of the arrow are matched with potential inputs to the operation, while the pattern to the right of the arrow defines the form of the output. The "$g(v_1, v_2, v_3) = 0$" matches equations linking the two input variables and the output variable. The "**(R A)**" in the input patterns again indicate that the operation is appropriate for either Required or Assured statements. The "**(RubA)**" in the output indicates that the output will be Required unless both inputs are Assured, in which case it will be Assured. Finally, the "RANGE" in the output pattern indicates that the numeric values are to be found by applying the RANGE operation to the input values.

Suppose again that in the Toscanini's Problem, we have available transmission ratios only from 2 to 4, and we know that torques above 10 would damage the stirrer. The specifications match our pattern:

$$\begin{array}{lcl}\langle\mathbf{R}\ [\ ]^{only}\ t_o\ 0\ 10\rangle & \sim & \langle(\mathbf{R\ A})\ [\ ]^{only}\ v_1\rangle\\ \langle\mathbf{A}\ [\ ]^{only}\ ratio2\ 4\rangle & \sim & \langle(\mathbf{R\ A})\ [\ ]^{only}\ v_2\rangle\\ \frac{t_o}{ratio}-t_i=0 & \sim & g(v_1,v_2,v_3)=0.\end{array}$$

This justifies applying the RANGE operation to form $\langle\mathbf{R}\ [\ ]^{only}\ t_i\ 0\ 5\rangle$. The elimination operations will use this new specification to eliminate any motor producing torques above 5.

The second operation is represented by the pattern

$$\langle(\mathbf{R\ A})\ \leftrightarrow^{every}\ s_1\rangle\ \&\ \langle(\mathbf{R\ A})\ [\ ]^{only}\ p_2\rangle\ \&\ g(s_1,p_2,s_3)=0$$
$$\longrightarrow\langle(\mathbf{RubA})\ \leftrightarrow^{every}\ s_3\ \textsc{Domain}\rangle.$$

Reading the pattern, we see that the first input must be an operating-region interval and the second a limit. The first input and the output variables must be state variables, while the second input variable must be a parameter. The output interval is formed by applying the DOMAIN operation to the input intervals. The **(RubA)** rule is applied again. The idea is that if we need a state variable to take on every value in a certain operating-region, and we have some limited choices of parameters in the equation, then the other state variable must take on values over a sufficiently large interval to satisfy the equation with at least one of the parameters available. If we need torques up to 8 to drive the stirrer, we can match the specifications with the pattern:

$$\begin{array}{lcl}\langle\mathbf{R}\ \leftrightarrow^{every}\ t_o\ 0\ 8\rangle & \sim & \langle(\mathbf{R\ A})\ \leftrightarrow^{every}\ s_1\rangle\\ \langle\mathbf{A}\ [\ ]^{only}\ ratio2\ 4\rangle & \sim & \langle(\mathbf{R\ A})\ [\ ]^{only}\ p\rangle\\ \frac{t_o}{ratio}-t_i=0 & \sim & g(v_1,v_2,v_3)=0.\end{array}$$

We therefore apply the DOMAIN operation to form $\langle\mathbf{R}\ \leftrightarrow^{every}\ t_i\ 0\ 2\rangle$; the motors are required to supply torques throughout the operating-region from 0 to 2. Note that this specification does not imply that the input torque t_i can never be greater than 2, but rather that all motors considered must be able to supply torques of at least 2. If at some point the transmissions of ratio 4 are eliminated from consideration, a new labeled interval requiring higher t_i will be generated.

Table 3 shows all the propagation operations. Symbols representing the associated equations are omitted for brevity. The list "(p s)" may be matched against either a parameter or a state variable.

The $\uparrow$ and $\downarrow$ operations, given intervals in a variable, extend the interval upward to infinity or downward to zero respectively. The $\overset{some}{\cdots}$ label indicates that the variable must take on at least one value in the interval; see [2] for details.

$\langle \mathbf{A} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{A} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle \mathbf{A} \overset{every}{\leftrightarrow} s_3 \text{ RANGE} \rangle$

$\langle \mathbf{R} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{R} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle \mathbf{R} \overset{every}{\leftrightarrow} s_3 \text{ RANGE} \rangle$

$\langle \mathbf{N} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle \mathbf{N} \overset{every}{\leftrightarrow} s_3 \text{ RANGE} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} (p_1\ s_1) \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} (p_2\ s_2) \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{only}{[\ ]} (p_3\ s_3) \text{ RANGE} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} (p_2\ s_2) \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{every}{\leftrightarrow} s_3 \text{ DOMAIN} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} s_2 \rangle \longrightarrow \langle \mathbf{R} \overset{only}{[\ ]} p_3 \text{ SUFPT} \rangle$

$\langle \mathbf{N} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{only}{[\ ]} p_2 \rangle \longrightarrow \langle \mathbf{N} \overset{every}{\leftrightarrow} s_3 \text{ DOMAIN} \rangle$

$\langle \mathbf{N} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\text{A R}) \overset{only}{[\ ]} (p_2\ s_2) \rangle \longrightarrow \langle \mathbf{N} \overset{every}{\leftrightarrow} s_3 \text{ RANGE} \rangle$

$\langle \mathbf{A} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{only}{[\ ]} p_2 \rangle \longrightarrow \langle \mathbf{N} \overset{only}{[\ ]} s_3 \text{ RANGE} \rangle$

$\langle \mathbf{R} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle \mathbf{R} \overset{only}{[\ ]} p_3 \text{ SUFPT} \rangle$

$\langle \mathbf{R} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle \mathbf{R} \overset{every}{\leftrightarrow} s_3 \text{ DOMAIN} \rangle$

$\langle \mathbf{N} \overset{only}{[\ ]} (p_1\ s_1) \rangle \;\&\; \langle (\text{A R}) \overset{only}{[\ ]} (p_2\ s_2) \rangle \longrightarrow \langle \mathbf{N} \overset{only}{[\ ]} (p_3\ s_3) \text{ DOMAIN} \rangle$

$\langle \mathbf{R} \overset{only}{[\ ]} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{only}{[\ ]} p_2 \rangle \longrightarrow \langle \mathbf{R} \overset{only}{[\ ]} s_3 \text{ DOMAIN} \rangle$

$\langle \mathbf{R} \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{only}{[\ ]} p_@ \rangle \longrightarrow \langle \mathbf{R} \overset{every}{\leftrightarrow} s_3 \text{ RANGE} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{only}{[\ ]} (p_2\ s_2) \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{some}{\cdots} s_3 \text{ SUFPT} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{every}{\leftrightarrow} s_2 \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{some}{\cdots} s_3 \text{ SUFPT} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{some}{\cdots} s_2 \rangle$
$\longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{every}{\leftrightarrow} s_3\ (\text{DOMAIN}(s_1,s_2) \cap \text{DOMAIN}(\uparrow (s_1),s_2)) \rangle$
$\cup$
$(\text{DOMAIN}(s_1,s_2) \cap \text{DOMAIN}(\downarrow (s_1),s_2))$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{every}{\leftrightarrow} s_1 \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{some}{\cdots} s_2 \rangle$
$\longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{some}{\cdots} s_3\ (\text{SUFPT}(s_1,s_2) \cap \text{SUFPT}(\uparrow (s_1),s_2)) \rangle$
$\cup$
$(\text{SUFPT}(s_1,s_2) \cap \text{SUFPT}(\downarrow (s_1),s_2))$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{only}{([\ ]} \overset{some}{\cdots}) (p_1\ s_1) \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{some}{\cdots} s_2 \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{some}{\cdots} s_3 \text{ RANGE} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{only}{([\ ]} \overset{some}{\cdots}) (p_1\ s_1) \rangle \;\&\; \langle (\mathbf{R}\ \mathbf{A}) \overset{some}{\cdots} s_2 \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{only}{[\ ]} p_3 \text{ RANGE} \rangle$

$\langle (\mathbf{R}\ \mathbf{A}) \overset{some}{\cdots} s_1 \rangle \;\&\; \langle \mathbf{N} \overset{only}{[\ ]} p_2 \rangle \longrightarrow \langle (\mathbf{R}\ \text{ub}\mathbf{A}) \overset{some}{\cdots} s_3 \text{ DOMAIN} \rangle$

Table 3: Inference patterns using equations.

5.4 Review of the literature

We have been influenced by a number of general ideas. De Kleer[4] argued that much of our knowledge about the physical world is left implicit by classical mechanics. "Constraint propagation" can be traced to Sutherland[5]. "Silicon compilers"[6] suggested that design operations could be regarded as transformations of formal descriptive languages. Chapman[7] argued that "partially completed plans" represent sets of possible plans; we have directly adapted this idea to physical artifacts.

Work using artificial intelligence methods to study mechanical design can be arranged along a spectrum of increasing abstraction from human design activity. At the most abstract point of this spectrum, Fitzhorn and his students are using Turing machine models to establish fundamental conclusions about the design process[8], while Yoshikawa[9] views design descriptions as topologies on a space similar to our artifact space. Conversely, at the "human model" end, Waldron and Waldron[10], and Ullman and Dietterich[11] study human designers using the methods of the social sciences.

Toward the "human model" end, Shin-Orr[12], Brown[13], and Mittal, Morjaria, and Dym[14], have developed "expert systems" to design multiple-spindle gear drives, air-cylinders, and paper-paths respectively. These programs use hierarchical control, trial solutions and back-tracking. They apply heuristics obtained by studying experts, and appear to give nearly expert performance in narrowly defined domains.

Near the center of the spectrum we might place work focusing on a single strategy. The Dominic series of programs by Dixon and his students[15], implement a modified "hill-climbing" procedure, searching from point to point in the design space. Problems like the Toscanini's are coded by the programmer, rather than assembled from schematics, but much of the system is independent of the particular problem. Also by Dixon and his students are a series of works on "design by features"[16]. Features are geometrically oriented entities (corners, bosses). It appears that compatibility between mating features must be maintained by "hard code", and that in general the systems warn if constraints are violated, rather than using constraints to set values. Papalambros[17], and Rinderle[18] are working on a variety of design support issues and tools, spread across the spectrum.

Our work belongs with a cluster slightly further toward the more abstract end of the spectrum. Ulrich and Seering [19] use "generate, test, and debug" schemes to transform differential equations into schematics, and schematics into more specific pictorial representations. The program does not use quantitative methods for elimination or optimization, instead presenting the human designer with a variety of alternatives. Wood and Antonsson[20, 21] have been exploring the use of fuzzy set theory and fuzzy arithmetic in analyzing designs.

A version of the idea that designs represent sets of artifacts appeared in Requicha's[22] theoretical study of geometric tolerancing.

Agogino and Cagan[23] extend formal optimization methods, for example deriving a torsion tube from a torsion bar by dividing the moment integral into two regions and optimizing over them.[23]

Finally, a good deal of work at about this level of abstraction uses constraint propagation. Gossard and his students explore "variational geometry", in which systems of equations are tied to geometric descriptions of parts. Much of this work has been directed to issues of computational efficiency, but see Serrano[24] for a system that allows the designer to use schematics in building an equation network for analysis (not compilation) of a mechanical design. Popplestone[25] et al have used an algebraic constraint propagator as part of a very large system with similar goals. Gross[26]proposes a similar system for architectural design. Fleming[27], propagates the geometric tolerances of parts. Steinberg et al[28] have partially integrated top-down refinement (in the sense of progressively dividing a design problem into modules) and constraint propagation with a hill-climbing mechanism.

These constraint propagation systems, and our earlier work[29, 30], propagate equalities only, or else give intervals the *limit* interpretation and propagate them using only equivalents to the *range* operation. However, see Lozano-Pérez et al[31] for a generalization of the *domain* operation, the *pre-image*, used to formulate robot motion plans under uncertainty.

5.5 A summary, with contrasts

We can now summarize our work, emphasizing points of contrast with the "mechanical design" programs mentioned above.

- We provide an explicit and "compositional" high level language in which designers can define new systems and problems. Formal operations on this language automatically transform high-level descriptions into detailed descriptions.
- Our design descriptions explicitly represent *sets of artifacts and operating conditions*, rather than a single or "archetypical" artifact under a single operating condition.
- We search by progressively narrowing volumes of the artifact space, rather than from point to point in that space.
- We add the *domain* and *sufficient-points* operations on intervals to the *range* constraint-propagating operation.
- We add the *operating region* interpretation of intervals to the *limit* interpretation.
- We divide specification statements into those which are true of all the artifacts represented (Assured), those which must be true of the final design (Required), and those which may or may not be true but which cannot be strengthened (No-stronger-possible).
- We divide variables into "causality classes" (parameters vs state-variables).

We believe these concepts are sufficient to support design compilation over a much of the power transmission system domain. (We have not attempted to design servo-systems, or use components which are not pre-cataloged.) The design compiler program we have written tests this hypothesis.

6 Results

We have tested our design compiler on more than a dozen arrangements of components. The data base for these tests includes specifications on such quantities as motor speeds and torques, pump pressure ratings, efficiencies, and displaced volumes, etc, as well as the related power transmission equations. Also included are equations for determining such quantities as ball screw critical frequency and buckling load.

There are often multiple solutions satisfying the specifications for a given problem. Also, because information is lost in abstracting, a group of alternatives may "hide" some unsatisfactory ones. The system therefore interleaves elimination steps with search steps, looking for the optimum solution as defined by a user-supplied objective function.

Fig. 7 shows an example component arrangement. The numbers in the schematic symbols indicate the number of line items in the catalogs represented; there are initially 404,352 possible combinations. With reasonable user specifications (for example on the speeds at which the loads should move and the forces required to move them) and a utility function (say summing the cost and weight), the system takes about 10 minutes[1] to find an optimal solution.

Testing continues; we defer a detailed discussion of the capabilities and limitations of the inference system and the search procedure to [2]. We present here conclusions based on example problems solved to date.

- Our language is powerful enough to capture most of the information provided in catalogs for a wide range of components.
- Our inference operations appear to result in only correct eliminations when components are appropriately modelled.

[1] On a Symbolics 3640 Lisp machine, running our unoptimized research code.

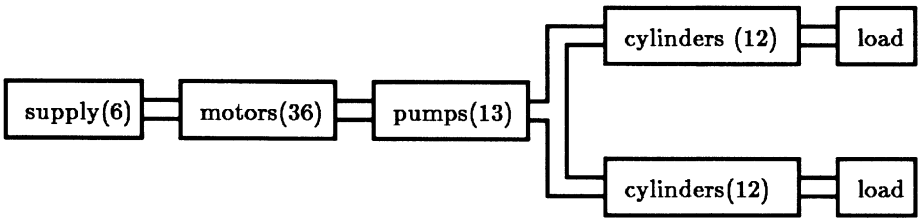

Figure 7: A hydraulic example

- The idea that design descriptions represent sets of artifacts and operating conditions is powerful. We could not have worked through the complexities of the theory without it.
- Our work is incomplete. For example, the adjustable position of the seat back in a automobile is neither a parameter nor a state variable. We are now exploring relationships between variables and intervals additional to Limits ($\overset{only}{[\ ]}$) and Operating-Regions ($\overset{every}{\leftrightarrow}$). We are at present restricted to invertible algebraic equations and to positive values for variables. We have explored geometric issues only very lightly. Our abstraction procedures can be made stronger. More subtly, "hard-edged" interval representations such as ours distort the real significance of some specifications[32].

References

[1] Gerhard Pahl and Wolfgang Beitz. *Engineering Design.* The Design Council, 1984.

[2] Allen C. Ward. *A Theory of Quantitative Inference for Artifact Sets, Applied to a Mechanical Design Compiler.* PhD thesis, Massachusetts Institute of Technology, 1989.

[3] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32, 1987.

[4] Johan de Kleer. Qualitative and quantitative knowledge in classical mechanics. Master's thesis, Massachusetts Institute of Technology, 1975.

[5] I. Sutherland. Sketchpad—a man-machine graphical communication system. Technical Report 296, Lincoln Laboratory, Massachusetts Institute of Technology, 1963.

[6] A. Jerraya, P. Varinot, R. Jamier, and B. Courtois. Principles of the SYCO compiler. In *Proceedings of the 23rd Design Automation Conference.* IEEE, 1986.

[7] David Chapman. Planning for conjuntive goals. Technical Report 802, MIT Artificial Intelligence Lab, 1985.

[8] Patrick Fitzhorn. A computational theory of design. *Design Computing*, 3(1), 1988.

[9] H. Yoshikawa. *Design Theory.* University of Tokyo, 1986.

[10] M. B. Waldron. Modeling of the design process. In *Proceedings of the 1988 IFIP 5.2 Intelligent CAD Workshop.* North-Holland, 1988.

[11] David G. Ullman and Thomas G. Dietterich. Mechanical design methodology: Implications for future developments of computer-aided design and knowledge-based systems. *Engineering with Computers*, 2, 1987.

[12] Chaim D. Shin-Orr. *Automatic Design of Complex Gear Trains.* PhD thesis, Massachusetts Institute of Technology, 1976.

[13] David Brown. Capturing mechanical design knowledge. In *Proceedings of the 1985 ASME International Computers in Engineering Conference*, Boston, MA, 1985. ASME.

[14] S. Mittal, C. L. Dym, and M. Morjaria. PRIDE: An expert system for the design of paper paths. In *Applications of Knowledge-based Systems to Engineering Analysis and Design.* ASME, New York, NY, 1985.

[15] M. F. Orelup, J. R. Dixon, and M. K. Simmons. Dominic II: More progress toward domain independent design by iterative redesign. In *Proceedings of the ASME Winter Annual Meeting.* ASME, 1987.

[16] J. R. Dixon, E. C. Libardi Jr., S. C. Luby, M. Vaghul, and M. K. Simmons. Expert systems for mechanical design: Examples of symbolic representations of design geometries. In *Applications of Knowledge-based Systems to Engineering Analysis and Design.* ASME, New York, NY, 1985.

[17] Panos Y. Papalambros. The design laboratory: Interdisciplinary research and education. In *Proceedings of the 1988 NSF Grantee Workshop on Design Theory and Methodology*, Rensselaer Polytechnic Institute, Troy, NY, 1988.

[18] James R. Rinderle, Eric R. Colburn, Stephen P. Hoover, Juan Pedro Paz-Soldan, and John D. Watton. Form-function characteristics of mechanical designs—research in progress. In *Proceedings of the 1988 NSF Grantee Workshop on Design Theory and Methodology*, Rensselaer Polytechnic Institute, Troy, NY, 1988.

[19] Karl T. Ulrich and Warren P. Seering. Conceptual design: Synthesis of novel systems of components. In *Proceedings of the 1987 ASME Winter Annual Meeting—Symposium on Intelligent and Integrated Manufacturing Analysis and Synthesis.* ASME, 1987.

[20] Kristin L. Wood and Erik K. Antonsson. Computations with imprecise parameters in engineering design: Background and theory. Technical Report 88-01, Engineering Design Research Laboratory, California Institute of Technology, 1988.

[21] Kristin L. Wood and Erik K. Antonsson. Computations with imprecise parameters in engineering design: Applications and examples. Technical Report 88-02, Engineering Design Research Laboratory, California Institute of Technology, 1988.

[22] Aristides A. G. Requicha. Toward a theory of geometric tolerancing. *International Journal of Robotics Research*, 2(4), 1983.

[23] Jonathan Cagan and Alice M. Agogino. Innovative design of mechanical structures from first principles. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 1988.

[24] David Serrano and David Gossard. Constraint management in conceptual design. In *Knowledge Based Expert Systems in Engineering, Planning and Design.* Computational Mechanics Publications, 1987.

[25] R. J. Popplestone. The Edinburgh Designer system as a framework for robotics: the design of behavior. *Artifical Intelligence for Engineering Design, Analysis, and Manufacturing*, 1(1), 1987.

[26] Mark Donald Gross. *Design as Exploring Constraints.* PhD thesis, Massachusetts Institute of Technology, 1985.

[27] Alan Fleming. Analysis of uncertainties in a structure of parts. In *International Joint Conference on Artificial Intelligence*, 1985.

[28] Jack Mostow, Lou Steinberg, Noshir Langrana, and Chris Tong. A domain-independent model of knowledge-based design: Progress report to the National Science Foundation. Technical Report Working Paper 90-1, Rutgers University, 1988.

[29] Allen C. Ward and Warren Seering. An approach to computational aids for mechanical design. In *Proc. 1987 International Conference on Engineering Design.* ASME, 1987.

[30] Allen C. Ward and Warren Seering. Representing component types for design. In *Advances in Design Automation—1987.* ASME, 1987.

[31] Tomás Lozano-Pérez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.

[32] Genichi Taguchi. *Introduction to Quality Engineering.* UNIPUB, White Plains, NY, 1986.

CS-TR Scanning Project
Document Control Form

Date : 4/27/95

Report # AIM-1062

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 16(22-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form(2) ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP (1) UN#'ED TITLE PAGE	
(2-16) PAGES #'ED 1-15	
(17-19) SCANCONTROL, DOD(2)	
(20-22) TRCTS (3)	

Scanning Agent Signoff:
Date Received: 4/27/95 Date Scanned: 5/1/95 Date Returned: 5/4/95

Scanning Agent Signature: Michael W. Cobb

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Completed 8/3/89

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AIM 1062	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A209900
4. TITLE (and Subtitle) Quantitative Inference in a Mechanical Design Compiler		5. TYPE OF REPORT & PERIOD COVERED memorandum 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Allen C. Ward and Warren Seering		8. CONTRACT OR GRANT NUMBER(s) N00014-86-K-0685
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE January 1989 13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

quantitative inference
qualitative reasoning
design
constraint propagation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents the ideas underlying a computer program that takes as input a schematic of a mechanical or hydraulic power transmission system, plus specifications and a utility function, and returns catalog numbers from predefined catalogs for the optimal selection of components implementing the design. Unlike programs for designing single components or systems, this program provides the designer with a high level "language" in which to compose new designs. It then performs some of the detailed design process for him.

The process of "compilation", or transformation from a high to a low level description, is based on a formalization of quantitative inferences about hierarchically organized sets of artifacts and operating

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Completed 8/5/89

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS

1. REPORT NUMBER: AIM 1062

2. GOVT ACCESSION NO.: AD-A 304 400

3. RECIPIENT'S CATALOG NO.:

4. TITLE (and Subtitle): Quantitative Compiler

5. TYPE OF REPORT: Block 20 cont.

conditions. This allows design compilation without the exhaustive enumeration of alternatives. The paper introduces the formalism, illustrating its use with examples. It then outlines some differences from previous work, and summarizes early tests and conclusions.

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Allen C. Ward and Warren Seering

8. CONTRACT OR GRANT NUMBER(s): N00014-86-K-0685

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS:
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

12. REPORT DATE: January 1989

13. NUMBER OF PAGES: 16

14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office):
Office of Naval Research
Information Systems
Arlington, VA 22217

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report):
Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:
None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number):
quantitative inference
qualitative reasoning
design
constraint propagation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

DD FORM 1473 (JAN 73) EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 5/1/1995

M.I.T. Libraries
Document Services