A Transformational Approach to Mechanical Design Using a Bond Graph Grammar

by

S. Finger, J. R. Rinderle

EDRC 24-23-90 C,3

hirst ASMb Design Theory and Methodology Conference, Montreal, Quebec, September 1989

A Transformational Approach to Mechanical Design Using a Bond Graph Grammar

Susan Finger
James R. Rinderle

Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

During the design process, a designer transforms an abstract functional description for a device into a physical description that satisfies the functional requirements. In this sense, design is a transformation from the functional domain to the physical domain; however, this transformation process is not well characterized nor understood for mechanical systems. The difficulty arises, at least in pan, because mechanical designs are often composed of highly-integrated, tightly-coupled components where the interactions among the components are essential to the behavior and economic execution of the design. This assertion runs counter to design methodologies in other engineering fields, such as software design and circuit design, that result in designs in which each component fulfills a single function with minimal interaction. Because of the geometry, weight, and cost of mechanical components, converting a single behavioral requirement into a single component is often both impractical and infeasible. Each component may contribute to several required behaviors, and a single required system behavior may involve many components. In fact, most mechanical components perform not only the desired behavior, but also many additional, unintended behaviors. In good mechanical designs, these additional behaviors often are exploited.

The long term goal of our research is to create a transformational strategy in which the design specifications for a mechanical system can be transformed into a description of a collection of mechanical components. To realize this goal requires formal representations for the behavioral and the physical specifications of mechanical systems as well as formal representations for the behaviors and the physical characteristics of mechanical components. Because the interactions of components are important in our synthesis strategy, the representation of the behaviors of mechanical components must be linked to the representation of their physical characteristics; that is, we are concerned with modeling the relationship between form and function of components. Finally, we need a strategy that enables us to transform an abstract description of the desired behavior of a device into a description that corresponds to a collection of available physical components.

In this paper, we present a graph-based language to describe both the behavioral specifications of a design as well as the behavior of the available physical components. We also briefly discuss a graph-based grammar for the representation of the physical characteristics of the components that enables us to guide the translation from specifications to components [Pinilla 89]. The transformation strategy is discussed in a companion paper [Hoover 89].

Introduction

During the design process, t designer transforms an abstract functional description for a device into a physical description that satisfies the functional requirements.[1] In this sense, design is a transformation from the functional domain to the physical domain [Mostow 85, Rinderle 82]; however, the basis for selecting appropriate transformations and methods for accomplishing transformations are not well understood. The implicit basis for design transformations in circuits [Steinberg 86], software [Wirth 71], and some architectural applications [Fcnvcs 87] result in a degree and type of modularity not well suited to mechanical devices [Rinderle 86].

Good mechanical designs are often composed of highly-integrated, tightly-coupled components where the interactions among the components are essential to the behavior and economic execution of the design. This assertion runs counter to design methodologies in other engineering fields, such as software design and circuit design, that result in designs in which each component fulfills a single function with minimal interaction. Because of the geometry, weight, and cost of mechanical components, convening a single behavioral requirement into a single component is often both impractical and infeasible. Each component may contribute to several required behaviors, and a single required system behavior may involve many components. In fact, most mechanical components perform not only the desired behavior, but also many additional, unintended behaviors. In good mechanical designs, these additional behaviors often are exploited.[2]

[1]Mechanical engineers tend to use the words function and behavior interchangeably. Qualitative physicists make a distinction between these »ords: that is, the *design's function* is what it is used for, while its *behavior is* *hai ü does. For example, the function of a clock is to display the time, but its behavior might be the rotation of hands. Similarly, a motor may be designed to function as a prime mover, but can also function as a door stop because it has additional behaviors due to its mass. In this paper. *Junction* is used to indicate ihe subset of *behaviors* which are required for the device to perform satisfactorily.

[2]This statement does not contradict the design axioms put forth by Suh (Suh 80, Suh 88]. The design axioms state that good designs maintain independence of *functional* requirements and minimize the information content of the design. Suh points out that by integrating functions into t single component, information content may be reduced without compromising the independence of functional requirements.

The direct transformation of behavioral requirements into physical components may result in undesirable designs for two reasons. The first is that matching individual behaviors directly to components does not enable the integration of behaviors into more compact or more economical collections of physical components. The second reason is that physical components have not only the desired behavior and physical characteristics but also many additional, incidental behaviors and characteristics. The appropriate device configuration and selection of components depends to some extent on exploiting or compensating for these incidental behaviors.

By creating a formal description of a limited set of behaviors for mechanical designs and a corresponding description of physical components, we can generate the description of a physical system that takes advantage of the multiple behaviors of its components. This paper focuses on the grammar that underlies the transformation from behavioral specifications to physical components. The transformation strategy is discussed in a companion paper [Hoover 89].

Overview of Our Approach

The goal of our research is to create a transformational strategy by which the design specifications for a mechanical system can be transformed into a description of a collection of mechanical components. Both behavioral and physical *requirements* as well as behavioral and physical *characteristics* of the available mechanical components must be represented to execute such a transformational approach to design. We are investigating the use of representations based on formal grammars to facilitate the characterization of our approach with respect to completeness, complexity, etc. and to take advantage of the advances in formal language theory. Because the interactions of components are important in our synthesis strategy, the representation of the behaviors of mechanical components must be linked to the representation of their physical characteristics; that is, we are concerned with modeling the relationship between form and function of components. Finally, we need a strategy that enables us to transform an abstract description of the desired behavior of a device into a description that corresponds to a collection of physical components. This paper primarily addresses the first issue, that of formal representations of behavior.

To realize the goal of formalizing the transformation from the behavioral to the physical domain, we have begun to explore a small domain within mechanical design, the domain of gear box design. Clearly, one reason for selecting this domain is that gear box design is a well-understood, highly-parameterized area of mechanical design. Nevertheless, we believe that our representation and transformation formalism will be applicable in other mechanical design domains, particularly to the class of design problems that we call *configuration design.* By configuration design, we mean designs composed from standard component families but for which allowable configurations are not specified *a priori.*

Our approach is based on the following assertions:

- The behavioral *requirements* of mechanical systems can be represented using a graph grammar based on bond graphs.
- The behavioral *characteristics* of components can be represented using a graph grammar based on bond graphs.
- The physical characteristics of designs and components can be represented using an augmented topology and geometry graph.
- The behavioral and physical graphs of components can be linked parametrically.
- The behavioral specifications graph can be transformed into a description of a physical system with associated behavioral and geometric representations.

Representational issues will be discussed at greater length in subsequent sections; however, we note here that our underlying representation for behavior is based on bond graphs [Paynter 61]. Usin^ bond graphs, we can construct a formal grammar that rives us a general representation of classes of mechanical behavfor. lit common practice, bond graphs are constructed to model the behavior of physical systems. We use bond graphs not only to model the behavior of physical systems, but also to represent behavior in the abstract, as with a design specification. Thus a device configuration can be generated by transforming a specification bond graph into a functionally equivalent graph which corresponds to a configuration of available components. The type of graph transformations used are those that decompose, aggregate, and redistribute graph primitives.

A major advantage of using bond graphs to represent design requirements is that we can define transformation rules that alter the structure of the bond graph but that do not alter the dynamic behavior of the system represented by the graph. The implications of this statement are important Because we can transform the specifications graph to represent many different physical systems, we do not impose an initial structure or configuration on the physical design; that is, we do not require an *a priori* decomposition of the design specifications.

To complete this methodology, we plan to represent the physical characteristics of designs and components using another graph grammar that is based on an augmented topology graph [Pinilla 89] and non-manifold geometry [Gursoz 89]. We plan to link parametrically the bond graph representation and the topology and geometry graph representation. With a geometric representation, characteristics such as the volume or mass can be modeled and computed, and from the bond graphs, the dynamic behavior of the final design can be modeled. In this paper, wfe do not discuss the geometric representation in detail because our work on linking the geometry and behavior is preliminary.

A Brief Introduction to Bond Graphs

Bond graphs, which were created by Paynter [Paynter 61], provide a convenient and uniform representation for the dynamic behavior of a broad class of physical systems, including those within the mechanical, electrical, hydraulic, thermal, and biological domains. Bond graphs have been used extensively in a wide variety of application areas including robotic manipulators [Margolis 79], torque conveners [Hrovat 85], and vacuum cleaners [Remmerswaal 85]. A brief introduction to bond graphs is given here. For a complete discussion, see Karnopp and Rosenberg [Kamopp 75].

Bond graphs enable mechanical and hydraulic systems to be represented in a manner equivalent to electric circuit diagrams. For example, a spring in a mechanical device acts like an electrical capacitor by storing and releasing energy. To illustrate the similarity, Figure 1 shows an RLC circuit with its equivalent bond graph and a mass-spring-damper system with its equivalent bond graph. While Figure 1 illustrates separate electrical and mechanical systems, one of the most powerful attributes of bond graphs is that they can be used to model integrated electrical, mechanical, and hydraulic systems.

Using bond graphs, physical systems are represented as a graph of lumped-parameter, idealized elements. Power is the currency of bond graphs; power flows through the bonds (edges) in the graph, and power is dissipated, stored, supplied, and transformed at the ports (vertices) in the graph.

The ports, or vertices, of bond graphs are divided into three categories:

- 1-port elements dissipate power, store energy, and supply power. Dampers, springs, and masses arc the mechanical elements represented by the passive 1-port elements. Force (effort) and velocity (flow) sources are represented as active 1-ports.
- 2-port elements transform power. Transformers axe 2-port elements that represent an imposed proportionate relationship between similar quantities, e.g. a gear pair constrains rotational speeds. Gyrators are a 2-port

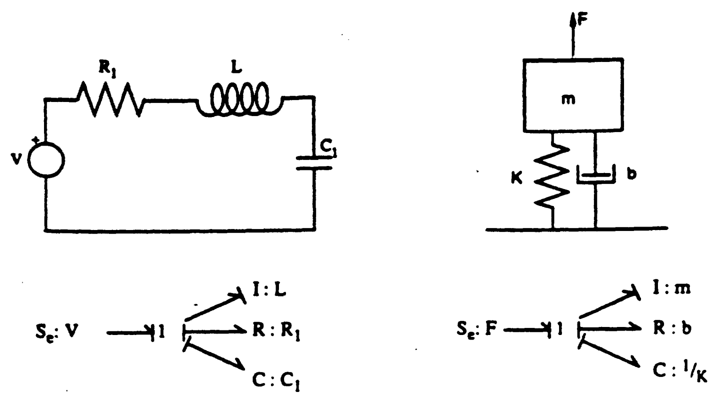

Figure 1: Simple systems with corresponding bond graphs

elements that impose a proportionate relationship between dual quantities, e.g. a torque converter constrains the relationship between torque and angular velocity. Power is conserved across a 2-port.

- N-port elements represent the structure of the system corresponding to the connections among the elements. There are two types of N-port elements: 0-junctions and 1-junctions, which correspond respectively to "same force" and "same velocity" connections. Equivalents of Kirkoff's laws apply to N-port elements: the sum of the efforts around a 1-junction is zero (the bonds share a common flow); the sum of the flows around a 0-junction is zero (the bonds share a common effort). N-port elements are power conserving.

Because bond graphs can be used to model electrical, mechanical, and hydraulic systems, problems with standard notation and terminology arise. For example the symbol, *v*, is commonly used for *velocity* in mechanical systems and for *voltage* in electrical systems. In circuits, it is common to speak of a voltage source, but speaking of a force source in mechanics is awkward. Table 1 gives a common notation for bond graph equivalents in the electrical and mechanical domains.

The bond graphs shown in Figure 1 consist of 1-port elements, n-port elements, and power bonds. Power bonds are indicated by the presence of a half-arrow, the direction of which establishes the sign convention for power flow. Power bonds are usually shown with a causal stroke at one end. The causal strokes play an important role in identifying device states, formulating system equations, and interpreting causal structure, but they, like the power direction half-arrow, may be considered as augmentations to the basic bond graph. Although they are included in the illustrations, neither the causal strokes nor the half-arrows enter into representational or grammatical considerations. Other bond graphs elements, notably field structures and active bonds, are not considered here.

Figure 2 shows a schematic of a motor, drive shaft, gear, and load system with an equivalent bond graph. In this bond graph, the motor and its controller have been idealized as a source of torque; however, the drive shaft compliance, gear inertia, and bearing losses have been explicitly included. The overall behavior of the transmission element is the aggregated behavior of these behavior primitives.

The use of bond graphs in our representation and grammar will be discussed in following sections.

		Generalized Bond Graphs	Electrical Systems	Mechanical Systems
Variables		effort, e	Voltage, v	Force, F
		flow, f	Current, i	Velocity, v
		Power = e f	Power = v i	Power = F v
Source Elements		Effort Source	Voltage Source	Applied Force
		Flow Source	Current Source	Prescribed Velocity
Passive Elements		Resistance R $e = R f$	Resistor R $v = i R$	Damper b $F = b v$
		Compliance C $f = C\,dv/dt$	Capacitor C $i = C\,dv/dt$	Spring 1/k $dF/dt = k v$
		Inertance I $e = I\,df/dt$	Inductor L $v = L\,di/dt$	Inertia m $F = m\,dv/dt$
Structure		0 Junction Common Effort	Parallel	Same Force
		1 Junction Common Flow	Series	Same Velocity

Table 1: Equivalents for electrical, mechanical and generalized bond graph elements

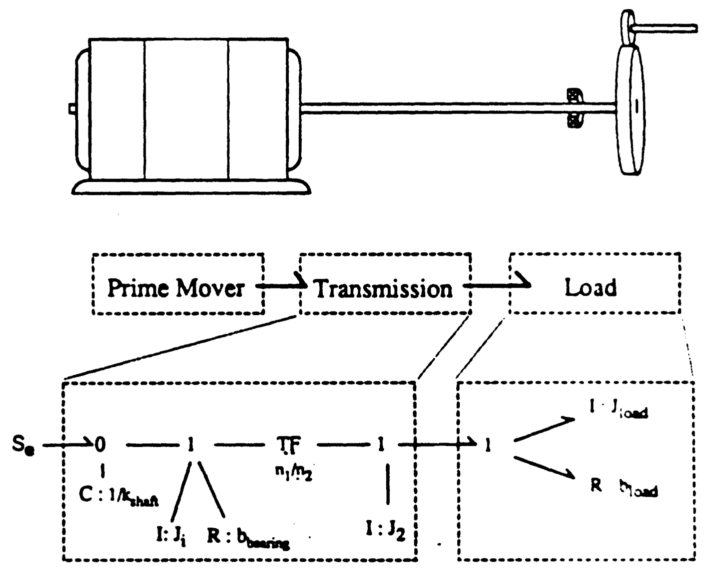

Figure 2: Drive system and associated bond graph

Related Work

Our research builds upon research from several different areas including representation of mechanical behavior, grammars for shape representation, configuration design, and bond graphs. In this section, we briefly discuss related research in these areas.

Representation of Function

A transformational strategy for design requires a representation of the device specifications and the components used in the device. The representation of the function and behavior of mechanical designs has been explored by, among others, Lai [Lai 87], Crossley [Crossley 80], Pahl and Beitz [Pahl 84]. The function structures of Pahl and Beitz provide a graphical system for laying out the functions of a design. In this system, functions such as "mix" or "deliver" are arranged in a graph to represent the overall function of the design. This is similar to our idea of a specifications graph, discussed below; however, there are several important differences. Pahl's work does not discuss how to integrate form specifications or functional constraints with this functional representation. Therefore, while the function structures are used to study various configurations in Pahl's synthesis strategy, there is little guidance for transforming the function structure to a physical description of the device. Lai has created a formal, English language-based system called FDL for representing the function and structure of mechanical designs. In FDL, nouns and verbs are used to create sentences that represent the function of a design, and *design* rules operate directly on the nouns and verbs in the sentence. Allowable verbs, for example "fasten," do not have physical or mathematical representations and so their meaning is determined by the rules that use them. While the FDL language can represent the function and form of a design it provides no assistance in transforming a functional description into a physical description.

More closely related to our approach for representing behavior is the work of Fenves and Baker [Fenvcs 87] and of Ulrich and Seering [Ulrich 87, Ulrich 88, Ulrich 89]. Fenves has created a spatial and functional representation language for structural designs. They use operators that execute a known grammar to generate architectural layouts as well as structural and functional configurations; however, they assume that the layout and structure are independent if they are generated sequentially. Ulrich and Seering also use a formal representation of function based on bond graphs; however, their goal is to create a system in which the bond graphs for single-input, single-output dynamic systems can be automatically synthesized and transformed into physical components. Using a strategy they call *design and debug,* they transform a graph of design requirements directly to functionally independent physical components. Reconfiguration for function sharing is performed after the components have been selected.

Grammars for Shape Representation

Formal grammars that can be used to represent, generate, and parse valid strings in a language have proven useful in a number of fields, most notably, linguistics and computer science. Recently, interest has been growing on the use of formal grammars in engineering design. Our work draws mainly from these engineering applications of grammars. We are specifically concerned with graph grammars, the class of grammars that operates on graphs. Tutorials on graph grammars and their applications are given by Ehrig [Ehrig 87] and by Nagl [Nagl 87].

One of the earliest uses of grammars in design was by Stiny [Stiny 75] who created shape grammars based on the formalisms of computational linguistics [Chomsky 57]. Architects in particular have been interested in shape grammars, using them to generate a family of floor plans or ornamentation. Fitzhorn [Fitzhorn 89] has shown the formal relationship between language theory and solid modeling systems. He shows that a variant of a graph grammar can produce three-dimensional solids. He creates three grammars, one of which generates the constructive solid geometry representation, the second of which generates the boundary representation, and the third of which generates plane models.

Pinilla *et al.* [Pinilla 89] have created a grammar that can be used to describe and represent the geometric features of a design. They use a non-manifold topological representation of a design to create a general, but formal, representation of form features. This system is currently being extended to enable feature-based designs to be generated, represented, and parsed. This extension if possible because the underlying representation of a feature is based on elements of a well-defined grammar.

Configuration Design

Configuration design and parametric design are active areas of research in mechanical engineering design. For a more complete discussion of this body of work, see [Finger 89]. Our research combines configuration design and parametric design because we generate both the structure of the design and the individual components. Many design systems, such as HI-RISE [Mahcr 85], AIR-CYL [Brown 85], and VT [Marcus 86], utilize either a set of predetermined decompositions of the structure of a design, or utilize design methods that generate, as part of the design process, a decomposition belonging to such a set. Therefore, all of the designs generated by these systems will share, at a relatively low level, a structural similarity. For design domains in which the most desirable structures can be enumerated or explicitly decomposed in advance, this approach proves useful. But there are many design problems where the structural decomposition of the most useful solution is not pre-determined.

Representation of Specifications, Components, and Devices

The transformational approach to design, which we have briefly introduced, imposes some representational requirements. Specifically, the representation must:

- Express formally the design requirements of mechanical systems.
- Be compatible with the representation of the behavior of components.
- Facilitate design validation.
- Represent the required system behavior without imposing a pre-defined structure on the physical realization of the design.

We make the observation that system specifications for engineering designs are of two types; *behavioral* and *physical.* "HUMS, some specifications describe at an abstract level the desired behavior of the overall system while others describe physical restrictions or requirements on the final design. For example, the requirements for a vibration absorber might include the *behavior* of the device in terms of frequency and rejection ratio and might also specify *physical* properties such as allowable size and weight. The physical and behavioral specifications express the design objective. Since the specifications are given without regard to design configuration, they are independent of each other.[3] Physical specificauons for a design may or may not be given, while at least some behavioral specifications must be given, since the behavioral specifications express the central aspect of the design objective.

Although the behavioral and physical specifications are independent in the *functional* domain, they are coupled in the *physical* domain, because any physical arrangement results in a specific set of behaviors. In the physical domain, the physical and behavioral characteristics of a individual components depend on one another, and the behavior of the whole design depends strongly on the configuration and interaction of components. The representation of interactions is essential for our purposes since we are investigating the effects of and the means for achieving functional integration in design

*That is, they are independent unless the requirements are contradictory, for example by virtue of physical law.

Physical and behavioral specifications **and** characteristics can be represented **as** combinations of **abstract** primitives. They are abstract because each primitive corresponds to only one behavioral or physical characteristic. Individually **they** do not correspond to any particular component or configuration of components, but collectively they may represent **the** design specifications or the form and behavior or components. There **are** two important criteria that a set of primitives must satisfy. One requirement is that the set of primitives chosen be complete; **that** is, all relevant behavioral and physical characteristics must be representable by some combination of primitives. The other is that the number of primitives must be small, although not necessarily minimal. The latter requirement minimizes combinatorial problems associated with representing a single behavior in different ways. In addition, the primitive behavioral and physical elements should enable commonly used components and typical specifications to be represented easily.

Representation of the Design Specification

If the behavioral primitives correspond to bond graph elements, the behavioral specifications can be represented by a bond graph. This specification bond graph is not unique, indeed any behaviorally equivalent graph is an acceptable representation of the specifications. The specifications graph represents only the desired behavior of the design. Physical specification will be represented as an augmented topology graph which is not discussed further in this paper.

Deriving a correct specifications graph for a design problem is, in itself, a major research problem. Initially, for transmission design, we assume that deriving a specifications graph from the problem statement is straightforward. Later, as less constrained design domains are explored, we will look at the problem of constructing the specification graph from the design requirements.

Figure 3 illustrates the system specification for a system required to drive two loads at a high speed relative to the input speed. The single physical requirement is a required shaft offset and is not shown.

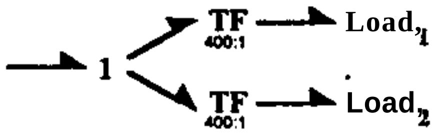

Figure 3: Specification graph for a system to drive two loads

Representation of Physical Components

Physical components, like specifications, have both behaviors and physical characteristics. Therefore, it is convenient in representing components to employ the same basic structure that is used for specifications. One crucial difference, however is that the behaviors and physical characteristics of a component are inherently linked; no single characteristic can be obtained in isolation. Each physical component is represented as an object that has a behavioral representation, a physical representation, and an explicitly represented interaction between the two. The relationships can take the form of design equations, analytical models that relate geometric characteristics to behavioral characteristics, or data base entries that prescribe a relation between the physical and the behavior characteristics. For example, the weight of a helical coil spring, which is a physical characteristic, is proportional to the product of stiffness and the square of allowable deflection, which are behavioral characteristics. These relationships are often critical during the design.

As an example consider **a** single spur gear **as a** component The behavior graph for the component is shown in Figure 4. In this example, **a** relationship is imposed between flow quantities rotational speed, co on the left and surface speed, v, on the right The topological graph, representing physical characteristics is not shown. In its place we simply show the geometric parameter corresponding to pitch diameter.

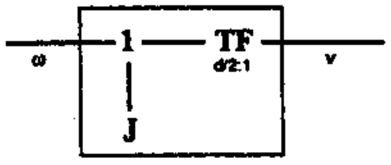

Figure 4: Behavior graph for a single spur gear

It is important to note that the completeness of the behavioral and physical representations determines the extent to which the additional behaviors can be exploited. For example, with a worm gear, the right angle shaft configuration jmd^ the self-locking property must be represented explicitly, or be derivable from the representation, for it to be recognized and utilized in the design process. Therefore, this methodology requires a known component base.

Representation of Designs

A complete design is the specified configuration of a set of components. Since the components have associated with them both behaviors and physical characteristics, the distinction between the representation of designs and the representation of components is in the representation of the behavioral and physical *configuration.* The behavioral configuration fundamentally consists of the kinematic connections, e.g. mounting to a frame, rigid connections, or rolling. Most of the common kinematic arrangements can be categorized [Reuleaux 54] and reduced to a bond graph junction structure. A 1-junction, for example, represents a common translational velocity. In this way, the behavior of complete devices may be represented in terms of the behavioral bond graphs of components and a number of bond graph junction structures representing kinematic connections. Partially complete designs also consist of components and junction structures and some number of behavioral primitives not yet associated with a component.

Ultimately, we will represent the physical characteristics of the device in much the same way, i.e. as a composite of the topology and geometry graphs of the components and configuration. To date we have represented the physical characteristics of devices by aggregating, on an *ad hoc* basis, the geometric parameters of the components and the layout. Figure 5 shows how two spur gears are represented as a combination of components joined through a kinematic connection. In this case, the components have 3 common speed at the pitch diameters of the gears.

At the device level, a physical or behavioral charactcrisnc describes an overall characteristic of the system. These characteristics may be one of two types: configuration-independent and configuration-dependent. Configuration-independent characteristics are those such as weight that depend only on the selected components and not on their configuration or interactions. Configuration-dependent characteristics, such as size and resonant frequency, depend on the configuration and interaction of the components in the final design. In either case the characteristics of the design can be determined from the completed behavioral and topological graphs, however, the implications of configuration independent characteristics can be utilized prior to design completion.

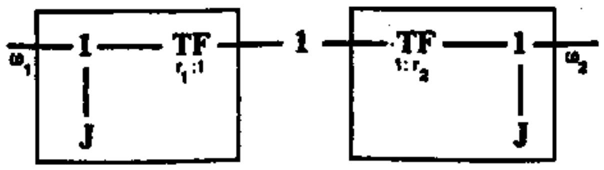

Figure 5: A behavior graph for **a** combination of two spur gears

A Bond Graph Grammar

In this section, we discuss the formalisms that enable bond graphs to be transformed from design specifications to component specifications while preserving the system behavior. We first define the graph, its elements, and some set operations on the graph such as inclusion and difference. We then define what it means for one graph (the component graph) to be derivable from another graph (the specification graph). Our primary interest is in the productions for transforming bond graphs of mechanical systems, rather than in demonstrating a grammar that generates all possible, valid bond graphs.

Graph Representation

A bond graph is a graph with vertices labeled to indicate correspondence to the various 1-ports, 2-ports and n-ports. The edges of the graph correspond to the power bonds and may be labeled to show, for example, causality. Loops may occur in a bond graph but multiple *edges* and self edges are not meaningful and are not allowed.

We define the bond graph to be an undirected graph with edge and vertex labels, that is, the bond graph, *g* is

*-<V.B).

where

V **c** X x Z_v is called the *vertex set*

B *Q* V x V x Z_e is the *edge set* with the conditions:

- V (v, v;, l_e) € B, (v,, v;, /,) * (v;, v,, /,)
- V 0v $v_{>f}$/,) € ^{B}>***J*
- if 3 (v, v;, /,) and (v,, v,, l_f) ***e*** B, /, = l_f

I_v is a finite, non-empty set of symbols called the *vertex labels*

l_e is a finite, non-empty set of symbols called the *edge labels*

and

X is a non-empty, enumeration set

V- is an element in the vertex set

l_e is an edge label in the edge alphabet

In this section, all graphs, g_{mt} are assumed to be defined on the alphabets, I_v and Z, with g_m = (B_m, VJ

For the grammar presented here, the labels of the edges are simple enumeration labels; however, in a more complete grammar for bond graphs, edge labels for causal strokes and power flow will be useful. We include in the set of vertex labels both terminal and non-terminal elements, that is, both the low level elements like resistors as well as the general elements like 1-ports. The set of vertex labels, which is large, can be generated by enumerating all the ports and port-types. Alternately, the labels could be generated by a simple string grammar that transforms n-ports to Q-junctions and 1-junctions. transforms 2-ports to transformers a**d **gyrators and so on.**

For the b'mited domain of gears, the alphabet for the vertices is:

I , - **{1-pon, source, torque source, angular velocity source, passive-element, rotary inertia, rotary damper, rotary spring,** 2-port, **transformer, gyrator, n-port,** 0-junction, 1-junction)

Basic Graph Relationships and Operations

Transformations of the specifications graph will require removing subgraphs of bond graphs and replacing them with other graphs. For example, if a specification included **a** graph representing a transformation ratio of 25:1, that subgraph would be removed, and **a** graph representing a sequence of 5:1 reductions might be inserted in its place. To perform such a transformation requires a definition of graph isomorphism so that different arrangements of the same graph can be recognized. It also requires **a** definition of graph inclusion so that subgraphs within a larger graph can be identified, that is, for example, to define what it means to say that the graph of the 25:1 reduction ratio is included in the specifications graph. Finally, it requires a definition of the difference operation between two graphs so that we can, for example, remove the 100:1 reduction from the specifications graph in order to replace it with the sequence of smaller reductions. Graph isomorphism, inclusion and difference are defined in Appendix A and are illustrated in Figure 6.

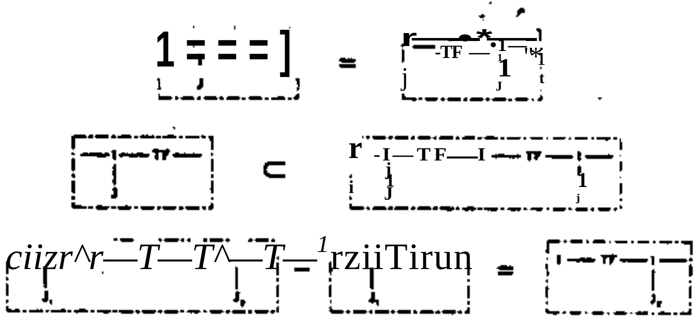

Figure 6: Graph isomorphism, inclusion, and difference

Graph Productions

We now define graph productions that enable us to remove one part of the bond graph and replace it with another. A graph *production* consists of a right hand side, a left hand side and an embedding transformation. That is, a production, *P*, transforms the left-hand graph, $g_{L\%}$ into the right-hand graph, g_R:

$$P = (g_L, g_R, E)$$

where *E* is the embedding transformation that gives the procedure for inserting the right-hand graph into the parent graph after the left hand side has been removed. In the productions defined so far, the embedding transformation involves a simple reconnection of bonds in the right-hand graph to vertices in the parent graph, so the embedding transformation will not be defined formally in this paper. A graph production is illustrated in Figure 7.

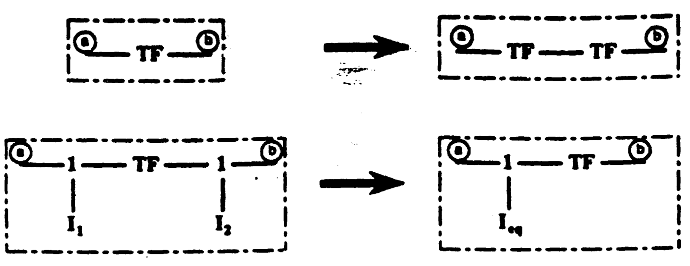

Figure 7: Graph productions

Graph Derivations

Now we say what it means to derive one graph from another. Given the definition of a production, we can define a *graph derivation*. Let

g_c = context graph

g_I = the initial graph

g_F = the final graph

then g_F is directly derivable from g_I through production $P = (g_L, g_R, E)$, iff

$$g_L \subseteq g_I$$

$$g_R \subseteq g_F$$

$$g_c = g_I - g_L = g_F - g_R$$

$\mathbf{B}_F - \mathbf{B}_R - \mathbf{B}_c$ = Set of connections specified by the embedding transformation

These conditions state that the left-hand graph of the production must be contained in the initial graph, that the right-hand graph of the production must be contained in the final graph, that the graph remaining after the left-hand graph has been removed from the initial graph is the same as the final graph before the right-hand graph has been inserted, and that the embedding transformation specifies the connections between right-hand graph and the context graph.

Finally, we can say that a graph, g_F, is derivable from g_I if there exists a set of graphs and productions such that:

$$g_I \xrightarrow{p_0} g_1 \xrightarrow{p_1} g_2 \xrightarrow{p_2} \cdots g_m \xrightarrow{p_m} g_F$$

For the mechanical design problem, we are looking for the sequence of productions that will transform the original specifications graph into the final graph of component specifications.

Graph Grammar

Using the elements already defined, defining a grammar for the class of graphs of interest here becomes straightforward. A grammar G on the set of graphs $\{g_i\}_{\Sigma_v \Sigma_e}$ is a quartet

$$G = (\Sigma_v, \Sigma_e, I, \mathbf{P})$$

where

Σ_v and Σ_e are the vertex and edge alphabets,

I is an element graph of $\{g_i\}_{\Sigma_v \Sigma_e}$ selected as the starting symbol of the grammar, and

$\mathbf{P}$ is a set of graph productions where $|\mathbf{P}|$ is finite.

This grammar defines a set of graphs or a *language* that can be derived from I through the set of productions $\mathbf{P}$. Formally, the *language*, $\mathrm{L}(G)$, is defined as:

$$\mathrm{L}(G) = \{g \mid g \in \{g_i\}_{\Sigma_v \Sigma_e} \wedge g \text{ is derivable from } I\}$$

Example of a Transformation from Specifications to Components

In this section, we illustrate how the graph productions are applied to a specifications graph resulting in a design alternative. In this paper, we do not discuss the strategy for selecting the sequence of transformations. The strategies are a topic of current research and are are discussed in [Hoover 89]. Briefly, the transformation strategy selects and configures components into a physical system based on two characteristics typical of good mechanical designs:

- Functional integration
- Utilization of incidental behaviors

Transformations that result in increased functional integration create a device configuration in which a single component contributes to more than one of the behavioral requirements of the design. Transformations that result in increased utilization of incidental behaviors create a device configuration in which secondary behaviors or physical characteristics of components are exploited.

Behavior-Preserving Transformations

The specification graph is not unique since there are many graphs which represent the same behavior. Figure 8 shows a specification graph for the design of a gear box that must drive two loads at a speed 400 times as great as the input speed. Other specification graphs are valid and, as many will immediately observe, the particular specification graph given is not likely to be the most convenient.

By transforming the specification graph, without altering behavior, we can explore design alternatives that have the same behavior. Some of the resulting configurations result in physically desirable designs and some do not. In addition to knowing the general rules for behavior-preserving transformations, we need to know which transformations to apply and the sequence of application. Guidance in selecting the behavior-preserving transformation to apply comes from the physical requirements of the system, from the physical characteristics of the components, and from the relationship between geometry and behavior of the component.

The transformation process described in [Hoover 89] is guided by the function integration and incidental behavior principles and by knowledge of the available components. It provides an approach to finding configurations that meet the physical requirements, thus eliminating blind search. This approach has been successful for the design of mechanical power transmissions and serves to illuminate several important issues for extending this technique to other domains.

The specification graph is composed of abstract behavioral primitives that do not yet correspond to any physical components. To arrive at a design we transform the specification graph, without changing function, so as to obtain a graph that more nearly corresponds to a collection of components. Figure 9 shows the resulting graph in which each of the rotational-rotational transformers has been replaced with a pair of rotational-translational transformers that might correspond to individual spur gears.

Component-Directed Transformations

Because mechanical designs are characterized by a high degree of integration, transformations should be directed toward this goal. Integration in a design requires the appropriate utilization of the behavioral and physical characteristics of its components. Therefore, intelligent application of transformations requires utilizing both the behavior and geometric graphs of the components to guide the selection process. Transformations selected in this way are *component-directed transformations* because their selection and

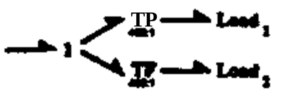

Figure 8: Specification graph for two-load design requirement

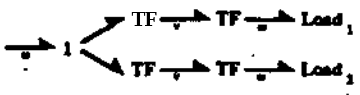

Figure 9: Transformed specification graph of Figure 8

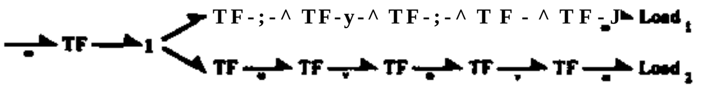

Figure 10: Component-directed splitting transformation of Figure 9

use are directed by knowledge of the available components. It is the class of component-directed transforms that ultimately enables the selection of a single component to fulfill multiple functions, e.g. selecting a worm gear to execute speed reduction, shaft offset and right angle functions as described more fully in [Hoover 89].

In this paper we have restricted the component domain to spur gears; nevertheless, the component directed transforms are important. The pair of transforms shown in Figure 9, if mapped directly to components would require a pair of gears with a 400:1 diameter ratio, which is clearly not practical. Directed by the allowable size range of available components, we apply a splitting transform to arrive at the graph shown in Figure 10.

The individual transformers in Figure 10 can then be mapped to spur gears (see Figure 4). The resulting design is shown in Figure 11. Note that as the mapping transforms are executed the dynamic behavior of the device changes due to the introduction of gear inertia. Because *the function* of the transmission is kinematic rather than dynamic, these mapping transforms are function preserving.

Other Transformations

Although the gearing system shown in Figure 11 is feasible, it is overly large, costly, and complex because there is no commonality of the power paths. Component-directed transforms do not have the scope to identify this type of deficiency. Another class of transforms are used to extend the function integration concepts to the device as a whole. Again, without discussing the means of selecting transforms we point out that the graph shown in Figures 8, 9, or 10 can be transformed in a function preserving manner to the one shown in Figure 12.

This specification can be mapped to an arrangement of gears as shown in Figure 13. This configuration has five fewer gears, one less shaft and is significantly more compact than the configuration shown in Figure 11.

The identification and selective application of this type of transformation is one of the most important and challenging aspect of our strategy for mechanical design.

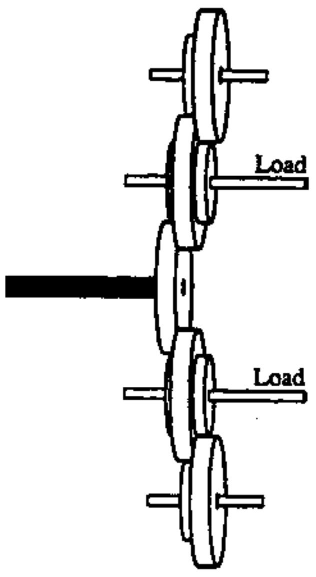

Figure 11: Physical system corresponding to Figures 8, 9, and 10

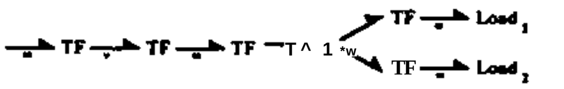

Figure 12: An alternative graph corresponding to Figure 10

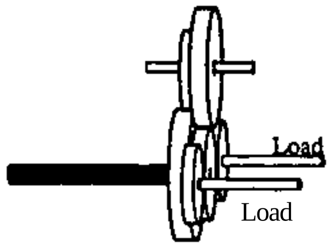

Figure 13: An alternative design corresponding to the specification given in Figure 8 and to the transformed graph shown in Figure 12

Results and Future Work

We have presented a behavioral and physical representation of the specifications and components for a limited class of mechanical designs. For transmission design, we have created the component database that contains graphs and form-behavior relations for bevel, spur, and worm gears, and we have implemented the transformation strategy discussed in [Hoover 89]. However, this paper presents preliminary work that must be expanded in many directions before we can prove that our transformational approach is valid for a larger class of mechanical designs.

Our research will continue in the following areas:

- **Expanding the behavioral md physical characteristics that can be represented***
- **Continuing to develop the representation of the interconnection between geometry and behavior.**
- **Increasing the number of components from** different mechanical **design domains represented** in our system, **and** representing **them at higher-levels of abstraction as** well **as at greater levels of detail.**
- Creating new transformation strategies, refining the existing ones, and in particular, creating general transformations that operate based on system-level characteristics rather than on component characteristics.

Acknowledgments

The authors are pleased to acknowledge the support of the Design Theory and Methodology Program of the National Science Foundation (NSF Grants DMC-84-51619 and DMC-88-14760) and the- Engineering Design Research Center at Carnegie Mellon University (NSF Grant CDR-85-22616).

References

[Brown 85]
Brown, D. and Chandrasekaran, B., "Expert Systems for a Class of Mechanical Design Activity," in *Knowledge Engineering in Computer-Aided Design*, Gero, J., ed., North Holland, 1985, pp. 259-290.

[Chomsky 57]
Chomsky, N., *Syntactic Structures.* Humanities Press, Atlantic Highlands, NJ, 1957.

[Crossley 80]
Crossley, E., "Make Science a Partner." *Machine Design*, April 24 1980.

[Ehrig 87]
Ehrig, H., Tutorial Introduction to the Algebraic Approach of Graph Grammars," *Graph-Grammars and their Applications to Computer Science*, Springer-Verlag, New York, Lecture Note Series 1987, pp. 3-14.

[Fenves 87]
Fenves, S. J. and Baker, N. C, "Spatial and Functional Representation Language for Structural Design," *Expert Systems in Computer-Aided Design*, Elsevier Science (North-Holland), IFIP 5.2 1987.

Finger 89]
Finger, S. and Dixon, J. R., "A Review of Research in Mechanical Engineering Design," *Research in Engineering Design*, Vol. 1, No. 1,1989.

[Fitzhorn89]
Fitzhom, P., "The Formal Language of Solid Representation," *Outfor review in Design Computing*, 1989.

[Gursoz89]
Gursoz, E. L. and ftinz, F. B., "Corner-Based Representation of Non-manifold Surface Boundaries in Geometric Modeling," Technical Report, Engineering Design Research Center, Carnegie Mellon University, 1989.

[Hoover 89]
Hoover, S. P. and Rinderle, J. R., "A Synthesis Strategy for Mechanical Devices," *Submitted to Research in Engineering Design*, 1989.

[Hrovat85]
Hrovat, D. and Tobler, W. E., "Bond Graph Modeling and Computer Simulation of Automotive Torque Converters," *Journal of The Franklin Institute*, 1985, pp. 93-114.

[Kamopp75]
Karnopp, D. and Rosenberg. R., *System Dynamics: A United Approach.* John Wiley & Sons, New York. 1975.

[Lai 87]
Lai, K. and Wilson, W. R. D., "FDL: A Language for Function Description and Rationalization in Mechanical Design," *Computers in Engineering*, ASME, New York. 1987. pp. 87-94.

[Maher85]
Mal»,M.UTO-RISEindBeyoixl: Direction f« Expert Systems in Design,* *Computer-Aided Design.* Vol. 17,1985, pp. 420-427.

[Marcus 86]
Marcus, S.. StouU. and McDermou, J., "VT: An Expert Elevator Designer that Uses Knowledge-Based Backtracking," Technical Report CMU-CS-86-169, Department of Computer Science, Carnegie Mellon University. 1986.

[Mtrgolis79]
Margolis, D. L. and Karnopp, D. C, "Bond Graphs for Flexible Mulibody Systems," *Transactions of the ASME*, Vol. 101.1979, pp. 50-57.

[Mostow85]
Mostow, J., "Toward Better Models Of The Design Process," *The Al Magazine*, Spring 1985.

[Nagl 87]
Nag], M., "Set Theoretic Approach to Graph-Grammars," *Graph-Grammars and their Applications to Computer Science*, Springer-Verlag, New York, Lecture Note Series 1987, pp. 41-54.

[Pahl 84]
Pahl, G. and Beitz, W., *Engineering Design*, The Design Council. Springer-Verlag, London, 1984.

[Paymer61]
Paynter, H. M., *Analysis and Design of Engineering Systems.* MIT Press, Cambridge, MA, 1961.

[Pinilla89]
Pinilla, J. M., Finger, S. and Prinz, F. B., "Shape Feature Descnpuon and Recognition Using an Augmented Topology Graph Grammar." *1989 NSF Engineering Design Research Conference*, University of Massachusetts, Amherst MA, June 11-14 1989.

[Remmerswaal 85]
Remmerswaal, J. A. M. and Pacejka, H. B., "A Bond Graph Computer Model to Simulate Vacuum Cleaner Dynamics for Design Purposes," *Journal of The Franklin Institute*, 1985, pp. 83-92.

[Reuleaux 54]
Kennedy, A. B. W., editor. *The Kinematics of Machinery.* Dover Publications, New York, 1854.

[Rinderle 82]
Rinderle, J. R., *Measures of Functional Coupling in Design*, PhD dissertation, Massachusetts Institute of Technology, June 1982.

[Rinderle 861
Rinderle, J. R., "Implications of Funcuon-Fbrm-Fabrication Relations on Design Decomposition Strategies," *Computers in Engineering. 1986*, American Society of Mechanical Engineers, Chicago, 1986, pp. 193-198.

[Steinberg 86]
Steinberg L., Langrana N., Mitchell, T., Mostow, J. and Tong, C. "A Domain Independent Model of Knowledge-Based Design," Technical report AIA"LS1 Project Working Paper No. 33, Rutgers University, March 1986.

[Stiny75]
Stiny, G., *Pictorial and Formal Aspects of Shape and Shape Grammars.* Birkhauser, Basel. 1975.

[Sun 80]
Suh, N. P., Wilson, D. R., Tice, W. W.. Yasuhara, M., and Bell, A. C, "Application of Axiomatic Design Techniques to Manufacturing.* *Production Engineering Division, Winter Annual Meeting*, American Socict> of Mechanical Engineers, Washington, *D.C.*, 1980.

[Suh 88]
Suh. N. P., *The Principles of Design*, Oxford University Press, Oxford. UK. 1988.

[Ulrich87]
Ulrich, K. and Seering, W. P., "Conceptual Design: Synthesis of Systems Components," *Intelligent and Integrated Manufacturing Analysis and Synthesis*, American Society of Mechanical Engineers, New York, 1987. pp

57-66.

[Ulrich8S]
Ulrich, K. T. and Seering, W. P.. Tunction Sharing in Mechanical Design/ *7th National Conference on Artificial Intelligence*, AAAI-88, Minneapolis, MN, August 21-26 1988.

[Ulrich 89]
Ulrich, K. T. and Seering, W. P., "Synthesis of Schematic Descriptions in Mechanical Design/ *Research in Engineering Design*, Vol. 1. No. 1,1989.

[Winh71]
Wirih, N.f "Program Development by Stcpwise Refinement," *Communications of the ACM*, Vol. 14, No. 4,1971. pp. 221-227.

Appendix A: Basic Graph Relationships and Operations

Let *g* = (V, B) and g^m • (V\ B*) The graph, *g*, is defined to be *isomorphic* to $g*_9$ that is,

*g=g** iff:

$$\exists \phi : V \rightarrow V^*$$

v€ $V \rightarrow \phi(v) \in V^*$

such that:

<> is bijective

V$(v_p v_2$,/,)$_€$B 3(<J)(v_I),0(v_2),/$_€$)€B-

^(^{v}l» ^{V}2» $0^€$ ^* 3 ^"'(Vj). 0"'(V_2), /,)€ B

That is, two graphs are isomorphic if for every venex in either graph there is an equivalent venex in the other, and for every edge connecting two venices in either graph there is an equivalent edge in the other.

Again assuming that the graphs are defined on the same alphabets, the graph, *g\ is* defined to be *included* in *g,* that is,

g'Qg iff

$$\exists g^{**} = (V^{**}, B^{**})$$

where:

r * *r*

V £ V

B^{mm} c B

V v,, v_2 € V, V /, € I, iff ($v_{|f} v_2$,/,) € B then
(v,, v_2, l_t) € B"

That is, calling *g** the subgraph and *g* the parent graph, the subgraph is said to be included in the parent graph if for every labeled edge between labeled vertices in the subgraph, or some graph isomorphic to it, there is an equivalent labeled edge between labeled venices in the parent graph.

Finally, given that g^9 is included in *g,* that is, *g'Qg** the *graph difference* is defined to be:

g-g^m = (V'\ B")

where:

V s V - V set difference

B- = B - (B- u {(v,, v_2, /,) | v, € V, v_2 € V J)

That is, the difference between the parent graph, *g,* and the subgraph, *g** is the parent graph with the vertices of the subgraph removed and with the edges of the subgraph removed along with the edges connecting the subgraph to the parent graph.