Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2002), *16*, 343–362. Printed in the USA.
Copyright © 2002 Cambridge University Press 0890-0604/02 $12.50
DOI: 10.1017/S0890060402165024

Function and behavior representation in conceptual mechanical design

Y.-M. DENG

CAD/CAM Lab, School of Mechanical and Production Engineering, Nanyang Technological University, Singapore 639798

(Received November 15, 2001; Accepted September 3, 2002)

Abstract

Conceptual design seeks to deliver design concepts that implement desired functions. Function and behavior are two dominant terms used in the research of this design phase. However, there are still some fundamental ambiguities and confusions over their representation, which have greatly hindered the interchange of research ideas and the development of design synthesis strategies. For conceptual design of mechanical products specifically, this paper attempts to clarify these ambiguities. It classifies function as purpose function and action function and relates them to the different levels of design hierarchy and abstraction. It distinguishes between semantic and syntactic representations of function and behavior and summarizes basic representation schemes. It also proposes an input–output action transformation scheme for semantic function representation and an input–output flow of action scheme for semantic behavior representation. Based on these discoveries, a refined framework is proposed for conceptual mechanical product design, where a function–decomposition–mapping process is elaborated to demonstrate the necessities and usefulness of the presented work.

Keywords: Action Function; Behavior; Conceptual Design; Purpose Function; Semantic Representation; Syntactic Representation

1. INTRODUCTION

Conceptual design is the most critical phase of a design process. In this phase, a design specification in the form of functional and other requirements is transformed into one or more design descriptions, each of which consists of a number of conceptual variants. Satisfying functional requirements is one of the primary design tasks. To achieve this, the behavior of the design often needs to be explored. Function and behavior are the two most fundamental and dominant concepts. Therefore, they need to be properly represented.

Representation provides semantics and syntax for specifying and manipulating an object of interest. Semantically, the representation is a description of the characteristics of the object, which reflects the user's understanding and perception. Syntactically, the representation is aimed at how the object can be represented in a computer program or other media so that it can be easily defined, retrieved, edited, and manipulated by a designer or a computer program in specific design situations. Hence, representation can be semantic, syntactic, or both.

Because of the importance of function and behavior, there have been countless definitions, descriptions, and discussions on them in the research community. However, there are still some fundamental ambiguities and confusions over their representation. It is a rather difficult task to clarify these ambiguities, given that design is so broad and deep. As such, this paper attempts to address the problem by restricting the designs to the conceptual stage of mechanical products. By a common understanding, the mechanical product should be an artifact (human made), it should be a physical device (with a physical structure and involving only physical behavior during its operational process), it should be of solid state (although liquid state or gas state objects can be used during its operational process), and it should be discrete.

Reprint requests to: Y.-M. Deng, Singapore–MIT Alliance (SMA) Office, N2-B2C-15, Nanyang Technological University, Singapore 639798. E-mail: mymdeng@ntu.edu.sg

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

To demonstrate the necessities and usefulness of the presented work, a framework of conceptual mechanical product design is also proposed.

2. FUNCTION

2.1. Purpose function and action function

Function is an overloaded term, with different researchers attributing different meanings to it (Chittaro & Kumar, 1998). Winsor and MacCallum (1994) state that function can be viewed as either *purpose* or *action*. Chakrabarti (1998) proposes two views of function, that is, function at the same level of abstraction as behavior (intended behavior) and function at a higher level (purpose). Chittaro and Kumar (1998) distinguish function as being *purposive* or *operational*. Chandrasekaran and Josephson (2000) propose an environment-centric viewpoint of function (function as *effect*) and a device-centric viewpoint of function (function as *role*). Hubka and Eder (2001) suggest eight types of functions according to the categories of purpose of the function, among which there is a *purpose* function and a *working* function. All these researchers view function as being either purposive or operational. However, few have recognized that there are not just two such *views* of function; rather, there could be two such function *types* in a same design. In addition, there is no existing design synthesis strategy or framework that explicitly supports these two function types for the same design problem.

The author advocates both two views and two types of function and adopts Chakrabarti's proposal (1998) that both functions may reside in the same design. For mechanical product design specifically, the function at the same level of abstraction as behavior, that is, operational function, can be more specifically referred to as action function. Here action is defined as *a physical interaction between two objects of interest, each of which may be a component of a design or the design itself and its environment*. Hence, function can be semantically classified into two types: purpose function and action function.

Purpose function is *a description of the designer's intention or the purpose of a design*. It is thus abstract and subjective. It is teleological knowledge and is human oriented. Action function is *an abstraction of intended and useful behavior that an artifact exhibits*. It is not human oriented, but it is related. The two functions relate to the different levels of a design hierarchy and abstraction. Generally, the overall function and some of its subfunctions at the upper level design hierarchy are expressed as a design intention, with the relevant concrete actions not known; hence, they are purpose functions. This overall function and the upper level subfunctions may need to be implemented by some lower level subfunctions via certain physical behaviors. These lower level subfunctions are thus action functions. For example, in designing a time-telling device, the overall required function may be expressed as "to tell time." This is a purpose function. It may be embodied by an action function at the lower design level, such as "to rotate the hour, minute, and second hands around the same pivot, each with a specific rotary velocity."

It is noted that the distinction between purpose function and action function is not clear-cut. This is because an action function may also contain design intention, because it relates to only the *intended* and *useful* behavior. On the other hand, a purpose function may also imply information relating to certain required action, even though this is not explicitly mentioned. For example, in the time-telling design case, "to tell time" also implies that the device should have a behavior that a human is able to interpret as telling time. On the other hand, "to rotate the hour, minute, and second hands" also annotates a design intention. This is because in achieving this behavior, the device may also exhibit other behaviors such as generating heat because of the friction. However, only the behavior of rotation is what the designer intends to use.

Nevertheless, a distinction between these two functions is still necessary and important. First, purpose function is a higher level design abstraction; thus, it facilitates a wider range of selection of design solutions. With carefully elaborated functional vocabulary, either domain specific or oriented to more domains of interest, purpose function may be used to enable wider and more flexible design synthesis, for example, the work of Hirtz et al. (2002). Action function, on the other hand, is specific to mechanical design and some other design domains where an artifact's behavior is related to certain actions. It is more directly related to physical principles by which more concrete design solutions may be sought.

Second, the perception of purpose function and action function is useful in clarification of function representation, as will be discussed in the following sections.

Third, and most importantly, employing both functions in a same design facilitates design synthesis. This will be demonstrated in the proposed conceptual design framework.

Note that the notion of purpose function and action function is slightly different from that of Chakrabarti's two views of function. Chakrabarti (1998) views an operational function as the one that is at the same level of abstraction as the expected or actual behavior (i.e., shares the same state variables and is required to use components only within the system boundary of the design to explain the behavior); whereas a purpose function is at the other levels of abstraction, requiring components of both design and its context for justification of its fulfillment. Hence, his purpose function is a function defined across system boundary, whereas the action function is defined within the system boundary. Both are operation oriented. Similarly, Kannapan and Marshek (1991) regard function as a mapping from a part of the behavior of the system to a part of the behavior of its supersystem. They characterize a mechanical system and its environment by the system interior, the system–environment interface, and the environment. With this char-

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

acterization, they associate behavior with the system–environment interface and function with the environment.

The author's action function is similar to that proposed by Chakrabarti (1998) and Kannapan and Marshek (1991), namely, it is an abstraction of system behavior and thus is operation oriented. It may be defined within or across the system boundary, depending on whether the environment is explicitly involved in the system behavior. However, the purpose function is slightly different. Although purpose function is at a higher level of design hierarchy and abstraction, it is not an abstraction of behavior, whether this is a higher level abstraction or not. Hence, it is not operation oriented. It is only a description of design intention, even though this description may imply information relevant to behavior. For example, for an acceleration sensing device, the purpose function may be something like "to sense acceleration," whereas one of the action functions may be expressed as "input: acceleration, output: voltage." These two functions are at different levels of the design hierarchy; that is, a partial design solution has already been identified or determined from the purpose function to the action function. The purpose function is not only at a higher level of design hierarchy (e.g., it is given as a design requirement or need), but it is also at a higher level of design abstraction (e.g., it does not mention what the concrete input and output are).

On the other hand, a few researchers explicitly support operational semantics of function only. For example, Rosenman and Gero (1994) distinguish function from design intention or purpose. Their function is equivalent to the author's action function, whereas their design intention is equivalent to the purpose function. Kirschman and Fadel (1998) classify four basic types of mechanical functions, which are oriented to action functions only. Interestingly, Maier and Fadel (2001) propose a more fundamental concept of "affordance" to subsume function, where their notion of function is, in effect, that of an action function. Affordance carries the meaning of the usefulness and purpose of an artifact; hence, it relates not only to the artifact but also to the user and the designer. In this sense, it unifies both purpose function and action function. In fact, as Maier and Fadel (2001) claimed, "affordance can be used as a unifying concept to stitch together an array of disparate design approaches." However, unlike the popular concept of function, a verified design synthesis framework is lacking based on this new concept.

2.2. Semantic representation of function

There has been a long-standing ambiguity in function representation. For example, Chakrabarti and Blessing (1996) summarize that, "Traditionally there have been three approaches in representing function," namely, verb–noun pairs, input–output flow transformations, and transformations between input–output situations. It is true that these types of representation have been widely used by the design researchers. However, there has been no awareness that the verb–noun pair is actually a syntactic representation and the other two are a semantic representation. Hence, such a classification causes ambiguity. Unfortunately, a similar classification has frequently appeared in the literature. In a recent article, Chakrabarti and Bligh (2001) state that function can be represented by a natural language like representation or by a mathematical representation expressed as a transformation between input and output. This statement is rigorous because both of these are syntactic representations.

Semantically, an action function can be represented by an input–output flow transformation (energy, material, and signal) and an initial–ending state transformation (initial physical state and ending physical state). For example, the function of a lever may be represented by an input force and an output force, which characterizes the effect of the lever when it is put into use. Initial–ending state transformation is used because in delivering a function, an artifact may undergo a series of physical state changes. The initial and ending state characterize the designer's intention in achieving a required function.

It is notable that the input–output flow (of energy, material, and signal) representation has its limitations. This is because, in the mechanical design domain, it is the physical interactions between the components (and also between the design and its environment) that contribute to the achievement of a function. In this paper, these interactions are referred to as *input actions* or *output actions*, depending on whether they act to or from an object of interest. The input–output flow of energy, material, and signal only characterizes some attributes (or properties) of an input–output action, not the action itself.

For example, if we want to more completely characterize the function of a lever used "to magnify force," then there should be an input–output action. The input action to the lever can be expressed as "to push/pull the lever at one end," whereas the output action from the lever may be expressed as "to push an object by the lever at the other end." The attributes of these actions may be represented by their associated forces (the input force and the magnified output force), the lengths between the respective ends of the lever and its pivot, the distances that the two ends move, the time that each of the two actions take, and possibly some others, depending on the requirement of the specific design problem. Clearly, this approach has stronger representational capability than an input–output flow of energy scheme, where an input force and an output force are commonly used. More importantly, the explicit representation of input action and output action facilitates synthetic reasoning, as will be explained in the section on behavior representations.

As such, the author proposes a new method to represent an action function: input–output flow of *action* transformation. The flow of action includes the input and output actions involved in the achievement of the action function. The conventional input–output flow of energy, material, and signal can be referred to as an input–output flow of *object* transformation. The distinctions between flow of

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

action and flow of object will be further elaborated in Section 3.1, where the lever example will be further discussed.

Furthermore, the input and output actions used in describing an action function are the intended actions. Unintended input action to an artifact may exist, and the artifact may also produce side effects that may affect the artifact's performance or its environment. Nevertheless, all these unintended actions are not supposed to be contained by the concept of function. This is because, after all, function is a description of the designer's intention (purpose function) or abstraction of intended and useful behavior (action function). The intended input action is one of the prerequisites for the output action to occur. The intended output action is where a function is eventually delivered. Hence, the author refers to them as driving input action (or *driving input*) and functional output action (or *functional output*), respectively.

Similarly, the input–output flow of object transformation represents only the intended input object and the intended output object, whereas the state transformation representation characterizes only those state changes that are of interest to the designers.

In some situations, the input flow of either object or action may be achieved earlier than when a function is being implemented, and thus not at the designer's interest. This kind of function may be represented without input flow. The devices providing this kind of function are intuitively called a "source," for example, a battery providing electricity, a flywheel providing mechanical energy, a water tank providing water, and so on. However, it is obvious that every function represented by an input–output flow transformation must at least have an output flow (object or action).

As to the semantic representation of the purpose function, unfortunately there is no existing method. This is due to the subjective and abstract nature of the purpose function itself, or we may say that a semantic representation of a purpose function is the function itself.

To conclude, only an action function has a semantic representation method, which includes three basic forms (schemes):

- input–output flow of object transformation: intended input and intended output (object = energy/material/signal);
- input–output flow of action transformation: driving input and functional output; and
- initial–ending state transformation: initial physical state and ending physical state.

2.3. Syntactic representation of function

A syntactic representation is the instantiation and embodiment of a semantic representation. Hence, it is more versatile. It could be data, variable, equation, expression, graph, table, and so on. It could be numerical or symbolic. It could be an array, list, frame (as in a frame-based scheme), logic, rule, object (as in an object-oriented scheme), and so on. This article only discusses some basic methods or schemes of syntactic function representation. In fact, many of the existing function representation methods proposed in the literature are merely a mutation of the basic forms discussed below, because there is no semantic difference between them.

As mentioned before, function can be syntactically represented by natural language sentences. This is one of most commonly used methods, and is used dominantly by human designers in their design documents, in design researchers' literature (e.g., this article), as well as in computer programs. Because of the expressiveness and generality of natural language, it can be used to represent both the purpose function and action function. Moreover, because of its subjective and abstract nature, a purpose function can only be represented by natural language sentences.

The syntactic representation of an action function must signify its semantics, namely, the input–output flow transformation or state transformation. For example, if we use a natural language sentence approach, an action function may be represented by a sentence describing its input flow or initial state and another sentence describing its output flow or ending state. Alternatively, and more often, we can use a single sentence with a carefully chosen verb, for example, the action function of a lever can be expressed as "to magnify force," where the verb magnify signifies the difference between input force and output force. For the initial–ending state transformation, the function can be expressed as "to transform from an initial state to an ending state," where the verb transform signifies the change of the two states.

Besides using natural language sentences, an action function can also be represented by other means. We can use input and output (state) variables. These variables can be fixed, or they can change temporally. Apart from directly specifying the input and output, the representation can also use a state table specifying in and out parameters of a transformation (Schmekel, 1989); or it can use a matrix, charts, sketches, and so forth for the same purpose (Kota, 1990; Hashim et al., 1994; Chiou & Kota, 1999). Under certain situations the function representation can also rigorously employ mathematical functions, such as formulas and equations. Adopting the term used by Chakrabarti and Bligh (2001), the author refers to all these forms of representation as *mathematical forms of input–output/state transformation*.

To conclude, there are two basic schemes of syntactic representation of function: a natural language sentence and a mathematical form of input–output/state transformation.

2.4. Comparison between two syntactic representations of function

The natural language sentence approach has the advantage of generality, flexibility, and expressiveness. However, it has several limitations.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

First, it lacks rigor in defining a function. Linguistic description is only a label to the true meaning of the thing to be described. For example, the function of a bearing can be described as "to support the shaft." Here the word "shaft" only serves as a label or index to the true meaning of a shaft: a shaft is a straight, round part; a shaft often works by rotating around its axis; a shaft is normally installed on the bearing; and so on. All these meanings are the knowledge relating to shaft, which are supposed to be known to the designer. Similarly, another word "support" also has its own meanings. Hence, simply putting the word support and the word shaft together to form a verb–noun pair style of expression is not rigorous enough to represent the function completely.

However, it should be noted that, from a philosophical viewpoint, this characteristic is also a benefit during the design synthesis process. In this process, several levels of a function structure, that is, from purpose functions to action functions, might need to be generated. The designer may not know all these levels of function precisely and completely. In these situations, natural language sentences provide a good tool in function representation.

Second, the approach lacks uniqueness. For example, the function of a motor can be described as "to provide mechanical power," "to change electrical power to mechanical power," "to provide rotary motion," and so on. In addition, there are many synonyms, such as furnish, generate, offer, produce, and supply for the word "provide" and alter, convert, transfer, transform, and turn for the word "change." In this regard, the concept of affordance proposed by Maier and Fadel (2001) may provide a good solution because it has a more complete notion than function.

The mathematical form of the input–output/state transformation approach, on the other hand, has the advantage of describing a function rigorously. As pointed out by Ullman (1993*a*), all methodical design authors use a system transformational definition of function. Examples of using the initial–ending state transition approach include Goel and Stroulia (1996) and Prabhakar and Goel (1998).

The rigor of this representation is due to the concrete nature of an action function. Hence, it is advisable to represent an action function by using some forms of input–output/state transformation; more precisely, some mathematical forms of input–output/state transformation (because a semantic input–output/state transformation may also be syntactically represented by natural language sentences).

As mentioned before, mechanical functions at different levels of a same design hierarchy may be viewed as either a purpose function or action function. A natural language sentence scheme can be used because it can represent both types of function. However, if we want to use a mathematical form of input–output/state transformation to represent the action functions, then different schemes have to be used during the design process, because purpose functions can only be represented by natural language sentences. For example, Yoshioka et al. (1999) proposed a prototype functional modeling system, where two representation schemes, natural language sentence and input–output flow transformation, are used at the different phases of the functional modeling process.

The author believes that a unified approach that incorporates the characteristics of the two basic schemes would be preferable because no model transformation is required between the different schemes. Deng et al. (1998) proposed an object-oriented representation scheme, where both the natural language sentence information (via attributes of the *function name*) and input–output transformation information (via *methods* encapsulated in the object) can be incorporated in the scheme.

3. BEHAVIOR

3.1. Semantic representation of behavior

A dominant view of behavior is that behavior is related to the physical state of a design: either it changes temporally, thus causing a state transition (Bobrow, 1984; Keuneke, 1991; Tomiyama et al., 1993; Ullman, 1993*b*; Umeda et al., 1996), or it can also remain unchanged, thus maintaining a static state (Yan, 1993; Deng, 1994; Rosenman & Gero, 1994; Qian & Gero, 1996). To simplify discussion, this article will use the term "state transition" to refer to both situations.

Umeda and Tomiyama (1997) gave a rather comprehensive summary of behavior representations, but they also failed to clarify whether a representation is semantic or syntactic. The types of representation given by them include state transitions, bond graphs (a sophisticated version of input–output formalism), physical phenomena, a mixture of state transition and physical phenomena, and the relative motion used by Chakrabarti and Bligh (1996). Here bond graph and relative motion can be unified into input–output flow of object (energy, material, and signal). As to physical phenomena, it is problematic because the phenomenon itself needs to be represented in the first place.

From a semantic perspective, the author summarizes existing behavior representation into the following two types: input–output flow of object (object = energy, material, signal) and state transition.

The input–output flow of object has the same formalism as that used for function representation, examples of which can be seen in the previous section. The state transition behavior representation is in accordance with the state transition view of behavior discussed above. It characterizes behavior as a chain or network of physical state change. Examples can be seen in Umeda and Tomiyama (1997).

The author believes that the physical state view is only one general view of behavior. For a specific design domain, there may be other views more helpful to conceptual design synthesis. For mechanical design, another useful view of behavior is the physical interactions between the compo-

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

nents of a design as well as between the design and its environment. An artifact exhibits certain behaviors not only by the change (or maintaining) of its physical state, but also by a number of interactions that take place inside the artifact, as well as with its environment, as governed by physical laws.

Like the input–output flow transformation function representation, the input–output flow of energy, material, and signal only characterizes the attributes of the physical interactions, not the physical interactions themselves. Hence, behavior can also be more precisely represented by an input–output flow of action.

It is different from the input–output flow of action transformation used in the function representation that an input action for behavior representation may be useful or harmful, whereas an output action may be an intended action or a side effect. Deng et al. (1999, 2000*a*) represent a behavior by a set of driving inputs (intended input actions) and a set of functional outputs (intended output actions), as well as possible harmful inputs (unintended and harmful input actions) and side effects (unintended and undesirable output actions). The following is a simple design example to show the difference between the conventional *flow of object* and the proposed *flow of action* behavior representation (Deng et al., 2000*a*):

Consider the behavior of a domestic washing machine. The input and output flow of object is the following:

- *input*: dirty laundry to be washed, clean water, detergent, and electricity;
- *output*: clean laundry, dirty water with detergent, noise, and so forth.

The input and output flow of action is as follows:

- *driving input*: the starting action from the user or the customized timer to initiate the washing cycle, the action of providing electricity from the power supply, and the support action from the floor where the washing machine is installed;
- *functional output*: the washing machine creates a certain pattern of motion of the laundry, as well as that of the water and the detergent used;
- *harmful input*: the washing machine is provided with too heavy a load, too much detergent, and so on;
- *side effect*: the washing machine generates noise and produces dirty water.

All these actions have their respective attributes, which can be represented as well. Together with this input–output flow of action, there should be a representation of the physical structure of the washing machine, which needs to be developed during the design process. The environment also needs to be represented, including the floor that the washing machine is on and the objects that are being processed by the washing machine (the laundry, water, and detergent). The relationship between the washing machine and the environment is yet another piece of information, involving factors such as how the washing machine is put on the floor (fixed on the floor or able to move freely on the floor, etc.) and whether the laundry, water, and detergent are inside or outside the washing machine. Of course, all this information is part of a design model, which is not for the behavior itself to represent.

This example shows the situation of a design at an overall artifact level. Another example is the behavior of a lever in producing the function of "magnifying force" (see Figure 1). For this example, the representation will be the same as its corresponding function representation, because there is no harmful input and no side effect:

- *structure:* C1 = lever with pivot O and end A and end B;
- *attributes of C1:* C1V1 = length of OA, C1V2 = length of OB;
- *driving input:* D1 = the lever is provided with a push/pull action at end A;
- *attribute of D1:* D1V1 = force of D1;
- *functional output:* F1 = the lever generates a push action at end B;
- *attribute of F1:* F1V1 = force of F1;
- *attribute expression:* F1V1 = D1V1*C1V1/C1V2.

Note that the physical structure is also listed. And for brevity, only a small portion of the attributes of the input–output action is listed.

To conclude, there are three basic schemes for the semantic representation of behavior:

- input–output flow of object: intended/unintended input, intended/unintended output (object = energy/material/signal);
- input–output flow of action: driving input, harmful input, functional output, side effect;
- state transition: a chain or a network of physical states.

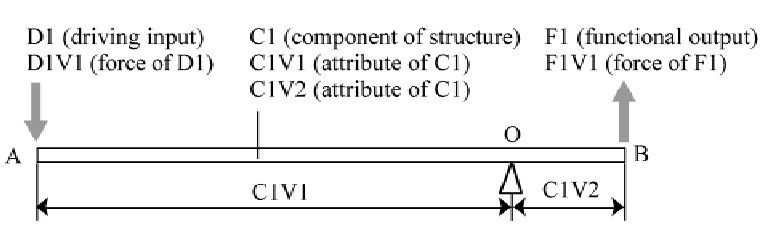


Fig. 1. The input–output flow of action demonstrated by a lever.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

3.2. Comparison between three semantic representations of behavior

The state transition type of behavior representation is very effective when there is distinct and finite change of physical states for a design problem. However, when there is no physical state change or the state change is trivial to the design problem solving, such as the behavior of a bearing in supporting a shaft, the approach is not effective in capturing the true meaning of behavior.

On the other hand, the input–output flow of object approach is very effective when there is a distinct flow of energy, material, or signal. An example of this is given by Kota and Lee (1990, 1993), who use a flow of material to represent the behavior of the hydraulic system. However, this approach cannot be used for systems where flows are difficult to identify. Eder and Gosling (1965) define such systems as *associative systems* (association of parts to form a whole unity). They pointed out that any attempt to define inputs and outputs for these systems is rather artificial. Umeda and Tomiyama (1997) also argue that the input–output definition of function (and thus behavior) has trouble representing a function that does not transform anything, such as the function of a fixture and the function of a linear guide.

The input–output flow of action approach is more comprehensive than the input–output flow of object representation, because it represents the physical interactions, not just their attributes that are restricted to energy, material, and signal only. Another strong point is that it can represent a behavior even when there is no flow of object involved, because it is possible to represent the behavior by an action that does not have any explicit attribute. Third, and more importantly, the input–output flow of action representation is a higher level design abstraction than that of the flow of energy, material, and signal. Thus, it facilitates design synthesis, as will be demonstrated in the proposed conceptual design framework.

Because of their respective merits and demerits, the three basic representation schemes are not supposed to substitute each other. They are simply different approaches and should be applied in their respectively suitable design problems.

3.3. Syntactic representation of behavior

Just as with the syntactic representation of action function, behavior can also be represented by using natural language sentences or the mathematical form of input–output flow or state transition. Note that, although the natural language sentence is widely used in representing (at least in *expressing*) behavior, such as in designers' verbal communications, design documents, and research articles, surprisingly, this has never been recognized in the literature. The reason behind this phenomenon, as the author believes, is also the ambiguity between semantic and syntactic representation.

The above example of the household washing machine gives an example of natural language sentence representation, whereas the example of lever shows a mathematical form of input–output flow of action representation. The merits and limitations of each were described in Section 2.4.

Figure 2 illustrates the relationship between different types of function, behavior, and the relevant function and behavior representation schemes. Note that, because the *input–output flow of object/action transformation* function representation characterizes only the input and output sides of the flow, we can leave out the word *flow* and simply refer to this representation scheme as *input–output object/action transformation*. This distinguishes it from the statement of *input–output flow of object/action* behavior representation.

4. CONCEPTUAL MECHANICAL DESIGN FRAMEWORK

In this section, the author proposes a framework for the conceptual design of mechanical products to demonstrate the necessities and usefulness of the above discoveries. This is a modification to a two-stage functional modeling framework proposed in Deng et al. (1999) and Deng (2000). Functional modeling of a conceptual mechanical product can be regarded as a process of establishing a functional model of a desired, but physically undefined, product. From this perspective it is also a design synthesis process. In the following, the author first briefly introduces this functional modeling framework and then presents the modification that makes it more consistent and comprehensive for conceptual mechanical product design.

4.1. Existing two-stage functional modeling framework

One prominent characteristic of conceptual mechanical product design is that the design problem is, in general, weakly coupled or uncoupled at the upper levels of a design hierarchy and strongly coupled at the lower levels (Deng et al., 2000*b*). A feasible design strategy should thus enable the designers to decouple a design problem at the upper levels of the design hierarchy and to solve strongly coupled design problems at the lower levels. In the existing framework, conceptual mechanical design is stratified into two functional modeling stages, with the first stage for the upper level design hierarchy and the second stage for the strongly coupled design problems at the lower design levels.

In the first stage, a functional decomposition and mapping approach is generally applicable. The existing framework achieves decomposition by identifying functionally independent design subtasks. For example, in designing a domestic washing machine, the overall function is "to wash laundry." This function might be initially decomposed, by focusing on the set of functionally independent subtasks, into subfunctions such as "to add water," "to wash," "to drain," "to rinse," and "to spin." The subtasks associated with these subfunctions might need to be performed in a cycle (that is, with repetition of certain subtasks). To assist

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

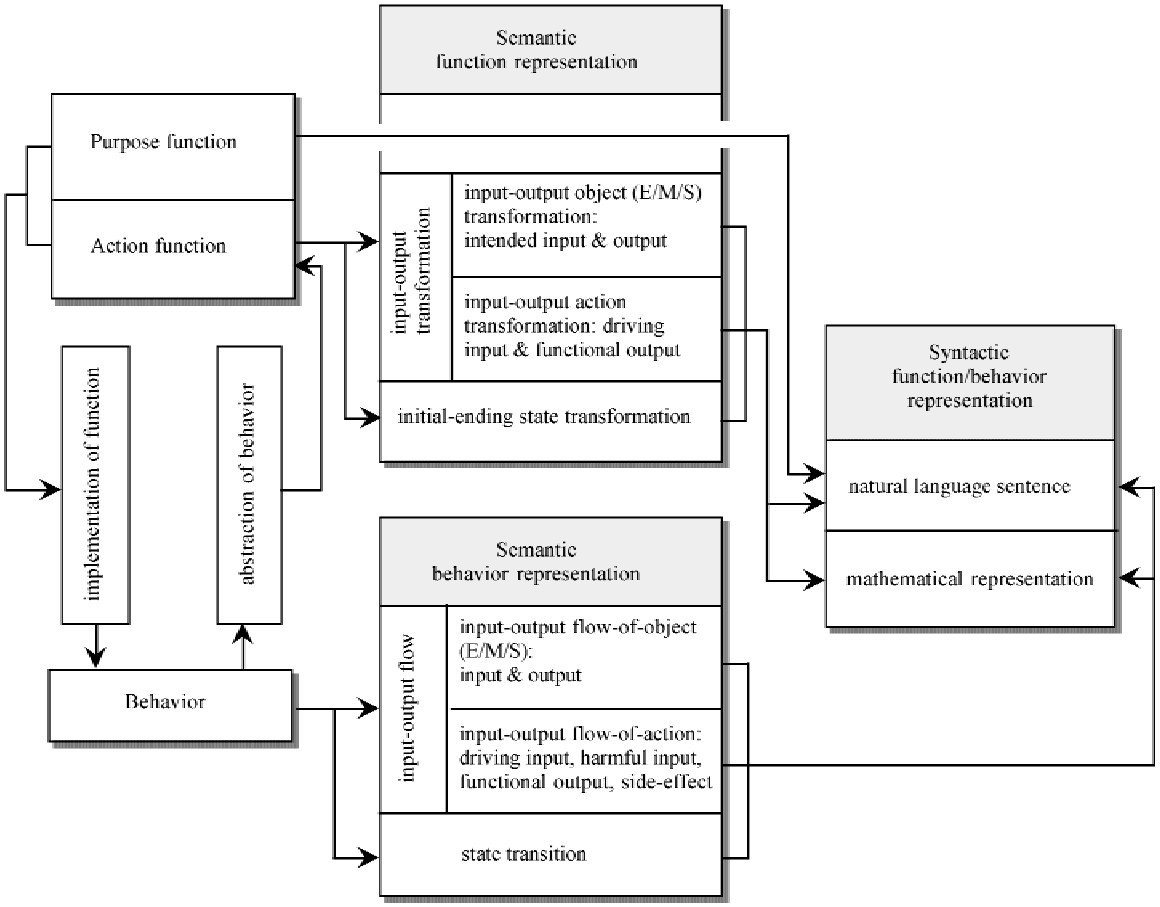


Fig. 2. The relationship between function–behavior and their representations.

the decomposition, domain-specific function decomposition knowledge may be applied (Deng et al., 2000*b*). This knowledge is formalized knowledge of working principles or past design scenarios that store the alternatives of necessary subtasks, independent of each other, for the domain-specific design tasks.

According to axiomatic design theory (Suh, 1990, 1995), the design process is a mapping between the functional domain (functional requirements) and physical domain (design parameters satisfying functional requirements). During the functional decomposition process, function–structure mapping should also be carried out in order to determine the implementing physical structures.

The second-stage functional modeling further develops those subfunctions that are not mapped to any physical structure by generating the desired product's causal behavioral process (CBP). A CBP refers to a network of subbehaviors required for achieving one or more functions identified from the first stage. The subbehaviors are causally connected, because a preceding subbehavior is causally required by the succeeding subbehavior: The functional output action from the preceding subbehavior is used as the driving input action to the succeeding subbehavior. The rationale behind this concept is that a function is not just achieved by a certain physical structure, but rather that this physical structure must be under its working environment and must undergo a behavioral process.

The subbehaviors used in the CBP are represented by an input–output flow of action approach. With this representation, a CBP can be generated by using a backward reasoning procedure. Briefly, the procedure starts from generating the subbehaviors (called *output subbehaviors*) that can produce the required functional outputs of the design, which correspond to the unmapped functions from the first-stage modeling process. After that, a recursive process is applied to generate the necessary preceding subbehaviors, whose functional outputs can be used as the driving inputs to their succeeding subbehaviors. The procedure continues until all the initial driving inputs to the generated subbehaviors can be provided by the working environment specified in the design specification.

In generating each subbehavior, the designer can search for relevant physical phenomena stored in a physical phenomena library, which acts as a supporting design knowledge base for the synthesis of the causal behavioral process. A physical phenomenon is defined as a distinctive and

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

generic characterization of a fundamental physical behavioral process guided by a certain physical/action principle, which involves a behavior actor (i.e., physical structure, which may consist of a number of components), certain working conditions, and a physical effect or effects arising from the behavioral process. The physical effect or effects correspond to the functional output action of a behavior.

After the above two functional modeling stages, all the physical structures corresponding to the subbehaviors of the generated CBPs, as well as the physical structures identified by the function–structure mapping, can be assembled with respect to the assembly relations identified during the modeling process. The results are then verified to ensure that they achieve all design requirements (Deng et al., 2000*c*).

For example (Deng et al., 1999, 2000*c*), in designing a paper separation device, the overall function is "to separate paper," denoted by the function F111. By focusing on the functionally independent subtasks, this function is decomposed into three subfunctions:

- F211 = transport the paper pack into a specified place,
- F212 = position the paper pack precisely, and
- F213 = separate the paper pack into individual pieces.

These three subfunctions cannot be mapped onto physical structures and hence must be further developed in the second modeling stage. As such, one or more CBPs should be developed. The CBP is represented and visualized by a CBP graph consisting of design nodes of both subbehaviors (denoted as B nodes) and environmental elements (denoted as E nodes), including their relevant function, behavior, and structure information. These nodes are causally connected to form a digraph. For example, Figure 3 shows a CBP graph developed for function F213.

The procedure for developing this CBP starts from the output subbehavior B1. Assume that, after searching the physical phenomena library, the designer decides to use a phenomenon called "separation by friction," which separates a paper pack into individual pieces by using a rotating drum and the friction between the rotating drum and the paper pack. By retrieving information from this phenomenon, B1 is specified; it is then represented by an input–output flow of action scheme:

- driving inputs: B1D1 = rotary motion to the rotating drum; B1D2 = pressure and support to the paper pack;
- functional output: B1F1 = move the top piece of paper off the paper pack;
- side effect (SE): B1F2 = move the pieces of paper, except for the top piece, off the paper pack (SE).

Note that the friction between the top piece of paper and the second piece of paper might cause the second piece of paper to be moved off also. This in turn might cause the remaining pieces of paper to be moved off accordingly. Hence, it is necessary to represent the side effect (B1F2).

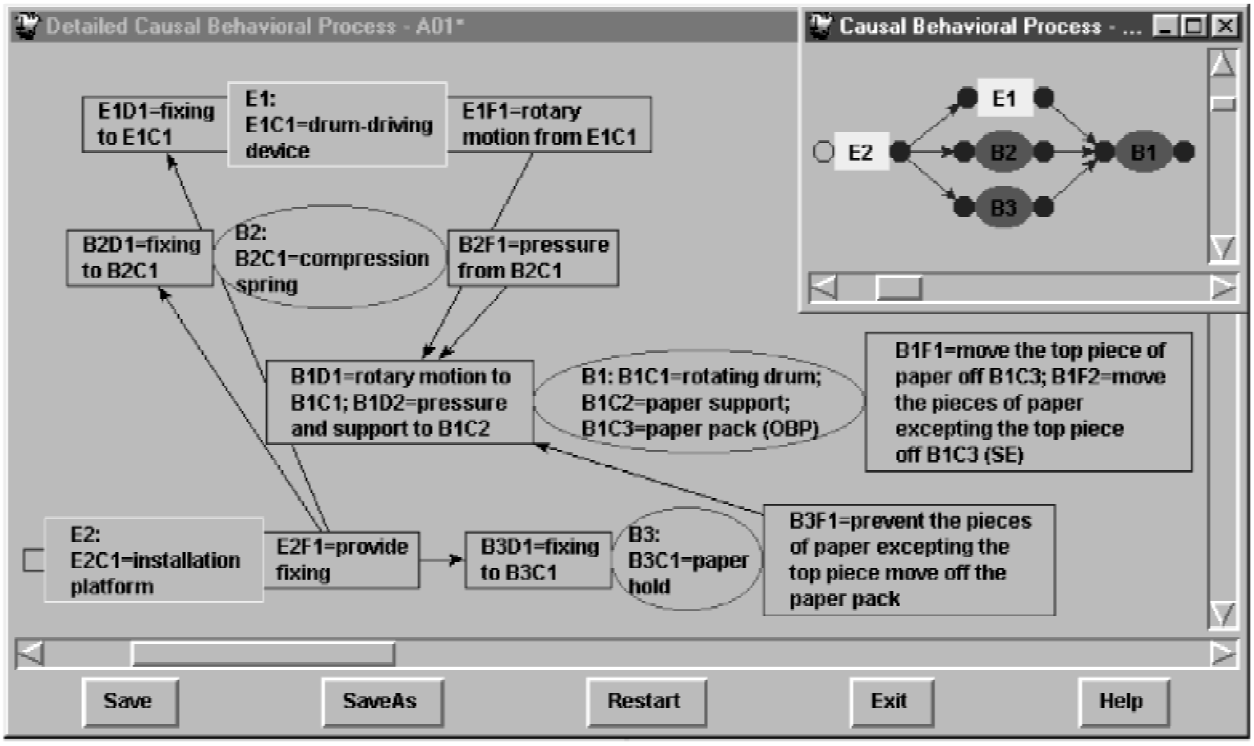


Fig. 3. A CBP graph developed for function F213.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

The physical components identified from this subbehavior are:

- B1C1 = rotating drum;
- B1C2 = paper support; and
- B1C3 = paper pack (OBP), where OBP stands for object being processed, meaning it will not be used when constructing the physical structure of the design later.

After that, more subbehaviors are generated whose functional outputs are used to provide the required driving inputs of B1 (B1D1 and B1D2). Assume that the design specification has stated a working environment: E1C1 = drum-driving device, which has a functional output: E1F1 = rotary motion from E1C1. Hence, E1 is connected to B1, and E1F1 is used as B1D1. Further on, E1 also needs a driving input (E1D1 = fixing to E1C1); thus, another environmental element, E2C1 = installation platform, is identified and connected to E1.

Regarding another driving input of B1, namely, B1D2 = pressure and support to the paper pack, the designer needs to find another relevant physical phenomenon so that another subbehavior can be generated. Assume that a compression spring is identified. For brevity, its behavior is not elaborated, but can be seen from Figure 3, where it is denoted as B2. The required driving input of B2 can be provided by environmental element E2 (installation platform), where the spring is fixed.

For the side effect from B1, which is B1F2 = move the pieces of paper except the top piece off the paper pack (SE), yet another subbehavior is needed. As such, subbehavior B3 is generated, where a component B3C1 = paper hold is identified. The driving input of B3 can also be provided by E2; that is, this component is also fixed on the installation platform.

With the CBP being developed, the identified physical components can be assembled. Note that during the CBP generation process, much more information is identified and formalized than is listed above, for example, the attributes of physical components and those of the input–output actions, the constraints on these attributes, the assembly and spatial relationships between the physical components of the design and between the design and the environment, and so on. Refer to Deng et al. (1999, 2000*a*) for a detailed description of the CBP generation process. Figure 4 shows a schematic of the physical structure identified from this CBP example, which achieves function F213 developed from the first modeling stage.

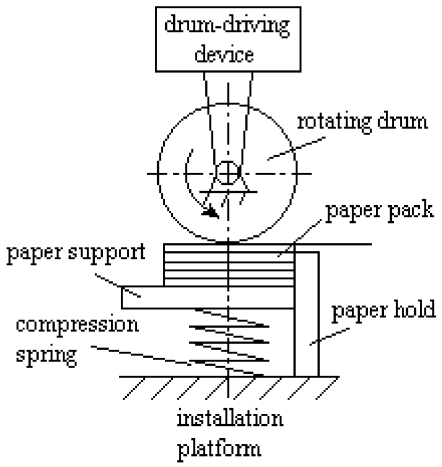


Fig. 4. A schematic of a paper separation device achieving function F213.

4.2. Refined framework

The stratification of conceptual mechanical design into two stages is useful in making design efforts more focused, in organizing the design process, and in formalizing design knowledge (Deng et al., 2000*b*). However, it also leaves behind a gap between the two stages. This includes a lack of clarification of what functions should and can be further developed in the second-stage functional modeling and a lack of methodology (or at least a guideline) of how these subfunctions can initiate the second-stage modeling process.

Recall that Section 2 classified function into purpose function and action function. It also states that for mechanical design, functions at the upper level design hierarchy are generally purpose functions. It is now clear that, in general, it is purpose functions that are involved in the first-stage functional modeling process, including functional decomposition and function–structure mapping (note that this does not completely exclude the involvement of action functions). Those subfunctions that cannot be mapped to physical structure and are purpose functions should be mapped to action functions before further development can be initiated in the second-stage functional modeling. That is to say, there should be a mapping from purpose function to action function. This is because in the second modeling stage, each subbehavior (represented by driving input, functional output, harmful input, and side effect) is actually an implementation of its corresponding action function (represented by the intended driving input and functional output). For example, in order to further develop the subfunction of "to add water" for the washing machine example, this purpose function should be mapped onto some kinds of action functions, an example of which is "to remove the blockage of the inlet to let water in." Obviously, this action function characterizes a movement action of the blockage of the inlet. To achieve this action, some driving input actions are required, which invariably initiates the second-stage functional modeling process.

To facilitate mapping from purpose functions to action functions, domain-specific mapping knowledge may be stored *a priori*. This is similar to the widely practiced *catalogue lookup* (Chandrasekaran, 1990) or *catalogue-driven* (Gui & Mäntylä, 1994) approaches in enabling function–structure mapping, where a vocabulary of domain-specific fundamental functions and a catalogue of their correspond-

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

ing physical structures are predefined and organized. An example is the aforementioned acceleration sensing device, where the purpose function "to sense acceleration" may be directly mapped to the action function "input: acceleration, output: voltage" or "input: acceleration, output: current," given that the knowledge of the commonly used sensing devices is generally known to the designer.

Besides a direct mapping, behavior can be used to facilitate the process. In fact, behavior is commonly utilized in assisting mapping from function to structure, which is often referred to as *function–behavior–structure* mapping. By identifying the behavioral specification of a design and exploring the behavior of existing physical components, designers can gain useful ideas about the means by which the overall design task can be achieved.

To assist functional mapping via behavior or physical domain knowledge, systematic design theory suggests using physical or working principles (Pahl & Beitz, 1988), which is adopted by many researchers, for example, Kuttig (1993) and Bracewell and Sharpe (1996). Similarly, design science also suggests identifying an action principle and/or a mode of action (Hubka & Eder, 1988, 2001).

The author embodies physical or action principles as physical phenomena and uses physical phenomena to assist mapping from function to structure, as well as from purpose function to action function. The same method is also applied in generating causal behavioral processes. For example, in the aforementioned paper separation design case, both the overall function F111 and its three subfunctions are purpose functions. For function F213 (separate the paper pack into individual pieces), depending on the different physical phenomena identified, the corresponding action function can be the following:

- F331 = slide off the top piece of paper,
- F332 = blow off the top piece of paper,
- F333 = shovel off the top piece of paper,
- F334 = adhere and shift off the top piece of paper,

and so on. These subfunctions are OR connected with F213. Each is associated with a specific action of the corresponding device, which enables the initiation of second-stage functional modeling process. For example, F331 actually corresponds to the CBP elaborated in the previous discussion.

To conclude, there are four types of functional mapping operations:

- mapping from function to physical structure,
- mapping from function to physical phenomenon to physical structure,
- mapping from purpose function to action function, and
- mapping from purpose function to physical phenomenon to action function.

Note that in the first mapping, function can be both purpose function and action function, hence it can be detailed into two mappings. In the second mapping, the function refers to purpose function only. By combining the first mapping with the last two mappings, two more mappings become possible:

- mapping from purpose function to action function to physical structure and
- mapping from purpose function to physical phenomenon to action function to physical structure.

In addition, because purpose function and action function are at different levels of design abstraction and function and physical means (structure and phenomena) are of different domains of knowledge, the actual mappings are not exhaustive; that is, various mappings may be necessary from a purpose function to a specific action function. Nevertheless, the mappings listed above have explicitly stated the different mapping types and also demonstrated the direction of the mappings, that is, from purpose function to action function to physical structure.

To implement these various functional mapping operations in the first-stage functional modeling, the author proposes to use a *function–decomposition–mapping* (FDM) graph to represent and organize the modeling process (Deng, 2000). Hence the first modeling stage is also called a *FDM process*. An FDM graph is a tree of both functions and nonfunctions, as is illustrated by an example shown in Figure 5.

The FDM graph includes *function* nodes (F), which can be either purpose or action function, and three nonfunction nodes, namely, *physical structure* nodes (S), *physical phenomenon* nodes (P), and *functional route* nodes (R). Each node is associated with an *AND* or *OR* operator (hence, the node is called an *AND node* or *OR node*, respectively), which signifies the operation between its subnodes. As the OR operation is already included in the graph, the functional route node (R) is only used to simplify the graph when an operand of an OR operation involves more than one node, such as nodes R1 and R2 shown in Figure 4 (refer to the Appendix for a detailed explanation).

Note that the functional route node is not a physical structure. For example, in designing a time-telling device, possible functional routes may include "using a mechanical approach" (thus, a mechanical clock), "using an electronic approach" (thus, an electronic clock), "using a biological approach" (thus, something the author does not know either), and so on. The case study discussed in the following section gives a practical example to illustrate this further.

An explanation of the functional modeling process shown in Figure 5 is given below:

- Overall required function F1 (an OR node) is developed via functional route 1 (denoted as node R1) or functional route 2 (denoted as node R2).
- Via route R1 (an AND node), F1 is decomposed into F2 and F3. F2 (an OR node) can be mapped to either structure S1 or structure S2. F3 (an AND node) is decomposed into F6 and F7. F6 is a purpose function, which is mapped to an action function, F12, via phenomenon P1. F7 itself is an action function.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

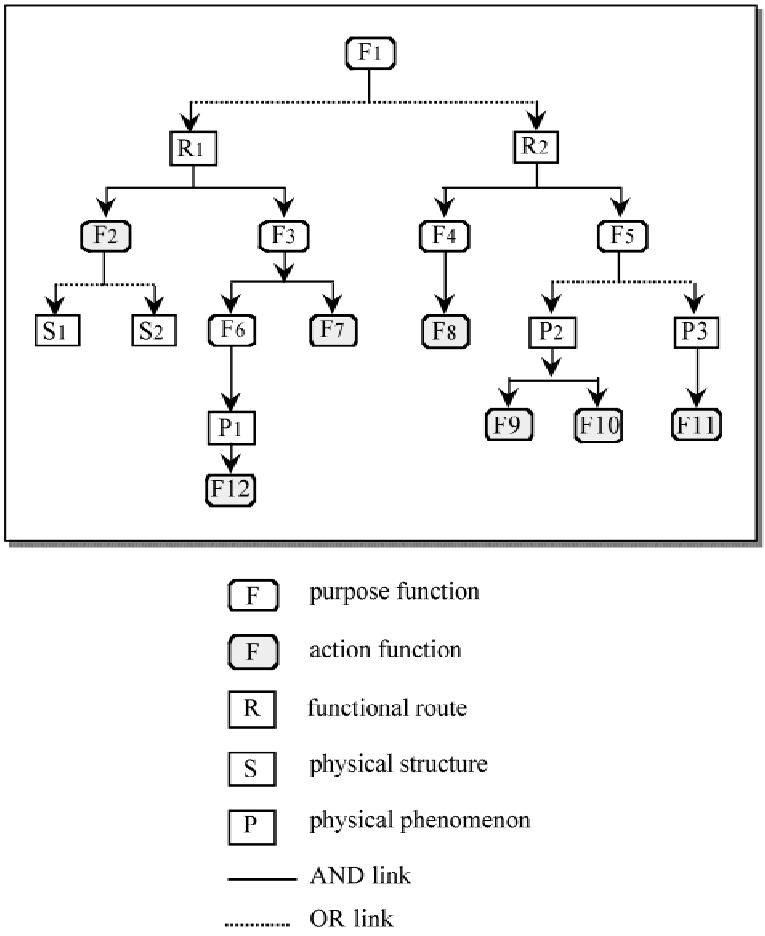


Fig. 5. An illustration of an FDM graph.

- Via route R2 (an AND node), F1 is decomposed into F4 and F5, both of which are purpose functions. F4 is directly mapped to an action function F8. F5 (an OR node) is mapped to either phenomenon P2, which is associated with two action functions, F9 and F10, or phenomenon P3, which is associated with action function F11.

The rules for constructing the FDM graph are:

- Rule 1: if a node is an R node, then its subnodes may be of any types (F, S, P, or R nodes).
- Rule 2: if a node is an S node, then its subnodes can only be S nodes. That is, the FDM graph allows physical structure decomposition to be incorporated.
- Rule 3: if a node is a P node, then its subnodes can only be F nodes or S nodes. The F nodes correspond to action functions. The S nodes indicate that the designer is mapping a function to its implementing physical structure through a certain physical phenomenon.
- Rule 4: If a node is an F node and none of its supernodes is a P node, then its subnodes can be of any type (F, S, P or R nodes).
- Rule 5: If a node is an F node and one of its supernodes is a P node, then its subnodes can only be F nodes or S nodes. This is because the current node is an action function and can only be decomposed or mapped to a physical structure.

The first-stage functional modeling should be carried out until all the lowest level nodes are either physical structures (in the example, S1 and S2) or action functions (in the example, F7–F12).

Because the presented work on function and behavior only applies to the first-stage functional modeling, there is no modification to the second-stage modeling framework.

4.3. Case study and software implementation

A design case is studied to illustrate the FDM process (first-stage functional modeling) and its software implementation. This is a terminal insertion device, which is part of an automatic assembly system for manufacturing electronic connectors. A terminal insertion device is used to insert

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

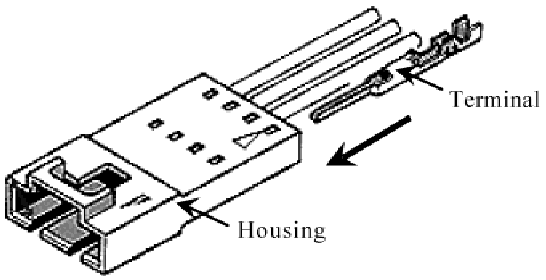


Fig. 6. The insertion of a terminal into a housing.

terminals into a housing to make the conductor and insulator one unit. Figure 6 illustrates the insertion of a terminal into a housing. (Refer to http://www.molex.com/training/bce/index.html for more details.)

Figure 7 shows the software implementation of the proposed FDM process. This is part of the refined version of a computerized design environment reported in Deng et al. (1999). The software uses an oval to represent the function nodes (F nodes) and a rectangle to represent nonfunction nodes (R nodes for functional routes, S nodes for physical structures, and P nodes for physical phenomena). The AND links are visualized by the black lines and the OR links are represented by white lines. The implemented FDM graph is used to represent, organize, and visualize the first-stage functional modeling process.

Note that the three-digit serial numbers used in the design nodes are automatically generated by the software, with the first number representing the number of design level, the second number representing the number of the group (each group relates to one node at the immediate next higher level) the node is in, and the last number representing the serial number of the node in its corresponding group.

The software is used in an interactive manner. A designer can start a conceptual design process by specifying functional requirements over the topmost design node, F111, and develop the FDM graph in a stepwise manner. Domain-specific functional decomposition knowledge and functional mapping knowledge are stored *a priori* by using functional knowledge libraries (Deng et al., 2000*b*), with a functional vocabulary library is also used to index these libraries. The functional vocabulary library is now upgraded based on the distinction between purpose function and action function, which means that the domain-specific purpose functions and action functions are stored under different classification schemes. The functional knowledge libraries are extended with new types of functional mapping knowledge to support mapping from purpose function to action function. Figure 8 shows the user interface for specifying design nodes of the FDM graph. Apart from directly specifying design nodes, the designer can also search the design knowledge libraries to retrieve the relevant data to assist the design process.

The FDM graph shown in Figure 7 corresponds to a partially developed FDM process of the design case. The overall required function is F111 = insert terminal into housing to make them one unit.

This is a purpose function, which does not state the concrete input–output actions needed for its implementation. From past design experience, which is well documented in

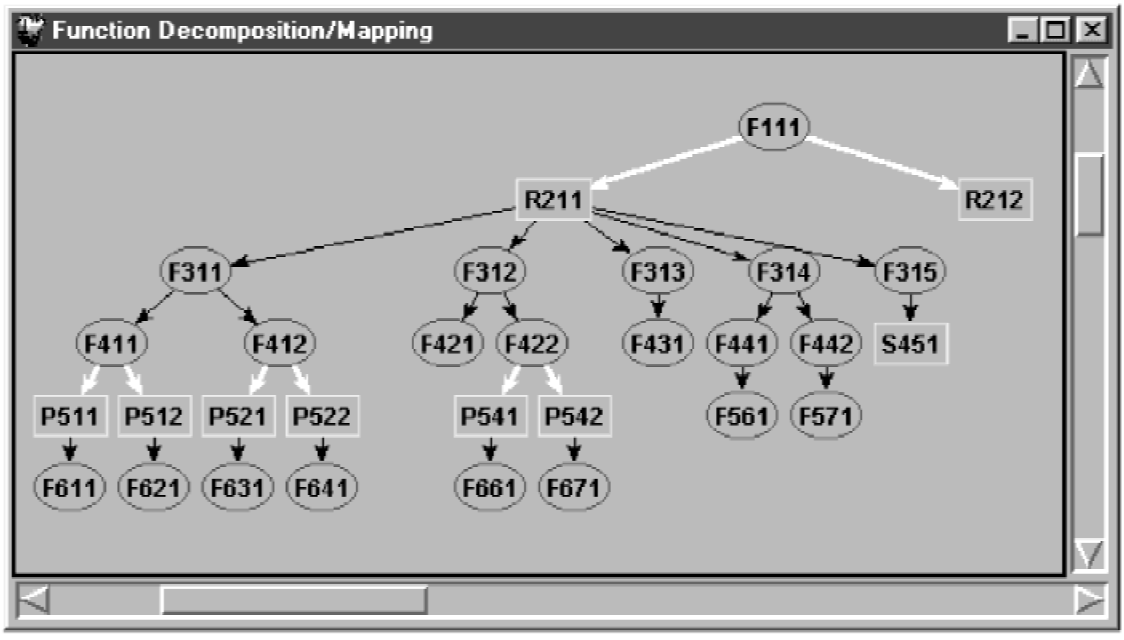


Fig. 7. The software implementation of the FDM process.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

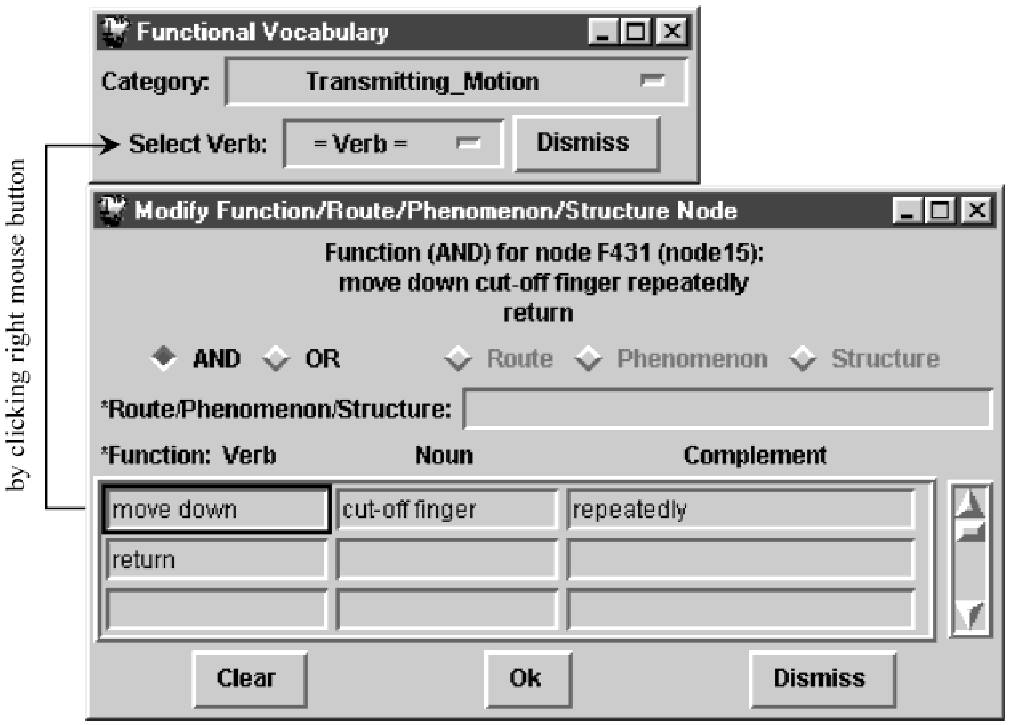


Fig. 8. The specification of design nodes.

connector engineering, there are two design alternatives for further developing this function. That is, there are two functional routes (Deng et al., 2000*b*):

- Functional route 1: R211 = gang insertion approach. Insert an entire row (or gang) of terminals with each insertion cycle.
- Functional route 2: R212 = terminal stitching approach. The terminals or pins are inserted one circuit at a time, resembling the stitching work of a sewing machine.

For brevity, only functional route 1 is discussed, where only a small fraction of possible design alternatives are discussed. F111 is decomposed through this functional route by considering the functionally independent design subtasks. The set of subfunctions (all are purpose functions) is as follows:

- F311 = locate and clamp housing,
- F312 = track and feed terminal in gang,
- F313 = cutoff terminal in gang,
- F314 = hold and insert terminal into housing in gang, and
- F315 = support the system.

The subfunction F311 (locate and clamp housing) needs both locating and clamping operations; hence, it can be further decomposed into two subfunctions, F411 = locate housing and F412 = clamp housing. Similarly, F312 (track and feed terminal in gang) can be decomposed into F421 = track terminal in gang and F422 = feed terminal in gang. F314 (hold and insert terminal into housing in gang) can be decomposed into F441 = hold terminal in gang and F442 = insert terminal in gang.

Together with this function decomposition process, various mappings are also applied. For example, function F315 (support the system) is directly mapped to a physical structure, S451 = supporting structure, which needs to be further developed during the embodiment design phase. Some purpose functions are directly mapped to action functions: F313 to F431, F441 to F561, and F442 to F571. For example, F313 (cutoff terminal in gang) is mapped to an action function, F431 = move down and return cutoff finger repeatedly. F431 is an action function because an output action is explicitly specified, which can be further developed in the second-stage functional modeling process.

Other purpose functions are mapped to physical phenomena first, then further to action functions, including F411–P511–F611, F411–P512–F621, F412–P521–F631, and F412–P522–F641. For example, by searching the physical phenomena library, F411 (locate housing) can be mapped to either of the following two phenomena:

- P511 = housing insert-locating phenomenon (locating the housing by inserting the locator into the housing) or
- P512 = housing house-locating phenomenon (locating the housing by putting the housing in the locator).

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

Figure 9 shows the housing insert-locating phenomenon stored in the physical phenomena library. As briefly mentioned before, a phenomenon incorporates information of the involved physical structure (including a number of components), its corresponding schematic geometry, an indexing function corresponding to the physical effect of the phenomenon, input–output flow of actions (driving input, functional output, harmful input, and side effect); their attributes; attribute expressions; constraints on the attributes to make the phenomenon valid; environmental elements; assembly relations between the components of the physical structure and between the components and the environmental elements; and so on.

By retrieving the phenomenon information and adapting it to the current design, the mapped action functions are identified. For example, action function F611 = input: translational motion and support, and output: insert locator into the housing. Similarly, other mappings are also developed.

Note that the rules for constructing the FDM graph discussed above are incorporated in the implemented software. Hence, the software will automatically monitor the design process when new design nodes are added. For example, if the designer tries to specify another phenomenon node connecting to F611, the software will warn the designer and stop him or her from doing so because this violates rule 5: if a node is an F node, and one of its supernodes is a P node, then its subnodes can only be F nodes or S nodes.

For those functions that cannot be mapped to a physical structure (all are action functions), they are further developed in the second-stage functional modeling process. By traversing all the design nodes, the software is able to automatically report all alternatives of these action functions (including the identified physical structures) from the FDM graph. For example, there are nine design alternatives from the above FDM graph (eight of them through functional route 1), as shown in Figure 10.

The designer can then choose one or more design alternatives to conduct the second-stage functional modeling. The physical structures identified from this process should be assembled with S451 (from the FDM process) to derive the physical structure of the whole design. Because there is

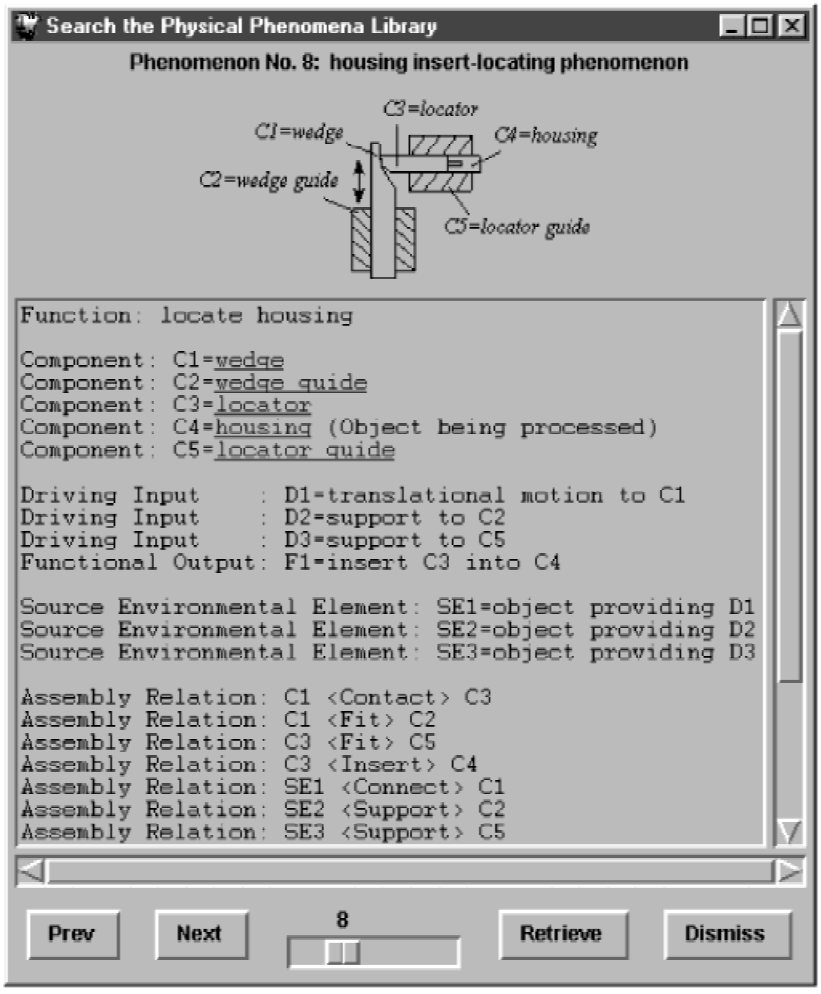


Fig. 9. The physical phenomena library to assist purpose–action function mapping.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

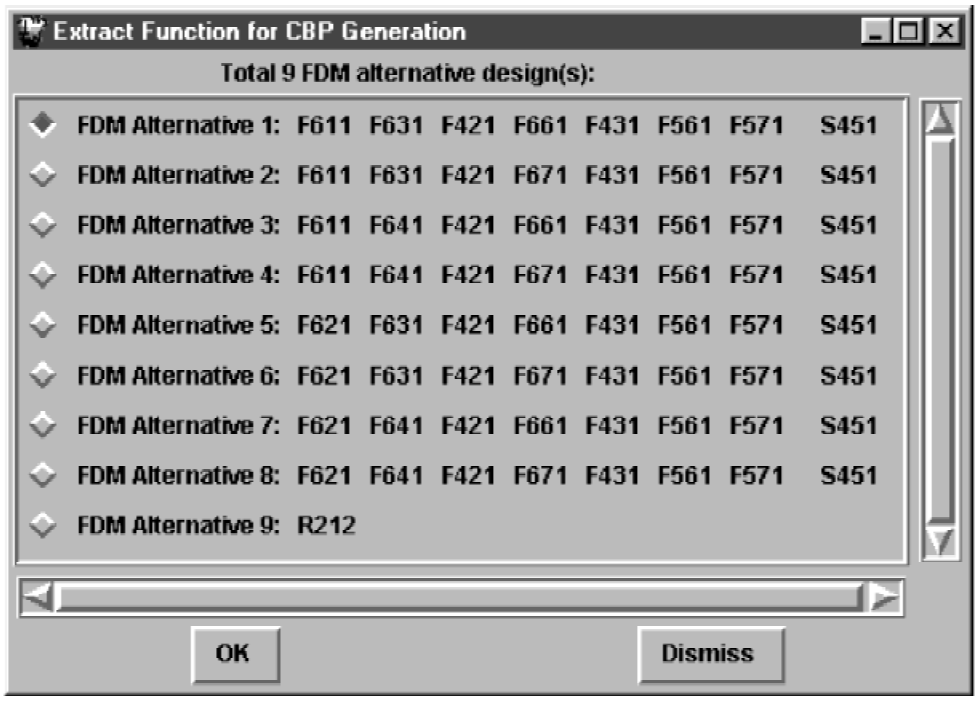


Fig. 10. The automatically listed design alternatives from the FDM graph.

no modification to this process and it was briefly discussed in the previous section, it will not be elaborated again. Refer to Deng et al. (2000*b*) for more details.

4.4. Related work

Function decomposition and mapping have been given considerable attention by design researchers, where AND and OR operators are commonly used in manipulating design alternatives derived from the decomposition (Kota & Lee, 1990, 1993; Gui & Mäntylä, 1994). For example, Kota and Lee (1990, 1993) use two building blocks for the mapping: *functional building block* (FBB) and *structural building block* (SBB). Mapping is carried out in two stages: from FBBs to SBBs and from SBBs to physical structures. Apart from function, the model incorporates various levels of abstraction of physical structure. However, it does not incorporate physical means in enabling the mapping.

By contrast, the well-known *function–means tree* model advocates the coevolution of function and its implementing means (Andreasen, 1980). Function and means are represented integrally in the function–means tree. Bracewell and Sharpe (1996) have developed a conceptual design environment called *SchemeBuilder*, which is based on the function–means tree. Rules for functional decomposition have been formalized from decomposition and working principles that are represented as means in the model.

Sturges et al. (1996) present an extended functional logic approach to represent functional relationships both within the same level and between the different levels of a design hierarchy. Their functional logic is represented by a general form of *functional block* diagram using a directed graph. A functional block is semantically equivalent to "why" at the next higher level and "how" at the next lower level. This function logic forms a chain of reasoning from required functions to the structural components that implement them. Connections between functions are defined by various links, such as the and/or link, temporal link, causal link, information link, and revisional link.

Contrary to the above top-down reasoning approach, Gero (1990) presents a bottom-up synthesis approach by using two mappings: from actual behavior directly to a structure and from expected behavior to a function. Expected behavior is derived from a formulation or specification of the design problem. Actual behavior is derived from analysis of a database of existing structures. A synthetic technique is proposed to generate the actual behavior from the expected behavior.

The function decomposition and mapping approach is perhaps the most ubiquitous design model in conceptual design (Maher, 1990). The above overview only listed some of the research that is most relevant to the proposed FDM process (part of a conceptual mechanical design framework). First, it shows that the proposed FDM graph representation is quite similar to the function–means tree, that is, physical means is incorporated in the graph. The major difference is that the FDM graph distinguishes between purpose function and action function, hence more mapping operations are supported.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

Second, physical phenomenon is proposed as a formalism of the mapping knowledge to assist both mapping from function to structure and mapping from purpose function to action function.

Third, the function decomposition and mapping operations are only carried out in the first modeling stage. This is because decomposition and mapping have limited reasoning capability when strongly coupled design problems are involved. For mechanical design specifically, this often occurs at the lower levels of the design hierarchy, where action functions are generally involved. Once one or more action functions are identified and are not mapped to any physical structure, the methodology of the causal behavioral process (CBP) can be employed. This is because action functions are more directly related to the physical principles by which more concrete solutions can be sought. As shown in the previous section, CBP is a network of design nodes consisting of function, behavior, and structure of both the design object and the environment. For those strongly coupled action functions at the lower design levels, CBP has stronger representational and reasoning capabilities than an AND–OR tree approach. This has been demonstrated in Deng (2000) and Deng et al. (2000*a*, 2000*b*, 2000*c*).

5. CONCLUSIONS

The purpose of this article is to clarify some ambiguities and confusions relating to the representation of function and behavior used in conceptual mechanical product design. It is suggested that there are not only two views of function, but also two types of function. Mechanical function can be classified into purpose and action functions, which relate to the different levels of the same design hierarchy and abstraction. It proposes a distinction between the semantic and syntactic representations of function and behavior. The commonly used natural language sentence scheme is shown to be only a syntactic representation, whereas the semantic form of input–output flow or state transition representation scheme (both for action function and behavior) can also be instantiated by natural language sentences. The article summarizes basic representation schemes. It also proposes an input–output action transformation scheme to semantically represent the action function. This provides a complement to the input–output flow of action scheme of behavior representation. Together the two schemes provide a consistent and comprehensive means for semantic representation of mechanical function and behavior.

The work is useful not only in understanding the fundamental concepts of function and behavior but also in developing conceptual design synthesis strategies. To demonstrate this point, a refined framework is proposed for conceptual mechanical product design by using both function types for the same design problem. A previous work involving a two-stage functional modeling framework is modified to bridge the gap between the two modeling stages. Specifically, it distinguishes functions involved in the function decomposition and mapping process as being purpose functions or action functions and suggests that only the action functions can be further developed by generating causal behavioral processes. Hence, there should be a mapping from a purpose function to an action function. As such, two new mapping operations are proposed: direct mapping from a purpose function to an action function and mapping from a purpose function to physical phenomena first and then to an action function.

With this refinement, the first-stage functional modeling process is represented and visualized by a FDM graph. Several construction rules are proposed for the development of the FDM graph, so as to organize and guide the modeling process. Software implementation of this modeling process is also briefly discussed, where a functional vocabulary library based on purpose and action functions is employed. The design knowledge libraries, such as the physical phenomena library, can also be enhanced with knowledge to support the new mapping operations.

The proposed framework is useful for the early stage of conceptual mechanical product design from both a research perspective and a practice perspective. For practice, the framework guides the designers in their conceptual design process, by which they can explore their design alternatives through decomposing design functions, conducting various functional mapping operations, and generating causal behavioral processes for those strongly coupled action functions. The designers can compile their past design experience by using the proposed formalized functional knowledge libraries to assist their future designs. The framework is also helpful for instructional purpose for novices.

From a research perspective, the main thrust of this article lies in the clarification of function and behavior representation, the suggestion of two function types for the same design rather than two function views for different designs, the perception of behavior as involving interactions rather than as state transitions, and the proposed input–output flow of action representation scheme. The proposed design framework contributes both to the understanding of conceptual mechanical product design from a functional and behavioral perspective and to the formal representation of this process. More importantly, the work is not restricted to the mechanical product design domain alone. By adjusting the notion of an action function to the operational function specific to other design domains and modifying the functional operations involving the action function, a new design support framework may be developed for other design domains as well.

Design is a very broad, deep, and complex intellectual activity (Duffey & Dixon, 1990). This article only touched on a small portion of design research issues. Nevertheless, it is the author's belief that clarification of any ambiguity in fundamental concepts such as function and behavior will be useful in broad areas of design research, especially for conceptual design.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

ACKNOWLEDGMENTS

The author would like to thank the Editor, Prof. David C. Brown, for his encouragement and the anonymous reviewers for their valuable comments and suggestions. This article is based on the author's previous work during his PhD study, which was guided by his two advisors, Prof. Tor Shu Beng and Prof. Graeme A. Britton. The author would like to express his sincere gratitude for their guidance and immense help.

REFERENCES

Andreasen, M.M. (1980). *Machine design methods based on a systematic approach*. PhD Thesis. Lund, Sweden: Lund Technical University.

Bobrow, D.G. (1984). Qualitative reasoning about physical systems: An introduction. In *Qualitative Reasoning about Physical Systems* (Bobrow, D.G., Ed.), pp. 1–5. Amsterdam: North-Holland.

Bracewell, R.H., & Sharpe, J.E.E. (1996). Functional descriptions used in computer support for qualitative scheme generation—"Schemebuilder." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 333–345.

Chakrabarti, A. (1998). Supporting two views of function in mechanical designs. *AAAI Workshop on Reasoning About Function, AAAI Workshop on Functional Modeling and Teleological Reasoning*. Madison, WI.

Chakrabarti, A., & Blessing, L. (1996). Special Issue: Representing functionality in design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 251–253.

Chakrabarti, A., & Bligh, T.P. (1996). An approach to functional synthesis of solutions in mechanical conceptual design. Part III: Spatial configuration. *Research in Engineering Design 8*, 116–124.

Chakrabarti, A., & Bligh, T.P. (2001). A scheme for functional reasoning in conceptual design. *Design Studies 22*, 493–517.

Chandrasekaran, B. (1990). Design problem solving: A task analysis. *AI Magazine Winter*, 59–71.

Chandrasekaran, B., & Josephson, J.R. (2000). Function in device representation. *Engineering with Computers 16*, 162–177.

Chiou, S.-J., & Kota, S. (1999). Automated conceptual design of mechanisms. *Mechanism and Machine Theory* 34, 467–495.

Chittaro, L., & Kumar, A.N. (1998). Reasoning about function and its applications to engineering. *Artificial Intelligence in Engineering 12*, 331–336.

Deng, Y.-M. (2000). *Functional design of mechanical products: Design model and modeling framework*. PhD Thesis. Singapore: Nanyang Technological University.

Deng, Y.-M., Britton, G.A., & Tor, S.B. (1998). A design perspective of mechanical function and its object-oriented representation scheme. *Engineering with Computers 14(4)*, 309–320.

Deng, Y.-M., Tor, S.B., & Britton, G.A. (1999). A computerized design environment for functional modeling of mechanical products. *Proc. 5th ACM Symposium on Solid Modeling and Applications*, pp. 1–12, Ann Arbor, MI.

Deng, Y.-M., Tor, S.B., & Britton, G.A. (2000*a*). Abstracting and exploring functional design information for conceptual mechanical product design. *Engineering with Computers 16(1)*, 36–52.

Deng, Y.-M., Tor, S.B., & Britton, G.A. (2000*b*). A dual-stage functional modeling framework with multi-level design knowledge for conceptual mechanical design. *Journal of Engineering Design 11(4)*, 347–375.

Deng, Y.-M., Britton, G.A., & Tor, S.B. (2000*c*). Constraint-based functional design verification for conceptual design. *Computer Aided Design 32(14)*, 889–899.

Deng, Y.-S. (1994). *Feature-Based Design: Synthesizing Structure from Behavior*. PhD Thesis. Pittsburgh, PA: University of Pittsburgh.

Duffey, M.R., & Dixon, J.R. (1990). A program of research in mechanical design: Computer-based models and representations. *Mechanical Machinery Theory 25(3)*, 383–395.

Eder, W.E., & Gosling, W. (1965). *Mechanical System Design*. Oxford, UK: Pergamon

Gero, J.S. (1990). Design prototypes: A knowledge representation schema for design. *AI Magazine Winter*, 26–36.

Goel, A.K., & Stroulia, E. (1996). Functional device models and model-based diagnosis in adaptive design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 355–370.

Gui, J.-K., & Mäntylä, M. (1994). Functional understanding of assembly modeling. *Computer-Aided Design 26(6)*, 435–451.

Hashim, F.M., Juster, N.P., & de Pennington, A. (1994). A functional approach to redesign. *Engineering with Computers 10*, 125–139.

Hirtz, J., Stone, R.B., McAdams, D.A., Szykman, S., & Wood, K.L. (2002). A functional basis for engineering design: Reconciling and evolving previous efforts. *Research in Engineering Design 13*, 65–82.

Hubka, V., & Eder, W.E. (1988). *Theory of Technical Systems*. London: Springer.

Hubka, V., & Eder, W.E. (2001). Functions revisited. *Proc. ICED'01*, Glasgow, Scotland.

Kannapan, S.M., & Marshek, K.M. (1991). Design synthetic reasoning. *Mechanism and Machine Theory 26(7)*, 711–739.

Keuneke, A.M. (1991). Device representation, the significance of functional knowledge. *IEEE Expert 6(2)*, 22–25.

Kirschman, C.F., & Fadel, G.M. (1998). Classifying functions for mechanical design. *Journal of Mechanical Design 120*, 475–482.

Kota, S. (1990). A qualitative matrix representation scheme for the conceptual design for mechanisms. *Mechanism Synthesis and Analysis 25*, 217–230.

Kota, S., & Lee, C.L. (1990). A functional framework for hydraulic systems using abstraction/decomposition hierarchies. *Computers in Engineering 1*, 327–340.

Kota, S., & Lee, C.L. (1993). General framework for configuration design: Part 1—methodology. *Journal of Engineering Design 4(4)*, 277–294.

Kusiak, A., & Szczerbicki, E. (1992). A formal approach to specifications in conceptual design. *Transactions of the ASME Journal of Mechanical Design 114*, 659–666.

Kuttig, D. (1993). Potential and limits of functional modeling in the CAD Process. *Research in Engineering Design 5*, 40–48.

Maher, M.L. (1990). Process models for design synthesis. *AI Magazine Winter*, 49–58.

Maier, J.R.A., & Fadel, G.M. (2001). Affordance: The fundamental concept in engineering design. *Proc. of DETC'01, DETC2001/DTM-21700*, Pittsburgh, PA.

Pahl, G., & Beitz, W. (1988). *Engineering Design: A Systematic Approach*. London: Springer-Verlag.

Prabhakar, S., & Goel, A.K. (1998). Functional modeling for enabling adaptive design of devices for new environments. *Artificial Intelligence in Engineering 12*, 417–444.

Qian, L., & Gero, J.S. (1996). Function–behavior–structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 289–312.

Rosenman, M.A., & Gero, J.S. (1994). The what, the how, and the why in design. *Applied Artificial Intelligence 8 (2)*, 199-218.

Schmekel, H. (1989). Functional models and design solutions. *Annals of the CIRP 38(1)*, 129–132.

Sturges, R.H., O'Shaughnessy, K., & Kilani, M.I. (1996). Computational model for conceptual design based on extended function logic. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 255–274.

Suh, N.P. (1990). *The Principles of Design*. New York: Oxford University Press.

Suh, N.P. (1995). Axiomatic design of mechanical systems. *Transactions of the ASME, Journal of Mechanical Design 117*, 2–10.

Tomiyama, T., Umeda, Y., & Yoshikawa, H. (1993). A CAD for functional design. *Annals of the CIRP 42(1)*, 143–146.

Ullman, D.G. (1993*a*). A new view on function modeling. *Proc. ICED'93*, pp. 21–28, The Hague.

Ullman, D.G. (1993*b*). The evolution of function and behavior during mechanical design. *Design Theory and Methodology 53*, 91–103.

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., & Tomiyama, T. (1996). Supporting conceptual design based on the function–behavior–state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(5)*, 275–288.

Umeda, Y., & Tomiyama, T. (1997). Functional reasoning in design. *IEEE Expert 12(2)*, 42–48.

Winsor, J., & MacCallum, K. (1994). A Review of functionality modeling in design. *The Knowledge Engineering Review 9(2)*, 163–199.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

Yan, M. (1993). Representing design knowledge as a network of function, behavior and structure. *Design Studies 14(3)*, 314–329.
Yoshioka, M., Lossack, R.-S., & Umeda, Y. (1999). Functional modeling on the basis of functional flow and verb–noun models. *Proc. ICED'99*, Vol. 2, pp. 1195–1198, Munich.

APPENDIX

A1. Explanation of functional route node used in FDM graph

Theoretically there is no need to employ a route node in an AND–OR tree because an OR operator already has the functionality to express different routes. The only reason for using a functional route node (R node) in the FDM graph is that it simplifies the graph when an operand of an OR operation involves more than one node. For example, in Figure 5, the design intention is to decompose F1 into (F2 AND F3) OR (F4 AND F5). If nodes R1 and R2 are not used, then all four subfunction nodes need to be connected to F1 directly. Figures A.1 and A.2 show two common methods that do not use R nodes (Kota & Lee, 1990; Kusiak & Szczerbicki, 1992).

Both of these methods have limitations. The method by Kota and Lee (1990) leads to redundancy. For example, if we want to decompose F1 into (F2 OR F3) AND F4, then F4 needs to be represented *twice* because it needs to be *ANDed* with both F2 and F3. This is because a direct representation is not possible using this method; hence, some logical calculation has to be done first.

It looks like the method by Kusiak and Szczerbicki (1992) can eliminate this redundancy (although the logical calculation is still necessary), because the fourth example shown in Figure A.2 (A AND B OR B AND C) clearly depicts that B can be *ANDed* with both A and C.

However, if we extend this expression as

A AND B OR B AND C OR A AND C,

and represent it in Figure A.3, it can be seen that the method cannot distinguish between A AND C (which is the desired

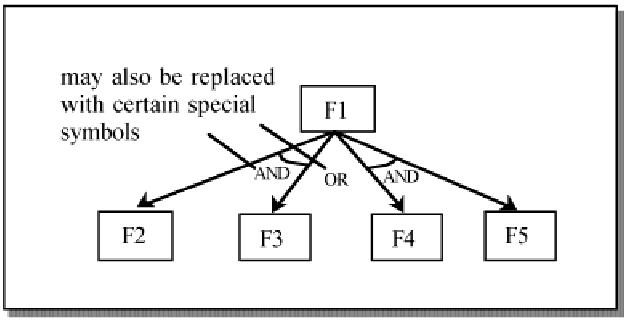


Fig. A.1. The AND–OR representation by Kota and Lee (1990).

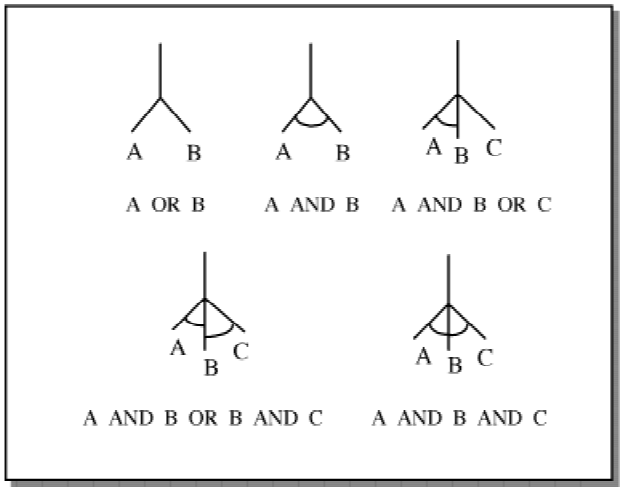


Fig. A.2. The AND–OR representation by Kusiak and Szczerbicki (1992).

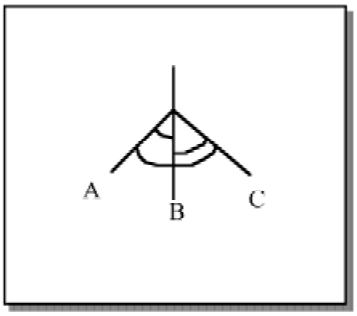


Fig. A.3. An example using the method of Kusiak and Szczerbicki (1992).

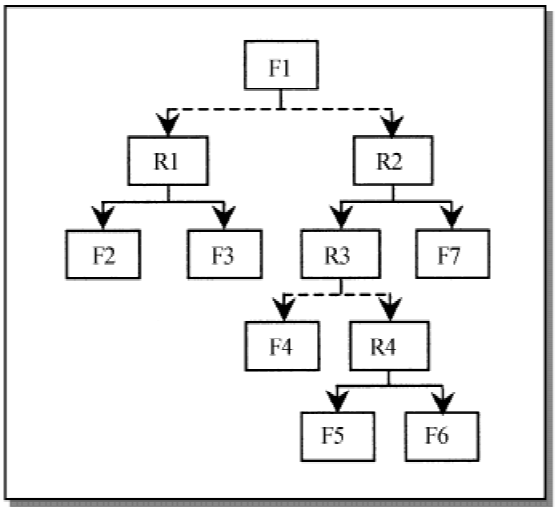


Fig. A.4. An example using route nodes in the AND–OR graph. (—) An AND operation; (- - -) an OR operation. For a real design case, R3 and R4 should be replaced with function nodes (F nodes), which represent certain subfunctions.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press

alternative) and A AND B AND C (just like the last example shown in Figure A.2).

Things can turn intangible if more subnodes are encountered where the logical relations between them are more complicated, such as the example of decomposing F1 into the following:

(F2 AND F3) OR ((F4 OR (F5 AND F6)) AND F7).

On the other hand, if we employ R nodes in the graph, such as the one shown in Figure A.4, then no logical calculation is necessary. This is very important because it helps to retain the design intention in the graph. Besides, the R nodes only bear the information of OR operators, and no additional information is required.

Yimin Deng (Y.-M. Deng) is currently a research fellow at the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. He received his BEng and MEng from the University of Science and Technology of China in 1987 and 1990, respectively, and his PhD from Nanyang Technological University in 2000. His research interests include design theory and methodology, geometric modeling and CAD/CAE/CAM applications, and injection molding design and optimization.

https://doi.org/10.1017/S0890060402165024 Published online by Cambridge University Press