Representing Function as Effect

B. Chandrasekaran and John R. Josephson

Laboratory for AI Research
The Ohio State University
375, Dreese Laboratories
2015 Neil Avenue
Columbus, OH 43210 USA
{Chandra, jj}@cis.ohio-state.edu

Abstract

In this paper we start with a brief introduction to the Functional Representation framework that a number of colleagues and we have been developing for over a decade or so. Then we move on to the need for a precise language for describing objects, properties and causal relations. We argue that such a language is needed to express the meaning of terms such as "function" and "behavior." We briefly describe a formal framework for and definition of device function that we have recently developed. We have sought the smallest ontological framework that is sufficient for developing an idea of function. Functions are defined in terms of the effects of objects on their environments. The definition of function is sufficiently general to express static and dynamic functions, intended and natural functions, and functions of abstract and physical objects.

FUNCTIONAL MODELING

The major concern of this international workshop series is "Functional Modeling." The idea of function is central to engineering: designers' job is to design artifacts that are intended to have certain functions. When systems fail, reasoning about how to fix them involves reasoning about malfunctions. When designers are designing, they look for components that can achieve certain functions. Predicting how systems would behave under various conditions of use or abuse also often requires knowing what the functions of the devices are. Risk analysis calls for calculating the impact on functions of various kinds of internal and external events Thus the notion of function is certainly of fundamental importance in engineering. This series of workshops is an example of the increasing concern engineers are showing in understanding explicitly what, as engineers, they understand implicitly and use in their everyday work. Making such knowledge explicit would help in automating or facilitating many engineering tasks. As it happens, one of the focus areas in the field of artificial intelligence is knowledge representation, and many people working in the intersection of AI and engineering have found that engineering problem solving is a rich domain to apply knowledge representation and problem solving ideas. Much work has been done in understanding the representational foundations for engineering knowledge (see Forbus[1] for a survey of work in this vein). In our own Laboratory, for over a decade, one of our focus areas has

been functional reasoning, dealing with the same set of issues that the participants at this workshop series have been grappling with.

Our representational framework has been called Functional Representation (FR, for short) and has been widely applied to a number of different domains. That work has been widely reported on and a couple of surveys[2,3] describe progress up to 1995 or so. I will just very briefly describe the goals of that line of work. However, the main purpose of this paper is to focus on the problems of defining function in a formal way, and to propose a specific definition. As the reader will see, the real goal in defining function is not to try to legislate how everyone should use the term, but to make clear what are the various distinctions that need to be recognized, whatever the names different people might to various distinctions.

FUNCTIONAL REPRESENTATION

FR is a framework in which to represent an agent's understanding of how some device works. Often, when people understand 'how things work," they give information of the following three types: (i) what the device is intended to do (function), (ii) the causal process by which it does it, and (iii) how the process steps are enabled by the functions of the components of the device, the way they are connected together and the underlying domain laws. FR provides language primitives using which the above elements can be represented.

For example, in the case of the familiar electric buzzer, the function might be represented as:

`Function` (*buzz*) `of` (*buzzer*):
    `To_Make` (*sound_at(clapper)*)
    `If` (*switch(pressed)*)

In the above, the items in the `Courier` font, are keywords of the FR language. The items in *italics* are device-specific terms. The process by which it accomplishes this might be represented (somewhat informally here) as:

1. *Switch(pressed) → circuit (closed) → current-through-coil (yes) → space_around_coil (magnetized) → clapper_arm (raised)*
2. *Clapper_arm (raised) → circuit (open) → current-through-coil (no) → space_around_coil (*`NOT` *magnetized) → clapper_arm (dropped)*
3. *Clapper_arm (dropped) → circuit (closed) → <repeat remainder of 1>*
4. Abstraction*: {clapper_arm (raised) → clapper_arm (dropped)*`}*` *→ sound_at(clapper).*

Together they explain how the functional state comes about. This kind of process explanation is called a causal process description (CPD) in FR. Each of the transitions may be additionally explained by annotating it with a statement about how that transition comes about. In FR, we have a taxonomy of such explanations. `By-Function` <..> `of Component` <..> explains a transition by attributing it to the function of some component, i.e., the transition is causally enabled by the fact that some component has the specified function. `By-CPD` <..> annotation explains a transition by saying, essentially, that the transition is a shorthand for a longer process, which is already understood, and if one wants the details,

look at *that* process. By-Domain_Law <..> explains a transition by appealing to a law, such as Ohm's Law equation. There are other types of annotation, but the examples above should suffice to give an idea of what sort of thing an annotation is. These annotations are not meant to be complete explanations for the transitions, but ones that the modeler wants to focus on. They not simply informal things for human comprehension – they are intended to be processed by reasoning modules for simulation, diagnosis and other problem-solving activities.

Let us look at some examples of annotation. Consider the clapper as a component. The clapper may be said to have an *electrical* function (conduct electricity between its terminals when it is in the dropped position), a *magnetic* function (create magnetic field in a certain region when it is electrified), a *mechanical* function (proper rising and falling of the clapper arm, as the clapper is magnetized and demagnetized) and an *acoustic* function (make sound when clapper arm hits the clapper repeatedly).

In the abstraction step 4 above (where * represents repetition, as in Kleene star in finite state machines), one might attach the following explanatory annotation to the above transition:

By_Function(*acoustic*) of component (*clapper*).

The transition,

space_around_coil (NOT *magnetized*) → *clapper_arm* (*dropped*)

might be annotated as

By-Function(*mechanical*) of component(*clapper*).

The transition

circuit (*closed*) → *current-through-coil* (*yes*)

might be annotated as:

By-Domain_Law (*Kirchoff's Law*)

For many purposes, the annotation involving By_Function is very useful. For example, suppose the buzzer has a malfunction, i.e., it is not buzzing when the switch is pressed. FR can be used to make a diagnostic analysis as follows.

i. The predicate involved in function definition is To_Make (*sound_at*(*clapper*)). This predicate appears in process description 4. That means somewhere in the chain *clapper_arm (raised)* → *clapper_arm (dropped)* there is a problem.

ii. This chain is the process fragment 2. Each of the transitions in this chain is checked. If the predicate in node *i* is true, but node *i+1* is not, then whatever structural element is responsible for that transition is the cause of the problem. For example, if *space_around_coil* (NOT *magnetized*) is true, but *clapper_arm* (*dropped*) is not, then looking at the annotation, By-Function(*mechanical*) of component(*clapper*), we can then generate the hypothesis that the problem is due to the clapper malfunctioning mechanically.

In various papers, summarized in[2, 3], we have discussed the use of FR for diagnosis, case-based design, simulation, explanation and so on. In addition to work in the FR tradition, there has been work in AI identifying specific functional vocabularies in various domains. Much of this work can be thought of as content-theory that can fit comfortably with the notions developed here. A set of basic roles in mechanical interactions is provided in Hodges[4] and, in a somewhat different subdomain, in Goel[5]. Roles that components of loops play in algorithms are given in Allemang[6], and roles that seem to be common to different domains sharing the ontology of flows are given in Chittaro, et al[7]. A similar

vocabulary of functional roles common to flow systems is provided in Lind[8] and Kumar[9]. A line of research in visual understanding (see, e.g., Stark[10]) focuses on recognitional algorithms that use a functional rather than a structural definition of objects to be recognized. Thus a recognizer for chair would see if the object could in fact support the sitting of a person rather than look for specific subparts.

Keuneke[11] has proposed a number of categorizations of functions. She identifies types of functions such as `To_Make`, `To_Control`, `To_Maintain`, `To_Prevent`, etc., each enforcing somewhat different relations between the functional predicates. In Iwasaki and Chandrasekaran[12], these function types are formalized and the formalization used to verify designs. In particular many FR ideas have been applied to process systems, both for diagnostic analysis and design. Two recent references that deal with the application of FR ideas for the construction and use of component libraries for design are Miller, et al[13, 14].

DEFINITION of FUNCTION

With this very brief introduction to the FR framework, I want to move on to another issue of fundamental importance for functional modeling, and that is the question of what exactly is meant by the term *function*. The FR framework treats function as a property of a device, which may be appropriate in many situations. However, we also often have a desired function in mind even when we don't have a device for it. In fact, this is how design tasks get their beginning. Function appears to be the name for a *set of related ideas*, rather than a single concept. In different situations where the term is used in engineers' activities, it may mean slightly different (though related) things. There is also quite a bit of confusion between function and behavior in the literature. We hear or see statements like, "Behavior is based on a science model while function is subjective," or "Function is some abstraction of the behavior of the object." Being able to talk precisely about the family of concepts surrounding function requires establishing a precise language of talking about objects, properties, behavior, connections, causal effects, etc., because it is in terms of these other concepts that the notion of function itself would be defined. Such a language has been called *domain ontology* in AI.

What we want to do in the rest of the paper is to ask the following question. What is the minimal ontology within which we can talk about function? We will propose a definition for function, but do concede in advance that it is for only one of the related meanings of the term. However, we hope to persuade you that the distinctions we make in getting to our definition would be useful to clarify the other members of this family of definitions. For work in particular domains and applications, additional constructs and content theories will be needed, but, if the effort is successful, the minimalist ontology will be usable by all.

Some Complexities

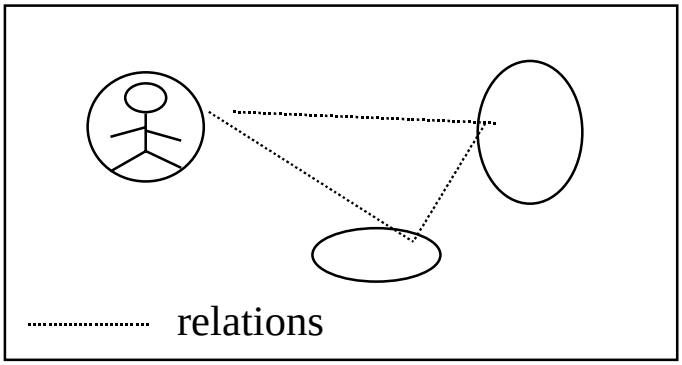
In defining function we need to take account of some complexities:

1. *Dynamic versus static functions*. Much of the work on functions has focused on situations where the function arises from dynamic activities of the object, i.e., state changes. For example, the buzzer achieves its function by virtue of the various state changes that occur when the switch is pressed. On the other hand, there are objects that achieve their function simply due to their structure and not due to state-change behavior. For example, a chair or a paperweight achieves its function by virtue of its structural properties. A definition of function should ultimately cover both the types of function.

2. *Functions of objects versus functions of processes*. Not only objects but processes have functions. For example, boiling is a process that can be used to create steam. Distillation is a process that can be used to separate components.
3. *Functions as states versus function as processes*. In the case of the buzzer, the function of buzzing can be defined in terms of achieving a certain state. But states themselves, on closer look, can turn out to be processes. In the case of buzzing, the underlying process is one of repeated state changes involving the clapper arm hitting the clapper.
4. *Functions involving users (intentional agents) versus functions of subsystems in larger systems*. Often functions are defined in terms of external users, such as in the case of the buzzer. On the other hand, components within a device have functions as well, but their functions are not defined in terms of an external intensional agent.
5. *Functions involving creating and destroying of abstract or concrete "matter."* Much of the work on function has so far involved only situations in which the total amount of matter does not change. Again, the buzzer is a good example. On the other hand, we have the function of the immune system, where the task is to destroy bacteria. In process engineering, it is common for matter of one type to be destroyed and matter of another type to be created.

The above is a list if issues that a formal definition of function should eventually be able to handle, not a list of problems for which we will offer a solution in this paper. The list is intended to be a consciousness-raising effort.

A Simple Ontology



Let us consider an extensible domain of discourse or a world W. The world is characterized by the following:

- Objects
  - which have *attributes* (some of them constants, some state variables), and
  - which may be physical, informational or intentional
- Inter-object *relationships* ("connections" or "interactions")
- *Causal models* consisting of *dependency relations* between the attributes under various relations between objects.

By extensible, we mean that new objects may be added or objects deleted, new relationships defined and so on. Physical objects are those that are made of matter, information objects are entities such as

software and plans, and intensional objects are those with goals, such as human beings.

Behavior

The term "behavior" is used often in the description of devices. Referring to the distinction that we made earlier between static devices and dynamic devices, it is interesting that people don't seem to use the term behavior to describe the values of static attributes. Behavior as some kind of description of state variables is common theme in all the usages that I have come across. However, the term is used to refer to a large variety of ideas relating to dynamic systems.

Consider a dynamic W, where causal relations involve state changes. *Behavior* has been used in the literature to refer to one or more of the following:

- the values of all the state variables at some instant.
- the values of state variables associated with specified variables (observables, output variables,..) at some instant.
- the values of either some or all state variables over several instants
  - behavior trajectory
  - behavior as a sequence of partial states (where partial states are state descriptions in which only some of the state variables are specified, thus they are equivalence classes of states.
- temporal sequence of interactions between two objects (such as the series of interactions between a user and an ATM machine for the user to deposit or withdraw money).

Generating the above list is an example of how a clear and simple ontology helps us to make precise distinctions and be clear about what one might mean by a term such as "behavior."

Constraints

Design specifications often include not behavior or properties as such that are desired, but *constraints* between properties or behaviors. However, behavior in one sense is really *constraints on behavior* in another sense. For example, behavior in terms of partial states defines equivalence classes of behavior in another sense.

Constraints are often specified using the language of predicates. Typically, constraints declare that certain predicates defined over object properties or behaviors need to be true. *Examples*: Using notation, p(O) to refer to property p of object O, and P(p(O_i), q(O_j),..) to refer to predicates asserting relations between different properties

1. P(p_8(O_1). This asserts a constraint on a single property of a single object, e.g., *diameter of pipe has to be greater than 5*.
2. If P_9(p_5(O_1),p_6(O_1)) then P_7(p_9(O_2),p_5(O_3)). This asserts a constraint on the properties between different objects, e.g., *If diameter of pipe 1 is greater than the length of pipe 1, then the size of pipe 2 and should be greater than the size of pipe 3*.
3. If <..>, then P(…) followed by P(..), followed by P(..). Example: *If switch is pressed, then the coils should be magnetized, followed by the lifting of the clapper,…*

A Definition of Role or Function

Now we are ready to move on to a definition of function.

- Let F be a set of constraints defined in some W. Let Θ be a new object introduced into W, in a relationship R(Θ,W) and let W_Θ be the extension of W when Θ is included. If Θ causes F to be

satisfied in W, we say that Θ plays, performs or has the *role* F in W_{Θ}. If F is intended or desired by some agent A, then we say that Θ has or performs, the *function* F in W for A. Often A is understood to refer to a designer or a user, and is dropped.

Comments on the definition

Function versus Role. In engineering language, unlike in mathematics, function has a design connotation to it, i.e., it is something someone – a designer, a user – intends. On the other hand, role is a descriptive term, neutral with respect to intentions, used in discourse in physical science. One can talk about the "The role of the moon in causing tides," or, "The role of the meteors in causing craters.." Most people would find "The function of the..." here strange, because physical science does not postulate that someone intends for the craters to be formed. On the other hand, if one were making an artificial solar system, then one might actually design meteors and cause them to make craters. It is OK to say "The ledge functioned as a chair," since some user intended that role. Biological systems are an exception: "The function of the heart is to pump blood. " Here evolution is generally taken to play the role of a *defacto* designer.

The above definition treats roles and functions as the *same kind of entities*. They are both *effects* that the object under discussion has on its environment. Whether something is a role or a function does not depend on the object configuration.

Object-independence of the Definition

The definition of F does not mention anything at all about the property or structure of Θ. Compare: "The function of the buzzer is to have the clapper make a sound when the buzzer's switch is pressed," with "When a user wants, sound is to be produced at spatial location L (where L is in W, not part of the object to be designed)." The proposed definition of function enables us to specify the function before the buzzer is designed, or to search a device library for a component with a desired function without having any idea of the structure of the component. Perhaps component C_1 requires it to be twisted at point A for the function to be achieved, while component. C_2 requires it to be pushed at point B for the same function.

Mode of Deployment

In the definition of function, we talked about the *relation* R(Θ,W). We intend by this term the specification of all the *causal interactions* between Θ and objects in W; that is, specification of what one normally thinks of as connections between objects (such as electrically connecting electrical terminal t_1 of Θ with electrical terminal t_2 of object O_5 in W) and, more generally, a specification of all causal interactions between Θ and W, such as *interactions between a user and* Θ (often called *usage scenarios*). In the context of design, we term R(Θ,W) the *mode of deployment* of Θ in W.

Different functions may be attributed to an object under different modes of deployments (MoD). For example, given a battery, with a causal model that describes its mechanical, electrical and thermal properties, functions *provide_voltage(t1,t2)* and *support_paper* can be attributed to the object. The former function is achieved under MoD: t_1,t_2 *electrically_connected* to *electrical terminals* of object in W1, and the latter under the MoD: *bottom_surface* in relation *atop* with object *paper* in W2.

The Design Task

The notion of function is used in many engineering tasks: design and diagnosis being the most prominent. It would be interesting to see how this definition of function clarifies the design task definition.

Definition. Given,

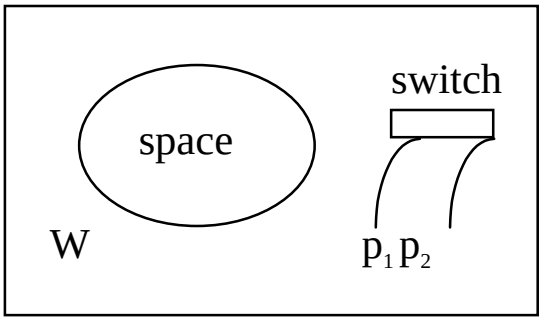
- a set F of property/behavior constraints, called a *function*, defined in some (generic) W
- a set C of constraints on construction of objects
- (optional) a set N of constraints defined over W

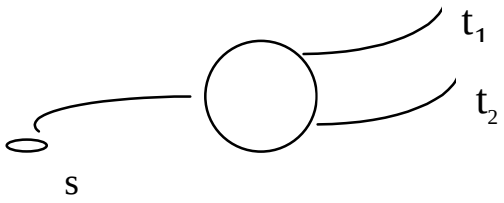
the design task is that of describing

- an object Θ, satisfying C, and
- a MoD for relating Θ to W
  - such that when Θ satisfies the MoD, Θ has the function F, I.e., it *causes* F to be satisfied; and Θ also satisfies N.

Comments on the definition. Construction constraints specify such things as the kinds of component objects from which to make Θ and constraints on causal connections between them. N specifies the so-called non-functional constraints (example, "total weight is less than 5 lbs"). Note that we didn't say that Θ *causes* them to be satisfied. Unlike definitions of the design task in the literature, including in Chandrasekaran[15], which only focused on specifying the structure of Θ, the current definition emphasizes MoD as well.

Examples of the Definition of Function





Let us consider a simple example of a thermostat to explain the definition. Let O be an object with structural elements, *s*, t_1 and t_2. Let the world W contain objects

space, with attribute *Temperature*

electrical_switch, with attribute, *switch_state* which can take values, *ON* and *OFF* , and structural elements, p_1 and p_2, called terminals.

We can attribute the function:

Function *Regulate*(*switch*, T_{set}) defined by the constraints, C:

If *Temp*(*space*) < T_{set}, *switch* is *ON*

If *Temp*(*space*) >= T_{set}, *switch* is *OFF*

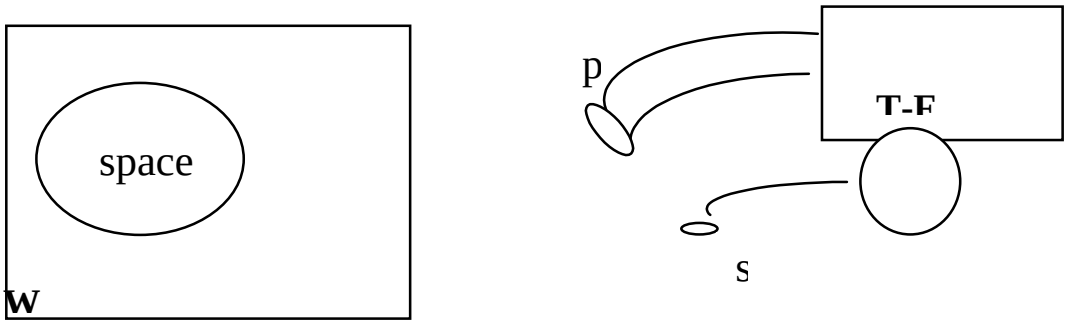
to the object O, if, when O is embedded in W with the mode of deployment:

s contained in *space* of W

terminals of *Thermostat electrically_ connected* to *terminals* of *switch*

the functional constraints C are satisfied.

In fact, an object O to which the above function can be attributed is called a thermostat. A thermostat's function is to regulate the opening and closing of a switch based on a set temperature. Let us see what the specification would like if we considered a furnace-thermostat combination.



In this case, W contains object *space*, with attributes *Tempr, heat_being_added(t).*
The function is defined by the constraint:

If *Temp*(*space*) < T_{set}, *heat_being_added(t)* increases
If *Temp*(*space*) >= T_{set}, *heat_being_added(t)* is 0

The object O has structural elements s and p. If the object O is embedded in W using the Mode of Deployment, p and s *contained_in space* of W, and the functional constraints are satisfied, we can attribute the function *Regulate (tempr, space)* to the object O. In fact objects with this function are furnaces with thermostats set to T_{set}.

DISCUSSION

The work in the FR tradition and on CFRL[12], [16], [17] treat function as an abstraction of the behavior of a device, as does reference[18]. This is entirely adequate for many purposes, but the separation of function from the object's properties will help those investigations to reach wider applicability. Among recent papers, references[19], [20] share some essential intuitions with the present paper in that they make an effort to separate the behavior of an object from its function. The present paper proposes what appears to be the simplest ontology in which this notion can be explicated, and also makes various kinds of unifications, such as between static and dynamic objects and between intended functions and functions observed in natural systems.

Multiple Realizability

Locating the function in the effects on the environment, rather than in the object, clarifies the notion of multiple realizability of functions. When the function is defined without any reference to the structure of an object, different realizations of the function become possible. This multiple realizability also suggests

criteria by which one could decide at what level of organization the effects of an object should be described. For example, we may describe the role of the thermostat as "When the room temperature is below T_{set}, the furnace is on." Note that both predicates are of objects in the environment. However, an effect of the thermostat so configured is also eventually to make a person in the room warm. Does this mean that it is equally plausible to attribute to the thermostat the function, "make a person warm"? Suppose that "make a person warm" can be multiply realized, for example, one, by covering the person with a woolen blanket and two, by using a thermostat-controlled furnace. The thermostat's effect is actually on the furnace, while the thermostat-furnace configuration as a whole has the effect of keeping the person warm. Thus, in the given modeling context, the thermostat's function is best stated as its role in linking the temperature of the room to the starting of the furnace.

In this paper, we have just touched on the issues involved in the formulation of an underlying ontology, and using it to be precise about a number of issues that are central to functional modeling. Our goal is to continue to develop this ontology and use it to be clear about the meaning of many terms used in functional modeling and reasoning about systems and devices.

ACKNOWLEDGMENTS

The research is supported by ONR grant no. N00014-96-1-0701, DARPA order no D594. The authors thank Susan Josephson for a number of useful discussions.

REFERENCES

1. Forbus, K.D., *Qualitative Physics: Past, present and future*, in *Exploring Artificial Intelligence*, H. Shrobe, Editor. 1988, Morgan Kaufmann: San Mateo, CA. p. 239-296.
2. Chandrasekaran, B., *Functional representation and causal processes*, in *Advances in Computers*, M.C. Yovits, Editor. 1994, Academic Press. p. 73-143.
3. Chandrasekaran, B., *Functional representations: A brief historical perspective.* Applied Artificial Intelligence, 1994. **8**: p. 173-197.
4. Hodges, J., *Naive mechanics: A computational model of device use and function in design improvisation.* IEEE Expert, 1992. **7**(1): p. 14-27.
5. Goel, A.K., *Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving*, . 1989, The Ohio State University.
6. Allemang, D. and B. Chandrasekaran, *Functional Representation and Program Debugging*, in *Proceedings of the 6th Knowledge-Based Software Engineering Conference*. 1991, IEEE Computer Society Press. p. 136-152.
7. Chittaro, L., C. Tasso, and E. Toppano, *Putting functional knowledge on firmer ground.* International Journal of Applied Artificial Intelligence, 1994. **8**: p. 239-258.
8. Lind, M., *Modeling goals and functions of complex industrial plants.* Applied Artificial Intelligence, 1994. **8**(2): p. 259-283.
9. Kumar, A.N., and S. J. Upadhyaya, *Function-based discrimination using model-based diagnosis.* International Journal of Applied Artificial Intelligence, 1995. **9**(1): p. 65-80.
10. Stark, L., and K. W. Boyer, *Achieving generalized object recognition through reasoning about association of function to structure.* IEEE Trans. Pattern Analysis and Machine Intelligence, 1991. **13**: p. 1097-1104.

11. Keuneke, A., *Device Representation: The significance of functional knowledge*, in *IEEE Expert*. 1991. p. 22-25.
12. Iwasaki, Y. and B. Chandrasekaran, *Design Verification Through Function- and Behavior-Oriented Representation: Bridging the gap between function and behavior*, in *Artificial Intelligence in Design '92*, J.S. Gero, Editor. 1992, Kluwer Academic Publishers. p. 597-616.
13. Miller, D.C., *et al. A Sharable Knowledge Repository for Decision Support in Life Cycle Design*. in *AIChE Mtg.* 1996. Chicago, IL.
14. Miller, D.C., *et al. Sharable engineering knowl-edge databases for intelligent system applications*. in *Computers chem. Engng. 21*. 1997. Trondheim, Norway.
15. Chandrasekaran, B., *Design Problem Solving: A task analysis*, in *AI Magazine*. 1990. p. 59-71.
16. Vescovi, M., *et al., CFRL: A Language for Specifying the Causal Functionality of Engineered Devices*, in *Eleventh National Conference on AI*. 1993, AAAI Press/MIT Press. p. 626-633.
17. Iwasaki, Y., *et al., How Things Are Intended to Work: Capturing functional knowledge in device design*, in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. 1993, Morgan Kaufmann: Mountain View, CA. p. 1516-1522.
18. Umeda, Y., H. Takeda, T. Tomiyama, and H. Yoshikawa, *Function, behavior and structure*, in *AI in Engineering*. 1990, Computational Mechanics Publications and Springer Verlag. p. 177-193.
19. Sasajima, M., Y. Kitamura, M. Ikeda, and R. Mizoguchi, *FBRL: A function and behavior representation language*, in *International Joint Conf on Artificial Intelligence*. 1995, IJCAI, Inc. and Morgan Kaufmann: Montreal. p. 1830-1836.
20. Larsson, J.E., *Diagnosis based on explicit means-end models*. Artificial Intelligence, 1995.