Engineering with Computers (2000) 16: 162–177
© 2000 Springer-Verlag London Limited

Function in Device Representation

B. Chandrasekaran and John R. Josephson

Laboratory for AI Research, Department of Computer & Information Science, The Ohio State University, Columbus, OH, USA

Abstract. *We explore the meanings of the terms 'structure', 'behaviour', and, especially, 'function' in engineering practice. Computers provide great assistance in calculation tasks in engineering practice, but they also have great potential for helping with* reasoning *tasks. However, realising this vision requires precision in representing engineering knowledge, in which the terms mentioned above play a central role. We start with a simple ontology for representing objects and causal interactions between objects. Using this ontology, we investigate a range of meanings for the terms of interest. Specifically, we distinguish between* function as effect *on the environment, and a device-centred view of device function. In the former view, function is seen as an intended or desired role that an artifact plays in its environment. We identify an important concept called* mode of deployment *that is often left implicit, but whose explicit representation is necessary for correct and complete reasoning. We discuss the task of design and design verification in this framework. We end with a discussion that relates* needs *in the world to functions of artifacts created to satisfy the needs.*

Keywords. Causal reasoning; Compositional modeling; Device ontology; Device simulation; Functional reasoning; Functional representation

1. Introduction

Design exists in order to deliver artifacts that have desired functionalities. The concept of function is thus fundamental in engineering practice. Engineers' intuitive understanding of this concept has, until recently, been sufficient for the practice of engineering. While precision in expressing particular functions for particular devices has always been of value – if for no other reason than to avoid misunderstandings between the customer and the designer – such precision could be obtained by means that were highly specific to the class of devices. However, in recent years, the prospect of computers taking on more and more of the *reasoning* tasks involved in engineering design has placed a premium on being explicit and precise about many of the intuitive concepts related to design, concepts such as *function*, *behaviour*, *structure*, and *causal relations*. This is because reasoning requires explicit representation of knowledge, and knowledge representation requires terms whose meanings are as clear as possible. Today's systems and devices may have components from a number of domains – mechanical, chemical, electrical, electronic and software elements are freely intermixed in contemporary complex devices. With the increase in such multidisciplinary design, if there is any hope at all of having the computer assist designers with reasoning tasks in engineering, then it is important to develop as general a framework as possible. If the concepts are captured in some generality, then the same representational framework may be used for reasoning about a wide variety of devices and processes. Otherwise, the representation of function in one domain, say chemical engineering, may not be compatible with the definition, say, in electrical engineering.

What kinds of tasks in design might computational reasoning systems help with? [1] One of us has presented a detailed task analysis of design, along with the knowledge requirements for its various subtasks. For current purposes, we will consider two subtasks as paradigmatic for the potential of the use of reasoning systems. Both of them can best be described in the context of compositional design.

In compositional design, the designer uses components from a component library (or components that are implicitly defined in terms of a 'technology', which in turn determines the type of components that are available for use) to specify a set of components and relations between the components as a

Correspondence and offprint requests to: Professor B. Chandrasekaran, Laboratory for AI Research, Department of Computer & Information Science, The Ohio State University, Columbus, OH, 43210, USA. E-mail: chandra@cis.ohio-state.edu

design. As the designer creates candidate designs by composing components, he needs to verify that the device in fact has the properties or the behaviour that can satisfy the functionality requirements. Performing this reasoning requires in turn a precise representation of the functions and behaviour involved. A powerful framework for compositional reasoning and a language – called CML – for representing the behaviours of components and composing the behaviour of devices has been proposed [2]. However, these frameworks still lack the ontology of function and behaviour in as much as a generality as might be useful. A recent enhancement of the CML language [3], based as it is on some of the ideas developed here, has provisions for representing functions.

A second subtask is that of choosing components from a component library. The designer might come up with a design in which only the function of a certain component is identified, but not yet the component itself. If the component library is indexed with the possible functions of the components, then an appropriate component may be retrieved if it is available in the library. In the current state of the art, such retrievals are possible in well-defined and restricted design domains. For multidisciplinary design, as mentioned, greater generality in representing functions, structure and behaviour will be necessary.

The main goal of this paper is to present an analysis of the notion of function in engineering, and to present a representational framework for it. Using a simple ontology for representing objects, their structure, behaviour and their compositions, we clarify the notions of structure and behaviour. The basic ontology for representing and reasoning about objects is not itself novel; most of it comes from the CML framework. Our goal is not to present the complete ontology that CML supports, but just those that are needed to support the distinctions which are needed to clarify the concepts of interest. A major problem in the current state of the art is that not only are there are deep ambiguities in the ways terms such as ‘function’, ‘structure’ and ‘behaviour’ are used by different researchers, but the researchers are often unaware of the ambiguities. Our goal is to identify some of the different senses, and to show how precision in representation can help in bringing out the different meanings.

2. A Simple Ontology

The world of devices is quite complex: objects, fields, flows and flow substances, actions, events, properties and causal connections are just a few of the elements of the ontology of this world. In our model, we will forgo much of the complexity, and use a simple model that is sufficient to make most of the distinctions. In this simple model, we restrict ourselves to devices made up of objects in some causal interaction. That is, we will not consider devices involving fields (such as electrical conductors in a magnetic field), or those involving flow substances (such as steam in heat exchangers). The analysis of the various terms that we propose can be extended to the situations involving these additional elements in a non-problematic way.

The compositional modelling framework facilitates representing causal knowledge modularly. This is achieved by representing knowledge about classes of objects, specifically about how the properties of an object causally influence each other; and representing knowledge about how objects interact with each other.

The basic element in the ontology is an *object*, represented in a *view*. Objects may be physical or abstract. Objects interact with other objects. A collection of objects in some causal relationship may itself be abstracted as an object. Any representations of real world objects are incomplete in principle, and what is included and what is omitted determine the point of *view*. All representations are in some view. Views can be related. A relation of particular interest, one we discuss in more detail later in the paper, is where a view is an *abstraction* of another. For example, representing a calculator as a device that performs calculations is one view, and as an electronic circuit is another view. These views can be related by indicating how the calculator variables are related to specific voltage variables in the circuit view. In principle, at least, the causal rules that relate input numbers and operations may be derived from the causal rules that relate the voltage and current variables in the circuit view. However, the same object may be represented in two quite unrelated views. A pocket radio can be represented in a view corresponding to its being a radio, but it may also be represented in a view corresponding to it being a paperweight. These two views are quite unrelated.

2.1. Objects

We start with an object representation with the following elements in it: ({causal variables}, {causal relations}). Causal variables are those properties of the object that may change because of a causal interaction. The causal variables may be continuous, discrete, qualitative or logical. Causal relations describe how changes in the values of a causal variable affect the other variables of the object. A variety of mathematical formalisms may be needed to represent causal relations: arithmetic, differential, logical formulas and algorithms are some examples. Given the values of independent causal variables, the values of the dependent variables can be calculated or inferred by making use of the causal relations.

Some of the causal variables may be identified with *terminals* or *ports* of the object. For example, one may talk about the voltage at *terminal* p_1 of an electrical resistor, the frictional force at *side* S_1 of a mechanical component, or value of program variable x at the *output* of a program module. Such talk is especially convenient in the case of physical objects, i.e. objects that have an extension in space, or abstract objects for which we may use spatial metaphors. An example of the latter would be program variables that are often associated with certain *ports* of software modules. Many causal variables may have no location associated with them at all. For example, the *resistance* of a *thermal resistor* is a causal variable that is associated with the object as a whole.

The variables as well as the relations may be *typed* for convenience. For example, an object, variables or sets of causal relations may be defined as *electrical* and generic representations may be provided. Then, when an object, variable or set of relations is declared 'Type: *electrical*', the generic template may be simply instantiated.

The causal variables may or may not be explicitly represented as state variables. For a simple electrical resistor, a causal relation may just state, '$I * R = v$' where v is the variable standing for the voltage between the two terminals, I, the current through the resistor, and R is the resistance value. On the other hand, if we wish to reason in the context of an alternating current source, the variables may be explicitly written as I(t) and v(t). For an electrical capacitor, an appropriate representation would treat the electrical variables as state variables. Thus, the corresponding causal relation might say, '$I(t) = C* dv(t)/dt$' where C is the capacitor and dv/dt is the time derivative.

Examples

We now give examples from the electrical domain to illustrate the representational ideas:

Electrical-Object
Object: *Electrical-object*, with terminals p_1 and p_2
Variables: $v_1(t)$, $v_2(t)$, type *voltage*, at terminals p_1 and p_2
$I_1(t)$ and $I_2(t)$, type current, into terminals p_1 and p_2 (1)

Thermal-Object
Object: *Thermal-Object*, with terminal *surface*
Variable: T_S, temperature at surface (2)

Electrical Resistor (Fig. 1)
Object: *Resistor*
Type-of: *Electrical-object*
Type-of: *Thermal-object*
Variables: R, resistance value
Causal Relations:
$I_1(t) = - I_2(t) = (v_1(t) - v_2(t))/R$
$T_S = f_R(I_1)$, where f_R is an appropriate function (3)

Because the *resistor* is a type-of *electrical-object*, it will inherit the *electrical* terminals as well as the associated *voltage* and *current* variables; and because it is a *thermal-object*, it will inherit the *surface* terminal and the *temperature at surface* variable. Additional representational elements that apply only to *resistors* (such as the additional variable R) are represented as part of *resistor*. Inheritance is a representational convenience, not part of the basic ontology of device representation. Inheritance as a general representational tool has to be used carefully: tangled hierarchies and circular definitions may occur, causing difficulties during reasoning. However, these are problems that are not specific to inheritance in device representations. Much has been said about how to construct safe and useful object hierarchies in the object-oriented programming literature. So, we forego further discussion of this subject here.

Battery (Fig. 2)
Object: *Battery*
Type-of: *Electrical-object*
Variables:
B: *battery voltage*
r: *battery internal resistance*

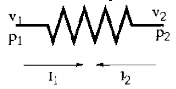

Fig. 1. A simple resistor with resistance value R, electrical terminals p_1 and p_2, voltages v_1 and v_2 at the terminals, currents I_1 and I_2 into the terminals, and temperature T_s at the thermal terminal *surface*.

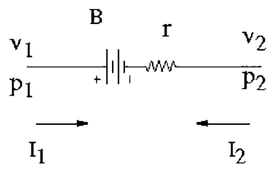

Fig. 2. A simple battery with voltage B and internal resistance r, terminals p_1 and p_2, voltages v_1 and v_2 at the terminals, and currents I_1 and I_2 into the terminals.

Causal Relations:

$$I_1(t) = -I_2(t) = (v_1(t) - v_2(t) - B)/r \quad (4)$$

Electrical Switch (Fig. 3)
Object: Electrical Switch
Type-of: *Electrical-object*
Terminal: *Push-button*
Variables: *Sw* (*Switch-position*) type: Boolean (*open, closed*); *action-on* (*push-button*), type: Boolean (*push, pull*);
Causal Relations:

If *action-on(push-button) = push, Sw = closed*

If *action-on(push-button) = pull, Sw = open*

If Sw = open, $I_1 = I_2 = 0$ (5)

If Sw = closed, $v_1(t) = v_2(t)$

In addition to its *electrical* terminal, the *Switch* has a terminal *push-button*, with an associated boolean variable that specifies whether the action on the *button* is *push* or *pull*. The description in Example (5) economises by mentioning only this additional terminal, since the description includes the element that the *switch* is a type-of *electrical-object*.

Causal relations for *resistor* and *battery* are expressed in terms of arithmetical, two-way, relations. On the other hand, the switch calls for one-way rules. That is, in the case of voltages and currents, any one of them can be independent, and the others can then be treated as dependent. In the case of the relation between the switch variables and currents, however, currents do not have a causal effect on the switch variable, while the latter has an effect on the former. The one-wayness of such causal relations needs to be distinguished from the two-wayness of the inferences. That is, knowing that there is no current through the switch, using the causal relations, one can *infer* that *switch* is *closed*, but one cannot say that lack of current *caused* the *switch* to be in *closed* state.

The causal relations in general can be quite complex, and might need additional representational elements. For example, representing probabilistic effects would require new elements in the ontology. Representing situations where objects can be created or destroyed similarly calls for a more complex representational scheme.

2.1.1. Completeness of Object Models

While no model can be complete even in principle, there is a requirement that might be called *relative completeness*. For example, while we have the freedom not to include variables such as size and material strength in the resistor model, once we include the *voltage* variable, we need to include the other variables that can affect *voltage*, consistent with the degree of approximation that we are aiming for. That is, we need to include the *currents* and the *resistance value* as part of the model, otherwise we will not be able to reason about *voltage* changes at all. Once we decide on representing a causal phenomenon – in this case the causal phenomenon captured by Ohm's law – all the variables that participate in the phenomenon need to be included in the model. While relative completeness is a requirement for a reliable representation, ensuring that a representation is in fact relatively complete may not be easy. In fact, causal models in science are often revised, since new variables might be discovered that have an effect on the variables of interest; thus, what was assumed to be a relatively complete model is seen in hindsight not to have had that property after all. However, in well understood engineering domains, it is generally possible to identify the variables that need to be represented in order to be able to reason about the phenomena of interest.

2.2. Causal Interaction Between Objects

To reason about the behaviour of configurations of objects, we need to represent causal interactions *between* objects. For physical objects, what kinds of interactions take place between them is ultimately a function of some physical relationship between them. An object may be in a field created by another, an object may be in electrical contact with several other objects at certain locations, a specified

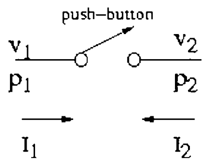

Fig. 3. A simple switch with electrical terminals p_1 and p_2, physical terminal 'push-button', voltages v_1 and v_2 at, and currents I_1 and I_2 into, the electrical terminals.

side of an object may be welded to a specified side of another object, and so on. Each of these types of physical relations corresponds to specific ways in which properties of one object affect the properties of the other object. A physical relationship type is characterised by, among other things, how many and what types of objects are to be involved in the relationship.

Representationally, it is useful to identify types of relations between objects generically, and describe the causal interactions that the relations correspond to. Thus, a generic relation type specification would look like:

Relation Type *Rel: Rel*($O_1, O_2, \ldots O_n$, specification of types and numbers of objects, and the generic physical parameters of objects in the relation)

Causal Interactions: Causal constraints on the variables of $O_1, O_2, \ldots O_n$

For example, one might identify a type of physical connection between a finite number of objects called *electrical-connection* and associate specific causal interactions with that type of physical relation as follows:

Relation: *Electrically-Connected*($p_{i1}(E_1)$, $p_{i2}(E_2) \ldots p_{in}(E_n)$), no other terminals in the relation: $p_{ij}(E_j)$ is the terminal p_{ij} of object E_i, physically connected to other terminals such that they are in electrical contact.

Causal Interactions: $v(p_{i1}(E_1)) = v(p_{i2}(E_2)) = \ldots v(p_{in}(E_n))$ where v(p) is the voltage at the terminal p and I(p) is current into terminal *p*.

$$\sum_{j=1}^{n} I(P_{ij}(E_j)) = 0$$

In Eq. (6), the causal effects are in the form of changes to the values of the variables of the objects involved. In general, the interactions may be more complex. Physical relations may be changed (e.g. a diode may open in a circuit); objects may be destroyed (a bacterium may be killed by the immune process), created, and split (a cell may split into two cells); an object may go into modes in which a different set of causal relations comes into play; and so on.

Another example of a relation is the following (illustrated in Fig. 4).

Relation: *Spatially-immersed* (terminal *surface* of *thermal-object*, *vol*, volume of space with air). (*Thermal_object* is described in Example (2).)

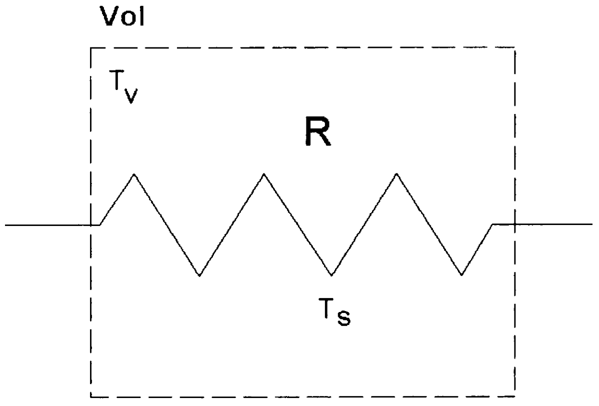

Fig. 4. A simple resistance with surface temperature T_s is in a volume of space *Vol* whose temperature is T_v.

Causal Interactions: T_v, the temperature of *Vol*, increases from T_A, its ambient value, but stays below T_s, the temperature at *Surface* of *Resistor*. (7)

2.2.1. Actions

Sometimes, objects (including intensional objects, such as humans) *act* on other objects. For example, a user may *push* a button on a device; during the operation of a device, one part may *hit* another part. For the purposes of causal reasoning, we can regard actions as instantiating specific types of relations. Thus, the action of pushing a button might be identified with a specified terminal of an object coming into an *electrically-connected* relation with a terminal of another object. The physical relation in turn corresponds to the causal constraints in Eq. (6) becoming operational. We can thus reason about how an action sets in motion causal consequences.

2.2.2. Interfaces and Interface Variables

Certain variables of a device may at times be explicitly identified as interface variables, variables that mediate causal interaction with other objects. For example, while voltages at different locations in an electrical object may be of interest to a modeller, *terminals* are identified are as loci of interaction with other electrical objects. Such a distinguished location is an *interface*, and the variables associated with such interfaces are interface variables. An interface may itself be an object that belongs to an object configuration. For example, electrical devices, which are compositions of many electrical objects, often have two interfaces: one a switch that may be operated by a user, and the other a pair of terminals that may be connected to an electrical source.

With this basic ontology of objects and their causal interactions in place, we are in a position to investigate the terms *structure, behaviour* and *function* as used in engineering practice.

3. Structure

3.1. Structural Relations

As mentioned earlier, when objects interact, causal effects may simply change the values of the variables, or they may include changes in one or more of the relations. For example, in an electric buzzer, closing the circuit causes the circuit to become open at a later instant. In general, some relations may no longer apply, and new relations may come into play. When a relation no longer applies, the causal interactions associated with the relation, as in Example (7), no longer apply. By the same token, as a new relation is established, a new set of causal interaction rules comes into play. It is often useful to categorise the relations in the initial configuration into those that remain stable through causal interactions of interest, and those that can change during such interactions. The former are often specifically called *structural relations*. Of course, whether a relation is stable, and thus structural, is not a permanent fact about the object configuration, but with respect to a modelling view.

3.2. Structure of an Object Configuration

A new object may be created by composing some objects into a larger configuration. Composing objects amounts to specifying relations between the objects that set up the causal interactions between the objects. The specification of the objects in the configuration and the structural relations between them is called the *structure* of the configuration. Structure defines the basic interaction framework, within which other, temporary, interactions take place.

Example

Suppose we build a simple circuit using the three electrical components defined earlier. Using the notation terminal(object), if we connect terminals p_1(*resistor*) with p_1(*battery*), p_2(*resistor*) with p_2(*switch*), and p_2(*battery*) with p_1(*switch*), we will have a *circuit*. The composition of the *circuit* is given in Fig. 5, where the double lines indicate the structural connections. The structure of the circuit is then given by:

Object: *Electrical-Circuit*
Structure:
Component objects: *resistor* R_1, *battery* B_1, *switch* S_1
Structural relations: *electrically-connect* ($p_1(R_1)$,$p_1(B_1)$), *electrically-connect* ($p_2(R_1)$, $p_2(S_1)$), and *electrically-connect* ($p_2(B_1)$, $p_1(S_1)$). (8)

3.2.1. Causal Model of Composed Object

A configuration of objects, with a set of structural relations between them, is itself an object. The composed object's causal model can in turn be composed out of the models of the component objects and the models of the relations. Thus, if a configuration O has component objects O_1,O_2, . . .O_n and R_k, $k = 1$,. . .m, are the set of m structural relations between the objects, then the causal model of O is:

Object: O
Structure:
Component-Objects: O_1, O_2, . . .O_n
Structural Relations: R_k, $k = 1, \ldots m$

Variables:

$\bigcup_{i=1}^{n}$ Set of variables for component O_i

(Note: the variables are renamed to keep the variables of one component distinct from those of another component.)

Causal Relations:

$\{\bigcup_{i=1}^{n}$ {set of causal relations of O_i}$\} \cup \{\bigcup_{k=1}^{m}$
{{set of causal interactons that (9)
characterize R_k}}

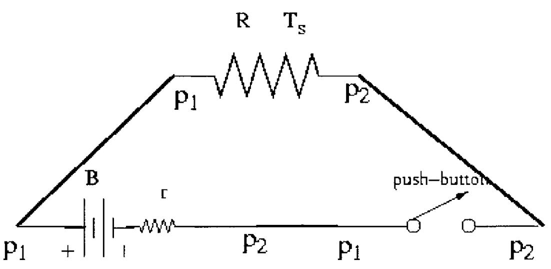

Fig. 5. The structure of a circuit, showing its components and how they are structurally related. Thick lines indicate which terminals of components are electrically connected.

For example, the representation of *Electrical-Circuit* will have as variables the *current_into_* and *voltage_at_terminal* variables of all the components. The set of causal relations will include the causal relations of the components (as described in (3)–(5)), plus the three sets of causal interactions that instantiate the *electrically-connected* interaction in (6) for the three connections described in the structural relations in (8). The new set of variables will look something like

{I_i into and v_i at terminal p_i of *Switch*, i = 1,2; I_i into and v_i at terminal p_i of *Battery*, i = 1,2; I_i into and v_i at terminal p_i of *Resistor*, i = 1,2; T_s, *temperature* at *surface* of *resistor*}

In many cases, it is useful to represent the composed object in a new abstract view. This abstraction can be as simple as renaming the variables and algebraically simplifying the causal relations, or may involve introduction of new abstract variables. To take the simple example first, the composed object may be re-represented as follows.

Object: *Electrical-circuit*

View: Abstraction1

Variables: I (current from the positive terminal of the *Battery*), T_s (temperature at surface of *Resistor*), action variable *action-on*(*push-button*).

Causal Relations:

If *action-on*(*push-button*) = *push*, I = (B/R+r);
If *action-on*(*push-button*) = *pull*, I = 0
$T_s = f_T(I)$. (11)

The representation in (11) does not use all the variables in (10). The set of causal relations is much simplified in comparison to the one that would follow from (9). The representation has chosen to discard the *voltage* variables, and to collapse the individual *current* variables into one variable, since algebraic solution of the equations in (9) for the circuit shows that the currents through the components have the same value. Figure 6 shows the circuit in this abstract view.

Abstraction operations may be more complex. Continuing with the circuit example, an abstraction of it, let call it *heater*, may be defined. In this representation, we might retain only the terminals *surface* and *Push-button*, and the associated variables. Figure 7 illustrates this object. The object is described as follows.

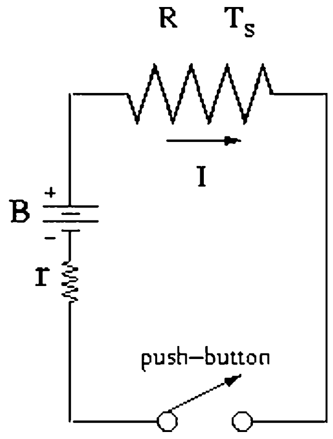

Fig. 6. The composition is abstracted into a new representation. In this example, the *voltages* are dropped, a single *current* variable is used, and terminals *surface* with variable *temperature* T_s, and *Push-button* are the only terminals retained.

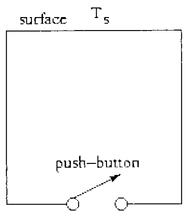

Fig. 7. *Heater*, a further abstraction of *circuit* in Fig. 6. The object is described only in terms of the variables associated with the *push-button* and *surface* terminals. Abstraction relations that relate the *heater* description to the *circuit* description enable moving from reasoning about *circuit* phenomena to *heater* phenomena.

Object: *Heater*

Variables: T_s (temperature at *surface* of *heater*), Boolean action variable *action-on* *(push-button* with values *push* and *pull*).

Causal Relations:

If *action-on*(*push-button*) = *push*, $T_s = T_{high}$
If *action-on*(*push-button*) = *pull*, $T_S = T_{ambient}$
(12)

The abstraction mapping between the objects in Figs 6 and 7 is quite straightforward: the variables included in the description for the *heater* are a subset of the variables for *electrical-circuit*. However, in general, for abstractions this need not be the case. Consider the electronic device called the *Adder*. As an electronic device, its components are all electrical, with *voltages* and *currents* as the relevant variables. However, at the device level, the relevant variables are *addend* and *sum* numerical digits. Thus, the device may be represented in two views, one that retains the individual views of the components, so that the composed device has all the current and voltage variables, and the union of all the relevant causal relations. The device might be represented in another view, an abstract *Adder* view, in which only the *Adder* level variables and relations are included, so that the behaviour of the *Adder* may be reasoned about without reference to voltages and currents. Finally, a mapping may be provided between the two levels. The mapping would show the relationship between the *Adder* level variables and the component view variables. Once all this is established, in theory, it should be possible to derive the *Adder* level causal relations from the causal relations at the component view. For example, in an Adder in which the numerals correspond to voltage levels at specific locations of the circuit, the mapping may look like:

Abstraction–Mapping (View: *Adder*, Abstraction-of View: *electrical-components*)
Numeral at *locationi* of *addendj* in View *Adder* $\leftrightarrow$
voltage v at *locationL*$_{ij}$ in View *electrical-components* (13)

Formally, given $O_{\text{component-level}}$, an object representation at the component compositional level view, O_{abstract}, a representation of the same object at an abstract view, and a mapping between the variables of the two views:

Set of causal relations for $O_{\text{component-level}}$
+ Mapping rules $\longrightarrow$ (14)
Set of causal relations for O_{abstract}

That is, the component-level view causal relations and the mapping information together imply the set of abstract view causal relations.

Re-representing objects at new abstraction levels and linking the representations at different abstractions levels are essential for reducing the complexity of reasoning. For example, we can reason about the behaviour of the adder using the *Adder* abstraction level without bringing in voltages and currents. However, if needed – for example, when there is a malfunction in the *Adder* – we can move to the lower abstraction level, and relate what is happening at the *Adder* level to what is happening to the components. An example of diagnostic reasoning by moving between abstraction levels is given in Chandrasekaran [4].

4. Behaviour

This term is used in the literature in different ways in different contexts. The intuition that the term appeals to is what the object *does*, but 'what an object does' is ambiguous in many ways. There is one distinction between 'behaviour' in the sense of an object's causal model versus 'behaviour' in the sense of a specific thing an object does. For example, one can say that Ohms Law, (i.e., the causal model in (3)) describes the behaviour of an electrical resistor. Here 'behaviour' refers to the underlying set of rules. On the other hand, saying 'The resistor's behaviour when 1 volt is applied between its terminals is to pass a current of 1 amp' is also a description of behaviour, but one that is expressed in terms of specific values. Even when we talk about behaviour in the sense of specific values, engineering discourse still abounds in ambiguities. Here is a range of meanings for the term 'behaviour'.

- *Beh-i*. The value(s), or relations between values, of state variables of interest at a particular instant: 'How did the car behave? It rattled when I hit the curve'. 'How does the widget behave? The ratio of output to input voltage is greater than one'.
- *Beh-ii*. The value(s), or relations between values, of properties of an object. This is closely related to sense (i) above. For example, one might say, a lintel *distributes* the load to the two sides. A window *transmits* the light from outside to inside the house. A paperweight *keeps* the paper in place. The italicised words are verbs, and as such, the descriptions are behavioural. In such descriptions, however, time is not explicitly mentioned. Thus, instead of thinking of these behaviours as state variable values at any specific instant, it is more perspicuous to think of them as relations between specified properties of an object.
- *Beh-iii*. The value(s) of state variables of interest over an interval of time. 'What did you notice about the behaviour? The BHP increased for a while, but then started decreasing'.
- *Beh-iv*. The value(s) of state variable(s) specifically labelled 'output' state variables, either at an instant or over an interval of time. 'The amplifier is behaving well – the output voltage is constant'.
- *Beh-v*. The values of all the state variables in the object description, either at an instant or over an interval of time. A graph that plots all the variables over time is often called the behaviour graph.

To complete the list, we include below the meaning discussed at the beginning of this section.

- *Beh-vi*. The causal rules that describe the values of the variables under various conditions.

In everyday use of the term, even in engineering discourse, the context removes any ambiguities.

The concept of function, which we discuss next, is closely related to that of behaviour. Unless otherwise specified, in the following sections we will use the term 'behaviour' to refer to *Beh-i* to *Beh-v* in the above list.

5. Function

5.1. Function as Effect vs. Function as What a Device Does

Our interest is in the use of the term 'function' within engineering. In engineering, function is an important concept in two ways. In design, the task is to make, or acquire, devices that have certain functions. In diagnosis, the task is to understand why a device is not functioning as expected. However, the term has been a source of endless discussion and attempts at definition.

The problem, in our view, is that the term that has many, but related, meanings, and attempts to identify 'one true meaning' are bound to fail. This sort of situation is quite common in ordinary language. Before the advent of Newtonian physics, terms such as 'force' and 'power' were part of everyday language, and had a number of closely related meanings. It is not that Newton gave a precise definition to, found the one true meaning of, the everyday term 'force', but rather he appropriated that word for a specific sense for which he needed a technical term. We would like to identify, in as precise a way as possible, the range of meanings for the term 'function' in engineering discourse.

It is useful to start by saying what is common to all meanings: they all arise from the idea of a machine, system or a person *doing* something or having a property that is *intended* or *desired* by someone, or deemed as appropriate from someone's point of view. Thus, the ontology of function starts with the ontology of behaviour, but it is distinguished by the fact that some agent regards it as desirable or intends the behaviour. All the other terms – structure, beahviour, causal models – are neutral with respect to intent (except, of course, the intention of the modeller, who as usual might include some aspects and exclude others).

However, the intended behavior of what? One might at first blush think that what we care about is the behaviour of the device by itself. On the other hand, a device is used because someone desired that something desirable happen *outside* the device. Thus, a central meaning of function is *function as* (*desired*) *effect*. However, functions are also often described in terms of the device's properties or behaviour, without any explicit mention of what the device might help achieve in the world outside it. Thus, functions can be described from a *device-centric* or an *environment-centric* viewpoint, or even in a mixture of the two viewpoints. Let us consider an example, an *electric buzzer*, which we assume everyone is familiar with.

Functional Descriptions of an Electrical Buzzer
Supose we see a device with a small box, say *box1*, a *switch* on the face of *box1*, a wire that leads from *box1* to another box, say *box2*. We ask the owner or the designer what the function of the device is. Consider the following answers:

- *Buzzer-Function i*. Its function is to make a sound come from box2, when the switch in box1 is closed.
- *Buzzer-Function ii*. Its function is to make box2 fill location2 with sound, when box1 is placed at location1 and box2 at location2, and when the switch in box1 is pressed.
- *Buzzer-Function iii*. Its function is to provide a means by which a person at location1 may cause sound to be produced at another location.
- *Buzzer-Function iv*. Its function is to enable a visitor to a house inform the person inside that someone is at the door.

In (i), the desired behaviour is given entirely in terms of the device. The device behaviour is stated in terms of variables associated with specific structural elements. On the other hand, in (iii) and (iv), the desired beahviour is couched entirely in terms of elements external to the device. These functional descriptions could have been written before the device was invented. While (iii) and (iv) are both environment-centric, they differ from each other in the degree of abstraction. A completely different device, say one that flashes lights inside the house as a visitor comes to the door, would also satisfy the functional description in (iv).

In (ii), there is a mixture of both the device and environment. Location2, outside of the box, is to be filled with sound, but the initiating action is described in terms of a structural element of the device.

For our purposes, we can take (i) and (iii) as extreme points, and focus on the relationship between them. Cases such as (ii), which are mixtures, and such as (iv), which are further abstractions, can then be placed in the context of these two descriptions. All meanings of function are defined in terms of behavioural constraints: properties or behaviours we wish to be satisfied under certain conditions.

Behavioural Constraints

Let W be an object or an object configuration. A behavioural constraint is any constraint on the behaviours in W (behaviours defined in any of the senses defined earlier). Some common forms in which behavioural constraints might be expressed are given in the examples below. (In the following, C stands for conditions on the values of the variables of, and actions on, the objects in W, B for behaviours, and P(B) for predicates defined on behaviours.):

- Behavioural_constraint i. P(B) is a behavioural constraint. *Examples*: the value of output voltage is greater than 5 volts. The average value of the variable P over time is 6.5 psi.
- Behavioural_constraint ii. 'If C, P(B)' is a behavioural constraint. *Example*. if the input voltage is above 5, the output voltage is be a sinusoid. If the switch is pressed, the voltage goes up to 5 volts, and stays at that level.
- Behavioural_constraint iii. 'If C_1, then $P_1(B)$; then if C_2, then $P_2(B)$, . . . then if C_m, then $P_m(B)$, *Example*: if switch is pressed, the ATM flashes, 'insert card'. If card inserted, ATM flashes, 'Enter ID'. If ID is entered, ATM . . .

We say that W satisfies the behavioural constraint *F* or *F* is satisfied in W if the values of the relevant variables in W satisfy the predicates under the conditions specified in F. In the software engineering literature, Behavioural—constraint iii is at times called a *usage scenario* or just a *scenario* [5].

5.1.1. *Function as Effect*

The concept of *mode of deployment* is intended to capture the notion of how to use the object so that it produces the intended effect.

Mode of Deployment

The mode of deployment of an object (device) D in some world W, represented as M(D, W), is the specification of all the ways in causal interactions between D and the entities in W are instantiated. Such a specification might include one or more of the following:

(i) Structural relations between D and the entities in W. Example: *Locate* part p of device D at location L_1, *electrically-connect* terminal t of D to terminal *of electrical outlet*, . . .

(ii) Actions or action sequences by entities in W on D. Examples: (1) Push Switch of D. (2) Insert card into slot of D, when 'insert card' flashes on screen. (3) . . .

Mode of deployment describes an object's 'connections' to its environment, and the actions taken by the objects in the environment on the object. Sometimes the mode of deployment is indirectly represented by specifying the boundary conditions on the object's variables. Specifying the relations with the environment often makes use of the interface variables that we described earlier. Associated with a mode of deployment, M, is a set of causal relations that are instantiated by the relations and interactions specified in M.

Role

Let F be a set of behavioural constraints defined in some W. Let D be an object introduced into W, in a mode of deployment M(D,W). If D causes F to be satisfied in W, we say that D plays, performs or has the *role* F in W.

Role is a purely descriptive term, neutral with respect to intention, often used in physical science. *Examples*: the role of the moon in causing tides. The role of the meteors in causing craters. The heart plays the role of a pump. The role captures an aspect of the effect an object has on its environment.

Function as Role

Let F be a set of behavioural constraints that an agent, say A, desires or intends to be satisfied in some W. Let D be an object introduced into W, in a mode of deployment M(D,W). If D causes F to be satisfied in W, we say that D has, or performs, the *function* F in W. Often A is understood to refer to a designer or a user, and is not explicitly mentioned.

We can talk about the function of D – function as role or effect – even before D is designed, or we know anything about its structure.

Suppose a person finds a ledge to sit on after a tiring hike. In this case, not only is it the case that the ledge plays a *role* of a chair, but that it serves the *function* of a chair, because this role is intended by the person. This definition of function as role applies to devices explicitly designed for a function, for objects that are just used by someone for a certain function (such as the use of a ledge for sitting on), as well as to biological objects. 'The function of the heart is to pump blood'. Here evolution is generally taken to play the role of a designer.

Different functions may be attributed to an object under different modes of deployment. For instance, an electrical battery may deliver the function, 'Provide a *voltage of B volts between external electrical terminals* p_1 and p_2, under the mode of deployment, 'p_1,p_2 *electrically_connected* to *electrical terminals of battery*'. It may also provide the function '*Support paper*', under the mode of deployment, '*bottom_sur-*

face of battery in relation *Atop*(*Battery, paper*)'. In fact, in engineering, the same object is often used for multiple functions, precisely because they satisfy more than one mode of deployment at the same time with respect to other objects.

The idea of 'role' helps to unify intention-neutral scientific talk and intention-intensive device talk under the same framework. Role is the common link. *Functions are Roles + Intentions*. All predictive reasoning can stick to the role talk.

In later discussion we use the terms 'function as effect' and 'environment-centred functional description' interchangeably.

The definition, as given, of funtion as role permits one to attribute the function of, say, a radio to a battery if the battery is inserted into a batteryless radio that is already a part of W. To avoid this, the definition may specify that W is in some sense minimal, i.e., it only involves objects that are explicitly implicated in the specification F.

5.1.2. Device-Centred Functional Description

Let F be a set of behavioural constraints defined on, and satisfied by, an object D. If F is intended or desired by an agent A, then D has function F for A.

Defining a function in this device-centred way does not imply that the device achieves its function without any interaction with the external environment. Behavioural constraints often have conditions, e.g. C in behavioural constraints (ii). These conditions may be defined purely in device terms (such as '*Switch* state = *closed*'), but satisfying the condition may require action by an external object (such as 'action *push* on *switch*). Hence, even if the behavioural constraints are defined exclusively in device terms, their satisfaction might still hinge on appropriate interaction of the device with its environment.

Example
Let F_E be the following behavioural constraint:

> F_E: T_v, the temperature of a specific volume of space *Vol* in some world W $> T_{ambient}$, the ambient temperature. (That is, we wish to heat a given volume of air in a space.) (15)

Given this functional description, a designer might propose the device, *heater*, whose causal model is described in (12), with the following mode of deployment:

> M (*heater, volume of space*): Relation *Spatially_immersed* (*surface of resistor, Vol*). (See (7) for a description of this relation.) (16)

We will see in the next section how to determine if this device will deliver the function. A device-centred functional description for *heater*, corresponding to the above F_E, is:

> F_D: If *Switch* is *closed*, $T_S(t_0) > T_S(t)_0)$, where T_S is the temperature of the *surface* and t_0 is the initial time. (17)

5.2. Relation between Device-Centric and Environment-Centric Functional Descriptions

Suppose that we desire a device that could play the role F_E in some world E, and that there is a device D with a device-centric functional description F_D. How do we decide if D can play the role F_E in E. The following proposition shows how.

Proposition. Let F_E be a functional description in some world E, and let D be a device with a device-centric functional description F_D. Let M(D,E) be a mode of deployment. If F_E can be inferred from the model of D, F_D, model of E, and causal relations associated with M(D,E), then we can say that D can provide function F_E in E under M.

Example
Device *heater* with the mode of deployment described in (16). The designer can proceed along the following steps: (i) Using the model of *heater* in (11), show that, if *Switch* is *closed*, the *temperature* T_s rises above $T_S(t_0)$ (i.e. F_D is satisfied); and (ii) using the model of the relation in (7), show that if T_S is high and if the device is *Spatially-immersed* (*surface of resistor, Vol*) (the mode of deployment), then the temperature of *Vol* increases (i.e. F_E is satisfied).

Example
Consider a buzzer with F_D as in *Buzzer_function_i*, i.e. 'make a sound come from box2, when the switch in box1 is closed'. Consider the mode of deployment, 'locate box1 at door of house, locate box2 at wall inside the house, and let the visitor perform action, "Press Switch" '. Under this mode of deployment, we can infer from *Buzzer_function_i* the satisfaction of the functional description F_E in *Buzzer_function_iv*, 'enable a visitor to a house inform the person inside that someone is at the door'.

This inference calls on additional knowledge, such as:

- The visitor and person inside the house share the convention that a visitor presses the switch to announce his arrival.

- The person inside the house knows that hearing the sound means that someone pressed the switch.
- Sound produced at the component box2 of the buzzer travels throughout the inside of the house.

Often, the customer (or previous practice) partially solves the design problem and transforms an environment-centric specification into a device-centric functional description. Suppose we need F_E to be satisfied in some world, and that we decide on an interface for the device and a mode of deployment in terms of the interface. Now, using the interface and deployment specifications, we might identify an F_D such that $F_D \rightarrow F_E$. Using the buzzer example again, suppose we decide that the device will have a switch that will be pressed by the visitor and a sound-producing element that will be placed inside the house. Thus, we have specified both a structural interface and a mode of deployment. An F_D along the lines of *buzzer_function i* can be shown to support the inference of F_E under the assumptions. We can now give the designer F_D as the functional specification, a specification that is entirely device-centric.

Thus, moving from an environment-centric functional description towards to device-centric description calls for partially solving the design problem, specifically to the extent of identifying the interfaces and the mode of deployment. The designer can the focus his design and the verification of his design to F_D, since the mapping from F_D to F_E has already been done.

In many cases, the interface design stage may only be partially accomplished, and correspondingly, the mode of deployment is only partially specified. In those cases, the functional specification handed to the designer will be a mixture of device-centric and environment-centric terms. Suppose we decide on the following partial interface design for the buzzer: 'The device will have an electrical switch', and a partial specification of the mode of deployment, 'The visitor will act by pressing the switch'. In this case, the best we can do in moving from F_E as given in *buzzer_function_iv* is the following F: 'When the switch is pressed, the person inside the house should get the information that the switch is pressed'. F is part device-centric ('switch is pressed') and part environment-centric ('person at house should know'). The designer has more freedom with this functional description e.g. he can design something that flashes a light inside the house.

The design of an ATM (Automated Teller Machine) provides an example of why functional descriptions may usefully combine both device-level and environmental-level terms. In the sense of function as effect, the function of an ATM is to enable a bank customer to be able to withdraw money from his account. This can even be expanded to, 'enable a bank customer to withdraw money from his account, provided the bank is able to check the identity of the customer, and that he has an adequate balance in his account'. All the terms here, *customer*, *bank*, *account*, *balance*, etc., relate to the world outside the ATM, and thus could provide the basis for a functional specification. However, before the electro-mechanical design proceeds, the bank (or the industry or the design firm itself) might decide on an interaction sequence between the customer and the machine, based partly on the fact that customers already have a machine-readable bank-card. The suggested interaction sequence might be: customer inserts card, the system verifies that the individual is a customer of bank, the customer inputs desired cash for withdrawal, the system checks that there is adequate balance, and if balance is adequate, gives him the cash. The most natural way of describing this functional requirement is to couch it as a sequence of device- and environment-centric conditions and behaviours.

5.3. From Needs to Functions: Levels of Abstraction

Use or design of any artifact is triggered by a *need* or a *desire* felt by some human in some context. People need to stay warm when it turns cold. People need easy access to water in their homes. People need or desire cash from their bank accounts when the bank is closed or too far. People desire that their plants be watered when they soil gets dry. And so on. In what follows, we will use the single term 'need' to stand for both 'need' and 'desire'.

The ontology of needs and goals is the same as that of functional descriptions; both are represented as desired behavioural constraints in some universe: 'Someone desires to be warm, if it is colder than 70 degrees', 'Someone desires water to be available in the house whenever he needs it'. These describe what kinds of behaviours are expected under what conditions. Thus, one might think that needs directly become the functional descriptions that are then handed to a designer for design, or become the basis for acquiring a device. As it turns out, needs often undergo a sequence of transformations before they become the specifications for an artifact.

First, the need has to be recognised as something to be satisfied, and some human has to set up a *goal* or have a *purpose* to satisfy the need. The goal is a representation in the cognitive structure that initiates, drives and mediates problem-solving activities towards the achievement of the goal. The goal can be created in the mind of the human with the need, or it can be created in the mind of a designer, who is solving the problem of helping a generic person with a need.

Secondly, problem solving produces a sequence of transformations of the need such that objects or objects configurations, and means of interacting with them, can be identified so that the need can be satisfied. Consider the need someone might have to possess some merchandise found in a shop. Let us call this need, N_1. The person realises that he needs cash to buy this, and doesn't have it. So N_1 creates another need, say N_2, which is to have cash. One way to have cash is to withdraw money from his bank account, let us call this need N_3. The functional specifications of ATM match N_3 exactly. By using the ATM, he meets need N_3, which meets need N_2, which helps meet need N_1. While we feel comfortable thinking of N_3 as the ATM's function, we don't feel that it is appropriate to say that the function of ATM is N_1, i.e. to help buy merchandise. However, the ATM in the appropriate mode of deployment did result in the functional requirement N_1 to be satisfied.

To generalise the issue, we often use device D with an associated (environment-centric) function F for achieving some need N. When is it appropriate to associate N also as a function of D?

Consider N_1, the need to own a specific item in a shop. There are many ways to satisfy this need: buy it, get it as a present, steal it, and so on. There are at least three ways to buy: paying cash, paying by check and charging it. There are many ways to have cash, borrowing it and withdraw money from account (if the account has sufficient balance) being two. In this tree of alternatives starting from N_1, only one, withdrawing money from one's account, was associated with the ATM as its function in our account. There are basically only two ways of getting cash from one's bank account in the US: go to the bank when it is open and stand in line; and use an ATM. Out of the two, only the ATM is a device. In general, in human assignment of functions, the following heuristic seems to come into play: a device is to be assigned a function at the lowest level of abstraction in the chain of needs it can satisfy. Consider the chain: N_1, N_2 . . . N_k, where N_i stands for a way in which to achieve N_{i-1}. Suppose a device can in fact be used to satisfy any of the needs in the sequence. The heuristic suggests assigning N_k as the device function. A device that accepts cash or credit card to dispense the shirt would in fact be assigned the function of enabling buying shirts, since whatever the need a new shirt would satisfy, shirt-buying is the need that the device satisfies, stated at lowest level of abstraction.

Of course, this is only a heuristic. A chair, which is normally assigned the function of supporting a human in a certain position, can sometimes be used, in the absence of something that is more appropriate, to break windows in a room in case of an emergency. The chair satisfies the need that a piece of glass be broken, but we would not assign this as a function of the chair. A chair is certainly not the best tool for window breaking, and, not incidentally, it is not normally used for this purpose. (A designer's task is to come up with a device that is tuned to achieve the desired function. Thus, while the device may also achieve other functions, there may well be other devices that are better suited for the other functions.) Functions get assigned to devices because they help in retrieval from memory or from a device library. It is not effective to associate with objects functions that are theoretically possible, but for which the objects are not the best suited. So, another heuristic is that objects are not assigned functions for which they may be used but only inefficiently (and thus rarely). A hammer would be the best tool for breaking windows, but a hammer is not assigned window breaking as a function either. However, this is explained by the previous heuristic, that a device be assigned a function at the lowest level of generality. Thus, a hammer has the function of exerting great force focused on a small area. Given this functional description, it can be used to hit nails as well as break windows.

In summary, humans have needs or desires, which in turn results in goals or purposes, which are representations in their cognitive architectures, to help achieve the constraints defining the need or desire. The need/desire/goal/purpose specifications undergo a number of abstraction transformations. At the end of these transformations, there is a specification of needs that corresponds to the function of a device in the sense of function as effect. If such a device and a mode of deployment exist with this function as effect, this pair can be the basis for a solution to the problem initiated by the need or desire. If not, then the function as effect becomes the charge to a designer.

5.4. Function Specification for the Design Task

The Design Task

Let W be an environment and let F be a set of behavioural constraints defined for W. Let a designer have a goal to have F be true in W. This sets up the functional specification for the design task: to specify an object D, and specify a mode of deployment M(D,W), such that under M, D causes F to be satisfied in W. We can say that D has the function F under mode of deployment M.

Traditional definitions of the design task focus on the need to specify the object, e.g. to provide a list of components from some component library and a way of composing them. Our definition additionally requires that a way of embedding D in W and interactions between D and W be specified; the design task is not complete until the designer specifies this mode of deployment. The mode of deployment makes the connection between the properties and structure of D and the achievement of F in W. The requirement that the designer specify the mode of deployment is equivalent to the requirement that he tell the user how to use the device to meet the need for which it was designed.

If the function that is given to the designer is completely device-centred, the requirement about the mode of deployment no longer applies, since the translation from the original need to a device-centred description entails the specification of the mode of deployment. If the functional specifications are a mixture of device- and environment-centred descriptions, then it is only necessary for the designer to specify the mode of deployment partially, for those aspects of functions that are environment-centric.

Design Function Verification

How does the designer or the customer verify (short of constructing the artifact as per proposed design and seeing if it does the job) that a design will deliver the function? First, we need to show that the models of the components, the relations specified in object composition, and M(D,W) together imply the functional specification F. We also need to show that W alone does not imply F. The second step ensures that D is really needed, i.e. it really does play a causal role in delivering the function. There is usually an additional implicit requirement, that in some sense, D is the simplest such device that will deliver the function. This is to avoid Rube Goldberg-like devices as satisfactory solutions to a design problem.

6. Explanation of How a Device Works

Suppose a device D consists of components C_1, $C_2, \ldots C_n$ in relation $R(C_1, C_2, \ldots C_n)$. Let us say that D delivers a function F (be it device-centred or environment-centred). There are different ways of explaining *how* the device works, i.e. how the device delivers its function.

(i) *Derive F from the composed causal model of D* if F is device-centric, or from the model of D, M(D,E) and E, where E is the environment, if F is environment-centric. That is, we show that the causal model of D implies F (device-centric), or that the models of D, E and M(D,E) together imply F (environment-centric).

Example

Consider the device *heater*, illustrated in Fig. 7 and described in (12), in the mode of deployment described by the relation in (16). A way to explain *how* it works is to show that the causal model of the device constructed as described in (9) implies its F_D, described in (17); or, show that the model of D and the relationship (16) together imply F_E, given in (15). It is straightforward to show that these implications follow.

(ii) *Show the causal dependency structure of the function on the variables of the components*. In this form of explanation, we display how the behaviours in F causally depend upon the properties of the components or the relations in D, and if F is environment-centric, also on M(D,E) and E. The general way to show this is by use of a technique known as causal ordering [6,7]. Causal ordering identifies which variables causally precede which other variables, and in what order. This form of explanation is often helpful in identifying which other variables will be affected as we contemplate a change in the value of some variable.

Example

For the device described in (16), T_v is the relevant variable that is implicated in F_E. A causal ordering of this variable with respect to the component level variables will be as in Fig. 8.

(iii) *Annotated causal process descriptions*. The quality of the explanation in (ii) can often be

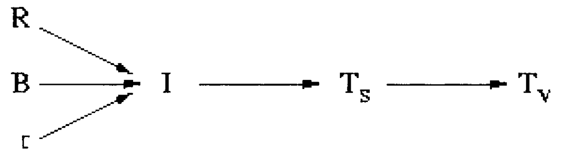
$$\begin{tikzcd} R \arrow[dr] & & & \\ B \arrow[r] & I \arrow[r] & T_S \arrow[r] & T_V \\ r \arrow[ur] & & & \end{tikzcd}$$

Fig. 8. A causal dependency diagram that shows how the variables in *electrical-circuit* are related. The current I is determined by the values of the two resistances R and r as well as the battery voltage B. T_s depends upon the current I. T_v depends on T_s. Such a diagram is part of explaining how *heater* works.

enhanced by annotating the causal dependencies in various ways. One way to annotate is to have the precise functional relation between the variables on the left and right sides of a dependency arrow. This will help to calculate changes in the dependent variables as independent variables change. Another form of annotation is to identify the function or functions of components that support the causal transition. Additional such annotations include structural connections that play a role in the transition, and physical laws that govern the values of the dependent variable. In the buzzer example, a piece of causal dependency diagram might be that 'Switch state being closed causes current to flow through the circuit which in turn causes a magnetic field around the coil'. The first causal transition might be explained by annotating it as due to structural connections between the components and the voltage providing function of the battery, and the second transition might be annotated as being due to the magnetic function of the component coil. The advantage of such annotation is that if there is a malfunction, we can systematically check with link in the dependency diagram is not working, and for such a link, the annotation suggests component malfunctions or structural failures. This style of explanation was used extensively in the FR work summarised in Chandrasekaran [4].

7. Discussion

The goal of this paper has been to explore the meanings of certain basic terms in engineering practice, specifically the term 'function'. The motivation is provided by the prospect of automating more and more of the reasoning processes in the practice of engineering, and the consequent need for representing engineering knowledge in precise ways. We started with a relatively simple ontology based on the CML work in AI, and used that framework to define the terms, ending with a range of meanings of the terms of interest. A central meaning is *function as effect*, emphasising that the reason we wish to use a device at all is so that certain desired things may come to pass in the world external to the device. Then, we account for a device-centric use of the term, and finally, for a use that mixes device-centric and environment-centric descriptions. Using the set of definitions, we were able to clarify the tasks of design and design verification. Discussions on function over the years often take the form of 'This is what function *really* means'. Instead, we have proposed that it is a concept that has a range of meanings, and what is important is not what it *really* means – because there is no one meaning – but to develop a framework in which what one means by such terms in a certain context may be made clear.

We also referred in passing to how the function of a device may be explained in terms of the properties of the components and the structure. This topic requires a more complete treatment in a subsequent paper. Another topic that we only referred to in passing is that of building component libraries in which the components are indexed by function. Thus, a designer, looking for a component that might deliver a required function, may be able to retrieve candidate components. Representing the function precisely is a prerequisite for this capability.

We have also discussed how design – and functional specifications for design – arise from attempts to meet needs in the world. Needs and functional requirements are similar in the sense that both state desired conditions on behaviours, but needs undergo several stages of reformulation before they become statements about functions for devices. We discussed the steps involved in such reformulations.

In our own earlier work on function – the work that has come to be called Functional Representation (FR) [4,8–10] – function was defined largely in a device-centric sense. The emphasis in that work was on explanation of how the function of a device arises from those of the components and they are configured. The representational framework of the current work can be used to restate in more precise terms the ideas in the FR approach. The FR work makes clear the usefulness of explicitly representing functions of devices and their relations to component functions for the tasks of diagnosis, design and explanation. In later work [11], we developed the framework for function as role. The current paper synthesises both points of view.

There has been a substantial amount of recent research on representing function. We have reviewed them elswhere [4]. The notion of function as effect, one of the meanings discussed in the current paper, is close to the ideas contained elsewhere [12–14]. All these authors, in their discussion on function, focus their attention on what a device does in the external world. What we add here, as we do to the FR work, is precision in definitions by exploiting a simple ontological framework.

The ontology that we have used for modelling the world was quite simple: objects, properties and state variables, causal relations between the variables in an object, and inter-object causal interactions. These by no means exhaust the terms needed to describe the causal phenomena in the world. It is easy to come up with terms that we have not considered, but will surely be needed for a fuller description, terms such as field, events, unindividuated things such as liquids and gas, flow and so on. However, the goal in the paper is not an ontology for describing the world, but rather to clarify certain terms used in engineering discourse, especially function. We believe that the essential intuitions about these concepts will survive an enriched ontology.

We have focused entirely on the meaning of function in general, not on specific functions. There is an orthogonal body of work that focuses on what we might call *content theories* of functions, i.e. on proposals about specific functions in specific domains or classes of domains. For example, the flow-container-conduit domain work in Chittaro et al. [15] and Kumar and Upadhyaya [16], the substance-flow work in Goel [17], and the work on meta-functions [18] describe very general roles that are common to different domains in which the notion of flow is important. The work in the current paper can help the research in this direction to be more precise about the specific content proposals for functions in various domains.

In another direction, researchers have attempted various typologies of functions. Distinctions between functions that *Achieve* vs. *Prevent* vs. *Maintain* are identified in Keuneke [19], the function type *Allow* was added to this taxonomy in Iwasaki and Chandrasekaran [8]. Again, the work in this paper can help such taxonomies to be stated more precisely. The *Achieve*, *Prevent*, *Maintain* and *Allow* functions may be device-centric or stated in terms of constraints on behaviours on the environment.

Acknowledgements

The research reported in this paper was supported by DARPA under two contracts: one contract no. F30602-93-C-0243 monitored by USAF Material Command, Rome Laboratories, DARPA order no. A714, and by grant no. N00014-96-1-0701, DARPA order no D594, monitored by Office of Naval Research. We are thankful for the useful comments of the reviewers and of Hermann Kaindl, and to the editors of this Special Issue.

References

1. Chandrasekaran, B. (1990) Design problem solving: a task analysis. AI Magazine, 11, 4, 59–71
2. Falkenhainer, B., Farquhar A., Bobrow D., Fikes R., Forbus K., Gruber T., Iwasaki Y., Kuipers B. (1994) CML: A Compositional Modeling Language. Technical Report, Stanford University Knowledge Systems Laboratory
3. Carroll, M. (2000) A Brief Introduction to OSU-CML. Final Report submitted to Office of Naval Research, Grant No. N00014–96–1–0701, Functional and Diagrammatic Representation for Device Libraries
4. Chandrasekaran, B. (1994) Functional representation and causal processes. In: Yovits, M. C. (ed.), Advances in Computers, pp 73–143. Academic Press
5. Kaindl, H. (2000) A design process based on a model combining scenarios with goals and functions. IEEE Trans. SMC Part A: Systems and Humans, SMC, 30(5), September 2000, 537–551
6. Simon, H. A. (1953) Causal ordering and identifiability. In: Koopman, T., Hood, W. (eds), Studies in Econometric Methods. Wiley
7. Iwasaki, Y., Simon, H. A. (1986) Causality in device behavior. Artificial Intelligence, 29; (1) 3–32
8. Iwasaki, Y., Chandrasekaran B. (1992) Design verification through function-and behavior-oriented representation: Bridging the gap between function and behavior. In: Gero, J. S. (ed.), Artificial Intelligence in Design, 597–616
9. Iwasaki, Y., Fikes, R., Vescovi, M., Chandrasekaran, B. (1993) How things are intended to work: Capturing functional knowledge in device design. Proc. 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann pp. 1516–1522
10. Vescovi, M., Iwasaki Y., Fikes R., Chandrasekaran, B. (1993) CFRL: A language for specifying the causal functionality of engineered devices. 11th National Conference on AI, pp 626–637, AAAI Press/MIT Press
11. Chandrasekaran, B., Josephson, J. R. (1997) Representing function as effect. In: Modarres, M. (ed.), Proc. Functional Modeling Workshop, Paris, France
12. Umeda, Y., Takeda H., Tomiyama T., Yoshikawa H. (1990) Function behavior and structure. AIENG'90, Applications of AI in Eng., Computational Mechanics Publications and Springer Verlag, 177–193
13. Sasajima, M., Kitamura Y., Ikeda M., Mizoguchi, R. (1995) FBRL: A function and behavior representation language. International Joint Conf on Artificial Intelligence, 2, Morgan Kaufmann pp. 1830–1836.
14. Larsson, J. E. (1996) Diagnosis based on explicit means-end models. Artificial Intelligence, 80(1), 29–93
15. Chittaro, L., Tasso C., Toppano, E. (1994) Putting functional knowledge on firmer ground. Int. J. Applied Artificial Intelligence, 8, 239–258
16. Kumar, A. N., Upadhyaya, S. J. (1995) Function-based discrimination using model-based diagnosis. Int. J. Applied Artificial Intelligence, 9, 65–80
17. Goel, A. K. (1989) Integration of cased-based reasoning and model-based reasoning for adaptive design problem solving. PhD Thesis, Ohio State University
18. Kitamura, Y., Mizoguchi R. (1999) Meta-functions of artifacts. 13th International Workshop on Qualitative Reasoning (QR-99), pp 136–145
19. Keuneke, A. (1991) Device representation: The significance of functional knowledge. IEEE Expert, 6, 22–25