CFRL: A Language for Specifying the Causal Functionality of Engineered Devices

Marcos Vescovi **Yumi Iwasaki** **Richard Fikes**
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg C
Palo Alto, CA 94304
vescovi,iwasaki,fikes@ksl.stanford.edu

B. Chandrasekaran
Laboratory for AI Research
The Ohio State University
217 B, Bolz Hall, 2036 Neil Avenue
Columbus, OH 43210-1277
chandra@cis.ohio-state.edu

Abstract*

Understanding the design of an engineered device requires both knowledge of the general physical principles that determine the behavior of the device and knowledge of what the device is intended to do (i.e., its functional specification). However, the majority of work in model-based reasoning about device behavior has focused on modeling a device in terms of general physical principles or intended functionality, but not both. In order to use both functional and behavioral knowledge in understanding a device design, it is crucial that the functional knowledge is represented in such a way that it has a clear interpretation in terms of actual behavior. We propose a new formalism for representing device functions with well-defined semantics in terms of actual behavior. We call the language CFRL (Causal Functional Representation Language). CFRL allows the specification of conditions that a behavior must satisfy, such as occurrence of a temporal sequence of expected events and causal relations among the events and the behavior of device components. We have used CFRL as the basis for a functional verification program which determines whether a behavior achieves an intended function.

Introduction

Understanding the design of an engineered device requires both knowledge of the general physical principles that determine the behavior of the device and knowledge of what the device is intended to do (i.e., its functional specification). However, the majority of work in model-based reasoning about device behavior has focused on modeling a device in terms of general physical principles *or* intended functionality, but not both. For example, most of the work in qualitative physics has been concerned with predicting the behavior of a device given its physical structure and knowledge of general physical principles. In that work, great importance has been placed on preventing a pre-conceived notion of an intended function of the device from influencing the system's reasoning methods and representation of physical principles in order to guarantee a high level of "objective truth" in the predicted behavior. In contrast, in their work based on the FR (Functional Representation) language (Sembugamoorthy & Chandrasekaran 1986) (Keuneke 1986), Chandrasekaran and his colleagues have focused mostly on modeling a device in terms of what the device is intended to do and how those intentions are to be accomplished through causal interactions among components of the device.

Both types of knowledge, functional and behavioral, seem to be indispensable in fully understanding a device design. On the one hand, knowledge of intended function alone does not enable one to reason about what a device might do when it is placed in an unexpected condition or to infer the behavior of an unfamiliar device from its structure. On the other hand, knowledge of device structure and general physical principles may allow one to predict how the device will behave under a given condition, but without knowledge of the intended functions, it is impossible to determine if the predicted behavior is a desirable one, or what aspect of the behavior is significant.

In order to use both functional and behavioral knowledge in understanding a device design, it is crucial that the functional knowledge is represented in such a way that it has a clear interpretation in terms of actual behavior. Suppose, for example, that the function of a charge current controller is to prevent damage to a battery by cutting off the charge current when the battery is fully charged. To be able to determine whether this function is actually accomplished by an observed behavior of the device, the representation of the function must specify conditions that can be evaluated against the behavior. Such conditions might include occurrence of a temporal sequence of expected events and causal relations among the events and the components. Without a clear semantics given to a representation of functions in terms of actual behavior, it would be impossible to evaluate a design based on its predicted behavior and intended functions.

While it is important for a functional specification to have a clear interpretation in terms of actual behavior, it is also desirable for the language for specifying functions to be independent of any particular system used for simulation. Though there are a number of alternative

* The research by the first three authors is supported in part by the Advanced Research Projects Agency, ARPA Order 8607, monitored by NASA Ames Research Center under grant NAG 2-581, and by NASA Ames Research Center under grant NCC 2-537. Chandrasekaran's research is supported by the Advanced Research Projects Agency by means of AFOSR contract F-49620-89-C-0110 and AFOSR grant 89-0250.

methods for predicting behavior, such as numerical simulation with discrete time steps or qualitative simulation, a functional specification at some abstract level should be intuitively understandable without specifying a particular simulation mechanism. If a functional specification language was dependent on a specific simulation language or mechanism, a separate functional specification language would be needed for each different simulation language, which is clearly undesirable. What is needed is a functional specification language that has sufficient expressive power to support descriptions of the desired functions of a variety of devices. At the same time, the language should be clear enough so that for each simulation mechanism used, it can be given an unambiguous interpretation in terms of a simulated behavior.

An essential element in the description of a function is causality. In order to say that a device has achieved a function, which may be expressed as a condition on the state of the world, one must show not only that the condition is satisfied but also that the device has participated in the causal process that has brought about the condition. For example, when an engineer designs a thermostat to keep room temperature constant, the design embodies her idea about how the device is to work. In fact, the essential part of her knowledge of its function is the expected causal chain of events in which it will take part in achieving the goal. Thus, a representation formalism of functions must provide a means of expressing knowledge about such causal processes.

We have developed a new representational formalism for representing device functions called CFRL (Causal Functional Representation Language) that allows functions to be expressed in terms of expected causal chains of events. We have also provided the language with a well-defined semantics in terms of the type of behavior representation widely used in model-based, qualitative simulation. Finally, we have used CFRL as the basis for a functional verification program which determines whether a behavior achieves an intended function.

This paper is organized as follows: We first describe the representation of behavior over time in terms of which the semantics of CFRL will be defined and our assumptions about the modeling and simulation schemes that produce such a behavior description. We then present the CFRL language and define its semantics in terms of behavior. We close with a discussion and summary.

Behavior Representation

Before describing CFRL, we briefly describe the behavior representation in terms of which the semantics of CFRL will be defined. A physical situation is modeled as a collection of *model fragments*, each of which represents a physical object or a conceptually distinct physical phenomenon, such as a particular aspect of component behavior or a physical process. A model fragment representing a phenomenon specifies a set of conditions under which the phenomenon occurs and a set of consequences of the phenomenon. The conditions specify a set of instances of object classes that must exist and a set of relations that must hold among those objects and their attributes for the phenomenon to occur. The consequences specify the functional relations the phenomenon will cause to hold among the objects and their attributes.

Model fragments can represent phenomena as occurring continuously while the fragment's conditions hold or as events that occur instantaneously when the conditions become true. The consequences of a model fragment that represents an event are facts to be asserted resulting from the event, whereas the consequences of a model fragment that represents a continuous process are sentences (e.g., ordinary differential equations) which are true while the phenomena is occurring.

When there exists at time t a set of objects represented by model fragments m_i to m_j that satisfy the conditions of a model fragment m_0, we say that an instance of m_0 is active at that time. We will call m_i through m_j the *participants* of the m_0 instance.

Representation of physical knowledge in terms of model fragments is a generalization of the representation of physical processes and individuals in Qualitative Process Theory (Forbus 1984). There are several systems, including the Device Modeling Environment (DME) (Iwasaki & Low 1991) the Qualitative Process Engine (QPE) (Forbus 1989), and the Qualitative Process Compiler (QPC) (Crawford, Farquhar & Kuipers), that use similar representations for physical knowledge to predict the behavior of physical devices over time. Though the ways these systems actually perform prediction differ, the basic idea behind all of them is the following: For a given situation, the system identifies active model fragment instances by evaluating their conditions. The active instances give rise to equations representing the functional relations that must hold among variables as a consequence of the phenomena taking place. The equations are then used to determine the next state into which the device must move.

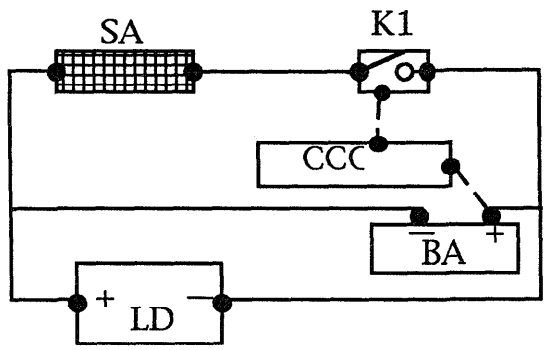
We assume that a behavior is a linear sequence of states. The output of a qualitative simulation system such as QPE, DME, and QPC is usually a tree or a graph of states. Each path through the graph represents a possible behavior over time. We will refer to such a path, i.e., a linear sequence of states, as a *trajectory*.

A state represents a situation in which the physical system being modeled is in at a particular time. "A particular time" here can be a time point or interval. We will not assume any specific model of time in this paper. The only assumptions about time that we make are: (1) the times associated with different states do not overlap; (2) when a state s_j immediately follows s_i in a behavior, there is no other "time" that falls between the times (periods) associated with s_i and s_j; and (3) every state has a unique successor (predecessor) unless it is the final (initial) state, in which case it has none.

In our modeling scheme, each state has a set of variable values and predicates that hold in the state. In addition, each state has a set of active model fragment instances representing the phenomena that are occurring in the state.

An Electrical Power System

This section presents the device that we will use throughout the rest of this paper as an example. The device is the electrical power system (EPS) aboard an Earth orbiting satellite (Lockheed 1984). A simplified schematic diagram of the EPS is shown in Figure 1. The main purpose of the EPS is to supply a constant source of electricity to the satellite's other subsystems. The solar array generates electricity when the satellite is in the sun, supplying power to the load and recharging the battery. The battery is a constant voltage source when it is charged between 6 and 30 ampere-hours. When the charge level is below 6 ampere-hours, the voltage output decreases as the battery discharges. When the charge level is above 30 ampere-hours, the voltage output increases as it is charged.



SA: Solar array
LD: Electrical load on board
BA: Rechargeable battery
CCC: Charge current controller
K1: Relay

Figure 1: An Electrical Power System.

Since the battery can be damaged when it is charged beyond its capacity, the charge current controller opens the relay when the voltage exceeds a threshold to prevent the battery from being over-charged. The controller senses the voltage via a sensor connected to the positive terminal of the battery. When the voltage is greater than 33.8 volts, the controller turns on the relay K1. When the relay is energized, it opens and breaks the electrical connection to prevent further charging of the battery, thereby switching the current source for the load from the solar array to the battery. When the relay is open or when an eclipse period begins, the battery's charge-level starts to decrease. When the battery becomes under-charged, the voltage decreases. When it reaches 31.0 volts, the CCC turns relay K1 off to close it.

CFRL

We now describe the syntax and semantics of CFRL. Figures 2 shows an example of the representation of a function of the EPS.

```
DF:  ?eps: Electrical-power-system
CF:  Object-set: ?sun: Sun    ?l: electrical-load
     Conditions: T
GF:
  (ALWAYS
    (AND
      (-> (AND (Shining-p ?sun)
               (Closed-p (Relay-component ?eps)))
          CPD1)
      (-> (OR (NOT (Shining-p ?sun))
              (Open-p (Relay-component ?eps)))
          CPD2)
      (-> (AND  (> (Electromotive-force
                     (Battery-component ?eps))
                   33.8)
                (Closed-p (Relay-component ?eps)))
          CPD3)
      (-> (AND (< (Electromotive-force
                    (Battery-component ?eps))
                  31.0)
               (Open-p (Relay-component ?eps)))
          CPD4)))
```

Figure 2-a: Function F_1 of EPS

We consider a function to be an agent's belief about how an *object* is to be used in some *context* to achieve some *effect*. Thus, our representation of a function specifies the object, the context, and the effect. However, it does not specify an agent, which is implicitly assumed to be whoever is using the representation. Formally, a function is defined as follows:

Definition 1: Function

A function F is a triplet $\{D_F, C_F, G_F\}$, where:

D_F denotes the device of which F is a function.

C_F denotes the context in which the device is to function.

G_F denotes the functional goal to be achieved.

The device specification, D_F, specifies the class of the device and the symbol by which the device will be referred to in the rest of the definition of F. The example in Figure 2-a states that the function is of an Electrical-power-system which will be referred to as *?eps* in the rest of the definition.

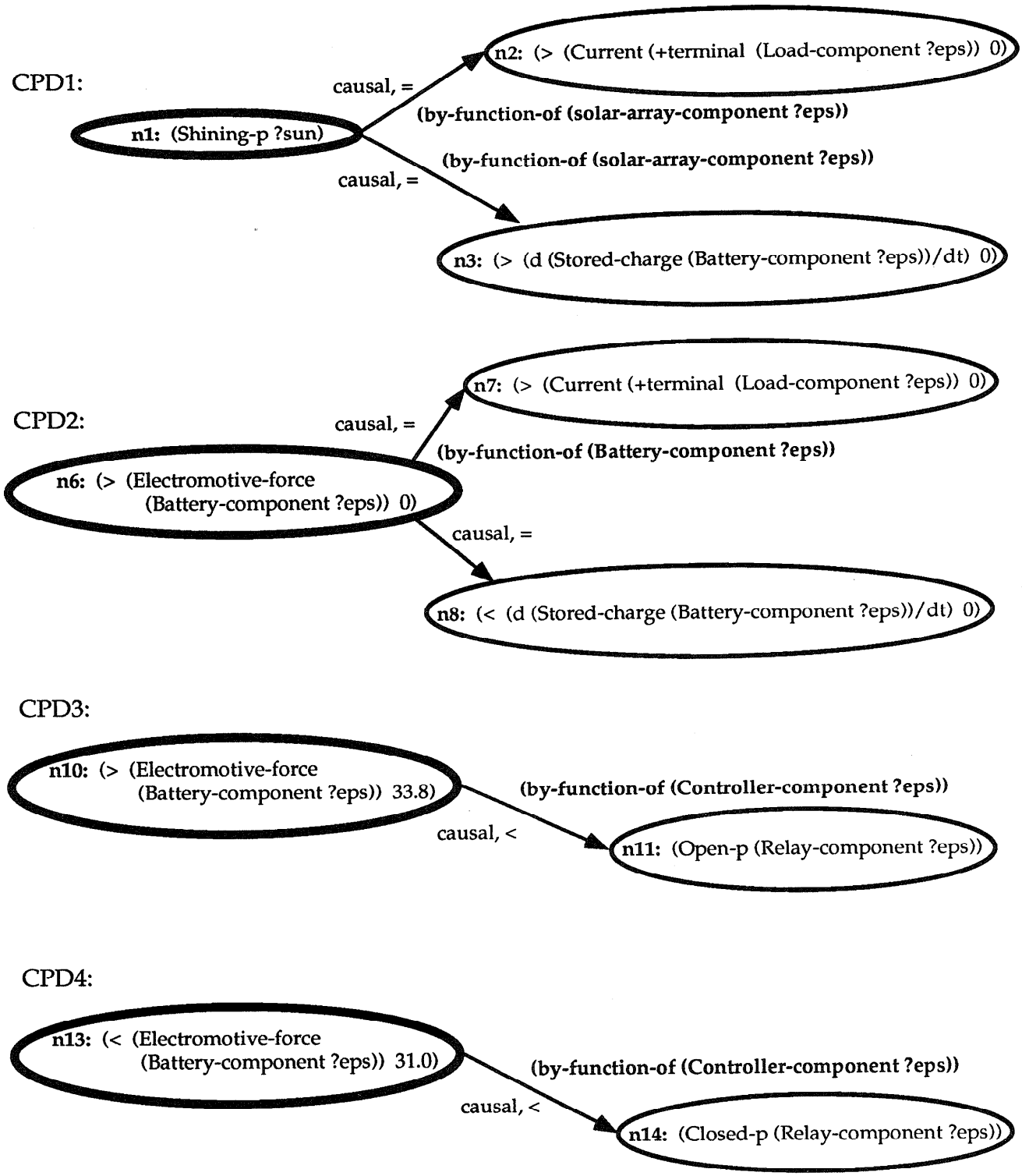


Figure 2-b: CPD's of Function F_1 of EPS.

The notion of a device function assumes some physical context in which the device is placed, and C_F is a specification of such a context. C_F consists of two parts, a set of objects and a set of conditions on those objects. For example, Figure 2-a states that there must exist an instance of Sun and an instance of electrical load. The conditions must hold throughout a behavior in order for the function to be verified in the behavior.

Formally, the *Object-set* of a C_F is a list of pairs *{var, type}*, where *var* is a symbol to be used in the description of F to refer to the object, and *type* is the type (class) of the object. *Conditions* is a logical expression involving the variables defined in the *Object-set* and D_F.

The third part of the function definition, G_F, specifies the behavior to be achieved by the device used in a specific manner. G_F of a function is represented as a Boolean combination of *Causal Process Descriptions* (CPDs) and conditions involving the variables defined in D_F and the *Object-set* of C_F. Each CPD is an abstract description of expected behavior in terms of a causal sequence of events. In the following, we formally define a CPD.

Causal Process Descriptions (CPD's)

Figure 2-b shows examples of CPD's which are part of the functional specification of the EPS. A CPD is a directed graph, in which each node describes a state and each arc describes a temporal and (optionally) a causal relation between states.

A node specifies a condition on a state. The condition is a logical sentence about the state of the world at some time using the variables defined in the D_F and C_F portions of the function. For example, the node n_1 in Figure 2-b states the condition that the sun be shining. One or more nodes in each CPD are distinguished as the initial node(s). In the figures, the initial nodes are indicated with a thick oval. A condition specified by a node can contain AND and OR as logical connectives. When the meaning is clear, we will use the name of a node to refer to the condition represented by the node.

The arcs in a CPD are directed and specify temporal and causal relations among nodes. An arc has the following attributes:

source: The node at the tail of the arc.

destination: The node at the head of the arc.

causal-flag: An indicator of whether the relationship between the states described by the source and destination nodes is causal. (The relationship is always temporal.)

temporal-relation: =, <, or ≤, indicating the temporal relation between the states described by the source and destination nodes. = means that the states described by the two nodes are to be the same state, < means the state described by the source node must strictly precede the state described by the destination node, and ≤ means the state described by the source node must either be the same as or precede the state described by the destination state.

causal-justification: If an arc is "causal", one can attach a justification for the causal relation. A justification takes the form of a Boolean combination of the following predicates:

(by-function-of <model-fragment>),
(with-participation-of <model-fragment>).

The meaning of these predicates will be explained after we give a precise definition of a *causal relation* among nodes.

In order to refer to attributes of arcs, we will use the attribute name (e.g., source, destination, etc.) as a function of the arc as in "*source*(a_1)".

We will write $n_i \Rightarrow_c n_j$ when there is a causal arc from n_i to n_j. As a condition specified by a node can be a Boolean combination of conditions, the following defines the meaning of causal relations among them, where e_1, e_2, and e_3 are conditions:

a) $(\text{AND } e_1\ e_2) \Rightarrow_c e_3 \equiv (\text{AND } (e_1 \Rightarrow_c e_3)\ (e_2 \Rightarrow_c e_3))$

b) $e_1 \Rightarrow_c (\text{AND } e_2\ e_3) \equiv (\text{AND } (e_1 \Rightarrow_c e_2)(e_1 \Rightarrow_c e_3))$

c) $(\text{OR } e_1\ e_2) \Rightarrow_c e_3 \equiv (\text{OR } (e_1 \Rightarrow_c \text{e3})\ (e_2 \Rightarrow_c e_3))$

d) $e_1 \Rightarrow_c (\text{OR } e_2\ e_3) \equiv (\text{OR } (e_1 \Rightarrow_c e_2)\ (e_1 \Rightarrow_c e_3))$

Semantics of a CPD

A CPD can be considered to be an abstract specification of a behavior. Unlike a trajectory, it does not specify every state or everything known about each state. It only specifies some of the facts that should be true during the course of the behavior and partial temporal/causal orderings among those facts. The intuitive meaning of a CPD is that:

- For each node in the CPD, there must be a state in the trajectory in which the condition specified by the node is satisfied, and
- For each pair of nodes directly connected by an arc, the causal and temporal relationships specified by the arc must exist in the trajectory.

In order for us to evaluate these conditions against a behavior, we must define their meanings in terms of the languages used to describe a (simulated or actual) behavior. In this paper, we will do so in terms of the behavior representation formalism described earlier.

However, note that CFRL itself is independent of the particular behavior representation language used, and that one would need to provide different definitions in order to evaluate functional specifications in CFRL against behaviors generated by a different scheme.

We first present the definition of a *causal dependency* relation between sentences in a trajectory and the *causality constraints* that can be associated with a CPD arc. We then define the requirements for a trajectory to match a CPD and for a trajectory to match a function goal. Finally, we use those definitions to define the requirements for a trajectory to achieve a function.

A few words about notation: We will attach *[s]* to a sentence to denote the sentence holds in state *s*. Therefore, *p[s]* means that *p* holds in state *s*. We will also associate a state with models and variables to denote sentences as follows:

m[s] : An instance of model fragment *m* is active in *s* .

v[s] : The value of variable *v* in *s*. (i.e., an axiom of the form *(= (value v s) c)* for some constant *c*.)

We will use the relations $<$, $>$, $=$, and $\leq$ to express temporal ordering among states in a trajectory. For example, for states $s1$ and $s2$ in a trajectory, "$s1 < s2$" means that $s1$ strictly precedes $s2$ in time. Note that ordering is total for states in a trajectory because a trajectory is a linear sequence of states, while the ordering is partial for states in a CPD.

Intuitively, we say $p2$ is *causally dependent* on $p1$ in trajectory *Tr*, written $p1 \Rightarrow p2$, when it can be shown that $p1$ being true in *Tr* eventually leads to $p2$ being true in *Tr*.

Definition 2: Causal Dependency

The causal dependency relation, $\Rightarrow$, is a binary relation between sentences in a trajectory with the following properties:

1. For all atomic sentences *p*, states *s*, model fragments *m*, and variables *v*:

   a) If $p[s0], p[s1], \ldots p[s]$ (i.e., if *p* is part of the initial conditions and is never changed), then $\phi \Rightarrow p[s]$. (And we say that *p[s]* is exogenous.)

   b) If model fragment *m* represents an event and asserts *p*, and if there exists a state s_j such that $s_j < s$, $\neg p[s_j]$, $m[s_j]$, and $p[s_k]$ for all $k > j$ (i.e., *p* became true at some point before *s* due to *m*), then $m[s_j] \Rightarrow p[s]$.

   c) If model fragment *m* represents a continuous process and has *p* as a consequence, and if there exists a state s_j such that $s_j < s$, $\neg p[s_j]$, $m[s_j]$, and $p[s_k]$ for all $k > j$ (i.e., *p* became true at some point before *s* due to *m*), then $m[s_j] \Rightarrow p[s]$.

   d) If model fragment *m* has *p* as a condition, then $p[s] \Rightarrow m[s]$.

   e) If *v* occurs in *p* as a term and *p* is not *v[s]*, then $v[s] \Rightarrow p[s]$.

   f) If *v* is an exogenous variable, $\phi \Rightarrow v[s]$.

   g) For all variables *v'* such that *v'* -> *v* is in the causal ordering[1] in *s* :

      (i) v'[s] $\Rightarrow$ v[s] ;

      (ii) If *p[s]* is the equation through which *v* depends on *v'*, then $p[s] \Rightarrow v[s]$.

   h) For all variables *v'* such that *v* and *v'* are in a feedback loop in the causal ordering in *s*:

      (i) v'[s] $\Rightarrow$ v[s] and v[s] $\Rightarrow$ v'[s] ;

      (ii) For each equation *p* such that *p* is part of the feedback loop and *v* appears in *p*, $p[s] \Rightarrow v[s]$.

   i) If $s1$ is the state immediately following *s*, and *dv* is the time-derivative of *v* in *s*, then $dv[s] \Rightarrow v[s1]$.

2. $\Rightarrow$ is transitive.

When $p_i \Rightarrow p_j$, we will say that p_j is *causally dependent on* p_i or that p_i *causes* p_j. Given statements $p[s_i]$ and $p[s_j]$ such that $p[s_i] \Rightarrow p[s_j]$, we call the causal sequence of statements starting from $p[s_i]$ and leading to $p[s_j]$ the *causal path* from $p[s_i]$ to $p[s_j]$.

Having defined the meaning of a causal relation among statements, we can now explain the meaning of the predicates used to justify causal arcs in a CPD.

Definition 3: Causality constraints

Given an arc *a* from node n_i to n_j in a CPD and a model fragment *m*, causality constraints of the following form can be associated with *a* :

a) ***(by-function-of m)*** -- meaning that the causal path from n_i to n_j includes a consequence of an instance of *m*;

b) ***(with-participation-of m)*** -- meaning that the causal path from n_i to n_j includes a consequence of an instance of a model fragment in which an instance of *m* participates.

These predicates do not imply specific commitments as to *how* the components participate in the causal process. They give the designer the capability of using whatever component has the desired function, independent of its particular mechanism.

We can now present the definitions on which verification of a trajectory with respect to a CPD is based.

Definition 4: Matching of a state and a node

A state *s* in a trajectory and a node *n* in a CPD are said to *match* if the condition specified in *n* is true in *s*.

Having defined the meaning of a causal relation among statements in a trajectory, we can now define the meaning

[1] Causal ordering is a technique for determining causal dependency relations among variables in a set of equations (Iwasaki & Simon 1986).

of the causal and temporal relations between linked nodes of a CPD.

Definition 5: Satisfying the constraints of an arc

If a is an arc from node n_i to n_j in a CPD, then the causal and temporal constraints of a are satisfied at states s_i and s_j if both of the following conditions are satisfied:

a) s_i < (= or ≤) s_j when n_i < (= or ≤) n_j , respectively.

b) If arc a is causal and if n_i and/or n_j are Boolean combinations of conditions, then the causal relation between n_i and n_j can be rewritten as a Boolean combination of causal relations of the form $e_i \Rightarrow_C e_j$, where e_i and e_j are atomic conditions. $e_i[s_i] \Rightarrow_C e_j[s_j]$ is satisfied if for every variable[2] v_i used in e_i and every variable v_j used in e_j, $v_i[s_i] \Rightarrow v_j[s_j]$ and the causal path from $v_i[s_i]$ to $v_j[s_j]$ satisfies the causal justification on a.

Definition 6: Matching of a CPD and a trajectory

Let T be a trajectory consisting of a linear sequence of m states, s_1 through s_m. Let CPD_1 be a CPD consisting of a set of nodes, N_1, and a set of arcs, A_1. CPD_1 and T are said to match iff all the following conditions are satisfied:

a) The initial nodes of CPD_1 match the initial state s_1 in T.

b) For each remaining node n in N_1, there exists a state in T that matches n such that for every arc a in A_1 from nodes n_i to n_j, the temporal and causal constraints specified by a are satisfied by the states matched to n_i and n_j.

Representation of the Functional Goal (G_F)

The functional goal of a function (denoted by G_F) is represented as an expression consisting of CPDs, conditions, quantifiers, and Boolean connectives. Nested expressions using connectives are allowed, but a quantifier cannot appear in the scope of another quantifier. Each CPD must appear in the scope of one and only one quantifier. There are two quantifiers, ALWAYS and SOMETIMES. Connectives are AND, OR, IMPLIES, and NOT. Syntactically, the connectives are used in the same way as ordinary logical connectives. The following are example G_F expressions:

(ALWAYS (AND *cpd1 cpd2* (OR *cpd3 cpd4*)))
(OR (ALWAYS *cpd1*)
    (SOMETIMES (AND *cpd2 cpd3*)))
(ALWAYS (NOT *cpd1*))

[2] The variables used in CFRL can be different from the variables in terms of which the trajectory states are defined, since CFRL descriptions represent a device-level perspective, while states in the trajectory represent a component or physical process-level perspective. Correspondences between CPD variables and trajectory variables are made when the function is matched against a specific trajectory.

Quantifiers align the initial nodes of the CPDs in their scope as well as specify whether the described behavior must hold in every subsequence of the trajectory or only in some of them. The connectives and quantifiers are to be interpreted as specified in the following definition of matching a G_F and a trajectory.

Definition 7: Matching of a G_F and a trajectory

Let T be a trajectory consisting of a linear sequence of m states, s_1 through s_m; T_i denote subsequences of T from s_i through s_m; and <cpd-exp> denote a Boolean combination of CPD's and conditions. Then:

a) *(ALWAYS <cpd-exp>)* matches T iff *<cpd-exp>* matches T_i for each T_i (i = 1 to m).

b) *(SOMETIMES <cpd-exp>)* matches T iff *<cpd-exp>* matches T_i for some T_i (i = 1 to m).

c) *(AND <cpd-exp0> <cpd-exp1> ...)* matches T iff every conjunct matches T.

d) *(OR <cpd-exp0> <cpd-exp1> ...)* matches T iff at least one of the disjuncts matches T.

e) *(NOT <cpd-exp>)* matches T iff *<cpd-exp>* does not match T.

f) *(IMPLIES <cpd-exp0> <cpd-exp1>)* matches T iff *<cpd-exp0>* does not match T or *<cpd-exp1>* does match T.

g) Condition c matches T iff c is true in the initial state of T.

Finally, we complete the definition of the meaning of a function, as follows:

Definition 8: A trajectory achieving a function

A trajectory T achieves a function F when the condition specified in C_F holds throughout T and G_F matches T.

Discussion and Summary

In this paper, we have presented CFRL, a language for specifying an expected function of a device and defined its semantics in terms of the type of behavior representation widely used in model-based qualitative simulation. The language allows one to explicitly state the physical context in which the function is to be achieved and to describe the function as an expected causal sequence of events. Since the concept of causal interactions among components is essential to the understanding of a function, the language allows explicit representation of causal interactions and constraints on such interactions.

CFRL is based on the work on Functional Representation (Sembugamoorthy & Chandrasekaran 1986), and it is a further extension of the work presented in (Iwasaki & Chandrasekaran 1992). We have extended the expressive power of the function specification languages

described in those papers and have provided a formal foundation for the semantics of the resulting language.

Franke (Franke 1991) also proposed matching design intent with simulated behavior. Unlike other work on functional representation, he focuses on representing the purpose of a design *modification* and not that of a device itself. He developed a representation scheme, called TED, in which he expresses the purpose for making a modification in a structure. TED's representation of a function can be a sequence (not necessarily a linear) of partial descriptions, which is matched against states in a sequence of qualitative states generated by QSIM. To prove that a function is achieved by a modification, he compares the behavior of the original structure and that of the modified structure.

Bradshaw and Young (Bradshaw & Young 1991) also represent the intended function in a manner similar to Functional Representation. They built a system called DORIS, which uses the knowledge generated by qualitative simulation for evaluating device behavior as well as for diagnosis and explanation.

The most important characteristic that distinguishes our work from those by Franke and by Bradshaw and Young is the central role causal knowledge plays in CFRL. We conjecture that causal relations are an essential part of functional knowledge, and that representation of functional knowledge must allow explicit description of the causal processes involved. Furthermore, verification of a function must ascertain that the expected causal chain of events take place, since the satisfaction of the functional goal alone does not necessarily indicate that the device is functioning as intended.

Because the semantics of CFRL is defined in terms of matching between a behavior and a functional specification, the language is immediately useful for the purpose of behavior verification. We have designed and implemented an algorithm that verifies a behavior produced by the DME system with respect to a function specified in CFRL as defined in this paper. Initial testing of the algorithm has included verifying the functional specifications of the EPS as given above. Care must be taken in designing such an algorithm to assure that exponential search is not required to find a match between a trajectory and a CPD. We are currently in the process of analyzing the computational complexity of the problem and our algorithm.

We expect formal functional specifications to have many uses throughout the life cycle of a device (Iwasaki et al. 1993). For example, in the early stages of the design process, designers often do "top down" design by incrementally introducing assumptions about device structure and causality relationships. Such design evolution could be expressed as incremental refinements of a CFRL functional specification. DME could assist a designer in this functional refinement process by assuring that each successive specification is indeed a refinement of its predecessor so that any device that satisfies the refinement also satisfies the predecessor.

References

Bradshaw J.A.; and Young R.M. 1991. Evaluating Design Using Knowledge of Purpose and Knowledge of Structure. *IEEE Expert* April.

Crawford J.; Farquhar A.; and Kuipers B. 1990. QPC : A Compiler from Physical Models to Qualitative Differential Equations. In *Proceedings of the Eight National Conference on Artificial Intelligence.*

Forbus K.D. 1984. Qualitative Process Theory. *Artificial Intelligence* 24.

Forbus, K. D. 1989. The Qualitative Process Engine. In *Readings in Qualitative Reasoning about Physical Systems.* Weld, D. S., and de Kleer, J. Eds. Morgan Kaufmann.

Franke D.W. 1991. Deriving and Using Descriptions of Purpose. *IEEE Expert* April.

Iwasaki Y.; and Simon H.A. 1986. Causality in device behavior. *Artificial Intelligence* 29:3-32.

Iwasaki Y.; and Low C.M. 1991. Model Generation and Simulation of Device Behavior with Continuous and Discrete Change. Technical Report, KSL, Dept. of Computer Science, Stanford University.

Iwasaki Y.; and Chandrasekaran B. 1992. Design Verification through Function and Behavior-Oriented Representations : Bridging the gap between Function and Behavior. In *Proceedings of the Second International Conference on Artificial Intelligence in Design*, Pittsburgh.

Iwasaki Y.; Fikes R.; Vescovi M.; and Chandrasekaran B. 1993. How Things are Intended to Work : Capturing Functional Knowledge in Device Design. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.*

Keuneke A. 1989. Machine Understanding of Devices; Causal Explanation of Diagnostic Conclusions. Ph.D. thesis, Laboratory for AI Research, Dept. of Computer & Information Science, The Ohio State University.

Lockheed Missiles and Space Company. 1984. SMM Systems Procedure for Electrical Power Subsystem. doc #D889545A, SE-23, Vol. 3.

Sembugamoorthy V.; and Chandrasekaran B. 1986. Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In Kolodner J.L. and Riesbeck C.K. (editors), *Experience, Memory and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ.