Design Verification Using Functional Knowledge

Y. IWASAKI
Knowledge Systems Laboratory
Stanford University
701 Welch Rd.,
Palo Alto, CA 94304

B. CHANDRASEKARAN
Laboratory for AI Research
Dept. of Computer & Information Science
The Ohio State University
217 B, Bolz Hall
2036 Neil Avenue
Columbus, OH 43210-1277

Abstract. This paper focuses on the task of design verification using both knowledge of the structure of a device and its intended functions. In particular, it addresses the question of when one can say a behavior predicted by a prediction system achieves the desired function in the manner intended by the designer. We use Functional Representation (Sembugamoorthy & Chandrasekaran 1986) to represent the function of a device and the expected causal mechanism for achieving it. We present a formal definition of matching between a system trajectory generated by a simulation system and the description of a causal process to achieve a function expressed in Functional Representation. We believe that evaluating a behavior with respect to the expected causal process as well as the function improves the chances of uncovering hidden flaws in a design that may otherwise go undetected at an early stage.

1. Introduction

Simulation of the behavior of the design of a structure is an important means for design evaluation, which must ascertain that the design achieves the intended function. To achieve a robust modeling and simulation capability, a system must be able to compose a simulation model from pieces each of which may be applicable to a variety of situations. At the same time, in order to provide a useful feedback about the design of a device based on the result of simulation, the system must be able to evaluate the predicted behavior with respect to the knowledge of the function.

In this paper, we focus on the task of design verification using both knowledge of the structure of a device and its intended functions. In particular, we address the question of when one can say a behavior predicted by a prediction system achieves the desired function in the manner intended by the designer. We use Functional Representation (Sembugamoorthy & Chandrasekaran 1986) to represent the function of a device and the expected causal process for achieving the function.

Research in model-based reasoning about physical systems has emphasized representation of structures and reasoning about behavior from the knowledge of their structures and physical principles. Several model-based reasoning systems have been built (Falkenhainer & Forbus 1991, Crawford et al. 1990, Iwasaki & Low 1992) that formulates a model of a device based on its structure and predict its behavior. An important requirement in the approach taken in these systems, which we shall call the *behavior-oriented* approach, is that the knowledge is stored in small pieces, each representing a conceptually independent physical phenomenon such as a physical process or an aspect of the behavior of a component. For the pieces to be composable, each of them must be defined in a context-independent manner as much as possible in the sense that there is no unstated assumption about the surroundings of a component or the function of the whole device. These systems predict a behavior in terms of a sequence or a graph of states, each of which is characterized by the set of applicable knowledge pieces, implied constraints, and variable values.

This type of model-based reasoning capability is useful for a system aimed to help in design, since it allows the system to formulate a behavior model automatically and to simulate its behavior so that the designer can discover behavioral implications of design decisions easily. However, an account of behavior in the form of a sequence of states must be evaluated to be useful for further development of the design. Does the predicted behavior achieve the desired function? Does it do so in the way the designer intended? These are crucial questions in providing a useful feedback to the designer. In order for a model-based reasoning system to answer such questions, it must have knowledge of the function of the device -- WHAT it is supposed to do -- and the expected behavior -- HOW it is supposed to achieve the function.

Functional Representation (FR) is a representational scheme for the functions and expected behavior of a device. FR represents knowledge about devices in terms

of the functions that the entire device is supposed to achieve and also of the sequence of causal interactions among components that lead to achievement of the functions. FR takes a top-down approach to representing a device in contrast to the bottom-up approach of behavior-oriented knowledge representation and reasoning schemes. In Functional Representation, the function of the overall device is described first and the behavior of each component is described in terms of how it contributes to the function, while in a behavior-oriented approach, the behavior of the entire device is inferred from those of individual components.

In order to evaluate a design, one must be able to predict the possible behavior of the design, as well as to determine whether the predicted behavior achieves the expected functionality. Verification that a behavior of a designed artifact achieves the desired goal must ascertain the following:

(1) the overall function of the device is achieved,
(2) the expected chain of events happen in the predicted behavior, and
(3) the causal connections expected between events exist in the predicted behavior.

The purpose of this paper is to investigate this concept of behavior verification and provide a formal definition of behavior verification of a design with respect to its intended functions and the expected causal processes for achieving the functions. As an example of a model-based reasoning system, we use DME (Device Modeling Environment) developed at Stanford University (Iwasaki & Low 1992). Given the topological description of a device and initial conditions, DME formulates a mathematical model and simulates its behavior.

In DME, knowledge about physical phenomena is organized into *model fragments* in the knowledge base. Each model fragment represents knowledge of a conceptually distinct physical phenomenon such as a physical process, component behavior characteristics, etc. DME takes an input description of the initial state, including the topological model of the device, and searches the knowledge base for model fragments that are applicable to the given situation. Equations to describe the behavior of the device are formulated from the constraints associated with the set of model fragments thus found. The equations are used to predict the behavior of the device numerically or qualitatively using QSIM (Kuipers 1986). During prediction, if there are any changes in the set of applicable model fragments, the set of equations is updated accordingly and prediction continues with the new equation model.

Some model fragments represent instantaneous changes, which are phenomena that take place too quickly to model as continuous phenomena. Such model fragments do not have constraints but they have consequences, which are facts to be asserted. When an instantaneous model fragment becomes active, a new state is generated immediately to follow the current state, and the consequences are asserted in the new state. A model fragment m can be interpreted as an implication of the form $P_m \Rightarrow E_m$ or $P_m \Rightarrow R_m$, where P_m, E_m, and R_m denote respectively the conditions for the applicability of the model fragment, the behavior constraints (equations in the case of a continuous phenomenon), and the consequences of m (in the case of a discontinuous phenomenon).

Definition 1. A device state is represented as a set of state variables $\{V_S\}$ consisting of values of all the variables of interest in the description of the device. State variables can be either continuous or discrete.

Definition 2. A device trajectory, T_r, represents the behavior of the device over time. It is a linear sequence of states.

2. What does it mean to verify that a design achieves an expected function?

In this section, we define what it means for such a simulated behavior to achieve the expected behavior represented in FR. This requires introduction of the notions of a *causal process description* (CPD) and a *function* in FR. A CPD is a causal explanation of how certain states of interest come about by exhibiting a sequence of causal transitions. The transitions are annotated by different types of causal explanation.

Definition 3. A Causal Process Description (CPD) is defined as a pair $\{C, G\}$, where C is the applicability condition and G is a directed graph $G = \{N, L\}$. C specifies the condition under which the device is expected to behave as specified by G. C is a necessary condition for applicability of CPD but not a sufficient condition. N is a set of nodes and L is a set of directed links among nodes. Each node represents a partial description of a state. There are two distinguished nodes in N, the initial node, N_{init}, and the final node, N_{fin}. Each link represents a causal connection between nodes. The graph may be cyclic, but there must be a directed path from N_{init} to N_{fin}.

A link may have an attached *qualifier*, *By-function-*

<f>-of(c), where *c* is a component, to indicate the conditions under which the transition will take place. A link can also have *annotations* of the types, *Provide(p)*, *If(p)* and *Trigger(p)*, where *p* is a wff, to indicate the type of the causal explanation to account for the transition.

In order for us to be able to relate a *Tr* and a CPD, we require that *each node in a CPD must be given a definition in the form of a wff about objects and predicates defined in terms of model fragments attributes.* We let *def(n)* denote such a definition of a node *n*. For example, the node "Battery charging" is defined as $dC/dt > 0$, where *C* is the variable, charge-level of the battery. With such a definition, a node in a CPD becomes a partial description of a state in *Tr* using attributes defined in the model fragment library.

Definition 3 mentions qualifiers and annotations that can be attached to the links between the nodes in CPD. The full list of proposed annotations can be found in (Sembugamoorthy & Chandrasekaran 1986) and (Keuneke 1991). For a link from n_i to n_j, an annotation *By-function-<f>-of(c)* means the causal interactions going from n_i to n_j must involve *c* achieving its junction. The purpose of a qualification is to allow a causal transition to be explained in further detail by CPD's of component *c*. In contrast, qualifiers allows one to specify further conditions on the causal transition. *Provided(p)* means that condition *p* must hold during the causal transition. *If(p)* means that the condition *p* must hold at state n_i. *Trigger(p)* means that *p* must not hold before n_i, but must hold at some point after n_i (inclusive).

In summary, a CPD describes a causal process from some perspective at the device level, and the conditions on the causal transitions and explanations of them are given as part of the description. Figure 1 shows an example of a CPD. It is for the electrical power system (EPS) aboard a satellite. N_{init} and N_{fin} in each CPD are indicated by a box in dashed lines and a box in thick lines respectively.

Definition 4: A function *F* is defined as a quintuple $\{Type_F, P_F, Dev_F, C_F, G_F\}$, where

- $Type_F$: One of {ToMake, ToMaintain, ToPrevent, ToControl}.
- P_F: The functional goal, i.e. the wff that the function is to make true.
- Dev_F: The device that this function is a function of. It must be a model fragment in the DME's knowledge base.
- C_F: The condition which specifies when the function must be achieved.
- G_F: The set of CPD's describing the causal mechanism to achieve the function.

We consider four types of functions; *ToMake*, *ToMaintain*, *ToPrevent*, and *ToControl* (Keuneke 1991). Note that a device can have multiple functions, in which case each function will be represented separately. In case of a function of type *ToControl*, P_F must be of the following form:

$(= v_0\ f(v_1 \ldots v_n))$, where v_i's are variables and *f* is some function of its arguments. Following is the function of EPS:

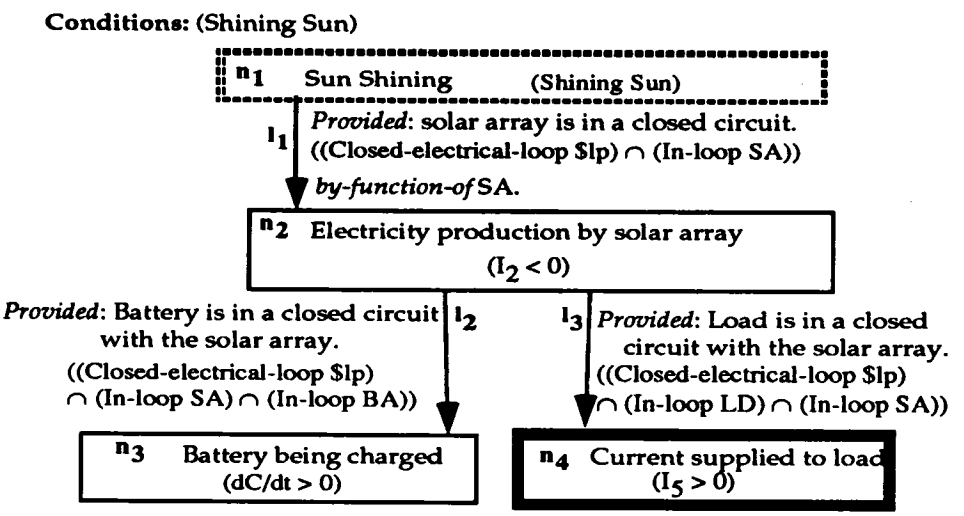

Figure 1: CPD_1 of EPS

EPS Function

$Type_F$: *ToMaintain*	P_F: (Powered Load)
C_F : T Dev_F: EPS	G_F: CPD_1, CPD_2

Given a device description and initial conditions, we can generate a Tr. Suppose we also have an intended function for the device and associated CPD's. Intuitively, we would like to say that the device achieves the function in Tr if (1) the functional goal is achieved, (2) there are states in Tr matching all the nodes in the CPD in the specified temporal order, and (3) for each causal link in the CPD, there is a causal path in Tr that connects the cause to the effect. In order to make these conditions more precise, we must define the concept of a causal path in a trajectory. Then, we will define what it means for a trajectory to match a CPD.

For the rest of this paper, we use the following notation: We will attach $[s]$ to wff's, model fragments and variable to denote the axiom that asserts the truth of the wff, the activity of the model fragment, or the value of the variable. For example, we will write $p[s]$ to denote that p holds in the state s, $m[s]$ to denote that the phenomenon represented by m is active in s , and $v[s]$ that asserts the value of v holds in s. We will use notations such as $<$, $>$, $\leq$, and $\geq$ to express ordering among nodes in a CPD and states in a trajectory. We write "$n_1 < n_2$" where n_1 and n_2 are nodes in a CPD to indicate that n_1 is strictly causally upstream of n_2. For states s_1 and s_2 in a trajectory, "$s_1 < s_2$" means that s_1 strictly precedes s_2 in time. Note that the ordering is partial for nodes because a node can have multiple incoming and outgoing nodes. Ordering is total for states because a trajectory is a linear sequence of states.

2.1 CAUSAL DEPENDENCY IN A TRAJECTORY

We now present the definition of a causal dependency relation between axioms p_1 and p_2 in a trajectory, Tr. Intuitively, we say p_2 is causally dependent on p_1, written "$p_1 \Rightarrow_C p_2$", when it can be shown in Tr that p_1 being true eventually leads to p_2 being true in Tr. The formal definition makes use of the concept of causal ordering (Iwasaki & Simon 1986). The theory of causal ordering defines the causal dependency relations among variables in a self-contained set of equations. We will write $v_1 \rightarrow_C v_2$ when v_1 depends on v_2 according to the definition of causal ordering.

Definition 5. The causal dependency relation, denoted $a_i \Rightarrow_C a_j$, is defined between two wff's, a_i and a_j, in the descriptions of states in a trajectory Tr. We write "$a_i \Rightarrow_C a_j$" and say "a_j *depends on* a_i" or "a_i *causes* a_j." The relation $\Rightarrow_C$ is transitive.

The following conditions specify when a wff can be said to be causally dependent on another in Tr:

(a) If $p[s_0], p[s_1] \ldots p[s]$ for all states from s_0 up to s, (in other words, p was part of the initial conditions and never changed), we say that $p[s]$ is exogenous, and write $\phi \Rightarrow_C p[s]$.

(b) If there exists a state $s_j < s$ such that, $\sim p[s_j]$, and $p[s_{j+1}]$, where s_{j+1} is the immediate successor of s_j in Tr, and there exists $m[s_j]$, such that $p \in R_m$, and $p[s_i]$ for all s_i between s_{j+1} and s inclusive (in other words, p becomes true at some point before s as a consequence of the phenomenon represented by m.), we say $m[s_j] \Rightarrow_C p[s]$.

(c) For each $p \in P_m$, we say $p[s] \Rightarrow_C m[s]$. In other words, for each phenomenon active in s, we say that the phenomenon being active is dependent on its precondition being satisfied.

(d) If $v_1 \rightarrow_C v_2$ according to the definition of causal ordering, we say $v_1[s] \Rightarrow_C v_2[s]$.

(e) For each equation e in $D(v)$ for a variable v and a phenomenon m such that $e \in E_m$, we say $m[s] \Rightarrow_C v_2[s]$. In other words, we say v depends on the phenomenon giving rise to the causal relation between v and whatever other variables v depends on.

(f) $v_2[s] \Rightarrow_C v_1[s]$ and $v_1[s] \Rightarrow_C v_2[s]$ if $v_2 \leftrightarrow_C v_1$ in the causal ordering in s.

(g) $v'[s_1] \Rightarrow_C v[s_2]$, where s_2 is the state immediately following s_1, and v' the time-derivative of v in s_1.

2.2 WHEN IS A FUNCTION ACHIEVED?

We listed in Definition 4 different types of functions of devices and components. In this subsection, we spell out the conditions under which a device is said to achieve each type of function in a trajectory.

Definition 6: Let s_z denote the final state in Tr, and Dev denote either Dev_F or one of its components. A function F is said to be achieved in a trajectory Tr in any of the following cases depending on $Type_F$. In all the cases C_F must hold in the initial state of Tr. In the following,

Case 1: When $Type_F$ = *ToMake*, F is achieved by Tr if

1.1 P_F, the functional goal, holds in the final state s_z. We denote this by $P_F[s_z]$. And,

1.2 There is some device variable v, and some state s in Tr, such that $v(s) \Rightarrow_C P_F[s_z]$ *(i.e.* this fact causally depends on the operation of the device).

Case 2:When $Type_F$ = *ToMaintain*, F is achieved in Tr if

in all states s_i in Tr , the following is true: For some s_j such that $s_j \leq s_i$ in Tr,

2.1 $P_F[s_i]$, and

2.2 There is some device variable v such that $v[s_j] \Rightarrow_C P_F[s_i]$.

Case 3: When $Type_F = ToControl$, F is achieved in Tr if

3.1 $v_0[s_Z] = f(v_1[s_Z], \ldots v_n[s_Z])$ *(i.e.* the functional relation holds between the value of the controlled variable and the values of the controlling variables in the final state),

3.2 $v_i[s_Z] \Rightarrow_C v_0[s_Z]$ for $1 \leq i \leq n$ *(i.e.* the value of the controlled variable in the final state causally depends on the controlling variables), and

3.3 There is some device variable v and some state s in Tr, such that $v(s) \Rightarrow_C v_0[s_Z]$

Case 4: When $Type_F = ToPrevent$, F is achieved in Tr if $\sim P_F[s]$ for any state s in Tr (*i.e.* F is achieved in Tr if the functional goal of F does not hold in any state). We make the closed world assumption that $\sim p$ unless p is explicitly known to hold.

Definition 7: A trajectory Tr is said to match a CPD if there is a mapping st from nodes in the CPD to the states in Tr that satisfies the following conditions:

1. for each node n in the CPD there is a state $st(n)$ in Tr where $def(n)$ holds, and
2. for any nodes n_1 and n_2 in the CPD, $st(n_1) \leq st(n_2)$ iff $n_1 < n_2$, and
3. for each causal link l from n_1 to n_2, there is a causal path $def(n_1)[st(n_1)] \Rightarrow_C def(n_2)[st(n_2)]$ in Tr, where $def(n)[st(n)]$ denotes that $def(n)$ holds in the state $st(n)$. Furthermore, if l has an attached qualifier, $Provided(p)$, p must hold for all states between $st(n_1)$ and $st(n_2)$ inclusive. If l has an attached annotation, *By-function-of*, which points to a component o, there must be a causal path $o[s] \Rightarrow_C def(n_2)[st(n_2)]$ for some state s such that $st(n_1) \leq s \leq st(n_2)$.

Clause 1 of the above definition ensures that for each node in the CPD, there is a state in Tr that matches it. Clause 2 makes sure that the temporal ordering of causes and effects in the CPD is preserved in the temporal ordering of their corresponding states in Tr. Finally, Clause 3 ensures that the causal paths exist in Tr that correspond to the causal links in the CPD. We are now ready to state precisely what it means to verify that a predicted behavior achieves the expected behavior.

Definition 8: We say that a trajectory Tr of a device achieves the expected behavior with respect to a function F when the following conditions are met. Tr_i denotes the subsequence of Tr from the initial state up to and including state s_i:

1. F is satisfied in Tr according to Definition 6, and
2. F is achieved in the expected manner, which is verified as

Case 1: if $Type_F$ is not *ToMaintain*, Tr matches one of the CPD's of F according to Definition 7,

Case 2: if $Type_F$ is *ToMaintain*, for each state s in Tr, a match between Tr_i and one of the CPD's exists such that $s_i = st(N_{fin})$ for the final node N_{fin} of the CPD.

Clause 1 of this definition makes sure that the function is achieved in the trajectory. Clauses 2 ensures that the function is achieved in the way the designer intended.

Using above definitions, one can prove whether or not a given function of a device is achieved in a behavior predicted by a simulator such as DME. A detailed example of such a proof is found elsewhere (Iwasaki & Chandrasekaran 1992).

3. Discussion

In this paper, we formalized a number of notions that were relatively informally specified in the Functional Representation language and defined matching between an expected behavior represented in Functional Representation and a predicted behavior. Our primary goal is to use the knowledge of functions and expected behavior for the purpose of design verification. It is important that the definition of behavior verification we have presented is not biased towards any particular perspective about what are more important than others as a causal factor. In other words, it does not require that the function or the expected behavior be described from a particular point of view. This definition of verification of a predicted behavior with respect to a function and an expected behavior is inclusive enough to allow a trajectory to match many representations of functions or expected behaviors. Likewise, there can be any number of trajectories that can be shown to match a given expected behavior as there can be any number of designs that accomplish the same functionality. Thus, the mapping between trajectories and an expected behavior is many to many. However, if the goal is to verify that a predicted behavior achieves a given expected behavior, this non-uniqueness of a match is not a problem. Our definition does not establish that the given expected behavior is the only correct causal story for a given trajectory, nor that the trajectory is the only correct way to achieve the function. However, the definition does

establish that a given design achieves the function in an expected manner, which is what is needed for our purpose of design verification.

We are in the process of implementing a program that takes a functional representation and a trajectory and automatically proves whether or not the expected behavior is realized in the trajectory.

3.1 RELATED WORK

Bradshaw and Young (1991) and Franke (1991) have also proposed representations of the knowledge of a purpose and their use in design. They represent the intended function in a manner that is similar to the way functions are represented in Functional Representation. Bradshaw and Young built a system called Doris, which uses knowledge of purpose for evaluating behaviors generated by qualitative simulation as well as for diagnosis and explanation.

The focus of Franke's work on representing functions is slightly different from ours or Bradshaw and Young in that he represents the purpose of a design modification and not that of a whole device. He developed a representation scheme, called TED, in which he expresses the purpose for making a modification δ in a structure using the same function types as those in Functional Representation. Thus, in order to prove that a function is achieved by a modification δ, he must compare the behavior of structure M and that of M', which is M with the modification δ. Another important characteristic of TED's representation of functions is that it can be a sequence (not necessarily a linear) of partial descriptions. The representation of a function in TED typically says "δ guarantees σ," where σ is a sequence, called scenario, of partial descriptions. The sequence of partial descriptions is matched against states in a sequence of qualitative states generated by QSIM.

The most important difference between our work and the works by Franke's or by Bradshaw and Young's is that we take not only the functions but also the causal interactions into account in evaluating behavior. We feel that it is important to test whether it is in fact the causal processes intended by the designer that are responsible for bringing about the achievement of the functional goal, since the satisfaction of the functional goal does not necessarily indicate that the design is functioning as intended. We believe that evaluating a trajectory with respect to the causal process as well as the function allows one to uncover hidden flaws in a design which may otherwise go undetected.

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 into Qualitative Differential Equations. *Proceedings of the Eighth National Conference on Artificial Intelligence.*

Falkenhainer, B. and Forbus, K.: 1988, Setting up Large-Scale Qualitative Models. *Proceedings of the Seventh National Conference on Artificial Intelligence.*

Fikes, R., Gruber, T., Iwasaki, Y., Levy, A. and Nayak, P.: 1991, How Things Work Project Overview. Technical Report, KSL 91-70, Knowledge Systems Laboratory, Stanford University.

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

Iwasaki, Y. and Chandrasekaran, B.: 1992, Design verification through function and behavior-oriented representations: Bridging the gap between function and behavior. *Proceedings of the Second International Conference On Artificial Intelligence in Design.*

Iwasaki, Y. and Low, C. M.: 1992, Device Modeling Environment: An integrated model-formulation and simulation environment for continuous and discrete phenomena. *Proceedings of the First International Conference on Intelligent Systems Engineering.*

Iwasaki, Y. and Simon, H.A.:1986, Causality in Device Behavior. *Artificial Intelligence* 29.

Keuneke, A.: 1991, Device Representation: The Significance of Functional Knowledge. *IEEE Expert*, April.

Kuipers, B.: 1986, Qualitative Simulation. *Artificial Intelligence* 29.

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. (eds), *Experience, Memory, and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ.