Functional Representation and Causal Processes

B. Chandrasekaran

Laboratory for AI Research

The Ohio State University

Columbus, OH 43210

1. Introduction

Cognitive agents which are organized to achieve goals in the world have three fundamental activities to perform:

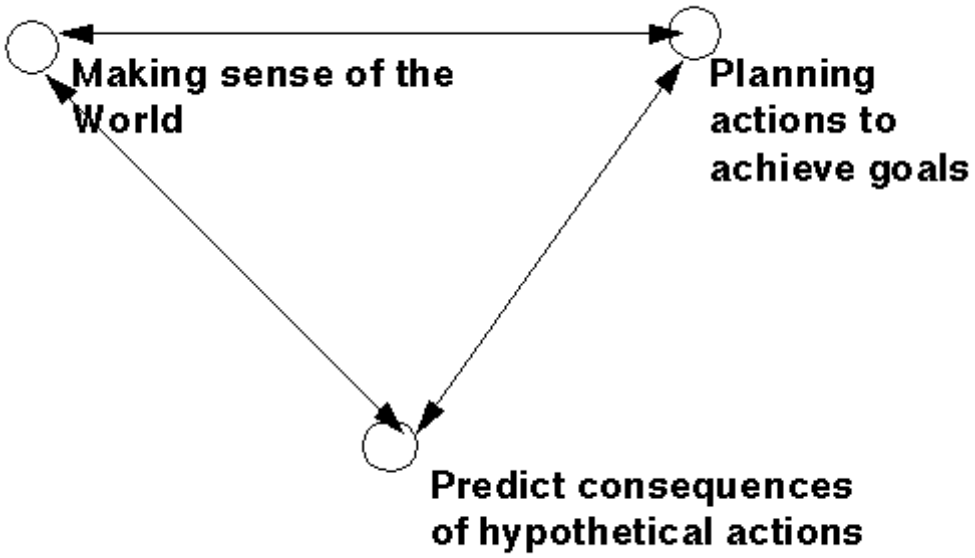

Figure 1. Some subtasks for a cognitive agent.

Making sense of the world: Using sensors and other information (including knowledge in memory), the agents have to form a theory of the world, about what is out there and how it works. This task has a number of different subtasks: sensory processing, forming a perception at the right level of abstraction that relates to the goals, and constructing an explanation of what is going on in the world. In this paper we are concerned with the form such explanations often take. Specifically, we propose that such explanations often take the form of a certain type of *causal story*, and the elements in the story are linked together following specific rules of composition.

Planning actions to achieve goals: The agent has to embark on actions on the world in order to achieve goals. One technique that is commonly employed is for the agent to synthesize a plan, a sequence of intended actions. Knowledge about the state and causal properties of the world is needed to generate the plans. In particular, causal stories can be inverted to reason from desired states to actions which are likely to produce them.

Predicting consequences: A common subtask in planning is to predict consequences of proposed actions to check that desired consequences arise and undesired consequences do not. A common subtask of "making sense of the world," is to evaluate hypotheses about states of the world by predicting the consequences of that state and checking if the consequences are indeed true in the world. If predicted consequences are true, that hypothesis of the state of the world would be given a higher plausibility. Thus *prediction* is an important and ubiquitous cognitive activity. Causal models play a role in prediction as well.

Forming causal models of the world and using them for prediction, planning and forming additional causal models are thus important activities of cognitive agents. The major goal of this paper is to review a theory about what form such causal models take and how they can be used for various problem solving purposes. An important underlying idea is that causal understanding of the world doesn't come simply in the form of "facts" about the world -- propositions or causal rules -- but in the form of causal packages which are organized from specific perspectives and which point to other casual packages. The work reviewed is based on the idea that these packages are basic units of comprehension and prediction. The functional representation (FR) theory is a proposal about the logical structure, representation and use of such casual packages. This article brings together work that has been developed and published in the last decade by a number of researchers.

Study of causality has many dimensions, and has an illustrious history in philosophy. It is important to be clear about what aspects of the construction of causal models are relevant to our purposes here. One major stream of philosophical work on causality grapples with the semantics of the term "cause," i. e., investigates exactly what it *means* to say "A causes B," or, equivalently, the necessary and sufficient conditions on A and B for us to assert that "A causes B." There continue to be different points of view on this issue, but this is not an issue that we will be concerned with in this paper. A second stream takes one or the other meaning of causality as given, and seeks to give computational formalisms for *deciding* if A causes B. In AI, Pearl and Verma (1991), for example, have adopted a probabilistic interpretation of causality, and they then go on to propose a Bayesian technique for computing the likelihood that A causes B. Our work is not concerned with formalizing the semantics of "cause." We take "cause" as a primitive. Our representations already start with some knowledge about causes, and we want to build other causal relations from this. Our work is also not concerned with quantification of the probabilities of causation. For now we take our causation to be deterministic. The research reported here has the goal of elaborating a theory of *representation* of causal processes and using the representation for problem solving of various sorts.

What is the connection between functions of devices and causal processes? So far I have been talking only about causal packages, without any mention of *function*. In the domain of devices, causal representation and reasoning serve several purposes. In design, the goal is to organize components in such a way that the causal processes that result cause the intended function to be achieved. In diagnosis, the causal process is analyzed to see why the intended function is not being achieved. Thus, understanding how to represent causal processes in general is a prerequisite for reasoning about the functions of devices. In reasoning about device functions, we use the same techniques as reasoning about causal processes in the world in general, but we have specific constraints on what kinds of effects are desired, intended, or to be avoided.

2. Human Reasoning About the Physical World

The research on device understanding and reasoning about causal processes is, for me, part of a larger agenda of creating a technology based on the same sources of power that the human agent taps into when reasoning about the world. I would like to make a brief detour describing a broader framework for thinking about human reasoning about the physical world, and along the way point to the role played by causal process packages in this

framework. Those interested only in the technical aspects of Functional Representation can skip this section and go right to Section 3.

2. 1. Human qualitative reasoning

Let us compare a trained physicist and an unschooled man-on-the-street. The physicist has a specialized vocabulary and a number of mathematical modeling and analytical techniques. The physicist might deploy his or her scientific knowledge and perspective selectively, either to reason about specialized domains or when precise answers to certain questions are needed. The special vocabulary and techniques of the physicist notwithstanding, there is a substantial overlap in the ways the physicist and the common man reason in everyday situations. They share an ontology and general purpose reasoning strategies that arise from the properties of the shared cognitive architecture.

Knowledge needed for reasoning about the world comes in many types:

1. A *commonsense ontology* which predates and is in fact used by modern science: space, time, flow, physical objects, cause, state, perceptual primitives such as shapes, and so on. The terms in this ontology are experientially and logically so fundamental that scientific theories are built on the infrastructure of this ontology. Early work in Qualitative Physics (QP) had as a main goal elaboration of such an ontology (Hayes, 1979, and Forbus, 1984 are examples). Even today, a good deal of QP research grapples with the development of ontologies for different parts of commonsense physical knowledge.

2. The *scientific ontology* is built on the commonsense ontology (and often gives specific technical meanings to some of the terms in it, such as "force"). Additional concepts and terms are constructed. Some of these are quite outside commonsense experience (examples are "voltage," "current," and "charm of quarks").

3. *Compiled causal knowledge* is knowledge of causal expectations that people compile partly from direct experience and partly by caching some results from earlier problem solving. Which causal expectations get stored and used is largely determined by the relevance of the causes and effects to the goals of the problem solver. There is a more organized form of causal knowledge that we build up as well: models of *causal processes*. By process model I mean a description in terms of temporally evolving state transitions, where the state descriptions are couched using the commonsense and scientific ontologies. For example, we have commonsense causal processes such as "boiling," or specialized ones such as "voltage amplification," "the business cycle," and so on. These are goal-dependent process descriptions, ones in which the qualitative states that participate in the description have been chosen based on abstractions relevant to the agent's goals. In particular, such descriptions are couched in terms of possible intervention options on the world to affect the causal process, or observation options to detect the process. The work on functional representations that I describe in this paper develops the terms to represent these causal process descriptions.

When the process model is based on pre-scientific or unscientific views, we have naive process models (such as models of sun rotating around the earth, or of exorcism of evil spirits). Many pre-scientific process models are not only quite adequate, but are actually simpler and more computationally efficient than the scientific ones, for everyday purposes. Some sciences, such as geology and biology, often give us their causal theories in the form of such process descriptions (e.g., how mountains are formed, how the infection-fighting mechanism works).

These process descriptions are great organizing aids as we will discuss in the paper: they focus the direction of prediction, help in the identification of structures to realize desired functions in design and suggest actions to enable or abort the process.

4. *Mathematical equations embodying scientific laws and expressing relations between state variables*. These equations themselves are acausal, and any causal direction is given by additional knowledge about which variables are exogenous.

2. 2. Modeling and prediction

The framework should provide support for three components of reasoning about the physical world, modeling, prediction and control of reasoning. In what follows, we discuss modeling and prediction. Control of reasoning is rather orthogonal to the main issues of concern here. The interested reader can refer to the section on control of reasoning in Chandrasekaran (1992)

2.2.1. Modeling

All modeling is done in the context of goals to be accomplished, i.e., states to be achieved or avoided in the world. Causal process knowledge plays an essential role in identifying aspects of the physical situation and perspectives that need to be represented. The process models can be used to identify states that should be represented and reasoned about. The heart of the modeling problem is to come up with tractable representations in a goal-dependent way. The aggregation levels (when dealing with populations) (Weld, 1986) , the abstractions, the approximations and the concepts in the representation are all jointly determined by the physical situation, the goals, and the rich storehouse of causal process knowledge that expert reasoners possess.

2.2.2. Prediction

The power of experts in prediction comes, not from wholesale formalization of the problem in terms of Physics and subsequent qualitative or other simulation (as much of current QP work tends to present the problem), but by the use of a substantial body of compiled causal knowledge in the form of causal process descriptions to hypothesize states of potential interest. Further, the state variables participating in causal relations may not all be continuous, and hence, even in principle, not all problems of prediction can be formulated as analysis of dynamical systems, as suggested by Sacks and Doyle (1992) in their critique of QP work. For example, a substantial part of our causal knowledge is about nominal variables ("vacations relax people," "lack of support causes objects to fall"). Simon (1991) describes a causal ordering scheme which works with such variables, but, as a rule, the most well-known qualitative reasoning models and the dynamic system analysis techniques work only with state variables which are continuous.

Humans, in their everyday life, rarely predict behavior in the physical world by generating a long series of causal chains. The reason is that qualitative reasoning about the world proliferates ambiguities rapidly. If you ask someone what will happen if I throw a ball at a wall, that person is likely to start off with the ball bouncing off the wall, move on to it dropping on the ground, and end with, "it will probably bounce a few more times on the floor and pretty soon will roll off." Very little of this sequence of predictions is the result of application of scientific laws of motion. Rather, a short series of causal sequences are constructed from compiled causal knowledge, instantiated to the specific physical situation. Two important sources of power that are available for human experts in generating successor states and handling ambiguities are discussed next.

2.2.2.1. Compilation of potentially interesting consequences

If we ask someone, "what will happen if I throw a rock at the glass window?" that person is likely to say, "the window might break." This answer is generally not a result of any kind of "simulation" of the mechanism of the

glass under impact. A number of such causal fragments, compiled from experience or from earlier problem solving episodes, are stored as part of our causal knowledge about domains of interest. An important aspect of such compilation is that the causal knowledge is, as a rule, no longer in the form assertions of the form "A will cause B" but rather of the form "A might cause B." Only causal paths that lead to interesting consequences (i.e., those that are likely to impact various goals of agents) are stored, but this in turn introduces uncertainty in the causal relation. This ambiguity is OK, since the goal of qualitative prediction is typically not accuracy or certainty, but identification of an interesting possibility that may be investigated more thoroughly if needed.

2.2.2.2. Handling ambiguity

Ambiguities in causal simulation are often handled not on the basis of what effect *will* happen, but on what *might* happen that may help or hurt the goals of the problem solver. Thus, when there is more than one successor state in simulation, the state that is related to goals of interest is chosen for further expansion. In the example involving the glass window, suppose a person was standing on the other side of the glass window, and you saw some one about to throw a rock at the window. You would most likely attempt either to stop the rock throwing or alert the person standing at the window. You would not be paralyzed with the ambiguities in prediction: the rock may not really hit the window, the window may not shatter, the rock may miss the person, the rock or glass fragments may not draw blood, and so on. Prediction of behavior in the physical world always takes place in the context of such background goals. The existence of these goals makes up for the fact that we rarely have enough information to be more than qualitative in our reasoning[1]. Lest one should think that this is only a phenomenon of interest in the commonsense world, it should be stressed that engineering reasoning is full of such goal-driven ambiguity handling. For example, in design analysis, one might use this form of ambiguity handling to identify the possibility that a component will make its way into a dangerous state. Of course, once this possibility is identified, quantitative or other normative methods can be used in a selective way to verify the hypothesis. Scientific first principles are embedded in process descriptions (in the form of explanations of causal transitions) in such a way that these principles can be formally used as needed for detailed calculation.

In engineering and scientific prediction problems, these techniques of ambiguity reduction are not always sufficient. Whenever reasoning about consequences reaches a point where relatively precise answers are needed for choices to be made the situation can be selectively modeled and analytical methods of varying degrees of complexity and precision can be employed. The models that are formed reflect the problem solving goal that is current, and typically represent only a small slice of the physical system. Mathematical techniques of various kinds, including dynamic system analysis techniques recommended in (Sacks and Doyle, 1992), will clearly form a part of this arsenal of analytical techniques.

3. Historical background

3.1. Need for "Models" in Diagnostic Reasoning

A brief historical background to the development of this body of ideas on representation of functions and causal processes might help motivate the ideas. In 1983, one of my main interests in problem solving was diagnostic reasoning. At that time, a discussion was getting started on so-called "deep" versus "shallow" representation of knowledge for diagnosis. Rules were said to be shallow because they were -- it was claimed -- just associations between symptoms and malfunctions (as in Mycin), without any indication of *how* the latter caused the former. It was proposed that, in contrast, there were so-called deep representations which provided a basis for explaining the associations. When diagnostic knowledge, i.e., knowledge that related the malfunction categories and symptoms, was incomplete or missing, it was proposed that these deep representations, also called models[2], might be invoked and the missing knowledge generated. (See Chandrasekaran and Mittal, 1983, for an early

presentation of these ideas, and Bylander, 1990, and Chandrasekaran, 1991, for a more recent analysis of the issues involved.) Model-based reasoning became a big subarea of research in AI over the last decade, precisely in response to the perceived need for a representation that was not just restricted to a narrow range of tasks.

Models were intended to be a description of *how the device worked*. I review two streams of work in representing models of devices (or physiological systems, since many of the ideas arose in the context of medical diagnosis).

3.2. Causal Nets

The work by Weiss, Kulikowski and Amarel (1978) is representative of this stream. (Belief nets of Pearl, 1986, are not meant explicitly as device models but they could be used to represent the causal relations underlying it.) In causal nets, a device's workings are represented as a network of causal relations between the variables describing the system. An effect could be the result of more than one cause, and a cause could result in more than one effect. One could introduce numerical weights in the links between nodes to represent some notion of likelihood of the effect given the cause. There are a number of technical issues in such representation which are not important for my current purpose, but I want to draw attention to two related aspects of such networks. The theories do not propose explicit criteria for the levels of abstraction that should be used for the variables, and for organizing the causal network. In this approach, these two decisions are left as part of the modeling problem, and are thus deemed to be domain-specific issues.

Let me illustrate the point by a simple example. A network in the domain of medicine may have two causal links coming into a node labeled "headache":

high blood-pressure at blood vessels at the temple headache (Link 1)

infectious diseases of certain kinds headache (Link 2).

Note that both are true causal relations, but do not represent different causal mechanisms. In the former, a causal explanation at the level of a physiological mechanism is being offered, while in the latter a disease-level explanation is represented. One of the ways in which diseases of certain kind might cause headache is in fact by activating processes that increase the blood pressure level at the temples. The two relations are best thought of as existing at different levels of abstraction and indexed by different goals. The relation involving infectious diseases is probably most usefully indexed by the goal of "diagnosis." The relation involving the pressure level at the temple might not be activated at all during diagnosis. On the other hand, for research on drug mechanisms, knowledge in Link 1 might be directly relevant and hence should be activated. The FR theory in this paper uses causal networks, but provides task-specific criteria for abstraction levels for the variables and organization of causal knowledge.

3.3. Qualitative Device Models

Another stream of ideas on representing models is the work of de Kleer (de Kleer, 1984; de Kleer and Brown, 1984) and others in that school. Devices (or physical systems) are modeled as having components connected in specific ways. The components are specified as satisfying certain input-output relations. A description of the device in terms of its components and their relations is called the *structure* of the device. The device can be simulated, i.e., values of all the variables could be derived using knowledge about components' input-output properties and about component interconnections. Such a description is termed the *behavior* of the device.

Contrastingly, Forbus (1984) introduced the idea that the way to model phenomena is as a set of *processes* . The physical situation is described in terms of processes at work in some domain of interest and their interaction. This is the structural representation. Behavior is generated by working out the consequences of interacting processes. The process and the component perspectives are seen within the qualitative reasoning community as compatible alternatives: some physical situations are best modeled as components interacting, some others as processes interacting, while yet other as perhaps having both components and processes. I think, however, that it is best to take the process view as the basic one and think of a component view as a special case. Component input-output relations are then a specific type of process and physical connections between components a specific type of process interaction.

Kuipers' (1986) representation of structure is much simpler than the previous two: It is simply a list of state variables and a description of how a change in the value of a state variable affects other state variable values. The three representations, i.e., the componential, the process, and the state variable relations, could be mutually translated. One could take the set of all input and output variables of the components in the de Kleer representation as the state variables of Kuipers and use the input-output component descriptions to generate the description of how state variable values change as a function of changes in the values of the other state variables.

Representationally, all the above approaches describe the world as a set of state variables and the underlying causal process as a set of relations between changes in the state variables. This is of course the standard ontology[3] of physics and system engineering. In the de Kleer picture, the state variables belong to the components, the component descriptions give the causal relations between variables, the component connections describe how the changes propagate. In the Forbus picture, the state variables belong to the processes which also describe the causal relations between changes in the variables. In the Kuipers picture, the state variables and their relations are simply described, without a particular theory of whether the variables arise from components or processes.

The AI researchers in qualitative reasoning proposed an additional set of ideas about simulation, specifically qualitative simulation. In their view, human intelligence is characterized by the fact that such structural models are qualitative, i.e., the causal relations are only described in terms of ranges of values rather than in terms of actual values. Instead of giving the exact description of how a change in variable x causes a change in variable y, the causal relation is only given as "If x increases, y decreases," or some similar type of trend description. de Kleer and Kuipers propose techniques for generating behavior using only such qualitative relations. These approaches are well-documented (e.g., see Forbus, 1988) and there is no need to review those techniques in detail here. We will argue that goal-directed allocation of reasoning resources is a more useful characterization of human intelligence than reasoning in terms of qualitative ranges of values.

Another AI technique for reasoning about the world is consolidation (Bylander, 1988). This work is based on the observation that often, for predicting behavior, the structure of a device is simplified in certain systematic ways. Such a simplification can help avoid component-by-component simulation. For example, given two resistors in series we simplify that into one resistor. Especially in electrical devices and to some extent in electromechanical devices such repeated application of structural grouping operations results in much simplified descriptions which make simulation either much easier or even unnecessary. In circuits for example, we can simplify the resistance by repeated application of formulae for series and parallel resistors, and simply write down formulae for all the currents. Any qualitative analysis can then be applied to this equation.

In these approaches the behavior of the device is composed out of the behavioral description of the components, processes or state variable relations. These techniques need to be complemented by other techniques to provide the following additional capabilities that are often needed:

1. *Device-level Vs component-level abstractions*. In the qualitative reasoning approaches that we described, the terms in which the behavior of the device as a whole is described is the same as that of component-level descriptions. For example, suppose we have an electronic amplifier, and the device's structure is described in terms of the components: the transistors, resistors, capacitors, etc. Let us say that each of these component behaviors is described in terms of their currents and voltages. All of the above techniques for simulation would then produce a description of device behavior in terms of currents and voltages. However, the behavior of interest of the device as a whole is as an amplifier, a higher level of description. We need techniques by which the device-level behavioral abstractions are related to descriptions at the component level.

2. *Concern only with aspects relevant to the goal*. The simulation techniques of qualitative reasoning produce the values of *all* the component-level state variables which are part of the model. However, many of the state variables may not be of interest to the goal at hand. In the amplifier example, if we are interested in the value of "amplification ratio" for some configuration of parameters, there may be no need to generate the values of the currents and voltages in circuits which play no causal role in the production of amplification. The computational work involved in the generation of values of all the state variables may be reduced if we have a *goal-directed* simulation strategy, and a representation which helps in identifying the dependencies and focusing the simulation.

3. *Flexibility regarding detail.* Human reasoning, while it is largely qualitative in the sense of reasoning in ranges, is also capable of invoking techniques for precise calculation if quantitative information is needed. An engineer might perform some reasoning using qualitative information, formulate a well-posed numerical problem that she might solve on paper or computer, and proceed with qualitative reasoning again. That is, human reasoning flexibly integrates computations of different degrees of accuracy and precision in a goal-directed way.

The FR work that we review in this paper provides the complementary function-oriented view that is needed to provide the above capabilities.

4. Functional Representation

The Functional Representation framework is a proposal about the top-down representation for goal-directed, flexible reasoning that bridges abstraction levels. It was originally proposed in (Sembugamoorthy and Chandrasekaran, 1986) for the causal processes that culminate in the achievement of device functions. (Some devices achieve their functions by means of causal *processes*, while the function of others is explained directly from their structure. We discuss this distinction later, but for now consider only devices wherein causal processes are the means for achieving the functions.) In FR, 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.

4.1. Informal overview

FR for a device has three parts:

a description of intended *function* (independent of how the function is accomplished),

a description of the *structure* of the device, i.e., what the parts are and how they are connected to the degree of detail chosen by the describer, and

a description of *how* the device achieves the function, specifically a *process* description.

de Kleer (1985) introduced the terms "structure" and "behavior" formally in the study of devices, and he also discussed the idea of "functions" as having to do with the teleology of artifacts. de Kleer and Brown (1983) also proposed a process description of mental models of how a device worked. Our work builds on some of these notions.

The part of FR that describes the function treats the device as a blackbox, i.e., it makes no assumptions about its internal structure, not to mention any processes that take place in the device. This is because the same function may be achieved in different ways, and thus a description of the function itself should not make any commitments about the structure. This may be called the "No structure in function" principle, a kind of converse of the "No function in structure" principle that is due to de Kleer. Of course for describing a function we need to describe a certain minimum amount of structure: how the device is to be embedded in the environment (e.g., where it is to be placed and how it to be connected), where on the device the operations to initiate the function are to be performed (e.g., turning the switch on for an electrical lighting circuit or an electric iron), and where the functional output is to be realized (e.g., light near the bulb, heat at the ironing plate). Other than this, we just need to describe under what conditions what kinds of predicates are to be satisfied by the device in order for it to achieve the function of interest. That is, a function is represented by describing the context of its application, the initiating conditions and the predicates that the device has to satisfy for it to achieve the function.

Of course FR as a whole enables us to combine this description of function with how the specific device achieves it, by adding descriptions of structure and the causal processes which make the function happen in device. This independence of the *what* of the function from its *how* is important to emphasize since this a point that is widely understood to be an important desideratum for representing function (e.g. Brajnik, et al, 1991, Chittaro, et al, 1993, Kaindl, 1993), while some of these writers apparently and mistakenly believe that FR does not maintain this distinction.

Representing the structure is straightforward: we simply list the component names and their functions and indicate how the components are put together to make the device, i.e., describe the relations between the components. The components functions are described using the same ideas as in the description of the device function. We give examples of such descriptions later in the paper.

The basic idea in describing how the device achieves the function is that we need to construct a *causal process description* (CPD) of a certain type. That is, we describe how the device goes through a causal state transition process in which the initial state is the state of the device just at the time the device "starts," the final state is the state at which the device is achieving the function for which it is designed, and each state transition is explained by appealing to knowledge about components or the domain. The idea that a behavioral description of the device is the link between structure and function has been stated often in the AI literature -- from the early works of de Kleer to Gero et al, 1992 -- but our proposal on CPD takes a very specific stance about what kinds of behavioral descriptions have the explanatory power needed to explain the function and bridge the levels of abstraction between component-level and device-level descriptions.

A CPD can be thought of as a directed graph whose nodes are predicates about the states of the device, and links indicate causal transitions. The links have one or more annotations that explain the transition. The annotations, according to the theory, belong to certain specific types.

Consider a simple electrical circuit with a switch, a voltage source and a resistor. Let us say that the function we are interested in is production of heat. A causal account of how this function comes about might be given as follows:

```
A->B->C->D->E
```

where A is 'switch is on'

B is 'voltage applied between terminals'

C is 'current to flow in the circuit'

D is 'current flows through the resistor'

E is 'heat generated.'

Normally "->" can be interpreted as "causes." The CPD is not a neutral description, but one that is oriented towards explaining a selected function. If the function of interest is the production of light and the resistor was a filament inside a light bulb, the link D E, might read as follows:

D is 'current flows through the resistor'

E is 'light generated.'

Of course the heat through the resistor also produces heat. If we are interested in explaining the production both of light and heat, the graph might look as follows:

```
A->B->C->D->E
          \->E'
```

where E corresponds to production of light and E' to production of heat.

Now let us examine how the transitions might be explained.

"A -> B" : "Closing the switch causes voltage to be applied between the terminals of the circuit." In explaining this transition we use knowledge about "closing" function of switch (the component "switch" has two functions, "open" and "close") and the "voltage" function of "battery."

The transition "B -> C," "the application of voltage causes a current to flow through the circuit," invokes the connectivity-providing functions of the various connectors, and also uses a domain law, viz., Ohm's Law. Checking the connectivity functions of the connectors ensures that the structure is in fact a circuit. Ohm's Law can be used to provide qualitative or quantitative information about the quantities involved.

The transition "C -> D" asserts that "current flowing through the circuit means that there is current through the resistor." This particular transition is not normally viewed as a causal transition. The explainer is simply inferring a predicate of interest, in this case, that there is current through the resistor. The explanation appeals to the meaning of a circuit. If we want only causal links, we can collapse the two links into one, "B -> D," asserting, "Application of voltage causes current through the resistor." The explanation would still appeal to the connectivity functions of the connectors and to Ohm's law.

The transition "D -> E," asserting that "current through the resistor causes heat to be produced," can be explained in different ways. One possibility is to simply point to the scientific laws that relate electricity to heat production in a resistor, similar to the way Ohm's Law was used earlier. Such a law can be given in a numerical or qualitative

form, as needed. In this case, the link would be annotated as "By domain-law (current-to-heat equation)." In this explanation, the domain law is taken as a primitive explanation, satisfactory for the current problem solving goals.

The same transition can be explained by appealing to another process: how electricity gets converted to heat. The link would have the annotation "`By-CPD`(electricity-to-heat)." Someone who already understands this process can use it if needed and build the more detailed (and longer) causal story. This causal process can be explained separately for some one who doesn't already understand it.

To summarize the various explanatory annotations:

i. `By-CPD`: This points to another CPD that provides further details of the transition. The details of the CPD may not matter for the current purpose. If they matter, the CPD may be part of the prior knowledge of the agent, or can be explained separately. Potentially long process explanations can thus be hierarchically composed out of other process explanations, making explanation at each level shorter. (The Abel system of Patil, et al, 1981, uses a similar approach of describing causal chains with hierarchically increasing details.) CPD's (such as "boiling" in commonsense physics, or "electricity-to-light production" in the circuit example) can be reused, possibly after instantiating some parameters (e.g., the pressure at which the boiling is done, the liquid that is being boiled, and so on). Human expertise in a domain contains knowledge of a large number of such causal processes that can be parametrized and re-used.

ii. `By-Function-Of-`<component> : This annotation appeals to the function of a component as the causal explanation of the transition. A major goal of causal explanation in devices is to explain the behavior of the device in terms of the properties of the components and their inter-connections. Again, a large part of expertise of human experts in a domain is in the form of knowledge about generic components and their functions (though, in many cases, how the component functions may not be known). The ability to explain the device functions partly in terms of component functions, and to explain component functions in turn in terms of the functions of its subcomponents helps in the formation of functional/component hierarchies in explanation and design. Also, components with different internal structure but the same function can be substituted.

iii. `By-Domain-Law` <law>. Another form of explanation is by appeal to domain laws. In the domain of engineering, scientific laws are the ultimate basis of explaining why the device behaves as it does. For example, the state transition, "*5 Volts at the input -> 2 amps through the load*" might be explained as "`By-Domain-Law(`*Ohm's-Law: Voltage = Current * Resistance)*".

For a particular device, any realistic FR description will taper off at some level of components and CPD's. The terms that are used at the lowest level of description are themselves undefined. In explaining to humans, these terms are assumed to be part of the commonsense knowledge. For machine processing, the terms at the lowest levels of description are just strings of symbols. Thus every FR is inherently incomplete.

Non-causal links: Sometimes additional, noncausal, links may need to be added to arrive at the predicate of interest. For example, for an amplifier, we may have constructed the CPD,

Voltage 1 at the input `->` `. . .->` `. . .->` *Voltage 10 at the output*,

but the function that needs to be explained might be "*Amplification of 10*." A non-causal, definitional/abstraction link can be used to arrive at the node "*amplification of 10*" from "*Voltage 10 at the output."* Such links can be used to indicate an inference that follows from predicates in the earlier nodes.

We have used examples of four types of ways in which a link can be explained: appealing to the function of a component, a causal process, a domain law, or some non-causal inference. In the body of FR research, the set of annotations has been somewhat open-ended and evolving. In the original paper (Sembugamoorthy and Chandrasekaran, 1986), we had made finer distinctions for the `By-knowledge` link, and also proposed a "`By-structural-assumptions`" link. The last link was supposed to handle situations where the structure as given was not sufficient to justify the causal transition, but with some additional assumptions about the structure, the transition would work. In the newer versions of the FR language, we include such requirements under various kinds of qualifiers. Vescovi, et al (1993) have identified another link, called "`By participation-of` <component>," to account for the situation where some aspect of the component not explicitly identified as its function plays a role in some state transition.

Qualifiers: In addition to the explanatory annotations, the links may have *qualifiers* which state conditions under which the transition will take place. In FR the qualifier `Provided(p)` is used to indicate that condition `p` should hold during the causal transition for the transition to be initiated and completed, and `If(p)` to indicate that the condition `p` should hold at the moment when the causal transition is to start. The conditions can refer to the states of any of the components or substances. Many of these qualifiers are eventually translated as conditions on the structural parameters.

4.2. Components of FR

I will use a running example of a device called a Nitric Acid Cooler (NAC) (Goel, 1989) to illustrate the various aspects of FR. Figure 2 is a schematic of the device (NAC).

4.2.1. Structure of a device

The structure of a device is a specification of the set of components that constitute the device and the relations between the components. The components are represented by their names and by the names of their functions, which are all domain-specific strings. Components and functions can have variables for its parameters and thus may describe classes. In the NAC example, component class *pipe*(*l*,*d*) describes pipes with length *l* and diameter *d* , while *pipe2* is a specific instance of *pipe*(*l*,*d*) with specific values for *l* and *d*. Similarly, the device NAC as a class has a function *cool-input-liquid*(*rate, temperature-drop*) where *rate* and *temperature-drop* are capacity parameters of the function *cool-input-liquid*. A particular NAC might be identified by specific values for these parameters. Devices can have *substances* whose *properties* are transformed as part of their functions. Substances can be destroyed and new substances created. Substances can be physical (e.g., *nitric-acid*) or abstract (e.g., *heat*). In the NAC example, the substance *nitric-acid* has properties *temperature*, *flow rate,* and *amount of heat* (which itself is a substance).

Components have *ports* at which they come together with other components in certain relations. For example, the component type "pipe" might be written as *pipe*(*l*,*d*;*t1*,*t2*), where *l* and *d* are the length and diameter, and *t1* and *t2* are the input and output ports. Components are configured in specific *structural relations* to each other to make a device. In an electrical circuit, electrical components are *electrically-connected* at the defined terminals. In the NAC example, the relations include *conduit-connection*, *containment*, etc. (The relational vocabulary can also include unintended relations, e.g., *electrical leakage between electrical components.* The components can be in unintended relations to each other as a result of malfunctions.) The vocabulary of relations is domain-specific. The semantics of the relations are established by the domain laws that govern the behavior of the components in the given relations.

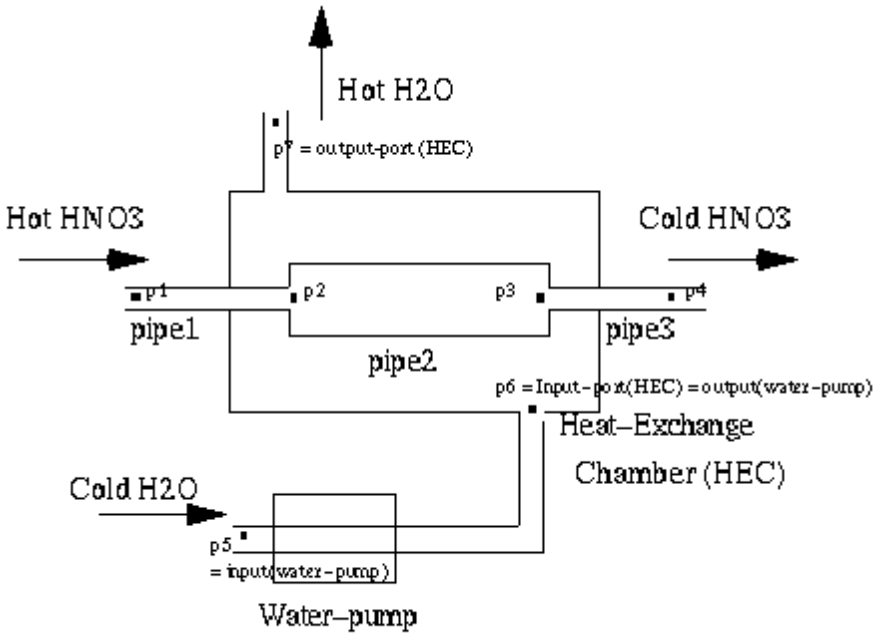

Figure 2. Schematic of a Nitric Acid Cooler

The FR language uses the following keywords for describing structure:

`Structure`((`Device`(<device-name>, <functional parameters>, <ports>)),

`Component`(<component-name>, <component parameters>, <ports>),

`Function`(<component>),

`Relation`(<relation{<component, port>,....<component, port>}>).

The structure of NAC is given in Figure 3.

`Structure`((`Device`(*NAC; cooling-capacity and temperature parameters;* ports: *p1, p4, p5, p7*))

`Components`: *pipe1*(*l1, d1*; *p1, p2*), *pipe2*(*l2,d2*;*p2,p3*),

pipe3(*l1*,*d1*;*p3*,*Output*)

Heat-exchange-chamber(*<dimensions>*, *Input-port, output-port*)

Water-pump(*Input*, *Output)*

`Function`(*pipei*): *Conduit* (*input*, *output)*

`Function`(*Heat-exchange-chamber*): *exchange-heat*(<parameters>)

`Function`(*Water-pump*):.....

`Relations:Component`(*pipe2*) *contained-in* `Component`(*Heat-exchange-chamber*)

`Component`(*pipe1*) *conduit-connected* (*pipe2*) {`Ports`: <information about ports>}

`Component`(*Water-pump*) *conduit-connected* `Component`(*Heat-exchange-chamber*) {`Ports`: <information about which ports of components are connected; e.g., *Input-port* of *Heat- Exchange Chamber* is the same as the *Output* of *Water-pump*>}

Figure 3: Structural description of NAC

In the figure, the terms in *italics* are domain-specific names for functions, components, relations, etc. The interpreter for FR treats them as strings. The terms in `Courier` are terms in FR. Additional domain-specific interpreters may be able to use the italicized terms as meaningful keywords. For example, a mechanical simulator can use terms such as *contained-in* and *conduit-connected* to perform simulations. For the purpose of this exposition, they are to be understood in their informal, English-language meanings. The syntax of the `Relations` keyword is that an *n*-ary relation has *n* components and a `ports` term indicates which ports of the components are connected. Note that the components are described purely in terms of their functions. This makes it possible in principle to replace components by structurally different but functionally identical components. Further, the components themselves can be represented as devices in their own terms.

4.2.2. States and Partial States

A device *state* is represented as a set of state variables {*Vs*} consisting of values of all the variables of interest in the description of the device. State variables can be either continuous or discrete. In particular, some of the variables may take truth values {T,F} as their values, i.e., they are defined by predicates. An example of a continuous variable is *water temperature* in a device that uses water for cooling a substance. An example of a variable defined by a predicate is "?*open(valve)*" This variable will take values T or F depending upon whether the valve is open or not.

In describing functions and causal processes, we generally talk in terms of *partial states* of the device. A partial state is given by the values (or some constraints on the values) of a subset of state variables. For example, the partial state (call it *state1*) of NAC (describing some relevant state variables at the input *p1* of the device can be given as {*substance: nitric acid*; *location(substance)*: *p1*, *temperature(substance)*: *T1*}. *State2*, describing the properties of nitric acid at location *p2*, will only differ in the location parameter, while the partial state description, *state3*, at location *p3* will be {*substance: nitric acid*; *location(substance)*: *p3*, *temperature(substance)*: *T2*}, where *T2* < *T1*.

4.2.2.1. State Description Languages

The language in which states are represented is itself not part of the representational repertoire of FR and is largely domain-specific. In economics, the state variables would be entities such as *GNP*, *inflation-rate*, etc; in nuclear-plants, an entity might be *radiation-level*. Goel (1989) has defined a state description language which is useful in describing devices that deal with material substances that change locations, e.g., those in which *substance flow* is a useful notion. The state representation that we just used for NAC uses this language. In NAC and many other domains a state can be represented by a real number, a binary predicate or a vector of real numbers and predicates.

4.2.2.1.1. Shape-based states

In principle the states can be images or shapes. For example., when we want to describe the causal process corresponding to the flow of hot molten metal into molds, the relevant intermediate states may be shape descriptions. (See Chandrasekaran and Narayanan (1990) and Narayanan and Chandrasekaran (1991)) for some descriptions of work relating to such visual simulations.)

4.2.2.2. State Abstractions

Consider a device which, at one level of description of states, has one of its state variables, say *s*, going through the following partial states repetitively: {-1, 0, 1}. That is, the state variable is oscillating around 0. Suppose we define another state variable *?oscillating* by a process of abstraction from the values of *s* over time. This would be a binary variable taking on the value *Yes* if the values of *s* are cycling through {-1, 0, 1} and *No* otherwise. Allemang and Keuneke (1988) discuss a number of issues in creating such abstractions. (See also Weld, 1986.) Which state variable behaviors are abstracted and in what way are determined by considerations outside of the physical system itself. In fact, all descriptions of physical systems at any level presuppose and arise from a set of problem solving goals. Whatever state variables we use to describe a system at the lowest level can themselves be defined from such a process of abstraction from physical behavior at even lower levels; there really is no way of representing any real-world system in a truly neutral way.

4.2.3. Causal Process Description

Formally, the causal process description (CPD) (see Iwasaki and Chandrasekaran, 1992) is a directed graph with two distinguished nodes, *Ninit*, and *Nfin*. Each node in the graph represents a partial state of the device. *Ninit* corresponds to the partial state of the device when the conditions for the function are initiated (such as turning a switch on). *Nfin* corresponds to the state where the function is achieved. Each link represents a causal connection between nodes. One or more qualifiers are attached to the links to indicate the conditions under which the transition will take place, and one or more annotations can be attached to indicate the *type* of causal explanation to be given for the transition. The graph may be cyclic, but there must be a directed path from *Ninit* to *Nfin*.

In the NAC example, let nodes *state1*, *state2* and *state3* correspond to the states of nitric acid at the input to *pipe1*, at location *p2* and and location *p3*, respectively. Figure 4 depicts the CPD graph (without any annotations or qualifiers) describing what happens to the nitric acid and the water as they flow through the chamber. In the figure, the nodes are described in informal English, but they can be described more formally similar to my earlier description of *state1*.

4.2.3.1. Annotation of links in CPD

This example will illustrate the use of three types of annotation for explaining causal transitions: appealing to another causal process, to a function of a component, and to domain laws (so-called first principles of the domain). It will also illustrate the use of some qualifiers, which are conditions on states or device parameters for the transition to take place.

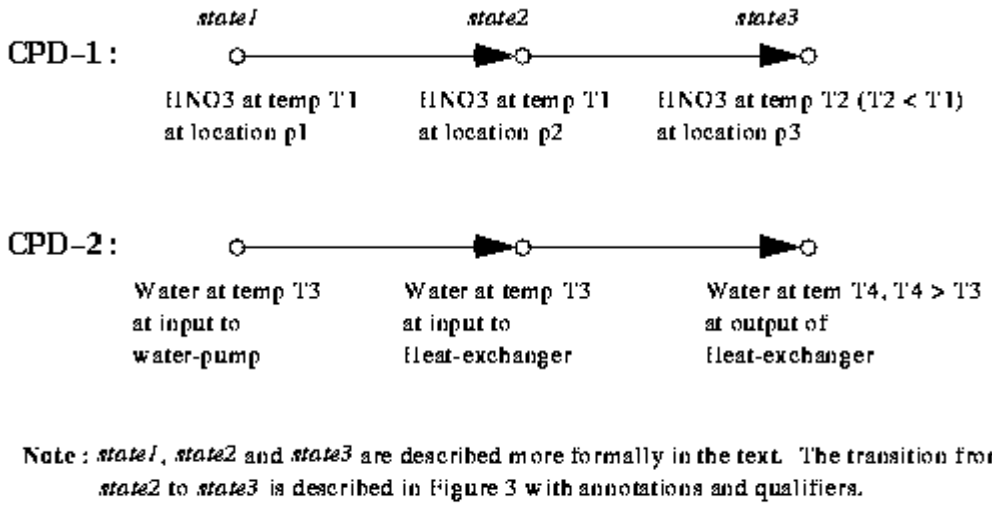

Figure 4: CPD's for device NAC (without link annotations). *state1*, *state2* and *state3* are described more formally in the text. The transition from *state2* to *state3* is described in Figure 3 with annotations and qualifiers.

Figure 5 shows one fully annotated causal transition in the Nitric-Acid-Cooler. It uses two functional and one domain law annotations, and uses conditions on the structure and substances as qualifiers. The qualifiers include conditions on the properties of the substance (it should be a *liquid of low acidity*) and structural conditions (*the chamber fully encloses pipe2*). Note that a transition may have more than one annotation or qualifier.

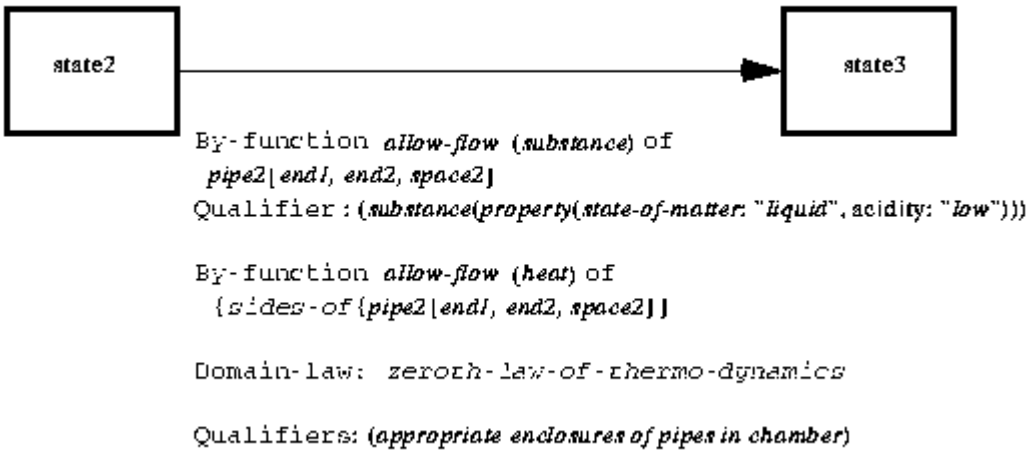

Figure 5. Annotations and qualifiers for a causal transition in NAC

4.2.4. On Functions

4.2.4.1. Types of Functions

Keuneke (1989, 1991) has identified four types of functions: `ToMake, ToMaintain, ToPrevent`, and `ToControl`. (Franke, 1991, on device properties such as "guarantee," "prevent," etc. is motivated by quite similar ideas.) Formal definitions of these function types have been developed (Iwasaki and Chandrasekaran,

1992), but for our current purposes the following informal ones should suffice. All the function types above except `ToControl` take as argument a predicate, say *PF*, defined over the state variables of the device. The function is of the `ToMake` type if the goal is to make the device reach a state in which *PF* is true, and after that state is reached no specific effort is needed to keep the predicate's value to True, or it doesn't matter what state the device goes to after the desired state is reached. A function is of type `ToMaintain` if the intention is to take the device to the desired state and the device has to causally ensure that the predicate remains True in the presence of any external or internal disturbance which might tend to change the device state. The function type is `ToPrevent` if the goal is to keep *PF* from ever being true, and some active causal process in the device is required to ensure it. (While logically `ToPrevent` *P* can be written as `ToMaintain` (Not *P*), there are important differences in practice. Pragmatically, a designer charged with making sure that the device doesn't explode uses knowledge indexed and organized for this purpose. This prevention of explosion -- say by using a thick pipe -- is not the same as maintaining some dynamic state variable in a range. The function type `ToControl` takes as argument a specified relation *vo = f(v1,...vn)* between state variables *vo, v1,...vn*, and the intent is to maintain this relationship. That is, we wish to control the values of specific variables as a function of the values of some other variables.

A function *F* thus has the following descriptive elements:

`Function` (<function-name>)

`Device` (<device-name>)

`Type` (<one of the above types>)

`Start-Conditions`(<conditions>; the conditions under which the function will be initiated)

`Function-Predicate` or `Control-Relation`(<predicate|control relation>; the predicate that has to be reached, maintained or prevented, or the control relation that has to be maintained)

`By-CPD`(<set of causal-process-descriptions>; explains how the function is achieved)

Consider the example of a Nitric Acid Cooler in Figure 2. Hot nitric acid goes into a heat exchanger and exchanges heat with the water that is being pumped in. The water gets hotter while the acid gets cooler. The functional definition of NAC can be given as follows.

`Function` (*Nitric-acid-cooling*)

`Device` (*NAC*)

`Type` (`To-Make`)

`Start-Conditions`(*Input temperature of Nitric Acid =T1*)

`Function-Predicate`(*Outlet temperature of Nitric Acid = T2, T2 < T1*)

`By-CPD`(CPD-1 in Figure 4)

The complete FR is given by specifying the device name, its structure, the state variables of interest, the functions

of interest, and the functional template including the CPD's. We have described the representational primitives needed for all the above. Many implementations exist for the FR language, with somewhat different syntax in each implementation. We have used a composite syntax, chosen mainly for expository effectiveness. We have suppressed many details by giving English language descriptions of the intended information within parentheses or curly brackets. For example, we say "`qualifiers`: (appropriate enclosures of pipes in chamber)" in Figure 5. A detailed syntax for representing the relevant relations about pipes is in fact available and used in (Goel, 1989).

Function types such as "`To-Maintain`" and "`To-Prevent`" apply not only to engineered artifacts, but to reasoning about natural phenomena as well: e.g., "The centrifugal force *prevents* the satellite from escaping into space," and "the rain/evaporation cycle *maintains* the salinity of the oceans."

4.2.4.2. Passive Functions

So far, all of our discussion of function has been in the context of temporally evolving causal processes. Keuneke (1989) made a distinction between "active" and "passive" functions. A chair satisfies the function of providing a seat for a person, but normally we don't explain this functionality by giving a description of state changes that the chair goes through as a person sits on it. This is not to deny that such a description can be given, but simply to point out that the function is normally explained as a match between the structural properties of the chair and how that meets the need for providing a seat. Similarly, a flower arrangement provides the function of decorating a room. Again, a description of how it achieves this function can be given by describing some process in which a viewer goes into a state of enjoyment, but the flower arrangement as such is not explained as something that undergoes a process involving state transitions in order for it to achieve the function.

In these cases, the object achieves a function simply by virtue of its structural properties. Such devices can still have parts and parts may have functions as well, and the device function as a whole will still arise from the parts achieving their functions. For example, in describing how a chair achieves its function, one might say at the top level that a chair has the structural property that a component with function "support human bottom comfortably" is attached in a certain physical relation to a component with function "elevate the device from the floor at a height equal to the length of average human legs," and optionally and in a certain physical relationship to two components with functions "support arms." Each of the component functions can be defined in terms of certain structural properties.

With the exception of a recent thesis by Toth (1993) wherein she considers the functions of mechanical structures (such as frames and trusses), there has not been much work in the FR framework in representing such passive functions. In visual recognition, there has been some work (e.g., Stark and Bowyer, 1991, Brand, et al, 1992, Rivlin, et al, 1993) in using such functional notions to recognize the identity of objects in a visual scene. Traditionally, programs for recognizing chairs in scenes would have some sort of a structural template or description of a chair, and the recognition process would consist of trying to match this template against the visual scene. In this approach the recognition system can only recognize types of chairs for which it has a structural description. On the other hand, a person who has never seen a bean-bag chair might recognize it as a chair. Function-based recognition may actively use the hierarchical functional model of what makes a chair a chair, namely, it has parts that serve the role of a seat, and so on, to check if the object can serve the function.

4.2.4.3. Content-theory of functions

In specific domains, we can develop theories of elementary functions that can be combined to make more complex functions. When the domain involved is of great generality, such as mechanical force transmission, such a content theory can be widely useful. Hodges (1992) has developed a set of useful basic functions in the domain

of physical objects interacting by means of their shapes. Examples of such elementary functions are: *linkage, lever, gear, pulley, screw, spring,* and *container*. These functions themselves are defined in terms of a vocabulary of state change operations, such as *move, constrain, transform,* and *store*.

4.2.4.4. Functions: Intrinsic to Objects?

How closely should functions be attached to object descriptions? There has been much debate about the distinctions between "function," "behavior," and "use." There is a view that functions of an object are strictly outside of a supposedly more intrinsic behavioral description of an object. That is, functions are a separate ontological category from behaviors, and behaviors are part of a neutral description of objects. Yet another proposal attempts to make a distinction between the "use" made of an object and its intrinsic function.

Any description of an object makes choices from available descriptive terms. The behavior of a physical object might be described by one observer in terms of currents and voltages, while another observer might describe it in terms of amplification. The former description is no more intrinsic that the latter, since the description in terms of currents and voltages is already based on a point of view which chooses both to omit certain things about the object (e.g., color, weight, etc., in the case of a circuit) as well as to commit to a certain level of abstraction (the same object could have been described in terms of more fundamental physical phenomena, say its atomic behavior).

There is thus no completely neutral description of an object's behavior. We have proposed to interpret "function" as a distinguished behavior of interest for some observer. There is no commitment to the use of the object for a purpose. However, someone who uses an object in a certain way is an observer for whom the behavior of interest is the one that corresponds to her use of it. Thus, in the sense in which we propose to use the terms, the representational terminology for "use" is the same as that for "function" which is the same as that for "behavior." Function is simply a behavior of interest[4], and use of a device in a certain way is possible because it is capable of behaving in that way.

What is the function of a thermostat? To keep the temperature in a room in a given range? To regulate the flow of electricity to the heater as a certain function of temperature? To regulate the flow of electricity as a function of a specified spatial distance inside bimetallic strip? None of these descriptions is intrinsic to the piece of matter that we call thermostat. Depending upon how we model the embedding context, all of the above descriptions can be supported. Consider the following CPD: "(T < T1) strip curls in end makes physical contact switch closed furnace is on temperature increases." If we model the thermostat as embedded in the physical circuit, we can stop the "intrinsic" description of the thermostat at "end makes physical contact." If the embedding context is modeled as the electrical circuit, the thermostat description can go up to "switch closed." And so on. There is nothing intrinsic about it, though clearly there may be some conventions regarding well-known devices about where we normally stop.

The notion of an intrinsic level of description for physical objects comes from a belief in reductionism that all of us have been raised on, namely, that physics gives us the fundamental level of reality, and all other higher levels can be reduced to it. That might be an appropriate doctrine about the nature of reality, but reasoning about the world requires levels of description corresponding to levels of interest. Our knowledge of causality ties descriptions at various levels: events both within levels and between levels are causally linked. A theory of reasoning about the physical world cannot be based on a reductionistic "intrinsic" physical description.

4.3. Remarks on FR

1. *Nodes in CPD are partial states*. The nodes in the graph are partial states, i.e., predicates about some aspect of the device, not a complete description of all the states of the device.

2. *The level of abstraction of the predicates in the CPD is at the level needed for the causal story*. Some of them may directly correspond to component-level state variables, while others may have only a device-level existence. In the CPD in the electrical circuit example, heat and light are device-level state variables, while voltages and currents and the off or on status of the switch are component-level variables.

3. *FR is underdetermined by the underlying structural description*: The form an FR takes is not unique. For the same physical system, not only are there different FR's for different functions, but even for a given function, different FR's can be written. The differences would reflect different assumptions about background knowledge, different decomposition strategies, and, to some extent, different intended uses for the FR. Suppose the FR model is going to be used for diagnostic purposes. Appealing to the domain laws that relate current to heat might be sufficient if all we want to do is to check whether the resistor is faulty and if so to replace it. However, for inventing different types of materials for making the resistors, a more detailed CPD that refers to the properties of the resistor material may be more useful as an annotation for the same link. It should be emphasized, however, that being goal-dependent is not the same as being arbitrary or simply subjective. The FR is still intended to represent reality, but what aspects are chosen and what levels of abstraction they are represented in depend on the problem solving goal.

4. *The CPD integrates the "object" and the "process" views*. The CPD's in the FR view integrate the process (Forbus, 1984) and object (de Kleer, 1984) views in modeling a physical system. Components have functions which are realized through CPD's which in turn appeal to functions of other components.

5. *FR and CPD's Capture Causal Understanding in General*. So far, we have talked about functions of devices, i.e., roles intended by designers or users for some physical objects. But, as stated earlier, the FR framework is really a framework for causal understanding, not just for representing functions of engineered artifacts. Consider the following questions:

i. How does this device work (i.e., deliver the intended function)?

ii. How does cancer "work"? (i.e., what is the mechanism of cancer?)

iii. How do clouds make rain? How are mountains formed?

iv. How does the immune system work?

v. How does this program work? this algorithm work?

vi. How is sticking pins in the doll going to bring my lover back? (i.e., how does voodoo work?)

Question *i* of course captures the traditional notion of function of an engineered device. In question *ii*, cancer is hardly an intended function. The questions in *iii* are about natural phenomena. The scientific temper of our times would be inimical to talking as if clouds have an intended function to make rain, or that geological processes were intended to form mountains. Regarding question *iv*, the theory of evolution allows to talk about the function of immune systems (to give bodies immunity from infections) and the function of heart (to pump blood). In question *v*, we do not have a physical object at all, but an abstract object of some sort, and we still often talk about it in the same way as we talk about causal processes in physical objects. In question *vi*, while the particular voodoo

practice has an intended function and the voodoo practitioner is likely to give a causal account, it is not something we would believe in.

Let us first consider cases in the above which correspond to physical phenomena (*i* to *iv*). If we interpret the term "function" not in the sense of an intended state, but in the sense of a role or state of interest, then we can see that the casual accounts that are needed to explain the phenomena are of the same type as those we are attempting to capture in CPD's. That is, we are interested in explaining the occurrence of a *distinguished state of interest*: in *i*, the device reaches a state corresponding to the intended function, in *ii*, the cellular mechanisms reach a cancerous state, in *iii*, a collection of water molecules with certain properties reach a state corresponding to forming rain, and in *iv*, the body reaches a state in which invading organisms are destroyed. The functional state in an artifact is just a type of distinguished state of interest.

In the case of abstract systems such as programs (question *v*), we still have components (modules in programs), their functions (i.e., they have identified roles to play in the achievement of the goal), and structure (control structures for programs that define how the modules are invoked and their results used). We talk about program states changing as a result of certain operations. Allemang (Allemang, 1990; Allemang, 1991; Allemang and Chandrasekaran, 1991) constructs FR's and CPD's for computer programs. It is beyond our current scope to discuss the metaphysics of how notions of a causal process apply to mathematical objects such as programs, but, at least formally, the FR machinery is both applicable and useful.

Question *vi* is interesting in another way. The causal story it tells is, at least according to the lights of science, a false one. But what is interesting is that such an explanation has the same logical structure as the causal story that a scientist would construct for a natural phenomenon. The explanation of the voodoo process probably goes something like, "Sticking of the pin causes such and such spirits to be awakened who in turn do such and such things ..." If asked for an explanation of each causal transition, explanations are likely to appeal to the functions of the various spirits, some recognized causal process in the voodoo theology or some recognized domain relations. That is, what makes the voodoo explanation false is not the form or logic of the explanation, but the phenomena that are appealed to in the explanations.

The point of the above discussion is that the underlying framework for capturing the logic of causal mechanism explanation has broader applicability than just to engineered artifacts. The framework itself can be thought of in terms of explaining how certain distinguished states of interest are or are not caused, and what roles parts of the configuration play in this process. When we design an artifact with a function in mind, we want to create a configuration which can reach states that satisfy the predicate corresponding to the function of interest, and we want the various parts of the device to play certain causal roles in this process. When we debug an artifact that is malfunctioning, we want to know why the configuration is not reaching certain distinguished states of interest. But, the logical form of the explanation of the causal processes in the engineering domain is itself independent of the notions of desired or undesired functions.

In what follows, we will in general talk about engineering artifacts and their intended functions, but it should be kept in mind that many of the points we make can be restated with respect to the general problem of understanding causal processes of configurations, physical or abstract.

4.7. Applications of FR

The FR framework in its various forms has been used for a variety of problem solving tasks. In this section, I review, in varying detail, a number of tasks for which it has been used.

4.7.1 Generating Causal Explanation By Simulation

4.7.1.1. The Problem Statement

Consider the following problem: Given a set of observations and a diagnostic hypothesis, construct an explanation of how the hypothesized malfunction caused the observations. That is, construct a set of causal stories each oh which starts with the hypothesized malfunction and ends in one or more observations. In the following, I describe the work of Keuneke (Keuneke, 1989) on the use of FR for solving this problem. Technical definitions of a few terms may be useful:

Observations: observable state variables. Some of the observations are called Symptoms, which are abnormal state variable values indicative of malfunctions and trigger the diagnostic process, e.g., specification of a drop from normal pressure. Malfunctions are observations which correspond to device-level functions not being delivered. (Malfunctions are symptoms as well.) The rest of the observations give information about the state of the device, but are not immediately classifiable as abnormalities. Most observations on a complex system are of this type.

Diagnostic Hypotheses: These are malfunctions of components or missing (but expected) relationships between components. A missing relationship would eventually result in the enclosing subsystem being a malfunctioning component.

Causal Explanation: A CPD that starts with a diagnostic hypothesis and ends with one or more observations to be explained. The explanation sought can be formally stated as follows:

diagnostic hypothesis x1 -> ....-> xi ....-> xN ...

where each xi is either (1) a state which is causally relevant in producing an observation, but is itself not a malfunction, (2) a component or subsystem malfunction, or (3) an observation at the device-level[5]. In Keuneke's work, the diagnostic hypothesis is assumed to have been generated by some form of "compiled" reasoning, and FR is used to verify if the hypothesis makes causal sense.

4.7.1.1.1. Generating the Malfunction Causal Chain

The organization of a functional representation gives both forward and backward reasoning capability, i.e., it can trace from the hypothesized malfunction to observations (forward), or it can trace from observations to the hypothesized malfunction (backward). This section describes an algorithm which demonstrates the forward simulation potential[6].

Specifically, if device A is malfunctioning, then devices which use device A (say devices B and C) have a high probability of malfunctioning as well. Similarly, devices which use B and C may malfunction, etc. The malfunction causal chain is achieved through the following algorithm which has been condensed to illustrate main points.

1. Set Observations to the symptoms to be explained, and set HypothesisList to the set of diagnostic hypotheses. Initialize HypothesisObject to an individual diagnostic hypothesis in this set (diagnosed hypotheses and their relationship to observations are considered individually).

2. Identify the function that the HypothesisObject names as missing. From the FR of the device, find all functions which make use of this function, call this set PossibleMalfunctions.

3. For each element in PossibleMalfunctions (call the specific function PossMal) consider the significance of the effect of HypothesisObject on the function.

if no effect on PossMal then remove from PossibleMalfunctions --- HypothesisObject is not causing future problems. Consider the next element in PossibleMalfunctions.

else maintain (Malfunction -> Malfunction) explanation chain; HypothesisObject is now known to cause a malfunction to PossMal. Specifically HypothesisObject -> PossMal is appended to chain. Note that this step will ultimately place any potential malfunctions in a malfunction chain, including those which are in the set of Observations. Continue.

4. Check the states in the causal process description of the affected PossibleMalfunction. Would noncompletion of these states explain any symptom(s) in Observations?

if yes, append to ExplainedSymptoms and print the chain which led to this symptom. Complete the malfunction explanation chain by continuing.

5. Set HypothesisObject to PossMal.

6. Repeat process from step 2 until all symptoms are in ExplainedSymptoms or the top level causal process description of the device has been reached.

7. The Process from step 1 is repeated until all elements of HypothesisList have been considered.

Step 2 is easily accomplished through the component hierarchy of the functional representation (see example to come soon). Steps 3 and 4 are more intricate and involve knowledge of function type (such as whether it is `To-Make`, `To-Prevent`, etc.) and the achievement of the intended causal processes.

For example, in Step 3, to determine the effects of a malfunction on other functions, one must consider the possible consequences of malfunctioning components. In general, the malfunction of a component in a device can cause one or more of the following three consequences:

> NOT Function: expected results of the function will not be present. Given that the malfunction is not producing the expected results within the causal process, what states in those causal processes will not occur? And, will the lack of this functionality cause the malfunctions of functions in which the malfunctioning component was used?
>
> Parameter Out-of-Range: expected results of the function are affected, but function is still accomplished to a limited degree. Sometimes components may be considered malfunctioning yet can still perform the function (or value of some substance parameter) to the extent needed for future use.
>
> New Behaviors: the malfunction results in behaviors and states which were not those intended for normal functioning.

The determination of whether a proposed malfunction can explain a symptom, step 4 in the explanation algorithm, can be established by a number of means. The following is a non-exhaustive list:

1. Check each state in the causal process description where the malfunctioning component is used to see if there is a direct match between a symptom and *not* achieving an expected state.

2. Check to see if the function which is malfunctioning has an explicit malfunction causal process description and if the symptom is included therein.

3. Check to see if side effects of the function's causal process description refer to the variables involving the symptoms.

4. Check each state in the malfunction causal process description and its provided clause to see if expected states point to general concepts or generic classes of behavior (such as leak, flow, continuity) and if the symptom pertains to or is explained by such concepts.

4.7.1.1.2. Representation of a Chemical Processing Plant

This section provides the output for an example explanation in the domain of chemical processing plants (CPP). The hierarchy in Figure 6 shows a partial representation of the functional components with their intended functions (functions are specified under component names). The top level function, *produce.acid*, is achieved by the causal process *oxidation* shown in Figure 7. The function hierarchy is generated from the CPD's for the FR of the plant. For example, the CPP uses the functional components LiquidFeedSystem, AirFeedSystem, TransferSystem, etc. in the process *oxidation* which represents the causal chain used to achieve the function *produce.acid*; the TransferSystem uses the functional components, AirFeedSystem, MixingSystem, etc., in its causal process to achieve the function *extraction*, and so on.

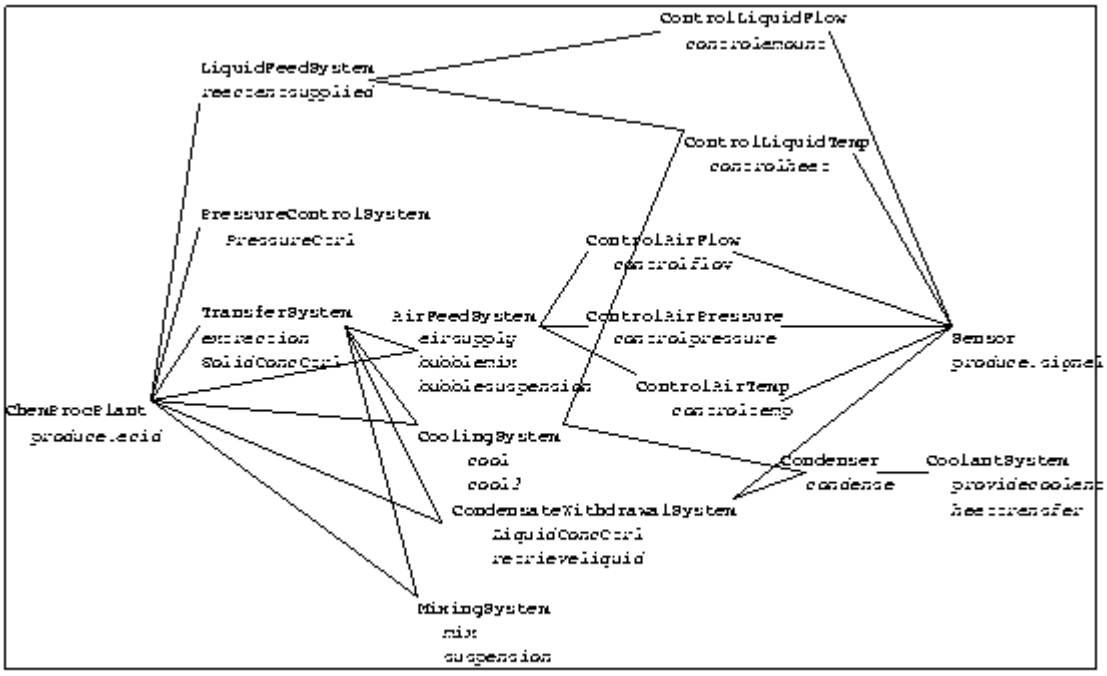

Figure 6. Partial functional hierarchy of the chemical processing plant

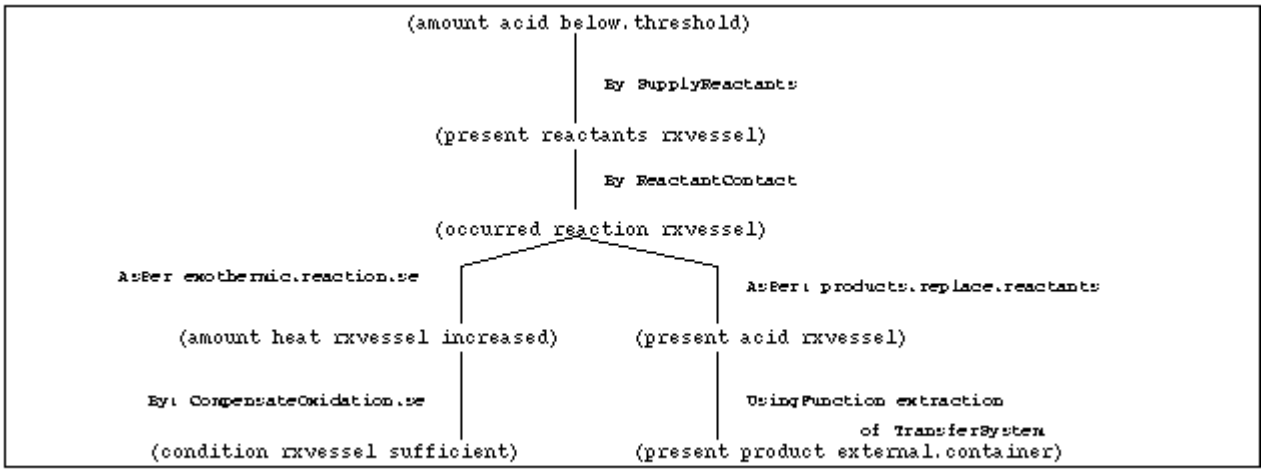

Figure 7. Causal process *oxidation* used by function *produce.acid*

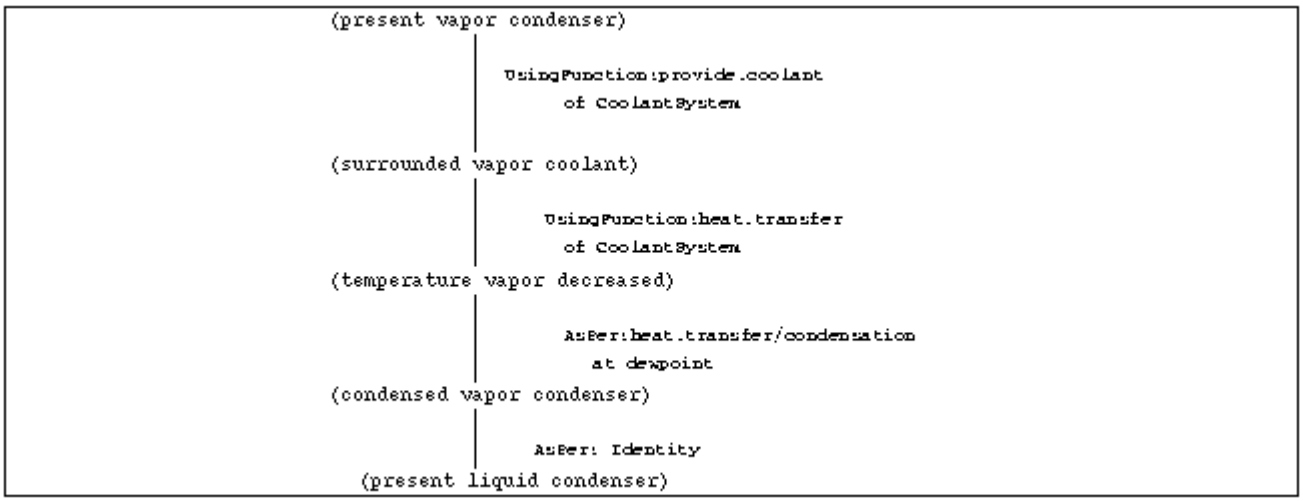

Figure 8. *Remove-heat* function of *Condenser.*

The Problem

The Coolant System (identified at the right of Figure 6) is used to provide coolant water to a Condenser so that it can be used to transfer heat from the vapor in the Condenser (see Figure 8). Suppose the coolant water has been completely cut off. A diagnostic system has concluded that a malfunction of the function *provide.coolant* of the Coolant System explains the symptoms of NOT (present product external.container) and NOT (temperature rxvessel at.threshold). Specifically, the HypothesisObject is *provide.coolant* of Coolant System and the observations to be explained are {NOT (present product external.container), NOT (temperature rxvessel at.threshold) }. The system produces three casual stories.

Causal Story 1: Generation of Causal Connections

The causal process SupplyReactants uses the functions *retrieveliquid* and *LiquidConcCtrl*, in addition to the

LiquidFeedSystem and AirFeedSystem. The explanation system generates the following:

```
The symptom NOT (present product external.container)

is explained by the following chain:

NOT provide.coolant causes malfunction in condense causing malfunction in
retrieveliquid causing malfunction in LiquidConcCtrl causing problems in behavior
SupplyReactants which is used in behavior oxidation and indicates malfunction of the
top level function and results in

NOT (present product external.container)
```

The idea here is that if the required amount of reactants is not available, the product is not produced as desired and thus can not be retrieved. The explanation system generates this chain by using the following information: *Provide.coolant* caused a malfunction in *condense* because it caused a failure in *condense*'s behavior. A malfunction in *condense* caused a malfunction in *retrieveliquid* because its achievement was required to attain the desired CPD for *retrieveliquid*. *Retrieveliquid* caused a malfunction in *LiquidConcCtrl* because it was needed to provide the preconditions for *LiquidConcCtrl* and it preceded the use of *LiquidConcCtrl* in the behavior SupplyReactants. SupplyReactants was used in the causal process Oxidation, (see Figure 7), to achieve the state (present reactants rxvessel). This state was necessary for the completion of the CPD and thus non-achievement here denotes non-achievement of further states in the CPD, particularly NOT (present product external.container).

Causal Story 2: The Use of Side Effect Inspection

The explanation system continues and finds a causal connection for the second symptom, NOT(temperature rxvessel at.threshold).

```
The symptom NOT (temperature rxvessel at.threshold)

is explained by the following chain:

            NOT provide.coolant causes malfunction in condense

causing problems in behavior removeheat of function cool.
```

Since *cool* is not a top level function of the chemical processing plant, the trace continues until all consequences are determined.

```
The symptom NOT (temperature rxvessel at.threshold)

is explained by the following chain:

            NOT provide.coolant causes malfunction in condense

causing malfunction in cool causing problems in behavior
compensate.oxidation.se, a notable side effect behavior used in oxidation
and indicates

                  NOT (temperature rxvessel at.threshold)
```

Notice that this explanation identifies that the symptom was observed in a *side effect* behavior (compensation for

effects of the reaction) rather than a behavior of the main functionality (production of acid).

Causal Story 3: Using Subfunction Connections for Causal Focus

A final statement is made when the system has inspected all pertinent causal chains. The final causal path is achieved via causal connections obtained specifically through the knowledge of subfunctions. In its specification, the function *extraction* has a `Provided` clause which specifies that the solid acid slurry must have the proper consistency so that flow through the extraction tube is possible. The function *SolidConcCtrl* is present in this device for the sole purpose of producing these conditions for *extraction*.

The purpose of *SolidConcCtrl* is to keep the solid suspended and the concentration in the reaction vessel at the proper consistency. In the CondensateWithdrawalSystem, the *retrieveliquid* function uses the Condenser to retrieve the condensate from the vapor produced. The *MixtureLevelCtrl* function then uses a feedback controller to maintain the flow and thus the desired amount of liquid in the reaction vessel --- which ensures that the acid slurry has the proper consistency. If the liquid is not retrievable, then obviously the condensate flow cannot be controlled and consistency of the acid in the vessel is not maintained. The explanation system provides this explanatory story as follows:

```
One function affected by provide.coolant is SolidConcCtrl which is a
necessary subfunction of extraction. The symptom NOT (present product
external.container) is explained by the following chain: NOT
provide.coolant causes malfunction in condense causing malfunction in
retrieveliquid causing malfunction in MixtureLevelCtrl causing malfunction
in SolidConcCtrl causing malfunction in extraction causing malfunction in
produce.acid causing

NOT (present product external.container)
```

Discussion

The intrinsic limitations of a functional representation for explanation arise from its intrinsic limitations for simulation. The representation uses prepackaged causal process descriptions which are organized around the expected functions of a device. Simulations of malfunctioning devices are thus limited to statements of what expectations are "not" occurring.

This limitation effects the capabilities for explanation in two significant ways. First, the functional representation is not capable of generating causal stories of malfunctions which interact unless the device representation has this interaction explicitly represented. Similar problems regarding the interactions of malfunctions arise in diagnosis (Sticklen, et al, 1985) Secondly, "new" CPD's, i.e., CPD's which are not those intended for normal functioning but which arise due to a change in device structure, could potentially lead to symptoms which cannot be explained using the functional representation. Additional research focusing on how a functional organization might be used to determine these new behavioral sequences, in addition to how conventional methods of qualitative reasoning may be integrated, is needed.

4.7.2. Parametric Simulation

Let us say we have a CPD associated with a function:

sI ->... si -> ... -> sF

where sI is the initial state and sF is the functional state desired. This CPD packages a simulation sequence. If all the conditions for sI and the next transition are satisfied, the partial state corresponding to its successor will be achieved and similarly for the next transition and so on. The nodes in CPD and the conditions on the links may be represented parametrically. In that case, the CPD becomes a parametrized family of behaviors. Specific behaviors can then be derived for particular situations represented by a particular set of parameters.

DeJongh (1991) uses the idea of parametric simulation for reasoning about *classes* of functional systems in the domain of blood typing and testing. The members of the class follow the same causal process but they differ in the various parameters associated with the values of the variables. For example how the function of preventing spontaneous agglutination is achieved by the red cells is represented parametrically. Similarly the class of test procedures is also represented as a class of devices that uses specific causal processes to achieve the test functions.

Sticklen and his group [Pegah, et al, 1993] have been most active in developing the techniques for the use of FR for simulation. Each CPD is a prepackaged simulation sequence of states of interest in some context, and in that sense following a CPD is a limited form of simulation. However, the very fact that the CPD's are organized with respect to goals of interest provides advantages in many simulation problems. In particular, when the FR represents a class of devices (or a class of contexts and initial conditions), then the CPD and the hierarchical organization implicit in FR enable efficient situation-specific and goal-directed simulation.

Suppose a device has a number of functions, each with different `Provided` clauses, i.e., the different functions are invoked under different conditions. Note that a similar situation might prevail with respect to the components of the device, i.e., each of them might have different functions with different `Provided` clauses. Let us also assume that the FR is represented parametrically, i.e., the partial states and the various conditions in the CPD's have parameters in them, corresponding to structural parameters that can be instantiated to correspond to specific versions of the device.

Sticklen describes an algorithm which, in outline, works as follows:

1. Given the operating conditions and the parameters, the simulator starts with top-level functions, and identifies all the functions whose `Provided` clauses are satisfied. Any missing functions or altered functions can also be used to initialize the simulation. If a function uses another function as part of one of its links, the calling function is included, but not the called function.

2. Using the specific parameters and the conditions, the top level CPD's of the functions that are chosen in 1 above are instantiated. Note that this is not merely copying the CPD. Some of the paths in the CPD may not be taken because the conditions for the links are not satisfied, either because of the conditions in the `Provided` clauses of functions that are called or because the functions are otherwise not available. The instantiation of the CPD for the specific situation is a *Particularized State Diagram* (PSD).

3. Any link in the CPD with a By-CPD or a By-Function annotation can be expanded by accessing the referenced CPD or the CPD of the referenced function, and instantiating it by using the values of the parameters and eliminating inapplicable paths. This now produces a PSD at another level of detail. This can be continued to as much detail as necessary. As PSD's are built, the values of the state variables are appropriately updated.

Thus the simulation itself is available in a hierarchical fashion to many levels of detail, mirroring the functional decomposition of the device. Pegah, et al (1993) describe the simulation of the fuel transport system of an F-18 aircraft using this technique. Sticklen, Kamel and Bond (1991) also describe integrating qualitative and

quantitative simulations in an FR framework. Basically, the transitions in the CPD help the simulation system focus on which specific quantitative equations are to be used in the actual computation. See also Sun ad Sticklen (1990).

Toth (1993) has recently shown how to use FR as an organizing principle for intelligent simulation where the computational effort in simulation can be allocated in response to the goals. For example, in a structural engineering problem, we might be interested in knowing if the stress in some element is going to be more than maximum allowed. Engineers reasoning about such problems typically combine quantitative and qualitative techniques. By using purely qualitative techniques, they might rule out many parts of the structure, but zero in on a member with a high likelihood of excessive stress. They might then use numerical techniques for a precise computation of stress in restricted parts of the original structure. Toth shows how the CPD of devices can be organized to include pointers to both qualitative and quantitative methods for computing the state variables involved in the causal transitions.

4.7.3. Diagnostic reasoning

Davis (1984), Fink and Lusth (1987) and Steels (1989) are among a number of authors who have used functional notions explicitly in diagnostic reasoning, but work in this vein does not include the relationship between functions and causal processes that our work FR elaborates. In the use of FR in diagnosis, the causal process description plays an important role.

4.7.3.1. Simple Use of FR for Diagnosis

The first use to which FR was put was in diagnostic reasoning. In Sembugamoorthy and Chandrasekaran (1986) a diagnostic knowledge structure was compiled for an electronic buzzer from its FR. The diagnostic knowledge structure was a malfunction tree, with a set of diagnostic rules for each of the malfunctions. Sticklen (1987) used a similar idea, but his problem was one where the diagnostic knowledge structure was incomplete. He used FR to generate the diagnostic knowledge that was needed for a diagnostic situation. The distinction between the use of FR for generating the complete diagnostic knowledge structure in advance and generating only the fragments of diagnostic knowledge as needed for a problem instance could be described as the distinction between compilation and interpretation, but this distinction is not my focus here. The various issues in using FR for diagnosis are explored in Chandrasekaran, et al (1989), Sticklen and Chandrasekaran (1989), Sticklen, et al (1989), Sticklen and Tufankji (1992) and Sticklen, et al (1993). The central idea in the application of FR for diagnosis can be summarized as follows.

For simplicity, let us first consider a CPD in which each transition has only one annotation. Consider a transition in a CPD of the form

`By-function` *F* `-of-component` *c*

n1----------> *n2*

Suppose the device is in the partial state *n1*, i.e., the device is in a state that satisfies the predicates corresponding to *n1*. Suppose we test the device and observe that the device fails to reach *n2*. What conclusions can we draw? Because the CPD asserts that the device goes from partial state *n1* to *n2* because of the function *F* of component *c*, we can hypothesize that the failure to reach *n2* is due to the component *c* not delivering the function *F*. Corresponding to this transition we can identify a possible malfunction state "Component C not delivering function F." The diagnostic rule "device satisfies *n1* but not *n2*," can be used to establish this malfunction mode of the

device. If the annotation had instead been "`By-CPD` *CPD-1*," where *CPD-1* is a specific CPD, we could similarly examine *CPD-1* to see why this transition failed (some transition in *CPD-1* should fail if the transition from *n1* to *n2* failed). Ultimately, we can identify some function of some component that would have to be responsible for the failure of the device to reach n*2*.

There is no malfunction corresponding to a transition with the annotation `By-Domain-Law;` a domain law cannot fail to hold. Of course, the designer's account of the role played by the domain-law could be incorrect, but we are assuming here that the FR itself is correct. How to verify the FR itself is an interesting issue in its own right, but is not the subject of current discussion.

The technique of identifying a component malfunction either directly from the annotation `By-Function` or by recursive application of `By-CPD` leads to a diagnostic tree that will have as tip nodes malfunctions of components or subcomponents. The diagnostic rule for each malfunction will be composed of rules of the form "If the predicates corresponding to node *ni* are true, but those corresponding to *nj* are not true, then.."

What happens when we have more than one annotation, e.g., as in Figure 5 where the transition appeals to more than one function? In this example, the transition can fail because *pipe2* is blocked, its thermal (heat exchange) function fails, or because the conditions in the qualifiers are not satisfied. In this case, the failure of the transition can only identify these as possible malfunctions, but cannot establish them. Additional information will be necessary.

Not all diagnostic knowledge can be derived from design information alone. For example, rank-ordering diagnostic hypotheses in terms of likelihood and pursuing them in the order of most probable to least probable is quite common in diagnostic reasoning. But this ordering requires knowledge of probabilities of failure for components. This information is not derivable from a causal model of how a device works. Additional information in the form of failure rates is needed. Conversely, not all diagnostic knowledge derived from causal models is directly usable, since some variables mentioned in the diagnostic rules generated from causal models may not be directly observable. Additional inference may be required. For example, in medical diagnosis, from an FR of liver functions, one might derive a diagnostic rule, "If bile is generated in the liver but is not delivered to the duodenum, then establish 'blockage of bile duct'." However, "bile in the duodenum" is not directly observable. Additional reasoning about the consequences of bile in the duodenum, perhaps by using FR's of other physiological mechanisms, can result in observable tests which can then be used as diagnostic knowledge.

DeJongh (1991) discusses the use of FR-like simulations of physiological mechanisms (like the "Prevent-agglutination" function of red blood cells) to verify abductive hypotheses in a blood typing system. His work is significant in that both compiled diagnostic problem solving as well as causal simulation using FR is done in a uniform formalism of problem spaces in the Soar (Laird, et al., 1987) framework. This enables him to use Soar's chunking mechanism to transfer the results of causal simulation to the compiled diagnostic knowledge structure.

Debugging proposed designs involves a form of diagnosis. See (Stroulia, et al, 1992) for a discussion of the use of FR in this task.

4.7.3.2. Debugging Computer Programs

Allemang (Allemang, 1990; Allemang and Chandrasekaran, 1991; Allemang, 1991) considered the problem of understanding how computer programs work. Computer programs have components (modules at higher levels, and programming language statements at the lowest level) just as physical devices do. We understand the

functioning of a program by building a process description. Because one of the basic principles behind FR is that how a component achieves its functions is irrelevant to understanding its role in a device, and the component may be replaced by another that provides the same functionality, an FR actually does not represent a single device, but a class of devices that share the same functional breakdown. Allemang proposed that an FR in the programming domain would correspond to several programs, all sharing the same strategy. The proofs of correctness of these programs would share many features in common as well. In that sense FR can be viewed as an organization of the proofs of correctness of this class of programs. The partial states in the CPD of an FR correspond to intermediate formulas that appear in all the proofs. Typical examples of these states of computation include assertions about values of variables and loop invariants.

In this section, I will use some examples from Allemang (1990) and Allemang and Chandrasekaran (1991) to illustrate FR of programs. See also Liver (1993) for application of FR to programs in the domain of telecommunications.

4.7.3.2.1. An Example of a Functional Representation for a Program

Consider the problem of moving the contents of each element of an array between indices k and n - 1 to the next higher position, that is, i [k, n - 1], a[i + 1] = #a[i] (#a denotes the original values of the array a). At the end of the program, a[n] has the value that used to be at a[n - 1].

Three possible solutions to this are shown in Figure 9. The first solution iterates backwards over the relevant fragment of the array, moving the element with the highest index (n - 1) into place first, leaving room for the next to last element, and so on. The second solution attempts to move the element with the least index (k) first, but is buggy, since this clobbers the contents of the next array location. The final example is a corrected version of the second case, in which some auxiliary variables have been introduced to take care of the value clobbering problem.

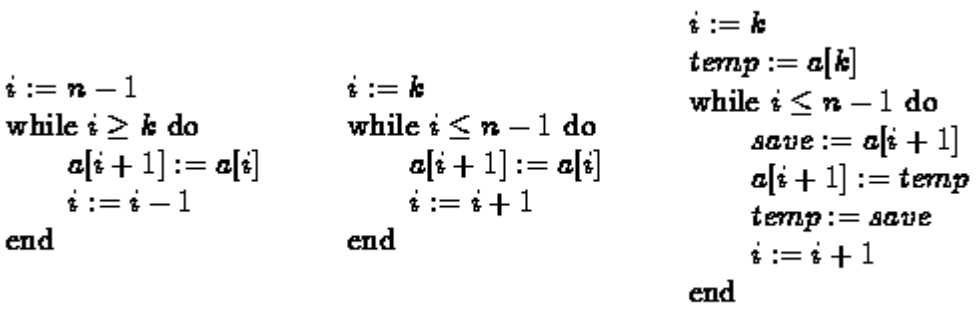

Figure 9. Three solutions to the shift problem, including one incorrect solution (center)

We have some knowledge about a large range of solutions to this assignment, including the three solutions given above. We know that the solutions must treat data in a conservative way, that is, data must not be overwritten. These solutions must move the data within one data structure, rather than construct a new one. They will iterate through the set, treating each element individually. How can we take advantage of such knowledge to help to recognize the correctness of the actual code? We begin by representing them with the same FR.

All three of these programs cover the set of relevant indices of the array with the index i ; for each element covered by i they move the current element up one place in the array. These two operations are coordinated by

the overall structure of the loop. Notice that the programs differ in the choices they make for each of these functions; the first program counts *down* the set while the others count *up*; the first two simply move the value to the appropriate place in the array, while the third uses a more complex swapping solution. This suggests three devices in the functional representation; the *index*, the *mover*, and the *loop*.

The role of the *index* in this problem is to cover the part of the array that is of interest. It does this by 1) starting somewhere in the set, 2) moving from one item in the set to another, and 3) checking when it has covered the entire set. This suggests three functions for the index variable. These, along with some samples of code that could support these functions, are shown in Figure 10. Two options for the index are specified; one each for moving up and down the set. We will use the notation U(i) to refer to the part of [k...n - 1] visited so far by the index; for ascending index, U(i) = [k...i - 1], for descending index, U(i) = [i + 1....n - 1] (by convention, U(i) never contains i).

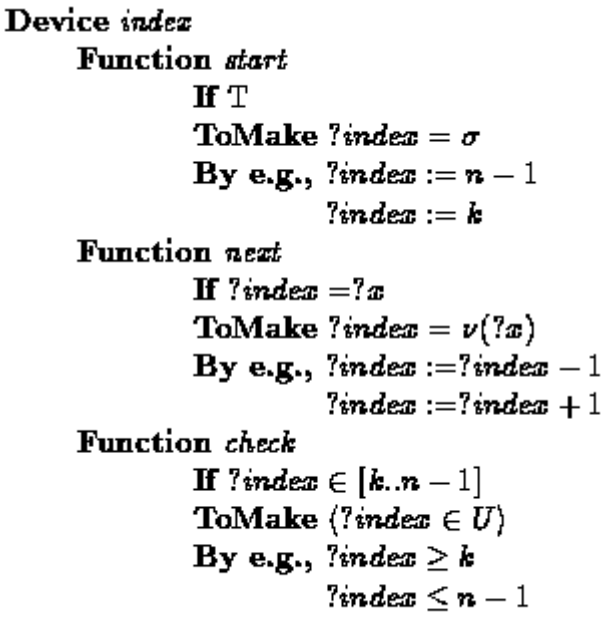

Figure 10.

The role of the *mover* in this program is to guarantee that if the previously visited cells in the array have been moved correctly, then all the cells, including the current cell, are moved correctly. It has one function to do this, which in Figure 11 is called *move* . In the figure, refers to the predicate on subsets of [k...n - 1] that all cells in a subset have been appropriately moved, while no others have been moved. For use in a correctness proof, is defined by

(S) = j [k...n - 1] \ S, a[j] = #a[j]

j s, a[j + 1] = #a[j] (1)

Device *mover*
  Function *move*
    If $\Gamma(U(?index)) \wedge (?index \in U)$
    ToMake $\Gamma(U(\nu(?index))) \wedge (\textbf{empty } a[?index])$
    By e.g., $a[?index + 1] := a[?index]$
    Provided $(\textbf{empty } a[?index])$

Figure 11.

Because of the possibility of a mismatch between some choices that might be made for the *index* (e.g., ascending) and this definition for the mover, it is not possible to justify the part of the proof corresponding to this function, i.e.,

{(U(?index))} a[?index+1] := a[?index] {(U((?index)))} (2)

before knowing the details of the actual program. Thus, in order for this FR to be consistent, it would be necessary to place (2) as the proviso for function *move*. In Figure 11, a weaker proviso is used which does not entail the consistency of the FR, but shows the capability of the system to use plausible explanations in place of actual proofs.

Finally, the overall loop has the job of coordinating the functions of these two components. It is the loop that sequences the various checks and assignments from the *mover* and the *index* so that the overall task of moving the array fragment is done. A functional representation for the loop is shown in Figure 12.

4.7.3.2.2. Debugging Using Device Understanding

Allemang also discussed the use of FR for program debugging. The debugger matches intentions to programs, and then resorts to a weak theorem prover only when the match cannot be completed. It uses provisos to simplify the job of the theorem prover as much as possible.

When presented with the first program in Figure 9, the debugger determines which choices of navigator and collector match the actual program. We omit the details of this unification process. Since the structure of the FR matches this program quite well, the loop invariant for this induction proof is already known, and the only work for the theorem prover is to verify that the proviso (empty a[i]) is satisfied whenever the line a[i + 1] := a[i] is executed. So the debugger presents the following fragment of a proof:

Presuming that a[n] is empty, the loop initialization tells us that i is n - 1. So, at the start of the loop, a[i + 1] is empty.

From the previous iteration, the line a[i + 1] := a[i] gives us that a[i - 1 + 1] is empty. The line i := i - 1 gives us that a[i + 1] is empty.

Thus, a[i + 1] is empty for the current iteration.

We will skip what the debugger does with the second program, and move to the third one, in which the programmer has introduced three novel lines and two new variables to the problem. The debugger has no

problem recognizing this as a correct program. It finds three lines in the body of the loop where it expected to find just the one line,

a[i + 1] := a[i].

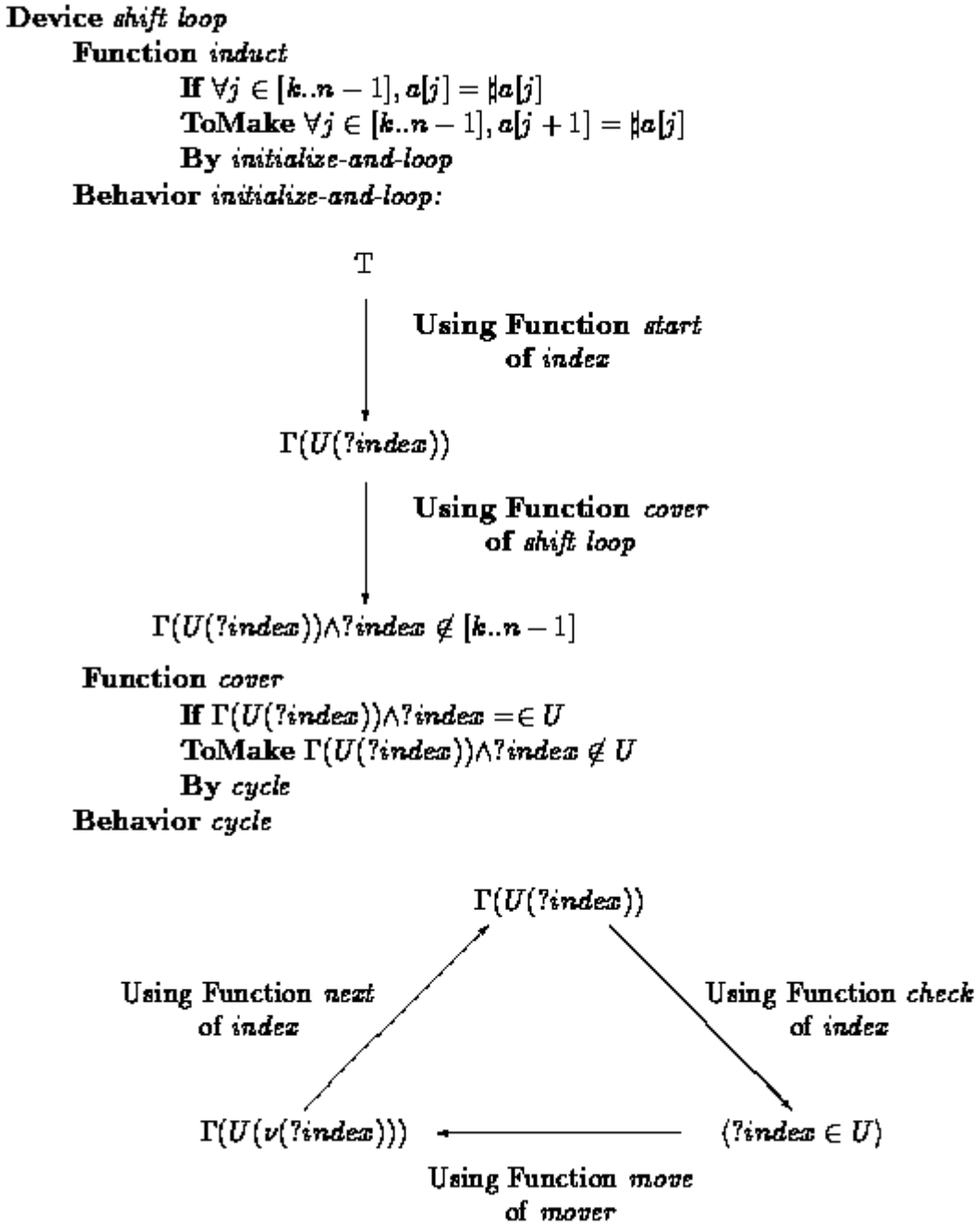

Figure 12.

First, it notices that the second of these three lines might be able to provide the function expected by the missing line, provided that the variable *temp* contains the value that was expected on the right hand side of the assignment, that is, #a[i]. This proviso is treated as any other, and the debugger traces back through the loop to

generate the following proof:

The line *temp* := a[k] gives us that *temp* contains #a[k].

The loop initialization tells us that i is k , so at the start of the loop, *temp* contains a[i].

From the previous iteration,

the line *save* := a[i + 1] gives us that *save* contains #a[i + 1]

the line *temp* := *save* gives us that *temp* contains #a[i + 1]

the line i := i + 1 gives us that *temp* contains #a[i].

Thus, *temp* contains #a[i] for the current iteration.

The separation of functions from the process description allows FR to act like a plan representation for programs (Johnson, 1986); the functions specify pre- and post- conditions. The links in CPD index other functions, based on their pre- and post- conditions, just as subgoals index other plans. Allemang goes on to argue how FR allows for a way of combining the power of the plan-based representation and the traditional programming language semantics to define what he calls a *functional semantics*, which allows a debugger to consult a proof of correctness without having to deal with all the complexities of the traditional programming language semantics.

4.7.4. Device Libraries

The FR framework leads to the prospect of technical domain libraries of generic devices. Device classes at different levels of system description would be represented along with parametrized structural representation and the corresponding CPD's. Specific device representations can be constructed by choosing, instantiating and composing elements from the library. In most cases, the FR for a device would simply be an instantiation of an abstract generic device, but in cases where the design is novel, new CPD's can be composed. This device can be abstracted to a generic device and be made available for future use.

DeJongh (1991) represents classes of devices by parametrized FR descriptions, and specific devices inherit the causal structure but the variables assume the specific values for the particular device. Pegah, et al (1993) report on the use of device libraries for constructing representations. Toth (1993) outlines the construction of such libraries in an object-oriented framework, and uses such libraries extensively in her work on simulation. Josephson (1992) reports on the use of abstract data types as the basis for building such device libraries. The Kritik system of Goel (1992b) uses a library of about 25 designs for case-based design.

4.7.5. Uses in Design

There are a number of design subtasks for which the FR framework is useful (see Chandrasekaran, 1990, for a description of the task structure of design, Goel and Chandrasekaran, 1992 for a more detailed task analysis of case-based design, and Freeman and Newell, 1971, for an early discussion of the role of functional reasoning in design). Iwasaki, et al (1993) describe a design framework in which again FR plays an important role. Chandrasekaran, Goel and Iwasaki (1993) discuss the use of FR for representing causal aspects of a design rationale, i.e., explanations of the design choices. Levi, et al (1993) discuss how FR helps in bridging planning, execution and device implementation in planning systems. In this section, we discuss the use of FR in case-based

design and in design verification.

4.7.5.1. Redesign

The need to redesign can come about in a number of ways. For one thing, it is a subtask of the technique of case-based design. In this technique, when a new design problem comes along, a search is made in memory of problems similar to the current one, the "closest" one is retrieved and the design solution for that problem is redesigned, i.e., modified to fit the current needs. Redesign can also occur when the use environment has changed and it is desired to modify a device so as to deliver slightly different functions. In either case, the goal in the task of redesign is to modify the artifact so that it meets somewhat different functions. If the required changes in function are drastic, then perhaps equally drastic structural alterations will be needed, possibly requiring another design from scratch. However, if the needed changes are small, redesign can be accomplished by relatively simple modifications to the existing structure, perhaps by parametric changes to the components and substances. In this section, we examine the role of FR in the redesign problem, assuming that the required changes are parametric changes to the components.

Redesign has three subtasks: identifying the substructure that requires modification, identifying the modifications that need to be made, and verifying that the changes in fact produce the desired changes in function. Taking the last two subtasks first, deciding on the appropriate modifications requires knowledge outside of the FR for the device. Perhaps, a design library containing FR's of a number of functional assemblies could help. If it is decided that a component or a subsystem of the previous design needs to be changed to reflect a different functionality, an appropriate solution from the library may be available. Regarding verification, Sticklen's use of FR for parametric simulation (Pegah, et al, 1993) is relevant. As we discussed in the earlier section on parametric simulation, he shows how FR can be viewed as a form of compiled simulation, and suggests ways in which FR can incorporate information about the behavior of the device over ranges of component parameters. With this information, it is straightforward to derive device behavior when component parameters are changed.

Let me now get back to the first subtask, viz., identifying the substructures or components that need modification. Use of FR for retrieval and case analysis was first discussed in Goel and Chandrasekaran (1989), and Goel developed the idea in detail in his thesis (Goel, 1989; see also Goel, 1992) where he described Kritik, a system that performs a form of case-based design in the NAC domain. I will describe how Kritik uses FR for case retrieval and analysis. Suppose we want to modify NAC to cool high-acidity sulphuric acid instead of the low-acidity nitric acid. Kritik first compares the desired functions and those actually delivered by the candidate design (NAC). It notes that they differ in (i) the substance to be cooled (*sulfuric acid* instead of *nitric acid*), and (ii) a property of the substance (*high-acidity* instead of *low-acidity*). Since the substance property difference occurs in the function *cool (low-acidity) nitric acid*, Kritik uses this function to access the CPD responsible for it. A fragment of this CPD, the transition from state2 to state3, is shown in Figure 5. Kritik traces through this CPD, checking each state transition in it to determine whether the goal of reducing the substance property difference (*low-acidity high-acidity)* can be achieved by modifying some element in the transition. For example, in the transition *state2 state3* it finds that *pipe2* has an *allow* function but it is restricted to *low-acidity* substances.

Kritik has a typology of modifications to device components: (i) the parameters of a component can be tweaked, (ii) the modality of operation of a component can be changed, and (iii) a component can be replaced by another one. It correspondingly generates the following modification hypotheses: (i) *pipe2* can allow the flow of high-acidity substances in a different parameter setting; (ii) *pipe2* can allow the flow of high-acidity substances in a different mode of operation; and (iii) *pipe2* has to be replaced with some *new-pipe2* which can allow the flow of high-acidity substances. How the choice of modification is made is not directly related to the design rationale,

so we omit a discussion of that issue. The replacement of nitric acid with sulphuric acid is straightforward as is the similar modification needed to handle the difference in the functional specification. See Goel (1991a) for additional discussion of case adaptation.

While Kritik is limited to `local' changes in a design, new members in the Kritik family of systems go beyond this. Stroulia and Goel (1992), for example, show how FR-like representations of generic mechanisms such as cascading can help make certain kinds of non-local modifications to a design. Suppose, for example, a designer wanted to create a device that will be able to cool Nitric Acid by a much larger range than the device illustrated in Figure 2. Suppose also that in addition to the design shown in Figure 2, the designer knows of the generic mechanism of cascading - the replication of a device structure to achieve a larger function. Stroulia and Goel show the cascading mechanism can be represented in the FR language and used to replicate the water pump in Figure 2 to cool Nitric Acid by a larger range.

Note that the cascading mechanism is not only device-independent but also domain-independent. Bhatta and Goel (1993) have studied how the cascading can be learned from the design of one type of device such as a Nitric Acid Cooler, and used to design another type of device, such as an electrical circuit. In their work, the FR of the first device, the Nitric Acid Cooler, guides both the learning of generic mechanisms such as cascading, and the transfer of this knowledge across devices and domains.

4.7.5.2. Design Verification

Lets us consider a designer who has just completed a design, i.e., she has put together a description which specifies what components to use and how they are to be connected. Unless the design was done by a very simple process such as table look-up, the designer is likely to have her own explanation of why she thinks the design would work. In our framework, having such an explanation would correspond to her having a causal story of how the device as designed will meet the functions.

The intended behavior of the device as described in the CPD can be verified by simulating the device behavior based on component descriptions. As we discussed earlier, there are two problems in using component behavioral specifications for device simulation. First, there is a possible gap in levels of abstractions between device-level behaviors we are interested in verifying, and the component-level behavioral descriptions. For example, the language in which the behavior of transistors and resistors is described is that of currents and voltages, but a circuit as a whole might be described functionally as an oscillator or as an adder. We could add a number of abstraction rules, but the simulation needs guidance on which abstraction rules to apply and when. Second, the component models may describe aspects of behavior that may not be relevant for the device-level behaviors of interest. Without guidance from the device-level functional description, the simulation may become quite complex and unwieldy, generating behaviors that do not contribute to design verification. For example, a *pipe* as a component may have two sets of behavioral descriptions, the first involving its capability to support flow and the second regarding its thermal properties. If the device function is only concerned with flow, we would want to use this information to avoid pursuing behavior composition of thermal properties.

The CPD can be used as follows in the design verification task (Iwasaki and Chandrasekaran, 1992). The predicates that appear in the definition of the nodes in the CPD and the functional predicate, say *PF*, are terms that are of interest at the device-level. We first need to define these predicates in terms of objects and predicates that appear in the definition of components. For example, suppose the predicate `Amplification-Level` appears in the description of a node in a CPD, and that the component behaviors are in terms of voltages and currents. We first define the predicate in terms of voltages at the input and output of the relevant components. Once a correspondence has been established between device- and component-level terms, we need to establish

that the partial states corresponding to the CPD nodes occur in the device, and that the transition occurs for the reasons mentioned in the annotation. If the annotation said, "By-Function F of component c", we want to verify that the component did play the indicated causal role in the transition.

We thus have the designer's CPD and the structural description. We can generate a description of device behavior from a description of component behaviors using a simulator that composes the behaviors of the components in the structural description. A number of component description and simulation systems have been described in the AI literature, e.g., Fikes, et al, (1991) and Low and Iwasaki (1993). However, as mentioned earlier, this behavioral description will be in component terms. We can then verify that the CPD is supported by this simulated behavioral description, i.e., the predicates mentioned in the CPD occur in the behavior and in the appropriate causal and temporal relationship.

One can imagine using the simulator in two modes. In one, the simulator is run first and the entire set of state variable values for all relevant discrete instants of time,

$St\{x1,...xj,..xN)$, $t = 1, 2,.., ..T$,

where *St* is the state vector at time *t* and *xj* is the j-th state variable, is generated. (The state vector is simply the set of all the state variables that describe all the components in the device.) This description is a *trajectory* of the behavior of the device. Once this simulation is available, then we can proceed to establish that the CPD is satisfied by the simulation.

A second mode is one where the simulation itself is guided by the CPD. That is, we first verify the initial conditions of an FR are satisfied by the initial values of the state vector. The first transition of the CPD is then used to drive the simulator in relevant directions, i.e., to compute the values of the relevant state variables needed to establish the next node in the CPD and the transition. Once these are verified, the simulation can be guided by what is needed for establishing the next node in the CPD and so on. In order for performing this form of guided simulation, we need to have component-level simulation techniques that can be used in selected directions. Development of these techniques is an area of needed research.

The work reported in Iwasaki and Chandrasekaran (1992) uses component simulation in the first mode to verify the CPD. Let us say that the trajectory of behavior has been generated by the simulator, and that we are looking at a transition from node *ni* to node *nj* in the CPD. Verifying that the two nodes are satisfied by the trajectory involves showing that there are instants *k* and *l*, *k l*, such that *ni* is true in state *Sk* and *nj* is true in state *Sl*. Proving this will make use of the predicates used in the definition of the nodes in the CPD and the abstractions defined between the variables used in these predicates and the component-level variables in the trajectory.

Even if we have verified that the device (in simulation) has gone from node *n1* to *n2*, we can't yet claim that the transition in the CPD has been verified. The reason is that we still need to show that this transition occurred as a result of, or was caused by, the reason mentioned in the annotation. For example, say that the annotation is "By-Function *F* of the component *c*." If component *c* had no role to play in the transition, then it is possible that the component *c* was not needed, and in any case the designer's account of why the device worked would be incorrect.

How to decide if an event causes another is a contentious philosophical issue, but the following weak criterion is sufficient for most purposes: *pi* causes *pj* if *pi* had anything to do with eventually bringing about *pj*, where *pi* and *pj* are predicates in nodes *ni* and *nj*, $j > i$. In Iwasaki and Chandrasekaran (1992) techniques of causal ordering -- originally presented in Iwasaki and Simon (1986) -- are used to show this kind of relationship between *pi* and

pj. Showing that the function of a component causes some aspect of a state to be true requires careful use of the semantics of the type of function.

A slightly different kind of design verification task occurs in the context of incremental modification of a design. Suppose that a designer designs a new device for achieving a desired function F2 by tweaking the design of a known device which delivers a function F1 very similar to F2. The designer may now want to verify whether the proposed design for the new device will result in desired functionality F2. Note that the designer already knows that the old design results in F1. If the designer has access to the FR of the old design, then he may modify the FR to reflect the design tweak, and then simulate the revised FR by forward tracing to determine whether the proposed design will deliver the desired function F2. Kritik (Goel, 1989), which designs new devices in this manner, uses this method for verifying whether the proposed design tweaks help in achieving a desired function.

4.7.6. Representing Problem Solvers as Devices

I earlier described Allemang's work in representing computer programs in the FR framework and using it to reason about errors. An AI problem solving program is of course a specific type of computer program, and thus has a device-like representation. Weintraub (1991), Johnson (1993) and Stroulia and Goel (1993) have built systems which use the FR of the problem solvers as a basis for critiquing the problem solving behavior. Weintraub's system uses this critiquing for credit assignment (actually blame assignment) in learning. Johnson uses it as the basis for student monitoring in a tutorial system for problem solving. Stroulia and Goel use it for self-monitoring, credit assignment, and self-adaptation in the context of a planning system. In all cases the FR is used as an abstract representation of the strategy of the problem solver, i.e., the goal-subgoal structure of the problem solving method. Because the problem solvers involved in all these cases use goals and subgoals explicitly during problem solving, the organization of the problem solver has the same function-subfunction decomposition similar to devices. In Weintraub's work, the transitions of the CPD represent the execution of the method's subtasks. Associated with each transition is a set of error candidate rules. If the problem solving system that is being critiqued fails to make a transition or makes it with erroneous parameters, the credit assignment system uses these rules to make hypotheses about the sources of error in the problem solver. The FR thus provides an abstract road map about the strategy for the critic. Johnson uses the FR of the problem solver in a similar way for monitoring a student's problem solving behavior. Stroulia and Goel use the FR for self-monitoring by a planning system, for generating hypotheses about the causes of error when the planner either fails to produce a plan or the plan fails upon execution, and for modifying the planner using generic repair plans.

There are some interesting differences between the FR's used by Weintraub, Johnson, and Stroulia and Goel in representing problem solvers. Weintraub uses CPD's in FR in the way we have used them for devices and programs so far: the method is a state transition diagram where the nodes are partial states at the appropriate level of description. On the other hand, Johnson wants to model problem solving methods whose behaviors are not completely specified in detail beforehand. Her problem solver is built in the Soar framework (Laird, Newell and Rosenbloom, 1987). In this framework a method is specified abstractly as a set of subgoals and some type of "search-control" knowledge, and out of this information an actual search strategy emerges at runtime in response to the specifics of the problem situation. This kind of flexibility is not normally part of ordinary devices and programs, and hence their FR's have their CPD's completely specified in advance. On the other hand, the goal of the Soar framework is to build flexible problem solvers, i.e., problem solvers which are not committed to a fixed procedure to achieve a goal. Only to a high level strategy is specified and the detailed behavior is determined at run-time. Johnson represents CPD's just by their subgoals and some general knowledge about the constraints on sequencing them. The following example illustrates this.

`Function: To-achieve` B Method B1:

If: Desired R-B Goals: F, G

By: Method B1 or Method B2 Control: F precedes G

Method B2

Goals: H, I, J

Control: No prior constraints

Figure 13. Goals, multiple methods and method selection information

The specification in the figure says that Goal B can be achieved by either of the methods B1 or B2. Method B1 has two subgoals to be achieved in that order, while method B2's subgoals can be achieved in any order depending upon the circumstances. In the case of B2, this abstract representation actually corresponds to 6 distinct sequences and hence is a more compact representation.

Stroulia and Goel's Autognostic system (Stroulia and Goel 1993), uses FR to describe the reasoning process of a robot planner. Autognostic uses the FR model to monitor the planner's problem solving in a manner similar to Johnson's. If the planner fails to produce a plan or if the plan produced by it fails upon execution, then, like in Weintraub's work, Autognostic uses the FR model of the planner for assigning blame and generating hypotheses about the causes of failure. The process of blame assignment however is different. In Weintraub's work, the transitions in the CPD are annotated by associative rules which indicate the likely sources of error. In contrast, Autognostic uses the derivational trace of problem solving in conjunction with the FR model to identify the sources of error. A major aspect of this work is the redesign of the robot planner after the causes of failure have been identified. The FR model provides the vocabulary for indexing repair plans that correspond to different types of failure causes. In addition, the semantics of the FR model enable a modification of the planner in a manner that maintains the consistency of problem solving.

4.7.7. Representation of Scientific Theories

Darden (Darden, 1990; 1991, 1992, Moberg and Josephson, 1990) has used FR to represent scientific theories and to capture some aspects of theory change in science. FR is a natural medium for representation of theories, especially in domains such as biology and geology where the objects of study are causal processes. Debugging a theory is akin to debugging a mechanism.

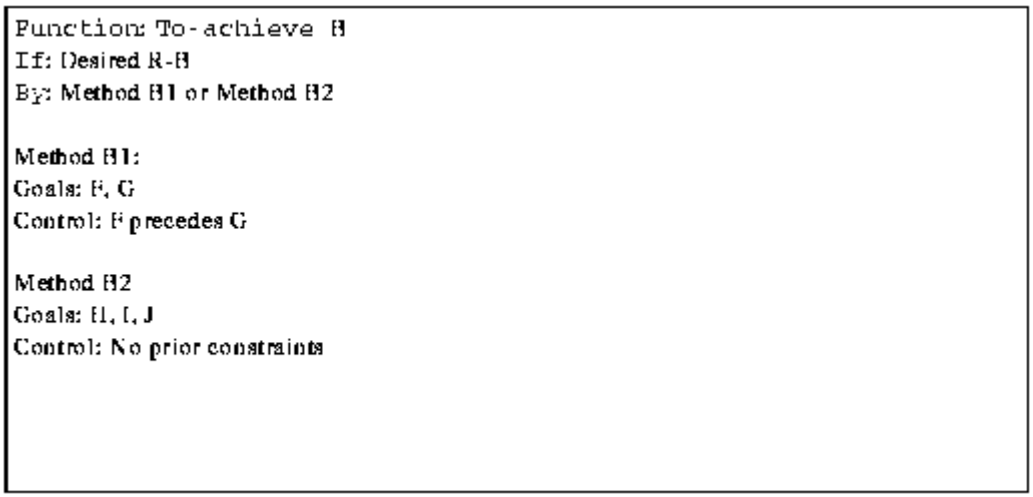

Figure 13

4.8. Generating FR's for New Devices

In all of our discussion so far, we have assumed that the designer or someone else has constructed an FR for the device that is being reasoned about. For diagnosis and simulation, the FR is assumed to be given, and the reasoning mechanisms produce diagnostic knowledge or a simulation of the device. For design verification, the component level simulation mechanism produces a simulation of all the state variables at the component level, but the FR as proposed by the designer is matched with the simulation to see if the CPD in FR is supported by the generated behavior. Human experts, however, are often able to construct functional representations of devices that they have not seen before in the domains in which they have expertise. For example, if a circuit is shown to an electronic specialist, she might, after some hypothesis making and verification, identify the circuit as, say, an amplifier and proceed to explain what components play what role and how she thinks it works as an amplifier.

I remarked earlier that, given a structure, i.e., a set of components connected in some way, the FR of the device as a whole is determined partly by considerations that are outside the device itself. The levels of abstractions of device level variables, i.e., the terms of description, are determined both by the problem solving goals and by what can be supported by the component-level descriptions. Given a circuit with two resistors in parallel, a possible device-level hypothesis is that one of the resistances is providing a current shunt, and an FR that focuses on that aspect can be constructed. Two loads in parallel is another possibility, and an FR that reflects it is also possible. If the goal is to construct an FR as intended by the designer, some form of "best explanation" reasoning may be necessary in general. That is, a form of abductive reasoning in which, using a number of cues, a hypothesis is made about the intended function of the device and a corresponding FR is constructed.

One approach to constructing an FR for a device is to adapt the FR of a similar device, if such information is available in memory. The Kritik system of Goel (1991b) explores this method. Prabhakar and Goel (1992) investigate how the process of adapting the FR of a known device to obtain the FR of a new device can be facilitated by FR-like representations of generic physical processes and mechanisms.

For the more general version of the problem of constructing FR's for novel devices, Thadani and Chandrasekaran (1993) propose a set of techniques, and Thadani (1993) has built a system that does this in the domain of passive electrical circuits. A central idea is that expertise in domains partly consists of *structure-function-CPD templates* at various levels of description. These templates consist of structural skeletons, functions that they can

help achieve, and an abstract CPD that describes how the structural skeleton might achieve that function. Phrases such as "skeleton," "template" and "abstract" in the above are intended to refer to the fact that the templates and the CPD's may refer to classes of objects and behaviors, and also may not have all the details filled in. They are simply organized fragments of knowledge about the domain, fragments embodying pieces of understanding about structural configurations and their relations to various functions.

As a new physical situation is presented, the reasoning proceeds as follows. Templates from memory are matched to the description. All templates that match parts or the whole of the device are retrieved and ranked according to the degree of match. The templates that have the highest degree of match are considered first. The templates are instantiated with as much detailed information from the device as available. Instantiated CPD's may suggest additional hypotheses about the possible roles of other structural parts. These hypotheses may be partially or completely verified by checking the conditions associated with the selected CPD's. The hypotheses may also be verified by simulating the CPD with instantiated parameters, but the current implementation does not use do any simulation. In this process, additional hypotheses might be generated about To use an example, let us say that the original structural description of a circuit is in terms of resistors and voltage sources. Suppose that, as a result of template matching and additional verification, portions are labeled as voltage dividers and current shunts. The hypothesis about the shunt might be accepted or rejected based on the typical values of the resistors in the shunt and whether the resistors in the device satisfy the typical relation. In another example, the CPD for a hypothesized template might have a transition based on some function of a component. We can now check to see if there is structural evidence of the component. If there is, that component structure is so labeled. If there is no evidence, the template is rejected as inapplicable, along with the corresponding hypothesis about the structural fragment. The surviving CPD's are used to generate hypotheses about the rest of the device.

If the predictions are confirmed, that part of the device is labeled with the function from the template. When the cycle of the identifications and verifications is over, we may have a set of alternate hypotheses for parts of device. Each consistent set of interpretations gives rise to labeling the parts of the structure differently. The relabeling for a specific interpretation changes the structural description, raising the level of abstraction in which the structure is described. There will be such a relabeling for each of the alternative sets of interpretations.

This relabeling enables a new round of matchings to be activated, and a new set of structure-function templates to be retrieved. Because of the constraints that come up with each such hypothesis, a number of earlier alternatives would typically not survive, but perhaps other alternatives at this higher level might be constructed. In any case, this process is repeated at each level. From the level of resistors and transistors, configurations at the level of voltage dividers and amplifiers may be hypothesized. At the next level of reasoning, the higher level structural description might enable the reasoner to identify higher level functional units, aided by templates in the knowledge base that relate structures at this level to higher level functional units. Each stage prunes away some of the hypotheses from the previous levels, and might add a few hypotheses, but in general, with a sufficient body of domain knowledge in the form of templates and their constraints, the number of possible interpretations converges to a small number of highly plausible and consistent ones.

The above picture of top-down recognition alternating with bottom-up hypothesis verification is one which I think models our general reasoning about the behavior of the physical world. In this view, we are armed with a large library of skeletal FR's which relate behaviors, structural constraints and CPD's at various levels of abstraction in a given domain. Our knowledge of the world is, as I said at the beginning of the paper, in the form of such causal packages. Use of a large repertoire of FR's, spanning multiple levels of abstraction and goals, gives the agent the capability for very goal-directed and efficient simulation of just the relevant parts of the world for predicting behavior.

4.9. Generalization to Non-Dynamic Causal Structures

Most of the devices that we have considered so far in the paper are what one might call "dynamic" devices, i.e., devices whose function is defined in terms of states and state transitions of the device. The device has an initial state and undergoes a causal process of state changes and reaches states in which the functional specification is satisfied.

The exception was our discussion of *passive devices* which achieve their functions simply by virtue of their structure. We used the example of a flower arrangement serving the function of decoration. Here the notion of causation involves the device's causal effect on human agents, the occupants of the room. As I discussed earlier, presumably one could give a causal process account of how the flower arrangement actually ends up creating a sense of beauty in the perceivers. However, for many such devices, we develop direct mappings from structure to function without involving the causal processes in the user. In any case, the flower arrangement does not itself undergo a causal process.

Another example of a passive function is the structural frame which gives strength to a structure by distributing the loads in its members. Suppose we want to explain how it is able to support a heavy load. Engineers often give a causal account that goes something like:" this member divides the load and transmits each half to these two members, and because of the thickness of the beam the stress is pretty small." This is a causal account, and the phenomena involved are intrinsic to the frame, unlike in the flower arrangement example. But the account is not a description of a dynamic causal process, i.e., the frame is not described as undergoing state changes over time in order to explain its ability to support the load.

Nevertheless, the explanation by the engineer has the syntactic form of a CPD: something in the device causes something else in the device until the function we are interested in (in this case, relatively low value of the stress) is shown to be caused, thus explaining the function. Each of the transitions between causes and effects in the explanation can be further explained in the way we described for CPD's: by appeals to functions of substructures, other such "CPD's," or domain laws. For representation and reasoning, the CPD's play the same role as they do in the case of devices with dynamic state transitions. Toth (1993) in fact constructs FR's and CPD's for mechanical structures and uses them to simulate their properties.

There are examples which are not even causal in the way that the structural frame example is, but still a CPD-like explanation of why it works can be given, and such explanations can be used for predictive problem solving. When we understand the proof of a theorem, we create subproofs which prove various lemmas. We talk about how the assumptions lead to certain conclusions which lead to other conclusions. In the process of explaining how the conclusions lead to other conclusions, we may appeal to lemmas, (which serve the role of functions of components) or to inference rules of logic (domain laws) or to other proofs (other CPD's). Thus it appears that the structure of FR and CPD's captures a general logic of comprehension and explanation, with causal explanations being a special case.

Both in the case of the structural frame and in the case of the mathematical proof, the explanation itself has the structure of a *process*: things are explained one after another, one causing another or one implying another, though of course there is no such sequentiality in the phenomena themselves. In the case of the structural frame, all of the stresses and strains are simultaneously in balance, even though the causal account has an inevitably sequential character. In the case of the proof, all the truths about a mathematical domain are eternally true: one conclusion doesn't *cause* another conclusion, let alone in a sequential way.

So where do the sequentiality of the explanations and their formal similarity to causal processes come from in

these cases? The answer is that these are descriptions that are generated or used by cognitive agents with a highly sequential deliberative architecture. When we reason about the world, we move from conclusion to conclusion (or hypothesis to hypothesis). The knowledge state of the agent changes as the agent traverses through the knowledge space in this manner. In the case of a real world phenomenon with an intrinsic causal process, there is often a mapping from the knowledge state of the agent to the causal state of interest. When a state changes into another state in a device, the agent's knowledge state, as he traces this state transition, changes in a similar way. This ability to make the sequence of knowledge state transitions mirror the causal structure of the world is in fact one of the major sources of the power of thinking in dealing with the world, as Craik pointed out in 1943 (Craik, 1967, is a reprint, with a postscript, of the 1943 edition). The power of goal-directed explanation is not restricted to dynamic causal phenomena, as the examples of the structural frame and mathematical proofs indicate. What the explanations capture is an organization of dependency relations in the domain of interest to arrive at conclusions of interest. Causal state change relations are just one form of such dependency relations.

5. Related Work

There has been quite a bit of work in AI and related areas in trying to understand the relation between function and structure of devices. The following is a representative but by no means exhaustive selection of such articles in the bibliography: Brajnik et al (1991), Chittaro, et al (1993), Gero et al (1992), Hunt and Price (1993), Jordan (1991), Kaindl (1993), Lind (1990), Navinchandra and Sycara (1989), Malin and Leifker (1991), Umeda, et al (1990), and Welch and Dixon (1992). They all build on the intuition that functions are made possible by behaviors and the properties of components (in the structure of the device) make the behaviors happen. Our work is characterized by an emphasis on the representation of causal processes that underlie the functioning of the device. We also emphasize the issues of levels of abstraction, of integrating functional and process description into device-specific packages, and formal representation, and use in a variety of problem solving tasks, of explanatory annotations.

Work like that of Jordan (1991) and Stadlbauer (1991) emphasizes the relationship between shape and function, and issue that we have not been concerned with much in this paper, though of course it is an important one. We have also earlier related our work to that of Hodges (1992) by pointing out that he is attempting to come up with a set of mechanical function primitives. One can think of his work as a content theory of function in a shape-based mechanical design domain where shapes play a role in the transfer of force and motion. Abu Hanna et al (1991) discuss how functional model is not sufficient for diagnosis and additional information is needed, a point we also made in our work on diagnostic systems based on functional models (Sembugamoorthy and Chandrasekaran, 1986). Hunt and Price also make points similar to Abu Hanna, et al, about the need in diagnosis for knowledge beyond a purely functional level. Their device representation uses ideas for the representation of function and structure similar to ours, but does not have the causal process description. They make the point that the CPD of FR could describe a system's working incorrectly (after all it is a theory of how the device works by the designer or the diagnostician) and hence may lead to incorrect diagnosis, so they want to use component descriptions for simulating device behavior. The problems that we have identified regarding levels of abstraction in the description could arise and additional inferences from the component-level behavioral descriptions might be needed. An approach based on integrating the FR representation for focusing diagnostic problem solving, and component-level behavior simulation for deriving new behaviors that are not explicitly mentioned in the CPD's can be profitable. In fact, the work that we describe on design verification shows how the FR view and the component simulation view can be integrated.

Bonnet (1991, 1992), Franke (1991), and Bradshaw and Young (1991) are closest to the kinds of concerns that we have been dealing with in this paper. Bonnet's work is actually built on FR and he makes additional representational suggestions, including representations for what we have called passive functions. Franke focuses

on representing the purpose of a design modification and not that of the device itself. Similar to the work of Iwasaki and Chandrasekaran (1992) on design verification, Franke also matches the description of changes in function against a qualitative simulation of behavior changes from component descriptions.

Bradshaw and Young represent the intended function in a manner quite similar to FR. The most important difference between the FR work and that of Bradshaw and Young and also that of Franke is the central role causal process descriptions play in explaining how a function is achieved. Verification of device design involves not only checking that the function is achieved, but also that the device structure played a causal role in the achievement of the function.

Borchardt (1993) and Doyle (1988) are also relevant though their specific concerns are rather different from ours. Borchardt wants to understand how to go from natural language descriptions of causal processes to a more precise and complete representation of the details of the process. Doyle's approach consists of a set of device models for individual physical mechanisms. His program uses a collection of heuristics for synthesizing, from the mechanism descriptions, a device model. The hypothesized model is checked by checking if the appropriate constraints are satisfied. His specific mechanism representations use constraints between variables, while FR additionally emphasizes representation of causal processes. Except for this difference, the work of Thadani and Chandrasekaran described in the paper in constructing device-level FR's has similarities to Doyle's work.

6. Concluding Remarks

We have reviewed over a decade of work on device understanding from a functional perspective. It should be clear from the review that the research causal and functional representations is just beginning. It might be useful to describe a research agenda for the immediate future. We take this up in the next subsection.

6.1. A Research Agenda

The specific representational elements and organizing principles in the language developed so far have been applied to relatively simple devices and processes. The framework needs to be exercised and expanded by applying it to a larger variety of phenomena and devices. Following is a list of extensions on which work is either being done or needs to be done in the near future. It is likely that many of these problems and issues can be handled within the current ontology, but assuredly some of them will require additional representational and organizational ideas.

i. *Functions that arise largely from shape*. Suppose we want to explain how a gear train works. We certainly can give a CPD whose nodes are simply symbolic predicates such as "tooth A exerts force on tooth B." In fact the work by Hodges (1992) can be viewed as a catalog of the kind of influences in force transfer that physical shapes have on each other. However, in human understanding of this function the shapes of the teeth and how they mesh in transmitting the force play an important role. It is important to extend state representations to include components which are shapes, rather than high level predicates about the effects of shapes. The transitions from shapes to shapes may require appeal to visual or spatial simulation.

ii. *Function-sharing*. Clever designs often have components that are used differently to achieve different functions (Ulrich and Seering, 1988). It is true that we can write an FR for each function, but an integrated view of the role of the component would be missing in such a representation. This brings up a more general problem in FR, where, for each explicitly defined function, we can write a CPD that captures how that is accomplished. We also need a higher level integrative perspective in which these individual functions are seen to be part of a larger function which has a unitary representation.

iii. *Representation of mutually dependent functions*. For example, the car battery system requires that the engine be running regularly to keep the battery charged, while the running of the engine requires that the battery system is functioning normally. We need more experience with FR's that deal with such dependencies.

iv. *Abstract or generic devices*. Part of engineering expertise is knowledge of device frameworks, not simply specific devices. Examples are electrical circuit, voltage divider, regulator, and feedback loop. These frameworks can be instantiated in different ways, but we understand how they work at an abstract level without the instantiation. Building device libraries requires representational and instantiation techniques to be developed for such abstract devices.

v. *Representation of functions that arise from a large number of individual elements.* The interactions of bacteria and white cells can be individually represented (Sticklen, 1987), but, given that there are millions of these entities, it is impossible to reason about them on an individual basis. Gross behavior has to be explained as arising from the behavior of numerous elements without individually representing the elements.

vi. *Problem-specific FR construction*. We have been talking as if there is a fixed, possibly parametrized, FR for each device -- at least for each functional perspective -- which we retrieve from memory and apply as needed for specific tasks. However, it appears to me that, even for devices that we thoroughly understand, we construct versions of FR that are appropriate for the particular problem solving task that we face. For example, depending upon what aspects we expect to be reasoning about, we may have qualitative conditions on state transitions, or we may represent them with great numerical accuracy. We may explain how the lamp's filament produces light by appealing to an equation that relates current to lumens, or we may appeal to a CPD that uses the properties of the filament. Further, for each transition in a CPD, we usually only include some conditions that we think are worth mentioning explicitly, but we are aware of a number of background assumptions which are not stated explicitly. For example, if we are explaining that "liquid input at the top of a sloping pipe" causes "liquid to emerge at the other end," we may annotate it in any number of ways depending upon the tasks that we expect to use it for: "By-Function <conduit> of pipe" is one such annotation. But we also assume that the liquid doesn't evaporate in the meantime, that the pipe's surface is not too absorbent in relation to the amount of liquid, and so on and so forth. These assumptions are not stated explicitly, but are available to us when we need to think of them for debugging a faulty pipe. This kind of goal-driven construction of the FR is a capability that is important to understand.

vii. *Need for formalizing different senses of ports.* We have mostly discussed devices that are constructed by composing components. The components are in turn devices as well, i.e., they have functions which can be analyzed similar to the functions of the device they are part of. In addition, the devices are modeled as having *input ports* and *output ports*. Ports serve two distinct roles which are often conflated. In one role input ports are the places where actions are done by the user that invoke the function of the device, while output ports are the places where the device delivers the function. For example, we put fruits into the mouth of the juicer and out comes the juice. Or, we input a low amplitude signal at the input of an amplifier and out comes the amplified signal. There is also another role for the ports, and that is as the locus of connection with other devices to make additional devices. We can cascade two amplifiers -- the output of the first amplifier is connected to the input of the second amplifier -- to make an amplifier of higher amplification factor.

In many devices it happens that the "input-output" and "locus-of-connection" roles go together naturally. However, this is in general not the case. For example, when we build a lamp circuit by connecting a bulb with a voltage source and a switch, the input port for invocation of the function is the switch, and the output port is not electrical at all, but spatial, i.e., the immediate region around the bulb. On the other hand, the device as a whole is

composed out of parts (switch, battery, bulb) which are connected together at various connection ports. These parts are themselves not devices, i.e., they cannot be defined independent of the circuit as having a behavior, let alone a function, unlike the individual amplifiers in the cascaded amplifier example. In the case of the cascaded amplifier, we can trace the behavior of the device by tracing the flow of some entity -- in this case, the signal -- from the input to the output of each component. In the case of the circuit, however, we do not explain the function by starting from the positive terminal of the battery and tracing the flow of electricity through each of the resistors and connecting wires. If there is a break in the circuit, we don't say, "..electricity starts at the positive terminal, moves across the two resistors, and then, oops, it can't go any further because of the break." The reason we don't do this is that electricity is not modeled as flowing until the whole circuit is complete. At that point, we model current as flowing through each of the parts. The notion of component ports as places for connections to make the device is separated from that as places for delivering or invoking functions.

In the case of the lamp circuit, the circuit as a whole can be composed with other components to make a new device. For example, the light from the lamp circuit may be detected by a photo-diode in another circuit which might activate a switch. In this case, the input-output for the lamp circuit, viewed as signal flow, would be the switch and the region around the bulb. The locus of connection is the region around the bulb as well, since the photo-diode will be positioned there. The output of the device as a whole is the output of the diode circuit, within which presumably a switch is activated. On the other hand, each of the circuits have components and loci of connections which are quite different from the ports for the signal flow perspective.

We need to formalize the representation of components, ports and devices so that the more general sense of devices is captured, or at least the formalization supports the distinctions that I just described.

viii. *Functions which involve time.* We have discussed examples where the predicates have to satisfy some temporal relations. For example, the `To-Maintain` function is defined in terms of certain predicates being always true. We also discussed dynamic state abstractions where a repeating sequence of states was defined as a new state at a higher level of abstraction, say "oscillating." But we have not discussed examples where the predicates involve specific quantitative temporal relations between state variables. For example, the function of a sawtooth generator is to generate output over time with specified relations over the values at different times. We need to exercise the FR framework in devices of this type. Further, the work of Rieger and Grinberg (1978) in identifying different types of temporal constraints in transitions needs to be integrated with the FR work.

ix. *Multiple and redundant causal paths in CPD.* There are devices where a function is achieved by different parallel processes, providing redundancy. There are several versions of this type of parallelism. In one, the function is a quantitative one, e.g., so many units of x is to be produced. A number of different causal processes, using different subsystems, would each contribute some amount to the functional requirement. Another version is similar, but if any of the processes fail, the remaining ones pick up the slack. This requires interesting feedback. Such mechanisms are common in biology.

6.2. Logic of Understanding

To what degree do computer programs which contain functional representations of devices and make use of them to solve problems really understand the devices they are reasoning about? Whether symbolic representations of this type alone can give computers understanding, even when they perform reasoning feats that correspond to humans', is currently a topic of heated philosophical debate in circles concerned with the foundations of AI and cognitive science. The minimal claim I would like to make is that the FR framework is an attempt to capture the *logical structure* of understanding of certain types of causal processes in the world. The FR displays the elements that participate in the causal process, and highlights the relationships between them that the understander

believes to exist. As the work I have described has demonstrated, possession of the logical properties enables the representation to be the basis for many different problem solving activities.

6.3. Rich Variety of Reasoning Phenomena

I have surveyed a body of work built over the last decade on reasoning about artifacts, their functions and their causal processes that underlie the functions. The work is based on the assumption that there is a continuity between reasoning in common sense domains and technical fields, that technical experts and ordinary people share ontologies and cognitive organizing principles for the generic task of modeling the world as a causal phenomenon, predicting behavior and synthesizing artifacts. I have emphasized that the real power of intelligent behavior arises from the agent's ability to organize reasoning and computational resources in a goal-directed way, and qualitativeness of reasoning is just one aspect of it. FR investigates issues about how causal knowledge is indexed and packaged functionally. I have also indicated how the work reported is complementary to the work on qualitative device models. Together they provide an integration of top-down and bottom-up reasoning techniques for efficient goal-directed reasoning. The research area is rich in topics for further expansion and exploitation.

The FR framework is just part of a larger framework that is hinted at in Section 2, one in which reasoning, action and perception are seen as an integrated whole. AI has been too closely associated with just a "reasoning" paradigm, and achieving various goals in the physical world is an area of research which can serve as a great arena for developing an integrated AI perspective.

7. Acknowledgments

I acknowledge the contributions of my colleagues and collaborators in FR research over the years, V. Sembugamoorthy, Jon Sticklen, Jack Smith, William Bond, Ashok Goel, Dean Allemang, Anne Keuneke, Lindley Darden, John Josephson, Mike Weintraub, Kathy Johnson, Matt DeJongh, Sunil Thadani, Susan Toth, Yumi Iwasaki, Marcos Vescovi and Richard Fikes. I thank Ashok Goel, Sunil Thadani, Dean Allemang and Marshall Yovits for reading a draft of this paper and providing a number of useful suggestions for improvement. I also thank Marshall Yovits for inviting me to write this article and being patient with me until I completed it. My research reported here has been supported over the years by DARPA (contract F30602-85-C-0010 monitored by Rome Air Development Center and contract F-49620-89-C-0110 monitored by Air Force Office of Scientific Research), Air Force Office of Scientific Research (grant 89-0250), McDonnell Douglas and BP America's graduate fellowships.

8. Bibliography

(Abu-Hanna et al., 1991) Abu-Hanna, A., Benjamins, V.R., and Jansweijer, W.N.H., Device understanding and modeling for diagnosis. *IEEE Expert*, Vol. 6, no. 2, 1991, pp. 26-32.

(Allemang, 1990) Allemang, D., Understanding Programs as Devices. Ph. D. thesis, The Ohio State University, 1990.

(Allemang, 1991) Allemang, D., Using functional models in automatic debugging. *IEEE Expert*, Vol. 6, no. 6, 1991, pp. 13-18.

(Allemang and Chandrasekaran, 1991) Allemang, Dean and Chandrasekaran, B., Functional representation and program debugging, In *Proceedings of the 6th Annual Knowledge-Based Software Engineering Conference*,

IEEE Computer Society Press, 1991, pp. 136-152.

(Bhatta and Goel, 1993) Sambasiva Bhatta and Ashok Goel. Learning Generic Mechanisms from Experiences for Analogical Reasoning. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, 1993, pp. 237-242.

(Bonnet, 1991) Bonnet, Jean-Charles, Functional Representations: a Support for Enriched Reasoning Capabilities. Stanford University, Knowledge Systems Laboratory, Technical Report KSL 91-58, 1991.

(Bonnet, 1992) Bonnet, Jean-Charles, Towards a formal representation of device functionality. Stanford University, Knowledge Systems Laboratory, Technical Report KSL 92-54, 1992.

(Borchardt, 1993) Borchardt, Gary C., Causal reconstruction. Massachusetts Institute of Technology, AI Lab, memo 1403, 1993.

(Bradshaw and Young, 1991) Bradshaw, J.A., and Young, R.M., Evaluating design using knowledge of purpose and knowledge structure. *IEEE Expert*, Vol. 6, no. 2, 1991, pp. 33-40.

(Brajnik et al., 1991) Brajnik, G., Chittaro, L., Tasso, C., and Toppano, E., Representation and use of teleological knowledge in the multi-modeling approach. *Trends in Artificial Intelligence,* E. Ardizzone, S. Gaglio, and F. Sorbello, editors, Berlin: Springer Verlag, 1991, pp. 167-176.

(Brand et al., 1992) Brand, M., Birnbaum, L., and Cooper, P., Seeing is believing: Why vision needs semantics. In *Proceedings of the Fourteenth Meeting of the Cognitive Science Society*, pp. 720-725, 1992.

(Bylander, 1988) Bylander, T., A Critique of Qualitative Simulation from a Consolidation Viewpoint. *IEEE Trans. Systems, Man and Cybernetics*, Vol. 18, no. 2, 1988, pp. 252-263.

(Bylander, 1990) Bylander, T., Some causal models are deeper than others. *Artificial Intelligence in Medicine*, 2(3), 1990, pp. 123-128..

(Chandrasekaran and Mittal, 1983) Chandrasekaran, B. and Mittal, S., Deep versus compiled knowledge approaches to diagnostic problem solving. *Int. J. Man-Machine Studies*, pp. 425-436, 1983.

(Chandrasekaran et al., 1989) Chandrasekaran, B., Smith, Jack W. Jr., and Sticklen, Jon, 'Deep' Models and their relation to diagnosis. *Artificial Intelligence in Medicine*, Vol. 1, no. 1, 1989, pp. 29 -40.

(Chandrasekaran, 1990) Chandrasekaran, B., Design Problem Solving: A Task Analysis. *AI Magazine*, Vol. 11, no. 4, 1990, pp. 57-71.

(Chandrasekaran and Narayanan, 1990) Chandrasekaran, B. and Narayanan, H. R. Integrating imagery and visual representations. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, Boston, MA, 1990, pp. 670-678.

(Chandrasekaran, 1991) B. Chandrasekaran, Models Vs rules, deep versus compiled, content versus form: Some distinctions in knowledge systems research. *IEEE Expert*, Vol. 6, no. 2, April 1991, pp. 75-79.

(Chandrasekaran, 1992) B. Chandrasekaran, QP is more than SPQR and dynamical systems theory: Response

to Sacks and Doyle. *Computational Intelligence*, 8(2), 1992, pp. 216-222.

(Chandrasekaran et al., 1993) Chandrasekaran, B., Goel, Ashok, and Iwasaki, Yumi. Functional representation as design rationale. *IEEE Computer, Special Issue on Concurrent Engineering*, January 1993, pp. 48-56

(Chapman, 1990) Chapman, D., Vision, Instruction and Action. MIT AI Lab, Cambridge, MA, 1990.

(Chittaro et al., 1993a) Chittaro, L., Tasso, C., and Toppano, E., Putting Functional Knowledge on Firmer Ground. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence-93 Workshop Program, pp. 23-30.

(Chittaro et al., 1993b) Chittaro, L., Guida, G., Tasso, C., and Toppano, E., Functional and teleological knowledge in the multi-modeling approach for reasoning about physical systems: a case study in diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, to appear.

(Craik, 1967) Craik, Kenneth, *The Nature of Explanation*. Cambridge University Press, Cambridge, London, UK and New York, NY, 1967.

(Darden, 1990) Darden, Lindley, Diagnosing and fixing faults in theories. *Computational Models of Scientific Discovery and Theory Formation*, J. Shrager and P. Langley, eds., Hillsdale, NJ: Lawrence Erlbaum, 1990, pp. 319-346.

(Darden, 1991) Darden, L., *Theory Change in Science Strategies from Mendelian Genetics*. Oxford University Press, Oxford, 1991.

(Darden, 1992) Darden, L., Strategies for Anomaly Resolution, .in R. Giere (ed.), *Cognitive Models of Science*, Minnesota Studies in the Philosophy of Science, Vol. 15. Minneapolis: University of Minnesota Press, pp. 251-273.

(Davis, 1984) Davis, R., Diagnostic Reasoning Based on Structure and Function. *Artificial Intelligence*, Vol. 24, 1984, pp. 347-410.

(DeJongh, 1991) DeJongh, M., Integrating the Use of Device Models with Abductive Problem Solving, Ph. D thesis, Department of Computer & Information Science, The Ohio State University, 1991.

(de Kleer and Brown, 1983) de Kleer, J., and Brown, J.S., Assumptions and Ambiguities in Mechanistic Mental Models. *Mental Models*, D. Gentner and A. Stevens, eds., Hillsdale, NJ: Lawrence Erlbaum, 1983, pp. 155-190.

(de Kleer and Brown, 1984) de Kleer, J. and Brown, J.S., A Qualitative Physics Based on Confluences. *Artificial Intelligence*, Vol. 24, 1984, pp. 7-83.

(de Kleer, 1985) de Kleer, J. , How Circuits Work. *Qualitative Reasoning about Physical Systems*, D. G. Bobrow, MIT Press, 1985

(Di Manzo et al., 1989) Di Manzo, M., Trucco, E., Giunchiglia, F., and Ricci, F., FUR: Understanding Functional Reasoning. *International Journal of Intelligent Systems*, Vol. 4, 1989, pp. 431-457.

(Doyle, 1988) Doyle, R.J., Hypothesizing device mechanisms: Opening up the black box. Ph. D. dissertation AI-TR 107, MIT Artificial Intelligence Laboratory, 1988.

(Faltings, 1992) Faltings, B., A symbolic approach to qualitative kinematics. *Artificial Intelligence,* Vol. 56, no. 2-3, 1002, pp. 139-170.

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

(Fink and Lusth, 1987) Fink, P.K., and Lusth, J.C., Expert systems and diagnostic expertise in the mechanical and electrical domains. *IEEE Transactions on Systems, Man and Cybernetics,* SMC-17(3), pp 340-349.

(Forbus, 1984) Forbus, Kenneth D., Qualitative Process Theory. *Artificial Intelligence,* Vol. 24, 1984, pp. 85-168.

(Forbus, 1988) Forbus, K., Qualitative Physics: Past, Present and Future. *Exploring Artificial Intelligence*, H. Shrobe, ed., San Mateo, CA: Morgan Kauffman, 1988, pp. 239-296.

(Franke, 1991) Franke, David W., Deriving and using descriptions of purpose. *IEEE Expert*, Vol. 6, no. 2, April 1991, pp. 41-47.

(Freeman and Newell, 1971) Freeman, P., and Newell, A., A model for functional reasoning in design. In *Proceedings of the Second International Conference on Artificial Intelligence (IJCAI-71)*, London, England, September 1971.

(Gero et al., 1992) Gero, J.S., Tham, K.W., and Lee, H.S., Behaviour: a link between function and structure in design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron, and H. Yoshikawa, editors, Amsterdam, Netherlands: North-Holland, 1992, pp. 193-225.

(Goel, 1989) Goel, A.K., Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph. D. thesis, The Ohio State University, LAIR, 1989.

(Goel and Chandrasekaran, 1989) Goel, A.K., and Chandrasekaran, B., Functional Representation of Designs and Redesign Problem Solving. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, August 20-25, 1989, pp. 1388-1394, Los Altos, CA: Morgan Kaufmann

(Goel, 1991a) Goel, A.K., A Model-Based Approach to Case Adaptation. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* Chicago, August 7-10, 1991, pp. 143-148, Hillsdale, NJ: Lawrence Erlbaum.

(Goel, 1991b) Goel, A.K., Model Revision: A Theory of Incremental Model Learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, Chicago, June 27-29, 1991, pp. 605-609, Los Altos, CA: Morgan Kaufmann.

(Goel, 1992) Goel, A.K., Representation of Design Functions in Experience-Based Design. *Intelligent Computer Aided Design*, D. Brown, M. Waldron, and H. Yoshikawa, eds. Amsterdam, Netherlands: North-Holland, 1992, pp. 283-308.

(Goel and Chandrasekaran, 1992) Goel, A. K., and Chandrasekaran, B., Case-Based Design: A Task Analysis. In *Artificial Intelligence Approaches to Engineering Design*, Volume II:, *Innovative Design*, C. Tong and D. Sriram (editors), pp. 165-184, San Diego: Academic Press, 1992.

(Hayes, 1979) Hayes, P., The Naive Physics Manifesto. *Expert Systems in the Micro-Electronic Age*, D. Mitchie, ed., Edinburgh: Edinburgh University Press, 1979, pp.

(Hodges, 1992) Hodges, J., Naive Mechanics: A Computational Model of Device Use and Function in Design Improvisation. *IEEE Expert*, Vol. 7, no. 1, pp. 14-27.

(Hunt and Price, 1993) Hunt, J.E., and Price, C.J., Integrating Functional Models and Structural Domain Models for Diagnostic Applications. To appear in *Second Generation Expert systems*, J.M. David, J.P. Krivine, and R. Simmons, editors, 1993, Springer Verlag.

(Iwasaki and Chandrasekaran, 1992) Iwasaki, Yumi and Chandrasekaran, B., Design verification through function- and behavior-oriented representations: Bridging the gap between function and behavior. *Artificial Intelligence in Design '92*, John S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 597-616.

(Iwasaki et al., 1993) Iwasaki, Y., Fikes, R., Vescovi, M., and Chandrasekaran, B., How things are intended to work: Capturing functional knowledge in device design. In *Proceedings of the 13th International Join Conference of Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, 1993, pp. 1516-1522.

(Johnson, 1986) Johnson, W. Lewis, Intention-Based Diagnosis of Novice Programming Errors. *Research Notes in Artificial Intelligence*, Los Altos, CA: Morgan Kaufmann, 1986.

(Johnson, 1993) Johnson, Kathy Price, Exploiting a Functional Model of Problem Solving for Error Detection in Tutoring. Ph. D Thesis, Department of Computer & information Science, The Ohio State University, Summer, 1993.

(Jordan, 1991) Jordan, D.S., The Role of Physical Properties in Understanding the Functionality of Objects, Ph. D thesis, Stanford University, 1991.

(Josephson, 1993) Josephson, John R., The Functional Representation Language FR as a Family of Data Types, The Ohio State University, Laboratory for Artificial Intelligence Research, Columbus, OH, tech report, 1993.

(Kaindl, 1993) Kaindl, H., Distinguishing Between Functional and Behavioral Models. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence, 1993 Workshop Program, pp. 50-52.

(Keuneke and Allemang, 1988) Keuneke, A., and Allemang, D., Understanding devices: Representing dynamic states. Technical Report, The Ohio State University, Laboratory for Artificial Intelligence Research, Columbus, OH, 1988.

(Keuneke, 1989) Keuneke, A., Machine Understanding of Devices: Causal Explanation of Diagnostic Conclusions. Ph. D thesis, The Ohio State University, 1989.

(Keuneke and Allemang, 1989) Keuneke, A., and Allemang, D., Exploring the `No-Function-In-Structure' Principle. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, 1989, pp. 79-89.

(Keuneke, 1991) Keuneke, A., Device Representation: The Significance of Functional Knowledge. *IEEE Expert*, Vol. 6, no. 2, pp. 22-25.

(Kuipers, 1986) Kuipers, B., Qualitative Simulation. *Artificial Intelligence*, Vol. 29, 1986, p. 289-388.

(Laird, et al., 1987) Laird, J.E., Newell, A., and Rosenbloom, P.S., SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1-64, 1987.

(Levi, et al., 1993) Levi, K. R., Moberg, D., Miller, C. A., and Rose, F., Multilevel causal process modeling: Bridging the plan, execution, and device implementation gaps. In *Proceedings of the 1993 Conference on Applications of AI: Knowledge-Based Systems in Aerospace and Industry*, Orlando, FL, Bellingham, WA: SPIE - The International Society for Optical Engineering, pp. 240-250.

(Lind, 1990) Lind, M., Representing Goals and Functions of Complex Systems - An Introduction to Multilevel Flow Modeling. Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark, 1990.

(Liver, 1993) Liver, B., Working-around Faulty Communication Procedures Using Functional Models. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence, 1993 Workshop Program, pp. 95-101.

(Low and Iwasaki, 1993) Low, C.M., and Iwasaki, Y., Device modeling environment: an interactive environment for modeling device behavior. *Intelligent Systems Engineering*, Vol. 1, no. 2, 1993, pp. 115-145.

(Malin and Leifker, 1991) Malin, Janet T. and Leifker, Daniel B., Functional modeling with goal-oriented activities for analysis of effects and failures on functions and operations, *Telematics and Informatics*, 8(4), 1991, pp. 353-364.

(Moberg and Josephson, 1990) Moberg, Dale and Josephson, John, Diagnosing and fixing faults in theories, Appendix A: An Implementation Note, in J. Shrager and P. Langley (eds.), Computational Models of Scientific Discovery and Theory Formation, San Mateo, California: Morgan Kaufmann, pp. 347-353.

(Murray, 1988) Murray, William R., Automatic Program Debugging for Intelligent Tutoring Systems. *Research Notes in Artificial Intelligence*, Los Altos, CA: Morgan Kaufmann, 1988.

(Narayanan and Chandrasekaran, 1991) Narayanan, Hari N., and Chandrasekaran, B., Reasoning visually about spatial interactions, In *Proceedings of the 12th International Joint Conference on Artificial Intelligence* , Sydney, Australia, August 1991, pp. 360-365. Mountain View, CA: Morgan Kaufmann.

(Navinchandra and Sycara, 1989) Navinchandra, D., and Sycara, K., Integrating case-based reasoning and qualitative reasoning in design. J. Gero, ed. *AI in Design*, U.K., 1989.

(Patil et al., 1981) Patil, R.S., Szolovits, P., and Schwartz, W.B., Causal Understanding of Patient Illness in Medical Diagnosis. In *Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, 1981, pp. 893-899.

(Pearl, 1986) Pearl, J., Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, Vol. 29,

no. 3, 1986, pp. 241-288.

(Pearl, 1991) Pearl, J. and T. S. Verma, A theory of inferred causation. In *Proceedings of the International Conference on Knowledge Representation*, 1991.

(Pegah et al., 1993) Pegah, M., Sticklen, J., Bond, W., Functional Representation and Reasoning About the F/A-18 Aircraft Fuel System. *IEEE Expert*, Vol. 8, no. 2, 1993, pp. 65-71.

(Pittges et al., 1993) Pittges, J., Eiselt, K., Goel, A.K., Garza, A.G.S., Mahesh, K., and Peterson, J., Representation and Use of Function in Natural Language Understanding. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence, 1993 Workshop Program, pp. 114-120.

(Prabhakar and Goel, 1992), Prabhakar, S. and Goel, A. K., Integrating Case-Based and Model-Based Reasoning for Creative Design: Constraint Discovery, Model Revision and Case Composition. In *Proceedings of the Second International Conference on Computational Models of Creative Design*, Heron Island, Australia, December 1992; Kluwer Academic Press, pp. $$.

(Rieger and Grinberg, 1978a) Rieger, C. and Grinberg, M., A system of cause-effect representation and simulation for computer-aided design. *Artificial Intelligence and Pattern Recognition in Computer-Aided Design*, Latombe, editor, North Holland, 1978, pp. 299-333.

(Rivlin et al., 1993) Rivlin, E., Rosenfeld, A., and Perlis, Recognition of Object Functionality in Goal-Directed Robotics. In *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence, 1993 Workshop Program, pp. 126-130.

(Sacks and Doyle, 1992) Sacks, Elisha P. and Doyle, Jon, Prolegomena to any future qualitative physics. *Computational Intelligence*, Cambridge, MA and Oxford, UK: Blackwell Publishers, Vol. 8, no. 2, May 1992, pp. 187-209.

(Sembugamoorthy and Chandrasekaran, 1986) Sembugamoorthy, V., and Chandrasekaran, B. Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. *Experience, Memory, and Learning*, J. Kolodner and C. Reisbeck, editors, Lawrence Erlbaum Associates, 1986, pp. 47-73.

(Simon, 1991) Simon, H.A., Nonmonotonic Reasoning and Causation: Comment. *Cognitive Science*, Vol. 15, no. 2, 1991, pp. 293-300.

(Stadlbauer, 1991) Stadlbauer, H., Functional skeletons: From specification to design. INFA-report, Institute for Flexible Automation, Technical University of Vienna, Austria, 1991.

(Stark and Bowyer, 1991) Stark, L., and Bowyer, K.W., Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Trans. Pattern Analysis and Machine intelligence*, Vol. 13, 1991, pp. 1097-1104.

(Steels, 1989) Steels, L., Diagnosis with a function-fault model. *Applied Artificial Intelligence Journal*, Vol. 3, 1989.

(Sticklen, 1987) Sticklen, J.H., MDX2, an integrated medical diagnostic system, Ph. D. dissertation, The Ohio State University, 1987.

(Sticklen and Chandrasekaran, 1989) Sticklen, Jon and Chandrasekaran, B., Integrating classification-based compiled level reasoning with function-based deep level reasoning. *Applied Artificial Intelligence*, 1989, Vol. 3, no. 2-3, pp. 275-304.

(Sticklen et al., 1989) Sticklen, J., Chandrasekaran, B., and Bond, W.E., Distributed causal reasoning. *Knowledge Acquisition*, 1989, Vol. 1, 139-162.

(Sticklen, Kamel, and Bond, 1991) Sticklen, J., Kamel, A., & Bond, W. E. (1991). Integrating Quantitative and Qualitative Computations in a Functional Framework. *Engineering Applications of Artificial Intelligence*, 4(1), 1991, pp. 1-10.

(Sticklen and Tufankji, 1992) Sticklen, J., & Tufankji, R., Utilizing a Functional Approach for Modeling Biological Systems. *Mathematical and Computer Modeling*, 16, 145-160.

(Sticklen et al., 1993). Sticklen, J., McDowell, J.K., Hawkins, R., Hill, T., and Boyer, R., Troubleshooting Based on a Functional Device Representation: Diagnosing Faults in the External Active Thermal Control System of Space Station FREEDOM. U. Fayad, ed., *SPIE Applications of Artificial Intelligence XI: Knowledge-based Systems in Aerospace and Industry*, Orlando, FL, SPIE, 1993, pp. $$.

(Stroulia et al., 1992) Stroulia, E., Shankar, M., Goel, A.K., and Penberthy, L., A Model-Based Approach to Blame Assignment in Design. In *Proceedings of the Second International Conference on AI in Design*., Kluwer Academic Press, 1992, pp. 519-538.

(Stroulia and Goel, 1992) Stroulia, E., and Goel, A.K., Generic Teleological Mechanisms and their Use in Case Adaptation. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 1992, pp. 319-324.

(Stroulia and Goel, 1993) Stroulia, E., and Goel, A.K., Using functional models of problem solving to learn from failure. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence, 1993 Workshop Program, pp. 157-163.

(Sun and Sticklen, 1990) Sun, J., and Sticklen, J., Steps toward tractable envisionment via a functional approach. In *The Second AAAI Workshop on Model Based Reasoning*, American Association for Artificial Intelligence, Boston, pp. 50-55.

(Thadani, 1993) Thadani, Sunil, Understanding function from structure, Ph. D Thesis, Department of Computer & Information Science, The Ohio State University, 1993 (forthcoming).

(Thadani and Chandrasekaran, 1993) Thadani, S., and Chandrasekaran, B., Structure-to-Function Reasoning. *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence 1993 Workshop Program, pp. 164-171.

(Toth, 1993) Toth, Susan, Using Functional Representation for Smart Simulation of Devices, Ph. D Thesis, Department of Computer & information Science, The Ohio State University, Summer, 1993.

(Ulrich and Seering, 1988) Ulrich, K.T., and Seering, W.P., Function Sharing in Mechanical Design. In *Proceedings of AAAI-88*, American Association for Artificial Intelligence, 1988, pp. 450-454.

(Umeda et al., 1990) Umeda, Y., Takeda, H., Tomiyama, T., and Yoshikawa, H., Function, Behavior and Structure. *Applications of Artificial Intelligence in Engineering V, Vol. 1*: *Design*, Computational Mechanics Publications, Southampton, 1990, pp. 177-193.

(Vescovi, et al., 1993) Vescovi, M., Iwasaki, Y., Fikes, R., and Chandrasekaran, B., CFRL: A language for specifying the causal functionality of engineered devices. In *Proceedings of the Eleventh National Conference on AI,* American Association for Artificial Intelligence, July 93, AAAI Press/MIT Press, pp. 626-633.

(Weintraub, 1991) Weintraub, Michael A., An Explanation-Based Approach to Assigning Credit, Ph.D. dissertation, The Ohio State University, Department of Computer and Information Science, Columbus, OH, 1991.

(Weiss et al., 1978) Weiss, S., Kulikowski, C., and Amarel, S., A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, Vol. 11, 1978, pp. 145-172.

(Welch and Dixon, 1992) Welch, R.V., and Dixon, J.R., Representing function, behavior and structure during conceptual design. *Design Theory and Methodology DTM'92*, D.L. Taylor and L.A. Stauffer, eds, American Society of Mechanical Engineers, 1992, pp. 11-18.

(Weld,1986) Weld, D., The Use of Aggregation in Causal Simulation. *Artificial Intelligence*, Vol. 30, no. 1, October, 1986.