Constructing Functional Models of a Device from its Structural Description

Sunil Thadani B. Chandrasekaran
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277
thadani/chandra@cis.ohio-state.edu

Topic: Representation and reasoning
Domain: Electrical circuits
Contribution: This paper identifies a useful and ubiquitous version of structure-to-behavior reasoning problem and proposes a novel solution approach to it.

Abstract

This paper addresses, and proposes a solution for, a version of the device understanding problem. Given the structural description of the device, the system generates hypotheses about its functions and how it achieves them. These function-specific device models (FR models) are constructed at multiple levels of abstraction. The proposed method uses the knowledge of frequently encountered abstract devices in the domain to derive functions and FR models from structure. Using a decomposition-based strategy a device is viewed as consisting of a combination of the instances of the already known abstract devices whose functions and FRs are composed to derive the functions and FRs of the whole device.

1 The problem

The importance of the problem of understanding how devices work cannot be overstated. Given an unfamiliar device one has to first go through the process of understanding the working of the device, that is, what the device does and how it does that, before one can perform tasks such as diagnosing the device, predicting the behavior of the device, and explaining the working of the device. In this paper we are interested in the process of understanding itself, that is, given the structural description, henceforth referred to as *struc-desc*, of a device, how an agent acquires the understanding of the working of the device. More specifically, by "understanding a device" we mean determining the functions of the device and for each function generating a causal account of how the device function arises from the functions of its components.

2 Situating this work

The motivations to address this problem in the field of AI have been quite varied ranging from building more flexible and robust expert systems to understanding the cognitive aspects of common sense reasoning. The version of the problem addressed by most of the earlier work in AI in this area [dK85, For84, Kui86] was that given the *struc-desc* of the device and given an input perturbation, determine the resulting behaviors of the device. And some [dK85, IS86] also addressed the problem of generating an account of how specific behaviors of the device come about from its *struc-desc*. Even though the various approaches proposed by these various researchers differed along certain dimensions, they all shared the following characteristics. *Device models were composed out of context-independent models of the parts. Simulation was proposed as the primary method of going from structure to behavior. The device models and the behavior descriptions generated were all at a single level of abstraction.* Further, *the device models and the behavior descriptions generated were general purpose, in the sense that they were independent of the specific aspects of device behaviors that one might be interested in.* Such an approach to reasoning about devices is useful for, say, discovering the behavioral implications of a new design. But there are several scenarios involving reasoning about devices in which device models with the above characteristics are not adequate for the purpose [VIFC93, Cha94, CGI93, IC92, IFVC93].

For any complex device, most tasks require one to reason about the device at multiple levels of abstraction. For example, to diagnose a complex device, one first rea-

sons about the device at a high level of abstraction to localize the malfunctioning portions, and then reason at a more detailed level with the malfunctioning portions only to further narrow down the fault. Device models at multiple levels of abstraction improve efficiency of such tasks. Further, there are a lot of scenarios in which one is interested only in specific behaviors of the device. In such cases there is no need to simulate the general purpose device model generating several irrelevant device behaviors, because for most complex devices simulating such general purpose models is computationally very expensive. To efficiently generate only the behaviors of interest, we need to be able to generate function-specific device models. And finally, for most complex devices component-level behavior descriptions at a single level of abstraction are usually quite large and do not lend themselves to any efficient strategy for organizing the behavior descriptions of a large number of devices in memory.

Depending on the specific task and the specific domain, different researchers have focused on different aspects of the issues mentioned above. For example, [KC87, Str88] use information about higher order derivatives to eliminate the generation of some spurious behaviors during simulation. [ACP91, NJA91] have proposed a mechanism for using context specific component models. [FF91] builds device models at the right level of abstraction appropriate for answering specific questions about the behavior of the device. [Wel86] proposes an abstraction technique for repeating behaviors. Even though various versions of the issues mentioned in the previous paragraph have been addressed by these researchers, most of them assume the context that makes specific commitments, indicated in italics earlier in this section, to what constitutes understanding of a device.

What distinguishes our work on device understanding is the set of specific commitments made about what constitutes understanding of a device. There are at least three parts to the problem of understanding devices. The first one is determining what constitutes understanding of a device. That is, determining the form and content of the knowledge that constitutes understanding. The second part of the problem is to determine how this understanding can actually be generated starting from the *struc-desc* of the device. The third part is to determine how to use this understanding for various tasks mentioned earlier [IC92, IFVC93, CGI93, All90, VIFC93]. Naturally, the commitments made for the first part will determine the characteristics of the approach used to do the second and third parts.

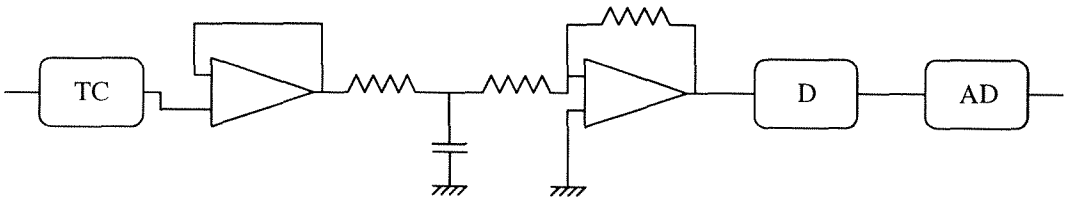
For the first part we have used the proposal on functional representation (FR) [Cha94, IC92, SC88] that makes clear commitments to the content and the form of the knowledge that constitutes understanding of a device, *independent of the specific task or the specific domain.* These commitments have been motivated by larger concerns about how the understanding of a large number of complex devices may be organized in memory and how this understanding may be used to efficiently perform various tasks mentioned earlier. Since the distinguishing characteristics of the output expected, the FR models, provide both the motivation as well as the justification for some of the characteristics of the approach proposed in this paper, a clear understanding of the commitments made in FR would help understand this work better.

3 Characteristics of the output

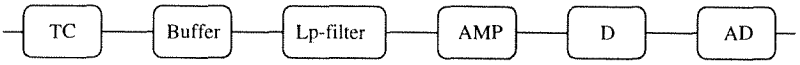
One of the important ideas stated in the proposal on FR is that the causal understanding of a device consists of a set of function-specific causal models of the device, also referred to as the FR models of the device, each of which may be at multiple levels of device-component abstraction.

Consider the circuit, shown in Figure 1(a), of a temperature measuring device. Even though the circuit is given in terms of components such as op-amps, resistors, and capacitors, an expert describing how this circuit works uses several abstractions, such as buffer, low-pass-filter, amplifier, and instrumentation-amplifier. An expert's description of how this device works would be similar to the one that follows. The device shown in Figure 1(a) is understood in terms of the abstract *struc-desc* shown in Figure 1(b) and 1(c). In the *struc-desc* in Figure 1(c) the thermocouple, *TC*, produces a voltage proportional to the input temperature. The instrumentation-amp, *IAMP*, filters and amplifies the voltage generated by *TC*. The analog display, *AD*, produces a deflection proportional to the voltage generated by the *IAMP*. The function of the instrumentation-amp in turn is explained as follows. The input is buffered. The output of the buffer, *B*, is filtered by the *lp-filter* and then amplified by the amplifier *AMP*. The output of the amplifier is further amplified by the driver, *D*, to generate the driver voltage. The functions of the amplifier, buffer, and low-pass filter are in turn explained in terms of the functions of the op-amp, resistors, and capacitors. Such a description of the function of the temperature measuring device is at multiple levels of abstractions and it bridges the descriptions at different levels.

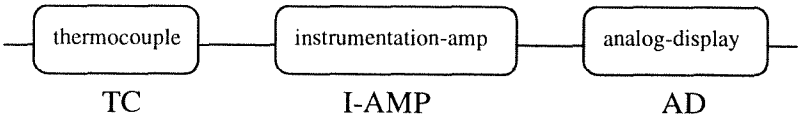
Further, this description is a function-specific description of the temperature measuring device. That is, even though there are several other aspects of this device, only the ones that are relevant to the temperature-measurement function, *tmp-meas-func*, of this device are used in this description. Such function-specific models



(a) Temperature-measuring-device circuit



(b) First level abstraction of temperature-measuring-device circuit



(c) Second level abstraction of temperature-measuring-device circuit

Figure 1: Structural descriptions of temperature measuring device

enable highly focused and computationally efficient reasoning in tasks that use these models.

The description of the temperature measuring device given above is an English-language version of the FR model of the *tmp-meas-func* function of the device shown in Figure 1(a). According to the commitments made in [SC88, Cha94] this is the kind of description an agent is expected to have if the agent claims to understand the *tmp-meas-func* function of the device shown in Figure 1(a). This is also the kind of description we expect our system to generate as an output.

4 Definitions

Some of the terms used here have been used differently by different researchers. For the sake of clarity we give our definitions of these terms here. A *component* is defined by a set of ports and parameters. A component connects and interacts with other components through its *ports*. *Parameters* of a component is a set of parameter-name and parameter-value pairs. For example, a component called *rechargeable-battery* is defined by two ports, the positive and the negative terminals, and parameter names like emf, internal resistance, and charge-capacity each of which has a value associated with it. Associated with a component are *state variables*. The state of a component is defined by the values of its state variables. A *partial state* of a component is a wff of predicates on the state variables of the component. Also associated with a component may be functions. A *function* of a component relates the dependent state variables to the independent state variables of the component. *Independent state variables* of a component are the state variables whose values are assigned or changed by an agent external to the component and *dependent state variables* are the state variables whose values depend on the values of independent state variables.

For each function the component also has a condition, called *provided*. A function of the component is applicable only if the corresponding *provided* condition evaluates to true. These conditions may check anything ranging from the values of certain component parameters to some global variables used to capture the agent's interests. For example, for a rechargeable battery in a closed circuit the charge level of the battery has to be above a certain threshold value, Q_P, for the *deliver-voltage* function of the battery to be applicable. So the *provided* condition for *deliver-voltage* function of the rechargeable battery would be "actual-charge $> Q_T$."

Components may be connected at the ports to form new components. *Struc-desc* of a component specifies the set of sub-components and the relationships between the sub-components. Relationships between the components are defined in terms of the connections between their ports. In the domain of electrical circuits, one

of the relations between the components is *electrically-connected* that corresponds to an electrical connection between *electrical* ports.

Associated with a specific function of a component may be a functional representation (FR), which contains, among other things, a description of how the function arises from the functions of its subcomponents. This description is called causal process description (cpd). A cpd is a directed acyclic graph [IC92, Tha93, Cha94] with nodes corresponding to the partial states of the components and directed edges, also called links, implying transition from one partial state to another partial state. Links of type *using-func-link* are annotated by the function of the component that causes the state transition, and the links of type *as-per-link* are annotated by the name of the domain law that explains the state transition. The *using-func-link* annotation is what enables the FR to relate cpds at various levels of abstractions. For a complete description of FR see [Cha94, IC92, SC88].

Components are organized in component class hierarchies. A component class may inherit ports, parameters, functions, and FRs from its parent class.

5 Getting to the Solution

To describe our approach we shall start backwards from the characteristics of the output expected, and identify the kinds of knowledge required to generate such an output. We shall then propose the form in which these various knowledge types can be acquired and used to perform the task. This is followed by a high level description of the algorithm.

For most non-trivial devices the FR models would have multiple levels of abstraction. FR models use both, structural and behavioral abstractions. For example, component *AMP* in Figure 1(b) abstracts the *struc-desc* in box *A* in Figure 1(a) by hiding the structural details. And the function *amplify-func* associated with the *AMP* abstracts the voltage level description given by the cpd in Figure 6(e), thus hiding the details of the cpd and also introducing a new term "amplify-func" for the input-output behavior described by the corresponding cpd. Since these abstractions are not given as part of the input *struc-desc*, *the system needs knowledge to build structural and behavioral abstractions* to build an FR like the one shown in Figure 6.

The other important characteristic of the output FR models is that these models are function-specific models of the device. To build such a model, the system has to select relevant aspects of the behaviors (subset of state variables and appropriate relations between them) for each abstract and primitive component, aspects that are relevant in describing the specific function of the device. For example, even though there are several state variables and parameters associated with the *struc-desc* of the component *amplifier*, in the context of the function *amplify-func* only a small subset of state variables are of interest. *For each device function the system needs knowledge to get appropriate relations between the relevant subset of state variables.*

We represent the above mentioned knowledge types using structure-function-FR (SFF) templates. Each SFF template represents the understanding of an abstract device in the domain. For example, an expert working with electronic circuits already understands several abstract devices, such as comparator, integrator, adder, amplifier, and lp-filter. So a system reasoning about electronic circuits would be provided with SFFs corresponding to each of these abstract devices. The SFF captures the understanding of an abstract device by associating the functions and FR templates of the device with the abstract *struc-desc* of the device. The abstract *struc-desc* represents a class of *struc-desc*'s. It is defined just like the *struc-desc*, except that the sub-components are specified only by their classes and it may also define a constraint on the values of the sub-component parameters. The corresponding function and FR templates also refer to the parameter names and functions of the sub-components defined in the abstract *struc-desc*. A function (or FR) template also defines a class of functions (or FRs). As described below, the SFF can be used to select specific state variables and relations for specific functions and can also be used to build structural and behavioral abstractions for the abstract device the SFF models.

The system has access to a large number of SFFs that correspond to the abstract devices an expert working in that domain already understands. The primary method used by the system to go from a *struc-desc* to its functions and FRs is by using SFFs. First the system tries to identify the set of SFFs such that the given *struc-desc* is an instance of the abstract *struc-desc* of the SFF. The functions of the matching SFFs are hypothesized to be the functions of the given *struc-desc*. Additional methods are used to verify which functions are indeed applicable to the given *struc-desc*. The templates of the verified functions and the corresponding FRs are instantiated to get the functions and FRs for the given *struc-desc*. This process is shown in Figure 2(a).

For example, the part of the circuit in box *A* in Figure 1(a) matches the abstract *struc-desc* of the SFF for the amplifier. The function templates associated with the amplifier SFF, *amplify-func* and *clipped-amplify-func*, are hypothesized to be the possible functions for the *struc-desc* in box *A*. The system then verifies which of the hypothesized functions are actually applicable for the given structural description in the given context. The

system, as an output, then returns an instance, A', of the abstract device *amplifier* with its abstract *struc-desc* instantiated to the *struc-desc* in box A. And instantiations of the verified and selected function and FR templates are also associated with the instance A'.

Obviously the system cannot be provided with an SFF for every structural description that the system would encounter. So there are going to be *struc-desc*'s for which the system does not have any matching SFFs. To be able to handle a large number of devices the system has to have some sort of a compositional method that enables the system to identify a given *struc-desc* as an instance of a combination of SFFs. The way our system achieves this is by decomposing the given *struc-desc* into parts, analyzing[1] each part separately using the method described in the previous paragraph, and then combining the analyzed parts to form a new *struc-desc* which can be analyzed again. This process is diagrammatically shown in Figure 2(b).

For example, after analyzing the *struc-desc* in box A, the part of the circuit in box A can be replaced by an instance, *AMP*, of the abstract device *amplifier*, with the function *amplifier-func* associated with the instance *AMP*. Similar things are done with other parts of the given circuit. This process results in a new *struc-desc* shown in Figure 1(b) consisting of abstract devices like *AMP*. The decomposition strategy can also be seen as a "structural description transformation" technique, because it takes in a *struc-desc* and generates a new *struc-desc*.

The above process is repeated on the new *struc-desc* generating more abstractions resulting in hierarchical FR models. The *struc-desc* in Figure 1(b) is further abstracted to the structural description in Figure 1(c), which is analyzed by matching it to the *measuring-instrument* SFF. The functions and FRs for the *struc-desc* in Figure 1(c) are obtained by verifying and instantiating the functions and FR templates associated with the *measuring-instrument* SFF.

The two strategies described above constitute the core of the algorithm used by our system to perform the task. In several domains, given the right set of SFFs, a large class of devices can be analyzed by recursive application of these two strategies alone. The method based on the above two strategies is complementary to the simulation based techniques for going from structure to behavior. There are at least two places where simulation can play a role. One is in function verification. Right now we use purely structural criteria to verify if a hypothesized function is in fact a function of the given *struc-desc*. A simulation-based function verification may be used where it may not be possible to verify function based on structural criteria alone. Another place simulation may be used is to determine the behavior of those parts of the given *struc-desc* for which the above method does not work. So if some parts of the given *struc-desc* cannot be analyzed using SFFs, one can use simulation locally to determine their behaviors.

Additional algorithms are used to achieve the following:

- To fetch SFF candidates that would match the given *struc-desc*. To make the search efficient we have organized the SFFs based on functions, the number of sub-components, and special sub-components.
- To match the given *struc-desc* to the abstract structural description of an SFF.
- To verify and select functions that are applicable to the given structural description and that are relevant in the given context.
- To decompose the given *struc-desc*. Various heuristics are used to control the decomposition. For example, the knowledge of the hypothesized function of the given *struc-desc* may be used to suggest decompositions.

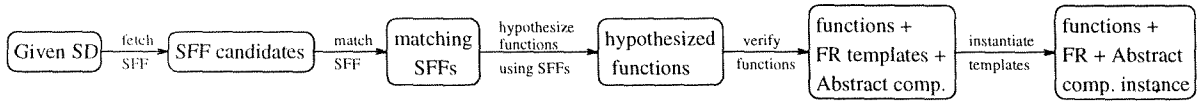
For more details on the algorithm see [Tha94]. As we will show in the following section the resulting control in this algorithm may be top-down or bottom-up depending on the knowledge available. The heuristics used for various steps keep the complexity of the algorithm linear in most cases [Tha94]. Assuming the component models and SFFs to be correct, and assuming the *provided* conditions associated with each function in the SFF correctly verify the hypothesized functions, the algorithm will always produce sound results. Since our decomposition heuristics do not try all the possible decompositions of the given device, the algorithm is not complete with respect to the knowledge provided to it.

6 Example

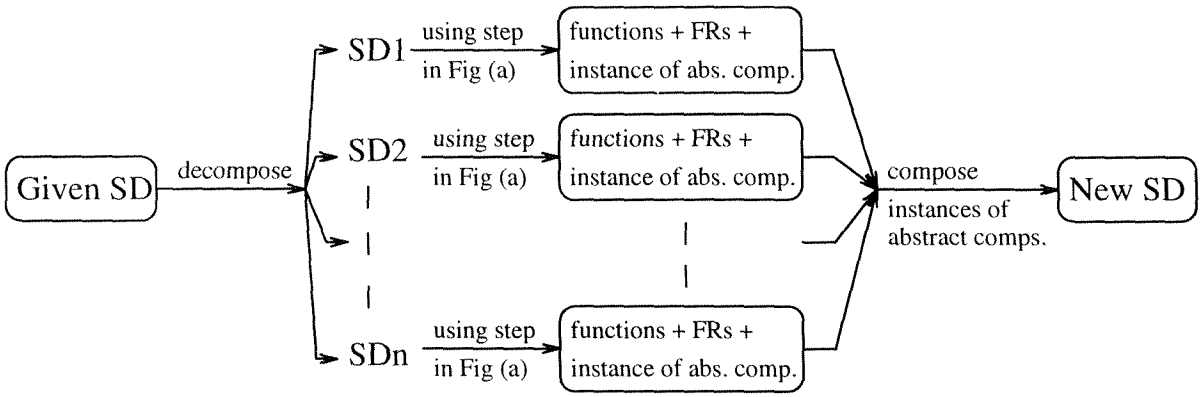
The input to the system is the *struc-desc* shown in Figure 1(a). The system is to determine its functions and FRs. The system uses the models/SFFs of the op-amp, resistor, capacitor, thermocouple, analog-display, instrumentation-amp, and measuring-instrument device, inverter, amplifier, lp-filter, and integrator to analyze this device.

We will show two different executions of this example. The two executions differ in the knowledge provided to the system. In the first execution, as we will see, the system considers a large number of hypotheses to generate the function and the FR of the whole device. In the second execution the system is provided with

[1] We use the phrase "analyzing the device" to refer to the process of understanding the device, that is, generating its functions and FRs



(a) The fetch-match-hypothesize-verify method



(b) The decomposition method

Figure 2: The two primary strategies

the additional knowledge that if the given structural description contains a sub-component of type transducer and a sub-component of type display, then hypothesize the given *struc-desc* to be an instance of the *measuring-instrument*. Once a function has been hypothesized for the given *struc-desc*, the given structural description is analyzed in a top-down fashion resulting in a highly focused reasoning.

Here are the main steps the algorithm goes through for the circuit in Figure 1(a):

- Since the given *struc-desc* does not match any SFF, partition the given circuit. A heuristic is used that produces six partitions corresponding to the *struc-desc* in the box P, Q, R, S, T, and U in Figure 3. Each partition is analyzed separately as shown in Figure 3 to get abstract devices.
- Since no subset of the analyzed partitions matches an SFF, it backtracks one step and tries to analyze the partitions again.
- The partitions are re-analyzed to get the results shown in Figure 4.
- A subset of analyzed partitions matches the SFF of a *sweep-generator*. A new *struc-desc* containing the instance of a *sweep-generator* is generated. The system tries to analyze the new structural description.
- Since there is no matching SFF, once again it backtracks and tries to re-analyze the partitions one more time. Like the first attempt, this also results in a failure.
- The fourth attempt, shown in Figure 5, is finally successful. The final *struc-desc* (having the *instrumentation-amp*) is matched with the *measuring-instrument* SFF and the relevant functions are selected. The verified and selected function and FR templates are instantiated to get the FR shown in Figure 6.

Executing the same example with the additional rule, results in a highly focused problem solving. Once the hypothesized function of the device, *measuring-instrument-func*, is known, the cpds corresponding to the hypothesized function of the device are used to hypothesize functions for the sub-components. With the added rule, the system directly gets to the steps shown in Figure 5.

7 Conclusions

We have proposed a method for generating understanding of devices (FR models) that is function-specific and is at multiple levels of abstraction. Further the understanding at multiple levels is bridged enabling smooth transitions between levels while reasoning. The two main characteristics of the method proposed are that it uses the knowledge of frequently encountered abstract devices in the domain to derive FRs from structure and that

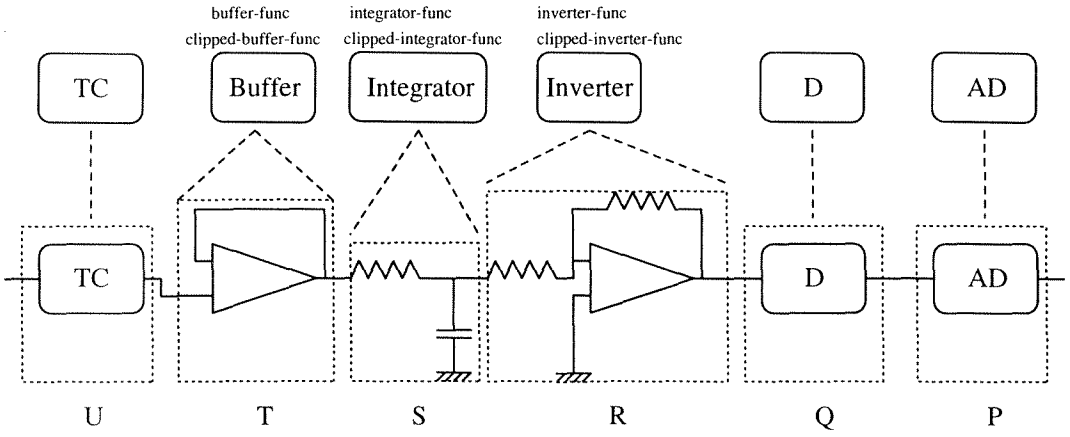


Figure 3: Analyzing the given structural description

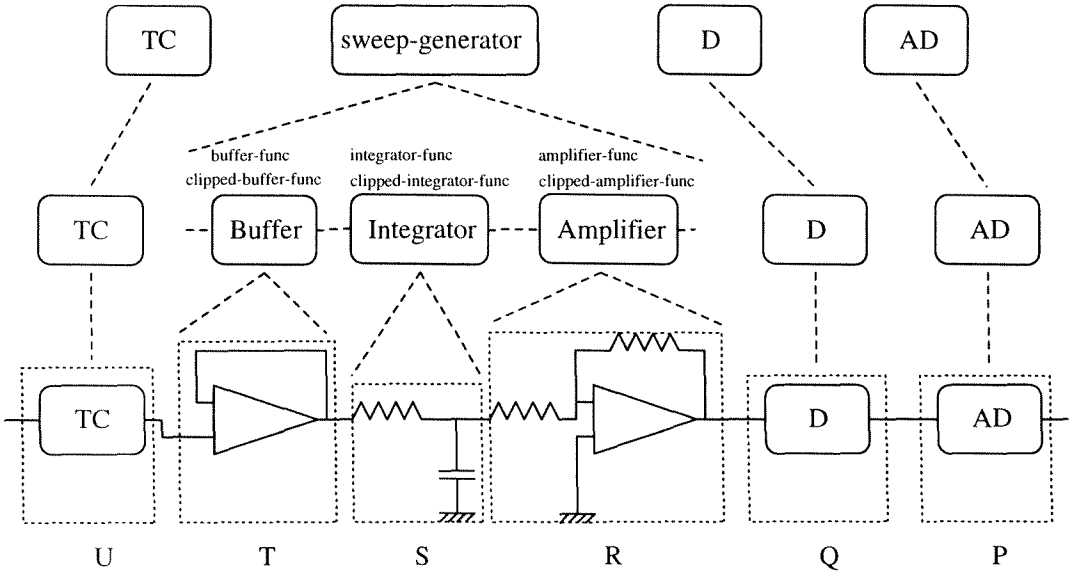


Figure 4: Analyzing the given structural description

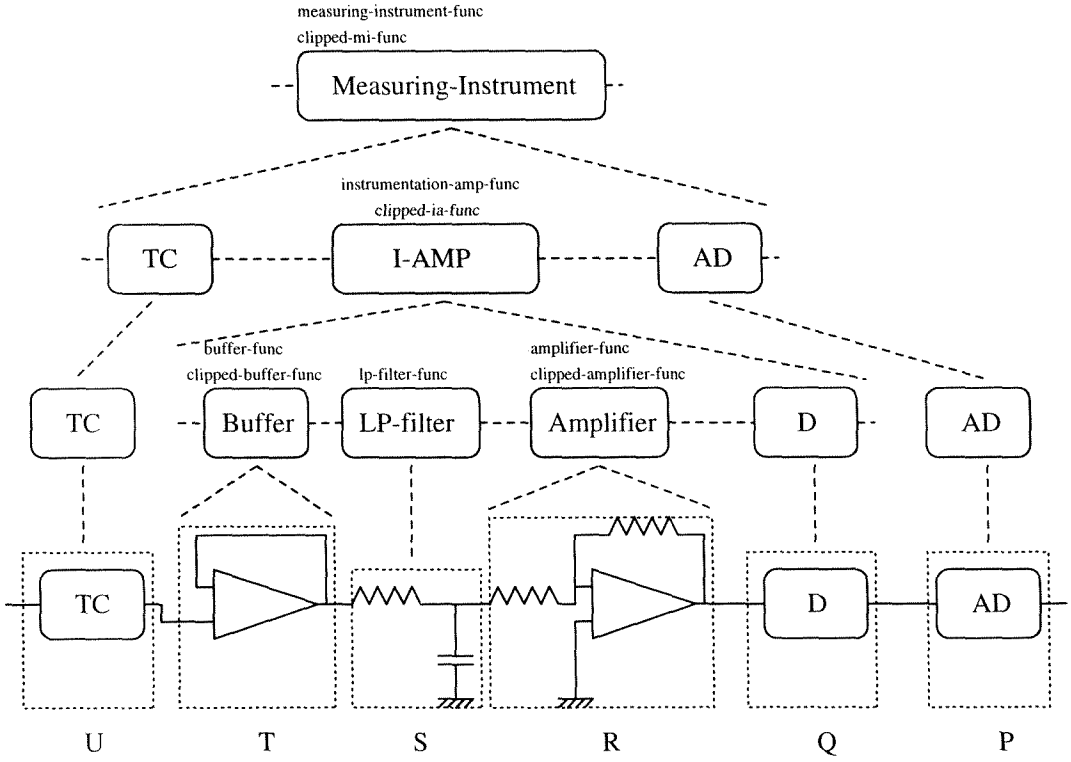


Figure 5: Analyzing the given structural description

it uses a decompose-analyze-compose strategy to transform *struc-desc*'s. Given the right set of abstractions, a large class of devices can be analyzed by recursive application of just these two strategies. The specific set of SFFs provided to the system in a way captures the behaviors the user is interested in. Thus, our method also proposes a way of capturing user's interest in specific behaviors and a way of bringing that knowledge to bear upon structure-to-function reasoning. We also described how our approach complements the simulation based approach in understanding devices [IC92].

References

[ACP91] Sanjay Addanki, Roberto Cremonini, and J. S. Penberthy. Graphs of models. *Artificial Intelligence*, 51(1-3):145–177, October 1991.

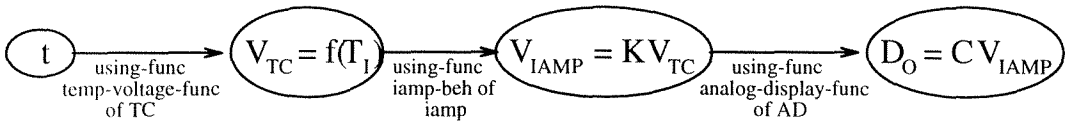
[All90] D. Allemang. *Understanding Programs as Devices*. PhD thesis, The Ohio State University, 1990.

[CGI93] B. Chandrasekaran, Ashok Goel, and Yumi Iwasaki. Functional representation as design rationale. *IEEE Computer, Special issue on concurrent engineering*, pages 48–56, Jan 1993.

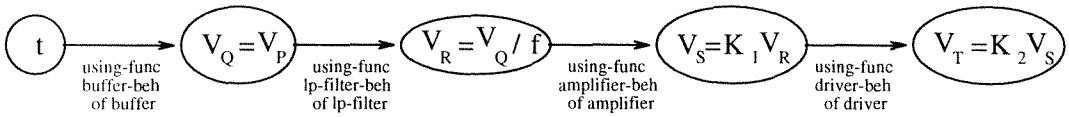
[Cha94] B. Chandrasekaran. Functional representation and causal processes. *Advances in computers*, 38, 1994.

[dK85] J. de Kleer. How circuits work. In D. G. Bobrow, editor, *Qualitative Reasoning About Physical Systems*. MIT Press, 1985.

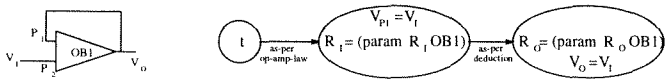
[FF91] B. Falkaenhainer and K. D. Forbus. Compositional modeling: finding the right model for the job. *Artificial Intelligence*, 51(1-3):95–143, Oct 1991.



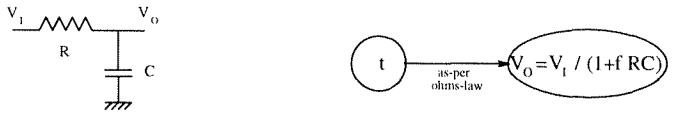
(a) ***measuring-instrument-cpd***



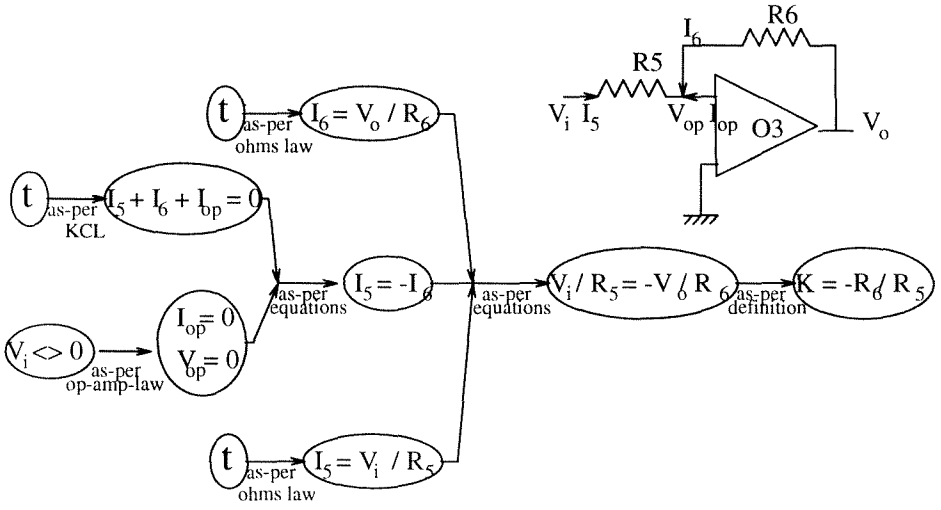
(b) ***instrumentation-amp-cpd***



(c) ***buffer-cpd***



(d) ***lp-filter-cpd***



(e) ***amplifier-cpd***

Figure 6: Cpds for the temperature measuring device

[For84] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[IC92] Yumi Iwasaki and B. Chandrasekaran. Design verification through function- and behavior-oriented representations: Bridging the gap between function and behavior. In *Proceedings of AI in Design Conference*, pages 597–616, 1992.

[IFVC93] Y. Iwasaki, R. Fikes, M. Vescovi, and B. Chandrasekaran. How things are intended to work: Capturing functional knowledge in device design. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1516–1522. Morgan Kaufmann, 1993.

[IS86] Yumi Iwasaki and Herbert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29, 1986.

[KC87] Benjamin Kuipers and Charles Chiu. Taming intractable branching in qualitative simulation. In *Proceedings of the IJCAI*, pages 1079–1085, 1987.

[Kui86] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–388, 1986.

[NJA91] P. Pandurang Nayak, Leo Jaskowicz, and Sanjay Addanki. Automated model selection using context-dependent behaviors. Technical Report RC 16906 (#74798), IBM, IBM Research division, T. J. Watson Research Center, Yorktown Heights, NY 10598, May 1991.

[SC88] V. Sembugamoorthy and B. Chandrasekaran. Functional representation of devices and compilation of diagnostic problem-solving systems. In J. Kolodner and C. Riesbeck, editors, *Experience, Memory, and Reasoning*, pages 47–73. Lawrence Erlbaum Associates, 1988.

[Str88] Peter Struss. Global filters for qualitative behaviors. In *Proceedings of the AAAI*, pages 275–279. AAAI, 1988.

[Tha93] Sunil Thadani. Extending causal sequences to make teleological distinctions. In *Proceedings of The NinthConference on Artificial Intelligence for Applications*, Orlando, Florida, March 1993. IEEE, IEEE Computer Society Press.

[Tha94] Sunil Thadani. *Constructing Functional Models of a Device from its Structural Description.* PhD thesis, The Ohio State University, 1994.

[VIFC93] M. Vescovi, Y. Iwasaki, R. Fikes, and B. Chandrasekaran. Cfrl: A language for specifying the causal functionality of engineered devices. In *Proceedings of the Eleventh National Conference on AI*, pages 626–633. AAAI, July 1993.

[Wel86] D. S. Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 30, 1986.