SPIE Intelligent Information Systems Conference
on Applications of Artificial Intelligence X:
Knowledge-Based Systems
April 22-24, 1992

Acquisition of Teleological Descriptions*

David W. Franke

Department of Computer Sciences
University of Texas at Austin, Austin, Texas 78712

Abstract

Teleology descriptions capture the *purpose* of an entity, mechanism, or activity with which they are associated. These descriptions can be used in explanation, diagnosis, and design reuse. We describe a technique for acquiring teleological descriptions expressed in the teleology language TeD. Acquisition occurs during design by observing design modifications and design verification. We demonstrate the acquisition technique in an electronic circuit design.

1 Introduction

For knowledge-based systems, acquiring the knowledge in a form usable by that system is a principal concern. This is particularly true for systems that rely on a database (knowledge base) of examples, such as analogical reasoning systems [12], design support systems, and case-based reasoning systems [19]. The teleology language TeD [8, 9] provides a means for addressing the acquisition problem in the context of the design process model of Figure 2. In particular, the essential elements referenced by teleological descriptions are available in this design process, namely design specifications and design modifications. Further, the process includes evaluation steps where the designer determines whether the design meets the specifications and where teleological descriptions can be captured.

In teleology we mean to capture the manner in which a component, at any level of the structure hierarchy, contributes to the behaviors of its ancestors in the structure hierarchy. We assume that real world systems are designed to achieve specific behaviors, and that each component and subsystem has been included in the design to contribute in some way to these behaviors.[1] When examining human-generated descriptions of systems or mechanisms, one finds that they are rich with descriptions of purpose, as well as descriptions of structure, behavior and causality. Descriptions of purpose are very valuable in communicating and understanding design descriptions, since they convey the designers' intent.

To demonstrate our acquisition goal, consider an electrical engineer designing an input selection circuit. The engineer begins with specifications describing the desired static (e.g. size) and dynamic (behavior) characteristics of the resulting circuit design. For example, one behavior specification for the input selection circuit is "invert the data signal when the control signal is high (logic true) and leave the output unchanged when the control signal is low (logic false)". The engineer also begins with specifications from the domain of CMOS circuit design, such as "the input value to a logic gate should not maintain a steady, intermediate value (voltage) between low and high, causing the gate to consume power by allowing current to flow".

Interacting with a design system, the engineer proposes a design, evaluates the design with respect to the specifications, and makes modifications to the design in attempt to make the design meet the specifications. In this interaction, teleological descriptions can be acquired and subsequently used to assist the engineer in completing the design, as demonstrated in the following example.

Engineer: Generate the possible behaviors of design 1 (*see schematic in Figure 1a.*) for signal `ctl` high and signal `in` transitioning from low to high.

*This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-8905494, IRI-8904454, and IRI-9017047, by NASA grant NAG 9-512, and by the Texas Advanced Research Program under grant no. 003658-175.

[1] As de Kleer points out [5], the goals of efficient design, manufacture, and maintenance of artifacts dictate that designers avoid superfluous components in the design, and hence each component contributes to the ultimate purpose of the design in some way.

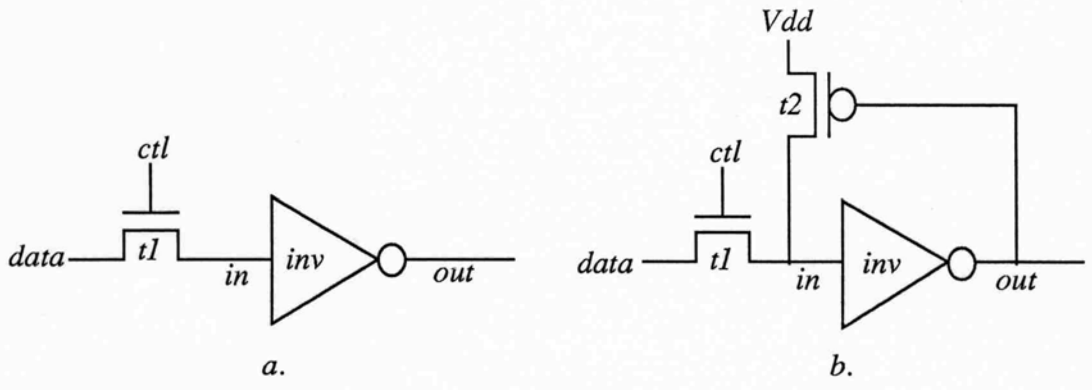


Figure 1: CMOS Input Selection Circuit - Schematics

Computer: The possible behaviors are: (*displays the behaviors*).
Engineer: Evaluate the behaviors with respect to the design specifications.
Computer: For signal *in*, the design does not meet specification 5: "the input value to a logic gate should not maintain a steady, intermediate value (voltage) between low and high, causing the gate to consume power by allowing current to flow". (*If the design system has a database of teleological descriptions and a teleological description referencing this specification exists in the database, a recommendation for modifying the design can be made.*)
Engineer: (*Modifies the design by adding feedback transistor t2, as shown in the schematic in Figure 1b.*) Generate the possible behaviors of design 2 for signal `ctl` high and signal `in` transitioning from low to high.
Computer: The possible behaviors are: (*displays the behaviors*).
Engineer: Evaluate the behaviors with respect to the design specifications.
Computer: The design meets all specifications. The purpose of the design modification transforming design 1 into design 2 is to guarantee specification 5.

2 TeD

The teleology language TeD relates design modifications (changes in structure) to design specifications (desired static and dynamic characteristics). With this language, we can formally express the designer's intent in modifying a design, namely to guarantee design specifications. Teleological operators are the language primitives for teleological descriptions. In the context of a modification to a design, a single teleological operator relates the unmodified design to the modified design in terms of *specification predicates.* Specification predicates express properties of a design which are desired, and verification of a design determines whether these properties hold for the design. Let ϕ be a specification predicate, d and d' be designs, δ a design modification such that d' is the design obtained by applying δ to d, and E and E$'$ be the envisionments of d and d', respectively.[2] We define the teleological operator **Guarantees** as:

$$\delta \textbf{ Guarantees } \phi \Leftrightarrow \begin{cases} \exists\, b \in \mathrm{E}, & \neg\phi, \\ \forall\, b' \in \mathrm{E}', & \phi. \end{cases}$$

One can understand the utility of teleological descriptions through questions or queries posed regarding a design. A teleological description addresses questions of the form "Why is this portion of the mechanism designed in this way?", or "What is the purpose of this piece of the mechanism?" This identifies one important use of teleological descriptions, namely explanation. The ability to express such explanations implies their use not only by humans but also by systems that automate problem-solving tasks. The task domains of diagnosis and design can use teleological descriptions to extend the applicability and performance of automated problem-solving systems.

[2] An envisionment is the set of behaviors exhibited by a mechanism or model of the mechanism.

2.1 Applying Teleological Descriptions

Deriving and utilizing causal relationships is an approach currently used in explanation systems and diagnosis systems. However, in mechanisms with highly inter-connected structure or feedback loops, causal relationships can exist between virtually every pair of components of the mechanism (and between variables of a mechanism model). If an observed symptom of a mechanism is considered either as an unwanted behavior (or a missing behavior), then a teleological description which relates a component of the mechanism with the prevention (introduction or guarantee) of that behavior provides a heuristic for selecting among potential causes. Teleological descriptions can provide a more productive initial focus of attention for diagnosis.

Mostow [17] discusses the potential for improving in the design process through capture and representation of design rationale, and Franck [6] points out that "Design is a form of teleological reasoning, in that from the intended purpose or anticipated behavior one can select elements that have the adequate structure to do so." Teleological descriptions provide a means for representing design rationale. Given the ability to capture and represent teleological descriptions (either generated by humans or programs), these descriptions can also be used to classify mechanism descriptions. The design reuse problem can be addressed by providing:

1. Techniques for capturing and representing information by which a design component should be classified for subsequent retrieval, and

2. A language for describing the characteristics of the design component the designer wishes to examine as a candidate for reuse.

Teleological descriptions add another dimension by which designs can be classified and retrieved. For example, a designer may wish to examine components which can control some variable (say tank fluid level) of a system under design. In the absence of a description of purpose, designers must rely on their mental inventory of likely components, structural features of likely components, or specific behaviors of likely components in order to construct a query for the search. Current reuse approaches are based on structure classification and hierarchy or on classifications organized around keywords that represent behavioral categories. Indexing designs and design components for reuse via teleological descriptions provides more semantic content for the reusing designer.

In reusing an existing design, if the design does not match the current requirements exactly, it will require some modification. As this design is being modified, knowledge of the purpose of components will benefit the (resuing) designer in much the same way teleological descriptions aid the diagnosis task. Teleological descriptions also help the designer understand the original purpose of components in the mechanism design.

[8, 9] discuss applying teleological descriptions to the tasks of explanation, design reuse, and diagnosis.

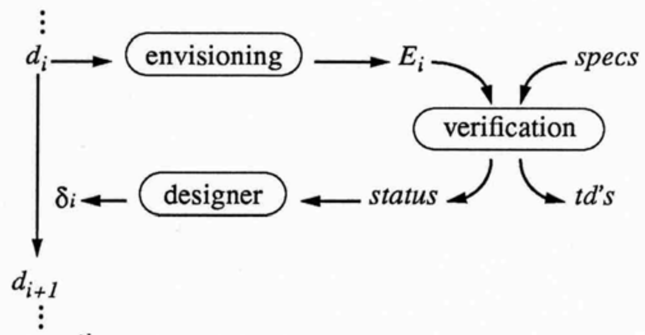
3 Design Modification and History

The design process model used here (see Figure 2) is of the Propose-Critique-Modify family described by Chandrasekaran [4]. Our design process model starts with a set of specifications for the design, including physical characteristics and descriptions of (required, prohibited, ...) behaviors. In addition to the specifications of a particular design, there is often a set of specifications, or design principles, that describe general engineering practice for the domain at hand. These specifications include characteristics of the design required for manufacturing, maintenance, standards conformance, and regulatory requirements.

Given specifications for the design, the design process proceeds as a series of structure modifications, starting from some initial structure. Accompanying this series of structure descriptions is a corresponding series of evaluations of the design with respect to the various design specifications.[3] The evaluation step requires a description of the behaviors of the design, called an *envisionment*. The evaluation step then examines the envisionment and the design specifications to verify the truth of the specifications. Ideally, this process continues until evaluation shows that all specifications have been satisfied, or amended so as to be satisfied.

A design structure language provides a means for describing a single point in the history or evolution of a design. A description of *design history* also requires a means for describing the transitions from one state of the design (distinct from a behavior state of an artifact or instance of the design) to another. The term *design modification* denotes

[3]Evaluation of dynamic characteristics (e.g. functional correctness, performance, thermal operating characteristics) and static characteristics may require complex computations like simulation, timing analysis, formal verification of function, and thermal modeling.



d_i - i^{th} version of the design
E_i - envisionment for design d_i
specs - design specifications
td's - teleological descriptions captured in verification
status - results of verification
δ_i - modifications generated by the designer

Figure 2: Design Process Flow (Single Step)

such a transition. A design history is a pair comprised of an initial design and a sequence of design modifications,[4] denoted

$$(d_0, \langle \delta_1, \delta_2, \ldots, \delta_n \rangle)$$

where d_0 is the initial design and δ_i are design modifications. This design history defines a sequence of designs

$$d_0, d_1, \ldots, d_{n-1}, d_n$$

where d_i is the result of applying design modification δ_i to design d_{i-1}. The design history is captured during the initial design and design modification steps of the design process model of Figure 2.

4 Acquisition

Several acquisition approaches are possible, and one has been implemented in this work. These approaches can be applied either interactively during design or to a replay of the design history. The acquisition approaches are:

- *Explicit description* - the designer identifies the design specification and the modification that comprise the teleological description. The acquisition system can verify that the modification did in fact result in the specification being met.
- *Explicit cue* or *Learn now* - the designer explicitly invokes generation of teleological descriptions at those points in the design process where a specification has been satisfied.
- *Implicit cue* - the designer states to the design system that the design specification being addressed is X, and the acquisition implementation determines when the specification has been met.
- *Automatic* - the acquisition program observes design activity, noting design modifications and evaluations, and automatically generates teleological descriptions.

The approach implemented in this work can be described as the *explicit cue* approach, and provides the implementation core for the more automated approaches.

[4] Representing a design history as an initial design and design modifications is common in commercial CAE and CAD systems, providing a record of design changes and the ability to undo design changes. For efficiency, new "initial" designs are created periodically in such systems by applying the modifications and explicitly representing the new "initial" design.

4.1 Comparative Analysis

Comparative analysis is used to recognize that a design specification has been met as a result of a design modification. We compare design evaluations performed before and after the modification to determine if a previously unsatisfied specification is now satisfied. Satisfaction of a specification predicate is determined by a model checking algorithm that computes the behavior abstraction relations described in [9, 10]. In the designs examined in this work, we use QSIM [13, 14] to model and simulate designs. The choice of QSIM allows strong statements about guarantees of the presence or absence of particular behaviors since QSIM guarantees that all possible behaviors of the model appear in the QSIM generated envisionment [15]. To evaluate a design, we compare each behavior with the specification predicate to determine the truth value of the specification predicate for that behavior. After evaluating the unmodified and modified designs, design specifications which were not met by the unmodified design and are now met by the modified design are attributed to the design modification.

It is possible that a modification does not guarantee a specification for all possible behaviors of a design, but does so for some behaviors. In this case, a conditional teleological description can be generated, with the condition describing an initial state or state sequence common to the behaviors now meeting the specification and not occurring in (i.e. abstracting) the behaviors that do not meet the specification. If ϕ' denotes the condition under which the specification predicate is true, we write the teleological description as

$$\delta \textbf{ Conditionally } (\text{in } \{\phi'\}) \textbf{ Guarantees } \phi.$$

Acquisition in this manner can be applied to modeling and evaluation techniques that do not guarantee that all possible behaviors are represented *if* the evaluation technique can state those initial conditions under which it can guarantee that the specification predicate is true. For example, a quantitative modeling and simulation approach may be restricted to making assertions about the truth of a specification predicate given a set of initial, quantitative values for the model (design).

5 Example

Consider the input selection circuit in Figure 1a, extracted from a CMOS arithmetic logic unit (ALU) design. The circuit contains a pass transistor *t*1 and an inverter *inv*. In the ALU design, the signal *ctl* determines (controls) whether the signal *data* is passed into the logic portion of the ALU. The desired behavior of this circuit in terms of signals *data*, *ctl*, *in*, and *out* is as follows:

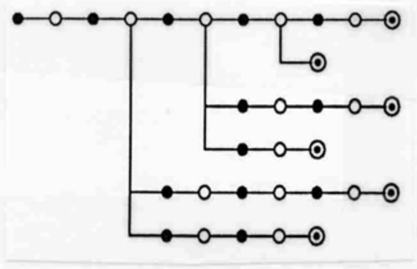
> When the value of signal *ctl* is HIGH, the value of signal *data* is transmitted to signal *in* (i.e. they are electrically connected). This value is then inverted by *inv* (HIGH $\rightarrow$ LOW or LOW $\rightarrow$ HIGH), and the inverted value then becomes the value of signal *out*.

The (logic) values of HIGH and LOW are landmarks of the quantity space in which the parameters *data*, *ctl*, *in*, and *out* range. These landmarks are the desired values for signals in the circuit when no signal transitions are occurring.

5.1 Design Specifications

A general domain constraint for CMOS design is that signals should take on an intermediate value between LOW and HIGH only during a transition from LOW to HIGH or HIGH to LOW. In particular, a logic component such as the inverter should not have an input signal with an intermediate value between LOW and HIGH and unchanging. The rationale for this design rule comes from the operating characteristics of the CMOS inverter implementation, in which current flows when the input has value between LOW and HIGH but does not flow when the input has value either LOW or HIGH.[5] Hence, CMOS circuits consume power only when switching, as opposed to other implementation technologies such as nMOS that consume power during signal transition and at other times as well. We denote the specification predicate describing the fact that no behavior should allow signal *in* to have an intermediate, steady value as ϕ_1.

[5]The actual value at which the inverter stops drawing current is determined by a threshold value set by the manufacturing process used to produce the circuit.



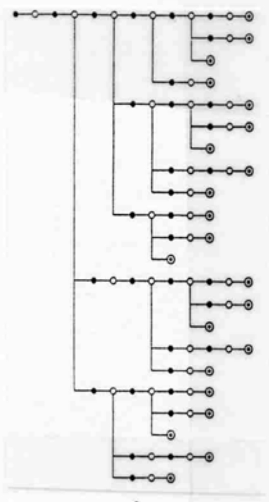


Figure 3: Behavior Tree

5.2 Behavior

We first examine the behavior of the circuit when the signal *in* has value LOW, signal *out* has the value HIGH, signal *ctl* has the value HIGH, and the signal *data* has just assumed the value HIGH. The desired behavior will be that the value HIGH is transmitted to *in*, and the inverter *inv* changes *out* to LOW.

The operating characteristics of $t1$ (an n-channel MOS transistor) are such that when the signal *data* has value HIGH and the signal *ctl* has the value HIGH, the value transmitted to signal *in* is HIGH minus the threshold value (> 0) of $t1$. Let HTH denote the value HIGH minus the threshold value of $t1$. The value HTH is between LOW and HIGH, and hence not a desired value for signal *in*. The envisionment (see behavior tree in Figure 3a) predicts six qualitatively unique behaviors, and each behavior exhibits the undesirable behavior of signal *in* reaching an intermediate value and remaining steady.

5.3 Evaluation 1

The model checking algorithm implemented in this work determines that no behaviors satisfy specification ϕ_1. The designer's task is to modify the design either structurally or via changes in parameter values to bring the behaviors in line with the specification.

5.4 Modification 1

The addition of the feedback transistor $t2$, a p-channel MOS transistor (see the schematic in Figure 1b) modifies the circuit behavior in the following way (in terms of signals *in* and *out*):

> As signal *in* transitions from LOW to HIGH, signal *out* transitions from HIGH to LOW. As signal *out* moves away from HIGH and towards LOW, transistor $t2$ electrically connects *in* with Vdd, enabling a current flow from Vdd to *in* which in turn increases the value of *in* to that of Vdd (HIGH).

The operating characteristics of a p-channel transistor are such that the value HIGH can be transmitted without degradation (i.e. not subject to any threshold value). Consequently, the designer's modification, the addition of $t2$, prevents the scenario in which *in* reaches a value less than HIGH and remains steady. This design modification is automatically captured and added to the circuit's design history.

The envisionment of the modified design characterizes 22 qualitatively distinct behaviors, shown in Figure 3b, with all behaviors having a final, quiescent state in which the signal *in* has value HIGH, the desired result.

The purpose of the design modification which adds $t2$ to the input selection circuit can be expressed in TeD as follows. Let δ_1 represent the design modification of adding $t2$ to the design. Then

$$\delta_1 \textbf{ Guarantees } \phi_1. \tag{1}$$

The behavior which electrically connects *in* to Vdd also addresses another problem that occurs when *in* has value HIGH and *ctl* transitions from HIGH to LOW. In this situation, *in* is no longer electrically connected to *data*, and becomes a memory element which should preserve its value, HIGH. However, in the absence of $t2$, the charge at *in* will dissipate and move the signal value away from the value HIGH, resulting in the value of signal *out* changing also (i.e. moving away from LOW). By introducing $t2$, the charge at *in* is maintained, and hence the behavior in which *in* decreases in value is prevented.

The purpose of the design modification that adds $t2$ can be expressed in TeD as follows. Let ϕ_2 denote the specification predicate describing the fact that signal *in* should behave as a memory element and not leak charge. Then

$$\delta_1 \textbf{ Guarantees } \phi_2. \tag{2}$$

The design modification of adding $t2$ to the circuit has been assigned the purpose of *guaranteeing* that a steady value between LOW and HIGH for signal *in* will not occur. Further, from starting conditions where *in* has value HIGH and *ctl* has value LOW, the modification *prevents* signal *in* from changing its value.

5.5 Evaluation 2

While the first design modification has addressed problems associated with signal *in* achieving and maintaining value HIGH, a new problem has been introduced by the first design modification. If signal *in* has value HIGH, signal *data* has value LOW, and signal *ctl* transitions from LOW to HIGH, the charge stored at *in* (representing the value HIGH) should be drawn off via the connection through $t1$. However, recall that current can flow from Vdd to *in* via the connection provided by $t2$. If current flows from Vdd to *in* at a sufficient rate, an intermediate value will be reached for *in* such that the complementary value at *out* is not high enough to "turn-off" $t2$ (a p-channel transistor is off when the gate voltage is HIGH).

5.6 Modification 2

The second design modification changes the channel resistance of $t2$ (to a high resistance value) to impede the current flow and hence prevent the scenario in which *in* reaches an equilibrium point between HIGH and LOW during the HIGH to LOW transition of *in*. A teleological description relates the design modification (changing the channel resistance of $t2$) to the desired change in behavior, namely that the circuit can successfully switch the signal value of *in* from HIGH to LOW.

The purpose of the particular channel resistance value for $t2$ can be expressed in TeD as follows. Let δ_2 denote the design modification of increasing the channel resistance of $t2$, let ϕ_3 denote the specification predicate describing the conditions under which *in* should be discharged, and let ϕ_4 denote the specification predicate describing *in* as completely discharged. Then

$$\delta_2 \textbf{ Conditionally } (\text{in } \{\phi_3\}) \textbf{ Guarantees } \phi_4. \tag{3}$$

5.7 Modification Teleology Summary

To summarize this example, $t2$ was added 1) to prevent the scenario in which *in* reaches a steady value between LOW and HIGH when transitioning from LOW to HIGH, and 2) to prevent the scenario in which the value of *in* decreases from HIGH when *in* is acting as a memory element storing the value HIGH. The channel-resistance of $t2$ was set high to prevent the scenario in which *in* reaches an equilibrium value between LOW and HIGH during the transition from HIGH to LOW. The complete design history in the context of the design process flow is shown in Figure 4.

6 The Issue of Scope

In considering acquisition of teleological descriptions, we must consider the appropriate level of behavior or specification at which to attribute a purpose of a design component or modification. The example of a spark plug's purpose in an automobile, suggested by Mooney [16], best demonstrates this issue. What is the purpose of a spark plug in an automobile? To make the car go? To make the engine produce force? To make a piston go up and down? In this example, the number of possible specifications or desired behaviors would seem to be endless, given all the

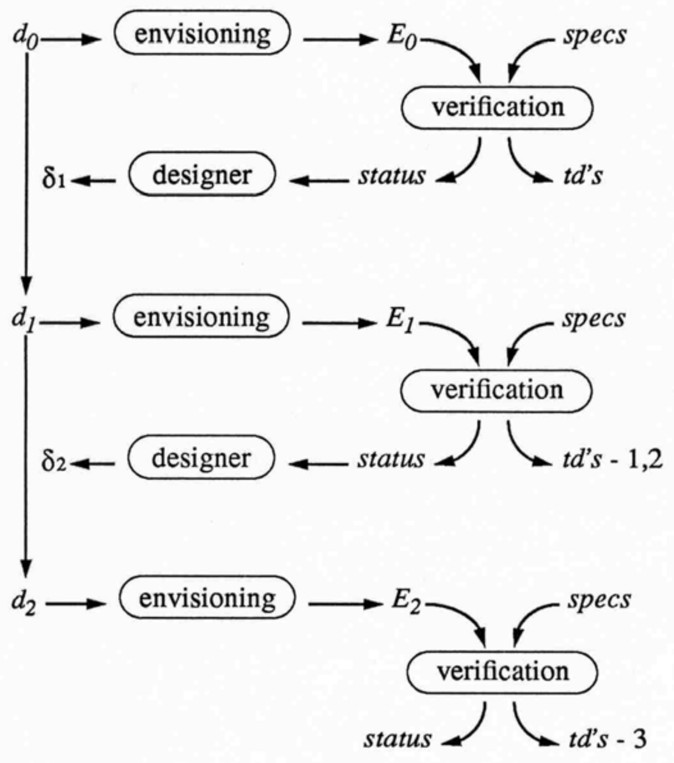


Figure 4: Complete Design Process Flow

potential behaviors of the automobile. We resolve this issue in the following paragraphs via a discussion concerning the nature of large system specifications, how they are developed, and how they evolve.

6.1 Design Specification Hierarchy

Although not always explicitly represented, the design specifications for a large, complex system describe the desired behavior and physical characteristics of the system at the level at which a user interfaces with that system. In the case of the automobile, these specifications (implicit or explicit) state such things as the expected behavior when the steering wheel is turned or when the accelerator pedal is pushed down, or the miles per gallon achieved by the vehicle. The designer or design team elaborates the design specifications based on past knowledge of such designs and on the initial functional and structural decompositions of the design (see discussions by Alford [2] and Rich and Shrobe [18]). For the automobile example, more detailed specifications for the steering column and linkage are generated (e.g. X degrees of rotation of the steering wheel translates to Y degrees of deflection in the front tires), the engine (e.g. power curve characterization), and other functional and structural components of the design. Each of these elaborations can be related to the higher level specification to which it contributes.

Teleological descriptions can be generated for any level of the specification hierarchy. For a specific component or modification, the associated specification (associated via the teleological description) will usually make a statement about the desired behavior of the functional or structural level at which the component is included. For example, a specification for the automobile might be that it translates chemical energy (gasoline) into mechanical energy (motion). The purpose of the engine is then to guarantee this behavior. As the engine design is created (via functional and/or structural decomposition), the specification is decomposed, eventually resulting in a specification for each individual cylinder of the engine. This level of specification will be referenced by a teleological description for the spark plug, namely to guarantee the behavior that the compressed fuel and air mixture is ignited and burns. Consequently, a teleological description will associate a modification with a specification of the "nearest" structural parent (hierarchically) within which the modification is made.

7 Related Work

Acquisition of descriptions of purpose or design rationale has been addressed by de Kleer's EQUAL system [5], the Conservation of Design Knowledge (CDK) Project [3] at NASA Ames Research Center, and by Gruber's ASK system [11].

de Kleer's EQUAL system [5] expresses teleological descriptions in terms of behaviors of a component. Each description is based on causal assumptions on the parameters of the component. EQUAL identifies a functional characterization (teleological description) by matching derived behavior with prescribed behavior prototypes which have been enumerated, named, and added as domain specific knowledge. Limitations of this approach are that teleological descriptions are prescribed, domain specific, and limited to describing relationships among variables of a single component.

The Conservation of Design Knowledge (CDK) Project [3] at NASA Ames Research Center addresses the problems of representing and acquiring design rationale using a philosophy similar to TeD. TeD provides a formal language for representing design rationale descriptions captured in the CDK acquisition work, and the CDK work complements TeD by providing acquisition techniques.

Gruber's ASK system [11] elicits justifications from experts via an interactive dialogue with the expert. TeD provides a formal language for representing ASK explanations (teleological descriptions), provides indexing capabilities for ASK explanations, and addresses acquisition of teleological descriptions during design.

8 Summary

The acquisition technique described here has been implemented, and classification and indexing of teleological descriptions has been implemented as well. Design models are described in the CC language [7], a component-connection structure language that generates QSIM [13, 14] QDE's. The author would like to thank Ben Kuipers for his guidance in this research, as well as the members of the Qualitative Reasoning Group at the AI Lab, University of Texas at Austin.

References

[1] Harold Abelson, Michael Eisenberg, Matthew Halfant, Jacob Katzenelson, Elisha Sacks, Gerald J. Sussman, Jack Wisdom, Kenneth Yip, "Intelligence in Scientific Computing" in *Communications of the ACM*, Vol. 32, No. 5 (May 1989), pp. 546-562.

[2] Mack W. Alford, "A Graph Model Based Approach to Specifications", in Distributed Systems: Methods and Tools for Specification, M. Paul and H. J. Siegert (eds.), Lecture Notes in Computer Science No. 190, G. Goos and J. Hartmanis (eds.), Springer-Verlag, New York, 1982, pp. 131-201.

[3] Catherine Baubin, Cecilia Sivard, Monte Zweben, "Model-Based Approach to Design Rationale Conservation", in Proceedings of the 1989 Workshop on Model-Based Reasoning, Detroit, August 20, 1989, pp. 88-90.

[4] B. Chandrasekaran, "Design Problem Solving: A Task Analysis", in *AI Magazine*, Vol. 11 No. 4 (Winter 1990), pp. 59-71.

[5] Johan de Kleer, "How Circuits Work", in Qualitative Reasoning About Physical Systems, Daniel G. Bobrow, ed., The MIT Press, Cambridge, MA 1985, pp. 205-280. Reprinted from *Artificial Intelligence* Vol. 24, 1984.

[6] Bruno Franck, "Qualitative Engineering at Various Levels of Conception for Design and Evaluation of Structures", in Proceedings of the Conference on Industrial and Engineering Application of AI and ES, ACM, 1989.

[7] David W. Franke, Daniel L. Dvorak, "CC: Component Connection Models for Qualitative Simulation, A User's Guide", TR AI90-126, Dept. of Computer Sciences, The University of Texas at Austin.

[8] David W. Franke, "Deriving and Using Descriptions of Purpose" in *IEEE Expert* (special track on Functional Reasoning), Vol. 6, No. 2 (April 1991), pp. 41-47.

[9] David W. Franke, "A Theory of Teleology", Ph.D. Dissertation, University of Texas at Austin, Feb. 1992.

[10] David W. Franke, "An Abstraction Relation for Behavior", submitted to *AAAI-92*.

[11] Thomas Gruber, "Learning Why by Being Told What", in *IEEE Expert* Vol. 6, No. 4 (August 1991), pp. 65-75.

[12] David H. Helman (ed.), Analogical Reasoning, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1988.

[13] Benjamin J. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure", in Qualitative Reasoning About Physical Systems, Daniel G. Bobrow, ed., The MIT Press, Cambridge, MA 1985, pp. 169-203. Reprinted from *Artificial Intelligence* Vol. 24, 1984.

[14] Benjamin J. Kuipers, "Qualitative Simulation", in *Artificial Intelligence*, Vol. 29, No. 3, (September 1986), pp. 289-338.

[15] Benjamin J. Kuipers, "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge", *Automatica*, Vol 25, No. 4 (1989), pp. 571-585.

[16] Raymond Mooney, private communication.

[17] Jack Mostow, "Towards a Better Model of the Design Process", in *AI Magazine*, Vol. 6, No. 1 (Spring 1985), pp. 44-57.

[18] Charles Rich, Howard E. Shrobe, "Initial Report on a LISP Programmer's Apprentice", in Interactive Programming Environments, D. Barstow, H. Shrobe, E. Sandewall (eds.), McGraw-Hiull, New York, 1984, pp. 443-463. Reprinted from *IEEE Transactions on Software Engineering*, Vol. SE-4, No. 6 (November 1978), pp. 456-467.

[19] Christopher K. Riesbeck, Roger C. Schank, "Case-Based Reasoning: An Overview", in Inside Case-Based Reasoning, Lawrence Erlbaum, Hillsdale, NJ, 1989.