GIT-CC-94/18

Situating Natural Language Understanding within Experience-Based Design

Justin Peterson
Kavi Mahesh
Ashok Goel
goel@cc.gatech.edu

To appear in the International Journal of Human-Computer Studies
(previously known as the International Journal of Man-Machine Studies).

Georgia Institute of Technology
September 1994

To appear in the International Journal of Human-Computer Studies

Situating Natural Language Understanding within Experience-Based Design

Justin Peterson
Kavi Mahesh
Ashok Goel
College of Computing
Georgia Institute of Technology

Abstract

Building useful systems with an ability to understand "real" natural language input has long been an elusive goal for Artificial Intelligence. Well-known problems such as ambiguity, indirectness, and incompleteness of natural language inputs have thwarted efforts to build natural language interfaces to intelligent systems. In this article, we report on our work on a model of understanding natural language design specifications of physical devices such as simple electrical circuits. Our system, called KA, solves the classical problems of ambiguity, incompleteness, and indirectness by exploiting the knowledge and problem-solving processes in the situation of designing simple physical devices. In addition, KA acquires its knowledge structures (apart from a basic ontology of devices) from the results of its problem-solving processes. Thus, KA can be bootstrapped to understand design specifications and user feedback about new devices using the knowledge structures it acquired from similar devices designed previously.

In this paper, we report on three investigations in the KA project. Our first investigation demonstrates that KA can resolve ambiguities in design specifications as well as infer unarticulated requirements using the ontology, the knowledge structures, and the problem-solving processes provided by its design situation. The second investigation shows that KA's problem-solving capabilities help ascertain the relevance of indirect design specifications, and identify unspecified relations between detailed requirements. The third investigation demonstrates the extensibility of KA's theory of natural language understanding by showing that KA can interpret user feedback as well as design requirements. Our results demonstrate that situating language understanding in problem solving, like device design in KA, provides effective solutions to unresolved problems in natural language processing.

Contents

List of Figures

1 Introduction and Overview

It has long been recognized that language understanding requires abilities far beyond what pure linguistic knowledge permits (Charniak and McDermott (1985); Grishman (1986); Kintsch (1974); Rich and Knight (1991); Winston (1992)). In the past, two approaches have been pursued to endow natural language understanding systems with such abilities as inference and problem solving. One approach has been to endow systems with a variety of domain and world knowledge as well as a range of inference, explanation, and reasoning capabilities (e.g., BORIS Lehnert, Dyer, Johnson, Yang, and Harley (1983)). All of the system's capabilities, in this approach, are solely employed in the service of "Understanding" natural language. Unfortunately, this approach to language understanding has not been particularly successful since the linguistic methods typically used for "Understanding" do not work well with the various types of non-linguistic knowledge that are required to understand language. Moreover, the specialized types of knowledge that these systems require for understanding a natural language text (Lehnert *et al.* (1983)) are not readily available, nor has it been demonstrated that they can be acquired by established methods. As a result, this approach has only resulted in prototype language understanding systems that show little promise of scalability or bootstrapping.

Another approach has been to make some other cognitive task (such as robot planning or expert decision making) the main task and add a natural language "Front-End" to the system. The front-end works in service of the rest of the system and has limited abilities to translate natural language inputs into a conceptual representation that is comprehensible to the rest of the system (e.g., Simon and Hayes (1979); Hendrix, Sacerdoti, Sagalowicz and Slocum (1978)). Such natural language front-ends are unable to solve many linguistic problems because they neither possess the requisite non-linguistic knowledge nor get any useful feedback from the other task. The linguistic problems remain hidden in the intermediate representation and are not resolved satisfactorily by the rest of the system with its non-linguistic methods and knowledge.

Our work on natural language understanding takes a third approach, a more modular one, in which language understanding and problem solving interact by communicating the results of their decision-making with each other. Language understanding uses the results of problem solving operations to resolve linguistic problems such as ambiguities. Problem solving in turn uses the decisions made by language understanding to direct the course of its own problem decomposition and problem solving process. A major difference between this approach to language understanding and previous approaches is that the language understander need not possess either the knowledge or the reasoning abilities to solve problems in reasoning. Nor does problem solving need to know how to solve linguistic problems left unsolved by a natural language front-end. All that the two need is to solve their own problems partially, be able to communicate their decisions and results with each other, and cooperate in an integrated architecture to arrive at a negotiated solution to the overall problem.

In this approach to building natural language understanding systems, we are not simply adding additional types of knowledge to a linguistic processor in the hope of making language understanding a feasible task. Nor are we passing on linguistic problems to a non-linguistic reasoning system in the hope that the reasoning system will somehow solve the problems in the natural language input. Instead, we are "Situating" natural language understanding in another task to make it more achievable—to exploit the knowledge *and the reasoning processes* running in the situation of another task to solve classical problems in natural language processing.

We have chosen to investigate the "situatedness" of natural language understanding within the design of simple physical devices such as electrical circuits. Design, like natural language understanding, is an oft-studied problem in AI, and many types of knowledge structures and reasoning methods have been developed for automating design, both in our own research and that of many others. More importantly, the knowledge of physical devices and their design that we are proposing to use in language understanding has been shown to be obtainable from prior problem solving experiences (Goel (1989, 1991a, 1991b)). Thus, the choice of physical device design for situating natural language understanding has a real promise of scalability and bootstrapping if the knowledge that can be acquired in the problem solving process can be used to solve classical problems in natural language such as ambiguity.

Taking this approach, we have built a natural language understanding system called KA for (i) understanding device specifications written in natural language (English) in the context of designing new devices and (ii) understanding user feedback in natural language in the context of device redesign. KA embodies an integrated model-based and case-based approach to design problem solving that we have been developing for many years now (Goel (1989, 1992); Goel and Chandrasekaran (1989, 1992)). KA uses the same approach to address well-known problems of natural language understanding such as resolving ambiguity, interpreting indirect statements, and inferring unspecified information. We have conducted several investigations to demonstrate that KA's design situation provides viable solutions to the problems of ambiguity, indirectness, and omission.

In this article, we describe three investigations in solving natural language understanding problems with KA. Our first investigation demonstrates that KA's models of physical devices and its reasoning for their design helps resolve ambiguities in design specifications as well as infer unarticulated requirements. Briefly, our approach was to construct an initial, tentative interpretation of the design specification using KA's language processing capabilities, locate a similar design in KA's case memory using model-based retrieval, and then use the retrieved design to countermand erroneous decisions made in the resolution of ambiguities. These retrieved designs were also used to augment the interpretation of requirements with those that were not articulated in the natural language specification. Our results in this first investigation indicate that, among other benefits, models of physical devices and the ability to reason about the function of devices aid ambiguity resolution in two specific ways. First, the ontology of physical devices employed in KA grounds the semantic representations of language processing, ensuring that decisions about the consistency of interpretations are made in accordance with the ontology of device design. Second, by allowing previous problem-solving experiences to be factored into linguistic decision-making, interpretations that are most compatible with past experience are produced.

The second investigation demonstrates that KA's problem-solving capabilities help ascertain the relevance of indirect design specifications, and identify unspecified relations between detailed requirements. Our approach to these problems relied extensively on KA's memory of design cases, case-specific models of devices, and model-based methods for design adaptation. Our results indicate that a memory of design cases and device models as well as the ability to adapt these descriptions in accordance with a deep understanding of the structure, function, and behavior of devices provides considerable leverage when dealing with indirect and ill-specified English descriptions of design problems. Using device descriptions in memory as baseline interpretations and the information extracted from the text as constraints on interpretation, model-based adaptation proves to be an effective means of producing both an interpretation of the text and a successful design solution. This result along with those of the first investigation demonstrates that situating natural language understanding in design problem solving provides tractable solutions to problems in understanding natural language specifications of design problems.

The third investigation demonstrates that KA's approach to situated natural language understanding can be extended to other, related situations. Natural language texts are used to achieve a variety of communication goals at different stages in the design process. For example, at later stages in the design, customers use English texts to communicate feedback to designers which the designers must understand in order to redesign the device. In the third and most recent investigation, we chose a problem in the design of a reaction wheel assembly for the Hubble space telescope and examined KA's ability to understand user feedback. We demonstrated that KA was in fact able to understand such user feedback and use the information it could extract from the feedback to redesign the reaction wheel assembly. This cost effective redesign was made possible by KA's repertoire of plans for incremental redesign to correct the malfunction and its ability to precisely identify the part of the design that is to blame for the malfunction. This investigation demonstrated that some of the same kinds of knowledge and methods that are used in understanding initial design requirements also enable the integrated system to understand natural language feedback from the customer for iterative redesign to meet customer requirements.

The organization of rest of this article is as follows. First, we provide a brief description of the design situation, the ontology of physical devices, and the problem solving methods used in KA's model of designing physical devices. Next, we show how classical problems in natural language understanding get redefined when the understanding is situated in the design task. The section after this describes the solution to these problems in terms of the architecture of the KA system, detailing the different components that make up KA. Following this description, we present the three investigations that substantiate our claim that situated understanding provides viable solutions to problems in natural language understanding. This presentation will include sample texts used in our work, an analysis of their difficulties, and a demonstration of KA's capabilities. Then, we describe the strengths and limitations of our work and compare it to that of other research projects. We conclude by articulating the contributions of our research. Throughout the paper, we focus on language processing. The reader may refer to our earlier papers that describe memory and problem solving in greater detail.

2 The Design Situation

The task in which we are situating natural language understanding is the design of simple physical devices such as electrical circuits and computer networks (Peterson, Mahesh, Goel, and Eiselt (1994)). Our work evolves from previous work on the Kritik project which used past design cases and associated device models for creating new device designs (Goel (1989)). Such prior experiences in design are stored in KA's memory in the form of cases and case-specific models (Goel (1989, 1991a, 1991b, 1992)). The memory uses a representation called the *SBF language* (Chandrasekaran, Goel and Iwasaki (1993); Goel (1989, 1992); Sembugamoorthy and Chandrasekaran (1986)) based on a component-substance ontology of the domain of physical devices (Bylander (1991); Bylander and Chandrasekaran (1985); Goel (1989)). In the SBF language, a device is represented in terms of its desired function, its internal structure (components and connections between them), and internal behaviors of its components. These behaviors are articulated in terms of the various substances[1] contained in the components, the states of the substances, and the flow of substances between components.

Given a functional specification of a device to be designed, previous designs of functionally similar devices are retrieved from the cases and models in memory. The models in the previous cases are adapted to the present problem using model-based methods (Goel (1991a, 1991b)). KA's design capabilities however extend far beyond mere case-based adaptation of prior designs. It can diagnose faulty designs, identify the components at error, and redesign the device by applying

[1]The term substance as used here not only includes material substances such as air and water, but also abstract substances and forms of energy such as heat, electricity, information, force, and so on (Bylander (1991)).

various design repair plans that it knows about, such as component replacement and component cascading (Goel (1991a, 1991b); Stroulia and Goel (1992); Stroulia, Shankar, Goel and Penberthy (1992)). Interestingly, it can also acquire new experiences and store away the new cases and models in its memory by learning appropriate indices to the cases and models (Bhatta and Goel (1992, 1993); Goel (1991a, 1991b)).

2.1 The SBF Language

Since our investigations will be using examples described in the SBF language, we make a brief digression in this section to describe the salient terms in the language. Models of physical devices are represented in terms of their structure, behavior, and function (SBF). These models are based on a *component-substance ontology.* In this ontology, the structure of a device is constituted of its *components* and *substances.* Substances have *locations* in reference to the various components of the device. They also have *behavioral properties,* such as *voltage* of *electricity,* and corresponding *parameters,* such as *1.5 volts, 3 volts,* and so on. This ontology gives rise to a SBF language.

Structure: The structure of a design is expressed in terms of its constituent components and substances and the interactions between them. Figure 1(a) shows the structure of a `1.5-volt electric circuit` (EC1.5) schematically.

Function: A function is represented as a schema that specifies the behavioral state the function takes as input, and the behavioral state it gives as output. Figure 1(b) shows the function "Produce Light" of EC1.5. Both the input state and the output state are represented as *substance schemas.* The input state specifies that electricity at `location` Battery in the topography of the device (Figure 1(a)) has the property `voltage` and the corresponding parameter `1.5 volts`. The output state specifies the property `intensity` and the corresponding parameter `6 lumens` of a different substance, light, at `location` Bulb. A third aspect of a functional specification is the *stimulus* which initiates the behaviors of the device. For example, the force on the switch is the stimulus to the electrical circuit in Figure 1. In addition, the slot *by-behavior* acts as an index into the causal behavior that achieves the function of producing light.

Behavior: The internal causal behaviors of a device are viewed as sequences of *state transitions* between *behavioral states.* The annotations on the state transitions express the *causal, structural,* and *functional context* in which the transformation of state variables, such as substance location, properties, and parameters, occur. Figure 1(c) shows the causal behavior that explains how electricity in Battery is transformed into light in Bulb. $State_2$ is the preceding state of $transition_{2-3}$ and $state_3$ is its succeeding state. $State_1$ describes the state of electricity at location Battery and $state_2$ at location Bulb. $State_3$ however describes the state of light at location Bulb. The annotation USING-FUNCTION in $transition_{2-3}$ indicates that the transition occurs due to the primitive function "**create** light" of Bulb.

3 Situating Natural Language Understanding

We now return to the task of natural language understanding and show how situating it in the design situation redefines classical problems such as ambiguity, indirectness, and incompleteness. We also show how the design situation suggests workable solutions to these linguistic problems. Real world tasks such as designing physical devices from written requirements specifications provide a context which refocuses many of the linguistic problems that have been central to the field,

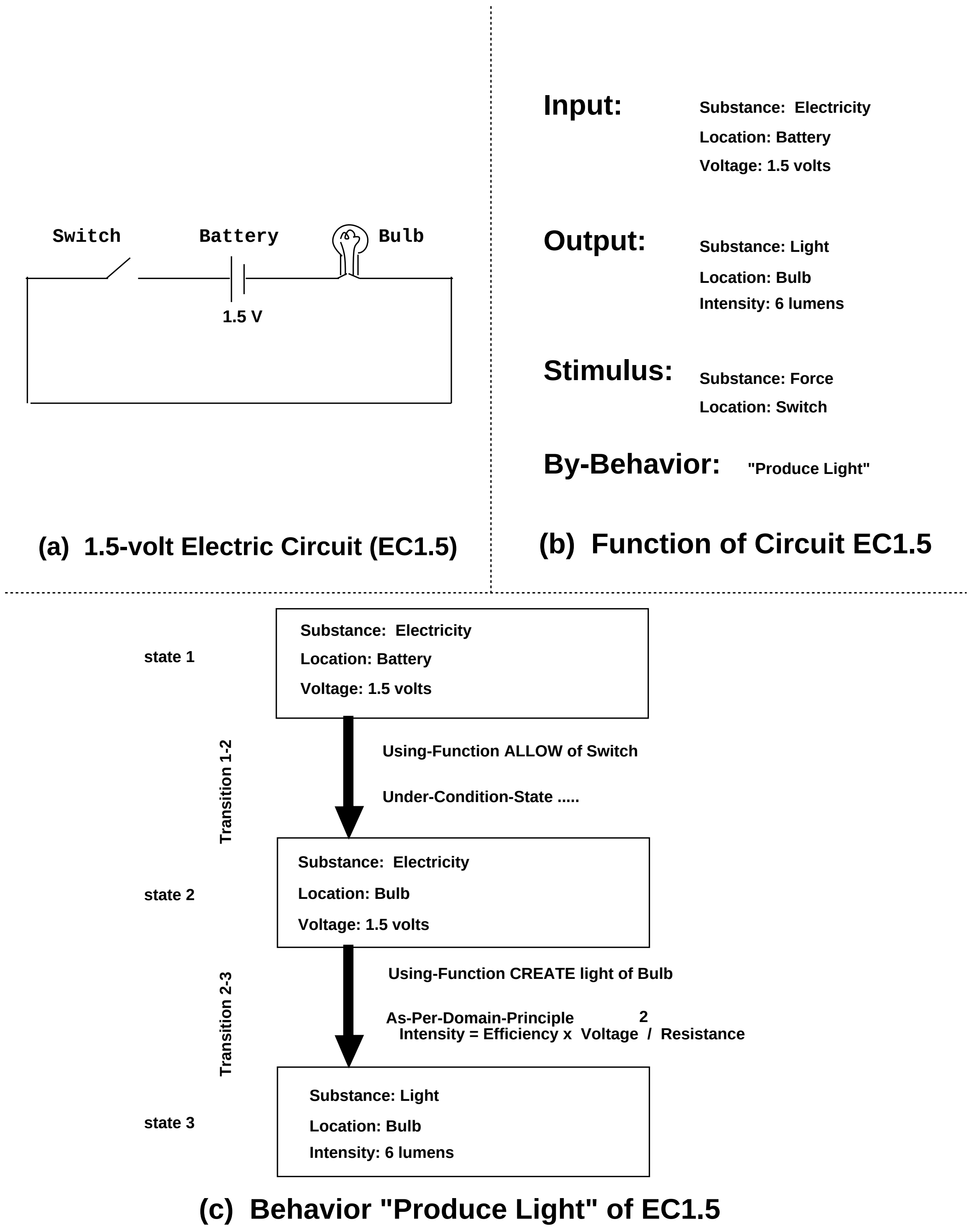

Figure 1: SBF Model of A 1.5-volt Electric Circuit (EC1.5)

allowing us to consider novel solutions to time-worn yet unresolved problems. It also allows us to consider problems particular to texts that are currently hampering efforts to develop robust text understanding systems.

In taking this situated approach to language understanding, we have found that linguistic problems and the problems of large texts that are inherent to written design requirements actually become problems which require reasoning about the design of the device. Requirements specifications are notoriously confusing and incomplete, providing poor articulations of the design requirements. In the KA project, we have encountered the following problems in requirements specifications:

- **Ambiguity.** The natural language surface form has multiple mappings into a conceptual representation of the device.
- **Incompleteness.** The natural language surface form fails to articulate a design requirement.
- **Indirectness.** The natural language surface form indirectly refers to a design requirement.
- **Underspecification.** The natural language surface form does not indicate certain relationships between design requirements.

Because research in natural language understanding has so decidedly separated the problems of linguistic analysis and sentence understanding from the other problems that must be resolved in the meaningful interpretation of texts, the linguistic solutions that have been proposed in most text understanding systems have been severely limited. For quite some time, the conventional wisdom has been that problems in natural language understanding are best addressed by constraint-based methods that employ a knowledge of natural language's distributional structure and rules of combination (Chomsky (1957)). In KA, we take a different approach where the design situation provides the knowledge and results of applying its reasoning methods that are then used to solve the above problems in language understanding. Below, we take each of these problems individually, identifying the conditions under which they occur and elaborating on their consequences.

In general, the mapping of language form to design requirements is *ambiguous*. For example, words as seemingly clear as "input" have multiple mappings into a SBF representation of function. The "input" to a device may refer to either an external stimulus (e.g., a force on a switch that initiates some causal behavior) or some entity that is transformed by the device (e.g., a substance like electricity consumed by an electrical device such as a light bulb). Because such ambiguities crop up often, requirement specifications written in natural language frequently specify several devices, rather than a single, unambiguous device. Leaving these ambiguities unresolved or failing to resolve them correctly cause a system to waste its resources pursuing a number of fruitless design efforts.

In general, requirement specifications written in natural language are also *incomplete*. For example, although electrical devices require a source of power, design problems can fail to mention how this power is to be derived. The device could use batteries, plug into an electrical outlet, or resort to some other electrical power source. It is critical that the system infer the appropriate design requirement because each of these designs would entail different structures and would be operable under differing conditions.

In general, requirements specifications state the design requirements *indirectly*. They refer to aspects of the device that are only distantly related to its principal features. For example, specifications for computing devices often identify principal components, specify the inputs and outputs of these components, and delineate their connectivity, but fail to define the big picture, viz., the general functions of the device. The writers of such requirements specifications can usually point out specific statements about the inputs and the outputs of the components, for example, that indicate the general functions of the device, but in no way are these requirements indicated in the natural language surface form. The system must be able to use the indirect statements given to infer the design requirements because, failing to do so, it would be unable to pursue a design solution.

In general, natural language specifications are *underspecified.* They identify detailed device requirements without articulating how these requirements relate to one another. For example, although baud rate, size of an information packet, and frequency of transmission have a well-defined relationship to one another in a computer network, requirements specifications for computing devices rarely, if ever, mention this relationship. A superficial analysis of the natural language surface form would produce three separate requirements (one for the baud rate, one for information packet size, and one for the frequency of transmission), entailing an extremely inefficient problem decomposition. If the system is to pursue designs efficiently, it must combine these disparate requirements into a coherent specification of the design.

In order to map requirement specifications to useful functional descriptions in the SBF language, KA must effectively resolve ambiguity, fill in missing details, identify the relevance of indirect statements, and combine related information. To do so efficiently, KA uses memory, comprehension, and problem solving processes in addition to purely language processes. In this way, the design situation in KA provides a robust context in which effective comprehension of natural language becomes feasible.

4 The KA Architecture

KA is a *case/model-based text interpretation and design problem solving system* which accepts a requirements specification written in English and produces a design expressed as a structure-behavior-function (SBF) model, which meets the design requirements. The functional architecture for KA is illustrated in Figure 2. It consists of several knowledge sources containing syntactic, conceptual, and episodic knowledge and employs memory, comprehension, and problem solving processes in addition to a language process.

4.1 Knowledge Sources in KA

The component processes in KA use different knowledge sources to bring about the capabilities to the system. The knowledge sources are,

- The **Lexicon** contains knowledge of the words in the language. It provides such information about words as their grammatical category, other linguistic markers such as number and person, and their conceptual meaning.
- **Syntactic knowledge** is knowledge of the grammar of the natural language. This is used by the language process to break up the input sentences into grammatical units such as phrases and clauses.
- **Conceptual knowledge** in KA is the knowledge of substances, components, their properties, states, and functions. Conceptual knowledge is essentially the content of the domain of physical devices and is expressed in the SBF ontology.
- **Case/Model Memory** is the episodic memory of past design cases and case-specific structure-behavior-function (SBF) device models. Like conceptual knowledge, this knowledge too is represented in the SBF ontology and indexed by items in the ontology such as functions and properties of substances and components.

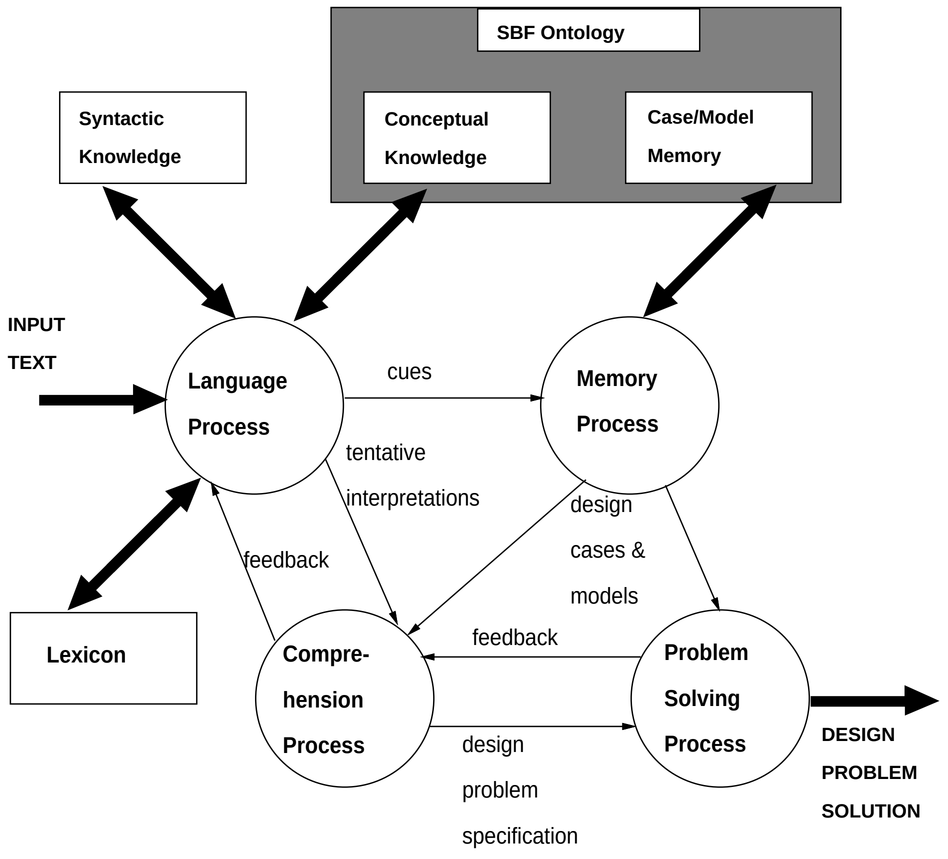

Figure 2: KA System Architecture

4.2 Processes in KA

Below, we describe in detail each of the processes in KA that utilize the knowledge sources to accomplish the task of KA.

4.2.1 Language

If the KA system is to effectively comprehend natural language texts, it must be able to resolve the different types of ambiguities (e.g., lexical and structural ambiguities), that arise in written texts. The language process in KA uses an early-commitment processing strategy with robust error-recovery to resolve word sense ambiguities (Eiselt (1987, 1989)). This mechanism has proved itself to be quite effective. Its early-commitment strategy provides the system with the ability of pursuing a tentative interpretation of the discourse. This allows the system to discover the entailments of this line of interpretation, bringing other processes on-line early in the course of language understanding. In situations where the early decision is incorrect, the error-recovery mechanisms may use feedback from the comprehension (or problem solving) process to reactivate a previously retained alternative interpretation. The output of the language processor serves both as a set of cues for the memory process and as a tentative interpretation for the comprehension process.

The language process consists of two components, a parser which produces syntactic structures and a semantic network that produces conceptual interpretations. Consistent with the early-commitment processing strategy, the semantic network resolves word-sense ambiguities by considering processing choices in parallel, selecting the alternative that is consistent with the current context, and deactivating but retaining the unchosen alternatives for as long as space and time resources permit. If some later context proves the initial decision to be incorrect, retained alternatives are reactivated without reaccessing the lexicon or reprocessing the text (Eiselt (1989)).

4.2.2 Memory

The memory process retrieves and stores design knowledge from an episodic memory that contains both design cases and case-specific device models. Design cases specify a design problem encountered by KA in the past and its corresponding solution. A case-specific SBF model of a known device specifies the causal behaviors that explain how the structure of the device produces the device functions. In order to ensure effective retrieval, the cases are indexed by the functional specification of the stored design and the SBF models are indexed by the cases.

4.2.3 Comprehension

The comprehension process provides feedback to the language process based on the retrieved cases and associated SBF models. It also generates new SBF device models by retrieving and adapting previously encountered design cases and their SBF models. Based on information provided by the language process, the retrieved design and its model are adapted using generic design repair plans. The comprehension process selects these modification plans by using the differences between the functions of the new device and the functions of the retrieved design as an index.

4.2.4 Problem Solving

The problem-solving process performs *function-to-structure* design tasks. It accepts a *functional* specification of the desired design as input and produces a *structural* specification that realizes the specified function as output. Both the specification of function and the specification of structure are articulated in terms of the SBF language.

The problem-solving process begins its task by soliciting the memory process for a case that most closely matches the functional specification of the desired design. The memory process returns an SBF model which problem-solving uses to adapt the design's structure so as to meet the given functional specification. Model-based diagnosis is used to identify the modifications needed to the retrieved design and repair plans are used to perform these modifications to the design's structure. Once the structural modifications are completed, this new design is verified by a qualitative simulation of its SBF model and produced as a solution to the design problem.

It may be noted from Figure 2 that the language process is neither a front-end to the problem-solving or comprehension processes, nor does it perform the entire understanding task by itself. What we have in KA is a highly interactive architecture in which language, memory, problem solving, and comprehension processes, each with its own sources of knowledge and its own capabilities, cooperate with each other, feeding back one's results and decisions to others, in order to arrive at an iterative solution to the overall problem of designing physical devices given their natural language specifications.

5 Investigation 1

In our first investigation, we examined whether KA's design situation and ability to reason about the design of physical devices could help resolve ambiguities in design specifications as well as infer unarticulated requirements. In this investigation, we sought to take advantage of the SBF ontology and KA's memory of past design cases and associated case-specific SBF models. Our results were very positive. They indicated the following benefits:

- Grounding the language process' conceptual knowledge in the SBF representation guarantees that decisions about the consistency of conceptual interpretations are made on the basis of their consistency as designs represented in the SBF ontology.
- Providing feedback to the language process in the form of past design cases, represented in the SBF language, ensures that the conceptual interpretations are compatible with past design experience and allows unarticulated design requirements to be inferred from previous design problems.

In this section, we discuss how each of these benefits was accrued in the implementation of KA. First, we present a sample text that is both ambiguous and incomplete along with its desired mapping into the SBF representation. Then, we demonstrate how KA performs the mapping from the natural language description to the functional specification of the desired design.

5.1 The Task

Figure 3 shows a sample *input* specification and Figure 4 shows its corresponding mapping, an SBF description of the desired design.

This simple example illustrates two general problems of natural language understanding that KA must solve. First, the requirement specification in Figure 3 is ambiguous. It states that the "input" to the device is a "small force on the switch," but in this domain, "input" can refer to one of two things, either an external stimulus (i.e., a force on a switch that initiates some causal behavior) or some entity that is transformed by the device (e.g., a substance like electricity consumed by an

Figure 4: The Output for Text 1 - SBF Functional Specification

electrical device). To produce the correct interpretation in Figure 4, KA must determine that in this instance "input" refers to an external stimulus by successfully resolving this lexical ambiguity.

Second, the requirement specification in Figure 3 is incomplete. There is no mention of how the device is to be powered. The description could be specifying a design which uses batteries or one which plugs into an outlet. In order to produce the functional specification in Figure 4, KA must infer that the design should use batteries. KA must effectively resolve the ambiguity and fill in the missing requirement in order to perform the mapping from the requirements specifications in Figure 3 to the functional description in Figure 4.

5.2 The Process

Briefly, KA achieves this mapping by performing the following actions iteratively. First, it reads a text word by word and sentence by sentence, building a syntactic and conceptual interpretation of the text. Structural and lexical ambiguities encountered along the way are resolved by combining information from lexical, syntactic, and conceptual knowledge. The result of this language process is a representation of the meaning of the text in the ontology of the domain captured by the SBF language. For the text in Figure 3, for example, the interpretation is a representation of a *tentative* functional specification of the device.

Second, the functional specification is sent to the memory process and the comprehension process. The memory process searches the case memory and retrieves a set of cases which at least partially match the tentative functional specification. These retrieved cases are sent to the comprehension process. The comprehension process uses the differences between the tentative specification of the new device and the specification of the retrieved cases (if any) to provide feedback to the language process. This feedback is in terms of the differences between the two specifications.

Third, this feedback is sent to the language process. The language process combines the feedback with its current tentative interpretation, filling in missing details. The parts of the current interpretation that are inconsistent with the feedback are re-examined and other alternatives are considered. It is in this way that the recovery from erroneous decisions in the resolution of ambiguities can be made. The language process communicates the results of its decision-making in the form of a new functional specification.

Once this text interpretation is consistent with the design experience, a complete interpretation is produced and sent to the problem solver. Below, we discuss these steps in further detail.

5.2.1 Producing a Tentative Interpretation

The language process begins by performing a syntactic parse of a sentence in the input. Parsing resolves any ambiguities in the word's syntactic categories and the sentence's syntactic structure. Once this parse has been completed, the concepts denoted by the content words[2] found in the lexicon are sent to semantic network. Choosing the syntactic categories of words, the parser, in effect, selects the word meanings that will be considered by conceptual processing. Only those concepts that are consistent with the syntactic categories chosen are sent to the semantic network.

After receiving all of the concepts denoted by the content words in the sentence, the semantic network begins by activating a semantic node for each concept. Since lexically-ambiguous words such as "input" denote multiple concepts, multiple nodes are activated by the appearance of words such as "input" in the sentence.

The semantic network identifies relevant conceptual relations between these active concepts (nodes) by *marker passing*, a standard method of inference used in semantic networks (Charniak (1981); Hendler (1986); Norvig (1989)). *Marker passing* identifies complex conceptual relations (paths) in the semantic network that connect active concepts (nodes), producing them as inferences. *Marker passing* is achieved by (1) initializing a marker for each active node, (2) sending copies of the marker to all of the nodes that maintain semantic links with the active node, and (3) continuing semantic link traversal until a maximum path length is reached. Semantic links correspond to primitive conceptual relations between concepts (e.g., part-whole relations, instance relations, property relations) and are the basic elements from which complex conceptual relations are formed. Inferences are produced when "marked" sequences of primitive conceptual relations (paths) that connect active nodes are identified. For example, in the network in Figure 5, an inference is generated for the "marked" sequence (**Circuit**, instance, Device, part, Function, part, Input, subject, Be, object, **Force**) which connects the active nodes **Circuit** and **Force**.

After a set of inferences (paths) has been proposed by *marker passing*, the semantic network begins resolving ambiguities in the interpretation. Ambiguities are marked by the words that evoke them. For example, "input" specifies that any interpretation may include either the concept **input** or the concept **stimulus** but cannot include both. In other words, either the node **input** or the node **stimulus** can appear in the paths that make up the final interpretation, but both cannot. These ambiguities are resolved by consistency-checking.[3] Consistency-checking is done in accordance with the SBF ontology. The semantic network identifies inferences that are deemed inconsistent by its SBF representation, resolves the inconsistency in favor of the inference that has the most in common with the current interpretation, and places the other on the retained list. Retained inferences can be recalled if the situation warrants it.

[2]Words are partitioned into two classes: *Function words* and *Content words*. Function words (e.g., Prepositions, Articles, Conjunctions, etc.) are also called *closed class words* because the addition of new words to the class is rare. Content words (e.g., Nouns, Verbs, etc.) are also called *open class words* because the addition of new words occurs frequently.

[3]See Wilks (1973) for a discussion of how semantic consistency-checking may be used to resolve ambiguity.

Figure 5: Resolving Ambiguity in the semantic network

To illustrate how this works, consider the network in Figure 5. In this network, there are two inferences that relate the active concepts **Circuit** and **Force**. Each contains a concept denoted by "input". The path (**Circuit**, instance, Device, part, Function, part, Input, subject, Be, object, **Force**) contains the concept *Input*, and the path (**Circuit**, instance, Device, part, Function, part, Stimulus, subject, Be, object, **Force**) contains **Stimulus**. As defined by the SBF ontology and representation, a force cannot be both the input and stimulus to the circuit, so these inferences are deemed inconsistent. The semantic network recognizes this by noting that the paths relate the same two active nodes. This simple method of inconsistency recognition is made possible by the fact that the network is specified in such way that it conforms to the SBF ontology. The network resolves this ambiguity in favor of the **input** inference because a number of inferences proposed by the network involve the concept of **input** but only a few include **stimulus**. The language process has a set of such heuristics for resolving semantic ambiguities by selecting between paths in the semantic network (Eiselt (1989)). Finally, **input** is kept in the current interpretation, and the inference containing **stimulus** is retained as noted by Figure 5.

Once consistency-checking has been completed and the ambiguities resolved, the inferences articulate a tentative functional specification which is consistent where consistency is defined by the principles of the SBF representation and ontology.

5.2.2 Model-based Retrieval and Comprehension

The tentative functional specification produced by the language process is sent to both the memory and comprehension processes. The memory process searches the case/model memory for a case that most closely matches the given specification. Given the tentative specification discussed in this example, the memory process finds a case that describes a device producing light with color blue and intensity 8 lumens. This is a partial match of the desired device since the specifications differ only in the intensity of light produced. This case is sent to the Comprehension process.

The comprehension process compares the tentative specification and the retrieved specification and notes the difference in light intensity. In an attempt to explain what about the behavior of the device causes this difference in intensity, the comprehension process performs a diagnosis on the retrieved case to determine the factors which contribute to the intensity of the output. In doing

Figure 6: Recovering from Error in the semantic network

Consider how this works given the network in Figure 6. The semantic network identifies an inconsistency between the "input is force" inference denoted by the path that travels from **Circuit** through **Input** to **Force** and the "input is electricity" inference denoted by the path from **Circuit** through **Input** to **Electricity**. Under the SBF representation, devices may have only one input. Both this inconsistency and the inconsistency between the retained inference "stimulus is force" and the "input is force" inference serve to make the "input is force" inference unlikely. To recover from this erroneous inference, the semantic network places the "input is force" inference on the retained list, recalls the "stimulus is force" inference, and keeps the "input is electricity" active, successfully resolving the lexically ambiguous word "input". In a similar way, the inference "electricity is provided by a battery" is also incorporated into the current interpretation.

Figure 7: Text 2 - Sample Requirements Specification

To successfully understand the passage in Figure 7 as a design specification, one must be able to determine the function of the device being described, its inputs, and its outputs. However, none of these characteristics are explicitly described in the text. The text describes the device (referred to as “the system”) in terms of its components (e.g., “computer A”, “computer B”), their connectivity (i.e., “Computer A shall send ... to Computer B”) and types of information they transmit (e.g., “request packet”). Nothing is stated about the function of “the system”. Its inputs and outputs are not even referred to. To achieve the mapping from the English description in Figure 7 to the

[4]Specific design details have been masked to protect proprietary information of our sponsor, Northern Telecom, who supplied this example as a test case.

Figure 8: The Output for Text 2 - SBF Functional Specification

functional specification in Figure 8, KA must be able to generate functional requirements from alternate information sources, using the information provided in the text (e.g., K byte request packet, L byte response packet) as constraints on the generation.

Before KA can use this information to constrain generation, however, it must determine its relevance to the function of the device. In the text, it is unclear what a "request packet" and a "response packet" have to do with the function of the device. On the face of it, they appear to have no relevance to the device's inputs and outputs, being simply relegated to its internal workings. An understanding of "requests" and "responses" reveals, however, that these information packets are the inputs and outputs of the device as indicated in Figure 8. If KA is to make use of such indirect statements about functional requirements, it must use its specific, episodic knowledge about computing networks to infer that a *request* message passed from one computer to another is the input to the system, and the message sent in *response* is the output.

Although the text devotes significant attention to design details such as the frequency of transmission "every M seconds", the size of the request packet "K byte", and the size of the response packet "L byte", it leaves the relationship between these details unspecified. Identifying the relationships between these design details is critical to producing a successful design. Using the frequency of transmission in combination with the size of the information packets, KA can infer the appropriate baud rate for the link between the system's two computers. Without this baud rate specification, KA may select a link that is too slow, producing an unusable design, or it may select a link that is much faster than needed producing an expensive and possibly unbuildable design. If a competent interpretation of this text and a successful design solution are to be produced, these design details must be combined into a coherent specification of the "XXXX" link's baud rate.

6.2 The Process

To produce the specification in Figure 8 from the text in Figure 7, KA must generate a functional specification that is constrained by the information provided in the text. To make use of the this information, KA must infer the relevance of indirect statements, and combine detailed design requirements into coherent specifications.

Briefly, KA performs the mapping from the text in Figure 7 to the functional specification in Figure 8 in the following manner. First, using its memory of past design cases and case-specific SBF models, KA employs a complete SBF model as a baseline from which the relevance of indirect statements about the function of the device can be inferred. The memory process extracts a relevant model from its case/model memory using the bits and pieces of a tentative interpretation produced by the language process and sends it to the comprehension process. This model is fed back to the language process. Using this model as baseline, the language process employs its inference generation capability (i.e., marker passing) to identify the relations between the feedback and the concepts specified by the text. Once the language process has finished its inference generation, it produces a tentative functional specification of the design which is sent to the comprehension process.

Second, KA performs model-based adaptation on the SBF model, generating a new case-specific SBF model that is consistent with the information provided in the tentative functional specification. The comprehension process identifies distinctions between the tentative functional specification and the SBF model. Then, it uses these distinctions to modify the SBF model. During adaptation, the comprehension process modifies only those aspects of the stored model that conflict with the tentative specification. This leaves a significant number of design details unaffected. In effect, design details are transferred from the stored SBF model to the new device model.

Third, during adaptation, KA identifies those distinctions that require changes to the new device's structure and adapts the tentative design specification accordingly. The comprehension process notes differences between the stored device model and the tentative functional specification such as a difference in output and input that may require changes in the structure of the new device. To accommodate these changes, it selects generic modification plans that modify device structure. Model-based diagnosis is performed on the stored device model, and the necessary modifications to the device structure are determined. Using the products of model-based diagnosis and the selected generic modification plans, the comprehension process adapts the tentative design specification such that it includes a structural description that is consistent with the functional specification.

Once this process of adaptation is completed, the new design is sent to the problem-solving process for verification and possibly further design. Then this new case is stored in KA's memory of case-specific models for later reuse. Below, we discuss each of these steps in detail.

6.2.1 Inferring the Relevance of Indirect Statements

The memory process begins by sending a relevant SBF model to the comprehension process which feeds it back to the semantic network in the language process. The semantic network activates the model's corresponding concepts and conceptual relations. For example, in the subsection of the semantic network displayed in Figure 9, the concepts **Old-Device**, **Y Byte**, **Response message**, and **Response** and the primitive conceptual relations that relate these concepts (e.g., *parameter*, *instance*, *part*) are activated by feedback from the Comprehension process.

Much like the previous example, the input is then parsed and the content words of each sentence are passed to the semantic network which initiates marker passing at each word's corresponding concept. Using the feedback as a bridge, the semantic network identifies conceptual relations between the concepts activated by the text and constructs a new set of inferences. The new inferences relate concepts specified in the text to the functional specification of the new device. Finally, a tentative functional specification of the new device is produced from these new inferences and sent to the comprehension process.

To see how the feedback acts as a bridge between the concepts activated by the text, consider the subsection of the semantic network displayed in Figure 9. In this semantic network, the concept

Figure 9: Identifying the Relevance of *response* and *L Byte*

L Byte is activated by the appearance of "L byte" in the input text in Figure 7, **Response** is activated by the appearance of "response", and **New-Device** is activated by the appearance of "system". Using only these active concepts, the semantic network would be *unable* to identify critical conceptual relations such as that between **L byte** and the **Output** of the **New Device** because the active concepts **L byte** and **New Device** are only distantly related to each other. Basing its decision on the length of the path between the two concepts, the semantic network would deem it unlikely that the text intends to relate these concepts without further evidence.

However, when the semantic network begins with feedback-activated concepts such as **Response message**, conceptual relations such as that between **L byte** and **New Device** as well as that between **Response** and **New Device** can be identified. The concepts activated by the text are more closely related to the concepts activated by feedback than they are to each other, so the semantic network can identify conceptual relations between the text-activated concepts and feedback-activated concepts (as indicated by the active paths in Figure 9). This produces inferences that serve to relate the text-activated concepts, inferences that identify the relevance of concepts such as **response** and **L byte** to the function of the **new device** .

6.2.2 Generating a New Functional Specification

Given a tentative and underspecified functional specification produced by the language process, the comprehension process compares the SBF model and this underspecified functional specification to determine the distinctions between the two device descriptions. It notes distinctions that are extremely significant such as the distinction between the size of the response package (*L bytes* versus *J bytes*) and those that are less significant such as the difference in the names of the components (*A* versus *C*).

Once all of these distinctions have been collected, the comprehension process begins adapting the stored SBF model. It modifies the component names such that they are consistent with the new functional specification, changes the sizes of the response package and request package, etc. In doing so, it transfers a large amount of the SBF model of the known device to the SBF model of the new device. For example, it transfers the types of the components in the old device to components of the new device. The result is that all of the design details are filled in, and a significant number of assumptions are made. The comprehension process assumes that the new device has the same behavioral descriptions as the stored device and the same structural description.

6.2.3 Identifying Relationships between Design Details

During the adaptation of the stored SBF model, the assumption that the structural specifications of the new and stored designs are equivalent is examined. The comprehension process considers each of the differences it has identified between the new specification and the stored specification, looking for those differences that may require modifications to the device structure. Differences that are particularly relevant are differences in device inputs and outputs. For example, in this example, the distinction between the size of the new design's output and the stored design's output (i.e., *L bytes* versus *J bytes*) imposes new constraints on the structure of the new design. The comprehension process collects these difference and orders them with respect to their priority.

Examining them in order of their priority, the comprehension process retrieves generic modification plans that rectify the differences between the new design specification and the stored design specification by adapting the stored SBF model. Generic modification plans are selected by the type of differences they reduce. In achieving their ends, generic modification plans manipulate, delete, and augment device structure. They include component replacement, substance substitution, parametric modification, component deletion and component insertion and cascading.

After the comprehension process has received the generic modification plans from the memory process, it begins to diagnose the new model's failure, in this particular example, its failure to produce the desired output. It is during diagnosis that the comprehension process recognizes the relationship between the design details *every M seconds* and *L byte response packet*. The comprehension process correctly assumes that since the new design specification identifies an output which differs from the stored specification, the new design's current structural specification will fail to produce its specified output. The design produces the output of the stored design because the comprehension process has assumed that they have equivalent structural specifications. While investigating the causes of the new design's failure, the comprehension process identifies the relationship between the frequency of transmission (i.e., *every M seconds*), the size of the response packet (i.e., *L bytes*) and the baud rate of the link component. Using the qualitative relations specified in the stored SBF model, it notes that baud rate of the link component limits the amount of information that can be transferred at a particular frequency. It concludes the baud rate of the current link component is too low and that increasing the baud rate of this component would provide for the size of the response packet in the new design and the desired frequency of transmission.

Given the diagnosis and the generic modification plans, the comprehension process "repairs" the structural specification of the new design, using component replacement. It replaces the link component in the stored SBF model with a link component that has a higher baud rate. This modification appears in the specification of structure in Figure 8.

7 Investigation 3

Natural language texts are used to achieve a variety of communication goals at different stages in the design process. For example, at later stages in the design, customers use English texts to communicate feedback to designers. Understanding these texts is essential for redesigning a product to meet customer needs. In keeping with our goal of "situating" natural language within design, our third and most recent investigation examined whether KA could interpret design feedback as well as design requirements.

We chose a problem in the design of a reaction wheel assembly for the Hubble space telescope and examined KA's ability to understand user feedback. We demonstrated that KA was in fact able to understand such user feedback and use the information it could extract from the feedback to redesign the reaction wheel assembly. This cost effective redesign was made possible by

1. KA's repertoire of plans for incremental redesign to correct the malfunction, and
2. KA's ability to precisely identify the part of the design that is to blame for the malfunction.

This investigation demonstrated that KA's theory of situated natural language understanding that was effective in understanding initial design requirements also serves to accomplish interaction with the customer and iterative redesign to meet customer requirements. This successful demonstration shows the extensibility of KA's theory of situated understanding.

7.1 The Task

In this investigation, we focused on interpreting and acting upon user feedback. The task is to redesign a malfunctioning component given feedback written in English text. This overall task decomposes into language interpretation and redesign. Given a description of an undesired output of the device and a model of the device's structure, function, and behavior, the redesign task is to modify the structure of the device so that it does not produce the undesired output. So as to successfully interpret a passage as a redesign problem, the system must be able to identify the relevant device and produce a description of the undesired output. A description of the undesired output includes identifying the device component that produces this output and a description of the output in terms of the relevant substance properties.

Let us consider the specific problem we examined in this third investigation: redesigning the Reaction Wheel Assembly (RWA) aboard the Hubble Space Telescope (Keller, Manago, Nayak and Rua (1988)). The Hubble Space Telescope contains four RWA's; a small portion of one is shown in Figure 10. The desired function of the RWA is to make the telescope point at a chosen area of the universe. The structure of the RWA consists of a rapidly spinning rotor mounted on a shaft. The functioning of the RWA is based on the law of conservation of angular momentum. The direction of the telescope is changed via a signal from Earth sent to the motor which changes the amount of power supplied to the motor. This causes a change in the motor's angular momentum which in turn affects the angular momentum and angular velocity of the shaft. Due to the conservation of angular momentum, the angular momentum of the telescope as a whole changes in the opposite direction. When the telescope nears its desired orientation, a change in the angular momentum of the telescope in the opposite direction reduces the telescope's angular velocity to zero. The vector sums of the angular momentum imparted by the four RWA's enable a rotation of the telescope about any axis.

A common problem in the operation of devices like the RWA arises due to friction in the bearing assemblies. The load on the bearings due to the rapid spin of the rotor causes deformation of the bearing balls which results in increased frictional forces in the bearing assembly. This causes generation of heat in the bearing assembly. The increased temperature in the bearing assembly is an extremely undesirable behavior.

Figure 11: Text 3 - Sample Requirements Specification

In our third investigation, we examined this redesign problem taking the the text in Figure 11 as the typical type of feedback that a user would provide. To produce the appropriate redesign specification given the sentence in Figure 11, KA must identify that

1. The relevant device is the RWA.
2. The components producing the undesired output are the ball bearings.
3. The undesired output is heat whose magnitude is too high.

7.2 The Process

KA performs the mapping from the user feedback in Figure 11 to the specification of the undesired behavior. It reads the sentence word by word and builds a syntactic and conceptual interpretation of the sentence. The result of this language process is a representation of the meaning of the text in terms of the SBF language.

Next, diagnosis determines the component parameter that is responsible for the undesired behavior and the parameter modification desired. In the RWA example, the diagnosis accepts as input a specification of excessive heat at the bearing assembly and returns as output the component parameter, the size of the ball bearings, and the parameter modification desired, an increase in the size of the the ball bearings.

Third, The repair task replaces the component with a component that has the desired parameter value. In the RWA example, this task replaces the old ball bearings with new larger ball bearings, thereby redesigning the RWA to eliminate the problem of excessive heat in the bearings which was noted by the user in the feedback.

8 Discussion

The current implementation of KA is able to extract both functional and structural specifications from design requirements or from user feedback written in English. To do so effectively, it overcomes many of the difficult problems that face any natural language understanding system. It can resolve semantic and syntactic ambiguities, correctly infer unarticulated statements, identify the relevance of indirect statements, determine the unspecified topic/theme of a passage from its constituent statements, and combine disparate statements into coherent interpretations.

We have tested the KA system with examples of design specifications and user feedback in several different domains including electrical circuits, computer networks, and mechanical dynamic systems. Many of the texts used in these tests, such as the one in Investigation Two came from real-life examples from the industry. We have investigated similar domains in our ongoing work on design (Bhatta and Goel (1992, 1993); Goel (1991a, 1991b)) where we have demonstrated that our design system can acquire knowledge about new devices in the form of cases and models as a result of design problem solving. Since KA has been shown to provide solutions to problems (such as ambiguity) in understanding design specifications of new (yet related) devices by using such previously acquired design knowledge, its methods for language understanding in the design situation show a strong promise of bootstrapping and scalability.

Apart from providing a new approach to attacking the natural language understanding problem, the task that KA addresses is also a problem of high practical relevance. There is a great need to build automated systems that can take design specifications for both physical hardware and software "devices" and represent them in formal ontologies that are comprehensible to the designers no matter whether they are human design teams or automated design systems or computerized design aids. KA is a demonstration that it is possible to build such systems. In addition, KA's ability to infer unarticulated requirements from its cases and models makes it a useful model of making sure that the "common" design knowledge that is supposed to be shared between customers and designers in each domain is included in the design process, thereby reducing the need for expensive redesign and customer dissatisfaction at later stages.

8.1 A Critique of KA

However, our work also raises certain issues addressing which is part of ongoing work in the KA project. These issues are representative of the open problems that exist in the field of natural language understanding and, more generally, artificial intelligence. They involve limitations in both the input that KA can accept and the outputs that it can produce. The system's representations also have some limitations. In addition, restrictions on interaction limit the extent to which system components can avail themselves of the system's resources. In hopes of clearly elucidating the capabilities of our work as well as its limitations, in this section, we discuss each of these problems and their impact. We close with a short summary of recent work within our research group that has sought to remedy these problems and outline our research plans for addressing them in the future.

8.1.1 Input

While KA is able to master many of natural language's lexical and structural ambiguities and ascertain aspects of meaning that are left unarticulated in texts, certain classes of natural language discourse remain beyond its reach. KA's method of resolving ambiguity relies on a text that is internally consistent. Single interpretations are produced when their alternatives are found to be inconsistent with a combination of linguistic evidence and the evidence provided by KA's memory of case-specific models. It is certainly possible, however, for the text to be internally inconsistent. Design specifications may describe a conflicting set of requirements that prevent a consistent interpretation. In this case, KA should communicate these inconsistencies to the user, and possibly pursue an interpretation that reflects a consistent subset of the text's content. Such a capability is beyond the current scope of our work.

8.1.2 Output

Although KA is able to produce both functional and structural design specifications given design specifications written in natural language, there are some types of design information that it is unable to deal with. Examples include design requirements that are not related to the function of the device such as costs, packaging, and aesthetics requirements which are typically part of design specifications that customers provide.

8.1.3 Representation

The semantic network representation used in the language process gives it the ability to precisely articulate the SBF representation and ontology in a such a way that inconsistencies in the interpretations can be easily recognized. However, this precision has its associated cost. The representation affords little generativity. All of the concepts and the relations that hold between them must be explicitly articulated. We would like KA to be able to extract automatically some of this knowledge in the semantic network from the cases and models it acquires through its design problem-solving experiences.

8.1.4 Interaction

The KA architecture ensures that productive interaction occurs between the system components. The memory process lends its ability to select the appropriate case-specific device models to the language process, comprehension, and problem solving process; the language process provides cues that assist the comprehension process; both the comprehension and problem-solving process share many of the same methods. However, the architecture does still maintain a few well-demarcated boundaries that decrease the effectiveness of the system. For example, communication between syntactic parsing and semantic inference is overly restricted. We would like semantic inferences to have a greater effect on syntactic decision making, and increase the influence syntactic decision making has on semantic inferences.

8.2 Recent Work

In recent work in natural language processing, we have sought to rectify some of the problems above. Members of our research group have developed a model of sentence understanding that uses semantic inferences to avoid and recover from errors in syntactic parsing (Eiselt, Mahesh and Holbrook (1993); Mahesh (1994); Mahesh and Eiselt (1994)). Others have developed a model of sentence understanding that uses syntactic evidence to infer semantic interpretations (Peterson and Billman (1994)).

9 Related Work

Our work is related to five different bodies of research in natural language understanding: situated natural understanding, integrated representations for natural language understanding and problem solving, Conceptual Information Processing, the understanding of natural language descriptions of physical devices, and the modularity of "Mind". Below, we describe the relationship between this work and our own.

9.1 Situated Natural Language Understanding

Our research was inspired in part by Winograd's SHRDLU system (Winograd (1973)), which was one of the first successful systems to situate language understanding in problem solving. SHRDLU formed plans for actions in a simulated blocks world based on its interpretation of external commands expressed in English. It explored certain interactions between language understanding and planning, and demonstrated the methodological usefulness of exploiting the constraints imposed by planning on language understanding, and vice versa. Of course, SHRDLU also suffered from a number of well-known problems. For example, it assumed a closed world, it represented knowledge procedurally, it lacked the capability of abstract reasoning, and it also lacked sufficient control over processing.

Since the construction of the SHRDLU system in the late sixties, research in Artificial Intelligence has led to a large collection of new results in the areas of representation of knowledge and control of reasoning. For example, languages for descriptively and explicitly representing models of a physical situation, and methods for revising stored models to meet the specifications of new situations, are of relatively recent origin. We believe that the needed technologies are now ripe for once again investigating situated language processing in the context of problem solving. Specifically, our research seeks to explore and exploit the use of the design situation for natural language understanding.

9.2 Integrated Representations for Natural Language Understanding and Problem Solving

Several attempts have been made to integrate natural language and problem solving using a common representation for both language comprehension and reasoning (Beck and Fishwick (1989); Charniak (1981); Rieger (1976); Simon and Hayes (1979); Wilensky (1983)). Our work continues in this direction by applying functional models and reasoning to the understanding process. The same ontology is used for understanding natural language and reasoning about devices. It is noteworthy, however, that in the past, common representations for language understanding and problem solving have generally implied a unified process for the two tasks. That is, the same process over the same representations is used for both language understanding and problem solving tasks. Our work differs in that while we use a common ontology, the processes for language understanding and problem solving are not identical nor even equivalent. While the language understanding and problem solving tasks in KA support each other and share some subtasks and subprocesses (e.g., memory retrieval), they are distinct in the subtasks they set up and the processes they use.

The goals of Grishman's PROTEUS system (Ksiezyk and Grishman (1989)), which comprehends failure reports, are not unlike our own goals. PROTEUS, however, did not implement diagnosis and repair. More importantly, language understanding in PROTEUS is driven merely by the templates they wish to fill. We are developing a more general theory of language and applying it to extract the information we need for the design process.

9.3 Conceptual Information Processing

Many language understanding theories have used high-level knowledge structures to guide the understanding process. Schank and Abelson (1977), for example, described the use of stored scripts. The script theory represented knowledge about stories as well as story interpretations in terms of temporally-orderered sets of events. It was employed in a story understanding system called SAM (for Script-Applier Mechanism, Cullingford (1978)). SAM identified the temporal relation between two events by assuming that the linear sequence of sentences in a story corresponded to the temporal ordering of events. SAM's ability to apply scripts and produce interpretations depended critically upon this seemingly simple assumption. In KA, we make no such assumption about temporal correspondences between the discourse and the knowledge structures.

In her work, Lehnert proposed an object representation called "Object Primitives" which assist in making inferences about objects described in natural language texts (Lehnert (1978)). Although there is merit in this object-centered representation, in our work, we have found causal relations between substances and components as well as the casual behaviors of devices to be much more effective aids in resolving problems in natural language understanding.

Other work in conceptual information processing has proposed theories of language understanding with an even stronger reliance on specific knowledge structures (Dejong (1983); Lebowitz (1980); Ram (1989); Wilensky (1978)). While these systems have demonstrated deep understanding abilities in small domains, they have not shown how each of the many types of knowledge structures they need for language understanding (Lehnert (1978); Martin (1990)) can be acquired without being hand-coded. As a result, this class of systems for language understanding shows little promise for bootstrapping or scalability.

9.4 Understanding Natural Language Descriptions of Physical Systems

While we believe that our approach to language understanding in the design situation is quite novel, it is not the first time that researchers have tied text understanding to models of physical systems. Lebowitz's RESEARCHER (Lebowitz (1983)), for example, read natural language texts in the form of patent abstracts, specifically disk drive patents, and updated its long-term memory with generalizations made from these texts. What RESEARCHER stored in its memory was a generalized representation of a disk drive, consisting of a *topographic model* of the disk drive which specified its components and the topographic relationships among them. RESEARCHER's knowledge representation scheme was oriented toward objects and their topographic relationships, which was a departure from most natural language understanding systems of that time which had typically focused on actors, events, and causal relationships. RESEARCHER then used this knowledge to aid in the top-down understanding of additional patent texts. However, RESEARCHER's emphasis on components and topographic relationships left it unable to build causal models of the mechanisms described. In other words, RESEARCHER effectively knew how a disk drive was constructed, but it did not know how it worked.

Dyer, Hodges, and Flowers (1987) and Hodges (1993) describe EDCA, a conceptual analyzer which serves as a natural language front-end for EDISON, a naive design problem solver. EDCA uses knowledge of the function of physical devices to produce an episodic description of a device's behavior as described by an input text. This episodic description can then be used to generate a new device model to be integrated into long-term memory. The result is a much more comprehensive understanding of the device's functionality than was possible with RESEARCHER, but EDCA's analysis of the device description is not fully integrated with the processes for generating new device models and incorporating them into memory. EDCA, in other words, is but a front end to

EDISON.

As Selfridge (1989) notes, separating the process of analyzing the input from generating and incorporating the new model is misguided — the process of understanding a device description *is* the process of building and incorporating a causal model of that device. This is the approach that we have followed in our work, and this approach led us to the KA system which, we believe, corrects the shortcomings of both RESEARCHER and EDCA.

9.5 Modularity of Mind

As well as providing a viable model for solving problems in natural language understanding, our work also addresses a contentious issue in cognitive science, viz., the modularity of "mind". Although it seems clear that language understanding requires cognitive abilities far beyond what pure linguistic knowledge permits, it is unclear in what manner, if any, linguistic and non-linguistic processes interact. Advocates for the modularity of "mind" have argued for a very limited form of interaction (Fodor (1983); Jackendoff (1987)). Others have contended that the interaction is so open-ended as to make any boundaries between linguistic processing and the other cognitive processes insignificant (Marslen-Wilson and Tyler (1989)). We propose a modular processing architecture that contains separate language understanding and problem solving components. These components interact in at least two significant ways. They share common knowledge, and they communicate the results of their reasoning to each other.

What lessons regarding the modularity of 'mind', even tentative ones, can be drawn from our work on KA? KA certainly is modular, but the nature of the modularity depends on the level at which it is analyzed. Modularity in KA can be viewed at the levels of task, process and knowledge, and representation. At the task level, 'language processing' and 'problem solving' are distinct modules, characterized by the types of information they take as input and give as output. At the next level, some of the processes are task-specific but others are shared. Language processing and problem-solving, for example, are both informed by the same memory processes which retrieve episodic and conceptual information. Similarly, some of the knowledge is task-specific and some of it is shared. Only the language processes use lexical and syntactic knowledge, and only the problem-solving processes use knowledge of the the repair plans used for redesigning devices. On the other hand, both the language and problem solving processes employ functional and causal knowledge of devices. Finally, at the level of representation, the language and problem-solving processes share the same SBF ontology for representing conceptual knowledge. Thus, from the viewpoint of KA, the issue of modularity is much more complex than either the orthodox 'modularists', such as Fodor and Jackendoff, or the 'non-modularists', such as Marlen-Wilson and Tyler, suggest.

10 Conclusions

At one level of abstraction, our work on KA leads to the following conclusions regarding the use of the design situation for natural language understanding:

- Ontologies and knowledge structures available in the design situation (and in intelligent design systems) can be used to resolve ambiguities in natural language inputs to the design system.
- Unarticulated design requirements can be inferred from past design problems described in case-specific SBF models.
- The relevance of indirect statements to design requirements can be inferred by using case-specific SBF models as the starting point for the interpretation of a requirements specification.

- A coherent functional specification can be produced from a disparate set of written structural requirements by applying model-based adaptation to case-specific SBF models.
- The above solutions can all be implemented in an integrated yet modular architecture in which different "modules" interact with each other by communicating their results and decisions with each other. Natural language understanding requires such an iterative cooperation between language-specific and non-linguistic reasoning processes.
- All of the knowledge structures needed for the above solutions (apart from the basic ontology of devices and the lexical entries and meanings of new words) can be acquired from previous design problem solving experiences of the system.

Our work on KA provides solutions to the classical problems of language understanding. Building useful systems with an ability to understand "real" natural language input has been an elusive goal for many years now. Well-known problems such as ambiguity, indirectness, and incompleteness of natural language inputs have thwarted efforts to build natural language front-ends to intelligent systems. KA solves these problems by exploiting the knowledge and problem-solving processes in the situation of designing simple physical devices. In addition, since the types of knowledge structures used by KA can in fact be acquired by the results of the very problem-solving processes that the system performs, we have shown that KA can be bootstrapped to understand design specifications and user feedback about new devices using the knowledge structures it acquired for similar devices it designed previously. We strongly believe that such an approach to situated natural language understanding where a problem solving situation, such as device design, provides the knowledge that is needed for language understanding is the right way to pursue research in building scalable intelligent systems with useful abilities to interact in a natural language.

11 Acknowledgements

This work has been supported by the National Science Foundation (research grant IRI-92-10925), the Office of Naval Research (research contract N00014-92-J-1234), and Northern Telecom (research gift). The authors would like to thank their colleagues Sambasiva Bhatta, Andres Gomez de Silva Garza, Kurt Eiselt, Jeff Pittges, and Eleni Stroulia for their efforts and their suggestions and criticisms on the KA project.

References

Beck, H. & Fishwick, P. (1989). Incorporating Natural Language Descriptions into Modeling and Simulation. *Simulation*, **52**(3), 102–109.

Bhatta, S. & Goel, A. (1992). Use of Mental Models for Constraining Index Learning in Experience-Based Design. *Proceedings of the AAAI workshop on Constraining Learning with Prior Knowledge*, 1–10, San Jose, July.

Bhatta, S. & Goel, A. (1993). Model-Based Learning of Structural Indices to Design Cases. *Proceedings of the IJCAI workshop on "Reuse of Designs: An Interdisciplinary Cognitive Approach"*, A1–A13, Chambery, France, August.

Bylander, T. (1991). A Theory of Consolidation for Reasoning about Devices. *International Journal of Man-Machine Studies*, **35**(4), 467–489.

Bylander, T. & Chandrasekaran, B. (1985). Understanding Behavior Using Consolidation. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 450–454.

Chandrasekaran, B., Goel, A. & Iwasaki, Y. (1993). Functional Representation As Design Rationale. *IEEE Computer*, 48–56, January.

Charniak, E. (1981). A Common Representation for Problem-Solving and Language-Comprehension Information. *Artificial Intelligence*, **16**, 225–255.

Charniak, E. & McDermott, D. (1985). *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley Publishing.

Chomsky, N. (1957). *Syntactic Structures.* The Hauge: Mouton.

Cullingford, R. (1978). *Script Application: Computer Understanding of Newspaper Stories.* Ph.D. thesis, Yale University, Department of Computer Science.

Dejong, G. (1983). *Skimming Stories in Real Time: An Experiment in Integrated Understanding.* Ph.D. thesis, Yale University, Department of Computer Science.

Dyer, M., Hodges, J. & Flowers, M. (1987). *Computer Comprehension of Mechanical Device Descriptions.* Technical Report UCLA-AI-87-7, University of California, Los Angeles, Artificial Intelligence Laboratory.

Eiselt, K. (1987). Recovering from Erroneous Inferences. *Proceedings of the Sixth National Conference on Artificial Intelligence*, 540–544.

Eiselt, K. (1989). *Inference Processing and Error Recovery in Sentence Understanding.* Ph.D. thesis, University of California, Irvine, Department of Information and Computer Science.

Eiselt, K.P., Mahesh, K. & Holbrook, J.K. (1993). Having your cake and eating it too: Autonomy and interaction in a model of sentence processing. *Proceedings of the Eleventh National Conference on Artificial Intelligence, AAAI-93,* 380-385, Washington, DC, July.

Fodor, J. (1983). *Modularity of Mind.* Cambridge, MA: MIT Press.

Goel, A. (1989). *Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving.* PhD Thesis, Ohio State University, Department of Computer Science.

Goel, A. (1991a). A Model-Based Approach to Case Adaptation. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 143–148, Chicago, August.

Goel, A. (1991b). Model Revision: A Theory of Incremental Model Learning, *Proceedings of the Eighth International Conference on Machine Learning*, 605–609, Chicago, June.

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

Goel, A. & Chandrasekaran, B. (1989). Functional Representation of Designs and Redesign Problem Solving. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 20–25, Detroit, August.

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

Grishman, R. (1986). *Computational Linguistics.* New York, NY: Cambridge University Press.

Hendler, J. (1986). *Integrating Marker-Passing and Problem-Solving: A Spreading Activation Approach to Improved Choice in Planning.* Technical Report CS-86-01, Brown University, Department of Computer Science.

Hendrix, G., Sacerdoti, E., Sagalowicz, D, & Slocum, J. (1978). Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems*, **8**(3).

Hodges, J. (1993). *Naive Mechanics: A Computaional Model for Representing and Reasoning about Simple Mechanical Devices.* Technical Report UCLA-AI-93-01, University of California, Los Angeles, Department of Computer Science.

Jackendoff, R. (1987). *Consciousness and the Computational Mind.* Cambridge, MA: MIT Press.

Keller, R., Manago, C., Nayak, P. & Rua, M. (1988). Internal Memo. Knowledge Systems Laboratory, Stanford University.

Kintsch, W. (1974). *The Representation of Meaning in Memory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Ksiezyk, T. & Grishman, R. (1989) Equipment Simulation for Language Understanding. *International Journal of Expert Systems*, **2**(1), 33–78.

Lebowitz, M. (1980). *Generalization and Memory in an Integrated Understanding System.* Ph.D. thesis, Yale University, Department of Computer Science.

Lebowitz, M. (1983). RESEARCHER: An Overview. *Proceedings of the Second National Conference on Artificial Intelligence*, 232–235.

Lehnert, W. (1978). *The Process of Question Answering.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Lehnert, W., Dyer, M., Johnson, P., Yang, J. & Harley, S. (1983). BORIS - An Experiment in In-Depth Understanding of Narratives. *Artificial Intelligence*, **20**(1), 15–62.

Mahesh, K. (1994). Reaping the Benefits of Interactive Syntax and Semantics. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 310-312, Las Cruces, New Mexico, June.

Mahesh, K. & Eiselt, K. (1994). Uniform Representations for Syntax-Semantics Arbitration. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, August.

Marslen-Wilson, W. & Tyler, L. (1989). Against Modularity. In J. Garfield, Ed. *Modularity in Knowledge Representation and Natural Language Understanding.* Cambridge, MA: MIT Press.

Martin, C. (1990). *Direct Memory Acess Parsing.* Ph.D. thesis, Yale University, Department of Computer Science.

Norvig, P. (1989). Marker Passing as a Weak Method for Text Inferencing. *Cognitive Science*, **13**, 569–620.

Peterson, J. & Billman, D. (1994). Correspondences between Syntactic Form and Meaning: From Anarchy to Hierarchy. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, August.

Peterson, J., Mahesh, K., Goel, A. & Eiselt, K. (1994). KA: Situtating Natural Language Understanding in Design Problem Solving *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, August.

Ram, A. (1989). *Question-driven Understanding: An integrated theory of story understanding, memory, and learning.* Ph.D. thesis, Yale University, Department of Computer Science.

Rich, E. & Knight, K. (1991). *Artificial Intelligence.* New York, NY: McGraw-Hill.

Rieger, C. (1976). On the Organization of Knowledge for Problem Solving and Language Comprehension. *Artificial Intelligence*, **7**, 89–127.

Schank, R. & Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Selfridge, M. (1989). Toward a natural language-based causal model acquisition system. In W. Horn, Ed. *Causal AI Models: Steps Toward Applications*, 107–128, Hemisphere Publishing Corporation.

Sembugamoorthy, V. & Chandrasekaran, B. (1986). Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In J. Kolodner & C. Riesbeck, Eds. *Experience, Memory and Problem solving*, 47–73, Hillsdale, NJ: Lawrence Erlbaum Associates.

Simon, H. & Hayes, J. (1979). *The Understanding Process: Problem Isomorphs*, volume 1. New Haven, CN: Yale University Press.

Stroulia, E. & Goel, A. (1992). Generic Teleological Mechanisms and their Use in Case Adaptation. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 319–324, Bloomington, August.

Stroulia, E., Shankar, M., Goel, A. & Penberthy, L. (1992). A Model-Based Approach to Blame-Assignment in Design. *Proceedings of the Second International Conference on AI in Design*, 519–537, Pittsburgh, June.

Wilensky, R. (1978). *Understanding Goal-Based Stories.* Ph.D. thesis, Yale University, Department of Computer Science.

Wilensky, R. (1983). *Planning and Understanding, A Computational Approach to Human Reasoning.* Reading, MA: Addison-Wesley Publishing.

Wilks, Y. (1973). An Artificial Intelligence Approach to Machine Translation. In R. Schank & K. Colby, Eds. *Computer Models of Thought and Language*, 114–151. San Francisco, CA: W. H. Freeman and Company.

Winograd, T. (1973). A Procedural Model of Language Understanding. In R. C. Schank and K. M. Colby, Eds. *Computer models of thought and language,* pp. 152-186. W. H. Freeman.

Winston, P. (1992). *Artificial Intelligence.* Reading, MA: Addison-Wesley Publishing.