Understanding System Specifications Written in Natural Language

John J. Granacki, Jr., Alice C. Parker and Yigal Arens
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-0781

Abstract

This paper describes research in understanding system specifications written in natural language. This research involves the implementation of a natural language interface, PHRAN-SPAN, for specifying the abstract behavior of digital systems in restricted English text. A neutral formal representation for the behavior is described using the USC Design Data Structure. A small set of concepts that characterize digital system behavior are presented using this representation. An intermediate representation loosely based on Conceptual Dependency is presented. Its use with a semantic-based parser to translate from English to the formal representation is illustrated by examples.

1 Introduction

This work describes research in understanding system specifications written in natural language. This research was implemented in the PHRAN-SPAN [1] natural language interface to ADAM, the USC Advanced Design AutoMation System [5]. PHRAN-SPAN is used to capture system specifications, with particular emphasis on the abstract behavior of the system being specified.

2 The System Specification Problem

System behavior can be described by one or more processes (independently executing environments) that compete and/or communicate. For example, a serial interface might contain two processes, one to communicate asynchronously over a serial line, and one to communicate synchronously over a parallel bus. A process can: be started asynchronously (whenever specified conditions become true); execute indefinitely; start, suspend and terminate other processes asynchronously; exclude other processes from executing; communicate with other processes; and be asynchronously terminated or suspended itself when some specified conditions become true. These processes may run at different (clock) rates. Processes communicate via shared data, synchronize at critical points, or compete for shared resources. Most existing techniques for describing or specifying these types of processes overly restrict the solution or they cannot be used for complex system specifications.

3 Relationship to Other Research

Previous natural language specifications have been concerned primarily with software systems [3], programs [1] and data types

In this research, we chose a limited domain, the behavior of digital systems. In addition, our system expects a structured input that has been checked for spelling errors and misty pings.

One prior endeavor involved the application of natural language processing as an input to a design system for digital electronics, [7], but this work focused on the construction of a circuit given predefined components rather than specification. Furthermore, use of hyphenated reserved phrases made it more like an application-oriented programming language.

Other recent work, like the UNIX ' Consultant *(VC)* [11], and CLEOPATRA [9], answer questions concerning a given body of knowledge. The PHRAN interface incorporated in this interface was developed for UC.

The research described here differs from UC and CLEOPATRA in that it is *creating* a design entity. To create this representation, semantic knowledge about system behavior has been encoded in the parser's knowledge base.

Overview of the PHRAN-SPAN Operation

The PHRAN-SPAN interface operation is shown in Figure 1.

English sentences are input to PHRAN. PHRAN detects patterns in the sentences and produces concepts, based on its database of pattern-concept pairs. SPAN analyzes these concepts and constructs an internal representation of design data for each sentence. SPAN also informs the user in English how each sentence has been interpreted.

5 Components of the Natural Language Interface

Ib understand the specification of digital systems in restricted English text requires a corpus (a collection of writings, in this case examples) for the domain of these specifications, a representation for the knowledge expressed in the corpus, a formal

[1]PHRasal ANalyser-SPeciflcation ANaly

[3]UNDx is a trademark of Bell Labs.

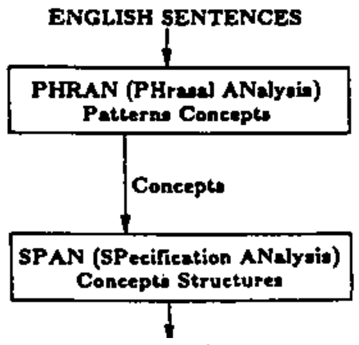

Figure 1: Overall Operation of the PHRAN-SPAN Interface

representation for the behavior of a digital system, and a parsing technique to map the natural language into the formal behavioral representation. Each of these will now be described.

5.1 The Corpus

The corpus for this natural language interface was developed by acquiring actual specifications, having students write specifications and constructing additional examples. These examples were based on a taxonomy of high-level system behavior and a 2000-1- word lexicon which were developed as part of this research.

Examples of the sentences taken from the actual specifications are provided in the following list:

1. *A block of data bytes is transferred by a sequence of data cycles.*
2. *The peripheral equipment shall sample the EF code word which is on the OD lines.*
3. *Each requestor communicates with the arbiter via two lines, a request line and a grant line.*
4. *Select shall be dropped 100 ns after the write is begun.*

5.2 The Corpus' Knowledge Representation and the Parsing Technique

The representation of the knowledge expressed in this corpus was constrained by the choice of a pre-existing semantic-based parsing technique which was implemented by Arens in PHRAN, a PHRasal ANalysis program, [2]. PHRAN is a knowledge-based approach to natural language processing. The knowledge is stored in the form of pattern-concept pairs. A pattern is a phrasal construct which can be a word, a literal string (e.g. Digital Equipment Corporation), a general phrase such as

<component><sends><data>to<component>

and can be based on parts of speech, for example, < noun-phrase > < verb >.

Associated with each phrasal pattern is a concept. The pattern-concept pair (PCP) encodes the semantic knowledge of the language. For example, associated with the pattern:

<component><sends><data>to<component>

is a conceptual template called *unidirectional value transfer* that denotes a transfer of data from one component to another physical component.

The concepts in PHRAN are expressed in a specification representation language (SRL) loosely based on Conceptual Dependency Diagrams (CD) as developed by Schank [10]. Our SRL is based on concepts of system behavior such as temporal constraints and value transfers and the information required to specify a digital system.

5.3 The Design Data Structure: A Modelling Tool

The Design Data Structure (DDS) [6], is the underlying representation used both to develop the SRL for understanding specifications and also to represent the behavior of digital systems. The DDS is a unified representation of design data. It has been designed to support and facilitate the synthesis of digital hardware systems. It is composed of four subspaces:

1. Data Flow (DFss), which covers data dependencies and functional definitions.
2. Timing and Sequencing (TSss), which covers timing, sequence of events and conditional branching. It is represented by a directed acyclic graph, which consists of nodes corresponding to events, and arcs which represent intervals and connect these nodes.
3. Structural (Sss), which covers the logical decomposition of a circuit. This subspace is similar to a schematic or block diagram.
4. Physical (Pss), which covers the physical hierarchy of components and the physical properties of these components.

TSss arc types There are four types of timing arcs. A sigma arc represents an interval of time (or range [8]) in the TSss. A theta arc represents a temporal constraint. A chi arc represents a casual relationship. A delta arc represents *inertial delay.*

Bindings			
Value	**Interval**	**Operation**	**Interval**
src cntl	σ_4	src	σ_3
snk cntl	σ_4	snk	σ_2
info	σ_5	cntl	σ_1

Table 1: The DDS Bindings for the UVT.

TSss node types There are seven types of nodes in the TSss. For example, a *w)* node is a simple node that may *join* two arcs. A beta node represents an *and fork* point or a *cobegin.* A mu node represents an *and join* point.

The relationships between these various spaces are made explicit by means of bindings. These bindings and the information in the four subspaces are believed to fully characterize the design.

5.4 The Knowledge of the Specification Domain

The desired result of processing the natural language text is to form a representation of the behavior in the DDS. Since PHRAN processes each sentence independently, it is only necessary to consider the fragments of DDS graphs or sub-graphs which may be created as a result of reading each sentence. Then the fragments can be related as a post-processing step.

A small set of *concepts* that characterize system-level behavior, constraints and other ancillary data were developed. These concepts can be grouped into classes as follows: Information Transfer consisting of Unidirectional Value Transfer (UVT), Bidirectional Value Transfer (BVT), and Nondirectional Value Transfer (NVT); Temporal Activities consisting of Asynchronous Temporal Activity (ATA), Causal Temporal Initiation (CTI), Causal Temporal Termination (CTT), and Single Temporal Event (STE); Temporal Constraints consisting of Single Temporal Relation (STR), and Dual Temporal Relation (DTR); Control consisting of If-then-else, While, Repeat, and Looping; Declarations consisting of Assignment or Inheritance Statements, and Structural or Physical Interconnection; and Abstractions of DDS Bindings consisting of Value-Carrier-Net-Range (VCNR), and Operation-Module-Block-Range (OMBR).

Formal semantic definitions of all these concepts have been developed and represented using DDS templates [6].

5.5 The Unidirectional Value Transfer

An example of these models is the DDS template for a UVT shown in Figure 2 and Table 1. This template spans two of the DDS subspaces, the DFss and the TSss. The template for the UVT in the DFss is composed of three values and three operations and their data link arcs. The control operation may be associated with the source operation, where the data flow value, info originates or the sink operation, the destination for the data flow value, info or the control may be associated with a third independent operation.

An example of a sentence which maps into a UVT is

The cpu transfers the block of data bytes from the disk to the control store.

If no timing information or constraints are specified in the same sentence then the TSss template shown in Figure 2 is used as the default. The default TSss template shows the timing for the three operations and the necessary constraints for a valid UVT. For example, the constraints labelled 0_1 and 02 represent the fact that the time interval for the operation control must begin before the end of the intervals associated with the source operation and the sink operation, respectively. If these constraints were not present it would be meaningless to associate the src.cntl value or the snk.ctl value with this particular transfer of the value info.

The fact that these constraint arcs emanate from a node labelled *0_C2* indicates that the two constraints, 0*1* and 0*2* and the interval labelled *o4* all begin concurrently.

5.6 PHRAN-SPAN Operation

PHRAN reads the sentence from left to right one word at a time. As each word is examined, existing patterns and concepts are checked for a match and retained, modified or discarded. The match may be based on lexical criteria, semantic criteria and/or syntactic criteria. PHRAN also provides look-ahead in the sentence to the next word and the ability to look back at previously matched terms with limited ability to modify those previously matched terms.

The patterns and concepts for PHRAN are stored in a knowledge-base of pattern-concept pairs (PCP). An example of the pattern for the UVT concept expressed by the verb transfer is

```
[(or (a_component) (df_opn) (root %transfer)
                                    (df_val)].
```

The subject of the sentence must belong to the semantic category of an a.component (abstract component) *or* a df.opn (data flow operation) for this pattern to match. The next part of the pattern indicates that some verb form with the root of transfer must be present. The verb may be in a different tense, *e.g.,* transferred or it may be combined with a modal verb like shall. The object transferred belongs to the semantic category df.val (data flow value). The abstract component is introduced to handle ambiguity. For example, a cpu may be a logical module or a physical block. An additional declaration or phrase like an appositive is required to resolve this type of ambiguity. If a cpu process had been specified this would be interpreted more precisely as a df.opn and the pattern would also match.

10

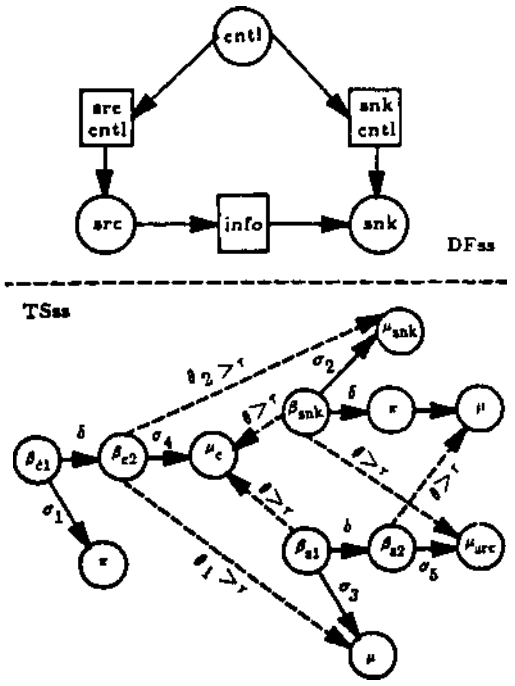

Figure 2: The DDS Template for a UVT.

Associated with each pattern is a concept that describes the meaning of the word phrase or sentence that matches the pattern. The concept is represented as a frame [12] using the specification representation language SRL, based on the set of concepts we introduced.

For example, the concept part of the pattern-concept pair for the UVT in SRL is

```
(uni_dir_vtrans
        (source  (a_component ?source))
        (sink    (a_component ?sink))
        (info    (df_val ?info))
        (control (a_component ?control)))
```

The facet slots in the frame are represented by variable names that are prefixed with a question mark. Fillers for these slots are obtained when the sentence matches the pattern. For example, consider the sentence,

The cpu transfers the command.

The resulting concept for this example is

```
(uni_dir_vtrans
        (source  (a_component *unspecified*))
        (sink    (a_component *unspecified*))
        (info    (df_val command1))
        (control (a_component cpu1)))
```

Note the facet values for the source and sink have defaulted to •unspecified* since they were not included in this sentence.

The source and sink are specified by adding two adverbial phrases to the sentence. These phrases must match the patterns associated with the desired concepts. The pattern for source is

[from (or (a_component) (df_opn))]

When these patterns are matched the concept associated with each of them modifies the UVT pattern by replacing the default value of * unspecified* with the value of source and sink found in the sentence.

The following sentence results in a completely specified UVT:

The cpu transfers the code word from the controller to the peripheral device.

An option in the prototype system is for SPAN to display the resulting concept in English instead of the frame-like data structure. In this case, SPAN'S output for the above sentence is

This sentence resulted in a data flow
subgraph for a unidirectional value transfer.

The source of the information is the
controllerl.

The sink for the information is the
peripheral-device 1.

The Information transferred is the
code-wordl.

The transfer is controlled by the *cpul.*

6 Current Status

The system currently recognizes simple sentences associated with all the primitive concepts of our specification language, *e.g.,* UVT, BVT, CTI and DTR which are required to describe behavior in the domain of digital systems. Actual pattern concept pairs have been built for 25 basic verb patterns common to specifications and 100+ nouns. In addition, we have added the ability to handle multi-noun sequences which occur frequently in specifications. Also the system has been extended to detect ambiguity that can arise from the use of nouns and verbs that have the same lexical stem, *e.g.,* transfer, interrupt, and signal.

The system is coded in Franz Lisp and is running in interpreted mode on a SUN/2 workstation under SUN's operating system, Version 1.4 (UNIX BSD 4.2). Typical sentences take approximately 15 to 35 cpu seconds to process. No attempt has been made to optimize the code.

7 Acknowledgements

This research was supported by the National Science Foundation under Computer Engineering grant #DMC-8310744 and John Granacki was partially funded by the Hughes Aircraft Co.

References

[1] R.J. Abbott. Program description by informal English descriptions. CACM, 26(31):882-894, November 1983.

[2] Arens. *CLUSTER: An Approach to Contextual Language Understanding.* PhD thesis, University of California, Berkeley, 1986.

[3] R. Balzer. A 15 year perspective on automatic programming (invited paper). *IEEE Transactions on Software Engineering,* SE-11(11):1257-1268, November 1985.

[4] J.R. Comer. *An Experimental Natural-Language Processor for Generating Data Type Specifications.* PhD thesis, Texas A&M University, May 1979.

[5] J. Granacki, D. Knapp and A. Parker. The ADAM design automation system: overview, planner and natural language interface. In *Proceedings of the 22nd A CM/IEEE Design Automation Conference,* pages 727-730, ACM/IEEE, June 1985.

[6] J.J. Granacki. *Understanding Digital System Specifications Written in Natural Language.* PhD thesis, University of Southern California, November 1986. Department of Elecrtical Engineering.

[7] M.R. Grinberg. *A Knowledge-Based Design Environment for Digital Electronics.* PhD thesis, University of Maryland, 1980. Dept. of Computer Science.

[8] D.W. Knapp, J.J. Granacki and A.C. Parker. An expert synthesis system. In *Proceedings of the ICC AD Conference,* pages 164-165, IEEE, September 1983.

[9] T. Samad and S.W. Director. Toward a natural language interface for CAD. In *Proceedings of the 22nd ACM/IEEE Design Automation Conference,* pages 2-8, ACM/IEEE, June 1985.

[10] R.C. Schank. *Conceptual Information Processing.* Volume 3 of *Fundamental Studies in Computer Science,* American Elsevier Publishing Co., New York, N.Y., 1975.

[11] Y. Arens Wilensky, R. and D. Chin. Talking to UNIX in English: an overview of UC. *CACM,* 27(6):574-593, June 1984.

[12] P.H. Winston and B.K.P. Horn. *LISP.* Addison-Wesley, 1984.