

Towards Knowledge Acquisition from Natural Language Documents

-- Automatic Model Construction from Hardware Manual --

Toyo-aki NISHIDA, Akira KOSAKA and Shuji DOSHITA

Department of Information Science
Faculty of Engineering, Kyoto University
Sakyo-ku, Kyoto 606, JAPAN

ABSTRACT

In this paper, we explore automatic model construction by analyzing natural language documents. The extracted model will be utilized by a CAD system. A system called *hmU*, in the course of development, is designed to allow knowledge on very complex hardware module like LSI or VLSI to be incorporated into its knowledge base. The acquired knowledge will be utilized for helping human designer understand the component from various levels of abstraction. The focus of this paper is attentioned more to issues on knowledge representation and model inference than that on natural language analysis. Hierarchical model is employed. In particular, cause-effect representation is used to make it clear how actions of each module and events are related to each other. A brief description is given to illustrate our approach.

1. Introduction

One of the difficulties with expert system is knowledge acquisition. The fact that most of human knowledge is integrated as natural language documents has led us to the development of a system which can automatically acquire knowledge from existing documents. In knowledge acquisition from documents, expressive power of natural language, in particular the ability of representing any complex object or situation from various levels of abstraction, should not be sacrificed but be effectively utilized by an expert system for CAD. The essential problem to be solved consists more in model representation and inference problem than in parsing. For example, the resolution of anaphoric expression depends more on domain specific knowledge than on linguistically general knowledge.

A system called *hmU* (hardware manual Understander) is in the course of development, which will analyze given natural language specification of LSI chips (such as microprocessor and the like), and which

will construct a knowledge structure specifying the behavior of the chip. The acquired model will be utilized by intelligent symbolic simulator to give human designer explanations about the chip. The intelligent symbolic simulator is also under development [Nishida 1983].

hmU's main components are natural language analyzer and model builder. Natural language analyzer utilizes domain specific knowledge to resolve ambiguities, anaphoras, etc. Accordingly, the model builder and the natural language analyzer communicates to each other. The details are described in [Nishida 1982]. The model builder receives internal representation obtained from natural language analysis and builds an automaton-based hardware model structure using common sense knowledge about hardware, time, action, events, etc. In what follows, we will describe the model representation and model building procedure to more detail.

2. A Model for Representing Multiple Agent Co-Operating Situation

This section describes the hardware model which we used. The well defined hardware domain can be modeled as a world where multiple agents are cooperating each other to attain a common goal. Actions are done in parallel and are synchronized through events. Use of hierarchical representation gives human designer good understanding of hardware. For example, a phrase like "MREQ is asserted" can be given a good explanation, if a statement like "it indicates that address to the memory becomes valid" is accompanied.

Our hardware specification model is hierarchical. Hierarchies are linked together by *indicates* relations. Each hierarchy consists of an *event* model and a set of *action* models. To each hardware module, an action model is defined to specify the behavior of the module. The notion of hierarchy is important in digital circuit design [McDermott 1978, Mitchell 1981, Stefik 1982, Sakai 1982, Sussman 1980]. In the event model, cause-

effect representation is used to correlate actions of each agent along with the time axis. The notion of cause-effect relationship has been advocated to be useful in understanding cooperating actions and events [Rieger 1978]. The action model of an agent represents the actions taken by the agent when a specified input event takes place. This representation is independent of the internal structure of the agent, and makes it possible to give the abstract level description of any complex hardware.

3. Model Inference Procedure

The model builder is given some information from the natural language analyzer and other information from the diagram analyzer (currently pictures like time chart are manually encoded into symbolic expressions), and it attempts to construct a consistent model from inputs. Since the information from the natural language analyzer may be vague or not specified to full details, the model builder has to make inference in the hardware domain. Sometimes model revision may be needed to correct arbitrary choices that are made due to the lack of information. Accordingly, the task has much in common with truth maintenance system [Doyle 1979]. The model builder mainly makes forward inferences using common sense knowledge. In this paper, we concentrate on the descriptions on hardware behavior and assume other parts of the hardware manual such as pin descriptions, have already been analyzed and converted into the knowledge structure. The below illustrates a part of the knowledge:

Action-Event Definition

Example 1. asserts

a asserts s at t (a: agent, s: signal, t: time)

presupposition

a is an agent for s.
P: s is available for s at t.
a believes P at t-.
s is not active at t-.

definition

<gate level operation is given here>

effect

s becomes active at t+.

Example 2. sends

a sends s to b through x at t
(a, b: agent, x: data transfer device, t: time)

presupposition

a has s at t-.
a believes b at t-.
a believes x is available for a at t.
...

definition

a puts s on x at t.
a requests b to sample s from x at t+.
effect
b has s at t+ +.

Module Definition

Example 3. bus

bus : [isa data-transfer-device
with (consists-of address-bus data-bus
*MREQ ...) ...]

Example 4. signal and line

*MREQ : [isa signal with (asserted-by CPU) ...]
*MREQ-line : [isa line with (type NEG)
(indicates *MREQ) ...]

Example 5. schema of a line

line : [isa : data-transfer-device
has-attribute (type NEG POS)
(state HIGH LOW (HZ))
where
if type=POS then
state=HIGH indicates ACTIVE
state=LOW indicates NEGATIVE
else
state=HIGH indicates NEGATIVE
state=LOW indicates ACTIVE]

The model inference procedure involves the following types of reasoning:

- (a) Seeking an action which causes a given event: this task will be done using action-event definitions.
- (b) Seeking values of case slots which were not filled by the natural language analyzer: this task will be done using definitions for individuals.
- (c) Linking descriptions of each hierarchy using *indicates* links.
- (d) Reasoning about cause-effect relations between events: this reasoning utilizes action-event definitions.
- (e) Making vague expressions more accurate: this task is done using axioms for event and action. Example of such axioms can be found in [McDermott 1982].
- (f) Revising a model if needed: each position where arbitrary advice was made is marked. Those positions are candidates for reconstruction when any contradiction takes place.
- (g) Verifying constraint condition.

Fig.1 illustrates how the description: "MREQ lines goes low at T1" is incorporated into the model. As is seen from the figure, action and event models of each hierarchy are revised so as to be able to give explanation

to the input. Symbols attached to the figure indicates which inference rule is made.

4. Conclusion

Currently, a simplified version of hmU is in the course of development, where the focus is mainly attentioned to rather basic issues, i.e., natural language analysis, discourse analysis, canonical transformation, and reasoning about action, time, and event. However, the initial experiments by hand reveal fundamental validity of our approach. Some of them are illustrated in the appendix.

References

- [Doyle 1979] Doyle, J., A Truth Maintenance System, *AI 12* (1979), 231-272.
- [McDermott 1978] McDermott, D., Circuit Design as Problem Solving, in *Lamtombe(ed.), Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North-Holland, 1978, 227-252.
- [McDermott 1982] McDermott, D., A Temporal Logic for Reasoning About Processes and Plans, *Cognitive Science* 6, 1982, 101-155.
- [Mitchell 1981] Mitchell, T.M. et al., Representations for Reasoning About Digital Circuits, in *Proc. IJCAI-81*, 1981, 343-344.
- [Nishida 1982] Nishida, T., Kosaka, A., and Doshita, S., On Automatic Extraction of Information from Hardware Manuals, Technical Report AL-82-68, *IECE of Japan*, 1982, (in Japanese).
- [Nishida 1983] Nishida, T., Kosaka, A., and Doshita, S., On Action Description Model and its Inference for Hardware Manual Understanding, in *Annual Convention Records of IPS Japan*, 7C-1, 1983, (in Japanese).
- [Rieger 1978] Rieger, C. and Grinsberg, M., A System of Cause-Effect Representation and Simulation for Computer-Aided Design, in *Lamtombe (ed.), Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North-Holland, 1978, 299-333.
- [Sakai 1982] Sakai, T., et al., An Interactive Simulation System for Structured Logic Design, in *Proc. ACM IEEE 19th Design Automation Conference*, 1982, 747-754.
- [Stefik 1981] Stefik, M. and Bobrow, D.G., Linked Module Abstraction: A Methodology for Designing the Architectures of Digital Systems, KB-VLSI-81-9,

XEROX PARC, 1981.

[Sussman 1978] Sussman, G.J., SLICES: At the Boundary between Analysis and Synthesis, in *Lamtombe (ed.), Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North-Holland, 1978, 261-298.

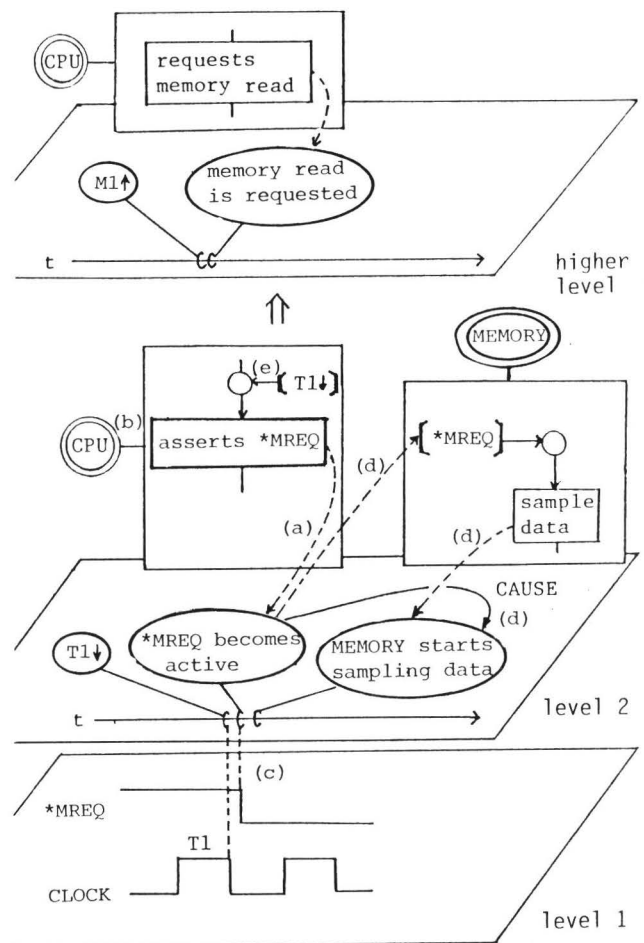


Figure 1. An Example of Model Inference.

Appendix: Overview of the Simplified Version of hmU

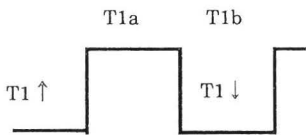
This appendix illustrates an overview of the simplified version of hmU. Usually, input text can be divided into diagrams and natural language text portion. The acquisition task consists of diagram analysis and natural language analysis. After each step is completed, the results are matched together and consistent model will be instantiated.

Diagram Encoding

Diagrams are assumed to be somehow encoded into symbolic expressions. The encoding rule is as follows:

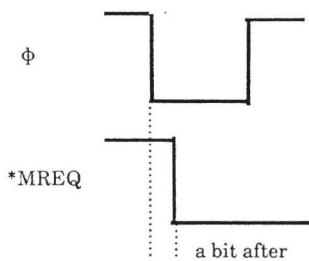
(step 1) labeling clock: generating new symbols to name each clock pulse.

Example 1.



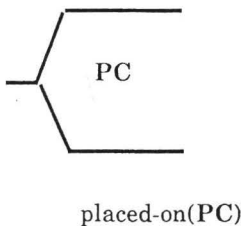
(step 2) assigning symbolic names to events other than clock: This labeling is carried out in terms of Active/Negative instead of High/Low.

Example 2.



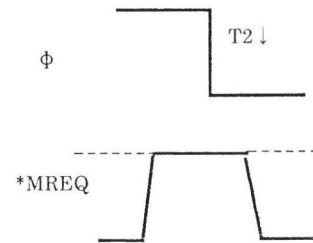
$\Delta\text{afterT1}\downarrow$: asserted(*MREQ)

Example 3. If a name is given to the event, it is copied to the symbolic expression.



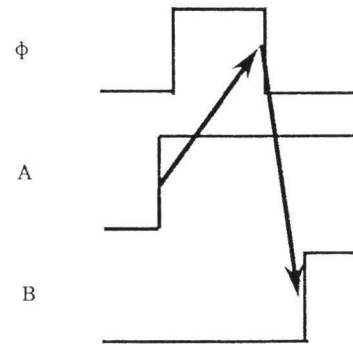
(step 3) special cases:

Example 4. indication of sampling a signal.



at $T2\downarrow$: sampled(*MREQ)

In time chart arrows are often used to indicate causal-effect relationships. Such information should be effectively utilized during the process of diagram encoding. For example,



state-1: $A\uparrow$: $\rightarrow$ state-2;
state-2: $\phi\downarrow$: asserts(B);

Analysis of Diagram

From encoded diagrams, simple model inference rules can be used to extract model structure. Here, automaton description is based on DDL [Duley 1968].

(rule 1) supplementing agents: signal specification is used. For example,

from: $\Delta\text{afterT1}\downarrow$: asserted(*MREQ),
infer: $\Delta\text{afterT1}\downarrow$: asserted(*MREQ) by CPU.

(rule 2) inference of automaton transition arc:

for example,

from: $\Delta\text{afterT1}\downarrow$: asserted(*MREQ),
infer: $T1a$: $\phi\downarrow$: asserts(*MREQ).

Natural Language Analysis

Natural language portion of input text is analyzed sentence by sentence. The result of the phrase structure analysis is translated into intermediate representation. The intermediate representation is designed using the formalism of lexical functional [Kaplan 1982]. Then it is further transformed into canonical representation. During the process, discourse analysis is carried out to solve simple cases of definite noun phrase reference and ellipsis. Intermediate structure is used as a discourse structure.

The below illustrates an intermediate representation and canonical form for a simple sentence:

The CPU turns off the *RD signal. ... (1)

The Intermediate Representation

```

Category = Sentence
Subject = Category = NounPhrase
    Determiner = [Lexicon = The, ... ]
    MainNoun = [Lexicon = CPU, ... ]
    SemanticFunction
        = Evaluate[↑ Determiner]
Predicate = Category = VerbPhrase
    MainVerb = [Lexicon = Turn, ... ]
    SemanticFunction = ↑ MainVerb
AdverbialParticle = Off
Object = Category = NounPhrase
    Determiner = The
    MainNoun = [Lexicon = *RD-Signal, ... ]
    SemanticFunction
        = Evaluate[↑ Determiner]
SemanticFunction = Evaluate[↑ Predicate]
  
```

Semantic Mapping Function for "turn"

```

AdverbialPhrase = 'On'
→ Create Unit: {Type ← 'Event',
    Self ← 'Asserts',
    Level ← 'DDL',
    Agent ← ↑ Subject,
    Object ← ↑ Object}
AdverbialPhrase = 'Off'
→ Create Unit: {Type ← 'Event',
    Self ← 'Negates',
    Level ← 'DDL',
    Agent ← ↑ Subject,
    Object ← ↑ Object}.
  
```

Canonical Form

```

Unit:a: Type = Event
    Modality = Description
    Self = Negates
    Level = DDL
    Agent = Unit:b
    Object = Unit:c
  
```

```

Unit:b: Type = Individual
    Isa = CPU
  
```

```

Unit:c: Type = Individual
    Isa = *RD
  
```

Comparing Outputs from Natural Language Analyzer and Diagram Analyzer

Roughly speaking, information from diagrams like time charts, tells a lot about the described hardware. So our strategy is first to construct a model based on the information from diagram and then to check it against natural language information. Figure A-1 illustrate this process for a version of sentence (1):

This same edge is used by the CPU to turn off the *RD and *MREQ signal. ... (1')

References of Appendix

[Kaplan 1982] Kaplan, R.M. and Bresnan, J., Lexical-Functional Grammar: A Formal System for Grammatical Representation, in Bresnan (ed.), *The Mental Representation of Grammatical Relations*, The MIT Press, 1982, 173-281.

[Duley 1968] Duley, J.R. and Dietmeyer, D.L., A Digital System Design Language (DDL), *IEEE Trans. Computers*, Vol. C-17, No.9, 1968.

*Examples are cited from: Z80-CPU/Z80A CPU Technical Manual.

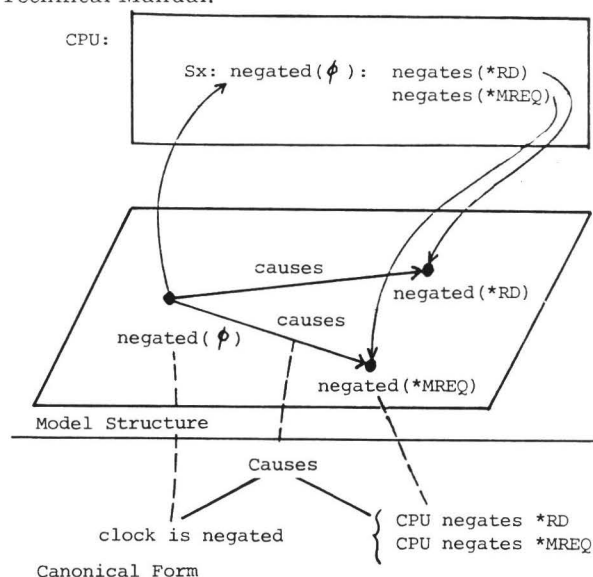


Fig.A-1. Matching input against the Model.