Management System of Knowledge Base in Intelligent Machining*

Yasumi NAGASAKA**, Happy WIBISONO**,
Hideyuki OHTAKI** and Yoshio ISHIKAWA**

This paper presents an approach to the management of knowledge bases in machining. This approach is aimed at developing a system functioning as a knowledge base management mechanism like TMS, and which can also manage different representations of knowledge bases collectively. First, we integrate the description of the knowledge base by using the symbolic-expression form. Secondly, in connection with the TMS knowledge management mechanism, we developed a new predicate-logic representation language based on PROLOG to be applied to TMS. As a result, a strategy for resolving the contradiction in TMS can be defined easily by using the meta level representation.

Key Words: Expert System, Artificial Intelligence, CAM, Knowledge Base, Milling, Cutting, Frame Representation, TMS

1. Introduction

The knowledge base in an expert system is built based on the knowledge and experience of experts. However, as we know, the knowledge of an expert is always being updated and changed according to the operating conditions of the equipment and financial circumstances. Consequently, a knowledge base also needs to be renewed from time to time. In connection with the above matter, it is important to create a management system to support coordination of knowledge and to rebuild the knowledge base without any contradictions. TMS (truth maintenance system)[1],[2], and ATMS (assumption-based on TMS)[3] are examples of management systems which have been developed and are being used in practical management (e. g., scheduling systems[4],[5]).

However, as far as the author knows, there has been no application of these systems to the management of knowledge bases in the mechanical engineering field. Experts in the mechanical engineering field must select the most suitable knowledge from a range of available information which consists of static knowledge that does not change once determined and knowledge which changes from time to time. For that reason, to build an expert system in the mechanical engineering field, many knowledge representations must be mixed. Consequently, current systems such as TMS and ATMS which use only limited knowledge representations are not sufficient.

Our research is aimed at developing a system which manages different knowledge representations of a knowledge base collectively. For example, a knowledge base which is built by frame representation[6]-[8] or rule representation[9],[10] and a knowledge base of numerical equations, which are written in C language, can be combined systematically. Thus, there is a possibility of the realization of a reasoning knowledge base among different knowledge representations. However, such a system does not have a mechanism to manage the knowledge base itself, which a system like TMS has.

In order to manage different knowledge representations of a knowledge base collectively, we first integrate the description of the knowledge base by using a symbolic-expression form which is conducted using LISP language. As a result, it effectively serves to combine knowledge bases based on frame representation; connect managing mechanism; and develop

* Received 20th May, 1993.
** Department of Mechanical Engineering, Saitama University, 255 Shimo Okubo, Urawa-shi, Saitama 338, Japan

algorithms. Secondly, in connection with the TMS knowledge management mechanism, we developed a new predicate-logic representation[11),(12)] language based on PROLOG to be applied to TMS.

This paper describes how to manage knowledge bases in a machining management system and how to treat the functions in a TMS using the above process mechanism. It also gives an example of the application of TMS in machining.

2. Predicate-Logic Representation (PROLOG) of the Developed Managing System

Prolog is predicate-logic representation language which has unification and backtracking as special functions. Figure 1 shows a comparison between the description by Prolog that we developed and that by common Prolog (DEC-10 Prolog).

Basic description in common Prolog is written in the form of

predicate-name (argument 1, argument 2, ...).

In comparison, in our managing system, description is written in the form of

(predicate-name argument 1 argument 2...).

In this description, since the predicate-name and arguments are expressed by a symbolic expression, predicate-name can be treated as a variable. Consequently, it is possible to realize a reflection function on meta level representation[13)-(15)]. Then, "node generation" and "contradiction resolution" of TMS can be realized easily (as will be described in section 3.4). Furthermore, an algorithm of contradiction resolution by inference engine (attached procedure) in frame representation can also be adapted easily in knowledge representation such as rule representation.

3. Machining Management System with Knowledge Base Management Mechanism

3.1 Composition of machining management system

Figure 2 shows the composition of the machining management system which provides a knowledge base management mechanism.

The machining management system consists of an inference engine of conditions, knowledge base (to store knowledge about machining conditions), and data base (to store inference results). The knowledge base is built based on the knowledge of experts in the description, as mentioned in section 2. However, the data base stores inference results of the knowledge base by frame representation. On the other hand, TMS consists of a node generation system and contradiction resolution system and operates when inconsistent machining conditions are detected.

3.2 Inference process of machining management system

Figure 3 shows an example of knowledge representation which is exchanged between inference processes of the machining management system.

Now, let us perform the inference process according to the numbers enclosed in parentheses in Fig. 3.

(1) Input the name of milling machine.

(2) Input machining conditions (material, hardness, etc.).

(3) Based on the input conditions, machining conditions (e.g., diameter of cutter, number of revolutions, etc.) which are suitable in the data base are offered to the user. If suitable machining conditions do not exist in the data base, step (4) is executed. Moreover, if machining conditions offered are still inconsistent, step (5) will be executed.

(4) Execute attached procedure (Attached-Procedure-1) by ISA-relation, and perform inference engine of the machining conditions. Then, load knowledge base which is suitable for machining conditions and create the nodes of TMS. Subsequently, initial conditions of node generation are expressed as a result (in this step, the reasoning of contradiction resolution is not performed). This result is kept in the new data base and simultaneously offered to the user. If the conditions expressed are inconsistent, step (5)

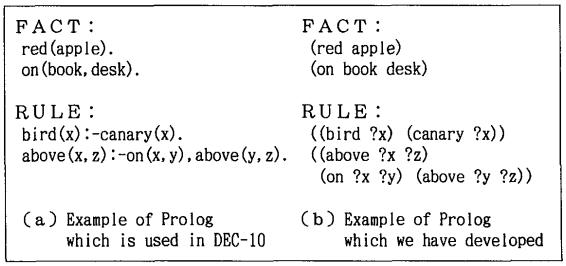

Fig. 1 A comparison between description by Prolog that we developed and by common Prolog (DEC-10 Prolog)

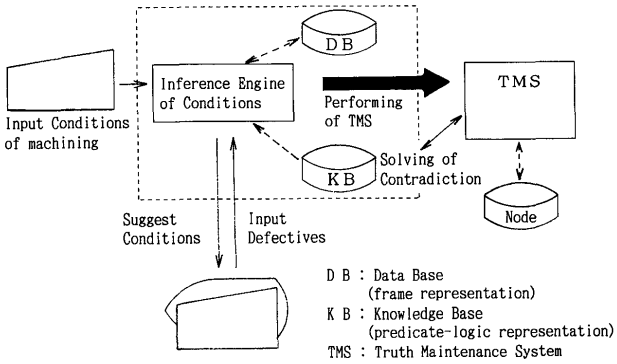

Fig. 2 The composition of machining management system which provides knowledge base management mechanism

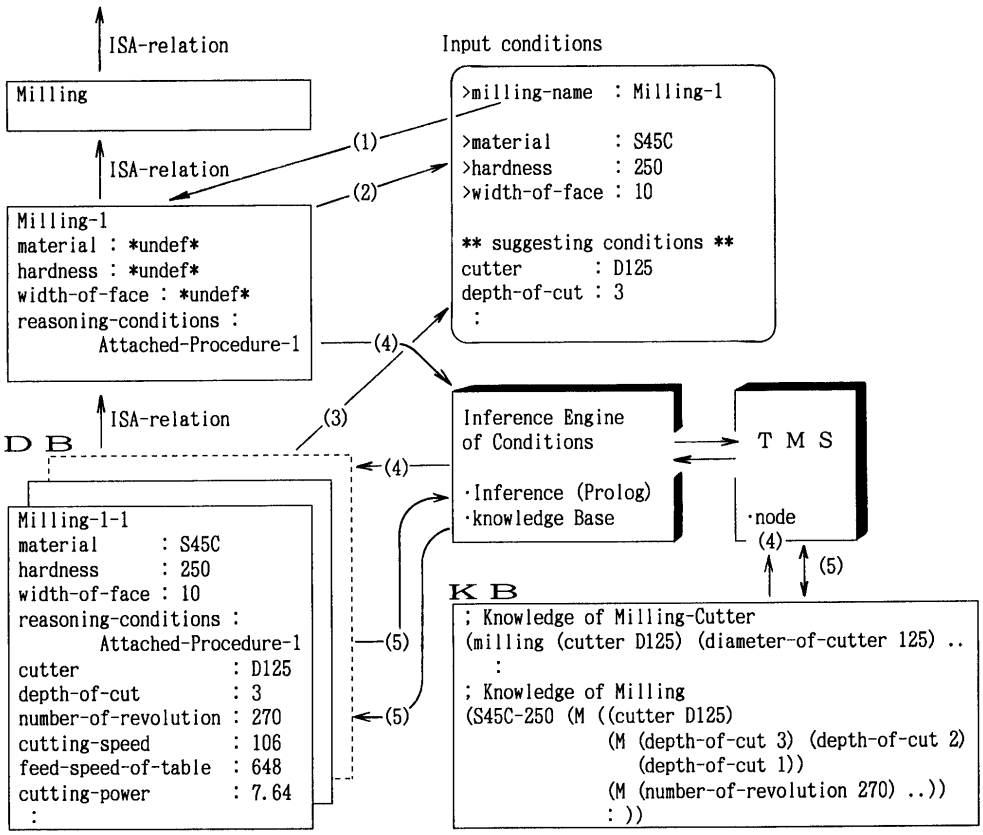

Fig. 3 Example of knowledge representation which is exchanged between inference processes of machining management system

is executed.

(5) Perform TMS (as will be described in section 3.4) and execute inference to resolve contradictions. As a result, data base is renewed and presented to the user simultaneously. If its conditions are not satisfied, step (5) will be repeated until a satisfactory result is obtained. In this case, data base and knowledge base are renewed.

Of these, knowledge base is managed by TMS of predicate-logic representation, and inference is performed by using attached procedures of frame representation. Since the knowledge of experts is expressed by frame representation, it is easily understood even by inexperienced operators. Moreover, since the inference engine of machining conditions are built by predicate-logic representation, it will produce an effective system by virtue of realizing an inference function (for resolving contradictions by using the backtracking function of the original representation language).

3.3 Data structure of knowledge base

Figure 4 shows the contents of the knowledge base for milling. From this, we know that expert knowledge on machining is presented as a "table" of predicate-logic representation, which is a kind of relational data base.

```
; Knowledge of Milling-Machine
(max-milling-machine (max-cutting-power 10) (max-feed-speed-of-table 1000)
                     ...)
  :
; Knowledge of Milling-Cutter
(milling  (cutter D80)  (diameter-of-cutter 80)  (number-of-cutter 5))
(milling  (cutter D100) (diameter-of-cutter 100) (number-of-cutter 6))
(milling  (cutter D125) (diameter-of-cutter 125) (number-of-cutter 8))
  :
; Knowledge of Milling
(S45C-250  (M  ((cutter D125)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
                (M  (number-of-revolution 270) (number-of-revolution 200)
                    (number-of-revolution 180) (number-of-revolution 140)
                    (number-of-revolution 100) (number-of-revolution 70))
                (L  (depth-of-cut 2) (number-of-revolution 270)))
               ((cutter D160)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
                (M  (number-of-revolution 200) (number-of-revolution 180)
                    (number-of-revolution 140) (number-of-revolution 100)
                    (number-of-revolution 70)))
               ((cutter D200)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
                    (number-of-revolution 180) (number-of-revolution 140)
                    (number-of-revolution 100) (number-of-revolution 70)))
               ((cutter D250)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
                (M  (number-of-revolution 140) (number-of-revolution 100)
                    (number-of-revolution 70)))
               ((cutter D315)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
                    (number-of-revolution 100) (number-of-revolution 70))))
(S45C-350  (M  ((cutter D125)
                (M  (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))
    :
```

Fig. 4 The contents of knowledge base for milling

For example, the knowledge of a milling cutter shown in Fig. 4 has the following meaning:

"(cutter D80), (diameter-of-cutter 80), (number-of-cutter 5)." The above three datums exist together in relation to "milling". In other words, corresponding

to the description mentioned in section 2, predicate-name is "milling" and the arguments are (cutter D 80), (diameter-of-cutter 80), and (number-of-cutter 5).

Here, in order to be able to express the representation of the experts, we have analyzed the knowledge representation of some experts. As a result we obtain two kinds of representations,

(1) Hypothetical knowledge in the form of "if...." ("M" in Fig. 4).

(2) Restrictive knowledge in the form of "however..." ("L" in Fig. 4).

This "M" and "L" representation is described as follows:

(M (depth-of-cut 3) (depth-of-cut 2) (depth-of-cut 1))

(L (depth-of-cut 2) (number-of-revolution 200))

The meaning of "M" is "if there is a fact which denies the assumption of (depth-of-cut 3), then (depth-of-cut 2) will be assumed". In comparison, the meaning of "L" is "however, it is no good in the case of (depth-of-cut 2) and (number-of-revolution 200) because an unsuitable condition occurs".

If this expression is accepted, it will be possible to realize, in the knowledge base, the expression form of the knowledge of machining experts such as "adjust number-of-revolution to 270 or 200, and depth-of-cut either to the value of 3, 2, or 1; however, when it is performed at the number-of-revolution 200 and depth-of-cut 2, an unsuitable condition will occur". With this description, it is easy to understand the contents of the knowledge base.

3.4 Knowledge base management system-TMS

TMS is a system used to maintain consistency in the knowledge base. When an inconsistency is detected, it evokes its own reasoning mechanism, dependency-directed backtracking, to resolve the inconsistency by altering a minimal set of beliefs.

Beliefs have two states: IN and OUT.

IN states (believed to be true) are those that have at least one justification that is currently valid. In comparison, OUT states do not have any currently valid justification (not believed to be true, either because there is no reason for believing it to be true or because none of the possible reasons is currently valid).

TMS uses these IN and OUT states to manage the coordination of justifications.

3.4.1 Justifications of TMS In TMS, each statement or rule is called a node. There are two kinds of justifications of node in the system.

— Support list (SL) justification

— Conditional proof (CP) justification

Based on these justifications the relations between nodes are expressed and contradiction resolution in the knowledge base is conducted.

The SL justification node is presented as

(node-number SL (IN-nodes) (OUT-nodes)).

SL justifications are valid if all of the nodes mentioned in the list of IN-nodes are currently IN and if all the nodes mentioned in the list of OUT-nodes are currently OUT.

The CP justification node is presented as

(node-number CP <consequent> (IN-hypotheses)).

CP justifications represent hypothetical arguments. They are valid if the consequent node is always IN whenever the nodes in IN-hypotheses are IN. For example, node 1001 in Fig. 7 indicates the argument that "if consequent nodes (1 2 5) are all IN, node 1000 is IN". This assumption is inconsistent with the fact that "node 1000 is OUT". This is because, in fact, consequent nodes (1 2 5) are not simultaneously all IN. In order to resolve the contradiction, TMS searches for one of the inconsistent assumptions and makes it OUT.

3.4.2 Strategy for resolving contradiction and node generation Figure 5 shows the nodes which are created based on knowledge of milling presented in Fig. 4. Here, hypothetical knowledge "M" and restrictive knowledge "L" (see Fig. 4) are expressed by meta level representation, and the nodes shown in Fig. 5 are created by its reflection function. By combining these two kinds of knowledge (M and L), a strategy to resolve contradiction similar to the strategy which is intended by the experts can be realized.

This resolution strategy is based on the three following principles.

(1) Renewing the nodes of knowledge elements such as depth-of-cut, number-of-revolution, etc. (Fig. 4) in hypothetical knowledge "M".

(2) Renewing the nodes with priority from the left-hand side of knowledge elements in hypothetical knowledge "M".

(3) Re-executing contradiction resolution when the nodes of knowledge elements in restrictive knowledge "L" are all IN.

```
(1   (cutter D125)              IN)   (1  SL () (11 12 ..))
(2   (depth-of-cut 3)           IN)   (2  SL (1) (3 4 11))
(3   (depth-of-cut 2)           OUT)  (3  nil)
(4   (depth-of-cut 1)           OUT)  (4  nil)
(5   (number-of-revolution 270) IN)   (5  SL (1) (6 7 8 9 10 11))
(6   (number-of-revolution 200) OUT)  (6  nil)
      :
(11 (not_cutter)                OUT)  (11 nil)
(12 (cutter D160)               OUT)  (12 SL () (1 ..))
      :
 (900  (not_pairs 1 3 5)        OUT)  (900  SL (1 3 5) ())
      :
(1000 (suggest-conditions)      IN)   (1000 SL (1 2 5) ())
```

Fig. 5 The nodes which are created based on knowledge of milling presented in Fig. 4

1. Node generation

Now, we describe the theory of the creation of nodes shown in Fig. 5. Hypothetical knowledge "M" is expressed as

$(\mathrm{M}\ A_1\ A_2\ A_3\ \cdots\ A_n)$,

where A_1, A_2, A_3, $\cdots$, A_n are knowledge elements. The relation between knowledge elements can be expressed as a logical expression of

$A_1 \cup A_2 \cup \cdots A_n \cup A_{\text{not\_all}} = 1$

$A_1 \cup (A_2 \cup \cdots (A_n \cup A_{\text{not\_all}})) = 1$,

where A_1, A_2, $\cdots$ A_n, $A_{\text{not\_all}}$ can be treated as independent knowledge elements. Now, $A_{\text{not\_all}}$ indicates all knowledge except hypothetical knowledge "M". That is, when the node $A_{\text{not\_all}}$ is IN, it means that it is impossible to create an IN node in hypothetical knowledge "M". As a result, it can be expressed as the following relation:

$\bar{A}_1 = (A_2 \cup \cdots (A_n \cup A_{\text{not\_all}}))$

$A_1 \cup \bar{A}_1 = 1$

As expressed in the above logical expressions, the generation of node "M" always chooses one knowledge element. Now a reasonable SL justification which chooses the knowledge element is performed. This justification comes from the above logical expression that "because $\bar{A}_1$ is OUT, A_1 is IN". Actually, it creates nodes by the logic that "because A_2, A_3, and so on are OUT, A_1 becomes IN" (nodes 2, 3, 4 and nodes 5, 6 in Fig. 5).

Here, if for some reason A_1 becomes OUT (contradiction occurs), similar to the above relation, it will create a node justification of "because A_3, A_4 and so on are OUT, A_2 is IN".

In comparison, restrictive knowledge "L" in Fig. 4 creates node 900 (not_pairs) (see Fig. 5). This node serves a function similar to that of node (contradiction) (as will described in section 4); that is, to resolve the contradiction. In other words, (contradiction) is a node to resolve the contradiction from the outside, while (not_pairs) is a node from the knowledge base itself.

2. Method for realizing contradiction resolution strategy

Knowledge "M" and "L" are strategies to resolve contradictions. They are realized by the reflection function on meta level representation which is described in section 2.

Figure 6 shows an example of the realization of resolution strategy by using the reflection function. In this example, it is shown that when we use the reflection function of knowledge "C" in the knowledge base, the results of resolution strategy are different. Here, knowledge "C" is described by meta level representation. Furthermore, by using this "C" knowledge (C 1 2), the knowledge base can be expressed easily to define which is selected, "2" or "3".

For this reason, this system can be used easily to define the contradiction resolution strategy corresponding to knowledge representation of the experts.

4. Examples of Applying TMS in Machining

TMS is provided by functions to resolve contradictions which occur due to either clear causes or unclear causes. Figures 7 and 8 show the change of nodes in TMS in the case of an unclear cause. In comparison, Figure 9 shows the change of nodes in the

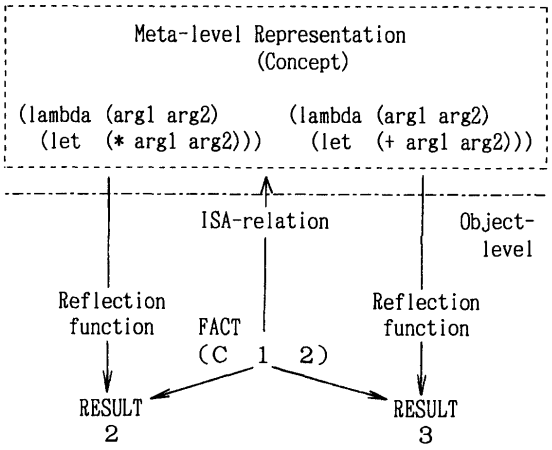

Fig. 6 Example of realization of resolution strategy by using reflection function

```
(1  (cutter D125)                IN)   (1  SL () (11 12 ..))
(2  (depth-of-cut 3)             OUT)  (2  SL (1) (3 4 11))
(3  (depth-of-cut 2)             IN)   (3  SL (1001) (4 11))
(4  (depth-of-cut 1)             OUT)  (4  nil)
(5  (number-of-revolution 270)   IN)   (5  SL (1) (6 7 8 9 10 11))
(6  (number-of-revolution 200)   OUT)  (6  nil)
     :
(11 (not_cutter)                 OUT)  (11 nil)
(12 (cutter D160)                OUT)  (12 SL () (1 ..))
     :
(1000 (contradiction 1000)       OUT)  (1000 SL (1 2 5) ())
(1001 (nogood 1000)              IN)   (1001 CP 1000 (1 2 5))
(1002 (suggest-conditions)       IN)   (1002 SL (1 3 5) ())
```

Fig. 7 The contradiction resolution of machining under the condition of node 1000

```
(1  (cutter D125)                IN)   (1  SL () (11 12 ..))
(2  (depth-of-cut 3)             OUT)  (2  SL (1) (3 4 11))
(3  (depth-of-cut 2)             OUT)  (3  SL (1001) (4 11))
(4  (depth-of-cut 1)             IN)   (4  SL (901) (11))
(5  (number-of-revolution 270)   IN)   (5  SL (1) (6 7 8 9 10 11))
(6  (number-of-revolution 200)   OUT)  (6  nil)
     :
(11 (not_cutter)                 OUT)  (11 nil)
(12 (cutter D160)                OUT)  (12 SL () (1 ..))
     :
(900  (not_pairs 1 3 5)          OUT)  (900  SL (1 3 5) ())
(901  (nogood 900)               IN)   (901  CP 900 (1 3 5))
     :
(1000 (contradiction 1000)       OUT)  (1000 SL (1 2 5) ())
(1001 (nogood 1000)              IN)   (1001 CP 1000 (1 2 5))
(1002 (suggest-conditions)       IN)   (1002 SL (1 4 5) ())
```

Fig. 8 The contradiction resolution of machining under the condition of node 1000 (restrictive knowledge exists)

case of a clear cause.

The following explains how to resolve the contradiction by the change of nodes.

Figures 7 and 8 show the contradiction resolution of machining under the condition of node 1000. Based on hypothetical knowledge that "an inconsistency is detected, but the cause is unclear", the most suitable machining conditions are sought in order to resolve the contradictions, which is conducted as follows.

(1) Because inconsistency has occurred, it is interpreted that present machining conditions are not suitable. Then, TMS creates node 1000.

(contradiction 1000)

It expresses that node 1000 causes a contradiction.

(2) TMS creates node 1001(nogood 1000) by CP justification to make node 1000 "OUT". Then, backtracking is executed. Thus, a searching route is constructed based on resolution strategy to choose one of the nodes in (1 2 5) as OUT.

(3) By using the resolution strategy mentioned above, reasoning mechanism of TMS searches IN-nodes and OUT-nodes, and tries to find a set of assumptions such that node 2 is OUT and node 3 is IN.

(4) Now, node 3 is renewed, as shown in the expression below, and the nodes are renewed in the order of 3, 2, 1000. Moreover, a new node, 1002, is created.

```
(3     SL (1001) (4 11))    to be IN
(2     SL (1)    (3 4 11))  to be OUT
(1000  SL (1 2 5) ( ))       to be OUT
(1002  SL (1 3 5) ( ))       to be IN
```

(5) Then, TMS tries to find the existence of an IN node which is adapted to restrictive knowledge. In the cases where the node is nonexistent (see Fig. 7), TMS is completed. However, if a node which is adapted to restrictive knowledge exists (node 900 in Fig. 8), it will return to step (2) to make this node "OUT". Figure 8 shows the conditions of TMS after the above process is completed.

If under these conditions, machining can be performed favorably, these conditions are recorded in the data base. However, if they are still unsatisfactory, TMS will be executed again and repeated until contradictions disappear.

```
(1   (cutter D125)              IN)   (1  SL () (11 12 ..))
(2   (depth-of-cut 3)           IN)   (2  SL (1) (3 4 11))
(3   (depth-of-cut 2)           OUT)  (3  nil)
(4   (depth-of-cut 1)           OUT)  (4  nil)
(5   (number-of-revolution 270) OUT)  (5  SL (1) (6 7 8 9 10 11))
(6   (number-of-revolution 200) IN)   (6  SL (1001) (7 8 9 10 11))
      :
(11  (not_cutter)               OUT)  (11 nil)
(12  (cutter D160)              OUT)  (12 SL () (1 ..))
      :
(1000 (contradiction 1000 5)    OUT)  (1000 SL (1 2 5) ())
(1001 (nogood 1000 5)           IN)   (1001 CP 1000 (5))
(1002 (suggest-conditions)      IN)   (1002 SL (1 2 6) ())
```

Fig. 9 The change of nodes when the cause of the contradiction is clear

In comparison, Fig. 9 shows the change of nodes when the cause of the contradictions is clear. For example, node

(contradiction 1000 5)

is created to represent a statement from the user's input that "it is clear that there is an inconsistency in the number-of-resolution". Thus, node 5 becomes OUT and the contradiction will disappear. Then, node 1002 is created and TMS is completed.

Figure 10 shows the last condition of TMS when the knowledge base is renewed to express the statement of "select cutter D 160, since milling cannot be done by using cutter D 125".

Figures 11, 12, and 13 are examples of the execution of TMS which we have developed, in the case of the condition mentioned in section 3.2. Figure 11 shows the condition of TMS just before execution; this condition refers to the contents of the data base to be expressed by frame representation.

Figures 12 and 13 show examples of executing the two conditions repeatedly. The first one is "inconsistency is detected, but the cause is unclear", and the second one is "clearly, there is an inconsistency in number-of-resolution". By execution of TMS, the last conditions of machining are recorded in the data

```
(1   (cutter D125)              OUT)  (1  SL () (11 12 ..))
(2   (depth-of-cut 3)           OUT)  (2  SL (1) (3 4 11))
(3   (depth-of-cut 2)           OUT)  (3  SL (1001) (4 11))
(4   (depth-of-cut 1)           OUT)  (4  SL (1003) (11))
(5   (number-of-revolution 270) OUT)  (5  SL (1) (6 7 8 9 10 11))
(6   (number-of-revolution 200) OUT)  (6  SL (1005) (7 8 9 10 11))
      :
(10  (number-of-revolution 70)  OUT)  (10 SL (1013) (11))
(11  (not_cutter)               IN)   (11 SL (1015) ())
(12  (cutter D160)              IN)   (12 SL () (1 21 ..))
(13  (depth-of-cut 3)           IN)   (13 SL (12) (14 15 21))
(14  (depth-of-cut 2)           OUT)  (14 nil)
(15  (depth-of-cut 1)           OUT)  (15 nil)
(16  (number-of-revolution 200) IN)   (16 SL (12) (17 18 19 20 21))
      :
(1000 (contradiction 1000)      OUT)  (1000 SL (1 2 5) ())
(1001 (nogood 1000)             IN)   (1001 CP 1000 (1 2 5))
(1002 (contradiction 1002)      OUT)  (1002 SL (1 3 5) ())
(1003 (nogood 1002)             IN)   (1003 CP 1002 (1 3 5))
(1004 (contradiction 1004)      OUT)  (1004 SL (1 4 5) ())
(1005 (nogood 1004)             IN)   (1005 CP 1004 (1 4 5))
(1006 (contradiction 1006)      OUT)  (1006 SL (1 4 6) ())
(1007 (nogood 1006)             IN)   (1007 CP 1006 (1 4 6))
(1008 (contradiction 1008)      OUT)  (1008 SL (1 4 7) ())
(1009 (nogood 1008)             IN)   (1009 CP 1008 (1 4 7))
(1010 (contradiction 1010)      OUT)  (1010 SL (1 4 8) ())
(1011 (nogood 1010)             IN)   (1011 CP 1010 (1 4 8))
(1012 (contradiction 1012)      OUT)  (1012 SL (1 4 9) ())
(1013 (nogood 1012)             IN)   (1013 CP 1012 (1 4 9))
(1014 (contradiction 1014)      OUT)  (1014 SL (1 4 10) ())
(1015 (nogood 1014)             IN)   (1015 CP 1014 (1 4 10))
(1016 (suggest-conditions)      IN)   (1016 SL (12 13 16) ())
```

Fig. 10 The last condition of TMS when knowledge base is renewed to express the statement of "select cutter D 160, since milling cannot be done by using cutter D 125"

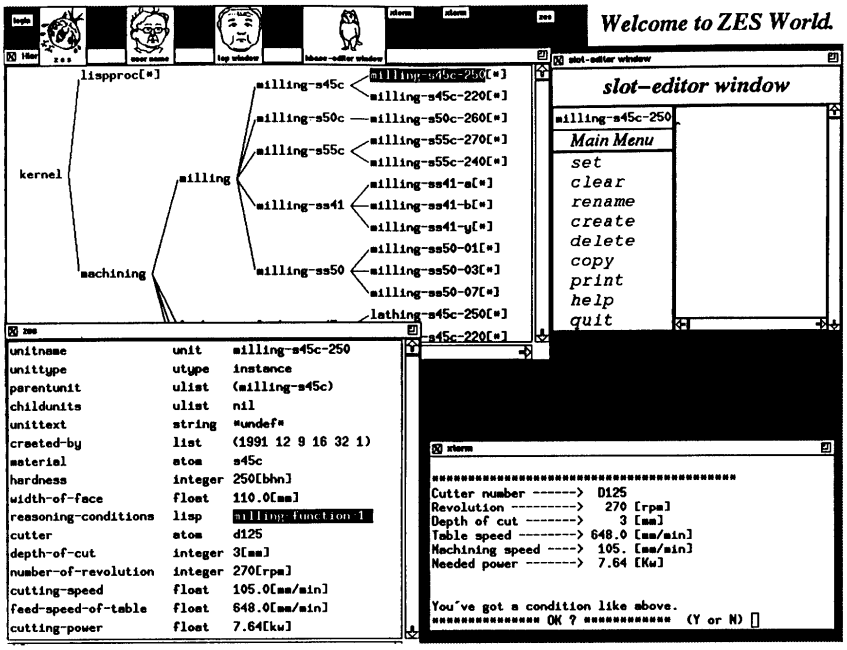

Fig. 11 Example of the condition of TMS just before execution

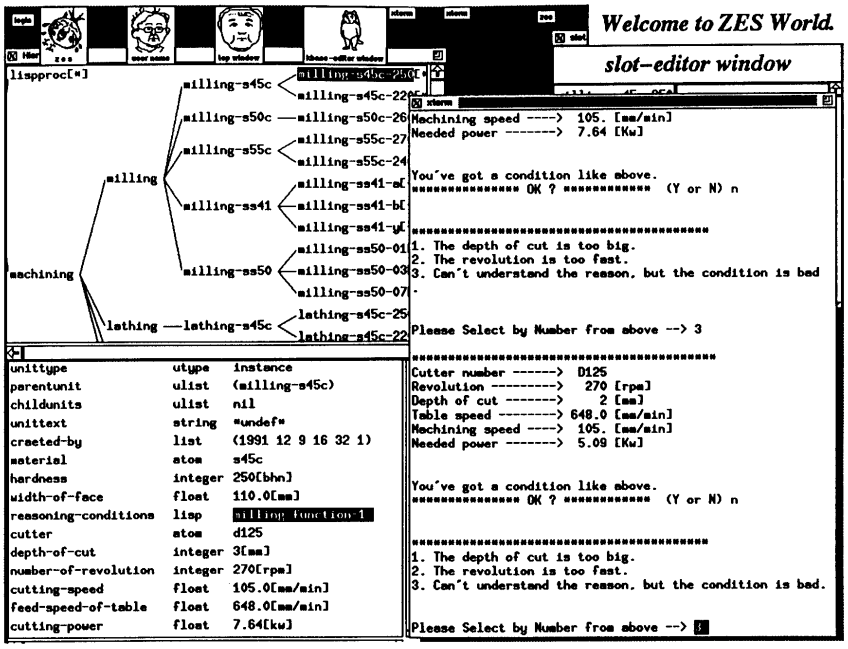

Fig. 12 Example of the execution of TMS when the cause of the contradiction is unclear

base as the newest conditions.

Here, when TMS is executed, cutter-number, depth-of-cut, number-of-revolution, cutting-speed, and other parameters are offered to the users.

Therefore, the condition of the knowledge base in this system becomes suitable for the machining process, because it can change the conditions of nodes by using resolution strategy.

5. Conclusions

In this paper, an approach to the management of the knowledge base to be used in machining has been outlined.

A system functioning as a knowledge base management mechanism has been generated and verified. In order to build this system, description of the knowledge base was unified. It was found that this system is useful for supporting coordination among knowledge sources and for rebuilding knowledge bases from time to time.

It is also possible to express expert strategies for resolving contradictions in machining in the

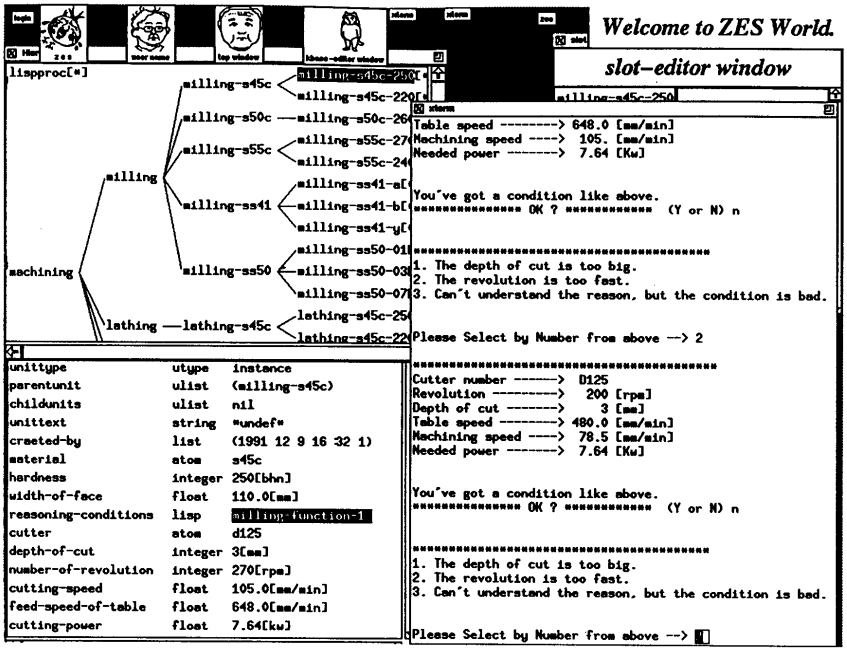

Fig. 13 Example of the execution of TMS when the cause of the contradiction is clear

knowledge base. Hence, this system should be highly appreciated among experts.

The aim of this study was to confirm that the management mechanism system which we developed to manage knowledge bases in intelligent machining can function without any contradictions. Therefore, in order to understand the principle and methods of the mechanism easily, here we imposed limitations on many of the details of the operation of contradiction resolution. However, the usefulness of this management system should be clear even from this simple example, and we believe that the same control mechanism can manage the knowledge base without contradiction in practical case.

References

(1) Doyle, J., A Truth Maintenance System, Artificial Intelligence, Vol. 12 (1979), p. 231.

(2) Doyle, J., The Ins and Outs of Reasoning Maintenance, Proc. Eighth International Joint Conference on AI, Karlsruhe, Germany, (1983), p. 349.

(3) de Kleer, J., Problem Solving with the ATMS, Artificial Intelligence, Vol. 28 (1986), p. 197.

(4) Orciuch, E. and Frost, J., ISA: Intelligent Scheduling Assistant, The First Conference on AI Applications, IEEE Computer Society, (1984), p. 314.

(5) Putnum, L.H., A General Empirical Solution to the Macro Software Sizing and Estimating Problem, IEEE Transaction Software Eng., Vol. SE-4 (1978), p. 345.

(6) Minsky, M., A Framework for Representating Knowledge in the Psychology of Computer Vision, (1975), McGraw-Hill.

(7) Ogawa, Y., Shima, K., Sugawara, T. and Takagi, S., Knowledge Representation and Inference Environment: KEE, An Approach to Integration of Frame, Prolog and Graphics, Proc. of the International Conference on Fifth Generation Computer Systems, (1984), p. 643.

(8) Stefic, M., An Examination of a Free-based Representation System, Proc. 6th IJCAI, (1979), p. 845.

(9) McDermott, J., R 1: A Rule-Based Configurer of Computer Systems, Artificial Intelligence, Vol. 18 (1982), p. 39.

(10) Marshall, I. S., Using Declarative Knowledge Representation Techniques: Implementing Truth Maintenance in OPS 5, The First Conference on AI Applications, IEEE Computer Society, (1984), p. 261.

(11) Cambell, J.A, Implementations of Prolog, (1982), Ellis Horwood Limited.

(12) Apt, R. and VanEmden, M.H., Contributions to the Theory of Logic Programming, JACM, Vol. 29, (1982), p. 841.

(13) Smith, B.C., Reflection and Semantics in Lisp, Proc. 11th ACM Symposium on Principles of Programming Languages, (1984).

(14) Jaques, F., Computational Reflection in Class Based Object Oriented Languages, Proc. ACM Conference on Object-Oriented Programming, Systems Languages and Applications, (1989).

(15) Pattie, M., Concepts and Experiments in Computational Reflection, Proc. ACM Conference on Object-Oriented Programming, System, Language and Applications, (1987).