日本機械学会論文集(C編)
58巻552号(1992-8)

論文 No. 91-1694

機械加工における知識ベースの無矛盾管理システム*

長坂保美*1, 秋山牧雄*2
大滝英征*2, 石川義雄*2

Managing System of Knowledge Base in Machining

Yasumi NAGASAKA, Makio AKIYAMA,
Hideyuki OTAKI and Yoshio ISHIKAWA

It is possible to incorporate a system which manages a knowledge base into an expert system. Because, by using this system, we can rebuild knowledge bases which have no logical contradictions with knowledge of experts and phenomena which we experience daily. However, it is difficult to build beyond systems in the machining field due to the existence of many kinds of knowledge, if we want to achieve rebuilding of the knowledge bases using current representation language. Thus, we developed a truth maintenance system for knowledge bases by using a new predicate-logic representation language. This paper describes how to manage knowledge bases and how to treat the functions in a truth maintenance system using this system when contradictions occur in the practical machining problem.

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

1. 緒　　言

エキスパートシステムの知識ベースは，専門家の知識や経験に基づいて構築される．しかし，専門家の知識は，装置の稼働状況とか経済的な周辺環境などをにらみながら更新，蓄積していくという過程を踏む．それに応じて知識ベースも更新される必要がある．そのためには，知識ベースを矛盾なく再構築し，知識間の整合性を常に維持するような管理機構が必要となる．このような管理機構として，TMS (Truth Maintenance System)[1]や，ATMS (Assumption - based TMS)[2]が研究開発され，スケジューリングシステム[3]などの管理分野で具現化されている．

しかし，機械工学分野の知識ベースの管理に適用した例は，著者の知る限りない．機械工学分野の専門家は，一度決定されれば変更のほとんどない知識と，逆に時々刻々変化する知識とが混在している雑多な知識の中から，最適な知識を選び出し利用している．それゆえ，エキスパートシステムを構築するとなると，必ず複数の知識表現を混在せざるを得ない．そのため，従来のTMSやATMSのように，限定された知識表現でしか利用できないものであっては，これに十分に対処しきれない．

そこで，著者らは知識表現の異なる知識ベースの総合的な管理[4][5]を目的として研究開発を行ってきた．そして，フレーム表現やルール表現で構築された知識ベース，さらにはC言語による数式などの知識ベースを有機的に結合し，知識表現の異なる知識ベース間の推論を可能にした．しかし，TMSのような知識ベースの管理機構を擁しているものではなかった．

そこで，本研究では，まず知識ベースの記述形式をフレーム表現の記述形式と同じS式（LISP言語で処理するデータ：Symbolic expression）で統一し，知識表現の異なる知識ベースを総合的に管理できるよう配慮した．これにより，フレーム表現を基にした知識ベース間の統合性や可読性，さらには知識ベースの管理機構の組込みやそのアルゴリズム開発に有効に機能することになった．ついで，知識ベースの管理機構TMSについては，新たに述語論理表現のProlog処理系を開発し，これを適用した．

本報告は，この処理系を用いて機械加工条件を知識ベース化した機械加工管理システムの知識表現，およびこれを管理するTMSの具体例を示した．

* 原稿受付　平成3年12月19日．
*1 学生員，埼玉大学大学院．
*2 正員，埼玉大学工学部（☎338 浦和市下大久保255）．

2. 開発した述語論理表現(Prolog)処理系

Prologは，ユニフィケーションとバックトラック機能を特徴とする述語論理表現の言語である．図1は，著者の開発したPrologの記述と，一般のProlog (DEC-10 Prologあるいは Prolog-KABA) の記述の比較例を示したものである．

一般のPrologが「述語名 (引数, 引数, …)」を基本記述形式としているのに対し，本処理系では記述形式を「(述語名 引数 引数 …)」とした．

この記述形式は，述語名と引数がS式内で表現されているので，述語名を変数として持つことができる．そのため，メタ知識表現が可能となるので，3・4節で後述するTMSの「ノード生成」と「矛盾解消」に適用できる．しかも，フレーム表現にある推論機能により矛盾解消のアルゴリズムをルール表現などの知識表現にも簡単に適用できる．

3. 知識ベースの管理機構を持つ機械加工管理システム

3・1 機械加工管理システムの構成　図2は，知識ベースの管理機構を持つ機械加工管理システムの構成を示したものである．

機械加工管理システムは，加工条件推論部，知識ベース（KB：機械加工条件に関する知識を格納），データベース(DB：推論結果を格納)から構成されている．KBは機械加工の専門家の知識を2章の記述形式で構築されたものである．DBはKBの最新の推論結果をフレーム表現で格納している．また，TMSはノード生成システムと矛盾解消システムから構成され，何らかの理由で加工条件に不都合が生じた場合に起動される．

```
FACT:                          FACT:
 red(apple).                    (red apple)
 on(book,desk).                 (on book desk)
RULE:                          RULE:
 bird(x):-canary(x).            ((bird ?x) (canary ?x))
 above(x,z):-on(x,y),above(y,z). ((above ?x ?z)
                                  (on ?x ?y) (above ?y ?z))
(a) Example Prolog             (b) Example Prolog
    which is used in DEC-10        which we developed
```

図 1　著者の開発したPrologの記述の例

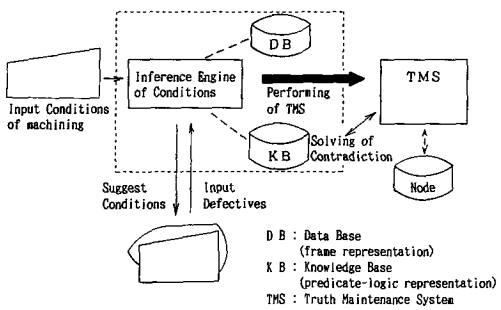

図 2　知識ベースの管理機構TMSを持つ機械加工管理システムの構成

3・2 機械加工管理システムの推論過程　図3は，図2の機械加工管理システムの推論過程と，その間でやりとりされる知識表現の例を示したものである．いま，図3の括弧の番号に沿って推論過程を追ってみると，

(1)　「フライス盤名」の入力を行う．

(2)　該当するフライス盤で切削する条件（材質，硬さなど）入力を行う．

(3)　入力を基にDB内の該当する加工条件（工具径，切込み，回転数など)がユーザに提示される．ここで，DB内に該当する加工条件が存在しない場合は，ISA関係により付加手続き（Attached-Procedure-1）が起動し(4)が行われる．また，提示された加工条件が不都合の場合は(5)が行われる．

(4)　条件推論の付加手続き（Attached-Procedure-1）がISA関係により実行され，加工条件推論部が起動される．すると，加工条件に該当するKBがロードされ，TMSのノード生成を行った後，ノード生成の初期状態(矛盾解消のための推論は実行されない)が結果として示される．この結果は，新たなDBに格納され，同時にユーザに提示される．もし，この提示された条件が不都合な場合は(5)が行われる．

(5)　3・4節で述べるTMSが起動され，矛盾解消

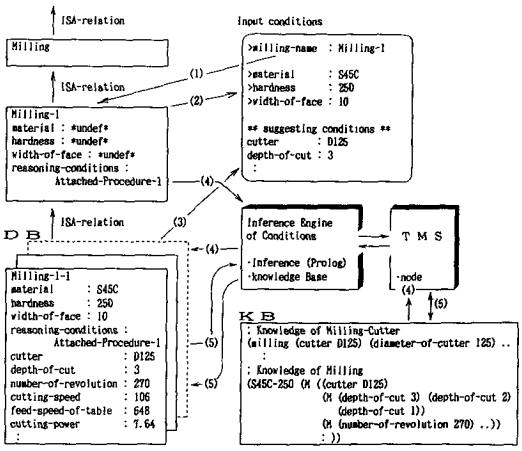

図 3　機械加工管理システムの推論過程

のための推論が実行される．これにより，DB 内部が更新され，同時にユーザに提示される．もし，この加工条件でも不都合な場合は，最終的に満足するまで(5)が繰返され，DB と KB とが更新される．このように，KB は述語論理表現の TMS により管理され，かつ推論はフレーム表現の付加手続きを用いて起動される．そのため，専門家の知識は記述性の高いフレーム表現で表示されるので，初心者でも容易に理解できる．また，加工条件推論部は述語論理表現で構築されているので，表現言語独自のバックトラック機能により容易に矛盾解消のための推論機能が実現され，効率的なシステムとなる．

3・3 知識ベースの内部構造 図 4 は，機械加工(フライス加工)の知識ベースの内容を示したものである．これからもわかるように，機械加工の専門家の知識は述語論理表現の「表」形式により表現されている．

例えば，図 4 中のフライス (Milling-Cutter) の知識は，以下のようなことを意味している．

「(cutter D 80)，(diameter - of - cutter 80)，(numbr-of-cutter 5) という三つのデータが milling という関係にある」

すなわち，2 章の記述形式と対応させると，述語名は milling，引数は (cutter D 80) (diameter-of-cutter 80) (number-of-cutter 5) が該当する．

ここでさらに，機械加工の専門家の表現にできる限り近づけるため，数名の専門家の知識表現を分析した．その結果，3・4 節で後述する「もし…」という仮定的な知識（図 4 中の「M」）と，「ただし…」という制約的な知識（図 4 中の「L」）の表現を取入れている．この「M」と「L」の表現は，以下のように記述される．

M (切込み 3) (切込み 2) (切込み 1)

L (切込み 2) (回転数 200)

この「M」の意味は，「もし，(切込み 3) を否定する事実があれば，(切込み 2) を仮定する」という内容を表している．これに対し「L」の意味は，「ただし，(切込み 2) で (回転数 200) とすると，何らかの不都合が生じて好ましくない」という内容を表している．

この表現を採用すれば，機械加工特有な「回転数は 270 か 200 とし，切込みは 3, 2, 1 のいずれかで調整する．ただし，回転数が 200 で切込み 2 で行うと不都合が生じる」といった表現を可能にしている．そのため，専門家の知識に沿って知識表現しやすく，初心者でも知識ベースの内容を容易に理解できる．

```
; 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))
  :
```

図 4 本システムの知識ベース(フライス加工条件の例)

3・4 知識ベースの管理機構 TMS TMS は，知識ベースの内容を矛盾なく管理するためのシステムである．具体的には，知識ベースの内容の理由づけ (Justification) の集合を新しく得られる理由づけの集合と整合させるように，信念 (belief) の集合を動的に変化させるシステムといえる．

信念には IN と OUT の二つの状態が持たされる．IN の状態は，信念を信じ得るだけの十分な根拠がある場合で，少なくとも一つの正当な理由づけを持っている．これに対し，OUT の場合は，信念を信じ得る根拠が全くないか，正当でないとする理由づけが明確に存在する．TMS はこの IN と OUT の状態を用いて，以下のようにして理由づけの整合性を管理する．

(a) TMS の理由づけ

知識ベースの知識に対して，有効な理由づけを行うためノード(node)を設ける．ノードには，仮定による理由づけ (SL 形理由づけ：Support - List justification) と，条件証明による理由づけ (CP 形理由づけ：Conditional-Proof justification) とがある．この二つの理由づけを基にノード間の関係を表現し，知識ベース内の矛盾を解消する．すなわち，

(イ) 仮定による理由づけのノード：

(ノード番号 SL IN リスト OUT リスト)

のように表現される．ノードの状態が IN であるためには，IN リストのノードがすべて IN で，OUT リストのノードがすべて OUT の場合である．

(ロ) 条件証明による理由づけのノード：

(ノード番号 CP 結論部 条件部)

のように表現される．ノードの状態が IN であるため

には，例えば，条件部に記述されているノードのすべてがINである場合，結論部がINである場合が該当する．そして，例えば後述する図6のノード1001は，「条件部（1 2 5）のノードがすべてINなら，ノード1000がINになる」ということが指示される．このことは，「ノード1000がOUTである」という事実に矛盾する．つまり，この矛盾は（1 2 5）が同時にすべてINにならないことによる．そこで，この中のいずれかがOUTになるように探索が実行され，矛盾解消が行われる．

（b） 矛盾解消のための戦略とノード生成

図5は，図4の加工（milling）に関する知識を基に生成されたノードとその状態を示したものである．つまり，図4中の仮定的な知識「M」と制約的な知識「L」は，2章で述べたメタ知識表現を適用して，図5のようなノードを生成する．そして，この二つの知識「M」と「L」とを組合せることによって，専門家の意図する矛盾解消戦略が実現される．この解消戦略は次の三つの原則に基づいている．

（1） 知識「M」内で知識要素（図4中の「切込み」や「回転数」など）のノード状態が変更される．

（2） 知識「M」内では，左側の知識表現からノード状態の変更が優先的に行われる．

（3） 知識「L」内の知識要素と同じノードがINの状態の場合，再度矛盾解消が起動される．

図5のノードは，TMSの矛盾解消過程が上記三つの原則を効果的に実現できるように生成される．ここで，図5に従ってこのノード生成の原理について述べる．

加工に関する知識「M」の知識要素をおのおのA_1, A_2, A_3, …, A_nとすれば，仮定的な知識「M」は以下のように示される．

$(M\ A_1\ A_2\ A_3\ \cdots\ A_n)$

知識要素間の関係は，以下の論理式で示すことができる．なお，$A_{not-all}$は知識「M」以外のすべての知識の集合を示している．つまり，$A_{not-all}$が仮定された場合は，この知識内ではIN状態のノードを生成することができない．

$A_1 \cup A_2 \cup \cdots A_n \cup A_{not-all}=1$

$A_1 \cup (A \cup \cdots (A_n \cup A_{not-all})\cdots)=1$

ここで，A_1, A_2, …, A_n, $A_{not-all}$は，おのおのが独立した知識要素としてとらえることができるので，以下の関係として考えることができる．

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

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

知識「M」のノード生成は，上記論理式の関係で示されるように必ず一つの知識要素が選択される．この際に，この知識要素を選択した正当な理由づけ（SL形理由づけ）を行う．つまり，このSL形理由づけは，上記論理式から「$\bar{A}_1$がOUT状態であるから，A_1はIN状態である」という論理づけを行う．実際には，上式から「A_2以下がOUT状態であるから，A_1はIN状態である」という論理づけのノード（図5中の2, 3, 4のノードと5, 6のノード）を生成する

ここで，もし何らかの理由でA_1がOUT状態になる（矛盾が生じる）と，上記の関係と同様に「A_3以下がOUT状態であるから，A_2はIN状態である」という理由づけのノードが生成される．

これに対し，図4中の知識「L」は，図5中のノード900（not_pairs）を生成する．このノードは，次章で後述する（contradiction）と同様な矛盾解消のためのノードである．つまり，（contradiction）が外部からの矛盾解消のためのノードであるのに対し，（not_pairs）は知識ベース内部からのノードである．

```
(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) ())
```

図5　図4のノードの状態（TMS起動前）

4. 機械加工におけるTMSの適用例

TMSには，原因が明確な場合と不明確な場合に対する矛盾解消の機能が組込まれている．図6と7は，原因が不明な場合のTMSのノードの状態変化を示したものである．図8は，原因が明らかなノードの状態変化を示したものである．ここで，これらのノードの状態変化による矛盾解消について述べる．

図6と7は，ノード1000の加工条件で加工を行った結果，「不都合が生じているが原因は不明」という仮定の基に，最適な加工条件を探索し矛盾解消を行う．

（1） 不都合が生じたために現加工条件は不適当であると判断し，その旨を入力した場合は，

（contradiction 1000）

というノードの知識を生成する．これは「ノード1000が矛盾を起こしている」ことを示している．

（2） TMSは，ノード1000が状態OUTになる

ようにCP形理由づけによってノードの知識1001 (nogood 1000) を生成する．そして，バックトラックを起動する．すると，ノード (1 2 5) のいずれかが状態OUTになるように，解消戦略に基づいて経路探索がなされる．

(3) 前述の解消戦略により，ノード2を状態OUTの候補とし，ノード内のIN/OUTリストを調べ，OUTリストのノード3を状態INへ変更することを試みる．

(4) ノード3は以下に示すノードに変更され，ノード3, 2, 1000の順に状態が変更される．そして，新たなノード1002が生成される．

```
(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) ())
```

図 6 第1回めのTMS起動後（制約なしの場合）

```
(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) ())
```

図 7 第1回めのTMS起動後（制約ありの場合）

```
(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) ())
```

図 8 第1回めのTMS起動後（原因が明らかな場合）

(3 SL (1001) (4 11)) → 状態IN

(1002 SL (1 3 5) ()) → 状態IN

(5) 制約的な知識に適合する（状態はIN）ノードが存在するかが探索される．ここで，適合しない場合は図6のようにTMSを終了する．しかし，制約的な知識に適合するノードが存在する場合（図7のノード900）は，このノードが状態OUTになるように(2)に戻る．図7はこの過程を経て，TMSが終了した状態を示したものである．

この加工条件で加工がうまくいけば，この条件がフレーム形表現のDBに格納される．しかし，この条件でも加工がうまくいかなかった場合は，再度TMSが起動され，矛盾解消まで繰返される．

これに対し，図8は原因が明らかな場合のノードの状態変化を示したものである．ここでは，ユーザ入力により「明らかに回転数に問題がある」という知識は，

(contradiction 100 5)

というノードの知識を生成する．そして，ノード5が状態OUTになり，矛盾が解消される．そして，ノード1002が生成され，TMSを終了する．

図9はTMSを再度繰返し，最終的にカッタ (D125) ではフライス加工ができなくなった状態を示しており，カッタ (D160) を使用するように知識ベース内が更新された状態になっていることを示している．図10～12は，本システムのTMS実行画面例で，

```
(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) ())
```

図 9 第8回めのTMS起動後（制約なしの場合）

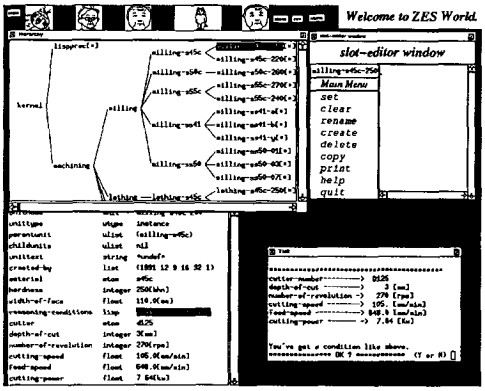

図 10　TMS 起動の直前状態の実行画面

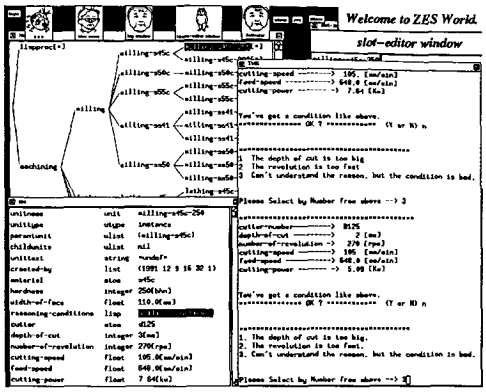

図 11　TMS 起動の実行画面（原因が不明）

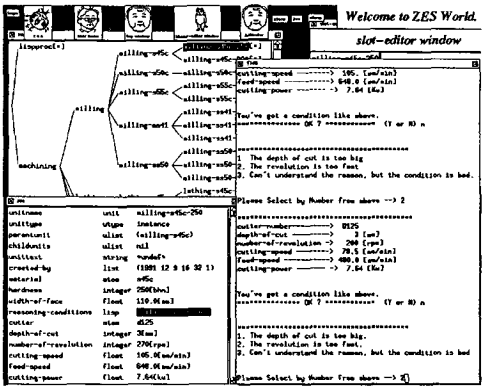

図 12　TMS 起動の実行画面（回転数に問題）

3･2 節の(5)の状態を示している．図 10 は，TMS 起動直前の状態を示しており，フレーム内部の状態が TMS 起動時の状態として示される．図 11 と 12 は，TMS 起動時の状態を示しており，「不都合が生じているが原因が不明」と「明らかに回転数に問題あり」という状態を繰返した実行例を示している．これら図中右部の TMS の起動により，最終的な加工条件が最新状態としてフレーム表現の内部に格納される．なお，TMS の実行では，カッタ，回転数，切込み，テーブル送り速度などの加工条件が提示される．

このように，本システムの知識ベースは，解消戦略によってノードの状態を変化させ，最適な知識ベースの状態になっていることがわかる．

5. 結　　言

フライス加工の専門家（平均 14 年経験）8 人を対象に，本システムの評価を行った．専門家の中には，「回転数を徐々に落とさず一度に半分程度にする」，「回転数を落としすぎる」などがあげられたが，大方の意見(知識)は矛盾解消として最良の解消戦略がとれているという評価を得た．また，無人化や安全性の点から，本システムが NC 加工にかなり有効であるという評価も得られることができた．なお，本報告は，知識ベースの管理機構を構築し，その管理機構が矛盾なく機能することを確認することを第一目的とした．そのため，管理機構の動作原理を理解しやすくするため，矛盾解消の対象を限定して述べてきたが，本来対象の大きさに関係なく同じ管理機構で無矛盾の知識ベースを管理することは可能である．ここで，本研究をまとめてみると以下のようになる．

（1）　知識ベースの管理機構を持つ本システムは，時々刻々変化するような管理機構として，その有用性や構築性に大いに期待される．

（2）　管理機構の矛盾解消において，専門家の解消戦略を知識ベース中に組込むことを可能にしている．これにより，専門家の知識を無理なく表現できるので，その構築性に専門家より高い評価を得た．

（3）　知識ベースの記述形式の統一により，知識ベース間の統合性や可読性，および管理機構の開発に有効である．特に，本記述形式によりメタ表現を可能にしているため，その実用性が高い．

文　　献

（1）　John, D., *Artificial Intelligencee,* 12(1979), 231-272.
（2）　Jonan de K., *Artificial Intelligence,* 28(1986), 197-224.
（3）　大場・都島・薦田・山中，情報処理学会論文誌，5(1989), 668-677.
（4）　長坂・ほか 2 名，機論，58-547, C(1991), 1151.
（5）　長坂・ほか 2 名，機論，58-549, C(1991), 1385.