論文 90-27-1-5

전문가 시스템 개발을 위한 Knowledge Base Editor의 구현

(A Knowledge Base Editor for Building Expert Systems)

金 在 熹* 申 東 弼**

(Jai Hie Kim and Dong Pil Shin)

要 約

전문가 시스템 개발도구의 지원환경으로써, 프로덕션언어인 OPS5를 이용하는 사용자가 보다 빠르고 쉽게 지식베이스를 구축할 수 있게 하는 knowledge base editor를 구현하였다. 구현된 knowledge base editor는 OPS5에서 사용하는 rule의 구문과 타입에 관한 지식을 갖고 이에 대한 오류를 점검하는 기능과 특정한 rule이 주어졌을때 그 rule들에 대한 탐색 기능, 그리고 rule의 생성자와 생성일시에 관한 정보를 자동적으로 기록하는 automatic bookkeeping 기능 등을 제공한다.

Abstract

In this paper, a knowledge base editor is presented as a supporting environment for an expert system building tool, OPS5. The knowledge base editor is especially useful for the fast and easy development of a knowledge base when the OPS5 production language is used. This knowledge base editor has some special facilities such as syntax and type checking, rule browsing and automatic bookkeeping. The syntax and type checking provides the facilities to find syntax and type errors in an edited knowledge base, respectively. The rule browsing facility offers various pattern matching schemes to see the causes and effects of a concerned rule. Automatic bookkeeping keeps the updated date and user name of a rule for the later reference whenever a user adds or changes a rule.

I. 서 론

최근에 이르러 인공지능의 연구는 여러분야에서 괄목할만한 성장을 보이고 있으며, 이중에서도 특히 주목을 받고 있는 것은 전문가 시스템일 것이다. 전문가 시스템은 전문응용분야에 관한 전문가의 특별한 지식(domain knowledge)을 저장하는 지식베이스(knowledge base)와 지식을 처리하고 문제를 풀이하며 시스템과 사용자와의 연결에 관한 지식을 저장하는 추론기관(inference engine)으로 구성된다.[1]

이와같은 전문가 시스템은 그 개발을 쉽고 빠르게 하기 위하여, 여러 분야에 걸쳐 다양한 전문가 시스템 개발도구가 제공되고 있으며, 이들은 추론기관을 제공하고 있으므로, 전문가 시스템을 개발하는이(지식공학자)는 개발도구가 요구하는 형태대로 지식베이스만을 구성하면 되는 것이 보통이다.

전통적으로 지식베이스를 구성하는 지식획득 과정은 지식공학자가 다루는 문제분야의 전문가와의 면

*正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)
**正會員, 韓國科學技術研究院 附設 시스템工學센타
(System Engineering Research Institute, KIST.)
接受日字 : 1989年 10月 10日

담을 통해서 이루어진다. 이러한 경우 지식공학자는 문제분야의 여러 개념 및 계약과 전문가가 사용하는 해법 등을 이해하고 이를 다시 프로그램하여 지식베이스를 구성하여야 하므로 지식획득은 전문가 시스템 개발에 대한 소요시간과 경비의 큰부분을 차지하게 된다. 이러한 지식획득에 대한 어려움과 문제점에 대한 인식및 이의 극복을 위한 시도는 전문가 시스템 연구의 초기에서 부터 진행되어 왔으며, 현재 이에 대한 연구는,

1. 지식공학자가 지식베이스를 구성하는 경우에 편의를 제공하는 knowledge base editor에 관한 연구[2,3,4,5,6]

2. 지식공학자의 관여없이 문제분야의 전문가가 직접 지식베이스를 구성하는 자동적 지식추출(automatic knowledge extraction)에 관한 연구[3,7,8,9,10]

3. 일단 구성된 지식베이스에 대하여 중복, 모순, 완전성등을 검토하여 지식베이스의 내용을 개선하기 위한 연구[11,12]

등으로 요약 될 수 있다.

Knowledge base editor는 간단한 경우에는 수동적인 작업으로써 지식베이스를 편집하는 보통의 문서편집 기능만을 갖지만, 좀더 정교한 경우에는 지식베이스의 특성에 맞는 여러 형태의 기능을 제공할수 있다. 예를 들면 EMYCIN[13]의 경우에는 automatic bookkeeping의 기능이 있는데, 이는 지식베이스의 내용을 변화시키거나 새로운 내용을 삽입시킨 사람과 그 일시에 관한 정보를 자동적으로 기록하여 추후에 참고자료로 이용하게 하며, 이는 여러 지식공학자가 공동으로 지식베이스를 구성할 때 유용한 기능이 된다. KuBIC[2]의 경우에는, 새로이 추가되는 지식에 대하여 이의 분류학적 위치가 현존하는 지식베이스의 어디에 해당하는지를 찾아내어 적합한 곳에 삽입시켜 주며 TDE[3]와 IMPULSE[4]는 각각 TEST[14]와 STROBE[5]에 관계된 editor로써 이들은 주요개념을 프레임(frame)들의 network으로 표시하고 다중 윈도우(multi-window)로써 각각의 윈도우에 개념들간의 계층적 관계나 각 개념의 속성들을 보여주며 편집할 수 있도록 되어 있다. 그밖에 GURU[6]는 rule단위로 지식베이스를 생성, 편집, 삭제할 수 있으며 특정 패턴(pattern)을 갖고 있는 rule만을 검색하여 편집할 수 있는 간단한 browsing 기능을 갖춘 모듈 editor를 제공한다. 그러나 이러한 기능들은 특정한 개발도구의 지원환경에 따라 부분적으로 제공되고 있으며 이에 반하여 종합적인 기능을 갖춘 knowledge base editor의 개발이 필요하다.

본 논문에서는 앞에서 예시한 여러 개발기구에서 제공되는 editor의 단편적 기능들을 종합하여, 상업용으로 널리 쓰이나 지원환경으로써 editor가 제공되지 않는 OPS5[15]를 사용하여 지식베이스를 구축하는 경우에 유용한 knowledge base editor를 구현하였다. 구현된 knowledge base editor는,

1. OPS5 구문(syntax)에 대한 지식을 갖고, 구문에 대한 오류를 검색하는 구분점검 (syntax checking) 기능

2. Object-Attibute-Value(O-A-V) 형태의 지식표현에서 타입(type)의 오류를 지적하는 타입점검 (type checking) 기능

3. Automatic bookkeeping 기능

4. 지정된 rule이 영향을 받을 수 있는 rule 들과 이 rule에 영향을 줄 수 있는 rule들의 탐색 및 지정된 작업 메모리(working memory)의 내용을 변경시키는 rule들의 탐색에 대한 rule browsing 기능

5. Rule 편집시에 윈도우를 설정하여 다른 모듈(function, literal등)을 함께 편집할 수 있는 기능 등을 갖는다.

특히 4의 rule browsing 기능은 지식베이스의 규모가 큰 경우에 rule 들간의 상호연관성을 찾는데 편리한 기능이며 이와 같은 기능은 일단 편집된 rule의 자료구조를 스트링 형태에서 association list로 변환하여 이용함으로써 다양한 탐색기능을 제공하게 된다. list로 구성된 rule들이 보조기억장치에 놓일 때에는 text 형태로 변환되어, 다른 기존의 text editor를 사용한 경우와 차이가 없게 된다. 반대로 text 형태로 보조기억장치에 놓인 내용들은 재편집되기 위하여 list 형태로 변환되어 main memory에 놓이게된다.

본 논문의 Ⅱ장과 Ⅲ장에서 각각 knowledge base editor의 개요와 구현에 대하여, 그리고 Ⅳ장에서는 구현된 knowledge base editor를 이용한 처리결과를, Ⅴ장에서는 결론과 향후 연구과제에 대하여 기술하였다.

Ⅱ. 시스템 개요

전체 시스템은 크게 Full edition과 Module edition의 두 부분으로 구성된다.

Full edition 부분은 기존의 text editor(예 : Turbo editor, Word Star)를 연결하여, 전체 지식베이스 화일을 신속히 편집하는 기능을 갖는다. 이와같은 기능은 숙련된 OPS5 사용자에게 있어서 전체화일의 내용(strategy부분, literalization부분, function부분, rule부분, main부분으로 구성되는 하나의 OPS5 프로그램)을 보고 편집하는 방법을 제공한다.

Module edition 부분은 전체 지식베이스 화일을 앞에서 언급한 strategy 부분, literalization 부분, function 부분, rule 부분, 그리고 top level에서 이용되는 명령어(command)로 구성되는 main 부분으로 나누어 편집하는 기능을 갖는다. 각 부분별 편집에는 단순한 편집기능외에 편집에 유용한 여러가지 기능들이 부가되며 이와같은 기능을 요약하면 다음 그림 1과 같다.

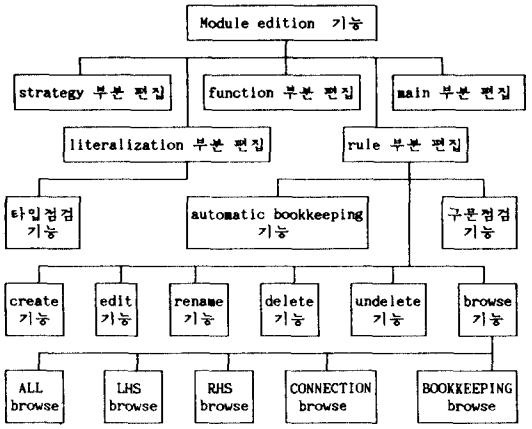


그림 1. Module 편집 기능
Fig. 1. Module edition functions.

Rule 편집부분은 OPS5 프로그램의 대부분을 차지하는 중요한 부분으로 그 효율적인 편집을 위해서 rule 단위로 생성, 삭제, 편집할 수 있는 기능, rule 이름 변경기능, 그리고 실수로 인한 rule의 삭제에 대해 그 삭제된 rule을 복구할 수 있는 기능들을 갖는다. 그 밖에 automatic bookkeeping 기능과 일단 편집된 rule의 내용을 검색하여 구문상의 오류를 발견하여 그 내용과 위치에 대한 정보를 제시하여 주는 구문점검 기능을 갖는다. 그러나 rule의 편집에 있어서 무엇보다도 중요한 것은 rule들간의 상호연관성이며, 이를 위해서 특정조건 element를 가진 rule에 대한 탐색(LHS browsing), 데이타 메모리상의 특정 내용을 변경시키는 rule에 대한 탐색(RHS browsing), 주어진 rule이 영향을 받는 rule과 영향을 줄 수 있는 rule에 대한 탐색(CONNECTION browsing) 그리고 특정 이용자가 생성한 rule이나 특정 날짜 이전 혹은 이후에 생성된 rule에 대한 탐색(BOOKKEEPING browsing) 등의 다양한 탐색기능을 부가하였다.

Ⅲ. 시스템의 구현

1. 모듈 편집 (Module edition)

전체 화일 내용을 각 부분별로 나누어 편집하고, 한 화면내에서 다른 부분도 함께 편집할 수 있는 모듈편집이 가능하기 위해서는 임의의 크기로 윈도우를 설정하여 편집할 수 있는 편집기능이 필요하며 이를 위해 다음 그림2와 같은 알고리듬에 따라 간단한 문서 편집기(text editor)를 구현하여 각 부분별 편집에 이용하였다.

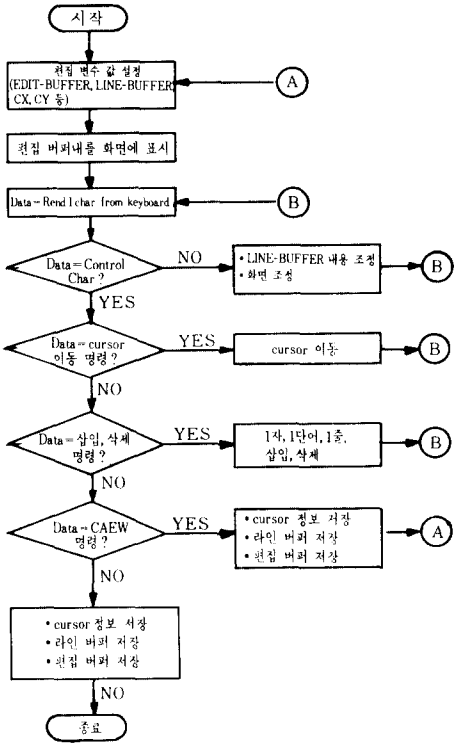


그림 2. 문서 편집기(text editor) 알고리듬
Fig. 2. Algorithm for the text editor.

우선 설정된 편집작업의 윈도우에 따라 프로그램에서 이용되는 주요 변수값들을 지정하고, 편집하고자 하는 내용을 편집 버퍼(EDIT-BUFFER)로 옮긴다. 프로그램에서 이용되는 변수로 MAX-COL과 MAX-LINE은 MAX-COL과 MAX-LINE을 넘는 경우를 처리하기 위해 LE와 PAGE 변수를 설정하여 수평 scroll과 수직 scroll시에 이 값들을 증가 시킴으로써, 편집 윈도우상의 커서 위치([CX, CY]) 와 이에 대응하는 편집 버퍼상의 위치([POSX, POSY])

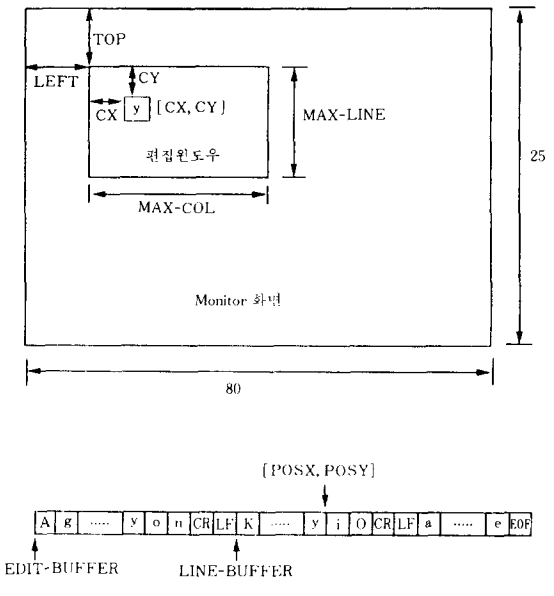


그림 3. 문서 편집기(text editor)에서 이용되는 주요변수

Fig. 3. Variables used in the text editor.

를 다음과 같은 방법으로 계산하여 이용한다(그림 3 참조).

편집 버퍼가 지정된 후 스택으로 부터 저장된 커서의 위치 정보(CX, CY, LF, PAGE)가 지정되며 위와 같은 계산 방법으로 POSY가 계산되고, 계산된 POSY 값에 따라 편집 버퍼로부터 한 줄분의 스트링이 라인 버퍼(LINE-BUFFER)에 지정된다.

위와 같은 과정에 따라 모든 편집 환경이 설정되면 편집버퍼의 내용이 편집 윈도우에 표시되고 키보드로부터 한 자씩 입력을 받아 입력된 코드에 따라 그림 2의 알고리듬에 따라 지정된 작업을 수행 한다.

자별, 단어별, 삽입과 삭제 작업은 커서의 X좌표에 따른 POSX 값이 계산되어 라인버퍼상에서 행해지며, 라인버퍼의 내용은 커서의 Y좌표가 바뀔 때마다 POSY 값을 계산하여 편집버퍼(EDIT-BUFFER)의 지정된 위치에 삽입되고, 새로운 내용이 편집버퍼로 부터 갱신된다.

키보드로부터의 입력이 CAEW(Create Another Editing Window; Ctrl-2 keycode)코드이면, 현재 편집 작업에 대한 상태 정보(EDIT-BUFFER, CX, CY, LF, PAGE)와 편집윈도우에 대한 정보(TOP, LEFT, MXX-COL, MAX-LINE)가 현 작업윈도우에 대한 정보(TOP, LEFT, MAX-COL, MAX-LINE)가 현 작업윈도우에 관계된 스택에 저장되며 새로이 지정된 편집윈도우에 따라 새로운 환경이 설정되어 같은 방법으로 편집 작업이 진행된다.

여러개의 편집윈도우가 설정되어 있는 경우, 한 작업윈도우에서 다른 작업윈도우로 이동되어 편집될 때는 각 윈도우에 관계된 스택에 현 상태에 관한 정보가 저장되고 새로운 환경이 스택으로부터 설정된다.

한 편집 윈도우에서의 편집작업의 종료시에는 현 라인버퍼의 내용이 편집버퍼에 삽입되고 스택에 편집 환경의 상태(버퍼내용과 커서 위치)가 저장 되며 편집된 내용은 스트링 형태로 편집버퍼에 남게된다.

2. Automatic Bookkeeping 기능

Automatic bookkeeping은 rule의 생성자와 생성 일시에 대한 정보를 자동적으로 기록하는 기능이며 이를 위해 editor의 기동 직후 사용자의 이름을 스트링 형태로 입력받아 광역 변수(global variable)인 USER에 저장한다.

한 rule이 편집되고, 편집된 rule이 정의된 rule structure에 저장 될 때마다, 변수 USER와 현재의 시간을 돌려주는 LISP의 TIME함수로 부터 생성자와 생성일시에 대한 정보가 MAKER 슬롯(slot)과 DATE 슬롯에 아톰과(년, 월, 일, 시, 분) 형태의 리스트로 각각 저장된다.

실제 rule 편집작업후에 automatic bookkeeping 에 대한 자료가 rule structure의 MAKER 슬롯과 DATE 슬롯에 저장되는 예는 다음과 같다.

```
#s(RULE:S-BODY "(p ru-15                    ;string slot
  (GOAL ^name POSS)
→
  (modify 1 ^name GOOD  value 100))
":L-BODY(P RU-15 (GOAL (NAME POSS)) →;associate list slot
(MODIFY 1 (NAME GOOD) (VALUE 100)))
:MAKER YOUNHAKSOO:DATE(1988 11 8 1 36);MAKER and DATE slot
:COM"")                                    ;comment slot
```

3. 타입 점검(Type checking) 기능

OPS5 언어에서 정의되는 데이타 element는 O-A-V 형태의 element class로 정의되며 모든 element class는 사용되기전에 반드시 선언되어져야 된다. 이와같은 element class 선언에는 다음 형태와 같은 일상적인 오류가 발생할 수 있다.

Type 1 : 부적절한 element 이름이나 속성 이름으로 선언된 오류.

Type 2 : 이미 존재하는 element class 이름으로 재선언된 오류.

Type 3 : 한 element class내 2개 이상의 벡터-속성을 선언한 오류.

위와 같은 오류는 타입에 관계된 오류들로서 다음과 같은 자료구조의 변환으로 쉽게 점검된다. 이용자가 literalization을 선언할 때마다 element 이름과 속성 이름에 대한 적합성이 점검되고(type1 오류), 선언된 스트링 형태의 내용은(element-name att-1 att-2…att-n)형태의 리스트로 변환된 후 다시 리스트에 삽입되어 association list, LT를 구성한다. 선언된 내용이 association list, LT에 삽입될 때마다 현재 삽입되는 key(element 이름)가 LT의 key list에 존재하는 가를 확인하여 Type2 오류를 점검한다. Type 3 오류는 벡터-속성의 선언시 선언된 속성 이름으로 리스트, VT를 만들어 association list, LT의 내용과 비교함으로써 점검된다.

4. 구문 점검(syntax checking) 기능

구문 점검기(syntax checker)는 그림 4와 같이 두 부분으로 구성된다.

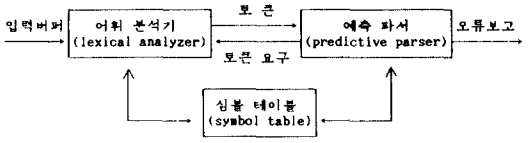


그림 4. 구문 점검기의 구성
Fig. 4. Structures of the syntax checker.

어휘 분석기는 파싱과정에서 이용되는 토큰(token; lookahead symbol)을 예측 파서의 요구에 의해, 입력 버퍼링 알고리듬[6]에 따라 하나씩 생성하여 돌려준다.

OPS5 rule의 문법은 컨텍스트 프리 문법(context-free grammar)이며 이를 파싱하기 위해서는 푸시-다운 오토메타(push-down automata)가 필요하다. 그러나 문법이 단순하고 구문 검색에 있어서는 파스 트리(parse tree)를 생성할 필요가 없으므로 간단한 예측 파서를 이용하였다. 예측 파서는 프러덕션(production)의 모든 넌터미날(nonterminal)들에 대한 각각의 프러시듀어(procedure)들로서 구성되며 OPS 5 구문 점검을 위한 이와같은 예측 파서의 구현은 다음과 같다.[16]

step 1) Left factoring 과정.

문법 상의 심볼 X에 대한 First 집합, First(X) 는 다음과 같이 정의된다.

i) X가 터미날이면 $First(X)=\{X\}$

ii) $X\rightarrow\varepsilon$ 이면 $First(X)=\{\varepsilon\}$

iii) X가 넌터미날이고 $X\rightarrow Y_1Y_2\cdots Y_n$가 프로덕션이며, 어떤 i에 대해 $a\in First(Y_i)$를 만족하는 a가 존재하고, 모든 i에 대해 $\varepsilon\in First(Y_1),\cdots,\ \varepsilon\in First(Y_{i-1})$이면 $a\in First(X)$이다.

예측 파서에 있어서 파싱할 문법에 $A\rightarrow\alpha\mid\beta$의 프러덕션이 존재하고, $First(\alpha)\cap First(\beta)\neq\Phi$이면 토큰(lookahead symbol) 만으로 어느 프로덕션을 전개할 지를 결정할 수가 없다. 이러한 문제점을 해결하기 위해서 $First(\alpha)\cap First(\beta)=\Phi$가 되도록 문법을 조정하는 left factoring의 과정이 선행되어야 한다.

OPS5 문법에 있어서 이러한 left factoring 과정은

```
lhs-term → ↑ constant-symbol-atom lhs-value
         | ↑ number lhs-value
```

와 같은 프로덕션이 다음과 같이 조정됨으로서 해결된다.

```
lhs-term  → ↑ lhs-term'
lhs-term' →   constant-symbol-atom lhs-value
          |   number lhs-value
```

step 2) Left-recursion의 제거

$A\rightarrow A\alpha\mid\beta$와 같은 left-recursion을 지닌 문법도 예측파서를 이용하기 위해서는 left-recursion이 제거되어야 한다. 이를 위해서는 OPS5 문법의 경우

expression→expression operator expression|num

와 같은 프로덕션은

```
expression → Num exp'
exp' →operator expression exp' | ε
```

형태로 조정된다.

step 3) 각 넌터미날에 대한 프로시듀어의 정의.

위와 같이 조정된 문법의 각 프로덕션에 대해 천이도(transition diagram)를 작성하고 이에 따라 lookahead symbol을 처리하여 최종상태로 천이하기 위한 프러시듀어를 정의한다. 스타팅 심볼(starting symbol), production에 대한 천이도는 다음 그림 5와 같으며 production에 관계된 프러시듀어는 상태 0에서 출발하여 어휘분석기로부터 토큰(lookahead symbol)을 받아 그 값에 따라 상태를 천이한다.

천이도 상의 LHS와 RHS는 넌터미날들로써 상태

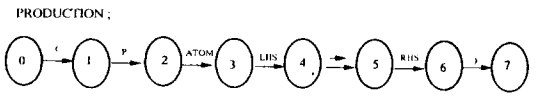


그림 5. 스타팅 심볼에 대한 천이도
Fig. 5. Transition diagram for starting symbol.

3과 5에서는 LHS와 RHS에 대응되는 프러시듀어가 수행된다. RHS에 대한 프러시듀어의 수행결과 상태 7에 천이되면 파싱이 성공적으로 종결된다.

이와 같은 방법으로 프러덕션의 각 넌터미날들에 대한 프러시듀어가 정의되며 예측파서는 이러한 프러시듀어들로 구성된다.

step 4) 구문 오류의 처리.

프러시듀어 수행중 한 상태로부터 다른 상태로 천이할 수 없으면 오류가 보고되며 rule의 크기가 그리 크지 않으므로 error recovery의 과정을 거치지 않고 어휘분석기로 부터 그 위치를 받아 오류 발생 위치와 오류 메시지를 이용자에게 보고한다.

5. Browsing 기능

Rule browsing 기능은 특정 조건을 만족하는 rule들에 대한 검색기능이며 이와 같은 검색기능은 rule의 조건부(LHS)나 행동부(RHS)에 대해서 O-A-V 형태의 데이타를 비교하게 된다. OPS5의 rule에 있어서 O-A-V 형태의 데이타 비교는 한 데이타 element내에서 그 attribute-value 쌍의 위치가 자유롭고 값으로서 변수도 포함될 수 있기 때문에 편집된 스트링 형태의 rule을 association list로 변환하여 매칭에 이용하였다. 비교할 데이타 element list로 변환하여 매칭에 이용하였다. 비교할 데이타 element들중에 변수가 있으면, 최초의 경우에는 무엇과도 매칭되나, 매칭된 내용을 관계된 변수에 binding시켜 놓고, 차후 같은 변수가 매칭을 시도하면 binding 된 내용이 매칭된다.

1) LHS browsing

LHS browsing 기능은 O-A-V 형태의 특정 데이타 element들을 조건부로 갖고 있는 rule들에 대한 검색기능으로 다음과 같은 방법에 따라 수행된다.

i) 비교할 데이타 element들을 받아 이를, 위에서 언급한 association list 형태로 변환한다.
ii) 비교할 데이타 element을 분류하여 보다 특정한 데이타 element(attribute-value 쌍의 수가 많은 데이타 element)순으로 정렬하여 리스트, L1을 만든다.
iii) 리스트, L1의 처음 데이타 element부터 association list로 변형된 각 rule의 조건부와 비교되어 부합되는 rule들만으로 리스트, L2을 형성하여 다음의 비교에 이용한다.
iv) 리스트, L2가 NIL이되면 비교작업을 중단하고 비교한 결과 값으로 NIL을 돌려준다.
v) 리스트, L2가 NIL이 아니면 L1에 있는 다음의 데이타 element에 대해 L2에 있는 각 rule과 비교되며 L1 리스트가 NIL이면 L2 리스트에 있는 rule들을 비교결과로서 돌려준다.

2) RHS browsing

RHS browsing은 특정 작업메모리의 내용을 변경하는 rule들에 대한 검색으로 이에는 특정 데이타 element를 작업메모리에 생성하는 rule에 대한 검색(Make browsing)과 특정 데이타 element를 작업메모리에서 삭제하는 rule에 대한 검색(Remove browsing)이 있다.

Make browsing은 LHS browsing 방법에서 데이타 element의 내용을 각 rule과 비교할 때 association list의 key가 Make인 부분과 비교되고 Remove browsing은 key로 Remove를 갖는 rule에서 Rement 뒤의 변수값에 해당하는 조건부의 데이타 element와 비교된다.

3) CONNECTION browsing

CONNECTION browsing은 rule들의 상호연관성에 대한 검색으로 지정된 rule이 영향을 받는 rule과 영향을 줄 수 있는 rule에 대한 검색기능이다.

지정된 rule이 영향을 받는 rule은, 지정된 rule의 조건 데이타 element에 대한 RHS browsing을 이용하여 검색되며, 지정된 rule이 영향을 줄 수 있는 rule은, 지정된 rule의 행동부(RHS)에 있는 Make와 Rement의 데이타 element에 대한 LHS browsing을 이용하여 검색된다.

4) Bookkeeping browsing

Bookkeeping browsing은 특정 이용자가 생성한 rule과 특정 날짜의 이전 혹은 이후에 생성된 rule에 대한 검색으로 각 rule의 maker slot과 date slot에 있는 automatic bookkeeping에 대한 자료를 검색하고자 하는 자료(생성자의 이름 혹은 생성일시)와 비교된다.

Ⅳ. 처리 결과

본 knowledge base editor는 LISP 프로그래밍 언어인 GCLISPLM[17]을 이용하여 PC용으로 구현하였고 다음은 각 부분 별 처리 결과의 예이다.

그림 6은 모듈편집 메뉴에서 rule을 선택하여 RU-10을 rule 이름으로 갖는 rule을 생성하는 경우로서 화면의 오른쪽, 함수 윈도우는 rule을 생성하면서 RU-10에 관계된 함수, test를 편집하는 예를 보여준다.

그림 7은 literalization 부분에 다음과 같이

Literalization 선언 :

(Literalize adr country city)

(Vector-attribute country)

의 literal이 선언된 후 city를 벡터-속성으로 선언할 경우, type 3 오류에 대한 타입점검의 한 예를 보여준다.

그림 8은 rule, RU-1에 대한 구문점검의 예로, 구현된 editor는 PHYCHAR에 오류가 있음을 지시하고, 지시된 위치에 RHS의 ACTION이 요구된다는 간단한 메시지를 제공한다. 그러나 구문점검에 있어서 다음과 같이 속성 지시자(attribute indicator)인 ↑가 생략된 오류나 조건부(LHS) 또는 행동부(RHS)에 NIL 리스트가 오는 오류는 검색할 수가 없었으며 이는 OPS5 rule 문법이 이들을 정상적인 문법으로 인식하는데서 오며 이를 해결하기 위해서는 원 문법의 내용을 수정하여 이들을 인식하지 못하도록 하는 과정이 필요하다.

그림 9는 LHS browsing 탐색에 대한 예로 다음의 data element

(PHYGOAL ↑age 50 ↑weeight 59)

(GOAL ↑name START ↑value T)

를 조건부로 갖고 있는 rule들이 RU-2와 RU-4임을 보여준다.

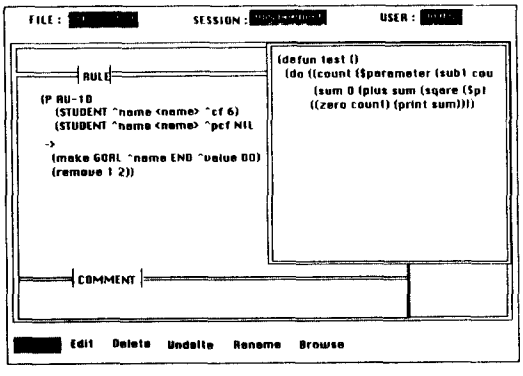


그림 6. rule과 함수의 편집 예
Fig. 6. An editing example of a rule and a function.

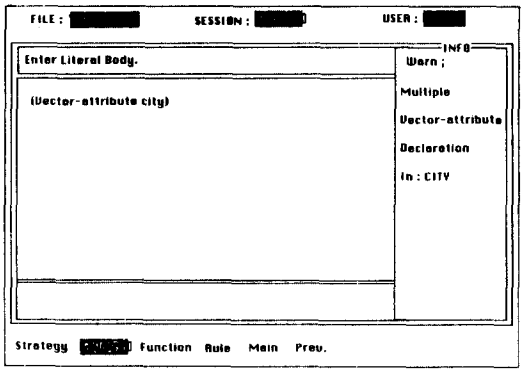


그림 7. 타입점검의 예
Fig. 7. An example of type checking.

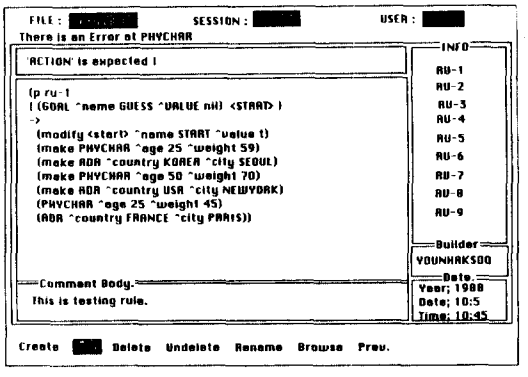


그림 8. 구문점검의 예
Fig. 8. An example of syntax checking.

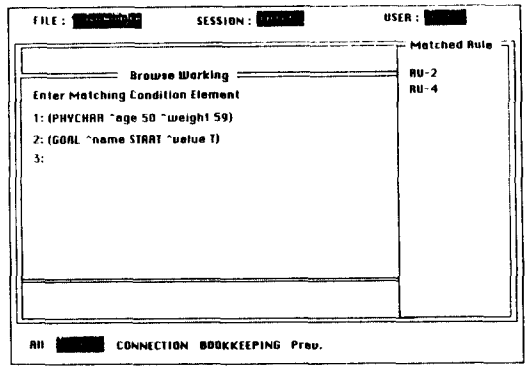


그림 9. LHS browsing의 예
Fig. 9. An example of LHS browsing.

V. 결론 및 연구과제

본 논문은 전문가 시스템 개발도구의 지원환경으로써, OPS5를 사용하는 이용자가 지식베이스를 보다 손쉽게 구축할 수 있는 knowledge base editor 의 설계 및 구현에 관한 것이다.

전체 프로그램 화일을 각 부분별로 나누어 편집하는 Module편집기능, OPS5 rule 구문에 대한 지식을 갖고 편집된 rule의 구문을 점검하는 구문점검 기능, O-A-V형태의 지식표현에서 타입의 오류를 점검하는 타입점검 기능, 지정된 rule이 영향을 받을 수 있는 rule들과 영향을 줄 수 있는 rule들에 대한 rule 탐색(browsing) 기능, rule 생성자와 생성일시에 대한 정보를 자동적으로 기록하는 automatic bookkeeping 기능, 그리고 rule부분 편집시 다른 부분을 동시에 편집할 수 있는 다중 윈도우등의 기능을 갖도록 구현된 editor는 이와같은 기능들을 통해서 OP OPS5를 이용하는 초보자의 경우에도 손쉽게 지식베이스 파일을 편집할 수 있는 환경을 제공할 수 있게 되었고, 전반적인 전문가 시스템 개발의 시간과 비용도 단축할 수 있게 되었다.

그러나 이러한 기능만으로는 전문가 시스템 개발에 있어서 가장 중요한 문제가 되는 지식획득(지식베이스의 구성)에는 여전히 어려움이 존재하게 되며, 이를 해결하기 위해서는 전문가와의 대화를 통해 rule을 학습함으로써 지식공학자의 도움없이 전문가가 직접 지식베이스를 구축할 수 있는 자동적 지식추출에 관한 연구와, 지식베이스내 지식간의 모순과 중복을 검출할 수 있는 지식개선에 대한 연구가 앞으로 함께 병행되어야 할 것이다.

參 考 文 獻

[1] D.A. Waterman, *A Guide to Expert Systems,* Addison-Wesley Publishing Company, U.S.A., 1986.

[2] T. Finin, and D. Shilverman, "Interactive classification: a technique for building and maintaining knowledge bases, *IEEE Proceedings workshop on principles of knowledge-based systems,* IEEE computer society press, pp. 107-114.

[4] G.S. Kah, "From application shell to knowledge acqusition system," *IJCAI-87,* pp. 355-358, 1987.

[4] E. Schoen, and R.G. Smith, "Impulse: a display oriented editor for strobe," *AAAI-83,* pp. 356-358, 1983.

[5] R.G. Smith, "Strobe: support for structured oriented knowledge representation," *IJCAI-83,* pp. 855-858, 1983.

[6] *GURU Reference Manual,* Micro Data base system, Inc., P.O. Box 248, Lafayelle, Indiana 47902, U.S.A., 1985.

[7] J.F. Brule, and F.A. Blount, "Problems of knowledge acquisition from human experts: A psychological perspective," *The Second Annual Artificial Intelligence & Advanced Computer Technology Conference,* pp. 62-68, 1986.

[8] A. Hart, *Knowledge Acquisition for Expert System,* Kogan Page Ltd., 120 Pentonville Road, London N19JN, 1986.

[9] K. Parsaye, and S. Murphree, *Automating the Knowledge Acquisition Process,* Intelligence Ware, Inc., 9800 S. Sepulreda Blrd. Suite 730, LA, LA 90045, U.S.A., 1987.

[10] S. Marcus, J. McDermott, and T. Wang, "Knowledge acquisition for construction systems,'. *IJCAI-85,* pp. 637-639, 1985.

[11] T.A. Wguyen "Verifying consistency of production systems," *IEEE Third Annual Expert Systems in Government Conference,* 1987.

[12] A. Ginsberg, "A new approach to checking knowledge bases for inconsistency and redundancy," *IEEE Third Annual Expert Systems in Government Conference,* 1987.

[13] W.V. Melle, E.H. Shortliffe, and B.G. Buchanan, "EMYCIN: A knowledge engineer's Tool for Constructing Rule-Based Expert Systems," in B. Buchanan and E. Shortliffe (eds.), *Rule-Based Expert Systems,* Addison-Wesley Publishing Company, U.S.A., 1984.

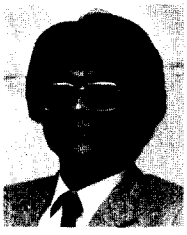
[14] G.S. Kahn, A. Kepner, and J. Pepper, "TEST: A model-driven Application shell," *AAAI-87, pp.* 814-818, 1987.

[15] L. Brownston, R. Farrell, and E. Kant, *Programming Expert Systems in OPS5,* Addison-Wesley Company, 1985.

[16] A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques, and Tools,* Addison-Wesley Company, 1986.

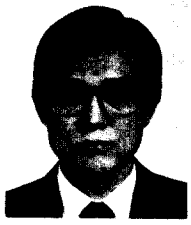
[17] *GCLISPLM reference manual,* God Hill Computers, Inc., Cambridge, Massachusetts, U.S.A., 1986.

著 者 紹 介



金 在 熹(正會員)

1979年 2月 연세대학교 전자공학과 졸업. 1982年 8月 Case Western Reserve University 석사 1984年 5月 Case Western Reserve University 박사. 1978~80年 금성사 중앙연구소. 1984 ~ 현재 연세대학교 전자공학과 조교수. 주관심분야는 인공지능, 패턴인식, Intelligent Tutoring System 등임.



申 東 弼(正會員)

1945年 5月 3日生. 1972年 2月 국민대학교 법학과 졸업. 1983年 5月 미국 Utah State University대학원 졸업 Computer Science 석사학위 취득. 1986年 5月 미국 University of Oklahoma 대학원 졸업 Computer Science 박사학위 취득. 1972年 ~1990年 현재 KIST 시스템공학 센터 인공지능 연구부장. 1987年 1月~1987年 11月 한국전산원 연구위원. 주관심분야는 Inference Machine, Parallel Computer Architecture, Pattern Recognition, Expert System 등임.