Journal of the Korean
OR/MS Society
Vol. 14, No. 2, Dec., 1989

지식베이스 구축을 위한 지식정의 언어와 지식생성

김창화* • 백두권**

The Knowledge Definition Language and Knowledge Creation for Knowledge Base Construction

Chang-Hwa Kim* and Doo-Kwon Baik**

Abstract

REA(Restricted Entity Aspect) model is a knowledge representation model to classify the aspect type, the EA model component, into five aspects(IS-A aspect, A-PART-OF aspect, attribute aspect, role aspect, and operation aspect).

EATPS, the knolwedge representation system, consists of user interface module, knowledge creation module, instance management module, schema management module, and integrity checking module.

EATPS creates and manages interactively REA model based knowledge base.

This paper shows the structure and functions of EATPS, the design and interactive construction of the knowledge definition language EAKDL, the functions and algorithm of class creation module, and the functions and algorithm of instance creation module to include inheritance inference mechanism.

요 약

REA 모델은 EA 모델의 구성요소인 측면을 역할측면, 연산측면, A-PART-OF 측면, IS-A 측면, 속성측면 등의 5개 측면으로 분류한 모델이다. EATPS는 사용자 인터페이스

* 강릉대학교 전자계산학과 전임강사
** 고려대학교 전산과학과 교수

모듈, 지식생성 모듈, 인스턴스 관리 모듈, 스키마 구조관리 모듈, 무결성검사 모듈등의 5개 모듈로 구성된 지능적 지식표현 시스템으로 REA 모델에 의해 대화식(interactive)으로 생성하고 관리한다.

본 논문에서는 EATPS의 구성과 기능, 지식 정의언어인 EAKDL의 구조 및 대화식 형성, 지식생성 모듈의 구성과 기능, 클래스 생성 모듈의 기능과 알고리즘, 그리고 상속추론 메카니즘을 도입한 인스턴스 생성 모듈의 기능 및 알고리즘을 제시하였다.

1. 서 론

EA(Entity Aspect) 모델은 시스템 개체구조(System Entity Structure)의 개념[9, 11, 17]을 도입한 지식표현을 위한 프레임 구조의 모델로서 개체, 측면, generalization, aggregation, multiplicity 등의 구성요소와 개념들로 구성된다[20, 22, 23, 24]. 개체란 실 세계로부터 구별되는 실제적 또는 추상적 대상체를 말하고, 측면이란 개체가 관찰되고 표현되는 관점을 말하며, 실세계의 개체는 일반적으로 다차원적 측면을 갖는다.

Generalization 개념은 IS-A 관계를 나타내는 추상화 개념으로 공통성질을 갖는 서브클래스들을 하나의 클래스로 통합시키는 추상화 개념이며, aggregation 개념은 A-PART-OF 관계로서 여러개의 구성 개체를 통합하여 새로운 조합된 개념을 창출하는 추상화 개념이다[9, 12].

이 모델은 실세계의 개체를 다양한 측면에서

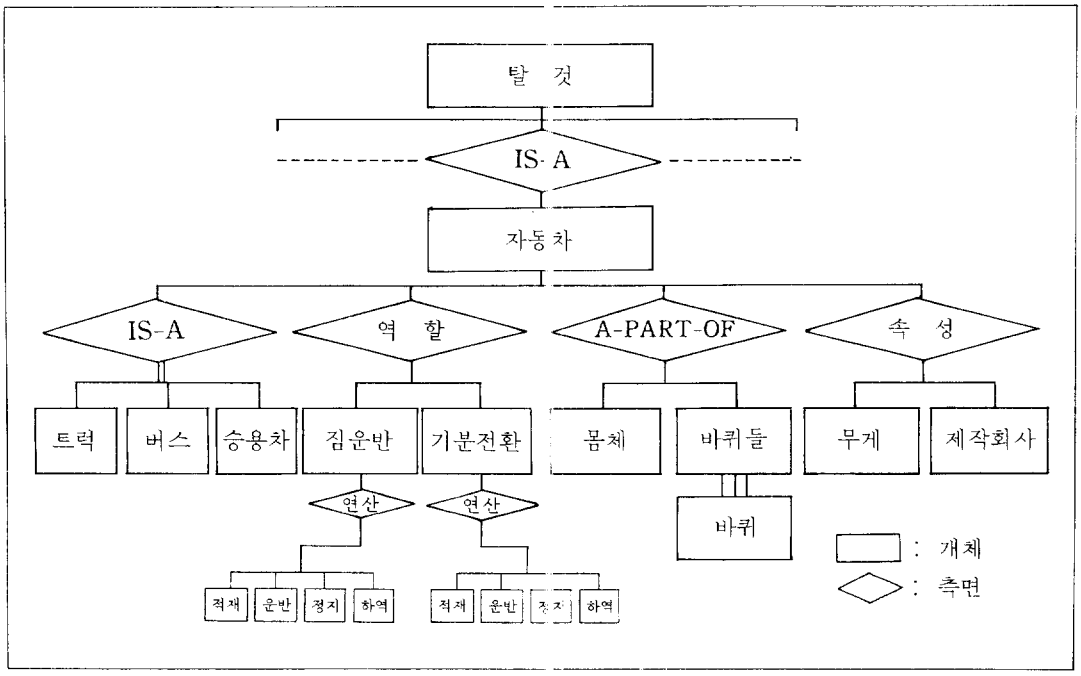

그림 1. '자동차'에 대한 REA 모델의 예

나타낼 수 있고, 개체에 대한 세부적이고 계층적인 지식구조를 잘 나타낼 수 있는 장점을 지닌다[20, 22, 23, 24]. 그러나, 이 EA 모델은 대부분의 모델과 마찬가지로 개체와 측면, 그리고 속성 구분의 불명확성으로 인하여, 1) 개체 타입과 측면 타입간의 모순발생, 2) 시스템 구현의 복잡성, 3) end-user가 지식정보를 추출하는데 대한 복잡성, 4) 무결성 유지의 어려움 등의 문제점들이 발생한다[18, 20]. 이러한 문제해결을 위하여 EA 모델의 측면을 개체 종류의 분석을 통하여 역할측면, 연산측면, A-PART-OF 측면, IS-A 측면 및 속성측면 등의 5개 측면으로 분류한 REA 모델(Restricted EA 모델) 개념이 제시되었다[18]. 그 예로서 그림 1은 '자동차'에 대한 REA 모델을 나타낸다.

EATPS(EA model To Prolog Knowledge base System)는 REA 모델에 의한 지식베이스를 형성하기 위한 지식표현 시스템으로, 1) 상속추론 메카니즘을 도입함으로써 자동 상속을 통한 REA 모델 지식베이스의 대화식 생성, 2) REA 모델 지식베이스의 대화식 관리, 3) 스키마 구조의 동적(dynamic) 변경, 4) 스키마와 지식베이스의 변경시 사용자와의 대화를 통한 무결성의 자동유지 등의 기능을 지원한다.

EATPS는 사용자 인터페이스 모듈, 지식생성 모듈, 인스턴스관리 모듈, 스키마 구조관리 모듈, 무결성 유지 모듈 등의 5개 모듈로 구성된다.

지식생성 모듈은 지식정의 언어인 EA-KDL(EA model Knowledge Definition Language)의 구조를 사용자와 대화식으로 형성하면서 REA 모델 기반의 prolog 지식베이스로 변환한다.

인스턴스관리 모듈은 인스턴스에 관련된 정보를 변경하는 기능을 담당하며, 사용자와 대화식으로 무결성을 유지한다.

스키마 구조관리 모듈은 REA 모델 스키마 구조를 동적(dynamic)으로 변경시키는 기능을 담당하며, 무결성 검사 모듈은 지식생성, 인스턴스 관리, 스키마 구조의 변경시 무결성을 자동검사하는 기능을 담당한다.

본 논문에서는 EATPS의 구성과 기능, EAKDL 구조, 지식생성 모듈의 구성과 기능 및 알고리즘에 대하여 소개한다.

2. 지식표현 시스템의 기능과 EATPS의 구성

Frost[4]가 소개한 지식베이스 시스템은 사용자와 KBMS(Knowledge Base Management System), 지식베이스의 3가지 요소들로 구성된다.

사용자는 질의시에 자연어 인터페이스와 접하게 되며 사용자 질의는 인트프리터에 의해 번역되어 접근전략(access strategy)이 형성되어 추론엔진에 의해 수행된다. 지식베이스는 단순 사실(simple fact)들과 일반규칙(general rule)들로 구성되며, 규칙의 호출 및 적용은 추론엔진이 담당하게 된다. KBMS가 지식베이스에 접근하거나 조작할 때에 무결성유지 모듈(integrity maintenance module)에 의해 지식베이스의 무결성이 항상 자동적으로 검사된다.

사용자가 지식베이스를 설계할 때에는 외부스키마 인터페이스와 접하게 되며, 사용자가 정의한 지식구조의 무결성이 자동적으로 검사되면서 KBMS에 의해 지식구조가 형성된다.

이 개념을 기반으로 지식표현 스템은, 1) 지식구조, 지식들간의 관계 및 내용을 표현하기 위한 지식표현 기능, 2) 실세계의 지식과 일관성 유지

를 위해 저장된 지식을 검색, 변경, 삽입, 삭제하기 위한 지식조작 지원기능, 3) 지식 접근을 위한 질의처리 기능, 4) 지식조작, 지식창출, 지식표현을 쉽게 할수 있도록 하는 환경을 시스템이 자동적으로 제공하기 위한 추론기능, 5) 스키마 구조의 변경, 확장 등을 위한 지식베이스의 확장 지원기능, 6) 무결성의 자동검사 및 유지기능 등의 기능들을 제공해야 한다.

앞에서 언급한 지식표현 시스템의 기능을 중심으로 EATPS(EA model To Prolog Knowledge base System)을 구성해 보자.

EATPS는 그림 2에서와 같이 사용자 인터페이스 모듈, 지식생성 모듈, 인스턴스 관리 모듈, 스키마 관리 모듈, 무결성 검사 모듈 등의 5개 모듈로 구성된다. 사용자 인터페이스 모듈은 사용자에게 메뉴중심(menu-driven)으로 시스템 기능을 지원하는 역할을 담당한다.

지식생성 모듈은 EAKDL(EA model Knowledge Definition Language)로 정의된 스키마를 번역하거나, 혹은 사용자가 대화식으로 지식구조를 정의할 수 있도록 안내하고 정의된 지식구조를 prolog 지식베이스로 변경하는 기능을 담당한다.

인스턴스 관리 모듈은 지식베이스에 지정된 인스턴스들을 삽입, 삭제, 변경, 검색하는 기능을 담당하고, 스키마 관리 모듈은 스키마의 구조를 동적(dynamic)으로 변경하는 기능을 담당한다. 또한, 무결성 유지 모듈은 무결성 검사를 담당하게 된다.

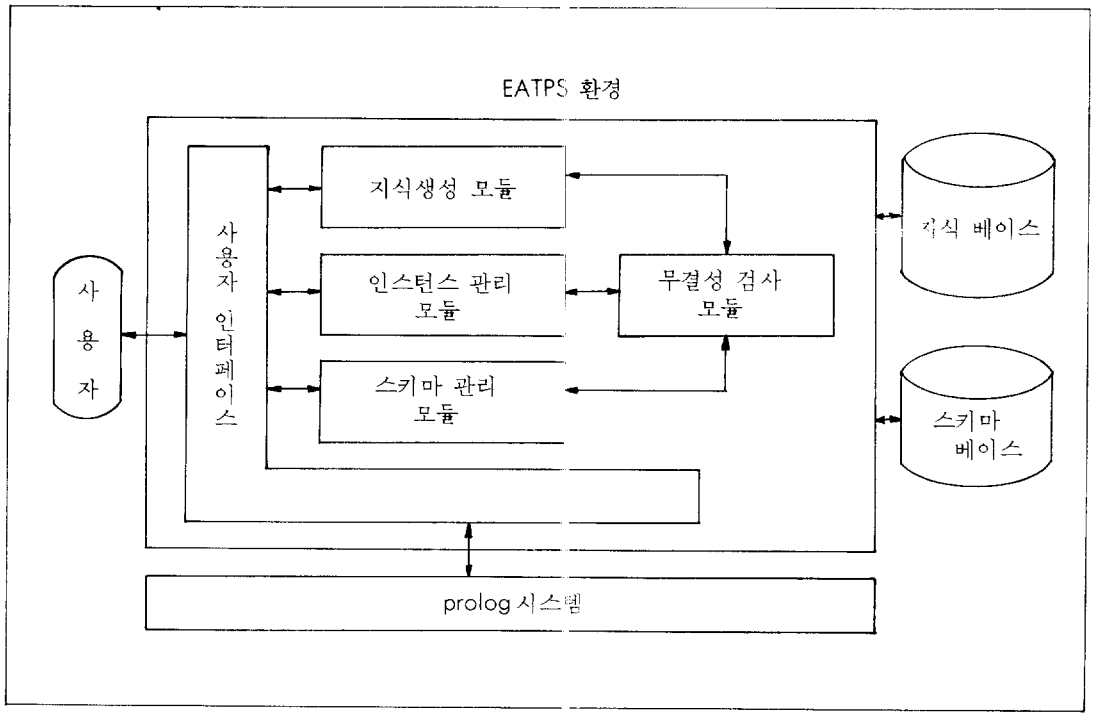

그림 2. EATPS의 구성 및 환경

3. 지식 정의언어의 구조 및 지식베이스의 생성

3.1 지식 정의언어의 구조

EAKDL(EA model Knowledge Definition Language)은 REA 모델 지식베이스 구조를 표현하는 언어로서 클래스 정의구조와 인스턴스 정의구조로 나뉜다. 클래스 정의구조를 통하여 클래스의 is-a 측면, a-part-of 측면, 속성측면, 연산측면 및 역할측면에 관련된 정보를 표현함으로써 클래스의 내부정보와 클래스들간의 계층관계 등을 나타냄으로써 지식구조가 형성되고 표현된다. 인스턴스 정의구조는 인스턴스와 클래스의 관계, 또는 인스턴스들간의 관계를 나타내며, 정의되는 인스턴스는 소속 클래스 및 상위 클래스로부터 is-a 측면, a-part-of 측면, 속성측면, 역할측면 및 연산측면에 관련된 정보와 구조를 상속받게 된다.

클래스 정보를 표현하는 클래스 정의구조는 다음과 같다.

[클래스의 EAKDL 정의구조]

```
CLASS: 클래스명
  IS_A: 상위 클래스
  A_PART_OF: 통합 클래스
  ATTRIBUTES:
    NAME  : 속성명_1
    VALUE : 속성값_1
      .....
      .....
    NAME  : 속성명_n
    VALUE: 속성값_n
  ROLE:
    role_1:-op_11, op_12,…,op_11
      .....
      .....
    role_m:-op_m1, op_m2,…,op_mk
ENDCLASS
```

위의 구조에서 IS_A, A_PART_OF ATTRIBUTES, ROLE 등은 각각 측면을 나타내고, role_i는 클래스의 역할을 나타내며, op_i1, …, op_ik는 각각 클래스의 역할 role_i를 수행하기 위한 연산을 나타낸다.

여기에서 CLASS, IS-A, ATTRIBUTES, NAME, VALUE, ROLE 등은 지식생성 모듈에 의해 사용자에게 제시되며, 클래스명, 상위 클래스, 통합 클래스, 속성명_i, 속성값_i, role_i :-op_i1, …, op_ik 등은 사용자가 입력하게 된다. 이에 대한 예로서 그림 1의 '자동차'에 대한 EAKDL 구조는 그림 3과 같다. 클래스 '트럭'과 '버스', '자동차'는 각각 클래스 '자동차'와 IS_A 측면 관계에 있으므로 '자동차'의 역할, 속성, A_PART_OF 측면의 모든 성질들은 상속성(inheritance)에 의해 '트럭', '버스', '자동차'로 각각 상속된다. 그림 3에 대한 EAKDL 구조의 상속관계는 그림 4와 같다.

한편, 인스턴스의 EAKDL 정의구조는 다음과 같다.

[인스턴스의 EAKDL 정의구조]

```
INSTANCE: 인스턴스명
  IS_A: 소속 클래스명
  A_PART_OF: /* 소속 클래스나 혹은 상위
    클래스로부터 상속되며, 이에 대하여 사용자
    가 입력하는 값은 이 인스턴스의 소속 클래스
```

```
CLASS: 탈것
  IS_A: none
  .....
ENDCLASS
CLASS: 자동차
  IS_A: 탈것
  A_PART_OF: none
  ATTRIBUTES:
     NAME  : 무게
     VALUE : _

     NAME  : 제작회사
     VALUE : _

     ROLE:
       짐운반  : _ 적재, 운반, 정지, 하역
       기분전환: _ 적재, 운반, 정지, 하역

ENDCLASS

CLASS: 트럭
  IS_A:  자동차
ENDCLASS /* A_PART_OF, ATTRIBUTES, OPERATION 측면은
            상위 클래스로부터 상속됨 */

CLASS: 버스
  IS_A: 자동차
ENDCLASS /* A_PART_OF, ATTRIBUTES, OPERATION 측면은
            상위 클래스로부터 상속됨 */

CLASS: 승용차
  IS_A: 자동차
ENDCLASS /* A_PART_OF, ATTRIBUTES, OPERATION 측면은
            상위 클래스로부터 상속됨 */
```

그림 3. '자동차'에 대한 EAKDL 구조

('몸체'와 '바퀴들'의 EAKDL 구조는 생략)

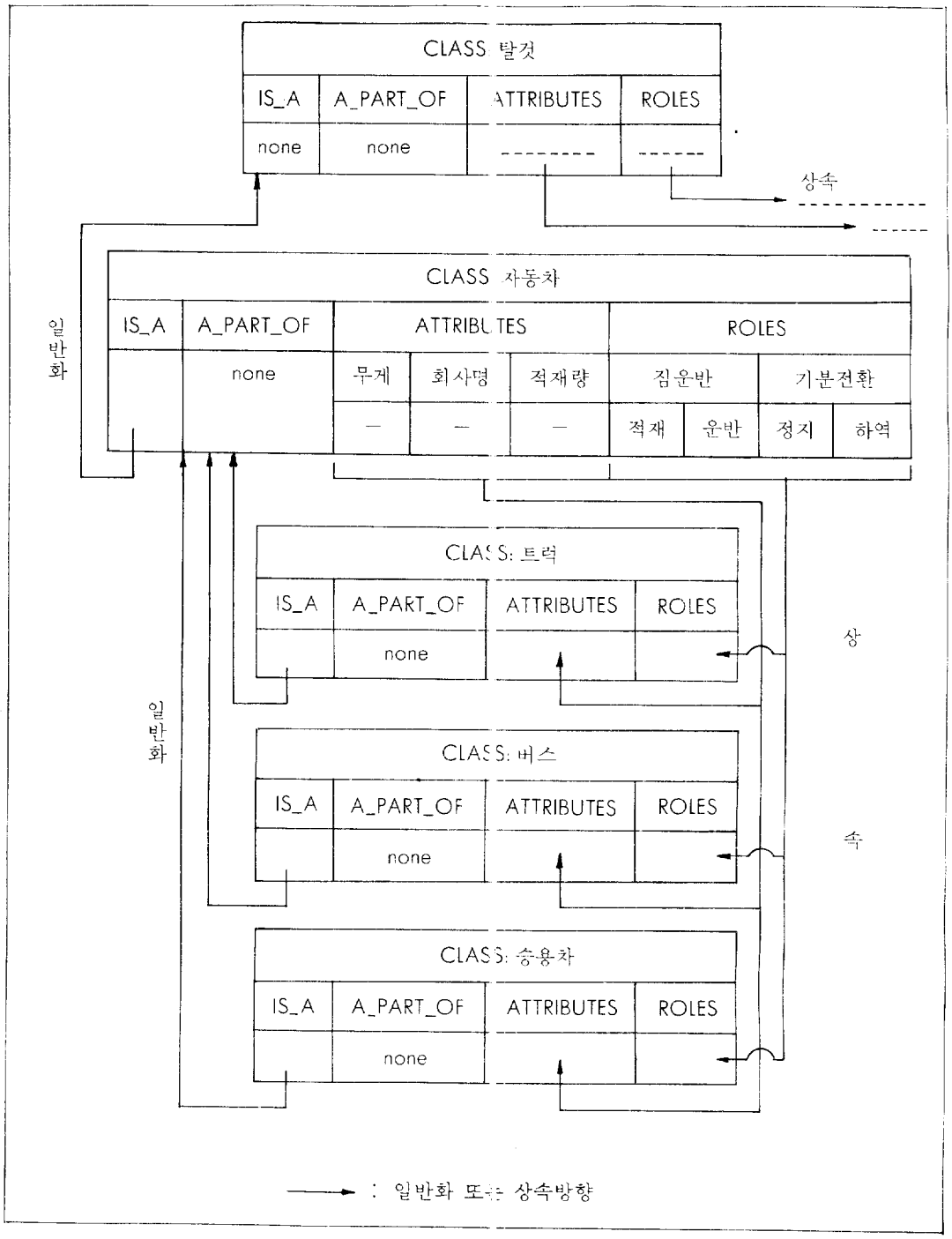

그림 4. '자동차'의 EAKDL 구조에 대한 일반화 및 상속관계

```
    나 혹은 상위 클래스가 구성하는 aggrega-
    tion 클래스의 인스턴스가 된다 */
  ATTRIBUTES:
  /* 소속 클래스 및 상위 클래스로부터 상속된
    속성명 및 사용자가 제시한 속성값 */
ENDINSTANCE
```

여기에서 INSTANCE 와 A_PART_OF 구조 및 IS-A 구조는 지식생성 모듈에 의해 사용자에게 제시되며, 인스턴스명과 소속 클래스명은 REA 모델구조에 따라 사용자가 정의한다. 인스턴스의 속성명 및 속성값과 A_PART_OF 측면 및 ROLE 측면은 소속 클래스 및 상위 클래스들로부터 지식생성 모듈의 상속 메카니즘에 의해 자동적으로 추론된다. 특히 속성명의 경우에는 지정하지 않더라도 소속 클래스 및 상위 클래스들의 속성들이 상속 메카니즘에 의해 자동 추론되므로, 사용자는 속성값만 제시하면 된다.

한편, EAKDL에 대한 BNF 표기법은 그림 5와 같다.

이 표기법에서 보면 클래스명, 상위 클래스명, 속성명 및 속성값, 통합 클래스명, 역할명 및 연산명은 모두 스트링으로 처리한다.

앞에서 제시한 EAKDL 구조는 3.2절에 제시된 변환 메카니즘에 의해 다시 REA 모델구조의 prolog 지식베이스로 변환된다.

```
<knowledge-base-schema> : : = <classes-schema><instances-schema>
<classes-schema> : : = <class><class-schema> | <class>
<instances-schema> : : = <instance><instances-schema> | <instance>
<class> : : = CLASS : <string>
                <is-a-clause>
                <a-part-of-clause>
                <attributes-clause>
                <role-operation-clause>
              ENDCLASS
<is-a-clause> : : = IS-A : <string> |ε
<a-part-of-clause> : : = A_PART_OF : <string>
                        |ε
<attributes-clause> : : = ATTRIBUTES :
                          <name-value-clauses>
                        |ε
<name-value-clauses> : : = NAME : <string>
                           VALUE : <string>
                           <name-value-list>
<name-value-list> : : = <name-value-clauses>
                      |ε
```

```
<role-operation-clause> : : = ROLE :
                                <role-op-clause>
                              |ε
<role-op-clauses> : : = <string> : - <string-list> ;
                        <role-op-list>
<role-op-list> : : = <role-op-clause>
                   |ε
<string-list> : : = <string><comma-string>
 comma-string> : : = ,<string><comma-string>
                    |ε
<instance> : : = INSTANCE : <string>
                 <is-a-clause>
                 <a-part-of-clause>
                 <attributes-clause>
                ENDCLASS
<string> : : = <character><string>
               <character>
<character> : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
                  a | b | c | d | e | f | g | h | i | j | k | l | m |
                  n | o | p | q | r | s | t | u | v | w | x | y | z |
                  blank | - | _ | * | / | =| < | > | $ | ₩
```

그림 5. EAKDL의 Syntax에 대한 BNF

3.2 prolog 지식베이스로의 변환

3.1절에서 제시한 EAKDL 정의구조는 지식베이스의 관리와 추론을 쉽게 하기 위해 그림 6의 변환 메카니즘에 의해 prolog의 clause 패턴으로 변환된다.

변환된 class(클래스) 형태의 clause들은 CLASS 부분 지식베이스를 형성하며, is_a(클래스, 상위 클래스) 형태의 clause들은 IS_A 부분 지식베이스를 형성한다. 또한 a_part_of 형태의 clause들은 A_PART_OF 부분지식베이스를 형성하고, attri(클래스, 속성명, 속성값) 형태의 clause들은 ATTRI 부분 지식베이스를 형성하며, role(클래스, 역할) 형태의 clause들은 ROLE 부분 지식베이스를 형성한다.

한편, operation(클래스, 역할, 연산) 형태의 clause들은 OPERATION 부분 지식베이스를 형성하고 instance(인스턴스명) 형태의 clause들은 INSTANCE 부분 지식베이스를 형성한다. 인스턴스의 속성에 대한 정보는 알고리즘 2에 도입된 상속추론 메카니즘에 의해 소속 클래스 및 상위 클래스들로부터 자동 상속되며, attri(인스턴스, 속성명, 속성값) 형태로 변환된

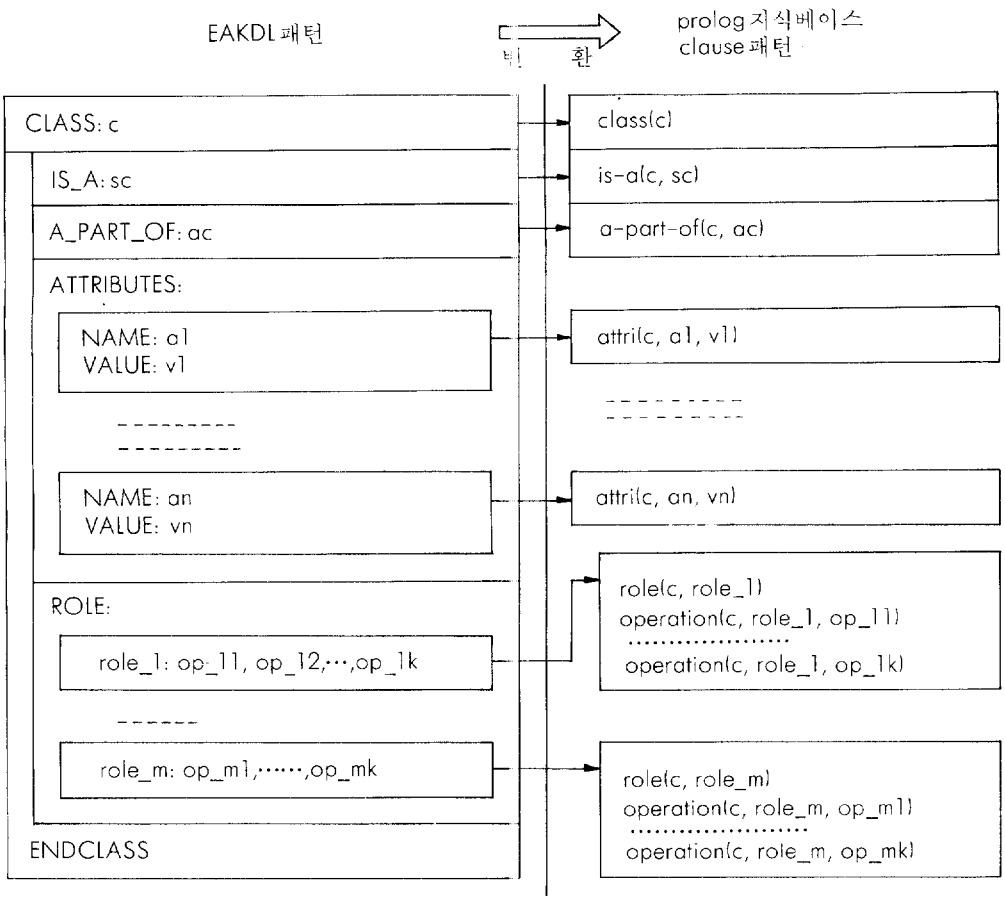

그림 6. EAKDL 구조를 prolog clause 패턴으로 변환하는 메카니즘

어 ATTRI 부분 지식베이스로 저장된다. 마찬가지로 인스턴스의 A_PART_OF 측면에 대한 정보도 자동 상속되어 사용자가 제시한 I-1, I-2,···, I-n의 형태가 a_part_of(인스턴스, I-1), a_part_of(인스턴스, I-2),···, a_part_of(인스턴스, I-n) 형태로 변환되어 A_PART_OF 부분 지식베이스로 저장된다.

4. 지식생성 모듈의 구조와 기능

사용자가 지식생성 모듈과 대화식으로 지식베이스를 형성해 나가는 과정은 그림 7과 같다. 사용자는 시스템이 제시하는 지식 정의구조인 EAKDL 구조에 따라 클래스 정보(클래스명, 상위 클래스, 속성명 및 속성값, 역할 및 연산 등)를 입력하면 시스템은 스키마 베이스를 형성

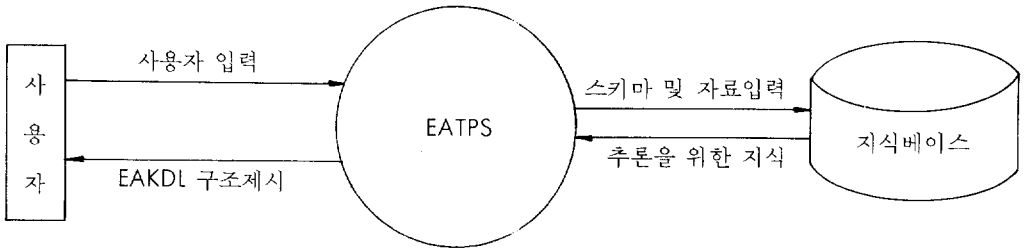

그림 7. EATPS의 지식생성 모듈을 통한 REA 모델 지식베이스의 대화식 생성

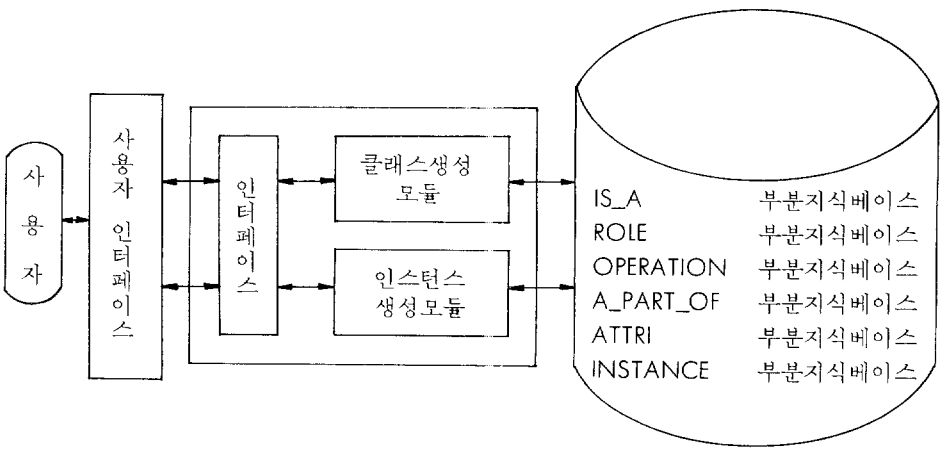

그림 8. 지식생성 모듈의 구성

하고, EAKDL 구조를 REA 모델 지식베이스로 변환함으로써 클래스들간의 REA 모델 구조를 형성한다. 이와 같은 REA 모델 구조에 상속(inheritance) 메카니즘을 이용하여 인스턴스를 생성함으로써 클래스들과 인스턴스들로 이루어진 지식베이스가 REA 모델구조로 형성된다.

지식생성 모듈은 그림 8과 같이 인터페이스, 클래스 생성 모듈, 인스턴스 생성 모듈들로 구성된다.

인스페이스는 사용자에게 지식베이스 생성을 위한 기능을 지원하며, 클래스 생성 모듈은 사용자와 대화식으로 형성된 EAKDL 정의구조를 부분 지식베이스들로 변환함으로써 prolog 지식베이스로 구조화 시킨다. 이 모듈에 의해 각 클래스의 구조 및 클래스들의 관계가 지식베이스로 형성된다. 이 모듈에 의한 사용자와의 대화식 EAKDL 형성 및 지식베이스로의 변환과정은 알고리즘 1과 같다.

알고리즘 1: 대화식 클래스 생성 모듈

```
REPEAT
1. WRITE "CLASS:" TO 스크린
   READ 클래스 CN FROM 키보드
   INSERT clause class(CN) INTO CLASS 부분지식베이스
2. WRITE "IS-A:" TO 스크린
   READ 상위 클래스 SC FROM 키보드
   INSERT clause is_a(CN, SC) INTO IS-A 부분지식베이스
3. WRITE "A_PART_OF" TO 스크린
   READ 통합 클래스 AC FROM 키보드
   INSERT clause a_part_of(C, AC) INTO A_PART_OF 부분지식베이스
4. WRITE "ATTRIBUTES:" TO 스크린
   READ Continue FROM 키보드
   /* 변수 Continue는 사용자가 속성명을 계속 제시할
      것인가의 여부를 나타내는 변수로서,
       Continue < > "Q": 속성명 계속 제시
       Continue = "Q": 속성명 제시 중지
      를 나타낸다 */
   WHILE Continue <> "Q" DO
     WRITE "NAME:" TO 스크린
     READ 속성명 AN FROM 키보드
     WRITE "VALUE:" TO 스크린
     READ 속성값 AN FROM 키보드
     INSERT clause attri(CN, AN, AV) INTO ATTRI 부분지식베이스
   DOWHILE
5. WRITE "ROLE:" TO 스크린
   READ Continue FROM 키보드
   WHILE Continue <> "Q" DO
     READ 스트링 "role-i: op-i1, op-i2,…,op-in"
          FROM 키보드
     INSERT role(CN, role-i) INTO ROLE 부분지식베이스
     FOR j=1 TO n DO
       INSERT operation(opij) INTO OPERATION 부분지식베이스
     DOFOR
     READ Continue FROM 키보드
   DOWHILE
6. WRITE "END CLASS" TO 스크린
UNTIL 클래스 생성완료
RETURN
```

인스턴스 생성 모듈은 사용자와 대화식으로 인스턴스 EAKDL 구조를 생성하면서 지식베이스를 형성한다. 인스턴스의 속성들은 인스턴스가 소속된 클래스 및 상위 클래스들로부터 상속 추론 메카니즘을 도입한 알고리즘 2에 의해 자동 추론되며, 사용자는 이 모듈에 의해 제시되는 속성명에 대한 속성값을 입력함으로써 지식베이스로 변환된다.

알고리즘 2: 인스턴스 생성 모듈

```
REPEAT
1. WRITE "INSTANCE:" TO 스크린
   READ 인스턴스 IN FROM 키보드
   INSERT clause instance(IN) INTO INSTANCE 부분지식베이스
2. WRITE "IS_A:" TO 스크린
   READ 소속 클래스 IC FROM 키보드
   INSERT clause is_a(IN, IC) INTO IS-A 부분지식베이스
3. /* 생성되는 인스턴스가 다른 인스턴스의 구성요소인지를
        추론을 통하여 검사하고 구성요소일 경우에는
          "A_PART_OF:" 패턴을 제시하여 사용자가
          통합 인스턴스를 입력하도록 한다. */
   IF∃a_part_of(IC, AC) IN A-PART-OF 부분지식베이스
THEN
  ACCESS is_a(IC, Sup) FROM IS-A 부분지식베이스
  WHILE(∃ a_part_of(Sup, AC) IN A-PART-OF
   부분지식베이스) OR (Sup <> "none") DO
   IC:=Sup
   UNBINDING(Sup)/* 변수 Sup에서 변수값을 unbind */
   ACCESS is_a(IC, Sup) FROM IS-A 부분지식베이스
  DOWHILE
  IF∃a_part_of(Sup, AC) IN A-PART-OF 부분지식베이스
   THEN
     WRITE "A_PART_OF: " TO 스크린
     READ 스트링 "I1, I2,…,In" FROM 키보드
     /* I1, I2,…,In은 각각 인스턴스를 나타냄 */
     FOR I:=1 TO n DO
       INSERT a_part_of(IN, Ii) INTO
       A-PART-OF 부분지식베이스
     DOFOR
  ENDIF
ELSE
```

```
  WRITE "A-PART-OF: " TO 스크린
  READ 스트링 "I1, I2,…,In" FROM 키보드
   /* I1, I2,…,In은 각각 인스턴트를 나타냄 */
  FOR I:=1 TO n DO
   INSERT a_part_of(IN, Ii)
   A-PART-OF 부분지식베이스
   DOFOR
ENDIF
4. /* 속성에 대한 상속추론 메카니즘: 인스턴스가 소속된
      클래스 및 상위 클래스로부터 속성들을 추론한다 */
  ACCESS is_a(IC, Sup Class) FROM IS-A 부분지식베이스
  WRITE "ATTRIBUTES:" TO 스크린
  WHILE SupClass <> "none" DO
   WHILE∃clause attri(IC, AN, AV) IN ATTRI 부분지식베이스
    DO WRITE "NAME:", AN TO 스크린
    WRITE "VALUE:" TO 스크린
    READ AV' FROM 키보드
    INSERT clause attri(IN, AN, AV')
      INTO ATTRI 부분지식베이스
   DOWHILE
   IC:=SupClass
   ACCESS is_a(IC, Sup) FROM IS-A 부분지식베이스
   SupClass:=Sup
  DOWHILE
  WHILE∃clause atrri(C, AN, AV) IN ATTRI 부분지식베이스 DO
   WRITE "NAME :", AN TO 스크린
   WRITE "VALUE:" TO 스크린
   READ AV' FROM 키보드
   INSERT attri(IN, AN, AV') INTO ATTRI 부분지식베이스
  DOWHILE
  WRITE "END OF 인스턴스" TO 스크린
UNTIL 인스턴스 생성완료
RETURN
```

5. 결 론

본 연구는 REA 모델 지식베이스를 형성하기 위한 지식 정의언어인 EAKDL의 구조와, 지식생성 모듈의 구성요소인 클래스 생성 모듈과 인스턴스 생성 모듈의 기능과 알고리즘에 관

하여 제시하였다.

지식 정의언어인 EAKDL은 클래스 정의구조와 인스턴스 정의구조로 나뉘며, 클래스 정의구조는 클래스에 대한 정보와 구조 및 클래스들 사이의 관계를 나타내고, 인스턴스 정의구조는 인스턴스에 대한 정보 및 인스턴스와 클래스와의 관계, 그리고 인스턴스들 사이의 관계를 나타내는 언어 패턴이다.

지식생성 모듈은 인터페이스, 클래스 생성 모듈, 그리고 인스턴스 생성 모듈로 구성된다. 클래스 생성 모듈은 클래스에 대한 EAKDL 구조를 사용자와 대화식으로 형성하면서 prolog REA 모델 지식베이스로 변환하며, 인스턴스 생성 모듈은 클래스 생성 모듈에 의해 생성된 클래스 구조를 중심으로 상속추론 메카니즘에 의해 인스턴스에 대한 EAKDL 구조를 형성하면서 prolog REA 모델 지식베이스로 변환하는 기능을 갖는다.

EATPS의 각 모듈은 PC XT, AT/MS-DOS 환경하에서 Turbo Prolog Ver 2.0을 이용하여 원시형태(prototype)로 구현되었다.

참고문헌

[1] Arte, Data Base: Structured Techniques for Design, Performance, and Management, Wiley-Interscience, 1980.

[2] Blaha, M.R., Premerlani, W.J. and Rumbaugh, J.E., "Relational Database Desing Using an Object-Oriented Methodology", ACM Comm., Vol. 31, No. 4, 1988.

[3] Broie, L., and Mylopoulos, J. and Schmidt, J.W.,. On Conceptual Modelling, Springerverlag New York Inc., 1986.

[4] Frost, R.A., Introduction to Knowledge Base Systems, Collins Professional and Technical Books, 1986.

[5] Kim, M.J., "A Relational Database Design Method Based on the Transformation of the Entity-Relationship Model", '87 동계 데이타베이스 학술세미나 논문집, Vol. 1, No. 1, 1987.

[6] Pressman, R.S., Software Engineering: A Practitioner's Approach, McGraw-Hill Inc., 1982.

[7] Rozenblit, J.W., "A Conceptual Basis for Integrated Model-Based System Design", Dep. Elec. Compt. Eng., Univ. Arizona, Tusson, Az, Tech. Rep., 1986.

[8] Rozenblit, J.W., Sevinc, S. and Zeigler, B.P., "Knowlege-Based Design of LANs Using System Entity Structure Concepts", Preceeding of the 1986 Winter Simulation Conference, 1986.

[9] Scherefl, M. and Neuhold, E.J., "Object Class Definition by Generalization Using Upward Inheritance", Proceedings Fourth International Conference on Data Engineering, Computer Society Press, 1988.

[10] Schubert, L.K., "Extending the Expressive Power of Semantic Networks", Artificial Intelligence 7, North-holland Pub. Com., 1976, pp.163-198.

[11] Sevince, S., and Zeigler, B.P., "Entity Structure Based Design Methodology: A Lan Protocol Example", IEEE Trans. Soft., Vol. 14, No. 3, 1988.

[12] Tsichritzis, D.C. and Lochovsky, F. M., Data Model, Prentice-Hall, 1982.

[13] Theory, T.J., Yang, T., and FRY, J. P., "A Logical Design Methodology for Relational Database Using the Extended Entity-Relationship Model", Computing Surveys, Vol. 18, No. 2, 1986.

[14] Waterman, D.A., "Generalization Learning Techniques for Automating the Learning of Heuristics", Artificial Intelligence 1, American Elsevier Pub. Com. Inc., 1970, pp.121-170.

[15] Zeigler, B.P., "Hierarchical, Modular Discrete Event Modelling in an Object-Oriented Environment", Simulation, 1987.

[16] Zeigler, B.P., "Hierarchical, Modular Modelling/Knowledge Representation", Preceedings of the 1986 Winter Simulation Conference, 1986.

[17] Zeigler, B.P., Multifacetted Modelling and Discrete Event Simulation, Academic Press, 1984.

[18] Zeigler, B.P., Theory of Modelling and Simulation, A Wiley-Interscience Pub., 1976.

[19] 김창화, 백두권, "지식표현을 위한 EA 모델의 확장", 한국정보과학회, '89 봄 학술발표 논문집, Vol. 16, No. 1, pp.119-122.

[20] 김창화, 백두권, 황종선, "데이타 베이스 설계를 위한 E-A(Entity-Espect) 모델에 관한 연구", 한국정보과학회, '86 봄 학술발표 논문집, Vol. 14, Vol. 1, 1987.

[21] 김성훈, "EA 모델을 이용한 데이타 베이스 스키마 통합 방법론에 관한 연구", 고려대학교, 석사학위 논문, 1988.

[22] 신대식, "EA 모델을 이용한 지식표현 시스템에 관한 연구", 충북대학교, 석사학위 논문, 1988.

[23] 신대식, 조용환, 백두권, "E-A(Entity-Aspect) 모델에 의한 지식표현 시스템에 관한 연구, 한국정보과학회, '87 가을 학술발표 논문집, Vol. 14, No. 2, 1987.

[24] 신대식, 조용환, 백두권, "지식표현 시스템의 설계 및 구현", 한국통신학회, 1987 년도 추계 학술발표회 논문집, 1987.