□ 특별기고 □

개념그래프 소개

목포대학교 양 기 철*

목 차

1. 서 론

자연언어가 컴퓨터에 의해서 처리되기에는 비효율적인 언어이지만 인간은 서로간의 의사소통을 위하여 자연언어를 사용하고 일상생활에서 대부분의 일들을 자연언어를 사용하여 처리하고 있으며, 인간에 의해 축적된 많은 지식들이 자연언어에 의해 기술되고 보존된다. 또한 컴퓨터를 사용하여 해결하고자 하는 많은 문제들이 인간으로부터 발생되고 자연언어로 기술되기 때문에 우리 인간을 위한 많은 문제들을 컴퓨터를 이용하여 쉽게 해결하려면 컴퓨터로 하여금 자연언어를 이해하도록 하고 자료(Data)뿐 만 아니라 지식(Knowledge)까지도 처리할 수 있도록 만들어야 한다.

근대 철학자이며 논리학자인 Peirce는 사람이 계산을 할 때 기호에 의한 것보다 도형에 의한 것이 더 효과적이라고 생각하여 논리를 도식으로 표현하는 존재그래프(Existential Graph)라는 도식언어를 창안하였다[7]. 도식언어는 논리언어(eg. Predicate Calaulus)보다 사람이 읽기에 쉽고 자연어보다 컴퓨터가 처리하기에 편리한 형태로 문장의 의미를 표현할 수 있다.

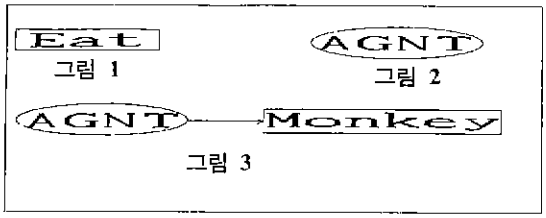
개념그래프(Conceptual Graph)는 플로우 챠트, 유한상태 기계, 또는 페트리 네트처럼 도형을 사용하여 의미를 전달하는 일종의 논리적인 도식언어(Graphical Language)로서 특정분야만을 위한 지식표현언어가 아닌 보다 더 일반적인 지식표현언어로 사용가능하다. 이 글에서는 개념그래프와 국내외 개념그래프 관련 연구 동향을 소개하므로써 독자들의 개념그래프에 대한 이해와 국내 개념그래프 관련연구의 활성화에 기여할 수 있기를 바라는 바이다.

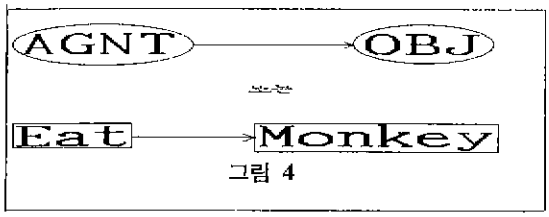
2. 개념그래프

개념그래프(Conceptual Graph)는 1984년 Sowa의 생각들이 책[9]으로 정리되어 나오면서 활발하게 연구가 진행된 일종의 의미망(Semantic Network)이다. 개념그래프는 노드들이 아크로 연결된 유한 이분 그래프로 표현되며 여기에는 두가지 형태의 노드가 있는데, 개념(concept)을 표현하는 개념노드와 개념노드 간의 관계(relation)를 표현하는 관계노드(node)가 그들이다.

개념노드는 그래프 상에서 사각형으로 표시되며, 관계노드는 원으로 표시된다. 하나의 개념노

*종신회원





드는 그 자체만으로 개념그래프를 형성하지만(그림 1), 관계노드는 그 자체만으로 개념그래프가 될 수 없다(그림 2). 따라서, 개념그래프 안에 있는 모든 관계노드는 개념노드와 연결되어 있는데 이 때 연결을 위하여 방향성을 갖는 아크(화살표)를 사용하며(그림 3), 개념노드와 관계노드 사이에서만 존재가 가능하다.

따라서, 같은 종류의 노드끼리 아크를 이용하여 연결되면, 이는 개념그래프가 아니다(그림 4).

2.1절에서 개념노드에 대해, 2.2절에서는 관계노드에 대해 설명하고 각개의 개념노드와 관계노드들 간의 관계설정을 위해 필요한 유형계층은 2.3절에서 설명한다.

2.1 개념노드

추상적 또는 사실적 실체(속성, 상태, 사건, 실체 등)를 나타내는 개념노드(Concept Node)는 그래프 상에서 사각형으로 표현되는데, 표기의 간결성을 위하여 그래프를 선형표현(Linear form)으로 표현할 때는 대괄호'[ ]'를 사용한다. 하나의 개념노드는 콜론':'에 의하여 두 부분으로 나뉘어 지는데 이때 콜론의 왼쪽 부분은 그 개념노드의 유형(Type)을 나타내고, 콜론의 오른쪽 부분에는 그 노드에 제약을 가하는 정보가 들어간다. 이 글 에서는 이 두 부분을 유형지대와 참고지대로 부르기로 한다(Sowa는 이 두 부분을

표 1 개념노드의 보기

No.	개 념 노 드	의 미
(1)	[PERSON]	사람
(2)	[PERSON : #123]	사람 #123(특정 개인)
(3)	[PERSON : John]	John이라는 이름의 사람
(4)	[PERSON : @1]	한 사람
(5)	[PERSON : {Tom, John}]	Tom과 John이라는 사람
(6)	[PERSON : {*}]	사람들
(7)	[PERSON : ?]	어느 사람
(8)	[PERSON : #]	그 사람

type field와 referent field라 불렀다).

예를 들어, [PERSON:John]이라는 개념노드는 유형이 PERSON이며 PERSON 중에서도 John이라는 개인을 다타낸다. 개념노드의 유형들은 계층적으로 조직되는데 이를 유형계층(type hierarchy)이라 한다. 하나의 유형계층에는 그 유형계층을 사용하는 시스템이 다루는 도메인(domain) 내의 모든 개념들의 유형이 속해 있다. 유형계층에 대해서는 2.3절에서 더 자세히 설명하기로 한다.

참고지대(referent field)는 여러 형태로 제약을 가하는 정보를 수록하는데 표 1에 그 예가 있다.

표 1에서 (1)은 일반적인 사람을 나타내며 [PERSON:*]와 같은 뜻이며 이 때 '*'는 총칭적 표시자(generic marker)이다. (2)는 개체표시자(individual marker)로 #기호 다음에 숫자를 사용하였다. (8)처럼 '#' 기호 다음에 숫자가 없으면 한정지시자(definite reference)로 쓰인 것이다. (4)는 양화사, (5)는 집합표시, (6)은 총칭적 복수표시자(generic plural markers)로 쓰였다.

이러한 참고지대의 형태들은 필요에 따라서 추가 또는 수정하여 쓸 수 있다. 또한 하나의 개념그래프 전체가 한 개념노드의 참고지대로 들어갈 수 있는데 그 예로 "John believes that Mary hit the ball"은 다음과 같은 개념그래프로 나타낼 수 있다.

[PERSON:John]←(EXPR)←[BELIEVE]-
(OBJ)→[PROPOSITION:[PERSON:Mary]-
(AGNT)←[HIT]→(OBJ)

→[BALL:#]].

기본적인 개념유형은 시스템 설계시 정의되는데 새로운 개념유형이 필요할 때에는 이를 정의하여 사용할 수 있다. 예를 들어 SMALL-PERSON이라는 유형을 PERSON의 하위유형으로 정의하려면

type SMALL-PERSON(X) is
[PERSON:*X]→(SIZE)→[SMALL]

형태로 정의할 수 있다. 이렇게 정의한 다음에는 [PERSON:*]→(SIZE)→[SMALL]이라는 패턴은 하나의 개념노드 [SMALL-PERSON]으로 대치하여 쓸 수 있다. 이를 유형축소(type contraction)라고 한다.

2.2 관계노드

개념노드 간의 관계를 말해 주는 관계노드(relation node)는 그래프 상에서 원으로 표시되며, 선형표현에서는 소괄호 '( )'를 쓴다. 개념노드와는 달리 관계노드에는 참고지대(referent field)가 없고 유형지대(type field)만 있다. 대부분의 관계노드는 두개의 개념노드 간의 관계를 정의하는 이원(binary)관계를 나타내지만 때로는 일원(unary) 또는 다원의 관계를 나타내기도 한다. 다음 개념그래프는 '철수가 그 공을 쳤다.'를 나타내는데

(PAST)→[[PERSON:Chulsoo]←(AGNT)←[HIT]→(OBJ)→[BALL:#]]

이때 (PAST)는 일원관계를 나타내는 관계노드이고 그 밖의 관계노드 (AGNT), (OBJ) 등은 이원관계를 나타낸다. 이때 각 관계노드들은 화살표를 이용하여 관련된 개념노드에 연결되는데 화살표 방향에 따라 같은 개념노드와 관계노드가 있는 그래프라도 서로 다른 의미를 갖는다.

(1)[HIT]→(OBJ)→[BALL:#]
(2)[HIT]←(OBJ)←[BALL:#]

그래프 (1)은 [HIT]의 object가 Ball이라는 의미이고 그래프 (2)는 [BALL:#]의 object가 Hit라는 표현이다.

관계노드는 관계유형에 따라 세 종류로 나눌 수 있는데, 원시적 관계(primitive relation), 초기집합(start set), 정의된 관계(defined relation) 등이 그들이다. 원시적 관계에는 LINK라는 유일한 관계노드가 있는데 원시적 관계노드만을 사용하면 'Mary hitting a ball'은

[PERSON:Mary]←(LINK)←[PATIENT]→(LINK)→[BALL]

로 표현되며 초기집합에 (AGNT), (PTNT) 등이 있어서 이들을 사용하면

[PERSON:Mary]←(AGNT)←[HIT]→(PTNT)→[BALL]

로 표현되며, (HITTING)이라는 관계유형을 정의하여 사용하면

[PERSON:Mary]←(HITTING)→[BALL]

로 표시할 수 있다.

여기에서 보듯이 같은 문장도 어떤 관계유형을 사용하였느냐에 따라서 개념그래프가 다르게 표현된다. 원시적 관계유형 'LINK' 만을 사용할 경우에는 개념그래프 내의 모든 관계노드는 (LINK)만으로 이루어 지며 관계노드의 특성이 개념노드에 의해 표현되어야 하므로 그래프가 가장 크게 나타나며, 초기집합에 LINK 이외의 여러 관계유형을 미리 설정해 놓으면 그만큼 다양한 관계를 쉽게 표현할 수 있다. 하지만, 너무 많은 관계유형을 만들어 놓으면 문장을 개념그래프에 사상시킬 때 그만큼 더 복잡한 처리과정을 필요로 한다. 따라서 적당한 수의 관계유형을 설정하면 시스템의 효율을 높일 수 있다. 2.1절에서 보았듯이 개념유형 뿐만 아니라 관계유형도 정의하여 사용할 수 있다. 초기집합에 없는 관계유형이 필요할 때에는 필요한 관계유형을 정의하여 사용하면 된다. 관계유형의 정의도 개념유형의 정의와 같은 형식으로 이루어 진다. 예를 들어 과거시제를 나타내는 'PAST' 라는 관계유형의 정의는 다음과 같다.

relation PAST(X) is

[SITUATION:*X]→(PTIM)→[TIME]→(SUCC)→[TIME:NOW].

2.3 유형계층

많은 인공지능 시스템들이 지식을 사용하는데 지식의 양이 방대할 때에는 이를 체계적으로 표현하기 위하여 응용분야에서 필요한 개념들의 계층(ontology)을 구성할 필요가 있다. 개념그래프에서는 이를 유형계층(type hierarchy)이라고 하는데 개념노드와 관계노드를 위한 별개의 유형계층이 필요하다. 유형은 사물의 집합이 아니라 사고의 추상적인 영역구분에 의하여 정의된다. 함수 δ는 유형을 사물의 집합으로 바꾸어 준다. 예를 들어 'DOG'가 개를 나타내는 유형이면 'δDOG'는 이 세상에 존재하는 모든 개의 집합을 나타낸다. 따라서 UNICORN처럼 유형은 존재하나 그에 대응되는 사물의 집합은 없을 수 있다.

유형계층은 유형집합 내 유형들 간의 반순서(partial order)관계로 정의된다. 유형계층은 임의의 두 유형 간에 직접 공통적인 상위유형(super type)과 하위유형(sub type)을 갖을 때 트리(tree)가 아닌 속(lattice) 형태를 갖는데 이는 두개의 특수한 유형, 즉 '⊤'(universal type)와 '⊥'(absurb type)을 포함한다. '⊤'는 다른 모든 유형의 상위유형으로 [⊤:*]로 표시할 수 있고, 참고지대(referent field)에 아무런 제약도 없으며 단지 무언가 존재한다는 뜻을 가지고 있다. '⊥'는 다른 모든 유형의 하위유형을 나타낸다. 그림 5는 [10]에 있는 유형계층의 윗부분만 예로 보인 것이다.

유형계층 내의 유형은 다시 자연유형(natural type)과 기능유형(role type)으로 구분하여 생각할 수 있는데, '사람', '고양이' 등은 자연유형에 속하고 '학생', '아버지' 등은 기능유형에 속한다. 때에 따라서는 하나의 유형이 서로 다른 자연유형과 기능유형에 속하기도 한다.

2.4 규범그래프

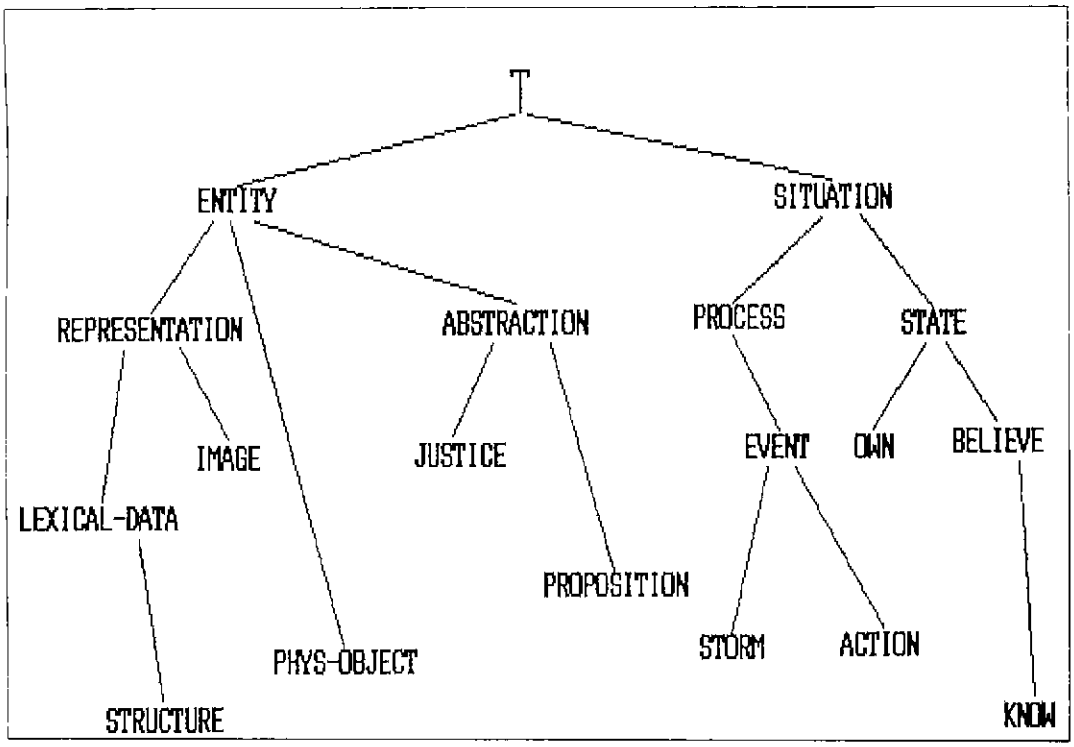


그림 5

개념그래프는 개념노드들이 관계노드들에 의해 서로 연결됨으로써 형성되는데 임의의 개념노드와 관계노드들이 아무런 제약없이 연결될 수 있다면 의미없는 개념그래프가 생성될 수 있다. 예를 들어 촘스키가 [2]에서 선보인 문장 'Colorless green ideas sleep furiously'을 개념그래프로 표현하면

[SLEEP]→(AGNT)→[IDEA]→(COLOR)→[GREEN]

이 된다. 위 문장이나 개념그래프는 구문상으로는 아무 이상이 없다. 하지만 실제 상황이나 우리의 상식에 비추어 옳지 않은 것이다. 따라서 문장은 구문규칙 뿐만 아니라 단어들의 의미도 고려하여 생성되어야 함을 알 수 있다. 이를 위하여 Katz와 Fodor[5]는 단어들에 선택제약을 부과하는 의미론을 개발하였고 이는 Fillmore[4]의 격문법(Case grammer)과 Wilks[11]의 선호의미론(Preference semantics)으로 발전하였다.

각 개념노드에 선택제약을 부과하여 생성된 개념그래프를 규범그래프(Canonical graph)라 한다. 규범그래프는 각 개념을 정의할 때 기본적으로 주어진 규범그래프가 있고 새로운 규범그래프는 규범그래프 형성규칙(Canonical formation rule)에 의해 생성될 수 있다. 다음은 GIVE라는 개념에 해당하는 규범그래프이다.

[GIVE]-
(AGNT)→[ANIMATE]
(RCPT)→[ANIMATE]
(OBJ)→[ENTITY].

Sowa는 다음 네 가지의 규범그래프 형성규칙을 설정하였다[9]. 다음 각 규칙에 의하여 u와 v라는 규범그래프가 있을 때 규범그래프 형성규칙에 의해 w라는 규범그래프가 도출될 수 있다.

◦ 복사(copy) : w는 v와 똑같은 개념그래프이다.

◦ 제한(restrict): u 안에 있는 개념노드의 유형을 하위유형으로 대치하거나 참고 지대(referent field)에 총칭적 표시자(generic marker)가 있을 때 이를 개체표시자(individual marker)로 대치한다. 이때 개체표시자는 그 개념노드의 유형에 속할 수 있는 것이어야 한다.

◦ 결합(join) : u에 있는 개념노드 c와 v에 있는 개념노드 d가 같을 때 d를 제거하고 d에 붙어 있던 모든 관계노드를 c에 붙여서 w를 생성한다.

◦ 단순화(simplify) : 하나의 개념그래프 상에 관계노드가 복사되어 있다면 이는 그 관계노드에 붙어 있는 아크와 함께 제거한다.

여기서 제한은 각개의 개념노드에 적용시켜 더욱 세부적인 정보를 표현하도록 하며 세부화(Specialization)라고도 한다. 어떤 개념그래프 u가 v로부터 규범그래프 생성규칙에 의해 생성됐다면, u는 v로부터 세부화된 개념그래프라고 말하고 그 관계는 u≤v로 표시한다. 이때 v는 u의 일반화(Generalization)된 개념그래프이다.
예를 들면

[PERSON:Mary]←(AGNT)←[HIT]→(PTNT)→[BALL]는

[PERSON]←(AGNT)←[HIT]→(PTNT)→[ENTITY]

보다는 세부화된 개념그래프이다.

2.5 추상화와 집합화

추상화(abstraction)란 프로그래밍 언어에서 쓰이는 프로시듀어(procedure)나 매크로(macro)처럼 복잡한 내용을 간단한 이름으로 대치하여 사용할 수 있도록 하는 기법이다. 추상화를 위해서 라므다수식(lambda expression)을 사용하는데, 이는 형식매개변수(formal parameter)를 잘 이용할 수 있기 때문이다. 라므다 표현 $\lambda_{a1,\cdots,an}$ U는 개념그래프 U를 몸체(body)로 그리고 총칭적 개념(generic concept) $a_1, \cdots, a_n$을 형식매개변수로 갖는다. 2.1절에서 보았듯이 SMALL-PERSON이라는 새로운 유형을 PERSON의 하위유형으로 정의하면

SMALL-PERSON = (λ_X)[PERSON:*X]→(SIZE)→[SMALL]

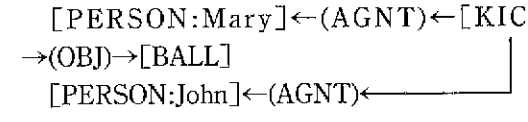
라고 정의되고 개념그래프 [PERSON:John]

→(SIZE)→[SMALL]은 [SMALL-PERSON: John]으로 줄여 쓸 수 있으며 이러한 현상을 유형축소(type contraction)라고 한다. 이렇게 축소된 그래프는 다시 확장(expansion)될 수 있는데 원래의 그래프가 아닌

[SMALL-PERSON:John]→(SIZE)→[SMALL]

로 확장된다. 이러한 축소와 확장은 개념유형뿐만 아니라 관계유형에도 적용된다.

집합화(aggregation)는 2.1절에서 보듯이 여러 객체를 참고지대에 표시할 수 있게 하는 기법이다. 예를 들어 'Mary와 John이 공을 찬다.' 라는 문장은



로 표현되는데, 이 때 [PERSON:Mary]와 [PERSON:John]을 하나의 개념노드 [PERSON:{Mary, John}]으로 표현하는 것을 집합화라 하고 주어진 개념그래프는

[PERSON:{Mary, John}]←(AGNT)←[KICK]→(OBJ)→[BALL]

로 변형될 수 있다.

2.6 개념그래프와 술어논리

개념그래프가 하나의 논리표현이므로 다른 논리표현으로의 변환이 꼭 필요한 것은 아니다. 하지만 상호변환이 가능하면 한 곳에서 발전된 이론을 서로 공유할 수 있고 하나의 표현을 기반으로 개발된 소프트웨어를 다른 표현을 위해서도 사용할 수 있게 된다[10]. 이 장에서는 개념그래프가 술어논리(Predicate Calculus)로 변환되는 예를 보인다.

예를 들어 '원숭이가 바나나를 먹고 있다.'는 문장은 개념그래프 u로 표현될 수 있다.

u=[MONKEY]←(AGNT)←[EAT]→(OBJ)→[BANANA:#123]

이를 함수 ϕ를 이용하여 술어논리로 바꾸면

$$u = \exists_x \exists_y (MONKEY(X) \wedge AGNT(y, x) \wedge EAT(Y) \wedge OBJ(Y, \#123) \wedge BANANA(\#123))$$

이 된다. 함수 ϕ는 [9]에 정의되어 있으며 [10]에도 설명되어 있다. 이때 개념그래프에서 [BANANA:#123]와 같은 개체표시자를 갖는 개념(individual concept)은 술어논리에서 상수로, 그리고 [MONKEY]와 같은 일반개념(generic concept)은 변수로 변환된다. 또한 (AGNT)와 같은 관계노드는 그 형(type)과 같은 이름을 갖는 술어(predicate)로 변환된다.

개념그래프가 술어논리보다 표현력이 크므로 Sowa가 정의한 ϕ는 술어논리로 표현가능한 개념그래프만을 변환할 수 있다. 자연언어는 개념그래프로 변환이 가능하지만 개념그래프로 표현된 모든 것을 술어논리로 표현하기는 쉽지 않다. 따라서 자연언어의 의미를 술어논리로 표현하는 것은 자연스럽지 못하다[10].

지금까지 개념그래프의 중요내용을 살펴보았는데 이밖에도 배경지식(background knowledge)의 표현을 위해 쓰이는 스키마(schema)와 계산이나 문제해결을 위해 쉽게사용될 수 있는 활동자(actor) 등이 있다. 이들에대한 설명은[9, 10]에 있으며 여기서는 생략하기로 한다.

3. 개념그래프의 연구현황

본 장에서는 국내와 국외의 개념그래프에 관한 연구현황을 소개한다.

3.1 국내 연구현황

국내에서는 아직 개념그래프 관련연구가 활성화되지 않은 상태이지만(저자가 알고있는 범위내에서) 현재 한국과학기술원 전산학과와 목포대학교 전산통계학과 그리고 전북대학교 전산학과에서 개념그래프를 응용한 시스템 개발을 위한 기초연구가 진행되고 있다. 한편 충남대학교 영문과에서는 언어학적인 입장에서 개념그래프의 표현력 향상을 위한 연구가 진행 중이다.

3.2 국외 연구현황

세계 여러 곳에서 개별적으로 진행되고 있는 개념그래프 관련 연구도 많지만 개념그래프에 관한 국제 프로젝트로 PEIRCE가 있다. PEIRCE 프로젝트는 실용가능한 첨단의 개념그래프 응용 시스템을 공동으로 개발하기 위해 세계 각국에서 40명 이상의 개념그래프 관련 연구자들이 함께 시작했으며, 1992년 7월 New Mexico State University에서 처음 모임을 갖았다[3].

개념그래프 이론의 부분들을 구현한 여러 시스템들이 있기는 하지만 아직 개념그래프를 이용한 응용시스템의 개발을 위한 도구들이 부족한 상태이다. PEIRCE 프로젝트는 다음과 같은 세 부그룹으로 나뉘어 있다.

3.2.1 표준화 그룹

PIERCE의 구현을 위해 많은 프로그래머들이 동원되므로 각기 다른 소프트웨어들이 같이 운영되려면 표준화가 요구된다. 표준화 그룹에서는 PEIRCE 구현을 위한 주언어로 C++를 선정하고 UNIX workstation 상에 X-Window를 구현환경으로 사용하기로 결정하였다. 후에 Macintosh와 IBM PC들 과도 호환성을 갖도록 할 계획이며 시스템 구현의 전반적인 관리를 담당한다.

3.2.2 인터페이스 그룹

인터페이스 그룹에서는 PEIRCE의 각 모듈 간의 인터페이스를 개발한다.

3.2.3 데이타베이스 그룹

본 그룹에서는 많은 수의 개념그래프를 저장, 검색하고 관리하는 시스템을 개발한다.

3.2.4 선형표현 그룹

선형표현 그룹에서는 개념그래프의 선형표현을 위한 문법을 표준화하고 선형표현을 위한 파서와 생성기를 개발한다.

3.2.5 도형표현 그룹

도형표현 그룹에서는 개념그래프를 도형으로 작성하고 표현할 수 있는 도구를 개발한다.

3.2.6 개념목록 그룹

본 그룹에서는 세상정보(Ontology)를 개념그래프 형태로 표현하고 저장하며 다른 지식베이스(CYC, KIF, KL-ONE, SNePS 등)와 상호 정보교환이 가능하도록 하는 변환기를 개발한다.

3.2.7 프로그래밍언어 그룹

본 그룹에서는 개념그래프를 사용하여 프로그램을 할 수 있도록 컴파일러를 개발한다. 개념그래프 프로그램언어는 Conceptual Graphs with Constraints(CGC)인데 Prolog와 비슷하다.

3.2.8 추론 그룹

본 그룹에서는 개념그래프를 증명에 사용할 수 있도록 하는 도구를 개발하며 이는 CGC를 사용하여 구현될 것이다.

3.2.9 학습 그룹

본 그룹에서는 개념그래프를 데이타베이스에 선택적으로 추가 또는 삭제할 수 있는 기능과 여러 예제들로부터 구조적인 개념들을 찾아낼 수 있는 능력을 갖춘 도구를 개발한다.

3.2.10 자연언어처리 그룹

본 그룹에서는 자연언어 문장을 개념그래프로 변환하고, 또 개념그래프 표현을 자연언어 문장으로 변환할 수 있는 분석기와 생성기를 개발한다.

이 밖에도 여러 그룹이 추가될 수 있고 이렇게 여러 그룹에서 개발한 시스템들을 하나로 모아 PEIRCE를 완성하게 된다. PEIRCE 프로젝트는 비영리적인 프로젝트이므로 성공여부는 미지수이지만 현재까지는 활발하게 진행되고 있다.

4. 결 론

이상에서 살펴본 바와 같이 개념그래프는 인간과 컴퓨터 사이에서 컴퓨터가 자연언어를 효율적으로 이해할 수 있는 방법을 제시한다. 지금까지 많은 지식표현언어가 개발되었지만 이들과는 달리 개념그래프는 자연언어와 상호변환이 용이하고 사람이 읽고 이해하기가 쉽다. 또한 개념그래프는 일종의 논리언어 이므로 기계가

처리하기에도 좋은 표현이다. 따라서, 자연언어를 사용하여 기술된 많은 지식을 컴퓨터를 이용하여 효과적으로 사용하려면 개념그래프를 빌어 지식을 표현하고 처리하도록 하는 것도 좋은 방법일 것이다.

PIERCE와 같은 개념그래프관련 국제 프로젝트도 진행 중에 있고 관련학회도 매년 열리고 있다. 세계적으로 활발하게 연구가 진행되고 있는 개념그래프 관련연구가 국내에서도 활성화되기를 바라는 바이다.

참고문헌

[1] Bach, E & R. T. Harms, Universals in Linguistic Theory, Holt, Rinehart & Winston, NewYork, 1968.

[2] Chomsky, N, Syntactic Structures, Mouton, The Hague, 1957.

[3] Ellis, G & R. Levinson, Proceedings of the First International Workshop on PIERCE:A Conceptual Graphs Workbench, Las Cruses, New Mexico, 1992.

[4] Fillmore, C. J. "The Case for Case" in Bach & Harms, pp. 1~88, 1968.

[5] Katz, J & J. Fodor, "The Structure of a Semantic Theory", Language, Vol. 31, pp. 170~210, 1963.

[6] Levinson. R. & G. Ellis, "Multi-level Hierarchical Retrieval", in Proceedings of the Sixth Annual Workshop on Conceptual Graphs, Binghamton, pp. 67~81, 1991.

[7] Peirce, C. S, "Manuscripts on Existential Graphs", in Roberts, D. D., 1897.

[8] Robert, D. D. "The Existential Graphs of Charles S. Peirce", Mouton, The Hague, 1973.

[9] Sowa, J. F., Conceptual Structures:Information Processing in Mind and Machine, Addison-Wesley, Massachusetts, 1984.

[10] Sowa, J. F., Conceptual Structures:Current Research and Practice, Ellis Horwood, 1992.

[11] Wilks, Y, "An Intelligent Analyzer and Understander of English", Comm.ACM, Vol. 18, No. 5, pp. 264~274, 1975.

양 기 철

전남대학교 전산학 학사
University of Iowa, 전산학 석사
University of Missouri 전산학 박사
현재 목포대학교 전산통계학과 전임강사
관심분야 : 인공지능, 분산처리

● 제 13회 정보산업리뷰 심포지움 ●

- **일 자** : 1994년 12월 8일
- **장 소** : 한국종합전시관(4층)
- **주 제** : "UR에 대비한 정보산업의 대응 방안"
- **문 의** : 한국정보과학회 사무국
  Tel : (02) 588-9246
  Fax : (02) 521-1352