컴퓨터 기반의 지식표현과 추론

Computer-based Knowledge Representation and Reasoning

김진현

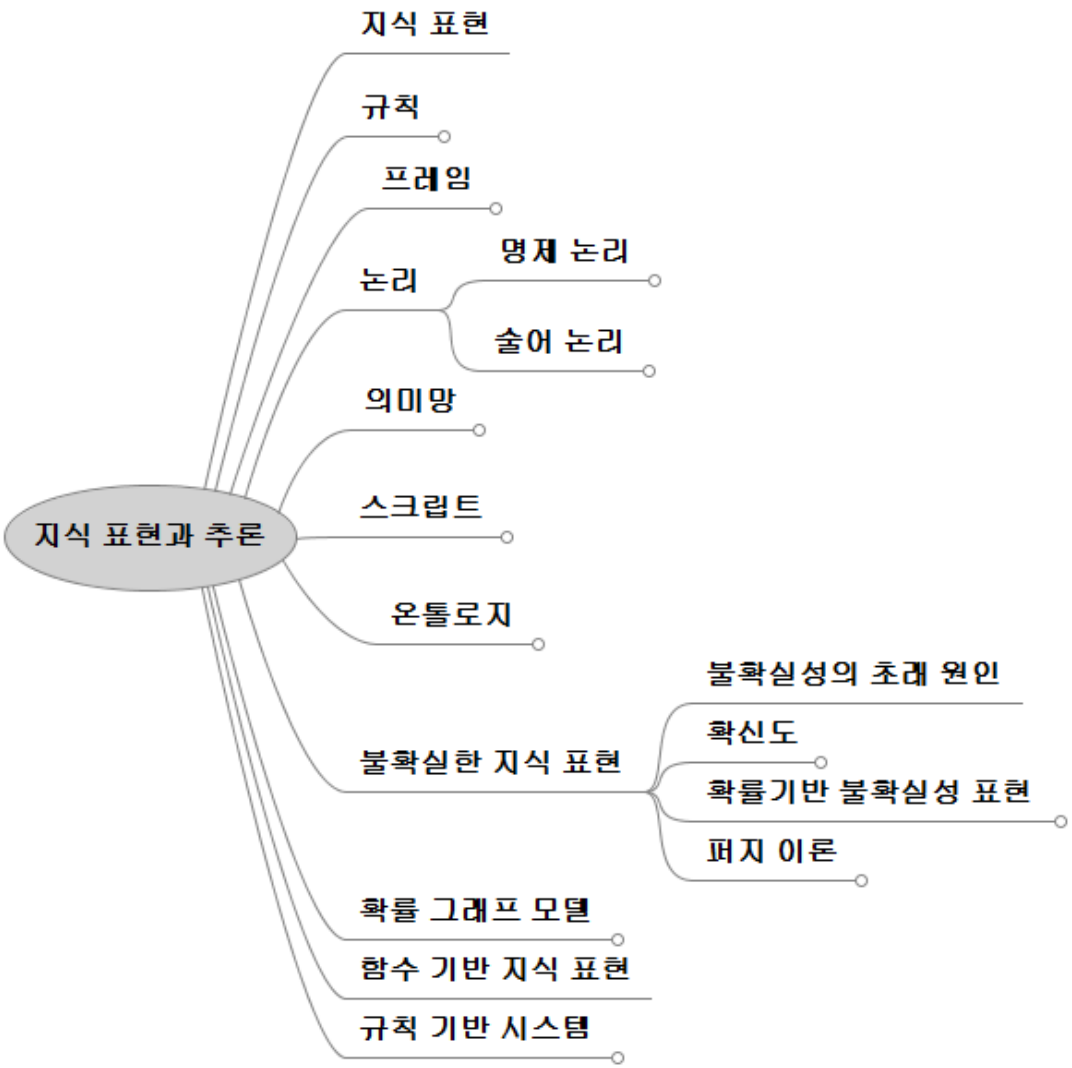
이 수업에서...

- 규칙 기반 시스템
- 논리 시스템
  - 명제논리
  - 술어논리

Reference

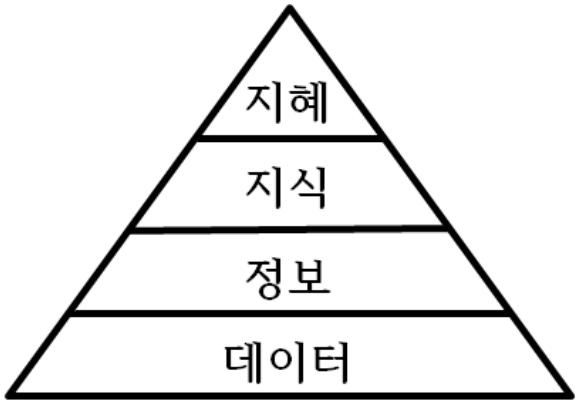
- 인공지능 – 튜링머신에서 딥러닝까지. 이건명 교수

컴퓨터 기반의 지식 표현과 추론



지식 표현

- **데이터 피라미드**
  - **데이터** (data)
    - 특정 분야에서 **관측된 아직 가공되는 않은 것**
    - 사실인 것처럼 관측되지만 **오류**나 **잡음**을 포함 ㄱ
  - **정보** (information)
    - 데이터를 **가공**하여 어떤 **목적**이나 **의미**를 갖도록 한 것
  - **지식** (knowledge)
    - 정보를 **취합**하고 **분석**하여 얻은 대상에 대해 사람이 **이해**한 것
  - **지혜** (wisdom)
    - 경험과 학습을 통해서 얻은 지식보다 높은 수준의 **통찰**



지식 표현

- **지식**(知識, knowledge)
  - **경험**이나 **교육**을 통해 얻어진 **전문적인 이해**(understanding)와 체계화된 **문제 해결 능력**
  - 어떤 주제나 분야에 대한 **이론적** 또는 **실제적인 이해**, 또는 현재 알려진 **사실**과 **정보**의 모음
  - **암묵지**(暗黙知, tacit knowledge)
    - **형식을 갖추어 표현하기 어려운, 학습과 경험을 통해 쌓은 지식**
  - **형식지(形式知, explicit knowledge)**
    - **비교적 쉽게 형식을 갖추어 표현될 수 있는 지식**

지식 표현

- **지식**(知識, knowledge) – cont'd
  - **절차적 지식(procedural knowledge)**
    - **문제해결의 절차 기술**
  - **선언적 지식(declarative knowledge)**
    - **어떤 대상의 성질, 특성이나 관계 서술**
  - **컴퓨터를 통한 지식 표현 및 처리**
    - **프로그램이 쉽게 처리할 수 있도록 정형화된 형태로 표현**
    - **규칙, 프레임, 논리, 의미망, 스크립트, 수치적 함수 등**

규칙 (Rule)

- **규칙 (rule)**
  - **'~이면, ~이다'** 또는 **'~하면, ~하다'**와 같은 **조건부의 지식**을 표현하는 **IF-THEN 형태**의 문장
  - 직관적이고 이해하기 쉬움
- **규칙 획득 및 표현**
  - **예:** 신호등이 녹색일 때는 건널목을 안전하게 건널 수 있고, 빨간색일 때는 길을 건너지 말아야 한다
  - **대상**, **속성**, **행동** 또는 **판단**의 정보 추출
    - 대상 : 신호등
    - 속성 : 녹색, 빨간색
    - 행동/판단 : 건넌다, 멈춘다.
  - **표현**
    - **IF** 신호등이 녹색이다 **THEN** 행동은 건넌다
    - **IF** 신호등이 빨간색이다 **THEN** 행동은 멈춘다

규칙

- **규칙 (rule)**
  - **IF** 신호등이 녹색이다 **THEN** 행동은 건넌다
    - **IF** trafficLight = green **THEN** action = cross
  - **IF** 신호등이 빨간색이다 **THEN** 행동은 멈춘다
    - **IF** trafficLight= red **THEN** action = stop
  - **IF 부분**
    - 주어진 정보나 사실에 대응될 조건
    - **조건부**(conditional part, antecedent)
  - **THEN 부분**
    - 조건부가 만족될 때의 판단이나 행동
    - **결론부**(conclusion, consequent)

규칙

- **규칙의 구성**
  - **조건부**
    - 둘 이상의 조건을 **AND** 또는 **OR**로 **결합**하여 구성 가능
      - **IF** <조건1> **AND** <조건2> **AND** <조건3> **THEN** <결론>
      - **IF** <조건1> **OR** <조건2> **OR** <조건3> **THEN** <결론>
  - **결론부**
    - 여러 개의 판단 또는 행동 포함 가능
      - **IF** <조건>

        THEN <결론1>

        AND <결론2>

        AND <결론3>

규칙

- **규칙을 통한 지식 표현**
  - **인과관계**
    - **원인**을 조건부에 **결과**는 결론부에 표현
      - **IF** 연료통이 빈다
        THEN 차가 멈춘다
  - **추천**
    - **상황**을 조건부에 기술하고 이에 따른 **추천 내용**을 결론부에 표현
      - **IF**여름철이다 **AND** 날이 흐리다
        THEN 우산을 가지고 가라
  - **지시**
    - **상황**을 조건부에 기술하고 이에 따른 **지시 내용**을 결론부에 표현
      - **IF** 차가 멈추었다 **AND** 연료통이 비었다
        THEN 주유를 한다

규칙

- **규칙을 통한 지식 표현**
  - **전략 (strategy)**
    - **일련의 규칙**들로 표현
    - 이전 단계의 판정 결과에 따라 다음 단계에 고려할 규칙이 결정
      - **IF** 차가 멈추었다
        THEN 연료통을 확인한다 **AND** 단계1을 끝낸다
      - **IF** 단계1이 끝났다 **AND** 연료통은 충분히 찼다
        THEN 배터리를 확인한다 **AND** 단계2를 끝낸다
  - **휴리스틱 (heuristic)**
    - **경험적인 지식**을 표현하는 것
    - 전문가적 견해는 최적을 항상 보장하는 것이 아니고 **일반적으로 바람직한 것**을 표현
      - **IF** 시료가 액체이다 **AND** 시료의 PH가 6미만이다 **AND** 냄새가 시큼하다 **THEN** 시료는 아세트산이다

규칙 기반 시스템

- **규칙 기반 시스템**(rule-based system)
  - 지식을 규칙의 형태로 표현
  - 주어진 문제 상황에 적용될 수 있는 규칙들을 사용하여 문제에 대한 해를 찾도록 하는 **지식 기반 시스템**(knowledge-based system)
  - **전문가 시스템**(expert system)을 구현하는 전형적인 형태
    - 특정 문제영역에 대해서 전문가 수준의 해를 찾아주는 시스템

추론

- **추론**
  - 구축된 지식과 주어진 데이터나 정보를 이용하여 새로운 사실을 생성하는 것
  - **전향 추론**(forward chaining, **순방향 추론**)
    - 규칙의 조건부와 만족하는 사실이 있을 때 규칙의 결론부를 실행하거나 처리
  - **후향 추론**(backward chaining, **역방향 추론**)
    - 어떤 사실을 검증하거나 확인하고 싶은 경우에 관심 대상 사실을 결론부에 가지고 있는 규칙을 찾아서 조건부의 조건들이 모두 만족하는지 확인

전향 추론의 예

R1: IF ?x는 체모가 있다 THEN ?x는 포유류이다.

R2: IF ?x는 수유를 한다 THEN ?x는 포유류이다.

R3: IF ?x는 깃털이 있다 THEN ?x는 조류이다.

R4: IF ?x는 난다 AND ?x는 알을 낳는다 THEN ?x는 조류이다.

R5: IF ?x는 포유류이다 AND ?x는 고기를 먹는다 THEN ?x는 육식동물이다.

R6: IF ?x는 포유류이다 AND ?x는 되새김질한다 THEN ?x는 유제류이다.

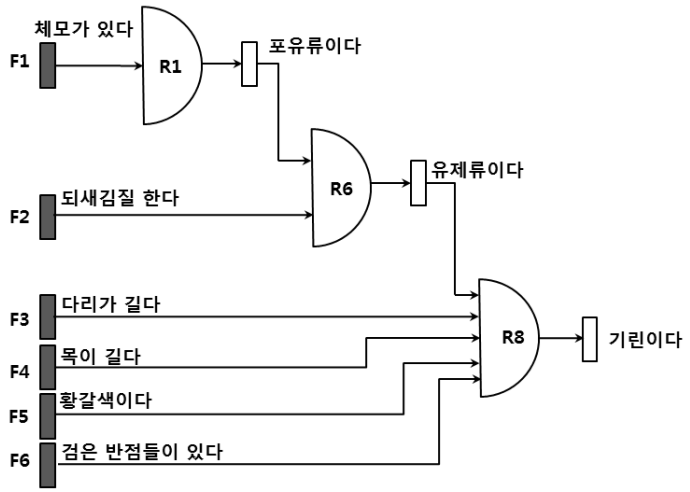
R7: IF ?x는 육식동물이다 AND ?x는 황갈색이다 AND ?x는 검은 반점들이 있다 THEN ?x는 치타이다.

R8: IF ?x는 유제류이다 AND ?x는 다리가 길다 AND ?x는 목이 길다 AND ?x는 검은 반점들이 있다
THEN ?x는 기린이다.

R9: IF ?x는 포유류이다 AND ?x는 눈이 앞을 향해있다 AND ?x는 발톱이 있다 AND ?x는 이빨이 뾰족하다
THEN ?x는 육식동물이다.

래더는 뭘까?

F1: 래더는 체모가 있다.
F2: 래더는 되새김질을 한다.
F3: 래더는 다리가 길다.
F4: 래더는 목이 길다.
F5: 래더는 황갈색이다.
F6: 래더는 검은 반점들이 있다



후향 추론의 예

R1: IF ?x는 체모가 있다 THEN ?x는 포유류이다.

R2: IF ?x는 수유를 한다 THEN ?x는 포유류이다.

R3: IF ?x는 깃털이 있다 THEN ?x는 조류이다.

R4: IF ?x는 난다 AND ?x는 알을 낳는다 THEN ?x는 조류이다.

R5: IF ?x는 포유류이다 AND ?x는 고기를 먹는다 THEN ?x는 육식동물이다.

R6: IF ?x는 포유류이다 AND ?x는 되새김질한다 THEN ?x는 유제류이다.

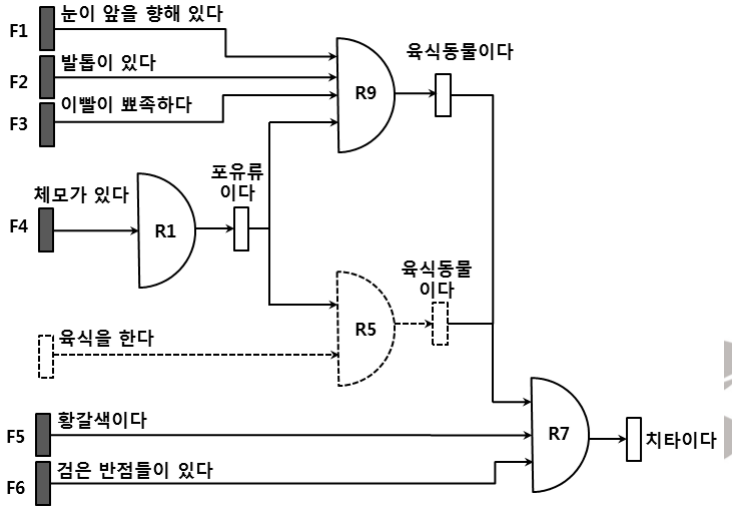
R7: IF ?x는 육식동물이다 AND ?x는 황갈색이다 AND ?x는 검은 반점들이 있다 THEN ?x는 치타이다.

R8: IF ?x는 유제류이다 AND ?x는 다리가 길다 AND ?x는 목이 길다 AND ?x는 검은 반점들이 있다
THEN ?x는 기린이다.

R9: IF ?x는 포유류이다 AND ?x는 눈이 앞을 향해있다 AND ?x는 발톱이 있다 AND ?x는 이빨이 뾰족하다
THEN ?x는 육식동물이다.

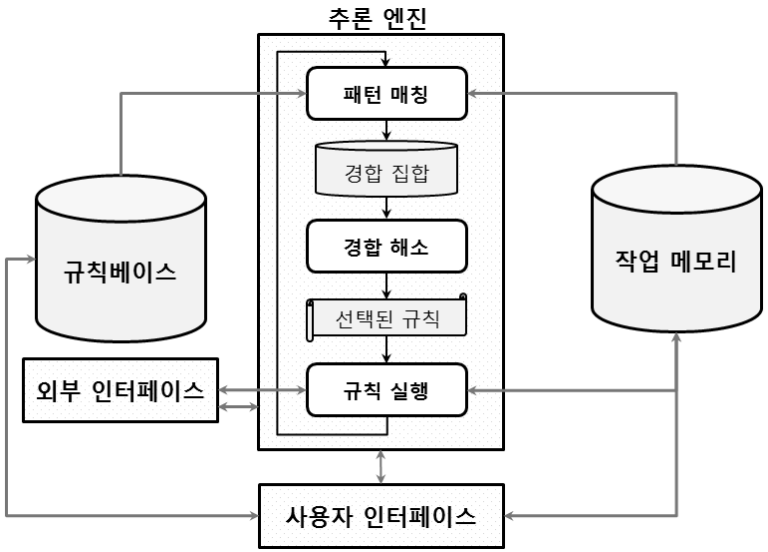
스프린터는 치타인가?

F1: 스프린터는 눈이 앞을 향해 있다.
F2: 스프린터는 발톱이 있다.
F3: 스프린터는 이빨이 뾰족하다.
F4: 스프린터는 체모가 있다.
F5: 스프린터는 황갈색이다.
F6: 스프린터는 검은 반점들이 있다.



규칙 기반 시스템 구조

- **규칙 기반 시스템 구조**
  - **지식**
    - 규칙과 사실로 기술
    - **규칙**(rule) : 문제 해결을 위한 지식
    - **사실**(fact) : 문제 영역에 대해 알려진 데이터나 정보

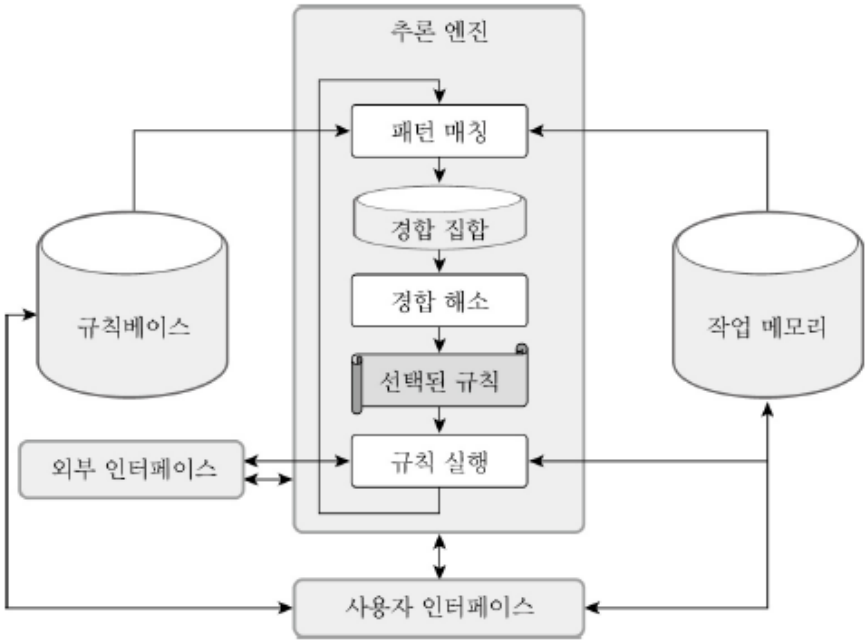


규칙 기반 시스템 구조

- **규칙 기반 시스템 구조**
  - **규칙베이스**(rule base)
    - 전체 규칙의 집합을 관리하는 부분
    - **생성 메모리**(production memory)라고도 함
  - **작업 메모리**(working memory)
    - 사용자로부터 받은 문제에 대한 정보를 관리
    - 추론과정의 중간결과를 저장하고, 유도된 최종해 저장
    - 작업 메모리에 저장되는 모든 것을 **사실**이라 함

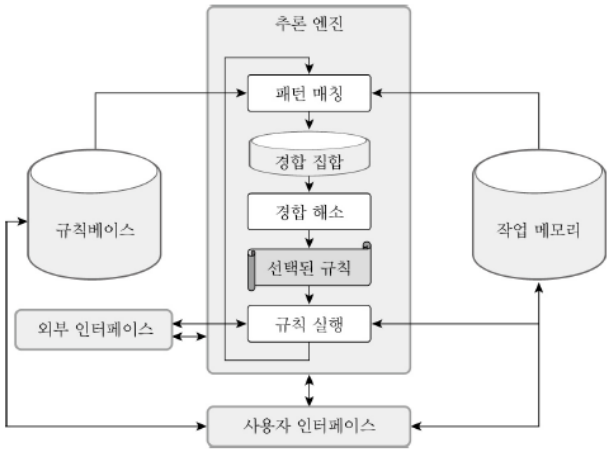
규칙 기반 시스템 구조

- **추론 엔진**(inference engine)
  - 실행할 수 있는 규칙을 찾아서, 해당 규칙을 실행하는 역할
  - ***패턴 매칭*** **- 경합 해소 - 규칙 실행**의 과정 반복



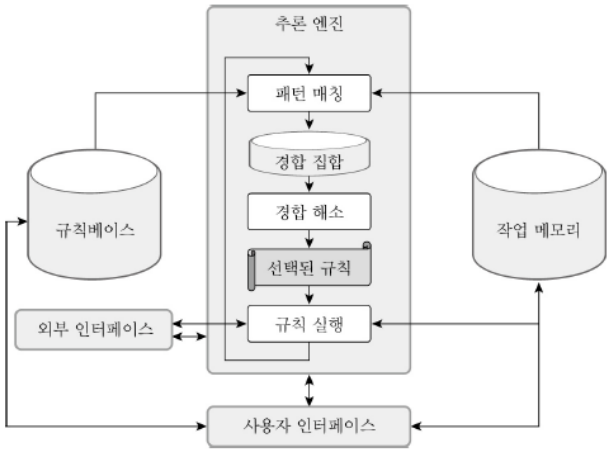
규칙 기반 시스템 구조

- **추론 엔진**(inference engine)
  - **패턴 매칭**(pattern matching)
    - 작업 메모리의 사실과 규칙베이스에 있는 규칙의 조건부를 대조하여 일치하는 규칙을 찾는 과정
  - **경합 집합**(conflict set)
    - 규칙들의 집합, 실행 가능한 규칙들의 집합
  - **경합 해소**(conflict resolution)
    - 경합 집합에서 하나의 규칙을 선택
  - **사용자 인터페이스**(user interface)
    - 규칙베이스 및 작업 메모리 관리 및 추론 엔진 조작
  - **외부 인터페이스**(external interface)
    - 외부 데이터나 함수의 기능 사용 지원



규칙 기반 시스템 구조

- **추론 엔진**(inference engine)
  - **경합 해소 전략**
    - **규칙 우선순위**(rule priority)
      - 미리 각 규칙에 우선순위 부여
      - 경합 집합에서 우선순위가 가장 높은 규칙 선택
    - **최신 우선**(recency, depth)
      - 가장 최근에 입력된 데이터와 매칭된 규칙 선택
    - **최초 우선**(first match, breath)
      - 경합 집합에서 가장 먼저 매칭된 규칙 선택
    - **상세 우선**(specificity)
      - 가장 상세한 조건부를 갖는 규칙 선택
      - 규칙의 조건부가 가장 복잡하게 기술된 것 선택



규칙 기반 시스템

❖ **경합 해소 전략 – cont.**

- **규칙 우선순위**(rule priority)
  - 규칙 1: 뇌막염 처방전 1 (우선순위 100)

    IF 감염이 뇌막염이다

    AND 환자가 어린이다

    THEN 처방전은 Number_1이다

    AND 추천 약은 암피실린(ampicillin)이다

    AND 추천 약은 겐타마이신(gentamicin)이다

    AND 뇌막염 처방전 1을 보여준다
  - 규칙 2: 뇌막염 처방전 2(우선순위 90)

    IF 감염이 뇌막염이다

    AND 환자가 어른이다

    THEN 처방전은 Number_2이다

    AND 추천 약은 페니실린(penicillin)이다

    AND 뇌막염 처방전 2를 보여준다

규칙 기반 시스템

❖ **경합 해소 전략 – cont.**

- **상세 우선**(specificity)
  - 가장 특수한 규칙 선택
  - **규칙 1**

    IF **가을이다**

    AND 하늘이 흐리다

    AND 일기예보에서는 비가 온다고 한다

    THEN 조언은'집에 머무르시오'
  - **규칙 2**

    IF **가을이다**

    THEN 조언은'우산을 가져가시오'

규칙 기반 시스템

❖ **경합 해소 전략 – cont.**

- **최신 우선**(recency, depth)
  - 가장 최근에 입력된 데이터(data most recently entered) 사용 규칙 선택
  - 각 **사실**에 **시간 태그** 부여
    - 규칙 1

      IF 일기예보에서는 비가 온다고 한다 [03/25 08:16 PM]

      THEN 조언은'우산을 가져가시오'
    - 규칙 2

      IF 비가 온다 [03/26 10:18 AM]

      THEN 조언은'집에 머무르시오'

규칙 기반 시스템

- **지식 표현**
  - 개발 도구에 따라 고유한 형식 사용
  - **사실**(fact)
    - 객체(object)나 프레임(frame)처럼 여러 개의 속성 포함 가능
    - 예. '이름이 멍키인 원숭이가 나이가 세 살이고 거실에 있다'

      (monkey (name 멍키) (age 3) (room 거실))
  - **규칙**
    - **Jess의 규칙 표현 예**

```
(defrule ruleBirthday
   ?c <- (monkey (name cheetah) (age ?old) (room ?where) (birthdate ?day))
         (calendar (date ?day))
      =>
         (bind ?newAge (+ ?old 1))
         (retract ?c)
         (assert (monkey cheetah ?newAge ?where)))
```

규칙 기반 시스템

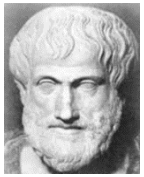
- **규칙 기반 시스템 개발 도구**
  - 규칙 기반 시스템의 기본 컴포넌트들을 미리 제공하여 규칙 기반 시스템을 쉽게 구현할 수 있게 하는 소프트웨어
  - 문제 영역의 지식을 잘 획득하여 정해진 형태로 표현만 하면 규칙 기반 시스템을 비교적 쉽게 구현 가능
  - **Jess**, CLIPS, EXSYS, JEOPS 등

Logic Systems

Jin Hyun Kim

논리

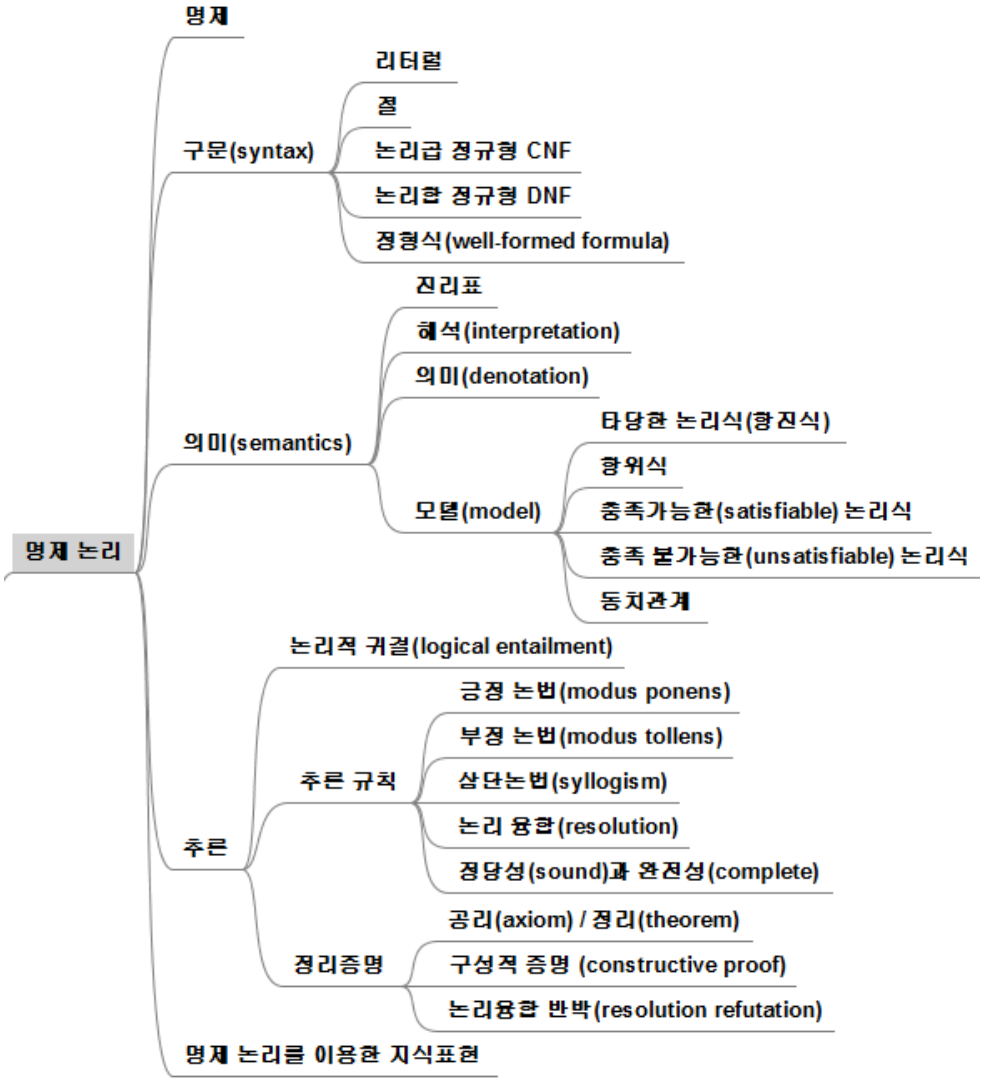
- **논리**(**論理**, logic)
  - **말로 표현**된 **문장**들에 대한 **타당한 추론**을 위해,
    기호를 사용하여 문장들을 **표현**하고
    기호의 **조작**을 통해 문장들의 **참 또는 거짓**을 **판정**하는 분야
  - **논리학의 역사**
    - **아리스토텔레스**(Aristotle, BC384-BC322)
      - 기호의 대수적 조작을 통해 추론을 하는 **삼단 논법**(syllogism) 도입



    - **부울**(George Boule, 1815-1864)
      - **명제 논리**(propositional logic)의 이론적 기초 확립

    - **프리게**(Gottlob Frege, 1848-1925)
      - **술어 논리**(predicate logic)의 이론적 기초를 확립

명제 논리


명제 논리
명제
구문(syntax)
리터럴
절
논리곱 정규형 CNF
논리합 정규형 DNF
정형식(well-formed formula)
의미(semantics)
진리표
해석(interpretation)
의미(denotation)
모델(model)
타당한 논리식(항진식)
항위식
충족가능한(satisfiable) 논리식
충족 불가능한(unsatisfiable) 논리식
동치관계
추론
논리적 귀결(logical entailment)
추론 규칙
긍정 논법(modus ponens)
부정 논법(modus tollens)
삼단논법(syllogism)
논리 융합(resolution)
정당성(sound)과 완전성(complete)
정리증명
공리(axiom) / 정리(theorem)
구성적 증명 (constructive proof)
논리융합 반박(resolution refutation)
명제 논리를 이용한 지식표현

명제 논리

- **명제 논리 (propositional logic)**
  - **명제**(命題, proposition)
    - 참, 거짓을 분명하게 판정할 수 있는 문장
      - 아리스토텔레스는 플라톤의 제자이다. (명제)
      - 1+1 = 3. (명제)
      - 일어나서 아침 먹자. (명제 아님)
  - 명제를 **P, Q**등과 같은 **기호로 표현**
  - 명제 기호의 **진리값**(truth value)을 사용하여 명제들에 의해 표현되는 문장들의 진리값 결정
  - 문장 자체의 **내용**에 대해는 **무관심**, 문장의 **진리값**에만 **관심**

명제 논리

- **기본 명제**(primitive proposition)
  - 하나의 진술(statement)로 이루어진 최소 단위의 명제
- **복합 명제**(compound proposition)
  - 기본 명제들이 결합되어 만들어진 명제
- 예.
  - 알렉산더는 아시아를 넘본다 ⇨ P
  - 징기스칸은 유럽을 넘본다 ⇨ Q
  - 알렉산더는 아시아를 넘보고, 징기스칸은 유럽을 넘본다 ⇨ $P \wedge Q$

명제 논리의 구문

- **논리식**(logical expression)
  - 명제를 기호로 표현한 형식
  - **명제기호**, 참과 거짓을 나타내는 **T**와 **F**, 명제 기호를 연결하는 **논리기호**인 $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\equiv$를 사용하여 구성

논리기호	이름	논리식	의미
$\neg$	부정(negation)	$\neg P$	P 가 아님
$\vee$	논리합(disjunction)	$P \vee Q$	P 또는 Q
$\wedge$	논리곱(conjunction)	$P \wedge Q$	P 그리고 Q
$\rightarrow$	함의(implication)	$P \rightarrow Q$	P 이면 Q
$\equiv$	동치(equivalence)	$P \equiv Q$	$(P \rightarrow Q) \wedge (Q \rightarrow P)$

명제 논리의 구문

- **리터럴**(literal)
  - 명제 기호 P와 명제 기호의 $\neg P$ 부정
- **절**(clause)
  - 리터럴들이 **논리합**으로만 연결되거나 **논리곱**으로 연결된 논리식

$$P \vee Q \vee \neg R \quad \text{(논리합 절)}$$

$$P \wedge Q \wedge \neg R \quad \text{(논리곱 절)}$$

- **논리곱 정규형 (conjunctive normal form, CNF)**
  - 논리합 절들이 논리곱으로 연결되어 있는 논리식

$$(P \vee Q \vee \neg R) \wedge (\neg Q \vee R \vee S) \wedge (P \vee R \vee S)$$

- **논리합 정규형 (disjunctive normal form, DNF)**
  - 논리곱 절들이 논리합으로 연결되어 있는 논리식

$$(P \wedge Q \wedge \neg R) \vee (\neg Q \wedge R \wedge S) \vee (P \wedge R \wedge S)$$

명제 논리의 구문

- **정형식**(well-formed formula, **wff**)
  - 논리에서 문법에 맞는 논리식

    (1) 진리값 T, F와 명제 기호들 $P, Q, R, \cdots$은 정형식이다.

    (2) p와 q가 정형식이면, 논리 기호를 사용하여 구성되는 논리식 $\neg p$, $p \vee q$, $p \wedge q$, $p \rightarrow q$, $p \equiv q$도 정형식이다.

    (3) (1)과 (2)에 의해 정의되는 논리식만 정형식이다.

  - **명제 논리에 대한 정형식**

$$(P \wedge Q) \rightarrow \neg P$$

$$P \rightarrow \neg P$$

$$P \vee P \rightarrow P$$

$$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$$

명제 논리의 의미

- **진리표**(truth table)
  - 논리기호에 따라 참, 거짓 값을 결합하는 방법을 나타낸 표

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \equiv Q$
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

명제 논리의 의미

- 논리식의 **해석**(interpretation)
  - **논리식**의 **진리값**을 **결정**하는 것
    - $P = \mathrm{T}, Q = \mathrm{F}, R = \mathrm{T}$
      - $(P \vee \neg Q) \wedge (Q \vee \neg R) = \mathrm{F}$
  - 우선 각 **명제기호**의 **진리값** 결정 필요
    - 명제 기호에 "명제"를 대응시키고, 해당 명제의 진리값을 결정
      - 예. $\boldsymbol{P}$ => "**토마토는 과일이다**", $P = \mathrm{F}$
    - **대응된 명제**를 명제 기호의 **외연**(外延) 또는 **의미**(denotation)라 함

명제 논리의 의미

- 논리식의 **모델**(model)
  - 논리식의 **명제기호**에 **참값**(T) 또는 **거짓값**(F)을 **할당**한 것
    - $(P \vee \neg Q) \wedge (Q \rightarrow \neg R)$: $P = \mathsf{T}, Q = \mathsf{F}, R = \mathsf{T}$
  - 모델이 주어지면, **진리표**를 사용하여 논리식의 진리값 결정, 즉 **해석** 가능

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \equiv Q$
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

  - n**개**의 **명제기호**가 **논리식**에 사용된다면, 각각 T 또는 F값을 가질 수 있기 때문에, 총 2^n**개**의 **모델**이 존재

명제 논리의 의미

- **타당한 논리식** (**valid** logical expression)
  - 모든 가능한 모델에 대해서 **항상 참**(T)인 논리식
  - **항진식**(恒眞式, tautology)
    - 예. $P \vee \neg P$

      P = T인 경우 : $P \vee \neg P$ = T

      P = F인 경우 : $P \vee \neg P$ = T

- **항위식**(恒僞式, contradiction)
  - 모든 가능한 모델에 대해서 **항상 거짓**이 되는 논리식
    - 예. $P \wedge \neg P$

      P = T인 경우 : $P \wedge \neg P$ = F

      P = F인 경우 : $P \wedge \neg P$ = F

명제 논리의 의미

- **충족가능한**(satisfiable) 논리식
  - 참으로 만들 수 있는 모델이 하나라도 있는 논리식
    - 예. $(P \vee \neg Q) \wedge (Q \vee \neg R)$

      $P = \mathsf{T}, Q = \mathsf{T}, R = \mathsf{F}$

- **충족불가능한**(unsatisfiable) 논리식
  - 참으로 만들 수 있는 모델이 전혀 없는 논리식
  - **항위식**인 논리식
    - 예. $P \wedge \neg P$

명제 논리의 의미

- **동치관계**(equivalence relation)의 논리식
  - **어떠한 모델**에 대해서도 **같은 값**을 갖는 두 논리식

(1) $\neg(\neg p) \equiv p$

(2) $p \vee \mathrm{F} \equiv p, \quad p \wedge \mathrm{T} \equiv p$

(3) $p \vee \neg p \equiv \mathrm{T}, \quad p \wedge \neg p \equiv \mathrm{F}$

(4) $\neg(p \wedge q) \equiv \neg q \vee \neg q, \ \neg(p \vee q) \equiv \neg q \wedge \neg q$

(5) $p \rightarrow q \equiv \neg p \vee q$

(6) $p \vee (q \vee r) \equiv (p \vee q) \vee r, \ p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$

(7) $p \vee (q \wedge q) \equiv (p \vee q) \wedge (p \vee q), \ p \wedge (q \vee q) \equiv (p \wedge q) \vee (p \wedge q)$

명제 논리의 의미

- **동치관계를 이용한 논리식의 변환**
  - 논리식의 동치관계를 이용하면

    임의의 논리식을 **논리곱 정규형**(CNF)과 같은 정형식으로 변환

$$
\begin{aligned}
& p \wedge (q \rightarrow r) \rightarrow p \wedge q \\
& \equiv \neg(p \wedge (\neg q \vee r)) \vee (p \wedge q) \\
& \equiv (\neg p \vee \neg(\neg q \vee r)) \vee (p \wedge q) \\
& \equiv (\neg p \vee (q \wedge \neg r)) \vee (p \wedge q) \\
& \equiv ((\neg p \vee q) \wedge (\neg p \vee \neg r)) \vee (p \wedge q) \\
& \equiv ((\neg p \vee q) \vee (p \wedge q)) \wedge ((\neg p \vee \neg r) \vee (p \wedge q)) \\
& \equiv ((\neg p \vee q) \vee p) \wedge ((\neg p \vee q) \vee q)) \wedge ((\neg p \vee \neg r) \vee p) \wedge ((\neg p \vee \neg r) \vee q)) \\
& \equiv (T \vee q) \wedge (\neg p \vee q) \wedge (T \vee \neg r) \wedge (\neg p \vee \neg r \vee q) \\
& \equiv (\neg p \vee q) \wedge (\neg p \vee q \vee \neg r)
\end{aligned}
$$

명제 논리의 의미

- **논리적 귀결**(logical entailment)
  - Δ : 정형식(wff)의 집합. $\Delta = \{P, P \rightarrow Q\}$

    ω : 정형식. $\omega = Q$

  - Δ에 있는 모든 정형식을 **참**(T)으로 만드는 **모델**이, ω를 **참**(T)으로 만든다

    = Δ는 ω를 **논리적으로 귀결**한다(**logically entail**)

    = ω는 Δ를 **논리적으로 따른다**(**logically follow**)

    = ω는 Δ의 **논리적 결론**(**logical consequence**)이다

- **표기법 :** $\Delta \models \omega$
  - $\{P, P \rightarrow Q\} \models Q$
  - Δ가 참이면, ω도 참이다
- **추론**
  - 참으로 알려진 Δ로 부터, 알려지지 않은 참인 ω를 찾는 것

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \equiv Q$
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

명제 논리의 추론

- **추론**(推論, inference)
  - **귀납적 추론**(inductive inference)
    - 관측된 복수의 **사실들**을 **일반화**(generalization)하여 **일반적인 패턴** 또는 **명제**를 도출하는 것
    - "맹자도, 석가도, 공자도 죽었다. 이들은 모두 사람이다. 곧 사람은 죽는다."
  - **연역적 추론**(deductive inference)
    - **참**인 **사실들** 또는 **명제들(전제)**로 부터 **새로운** 참인 **사실, 결론** 또는 **명제**를 도출하는 것
    - "모든 사람은 죽는다, 소크라테스는 사람이다. 곧 소크라테스는 죽는다."
    - "미인은 잠꾸러기이다. A는 잠꾸러기는 아니다. 그러므로 A는 미인이 아니다."
  - **논리**에서의 **추론**
    - 함의(→)의 논리적 관계를 이용하여 **새로운 논리식**을 **유도**해 내는 것
    - 함의 $p \rightarrow q$
      - p : **전제**(premise)
      - q : **결론**(conclusion, consequence)

명제 논리의 추론

- **추론규칙**(inference rule)
  - 참인 논리식들이 **논리적으로 귀결**하는 **새로운 논리식**을 만들어내는 **기계적**으로 **적용**되는 규칙
  - **긍정 논법** (modus ponens)

$$\begin{array}{l} p \rightarrow q \\ p \\ \hline q \end{array} \qquad \begin{array}{l} \text{새이다} \rightarrow \text{날 수 있다} \\ \text{새이다} \\ \hline \text{날 수 있다} \end{array} \qquad p \rightarrow q,\ p \vdash q$$

  - **부정 논법** (modus tollens)

$$\begin{array}{l} p \rightarrow q \\ \neg q \\ \hline \neg p \end{array} \qquad \begin{array}{l} \text{새이다} \rightarrow \text{날 수 있다} \\ \text{날 수 없다} \\ \hline \text{새가 아니다} \end{array} \qquad p \rightarrow q,\ \neg q \vdash \neg p$$

  - **삼단 논법** (syllogism)

$$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \hline p \rightarrow r \end{array} \qquad \begin{array}{l} \text{새이다} \rightarrow \text{날개가 있다} \\ \text{날개가 있다} \rightarrow \text{날 수 있다} \\ \hline \text{새이다} \rightarrow \text{날 수 있다} \end{array} \qquad p \rightarrow q,\ q \rightarrow r \vdash p \rightarrow r$$

명제 논리의 추론

- **추론규칙** – cont.
  - **논리 융합 (resolution)**
    - 일반화된 추론규칙
      - 긍정 논법, 부정 논법, 삼단 논법의 규칙을 포함한 추론 규칙
    - **두 개**의 **논리합절**이 **같은 기호**의 **긍정**과 **부정**의 **리터럴**을 서로 포함하고 있을 때, 해당 리터럴들을 제외한 **나머지 리터럴들**의 **논리합절**을 만들어 내는 것

$$P \vee q, \neg P \vee r \vdash q \vee r$$

논리융합식(resolvent)

명제 논리의 추론

- **추론 규칙의 정당성과 완전성**
  - **추론 규칙의 정당성 (sound)**
    - 추론 규칙에 의해 생성된 논리식은 주어진 논리식들이 논리적으로 귀결하는 것이다.
    - 즉, 추론 규칙이 만들어 낸 것은 항상 참이다.

$$\Delta \vdash \omega \rightarrow \Delta \models \omega$$

  - **추론 규칙의 완전성 (complete)**
    - 주어진 논리식들이 **논리적으로 귀결하는 것**들은 **추론 규칙**이 **찾아낼 수 있다**.
    - 즉, 추론규칙이 참인 것을 모두 찾아 낼 수 있다.

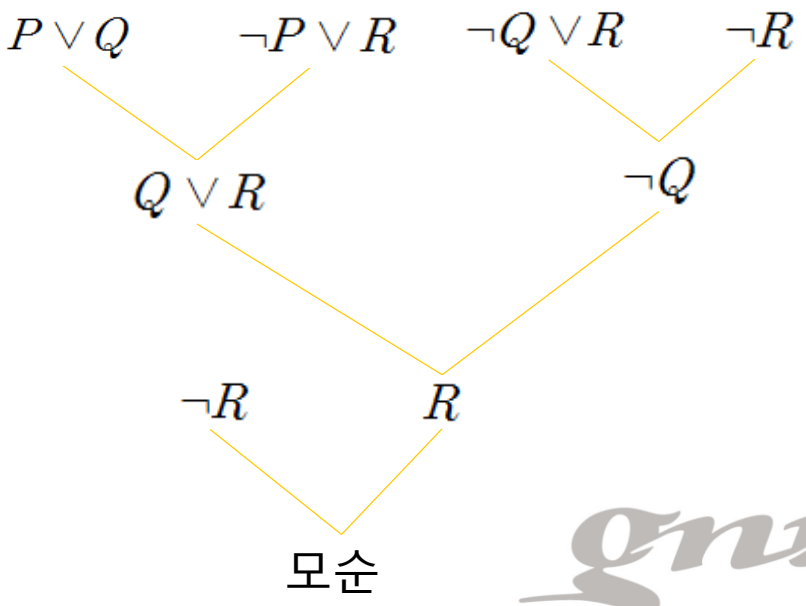
$$\Delta \models \omega \rightarrow \Delta \vdash \omega$$

명제 논리의 추론

- **정리증명**(theorem proving)
  - **공리**(axiom)
    - 추론을 할 때, 참인 것으로 주어지는 논리식
  - **정리**(theorem)
    - 공리들에 추론 규칙을 적용하여 얻어지는 논리식
  - **정리 증명**
    - 공리들을 사용하여 정리가 참인 것을 보이는 것
    - **구성적 증명**(**constructive proof**)
      - 공리들에 추론 규칙들을 적용하여 증명을 만들어 보이는 증명
    - **논리융합 반박**(**resolution refutation**)
      - 증명할 정리를 부정(negation)한 다음, 논리융합 방법을 적용하여 모순이 발생하는 것을 보여서, 정리가 참임을 증명하는 방법

명제 논리의 추론

- **논리융합 반박을 이용한 정리증명의 예**
  - **공리** $P \vee Q$, $P \rightarrow R$, $Q \rightarrow R$ $\quad\quad$ $\neg P \vee R$, $\neg Q \vee R$
  - **정리** $R \quad \Rightarrow \quad \neg R$



명제 논리의 지식표현

- **명제 논리를 이용한 지식 표현**
  - 문장으로 표현된 지식으로부터 **기본 명제**들을 **추출**
  - 각 명제에 대해 **명제기호 부여**
  - 기본 명제들의 **논리적 연결 관계**를 참고하여 대응되는 명제 기호들을 **논리기호**로 **연결**하여 논리식 구성
- **명제 논리로 표현된 지식에 대한 추론**
  - **명제 기호**가 나타내는 명제의 **의미**와는 **무관**
  - 대수적인 **기호 연산**을 통해서 **추론** 수행

다음

- 술어논리

Predicated Logic (술어논리)

Jin Hyun Kim

술어 논리

- **술어 논리**(predicate logic)
  - 명제의 내용을 다루기 위해 **변수**, **함수** 등을 도입하고 이들의 값에 따라 참, 거짓이 결정되도록 명제 논리를 확장한 논리
  - **술어**(述語, predicate)
    - 문장의 '주어+서술어'형태에서 **서술어**에 해당
    - 대상의 **속성**이나 **대상 간의 관계**를 기술하는 기호
    - **참**(T) 또는 **거짓**(F) **값**을 갖는 **함수**
    - 예.
      - **Student**(John)
      - **Friend**(John, Mary)

술어 논리의 구문

- **술어 논리**
  - **존재 한정사**(existential quantifier) ∃와 **전칭 한정사**(universal quantifier) ∀ 사용
    - 변수의 범위를 고려한 지식을 표현
    - $\exists x \ \text{Friend}(\text{John}, x)$
      - 'John은 친구가 한 명은 있다'
    - $\forall x \exists y \ \text{Friend}(x, y)$
      - '누구나 친구가 한 명은 있다'

술어 논리의 구문

- **함수**(function)
  - 주어진 인자에 대해서 참, 거짓 값이 아닌 일반적인 값을 반환
  - 술어나 다른 함수의 인자로 사용
- **항**(term)
  - 함수의 인자가 될 수 있는 것
  - 항이 될 수 있는 것 : 개체상수, 변수, 함수

(1) 개체상수, 변수는 항이다.

(2) $t_1, t_2, \cdots, t_n$이 모두 항이고, f가 n개의 인자를 갖는 함수 기호일 때, $f(t_1, t_2, \cdots, t_n)$은 항이다.

(3) (1)과 (2)에 의해 만들어질 수 있는 것만 항이다.

술어 논리의 구문

- **술어 논리식에 대한 정형식** (well-formed formula, **wff**)

(1) $t_1, t_2, \cdots, t_n$이 모두 항이고, p가 n개의 인자를 갖는 술어 기호일 때, $p(t_1, t_2, \cdots, t_n)$은 정형식이다.

(2) p와 q가 정형식이면, 논리 기호를 사용하여 구성되는 논리식 $\neg p$, $p \vee q$, $p \wedge q$, $p \rightarrow q$, $p \equiv q$도 정형식이다.

(3) $p(x)$가 정형식이고, x가 변수일 때, $\forall x\, p(x)$, $\exists x\, p(x)$는 정형식이다.

(4) (1), (2), (3)에 의해 만들어질 수 있는 것만 술어 논리의 정형식이다.

$\forall$x $\forall$y Horse(x) $\wedge$ Dog(y) $\rightarrow$ Faster(x,y)

$\exists$y Greyhound(y) $\wedge$ ($\forall$z Rabbit(z) $\rightarrow$ Faster(y,z))

Horse(Harry)

Rabbit(Ralph)

$\forall$y Greyhound(y) $\rightarrow$ Dog(y)

$\forall$x $\forall$y $\forall$z Faster(x,y) $\wedge$ Faster(y,z) $\rightarrow$ Faster(x,z)

술어 논리의 종류

- **일차 술어논리**(first-order predicate logic, **FOL**)
  - **변수**에만 전칭 한정사와 존재 한정사를 쓸 수 있도록 한 술어논리

- **고차 술어논리**(high-order predicate logic)
  - 변수뿐만 아니라 함수, 술어기호 등에 대해서 전칭 한정사와 존재 한정사를 쓸 수 있도록 한 술어논리
    - $\exists S\, S(x)$
    - $\exists g \forall x\, (f(x) = h(g(x))$

술어 논리의 지식표현

- **술어 논리를 이용한 지식 표현**

(a) Whoever can read is literate. (읽을 수 있으면 문맹이 아니다)

(b) Monkeys are not literate. (원숭이는 문맹이다)

(c) Some monkeys are intelligent. (어떤 원숭이는 지능적이다) ← **증명**

(d) Some who are intelligent cannot read. (지능적이어도 문맹일 수 있다)

(a) $\forall x[CanRead(x) \rightarrow Literate(x)]$

(b) $\forall x[Monkey(x) \rightarrow \neg Literate(x)]$

(c) $\exists x[Monkey(x) \wedge Intelligent(x)]$

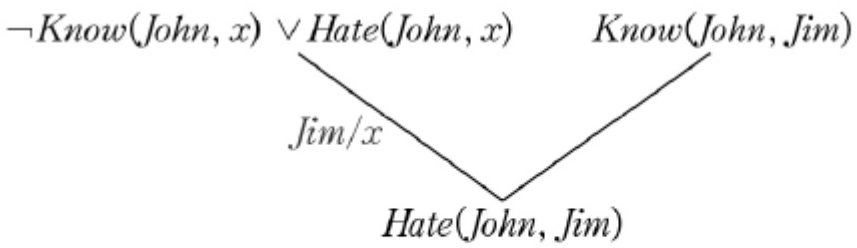
(d) $\exists x[Intelligent(x) \wedge \neg CanRead(x)]$

술어 논리의 추론

- **술어 논리식의 CNF로의 변환 과정**
  1. 전칭 한정사와 존재 한정사를 논리식의 맨 앞으로 끌어내는 변환
  2. **전칭 한정사**에 결합된 변수
     - 임의의 값 허용
  3. **존재 한정사**에 결합된 변수
     - 대응되는 술어 기호를 참(T)으로 만드는 값을 변수에 대응시킴
     - **스콜렘 함수(Skolem function)**
       - 존재 한정사에 결합된 변수를 해당 술어의 **전칭 한정사**에 결합된 다른 **변수**들의 새로운 **함수로 대체**
       - $\forall x \exists y [P(x) \wedge Q(x, y)] \Rightarrow \forall x [P(x) \wedge Q(x, s(x))]$
       - $s(x)$ - $Q(x, s(x))$를 어떤 x에 대해서도 참으로 만드는 마법의 함수(magic function)

술어 논리의 추론

- **단일화**(unification) 과정
  - 논리융합(resolution)을 적용할 때는 대응되는 리터럴이 같아지도록, 변수의 값을 맞춰주는 과정



단일화 과정의 예

x를 Jim으로 대체하면 논리융합에 의해 $Hate(John, Jim)$이 도출됨.

술어 논리의 추론

- **술어 논리로 지식의 증명**

(a) $\forall x[CanRead(x) \rightarrow Literate(x)]$

(b) $\forall x[Monkey(x) \rightarrow \neg Literate(x)]$

(c) $\exists x[Monkey(x) \wedge Intelligent(x)]$

(d) $\exists x[Intelligent(x) \wedge \neg CanRead(x)]$ ← 증명

- 논리곱 형태로 변환

(1) $\neg CanRead(x) \vee Literate(x)$

(2) $\neg Monkey(x) \vee \neg Literate(x)$

(3) $Monkey(A)$

(4) $Intelligent(A)$

(5) $\neg Intelligent(x) \vee CanRead(x)$

(d)를 **논리융합 논박**을 이용하여 증명하기 부정

- (1)과 (5)

(6) $\neg Intelligent(x) \vee Literate(x)$

- (2)과 (6)

(7) $\neg Intelligent(x) \vee \neg Monkey(x)$

- (3)과 (7)

(8) $\neg Intelligent(A)$

- (4)과 (8)

(9) $Intelligent(A) \wedge \neg Intelligent(A) \equiv nil$

(d)를 부정하여 모순이 발생하므로, (d)가 참임

술어 논리의 추론

- **술어 논리로 지식의 증명**

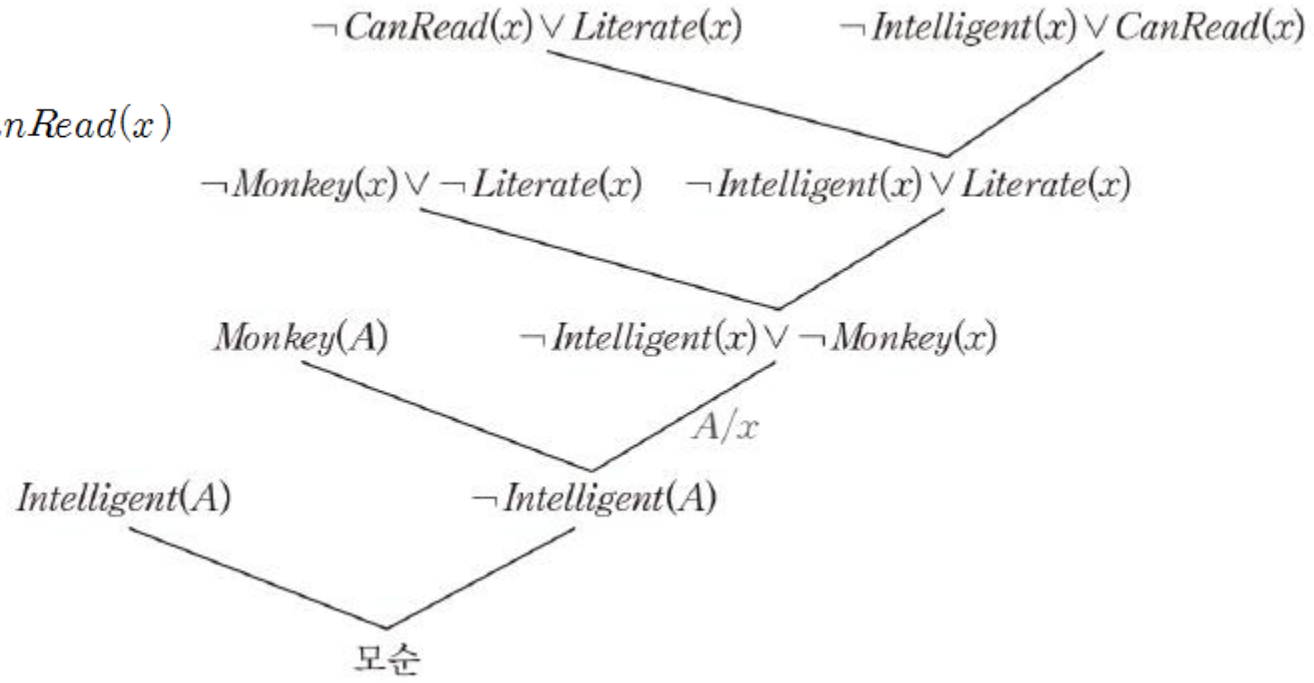
(1) $\neg CanRead(x) \vee Literate(x)$

(2) $\neg Monkey(x) \vee \neg Literate(x)$

(3) $Monkey(A)$

(4) $Intelligent(A)$

(5) $\neg Intelligent(x) \vee CanRead(x)$



논리 프로그래밍 언어

- **Horn 절 (Horn clause)**
  - 논리식을 논리합의 형태로 표현할 때, $\neg A(x) \vee \neg B(x) \vee C(x)$ 와 같이 긍정인 리터럴을 최대 하나만 허용
- **Prolog**
  - Horn 절만 허용하는 논리 프로그래밍 언어

```
father(noah, shem).
father(noah, ham).
father(shem, elam).
father(shem, arphaxad).
father(arphaxad, caina).
grandfather(X,Y) :- father(X,Z), father(Z,Y).
:- grandfather(X,Y).
```

  - 백트랙킹(backtracking)을 이용하여 실행