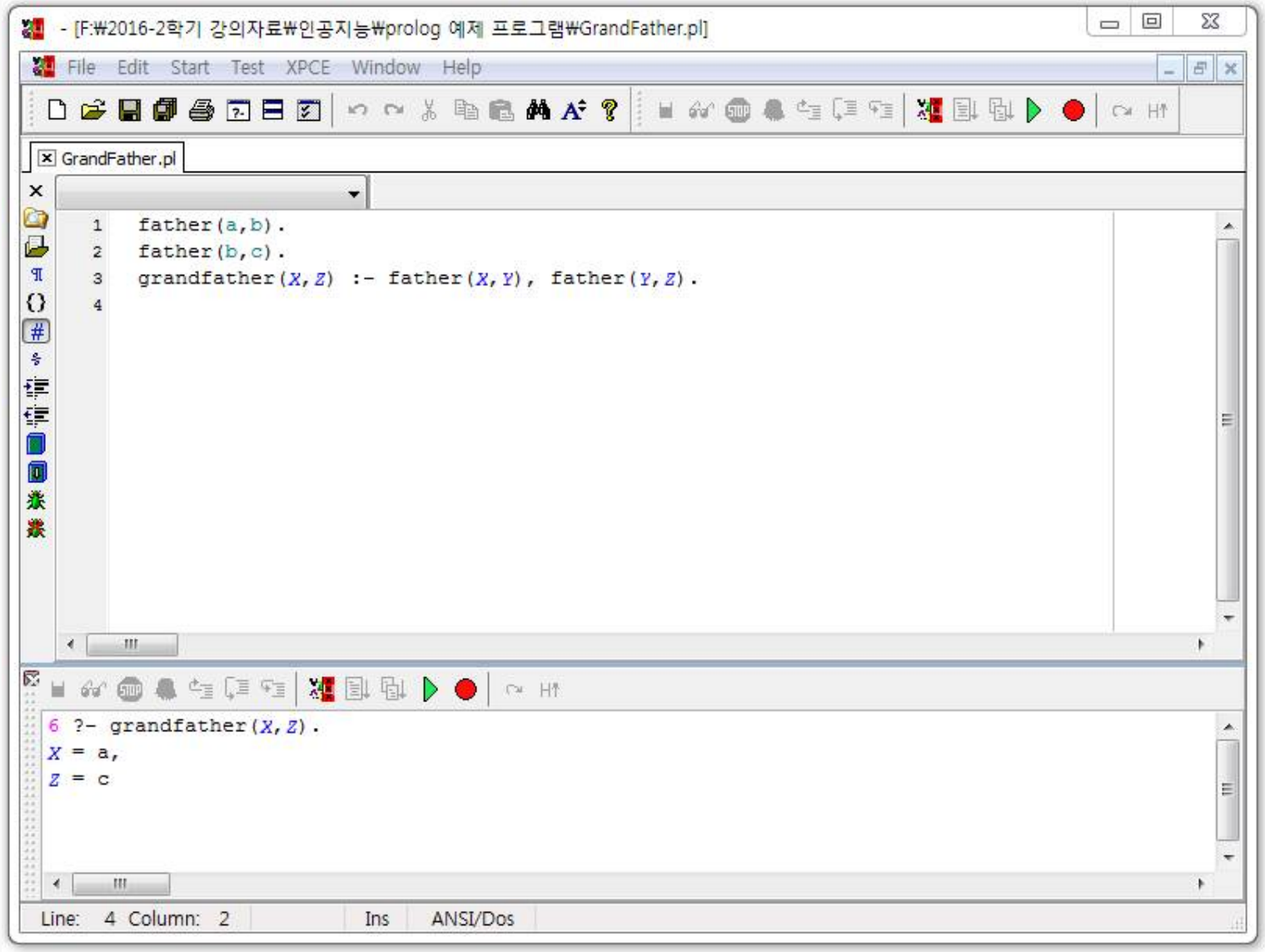
제7장. Prolog 언어

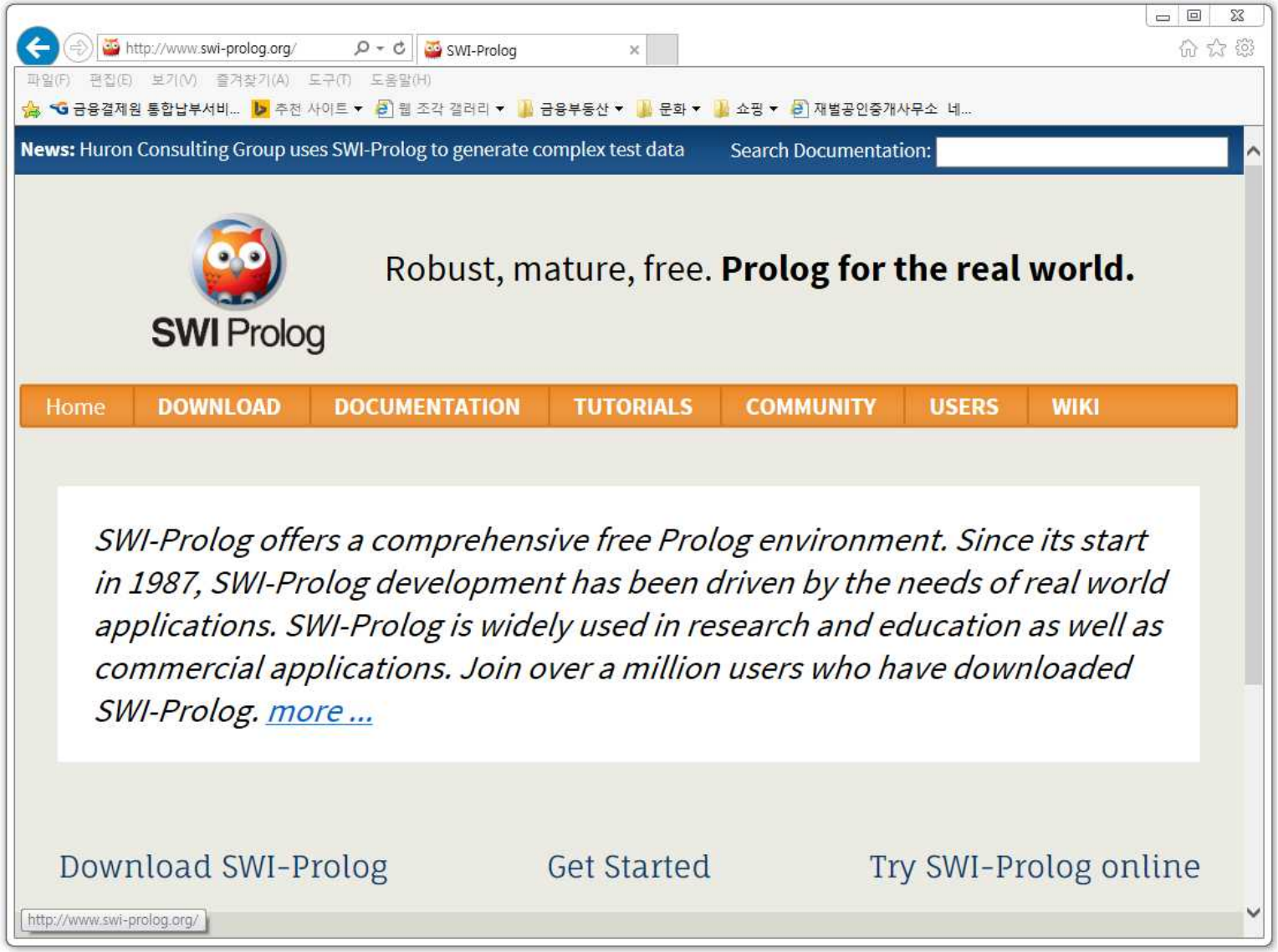
7.1 Prolog 소개

- 자연어 처리 시스템 구축을 위해 프랑스의 Colmeraurer에 의하여 1972년에 개발됨
- 1981년 일본이 5세대 컴퓨터 프로젝트에서 Prolog 언어를 주 언어로 채택함
  - Prolog 언어와 논리 프로그래밍은 국제적 관심을 모으기 시작함
- Prolog는 선언적 언어
  - 사실(fact)이나 규칙(rule) 등을 기술
  - 내장된 추론 기구에 의하여 문제해결

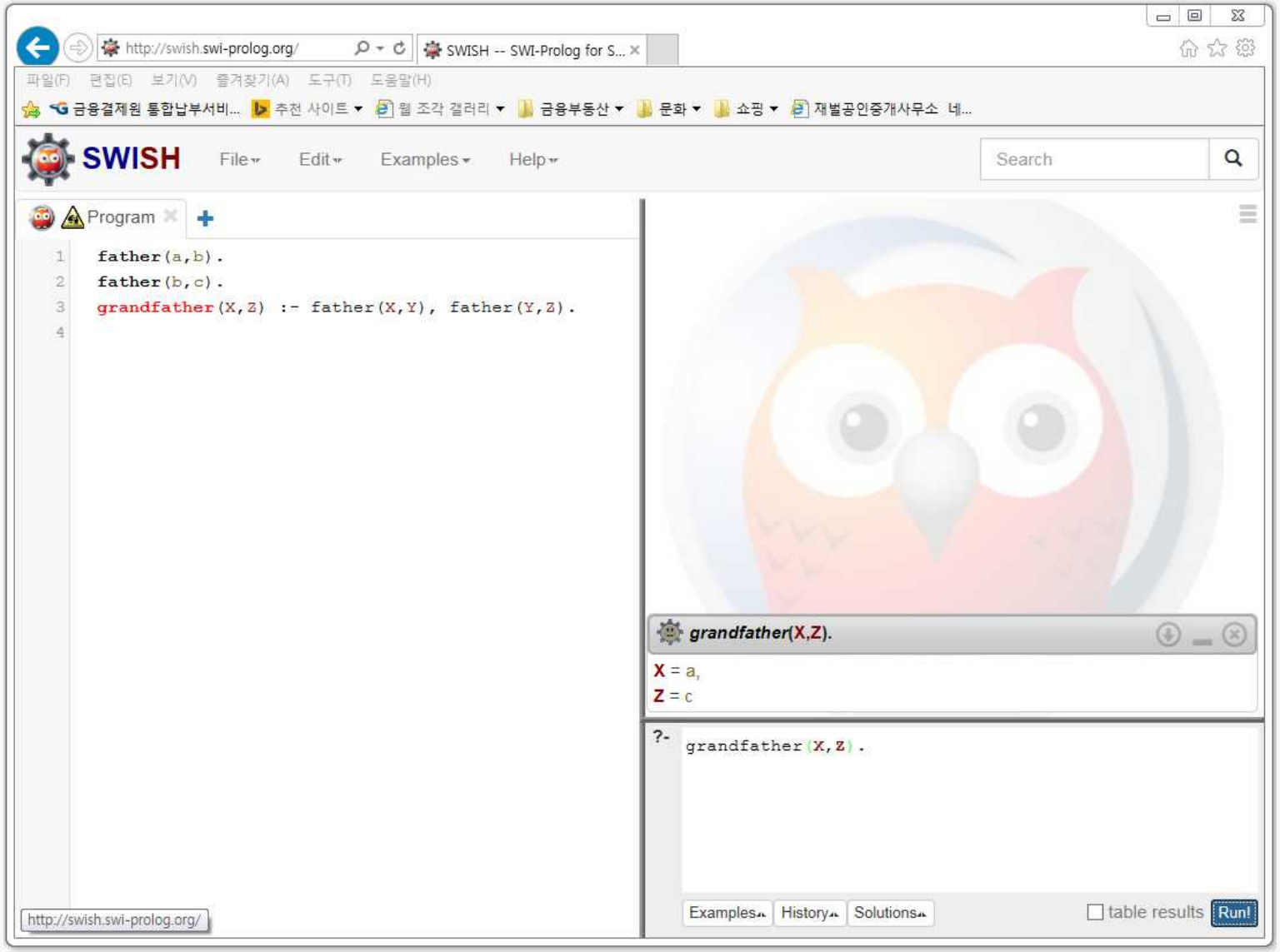
Prolog 개발환경1. swi-prolog (윈도우용)



Prolog 개발환경1 – swi-prolog (웹버전)



Prolog 개발환경1- swi-prolog (웹버전)



7.2 구성요소

- Prolog의 구성 요소를 이루는 기본적인 소자는 항(term)으로 이루어짐
  - 항은 상수(constant), 변수(variable), 구조(structure), 연산자(operator) 등으로 구성됨
  - 항은 연속적인 문자들로 이루어지며 다음과 같은 기호들을 사용할 수 있다.

```
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

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

0 1 2 3 4 5 6 7 8 9

+ - * / ₩ ^ < > ~ : . ? @ # $ &
```

7.2.1 상수

- 상수는 오브젝트(Object)나 오브젝트 사이의 관계를 나타내는 것으로, 원자(atom)와 숫자(number)의 두 가지 종류가 있다.
- 원자의 예

john	mary
likes	owns
book	wine
professor	student
can_work	sister_of

  - 원자는 알파벳, 숫자 특수 문자로 구성되며, **첫째 문자는 소문자로 시작**되어야 한다.
  - 대문자 혹은 숫자로 시작되는 원자를 만들 경우에는 인용 부호(‘ ’)를 사용
    - 인용 부호내의 문자열 전체는 한 개의 원자로 취급됨(예: ‘KIM’)

7.2.2 변수

- 변수는 **대문자 혹은 밑줄(_)로 시작**
  - 영문자와 밑줄(_), 0~9의 숫자로 표현
  - 변수의 예

```
X
Y
Value
Gross_Pay
_5_little_Indian
```

- 익명 변수(don't care): _

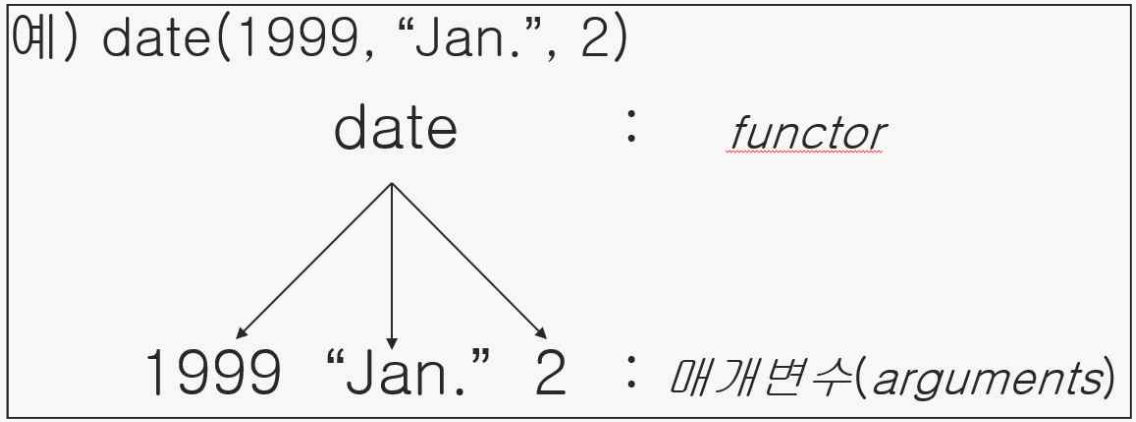
```
?- like(_, john).
```

  - John을 좋아하는 어떤 사람이 있는가를 알고 싶지만, 그사람이 정확히 누군지는 알 필요가 없는 경우에 익명 변수 사용

7.2.3 구조(Structures)

- 구조는 여러 개의 **매개변수**(arguments or component)와 **함수기호**(functor)로 구성되어 있음
  - 함수기호는 괄호 바로 앞에 쓰여지며, 매개변수는 소괄호 안에 기술되며 쉼표에 의해 구분됨

- 트리(tree)를 이용한 구조(structure)의 표현 예



7.2.3 구조(Structures)

- 사실 “John 이 책을 가지고 있다.”의 구조 표현

```
owns(john, book).
```

- 사실 “John이 Rogers가 쓴 Prolog 책을 가지고 있다.”의 구조 표현

```
owns(john, book(prolog, rogers)).
```

  - owns 안에 제목과 저자의 두 요소를 가지는 book이라는 명칭을 가지는 구조를 나타냄

- 학생 개인별 자료를 다음과 같은 구조 형태로 표시할 수 있다.

```
student(name(hong, kildong), stu_num(9760026), birthday(1979,4,21)).
```

7.2.4 연산자

- 산술연산자:
  - +, −, *, /, mod (예: X+Y, X−Y, X*Y, X/Y, X mod Y)
- 비교연산자:
  - = 또는 =:=, ₩= 또는 =₩=, <, >, =<, >=
- 대입가능 술어: /* C언어의 대입연산자와 비슷 */
  - is (예: "Z is X/Y"는 Z에 X/Y의 값을 저장)

```
X = Y          ; X와 Y는 같다.
X ₩= Y         ; X와 Y는 같지 않다.
X < Y          ; X는 Y보다 작다.
X > Y          ; X는 Y보다 크다.
X =< Y         ; X는 Y는 보다 작거나 같다.
X >= Y         ; X는 Y보다 크거나 같다.
```

7.2.4 연산자

Arithmetic	Prolog
2 + 3 = 5	?- 5 is 2+3.
3 x 4 = 12	?- 12 is 3*4.
5 – 3 = 2	?- 2 is 5-3.
3 – 5 = -2	?- -2 is 3-5.
4 : 2 = 2	?- 2 is 4/2.
1 is the remainder when 7 is divided by 2	?- 1 is mod(7,2).

7.2.4 연산자

```
?- 10 is 5+5.
yes

?- 4 is 2+3.
no

?- X is 3 * 4.
X=12
yes

?- R is mod(7,2).
R=1
yes
```

7.2.4 연산자

```
?- X = 3 + 2.
X = 3+2
yes

?- 3 + 2 = X.
X = 3+2
yes

?-
```

7.2.4 연산자

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- Result is 2+2+2+2+2.
Result = 10
yes

?-
```

7.2.4 연산자

```
?- is(X,+(3,2)).
X = 5
yes
```

7.2.4 연산자

```
?- 2 < 4+1.
yes

?- 4+3 > 5+5.
no
```

7.2.4 연산자

```
?- 4 = 4.
yes

?- 2+2 = 4.
no

?- 2+2 =:= 4.
yes
```

7.2.4 연산자

```
addThreeAndDouble(X, Y):-
    Y is (X+3) * 2.
```

```
?- addThreeAndDouble(1,X).
X=8
yes

?- addThreeAndDouble(2,X).
X=10
yes
```

7.3 기본문

- Prolog의 기본문은 세 가지로 구성된다.
  - 첫째는 오브젝트와 그들의 관계에 관한 **사실**(facts)
  - 둘째는 오브젝트들과 그들의 관계에 대한 **규칙**(rules)
  - 셋째는 오브젝트와 그들의 관계에 관한 **질문**(question)

7.3.1 사실(facts)

- “John likes Mary.”의 사실 표현
  - “likes (john, mary).”
    - 이것은 두 개의 대상(Object)와 하나의 관계로 구성됨

- 사실의 선언
  - 대상과 관계의 이름은 **소문자**로 시작
  - 관계가 앞에 나오고, 대상은 괄호 안에 묶어서 표시
  - 두 개 이상의 대상은 쉼표(,)로 구분
  - 사실의 마지막에는 마침표(.)가 옴

- 사실을 이용하여 대상들 사이의 관계를 정의할 때는 괄호 안에 나열되는 대상들의 순서에 주의
  - 예를 들면, “likes(john, mary)”와 “likes(mary, john)”은 서로 의미가 다름

참고: 사실(Facts)과 규칙(Rules)의 표현

- Prolog의 사실과 규칙은 논리식의 체계를 도입하여 표현하는데 그 기호는 다음과 같다.

PROLOG	논리식
,	and(∧)
;	or(∨)
:-	only if(←)
not	not(¬)

- 위의 기호를 사용한 표현 예)

likes(kim, pinky).

⇒ “kim은 pinky를 좋아한다”라는 fact의 표현

likes(X, pinky).

⇒ “변수 X는 pinky를 좋아한다”

참고: 사실(Facts)과 규칙(Rules)의 표현

- 위의 기호를 사용한 표현 예)

  likes(kim, X), likes(lee, X).

  ⇒ “kim이 X를 좋아하고 lee도 X를 좋아한다”

  likes(kim, X); likes(lee, X).

  ⇒ “kim이 X를 좋아하거나 lee는 X를 좋아한다”

  likes(kim, pinky):- likes(lee, pinky).

  ⇒ “만일 lee가 pinky를 좋아한다면 kim은 pinky를 좋아한다”

  likes(kim, pinky):- not(likes(lee, pinky)).

  ⇒ “만일 lee가 pinky를 좋아하지 않는다면 kim은 pinky를 좋아한다”

7.3.2 질문(question) 또는 목표(goal)

- 질문은 내부에 저장되어 있는 데이터베이스를 탐색
  - 두 개의 사실이 일치하면 yes, 그렇지 않으면 no로 응답함
- 질문의 형태
  - 물음표(?)와 하이폰(-)으로 구성됨 => “?-”
- 사실과 질문의 예

```
likes(jim, fish).
likes(jim, mary).
likes(jim, flowers).
likes(mary, books).
likes(mary, flowers).
?- likes(john, money).
no
?- likes(mary, books).
yes
?- likes(mary, X), liks(jim, X).
```

7.3.2 질문(question) 또는 목표(goal)

- 두 개 이상의 목표(goal)들을 결합하기 위해서
  - 목표들 사이에 쉼표(,)를 두며, 이를 “그리고(and)”라고 읽음
- 두 개 이상의 목표를 가지는 질문을 할 때
  - 목표 달성의 도중에 실패할 경우 백트래킹(backtracking)이 일어남

```
likes(mary, books).
likes(mary, flowers).
```

- 예를들어, “?- likes(mary, X), likes(jim, X). ”라는 질문을 했을 경우
  - 먼저 첫 번째 목표인 likes(mary, X)를 만족하는 사실을 데이터베이스에서 찾아 X=books로 X값을 대체하고, 두 번째 목표인 likes(jim, books)를 데이터베이스에서 찾음
  - 여기에서 실패하게 됨으로 X=books라는 X의 값을 버리고 다시 첫 번째 목표로 되돌아 가서, 첫 번째 목표인 likes(mary, X)를 만족하는 X=flowers로 값을 대체하고 두 번째 목표인 likes(jim, flowers)를 찾음.
  - 여기에서 성공하게 되므로 Prolog는 X=flowers를 응답하게됨

7.3.3 규칙(rule)

- 규칙은 머리(head)와 몸체(body)로 구성됨
  - 머리는 어떤 사실을 정의하고자 하는가를 표현
  - 몸체는 머리 부분이 참이 되기 위해 만족되어야 할 목표들을 기술
  - 머리와 몸체는 콜론(:)과 하이픈(-)으로 구성된 ":-" 기호로 나타내며, 규칙도 사실과 마찬가지로 마침표(.)로 끝남

- 규칙의 예1
  - "John likes anyone who likes wine." 이라는 문장에 대하여 "John likes X if X-likes wine."와 같이 변수를 사용할 수 있음
  - 이를 규칙으로 표현하면 "likes(john, X) :- likes(X, wine)."
    - 머리는 likes(john, X), 몸체는 likes(X, wine)

7.3.3 규칙(rule)

- 실습 예제1)

소크라테스(socrates)는 사람이다.

모든 사람(human)은 죽는다(mortal).

소크라테스는 죽는다.

- Prolog 표현

```
human(socrates).
mortal(X) :- human(X).

?- mortal(socrates).
true
```

7.3.3 규칙(rule)

- 규칙의 예2
  - “X가 Y의 누나가 되려면 X가 여성, Y가 남성이고, X가 Y보다 나이가 많아야 하며, X와 Y의 부모가 동일인이어야 한다.”라는 규칙

```
sister_of(X,Y) :- female(X), male(Y), older(X,Y), parent(X,P), parent(Y,P).
```

- 규칙 내에서 하나의 변수가 여러 번 나타날 경우에 이들은 똑같은 대상을 나타냄
  - ?- sister_of(alice, edward)라는 질문의 경우
    - 규칙 내에 있는 모든 X,Y에 대하여 X=alice, Y=edward로 대체됨
  - ?- sister_of(alice, Z)라는 질문의 경우
    - X=alice만을 대체하고, Y,P는 그대로 둔 채 네 개의 목표를 만족하는 Y,P의 값을 찾음
    - 만약 질의에 성공했을 경우 Y의 값을 Z에 대체하여 Z=Y의 값을 산출
- 규칙을 정의할 때 주석은 “/* ..... */”를 이용

7.3.3 규칙(rule)

- 실습 예제2)

홍길동(hong)은 운동선수(player)이다.
홍길동은 지성적(intelligent)이다.
건강(healthy)하고 지성적인 사람은 성공(succeed)한다.
모든 운동 선수는 건강하다.

- Prolog 표현

```
player(hong).
intelligent(hong).
succeed(X) :- healthy(X), intelligent(X).
healthy(X) :- player(X).

?- succeed(hong).
True
```

7.4 패턴 인식

- 상수와 상수는 두 값이 같은 경우, 패턴이 일치한다고 함

```
university = university
high_school ≠ elementary_school
1999 = 1999
2000 ≠ 2002
```

- 변수와 상수의 경우는 아무런 값이 주어지지 않은 변수와 상수는 항상 일치함

```
University = seoul_university
Variable = 1996
```

7.4 패턴 인식

- 상수 값을 가지고 있는 변수와 상수의 일치는 상수와 상수의 일치의 경우와 같음
  - Variable 변수가 1234 라는 값을 갖는 경우 Variable = 1234는 일치함
- 변수 New_var이 아무런 값을 가지지 않고, Variable이 2002 값을 가질 경우
  - New_var = Variable이 일치함
- 변수와 변수는 항상 일치함

```
X = Y
```

7.4 패턴 인식

- 두 구조가 같기 위해서
  - 함수 이름과 요소의 수가 같아야 하며, 또한 대응하는 모든 구성 성분들도 같아야 함

```
father(bill, john) = father(bill, john)
```

  - 예)

```
father(bill, CHILD) = father(FATHER, john)
```

    - FATHER = bill, CHILD = john에 일치하므로 전체가 일치하게 됨

7.5 Prolog의 동작

- Prolog의 동작은 도출 원리에 의한 정리 증명에 기반함
  (도출의 제어 전략은 깊이 우선 탐색을 기본으로 함)

  ① 우선 목표절의 리터럴 P_1이 증명되어야 할 목표로서 설정됨

  ② P_1과 단일화가 가능한 머리(head)를 갖는 사실 또는 규칙이 탐사됨

  ③ 만약 P_1이 사실과 패턴 일치를 할 경우,

  P_1은 증명되었으므로 다음의 목표 P2를 증명하게 됨

  ④ 만약 P_1이 어떤 규칙의 머리와 패턴 일치를 할 경우,

  그 규칙의 몸체(body)를 P_1에 대한 부목표로서 설정하고 이들의 부목표를 왼쪽부터 순서대로 ①~④의 순서에 따라 증명해야함.

  증명에 성공하면 P_1은 증명되었다고 하고 다음의 목표 P_2를 증명하게 됨

  ⑤ P_1의 증명이 실패하면 목표의 성립은 기각됨

  ⑥ P_1~P_n까지의 모든 것이 증명되면 성공하게 됨

7.5 Prolog의 동작

- [예제1]

```
dog(john).                              … (a)
cat(mary).                              … (b)
monkey(james).                          … (c)
friendly(john, james).                  … (d)
friendly(X, Y) :- dog(X), cat(Y).       … (e)

?- friendly(X, mary).                   … (f)
```

  - (a) ~ (d)가 사실, (e)가 규칙, (f)가 목표로서 주어졌을 때 다음과 같은 추론이 일어남
    1) (f)와 단일화 가능한 머리 부분을 갖는 절을 찾음
    2) (d)와는 패턴이 일치하지 않지만 (e)와는 Y=mary를 대입하면 패턴이 일치하게 됨
    3) 그 결과 (g)가 (f)의 부목표로 설정됨

```
?- dog(X), cat(mary).                   … (g)
```

    - 부목표(g)의 제 1목표 dog(X)는 X=john을 대입함으로써 (a)와 패턴이 일치하게 된다. (g)의 제2 목표 cat(mary)는 사실 (b)와 패턴이 일치한다. 원래의 목표 (f)의 성립이 증명됨

7.5 Prolog의 동작

- [예제2]

```
경부선의 시각표가 다음과 같은 사실로서 주어져 있다고 하자.
시각표(서울, 대전, 800, 930).
시각표(서울, 대전, 900, 1030).
시각표(대전, 광주, 940, 1240).
시각표(대전, 광주, 1040, 1340).
```

  - 여행 규칙은 다음과 같이 표현된다.

```
계획(출발역, 도착역, 출발시각, 도착시각)
        :- 시각표(출발역, 도착역, 출발시각, 도착시각).

계획(출발역, 도착역, 출발시각, 도착시각)
        :- 시각표(출발역, 환승역, 출발시각,환승역 도착시각),
           시각표(환승역, 도착역, 환승역 출발시각, 도착시각),
            >(환승역 출발시각, 환승역 도착시각).
```

  - ?- 계획(서울, 광주, 출발시각, 도착시각), <(도착시각, 1300).

- 실행결과: 출발시각 = 800, 도착시각 = 1240

7.5 Prolog의 동작

- [예제2]의 Prolog 코드

```
time_table(seoul, daejeon, 800, 930).
time_table(seoul, daejeon, 900, 1030).
time_table(daejeon, gangju, 940, 1240).
time_table(daejeon, gangju, 1040, 1340).

plan(S_Station, A_Station, S_Time, A_Time)
:- time_table(S_Station, T_Station, S_Time, T_A_Time),
   time_table(T_Station, A_Station, T_S_Time, A_Time),
    >(T_S_Time, T_A_Time).

?- plan(seoul, gangju, S_Time, A_Time), <(A_Time, 1300).
A_Time = 1240,
S_Time = 800
```

7.6 백트랙킹

- 미로의 문제를 해결하려고 하는 경우
  - 선택한 길이 막힐 경우가 발생하곤 한다. 이때에는 갔던 길을 다시 돌아 나와서, 다른 길을 선택하여 다시 목표 지점을 찾아 가게된다. => 이를 백트랙킹이라 함.

- Prolog 프로그램의 실행에서 백트랙킹
  - 어떤 목표를 호출할 때, 일치 하는 규칙이 있는 경우 우선 그 중 하나(순서대로 검색하여 처음 발견된 것)을 선택하고 그 몸체를 실행한다. 만약 목표를 호출하여도 더 이상 일치하는 주장이 없으면, 다른 주장을 시험하게 된다
  - 간단한 데이터베이스로부터 정보를 추출하는 경우
    - 하나의 조건을 만족하는 정보를 찾는다면 간단하지만, 두 개 이상의 조건을 만족하는 것을 찾을 때에는 백트랙킹이 발생할 수 있음

7.6 백트랙킹

- 예)

```
color(sky, blue).
color(leaf, green).
color(moon, yellow).
color(apple, red).
color(banana, yellow).
fruit(strawberry).
fruit(banana).
fruit(orange).
```

  - 위와 같은 사실에 대하여 다음과 같은 질문이 주어진다고 가정

```
?- color(X, yellow), fruit(X).
```

    - 위와 같이 목표가 두 개 이상일 경우에는 왼쪽부터 순서대로 실행함

7.6 백트랙킹

```
color(sky, blue).
color(leaf, green).
color(moon, yellow).
color(apple, red).
color(banana, yellow).
fruit(strawberry).
fruit(banana).
fruit(orange).
```

```
?- color(X, yellow), fruit(X).
```

1) color(X, yellow)를 실행하면, color(moon, yellow)가 발견됨[선택점 1].
2) 다음에 fruit(moon)을 실행하지만 이것은 일치하는 것이 없기 때문에[선택점 1]까지 백트랙킹하여 다른 해 color(banana, yellow)를 발견한다[선택점 2].
3) 이번에는 fruit(banana)를 발견하여 성공한다.

- 백트랙킹을 이용하여 정보를 검색하는 경우에는 프로그램이 정지하지 않을 경우가 발생할 수 있기 때문에 주의해야 함

7.7 컷(cut)

- 컷이란
  - 백트랙킹을 통해 다른 해를 구하는 것을 막는 것
  - 컷이라는 술어를 사용하며 “!”로 표기함

- 컷의 적용 여부에 따른 프로그램의 동작 원리
  - 컷을 적용하지 않은 경우의 예. (요소로 부터 짝수 찾기)

```
member_of (X, [1,3,4,5,12]), even(X).
```

1) 최초로 member_of(X,[1,3,4,5,12])가 실행되면 X=1 이 발견됨
2) even(1) 실행되지만 1이 짝수가 아니므로 실패
3) 백트랙킹 하여 다음의 해 X=3 이 발견, even(3)실행, 홀수 임으로 실패
4) 백트랙킹 하여 다음의 해 X=4 발견, even(4) 실행, 짝수 임으로 성공

7.7 컷(cut)

- 컷의 적용 여부에 따른 프로그램의 동작 원리
  - 컷을 적용한 경우의 예. (요소로 부터 짝수 찾기)

```
member_of (X, [1,3,4,5,12]), !, even(X).
```

1) 최초로 member_of(X,[1,3,4,5,12])가 실행되면 X=1 이 발견됨
2) even(1) 실행되지만 1이 짝수가 아니므로 실패
3) !로 인해 백트랙킹을 하지 못하고 전체가 실패로 끝남

★ !는 왼쪽부터 오른쪽을 일방 통행하는 성질을 가지고 있음

7.8 재귀적 구조

- 재귀적 구조는 규칙의 머리가, 같은 규칙의 몸체에도 나타나는 구문 구조를 의미함
  - 어느 특정한 프로세스가 수행 중에 그 프로세스를 반복하여 호출하는 것

- 재귀적 구조의 예: factorial

C언어	Prolog 언어

int factorial(int n)
{
if(n <=1) return;
else return (n* factorial(n-1));
}
	factorial(0,1) :- !. /* 컷 사용 */
	factorial(N, A) :-
	N > 0, /* 생략가능 */
	N1 is N-1,
	factorial(N1, A1),
	A is N * A1.

factorial(3);	?- factorial(3, Ans). Ans=6

실전 예: 감기, 독감 진단 프로그램

- 감기와 독감의 증세
  - 감기: 재채기, 기침, 콧물/코막힘 증세
  - 독감: 39도 이상의 고열과 심한 근육통 증세

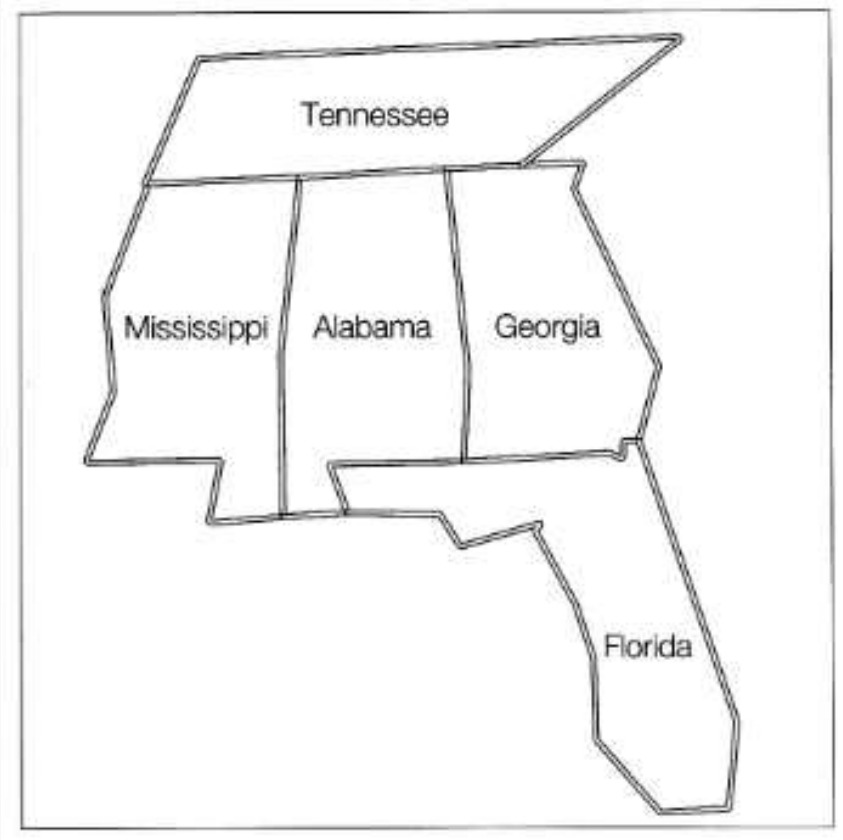
```
sneeze(me).
cough(me).
intenseHeat(me).
runnyNose(me).
nosalCongestion(kim).
muscularAche(kim).

cold(X) :- sneeze(X), cough(X), runnyNose(X).
cold(X) :- sneeze(X), cough(X), nosalCongestion(X).
flu(X) :- intenseHeat(X), muscularAche(X).

?- flu(me).
false
?- cold(me).
true
```

실전 예: 지도 색칠하기

- 지도 색칠하기
  - 조건: 서로 접하고 있는 지역끼리는 모두 다른 색으로 칠함
  - 색상: red, green, blue



실전 예: 지도 색칠하기 (계속)

- 지도 색칠하기: Prolog 코드

```
different(red, green). different(red, blue).
different(green, red). different(green, blue).
different(blue, red). different(blue, green).

coloring(Alabama, Mississippi, Georgia, Tennessee, Florida) :-
    different(Mississippi, Tennessee),
    different(Mississippi, Alabama),
    different(Alabama, Tennessee),
    different(Alabama, Mississippi),
    different(Alabama, Georgia),
    different(Alabama, Florida),
    different(Georgia, Florida),
    different(Georgia, Tennessee).
```


```
?- coloring(Alabama, Mississippi, Georgia, Tennessee, Florida).
Alabama = blue,
Florida = Tennessee, Tennessee = green,
Georgia = Mississippi, Mississippi = red
...
```

7.9 리스트

- 리스트
  - 원소들의 순서열을 나타내는 자료구조
    - 길이에 제한이 없음
    - 리스트의 원소는 상수, 변수, 구조, 다른 리스트 가 될 수 있음
    - 리스트의 예

```
[]    // 원소가 없는 리스트 => 공리스트
[1, 2, 3, 4, 5]
[서울, 부산, 인천, 대구, 광주, 대전]
[[1,2], 3, [4,5]]
[[경기도, 수원], [강원도, 춘천], [제주도, 제주]]
```

7.9 리스트

리스트(계속)

- 리스트의 머리와 꼬리 [Head|Tail]
  - 머리: 리스트의 첫 원소
  - 꼬리: 머리를 제외한 나머지 원소들

[리스트의 머리와 꼬리 예]

리스트	머리	꼬리	[x | y]형태
[a, b, c]	a	[b, c]	[a | [b,c]]
[a]	a	[ ]	[a | [ ]]
[[a, b], c]	[a, b]	[c]	[[a, b] | [c]]
[a, [b, c]]	a	[[b, c]]	[a | [[b, c]]]
[a, [b, c], d]	a	[[b, c], d]	[a | [[b, c], d]]

- 리스트 [a,b,c,d,e]에 대하여
  - [X \| L]을 패턴 일치시키면
    → X=a, L=[b,c,d,e]를 얻음
  - [X,Y \| L]을 패턴 일치시키면
    → X=a, Y=b, L=[c,d,e]를 얻음

7.9 리스트

- 리스트의 원소 출력

```
number_of(_,[]) :- !.
number_of(_,[Head|Tail]):- writeln(Head), number_of(_,Tail).
```

```
?- number_of(_,[1,3,4,5,12]).
1
3
4
5
12
```

7.9 리스트

- 리스트에서 특정 원소 찾기(1)

```
number_of(X,[X|_]).
number_of(X,[_|Y]) :- number_of(X,Y).
```

```
?- number_of(3,[1,3,4,5,12]).
true

?- number_of(3,[1,3,4,3,12]).
true
true

?- number_of(7,[1,3,4,5,12]).
False
```

7.9 리스트

- 리스트에서 특정 원소 찾기(2)

```
number_of(X,[X|_]) :- !.
number_of(X,[_|Y]) :- number_of(X,Y).
```

```
?- number_of(3,[1,3,4,5,12]).
true

?- number_of(3,[1,3,4,3,12]).
true

?- number_of(7,[1,3,4,5,12]).
false
```

7.9 리스트

- 리스트에서 짝수만 골라내는 프로그램

```
number_of(X,[X|_]) :- 0 is X mod 2.
number_of(X,[_|Y]) :- number_of(X,Y).
```

```
?- number_of(X,[1,3,4,5,12]).
X = 4
X = 12
```

실전 예: 세 개의 정수의 합과 평균

```
count(0,[]).
count(Count,[_|Tail]):-
        count(TailCount, Tail), Count is TailCount + 1.

sum(0,[]).
sum(Total, [Head|Tail]):-
       sum(Sum,Tail), Total is Head + Sum.

average(Average, List):-
       sum(Sum,List), count(Count,List), Average is Sum/Count.

?- count(Cnt,[10, 20,30]).
Cnt = 3
?- sum(Sum,[10, 20,30]).
Sum = 60
?- average(Avg, [10, 20,30]).
Avg = 20
```

[참고] 선택구조

- CASE나 IF-THEN-ELSE와 비슷한 기능을 갖는 프로그램
  - 예) 양수, 음수, 0 을 판단하는 프로그램

```
classify(0, zero).
classify(X, negative) :- X < 0.
classify(X, positive) :- X > 0.

?- classify(5,What).
What = positive
```

[참고] 선택구조

- 짝수/홀수 판단 프로그램

```
even(X, even):-
        N is X mod 2, N=0.
even(X, odd):-
        N is X mod 2, N=1.

?- even(4,What).
What = even
?- even(3,What).
What = odd
```

7.10.2 입출력문

```
hello_world :-
      writeln('Hello World!'),
      sleep(1),
      hello_world.

read_and_write :-
       read(Something),
       writeln(Something),
       read_and_write.

?- hello_world.
Hello Wordl!
Hello Wordl!
...

?- read_and_write.
```

[참고] 선택구조

- 짝수/홀수 판단 프로그램

```
even(X):-
        N is X mod 2, N=0, writeln("짝수").
even(X):-
        N is X mod 2, N=1, writeln("홀수").

?- even(4).
짝수
?- even(3).
홀수
```

[참고] 선택구조

- 임의의 값을 입력하여 짝수와 홀수를 판단하는 프로그램

```
start:-
    writeln(“### 짝수와 홀수를 판단하는 프로그램 ###"),
    write("임의의 값 X를 입력하시오: "),
    read(X), even(X), start.

even(X):-
        N is X mod 2, N=0, writeln("'짝수'"),nl.
even(X):-
        N is X mod 2, N=1, writeln("'홀수'"),nl.

?- start.
```

실전 예: 동물 식별 프로그램

- 139page 동물 식별 프로그램

P1: 만약 동물이 체모를 갖고 있다면, 그 동물은 포유동물이다.
P2: 만약 동물이 젖으로 새끼를 기른다면, 포유동물이다.
P3: 만약 동물이 날개를 갖고 있으면, 새이다.
P4: 만약 동물이 날 수 있고, 알을 낳는다면 새이다.
P5: 만약 동물이 포유동물이고, 굽을 갖고 있다면 유제동물이다.
P6: 만약 동물이 포유동물이고, 육식을 한다면 육식동물이다.
P7: 만약 동물이 유제동물이고, 흰색과 긴 목을 가지고 있다면 기린이다.
P8: 만약 동물이 유제동물이고, 흰색과 검은 색 줄무늬를 갖고 있다면 얼룩말이다.

1. 체모를 가지고 있다.
2. 굽을 갖고 있다.
3. 흰색과 검은 색의 줄무늬를 갖는다.

실전 예: 동물 식별 프로그램

- 139page 동물 식별 prolog 프로그램

```
positive(has,hair).              /*체모를 가지고 있다 */
positive(has,hooves).            /*굽을 가지고 있다 */
positive(has,black_stripes).  /*흰색과 검은색 줄무늬를 가지고 있다 */

animal_is(giraffe):-
   it_is(ungulate),
   positive(has,long_neck), !.

animal_is(zebra):-
   it_is(ungulate),
   positive(has,black_stripes), !.

it_is(mammal) :-
   positive(has,hair).

it_is(mammal) :-
   positive(does,give_milk).
```

1. 체모를 가지고 있다.
2. 굽을 갖고 있다.
3. 흰색과 검은 색의 줄무늬를 갖는다.

P1: 만약 동물이 체모를 갖고 있다면, 그 동물은 포유동물이다.
P2: 만약 동물이 젖으로 새끼를 기른다면, 포유동물이다.
P3: 만약 동물이 날개를 갖고 있으면, 새이다.
P4: 만약 동물이 날 수 있고, 알을 낳는다면 새이다.
P5: 만약 동물이 포유동물이고, 굽을 갖고 있다면 유제동물이다.
P6: 만약 동물이 포유동물이고, 육식을 한다면 육식동물이다.
P7: 만약 동물이 유제동물이고, 흰색과 긴 목을 가지고 있다면 기린이다.
P8: 만약 동물이 유제동물이고, 흰색과 검은 색 줄무늬를 갖고 있다면 얼룩말이다.

실전 예: 동물 식별 프로그램

- 139page 동물 식별 prolog 프로그램

```
it_is(bird) :-
   positive(has,feathers).

it_is(bird) :-
   positive(does,fly),
   positive(does,lay_eggs).

it_is(ungulate) :-
   it_is(mammal),
   positive(has,hooves).

it_is(carnivore) :-
   it_is(mammal),
   positive(does,eat_meat).

?- animal_is(X).
X = zebra
```

P1: 만약 동물이 체모를 갖고 있다면, 그 동물은 포유동물이다.
P2: 만약 동물이 젖으로 새끼를 기른다면, 포유동물이다.
P3: 만약 동물이 날개를 갖고 있으면, 새이다.
P4: 만약 동물이 날 수 있고, 알을 낳는다면 새이다.
P5: 만약 동물이 포유동물이고, 굽을 갖고 있다면 유제동물이다.
P6: 만약 동물이 포유동물이고, 육식을 한다면 육식동물이다.
P7: 만약 동물이 유제동물이고, 흰색과 긴 목을 가지고 있다면 기린이다.
P8: 만약 동물이 유제동물이고, 흰색과 검은 색 줄무늬를 갖고 있다면 얼룩말이다.