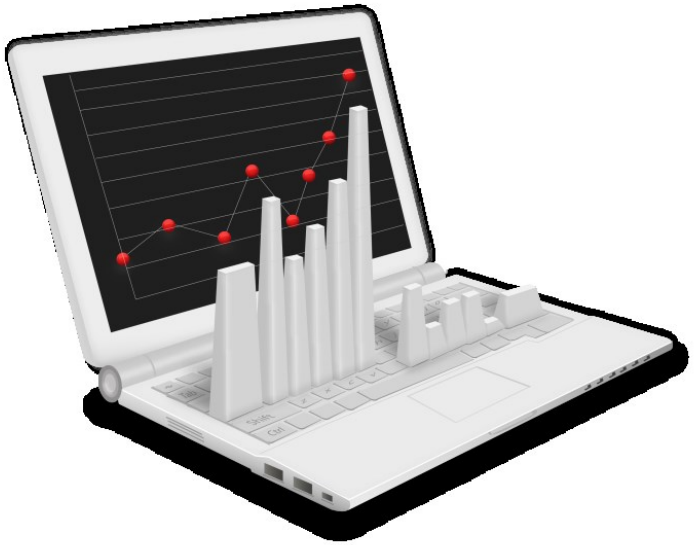
프로그래밍 언어론

논리형 프로그래밍

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

논리형 프로그래밍

학습목표

- 논리형 프로그래밍 언어를 사용한 프로그래밍 방법을 배운다.

학습내용

- 선언적 프로그래밍 방법
- backtracking(되추적) 및 unification(단일화) 를 사용하는 Prolog 프로그램의 수행과정

목 차

알고가기

다음과 같은 알고리즘에 대한 설명으로 틀린 것은?

```
function f(int i)
    if (i <= 0) return 1
    else return i * f(i-1)
```

(1) 재귀적 (recursive) 함수 호출을 사용했다.
(2) i의 i 제곱을 나타낸다.
(3) i가 100이면 f가 100번이상 호출된다.
(4) i의 값은 매 호출 마다 다른 값이다.

확인

| 선언적 프로그래밍

논리형 프로그래밍은 선언적 프로그래밍이다.

- 어떻게 수행하는 지를 기술할 수 없다.
- 각종 **fact, rule** 과 **query** 즉, 무엇이 필요한지만 기술 한다.

예

```
?-grandparent(X,세종)
```

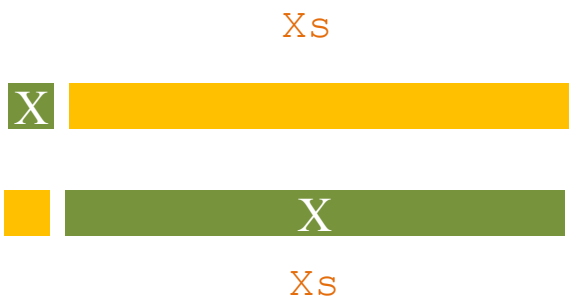
➔ *즉, 알고 싶은 것이 무엇만을 기술한다.*

("세종의 조부모가 누군지?")

프로그램 예 1

리스트에 특정 원소가 있는지 검사하는 프로그램

```
member(X, [X | _] ).
member(X, [_ |Xs]) :- member(X, Xs)
```




```
?- member(wed, [mon, wed, fri]
yes
?- member(X, [mon, wed, fri]).
X = mon;
X = wed;
X = fri;
no
```

프로그램 예 2

두 리스트를 병합하는 프로그램

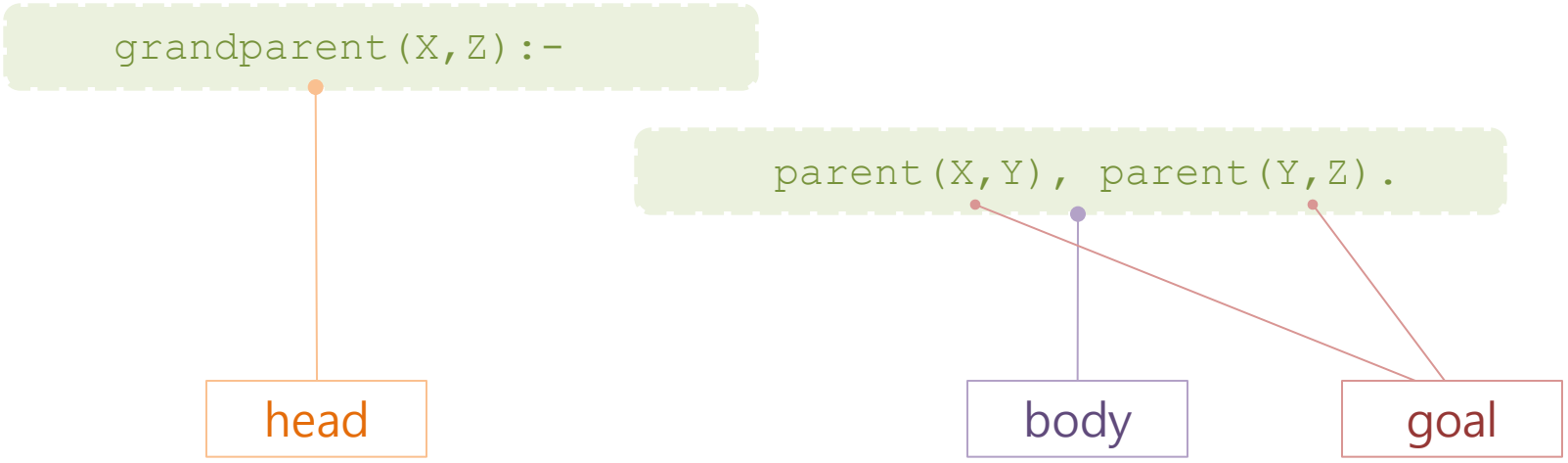
```
cat([], Z, Z).

cat([H|T], L, [H|Z]) :- cat(T, L, Z)
```

```
?- cat([mon, tue, wed], [thu,
fri, sat, sun], X)
```

X = [mon, tue, wed, thu, fri, sat, sun]

no

Prolog의 수행 과정



- body에 있는 각 부분을 다시 goal로 삼고 답을 구한다. **(subgoal이라 한다.)**
- **H:-G1, G2… Gn**

선언적 의미 : "G1, G2, … Gn 이 사실이면 H도 사실이다.

수행 과정 : "함수 H를 실행하려면 함수 G1, G2…Gn을 먼저 실행해야한다.

Backtracking (되추적)

```
member(X, [X | _] ).
member(X, [_ |Xs]) :- member(X, Xs)


?- member(wed, [mon, wed, fri]
```

query에 대응되는 rule이 여러 개 있으면 하나씩 "적용"한다.

- 위 예에서 첫번째 rule을 "적용"하고 성공하면 됐고, 만일 실패하면 그 다음에 두번째 rule을 적용
- 적용 : body내의 subgoal들과 해당 인자들에 대해, 주어진 query(즉, goal)에 있는 데이터와의 unification(단일화)를 시도한다. (재귀적으로 함)
- 이 때, body내의 subgoal들 중에서 하나라도 실패하면 그 rule은 실패(backtracking)

단일화 (unification)

변수와 상수

father (태종, 세종)과 father (X, 세종)을 unify 하면

➔ X를 태종으로 간주

변수와 변수

parent(X, Y)와 parent(X, Z)를 unify 하면

➔ Z를 Y로 간주 (혹은 그 반대)

상수와 상수

father(태종, 세종)과 father(태종, 세종)을 unify하면

➔ 같으므로 별도의 과정 필요 없음

father(태종, 세종)과 father(태조, 태종)을 unify하면

➔ 실패! 답이 없음

기타

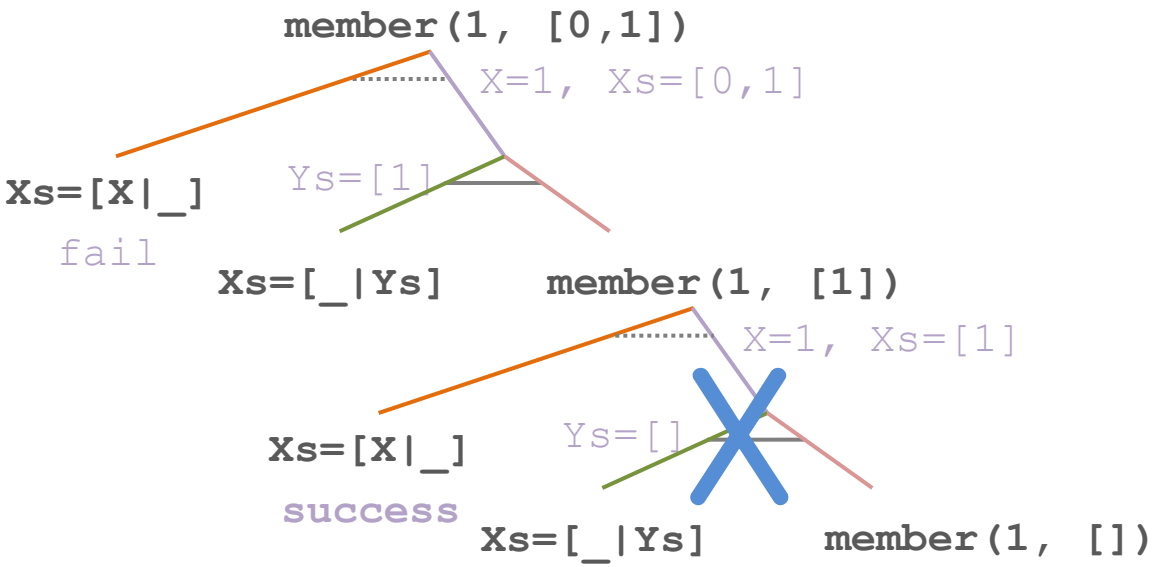
father(태종, 세종)과 mother(X, 세종)을 unify하면

➔ 실패! 답이 없음

__는 아무 값이나 상관없음

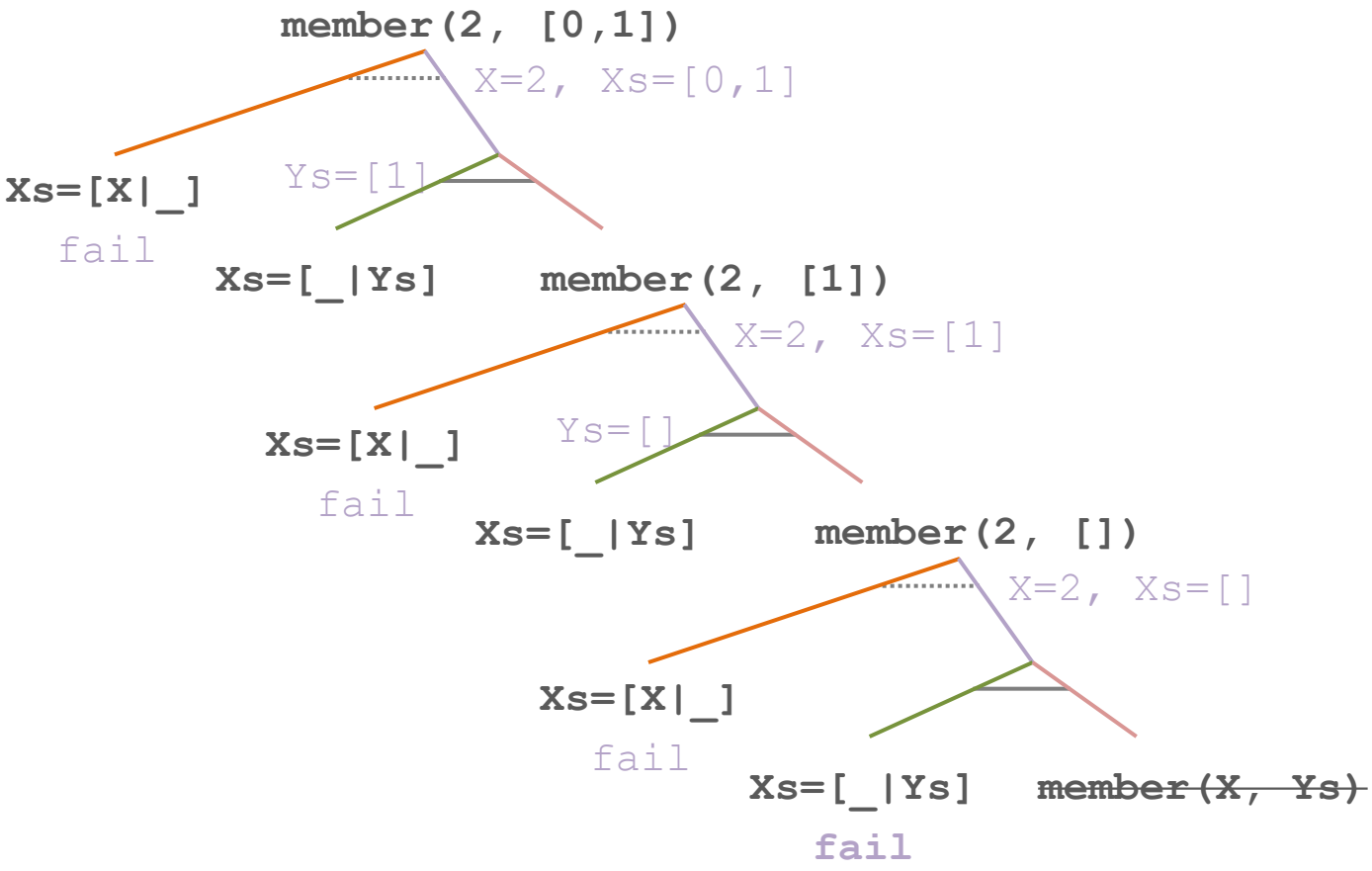
수행 예제

```
member(X, Xs) :- Xs=[X|_]

member(X, Xs) :- Xs=[_|Ys], member(X,Ys).
```



수행 예제 (계속)

```
member(X, Xs) :- Xs=[X|_]

member(X, Xs) :- Xs=[_|Ys], member(X,Ys).
```



Backtracking의 단점

앞의 예에서

- 더 이상 rule을 적용할 필요가 없는 fact (여기서는 Xs=[X|_] 와 Xs =[_|Ys])가 나올 때까지 계속 rule을 적용한다.

- 만일 원하는 답이 없으면?

  (이 예에서는 query에서 주어진 값이 해당 리스트의 member가 아니면)

  - 모든 가능한 rule 들에 대해 unification 해봐야 방법이 없다는 것을 알 수 있다 ➔ 비효율적이다.

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 다음 프로그램에 대한 설명으로 틀린 것은?

```
member(X, [X | _] ).

member(X, [_ |Xs]) :- member(X, Xs)
```

① 원소 X와 리스트를 인자로 받아 X가 리스트에 속하는지 여부를 알려준다.

② Xs는 주어진 리스트의 tail을 나타낸다.

③ 첫번째 rule이 성공했다는 뜻은 X가 첫번째 원소와 동일하다는 뜻이다.

④ 두번째 rule은 또다시 rule의 적용을 유발하여 무한 loop에 빠질 수 있다는 단점이 있다.

확인

평가하기 2

2. 다음중 Prolog 프로그램의 수행 과정에 대한 특징으로 옳지 않은 것은?

① H:- G1, G2... Gn의 의미는 H가 수행되고 나서 반드시 G1,G2, ...Gn이 수행되어야함을 의미한다.

② body에 있는 각 subgoal들 (G1, G2,.. Gn)을 수행하기 위해서는 해당되는 rule이 있는지 다시 찾아보고 여러 rule이 있으면 그 중 하나만 수행한다.

③ 한 rule이 실패하는 경우 다른 rule을 선택한다.

④ unification동안에는 query및 subgoal에서 주어진 인자와 rule의 인자를 동일하도록 만드는 시도가 포함된다.

확인

정리하기

- 논리적 프로그래밍언어로 프로그램을 작성하기 위해서는 선언적으로 프로그래밍을 해야한다.
- Prolog 프로그램의 수행 과정은 내부적으로 unification (단일화), backtracking(되추적) 등의 자료흐름과 제어흐름이 기본이 되어 동작한다.
- unification은 goal의 인자를 가지고 rule이나 fact 의 인자와 동일화 시키는 것을 시도한다.
- backtracking 은 여러 해당 rule 중 하나를 선택한후 만일 실패하면 다른 rule 을 선택하는 것으로, 효율성이 좋지는 않다.

"강의를 마치겠습니다. 수고하셨습니다."