CS370

Symbolic Programming
Declarative Programming

LECTURE 18: Language Processing

Jong C. Park

park@cs.kaist.ac.kr

Computer Science Department

Korea Advanced Institute of Science and Technology

http://nlp.kaist.ac.kr/~cs370

Language Processing with Grammar Rules

- Grammar rules in Prolog
- Handling meaning
- Defining the meaning of natural language

Grammar rules in Prolog

⊙ Grammar

- A formal device for defining sets of sequences of symbols
- Example: BNF (Backus-Naur Form)
  - Production rules
    - `<s> ::= a b`
    - `<s> ::= a <s> b`
  - Terminology
    - Non-terminals, Terminals, Sentences
    - Generation, Recognition
    - Parsing

Grammar rules in Prolog

⊙ Further Example: Motions of a robot arm

- Example command sequences
  - up
  - up up down up down
- Sample Grammar
  - <move> ::= <step>
  - <move> ::= <step> <move>
  - <step> ::= up
  - <step> ::= down

Grammar rules in Prolog

Definite Clause Grammar

- Example transformation for DCG
  - s --> [a], [b].
  - s --> [a], s, [b].
  - move --> step.
  - move --> step, move.
  - step --> [up].
  - step --> [down].
- In Prolog implementations that accept the DCG notation, the transformed grammars can be used as recognizers of sentences.

Grammar rules in Prolog

⊙ Definite Clause Grammar

```
?- s([a,a,b,b], [ ]).
yes
?- s([a,a,b], [ ]).
no
?- move([up,up,down], [ ]).
yes
?- move([up,up,left], [ ]).
no
?- move([up,X,up], [ ]).
X = up;
X = down;
no
```

Grammar rules in Prolog

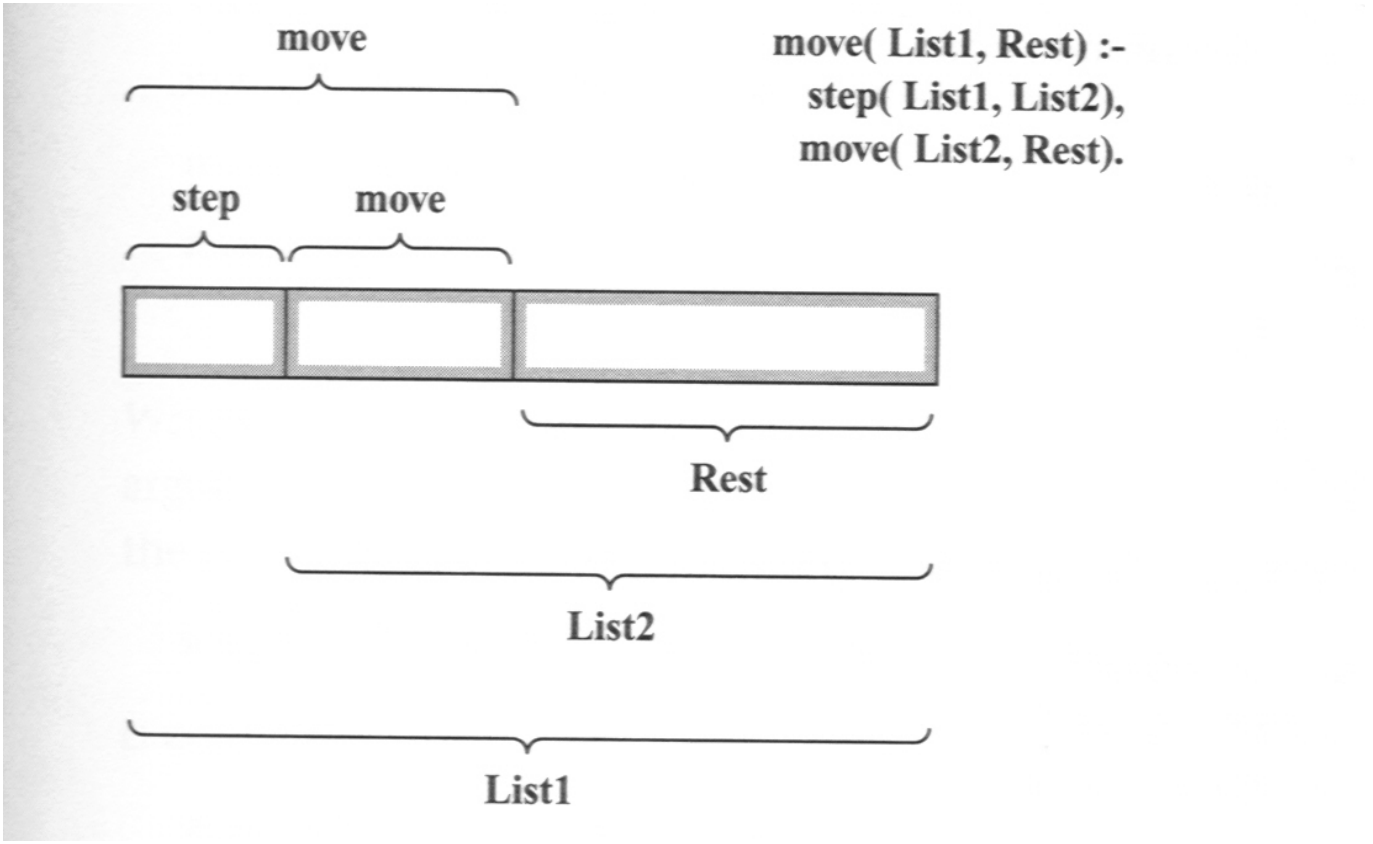
⊙ Definite Clause Grammar

- Prolog converts the given DCG rules into a program for recognizing sentences generated by the grammar.

```
move(List, Rest) :-
        step(List, Rest).
move(List1, Rest) :-
        step(List1, List2), move(List2, Rest).
step([up|Rest], Rest).
step([down|Rest], Rest).
```

Grammar rules in Prolog

⊙ Figure 21.1



Grammar rules in Prolog

⊙ Translation of DCG into standard Prolog

- Example 1

```
n --> n1, n2, ..., nn.
n(List1,Rest) :- n1(List1, List2),
                 n2(List2, List3),
                 ...,
                 nn(Listn, Rest).
```

- Example 2

```
n --> n1, [t2], n3, [t4].
n(List1,Rest) :- n1(List1, [t2|List3]),
                 n3(List3, [t4|Rest]).
```

Grammar rules in Prolog

⊙ DCG: Further Examples

- English grammar

```
sentence --> noun_phrase, verb_phrase.
verb_phrase --> verb, noun_phrase.
noun_phrase --> determiner, noun.
determiner --> [a].
determiner --> [the].
noun --> [cat].
noun --> [mouse].
verb --> [scares].
verb --> [hates].
```

Grammar rules in Prolog

- **Example sentences**
  - [the, cat, scares, a, mouse]
  - [the, mouse, hates, the, cat]
- **Extension**
  - [the, mice, hate, the, cats]
- **Extended Grammar**

```
noun --> [mice].
verb --> [hate].
```

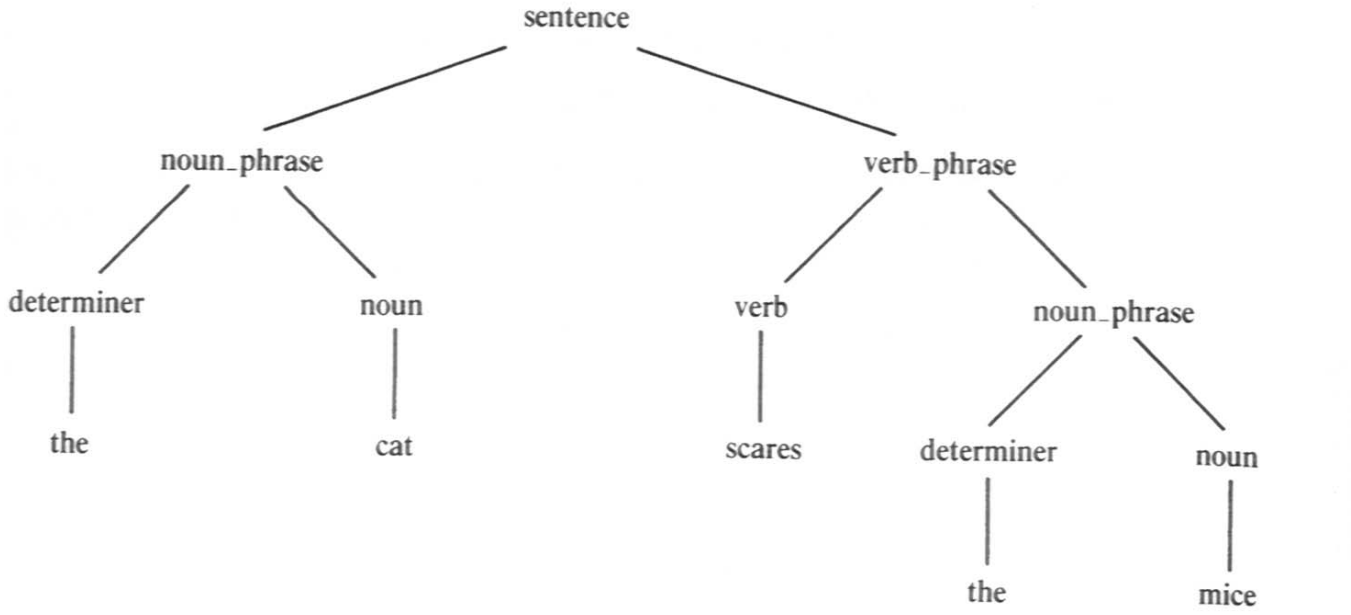
Grammar rules in Prolog

- **Problem**
  - [the, mouse, hate, the, cat]
- **Adding arguments to non-terminal symbols**

```
setence(Number) --> noun_phrase(Number),
                    verb_phrase(Number).
verb_phrase(Number) --> verb(Number),
                        noun_phrase(Number1).
noun_phrase(Number) --> det(Number),
                        noun(Number).
noun(singular) --> [mouse].
noun(plural) --> [mice].
```

Handling meaning

⊙Constructing parse trees



Handling meaning

⊙Constructing parse trees

- The parse tree of a phrase is a tree such that
  - All the leaves of the tree are labeled by terminal symbols of the grammar.
  - All the internal nodes of the tree are labeled by non-terminal symbols; the root of the tree is labeled by the non-terminal that corresponds to the phrase.
  - The parent-children relation in the tree is as specified by the rules of the grammar.

Handling meaning

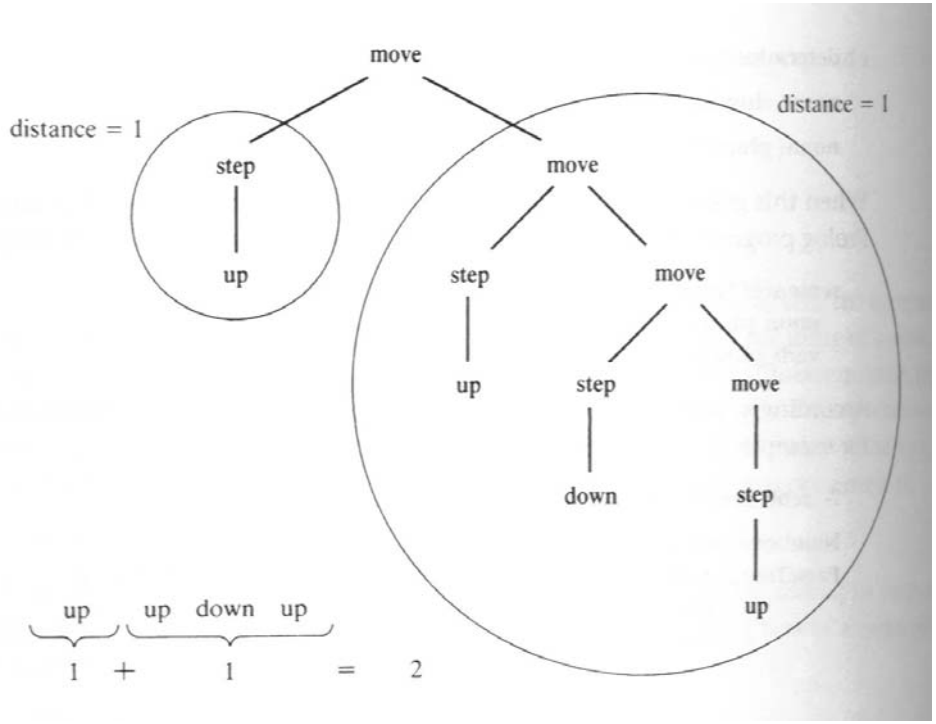
⊙Constructing parse trees

- Modification of a DCG grammar to generate a parse tree
  - noun_phrase(DetTree, NounTree)

```
noun_phrase(noun_phrase(DetTree, NounTree))
  -->
  det(DetTree), noun(NounTree).
```

Handling meaning

⊙From the parse tree to the meaning



Handling meaning

From the parse tree to the meaning

- One approach to extract the meaning
  - Generate the parse tree of the given sentence
  - Process the parse tree to compute the meaning

```
move(move(Step)) --> step(Step).
move(move(Step, Move)) -->
        step(Step), move(Move).
step(step(up)) --> [up].
step(step(down)) --> [down].
```

Handling meaning

From the parse tree to the meaning

- One approach to extract the meaning
  - Generate the parse tree of the given sentence
  - Process the parse tree to compute the meaning

```
meaning(move(Step, Move), Dist) :-
        meaning(Step, D1),
        meaning(Move, D2),
        Dist is D1 + D2.
meaning(move(Step), Dist) :- meaning(Step, Dist).
meaning(step(up), 1).
meaning(step(down), -1).
```

Handling meaning

From the parse tree to the meaning

- One approach to extract the meaning
  - Generate the parse tree of the given sentence
  - Process the parse tree to compute the meaning

```
?- move(Tree, [up, up, down, up], [ ]),
     meaning(Tree, Dist).
Dist = 2
Tree = move(step(up), move(step(up),
  move(step(down), move(step(up)))) )
```

Handling meaning

⊙ Interleaving syntax and semantics in DCG

- move(Dist)
  - A move phrase whose meaning is Dist

```
move(D) --> step(D).
move(D) --> step(D1), move(D2), {D is D1 + D2}.
step(1) --> [up].
step(-1) --> [down].
```

Handling meaning

- **Interleaving syntax and semantics in DCG**
  - move(Dist): Use gears
    - Examples

      stop

      g1 up up stop

      g1 up up g2 down up stop

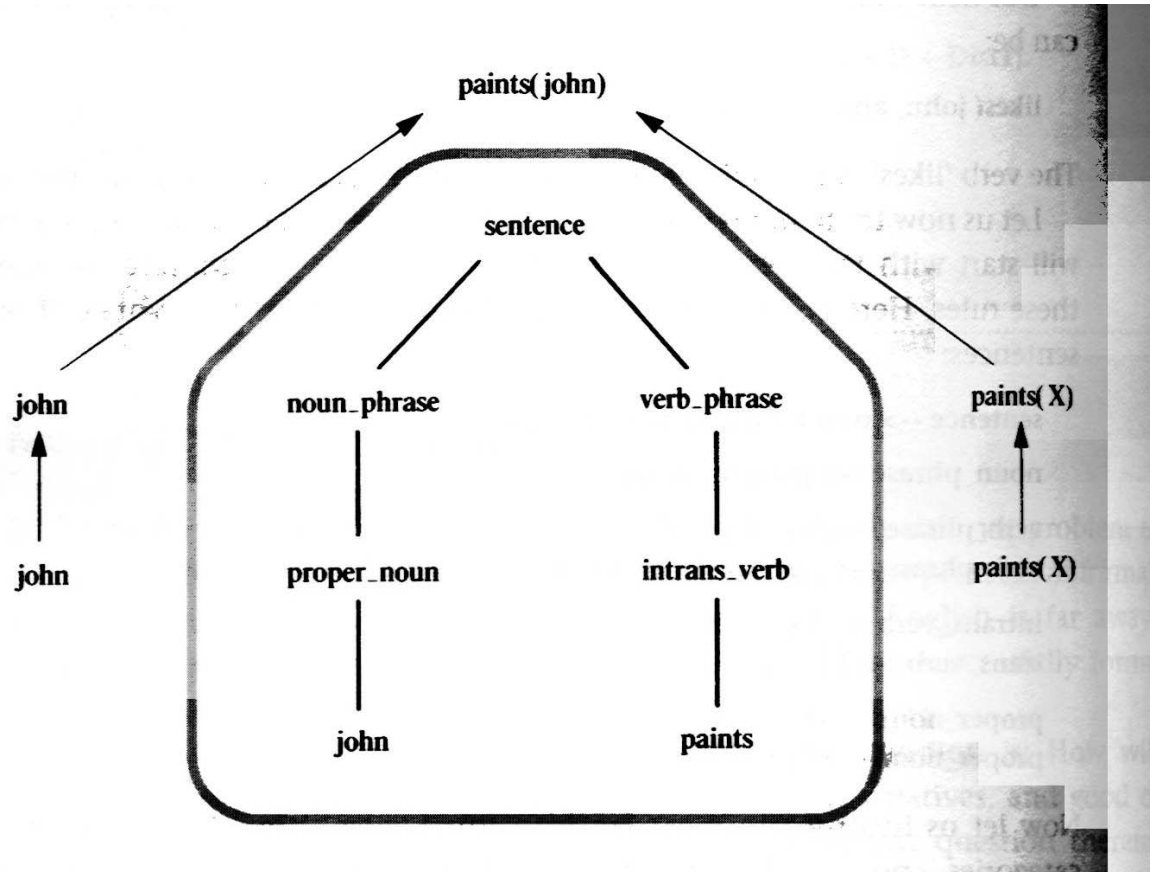
      g1 g1 g2 up up g1 up down up g2 stop

Handling meaning

- Program

```
prog(0) --> [stop].
prog(Dist) --> gear(_), prog(Dist).
prog(Dist) --> gear(G), move(D),
               prog(Dist1), {Dist is G*D + Dist1}.
gear(1) --> [g1].
gear(2) --> [g2].
```

Defining the meaning of natural language



Defining the meaning of natural language

⊙ Meaning of simple sentences in logic [1]

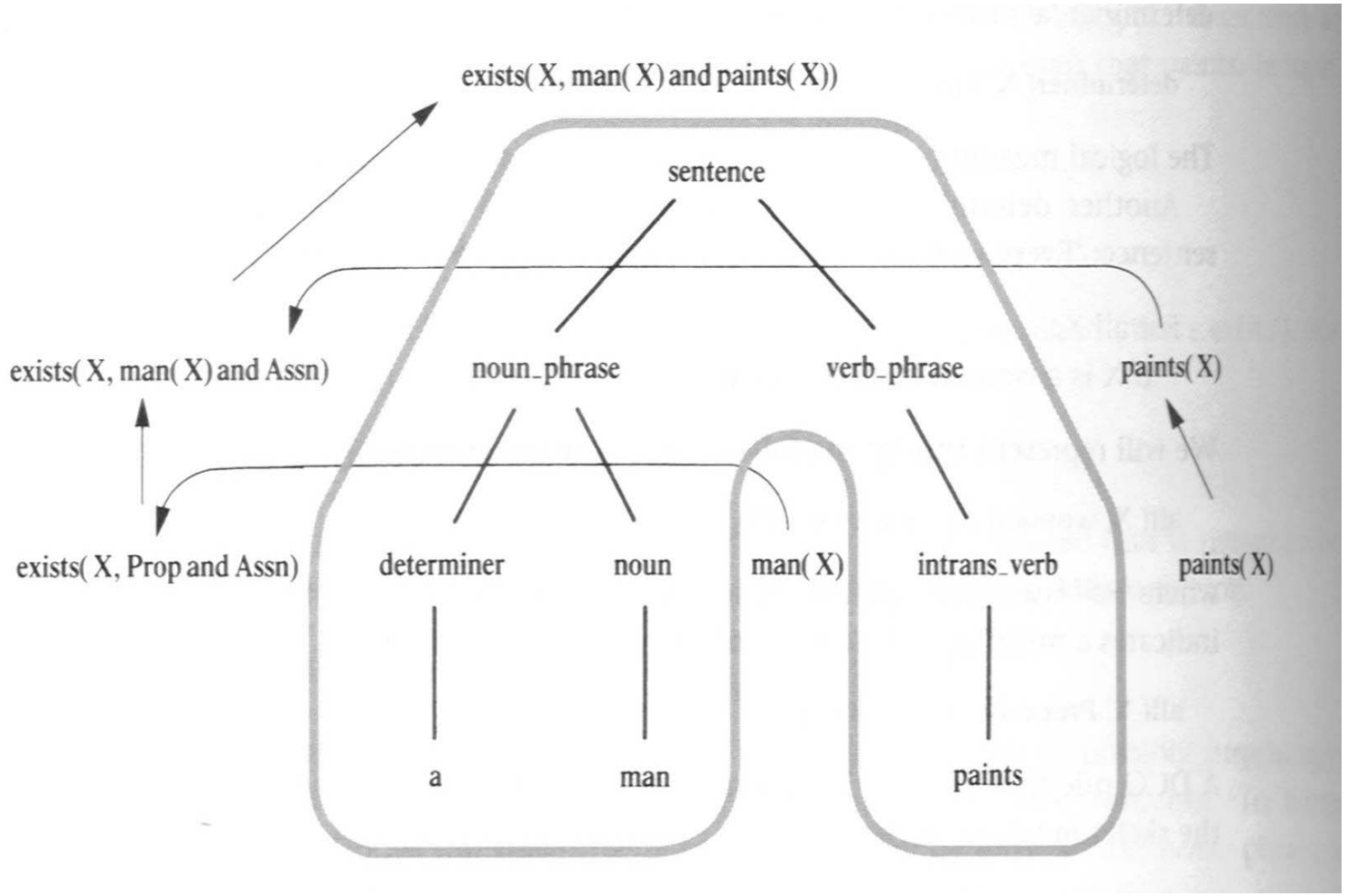
```
proper_noun(john) --> [john].
intrans_verb(paints(X)) --> [paints].
noun_phrase(NP) --> proper_noun(NP).
verb_phrase(VP) --> intrans_verb(VP).
sentence(S) --> noun_phrase(NP), verb_phrase(VP),
                    {compose(NP,VP,S)}.
actor(paints(X),X).
compose(NP,VP,VP) :- actor(VP,NP).
```

Defining the meaning of natural language

⊙ Meaning of simple sentences in logic [2]

```
proper_noun(john) --> [john].
intrans_verb(Actor,paints(Actor)) --> [paints].
noun_phrase(NP) --> proper_noun(NP).
verb_phrase(Actor,VP) --> intrans_verb(Actor,VP).
sentence(VP) --> noun_phrase(Actor),
                 verb_phrase(Actor,VP).
```

Defining the meaning of natural language



Defining the meaning of natural language

⊙ Meaning of determiners ‘a’ and ‘every’

- A man paints.
  - There exists an X such that X is a man and X paints.
  - exists(X,man(X) and paints(X))
  - exists(X,man(X) and Assertion)
  - exists(X,Property and Assertion)

```
:- op(100,xfy,and).
det(a,Prop,Assn,exists(X,Prop and Assn)) --> [a].
```

- Every woman dances.
  - all(X,woman(X) => dances(X))

```
det(a,Prop,Assn,all(X,Prop => Assn)) --> [every].
```

Defining the meaning of natural language

- **Meaning of determiners ‘a’ and ‘every’**

```
sentence(S) --> noun_phrase(X,Assn,S),
                verb_phrase(X,Assn).
noun_phrase(X,Assn,S) --> det(X,Prop,Assn,S),
                          noun(X,Prop).
verb_phrase(X,Assn) --> intrans_verb(X,Assn).
intrans_verb(X,paints(X)) --> [paints].
det(X,Prop,Assn,exists(X,Prop and Assn)) --> [a].
noun(X,man(X)) --> [man].
proper_noun(john) --> [john].
noun_phrase(X,Assn,Assn) --> proper_noun(X).
```

Summary

- Grammar rules in Prolog
- Handling meaning
- Defining the meaning of natural language