Programming Paradigms CSI2120 – Winter 2018

Jochen Lang

EECS, University of Ottawa

Canada

uOttawa

L'Université canadienne
Canada's university

Université d'Ottawa | University of Ottawa

uOttawa.ca

Logic Programming in Prolog

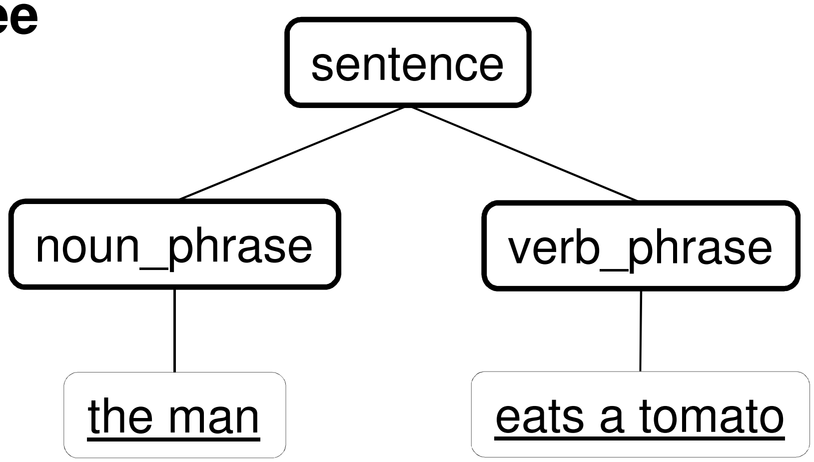
- **Language Processing**
  - Context free grammars
  - Parse trees
  - Definite clause grammars (DCG)

Grammar of (Simple) English Sentences

- **(Simple) English sentences consist of a noun phrase followed by a verb phrase**
- **Context free grammar rule**

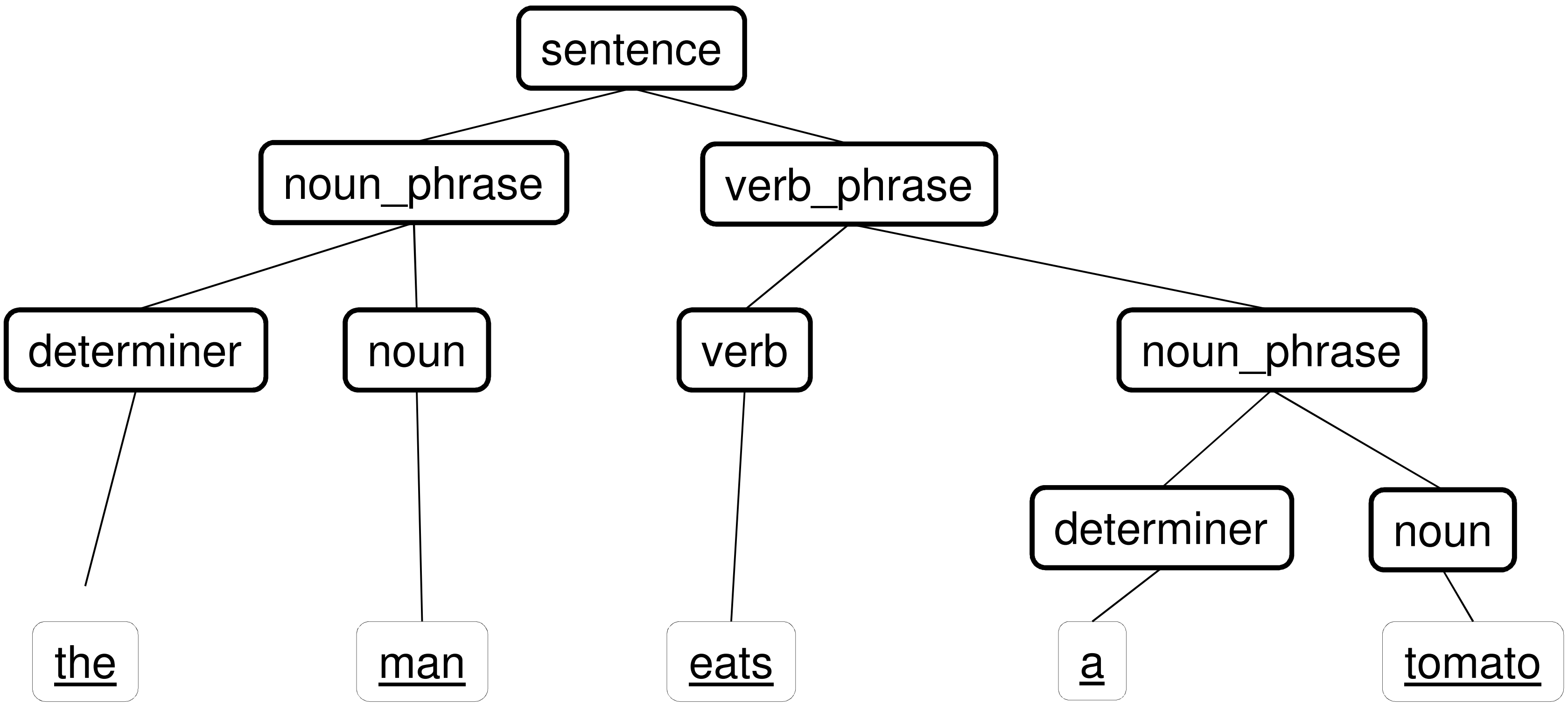
  sentence → noun_phrase,verb_phrase

  – which reads: sentence can take the form of noun_phrase followed by verb_phrase
- **As a tree**



Parse Tree

- **Decomposing the whole structure gives us a parse tree**
  - For the sentence from before:



Analyzing a Sentence in Prolog

- **We use a structure where we always work on the head of the list and return the rest for further analysis**

```
sentence(X,Z) :-
    noun_phrase(X,Y), verb_phrase(Y,Z).

noun_phrase(X,Z) :-
    determiner(X,Y), noun(Y,Z).

verb_phrase(X,Z) :-
    verb(X,Z).
verb_phrase(X,Z) :-
    verb(X,Y), noun_phrase(Y,Z).
```

Vocabulary

- **Need a vocabulary to store all words and their classification**
- **Example:**

```
determiner([the|Z],Z).
determiner([a|Z],Z).
noun([tomato|Z],Z).
noun([bird|Z],Z).
noun([man|Z],Z).
noun([cat|Z],Z).
verb([eats|Z],Z).
verb([sings|Z],Z).
verb([loves|Z],Z).
```

Example

- **Can ask if something can be parsed as a sentence**

```
?- sentence([the,man,eats,a,tomato],[]).
true
```

Notation DCG (definite clause grammar)

- **Prolog has a built-in operator which hides the extra (two) list parameters form us**
- **An equivalent program to before can be written as follows**

```
sentence -->  noun_phrase, verb_phrase.
noun_phrase --> determiner, noun.
verb_phrase --> verb.
verb_phrase --> verb, noun_phrase.
determiner --> [the].
determiner --> [a].
noun --> [tomato].
```

and so on

Arguments in DCG

- **For example, want to distinguish between singular and plural form for noun and verbs**

```
sentence --> sentence(_).
sentence(X) -->  noun_phrase(X),
verb_phrase(X).
noun_phrase(X) --> determiner(X), noun(X).
verb_phrase(X) --> verb(X).
verb_phrase(X) --> verb(X), noun_phrase(_).
determiner(_) --> [the].
noun(singular) --> [tomato].
noun(plural) --> [tomatos].
verb(plural) --> [eat].
verb(singular) --> [eats].
```
… and so on

Explicitly Constructing a Parsetree

- **Add an extra argument to hold the result of the parsing in a parse tree**

```
sentence(PT) --> sentence(_,PT).
sentence(X,sentence(NP,VP)) -->
noun_phrase(X,NP), verb_phrase(X,VP).
noun_phrase(X,noun_phrase(D,N)) -->
determiner(X,D), noun(X,N).
verb_phrase(X,verb_phrase(V)) --> verb(X,V).
verb_phrase(X,verb_phrase(VP,NP)) --> verb(X,VP),
noun_phrase(_,NP).
determiner(_,determiner(the)) --> [the].
noun(singular,noun(tomato)) --> [tomato].
noun(plural,noun(tomatos)) --> [tomatos].
verb(singular,verb(eats)) --> [eats].
verb(plural,verb(eat)) --> [eat]. … and so on
```

Example

```
?- sentence(T,[the, cats, eat, the, bird],[]).
T = sentence(noun_phrase(determiner(the),
noun(cats)), verb_phrase(verb(eat),
noun_phrase(determiner(the), noun(bird))))
```

– Our predicates do not perform pretty printing but this could be easily added for printing T, e.g.

```
T = sentence(
      noun_phrase(determiner(the),
                  noun(cats)),
      verb_phrase(verb(eat),
                  noun_phrase(determiner(the),
                              noun(bird))))
```

Simplify the Dictionary

- **We can change rules for each word into rules for classes of word and note all our vocabulary as simple facts.**

```
determiner(X,determiner(Y)) --> [Y],
                               {isDeterminer(Y,X)}.
noun(X,noun(Y)) --> [Y], {isNoun(Y,X)}.
verb(X,verb(Y)) --> [Y], {isVerb(Y,X)}.
isDeterminer(the,_).
isDeterminer(a,singular).
isNoun(tomato,singular).
isNoun(tomatos,plural).
isVerb(eats,singular).
isVerb(eat,plural). … and so on
```

  – Note the use of `{}` to exclude the extra list parameters

Another DCG Example: An Elevator

- **Other problems can be expressed in a DCG**
  - For example: Keeping track of the level an elevator is at

```
displacement(L) --> motion(L).
displacement(L) --> motion(L1), displacement(L2),
{L is L1+L2}.
motion(1) --> [up].
motion(-1) --> [down].
```

- **Query**

```
?- displacement(L,[up,up,up,up,down,down,up],X).
L = 1, X = [up, up, up, down, down, up] ;
L = 2, X = [up, up, down, down, up] ;
```

… and so on

```
L = 3, X = [] ;
```

Summary

- **Language Processing**
  - Context free grammars
  - Parse trees
  - Definite clause grammars (DCG)