Prolog 3

Fariba Sadri

Natural language Processing
Very Brief Introduction

- Input Text (or speech) in some language
- Ouput could be:
  - Whether or not text is grammatically correct
  - A translation to another language
    - To a logic language
    - To a natural language
  - A representation of the information in the text in some other format for further processing, e.g. Automatic Summarisation.

A Simple Syntactic Analysis

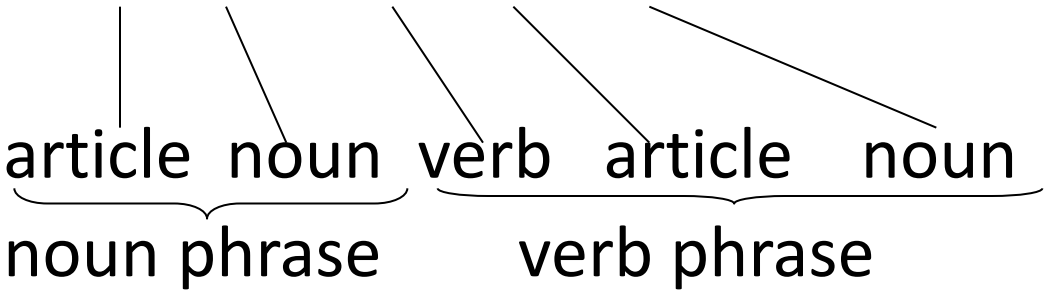
Phrase-structure grammar of very simple English:

sentence --> nounphrase, verb phrase

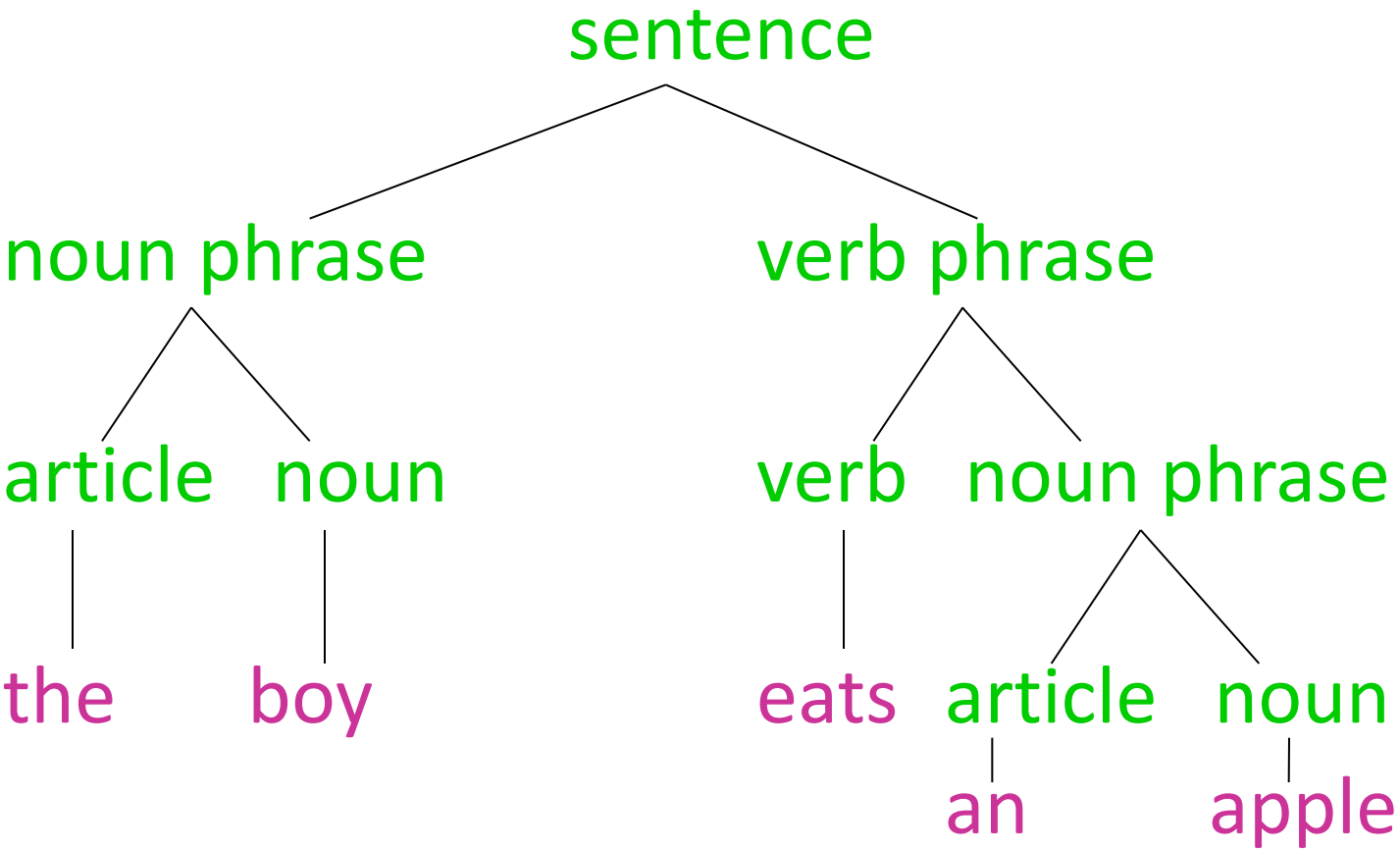
noun phrase --> article, noun

verb phrase --> verb| verb, noun phrase

"The boy eats an apple."



Parse Tree


sentence
noun phrase
verb phrase
article
noun
verb
noun phrase
the
boy
eats
article
noun
an
apple

A Simple Syntactic Analysis cntd.

A simple Lexicon:

article --> the | a | an

noun --> boy | apple | carrot | song

verb --> eats | sings

With this grammar, for example:

"the boy eats an apple"

is a grammatically correct sentence, but

"the boy eats a eats"

is not.

Of cours e, the grammar is too simple, and

"an apple eats a boy"

is also a grammatically correct sentence!

Never mind that for now.

Language Processing with *DCG Grammar Rules* in Prolog

- Many Prolog implementations, including Sicstus Prolog, provide specialised notation for language processing.
- This notation is called *DCG* – *Definite Clause Grammars*.
- This allows writing parsers in Prolog very easily and elegantly.
- Not part of the course.

Prolog DCG Rules

These can be written in the form:

head --> body

For example:

sentence --> noun_phrase, verb_phrase.

Example of Prolog DCG Notation

```
sentence --> noun_phrase, verb_phrase.

noun_phrase --> article, noun.
verb_phrase --> verb.
verb_phrase --> verb, noun_phrase.

article--> [a].
article--> [the].
article-->[an].

noun--> [boy].
noun--> [apple].

verb--> [eats].
```

- The DCG can be entered as a Prolog program directly and is itself a parsing program.
- Prolog automatically transforms this into a Prolog program that can be queried:

```
?- sentence([a,boy,eats,an,apple], []).
yes.

?- sentence([a,boy,apple,eats], []).
no.

?- sentence(X, []).
X=[a,boy,eats] …..
```

Some notes

- Notice the use of sentence/2.
- DCG implementations in Prolog expect this notation in the queries.
- *Difference lists* used by DCG parsers for efficiency.
- *Difference lists* are beyond the scope of this course.

Exercise: For the Tutorial Syntactic Analysis in Prolog

- We will not work with DCGs.
- But we can still do natural language processing.
- You can write an ordinary Prolog program to check whether or not an English sentence is grammatically correct, according to the grammar given in previous slides, and that can also generate grammatically correct sentences.

Exercise: For the Tutorial cntd.

- You can represent sentences as a list of words, e.g. [the, boy, eats, an, apple].
- Define a predicate *sentence/1*, and any other auxiliary predicate you need, such that *sentence(S)* succeeds if *S* is a correct grammatical sentence

So for example:

?- sentence([a, cow, eats, the, grass]).

Gets the answer yes.

Extending Your Grammar

The you can extend your grammar so it is more *context-sensitive* :

- To make sure the verb and the noun agree in being both singular or both plural

  For example to avoid the boys eats an apple.

- You can also think how you can prevent recognition of strings of words such as the following sentences:

  the apple eats a boy

  the carrot sings

  the song kicks the cow