Prolog
Natural language Processing

Fariba Sadri

Natural language Processing
Very Brief Introduction

- Input Text (or speech) in some language
- Ouput could be:

  ✓Syntactic analysis: grammatical criteria

  ✓A translation to another language

    - To a logic language
    - To a natural language

  ✓Query answering

  ✓Sentiment analysis, e.g. from social media

A Simple Syntactic Analysis

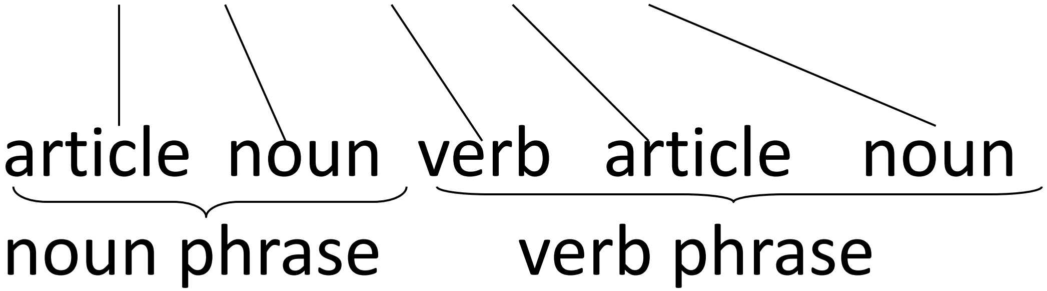
Phrase-structure grammar of very simple English:

sentence --> nounphrase, verb phrase

noun phrase --> article, noun

verb phrase --> verb | verb, noun phrase

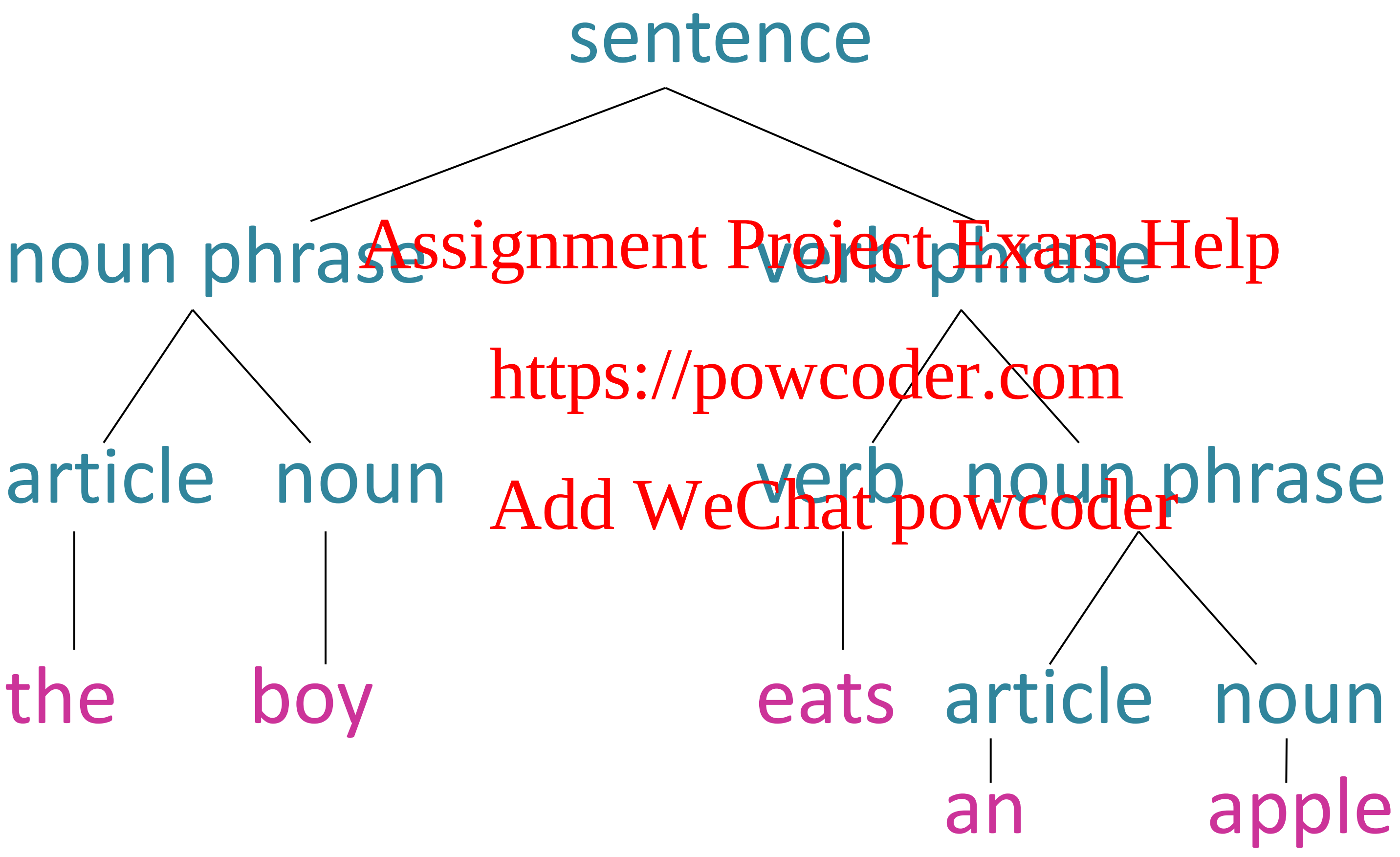
“The boy eats an apple.”



article noun verb article noun

noun phrase verb phrase

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 | song | cow | grass

verb --> eats | sings | kicks

With this grammar, for example:

“the boy eats an apple”

is a grammatically correct sentence, but

“the boy eats a eats”

is not.

Of course, the grammar is too simple, and

“an apple eats a boy”

is also a grammatically correct sentence!

Never mind for now.

Exercise: For the Tutorial Syntactic Analysis in Prolog

With what you know of Prolog so far:

- 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

Then you can extend your grammar so it is more sophisticated:

- Avoid *a, apple* and *an boy* as noun phrases.
- Make sure the verb and the noun agree in being both singular or both plural, e.g.

  the boys eats an apple.

- Avoid “non-active” nouns pairing with “active” verbs, e.g.

  the apple eats a boy.

  the carrot sings.

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.

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.

Extended Example: matching article to noun

✔ a boy

✔ the boy

✔ the boys

✘ a boys

```
noun_phrase --> article(N), noun(N).
article(single)--> [a].
article(single)--> [the].
article(multi)--> [the].
noun(single)--> [boy].
noun(multi)-->[boys].
| ?- noun_phrase([a, boys],[]).
no
```

A DCG for a Simple Formal Language

A formal language, e.g. logic or mathematics, is a set of strings, made up according to a clear grammar.

Consider a simple language S.

Suppose S has two symbols: *a* and *b*.

Suppose a sentence is S is of the form

a^nb^n,

i.e. a string of *a*s of length $n \geq 1$, followed by a string of *b*s of the same length.

For example

ab , aabb , aaabbb and aaaabbbb

are correct sentences, but aabbb is not.

Lets define this grammar in Prolog DCG.

Base case: s --> [a,b].

Recursive case s --> firsta, s, lastb.

Note recursion

firsta --> [a].

lastb --> [b].

```
| ?- s(X, []).
X= [a,b] ? ;
X = [a,a,b,b] ? ;
X = [a,a,a,b,b,b] ? ;
X = [a,a,a,a,b,b,b,b] ? ;
X = [a,a,a,a,a,b,b,b,b,b] ? ;
X = [a,a,a,a,a,a,b,b,b,b|...] ? ;
X = [a,a,a,a,a,a,a,b,b,b|...] ?
```