Communication among Agents

Communication: -The intentional exchange of information brought about by the production and perception of signs drawn from a shared system of conventional signs.

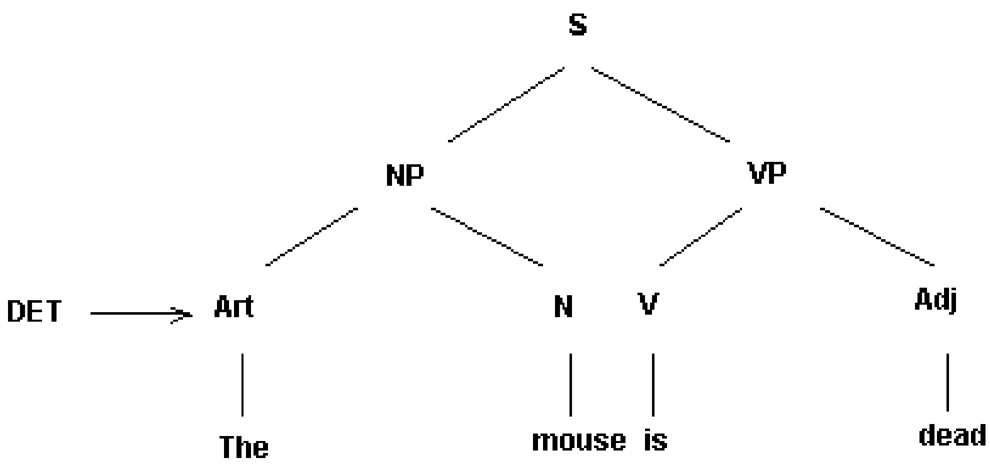
Parsing: recovering the phrase structure of an utterance given a grammar.

Applications of NLP

- Air Travel Information Systems (ATIS)
  1. Show me the flight from Atlanta to Boston on Friday.
  2. What is the cheapest fare?
- Machine Translation Systems – weather Rpts in Canada
- Front-Ends to Databases

Parse Trees

Here is a parse tree for an English sentence.



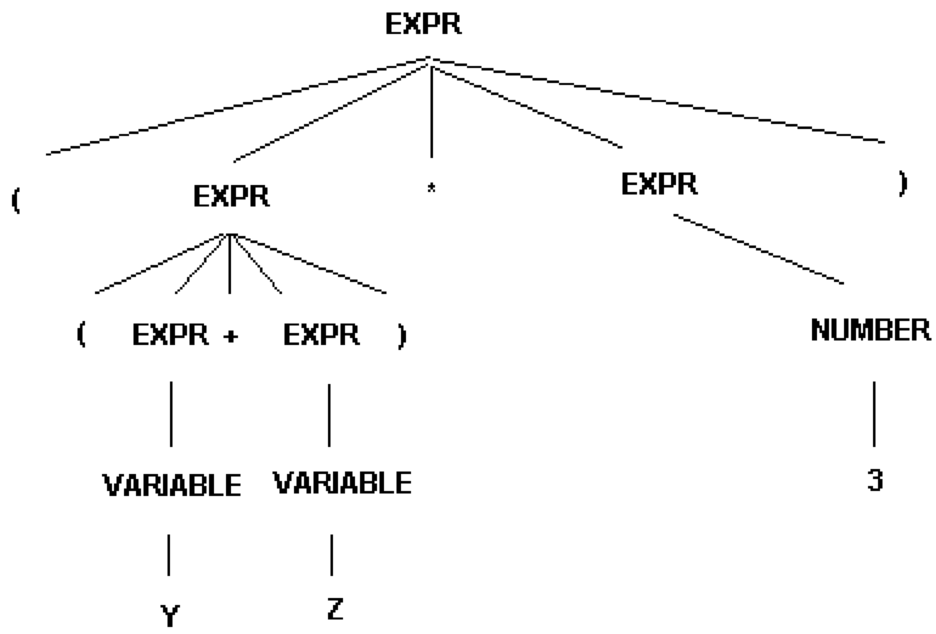
Context-Free Grammars

Below is a context free grammar.

$X \rightarrow AB$

1. $EXPR \rightarrow Number$
2. $EXPR \rightarrow Variable$
3. $EXPR \rightarrow (EXPR + EXPR)$
4. $EXPR \rightarrow (EXPR * EXPR)$

Context-Free Grammars (cont)



Above is a parse tree for the expression $((Y + Z) * 3)$ given the above grammar.

Grammar

A Grammar defines the legal expressions in a language.

The sequence of rewrite rules used to derive a sentence in this language reveals the structure of the sentence, and extractions this structure is called parsing the sentence.

The Lexicon for E

Consider the grammar below (from Russell and Norvig) for a fragment of English – E.

```
Noun  --> Stench|breeze|glitter|nothing|wvmpus|p

Verb  -->  is|see|smell|shoot|feel|stink|go|grab

Adjective --> right|left|east|south|black|smelly

Adverb  -->  here|there|nearby|ahead|right|left|

Pronnun --> me|you|I|it...

Article  -->  the |a|an|...

Preposition  --> to|in|on|near...

Conjunction  --> and |or|but|...

Digit  --> 0|1|2|3|4|5|6|7|8|9
```

The Grammar for E

```
S --> NP VP                 I + feel a breeze
     | S conjunction S      I feel a breeze + and +
                                      I smell a wum
NP --> Pronoun              I
     | Noun                 pits
     | Article Noun         the wumpus
     | Digit Digit          34
     | NP PP                the wumpus + to the east
     | NP Rel_Clause        the wumpus + that is sme
VP --> Verb                 stinks
     | VP NP                feel + a breeze
     | VP Adjective         is + smelly
     | VP PP                turn + to the east
     | VP Adverb            go + ahead
PP --> preposition NP       to + the east
RelClause --> that VP       that + is smelly
```

Parsing

Maintain a parse forest

initially list of words

At each iteration:

Match some subsequence of elements in the forest with the right-hand side of a grammar rule.

Then replace the subsequence with a single parse tree whose category is the left-hand

Parsing (cont)

So, starting off with the following sentence:

The wumpus is alive.

After matching with the rule below:

Article -> the

The first word in the sentence is replaced
by a tree with the parent
being Article and the child being the.

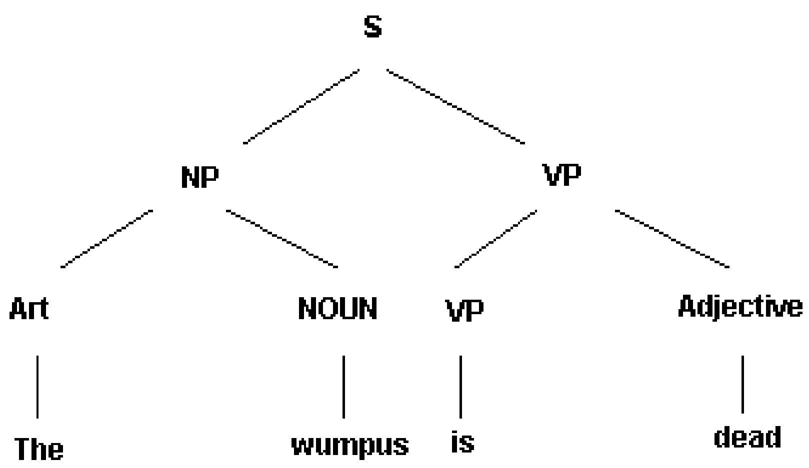
Parsing (cont)

The whole process is illustrated below:

```
       forest                       rule
The wumpus is dead.     Article -> the
Article wumpus is dead. Noun -> wumpus
Article Noun is dead.   NP --> Article Noun
NP            is dead.  Verb --> is
NP Verb       dead.     Adjective --> dead
NP Verb Adjective.      VP --> Verb
NP VP Adjective.        VP --> VP Adjective
NP VP                   S --> NP VP
S
```

Parse Tree Example

The final result is the following tree:



¡br¿

Definite Clause Grammar

Note that context-free rules have the form of definite clauses. These are clauses with exactly one positive literal. There is a ready encoding of such grammatical rules into Prolog.

We can define a predicate sentence that will allow us to determine whether or not a particular string of words (representated as a list of atoms) is a legal sentence in the language. Or we can ask Prolog to generate all legal sentences.

Definite Clause Grammar (cont)

```
?-sentence([the, man, eats, the, apple ])
?-Sentence(X)

A simple grammar is given below:

sentence(x) :-
      append(Y, Z, X), noun-phrase(Y),
                              verb_phrase(Z).

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

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

verb_phrase(X) :- verb(X).

determiner([the]).
verb([eats]).
noun([apple]).
verb([sings]).
noun([man]).
```

Semantic Analysis

Output of the parsing procedure is a representation of the meaning of the sentence in something like first-order logic.

```
‘‘block B is on block C and block B is clear’’

          ===>

On(B,C) ^  clear(B) ^ Block(B) ^ Block(C)
```

The Definite Clause Grammar given above in Prolog can be modified to output a semantic representation.