NATURAL LANGUAGE PROCESSING

Module 5

NLP

- Natural language processing (NLP) is a field of CS, AI and computational linguistics concerned with the interactions between computers and human (natural) languages,
- In particular, concerned with programming computers to fruitfully process large natural language corpora.

NLP

- A natural language is just like a human language which we use for communication like English, Hindi, Gujarati, Bengali, Arabic ….etc.
- The one of the *main aim of AI* is to build a machine that can understand commands written or spoken in natural Language.
- It is not an easy task to teach a person or a computer, a natural language.
- The problem is *Syntax* (rules governing the way in which the words are arranged) & understanding context.

NLP

- Challenges in natural language processing frequently involve
  - *speech recognition,*
  - *natural language understanding,*
  - *natural language generation (frequently from formal, machine-readable logical forms),*
  - *connecting language and machine perception, dialog systems*.

NLP

- Language processing problem can be divided as:
  - Processing written text, semantic and syntactic knowledge of the language.
  - Processing spoken language.
- Applications of NLP:
  - Voice recognition, Machine translation, Information visualization, Grammar checking System etc.

NLP

NLP encompasses anything a computer needs to understand natural language (typed or spoken) and also generate the natural language.

- **Natural Language Understanding (NLU) :**

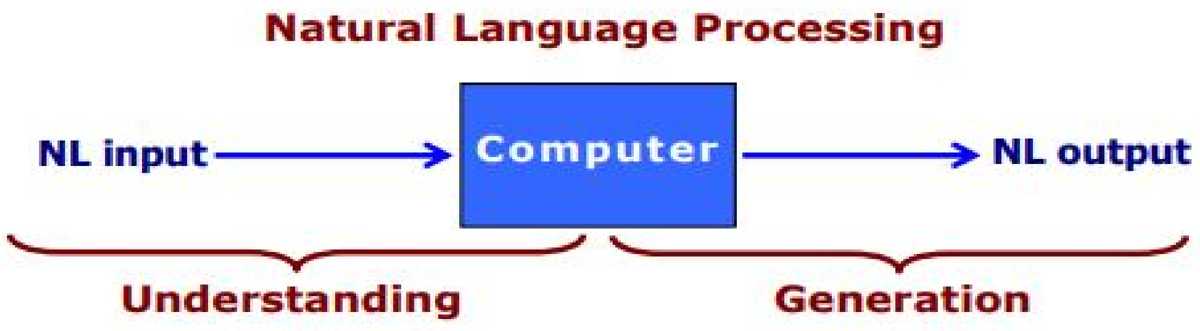
  The NLU task is understanding and reasoning while the input is a natural language.

  Here we ignore the issues of natural language generation.

- **Natural Language Generation (NLG) :**

  NLG is a subfield of natural language processing NLP.

  NLG is also referred to text generation.



Features of language

(that makes it difficult and useful)

- English sentences are incomplete description of the info that they intend to convey.
  - Eg. Some dogs are outside.
- The expression may mean different things.
  - Eg. Where's the water?
- No NL program can be complete bcoz new words, expressions etc can be generated easily.
- The are many ways to say the same thing.

Useful Features

- Language give flexibility to users to be vague or precise.
- Language helps us to communicate wrt infinite world with finite words.
- Language can evolve with experience.
- When a lot of information is known, facts imply each other.

Steps in NLP

- Null. Phonetics and Phonology
- 1. Morphological Analysis
- 2. Syntactic Analysis
- Null. Lexical Analysis
- 3. Semantic Analysis
- 4. Discourse Integration
- 5. Pragmatic Analysis

Phonetics

- Pronunciation of different speakers.
- Deals with physical building blocks of language sound system.
- Pace of Speech.
- Ex: I ate eight cakes, different 'k' sounds in 'kite', 'coat', That band is banned.
- If we speak in foreign English (I 8 8 cakes) similarly k (Hindi क for kite & coat).

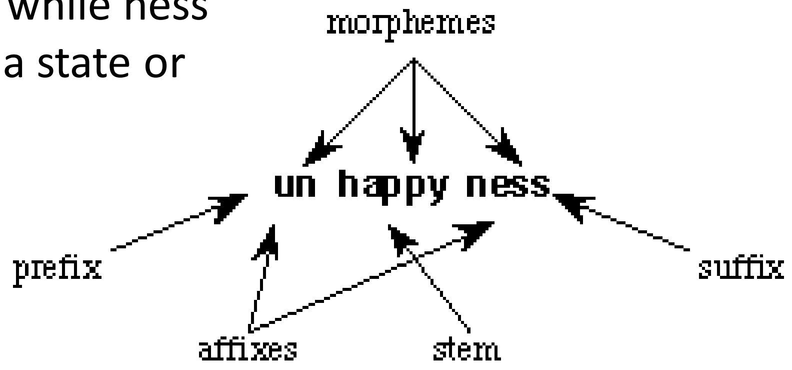
Phonology

- Processing of a speech.
- Organization of speech sound with in language.
- Ex: Bank (finance) v/s. Bank (River),
- In hindi, aa-jayenge (will come) or aaj-ayenge (will come today).

Morphological Analysis

- Morphology is the study of the structure and formation of words.
- One of the widespread task here is lemmatizing or stemming which is used in many web search engines.
  - Ex: dog-dogs, run-ran, bus-buses etc.
- Its most important unit is the *Morpheme*, which is defined as the "minimal unit of meaning”.
- Consider a word like: "unhappiness". This has three parts:

un means "not", while ness means "being in a state or condition"



Happy is a free morpheme because it can appear on its own

Syntactic Analysis

- It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words.
- The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.
- In other words, Syntactic Analysis exploit the results of Morphological analysis to build a structural description of the sentence.

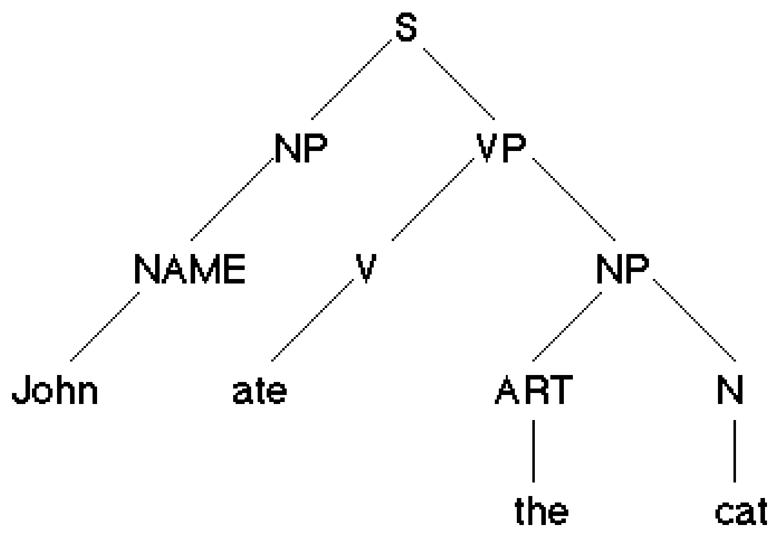
- sentence -> noun_phrase, verb_phrase
- noun_phrase -> noun
- noun_phrase -> determiner, noun
- verb_phrase -> verb, noun_phrase
- verb_phrase -> verb
- noun -> [mary]
- noun -> [apple]
- verb -> [ate]
- determiner -> [the]

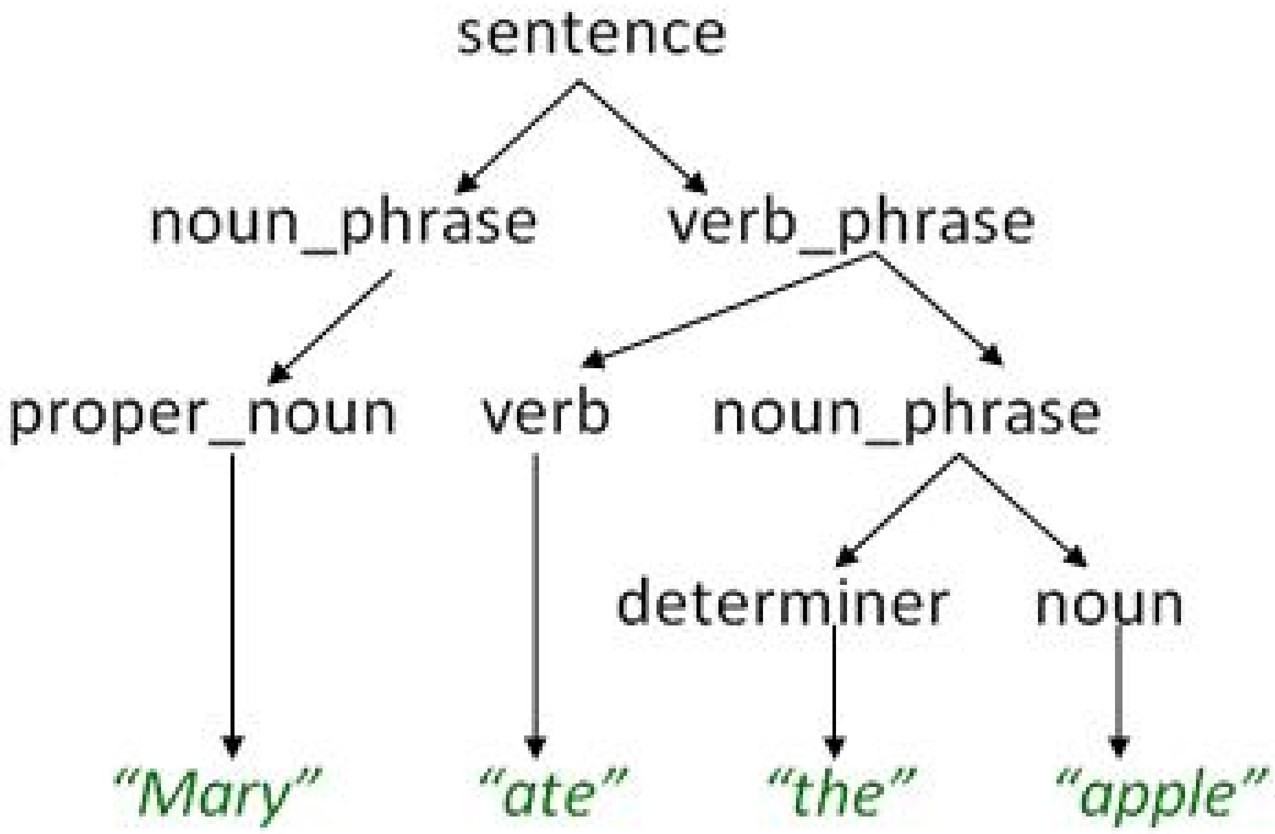
Syntactic Analysis

- Main problems on this level are:
  - part of speech tagging (POS tagging),
  - chunking or detecting syntactic categories (verb, noun phrases) and
  - sentence assembling (constructing syntax tree).

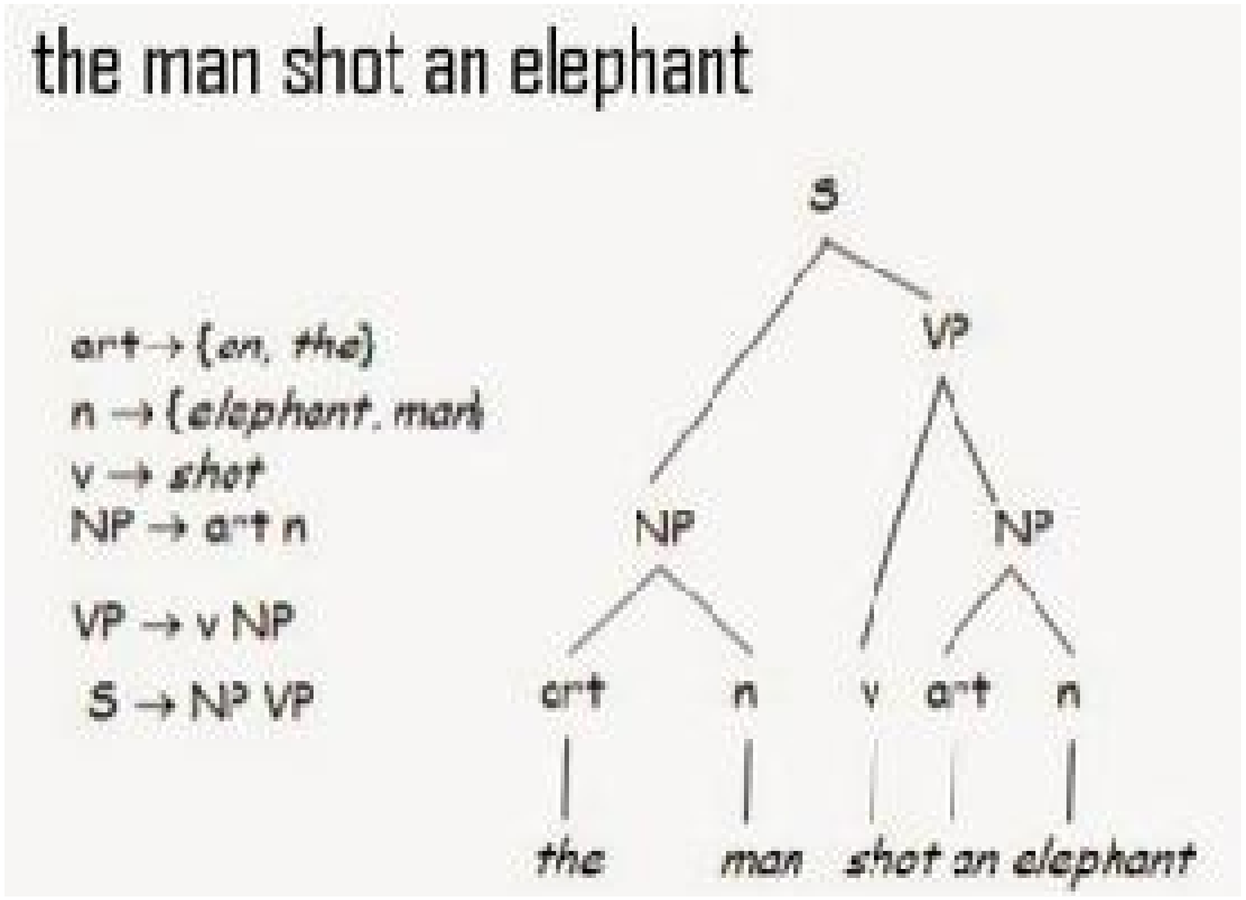
Parsing

- May be defined as the process of assigning *structural descriptions* to sequences of words in a natural language.
- The goal of Parsing it to convert the flat list of words that forms the sentence into a structure that defines the units that are represented by the flat list.
  - Converted into a hierarchical structure




sentence
noun_phrase
verb_phrase
proper_noun
verb
noun_phrase
determiner
noun
"Mary"
"ate"
"the"
"apple"

Syntactic Analysis



Explanation:
a sentence is construct from

- noun phrase (NP) & verb phrase (VP).
- Noun Phrase construct from article (art) and noun (n) .
- verb phrase are from verb (v) and noun phrase.

Lexical Analysis

- Obtaining the properties of word.
  - Ex: 'dog' then you can easily bring an image of dog & its properties like 4 leg, carnivore and animate.
- These properties also match with another animals like Lion.
- It involves identifying and analyzing the structure of words.
- Lexicon of a language means the collection of words and phrases in a language.
- Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.

Semantic Analysis

- The structures created by the syntactic analyzer are assigned meanings.
- A mapping is made between the syntactic structures and objects in the task domain.
- Semantic Analysis must do two important things:
  - Must map individual terms into appropriate objects in KB
  - Must create correct structures to correspond to correct meaning of words.

Semantic Analysis

- Semantics and its understanding as a study of meaning covers most complex tasks like:
  - finding synonyms,
  - word sense disambiguation,
  - constructing question-answering systems,
  - translating from one NL to another,
  - populating base of knowledge.
- Basically one needs to complete morphological and syntactical analysis before trying to solve any semantic problem.

Semantic Analysis

- Semantic and pragmatic analysis make up the most complex phase of language processing.

- One of the long-term projects of the NLP laboratory is the use of Transparent Intensional Logic (TIL) as a semantic representation of knowledge and subsequently as a transfer language in automatic machine translation.

Semantic Analysis

- The sentence "you have a colorless green ideas...." would be rejected semantically because colorless & green makes no sense.

Discourse Integration

- The meaning of an individual sentences may
  - depend on the sentences that precede it and
  - may influence the meanings of the sentence
  - (may depend on the sentences that precede it) that follow it.
  - Eg. Bill had a red balloon. John wanted it.
  - “ It” refers to red balloon and it should be identified like that only .
  - Such references are called *Anaphora*

Discourse Integration

- Another Example:
  - My house was broken into last week.
  - They took the TV and Music system.
- “ They” should be recognized as referring to burglars.

Pragmatic Analysis

- The structure representing what was said is reinterpreted to determine that what was actually meant.
- For example, the sentence
  - "Do you know what time it is?"
  - should be interpreted as a request to be told the time.
- This is the final step towards what to do as a result.
- One way is to record what was said as fact.(for some sentences intended effect is declarative and some times it is not).

Syntactic Processing

- Syntactic Processing:
  - Flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning. (this process is parsing)
  - Parsing a language presupposes the existence of a grammar for the language.
- The systems used for producing parse mainly have 2 components:
  - A declarative representation, called a grammar.
  - A procedure, called a parser, that compares the grammar against the i/p sentences to produce parsed structure.

Syntactic categories (common denotations) in NLP

- np - noun phrase
- vp - verb phrase
- s - sentence
- det - determiner (article)
- n – noun
- tv - transitive verb (takes an object)
- iv - intransitive verb
- prep – preposition
- pp - prepositional phrase
- adj - adjective

Syntactic Processing

- Most commonly grammars are represented as *production rules*.
- A context-free grammar (CFG) is a list of rules that define the set of all well-formed sentences in a language.
- Each rule has a left-hand side, which identifies a syntactic category, and a right-hand side, which defines its *alternative component parts*, reading from left to right.

CFG

- CFG is a finite collection of rules which tells us that certain sentences/strings are grammatical and what their grammatical structure is.
- A context free grammar is one in which all the rules apply regardless of the context.
- i.e. They would be of the type ‘Rewrite X as Y’, no further conditions being specified.

- Here's a simple context free grammar for a small fragment of English:

S -> NP VP

NP -> Det N

VP -> V NP

VP -> V

Det -> ***a***

Det -> ***the***

N -> ***woman***

N -> ***man***

V -> ***shoots***

What are the ingredients of this grammar?

It contains three types of symbol:

1. '**->**' = An instruction to rewrite whatever symbol appears to the left of the arrow as the symbol or string of symbols that appears to the right of the arrow.
2. Symbols written like: **S, NP, VP, Det, N, V**. These symbols are called ***non-terminal symbols***. *E*ach of these symbols is shorthand for a *grammatical category*.
3. Symbols in italics: ***a, the, woman, man,*** and ***shoots***. A computer scientist would probably call these ***terminal symbols*** and linguists would probably call them ***lexical items***.

Rule explanation

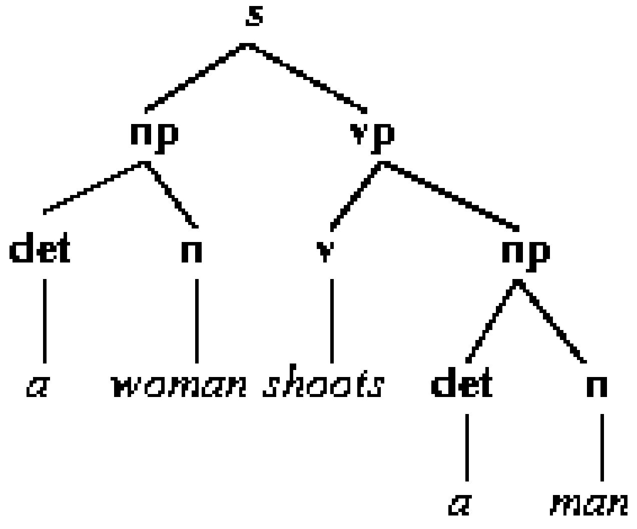
- Each rule consists of a *single non-terminal symbol*, followed by ->, followed by a finite sequence made up of terminal and/or non-terminal symbols.
- We interpret each rule X→Y as the instruction "rewrite X as Y."
- For example, rule (2) rewrites the symbol VP as the string of symbols Verb + NP, and defines Verb + NP to be a construction of the type VP.
- The symbol S (for "sentence") is designated as *the initial symbol.*
- It is necessary to *begin with a rule that has the initial symbol on the left.*
- Thereafter any rule may be applied in any order until no further rule is applicable.

Context Free Grammar (contd.)

Consider the string of words – ***a woman shoots a man.***

- Is this grammatical according to our little grammar?
- And if it is, what structure does it have?

The following tree answers both the questions:



Such a tree is called a *parse tree*, and it gives us two sorts of information:

1. Information about *strings* 2. Information about *structure*

Context-Free Grammars

We will look at the simplest Context-Free Grammars, without and with parameters. (Parameters allow us to express more interesting facts.)

```
sentence    → noun_phrase  verb_phrase

noun_phrase → proper_name
noun_phrase → article  noun

verb_phrase → verb
verb_phrase → verb  noun_phrase
verb_phrase → verb  noun_phrase
                     prep_phrase
verb_phrase → verb  prep_phrase

prep_phrase → preposition  noun_phrase
```

Context-Free Grammars

The still-undefined syntactic units are *pre terminals*. They correspond to parts of speech. We can define them by adding lexical productions to the grammar:

```
article → the  |  a  |  an
noun → pizza  |  bus  |  boys  |  ...
preposition → to  |  on  |  ...
proper_name → Jim  |  Dan  |  ...
verb → ate  |  yawns  |  ...
```

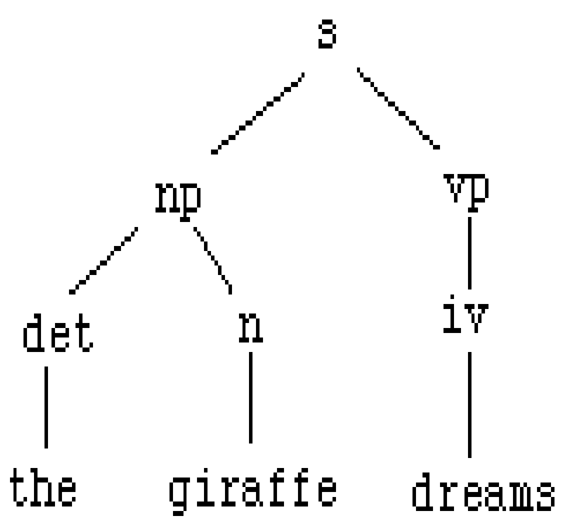
Not practical on a large scale. Normally, we have a lexicon (dictionary) stored in a database, that can be interfaced with the grammar.

```
sentence ↲
noun_phrase verb_phrase ↲
proper_name verb_phrase ↲
Jim verb_phrase ↲
Jim verb noun_phrase prep_phrase ↲
Jim ate noun_phrase prep_phrase ↲
Jim ate article noun prep_phrase ↲
Jim ate a noun prep_phrase ↲
Jim ate a pizza prep_phrase ↲
Jim ate a pizza preposition noun_phrase ↲
Jim ate a pizza on noun_phrase ↲
Jim ate a pizza on article noun ↲
Jim ate a pizza on the noun ↲
Jim ate a pizza on the bus
```

- E.g., the rule s --> np vp means that "a sentence is defined as a noun phrase followed by a verb phrase." Figure 1 shows a simple CFG that describes the sentences from a small subset of English.

Figure 1. A grammar and a parse tree for "the giraffe dreams".

s	→	np vp
np	→	det n
vp	→	tv np
	→	iv
det	→	the
	→	a
	→	an
n	→	giraffe
	→	apple
iv	→	dreams
tv	→	eats
	→	dreams



s => np vp => det n vp => the n vp => the giraffe vp => the giraffe iv => the giraffe dreams

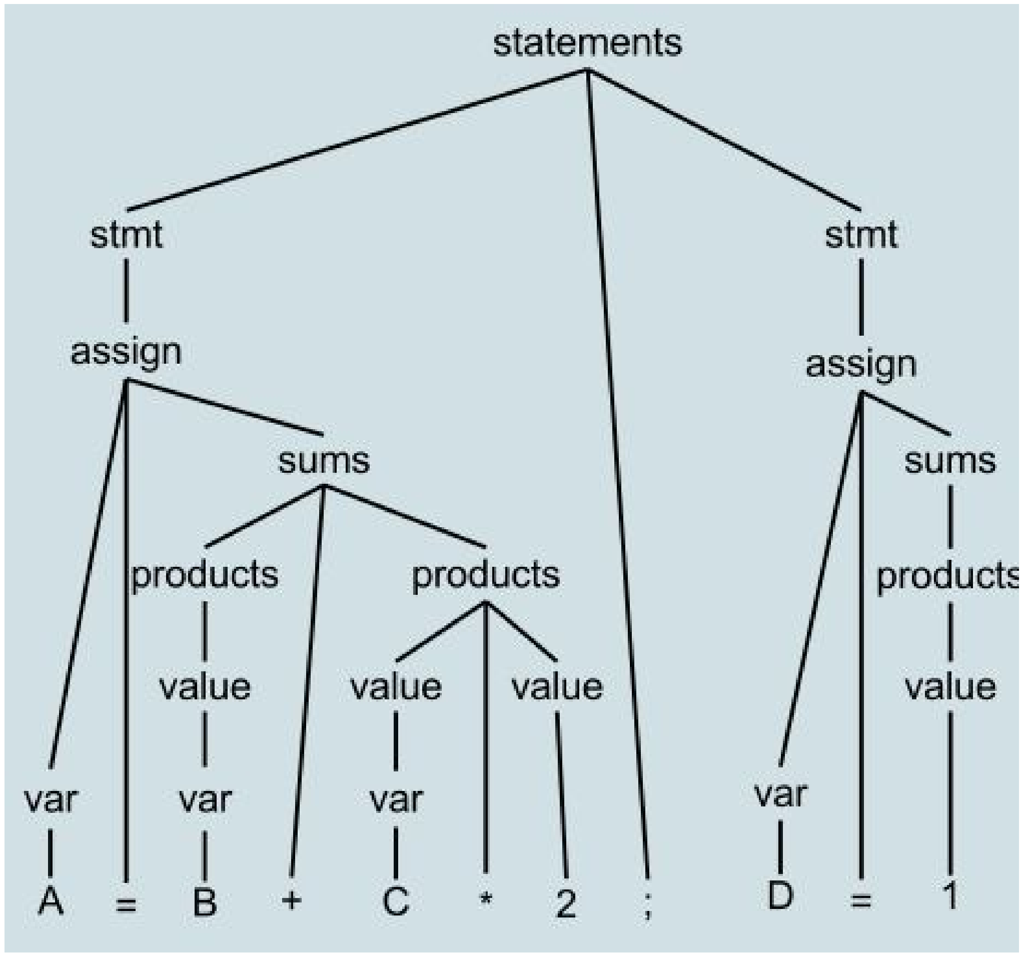
- A sentence in the language defined by a CFG is a series of words that can be derived by
- systematically applying the rules, beginning with a rule that has s on its left-hand side. A
- parse of the sentence is a series of rule applications in which a syntactic category is replaced
- by the right-hand side of a rule that has that category on its left-hand side, and the final

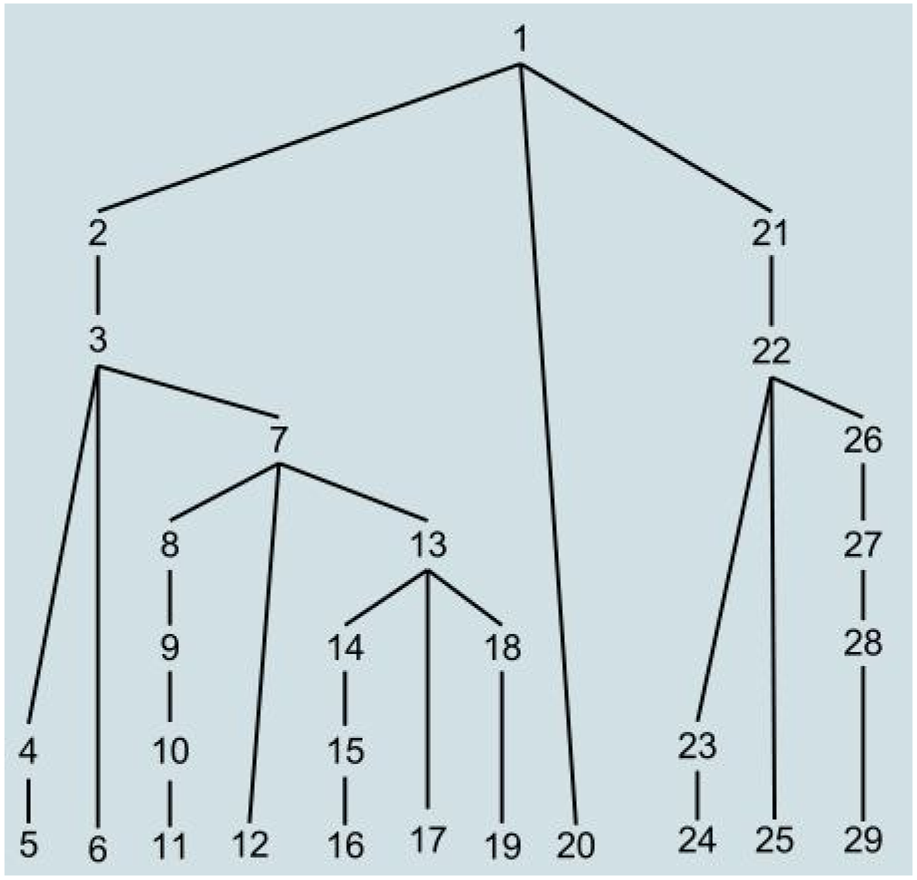
- rule application yields the sentence itself. E.g., a parse of the sentence "the giraffe dreams" is:

- s => np vp => det n vp => the n vp => the giraffe vp => the giraffe iv => the giraffe dreams

Top down vs Bottom up Parsing

- There must be a way to generate the sentence back from Start symbol. So for the purpose, there are 2 ways:
  - Top down parsing
  - Bottom up parsing
- Top-Down parsing
  - Start at the root of tree.(start symbol)
  - Picks a production and tries to match a rule forward till the symbols at the terminals of the tree correspond to the components of the sentence.
  - May require backtracking.


statements
stmt
stmt
assign
assign
sums
sums
products
products
products
value
value
value
value
var
var
var
var
A
=
B
+
C
*
2
;
D
=
1

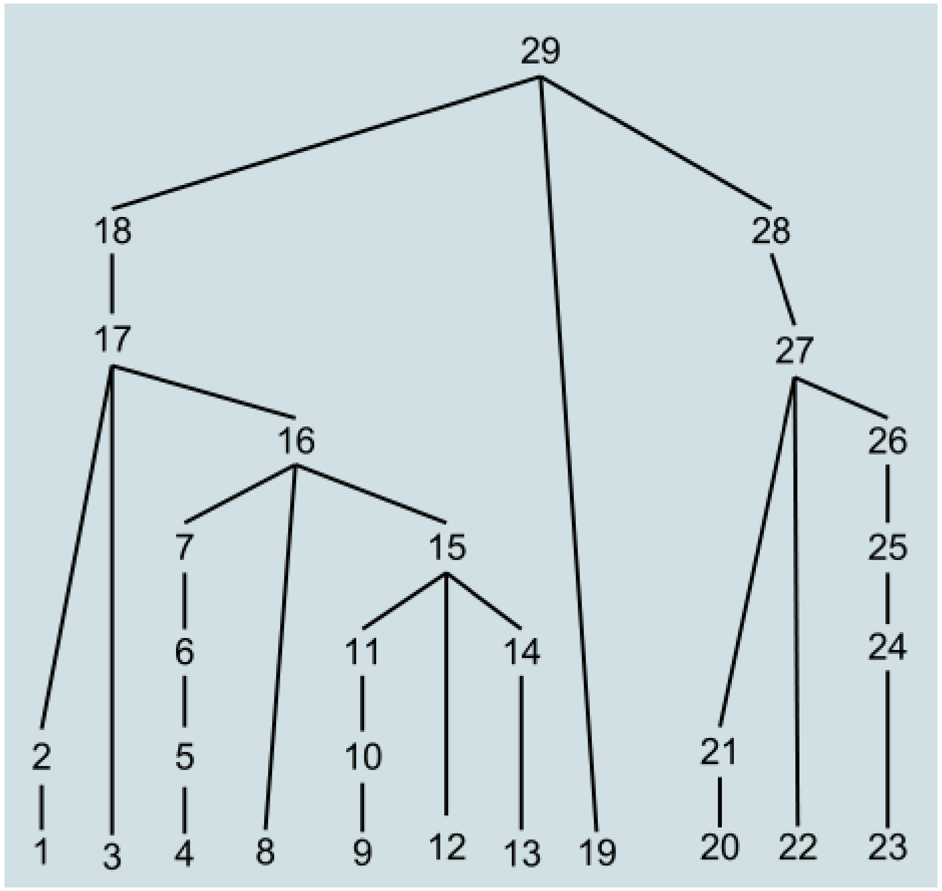

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

Bottom Up Parsing

- Starts at the leaves (ie with the sentence).

- Apply grammar rules backward till a single tree whose terminals are the words of the sentence.

- A bottom-up parse discovers and processes that tree starting from the bottom left end, and incrementally works its way upwards and rightwards


29
18
28
17
27
16
26
7
15
25
6
11
14
24
2
5
10
21
1
3
4
8
9
12
13
19
20
22
23

Finding one interpretation or finding many

- “ Have the students who missed the exam take it today”
  - “Have” is the main verb
- “Have the students who missed the exam taken it today?”
  - “ Have” is the auxiliary verb

- There are four ways for handling such cases:

- 1. All Paths:
  - Follow all possible paths
  - Build all possible immediate components.
  - Many components can be ignored later as other i/p require to use them doesn't appear.
- 2.Best path with backtracking:
  - Follow only one path at a time
  - But keep record of every choice point (so that later if one choice fails and another choice has to be made)

- 3. Best path with patch up:
  - Follow only one path at a time
  - When error is detected , components need to be shuffled that have already been formed.
- 4. Wait and see:
  - Follow only one path
  - Don't make decisions about the function of each components
  - Procrastinate the decision till enough info is available to make decision.

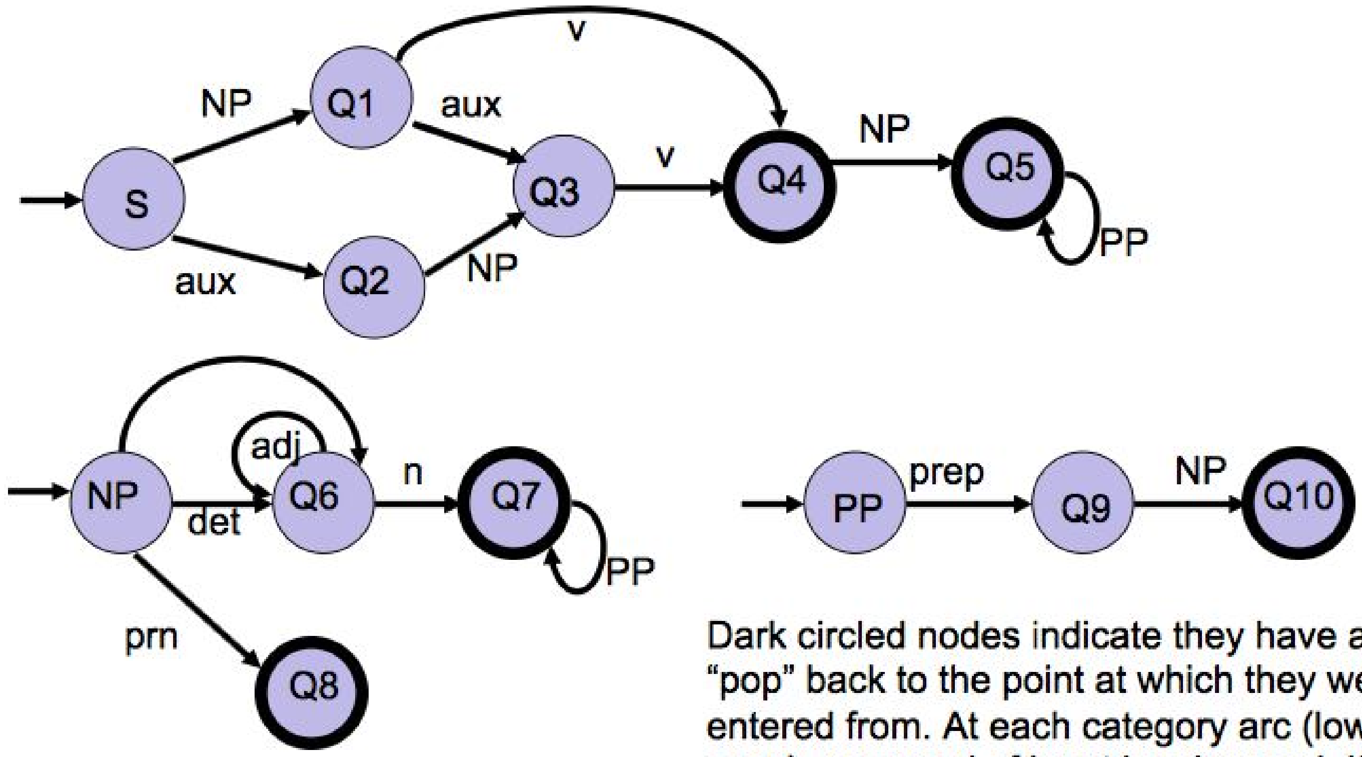
ATNs(another parsing procedure)

- An Augmented Transition Network is a type of graph theoretic structure used especially in parsing relatively complex natural languages, and having wide application in AI.
- ATNs build on the idea of using finite state machines to parse sentences.
- A grammatically correct sentence is parsed by reaching a final state in any state graph.

Components of an ATN

- ATNs are built on the idea of using finite state machines to parse sentences.
- ATNs have states to mark the progress in processing a string.
- Transitions correspond to individual words or phrases from a syntactic type.
- A collection of transition graphs are built for a particular sentence.
- A grammatically correct sentence is parsed by reaching a final state in any state graph.
- Transitions between these graphs are simply subroutine calls from one state to any initial state on any graph in the network.
- A sentence is determined to be grammatically correct if a final state is reached by the last word in the sentence.

An Example of ATN



Dark circled nodes indicate they have a "pop" back to the point at which they were entered from. At each category arc (lower case), one word of input is advanced. If no arc can be taken, the system backtracks.

- Tracing the execution of the above ATN for parsing the following sentence:

The long file has printed.

1. Begin in state S.
2. Push to NP.
3. Do a category test to see if "the" is a determiner.
4. This test succeeds, so set the DETERMINER register to DEFINITE and go to state Q6.
5. Do a cat test to see if "long" is an adjective.
6. This test succeeds, so append "long" to the list contained in the ADJS register. Stay in state Q6.

7. Do a cat test if “ file” is an adj. Test fails.

8. Do a cat test if “ file” is a noun. Test succeeds, set the NOUN regi to ‘file’ and go to state Q7.

9.Push to PP.

10. Do a cat test if ‘has’ is a prep. Test fails, so pop and signal failure.

11. Nothing can be done from state Q7, so pop and return structure (NP(FILE(LONG)DEFINITE))

(The return causes the m/c to be in state Q1(with SUBJ regi set to structure and TYPE regi to DCL)

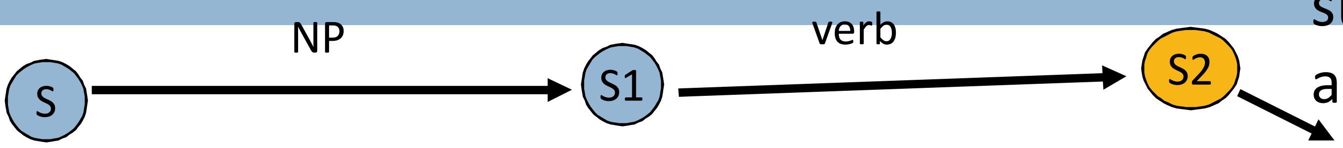
12. Do a cat test if ‘has’ a verb. Test succeeds, so set AUX regi to NIL and set V regi to ‘has’. Go to state Q4.

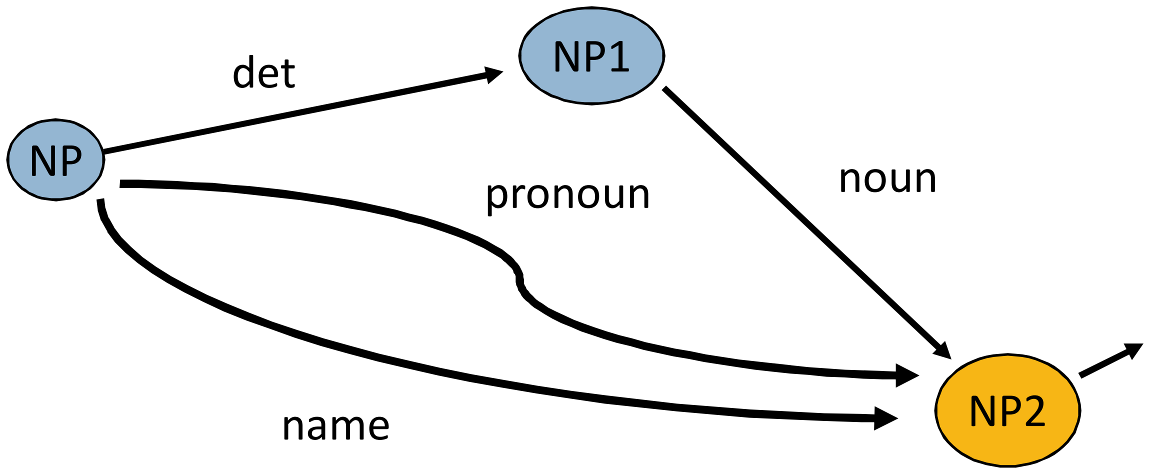
- 13. Push to state NP. As the next word 'printed' is neither a determiner nor a proper noun, NP will pop and return failure.
- 14. Halt in state Q4. But complete parse has not been found. Backtrack.
- 15. The last choice pointer was at state Q1, so return there. The regi AUX and V must be unset.
- 16. Do a cat test if 'has' is an auxiliary. Test succeeds., set AUX regi to 'has'. Go to state Q3.
- 17. Do a cat test if 'printed' is a verb. Test succeeds., set V regi to 'printed'. Go to state Q4.
- 18. Now as i/p is exhausted, Q4 is acceptable final state. Pop and return the structure .

  (S DCL (NP (FILE (LONG) DEFINITE))HAS(VP PRINTED))

Another Example of ATN

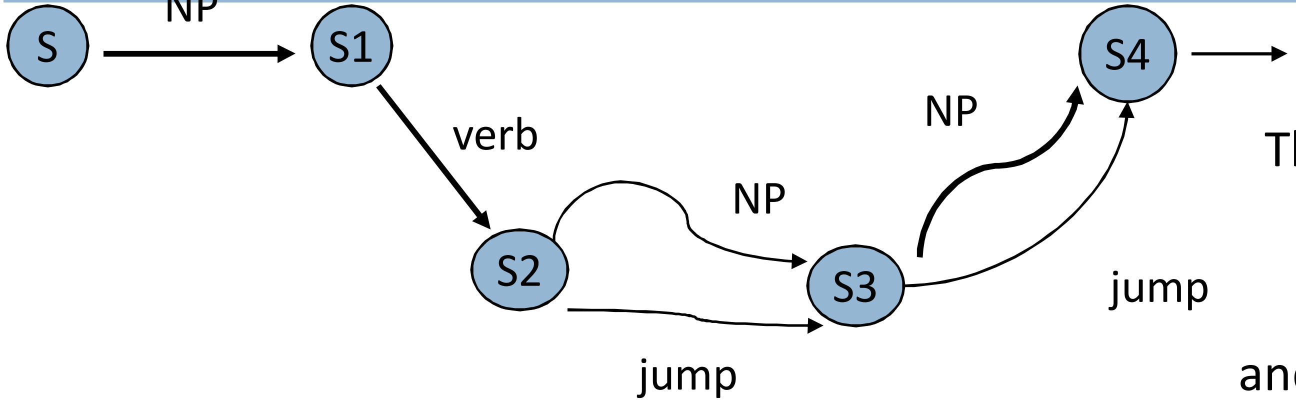
This ATN specifies the structure of a sentence.





•The bottom ATN corresponds to its NP transition and would be used to process a noun phrase as subject of the sentence.

A more complex example of an ATN for a sentence



The jump transitions are for optional components

and consume no input.

ATN for parsing

- Mary gave me a new picture
- Mary sings.
- etc

Parsing with an ATN

1. Set the ATN pointer to [S] and the source pointer to the 1st word of the sentence. The ATN pointer points to the current node.
2. Select an arc out of the current node which satisfies any associated test and, if the arc is labeled with a word category, the current word must be a member of that category.
3. Execute any actions associated with the arc and:
   - If the arc is a word category, update the current position in the source sentence by one word and change the current node to the destination of the arc
   - If the arc corresponds to another ATN, push the starting node of the ATN onto the ATN pointer
   - If the arc is jump, change the current node to the destination of that arc
   - If the arc is done, pop the current node off of the ATN pointer and set * to the value returned by this node. If the ATN pointer is now empty and all of the text has been processed, return *. If the ATN pointer is empty and there is remaining text, fail. Otherwise, return to step 2.

Example parse : Mary gave me a new picture

ATN Pointer	Source Pointer	Variables and Values
S	Mary	
NP S	Mary	
NP2 S	gave	Name = Mary
S	gave	* = (NP (Name Mary))
S1	gave	
S2	me	Verb = (V (Verb give))
NP S2	me	Pronoun = me
NP2 S2	a	
S2	a	
S3	a	OJB = (NP (Pronoun M))
NP S3	a	
NP1 S3	new	
NP1 S3	picture	DET = a
NP2 S3		ADJS = (new)
S3		NOUN = picture
S4		

Semantic Analysis

- Step1: Lexical Processing
  - Meaning of Words:
  - Word Sense Disambiguation
  - Semantic Markers

    (simple properties of word senses)

Semantic Analysis (1. List Processing)

- Lexical Disambiguation (word sense disambiguation)
- Word sense disambiguation (WSD) is the problem of determining which "sense" (meaning) of a word is activated by the use of the word in a particular context
- WSD is a natural classification problem: Given a word and its possible senses, as defined by a dictionary, classify an occurrence of the word in context into one or more of its sense classes.
- The features of the context (such as neighboring words) provide the evidence for classification.

- A famous example is to determine the sense of pen in the following passage (Bar-Hillel 1960):
- Little John was looking for his toy box. Finally he found it. The box was in the pen. John was very happy.
- *WordNet* lists five senses for the word pen:
  - pen — a writing implement with a point from which ink flows.
  - pen — an enclosure for confining livestock.
  - playpen, pen — a portable enclosure in which babies may be left to play.
  - penitentiary, pen — a correctional institution for those convicted of major crimes.
  - pen — female swan.

- Another example consider the sentence " *Put the apple in the basket on the shelf* ". There are two semantic interpretations for this sentence. Using a form of logic for the semantics:

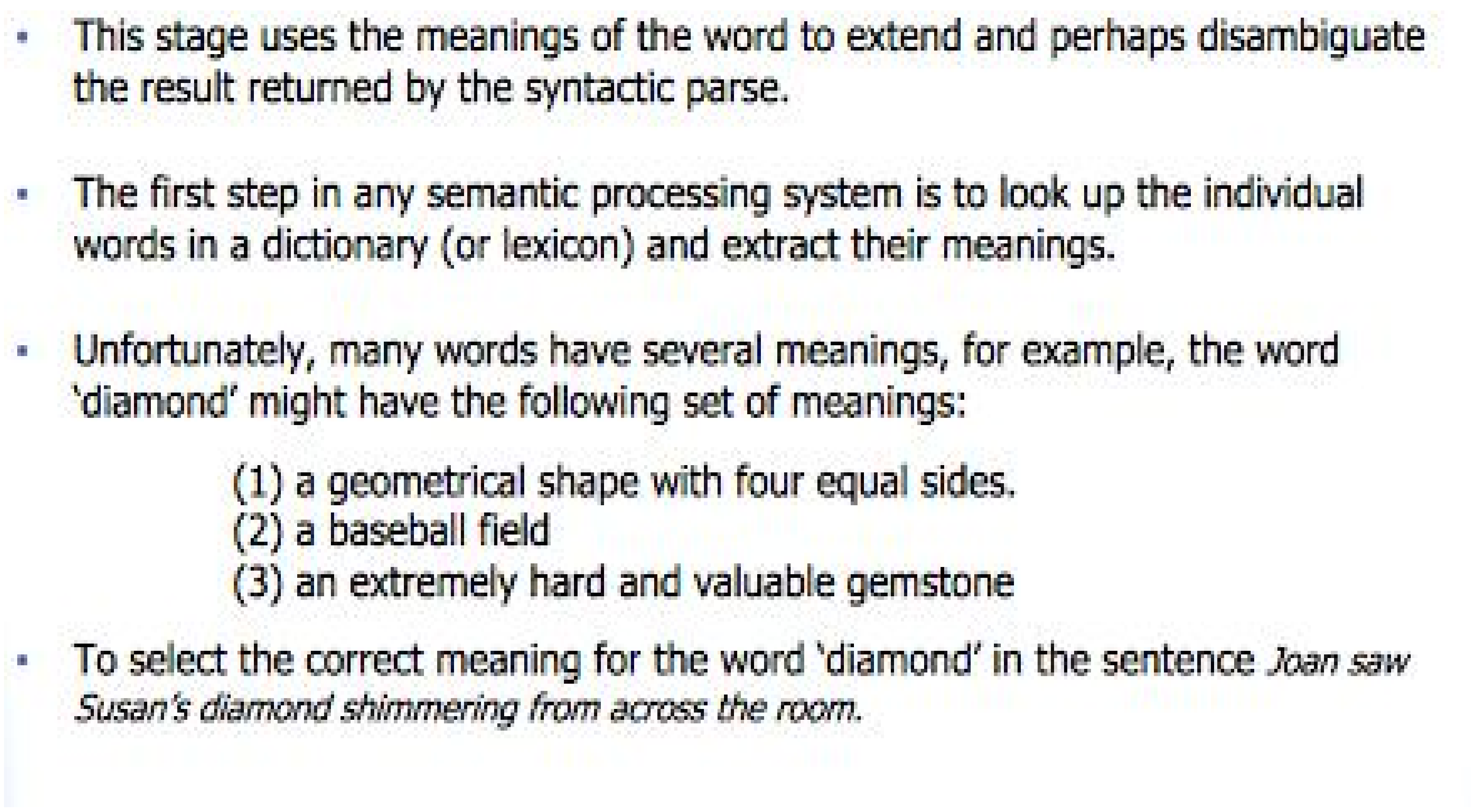
1. put the apple which is currently in the basket onto the shelf

$$(\exists_1 a : apple \mid \exists b : basket \wedge inside(a, b)) \wedge \exists_1 s : shelf \Rightarrow puton(a, s)$$

2. put the apple into the basket which is currently on the shelf

$$\exists_1 a : apple \wedge (\exists_1 b : basket \mid \exists_1 s : shelf \wedge on(b, s)) \Rightarrow putin(a, b)$$

Lexical Processing



- This stage uses the meanings of the word to extend and perhaps disambiguate the result returned by the syntactic parse.
- The first step in any semantic processing system is to look up the individual words in a dictionary (or lexicon) and extract their meanings.
- Unfortunately, many words have several meanings, for example, the word 'diamond' might have the following set of meanings:

  (1) a geometrical shape with four equal sides.
  (2) a baseball field
  (3) an extremely hard and valuable gemstone
- To select the correct meaning for the word 'diamond' in the sentence *Joan saw Susan's diamond shimmering from across the room.*

- It is necessary to know that neither geometrical shapes nor baseball fields shimmer, whereas gemstones do (process of elimination).
- The process of determining the correct meaning of an individual word is call word sense disambiguation or lexical disambiguation.
- It is done by associating, with each word in the lexicon, information about the contexts in which each of the word's senses may appear.
- Each of the words in a sentence can serve as part of the context in which the meanings of the other words must be determined.

2. Sentence Level Parsing

- There are Several Approaches
  - 1. Semantic Grammars
  - 2. Case Grammar
  - 3. Conceptual Parsing
  - 4. Approximately Compositional Semantic interpretation

Semantic grammar

- CFG in which choice of NTs and production rules is governed by syntactic as well as semantic func.

- There is usually a *semantic action* associated with each grammar rule.

- Combine *syntactic, semantic and pragmatic knowledge* into single set of production rules in the form of grammar.

Semantic Grammar

- Semantic description
- Meaning=Result of parsing + applying semantic actions
- Grammar rule based on key semantic concepts

Advantages n Disadvantages

- Adv.
  - Immediate results after parsing
  - Avoid many ambiguities
- Disadv
  - More rules
  - Parsing may be expensive

2. Case Grammar

- Case grammar focuses on the link between number of subjects, objects, etc., of a verb and the grammatical context it requires.
- This theory analyzes the surface syntactic structure of sentences by studying the combination of deep cases (i.e. semantic roles) -- Agent, Object, Benefactor, Location or Instrument etc—*which are required by a specific verb.*

- For instance, the verb "give" in English requires an Agent (A) and Object (O), and a Beneficiary (B).
- e.g. "Jones (A) gave money (O) to the school (B).

Consider the 2 examples:

- Susan printed the file.
- The file was printed by Susan.

The roles of "Susan" and " the file" are same in the two sentences.
But their syntactic roles are reversed.

Using a case grammar, the interpretation of both the sentences would be

(printed(agent Susan)
(object File))

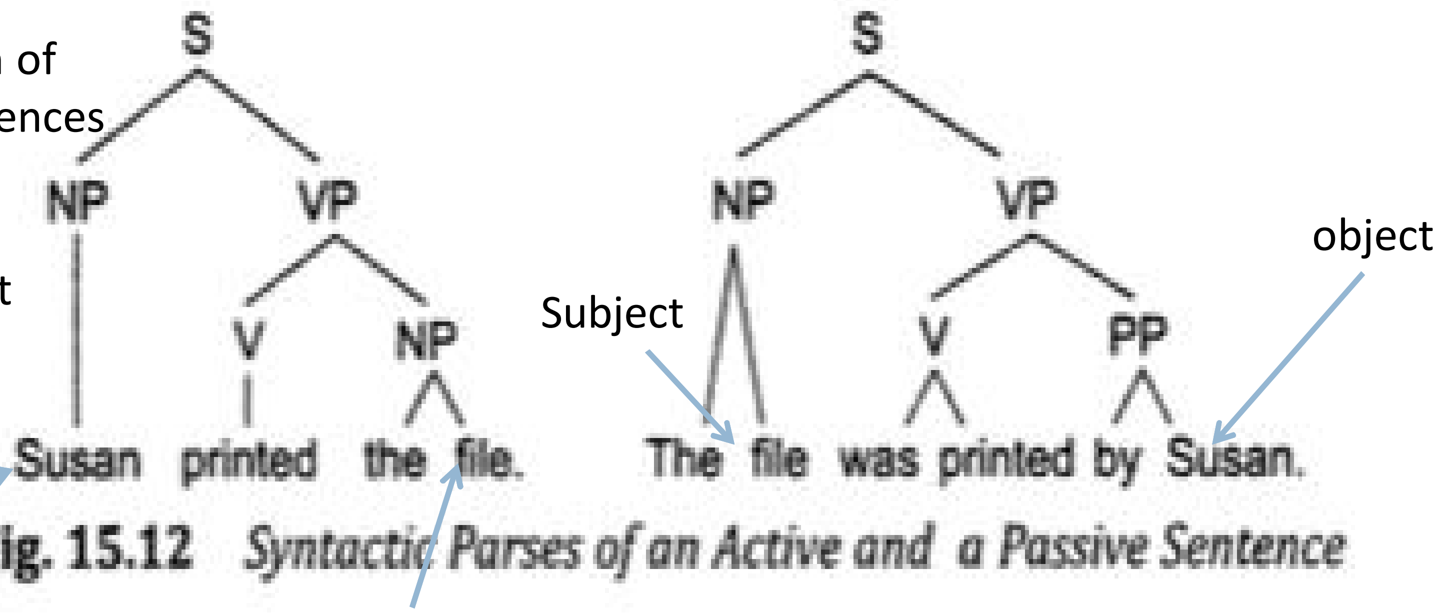


Fig. 15.12 Syntactic Parses of an Active and a Passive Sentence

Consider the following 2 similar sentences:
Mother baked for 3 hours.
The pie baked for 3 hours.

‘The pie’ is Subject of ‘baked’

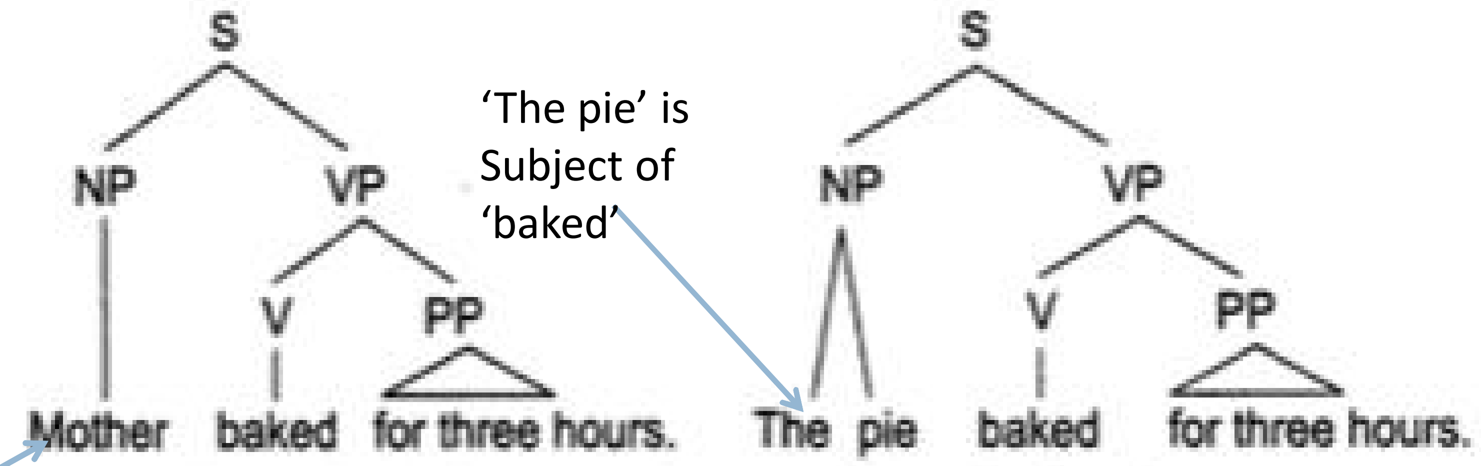


Fig. 15.13 *Syntactic Parses of Two Similar Sentences*

Subject of ‘baked’

The relation between Mother and baking is very different from the relation of pie and baking. A case grammar analysis of these 2 sentences capture the difference.
The first sentence would be interpreted as:
(baked (agent Mother)
(timeperiod 3 –hours))
The second:
(baked(object Pie)
(timeperiod 3-hours))

Case Grammar

- Case Grammar is expectation-driven.
- Once the verb of the sentence has been identified, it can be used to predict the NP that will occur and to determine the relationships of those phrases to the rest of the sentence.

Conceptual Parsing

- Similar to Semantic Grammar.
- Tries to find the structure and meaning of a sentence in a single step.
- The difference is, it is driven by a dictionary that describes the meanings of words as conceptual dependency structures.

- Parsing a sentence into CD representation is similar to the process of parsing in case grammar.

- Parsing process heavily relies on Sentence's main Verb, like in case grammar.

Statistical NLP

- Corpus is a large collection of texts. It is a body of written or spoken material upon which a linguistic analysis is based. The plural form of corpus is corpora. Some popular corpora are British National Corpus (BNC)
- European Corpus Initiative (ECI) corpus is multilingual having 98 million words in Turkish, Japenese, Russian, Chinese, and other languages.

NLP

- The corpus may be composed of written language, spoken language or both.
- Spoken corpus is usually in the form of audio recordings.
- A corpus may be open or closed.
- An open corpus is one which does not claim to contain all data from a specific area while a closed corpus does claim to contain all or nearly all data from a particular field.
- Historical corpora, for example, are closed as there can be no further input to an area.

Terminologies

- Sub language Corpora: Some corpora contain texts on a particular domain of study.
- A Treebank is a parsed text corpus that annotates syntactic or semantic sentence structure.
- Treebanks are often created on top of a corpus that has already been annotated with part-of-speech tags.

- Parallel Corpora:
  - A parallel corpus is a collection of texts, each of which is translated into one or more other languages than the original.
- Concordance:
  - A list showing all the occurrences and contexts of a given word or phrase, as found in a corpus. It also can give the notion of how often a word occurs.

- Collocation:
  - A collection of words that are often observed together in a text.
  - Eg. Crystal clear', 'middle management', 'nuclear family', and 'cosmetic surgery' , extremely beautiful are examples of collocated pairs of words.

Counting elements in a Corpus

- Frequency of the no. of words in a corpus can yield valuable info about the prob. Of the occurrence of a word for an incomplete string.
- There are no. of points that can be considered at this point:
  - Whether “book” and “books” be considered as one or 2 words.
  - Whether punctuation marks be considered.
  - Whether case sensitization be considered.

- Consider:
  - The former means the number of distinct words in the corpus while the latter stands for total number of words in the corpus.
  - Type: 14
  - Tokens: 24

Natural Language Processing

- Points
- Areas, problems, challenges
- Levels of language description
- Generation and analysis
- Strategies for analysis
- Analyzing words
- Linguistic anomalies
- Parsing
- Simple context-free grammars
- Direction of parsing
- Syntactic ambiguity

Areas, problems, challenges

- **Language and communication**
  - Spoken and written language.
  - Generation and analysis of language.
- Understanding language may mean:
  - accepting new information,
  - reacting to commands in a natural language,
  - answering questions.

Problems and difficult areas

- Vagueness and imprecision of language:
- redundancy (many ways of saying the same)
- ambiguity (many senses of the same data).

- Non-local interactions, peculiarities of words.

- Non-linguistic means of expression (gestures, ...).

□Challenges

- Incorrect language data—robustness needed.
- Narrative, dialogue, plans and goals.
- Metaphor, humour, irony, poetry.

Levels of language description

- Phonetic—acoustic:
  - speech, signal processing.
- Morphological—syntactic:
  - dictionaries, syntactic analysis,
  - representation of syntactic structures, and so on.
- Semantic—pragmatic:
  - world knowledge, semantic interpretation,
  - discourse analysis/integration,
  - reference resolution,
  - context (linguistic and extra-linguistic), and so on.
- Speech generation is relatively easy: *analysis* is difficult.
  - We have to segment, digitize, classify sounds.
  - Many ambiguities can be resolved in context (but storing and matching of long segments is unrealistic).
  - Add to it the problems with written language.

Generation and analysis

- Language generation
  - from meaning to linguistic expressions;
  - the speaker's goals/plans must be modelled;
  - stylistic differentiation;
  - good generation means variety.
- Language analysis
  - from linguistic expressions to meaning
- (representation of meaning is a separate problem);
  - the speaker's goals/plans must be recognized;
  - analysis means standardization.
- Generation and analysis combined: machine translation
  - word-for-word (very primitive);
  - transforming parse trees between analysis and generation;
  - with an intermediate semantic representation.

Strategies for analysis

- Syntax, then semantics (the boundary is fluid).
- In parallel (consider subsequent syntactic fragments, check their semantic acceptability).
- No syntactic analysis (assume that words and their one-on-one combinations carry all meaning) -- this is quite extreme...

- Syntax deals with structure:
- how are words grouped? how many levels of description?
- formal properties of words (for example, part-of-speech or grammatical endings).

- Syntactic correctness does not necessarily imply acceptability.
- A classic example of a well-formed yet meaningless clause:
  - Colourless green ideas sleep furiously.

Strategies for analysis (2)

- Syntax mapped into semantics
  - Nouns ↔ things, objects, abstractions.
  - Verbs ↔ situations, events, activities.
  - Adjectives ↔ properties of things, ...
  - Adverbs ↔ properties of situations, ...
- Function words (from closed classes) signal relationships.
- The role and purpose of syntax
  - It allows partial disambiguation.
  - It helps recognize structural similarities.
    - "He bought a car" — "A car was bought [by him]" —
      - "Did he buy a car?" — "What did he buy?"
- A well-designed NLP system should recognize these forms as variants of the same basic structure.

Analyzing words

- Morphological analysis usually precedes parsing. Here are a few typical operations.
  - Recognize root forms of inflected words and construct a standardized representation, for example:
    - books ⇔ book + PL, skated ⇔ skate + PAST.
  - Translate contractions (for example, he'll → he will).
- [We will not get into any details, other than to note that it is fairly easy for English, but not at all easy in general.]
- Lexical analysis looks in a dictionary for the meaning of a word. [This too is a highly simplified view of things.]
- Meanings of words often "add up" to the meaning of a group of words. [See examples of conceptual graphs.] Such simple composition fails if we are dealing with metaphor.

Analyzing words (2)

- Morphological analysis is not quite problem-free even for English. Consider recognizing past tense of regular verbs.
- blame → blame+d, link → link+ed, tip → tip+p+ed
- So, maybe cut off d or ed? Not quite: we must watch out for such words as “bread” or “fold”.
- The continuous form is not much easier:
- blame → blam-e+ing, link → link+ing, tip → tip+p+ing
- Again, what about “bring” or “strong”?
- give → given but mai → main ??
- Morphological analysis allows us to reduce the size of the dictionary (lexicon), but we need a list of exceptions for every morphological rule we invent.

Linguistic anomalies

Pragmatic anomaly

Next year, all taxes will disappear.

Semantic anomaly

The computer ate an apple.

Syntactic anomaly

The computer ate apple.

An the ate apple computer.

Morphological anomaly

The computer eated an apple.

Lexical anomaly

Colourless	green	ideas	sleep	furiously	WRONG
↑	↑	↑	↑	↑	
adjective	adjective	noun	verb	adverb	
↓	↓	↓	↓	↓	
Heavy	dark	chains	clatter	ominously	CORRECT

Parsing

Syntax is important: it is the “skeleton” on which we hang various linguistic elements, meaning among them.

So, recognizing syntactic structure is also important.

Some researchers deny syntax its central role. There is a verb-centred analysis that builds on Conceptual Dependency [textbook, section 7.1.3]: a verb determines almost everything in a sentence built around it. (Verbs are fundamental in many theories of language.)

Another idea is to treat all connections in language as occurring between pairs of words, and to assume no higher-level groupings. Structure and meaning are expressed through variously linked networks of words.

Parsing (syntactic analysis) is based on a grammar. There are many subtle and specialized grammatical theories and formalisms for linguistics and NLP alike:

Categorial Grammars	Indexed Grammars
Context-Free Grammars	Lexical-Functional Grammars
Functional Unification Grammars	Logic Grammars
Generalized LR Grammars	Phrase Structure Grammars
Generalized Phrase Structure Grammars	Tree-Adjoining Grammars
Head-Driven Phrase Structure Grammars	Unification Grammars

and many more

Simple context-free grammars

We will look at the simplest Context-Free Grammars, without and with parameters. (Parameters allow us to express more interesting facts.)

```
sentence    → noun_phrase  verb_phrase

noun_phrase → proper_name
noun_phrase → article  noun

verb_phrase → verb
verb_phrase → verb  noun_phrase
verb_phrase → verb  noun_phrase
                     prep_phrase
verb_phrase → verb  prep_phrase

prep_phrase → preposition  noun_phrase
```

Simple CF grammars (2)

The still-undefined syntactic units are *preterminals*. They correspond to parts of speech. We can define them by adding lexical productions to the grammar:

```
article → the  |  a  |  an
noun → pizza  |  bus  |  boys  |  ...
preposition → to  |  on  |  ...
proper_name → Jim  |  Dan  |  ...
verb → ate  |  yawns  |  ...
```

This is not practical on a large scale. Normally, we have a lexicon (dictionary) stored in a database, that can be interfaced with the grammar.

Simple CF grammars (3)

```
sentence ↵
noun_phrase verb_phrase ↵
proper_name verb_phrase ↵
Jim verb_phrase ↵
Jim verb noun_phrase prep_phrase ↵
Jim ate noun_phrase prep_phrase ↵
Jim ate article noun prep_phrase ↵
Jim ate a noun prep_phrase ↵
Jim ate a pizza prep_phrase ↵
Jim ate a pizza preposition noun_phrase ↵
Jim ate a pizza on noun_phrase ↵
Jim ate a pizza on article noun ↵
Jim ate a pizza on the noun ↵
Jim ate a pizza on the bus
```

Simple CF grammars (4)

Other examples of sentences generated by this grammar:

`Jim ate a pizza`

`Dan yawns on the bus`

These *wrong* data will also be recognized:

`Jim ate an pizza`

`Jim yawns a pizza`

`Jim ate to the bus`

`the boys yawns`

`the bus yawns`

... but not these, obviously correct:

`the pizza was eaten by Jim`

`Jim ate a hot pizza`

and so on, and so forth.

Simple CF grammars (5)

We can improve even this simple grammar in many interesting ways.

- Add productions, for example to allow adjectives.
- Add words (in lexical productions, or in a more realistic lexicon).
- Check agreement (noun-verb, noun-adjective, and so on).

$\texttt{rabbits}_{pl}\ \texttt{run}_{pl} \leftrightarrow \texttt{a rabbit}_{sg}\ \texttt{runs}_{sg}$

$\texttt{le bureau}_{m}\ \texttt{blanc}_{m} \leftrightarrow \texttt{la table}_{f}\ \texttt{blanche}_{f}$

An obvious, but naïve, method of enforcing agreement is to duplicate the productions and the lexical data.

Simple CF grammars (6)

```
sentence      → noun_phr_sg
verb_phr_sg
sentence      → noun_phr_pl
verb_phr_pl
noun_phr_sg → art_sg   noun_sg
noun_phr_sg → proper_name_sg
noun_phr_pl → art_pl   noun_pl
noun_phr_pl → proper_name_pl
art_sg        → the   |   a   |   an
art_pl        → the
noun_sg       → pizza   |   bus   |   ...
noun_pl       → boys   |   ...
```

and so on.

Simple CF grammars (7)

A much better method is to add parameters, and to parameterize words as well as productions:

```
sentence       →  noun_phr(Num)   verb_phr(Num)
noun_phr(Num)  →  art(Num)   noun(Num)
noun_phr(Num)  →  proper_name(Num)
art(sg)        →  the  |  a  |  an
art(pl)        →  the
noun(sg)       →  pizza  |  bus  |  ...
noun(sg)       →  boys  |  ...
```

and so on.

This notations slightly extends the basic Context-Free Grammar formalism.

Simple CF grammars (8)

Another use of parameters in productions: represent transitivity. We want to exclude such sentences as

```
   Jim yawns a pizza
   Jim ate to the bus
```

```
verb_phr(Num) → verb(intrans, Num)
verb_phr(Num)  →
        verb(trans, Num)
noun_phr(Num1)
verb(intrans, sg) → yawns  |  ...
verb(trans, sg) → ate  |  ...
verb(trans, pl) → ate  |  ...
```

Direction of parsing

Top-down, hypothesis-driven: assume that we have a sentence, keep rewriting, aim to derive a sequence of terminal symbols, backtrack if data tell us to reject a hypothesis. (For example, we had assumed a noun phrase that begins with an article, but there is no article.)

Problem: wrong guesses, wasted computation.

Bottom-up, data-driven: look for complete right-hand sides of productions, keep rewriting, aim to derive the goal symbol.

Problem: lexical ambiguity that may lead to many unfinished partial analyses.

Lexical ambiguity is generally troublesome. For example, in the sentence "Johnny runs the show", both *runs* and *show* can be a verb or a noun, but only one of 2*2 possibilities is correct.

Direction of parsing (2)

In practice, parsing is never "pure".

Top-down, enriched: check data early to discard wrong hypotheses (somewhat like recursive-descent parsing in compiler construction).

Bottom-up, enriched: use productions, suggested by data, to limit choices (somewhat like LR parsing in compiler construction).

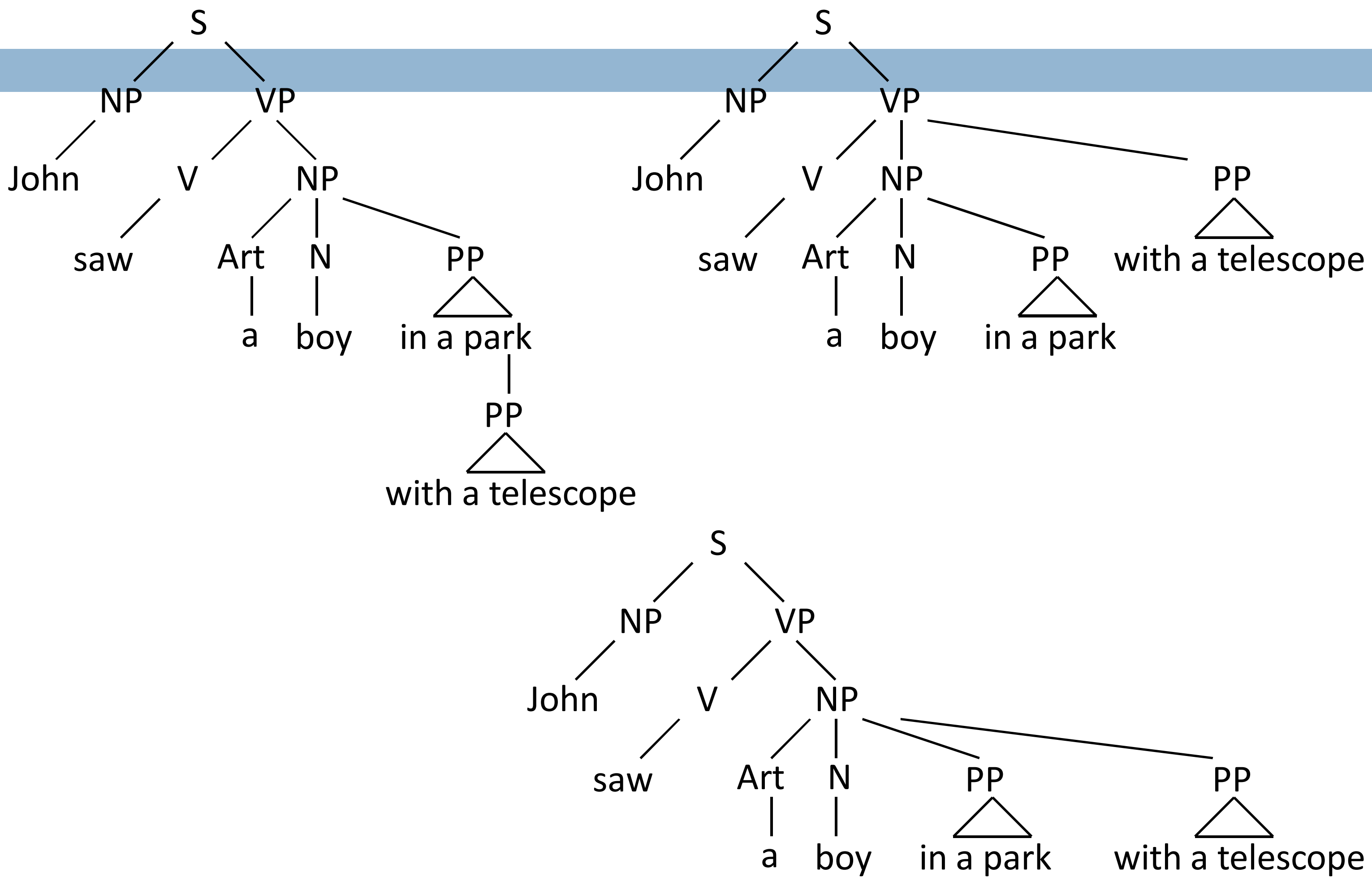
A popular bottom-up analysis method: chart parsing.

Popular top-down analysis methods:

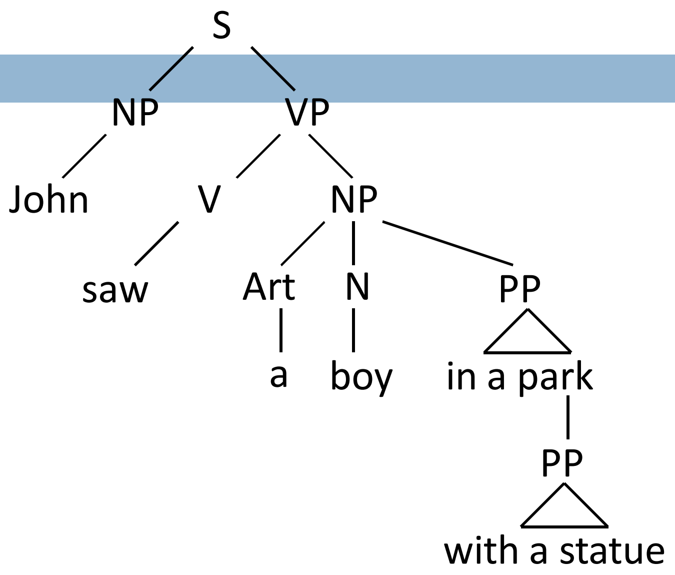
transition networks (used with Lisp),

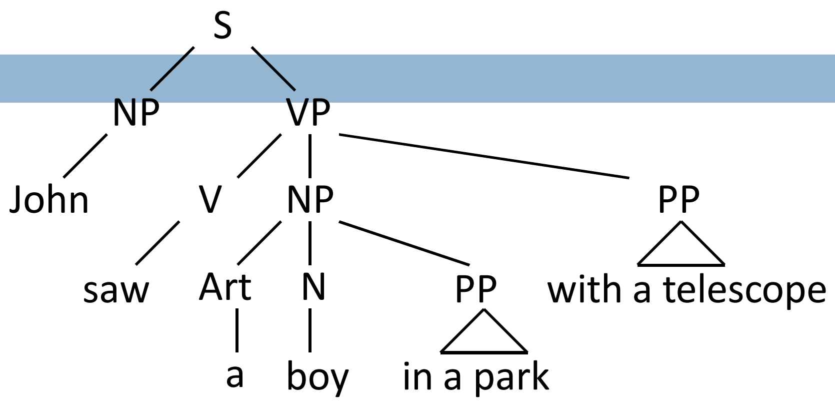
logic grammars (used with Prolog).

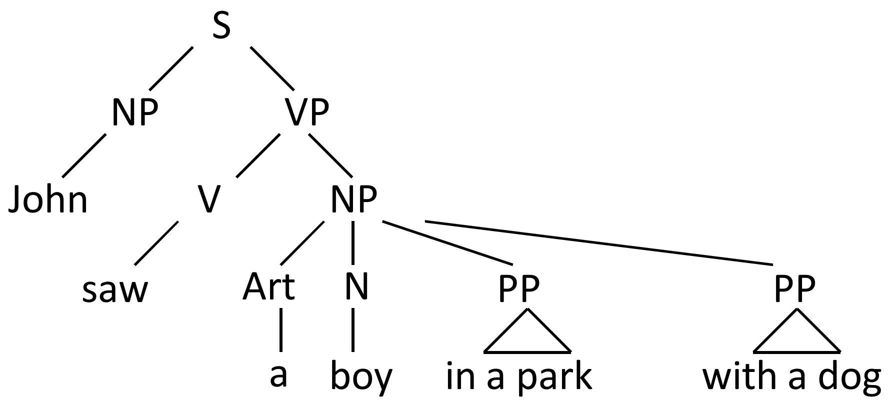
Syntactic ambiguity — a classic example



Syntactic ambiguity resolved semantically







On to Prolog

http://www.site.uottawa.ca/~szpak/teaching/4106/handouts/grammars/