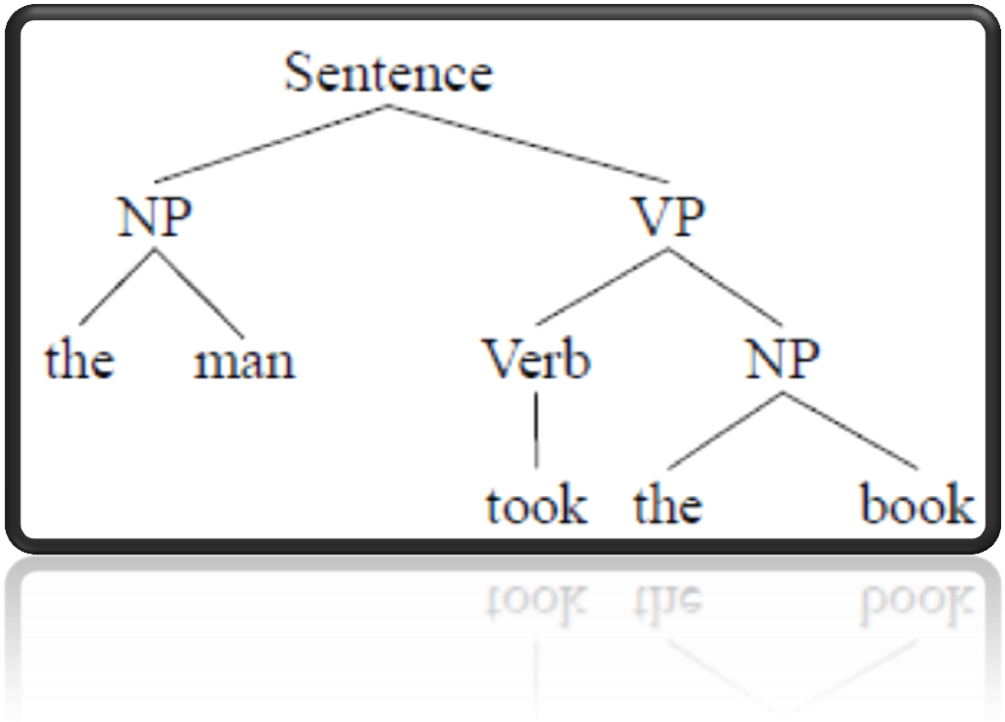
Natural Language Processing



Key Points

- What & Why NLP?
- Steps involves in language processing: Morphological, syntactic, semantic, pragmatic, discourse integration
- **Parsing**: bottom up vs. top down
- Parser Implementation: ATN
- Why semantic is important?
- CFG/CSG/TGG

Any Light at The End of The Tunnel?

- Yahoo, Google, Microsoft → Information Retrieval
- Monster.com, HotJobs.com (Job finders) → Information Extraction + Information Retrieval
- Systran powers Babelfish → Machine Translation
- Ask Jeeves → Question Answering
- Myspace, Facebook, Blogspot → Processing of User-Generated Content
- Tools for "business intelligence"
- All "Big Guys" have (several) strong NLP research labs:
  - IBM, Microsoft, AT&T, Xerox, Sun, etc.
- Academia: Research in an university environment

Natural Language Processing (NLP)

- Natural Language?
  - Refers to the language spoken/written by people, e.g. English, Japanese, Bengali, as opposed to artificial languages, like C++, Java, LISP, Prolog, etc.
- Natural Language Processing
  - Applications that deal with natural language in a way or another
  - Computers use (analyze, understand, generate) natural language
  - A somewhat applied field

- Computational Linguistics
  - Doing linguistics on computers
  - More on the linguistic side than NLP, but closely related
  - More theoretical

Why NLP?

- Language is meant for communicating about the world
  - NLP offers insights into language
- Language is the medium of the web
- Help in communication
  - With computers
  - With other humans (MT)
  - HCI/HRI

Kismet: MIT Media Lab

Why NLP?

Applications for processing large amounts of texts -require NLP expertise

- Classify text into categories
- Index and search large texts
- Automatic translation
- Speech understanding
  - Understand phone conversations
- Information extraction
  - Extract useful information from resumes
- Automatic summarization
  - Condense 1 book into 1 page
- Question answering
- Knowledge acquisition
- Text generations / dialogues

NLP Involves ….

Two tasks:

- Processing written text

-Lexical

-Syntactic

-Semantic

-Pragmatic

- Processing spoken language,

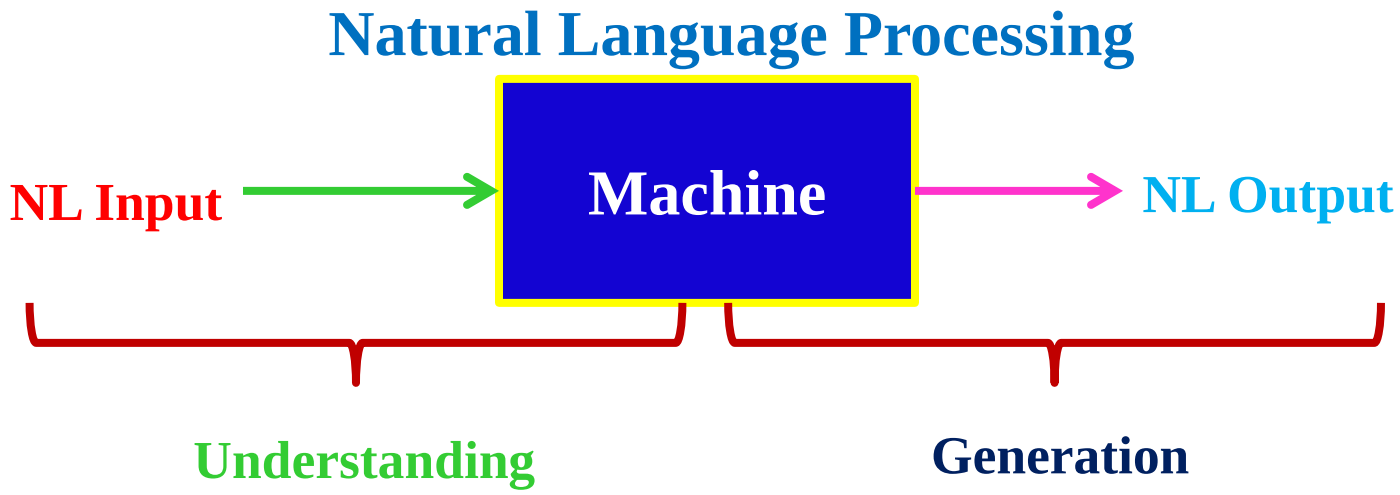
-Lexical, Syntactic, Semantic, Pragmatic+++

- Phonology

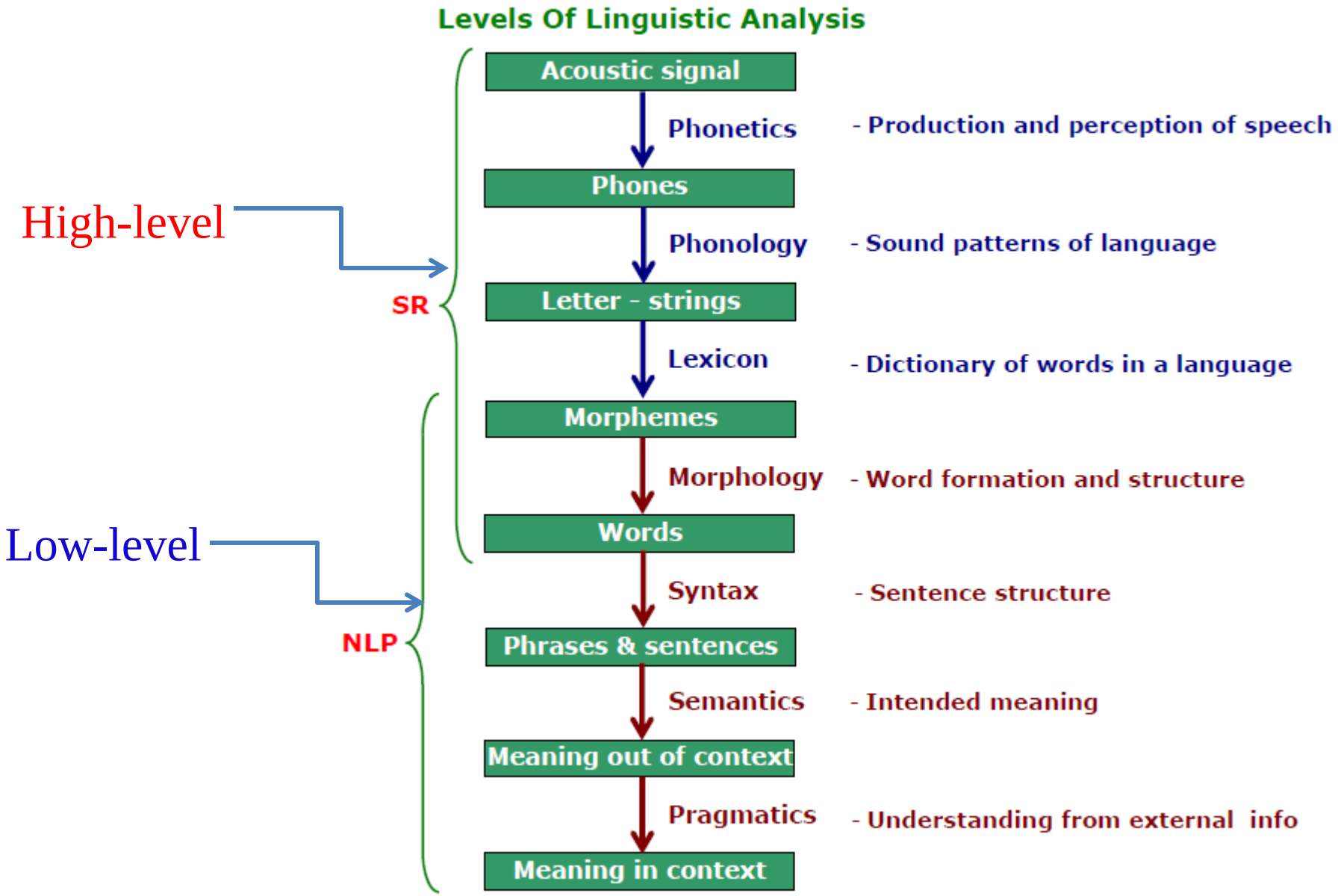
NLP Involves ….

Natural Language Understanding (NLU): understanding & reasoning while the input is NL
-Internal structure of the input NL

Natural Language Generation (NLG): generate other language



Linguistics and Language Processing



Steps in the NLP

- **Morphological/Lexical Analysis**:

  -Individual words are analyzed into their components

  -Non-word tokens such as punctuation are separated from the words.

  -Divide the text into paragraphs, sentences & words

Steps in the NLP

- **Syntactic Analysis:**
  - The word **syntax** comes from the Greek ***s`yntaxis***, *meaning "setting out together* or arrangement", & refers to the way words are arranged together.
  - Linear sequences of words are transformed into structures that show how the words relate to each other.

-Boy the go to store (rejected due to violate the rules)

Steps in the NLP

- **Semantic Analysis**: The structures created by the syntactic analyzer are assigned meanings.
  -Manush gass khai (syntactically correct but semantically not)

- **Discourse Integration**: The meaning of an individual sentence may depend on the sentences that precede it and may influence the meanings of the sentences that follow it.
  -you wanted it (it depends on the prior discourse context)

- **Pragmatic Analysis:** The structure representing what was said is reinterpreted to determine what was actually meant.
  -commonsense knowledge
  -Do you know what is it? (request)

Morphological Analysis

- **Morphology** is the analysis of words into morphemes & conversely the synthesis of words from morphemes

- **Morphemes**

-a smallest meaningful unit in the grammar

-a smallest linguistic unit that has semantic meaning

-a unit of language immediately below the ‘word level’

-a smallest part of a word that can carry a discrete meaning

Morphological Analysis

Example:

The word "unbreakable" has 03 morphemes

1. un- a bound morpheme
2. -break- a free morpheme
3. -able- a bound morpheme

Also, un: prefix, able-suffix--affixes

How many morphemes?
"aagniporikhha" "unreliable",

Types of Morphemes

- **Free Morphemes**:

-can appear stand alone/free

town, dog

-with other lexemes

town hall, dog house

- **Bound Morphemes**

-appear only together with other morphemes to form a lexeme

un-: may be prefix/suffix

- **Inflectional Morphemes**

-modify a word's tense, number, aspects etc

dog morpheme with plural marker

morpheme **s** becomes **dogs**

Types of Morphemes

- **Derivational Morphemes**

-can be added to a word to derive another word

Addition of "-nees" to "happy": "happiness"

- **Root Morphemes**

-primary lexical unit of a word. Roots can be either free or bound morphemes

chatters has the inflectional root chatter but the lexical root chat

- **Null Morphemes**

-if is an invisible affix, also called zero morpheme represented as either the figure zero (0), the empty set symbol ∅, or its variant ∅

-Adding a null morpheme is called null affixation/null derivation/zero derivation;

cat = cat + -0 = ROOT ("cat") + singular

cats = cat + - s = ROOT ("cat") + plural

Syntactic Analysis

❑ Syntax

-Syntax is the structure of language.

-It is the grammatical arrangement of the words in a sentence to show its relationship to one another in a sentence

-Syntax is a finite set of rules that specify a language

-**Rules**: Phase Structure Rules

Context-Free or Context-Sensitive

-Syntax rules govern proper sentence structure

-Syntax is represented by Parse tree(a way to show the structure of a language fragment) or by a List

Syntactic Analysis

He ate the pizza

Input String

PS Rules

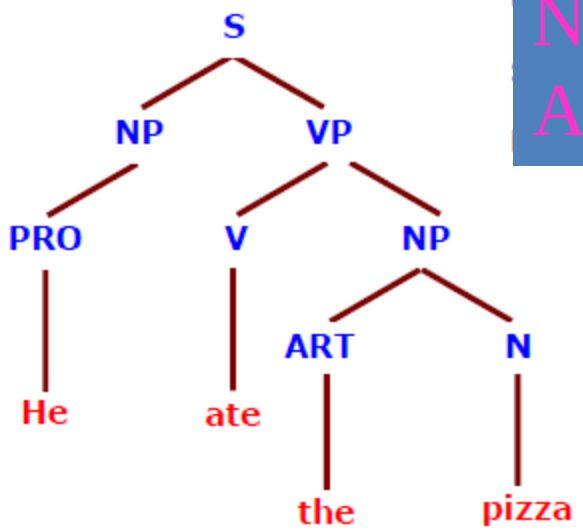
S → NP VP
NP → PRO
NP → ART N
VP → V NP

Parser

Lexicon

V-ate
PRO-he
N-pizza
ART-the

Parse Tree



Classification Phrases

- **Sentence (S)**
- **Noun Phrase** (NP): Noun/Pronoun as head

  -Optionally accompanied by a set of modifiers

  Determiners: article (the, a)

  Adjectives (the red ball)

  the black cat, **a** cat on the mat
- **Verb Phrase** (VP): verb as head

  "eat cheese", "jump up and down"
- **Adjective Phrase** (AP): adjective as head

  full of toys, **good** boy
- **Adverbial Phrase** (**AdvP**): adverb as head

  very carefully
- **Prepositional Phrase** (**PP**): preposition as head

  in love, **over** the rainbow

Classification Phrases

- **Determiner Phrase (DP)**: determiner as head

a little dog, **the** little dogs

- In English, determiners are usually placed before the noun as a noun modifier that includes:

-articles (the, a),

-demonstratives (this, that),

-numerals (two, five, etc.),

-possessives (my, their, etc.),

-quantifiers(some, many, etc.)

PS Rules

- PS rules are a way to describe language syntax.
- Rules determine what goes into phrase & how its constituents are ordered.
- They are used to break a sentence down to its constituent parts namely phrasal categories & lexical categories:

  Phrasal category include: noun phrase, verb phrase, prepositional phrase

  Lexical category include: noun, verb, adjective, adverb, others

PS Rules

- PS rules are usually of the form **A → B C**
  - -Meaning "constituent A is separated into sub-constituents B & C
  - -Or simply " A consists of B followed by C"
- **S → NP VP** Reads: S consists of an NP followed by a VP
  - -Means a sentence consists of a noun phrase followed by a verb phrase
- NP → Det N1 Reads: NP consists of an Det followed by N1
  - -Means a noun phrase consists of a determiner followed by a noun

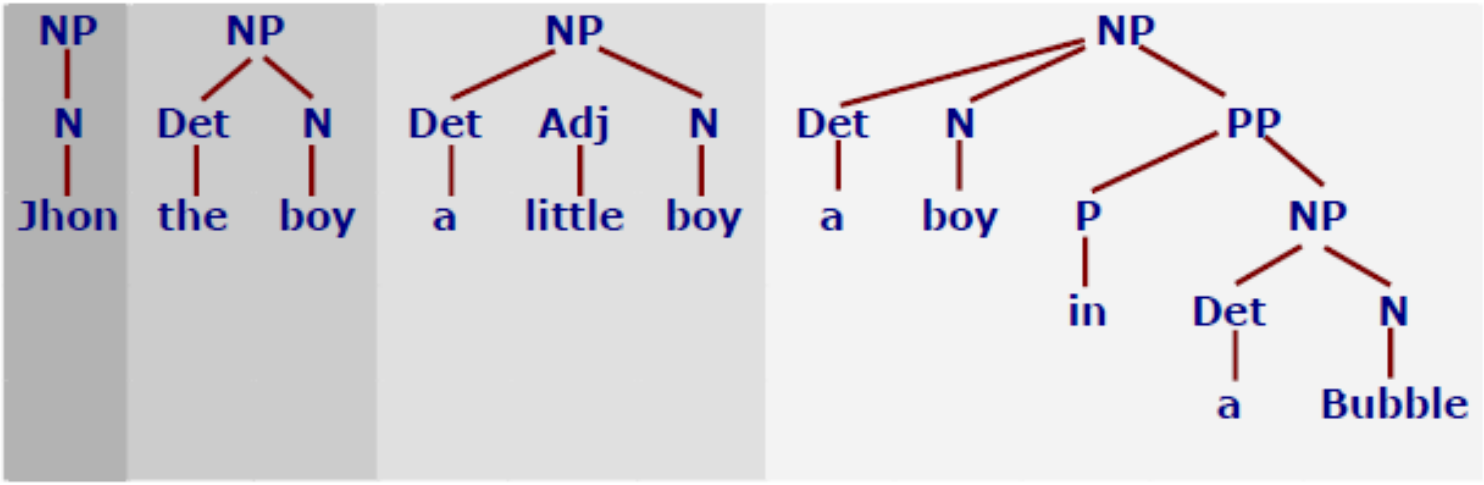
PS Rules and Tress for NP

Noun Phrase (NP)	
John	N
the boy	Det N
A little boy	Det Adj N
A boy in a bubble	Det N PP

Phrase Structure rules for NPs

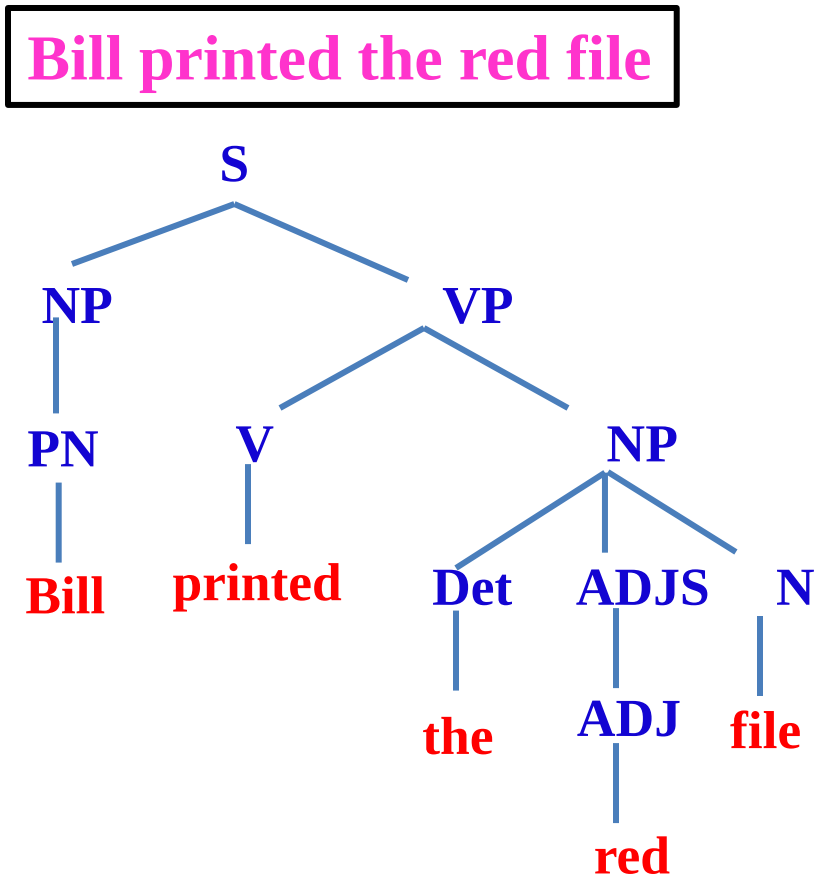
NP → (Det) (Adj) N (PP)

Phrase Structure trees for NPs



PS Rules

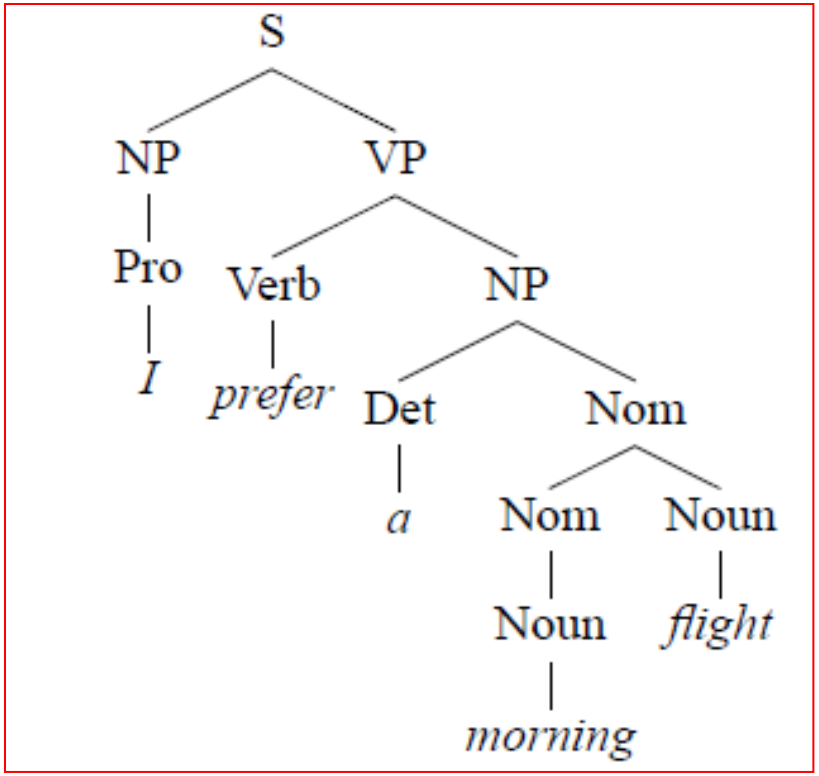
S→ NP VP
NP → (Det) (ADJS) N
NP → PRO
NP →PN
ADJS → ADJ |ADJS
VP → V (NP)
N → file|printer
PN → Bill|Rahim|Bird
PRO → I|he|she
Det → the|a|an
ADJ → short|long|want|red
V → printed|created|want



$$
\begin{aligned}
\textit{Noun} &\rightarrow \textit{flights} \mid \textit{breeze} \mid \textit{trip} \mid \textit{morning} \mid \ldots \\
\textit{Verb} &\rightarrow \textit{is} \mid \textit{prefer} \mid \textit{like} \mid \textit{need} \mid \textit{want} \mid \textit{fly} \\
\textit{Adjective} &\rightarrow \textit{cheapest} \mid \textit{non-stop} \mid \textit{first} \mid \textit{latest} \\
&\quad \mid \textit{other} \mid \textit{direct} \mid \ldots \\
\textit{Pronoun} &\rightarrow \textit{me} \mid \textit{I} \mid \textit{you} \mid \textit{it} \mid \ldots \\
\textit{Proper-Noun} &\rightarrow \textit{Alaska} \mid \textit{Baltimore} \mid \textit{Los Angeles} \\
&\quad \mid \textit{Chicago} \mid \textit{United} \mid \textit{American} \mid \ldots \\
\textit{Determiner} &\rightarrow \textit{the} \mid \textit{a} \mid \textit{an} \mid \textit{this} \mid \textit{these} \mid \textit{that} \mid \ldots \\
\textit{Preposition} &\rightarrow \textit{from} \mid \textit{to} \mid \textit{on} \mid \textit{near} \mid \ldots \\
\textit{Conjunction} &\rightarrow \textit{and} \mid \textit{or} \mid \textit{but} \mid \ldots
\end{aligned}
$$

S	$\rightarrow$	$NP\ VP$	I + want a morning flight
NP	$\rightarrow$	$Pronoun$	I
	$\mid$	$Proper\text{-}Noun$	Los Angeles
	$\mid$	$Det\ Nominal$	a + flight
$Nominal$	$\rightarrow$	$Nominal\ Noun$	morning + flight
	$\mid$	$Noun$	flights
VP	$\rightarrow$	$Verb$	do
	$\mid$	$Verb\ NP$	want + a flight
	$\mid$	$Verb\ NP\ PP$	leave + Boston + in the morning
	$\mid$	$Verb\ PP$	leaving + on Thursday
PP	$\rightarrow$	$Preposition\ NP$	from + Los Angeles

Parse tree: I Prefer a Morning flight



Bracket Notation: LISP/Prolog

[$_S$ [$_{NP}$ [$_{Pro}$ I]] [$_{VP}$ [$_V$ prefer] [$_{NP}$ [$_{Det}$ a] [$_{Nom}$ [$_N$ morning] [$_{Nom}$ [$_N$ flight]]]]]]

Formal definition of context-free grammar

A CFG, *G is* defined by four parameters *N, S, P, S (4-tuple):*

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol**

Parser

- A parser is a program, that accepts as input a sequence of words in a natural language and breaks them up into parts (nouns, verbs, & their attributes), to be managed by other programming.
- Parsing can be defined as the act of analyzing the grammatically an utterance according to some specific grammar
- Parsing is the process to check, that a particular sequence of words in sentence correspond to a language defined by its grammar
- Parsing means show how we can get from the start symbol of the grammar to the sentence of words using the production rules
- The output of a parser is a ***parse tree/Structural Representation (SR)***

Properties of a Parse Tree

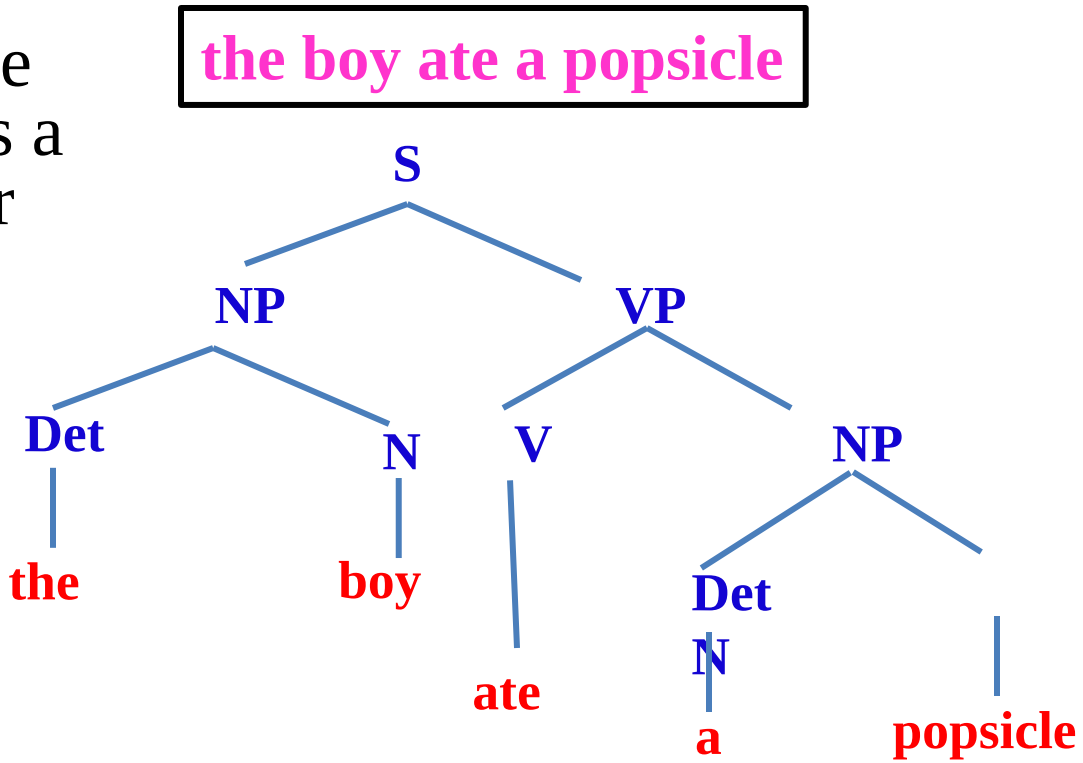
- Parse tree is a way of representing the output of a parser
- Each phrasal constituent found during parsing becomes a branch node of the parse tree
- Every node corresponds either to an input word or to a non-terminal
- Each level in the parse tree corresponds to the application of one PS rule
- The words of the sentence become the leaves of the parse tree
- There can be more than one parse tree for a single sentence

Parse tree/Structural Representation (SR)

- For computation, a tree must be represented as a record, a list or similar data structure.

List:

```
(S  (NP ((Det the)
         (N  boy))
    (VP  (V ate)
         (NP (Det  a)
             (N
popsicle)))))
```



Top-down vs. Bottom-up Parsing

- To parse a sentence, it is necessary to find a way in which the sentence could have been generated from the stat symbols
- Two Ways: Top-down and Bottom-up

- The choice between these two approaches is similar to the choice between forward and backward reasoning in other problem-solving tasks.
- Sometimes these two approaches are combined to a single method called "bottom-up parsing with top-down filtering (hybrid approach)".

Top-down vs. Bottom-up Parsing

- **Top-down Parsing:**

Begin with start symbol and apply the grammar rules forward until the symbols at the terminals of the tree correspond to the components of the sentence being parsed.

- **Bottom-up parsing**

Begin with the sentence to be parsed and apply the grammar rules backward until a single tree whose terminals are the words of the sentence and whose top node is the start symbol has been produced.

Top-down vs. bottom-up

- S→ NP VP
  → N VP
  → Kathy VP
  →Kathy V NP
  →Kathy jumped NP
  →Kathy jumped Det N
  →Kathy jumped the N
  →Kathy jumped the horse

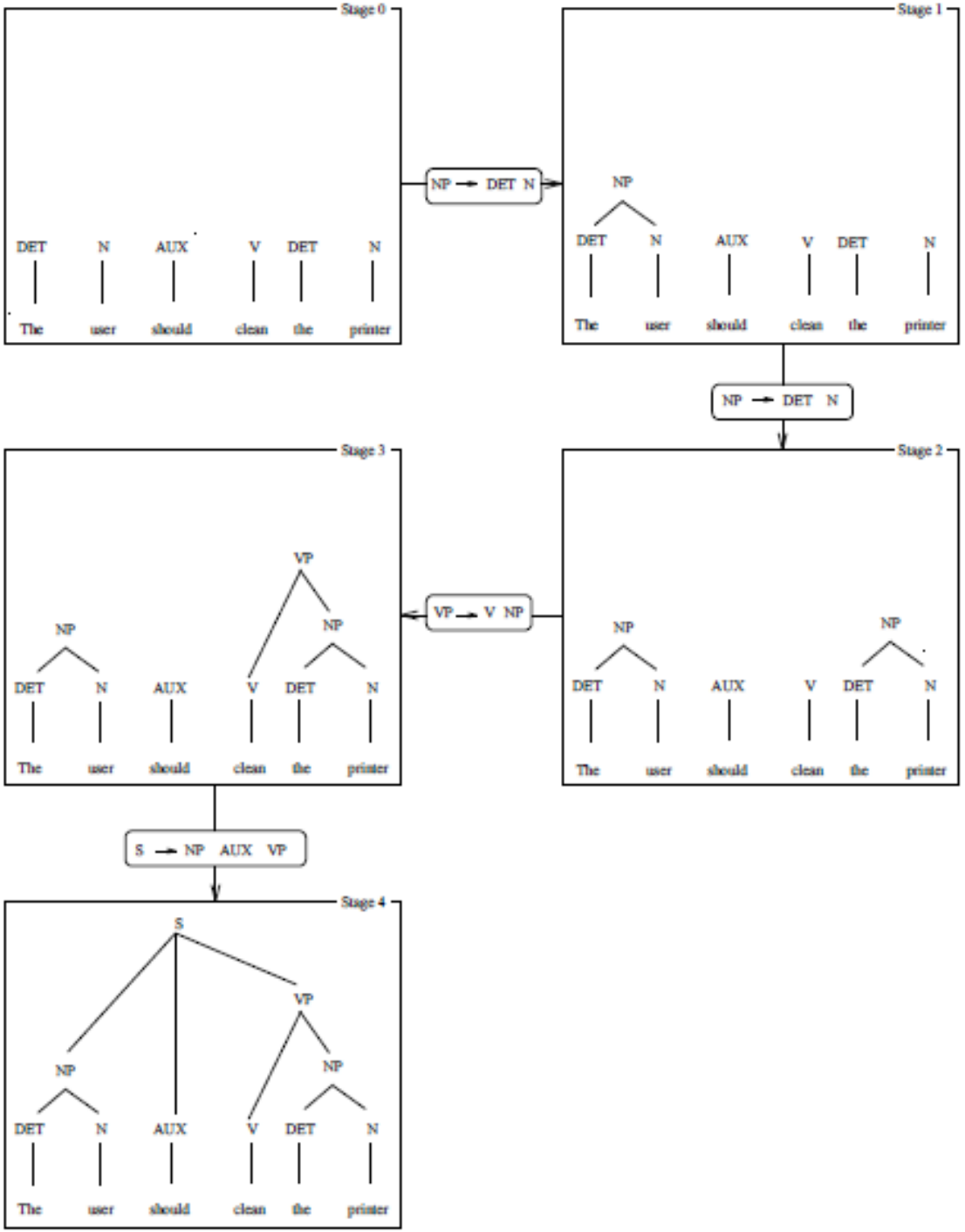
- → Kathy jumped the horse
  → N jumped the horse
  → N V the horse
  → N V Det horse
  → N V Det N
  →NP V Det N
  → NP V NP
  → NP VP
  →S

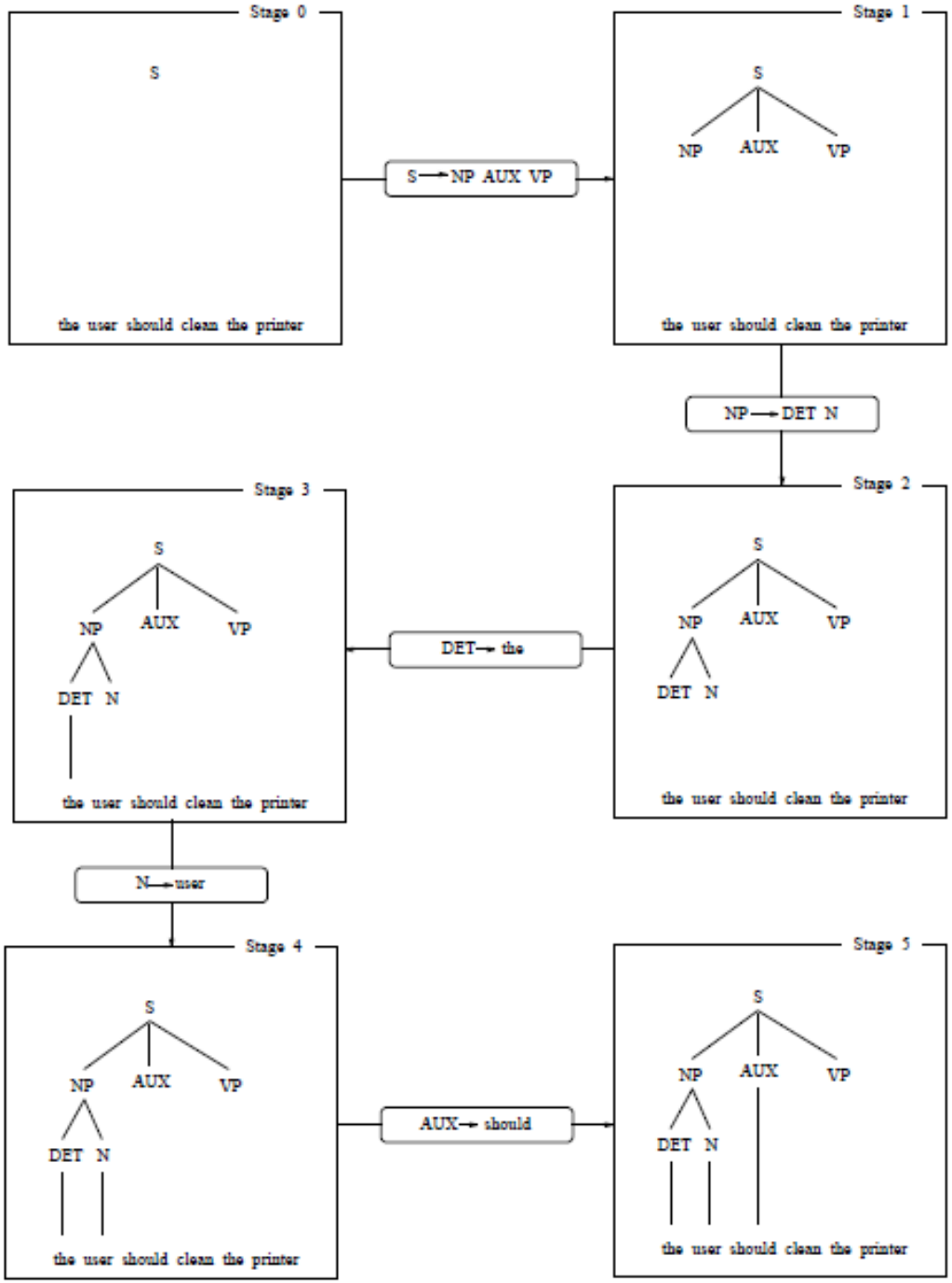
Bottom-Up Algorithm

1 For each word in the sentence, find a rule whose right hand side matches it. This means that every word would then be labelled with its part of speech (shown on the left hand side of the rule that matched it). This step is exactly equivalent to looking up the words in an English dictionary. Given rules of the type `N` $\rightarrow$ `user`, `N` $\rightarrow$ `printer`, and `V` $\rightarrow$ `clean`, this will produce a partial structure as we can see at the top left corner (Stage 0) of Figure 3.7.

2 Starting from the left hand end of the sentence, find every rule whose right-hand side will match one or more of the parts of speech (Stage 1 of Figure 3.7).

3 Keep on doing step 2, matching larger and larger bits of phrase structure until no more rules can be applied. (In our example, this will be when the sentence rule finally matches up with a noun phrase and a verb phrase which have already been identified). The sentence is now parsed (Stage 2-4 of Figure 3.7).


Stage 0
DET N AUX V DET N
The user should clean the printer
NP → DET N
Stage 1
NP
DET N AUX V DET N
The user should clean the printer
NP → DET N
Stage 2
NP NP
DET N AUX V DET N
The user should clean the printer
VP → V NP
Stage 3
NP VP
NP
DET N AUX V DET N
The user should clean the printer
S → NP AUX VP
Stage 4
S
NP VP
NP
DET N AUX V DET N
The user should clean the printer


Stage 0
S
the user should clean the printer
S → NP AUX VP
Stage 1
S
NP AUX VP
the user should clean the printer
NP → DET N
Stage 2
S
NP AUX VP
DET N
the user should clean the printer
DET → the
Stage 3
S
NP AUX VP
DET N
the user should clean the printer
N → user
Stage 4
S
NP AUX VP
DET N
the user should clean the printer
AUX → should
Stage 5
S
NP AUX VP
DET N
the user should clean the printer

Finding one Interpretation or Finding Many

- *"plant" = industrial plant*
- *"plant" = living organism*
- The process of understanding a sentence is a search process in which a large universe of possible interpretations must be explored to find one that meets all the constraints imposed by a particular sentence
- We must decide whether to explore all possible paths or to explore a single path

Four possible ways

- All paths
- Best path with backtracking
- Best path with patch up
- Wait and see

- Read books [Rich & Knight] for more details

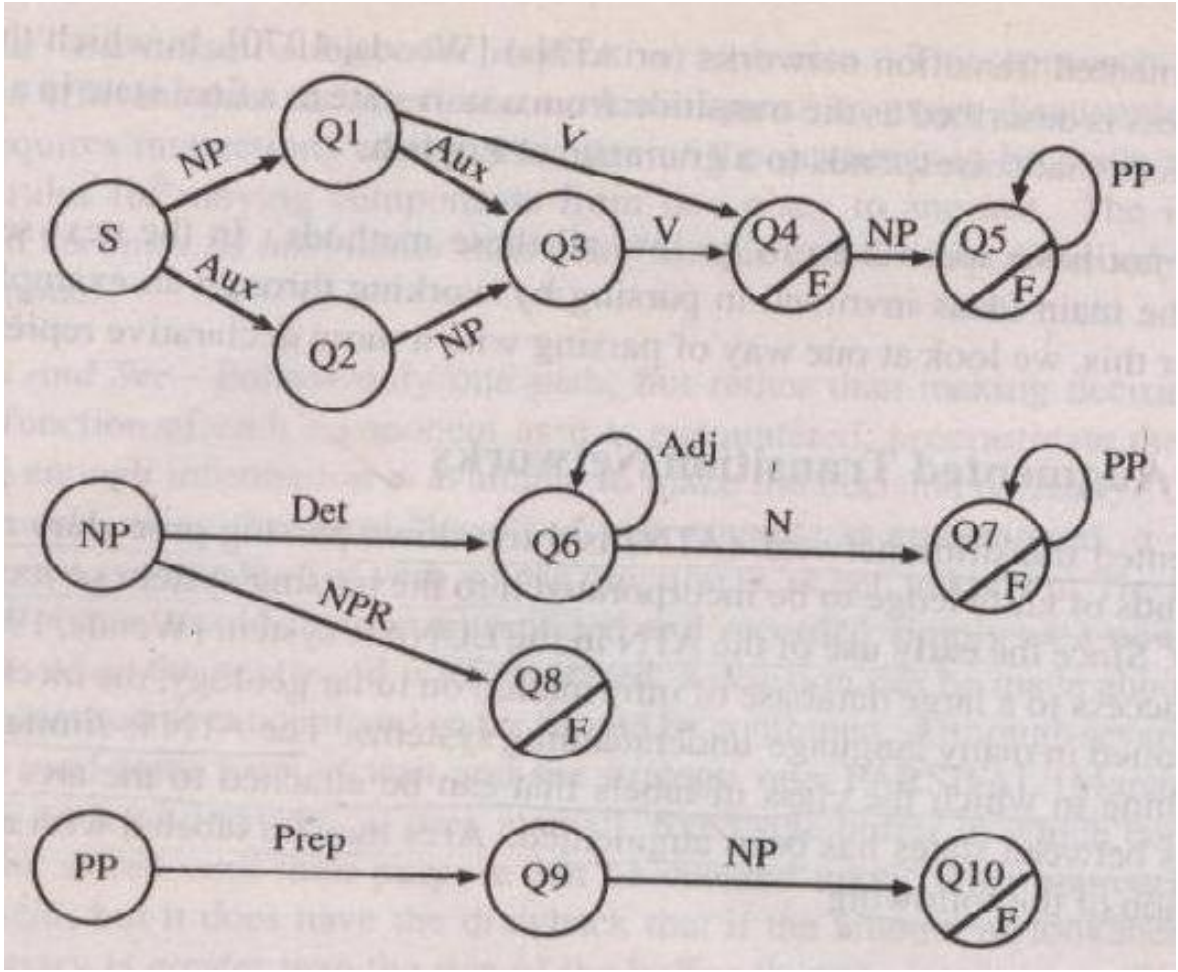
Augmented Transition Networks

- ATN: Augmented Transition Network
- Top-down parsing procedure that allows various kinds of knowledge to be incorporated into the parsing system so it can operate efficiently
- Similar to a finite state machine in which the class of labels that can be attached to the arcs that define transitions between states has been augmented.

Properties of ATN

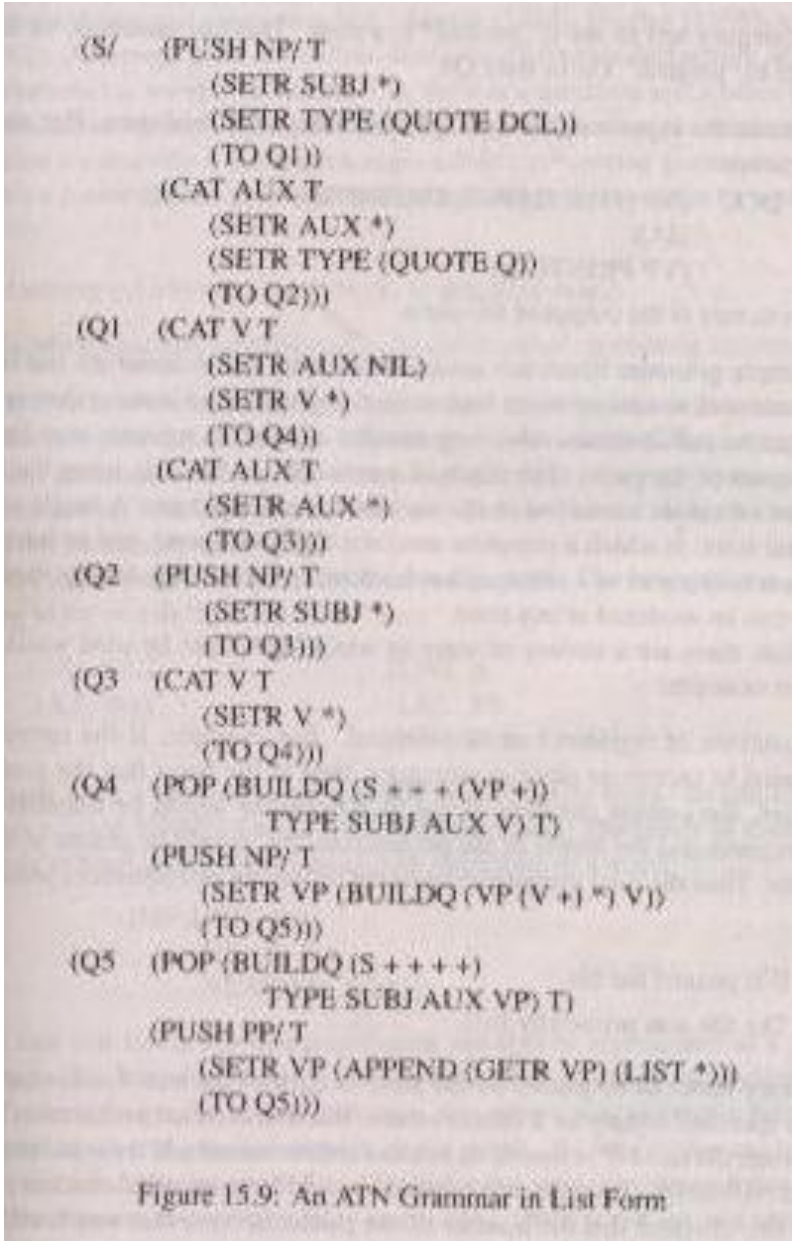
- Arcs may be labeled:
  - -Specific words: *in*
  - -Word categories (part-of-speech): *noun*
  - -Pushes to other networks that recognize significant components of a sentence. A network designed to recognize a prepositional phrase (PP) may include arc that asks for ("pushes for") a noun phrase (NP)
  - -Procedures that perform arbitrary tests on both the current input and on sentence components that have already been identified
  - -Procedures that build structures that will form part of the final parse

The long file has printed



An ATN Network for a Fragment of English

Augmented Transition Networks



```
(S/   (PUSH NP/ T
          (SETR SUBJ *)
          (SETR TYPE (QUOTE DCL))
          (TO Q1))
      (CAT AUX T
          (SETR AUX *)
          (SETR TYPE (QUOTE Q))
          (TO Q2)))
(Q1   (CAT V T
          (SETR AUX NIL)
          (SETR V *)
          (TO Q4))
      (CAT AUX T
          (SETR AUX *)
          (TO Q3)))
(Q2   (PUSH NP/ T
          (SETR SUBJ *)
          (TO Q3)))
(Q3   (CAT V T
          (SETR V *)
          (TO Q4)))
(Q4   (POP (BUILDQ (S + + + (VP +))
                TYPE SUBJ AUX V) T)
      (PUSH NP/ T
          (SETR VP (BUILDQ (VP (V +) *) V))
          (TO Q5)))
(Q5   (POP (BUILDQ (S + + + +)
                TYPE SUBJ AUX VP) T)
      (PUSH PP/ T
          (SETR VP (APPEND (GETR VP) (LIST *)))
          (TO Q5)))
```

Figure 15.9: An ATN Grammar in List Form

Semantic Analysis

- Producing a syntactic parse of a sentence is only the first step toward understanding it.
- We must still produce a representation of the meaning of the sentence.
- There is no single definitive language in which all sentence meaning can be described.
- Need to define vocabulary (features)
- The choice of a target language for any particular natural language understanding program must depend on what is to be done with the meanings once they are constructed.

Semantic Analysis

- Must do two important things

  -it must map individual words into appropriate objects in the knowledge base or database

  -it must create the correct structures to correspond to the way the meaning of the individual words combine with each other

Choice of Target Language in Semantic Analysis

- There are two broad families of target languages that are used in NL systems, depending on the role that the natural language system is playing in a larger system:
  - When natural language is being considered as a phenomenon on its own, as for example when one builds a program whose goal is to read text and then answer questions about it, a target language can be designed specifically to support language processing.
  - When natural language is being used as an interface language to another program (such as a db query system or an expert system), then the target language must be legal input to that other program. Thus the design of the target language is driven by the backend program.

Lexical processing

- The first step in any semantic processing system is to look up the individual words in a dictionary (or lexicon) and extract their meanings.
- Many words have several meanings, and it may not be possible to choose the correct one just by looking at the word itself.
- The process of determining the correct meaning of an individual word is called word sense disambiguation or lexical disambiguation.

Diamond

-a geometrical shape with 4 equal sides

-a baseball field

-an extremely hard & valuable gemstone

- It is done by associating, with each word in lexicon, information about the contexts in which each of the word's senses may appear.

Lexical Processing

- 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
- Sometimes only very straightforward info about each word sense is necessary.
- For example, baseball field interpretation of diamond could be marked as a LOCATION.
- Some useful semantic markers are :
  - PHYSICAL-OBJECT
  - ANIMATE-OBJECT
  - ABSTRACT-OBJECT

❑Using these markers, the correct meaning of a word in the sentence can be computed

Sentence-Level Processing

- Several approaches to the problem of creating a semantic representation of a sentence have been developed, including the following:
  - **Semantic grammars**, which combine syntactic, semantic and pragmatic knowledge into a single set of rules in the form of grammar.
  - **Case grammars,** in which the structure that is built by the parser contains some semantic information, although further interpretation may also be necessary.
  - **Conceptual parsing** in which syntactic and semantic knowledge are combined into a single interpretation system that is driven by the semantic knowledge.
  - **Approximately compositional semantic** interpretation, in which semantic processing is applied to the result of performing a syntactic parse

Semantic Grammar

- A semantic grammar is a CFG in which the choice of non-terminals and production rules is governed by semantic as well as syntactic function.
- A semantic action associated with each grammar rule.
- The result of parsing and applying all the associated semantic actions is the meaning of the sentence.

A semantic grammar

- **S-> what is FILE-PROPERTY of FILE?**
- **{ query FILE.FILE-PROPERTY}**
- **S-> I want to ACTION**
- **{ command ACTION}**
- **FILE-PROPERTY -> the FILE-PROP**
- **{FILE-PROP}**
- **FILE-PROP -> extension | protection | creation date | owner**
- **{value}**
- **FILE -> FILE-NAME | FILE1**
- **{value}**
- **FILE1 -> USER's FILE2**
- **{ FILE2.owner: USER }**
- **FILE1 -> FILE2**
- **{ FILE2}**
- **FILE2 -> EXT file**
- **{ instance: file-struct extension: EXT}**
- **EXT -> .init | .txt | .lsp | .for | .ps | .mss**
- **value**
- **ACTION -> print FILE**
- **{ instance: printing object : FILE }**
- **ACTION -> print FILE on PRINTER**
- **{ instance : printing object : FILE printer : PRINTER }**
- **USER -> Bill | susan**
- **{ value }**

Parsing with Semantic Grammars

- A semantic grammar can be used by a parsing system

- **Pros:**

  -when parse tree is complete, the result can be used immediately without the additional stage processing

  -Many ambiguities can be avoided

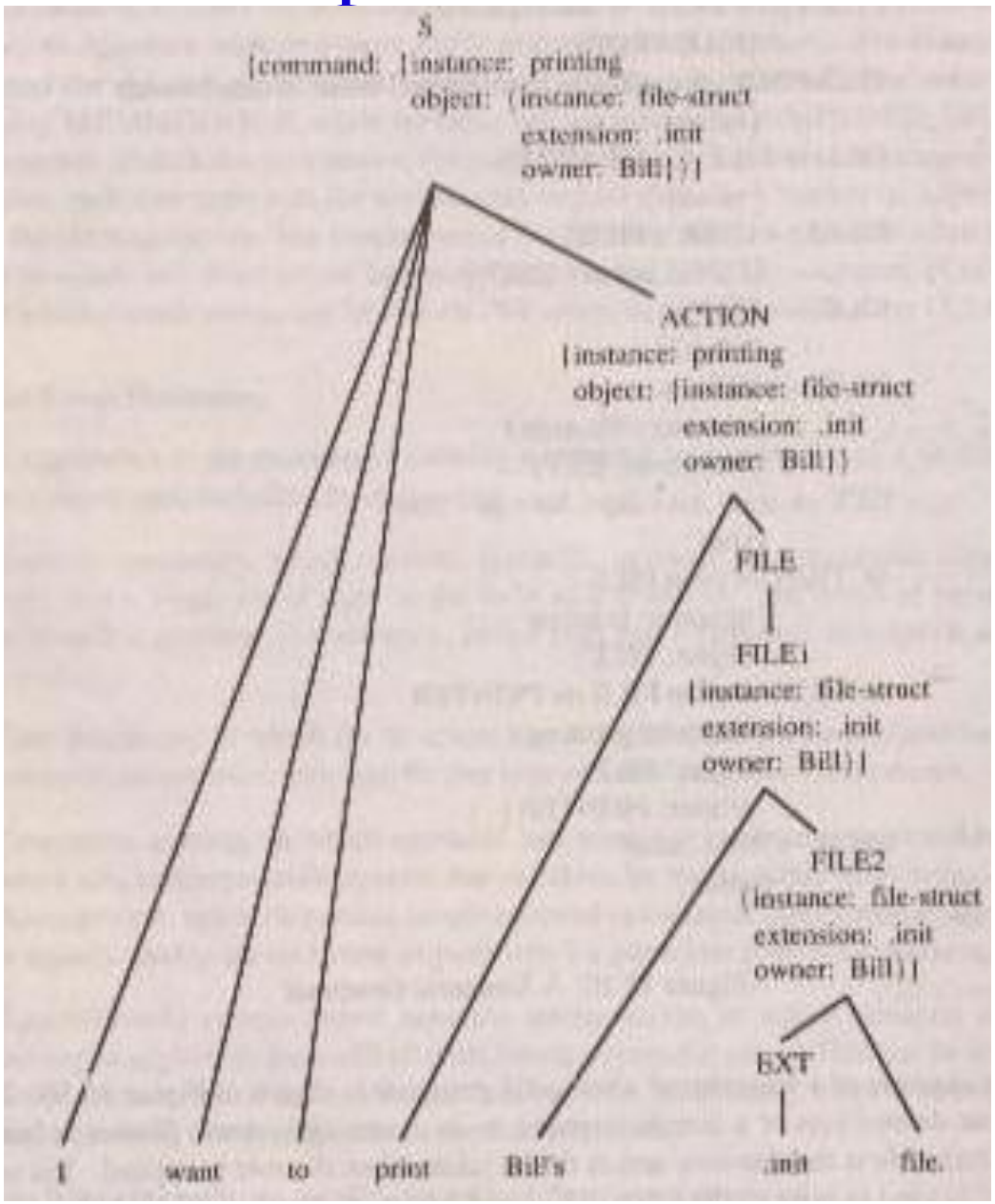
  -Syntactic issues that do not affect the semantic issues can be ignored

- **Cons**

  -the number of rules required can become very large

  -because the number of grammar rules may be large, the parsing process may be expensive

I want to print Bill's.int file



Case Grammars

- Provide a different approach to the problem of how syntactic & semantic interpretation can be combined
- Rules are written to describe syntactic rather than semantic regularities
- But the structures produce correspond to semantic relations rather than to strictly syntactic ones

An Example

- The semantic roles of "Susan" & "the file" are identical, but syntactic roles are reversed.

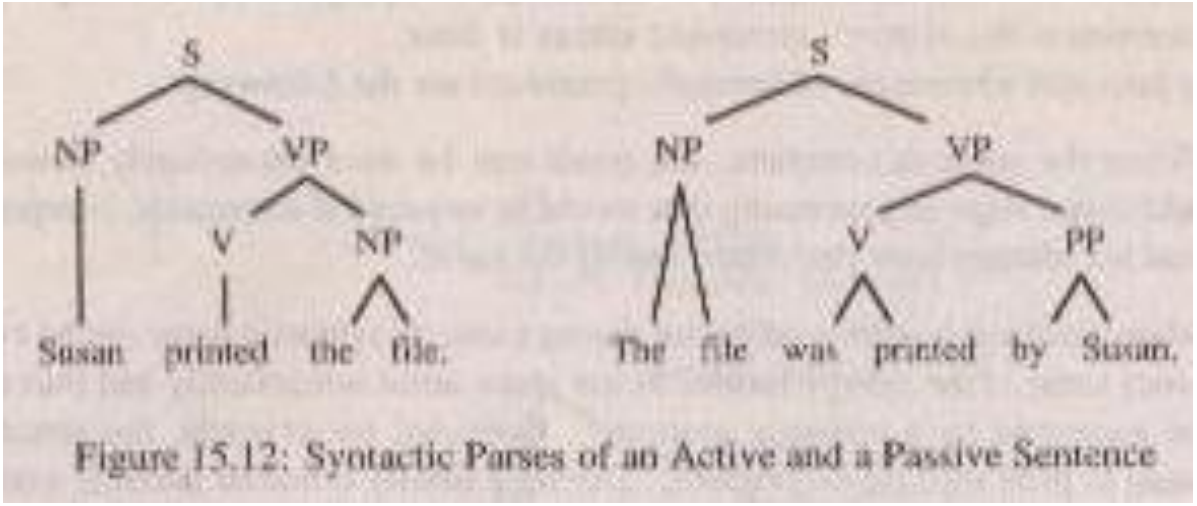


Figure 15.12: Syntactic Parses of an Active and a Passive Sentence

- **Case grammars**:

(printed (agent Susan)

(object File))

Example continued

- Syntactic structure are same
  - Case1: Mother is the subject of baked
  - Case2: the pie is the subject

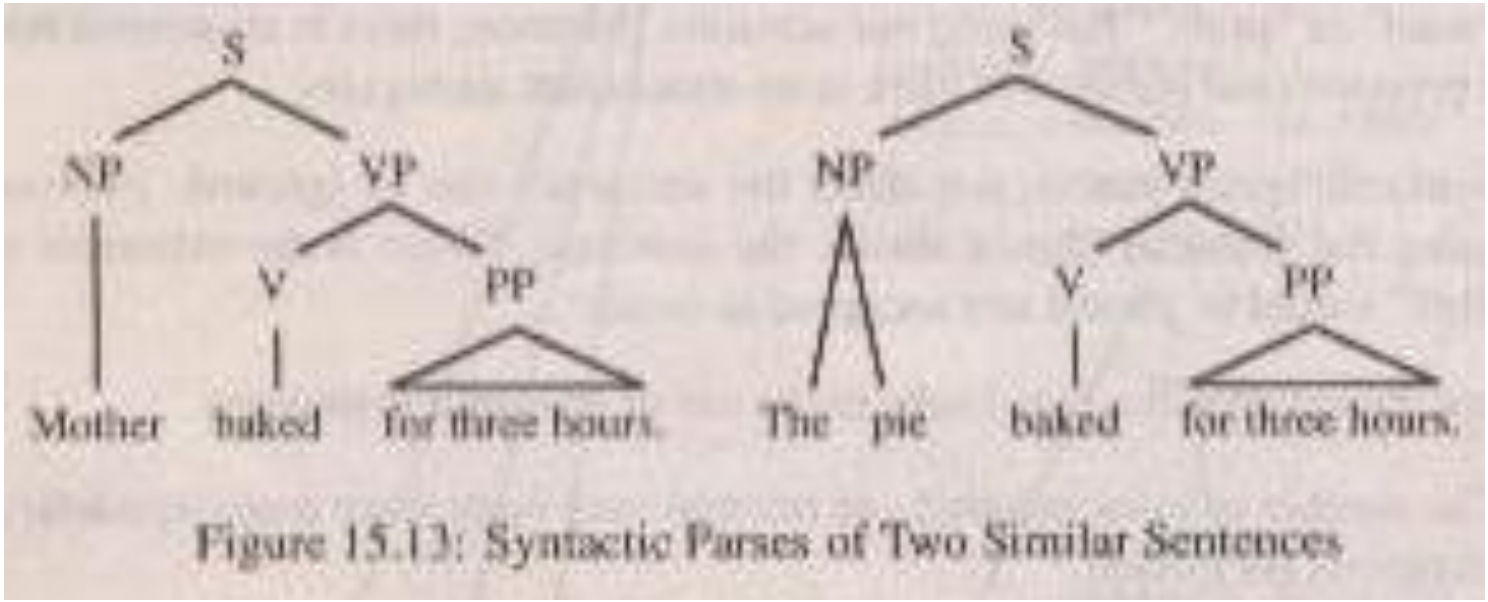


Figure 15.13: Syntactic Parses of Two Similar Sentences

- But the relationship between Mother and baking is very different from that between the pie and baking

Example Continued

- A case grammar analysis reflects this difference

(baked (agent Mother)
(timeperiod 3-hours))
(baked (object Pie)
(timeperiod 3-hours))

The semantic roles of “mother” and “the pie” are explicit

Deep cases

- **(A) Agent**-Instigator of the action (animate)
- **(I) Instrument-** Cause of the event/object used in causing the event (inanimate)
- **(D) Dative**-Entity affected by the action (animate)
- **(F) Factitive**-Object/being resulting from the event
- **(L) Locative**-Place of the event
- **(S) Source**-Place from which something moves
- **(G) Goal**-Place to which something moves
- **(B) Beneficiary**-being on whose behalf the event occurred (animate)
- **(T) Time**-time at which the event occurred
- **(O) Object**-Entity that is acted upon/that changes, the most general case

Parsing process directed by lexical entries associated with each verb

open [_ _ O (I) (A)]
The door opened
John opened the door
The wind opened the door
John opened the door with a chisel

Die [_ _ D]
John died

kill [_ _ D (I) (A)]
Bill killed John
Bill killed John with a knife

run [_ _ A]
John ran

want [_ _ A O]
John wanted some ice cream
John wanted Mary to go to the store

- Languages have rules for mapping from underlying case structure to surface syntactic forms
- If A is present, it is the subject. Otherwise, if I is present, it is the subject

Lexical and Compositional Semantics

- Input sentences are transformed through the use of domain dependent semantic rewrite rules which create the target knowledge structures.
  - The boy drunk a soda

(INGEST(ACTION(PP(MAME boy) (CLASS PHYS-OBJ)
(TYPE ANIMATE) (REF DEF)))
(OBJECT (PP (NAME soda) (CLASS PHYS-OBJ)
(TYPE INANIMATE) (REF INDEF)))
(TENSE PAST))

- The meaning of an expression is derived from the meanings of the parts of the expression. The target knowledge structures constructed are typically logic expressions
  - Sample24 contains silicon

(S DCL
(NP (N (Sample24)))
(AUX (TENSE (PREENT)))
(VP (V (contain))
(NP (N silicon))))

Try Yourself

- Design a set of PS rules to parse the Bangla simple sentences
- Generate the Parse tree for the following sentences by using your proposed PS rules:

- Rahim ekti boi porche
- CUET ekti engineering university
- Javed khub bhalo gan gay
- Ek kahana pakhi ahashe urche

Conclusion

- Why NLP is important?
- Steps involves in language processing
- Morphological analysis
- Syntactic analysis using PS rules
- Structural Representation (SR): Parse tree

Topics Covers:

Artificial Intelligence: Rich & Knight

Artificial Intelligent & Expert System: Patterson