Natural Language Processing

- NLP is the branch of computer science focused on developing systems that allow computers to communicate with people using everyday language.
- Also called Computational Linguistics
  - Also concerns how computational methods can aid the understanding of human language

Levels of Knowledge used in NLP

- Phonological
- Morphological
- Syntactic
- Semantic
- Pragmatic
- World

Phonological Knowledge

- This knowledge is needed if the computer is required to understand spoken language.
- Phonology is the study of the sounds that make up words and is used to identify words from sounds.
- Phoneme is the smallest unit of sound.
- Phones are aggregated into word sounds

Morphological Knowledge

- This is lexical knowledge which relates to word constructions from basic units called morphemes.
- A morpheme is the smallest unit of meaning.
- Morphological analysis is the task of segmenting a word into its morphemes:
  - carried ⇒ carry + ed (past tense)
  - independently ⇒ in + (depend + ent) + ly
  - Googlers ⇒ (Google + er) + s (plural)
  - unlockable ⇒ un + (lock + able) ?
    ⇒ (un + lock) + able ?

Syntactic Knowledge

- This knowledge relates to how words are put together or structured to form grammatically correct sentences in the language
- Parsing involves mapping a linear piece of text onto a hierarchy that represents the way the various words interact with each other.

Semantic Knowledge

- This knowledge is concerned with the meanings of words and phrases and how they combine to form sentence meanings

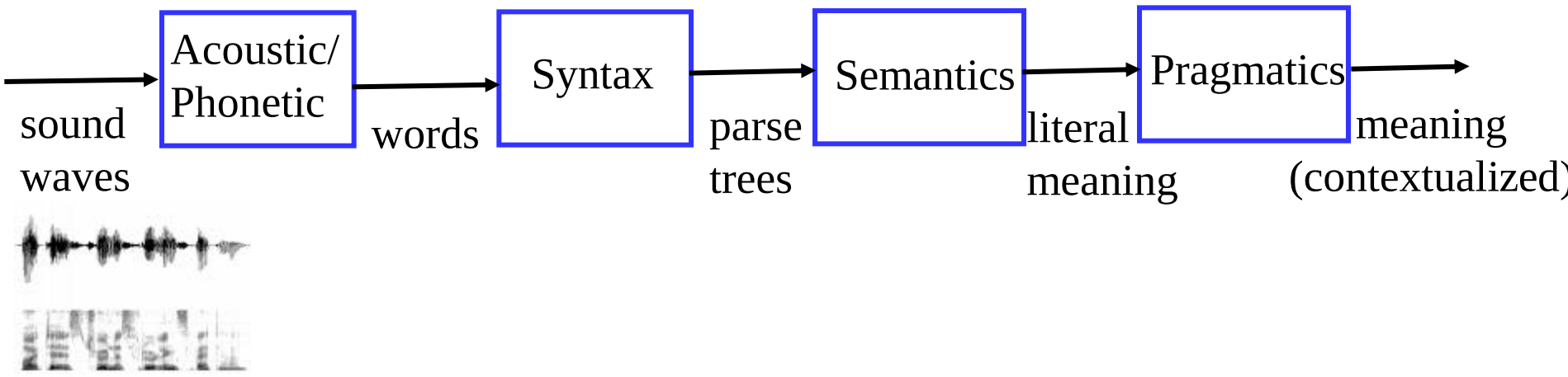
Pragmatic Knowledge

- This is high-level knowledge which relates to the use of sentences in different contexts and how the context affects the meaning of the sentences.
- This is the application of human-like understanding to sentences and discourse to determine meanings that are not immediately clear from the semantics.
- For example, if someone says, “Can you tell me the time?”, most people know that “yes” is not a suitable answer. Pragmatics enables a computer system to give a sensible answer to questions like this.

World Knowledge

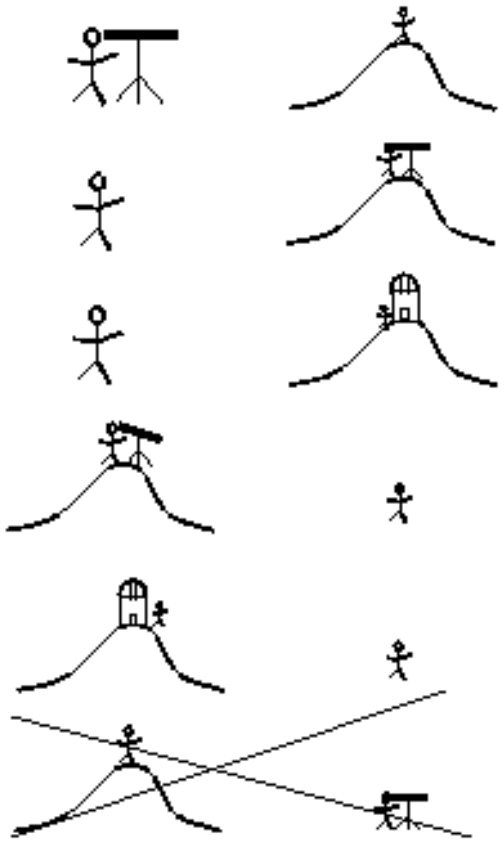
- World knowledge relates to the language a user must have in order to understand and carry on a conversation
- It must include an understanding of the other person's beliefs and goals.

Modular Comprehension



Ambiguity

- Natural language is highly ambiguous and must be *disambiguated*.
  - I saw the man on the hill with a telescope.
  - I saw the Grand Canyon flying to LA.
  - Time flies like an arrow.
  - Horse flies like a sugar cube.
  - Time runners like a coach.
  - Time cars like a Porsche.



Ambiguity is Ubiquitous

- Speech Recognition
  - "recognize speech" vs. "wreck a nice beach"
  - "youth in Asia" vs. "euthanasia"
- Syntactic Analysis
  - "I ate spaghetti with chopsticks" vs. "I ate spaghetti with meatballs."
- Semantic Analysis
  - "The dog is in the pen." vs. "The ink is in the pen."
  - "I put the plant in the window" vs. "Ford put the plant in Mexico"
- Pragmatic Analysis
  - **From "The Pink Panther Strikes Again":**
  - **Clouseau**: Does your dog bite?
    Hotel Clerk: No.
    Clouseau: [*bowing down to pet the dog*] Nice doggie.
    [*Dog barks and bites Clouseau in the hand*]
    Clouseau: I thought you said your dog did not bite!
    Hotel Clerk: That is not my dog.

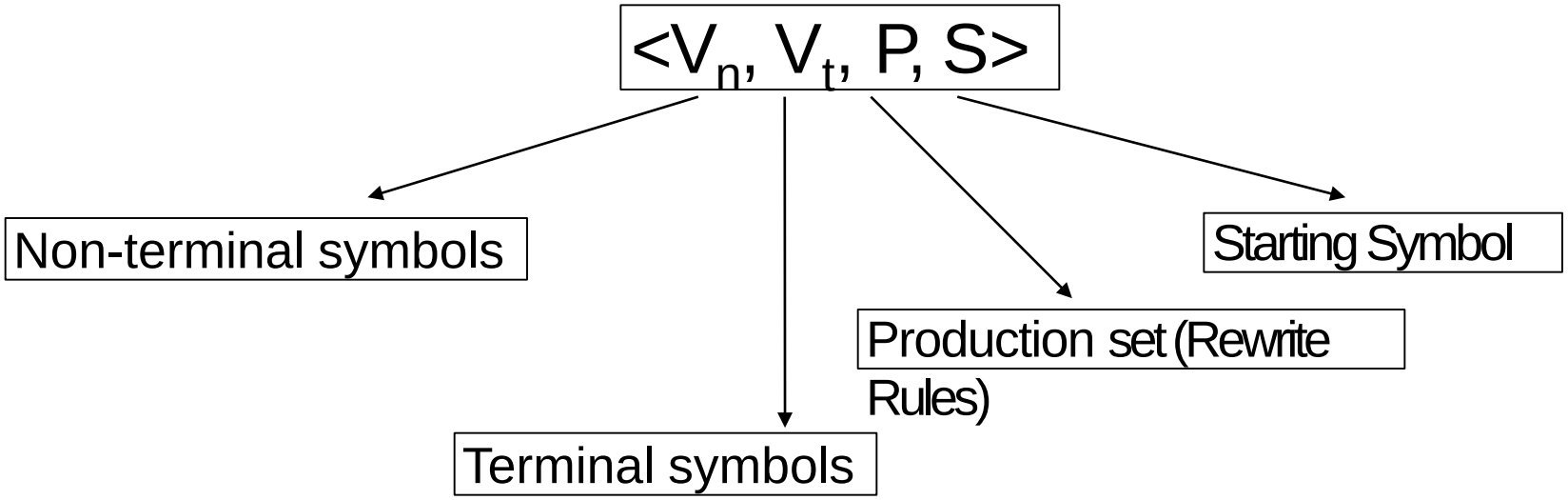
Natural Languages vs. Computer Languages

- Ambiguity is the primary difference between natural and computer languages.
- Formal programming languages are designed to be unambiguous, i.e. they can be defined by a grammar that produces a unique parse for each sentence in the language.
- Programming languages are also designed for efficient (deterministic) parsing, i.e. they are deterministic context-free languages (DCLFs).

Grammar and Languages

- Alphabet (vocabulary): Σ,set of symbols of the language
- String is constructed by Concatenating symbols
- Language L is a set of all strings that can be formed with symbols of Σ
- Well formed sentences are constructed using a set of rules called Grammar

Grammar G=(Vn,Vt,S,P)



Example of a Simple Grammar

- Production Rules:

S →NP VP

NP →ART N

VP →V NP

N →boy | popsicle| frog

V →ate | kissed | flew

ART →the | a

(S = starting symbol, NP = noun phrase, VP = verb phrase, N = noun, V = verb & ART = Article)

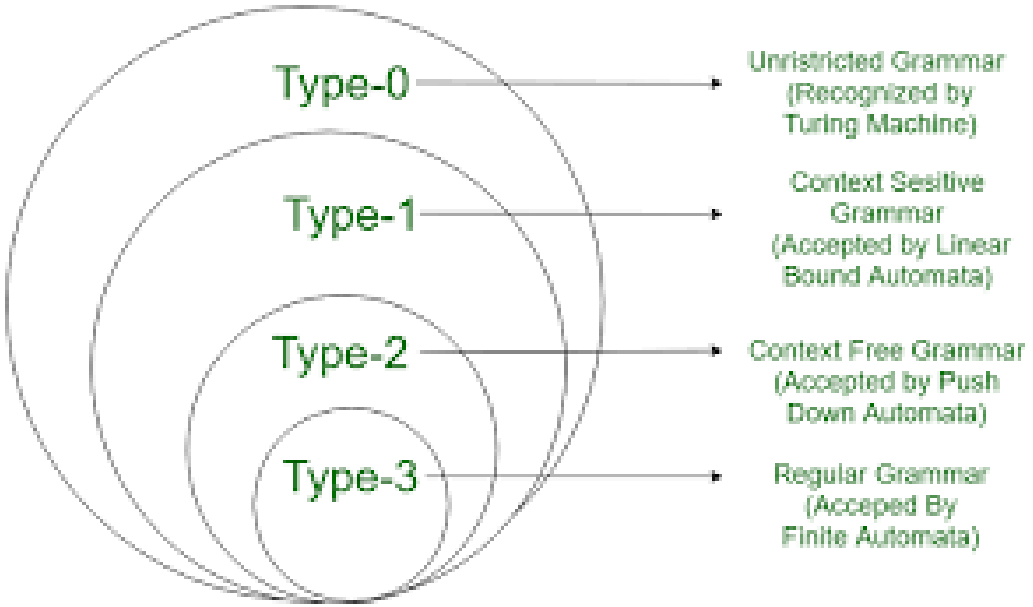
Sentence Generated : The boy ate a popsicle, The frog kissed a boy, A boy ate the frog.

Grammars and Languages

- **Grammar** is considered as a **generating formal system**.
  - The well-formed expressions or sentences of the language, are obtained (generated)from a start symbol by applying a set of productions or grammar rules.
- Conversely, **automata** are **accepting** (or recognizing) **formal systems**.
  - Their objective is to verify if a sentence belongs to a given language.

The Chomsky Hierarchy of Grammars

- The Chomsky hierarchy of languages reflects a certain order of complexity in some sense, the lower the language class is in the hierarchy; the simplest are its possible constructions.



The Chomsky Hierarchy of Grammars

Type 0 G :- xyz→ xwz , where y cannot be empty string.

Type 1 G :- S→ aS; AB →BA; |L.H.S|<=|R.H.S.|

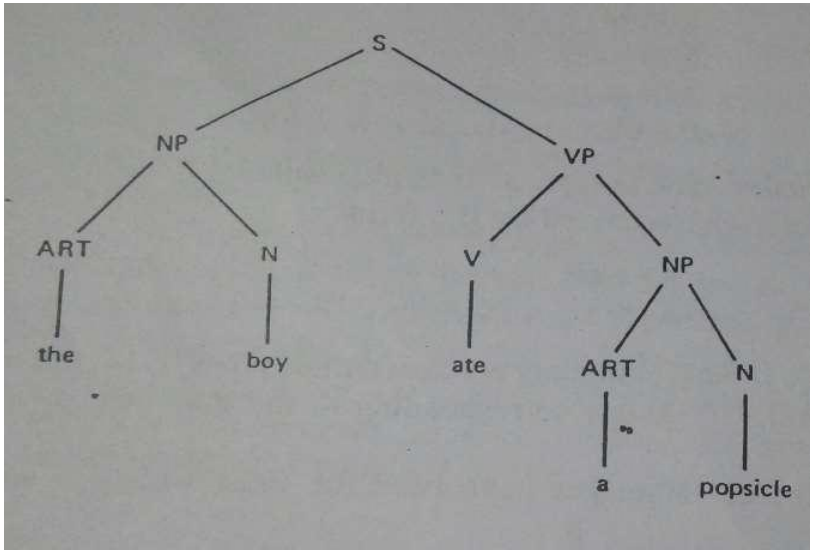
Type 2 G :- S→ aSb; A →BC; here |L.H.S.|=1

Type 3 G :- A→ aB; A→ a; Most restrictive grammar

Structural Representation

(Generative Grammar)

- It is convenient to represent sentences as tree or graph
- Example: Tree representation of sentence "The boy ate a popsicle" is shown below:
  - Root of the tree corresponds to whole sentence
  - Left subtree is a noun phrase
  - Right subtree is a verb phase
  - Leaf nodes contain terminal symbols
- Can also be represented as a list (S(NP ((ART the)
  (N boy))
  (VP (V ate)
  (NP (ART a)
  (N popsicle)))))



Drawback of Generative Grammars

Using generative grammar, we obtain different structures for sentences having different syntactical form , even if they have same content

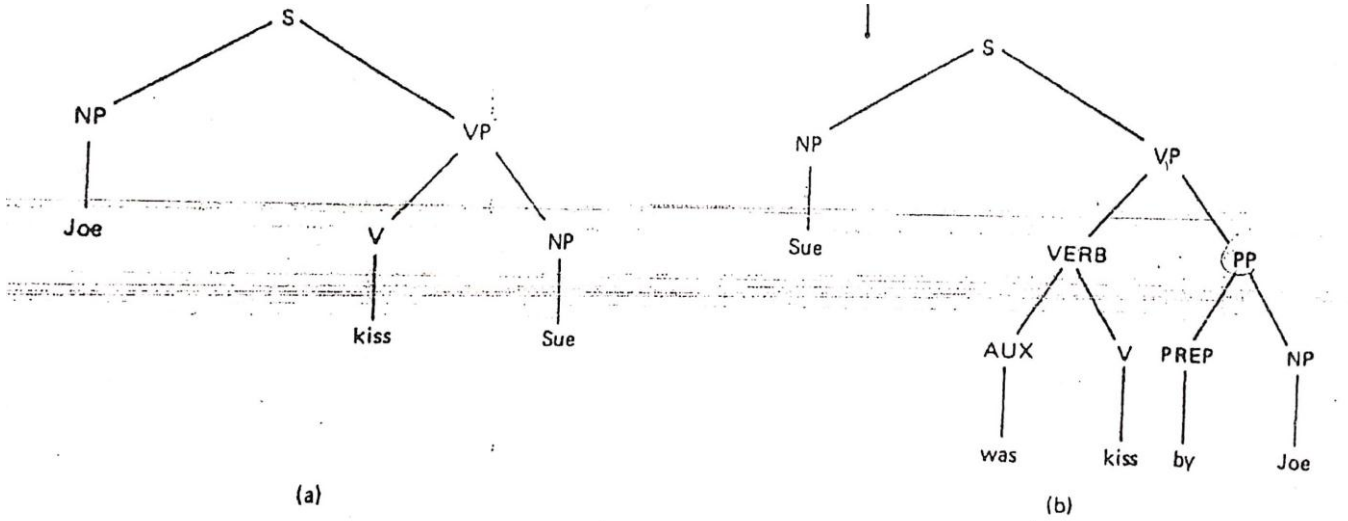
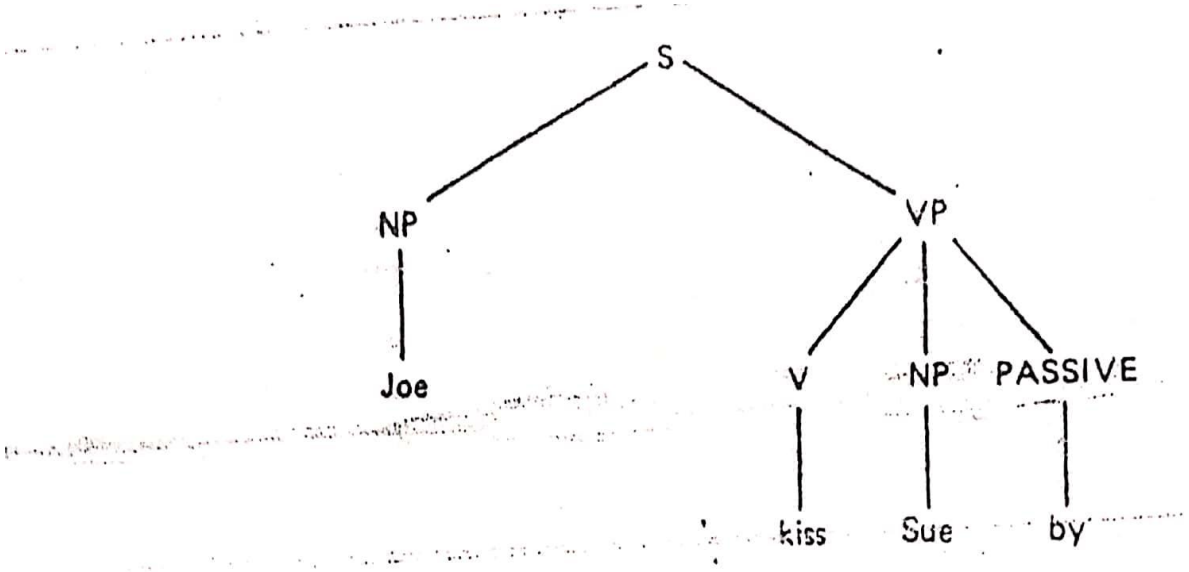


Figure 12.2 Structures for (a) active and (b) passive voice.

Transformational Grammars

- Two additional components are added to generative grammar(a semantic component and a phonological component)
- Added components produce single representation for sentences having same meaning
- This extended grammar is called Transformational Generative Grammar



Case Grammars

- Case is a semantic role that noun plays w.r.t. verbs and adjectives
- In English, cases are –
  - Nominative(subject of the verb)
  - Possessive(Showing possession/ownership)
  - Objectitve (direct and indirect objects)
- Sentence is composed of P(proposition) , a tenseless set of relationships between verbs and noun phrases and a Modality constituent M (mood, tense, aspect, negation etc.)
- S→M+P, where P is one or more cases C1,C2,….Ck
  - Agentive Case- Instigator of an action
  - Instrumental Case- Object used in the action
  - Objective Case –Object receiving the action

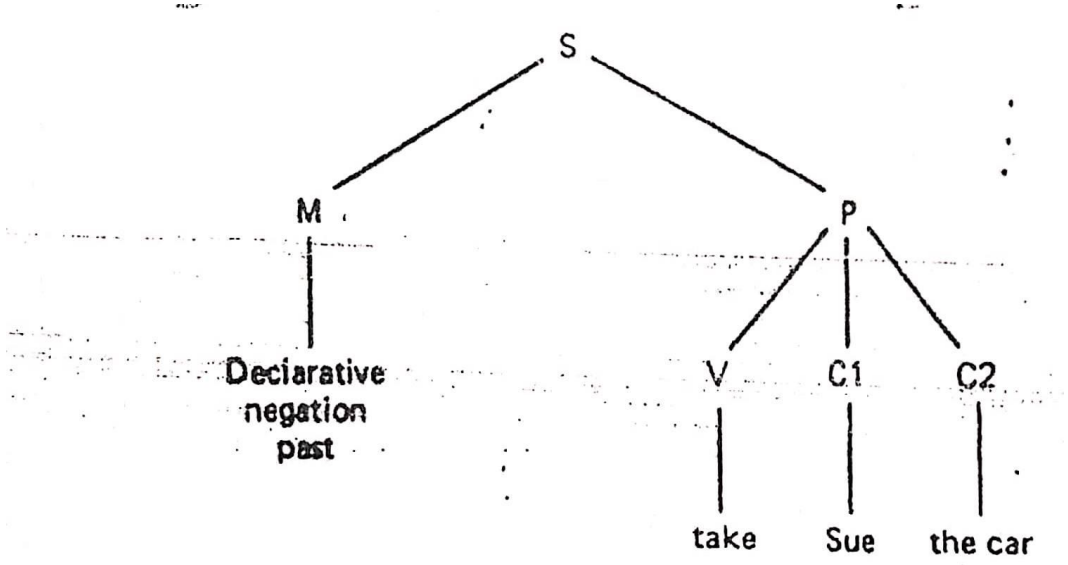
Case Grammars

Example- In this sentence “The soldier struck the suspect with the rifle butt”

Soldier –Agentive Case

Suspect – Objective Case

Rifle Butt – Instrumental Case



Case Grammar Tree for “ Sue did not take the car”

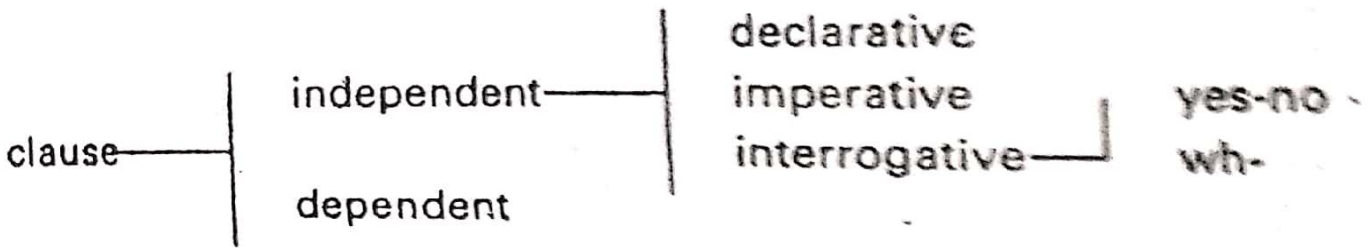
Systemic Grammar

- Emphasis on function and purpose in analysis of language
- Three functions
  i. Ideational Function- relates to content
  ii. Interpersonal function- concerned with purpose and mood
  iii. Textual Function- continuity and coherence between current and previous expressions

Systemic Grammar

- Proposed Model of Language contains 4 categories:
  - Language Units(Sentence, Clause, Word etc.)
  - Role Structure of Units(Subject, Predicate, Complement etc.)
  - Classification of Units(Verbal serves as predicate etc.)
  - System Constraints(constrains in combining component features)

- One Example of system constraints is depicted in below network structure:



Semantic Grammar

- Semantic grammars encode semantic information with syntactic grammar
- Rewrite rules are of following form:

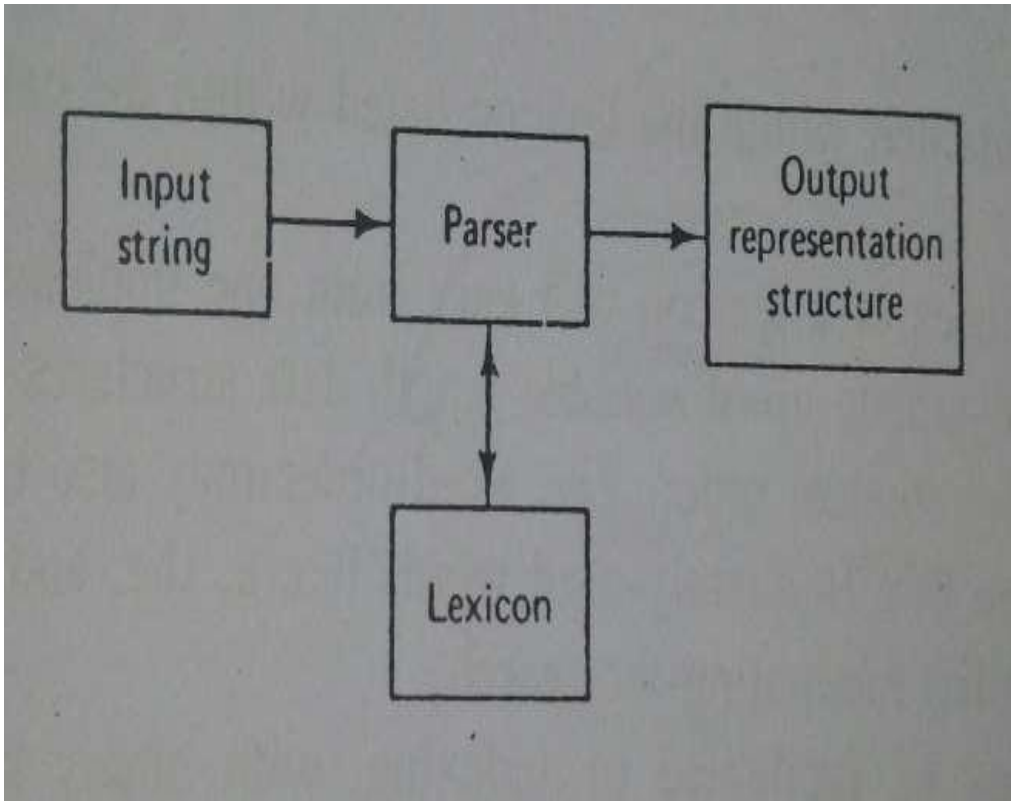
S → What is <OUTPUT-PROPERTY> of <CIRCUIT-PART>?
OUTPUT-PROPERTY → the <OUTPUT-PROP>
OUTPUT-PROPERTY → < OUTPUT-PROP>
CIRCUIT-PART → C23
CIRCUIT-PART → D12
OUTPUT-PROP → voltage
OUTPUT-PROP → current

- These rules handle different type of wh queries like –What is the name of carrier nearest to New York

Basic Parsing techniques

- Parsing is the method of determining syntactical structure of a sentence
- Process of analyzing a sentence by taking it apart word by word and determining its structure from its constituent parts and subparts
- Inverse of sentence generation process
- Lexicon is a dictionary of words(root words together with their derivatives)
- Information in Lexicon is needed to help determine the function and meaning of words

Basic Parsing techniques



Parsing an input to create output structure

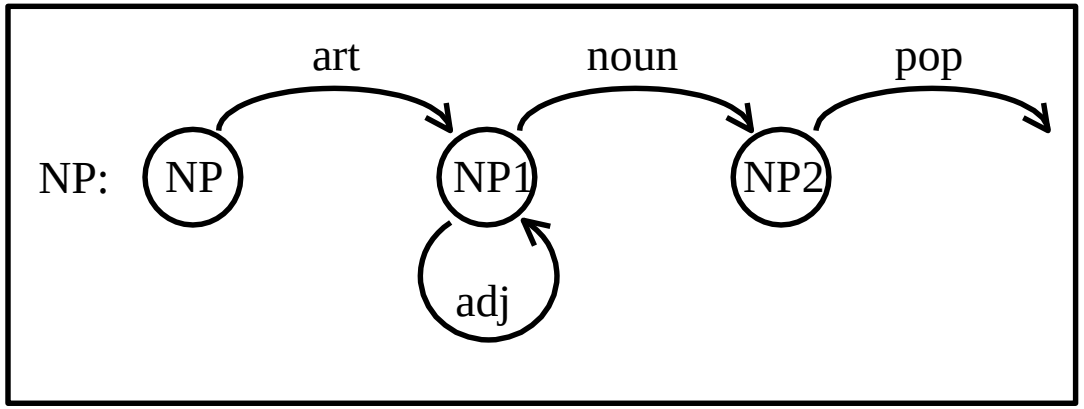
Word	Type	Features
a	Determiner	{3s}
be	Verb	Trans: intransitive
boy	Noun	{3s}
can	Noun	{1s, 2s, 3s, 1p, 2p, 3p}
	Verb	Trans: intransitive
carried	Verb	Form: past, past participle
.	.	.
.	.	.
.	.	.
orange	Adjective	
	Noun	{3s}
the	Determiner	{3s, 3p}
to	Preposition	
we	Pronoun	{1p}
		Case: subjective
yellow	Adjective	

Lexicon entries

Transition Networks

- Transition Network used to represent formal and natural language structures.
- They are formed using directed graphs and finite state automata.
- It consist of No. of nodes and labeled arcs.
- The nodes represent different states in traversing a sentence & the arcs represent rules or test conditions required to make the transition from one state to the next.
- A path through a T.N. corresponds to a permissible sequence of word types for a given grammar

Transition Network Grammar (con't)



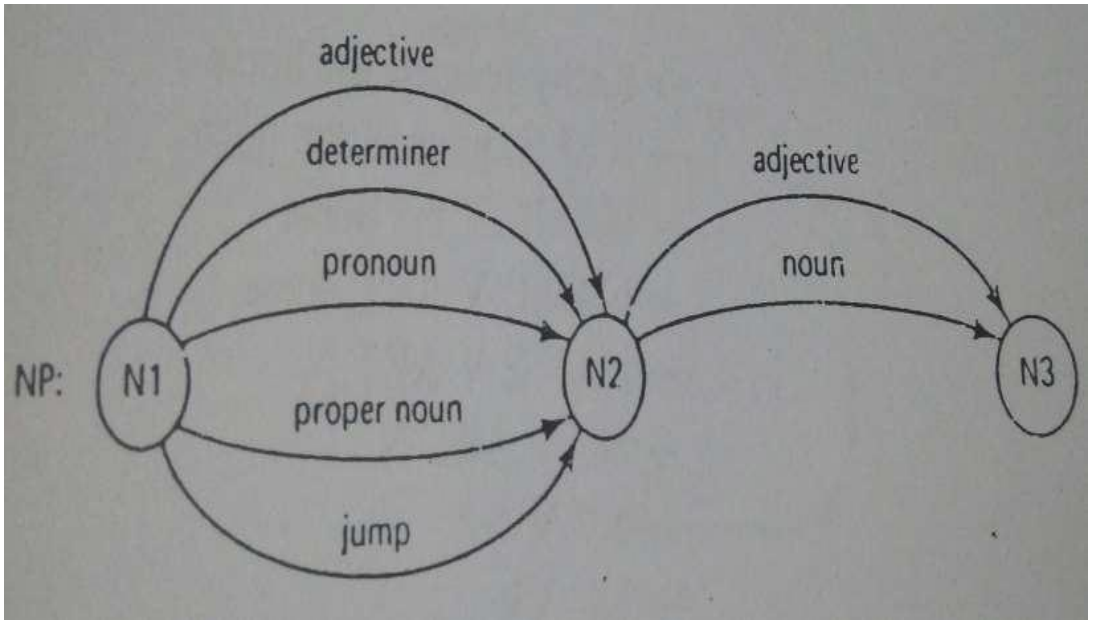
≡

$NP \rightarrow ART\ NP1$
$NP1 \rightarrow ADJ\ NP1$
$NP1 \rightarrow N$

transition network 1.

- The transition network 1 can be used to parse the NP "*a purple cow*".

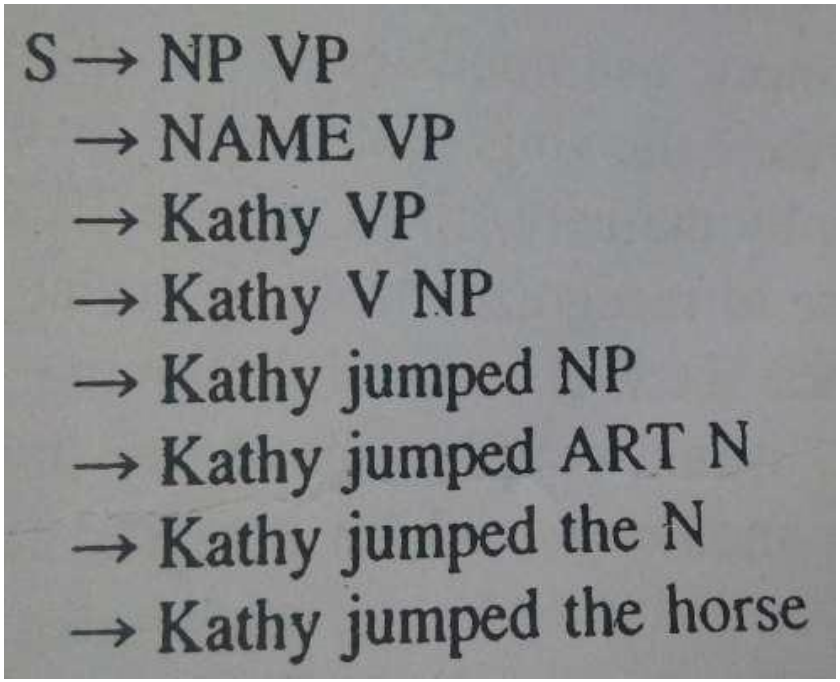
Transition Networks Example



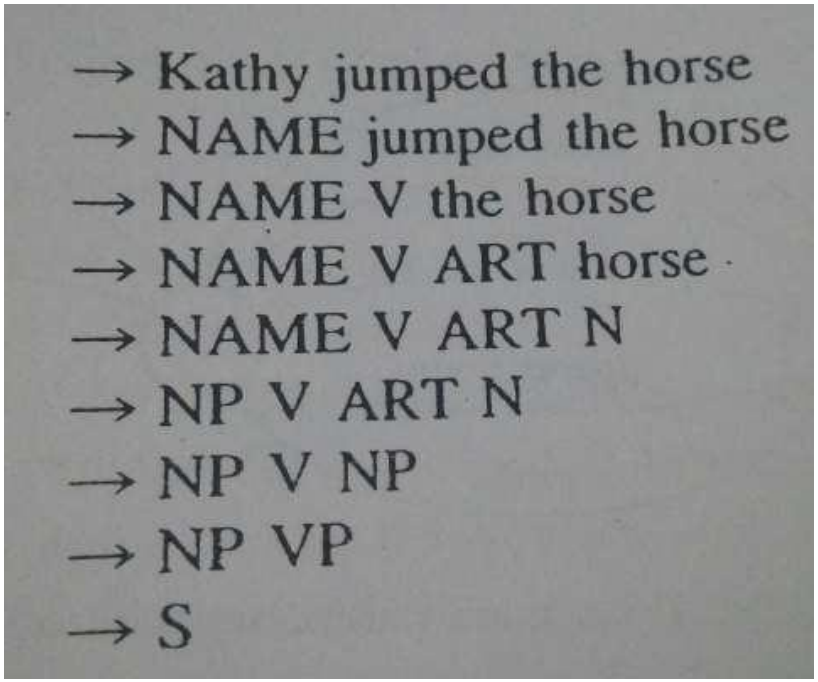
Sentences:-

a.Big white fluffy clouds.
b.Our bright children.
c.A large beautiful white flower.
d.Large green leaves.
e.Buildings.
f.Boston's best seafood restaurants.

Top Down Versus Bottom Up Parsing



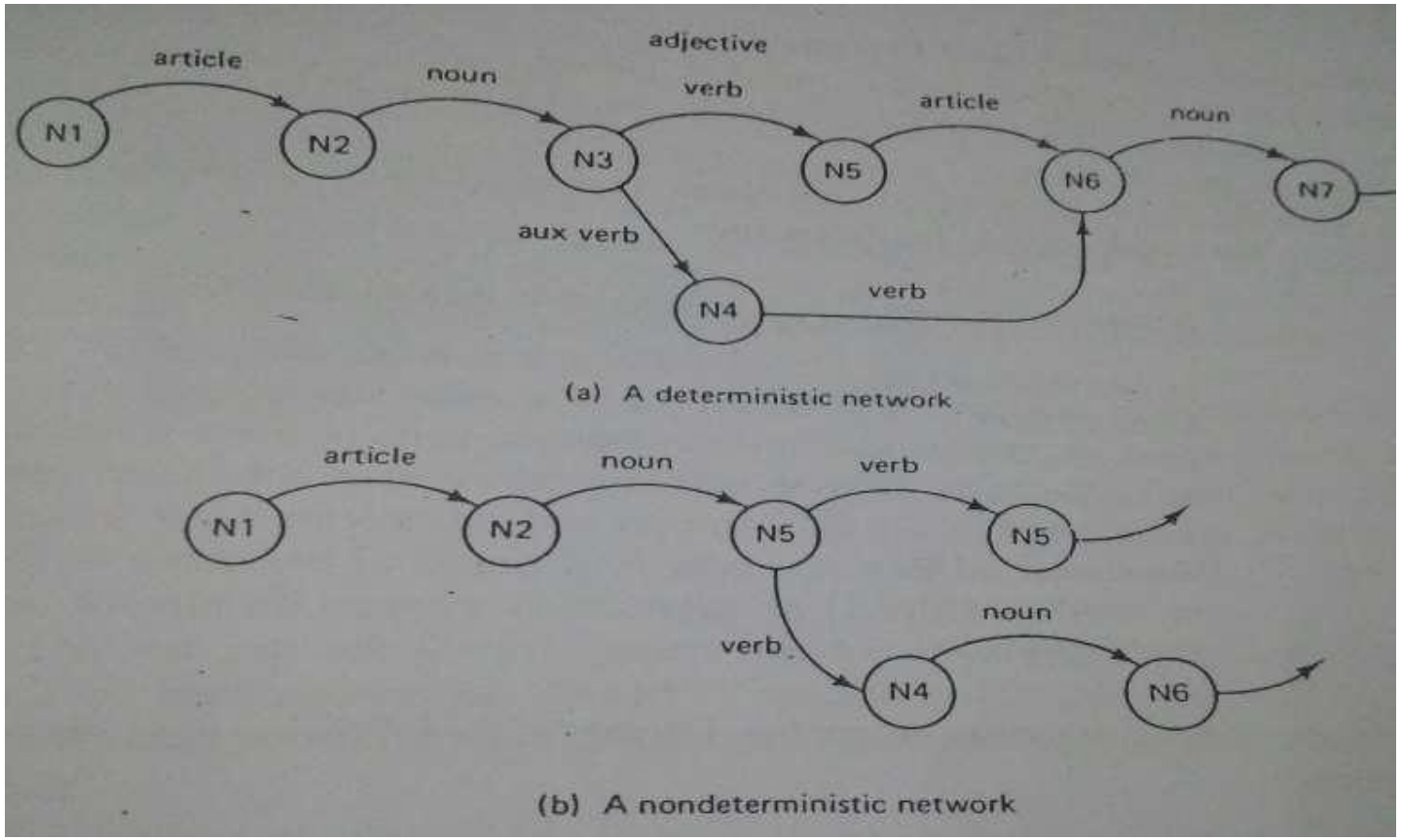
Top-Down Parsing



Bottom-Up Parsing

Deterministic Versus Non Deterministic Parser

- Deterministic parser permits only one choice for each word category. So each arc has a different test condition
- Non Deterministic parsers permit different arcs to be labelled with same test

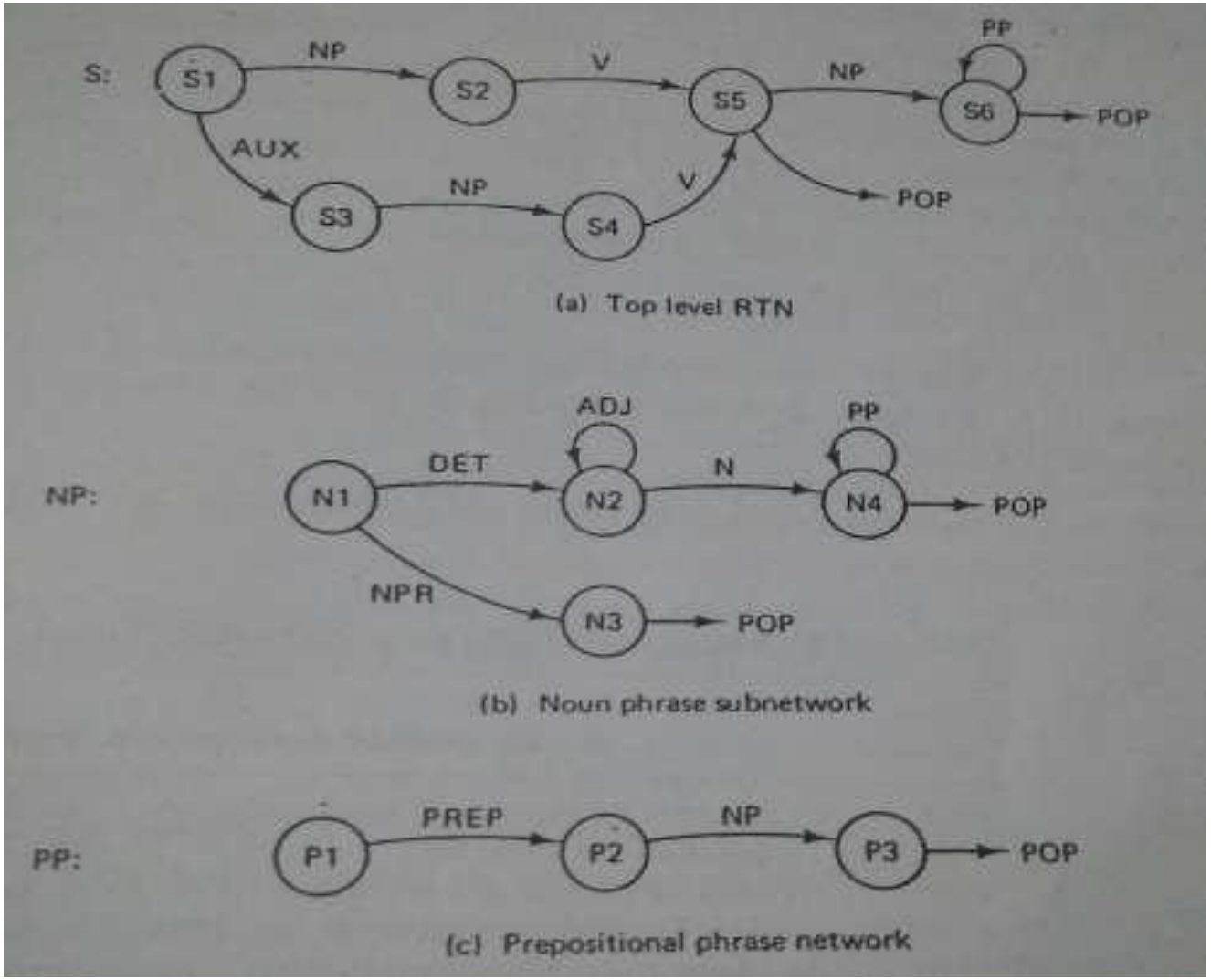


Recursive Transition Network

- RTN permits arc labels to refer to other networks.
- In RTN, one state is specified as a start state. A string is accepted by an RTN if a POP arc is reached and all the input has been consumed.

Type of arc	Purpose of arc	Example
CAT	a test label for the current word category	V
JUMP	requires no test to succeed	jump
POP	a label for the end of a network	pop
PUSH	a label for a call to a network	NP
TEST	a label for an arbitrary test	negatives
WORD	a label for a specific word type	from

RTN


S:
S1
NP
S2
V
S5
NP
PP
S6
POP
AUX
S3
NP
S4
V
POP
(a) Top level RTN
NP:
N1
DET
ADJ
N2
N
PP
N4
POP
NPR
N3
POP
(b) Noun phrase subnetwork
PP:
P1
PREP
P2
NP
P3
POP
(c) Prepositional phrase network

RTN Example

Let us consider a sentence "**The stone was dark black**".

The: ART

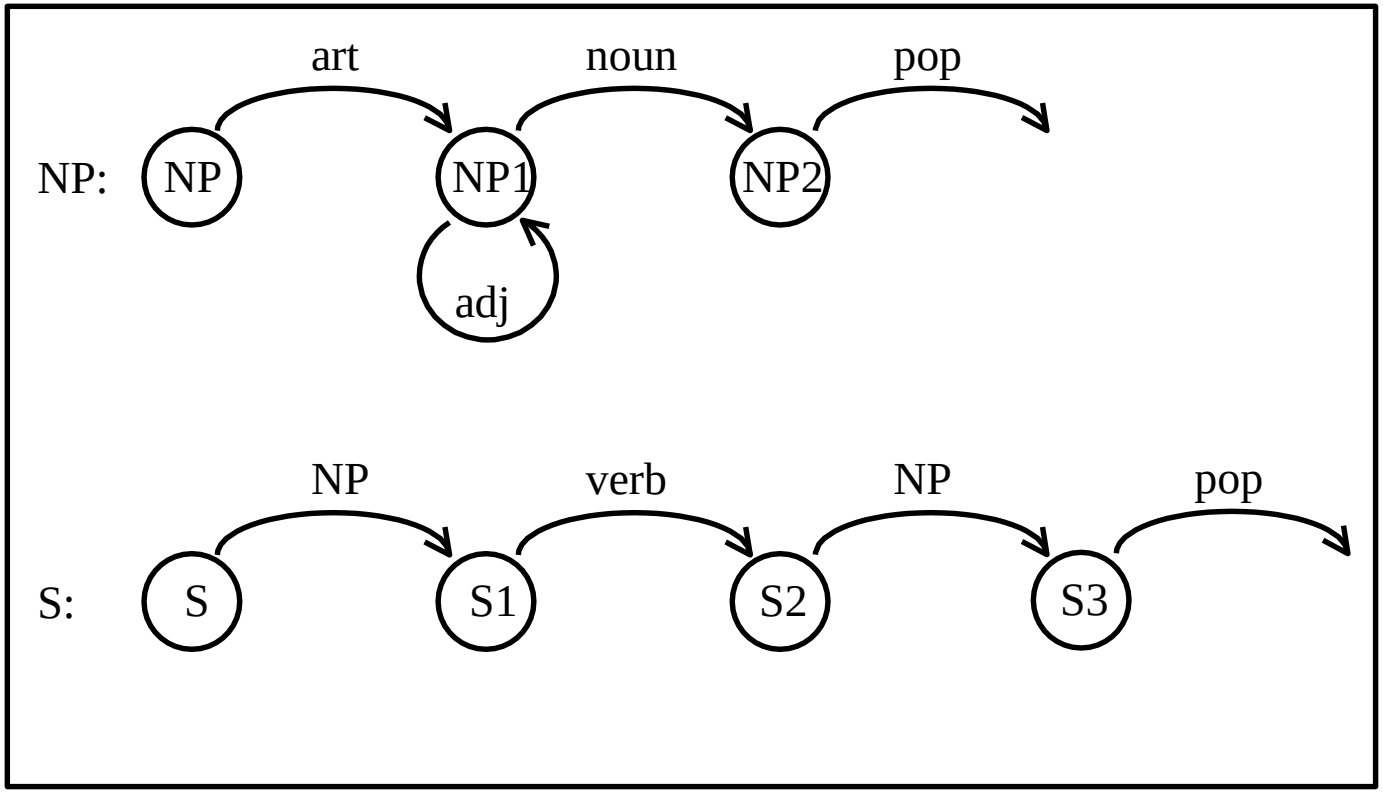
Stone: ADJ NOUN

Was: VERB

Dark: ADJ

Black: ADJ NOUN

An Example of RTN



transition network 2 (RTN)

Top-Down Parsing with RTN

- The algorithm for parsing with RTNs represents a parse state as follows:

-- current node : the node at which you are located in the network

-- current position : a pointer to the next word to be parse

-- return points : a stack of nodes in other networks where you will continue if you *pop* from the current network

Top-Down Parsing with RTN

- At each node, you can leave the current node and traverse an arc in the following cases:

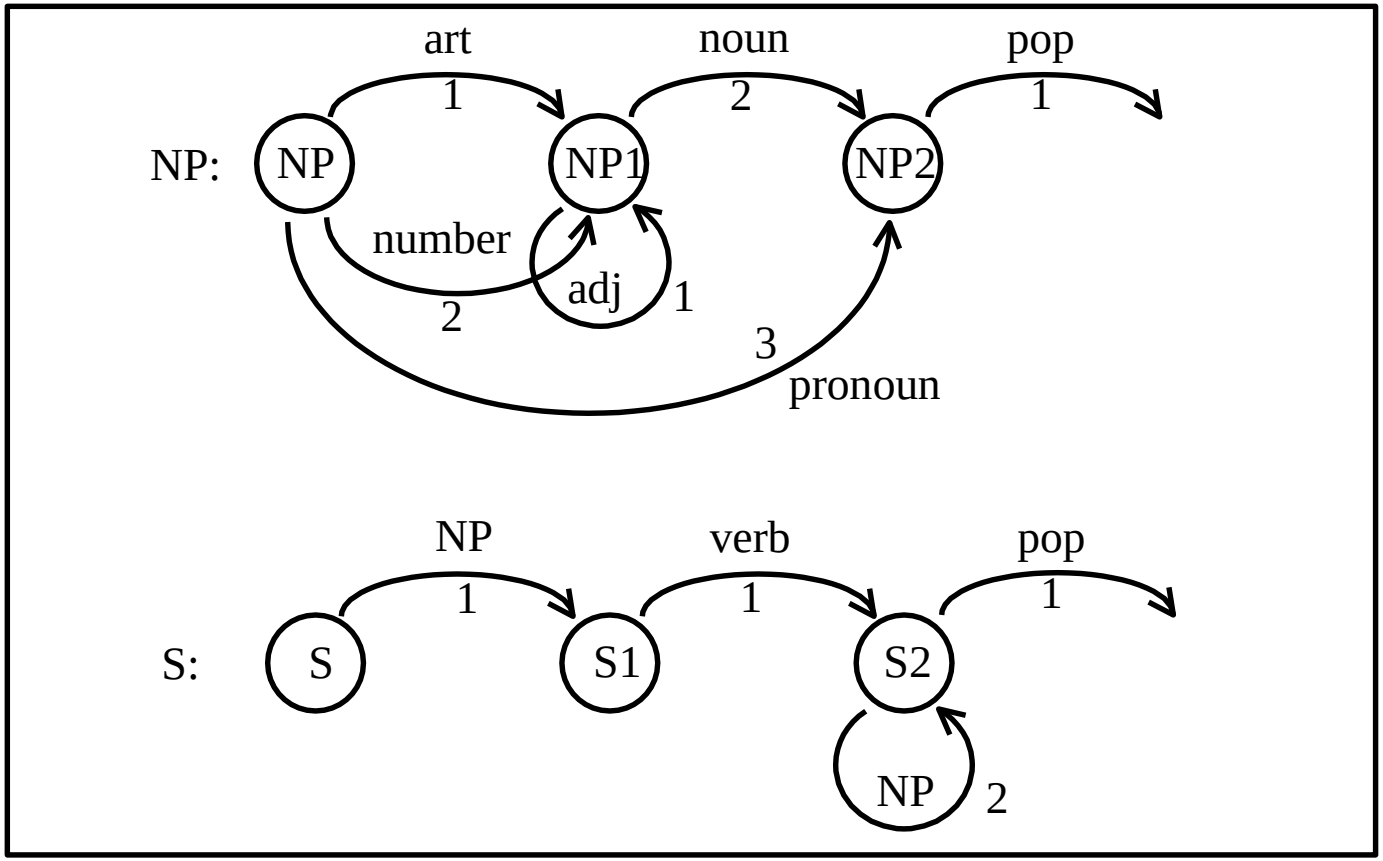
Case 1: IF arc is word category and next word in the sentence is in that category,
THEN (1) update *current position* to start at the next word
(2) update *current node* to the destination of the arc.

Case 2: IF arc is a push arc to a network N,
THEN (1) add the destination of the arc onto return points;
(2) update current node to the starting node in the network N.

Case 3: IF arc is a pop arc and *return points* list is not empty,
THEN (1) remove first return point and make it *current node*

Case 4: IF arc is a pop arc, *return points* list is empty and there are no words left
THEN (1) parse completes successfully

Another Example of RTN



transition network 3 (RTN)

A Trace Using Transition Network 3 of Parsing "$_1$ *The* $_2$ *old* $_3$ *man* $_4$ *cried* $_5$"

Step	Current Node	Current Position	Return Points	Arc to be Followed	Comments
1.	(S,	1,	NIL)	S/1	initial position
2.	(NP,	1,	(S1))	NP/1	followed push arc to NP network, to return to S1
3.	(NP1,	2,	(S1))	NP1/1	followed arc NP/1(*the*)
4.	(NP1,	3,	(S1))	NP1/2	followed arc NP1/1(*old*)
5.	(NP2,	4,	(S1))	NP2/1	followed arc NP1/2(man) since NP1/1 is not applicable
6.	(S1,	4,	NIL)	S1/1	the pop arc gets us back to S1
7.	(S2,	5,	NIL)	S2/1	followed arc S2/1(cried)
8.					Parse succeeds on pop arc from S2

Augmented Transition Network

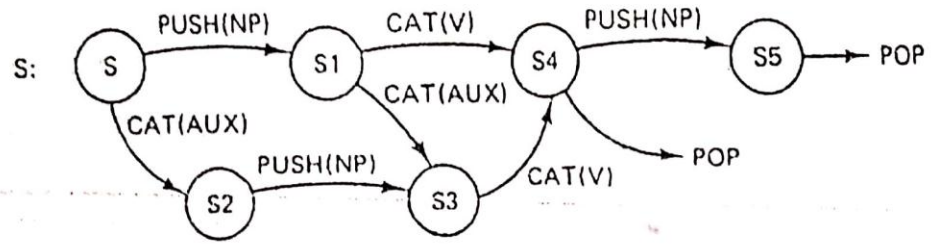
- When we include semantic information with grammar then performance of parser increases.
- We can achieve the additional capabilities by augmenting a RTN with the ability to perform additional tests and store immediate results as a sentence is being parsed, then resulting T.N. is called an Augmented Transition Network

Augmented Transition Network

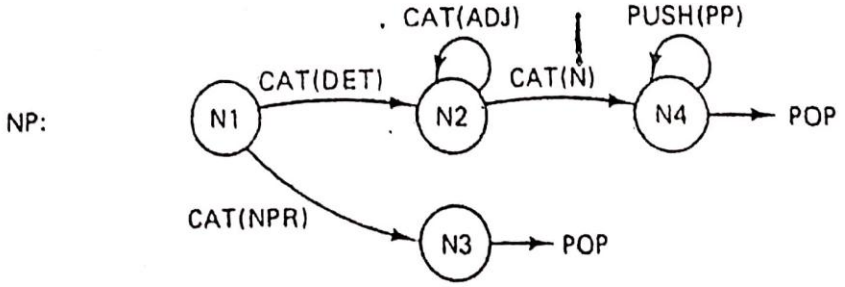
```
<transition net> → (<arc set><arc set>*)
<arc set> → (<state><arc>*)
<arc> → (CAT <category name><test><action>*<term act>)|
        (PUSH <state><test><action>*<term act>)|
        (TST (arbitrary label><test><action>*<term act>)|
        (POP <form><test>)
<action> → (SETR <register><form>)|
           (SENDR <register><form>)|
           (LIFTR <register><form>)
<term act> → (TO <state>)|
             (JUMP <state>)
<form> → (GETR <register>)|
         @|
         (GETF <feature>)|
         (BUILDQ <fragment><register>*)|
         (LIST <form>*)|
         (APPEND <form><form>)|
         (QUOTE <arbitrary structure>)
```

A specification Language for ATN

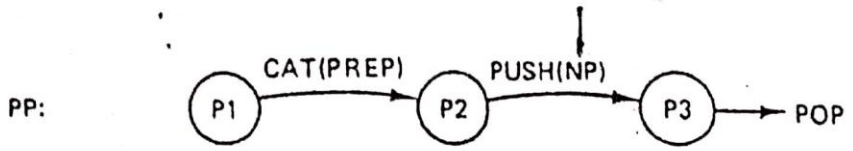
Augmented Transition Network



(a) Top level ATN



(b) Noun phrase subnetwork



(c) Prepositional phrase network

Augmented Transition Network

```
1. (S/  (PUSH NP/ T
2.          (SETR SUBJ @)
3.          (SETR TYPE (QUOTE DCL))
4.          (TO S1)
5.       (CAT AUX T
6.          (SETR AUX @)
7.          (SETR TYPE (QUOTE Q)))
8.          (TO S2)))
9. (S1 (CAT V T
10.         (SETR AUX NIL)
11.         (SETR V @)
12.         (TO S4)))
13.      (CAT AUX T
14.         (SETR AUX @)
15.         (TO S3)))
16. (S2 (PUSH NP/ T
17.         (SETR SUBJ @)
18.         (TO S3)))
19. (S3 (CAT V T
20.         (SETR V @)
21.         (TO S4)))
22. (S4 (POP BUILDQ (S+++(VP+)) TYPE SUBJ AUX V) T)
23.      (PUSH NP/ T
24.         (SETR VP BUILDQ (VP (V+) @) V))
25.         (TO S5)))
26. (S5 (POP (BUILDQ (S++++) TYPE SUBJ AUX VP) T)
27.      (PUSH PP/ T
28.         (SETR VP (APPEND (GETR VP) (LIST @)))
29.         (TO S5)))
```

Thank You!