Natural language processing

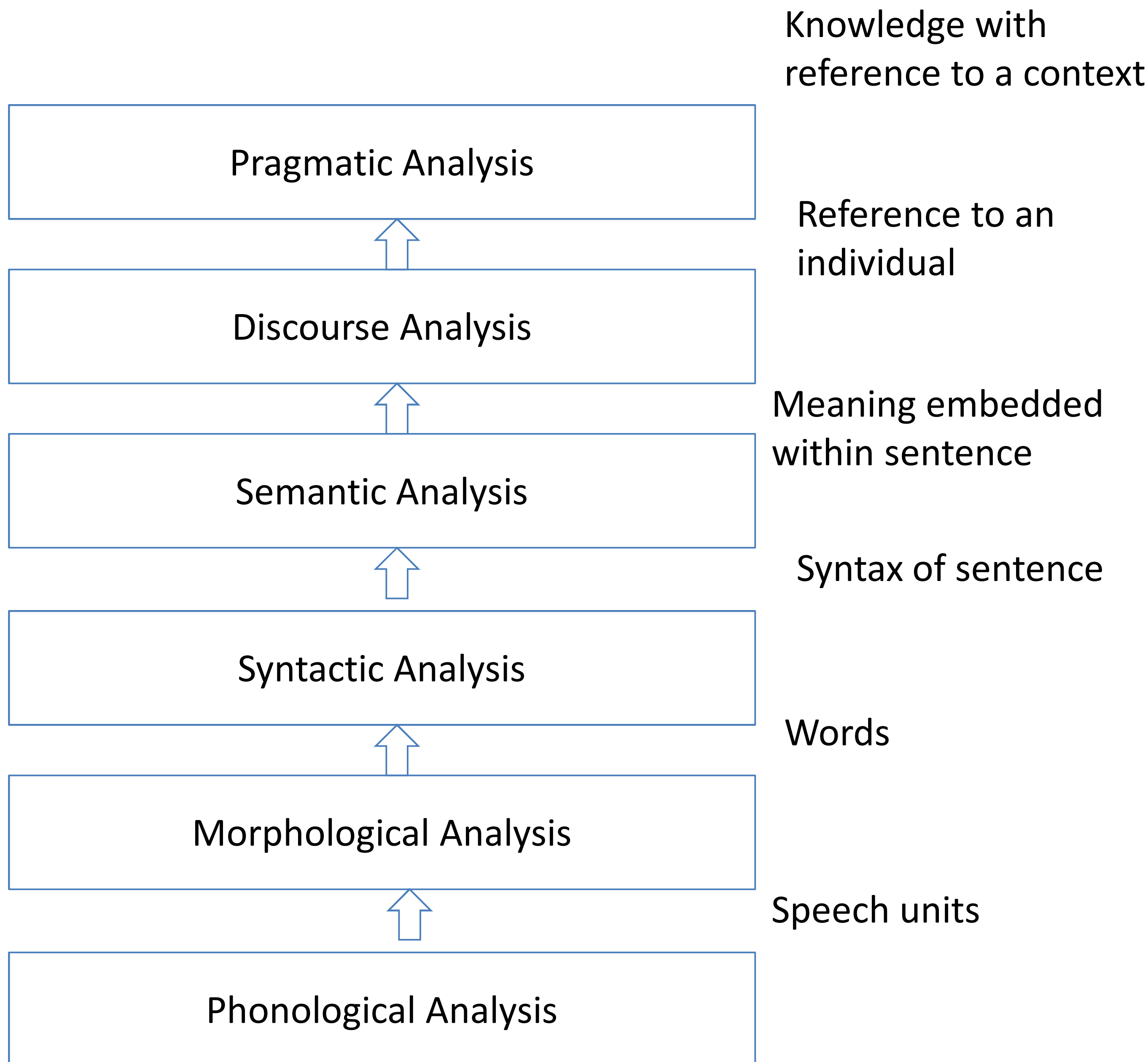
Natural Language Processing

- Perception and communication are important components of Intelligent behaviour
- Developing programs that understand natural language and comprehend visual scene are two most difficult task of AI research
- Entire language processing problem is divided in two task
  - Processing written text
    - Using lexical , syntactic, and semantic knowledge of language
  - Processing spoken language
    - Using all information stated above plus additional knowledge about phonology

Natural Language Processing

- A program understands a natural language if it behaves by taking correct action in response to given input
- Out of two categories of NLP , here focus will be on written text
- In order to understand language , program must have complete knowledge
  - about the structure of the language , grammar, phrases, etc.
  - Must know the meaning of the sentence and context with which they are referred.

Levels of NLP



Components of NLP OR levels of knowledge in NLP

- Phonological
  - Knowledge which related to sound to words we recognize
  - Phoneme is smallest unit of sound
  - Phones are aggregated into word sounds.
- Morphological
  - Lexical knowledge which relates to word construction from basic unit called ***morphemes***
  - *Morpheme* is smallest unit of meaning
  - Eg friendly = friend+ly
  - It gives syntactic category to all words
  - prefix or suffix may give this syntactic information
  - Prints here s indicate plural form hence –s can be marked with it

Components of NLP

- Syntactic knowledge
  - This knowledge is related to how words are put together or structured to form grammatically correct sentence.
  - Exploits the results of morphological analysis to build a structural description of sentence
  - This process is called parsing
  - Flat sentence is converted into structural representation (syntactic tree or network)

Components of NLP

- Semantic analysis
  - Structure created by syntactic analyzer are assigned meaning
  - Must do two things
    - Map individual words to appropriate object in knowledge base
    - Create structures to correspond to the way the meaning of individual words combine .
- Discourse integration
  - Meaning of individual word may depend upon sentence that precede it
  - It may also influence the meaning of following sentence
  - Correct reference to an individual
  - Requires a model to capture such information
  - Example : John wanted it. Meaning if it is clear from preceding sentences
  - Replacing pronoun he/she/them/it with their correct refences (noun)

Components of NLP

- Pragmatic analysis
  - Structure representing what was said is reinterpreted as what was actually meant.
  - High level of knowledge which relates to use of sentences in different context
  - How context effect the meaning of sentence
  - Example : “ Do you know what time it is?”
- These phases are performed in sequence or at once

Syntactic parsing

- It is step in which flat input sentences are converted into hierarchical structures that correspond to units of meaning in the sentence and it is called **parsing**
- Its need in NLP is helpful for following reasons
  - Semantic processing operates on syntactic constituents
    - Constraints the number of constituents
    - Syntactic parsing is computationally less expensive
    - Hence reduces the overall complexity

Approaches to NLU

- Three different method
  - Use of keyword and pattern matching
    - Eliza
    - Advantage : ungrammatical but meaningful information is captured
    - Disadvantage: No actual knowledge structures are created
  - Combined syntactic and semantic analysis
    - ATN and RTN(parsers are used to create structures)
    - Adv: power and versatility
    - Disadv: large computation
  - Comparing and matching input to real world situation
    - CD and frames

Syntactic parsing

- Most of the syntactic system have two components
  - A declarative representation called *grammar* of the syntactic facts about the language.
  - A procedure called *parser* that compare grammar against input sentence to produce parsed structure.

Grammar and language

- Most common way to represent grammars is set of production rules
- Language –A language L is set of strings of finite or infinite length where string is constructed by concatenating basic atomic elements called symbol
- Well formed sentences are formed using rules called grammar
- A grammar G of language L is defined as
  - G=(Vn,Vt,S,P) where
  - Vn=set of non terminal symbol
  - Vt = set of terminal symbol
  - S=starting symbol
  - P=finite set of production rules

Syntactic parsing

- Example of context free phrase structure grammar
- Qn={S, NP, N, VP, V, ART}
- Qt={boy , ate , apple , flew , a , an }
- P: S→ NP VP

  NP → ART N

  VP → V NP

  N → boy| apple

  V → ate | flew

  ART → a|an

  Sentence which can be generated using grammar is

  A boy ate an apple.

  An apple ate the boy

Syntactic parsing

- Irrespective of theory of grammar, parsing process takes rules of the grammar and compares them against the input sentence
- Each matched rule adds to structure being built for the sentence
- Simplest structure is to build *parse tree*
- It records the rules that match with given input.

Parse tree

- S→ NP VP

  ART N VP

  the N VP

  the boy VP

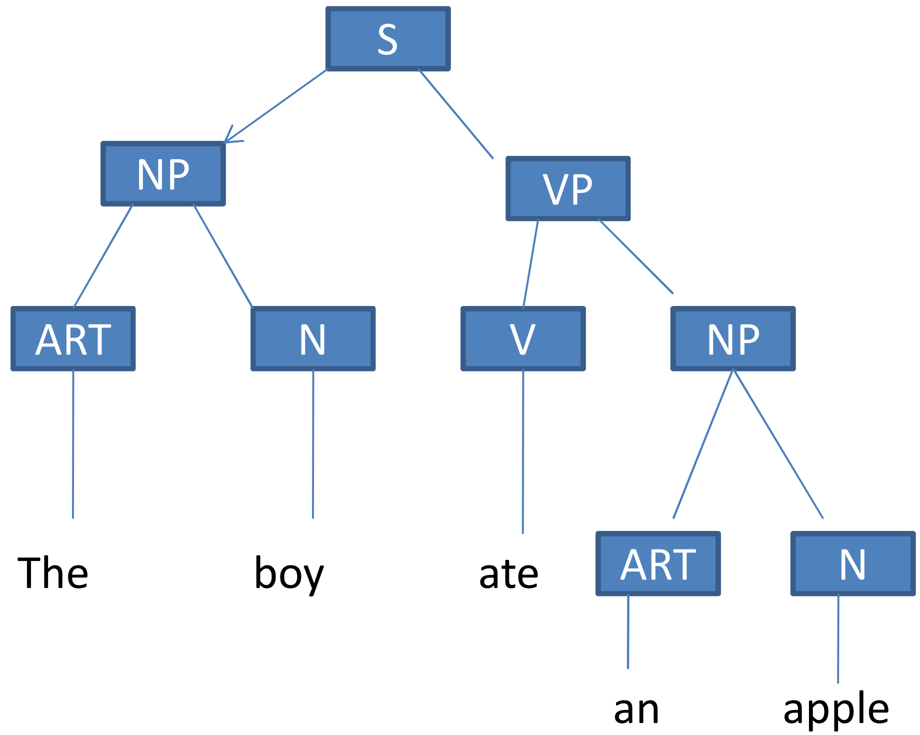
  the boy V NP

  the boy ate NP

  the boy ate ART N

  the boy ate an N

  the boy ate an apple



Structural representation :
A phrase marker or syntactic tree

Grammar and language

- A grammar generates grammatically correct sentences.
- No guarantee for meaningful sentences
- Natural language can not be formally characterized by simple grammar.
- Constrained / formal programming languages have been classified by grammar
- We learn language by learning the structure .
- Useful model of a language is one which characterizes the permissible structures through generative grammar.
- Chomsky has given few classes of languages

Chomsky hierarchy of generative grammar

- Type 0 :
  - is most general
  - Restriction that y cannot be empty in xyz $\rightarrow$ xwz
- Type 1 :
  - Context sensitive grammar
  - Restriction :
    - length of string on RHS of rewrite rue must be as long as string on LHS
    - Production of form xyz$\rightarrow$ xwz , y must be non terminal symbol
- Type 2
  - Context free grammar
  - Characterized by form
    - <symbol>$\rightarrow$<symbol1>....<symbol k> where k >=1
    - LHS is single non terminal
- Type 3
  - Finite state regular grammar
  - Characterized by the form A$\rightarrow$aB, A$\rightarrow$a

Structural representation

- More extensive English grammar can be obtained by adding other constituents like
  - Propositional phrases PP
  - Adjectives ADJ
  - Determiners DET
  - Adverbs ADV
  - Auxiliary verbs AUX
- Production rules can be enhanced as follows
  - PP → PREP NP
  - VP → V ADV
  - VP → V PP
  - VP → V NP PP
  - VP → AUX V NP
  - DET → ART ADJ
  - DET → ART

Structural representation

- Eg
  - The mean boy locked the dog in the house.
  - The cute girl worked to make some extra money .
  - The/DT cute/JJ girl/NN worked/VBD to/TO make/VB some/DT extra/JJ money/NN
- These will have the form
  - S $\rightarrow$ NP VP PP

Transformational grammar

- Generative grammar produce different structure for different sentence with same meaning
- Eg
  - Mother baked the cake.
  - Cake was baked by the mother.
- Sentence with same meaning , but different internal structure is undesirable in NLP
- Same meaning sentence must map to same internal knowledge structure.
- Chomsky extended the generative grammar by adding two components to it
  - Semantic component
  - Phonological component
- This was called transformational grammar

Transformational grammar

- Transformation rules can produce change from active to passive
- Other classification of grammar are
  - Case grammar
  - Systemic grammar
  - Semantic grammar

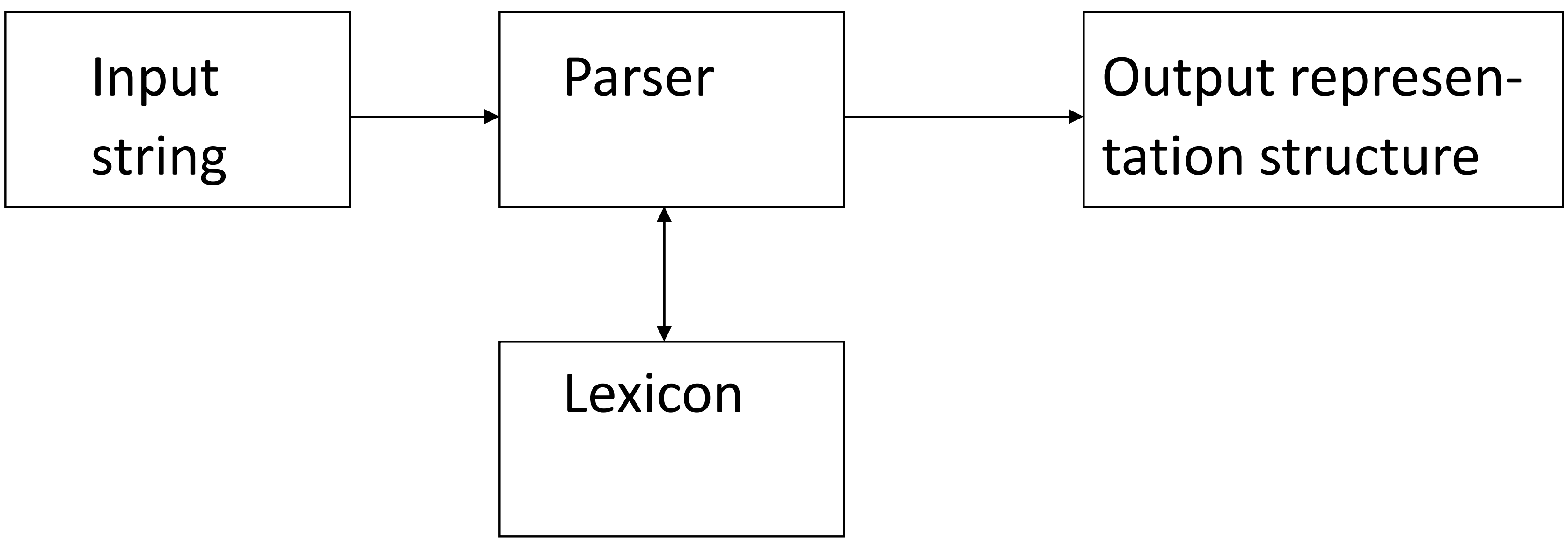
Basic Parsing Techniques

Parsing

- Determining the syntactical structure of a sentence.
- Inverse of sentence generation process.

Parser :

- Uses lexicon to determine the meaning of a word.



Parsing an input to create an output

The Lexicon

Lexicon :

A dictionary of words containing syntactic, semantic and pragmatic information. The entries of a lexicon may not be the same.

The Lexicon (Contd..)

Example Lexicon :

Word	Type	Features
a	Determiner	{ 3 s } 3 s means third person singular
be	Verb	Trans : Intransitive
boy	Noun	{ 3 s }
can	Noun	{ 1s, 2s, 3s, 1p, 2p, 3p }
	Verb	Trans : Intransitive
orange	Adjective	{ 3 s }
	Noun	

Parsing techniques

- To parse a sentence it is necessary to find the way in which that sentence is generated from start symbol
  - Top down parsing
  - Bottom up parsing
- A Top-down parser
  - begin with a start symbol
  - apply grammar rules in forward direction
  - Until symbol at terminal correspond to components of sentence
- Bottom up parsing
  - Begin with sentence to be parsed
  - Apply grammar rules backwards
  - Until a single tree whose terminals are words of sentence
  - And whose top node is start symbol is produced.

Example of top down parsing

“ Kathy jumped the horse “

S → NP VP
→ Noun VP
→ Kathy VP
→ Kathy V VP
→ Kathy jumped NP
→ Kathy jumped article N
→ Kathy jumped the N
→ Kathy jumped the horse

Top-Down Versus Bottom-Up Parsing

- A Bottom-up parser is data driven because it begins with the actual words in sentence, where as top down parsing begin by hypothesizing sentence S and predicting components.

  Kathy jumped the horse

  → name jumped the horse

  → name V the horse

  → name V art horse

  → name V art N

  → NP V art N

  → NP V NP

  → NP VP

  → S

Deterministic and non deterministic parser

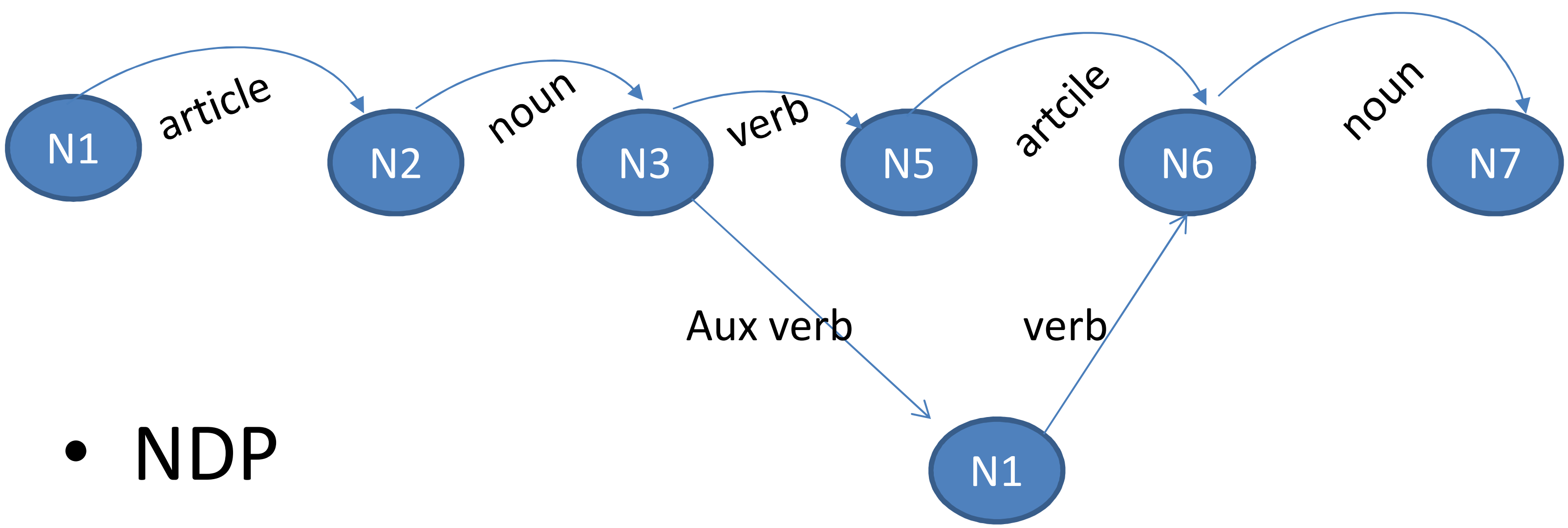
- Deterministic parser
  - Permits only one choice (arc) for each word category .
  - Each arc will have different test condition
  - If incorrect test choice is accepted at some state then parser will fail
  - Parser cannot backtrack to an alternative choice
  - Eg when word satisfies more than one category

Deterministic and non deterministic parser

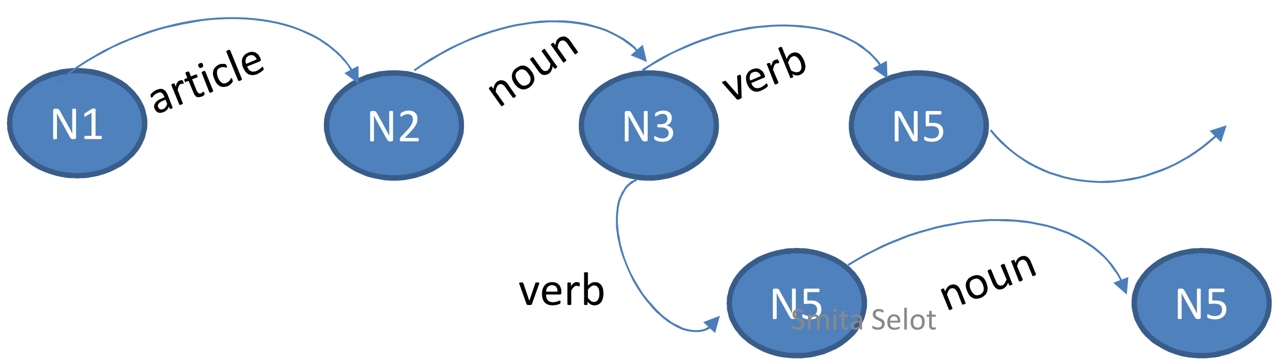
- Non-deterministic parser
  - Permit different arcs to be labeled with same test
  - Next test from any other state may not be uniquely determined
  - Parser makes a guess
  - If guess is wrong it backtracks
  - Require saving more than one potential structure during parts of network traversal
- Eg “ The strong bears the load”
  - If strong is taken as adjective and bear as noun the DP will fail as verb do not follow
  - NDP will back track and try a better option.

Example

- DP



- NDP



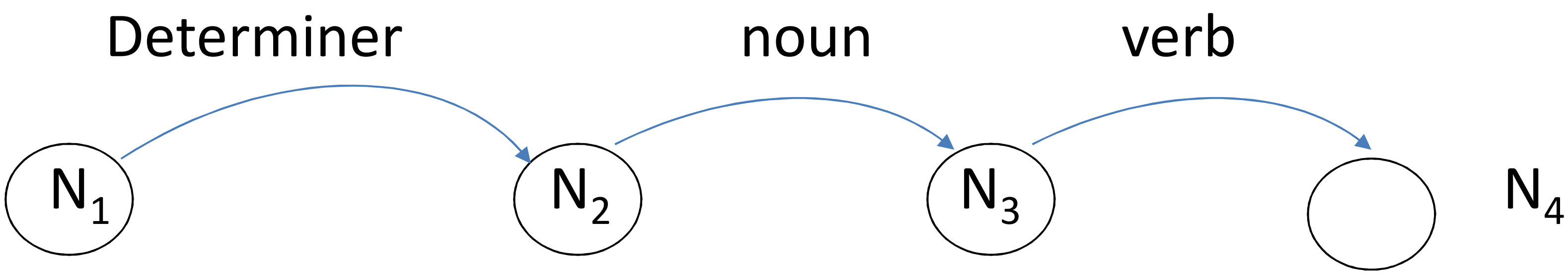
Transition Networks

- Used to represent formal and natural language structures
- Application of directed graph and finite state automata.
- Consists of a number of nodes and labeled arcs.
- Nodes represent different states in traversing a sentence
- Arcs represent rules or test conditions to make the transition from state to next state

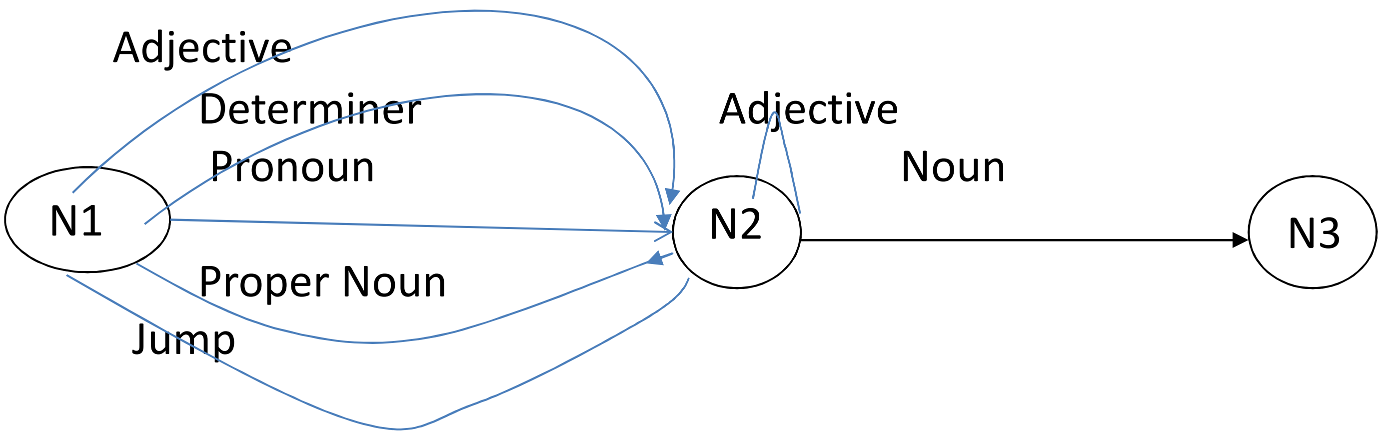
Transition network

- A path in TN = permissible sequence of word types for a given grammar
- If sentence is traversed successfully , it has a recognized permissible structure

Transition Networks (Contd..)



Sentence parsed: The Child Runs



Sentences parsed

Big white fluffy clouds

Our bright children

Large green leaves

Buildings

A large beautiful white flowers

Transition Networks (Contd..)

To move from N_1 to N_2 it is necessary to find an adjective, a pronoun, a determiner, a proper noun or none of these by jumping directly to N_2.

Examples

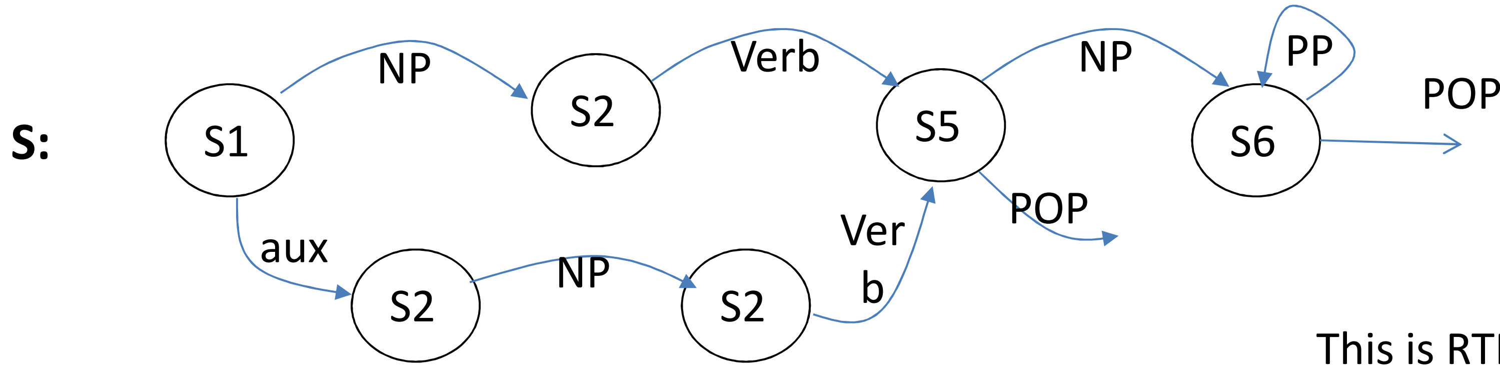
- Big white fluffy clouds
- Our bright children
- A large beautiful white flower
- Large green leaves

Types Transition Networks

1. Recursive Transition Networks (RTN)
   I. more powerful than simple networks
   II. RTN is a transition network which permits arc labels to refer to other networks and they in turn may refer back to the referring network.
2. Augmented Transition Network (ATN
   I. More power ful than RTN
   II. Requires storage
   III. Returns structure unlike RTN, useful for further analysis of sentences
   IV. Requires a specific language to process

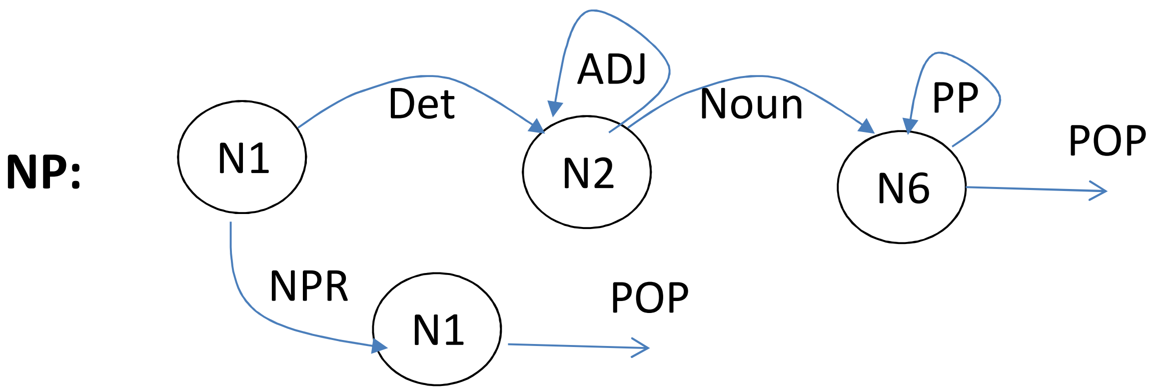
RTN

- To get descriptive power in TN , recursion is added
- Arc labels refer to
  - A particular category test
  - Call to another TN (may be recursive also)
  - Labels referring to other network are written in upper case

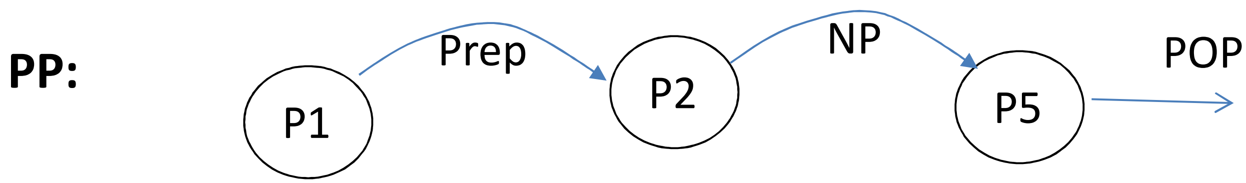


(a) Top level RTN

This is RTN described by William Woods in 1970 where main network calls two subnetwork



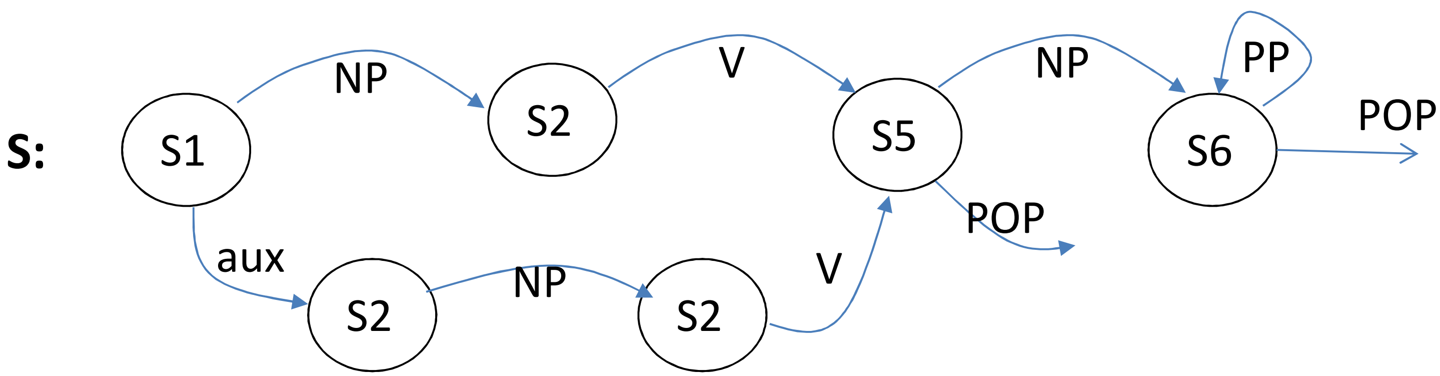
(b) Noun Phrase Network



© Propositional phrase Network

Recursive Transition Network

The big tree   shades   the small house   by the river



Three data structures used while processing the sentence through RTN are as follows:

- •POS : Current input word position
- •CND: current Node
- •RLIST: Stack data structure containing list of return points

RTN parser will terminate under two condition

(1) If end of sentence '.' is encountered

(2) word in the input sentence fail to satisfy any one of the available arc test from some node in the network

RTN

- Drawback
  - Only accept or rejects a sentence with respect to grammar
  - No definite structure is provided for further analysis
  - On failure alternate path is not available
  - Not very useful in language understanding
- Adv
  - Simple mechanism to test syntactic structure of sentence
  - No overhead is required once fundamental networks are built

Augmented Transition Network

- Number of sentence accepted can be extended if backtracking is allowed on occurrence of failures
- Parser must have the capability to build structure which will be used to create required knowledge entities
- Resulting data structure must contain more information than just syntactic information
- Semantic should also be included
- For this ,additional information is desired, like
  – Subject NP, Object NP, subject-verb number argument
  – Mood (declarative interrogative)
  – Tense, location etc

Augmented Transition Networks (Contd..)

- To include more semantic information into structure, RTN is augmented with these information
- Additional tests performed for semantic features and structure are stored.
- RTN with these additional capabilities are called ATN
- For building structure ,Temporary storage registers are used in ATN.
  - A set of registers for NP network
  - A set of registrar for PP network
  - Register for verb
- Using these register ATN builds a partial description of the sentence as it moves from one state to another
- These register provide temporary storage which can be modified , erased or discarded
- Register also holds flag and indicators used in conjunction with some arc
- After successful parsing, contents of registers are combined for final sentence structure

Augmented Transition Networks (Contd..)

Word	Definition	Word	Definition
a	Part-of-speech: article Root: a Number: Singular	Like	Part-of-speech: verb Root: like Number: Plural
men	Part-of-speech: noun Root: man Number: Plural	Likes	Part-of-speech: verb Root: like Number: Singular
		Dog	Part-of-speech: noun Root : dog Number: Singular

Dictionary of Entries for a Simple ATN

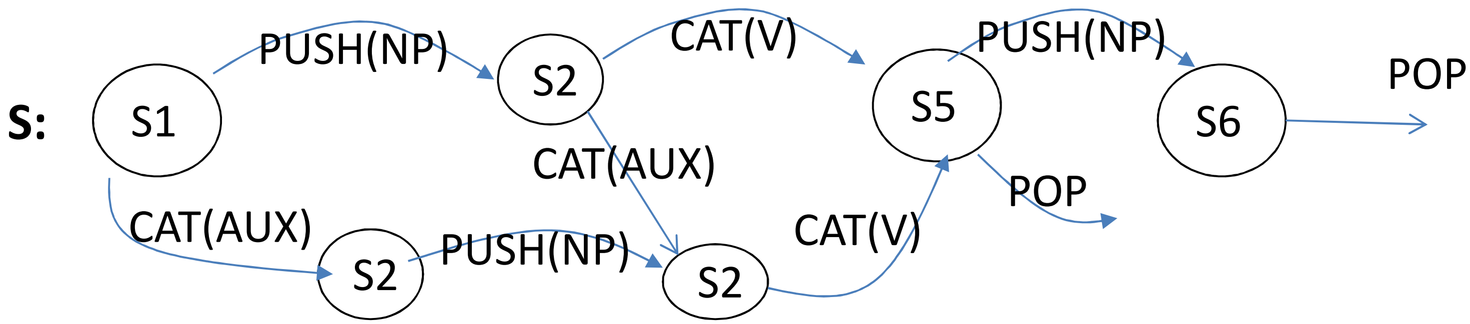
ATN specification language

- Given by woods it is extended context free grammar
- Using specification language we can represent the particular network with constituent abbreviation and functions described in from of a LISP Program

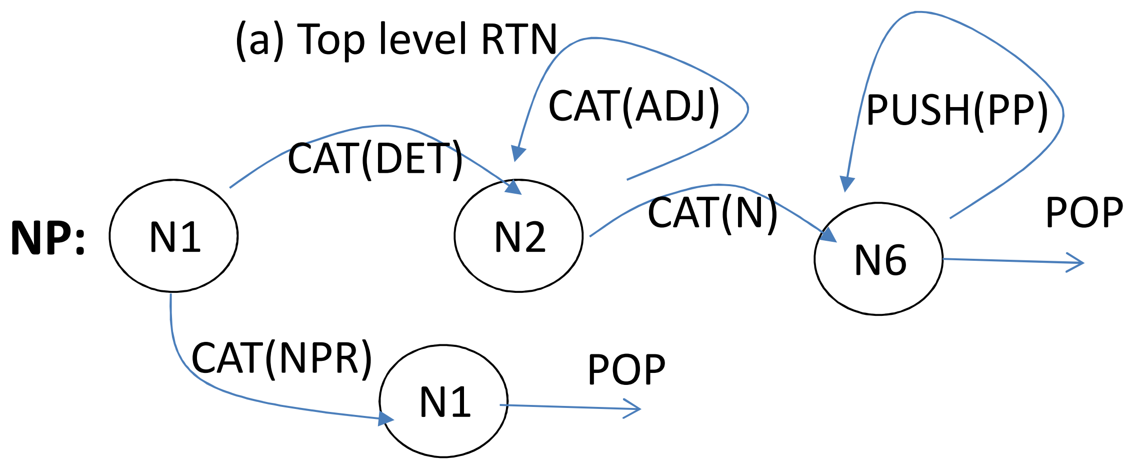
- | alternative choice
- * repeatable zeros
- () non terminals
- Function and test
- @ input word

ATN specification language

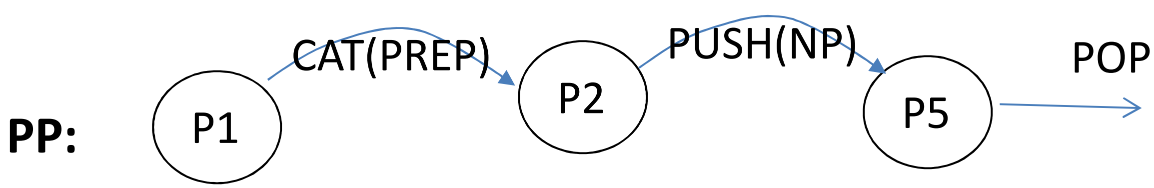
- Arc
  - CAT , JUMP, PUSH, TEST, WORD POP
- Action in form of function
  - **SETR**- which causes indicated register to be set to a value, done in the current level
  - **SENDR** causes it to be done by sending it to lower level
  - **LIFTR** return the information to next higher level of computation
  - **GETR** returns the value of indicated register
  - **BUILQ** takes the list from indicated registers
  - **TO** require input sentence pointer should be advanced
  - **JUMP** require pointer remain fixed and input word continue to scan

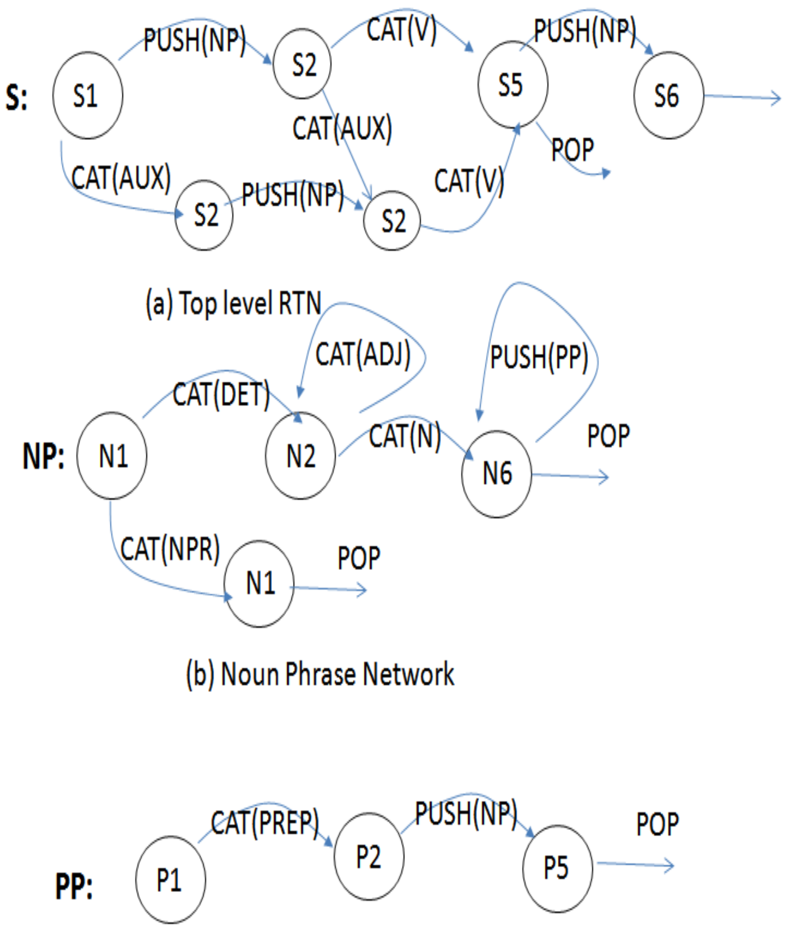


(a) Top level RTN



(b) Noun Phrase Network





(a) Top level RTN

(b) Noun Phrase Network

1. (S/ (PUSH NP/T
2. (SETR SUB @)
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)))

Example “ The big boy likes the small dog”

Output structure generated by BUILDQ is

(S DCL (NP(boy(big) DEF))

(VP (V likes)(NP (boy(small) DEF))))

Register used TYPE, SUBJ,AUX, VP

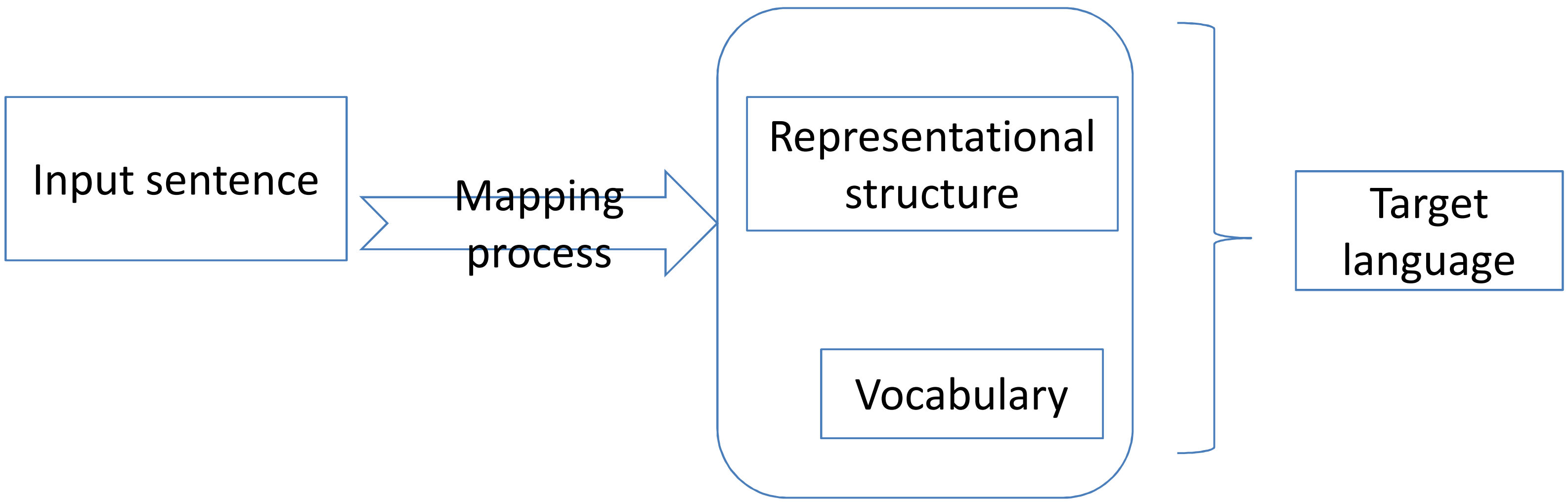
Other example

- The boy can whistle
- (S DCL (NP (boy) DET) (AUX CAN)(VP whistle)
- Constructed from TYPE, AUX, V AND SUBJ
- Adv
  - Use of arcs, test power of Turing machine
  - ATN can recognize any language
  - Build deep sentence structure rather than surface structure
  - Power and versatility made them more popular
- Disadv
  - Require computational and storage overhead
  - Backtrackings cause more time
  - Complex structure

Semantic Analysis

- Creating parse tree is first step for understanding
- Presentation of meaning of sentence is required
- Understanding is mapping process
- We must define the language in which we are trying to map
- There is no single language in which meaning of all sentence can fit
- We need a knowledge representation structure and the vocabulary

Semantic analysis



Two category of target language used in NL depending on the role of NL

When NL is considered a phenomenon of its own

Ex- question-answer system

primitive that correspond to distinction made in a language

When NL is used as in interface language to another program

Ex-expert system or database query system

Design of target language is depends on backend

In both the cases intermediate knowledge driven representation is significant

Function of semantic analysis

- Creation of target language representation of sentence meaning
- Imposes constraint on representational structure
- It provides a way to select a syntactic analysis which will suit the application

Lexical processing

- First step in semantic analysis
  - Look up the meaning of word in lexicon
  - One word may have different meaning
  - Results in lexical ambiguity
  - Process of determining correct meaning of the word is known as *word sense disambiguation* or *lexical disambiguation*
  - It is done by associating with each word the context in which the word sense may appear

Word Sense Disambiguation

- For example the word diamond may have following meaning
    - Geometrical shape
    - A stadium
    - Valuable gemstone
- To identify the correct meaning, one of parameter can be
    - nor the shape or stadium shimmers where as gemstone do
- Stadium diamond can be seen as *location* as the sentence *I will meet you at diamond*
- Occurrence of '*at*' indicated that word should be either a time or location
- Such properties of word senses are called *semantic markers*
- Ex of these markers are
    - PHYSICAL OBJECT
    - ANIMATE OBJECT
    - ABTRACT OBJECT
    - SHIMMERABLE/NON-SHIMMERABLE

Different approaches for Semantic Analysis

- Semantic grammar
- Case grammar
- Conceptual parsing
- Compositional semantic processing

Semantic grammar

- Is context free grammar in choice of non terminal and production rule is governed by the syntax and semantic function
- Close association between semantic and grammar rules
- This close coupling between semantic and grammar works as grammar of language is built around the key semantic concept

Example of 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
FILE1-> USER'S FILE2
ACTION -> print FILE
{instance printing
object : FILE}
ACTION -> print FILE on PRINTER
{instance: printing
object:FILE
printer: PRINTER}

Example : I want to print Bill’s .init file

{ command { instance : printing
object : { instance : file-struct
extension : .init
owner: Bill }

ACTION :
{instnce : printing
object : { instance : file-struct
extension : .init
owner: Bill }} a

FILE

FILE1
{instance : file-struct
extension : .init
owner: Bill}

FILE2
{instance : file-struct
extension : .init
owner: Bill}

I want to print Bill’s .init file

Semantic grammar

- Adv
  - After parsing result can directly be used without additional processing
  - Many ambiguities that arise due to pure syntactic parsing is avoided
  - Some syntactic issues can be ignored
- Disadv
  - Number of rules required can become very large
  - As a result parsing process may be expensive

Case grammar

- Case grammar given by Fillmore 1968
- Different approach for combing syntactic and semantic information
- Syntactic representation of
  - ‘Susan printed the file’ and
  - ‘File was printed by Susan’
- Will result in different syntactic structure
- But they have same meaning
- Semantic structure
  - (printed (agent Susan)
    (object File)

Case Grammar

- Cases describe the relation between verb and their arguments
- Set of cases
  - (A) Agent: Instigator of the action (animate)
  - (I) Instrument : used in causing event (inanimate)
  - (D) Dative: Entity effected by action (animate)
  - (F) Factive: resulting from event
  - (L) Locative: place of event

Case grammar

- Parsing is heavily dependent on lexical entities associated with verb
- Expectation driven parsing: once verb is located noun phrases can be predicted

LUNAR system

- NLP based system
- Designed as language interface for geologists
- To give direct access to database containing lunar rock and composition
- During NASA Apollo II moon landing mission
- Main three components
  - ATN parser with capability to handle large set of sentences
  - Rule driven semantic interpreter which transform syntactic representation to logic am form
  - Database retrieval and inference component