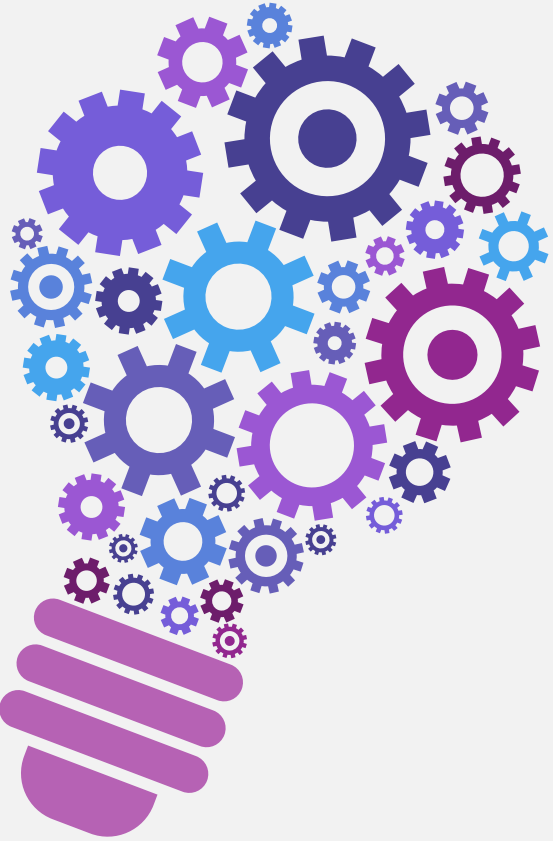
Natural Language for Communication

Presented To
Dipanita Saha
Assistant Professor
IIT,NSTU

Presented By
Nowal Benta Basher
Anwar Kabir
Akash Deb Nath
Khair Ahmed
Nadim Bhuiyan
Sawacca Pathan
Suvo Islam



Communication & Sign

- **Communication** is the intentional exchange of information brought about by the production **SIGN** and perception of signs drawn from a shared system of conventional signs.
- Communication can help agents be successful because they can learn information that is observed or inferred by others.
- We start with grammatical models of the phrase structure of sentences, add semantics to the model, and then apply it to machine translation and speech recognition.

Phrase Structure Grammar

- **Phrase structure grammar** is a type of generative grammar in which constituent structures are represented by phrase structure rules.
- The n-gram language models based on sequences of words. The big issue for these models is data sparsity. We can address the problem of sparsity through generalization.

A Simple Set of Phrase Structure Rules

S	⟶	NP +VP
NP	⟶	art + (adj*) + N
VP	⟶	V + NP + (PP*)
PP	⟶	Prep + NP

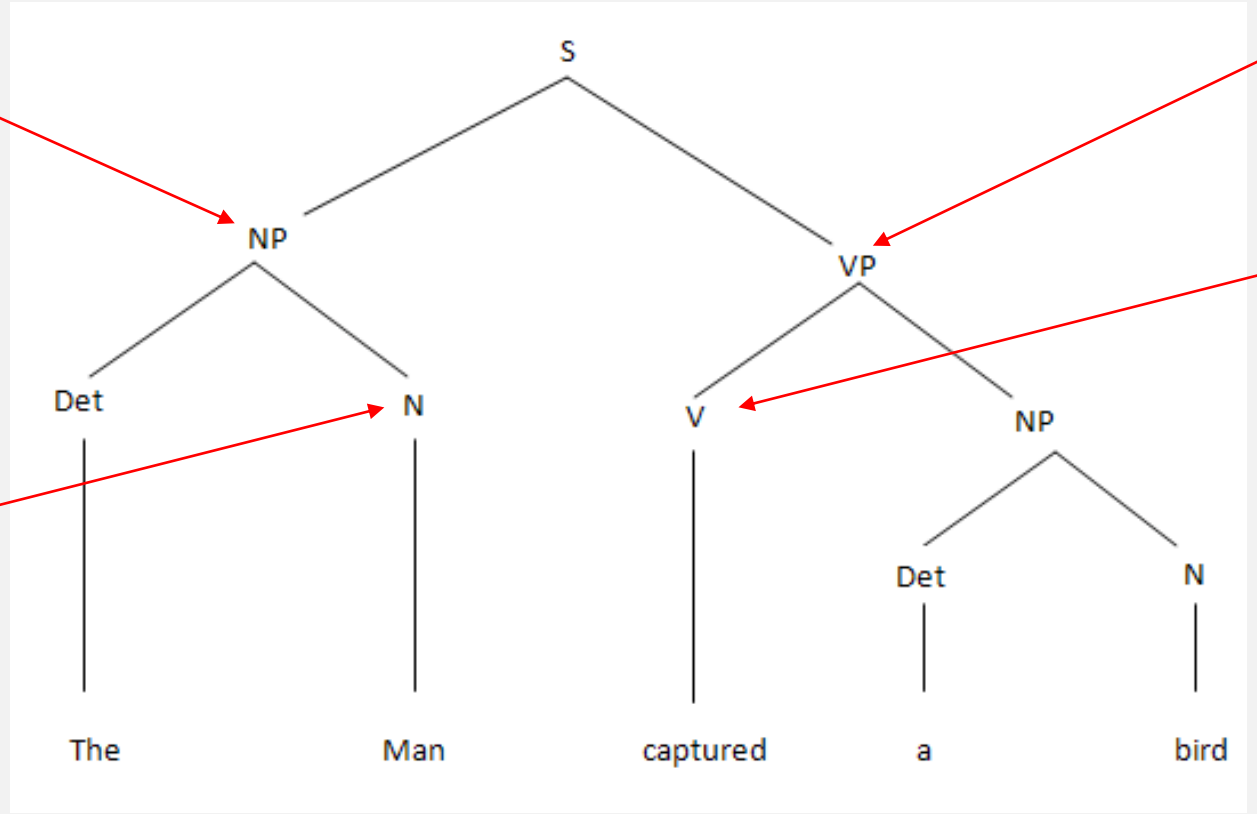
Phrase Structure Grammar

syntactic categories

syntactic categories

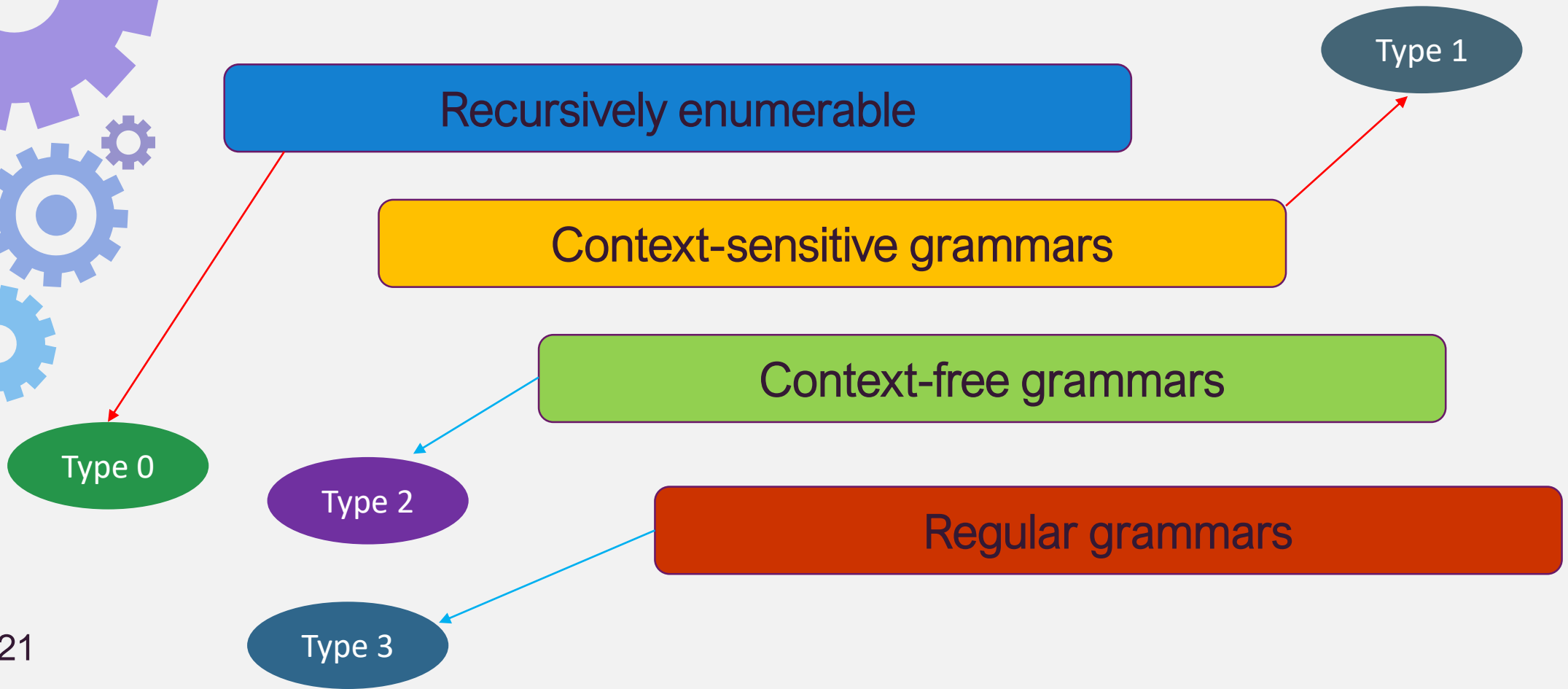
lexical category

lexical category



Grammatical formalisms

Grammatical formalisms can be classified by their generative capacity



Grammatical formalisms

Recursively enumerable

- ✓ unrestricted rules.
- ✓ both sides of the rewrite rules can have any number of terminal and nonterminal symbols.

α->β
α ε (V+T)*V(V+T)*
β ε (V+T)*
Where, α≠null

Example:
aAb->bB
aA->null

Context-sensitive grammars

- ✓ are restricted only in that the right-hand side must contain at least as many symbols as the left-hand side.

$$|\alpha| \leq |\beta|$$

Example:
aAb->bbb
aA->bbb

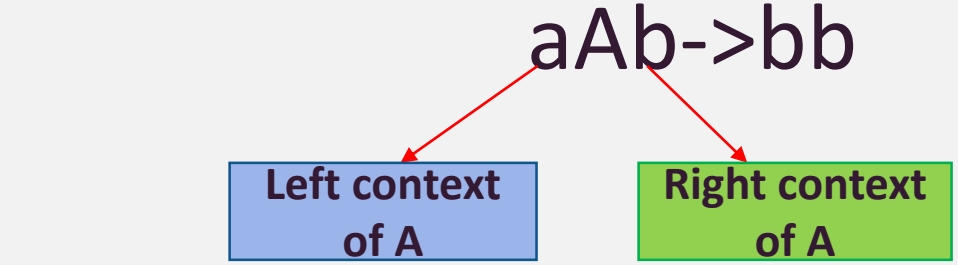
aAb->bb

This is allow REL but not allow CSG

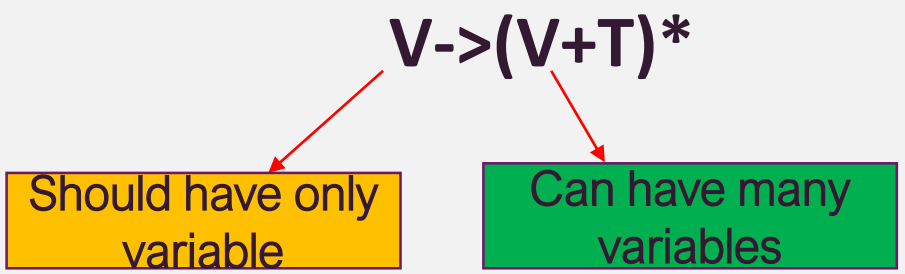
Grammatical formalisms

Context-free grammars

✓ The left-hand side consists of a single nonterminal symbol



In CFG, Production rule is,

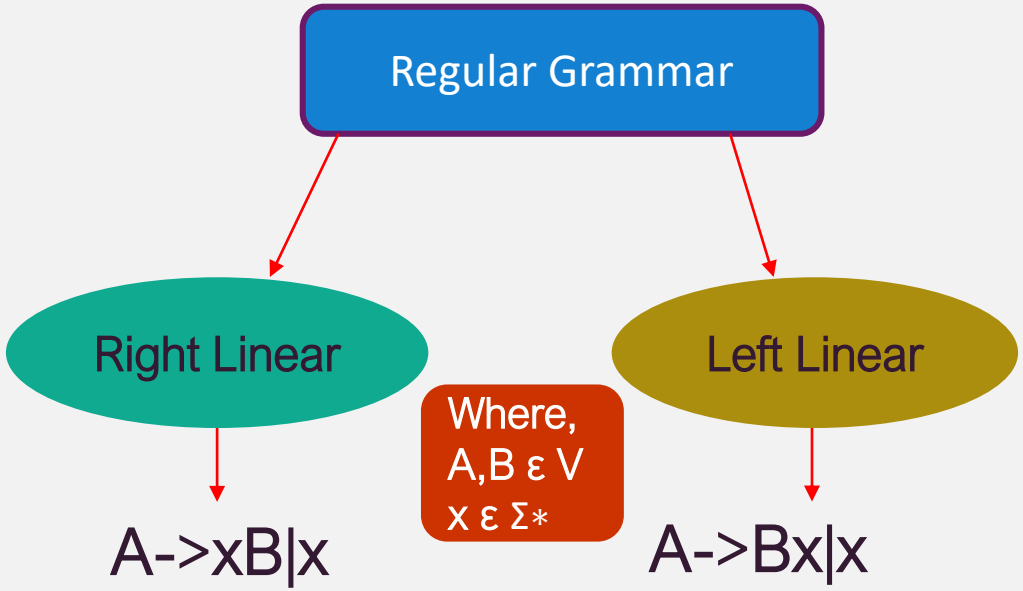


Example:
A->α
α ε (V+T)*
A-> null
A->BCD

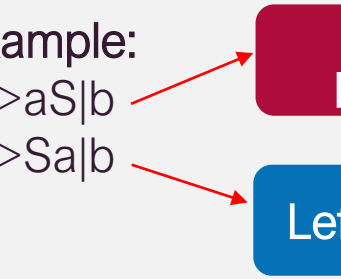
Grammatical formalisms

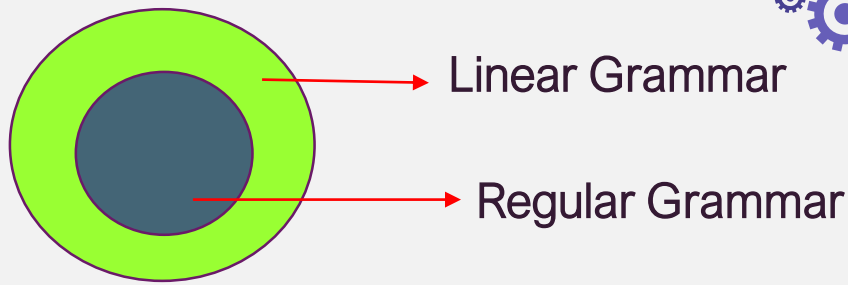
Regular grammars

- ✓ Exactly one variable on LHS and at most one variable must exist in RHS.
- ✓ Like as Linear Grammar



Example:
S->aS|b
S->Sa|b





Relation Between Four Grammars

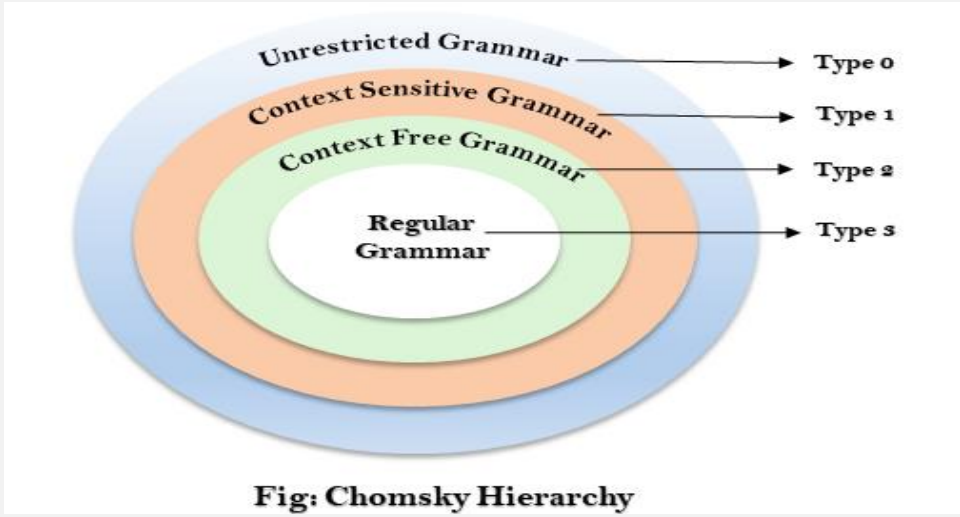


Fig: Chomsky Hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursive Enumerable	Turing Machine
Type-1	Context Sensitive	Context Sensitive	Linear-Bound
Type-2	Context Free	Context Free	Pushdown
Type-3	Regular	Regular	Finite

Type 3 ⊆ Type 2, Type 1, Type 0

Type 2 ⊆ Type 1, Type 0

Type 1 ⊆ Type 0

Expressive Power of Four Languages

Expressive Power : Number of languages accepted by a particular machine known as Expressive power of that machine.

Finite Automata: Regular Language

Push Down Automata: CFL,RL

Linear Bounded Automata: CSL,CFL,RL

Turing Machine: REL,CSL,CFL,RL

TM>LBA>PDA>FA

Identify Languages with Examples

Example 01:
S->as|a

This is Type 3 that means RL and so this language is automatic CSL,CFL,REL

Example 02:
S->AB
A->A
B->b

This is Type 2 that means CFL so this language is automatic CSL and REL

Example 03:
aAb->bBab

This is Type 1 that means CSL so this language is automatic REL

Probabilistic Context-free Grammar

- ✓ A **probabilistic context free grammar** consists of terminal and nonterminal variables.
- ✓ Grammar assigns a probability to every string.

Here is a PCFG rule:
VP → Verb [0.70]
| VP NP [0.30] .

Terminal Symbol

Non-Terminal Symbol

This rule is saying that with probability 0.70 a verb phrase consists solely of a verb, and with probability 0.30 it is a VP followed by an NP.

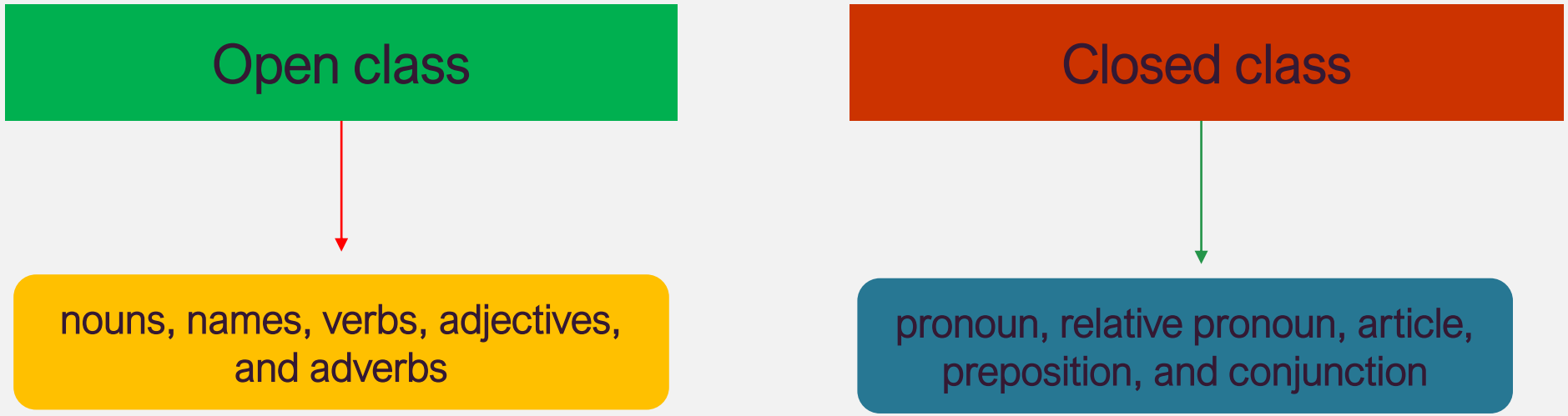
The Lexicon of E0

- ✓ A grammar for a tiny fragment of English that is suitable for communication, we call the language E0. Lexicon is list of allowable words.
- ✓ The words are grouped into the lexical categories. These are nouns, pronouns, verbs, adjectives, adverbs, articles, prepositions, and conjunctions.

Noun	→	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | . . .
Verb	→	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | . . .
Adjective	→	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] . . .
Adverb	→	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | . . .
Pronoun	→	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | . . .
RelPro	→	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] ∨ . . .
Name	→	**John** [0.01] | **Mary** [0.01] | **Boston** [0.01] | . . .
Article	→	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | . . .
Prep	→	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | . . .
Conj	→	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] ∨ . . .
Digit	→	**0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | . . .

The Lexicon of E0

In lexical categories there are two classes.



The Lexicon of E0

Open class

In open class, new words can be added to the class as the need arises. In the late twentieth century, developments in computer technology have given rise to many new nouns : Internet, website, URL, CD-ROM, email, newsgroup, bitmap, mode m, multimedia

New verbs have also been introduced : download, upload, reboot.

Closed class

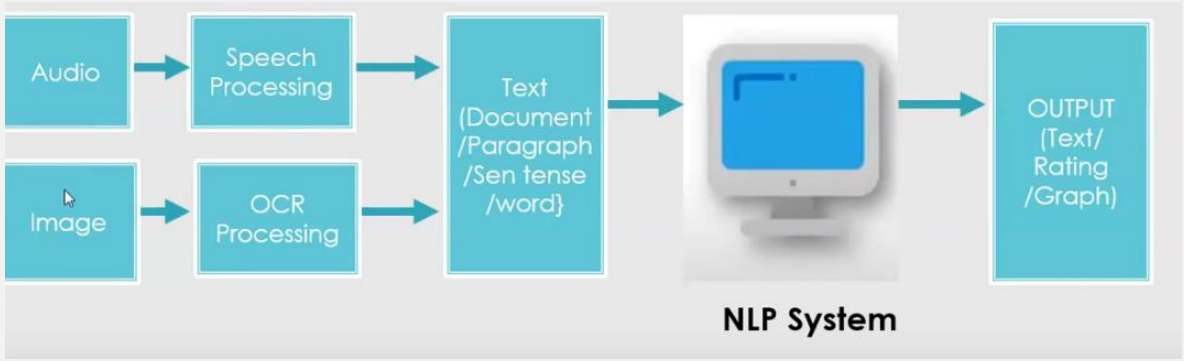
In close class, new words can not be added to easily and this class has finite words. we never invent or expanded new words easily. For example, “thee” and “thou” were commonly used pronouns in the 17th century, were on the decline in the 19th, and are seen today only in poetry and some regional dialects.

Grammar

CFG means A set of Rules/productions used to generate pattern of string.
CFG ->4 Taples(V,T,P,S)

V=Variable/Non terminal
T=Terminal
P= Production/Rules
S= Start variable

S ->AB
A ->BB|a
B ->AB|b



Syntactic Analysis

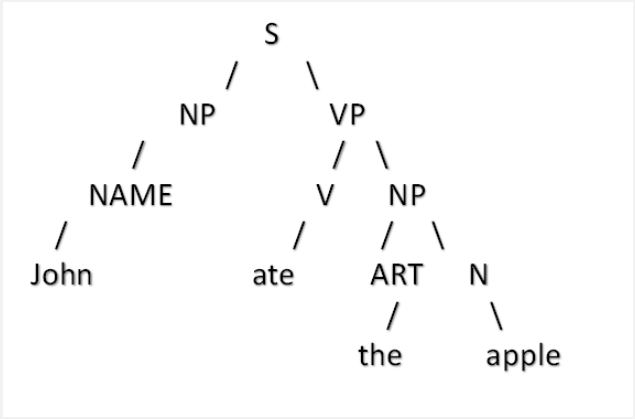
Whether the given sentence is grammatically right or wrong?

John ate the apple
Ate the app John

S->NP VP
VP->V NP
NP->NAME
NP->ART N

NAME->John
Verb->ate
ART->The
N->apple

A parse tree



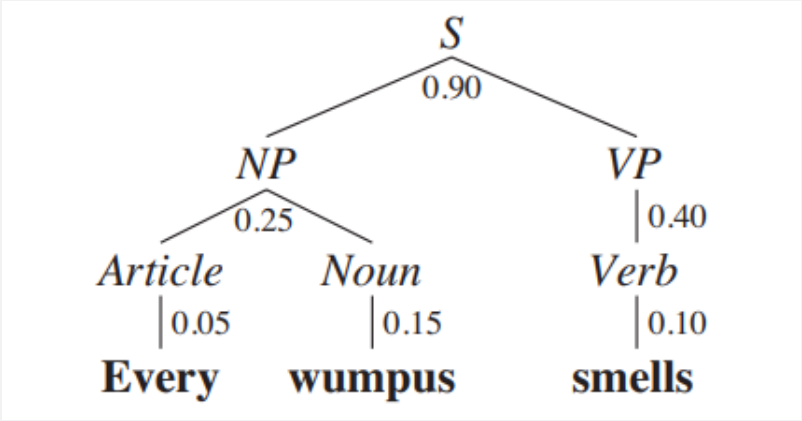
Grammar Rules

- The grammar for E0, with example phrases for each rule. The syntactic categories are sentence (S), noun phrase (NP), verb phrase (VP), list of adjectives (Adjs), prepositional phrase (PP), and relative clause (RelClause)

$\mathcal{E}_0$:	S	$\rightarrow$	*NP VP*	[0.90] I + feel a breeze
		|	*S Conj S*	[0.10] I feel a breeze + and + It stinks
	NP	$\rightarrow$	*Pronoun*	[0.30] I
		|	*Name*	[0.10] John
		|	*Noun*	[0.10] pits
		|	*Article Noun*	[0.25] the + wumpus
		|	*Article Adjs Noun*	[0.05] the + smelly dead + wumpus
		|	*Digit Digit*	[0.05] 3 4
		|	*NP PP*	[0.10] the wumpus + in 1 3
		|	*NP RelClause*	[0.05] the wumpus + that is smelly
	VP	$\rightarrow$	*Verb*	[0.40] stinks
		|	*VP NP*	[0.35] feel + a breeze
		|	*VP Adjective*	[0.05] smells + dead
		|	*VP PP*	[0.10] is + in 1 3
		|	*VP Adverb*	[0.10] go + ahead
	Adjs	$\rightarrow$	*Adjective*	[0.80] smelly
		|	*Adjective Adjs*	[0.20] smelly + dead
	PP	$\rightarrow$	*Prep NP*	[1.00] to + the east
	RelClause	$\rightarrow$	*RelPro VP*	[1.00] that + is smelly

Examples

Parse tree for the sentence “Every wumpus smells” according to the grammar E0.



The tree can also be written in linear form as
[S [NP [Article every] [Noun wumpus]][VP [Verb smells]]].

Examples

Fall leaves fall and spring leaves spring.

[S [S [NP Fall leaves] fall] and [S [NP spring leaves] spring]	NP NP
[S [S [NP Fall leaves] fall] and [S spring [VP leaves spring]]	NP VP
[S [S Fall [VP leaves fall]] and [S [NP spring leaves] spring]	VP NP
[S [S Fall [VP leaves fall]] and [S spring [VP leaves spring]] .	VP VP

If we had c two-ways-ambiguous conjoined sub sentences, we would have 2^c ways of choosing parses for the sub sentences

Parser

Parser is that phase of compiler which take tokens as i/p and with the help of CFG convert it into the corresponding parse tree . Parser also called Syntax analyser.

Types of parser

- Top down Parser
  - TDP with full backtracking Parser
    - Brute force Method
      Try all the possibilities
  - TDP without backtracking Parser
    - Recursive decent parser
    - Non Recursive decent parser
      - LL(1) Parser
- Bottom up parser(Shift Reduce Parse)
  - Operator Precedence Parser(Exception parse)
  - LR Parser
    - LR(0)
    - SLR(1)
    - LALR(1)
    - CLR(1)

LR Parser
L ->Scan string Left to Right
R->Reverse of Right most Derivation(RMD)

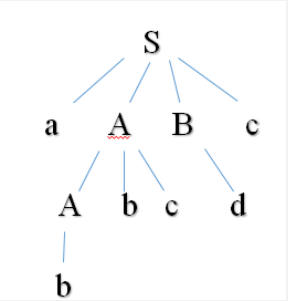
❖ Top down Parser :

- Generates parse tree for the given i/p string with the help of grammar productions by expanding the non-terminals i.e
- It starts from the start symbol and ends on the terminals.
- It uses LMD(left most derivation)

S->aABc
A ->Abc|b
B->d
String -> abbcdc

Top down parsing ->At every point we have to decide what is the next production we should use.

S ->aABc
->aAbcBc
->abbcBc
->abbcdc

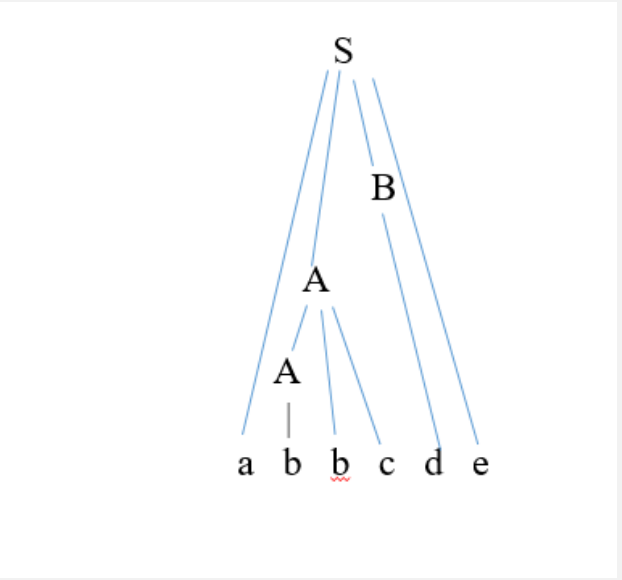


- Bottom up parser :
  - Generates parse tree for the given i/p string with the help of grammar productions by compressing the non-terminals .i.e
  - It starts from the terminals and ends on the start symbol.
  - It uses reverse of Right most derivation

S->aABe
A ->Abc|b
B->d
String -> abbcde

Main decision we have to make is when to **reduce** the given terminal.

S ->aABe
->aAde
->aAbcde
->abbcde



Check Box

Variety of Grammars	Top down parser	Bottom up parser
Ambiguous	x	x
Unambiguous	✓	✓
LR	✓	✓
RR	✓	✓
ND	x	✓
Deterministic	✓	✓

Parsing Algorithm

- we can start with the S symbol and search top down for a tree that has the words as its leaves, or we can start with the words and search bottom up for a tree that culminates in an S.

- Both top-down and bottom-up parsing can be inefficient.

Consider the following two sentences:

Have the students in section 2 of Computer Science 101 take the exam.
Have the students in section 2 of Computer Science 101 taken the exam?

- A left-to-right parsing algorithm would have to guess whether the first word is part of a command or a question .

- If the algorithm guesses wrong, it will have to backtrack all the way to the first word and reanalyse the whole sentence under the other interpretation

Dynamic Programming

- To avoid this source of inefficiency we can use dynamic programming:
- Every time we analyse a substring,
- Store the results so we won't have to reanalyse it later.

- For example, once we discover that
  "the students in section 2 of Computer Science 101" is an NP, we can record that result in a data structure known as a chart.

- Algorithms that do this are called chart parsers.
- There are many types of chart parsers; we describe a bottom-up version called CYK ALGORITHM

CYK Algorithm

- Membership algorithm
- Specific for context free grammar
- Grammar to be expressed in the Chomsky Normal From(CNF).That means CYK algorithm is only applicable on CNF
  - A ->BC or
  - A ->a
- It is Universal (applicable on all CNF)
- Time Complexity O(n^3)
- Space Complexity O(n^2)

CYK Algorithm : Check weather a string "abbb" is a valid member of following CFG

	4	3	2	1
1	1,4	1,1	1,2	1,1
2	2,4	2,3	2,2	
3	3,4	3,3		
4	4,4			

Fig:1

S -> AB
A-> BB/a
B -> AB/b

a b b b
1 2 3 4

	4	3	2	1
1	S,B	A	S,B	A
2	S,B	A	B	
3	A	B		
4	B			

Fig:2

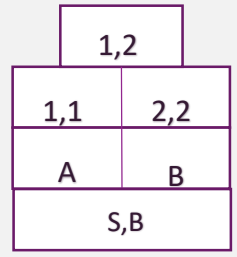


Fig:3

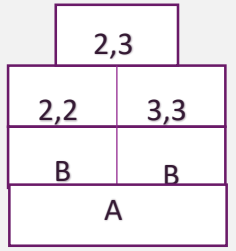


Fig:4

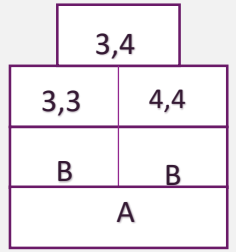


Fig:5

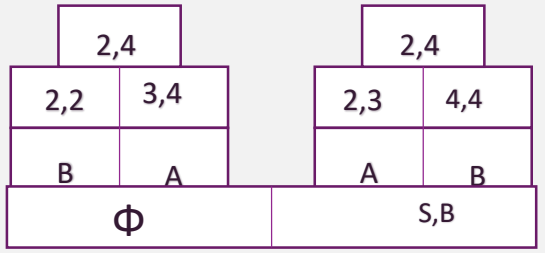


Fig:6

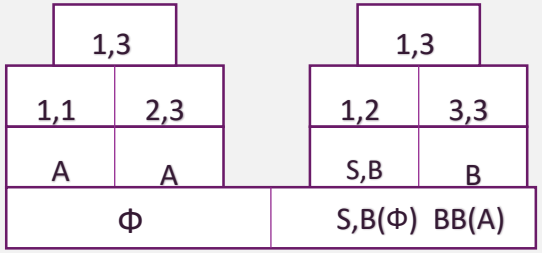


Fig:7

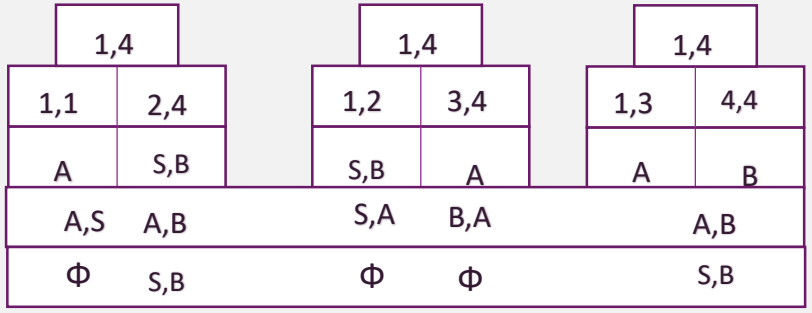


Fig:8

Learning probabilities for PCFGs

- A PCFG has many rules, with a probability for each rule
- Learning the grammar from data might be better than a knowledge engineering approach
- Learning is easiest If we are given a corpus of correctly parsed sentences, commonly called a Treebank
- Penn Treebank is the best known

What if a Treebank is not available?

- It is still possible to learn a grammar without Treebank
- But there will be two problems
- Learning the structure of the grammar rules and learning the probabilities associated with each rule
- The whole computation can be done in a dynamic-programming fashion with an algorithm called the inside–outside algorithm
- The inside–outside algorithm seems magical in that it induces a grammar from unparsed text.

Some Drawbacks of Inside-outside Algorithm

- The parses that are assigned by the induced grammars are often difficult to understand and unsatisfying
- It is slow. where n is the number of words in a sentence and m is the number of grammar categories
- The space of probability assignments is very large

The error rates for automatically learned grammars are still about 50% higher than for hand-constructed grammar, but the gap is decreasing.

Augmented Grammar(Example)

An **augmented grammar** is any grammar whose productions are augmented with **conditions** expressed using features.

If G is a grammar with start symbol S then G', the augmented grammar for G, is the grammar with new start symbol S' and a production S' -> S. The purpose of this new starting production is to indicate to the parser when it should stop parsing and announce acceptance of input.

Let a grammar be S -> AA

A -> aA | b

The augmented grammar for the above grammar will be

S' -> S

S -> AA

A -> aA | b **or** A -> aA

A -> b

Augmented Grammar(LR(0) Parser)

- **LR(0) Parser**
  We need two functions –
  Closure()
  Goto()

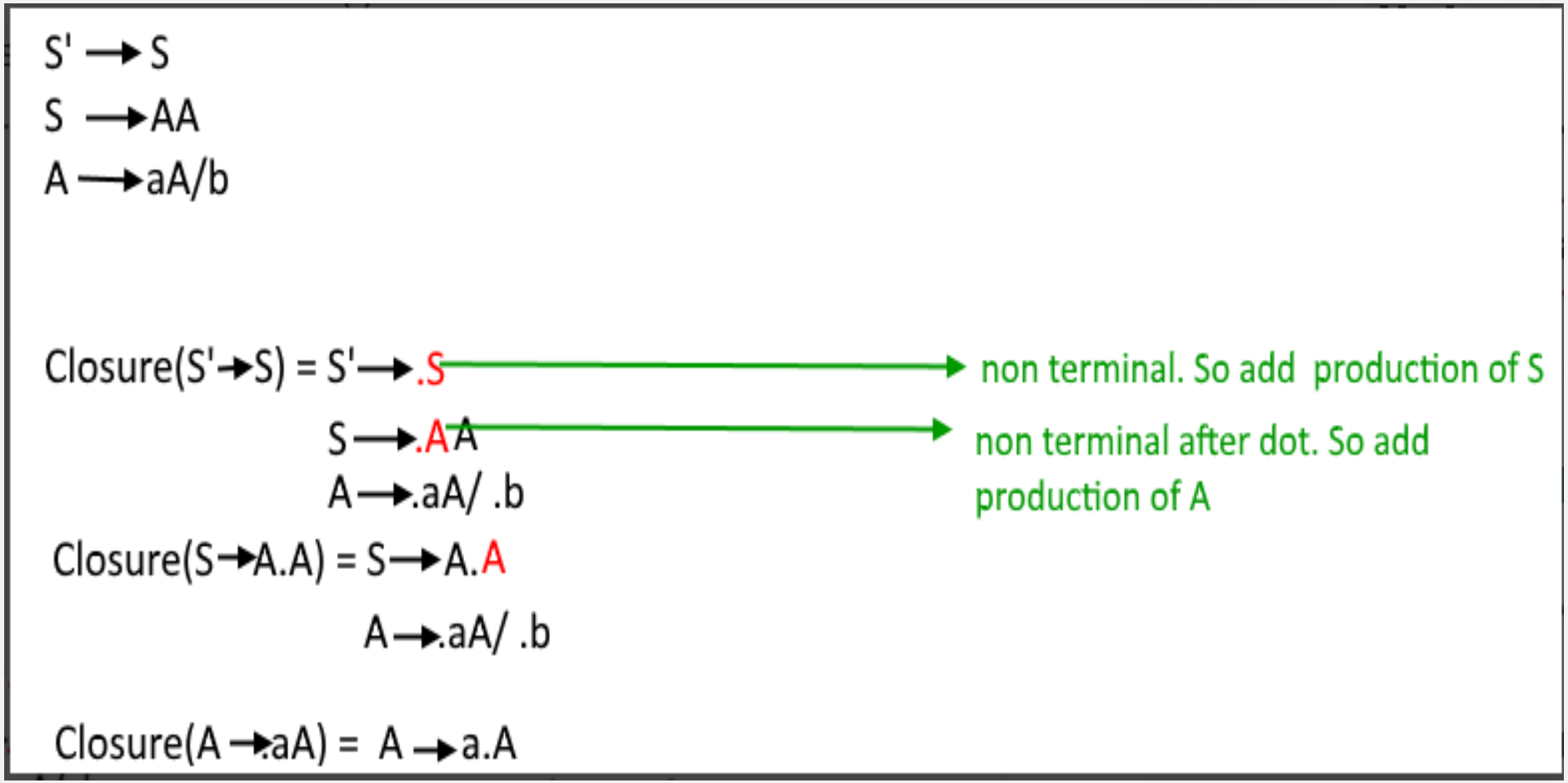
Closure Operation:

If I is a set of items for a grammar G, then closure(I) is the set of items constructed from I by the two rules:

1.Initially every item in I is added to closure(I).

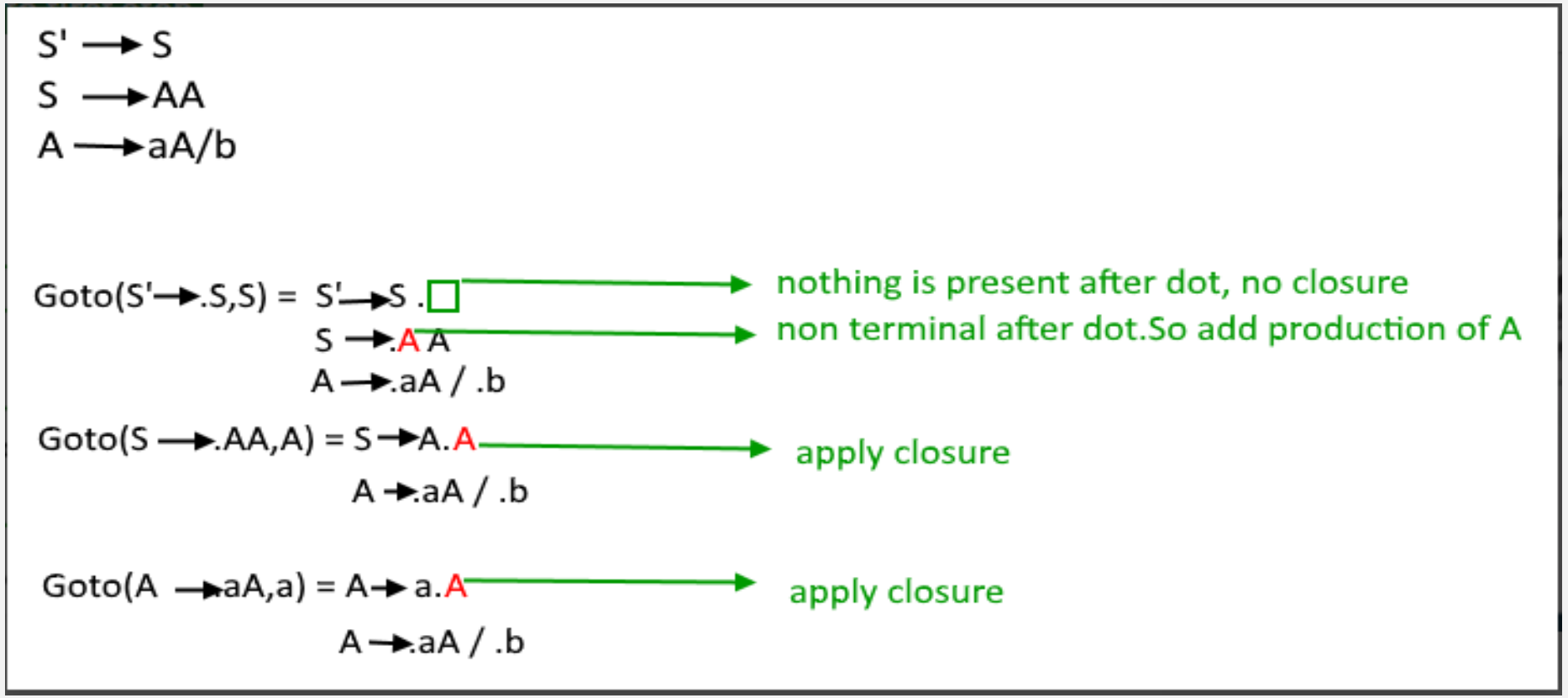
2.If A -> α.Bβ is in closure(I) and B -> γ is a production then add the item B -> .γ to I, If it is not already there. We apply this rule until no more items can be added to closure(I

Augmented Grammar(Closure Operation)

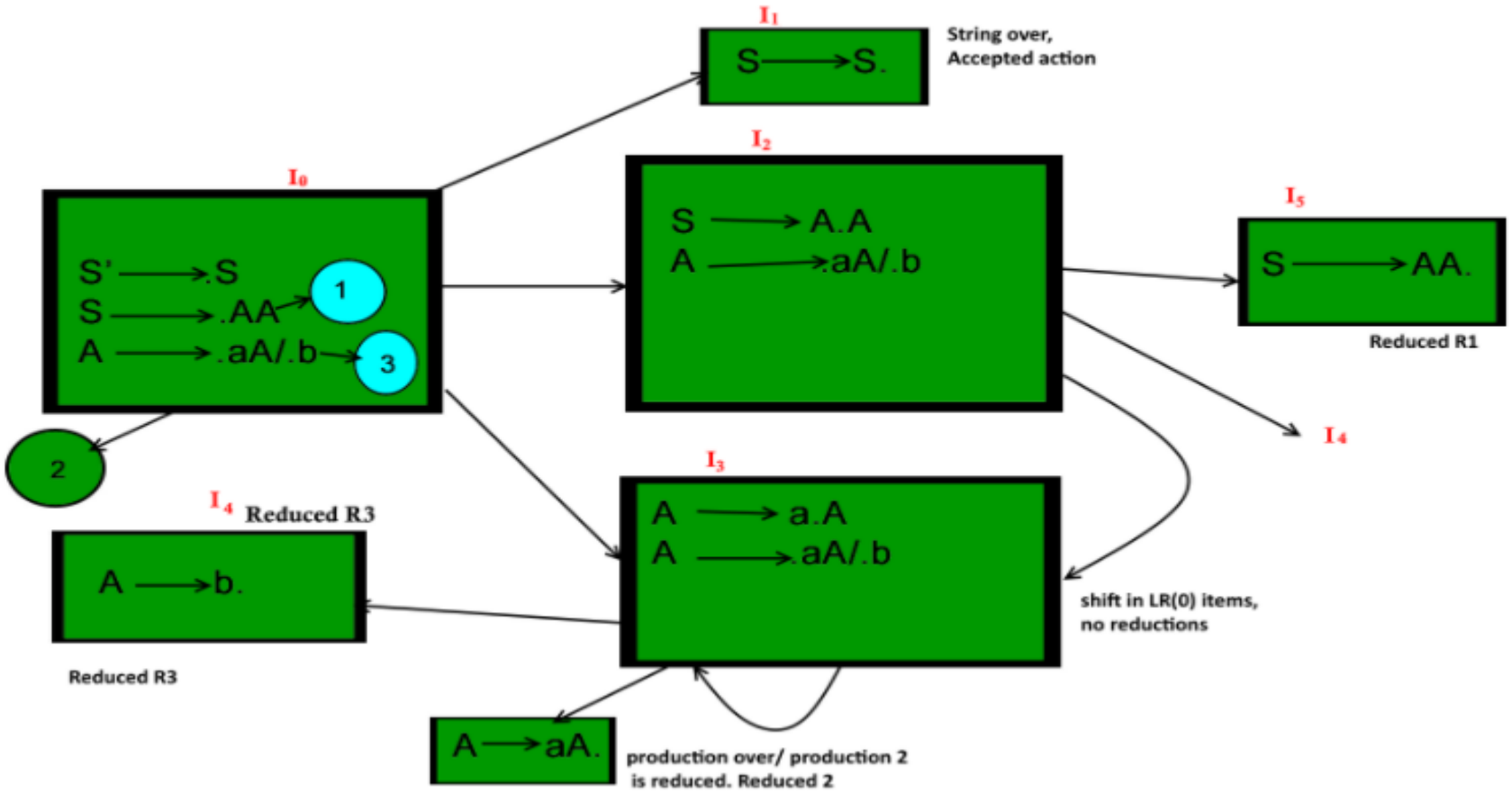


Augmented Grammar(**Goto Operation)**

- **Goto Operation** :
  Goto(I, X) = 1. Add I by moving dot after X.
  2. Apply closure to first step.



Augmented Grammar(**Goto Graph)**



1 Subject–verb Agreement

Subject-verb agreement refers to the relationship between the subject and predicate of the sentence.

2 Case Agreement

Case agreement refers in grammatical case between words in the same construction.

Difference

Subjective case

- Also called the nominative case
- NPs
- Pronouns

Objective case

- Also called the accusative case
- NPo
- Pronouno

Why we should to know Case and Verb aggrement?

The grammar $\mathcal{E}_1$, which handles subjective and objective cases in noun phrases and thus does not over-generate quite as badly as $\mathcal{E}_0$.

Another augmented grammar known $\mathcal{E}_2$, with three augmentations : case agreement, subject-verb agreement and head word. It used for omit $\mathcal{E}_1$.

Examples

$\mathcal{E}_0$ overgenerates,

“Him smells a stench”

$\mathcal{E}_1$ $NP_S\ VP$

“He smells a stench” ⟹ **“I smells a stench”**

$NP(Sbj, pn, h)\ VP(pn, head)$ $\mathcal{E}_2$

“I smell a stench”

“Them Walk through a jungle”

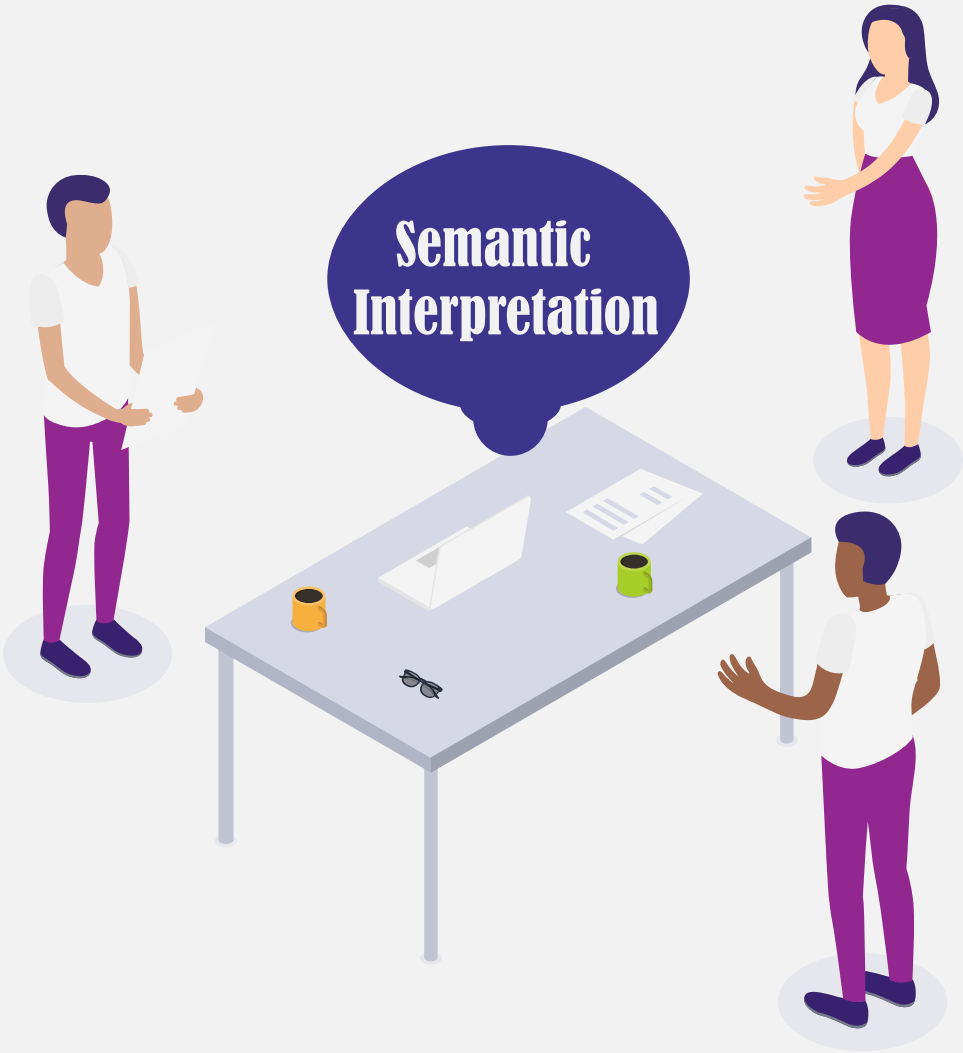
Examples


noun subject
verb
The dog chases the cat.
singular
no "s"
singular
"s"
The dogs chase the cat.
plural
"s"
plural
no "s"


He talks to the girl.
singular
singular
We talk to the girl.
plural
plural

Semantic interpretation

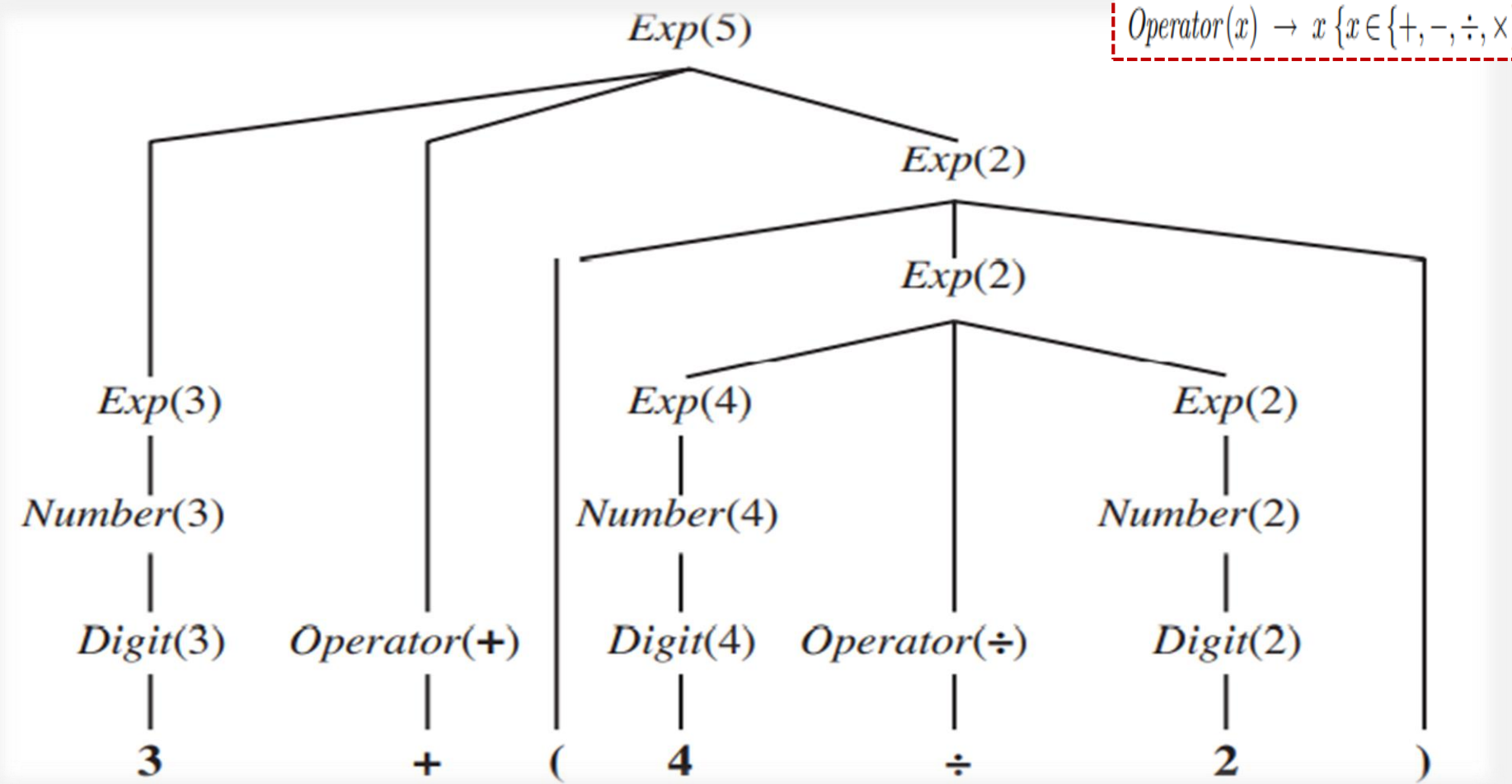
Figure 23.8 A grammar for arithmetic expressions, augmented with semantics. Each variable x_i represents the semantics of a constituent.



Let's Solve a problem?

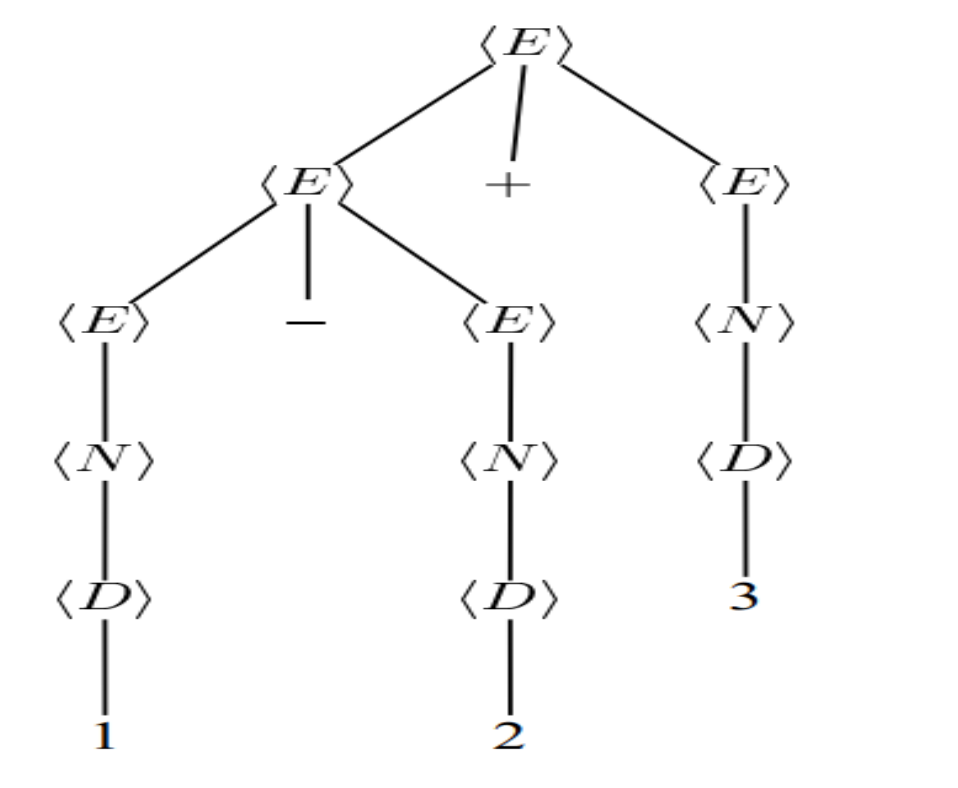
3 + (4 / 2)

$$Exp(x) \rightarrow Exp(x_1)\ Operator(op)\ Exp(x_2)\ \{x = Apply(op, x_1, x_2)\}$$
$$Exp(x) \rightarrow (\ Exp(x)\ )$$
$$Exp(x) \rightarrow Number(x)$$
$$Number(x) \rightarrow Digit(x)$$
$$Number(x) \rightarrow Number(x_1)\ Digit(x_2)\ \{x = 10 \times x_1 + x_2\}$$
$$Digit(x) \rightarrow x\ \{0 \leq x \leq 9\}$$
$$Operator(x) \rightarrow x\ \{x \in \{+, -, \div, \times\}\}$$

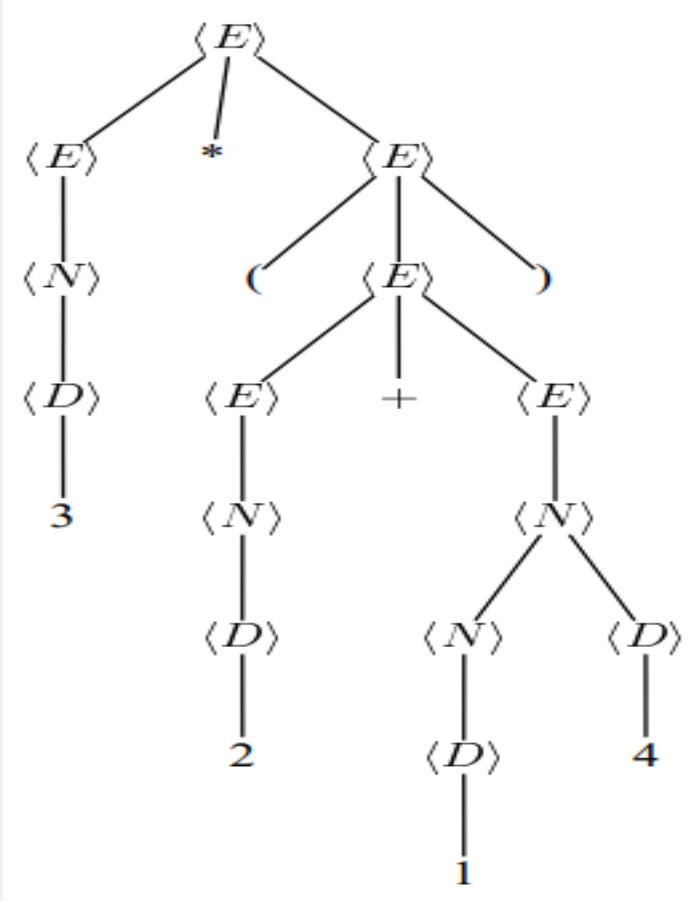


Examples

(1 – 2) + 3



3 * (2 + 14)



Semantic Interpretation

An atomic sentence is formed from a predicate symbol optionally followed by a ATOM parenthesized list of terms, such as –

Brother (Richard, John)

The intended interpretation is "Richard is the brother of John"

Using the λ-notation it can be represents "brother of John" as the predicate-

λx Brother(x, John)

The rule tells us that the semantic interpretation of "Richard is the brother of John" is –

(λx Brothers(x, John)) (Richard)

A λ-expression can also be defined and used as a predicate symbol.

Semantic Interpretation (2)

Let's see another example of sentence "John loves Mary". So its interpreted logical sentence should be -

Loves(John, Mary)

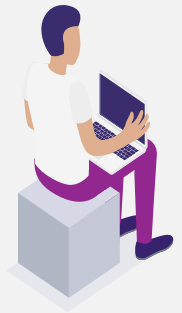
Using the λ-notation it can be represents "brother of John" as the predicate-

λx Loves(x, Mary)

The rule tells us that the semantic interpretation of "John loves Mary" is –

(λx Loves(x, Mary)) (John)

The verb "loves" is represented as λy λx Loves(x, y), the predicate that, when given the argument Mary, returns the predicate λx Loves(x, Mary).



CHILL


By "Zelle and Mooney", 1996 is an inductive logic program that learns a grammar and a specialized parser for that grammar
The target domain is natural language database queries
The training examples consist of pairs of word strings and corresponding semantic forms

Examples

∀ x King(x) ∧ Person(x)

would be equivalent to asserting

- Richard the Lionheart is a king ∧ Richard the Lionheart is a person.
- King John is a king ∧ King John is a person.
- Richard's left leg is a king ∧ Richard's left leg is a person.

Examples

What is the capital of the state with the largest population?

Answer: (c, Capital(s, c) ∧ Largest(p, State(s) ∧ Population(s, p)))

CHILL's task is to learn a predicate Parse(words , semantics).
CHILL can learn to achieve 70% to 85% accuracy on various database query tasks.

Complications

The grammar of real English is extremely complex in some sense which creates complications.

- Time and Tense
- Quantification
- Pragmatics
- Ambiguity

Time & Tense

- Difference between “John loves Mary” and “John loved Mary”
- We use typical tense (present, past , future) for indicating the time of events.
- One good way to represent the time of event is the event calculus notation:

John loves mary: E1 ∈ Loves(John, Mary) ∧ During(Now,Extent(E1))

John loved mary: E2 ∈ Loves(John, Mary) ∧ After (Now,Extent(E2))

.

Time & Tense

$S(pred(obj)) \rightarrow NP(obj)\ VP(pred)$
$VP(pred(obj)) \rightarrow Verb(pred)\ NP(obj)$
$NP(obj) \rightarrow Name(obj)$

$Name(John) \rightarrow$ **John**
$Name(Mary) \rightarrow$ **Mary**
$Verb(\lambda y\ \lambda x\ Loves(x, y)) \rightarrow$ **loves**

Figure 23.10 A grammar that can derive a parse tree and semantic interpretation for "John loves Mary" (and three other sentences). Each category is augmented with a single argument representing the semantics.

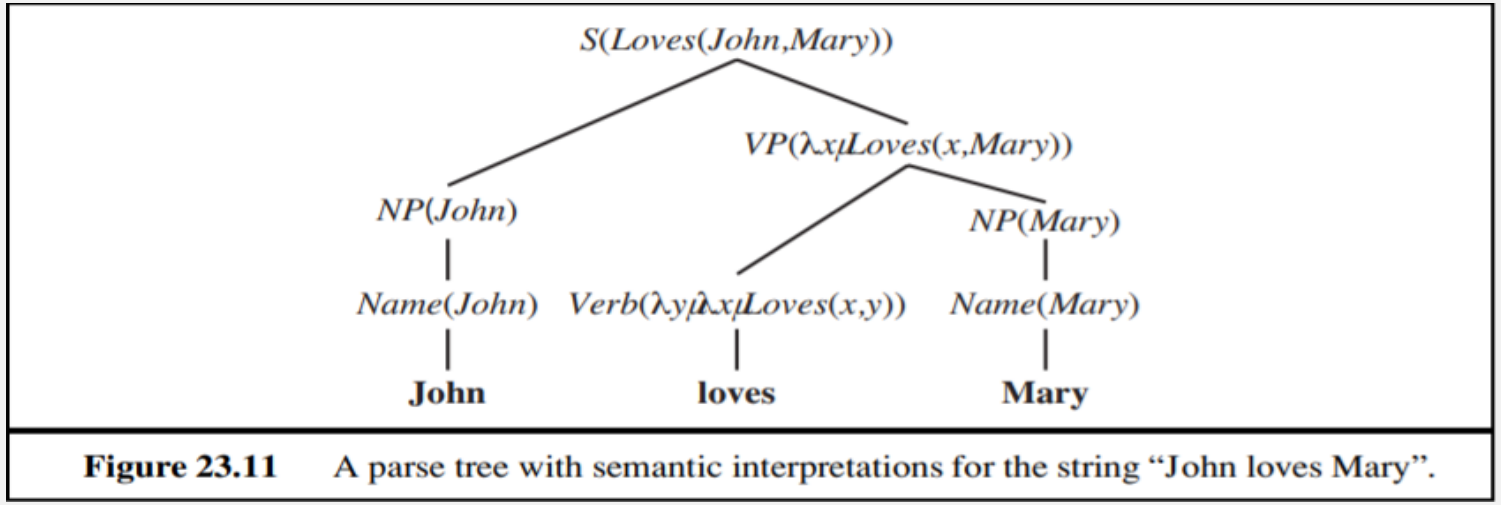


Figure 23.11 A parse tree with semantic interpretations for the string "John loves Mary".

Time & Tense

Distinction between process and discrete events. A grammar would reflect that fact by having a low probability for adding the adverbial phrase “a lot” to discrete events.

- “John Slept a lot last night” [process]
- “John found a unicorn a lot last night” [discrete]

Quantification

- “Every Agent feels a breeze” the sentence has only one syntactic parse but it is actually semantically ambiguous.
- Preferred meaning “For every agent there exists a breeze that the agent feels”. But an acceptable alternative is “There exists a breeze that every agent feels”

Quantification

These two alternatives can be represented as:

$$\forall\, a\ a \in \text{Agents} \Rightarrow \exists\, b\ b \in \text{Breezes} \wedge \exists\, e\ e \in \text{Feel}(a, b) \wedge \text{During}(\text{Now}, e)\ ;$$

$$\exists\, b\ b \in \text{Breezes}\ \forall\, a\ a \in \text{Agents} \Rightarrow \exists\, e\ e \in \text{Feel}(a, b) \wedge \text{During}(\text{Now}, e)\ .$$

So, the standard approach to quantification is defining a **quasi-logical form** not an **actual logical form**. The quasi-logical form then turned into a logical sentence.

Pragmatics

Problem of completing the interpretations by adding context-dependent information about the current situation

Two part of pragmatics:

* Indexicals: Phrases that refers directly to the current situations. “I am in Boston today” here both “T” and “today” are indexicals.

* Speech-act: The speaker’s action is considered the speech act. The hearer deciphers what type of action it is – *a question a statement, a promise, a warning, a command, and so on*.

Ambiguity

- In some cases, hearers are consciously aware of ambiguity in an utterance. Like the following:
  - Squad helps dog bite victim
  - Police begin campaign to run down jaywalkers
  - Helicopter powered by human flies
  - Teacher strikes idle kids
  - Hospitals are sued by 7 foot doctors

Ambiguity

- Language analyzing using computers in 1960's, researchers were quite surprised to learn that almost every utterance is highly ambiguous. A system with a large grammar and lexicon might find thousands od interpretations.
- Lexical Ambiguity: A word has more than one meaning which is pretty normal. "back" can be an adverb (go back) ,an adjective (back door), a noun (the back of the room) or a verb (back up your files).
- Syntactic Ambiguity: Refers to a phrase that has multiple parses. "I smelled a wumpus in 2,2" has two parses: one where the prepositional phrase "in 2,2" modifies the noun and one where it modifies the verb. The syntactic ambiguity leads to a semantic ambiguity, because one parse means that the wumpus is in 2,2 and the other means SEMANTIC AMBIGUITY that a stench is in 2,2. In this case, getting the wrong interpretation could be a deadly mistake for the agent.

Ambiguity

- Metonymy: Metonymy is a figure of speech in which one object is used to stand for another. When we hear “Chrysler announced a new model,” we do not interpret it as the companies can talk rather we understand that a spokesperson representing the company made the announcement.
- Metaphor: Metaphor is another figure of speech in which a phrase with one literal meaning is used to suggest a different meaning by way of an analogy. Thus, metaphor can be seen as a kind of metonymy where the relation is one of similarity.

Disambiguation

Disambiguation is the process of recovering the most probable intended meaning od an utterance. We already have the probability approach where the most probable outcome led to the interpretation. But probability failed to provide specific knowledge rather than general knowledge

We need to combine four models for doing disambiguation properly:

1. The World Model: The likelihood that a proposition occurs in the world. Given what we know about the world, it is more likely that a speaker who says “I’m dead” means “I am in big trouble” rather than “My life ended, and yet I can still talk.”

Disambiguation

2. The Mental Model: It's basically depends on the believes of speaker and hearer. For example, when a politician says, "I am not a crook," the world model might assign a probability of only 50% to the proposition that the politician is not a criminal, and 99.999% to the proposition that he is not a hooked shepherd's staff. Nevertheless, we select the former interpretation because it is a more likely thing to say.

Disambiguation

3. The Language Model: The likelihood that a certain string of words will be chosen, given that the speaker has the intention of communicating a certain fact.

4. The Acoustic Model: for spoken communication, the likelihood that a particular sequence of sounds will be generated, given that the speaker has chosen a given string of words.

THANK YOU!!