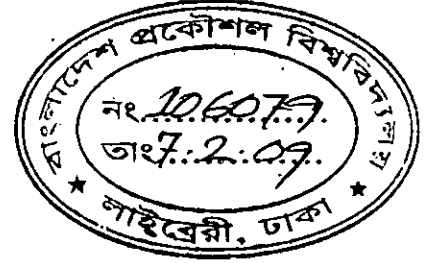


**Design of a Non-ambiguous Predictive Parser for Bangla Natural Language
Sentence with Error Recovery Capability**

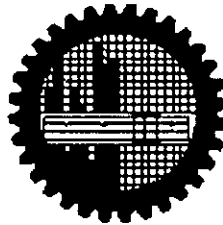
by

Fazle Elahi Faisal
Roll No: 040505037P



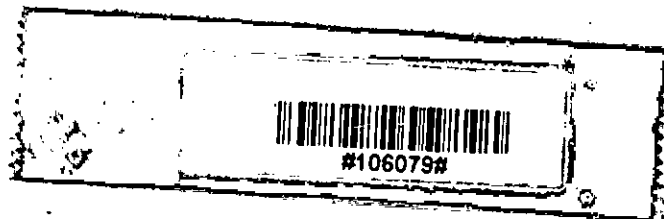
A thesis submitted for the partial fulfillment of the requirement for the degree of
Master of Science in Computer Science and Engineering (M. Sc. Engg.)

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



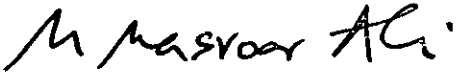
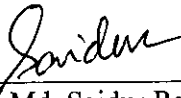
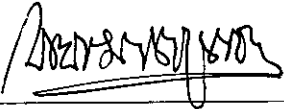
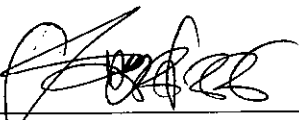

Department of Computer Science and Engineering (CSE)
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
(BUET)

Dhaka – 1000, Bangladesh
December, 2008



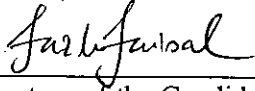
The thesis titled "Design of a Non-ambiguous Predictive Parser for Bangla Natural Language Sentence with Error Recovery Capability", submitted by Fazole Elahi Faisal, Roll No: 040505037P, Session: April 2005, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering (M. Sc. Engg.) on December 31, 2008.

BOARD OF EXAMINERS

1. 
Dr. Muhammad Masroor Ali
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka – 1000, Bangladesh
Chairman
(Supervisor)
2. 
Dr. Md. Saidur Rahman
Professor and Head
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka – 1000, Bangladesh
Member
(Ex-officio)
3. 
Dr. M. Kaykobad
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka – 1000, Bangladesh
Member
4. 
Dr. Md. Humayun Kabir
Associate Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka – 1000, Bangladesh
Member
5. 
Dr. Mohammad Zahidur Rahman
Associate Professor
Jahangirnagar University
Savar, Dhaka, Bangladesh
Member
(External)

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.



Signature of the Candidate

Fazle Elahi Faisal

Roll No: 040505037P

TO MY PARENTS

Table of Contents

1	Introduction	1
1.1	Machine Translation	1
1.2	MT Process	2
1.3	MT Techniques	2
1.3.1	Rule-based Machine Translation	3
1.3.1.1	Transfer-based Machine Translation	3
1.3.1.2	Inerlingual Machine Translation	4
1.3.1.3	Dictionary-based Machine Translation	5
1.3.2	Statistical Machine Translation	5
1.3.2.1	Word-based Machine Translation	6
1.3.2.2	Phrase-based Machine Translation	6
1.3.2.3	Syntax-based Machine Translation	7
1.3.3	Example-based Machine Translation	8
1.4	Word Sense Disambiguation	9
1.5	Applications of Machine Translation	9
1.6	Parsing	10
1.7	Machine Translation in Bangla	12
1.7.1	Prospective Applications of Bangla Translation	13
1.7.1.1	Social or Political Application	14
1.7.1.2	Commercial Application	14
1.7.1.3	Scientific Application	15
1.7.1.4	Philosophical Application	15
1.7.2	Bangla Parsing	17
1.7.3	Limitations of Bangla Parsing	18
1.7.4	Present State of Bangla Parsing and Contributions of the Thesis	19
1.8	Organization of the Thesis	21
2	Syntax-based Machine Translation	24
2.1	Syntax-based Machine Translation	24
2.1.1	Lexical Analysis	26
2.1.1.1	Scanner	26
2.1.1.2	Tokenizer	27
2.1.2	Syntax Analysis	27
2.1.3	Translation	27
2.2	Parser	27
2.2.1	Role of a Parser in a Machine Translator	29
2.3	Error Recovery in a Parser	30
2.3.1	Panic-mode Recovery	30
2.3.2	Phrase Level Recovery	31
2.3.3	Error Productions	31
2.3.4	Global Correction	31
2.4	Context-free Grammar	32
2.4.1	Terminal	33
2.4.2	Non-terminal	33

2.4.3 Start Symbol	34
2.4.4 Productions	34
2.4.5 Normal Forms	34
2.4.6 Undecidable Problems	34
2.4.7 Extensions	35
2.4.8 Linguistic Applications	36
2.4.9 Parse Tree	36
2.4.10 Derivations	37
2.4.11 Ambiguous Grammar	38
2.4.11.1 Elimination of Ambiguity	39
2.4.12 Left Recursion	40
2.4.12.1 Elimination of Left Recursion	42
2.4.13 Left Factoring	43
2.5 Context-sensitive Grammar	44
2.6 Top-down Parsing	46
2.6.1 LL Parser	48
2.6.2 Recursive Descent Parser	49
2.7 Bottom-up Parsing	50
2.7.1 LR Parser	51
2.7.2 LALR Parser	52
2.7.3 Shift-reduce Parser	52
2.8 Architecture of Non-recursive Predictive Parser	53
2.8.1 First	55
2.8.2 Follow	56
2.8.3 Construction of Predictive Parsing Tables	57
2.8.4 Error Recovery in Predictive Parsing	58
2.8.5 Remarks	59
3 Bangla Grammar	61
3.1 Bangla Grammar	61
3.2 Phrases in Bangla Grammar	61
3.2.1 Noun Phrase	61
3.2.2 Verb Phrase	62
3.2.3 Adjective Phrase	62
3.3 Context-free Grammar for Noun Phrase	62
3.4 Context-free Grammar for Verb Phrase	67
3.5 Context-free Grammar for Adjective Phrase	68
3.6 Context-free Grammar for Simple Sentence	69
3.7 Context-free Grammar for Complex Sentence	70
3.8 Context-free Grammar for Compound Sentence	75
3.9 Context-free Grammar for Bangla Sentence	78
3.10 Limitations of Present Context-free Grammar for Bangla Language	79
3.10.1 Ambiguity	79
3.10.2 Non-predictive	80
3.10.3 Lacking Error Recoverability	81
3.10.4 Unable to Handle Non-dictionary Word	81
3.10.5 Limited Use of Conjunctions	82

3.10.6 Unable to Handle Numeric Words	82
4 Non-ambiguous Grammar for Simple Sentence	83
4.1 Non-ambiguous Grammar for Noun Phrase	83
4.1.1 Ambiguity Elimination	84
4.1.2 Addition of Conjunctives	88
4.1.3 Unknown Word Handling	93
4.1.4 Left Factoring	97
4.2 Non-ambiguous Grammar for Adjective Phrase	101
4.2.1 Numeric Word Handling	103
4.3 Non-ambiguous Grammar for Verb Phrase	104
4.4 Non-ambiguous Grammar for Simple Sentence	107
4.5 Remarks	113
5 Non-ambiguous Grammar for Complex Sentence	114
5.1 Existing Grammar for Complex Sentence	114
5.2 Identifying Complex Sentence Patterns	115
5.3 Non-ambiguous Grammar for Complex Sentence	117
5.4 Remarks	133
6 Non-ambiguous Grammar for Compound Sentence	134
6.1 Existing Grammar for Compound Sentence	134
6.2 Enhancement of Grammar for Compound Sentence	135
6.3 Non-ambiguous Predictive Grammar for Compound Sentence	138
6.4 Remarks	138
7 Non-ambiguous Comprehensive Grammar	140
7.1 Existing Grammar for Bangla Sentence	140
7.2 Non-ambiguous Grammar for Bangla Sentence	140
7.3 Comprehensive Bangla Grammar	143
7.4 Significance of Each Non-terminal of Proposed Bangla Grammar	150
7.4.1 Significance of Non-terminals of Sentence Level	150
7.4.2 Significance of Non-terminals of Complex Sentence Level	152
7.4.3 Significance of Non-terminals of Simple Sentence Level	156
7.4.4 Significance of Non-terminals of Verb Phrase Level	157
7.4.5 Significance of Non-terminals of Noun Phrase Level	158
7.4.6 Significance of Non-terminals of Adjective Phrase Level	161
7.5 Remarks	162
8 Parsing Technique	163
8.1 Predictive Parser Architecture	163
8.2 Lexicon	164
8.3 Parsing Table	165
8.3.1 Calculation of First	166
8.3.2 Calculation of Follow	167
8.3.3 Building of Parsing Table	169

8.4 Lexical Analyzer	176
8.5 Syntax Analyzer	180
8.6 Parsing of Correct Sentence	180
8.7 Error Recovery Policy	182
8.8 Parsing of Erroneous Sentence	185
8.9 Remarks	189
9 Simulation	190
9.1 Overview	190
9.2 The Lexicon	190
9.3 Simulation Program	191
9.3.1 Initialization	192
9.3.2 Lexical Analyzer	192
9.3.3 Syntax Analyzer	192
9.4 Input of the Program	193
9.5 Output of the Program	194
9.6 Error Recovery	195
9.7 Remarks	198
10 Conclusion	199
10.1 Achievements of This Thesis	199
10.2 Future Works	202
A Bangla Unicode	203
Bibliography	205

List of Figures

1.1	Work flow of a standard translator.	2
1.2	Demonstration of the languages which are used in the process of translating using a bridge language.	4
1.3	Parse tree of the sentence "I eat rice", which is generated during syntax-based machine translation.	8
2.1	Parse tree of the sentence "I eat rice", which is generated during syntax-based machine translation.	25
2.2	Interaction of lexical analyzer with parser.	26
2.3	Parse tree of the sentence "John hit the ball".	37
2.4	Two ways of parse tree (a), (b) of the sentence segment "extremely very good".	38
2.5	Parse tree of the sentence segment "extremely very good" using left associative rule of new grammar.	40
2.6	Parse tree of the sentence segment "extremely very good" using right associative rule of new grammar.	40
2.7	A model of non-recursive predictive parser.	54
3.1	Tree structure for the rules $NP \rightarrow N$ (a), $NP \rightarrow N \text{ DET}$ (b).	63
3.2	Tree structure for the rules $NP \rightarrow NP \text{ Biv}$ (a) (b), $NP \rightarrow NP \text{ Biv NP}$ (c).	64
3.3	Tree structure for the rules $NP \rightarrow N \text{ PM}$ (a) (b) (c) (d).	64
3.4	Tree structure for the rules $SPR \rightarrow QFR \text{ PP}$ (a), $SPR \rightarrow QFR$ (b).	65
3.5	Tree structure for the rule $NP \rightarrow (\text{DEMO}) (\text{SPR}) (\text{AP}) \text{ NP}$, which cover the variations $NP \rightarrow \text{DEMO NP}$ (a), $NP \rightarrow \text{SPR NP}$ (b), $NP \rightarrow \text{AP NP}$ (c), $NP \rightarrow \text{DEMO SPR NP}$ (d), $NP \rightarrow \text{DEMO AP NP}$ (e), $NP \rightarrow \text{SPR AP NP}$ (f), $NP \rightarrow \text{DEMO SPR AP NP}$ (g).	66
3.6	Tree structure for the rule $VP \rightarrow (\text{NP}) (\text{AP}) \text{ VF}$, which cover the variations $VP \rightarrow \text{VF}$ (a), $VP \rightarrow \text{NP VF}$ (b), $VP \rightarrow \text{AP VF}$ (c), $VP \rightarrow \text{NP AP VF}$ (d).	68
3.7	Tree structure for the rule $AP \rightarrow \text{AD} / \text{ADs}$.	69
3.8	Tree structure for the rule $SS \rightarrow \text{NP VP}$.	70
3.9	Tree structure for the rules of complex sentences in different variations.	72
3.10	Tree structure for the rule $\text{COMS} \rightarrow \text{SS Conj SS}$.	76
3.11	Tree structure for the rule $\text{COMS} \rightarrow \text{SS Conj CS}$.	76
3.12	Tree structure for the rule $\text{COMS} \rightarrow \text{CS Conj SS}$.	77
3.13	Tree structure for the rule $\text{COMS} \rightarrow \text{CS Conj CS}$.	78
3.14	Ambiguity in the rule $NP \rightarrow NP \text{ Biv NP}$.	80
4.1	Ambiguity in the rule $NP \rightarrow NP \text{ Biv (NP)}$.	84
4.2	Ambiguity in the rule $NP \rightarrow NP \text{ Biv (NP)}$ (another example).	84
4.3	Tree derivation using the rules for PRE and NW.	86
4.4	Tree derivation using non-ambiguous rule for NP.	86
4.5	Tree derivation using the rule $NP \rightarrow NP \text{ Conj NP}$.	89
4.6	Tree derivation using the rule $NP \rightarrow NP \text{ Conj NP}$, where three smaller noun phrases are connected by conjunctives.	90

4.7	Ambiguity using the rule $NP \rightarrow NP \text{ Conj } NP$, where one is left associative (a), and another is right associative (b).	91
4.8	Elimination of ambiguity of the rule $NP \rightarrow NP \text{ Conj } NP$ using new rule $NP \rightarrow NPU \mid NPU \text{ Conj } NP$.	92
4.9	Tree derivation with grammar for noun phrase with unknown word (non-dictionary) handling.	95
4.10	Tree derivation with grammar for noun phrase with both single and multi-word unknown word (non-dictionary) handling.	96
4.11	Tree derivation with non-ambiguous grammar for noun phrase with predictive parsing.	100
4.12	Tree derivation with non-ambiguous grammar for adjective phrase.	102
4.13	Tree derivation with non-ambiguous grammar for adjective phrase with predictive parsing.	103
4.14	Tree derivation for a numeric word.	103
4.15	Tree derivation with non-ambiguous grammar for verb phrase with predictive parsing.	105
4.16	Ambiguity in the rule $SS \rightarrow NP \text{ VP}$.	107
4.17	Tree derivation with non-ambiguous grammar for simple sentence with predictive parsing.	111
5.1	Tree derivation for the sentence “আমি গেলে তুমি এসো” (ami gele tumi esO) using non-ambiguous grammar with predictive parsing.	125
5.2	Tree derivation for the sentence “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO) using non-ambiguous grammar with predictive parsing.	126
5.3	Tree derivation for the sentence “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO) using non-ambiguous grammar with predictive parsing.	126
5.4	Tree derivation for the sentence “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO) using non-ambiguous grammar with predictive parsing.	127
5.5	Tree derivation for the sentence “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO) using non-ambiguous grammar predictive parsing.	127
5.6	Tree derivation for the sentence “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO) using non-ambiguous grammar predictive parsing.	128
5.7	Tree derivation for the sentence “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO) using non-ambiguous grammar predictive parsing.	128
5.8	Tree derivation for the sentence “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO) using non-ambiguous grammar predictive parsing.	129
5.9	Tree derivation for the sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO) using non-ambiguous grammar predictive parsing.	129
5.10	Tree derivation for the sentence “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai) using non-ambiguous grammar predictive parsing.	130
5.11	Tree derivation for the sentence “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai) using non-ambiguous grammar predictive parsing.	130
5.12	Tree derivation for the sentence “তুমি তবে এসো আমি যদি যাই” (tumi tobe esO ami zodi zai) using non-ambiguous grammar predictive parsing.	131
5.13	Tree derivation for the sentence “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai) using non-ambiguous grammar predictive parsing.	131
5.14	Tree derivation for the sentence “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai) using non-ambiguous grammar predictive parsing.	132

5.15	Tree derivation for the sentence “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai) using non-ambiguous grammar predictive parsing.	132
6.1	Tree derivation for the compound sentence “আমি ঢাকা যাব এবং তুমি সিলেট যাবে” (ami Dhaka zabo ebong tumi sileT zabe) using existing grammar for compound sentence.	135
6.2	Tree derivation for the compound sentence “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahl zabe) using new grammar for compound sentence.	136
6.3	Ambiguity in the new rule COMS → S Conj S.	137
6.4	Tree derivation using non-ambiguous grammar for compound sentence.	138
7.1	Tree derivation for the simple sentence “আমি বেশী ভাত খাই” (ami beshI vat khai).	145
7.2	Tree derivation for the simple sentence “রবিন ও আমি ভাত ও পেপসি খাই” (robin O ami vat O peposi khai).	146
7.3	Tree derivation for the simple sentence “ঐ ছেলেটি খেলে” (OI chheleTi khele).	147
7.4	Tree derivation for the complex sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO).	147
7.5	Tree derivation for the complex sentence “তবে রবিন আসবে যদি জাকির যায়” (tobe robin ashobe zodi jakir zay).	148
7.6	Tree derivation for the compound sentence “আমি ঢাকা যাব অথবা তুমি ঢাকা যাবে” (ami Dhaka zabo othoba tumi Dhaka zabe).	148
7.7	Tree derivation for the compound sentence “আমি যাব অথবা তুমি যাবে অথবা সে যাবে” (ami zabo othoba tumi zabe othoba se zabe).	149
8.1	Architecture of a machine translator using predictive parser.	163
8.2	Work flow of lexical analyzer of predictive parser.	179
8.3	Tree derivation of erroneous sentence “আমি ঢাকা যা” (ami Dhaka za).	186
8.4	Tree derivation of erroneous sentence “ঐ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).	188
9.1	Lexicon used for simulation program.	190
9.2	Grammar used for simulation program.	191
9.3	Input file used for simulation program.	193
9.4	Output file showing parsed text by simulation program.	194
9.5	Output of display program using Java showing tree representation of parsed text.	195
9.6	Output productions for the sentence “আমি ঢাকা যা” (ami Dhaka za).	196
9.7	Tree representation for the sentence “আমি ঢাকা যা” (ami Dhaka za).	196
9.8	Output productions for the sentence “ঐ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).	197
9.9	Tree representation for the sentence “ঐ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).	198

List of Tables

2.1	Parsing table of a predictive parser.	58
2.2	Parsing table of a predictive parser with error recovery information.	59
8.1	Lexicon of Bangla language for a predictive parser.	164
8.2	Predictive parsing table for sentence portion.	170
8.3	Predictive parsing table for complex sentence portion.	171
8.4	Predictive parsing table for simple sentence portion.	173
8.5	Predictive parsing table for verb phrase portion.	174
8.6	Predictive parsing table for noun phrase portion.	174
8.7	Predictive parsing table for adjective phrase portion.	176
8.8	A sample lexicon.	178
A.1	Code in UNICODE for Bangla characters.	203

List of Abbreviations

Word	Abbreviation
AD	Adjective
AP	Adjective Phrase
AUX	Auxiliary
Biv	Bivokti, Non-extensive Bivokti
BivE	Extensive Bivokti
BS	Basic Sentence
CFG	Context-Free Grammar
COMS	Compound Sentence
Conj	Conjunctive
CSG	Context-sensitive Grammar
CS	Complex Sentence
DC	Dependent Clause
DD	Demonstrative Deictic
DEMO	Demonstrator
DO	Demonstrative Ordial
EBMT	Example-based Machine Translation
IC	Independent Clause
MT	Machine Translation
N	Noun
NP	Noun Phrase
NPU	Noun Phrase Unit
NW	Noun Word
P	Pronoun
PCFG	Probabilistic Context-free Grammar
PM	Plural Marker
PP	Post Preposition
PRE	Previous Portion of Noun Phrase
QFR	Quantifier
S	Sentence

SMT	Statistical Machine Translation
SPR	Specifier
SS	Simple Sentence
SUBORD	Subordinator
SUBCOM	Subordinator Complement
VF	Verb Form
VP	Verb Phrase
VR	Verb Root
UN	Unknown Word
UNG	Unknown Word Group

List of Symbols

Symbol	Elaboration
AD	Adjective
AP	Adjective Phrase
AUX	Auxiliary
Biv	Bivokti, Non-extensive Bivokti
BivE	Extensive Bivokti
BS	Basic Sentence
COMS	Compound Sentence
Conj	Conjunctive
CS	Complex Sentence
DC	Dependent Clause
DD	Demonstrative Deictic
DEMO	Demonstrator
DO	Demonstrative Ordial
IC	Independent Clause
N	Noun
NP	Noun Phrase
NPU	Noun Phrase Unit
NW	Noun Word
P	Pronoun

PM	Plural Marker
PP	Post Preposition
PRE	Previous Portion of Noun Phrase
QFR	Quantifier
S	Sentence
SPR	Specifier
SS	Simple Sentence
SUBORD	Subordinator
SUBCOM	Subordinator Complement
VF	Verb Form
VP	Verb Phrase
VR	Verb Root
UN	Unknown Word
UNG	Unknown Word Group
\Rightarrow	The expression $A \Rightarrow B$ indicates that B can be immediately derived from A
\Rightarrow^+	The expression $A \Rightarrow^+ B$ indicates that B can be derived from A after one or more steps of derivation
ε	Empty string
A^*	Several occurrences of A together, including ε
A^+	Several occurrences of A together, excluding ε

Acknowledgements

I am grateful to number of people, who supported me a lot towards successful completion of my thesis work and consequently acquiring a great achievement for me. I like to express unbounded gratitude to my thesis supervisor, Dr. Muhammad Masroor Ali, Professor and former Head of Department of Computer Science and Engineering (CSE), BUET. He supported me a lot with his depth of knowledge, lots of inspiration and encouragement. His knowledge and experience in the field of parsing and machine translation helped me to overcome many hindrances during my work. It was impossible for me to finish my work without his crucial ideas.

I like to express special gratitude to Dr. Md. Saidur Rahman, Head, Department of Computer Science and Engineering (CSE), BUET, for his inspiration and encouragement towards completion of my work. I am very grateful to Institute of Information and Communication Technology (IICT), my own institute, providing me enough technical support with software and internet facility. I am also grateful to Directorate of Advisory, Extension and Research Services (DAERS) for providing me necessary financial support.

And, of course my special gratitude goes to my wife, Farzana Sharmin. It was really impossible for me to complete my work without her mental support, inspiration and encouragement. She really sacrificed a lot for my work and always supported me weakening family burden from my mind.

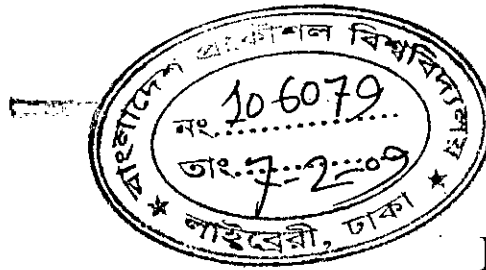
Abstract

Machine translation from Bangla to other languages is a promising field, but there are limited works in this area. Syntax-based machine translation is a suitable technique for machine translation system from Bangla to other languages, as Bangla grammar is nicely structured. This translation technique has two stages – parsing and generation. Parsing is the main challenge of a syntax-based machine translator. This thesis includes design of non-ambiguous predictive Bangla grammar, which is used to develop predictive parser with error recovery capability.

Analyzing previous works on Bangla grammar, it can be summarized that, previously designed grammars were non-comprehensive and ambiguous. Because of ambiguity, it does not fall into the category of LL(1) grammar. Predictive parser can not be developed without LL(1) grammar. Non-predictive parser uses backtracking technique, which takes exponential runtime. This is quite impractical for machine translation system, which generally deals with a large amount of data. Error recovery technique was never introduced in Bangla parsing technique. Unlike compiler, grammar of a natural language reflects only common patterns of sentences. To design a grammar to reflect all patterns of sentences, cause to grow the complexity of the grammar exponentially, because a single sentence can be written in different ways correctly. Without error recovery feature, parsing process stops when an error is detected in input sentence. Lack of error recoverability is a big hindrance to develop successful Bangla parser. Moreover, handling of non-dictionary words is a big challenge, which was not solved previously.

In this thesis, ambiguity is eliminated from previous grammar. Therefore, non-ambiguous predictive grammar is designed. Additionally, this grammar includes a nice mechanism to handle non-dictionary words. The grammar has been enhanced including some common patterns of sentences, specially including additional uses of conjunctive, number handling etc. A top-down predictive parser is designed using non-ambiguous predictive grammar. Predictive nature of the grammar ensures linear runtime of parsing process. Therefore, difficulty of parsing due to exponential runtime is over and parsing a massive volume of data is not a problem. Error recovery feature in Bangla parsing process has added a new dimension. This feature allows the parser to continue parsing after detection of error. Therefore, previously found problem of halting of parsing process is solved. Error recovery routine of the parser skips the error and parsing process again synchronizes with the rest of correct portion of input sentence, if error exists in that sentence. To make the error recovery process efficient, heuristic is applied. So, error may not be recovered correctly in all cases. But most of the cases error recovery is correct and most importantly parsing never stops due to error.

This thesis also includes some supporting modules of Bangla predictive parser, like structure of lexicon and strategy of lexical analysis for input Bangla sentences, which includes dynamic tagging of multiple meaning words. A simulation program justifies the correctness of grammar, parsing and error recovery mechanism.



1.1 Machine Translation

Machine Translation (MT) is a sub-field of computational linguistics that investigates the use of computer software to translate text or speech from one natural language to another. At its basic level, MT performs simple substitution of words in one natural language for words in another. Using corpus techniques [1], more complex translations may be attempted, allowing for better handling of differences in linguistic technology, phrase recognition and translation of idioms, as well as isolation of anomalies.

Current machine translation software often allows for customization by domain or profession (such as weather reports) – improving output by limiting the scope of allowable substitutions. This technique is particularly effective in domains where formal or formulaic language is used. It follows then that machine translation of government and legal documents more readily produces usable output than conversations or less standardized text.

Improved output quality can also be achieved by human intervention, for example, some systems are able to translate more accurately if the user has unambiguously identified which words in the text are names. With the assistance of these techniques, MT has been proven useful as a tool to assist human translators, and in some cases can even produce output that can be used “as is”. However, current systems are unable to produce output of the same quality as a human translator, particularly where the text to be translated uses casual language.

1.2 MT Process

MT process is stated as (1) Decoding the meaning of the source text, and (2) Encoding this meaning in the target language. Behind this ostensibly simple procedure lies a complex cognitive operation.

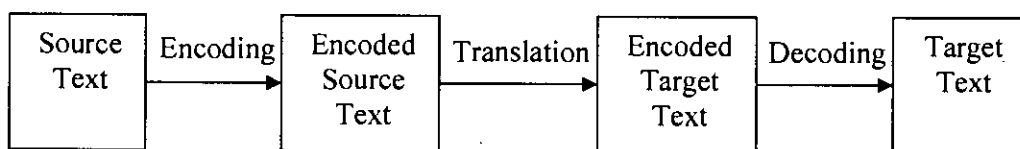


Figure 1.1: Work flow of a standard translator.

To decode the meaning of the source text in its entirety, the translator must interpret and analyze all the features of the text, a process that requires in-depth knowledge of the grammar, semantics, syntax, idioms, etc., of the source language, as well as the culture of its speaker. The translator needs the same in-depth knowledge to re-encode the meaning in the target language. Work flow of a standard translator is shown in figure 1.1.

1.3 MT Techniques

Machine translation can use a method based on linguistic rules, which means that words will be translated in a linguistic way – the most suitable words of the target language will replace the ones in the source language. It is often argued that the success of machine translation requires the problem of natural language understanding to be solved first.

Generally there are three popular techniques used for machine translation –

- Rule-based machine translation
- Statistical machine translation
- Example-based machine translation

1.3.1 Rule-based Machine Translation

Generally, rule-based methods parse a text, usually creating an intermediary symbolic representation, from which the text in the target language is generated. This method requires extensive lexicons with morphological, syntactic, and semantic information and, large sets of rules.

In a rule-based machine translation system the original text is first analyzed morphologically and syntactically in order to obtain a syntactic representation. This representation can then be refined to a more abstract level putting emphasis on the parts relevant for translation and ignoring other types of information. The transfer process then converts this final representation (still in the original language) to a representation of the same level of abstraction in the target language. These two representations are referred to as “intermediate” representations. From the target language representation, the stages are then applied in reverse.

Rule-based machine translation paradigm includes transfer-based machine translation, interlingual machine translation and dictionary based machine translation paradigm.

1.3.1.1 Transfer-based Machine Translation

Transfer-based machine translation is a type of machine translation, which is based on the idea of interlingua. It has the idea to make a translation it is necessary to have an intermediate representation that captures the “meaning” of the original sentence in order to generate the correct translation.

The way in which transfer-based machine translation systems work varies substantially, but in general they follow the same pattern: they apply sets of linguistic rules which are defined as correspondences between the structure of the source language and that of the target representation. The translation is generated from this representation using both bilingual dictionaries and grammatical rules.

It is possible with this translation strategy to obtain fairly high quality translations, with accuracy in the region of 90%, although this is highly dependent on the language pair in question – for example the distance between the two.

1.3.1.2 Interlingual Machine Translation

Interlingual machine translation is one of the classic approaches to machine translation. In this approach, the source language, i.e. the text to be translated is transformed into an interlingua, i.e. an abstract language-independent representation. The target language is then generated from the interlingua. Within the rule-based machine translation paradigm, the interlingual approach is an alternative to the direct approach and the transfer approach.

In the direct approach, words are translated directly without passing through an additional representation. In the transfer approach, the source language is transformed into an abstract, less language specific representation. Linguistic rules which are specific to the language pair then transform the source language representation into an abstract target language representation and from this the target sentence is generated.

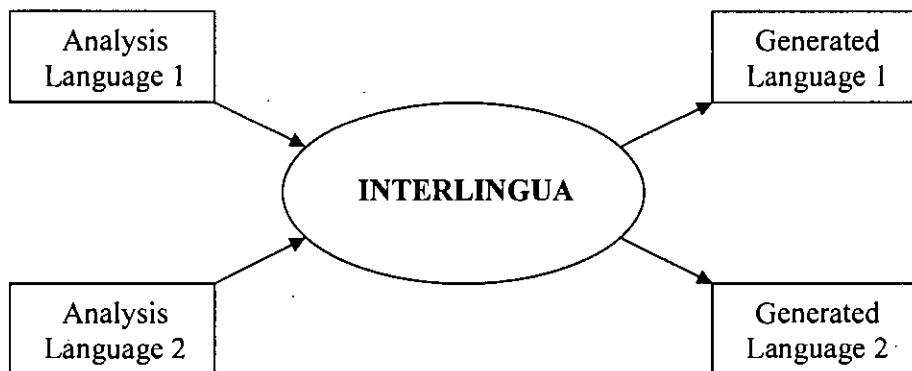


Figure 1.2: Demonstration of the languages which are used in the process of translating using a bridge language.

The interlingual approach to machine translation has advantages and disadvantages. The advantage in multilingual machine translation is that no transfer component has to be created for each language pair. The obvious disadvantage is that the definition

of an interlingual is difficult and may be even impossible for a wider domain. The ideal context for interlingual machine translation is thus multilingual machine translation in a very specific domain.

1.3.1.3 Dictionary-based Machine Translation

Machine translation can use a method based on dictionary entries, which means that the words will be translated as a dictionary does – word by word, usually without much correlation of meaning between them. Dictionary lookups may be done with or without morphological analysis or lemmatization. While this approach to machine translation is probably the least sophisticated, dictionary-based machine translation is ideally suitable for the translation of long lists of phrases on the subsentential (i.e. not a full sentence) level, e.g. inventories or simple catalogs of products and services [2]. It can also be used to expedite manual translation if the person carrying it out is fluent in both languages and therefore capable of correcting syntax and grammar.

1.3.2 Statistical Machine Translation

Statistical machine translation (SMT) is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual text corpora, such as the Canadian Hansard corpus, the English-French record of the Canadian Parliament and EUROPARL, the record of the European Parliament. Where such corpora are available, impressive results can be achieved translating texts of a similar kind, but such corpora are still very rare. The statistical approach contrasts with the rule-based approaches to machine translation as well as with example-based machine translation.

The first statistical machine translation software was CANDIDE from IBM. Google used SYSTRAN for several years, but has switched to a statistical translation method in October 2007. Recently, they improved their translation capabilities by inputting approximately 200 billion words from United Nations materials to train their system. Accuracy of the translation has improved [3].

As the translation systems are not able to store all native strings and their translations, a document is typically translated sentence by sentence, but even this is not enough. Language models are typically approximated by smoothed n -gram models, and similar approaches have been applied to translation models, but there is additional complexity due to different sentence lengths and word orders in the languages.

The statistical translation models were initially word based (Models 1-5 from IBM), but significant advances were made with the introduction of phrase based models [4]. Recent work has incorporated syntax or quasi-syntactic structures [5].

1.3.2.1 Word-based Machine Translation

In word-based machine translation, translated elements are words. Typically, the number of words in translated sentences are different due to compound words, morphology and idioms. The ratio of the lengths of sequences of translated words is called fertility, which tells how many foreign words each native word produces. Simple word-based translation is not able to translate language pairs with fertility rates different from one. To make word-based translation systems manage, for instance, with high fertility rates the system could be able to map a single word to multiple words, but not vice versa. For instance, if we are translating from French to English could produce zero or more French words. But there's no way to group two English words producing a single French word.

An example of word-based translation system is the freely available GIZA++ package (GPLed), which includes IBM models.

1.3.2.2 Phrase-based Machine Translation

In phrase-based machine translation, the restrictions produced by word-based translation have been tried to reduce by translating sequences of words to sequences of words, where the lengths can differ. The sequences of words are called, for

instance, blocks or phrases, but typically are not linguistic phrases but phrases found using statistical methods from the corpus. Restricting the phrases to linguistic phrases has been shown to decrease translation quality.

1.3.2.3 Syntax-based Machine Translation

In Syntax-based Machine Translation, the source language input is first parsed into a parse tree, which is then recursively converted into a string in the target language. The concept of Syntax-based Machine Translation was originally proposed in compiling (Irons, 1961; Lewis and Stearns, 1968; Aho and Ullman, 1972), where the source program is parsed into a tree representation that guides the generation of the object code. In other words, the translation is directed by a syntactic tree. In this context, a syntax-based translator consists of two components, a source language parser and a recursive converter which is usually modeled as a top-down tree-to-string transducer.

For example, a sentence "I eat rice" can be parsed according to following rules:

$S \rightarrow NP VP$

$NP \rightarrow P$

$P \rightarrow I$

$VP \rightarrow VB N$

$VB \rightarrow \text{eat}$

$N \rightarrow \text{rice}$

Here,

$S \equiv$ Sentence

$NP \equiv$ Noun Phrase

$VP \equiv$ Verb Phrase

$P \equiv$ Pronoun

$VB \equiv$ Verb

$N \equiv$ Noun

Equivalent parse tree is demonstrated as Figure 1.3.

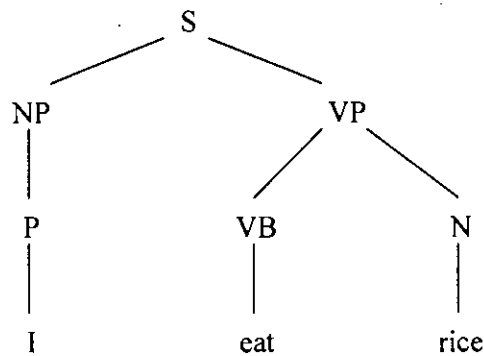


Figure 1.3: Parse tree of the sentence “I eat rice”, which is generated during syntax-based machine translation.

1.3.3 Example-based Machine Translation

The Example-based Machine Translation (EBMT) approach to machine translation is often characterized by its use of a bilingual corpus with *parallel texts* as its main knowledge base, at run-time. It is essentially a translation by analogy and can be viewed as an implementation of case-based reasoning approach of machine learning.

The foundation of example-based machine translation is the idea of translation by analogy. When applied to the process of human translation, the idea that translation takes place by analogy is a rejection of the idea that people translate sentences by doing deep linguistic analysis. Instead it is founded on the belief that people translate firstly by decomposing a sentence into certain phrases, then by translating these phrases, and finally by properly composing these fragments into one long sentence. Phrasal translations are translated by analogy to previous translations. The principle of translation by analogy is encoded to example-based machine translation through the example translations that are used to train such a system.

Example-based machine translation systems are trained from bilingual parallel corpora, which contain sentence pairs. These sentence pairs contain sentences in one language with their translations into another. There exist some sentence pairs known

as *minimal pair*, meaning that the sentences vary by just one element. These sentences make it simple to learn translations of subsentential units.

Example-based machine translation was first suggested by Nagao Makoto in 1984 [6]. It soon attracted the attention of scientists in the field of natural language processing.

1.4 Word Sense Disambiguation

Word sense disambiguation concerns finding a suitable translation when a word can have more than one meaning. The problem was first raised in the 1950s by Yehoshua Bar-Hillel [7]. He pointed out that without a "universal encyclopedia", a machine would never be able to distinguish between the two meanings of a word [8]. Today there are numerous approaches designed to overcome this problem. They can be approximately divided into "shallow" approaches and "deep" approaches.

Shallow approaches assume no knowledge of the text. They simply apply statistical methods to the words surrounding the ambiguous word. Deep approaches presume a comprehensive knowledge of the word. So far, shallow approaches have been more successful.

1.5 Applications of Machine Translation

There are now many software programs for translating natural language, several of them online, such as the SYSTRAN system which powers both Google translate and AltaVista's Babel Fish as well as Prompt that powers online translation services at Voila.fr and Orange.fr. Although no system provides the holy grail of "fully automatic high quality machine translation" (FAHQMT), many systems produce reasonable output.

Despite their inherent limitations, MT programs are used around the world. Probably the largest institutional user is the European Commission.

Togglertext uses a transfer-based system (known as Katakun) to translate between English and Indonesian.

Google has claimed that promising results were obtained using a proprietary statistical machine translation engine [9]. The statistical translation engine used in the Google language tools for Arabic ↔ English and Chinese ↔ English has an overall score of 0.4281 over the runner-up IBM's BLEU-4 score of 0.3954 (Summer 2006) in tests conducted by the National Institute for Standards and Technology [10][11][12]. Uwe Muegge has implemented a demo website [13] that uses a controlled language in combination with the Google tool to produce fully automatic, high-quality machine translations of his English, German, and French web sites.

With the recent focus on terrorism, the military sources in the United States have been investing significant amounts of money in natural language engineering. *In-Q-Tel* [14] (a venture capital fund, largely funded by the US Intelligence Community, to stimulate new technologies through private sector entrepreneurs) brought up companies like Language Weaver. Currently the military community is interested in translation and processing of languages like Arabic, Pashto, and Dari. Information Processing Technology Office in DARPA hosts programs like TIDES and Babylon Translator. US Air Force has awarded a \$1 million contract to develop a language translation technology [15].

1.6 Parsing

In computer science and linguistics, parsing, or, more formally, syntactic analysis, is the process of analyzing a sequence of tokens to determine grammatical structure with respect to a given (more or less) formal grammar. A parser is thus one of the components in an interpreter or compiler, where it captures the implied hierarchy of the input text and transforms it into a form suitable for further processing (often kind of parse tree, abstract syntax tree or other hierarchical structure) and normally checks for syntax errors at the same time. The parser [16] often uses a separate lexical analyzer to create tokens from the sequence of input characters. Parsers may

be programmed by hand or may be semi-automatically generated (in some programming language) by a tool (such as Yacc) from a grammar written in Backus-Naur form [17].

In some machine translation and natural language processing (NLP) systems, human languages are parsed by computer programs. For example, in syntax-based machine translation, parsing is very important as input sentence of source language is first parsed into hierarchical structure in this technique. Human sentences are not easily parsed by programs, as there is substantial ambiguity in the structure of human language. In order to parse natural language data, researchers must first agree on the grammar to be used. The choice of syntax is affected by both linguistic and computational concerns; for instance some parsing systems used lexical functional grammar, but in general, parsing for grammars of this type is known to be NP-complete. Head-driven phrase structure grammar is another linguistic formalism which has been popular in the parsing community, but other research efforts have focused on less complex formalisms such as the one used in the Penn Treebank [18]. Shallow parsing aims to find only the boundaries of major constituents such as noun phrases. Another popular strategy for avoiding linguistic controversy is dependency grammar parsing.

Most modern parsers are at least partly statistical; that is, they rely on a corpus of training data which has already been annotated (parsed by hand). This approach allows the system to gather information about the frequency with which various constructions occur in specific contexts. Approaches which have been used include straightforward PCGCs (probabilistic context free grammars) [19], maximum entropy, and neural nets. Most of the more successful systems use lexical statistics (that is, they consider the identities of the words involved, as well as their part of speech). However, such systems are vulnerable to over-fitting and require some kind of smoothing to be effective.

Parsing algorithms [16] for natural language can not rely on the grammar having 'nice' properties as with manually-designed grammars for programming languages.

As mentioned earlier some grammar formalisms are very computationally difficult to parse; in general, even if the desired structure is not context-free, some kind of context-free approximation to the grammar is used to perform a first pass. Algorithms which used context-free grammars often rely on some variant of the CKY algorithm [20], usually with some heuristic to prune away unlikely analyses to save time. However some systems trade speed for accuracy using, e.g. linear-time versions of the shift-reduce algorithm. A somewhat recent development has been parse re-ranking in which the parser proposes some large number of analyses, and a more complex system selects the best option. It is normally branching of on part and its subparts.

1.7 Machine Translation in Bangla

Machine translation in Bangla is a very promising field. A string machine translation system from Bangla to any other language will help to translate literary works, research works, articles, historical documents etc from Bangla to other languages. This will enrich Bangla as a language.

There are already some research and professional works done in this area. For example, ANUBADOK [21] is a machine translation software, which is capable of translating sentences from English language to Bangla language. First version of the software was released in 2006. Then second and latest version of the software released in 2008. The quality of translation of the software is moderately high. ANUBADOK is developed using freely available GIZA++ package (GPLed), which is a word-based machine translation system.

There are also some other works, but most of them are related to translating from English to Bangla. There are very few research works in the area of translating from Bangla to English or any other language.

1.7.1 Prospective Applications of Bangla Translation

Bangla is one of the most spoken languages (ranking 5th or 6th) of the world. Around 230 million people speak in this language. It comprises mainly in Bangladesh and Indian state of West Bengal. Bangla is the only language in the world for which native people sacrificed their lives. It happened in 21st February, 1952. For this reason, the 21st February is now observed all over the world as “International Day of Mother Language”. Besides, Bangla language is enriched with many famous literary works in its depth of history. Such works and documents are not property of native people now. This is property of people over the world. But widest possible distribution of Bangla around the world is impossible without machine translation. For human translator, this job is not feasible for a lot of reasons. Machines or automatic translator can do the job of translation efficiently.

A standard MT system divided into two subsystems: analysis or parsing and synthesis or generation. Parsing of Bangla natural language sentences can be used as a subsystem for Bangla to other language machine aided translation. In the parsing subsystem an input sentence is analyzed and converted into an SR. Once the SR is created for a particular sentence it is shown to the user as an intuitive attribute-value structure. The user can view the structure and make any necessary changes.

A parser module used the phrase structure (PS) rules and also find out how words in sentence related to each other. Therefore, parsing as sentence by using a set of PS rules produces underlying structure for sentence. The result of a Bangla natural language parser that produces a list of tokens from a Bangla sentence is then used as the input for an MT engine to produce other equivalent sentence of the given Bangla sentence. MT is an emerging paradigm for processing natural languages. Now a days, MT and NLP are being used by language industries for translation one language to another and allow people to interact with computers in a natural human language. An automatic translation will save a lot of time and cost of money.

Machine translation in Bangla can play an important role in social or political, commercial, scientific, philosophical etc, area.

1.7.1.1 Social or Political Application

The social or political importance of MT arises from the socio-political importance of translation in communities where more than one language is generally spoken. Here the only viable alternative to rather widespread use of translation is the adoption of a single common 'lingua franca', which is not a particularly attractive alternative, because it involves the dominance of the chosen language, to the disadvantages of speakers of the other languages, and raises the prospect of the other language often involves the disappearance of a distinctive culture, and a way of thinking, this is a loss that should matter to every one. So translation is necessary for communication for ordinary human interaction, and for gathering the information one needs to play a full part in society. Being allowed to express yourself in your own language, and to receive information that directly affects you in the same medium, seem to be an important, if often violated, right and it is one that depends on the availability of translation. The problem is that the demand for translation in the modern world far outstrips any possible supply. Part of the problem is that there are too few human expert translators, and that there is a limit on how far their productivity can be increased without automation. In short, it seems as though automation of translation is a social and political necessity for modern societies that do not wish to impose a common language on their members. Machine translation from Bangla to other languages will play an important role to disclose social activity, culture, customs, political views of Bengali nation.

1.7.1.2 Commercial Application

The commercial importance of MT is a result of related factors. First, translation itself is commercially important: faced with a choice between a product with an instruction manual in English, and one whose manual is written in Bangla, most English speakers will buy the former and in the case of a repair manual for a piece of

manufacturing machinery or the manual for a safety critical system, this is not just a matter of taste. Secondly, translation is expensive. Translation is a highly skilled job, requiring much more than mere knowledge of a number of languages, and in some countries at least; translator's salaries are comparable to other highly trained professionals. Moreover, delays in translation are costly. Estimates vary, but producing high quality translations of difficult material, a professional translator may average no more than about 4-6 pages of translation (perhaps 2000 words) per day, and it is quite easy for delays in translating product documentation to erode the market lead time of a new product. In brief, machine translation from Bangla to other languages will expedite translation of product tags, documentations, leaflets specially for export quality products, helping rapid shipment of products, and will contribute on national economy.

1.7.1.3 Scientific Application

Scientifically, MT is interesting because it is an obvious application and testing ground for many ideas in computer science, Artificial Intelligence, and linguistics, and some of the most important developments in these fields have begun in MT. To illustrate this: The origins of prolog, the first widely available logic programming language, which formed a key part of the Japanese 'fifth Generation' programmer of research in the late 1980s, can be found in the 'Q-Systems' language, originally developed for MT. In Bangla, there are many scientific works, specially in linguistics, history, economics, law, sociology, public administration, etc. Most of the works written in Bangla are not exposed to the world because of language translation problem. Machine translation from Bangla to other languages will enable exploration of Bangla works to world, will enable more interaction, will create more interest to researchers.

1.7.1.4 Philosophical Application

Philosophically, MT is interesting, because it represents an attempt to automatic and activity that can require the full range of human knowledge, that is, for any piece of

human knowledge, it is possible to think of a content where the knowledge is required. For example, getting the correct translation of “negatively charged electrons and protons” into Bangla depends on knowing that protons are positively charged, so the interpretation can not be something like “negatively charged electrons and positively charged protons”. In this sense, the extent to which one can automatic translation is an indication of the extent to which one can automatic ‘thinking’.

Despite this, very few people, even those who are involved in producing or commissioning translations, have much idea of what is involved in MT today, either at the practical level of what it means to have and use an MT system, or at the level of what is technically feasible, and what is science fiction. In the whole of the UK there are perhaps five companies who use MT for making commercial translations on a day-to-day basis. In continental Europe, where the need for commercial translation is for historical reasons greater, the number is larger, but it still represents an extremely small proportion of the overall translation effort that is actually undertaken. In Japan, where there is an enormous need for translation of Japanese into English, MT is just beginning to become established on a commercial scale, and some familiarity with MT is becoming a standard part of the training of a professional translator.

With the Advent of Internet technology and electronic commerce have increased the demand for automatic machine translation of sufficient quality for determining the content of a web page. Demand of machine translation has grown and will continue of a web page. Demand of machine translation has grown and will continue to grow steadily. A variety of authoring tools and document production techniques have also made linguistic information available on a variety of formats. The information placed on the web is needed to be translated to the language of the user for his/her understanding. If he/she tries to translate the whole contents word-by-word or sentence-by-sentence it will lose a lot of time. An automatic translator can do the job. Translation by machine will save a lot of time in this regard. A significant number of MT systems are found on browser over the Internet that can translate a

page into one or more languages. Moreover many language industries all over the world are engaged in multilingual translation of a language. Multilingual translation means the translation a language into several languages. The aim of machine translation of a particular language is to introduce the language to others who do not know the language. They will just use a machine for the job of translation. The explosion in electronic communication and the use of computers has explored the way of taking of technology to record, disseminate and may be even preserve the language and to exploit approaches and technique that are already tried and tested on more common languages.

In our country, every day many people go to abroad. For this purpose many documents are needed. Not all the documents are in English or in preferred language. Many documents are needed to be translated into English or any other languages. As manual translation is costly, laborious and time consuming, machine translation will be effective in this perspective. Moreover, machine translation in Bangla will be able to play vital role in our society and economy as described earlier.

1.7.2 Bangla Parsing

Bangla is one of the most enriched, organized and structured languages of the world. Bangla grammar is reasonably strict and organized. That is why a machine translation system involving Bangla language should follow syntax-based machine translation technique. Because, in this technique, input sentence of source language is first parsed according to the grammar of source language. Then translation technique is applied on parsed text.

To parse a Bangla text, a Bangla grammar is required. Parsing methodology of Bangla language sentences is already developed, which includes Bangla grammar and, parsing technique [22][23]. Current parsing methodology has many limitations which are discussed in the next section (1.7.3).

1.7.3 Limitations of Bangla Parsing

Existing parsing methodology of Bangla is capable of handling most type of Bangla sentences. But following features are missing in the methodology.

- **Non-ambiguity:** Existing Bangla grammar is ambiguous. A sentence can be parsed in different ways. For a single sentence, different parse trees may exist.
- **Predictive:** Existing Bangla parsing methodology is non-predictive. While parsing a Bangla sentence, all possible rules have to be examined to find appropriate sequence of rules to fit the input sentence. As a result, backtracking algorithm is required for parsing, which requires exponential runtime to parse a sentence. Using the methodology, a sentence can not be parsed in linear time.
- **Error recovery:** Existing Bangla parsing methodology lacks error recoverability. When any error is found in a Bangla sentence, parser can not continue parsing. Not only this, if any sentence can not be fit in corresponding grammar, parsing halts. In machine translation, it is not possible to write grammar to handle all types of sentences, as a correct sentence can be written in different ways. Writing such grammar is an NP-complete problem.
- **Unknown word:** There is always a certain probability of occurring some unknown words in a number of sentences of any natural language. Unknown words are basically words which are not found in dictionary. Existing Bangla parsing methodology is not capable of handling unknown words. If any unknown word is found, it is regarded as error and parsing stops.
- **Words with different meaning:** In most of the languages, there exist some words having different meaning. For example, in the sentence “ও আসবে ও তুমি

আস”, first “ও” represents a pronoun and second “ও” represents a conjuncture. Existing Bangla parsing methodology is not capable of handling such words with different meaning and different parts of speech. It assumes each word of a dictionary has single parts of speech.

1.7.4 Present State of Bangla Parsing and Contributions of the Thesis

Machine translation in Bangla is an emerging field. There are some research and professional works in this area. As mentioned earlier, ANUBADOK is an English to Bangla machine translation software. There are also some other research works in English to Bangla machine translation. But there are very few works in Bangla to English machine translation. There is no complete Bangla to English machine translation software.

There are many machine translation techniques as discussed earlier in this chapter. No technique can be chosen to be the best. There are many languages in this world. The languages are different in grammar, style, flexibility, etc. Different machine translation techniques are suitable for different languages. As being a discipline language, syntax-based machine translation system is most appropriate for Bangla language. In this technique, source language text is first parsed, then language translation technique is applied on the parsed text. So, parsing is the base method for syntax-based machine translation.

Bangla parsing methodology [22][23] is already developed. This methodology includes Bangla grammar and Bangla text parsing technique. There exist context-free and context-sensitive phrase structure rules, which can be used to parse almost all kinds of Bangla sentences. The methodology is capable of parsing complex and compound sentences, which was not possible before.

First complete Bangla parsing methodology [24] was developed in 1998. But the methodology was capable of parse only simple sentences. It was not capable of

parsing complex or compound sentences. Then phrasal category and Transformational Generative Grammars (TGG's) for simple sentences of Bangla language [25] was developed in 1999. Machine translation dictionaries for Bangla language [26] was developed in 2002. Syntax analysis using comprehensive approach [27] was presented in 2003. Then a knowledge based approach to Bangla-English machine translation for simple assertive sentences [28] was done in the same year. Parsing methodology for Bangla natural language sentences [22][23] was developed in the same year. The latest work is capable of parsing almost all types of sentences, which includes complex and compound sentences.

Later, there are also some other works in Bangla parsing. Structure based Bangla to English machine translation technique [29] was developed in 2005. Structure based machine translation is not so appropriate for Bangla language. Then a bottom-up parsing algorithm for Bangla parser applying context-sensitive transformation rules to maintain the freeness of word order [30] was developed in the same year. Latest contribution in Bangla parsing is parsing algorithm for Bangla parser to handle affirmative and negative sentences [31] developed in 2006.

There are many problems and limitations in existing Bangla parsing methodology. Existing Bangla Grammar is ambiguous. Because of ambiguous nature, existing Bangla parsing methodology is non-predictive. Existing Bangla parsing methodology can not continue parsing if any error occurs in input Bangla text. While translating using machine translation system, it can never be assumed that all input sentences are correct and can be fit in rules of grammar. Existing Bangla parsing methodology can not handle unknown words. Parsing halts if any unknown word is found in input sentence. Existing Bangla parsing methodology also can not handle words having different parts-of-speech.

This thesis overcomes all the above problems and limitations. This thesis presents a non-ambiguous grammar thereby predictive nature of Bangla parsing with error recovery capability. The parser uses a structured Bangla dictionary as lexicons. The

syntax analysis part is capable of handling complex words and pieces them into simple words.

As the grammar is non-ambiguous, just one parse tree can be generated for an input sentence. Because of predictive parsing nature, parsing can be done in linear runtime. Previous parsing methods used non-predictive parsing techniques, as a result exponential runtime is required to generate parse tree. Most important achievement is capability to continue parsing if any error occurs. Previous methods halt parsing, if any error is found in input sentence. In case of language translation, assumption of error-freeness is quite impractical. Parsing methodology in this thesis detects errors in sentences and continues parsing overlooking the error. This parsing methodology can handle unknown words. In case of language translation, some words are certainly found, which are not available in dictionary. Existing parsing methodologies halt parsing if any unknown word is found. Another important achievement of this thesis is to handle words having different meaning or different parts-of-speech. This parsing technique uses runtime parts-of-speech tagging for such kinds of words.

1.8 Organization of the Thesis

In this chapter, we have discussed basics of machine translation and parsing. We also have discussed prospects of machine translation in Bangla and also current state of Bangla parsing methodology. The rest parts of this thesis is organized as follows,

In chapter 2, we have discussed syntax-based machine translation and theories behind this category of machine translation like parsing, error recovery, context-free grammar, different parsing techniques and theories of non-recursive predictive parser.

In chapter 3, we have discussed existing Bangla grammar, which is ambiguous and non-predictive. Existing Bangla grammar is discussed in different levels like noun phrase, verb phrase, adjective phrase, simple sentence, complex sentence, compound

sentence and sentence level. We also have discussed limitations of existing Bangla grammar.

In chapter 4, we have eliminated ambiguity and non-predictive problem from Bangla grammar for simple sentences and designed a new non-ambiguous predictive grammar. We have incorporated the idea of handling unknown or non-dictionary words. We also have brought out some more new ideas in the grammar.

In chapter 5, we have analyzed all possible combinations of Bangla complex sentences. Then, we have designed non-ambiguous predictive grammar for complex sentence, which abolishes ambiguity and non-predictive problem of existing Bangla grammar for complex sentence.

In chapter 6, we have eliminated ambiguity and non-predictive problem from Bangla grammar for compound sentences, thereby have designed a new non-ambiguous predictive grammar. We also have facilitated more than two basic sentences in a compound sentence.

In chapter 7, we have eliminated ambiguity and non-predictive problem from Bangla grammar for all types of sentences. Therefore, non-ambiguous predictive comprehensive Bangla grammar is designed. We also have elaborated significance of each non-terminal in new proposed Bangla grammar.

In chapter 8, we have designed architecture of predictive parser, strategy and work flow of lexical analyzer and syntax analyzer and error recovery mechanism using new proposed non-ambiguous predictive Bangla grammar. We also have simulated the functionality of predictive parser for both correct and incorrect sentences.

In chapter 9, we have discussed the architecture and functionality of simulation program, which we have developed to simulate the correctness and usefulness of our proposed Bangla grammar and error recovery policy.

Finally, in chapter 10, we have discussed overall achievements of this thesis. We also have discussed further enhancement and future related research scopes in this area.

Syntax-based Machine Translation

2.1 Syntax-based Machine Translation

Syntax-based Machine Translation is one of the statistical machine translation paradigms. In this machine translation technique, the source language text input is first parsed into a parse tree. This parsed text will be recursively converted into a string in the target language.

The idea of syntax-based machine translation comes from the idea of a compiler design. E. T. Irons first worked on syntax-directed compiler design in 1961 [32]. Then P. M. Lewis and R. E. Stearns contributed more in this area in 1968 [33]. Finally, Alfred V. Aho and Jeffrey D. Ullman brought complete idea of parsing, translation and compiling using syntax-directed approach in 1972 [34].

In syntax-based translation, the source text is parsed into a tree representation that guides the generation of the object code. In other words, the translation is directed by a syntactic tree. In this context, a syntax-based translator consists of two components, a source language parser and a recursive converter which is usually modeled as a top-down tree-to-string transducer.

Syntax-based translation technique uses a context-free grammar to specify the syntactic structure of the input. With each grammar symbol, it associates a set of attributes, and with each production, a set of semantic rules for computing values of the attributes associated with the symbols appearing in that production.

For example, a sentence “I eat rice” can be parsed according to following rules:

$$S \rightarrow NP VP$$
$$NP \rightarrow P$$

$P \rightarrow I$

$VP \rightarrow VB N$

$VB \rightarrow \text{eat}$

$N \rightarrow \text{rice}$

Here,

$S \equiv$ Sentence

$NP \equiv$ Noun Phrase

$VP \equiv$ Verb Phrase

$P \equiv$ Pronoun

$VB \equiv$ Verb

$N \equiv$ Noun

Equivalent parse tree is demonstrated as Figure 2.1.

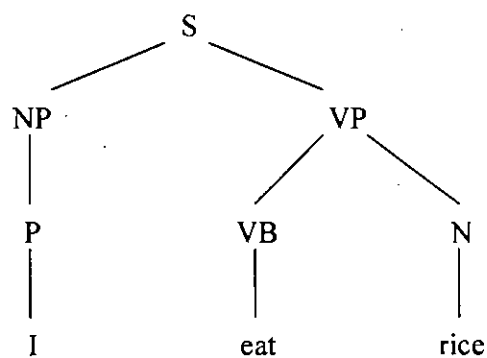


Figure 2.1: Parse tree of the sentence "I eat rice", which is generated during syntax-based machine translation.

Translation process using this technique can be modeled in three phases:

- Lexical analysis
- Syntax analysis
- Translation

2.1.1 Lexical Analysis

Lexical analysis is the process of converting a sequence of characters into a sequence of tokens. Programs performing lexical analysis are called lexical analyzers or lexers. A lexer is often organized as separate scanner and tokenizer functions, though the boundaries may not be clearly defined.

Lexical analyzer is the first phase of a syntax-based machine translator. Its main task is to read the input characters and produce as output a sequence of tokens that the parser uses for syntax analysis. This interaction, summarized schematically in figure 2.2, is commonly implemented by making the lexical analyzer be a subroutine or a co-routine of the parser. Upon receiving a “get next token” command from the parser, as mentioned in figure 2.2, the lexical analyzer reads input characters until it can identify the next token.

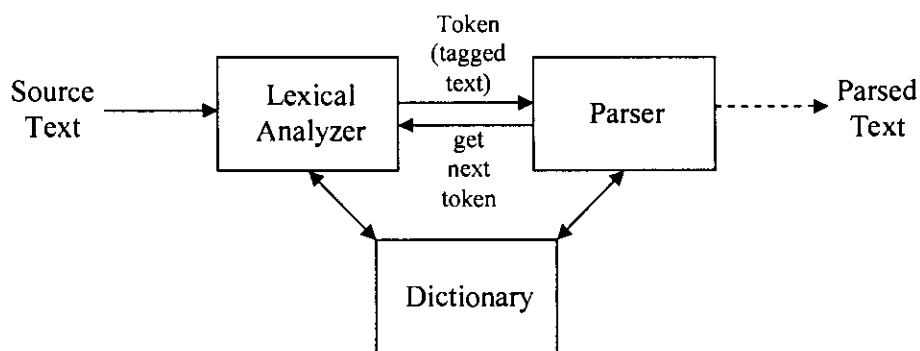


Figure 2.2: Interaction of lexical analyzer with parser.

2.1.1.1 Scanner

The first stage, the scanner, is usually based on a finite state machine. It has encoded within it information on the possible sequences of characters that can be contained within any of the tokens it handles (individual instances of these character sequences are known as lexemes). For instance, an *integer* token may contain any sequence of numeric digit characters. In many cases, the first non-whitespace character can be used to deduce the kind of token that follows and subsequent input characters are then processed one at a time until reaching a character that is not in the set of

characters acceptable for that token (this is known as the maximal munch rule). In some systems the lexeme creation rules are more complicated and may involve backtracking over previously read characters.

2.1.1.2 Tokenizer

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. The process can be considered a sub-task of parsing input. A process of tokenization could be used to split the sentence into word tokens.

2.1.2 Syntax Analysis

Hierarchical analysis is called syntax analysis or parsing. It involves grouping the tokens of the source sentence into grammatical phrases that are used by the translator to synthesize output. Usually, the grammatical phrases of the source text are represented by a parse tree.

2.1.3 Translation

This is the last phase in syntax-based machine translation. In phase takes parsed text as input. Then words in source language text are translated into destination language. Then some re-ordering of words is done using the grammar of target language. Then destination language text is found from parse tree by recursive conversion.

2.2 Parser

A parser is one of the components in an interpreter or compiler or machine translator, where it captures the implied hierarchy of the input text and transforms it into a form suitable for further processing (often some kind of parse tree, abstract

syntax tree or other hierarchical structure) and normally checks for syntax errors at the same time. The parser often uses a separate lexical analyzer to create tokens from the sequence of input characters. Parsers may be programmed by hand or may be semi-automatically generated (in some programming language) by a tool (such as Yacc) from a grammar written in Backus-Naur form.

A parser obtains a string of tokens from the lexical analyzer, and verifies that the string can be generated by the grammar for the source language. The parser reports any syntax errors in an intelligible fashion. It should also recover from commonly occurring errors so that it can continue processing the remainder of its input.

There are three general types of parsers for grammars. Universal parsing methods such as the Cocke-Younger-Kasami algorithm [20] and Earley's algorithm [19] can parse any grammar. These methods, however, are too inefficient to use in production compilers. The methods commonly used in compilers are classified as being either top-down or bottom-up. As indicated by their names, top-down parsers build parse trees from the top (root) to the bottom (leaves), while bottom-up parsers start from the leaves and work up to the root. In both cases, the input to the parser is scanned from left to right, one symbol at a time.

The most efficient top-down and bottom-up methods work only on subclasses of grammars, but several of these subclasses, such as the LL and LR grammars, are expressive enough to describe most syntactic constructs in programming languages. Parsers implemented by hand often work with LL grammars. Parsers for the larger class of LR grammars are usually constructed by automated tools.

The output of the parser is some representation of the parse tree for the stream of tokens produced by the lexical analyzer. In practice, there are a number of tasks that might be conducted during parsing, such as collecting information about various tokens into the symbol table, performing type checking and other kinds of semantic analysis.

A parser should report the nature of syntactic errors and adopt general strategies for error recovery. Two of these strategies, called panic-mode and phrase-level recovery, are used in most of the parsing methods.

2.2.1 Role of a Parser in a Machine Translator

In some machine translation (such as syntax-based machine translation) and natural language processing systems, human languages are parsed by computer programs. Human sentences are not easily parsed by programs, as there is substantial ambiguity in the structure of human language. In order to parse natural language data, researchers must first agree on the grammar to be used. The choice of syntax is affected by both linguistic and computational concerns; for instance some parsing systems use lexical functional grammar, but in general, parsing for grammars of this type is known to be NP-complete. Head-driven phrase structure grammar is another linguistic formalism which has been popular in the parsing community, but other research efforts have focused on less complex formalisms such as the one used in the Penn Treebank. Shallow parsing aims to find only boundaries of major constituents such as noun phrases. Another popular strategy for avoiding linguistic controversy is dependency grammar parsing.

Most modern parsers are least partly statistical; that is, they rely on a corpus of training data which has already been annotated (parsed by hand). This approach allows the system to gather information about the frequency with which various constructions occur in specific contexts. Approaches which have been used include straightforward PCFGs (probabilistic context free grammars), maximum entropy, and neural nets. However such systems are vulnerable to over-fitting and require some kind of smoothing to be effective.

Parsing algorithms for natural language cannot rely on the grammar having 'nice' properties as with manually-designed grammar for programming languages. As mentioned earlier some grammar formalisms are computationally very difficult to parse; in general, even if the desired structure is not context-free, some kind of

context-free approximation to the grammar is used to perform a first pass. Algorithms which use context-free grammars often rely on some variant of the CYK algorithm, usually with some heuristic to prune away unlikely analyses to save time. However some systems trade speed for accuracy using, e.g., linear-time versions of the shift-reduce algorithm. A somewhat recent development has been parse re-ranking in which the parser proposes some large number of analyzes, and a more complex system selects the best option. It is normally branching of one part and its subparts.

2.3 Error Recovery in a Parser

There are many different general strategies that a parser can employ to recover from a syntactic error. Although no one strategy has proven itself to be universally acceptable, a few methods have broad applicability. Here we introduce the following strategies:

- Panic-mode
- Phrase level
- Error productions
- Global correction

2.3.1 Panic-mode Recovery

This is the simplest method to implement and can be used by most parsing methods. On discovering an error, the parser discards input symbols one at a time until one of a designated set of synchronizing tokens is found. The synchronizing tokens are usually delimiters. The compiler designer must select the synchronizing tokens appropriate for the source language. While panic-mode correction often skips a considerable amount of input without checking it for additional errors, it has the advantage of simplicity and, unlike some other methods to be considered later, it is guaranteed not to go into an infinite loop. In situations where multiple errors in the same statement are rare, this method may be quite adequate.

2.3.2 Phrase Level Recovery

On discovering an error, a parser may perform local correction on the remaining input; i.e., it may replace a prefix of the remaining input by some string that allows the parser to continue. A typical local correction would be to replace a comma by a semicolon, delete an extraneous semicolon, or insert a missing semicolon. The choice of the local correction is left to the compiler designer. Of course, we must be careful to choose replacements that do not lead to infinite loops, as would be the case, for example, if we always inserted something on the input ahead of the current input symbol.

This type of replacement can correct any input string and has been used in several error-repairing compilers. The method was first used with top-down parsing. Its major drawback is the difficulty it has in coping with situations in which the actual error has occurred before the point of detection.

2.3.3 Error Productions

If we have a good idea of the common errors that might be encountered, we can augment the grammar for the language at hand with productions that generate the erroneous constructs. We then use the grammar augmented by these error productions to construct a parser. If an error production is used by the parser, we can generate appropriate error diagnostics to indicate the erroneous construct that has been recognized in the input.

2.3.4 Global Correction

Ideally, we would like a compiler to make as few changes as possible in processing an incorrect input string. There are algorithms for choosing a minimal sequence of changes to obtain a globally least-cost correction. Given an incorrect input string x and grammar G , these algorithms will find a parse tree for a related string y , such that the number of insertions, deletions, and changes of tokens required to string

form x into y is as small as possible. Unfortunately, these methods are in general too costly to implement in terms of time and space, so these techniques are currently only of theoretical interest. This problem can be considered as NP-complete problem.

2.4 Context-free Grammar

In formal language theory, a context-free grammar (CFG) is a grammar in which every production rule is of the form

$$V \rightarrow w$$

where V is a single non-terminal symbol, and w is a string of terminals and/or non-terminals (possibly empty). The term “context-free” expresses the fact that non-terminals can be rewritten without regard to the context in which they occur. A formal language is context-free if some context-free grammar generates it. Context-free grammars play a central role in the description and design of programming languages and compilers. They are also used for analyzing the syntax of natural languages.

The context-free grammar (or “phrase-structure grammar” as Chomsky called it) formalism developed by Noam Chomsky, in the mid 1950s [17], took the manner in which linguistics had described this grammatical structure, and then turned into rigorous mathematics. A context-free grammar provides a simple and precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the “block structure” of sentences in a natural way. Its simplicity makes the formalism amenable to rigorous mathematical study, but it comes at a price: important features of natural languages syntax such as agreement and reference cannot be expressed in a natural way, or not at all.

Block structure was introduced into computer programming languages by the Algol project [35], which, as a consequence, also featured a context-free grammar to describe the resulting Algol syntax. This became a standard feature of computer languages, and the notation for grammars used in concrete descriptions of computer

languages came to be known as Backus-Naur Form, after two members of the Algol language design committee. The “block structure” aspect that context-free grammars capture is so fundamental to grammar that the terms syntax and grammar are often identified with context-free grammar rules, especially in computer science. Formal constraints not captured by the grammar are then considered to be part of the “semantics” of the language.

A context-free grammar G is a 4-tuple:

$G = (V, \Sigma, R, S)$ where

1. V is a set of non-terminal characters or variables. They represent different types or clause in the sentence.
2. Σ is a finite set of terminals, disjoint with V , which make up the actual content of the sentence.
3. R is a relation from V to $(V \cup \Sigma)^*$ such that $\exists w \in (V \cup \Sigma)^* : (S, w) \in R$.
4. S is a start symbol, used to represent the whole sentence (or program). It must be an element of V .

In addition, R is a finite set. The members of R are called the rules of productions of the grammar.

2.4.1 Terminal

Terminals are the basic symbols from which strings are formed. The word “token” is a synonym for “terminal” when we are talking about grammars from linguistics or programming languages.

2.4.2 Non-terminal

Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar. They also impose a hierarchical structure on the language that is useful for both syntax analysis and translation.

2.4.3 Start Symbol

In a grammar, one non-terminal is distinguished as the start symbol, and the set of strings it denotes is the language defined by the grammar.

2.4.4 Productions

The productions of a grammar specify the manner in which the terminals and non-terminals can be combined to form strings. Each production consists of a non-terminal, followed by an arrow, followed by a string of non-terminals and terminals.

2.4.5 Normal Forms

Every context-free grammar that does not generate the empty string can be transformed into one in which no rule has the empty string as a product (a rule with ϵ as a product is called ϵ -production). If it does generate the empty string, it will be necessary to include the rule $S \rightarrow \epsilon$, but there need be no other ϵ -rule. Every context-free grammar with no ϵ -production has an equivalent grammar in Chomsky normal form [36] or Greibach normal form [36]. “Equivalent” here means that the two grammars generate the same language.

Because of the especially simple form of production rules in Chomsky Normal Form grammars, this normal form has both theoretical and practical implications. For instance, given a context-free grammar, one can use the Chomsky Normal Form to construct a polynomial-time algorithm which decides whether a given string is in the language represented by that grammar or not (the CYK algorithm).

2.4.6 Undecidable Problems

Although some operations on context-free grammars are decidable due to their limited power, CFGs do have interesting undecidable problems. One of the simplest

and most cited is the problem of deciding whether a CFG accepts the language of all strings. A reduction can be demonstrated to this problem from the well-known undecidable problem of determining whether a Turing machine [37] accepts a particular input. The reduction uses the concept of a computation history, a string describing an entire computation of a Turing machine. We can construct a CFG that generates all strings that are not accepting computation histories for a particular Turing machine on a particular input, and thus it will accept all strings only if the machine does not accept that input.

As a consequence of this, it is also undecidable whether two CFGs describe the same language, since we can't even decide whether a CFG is equivalent to the trivial CFG deciding the language of all strings. Another point worth mentioning is that the problem of determining if a context-sensitive grammar describes a context-free language is undecidable.

2.4.7 Extensions

An obvious way to extend the context-free grammar formalism is to allow non-terminals to have arguments, the values of which are passed along within the rules. This allows natural language features such as agreement and reference, and programming language analogs such as the correct use and definition of identifiers, to be expressed in a natural way. E.g. we can now easily express that in English sentences, the subject and verb must agree in number.

In computer science, examples of this approach include affix grammars [38], attribute grammars [39], indexed grammars [40], and Van Wijngaarden two-level grammars [39]. Similar extensions exist in linguistics. Another extension is to allow additional symbols to appear at the left hand side of rules, constraining their application. This produces the formalism of context-sensitive grammars.

2.4.8 Linguistic Applications

Chomsky initially hoped to overcome the limitations of context-free grammars by adding transformation rules [17]. Such rules are another standard device in traditional linguistics; e.g. passivization in English. However, arbitrary transformations must be disallowed, since they are much too powerful (Turing complete). Much of generative grammar has been devoted to finding ways of refining the descriptive mechanisms of phrase-structure grammar and transformation rules such that exactly the kinds of things can be expressed that natural language actually allows.

His general position regarding the non-context-freeness of natural language has held up since then [41], although his specific examples regarding the inadequacy of context free grammars (CFGs) in terms of their weak generative capacity were later disproved [42]. Gerald Gazdar and Geoffrey Pullum have argued that despite a few non-context-free constructions in natural language (such as cross-serial dependencies in Swiss German [41] and reduplication in Bambara [43]), the vast majority of forms in natural language are indeed context-free [42].

2.4.9 Parse Tree

A parse tree or concrete syntax tree is an ordered and rooted tree represents the syntactic structure of a string according to some formal grammar. In a parse tree, the interior nodes are labeled by non-terminals of the grammar, while the leaf nodes are labeled by terminals of the grammar. Parse trees may be generated for sentences in natural languages, as well as during processing of computer languages, such as programming languages.

A parse tree is made up of nodes and branches. Corresponding parse tree of an English sentence called "John hit the ball" is demonstrated in figure 2.3. The parse tree is the entire structure, starting from S and ending in each of the leaf nodes (John, hit, the, ball).

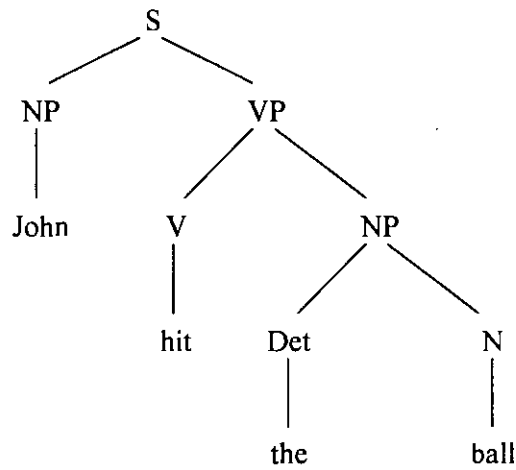


Figure 2.3: Parse tree of the sentence “John hit the ball”.

2.4.10 Derivations

The terms “parse tree” and “derivations” are closely related. To see the relationship between derivations and parse trees, we can consider any derivation $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n$, where α_1 is a single non-terminal A . For each sentential form α_i in the derivation, we construct a parse tree whose yield is α_i . The process is an induction on i . For the basis, the tree for $\alpha_1 \leq A$ is a single node labeled A . To do the induction, suppose we have already constructed a parse tree whose yield is $\alpha_{i-1} = X_1 X_2 \dots X_k$. Suppose, α_i is derived from α_{i-1} by replacing X_j , a non-terminal, by $\beta = Y_1 Y_2 \dots Y_r$. That is, at the i th step of the derivation, production $X_j \rightarrow \beta$ is applied to α_{i-1} to derive $\alpha_i = X_1 X_2 \dots X_{j-1} \beta X_{j+1} \dots X_k$.

To model this step of the derivation, we find the j th leaf from the left in the current parse tree. This leaf is labeled X_j . We give this leaf r children, labeled Y_1, Y_2, \dots, Y_r , from the left. As a special case, if $r = 0$, i.e., $\beta = \epsilon$, then we give the j th leaf one child labeled ϵ .

For example, the sentence “John hit the ball” (for which parse tree shown in figure 2.3), derivation can be done as follows:

$$\begin{aligned} S &\Rightarrow NP VP \\ &\Rightarrow NP V NP \\ &\Rightarrow NP V Det N \end{aligned}$$

2.4.11 Ambiguous Grammar

A grammar is said to be an ambiguous grammar if there is some string that it can generate in more than one way (i.e., the string has more than one parse tree). A language is inherently ambiguous if it can only be generated by ambiguous grammars.

For example, the context free grammar

$$AP \rightarrow AP AP \mid AD$$

is ambiguous (here, AP denotes Adjective Phrase and AD denotes Adjective), since the sentence segment “extremely very good” can be derived in at least ways.

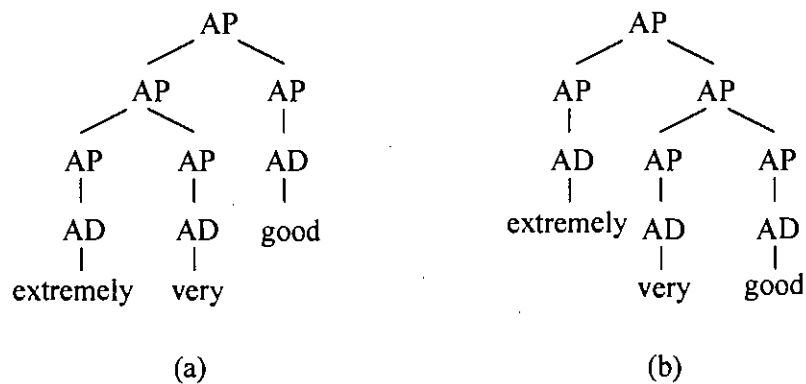


Figure 2.4: Two ways of parse tree (a), (b) of the sentence segment “extremely very good”.

First way:

$$\begin{aligned} AP &\Rightarrow AP AP \\ &\Rightarrow AP AP AD \\ &\Rightarrow AD AD AD \end{aligned}$$

Second way:

$$AP \Rightarrow AP AP$$

\Rightarrow AD AP AP
 \Rightarrow AD AD AD

Corresponding parse trees of the two ways are demonstrated in figure 2.4.

There is an obvious difficulty in parsing an ambiguous grammar by a deterministic parser but nondeterministic parsing imposes a great efficiency penalty. Most constructs of interest to parsing can be recognized by unambiguous grammars. Some ambiguous grammars can be converted into unambiguous grammars, but no general procedure for doing this is possible just as no algorithm exists for detecting ambiguous grammars.

2.4.11.1 Elimination of Ambiguity

In most of the cases, an ambiguous grammar can be rewritten to eliminate the ambiguity. There is no hard and fast rule to eliminate ambiguity of a grammar. Elimination technique of a grammar resides in the reason of ambiguity.

For example, the context free grammar

$$AP \rightarrow AP AP \mid AD$$

which is discussed in section 2.4.11, is ambiguous, because precedence of adjective(s) is not mentioned in the grammar. Ambiguity of the grammar can be eliminated by making the grammar left associative or right associative.

We can make the grammar left associative and can re-write the grammar as follows:

$$AP \rightarrow AP AD \mid AD$$

Using the grammar, the sentence segment “extremely very good” can be parsed in only one way, as demonstrated in figure 2.5. Thus ambiguity of grammar is eliminated.

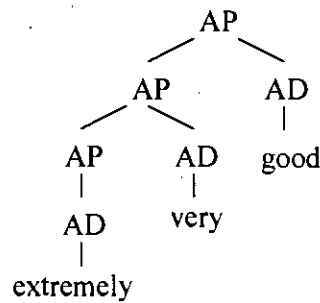


Figure 2.5: Parse tree of the sentence segment “extremely very good” using left associative rule of new grammar.

Same ambiguous grammar mentioned in section 2.4.11, can also be made non-ambiguous by making the grammar right associative. We can re-write the grammar as follows:

$$AP \rightarrow AD AP \mid AD$$

Using the grammar, the sentence segment “extremely very good” can be parsed in only one way, as demonstrated in figure 2.6. Thus ambiguity of grammar is eliminated.

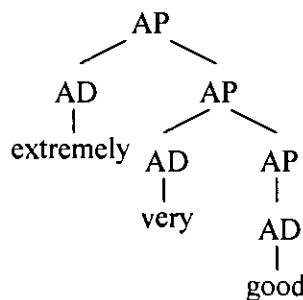


Figure 2.6: Parse tree of the sentence segment “extremely very good” using right associative rule of new grammar.

2.4.12 Left Recursion

A grammar is said to be left recursive if it has a non-terminal A such that there is a derivation $A \xRightarrow{+} A\alpha$ for some string α . In terms of context-free grammar, a non-terminal r is left-recursive if the left-most symbol in any of r 's ‘alternatives’ either immediately (direct left-recursive) or through some other non-terminal definitions

(indirect/hidden left-recursive) rewrites to r again. Top-down parsing methods cannot handle left-recursive grammars.

Immediate left recursion occurs in rules of the form

$$A \rightarrow A\alpha \mid \beta$$

Where α and β are sequences of non-terminals and terminals, and β does not start with A .

Indirect left recursion in its simplest form can be defined as:

$$A \rightarrow B\alpha \mid C$$

$$B \rightarrow A\beta \mid D$$

Possibly giving the derivation $A \Rightarrow B\alpha \Rightarrow A\beta\alpha \Rightarrow \dots$

More generally, for the non-terminals A_0, A_1, \dots, A_n indirect left recursion can be defined as being of the form:

$$A_0 \rightarrow A_1\alpha_1 \mid \dots$$

$$A_1 \rightarrow A_2\alpha_2 \mid \dots$$

...

$$A_n \rightarrow A_0\alpha_{(n+1)} \mid \dots$$

Where $\alpha_1, \alpha_2, \dots, \alpha_n$ are sequences of non-terminals and terminals.

A formal grammar that contains left recursion cannot be parsed by a naive recursive descent parser unless they are converted to a weakly equivalent right-recursive form. In contrast, left recursion is preferred for LALR parsers because it results in lower stack usage than right recursion. However, recent research demonstrates that it is possible to accommodate left-recursive grammars (along with all other forms of general CFGs) in a more sophisticated top-down parser by use of curtailment. A recognition algorithm which accommodates ambiguous grammars with direct left-recursive production rules is described by Frost and Hafiz in 2006 [44]. That algorithm was extended to a complete parsing algorithm to accommodate indirect as well as direct left-recursion in polynomial time, and to generate compact polynomial-size representations of the potentially-exponential number of parse trees for highly-ambiguous grammars by Frost, Hafiz and Callaghan in 2007 [45]. The

algorithm has since been implemented as a set of parser combinators written in the Haskell programming language. The implementation details of these new set of combinators can be found in a paper [46] by the above-mentioned authors, which was presented in PADL'08.

2.4.12.1 Elimination of Left Recursion

As discussed in the previous section (2.4.12), simplest form of left recursion can be defined as:

$$A \rightarrow A\alpha \mid \beta$$

This production can be replaced by the non-recursive productions:

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

without changing the set of strings derivable from A . This rule by itself suffices in many grammars.

The general algorithm to remove immediate left recursion follows. Several improvements to this method have been made, including the ones described in "Removing Left Recursion from Context-Free Grammars", written by Robert C. Moore.

For each rule of the form

$$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \dots \mid \beta_m$$

where A is a left-recursive non-terminal, α is a sequence of non-terminals and terminals that is not null ($\alpha \neq \varepsilon$), β is a sequence of non-terminals and terminals that does not start with A .

Left recursive A -production can be replaced by the following productions:

$$A \rightarrow \beta_1 A' \mid \dots \mid \beta_m A'$$

$$A' \rightarrow \varepsilon \mid \alpha_1 A' \mid \dots \mid \alpha_n A'$$

If the grammar has no ϵ -productions (no productions of the form $A \rightarrow \dots | \epsilon | \dots$) and is not cyclic (no derivations of the form $A \Rightarrow \dots \Rightarrow A$ for any non-terminal A), this general algorithm may be applied to remove indirect left recursion:

Algorithm 2.1:

```

Arrange the non-terminals in some fixed order  $A_1, \dots, A_n$ .
for i = 1 to n do
  for j = 1 to (i-1) do
    let the current  $A_j$  productions be  $A_j \rightarrow \delta_1 | \dots | \delta_k$ 
    replace each production  $A_i \rightarrow A_j \gamma$  by  $A_i \rightarrow \delta_1 \gamma | \dots | \delta_k \gamma$ 
    remove direct left recursion for  $A_i$ 
  end
end
end

```

The above transformations remove left-recursion by creating a right-recursive grammar i.e., this changes the associativity of our rules.

2.4.13 Left Factoring

Left factoring is a grammar transformation that is useful for producing a grammar suitable for predictive parsing. The basic idea is that when it is not clear which of two alternative productions to use to expand a non-terminal A , we may be able to rewrite the A -productions to defer the decision until we have seen enough of the input to make the right choice.

If $A \rightarrow \alpha\beta_1 | \alpha\beta_2$ are two A -production, and the input begins with a non-empty string derived from α , we do not know whether to expand A to $\alpha\beta_1$ or to $\alpha\beta_2$. However, we may defer the decision by expanding A to $\alpha A'$. Then after seeing the input derived from α , we expand A' to β_1 or to β_2 . That is, left-factored, the original productions become:

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \beta_1 | \beta_2$$

In general, left factorization of a grammar can be done using the following algorithm:

Algorithm 2.2:

For each non-terminal A , find the longest prefix α common to two or more of its alternatives. If $\alpha \neq \varepsilon$, i.e., there is a nontrivial common prefix, replace all the A productions $A \rightarrow \alpha\beta_1 | \alpha\beta_2 | \dots | \alpha\beta_n | \gamma$, where γ represents all alternatives that do not begin with α by

$$A \rightarrow \alpha A' | \gamma$$

$$A' \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$$

Here, A' is a new non-terminal. Repeatedly apply this transformation until no two alternatives for a non-terminal have a common prefix.

2.5 Context-sensitive Grammar

A context-sensitive grammar (CSG) is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and non-terminal symbols. Context-sensitive grammars are more general than context-free grammar but still orderly enough to be parsed by a linear bounded automation.

The concept of context-sensitive grammar was introduced by Noam Chomsky in the 1950s as a way to describe the syntax of natural language where it is indeed often the case that a word may or may not be appropriate in a certain place depending upon the context. A formal language that can be described by a context-sensitive grammar is called a context-sensitive language.

A formal grammar $G = (N, \Sigma, P, S)$ is context-sensitive if all rules in P are of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

where $A \in N$ (i.e., A is a single non-terminal), $\alpha, \beta \in (N \cup \Sigma)^*$ (i.e., α and β are strings of non-terminals and terminals) and $\gamma \in (N \cup \Sigma)^+$ (i.e., γ is a nonempty string of non-terminals and terminals).

In addition, a rule of the form,

$$S \rightarrow \lambda \text{ provided } S \text{ does not appear on the right side of any rule}$$

where λ represents the empty string is permitted. The addition of the empty string allows the statement that the context sensitive languages are a proper superset of the context free languages, rather than having to make the weaker statement that all context free grammars with no $\rightarrow \lambda$ productions are also context sensitive grammars.

The name “context-sensitive” is explained by the α and β that form the context of A and determine whether A can be replaced with γ or not. This is different from a context-free grammar where the context of a non-terminal is not taken into consideration.

If the possibility of adding the empty string to a language is added to the string s recognized by the non-contracting grammars (which can never include the empty string) then the languages in these two definitions are identical.

Every context-sensitive grammar which does not generate the empty string can be transformed into an equivalent one in Kuroda normal form. “Equivalent” here means that the two grammars generate the same language. The normal form will not in general be context-sensitive, but will be a non-contracting grammar.

The decision problem that asks whether a certain string s belongs to the language of a certain context-sensitive grammar G , is PSPACE-complete. There are even some context-sensitive grammars whose fixed grammar recognition problem is PSPACE-complete. The emptiness problem for context-sensitive grammars (given a context-sensitive grammar G , is $L(G) = \varepsilon$) is undecidable.

It has been shown that nearly all natural languages may in general be characterized by context-sensitive grammars, but the whole class of CSG’s seems to be much bigger than natural languages. Worse yet, since the aforementioned decision problem for CSG’s is PSPACE-complete [47], that makes them totally unworkable

for practical use, as a polynomial-time algorithm for a PSPACE-complete problem would imply $P=NP$. Ongoing research on computational linguistics has focused on formulating other classes of languages that are “mildly context-sensitive” whose decision problems are feasible, such as tree-adjoining grammars, combinatory categorical grammars, coupled context-free languages, and linear context-free rewriting systems. The languages generated by these formalisms properly lie between the context-free and context-sensitive languages.

2.6 Top-down Parsing

Top-down parsing is a strategy of analyzing unknown data relationships by hypothesizing general parse tree structures and then considering whether the known fundamental structures are compatible with the hypothesis. It occurs in the analysis of both natural languages and computer languages.

Top-down parsing can be viewed as an attempt to find left-most derivations of an input-string by searching for parse-trees using a top-down expansion of the given formal grammar rules. Tokens are consumed from left to right. Inclusive choice is used to accommodate ambiguity by expanding all alternative right-hand-sides of grammar rules [16].

Simple implementations of top-down parsing do not terminate for left-recursive grammars, and top-down parsing with backtracking may have exponential time complexity with respect to the length of the input for ambiguous CFGs [34]. However, more sophisticated top-down parsers have been created by Frost, hafiz, and Callaghan [45][46], which do accommodate ambiguity and left recursion in polynomial time and which generate polynomial-sized representations of the potentially-exponential number of parse trees.

An LL parser, also called a top-bottom parser or top-down parser, applied each production rule to the incoming symbols by working from the left-most symbol yielded on a production rule and then proceeding to the next production rule for each

non-terminal symbol encountered. In this way the parsing starts on the left of the result side of the production rule and evaluates non-terminals from the left first and, thus, proceeds down the parse tree for each new non-terminal before continuing to the next symbol for a production rule.

For example:

$$A \rightarrow aBC$$

$$B \rightarrow c | cd$$

$$C \rightarrow df | eg$$

would match $A \rightarrow aBC$ and attempt to match $B \rightarrow c | cd$ next. Then $C \rightarrow df | eg$ would be tried. As one may expect, some languages are more ambiguous than others. For a non-ambiguous language in which all productions for a non-terminal produce distinct strings: the string produced by one production will not start with the same symbol as the string produced by another production. A non-ambiguous language may be parsed by an LL(1) grammar where the (1) signifies the parser reads ahead one token at a time. For an ambiguous language to be parsed by an LL parser, the parser must look ahead more than 1 symbol, e.g. LL(3). The common solution is to use an LR parser (also known as bottom-up or shift-reduce parser).

When a top-down parser tried to parse an ambiguous input w.r.t. an ambiguous CFG, it may need an exponential number of steps (w.r.t. the length of the input) to try all alternatives of the CFG in order to produce all possible parse trees, which eventually would require exponential memory space. The problem of exponential time complexity in top-down parsers constructed as sets of mutually-recursive functions has been solved by Norvig in 1991 [48]. His technique is similar to the use of dynamic programming and state-sets in Earley's algorithm, and tables in the CYK algorithm of Cocke, Younger and Kasami. The key idea is to store results of applying a parser p at position j in a memo table and to reuse the result whenever the same situation arises. Frost, Hafiz and Callaghan [45][46] also use memorization for refraining from redundant computations to accommodate any form of CFG in polynomial time ($\Theta(n^4)$ for left-recursive grammars and $\Theta(n^3)$ for non-left-recursive grammars). Their top-down parsing algorithm also requires polynomial space for

potentially exponential ambiguous parse trees by “compact representation” and “local ambiguities grouping”. Their compact representation is comparable with Tomita’s compact representation of bottom-up parsing [49].

The most common parsers using top-down parsing methodology are LL parser and recursive descent parser.

2.6.1 LL Parser

An LL parser is a top-down parser for a subset of the context-free grammars. It parses the input from left to right, and constructs a leftmost derivation of the sentence. The class of grammar which are parsable in this way is known as the LL grammars.

An LL parser is called an $LL(k)$ parser if it uses k tokens of look-ahead when parsing a sentence. If such a parser exists for a certain grammar and it can parse sentences of this grammar without backtracking then it is called an $LL(k)$ grammar. Of these grammars, $LL(1)$ grammars, although fairly restrictive, are very popular because the corresponding LL parsers only need to look at the next token to make their parsing decisions. Languages based on grammars with a high value of k require considerable effort to parse.

A grammar whose parsing table has no multiply-defined entries is said to be $LL(1)$. This uses one input symbol of lookahead at each step to make parsing action decisions. $LL(1)$ grammar G produces only one entry in parsing table of a non-recursive predictive parser.

$LL(1)$ grammar have several distinctive properties. No ambiguous or left-recursive grammar can be $LL(1)$. It can also be shown that a grammar G is $LL(1)$ if and only if whenever $A \rightarrow \alpha | \beta$ are two distinct productions of G the following conditions hold:

1. For no terminal a do both α and β derive strings beginning with a .
2. At most one of α and β can derive the empty string.
3. If $\beta \xRightarrow{*} \varepsilon$, then α does not derive any string beginning with a terminal in $\text{Follow}(A)$.

There remains the question of what should be done when a parsing table has multiply-defined entries. One recourse is to transform the grammar by eliminating all left recursion and then left factoring whenever possible, hoping to produce a grammar for which the parsing table has no multiply-defined entries. The main difficulty in using predictive parsing is in writing a grammar for the source language such that a predictive parser can be constructed from the grammar. Although left-recursion elimination and left factoring are easy to do, they make the resulting grammar hard to read and difficult to use for translation purposes.

We have used LL parser using LL(1) grammar in this thesis. Because we have designed Bangla grammar in such way that Bangla sentences can be parsed using LL parser efficiently. Not only this, error recovery of Bangla parsing will be effective using this method.

2.6.2 Recursive Descent Parser

A recursive descent parser is a top-down parser build from a set of mutually-recursive procedures (or a non-recursive equivalent) where each such procedure usually implements one of the production rules of the grammar. Thus the structure of the resulting program closely mirrors that of the grammar it recognizes.

A predictive parser is a recursive descent parser that does not require backtracking. Predictive parsing is possible only for the class of $\text{LL}(k)$ grammars, which are the context-free grammars for which there exists some positive integer k that allows a recursive descent parser to decide which production to use by examining only the next k tokens of input. The $\text{LL}(k)$ grammars therefore exclude all ambiguous

grammars, as well as all grammars that contain left recursion. Any context-free grammar can be transformed into an equivalent grammar that has no left recursion, but removal of left recursion does not always yield an $LL(k)$ grammar. A predictive parser runs in linear time.

Recursive descent with backup is a technique that determine which production to use by trying each production in turn. Recursive descent with backup is not limited to $LL(k)$ grammars, but is not guaranteed to terminate unless the grammar is $LL(k)$. Even when they terminate parsers that use recursive descent with backup may require exponential time.

Although predictive parsers are widely used, programmer often prefer to create LR or LALR parsers via parser generators without transforming the grammar into $LL(k)$ form.

Some authors define recursive descent parsers as the predictive parsers. Other authors use the term more broadly, to include backed-up recursive descent.

2.7 Bottom-up Parsing

Bottom-up parsing is a strategy for analyzing unknown data relationships that attempts to identify the most fundamental units first, and then to infer higher-order structures from them. It attempts to build trees upward toward the start symbol. It occurs in the analysis of both natural languages and computer languages.

In linguistics, an example of bottom-up parsing would be analyzing a sentence by identifying words first, and then using properties of the words to infer grammatical relations and phrase structures to build a parse tree of the complete sentence. This means that rather than beginning with the starting symbol and generating an input string, we shall examine the string and attempts to work our way back to the starting symbol. We can gain some power by starting at the bottom and working our way up.

The common parsers using bottom-up parsing are LR parser, LALR parser, shift-reduce parser.

2.7.1 LR Parser

An LR parser is a parser for context-free grammars that reads input from left to right and produces a rightmost derivation. The term $LR(k)$ parser is also used; here the k refers to the number of unconsumed “look ahead” input symbols that are used in making parsing decisions. Usually k is 1 and is often omitted. A context-free grammar is called $LR(k)$ if there exists an $LR(k)$ parser for it. An LR parser is said to perform bottom-up parsing because it attempts to deduce the top level grammar productions by building up from the leaves.

In typical usage an “LR parser” means a particular parser capable of recognizing a particular language specified by a context free grammar. It is common, however, to use the term to mean a driver program that can be supplied with a suitable table to produce a wide number of different particular LR parsers. However, these programs are more accurately called parser generators.

LR parsers can be implemented very efficiently. Of all parsers that scan their input left to right, LR parsers detect syntactic errors (that is, when the input does not conform to the grammar) as soon as possible.

LR parsers are difficult to produce by hand; they are usually constructed by a parser generator. Depending on how the parsing table is generated these parser are called simple LR parser (SLR), look-ahead LR parser (LALR), and canonical LR parser. These types of parsers can deal with increasingly large sets of grammars; LALR parsers can deal with more grammars than SLR. Canonical LR parsers work on more grammars than LALR parsers.

Conceptually, an LR parser is a recursive program that can be proven correct by direct computation, and can be implemented efficiently as a recursive ascent parser,

a set of mutually-recursive functions for every grammar, much like a recursive descent parser. Conventionally, however, LR parsers are presented and implemented as table-based stack machines in which the call stack of the underlying recursive program is explicitly manipulated.

2.7.2 LALR Parser

A lookahead LR parser or LALR parser is a specialized form of LR parser that can deal with more context-free grammars than simple LR (SLR) parsers. It is a very popular type of parser because it gives a good trade-off between the number of grammars it can deal with and the size of the parsing tables it requires.

Like SLR, LALR is a refinement to the technique for constructing LR(0) parse tables. While SLR uses *Follow* sets to construct reduce actions, LALR uses *lookahead* sets are specific to an LR(0) item and a parser state.

Specially, the *Follow* set for a given LR(0) item I in a given parser state S contains all symbols that are allowed by the grammar to appear after I 's left-hand-side non-terminal. In contrast, the *lookahead* set for item I in state S , $Follow(I)$ is effectively the union of the *lookahead* sets for all LR(0) items with the same left-hand-side as I , regardless of parser states or right-hand-sides, therefore losing all context information. Because the *lookahead* set is specific to a particular parsing context, it can be more selective, therefore allowing finer distinctions than the *Follow* set.

2.7.3 Shift-reduce Parser

The most common bottom-up parsers are the shift-reduce parsers [50]. These parsers examine the input tokens and either shift (push) them onto a stack or reduce elements at the top of the stack, replacing a right-hand side by a left-hand side.

A shift-reduce parser uses a stack to hold the grammar symbols while awaiting reduction. During the operation of the parser, symbols from the input are shifted

onto stack. If a prefix of the symbols on top of the stack matches the RHS of a grammar rule which is the correct rule to use within the current context, then the parser reduces the RHS of the rule to its LHS, replacing the RHS symbols on top of the stack with the non-terminal occurring on the LHS of the rule. This shift-reduce process continues until the parser terminates, reporting either success or failure. It terminates with success when the input is legal and is accepted by the parser. It terminates with failure if an error is detected in the input.

The parser is a stack automaton which is in one of several discrete states. In reality, the parse stack contains states, rather than grammar symbols. However, since each state corresponds to a unique grammar symbol, the state stack can be mapped onto the grammar symbol stack mentioned earlier.

2.8 Architecture of Non-recursive Predictive Parser

Non-recursive parsing can be done by maintaining a stack. A model of a non-recursive predictive parser is demonstrated in figure 2.7. The basic idea is looking up the production to be applied in a parsing table. Such table can be constructed directly from grammar of language.

A table-driven predictive parser has an input buffer, a stack, a parsing table, and an output stream. The input buffer contains the string to be parsed, followed by \$, a symbol used as a right end-marker to indicate the end of the input string. The stack contains a sequence of grammar symbols with \$ on the bottom, indicating the bottom of the stack. Initially, the stack contains the start symbol of the grammar on the top of \$. The parsing table is a two-dimensional array $M[A, a]$, where A is a non-terminal, and a is a terminal or the symbol \$.

The parser is controlled by a program that behaves that behaves as follows. The program considers X , the symbol on top of the stack, and a , the current input symbol. These two symbols determine the action of the parser. There are three possibilities.

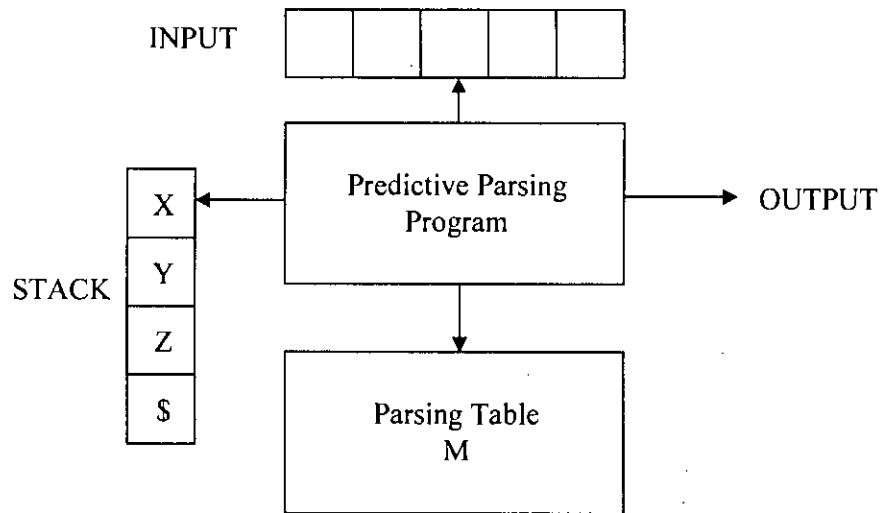


Figure 2.7: A model of non-recursive predictive parser.

1. If $X = a = \$$, the parser halts and announces successful completion of parsing.
2. If $X = a \neq \$$, the parser pops X off the stack and advances the input pointer to the next input symbol.
3. If X is a non-terminal, the program consults entry $M[X, a]$ of the parsing table M . This entry will be either an X -production of the grammar or an error entry. If, for example, $M[X, a] = \{ X \rightarrow UVW \}$, the parser replaces X on top of the stack by WVU (with U on top). As output, we shall assume that the parser just prints the production used; any other code could be executed here. If $M[X, a] = \text{error}$, the parser calls an error recovery routine.

Parsing algorithm for non-recursive predictive parser is described as follows:

Algorithm 2.3: Non-recursive predictive parsing.

Input. A string w and a parsing table M for Grammar G .

Output. If w is in $L(G)$, a leftmost derivation of w ; otherwise, an error indication.

Method. Initially, the parser is in a configuration in which it has $\$S$ on the stack with S , the start symbol of G on top, and $w\$$ in the input buffer. The program that utilizes the predictive parsing table M to produce a parse for given input.

set ip to point to the first symbol of $w\$$

repeat

 let X be the top stack symbol and a the symbol pointed to by ip

```

if  $X$  is a terminal or $ then
    if  $X = a$  then
        pop  $X$  from the stack and advance  $ip$ 
    else
        error( )
    end if
else //  $X$  is a non-terminal
    if  $M[X, a] = X \rightarrow Y_1Y_2\dots Y_k$  then
        pop  $X$  from the stack
        push  $Y_k, Y_{k-1}, \dots, Y_1$  onto the stack ( $Y_1$  on top)
        output the production  $X \rightarrow Y_1Y_2\dots Y_k$ 
    else
        error( )
    end if
end if
until  $X = \$$  // stack is empty

```

Parsing algorithm for non-recursive predictive parser used two important terms called *First* and *Follow* to build parsing table, which will be discussed in the next two sections (2.8.1 and 2.8.2).

2.8.1 First

$First(X)$ is defined as the set of terminals that begin the strings derived from X , where X denotes any string of grammar symbols. To compute $First(X)$ for all grammar symbols X , the following rules are applied until no more terminals or ϵ can be added to the $First(X)$ set.

1. If X is terminal, then $First(X) = \{X\}$.
2. If $X \rightarrow \epsilon$ is a production, then ϵ is added to $First(X)$.
3. If X is non-terminal and $X \rightarrow Y_1Y_2\dots Y_k$ is a production, then a is added to $First(X)$ if for some i , a is in $First(Y_i)$, and ϵ is in all of $First(Y_1), \dots, First(Y_{i-1})$; that is, $Y_1\dots Y_{i-1} \xRightarrow{*} \epsilon$. If ϵ is in $First(Y_j)$ for all $j = 1, 2, \dots, k$, then ϵ is added to $First(X)$.

Now, we can compute $First$ for any string $X_1X_2 \dots X_n$ as follows. All the non- ϵ symbols of $First(X_1)$ is added to $First(X_1X_2 \dots X_n)$. All non- ϵ symbols of $First(X_2)$

is added to $First(X_1X_2 \dots X_n)$, if ε is in $First(X_1)$. All non- ε symbols of $First(X_3)$ is added to $First(X_1X_2 \dots X_n)$, if ε is in both $First(X_1)$ and $First(X_2)$. In general, all non- ε symbols of $First(X_i)$ is added to $First(X_1X_2 \dots X_n)$, if ε is in each of $First(X_1), First(X_2), \dots, First(X_{i-1})$. Finally, ε is added to $First(X_1X_2 \dots X_n)$, if for all i , $First(X_i)$ contains ε .

For example, there is a context-free grammar as follows:

$$\begin{aligned} S &\rightarrow NP VP \\ NP &\rightarrow N | P \\ VP &\rightarrow VB C1 \\ C1 &\rightarrow N | \varepsilon \end{aligned}$$

where, S indicates sentence, NP indicates noun phrase, VP indicates verb phrase, N indicates noun, P indicates pronoun, VB indicates verb and C1 indicates a common factor. By the definition of *First*, we can calculate the *First* of the non-terminals as follows:

$$\begin{aligned} First(S) &= \{ N, P \} \\ First(NP) &= \{ N, P \} \\ First(VP) &= \{ VB \} \\ First(C1) &= \{ N, \varepsilon \} \end{aligned}$$

2.8.2 Follow

$Follow(A)$, for non-terminal A , is defined as the set of terminals a that can appear immediately to the right of A in some sentential form, that is, the set of terminals a such that there exists a derivation of the form $S \xRightarrow{*} \alpha A a \beta$ for some α and β . If A can be the rightmost symbol in some sentential form, then $\$$ is in $Follow(A)$. To compute $Follow(A)$ for all non-terminals A , the following rules are applied until nothing can be added to any *Follow* set.

1. $\$$ is placed in $Follow(S)$, where S is the start symbol and $\$$ is the input right endmarker.

2. If there is a production $A \rightarrow \alpha B \beta$, then everything in $First(\beta)$ except for ϵ is placed in $Follow(B)$.
3. If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B \beta$ where $First(\beta)$ contains ϵ (i.e., $\beta \Rightarrow^* \epsilon$), then everything in $Follow(A)$ is in $Follow(B)$. In this case, we can say $Follow(A) \subseteq Follow(B)$.

For example, according to the grammar mentioned in the previous section (2.8.1), using the definition of $Follow$ we can calculate $Follow$ of the non-terminals as follows:

$$Follow(S) = \{ \$ \}$$

$$Follow(NP) = \{ VB \}$$

$$Follow(VP) = \{ \$ \}$$

$$Follow(CI) = \{ \$ \}$$

2.8.3 Construction of Predictive Parsing Tables

The following algorithm can be used to construct a predictive parsing table for a grammar G . The idea behind the algorithm is the following. Suppose $A \rightarrow \alpha$ is a production with a in $First(\alpha)$. Then, the parser will expand A by α when the current input symbol is a . The only complication occurs when $\alpha = \epsilon$ or $\alpha \Rightarrow^* \epsilon$. In this case, we should again expand A by α if the current input symbol is in $Follow(A)$, or if the $\$$ on the input has been reached and $\$$ is in $Follow(A)$.

Algorithm 2.4: Construction of a predictive parsing table.

Input. Grammar G .

Output. Parsing table M .

Method.

1. For each production $A \rightarrow \alpha$ of the grammar, do steps 2 and 3.
2. For each terminal a in $First(\alpha)$, add $A \rightarrow \alpha$ to $M[A, a]$.
3. If ϵ is in $First(\alpha)$, add $A \rightarrow \alpha$ to $M[A, a]$ for each terminal b in $Follow(A)$.
If ϵ is in $First(\alpha)$ and $\$$ is in $Follow(A)$, add $A \rightarrow \alpha$ to $M[A, \$]$.
4. Make each undefined entry of M to be error.

For example, according to the grammar mentioned in section 2.8.1, we can construct parsing table of a predictive parser using algorithm 2.4, which is shown in table 2.1.

Table 2.1: Parsing table of a predictive parser.

	N	P	VB	\$
S	$S \rightarrow NP VP$	$S \rightarrow NP VP$		
NP	$NP \rightarrow N$	$NP \rightarrow P$		
VP			$VP \rightarrow VB Cl$	
Cl	$Cl \rightarrow N$			$Cl \rightarrow \epsilon$

2.8.4 Error Recovery in Predictive Parsing

The stack of a non-recursive predictive parser makes explicit the terminals and non-terminals that the parser hopes to match with the remainder of the input. We shall therefore refer to symbols on the parser stack in the following discussion. An error is detected during predictive parsing when the terminal on top of the stack does not match the next input symbol or when non-terminal A is on top of the stack, a is the next input symbol, and the parsing table entry $M[A, a]$ is empty.

Panic mode error recovery is based on the idea of skipping symbols on the input until a token is a selected set of synchronizing tokens appears. Its effectiveness depends on the choice of synchronizing set. The sets should be chosen so that the parser recovers quickly from errors that are likely to occur in practice. Some heuristics are as follows:

1. As a starting point, we can place all symbols in $Follow(A)$ into the synchronizing set for non-terminal A . If we skip tokens until an element of $Follow(A)$ is seen and pop A from the stack, it is likely that parsing can continue.
2. It is not enough to use $Follow(A)$ as the synchronizing set for A . For example, if full-stop(.) terminate sentences in English language, then terminals that begin sentences may not appear in the Follow set of the non-terminal generating expressions. A missing full-stop after a sentence may

therefore result in the terminal beginning the next statement being skipped. Often, there is a hierarchical structure on constructs in a language; e.g., expressions appear within statements, which appear within blocks, and so on. We can add to the synchronizing set of a lower construct the symbols that begin higher constructs. For example, we might add terminals that begin sentences to the synchronizing sets for the non-terminal generating expressions.

3. If we add symbols in $First(A)$ to the synchronizing set for non-terminal A , then it may be possible to resume parsing according to A if a symbol in $First(A)$ appears in the input.
4. If a non-terminal can generate the empty string, then the production deriving ϵ can be used as a default. Doing so may postpone some error detection, but cannot cause an error to be missed. This approach reduces the number of non-terminals that have to be considered during error recovery.
5. If a terminal on top of the stack cannot be matched, a simple idea is to pop the terminal, issue a message saying that the terminal was inserted, and continue parsing. In effect, this approach takes the synchronizing set of a token to consist of all other tokens.

With the idea of error recovery, we can reconstruct the parsing table (table 2.1) as table 2.2.

Table 2.2: Parsing table of a predictive parser with error recovery information.

	N	P	VB	\$
S	$S \rightarrow NP VP$	$S \rightarrow NP VP$		sync
NP	$NP \rightarrow N$	$NP \rightarrow P$	sync	
VP			$VP \rightarrow VB C1$	sync
C1	$C1 \rightarrow N$			$C1 \rightarrow \epsilon$

2.9 Remarks

In this thesis, top-down parsing algorithm will be used to design predictive parser for Bangla natural language. To design such a parser, ambiguity elimination is

important, as discussed in section 2.4.11.1. Then, left factoring technique, discussed in section 2.4.13, is important to achieve predictive parsing. This thesis will apply the architecture of predictive parser, as discussed in section 2.8. Parsing table construction will adopt the theory discussed in section 2.8.3. And, finally, error recovery mechanism will be applied to the parser according to the theory discussed in section 2.8.4.

3.1 Bangla Grammar

In syntax-based machine translation, parsing of source text is the task of first step. For parsing of an input text, a grammar is required. Therefore, grammar of source language is very important. A context-free grammar is capable of parsing all correct sentences of source language, if structure of all types of sentences is reflected in grammar of source language.

Context-free grammar of Bangla language is already developed [22][23]. In this chapter, we will discuss present context-free grammar of Bangla natural language.

3.2 Phrases in Bangla Grammar

In any language, any sentence can be sub-divided into smaller parts. These smaller parts are referred to as phrases. Like most of the languages of the world, Bangla natural language sentences contain three types of phrases. They are:

1. Noun phrase,
2. Verb phrase,
3. Adjective phrase.

Some other types of phrases may be present in other languages. But these three phrases are most important phrases in most of the languages.

3.2.1 Noun Phrase

Noun phrase is a kind of phrase whose head is a noun or pronoun, and optionally accompanied by a set of modifiers. In Bangla language, possible modifiers include

determiners, demonstrators, adjectives, quantifiers, bivoktis etc. Noun phrase is normally denoted by NP.

For example, “আমি” (ami), “আমার ভাই” (amar vai), “আমার ছোট ভাই” (amar chhOTO vai), “আমার বড় চাচার ছোট ছেলে” (amar boRo chachar chhOTO chhele) etc, are noun phrases.

3.2.2 Verb phrase

A verb phrase is headed by a verb. It may be constructed from a single verb. In other cases, it consists of the main verb and any auxiliary verbs, plus optional specifiers, complements, adjuncts etc. Verb phrase is normally denoted by VP.

For example, “আমি খাই” (ami khai), “আমি ভাত খাই” (ami vat khai), “আমি ভাত ও ডাল খাই” (ami vat O Dal khai) etc, are verb phrases.

3.2.3 Adjective phrase

An adjective phrase is a phrase with an adjective as its head. Adjectival phrases may occur as pre or post-modifiers to a noun, or as predicatives (predicate adjectives) of a verb. Adjective phrase is normally denoted by AP.

For example, “রবিন ভাল ছেলে” (robin valo chhele), “রবিন খুব ভাল ছেলে” (robin khub valo chhele), “সে খুব দ্রুত কাজ করতে পারে” (se khub druto kaj koroTe pare) etc, are adjective phrases.

3.3 Context-free Grammar for Noun Phrase

Context-free grammar for noun phrase in Bangla natural language is as follows:

NP → N (DET)

NP → (DEMO) (SPR) (AP) NP

NP → NP Biv (NP)

NP → N PM

SPR → QFR (PP)

DEMO → (DD) (DO)

In the rule $NP \rightarrow N (DET)$, a noun phrase (NP) consists of a noun (N), which is optionally followed by a determiner (DET). Determiner is placed to the right of noun. A determiner is a noun modifier that expresses the reference of a noun or noun phrase. In Bangla, “টি” (Ti) etc, are determiner. We can give example of sentence segment using this rule as, “ছেলে” (chhele), “ছেলেটি” (chheleTi) etc. Tree structure of the derivations is presented as figure 3.1.

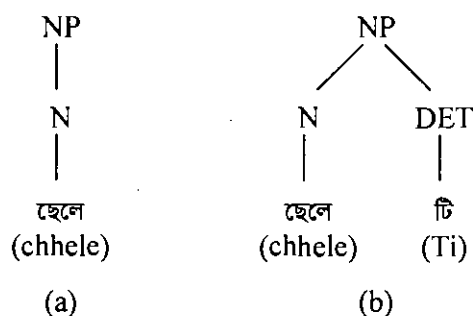


Figure 3.1: Tree structure for the rules $NP \rightarrow N$ (a), $NP \rightarrow N DET$ (b).

In the rule $NP \rightarrow (DEMO) (SPR) (AP) NP$, a noun phrase (NP) consists of a demonstrator (DEMO), a specifier (SPR), an adjective phrase (AP) and a noun phrase (NP). Here, placement of demonstrator, specifier and adjective phrase is optional. Example using this rule will be given later after stating some other rules.

In the rule $NP \rightarrow NP Biv (NP)$, bivokti (Biv) may appear after a noun phrase (NP) and optionally followed by another noun phrase (NP). Bivokti is a Bangla word, which is another type of modifier used after noun or noun phrase. In Bangla, “কে” (ke), “এর” (er) etc, are bivokti. We can give example of sentence segment using this rule as, “আমাকে” (amake), “ছেলেটিকে” (chheleTike), “ছেলেটির বই” (chheleTir boi) etc. Tree structure of the derivations is presented as figure 3.2.

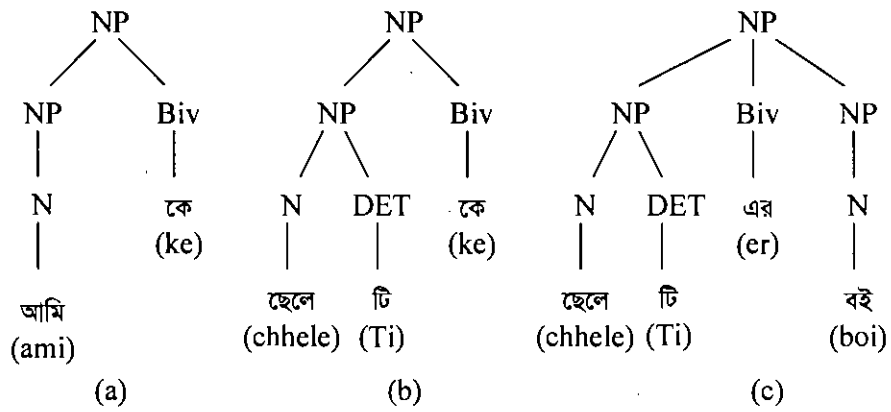


Figure 3.2: Tree structure for the rules $NP \rightarrow NP \text{ Biv}$ (a) (b), $NP \rightarrow NP \text{ Biv NP}$ (c).

In the rule $NP \rightarrow N \text{ PM}$, a noun phrase (NP) consists of a noun (N) and a plural marker (PM). Plural marker is used as suffix to a word, which modify meaning of the word as plural. In Bangla, “রা” (ra), “এরা” (era), “গুলো” (gulO), “দের” (der) etc, are plural marker. We can give example of sentence segment using this rule as, “ছেলেরা” (chhelera), “বোনেরা” (bOnera), “আমগুলো” (amgulO), “আমাদের” (amader) etc. Tree structure of the derivations is presented as figure 3.3.

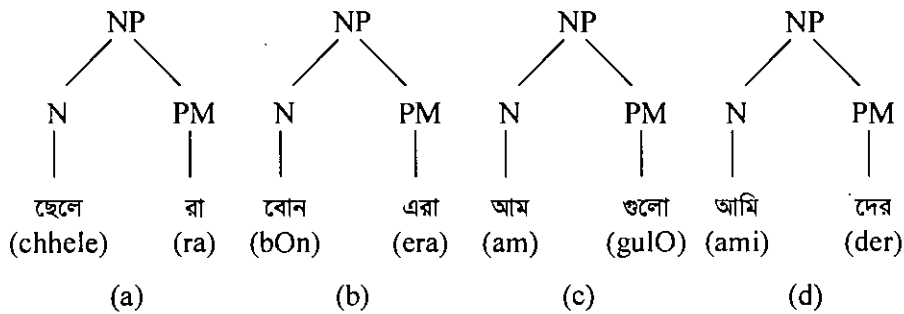


Figure 3.3: Tree structure for the rules $NP \rightarrow N \text{ PM}$ (a) (b) (c) (d).

In the rule $SPR \rightarrow QFR \text{ (PP)}$, a specifier (SPR) consists of a quantifier (QFR) and is optionally followed by a post preposition (PP). Numerical words, like “এক” (ek), “দুই” (dui), “তিন” (tin) etc, are quatifier. There are also some collective quantifiers, like “অনেক” (onek), “সব” (sob), “কয়েক” (koyek) etc. We can give example of sentence segment using this rule as, “একটি” (ekoTi), “অনেক” (onek) etc. Tree structure of the derivations is presented as figure 3.4.

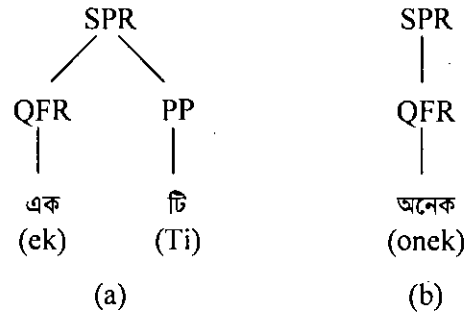


Figure 3.4: Tree structure for the rules $SPR \rightarrow QFR PP$ (a), $SPR \rightarrow QFR$ (b).

In the rule $DEMO \rightarrow (DD) (DO)$, a demonstrator (DEMO) consists of a demonstrative deictic (DD) and a demonstrative ordinal (DO). Here, placement of both demonstrative deictic and demonstrative ordinal is optional. In Bangla, “এই” (ei), “এ” (OI) etc, are demonstrative deictic. And, “প্রথম” (prothom), “দ্বিতীয়” (dwtiyo) etc, are demonstrative ordinal.

Now, we can give example of sentence segment using the rule $NP \rightarrow (DEMO) (SPR) (AP) NP$ as, “এ প্রথম ছেলেটি” (OI prothom chheleTi), “একটি ছেলে” (ekoTi chhele), “সুন্দর ছেলেটি” (sundor chheleTi), “এ একটি ছেলে” (OI ekoTi chhele), “এ সুন্দর ছেলেটি” (OI sundor chheleTi), “একটি সুন্দর ছেলে” (ekoTi sundor chhele), “এ প্রথম একটি সুন্দর ছেলে” (OI prothom ekoTi sundor chhele) etc. Tree structure of the derivations is presented as figure 3.5.

Main difficulty of using this grammar is its ambiguous nature. The rule $NP \rightarrow NP Biv (NP)$ is ambiguous, as using this rule same sentence segment can be parsed in different ways i.e., different parsing tree can be generated. Again these rules do not support predictive parsing, as when a word of a sentence is to be parsed, it is not possible to decide which rule is applicable instantly. For example, when a noun word is found, there are several eligible rules for parsing like, $NP \rightarrow N (DET)$, $NP \rightarrow NP Biv (NP)$, $NP \rightarrow N PM$.

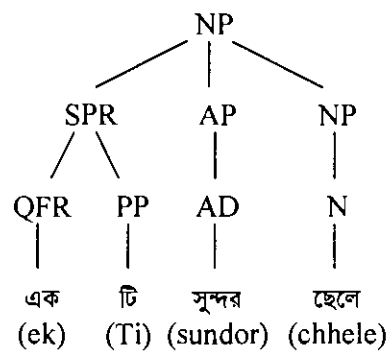
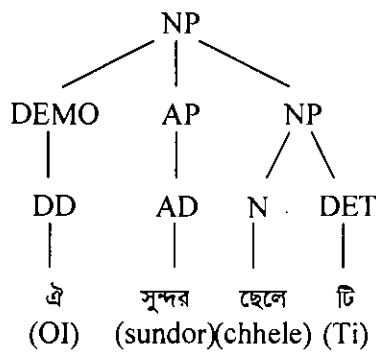
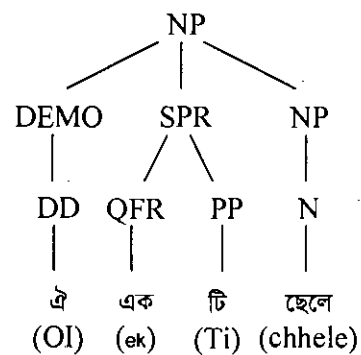
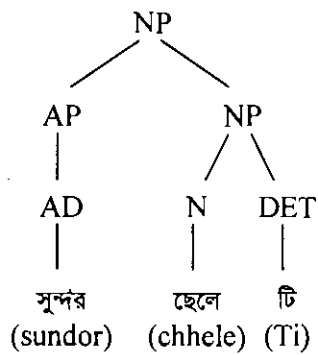
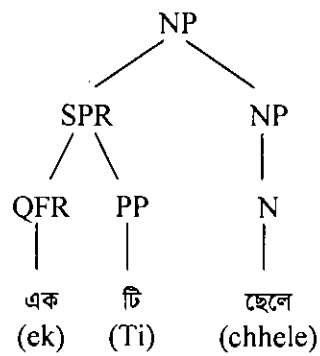
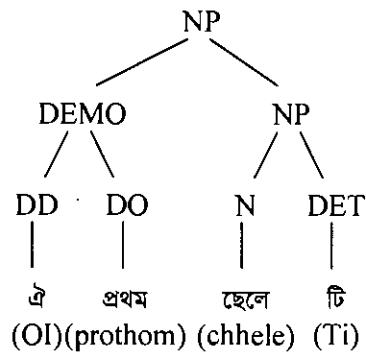
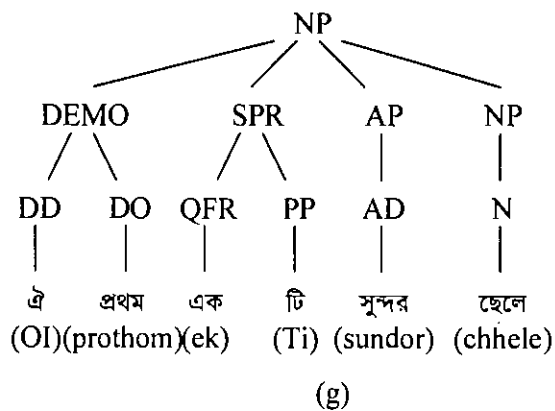


Figure 3.5: Tree structure for the rule $NP \rightarrow (DEMO) (SPR) (AP) NP$, which cover the variations $NP \rightarrow DEMO NP$ (a), $NP \rightarrow SPR NP$ (b), $NP \rightarrow AP NP$ (c), $NP \rightarrow DEMO SPR NP$ (d), $NP \rightarrow DEMO AP NP$ (e), $NP \rightarrow SPR AP NP$ (f) (continued).



Continued Figure 3.5: Tree structure for the rule $NP \rightarrow (DEMO) (SPR) (AP) NP$, which cover the variations $NP \rightarrow DEMO SPR AP NP$ (g).

3.4 Context-free Grammar for Verb Phrase

Context-free grammar for verb phrase in Bangla natural language is as follows:

$VP \rightarrow (NP) (AP) VF$

$VF \rightarrow VR AUX$

In the rule $VP \rightarrow (NP) (AP) VF$, a verb phrase (VP) consists of a noun phrase (NP), an adjective phrase (AP) and a verb form (VF). Here, placement of noun phrase and adjective phrase is optional.

In the rule $VF \rightarrow VR AUX$, a verb form (VF) consists of a verb root (VR) and an auxiliary (AUX).

We can give example of sentence segment using the rule $VP \rightarrow (NP) (AP) VF$ as, “সে খায়” (se khay), “সে ভাত খায়” (se vat khay), “সে বেশি খায়” (se beshi khay), “আমি তাকে ভালভাবে চিনি” (ami take valovabe chini) etc. Tree structure of the derivations is presented as figure 3.6.

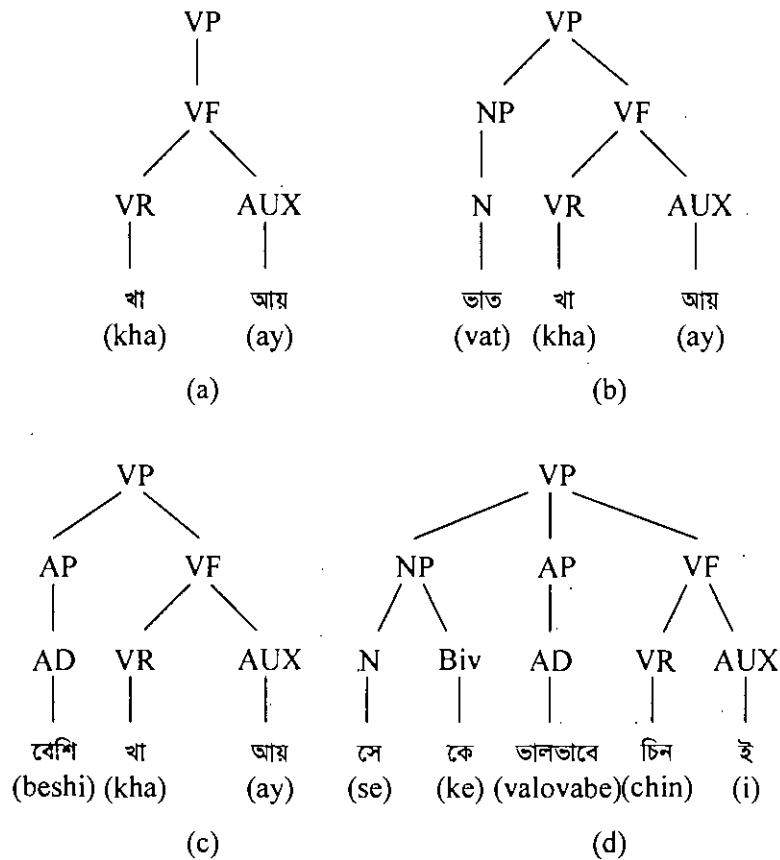


Figure 3.6: Tree structure for the rule $VP \rightarrow (NP) (AP) VF$, which cover the variations $VP \rightarrow VF$ (a), $VP \rightarrow NP VF$ (b), $VP \rightarrow AP VF$ (c), $VP \rightarrow NP AP VF$ (d).

This context-free grammar for verb phrase is ambiguous because of the rule $VP \rightarrow (NP) (AP) VF$. In a sentence noun phrase is followed by verb phrase. Therefore, actual noun phrase and noun phrase inside of verb phrase may allow a sentence to be parsed in different ways, making the rule ambiguous.

3.5 Context-free Grammar for Adjective Phrase

Context-free grammar for adjective phrase in Bangla natural language is presented as:

$$AP \rightarrow AD / ADs$$

According to this rule, adjective phrase (AD) consists of one or more adjectives.

We can give example of sentence segment using the rule as, “সে ভাল ছেলে” (se valo chhele), “সে খুব ভাল ছেলে” (se khub valo chhele) etc. Tree structure of the derivations is presented as figure 3.7.

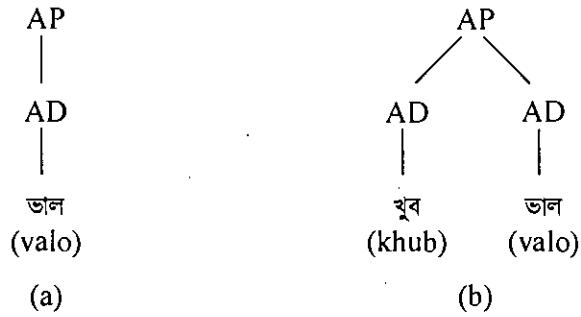


Figure 3.7: Tree structure for the rule $AP \rightarrow AD / ADs$.

This context-free grammar for adjective phrase is certainly ambiguous. When more than two adjectives appear together in a sentence, adjective phrase can be parsed in several ways.

3.6 Context-free Grammar for Simple Sentence

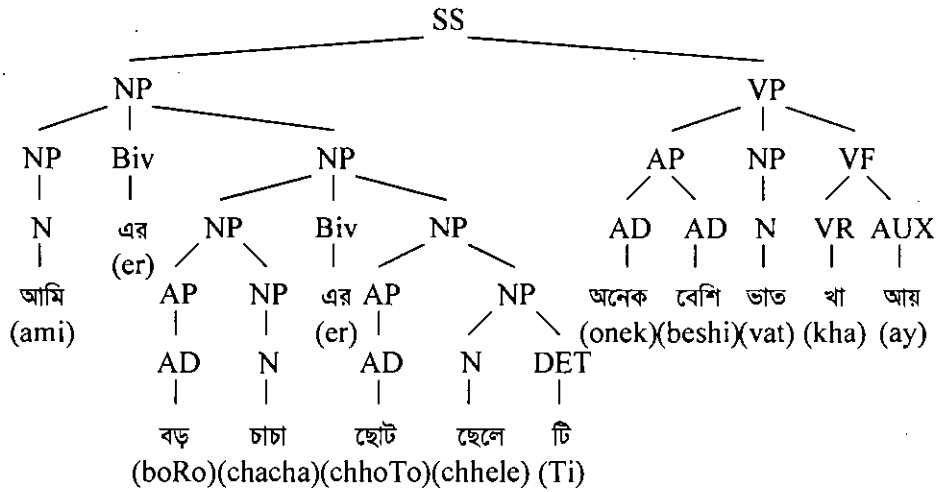
Context-free grammar for simple sentence in Bangla natural language is presented as:

$$SS \rightarrow NP VP$$

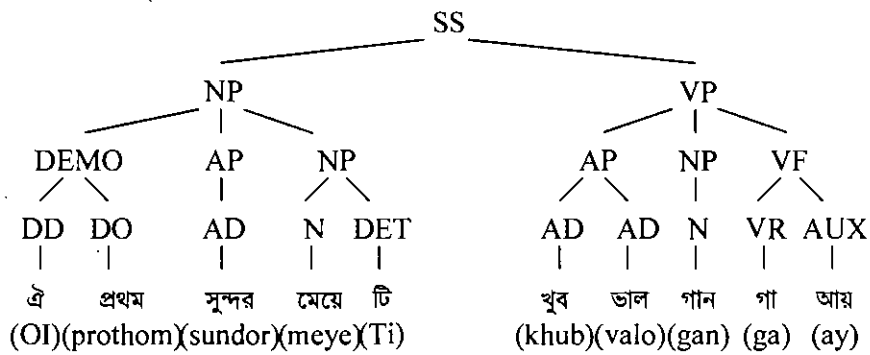
According to this rule, a simple sentence (SS) consists of a noun phrase (NP), followed by a verb phrase (VP).

We can give example of simple sentence as, “আমার বড় চাচার ছোট ছেলেটি অনেক বেশি ভাত খায়” (amar boRo chachar chhoTo chheleTi onek beshi vat khay), “ঐ প্রথম সুন্দর মেয়েটি খুব ভাল গান গায়” (OI prothom sundor meyeTi khub valo gan gay) etc. These sentences can be parsed using the rule $SS \rightarrow NP VP$. Tree structure of the derivations for the example sentences is presented in figure 3.8.

Grammar for simple sentence is certainly ambiguous, as simple sentence is composed of noun phrase (NP), and verb phrase (VP) and grammar for both phrases are ambiguous.



(a)



(b)

Figure 3.8: Tree structure for the rule $SS \rightarrow NP VP$.

3.7 Context-free Grammar for Complex Sentence

Context-free grammar for complex sentence in Bangla natural language is as follows:

$CS \rightarrow DC IC$

$CS \rightarrow IC DC$

$DC \rightarrow NP (SUBORD) VP$

$DC \rightarrow (SUBORD) SS$

IC → NP (SUBCOM) VP

IC → (SUBCOM) SS

If we analyze a complex sentence, we find a complex sentence is composed of a dependent clause or principle clause and an independent clause or subordinate clause. The meaning of dependent clause is dependent to the meaning of independent clause. These two clauses are optionally connected through a subordinate and a subordinate complement, where subordinate is a part of dependent clause and subordinate complement is a part of independent clause.

We can give example of complex sentence as, “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO). In the example, “যদি আমি যাই” (zodi ami zai) is dependent clause and “তবে তুমি এসো” (tobe tumi esO) is independent clause. Because, the meaning of sentence segment “তুমি এসো” (tumi esO) is independent. But the meaning of sentence segment “আমি যাই” (ami zai) is dependent to the meaning of “তুমি এসো” (tumi esO). In this example, “যদি” (zodi) is subordinate, which is part of dependent clause and “তবে” (tobe) is subordinate complement, which is part of independent clause. Some other examples of subordinate are, “যেখানে” (zekhane), “যে” (ze), “যাতে” (zate), “যেহেতু” (zehetu), “যখন” (zokhon) etc. Some other examples of subordinate complement are, “তাহলে” (tahole), “তবুও” (tobuO), “সেখানে” (sekhane), “তখন” (temon), “সেহেতু” (sehetu), “তখন” (tokhon) etc.

In the rule CS → DC IC, a complex sentence (CS) consists of a dependent clause (DC) followed by an independent clause (IC).

In the rule CS → IC DC, a complex sentence (CS) consists of an independent clause (IC) followed by a dependent clause (DC).

In the rule DC → NP (SUBORD) VP, a dependent clause (DC) consists of a noun phrase (NP), a subordinator (SUBORD) and a verb phrase (VP). Here, placement of subordinator is optional.

In the rule $DC \rightarrow (SUBORD) SS$, a dependent clause (DC) consists of a subordinator (SUBORD) and a simple sentence (SS). Here, placement of subordinator is optional.

In the rule $IC \rightarrow NP (SUBCOM) VP$, an independent clause (IC) consists of a noun phrase (NP), subordinator complement (SUBCOM) and a verb phrase (VP). Here, placement of subordinator complement is optional.

In the rule $IC \rightarrow (SUBCOM) SS$, an independent clause (IC) consists of a subordinator complement (SUBCOM) and a simple sentence (SS). Here, placement of subordinator complement is optional.

An example of complex sentence is, “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO). Same sentence can be written in some other variations like, “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai), “আমি গেলে তুমি এসো” (ami gele tumi esO), “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO), “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai), “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO) etc. Tree structure of the derivations for the examples is presented in figure 3.9.

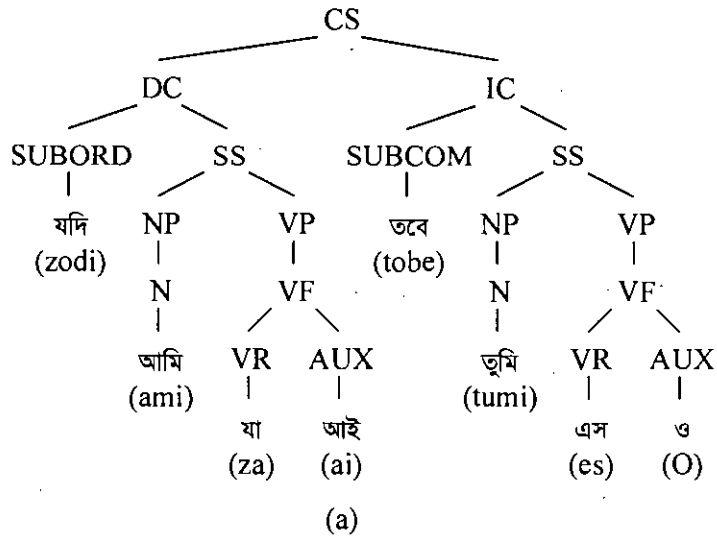
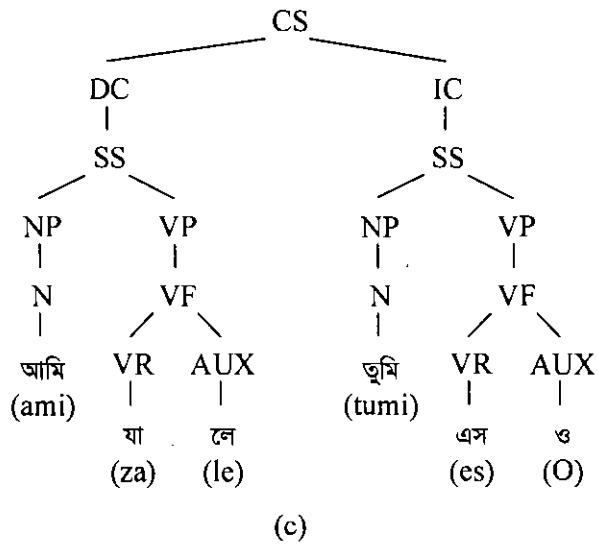
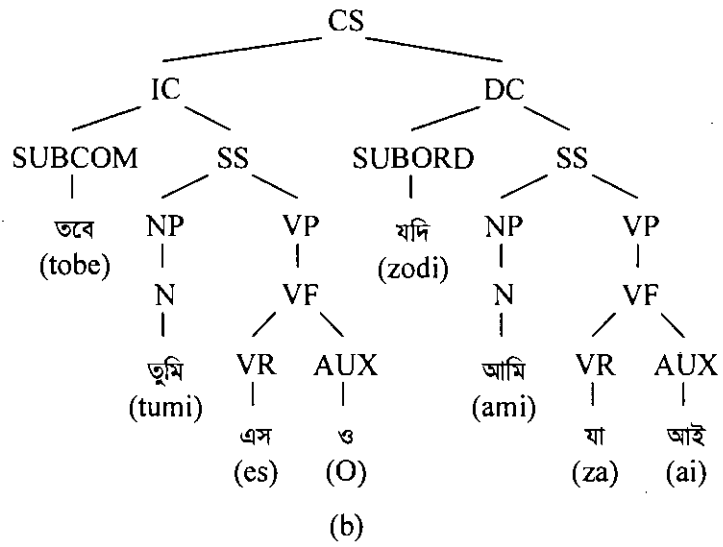
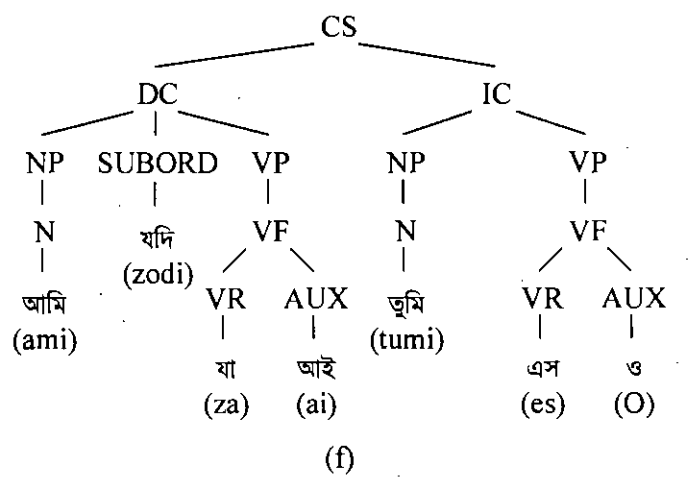
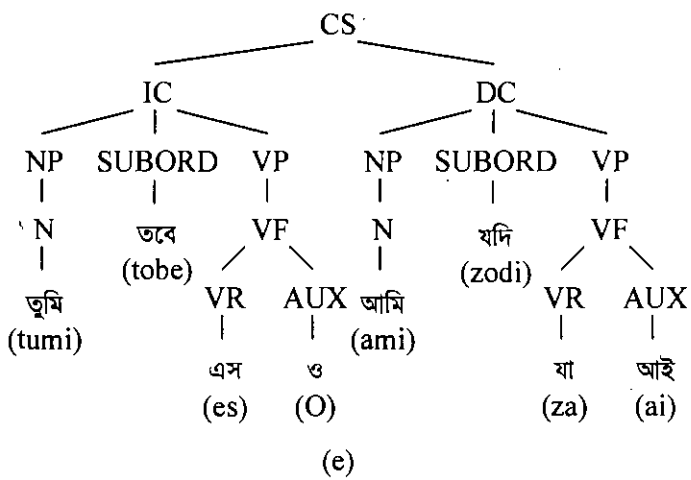
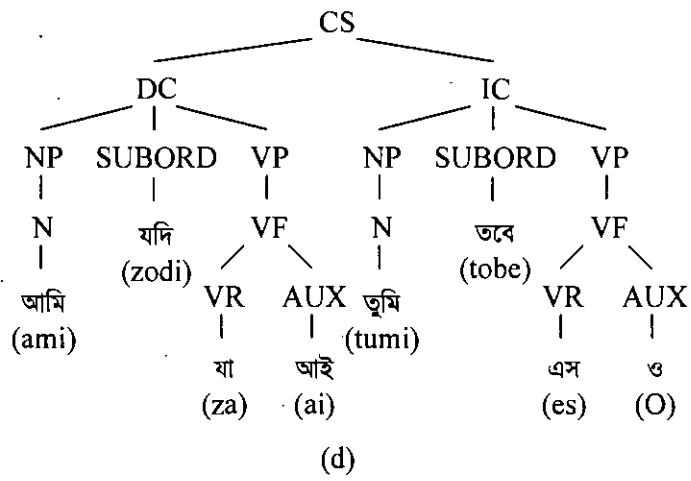


Figure 3.9: Tree structure for the rules of complex sentences in different variations (continued).



Continued Figure 3.9: Tree structure for the rules of complex sentences in different variations (continued).



Continued Figure 3.9: Tree structure for the rules of complex sentences in different variations.

Context-free grammar for complex sentence is ambiguous. Because, complex sentence is composed of noun phrase (NP) and verb phrase (VP). And grammar for both the phrases is ambiguous. Not only this, this grammar does not support predictive parsing.

3.8 Context-free Grammar for Compound Sentence

Context-free grammar for complex sentence in Bangla natural language is as follows:

COMS \rightarrow SS Conj SS

COMS \rightarrow SS Conj CS

COMS \rightarrow CS Conj SS

COMS \rightarrow CS Conj CS

where, COMS indicates compound sentence, SS indicates simple sentence, CS indicates complex sentence and Conj indicates conjunctive. Conjunctive is a type of word which is used to connect two simple or complex sentence without changing the meaning of corresponding simple or complex sentence. Some examples of conjunctive are, “ও” (O), “এবং” (ebong), “নইলে” (noile), “কিন্তু” (kintu), “নতুবা” (notuba), “সুতরাং” (sutorang) etc.

According to the rule COMS \rightarrow SS Conj SS, a compound sentence (COMS) can be composed of a simple sentence (SS) with another simple sentence (SS) through a conjunctive (Conj). We can give example of sentence using this rule as, “আমি ভাত খাই এবং সে রুটি খায়” (ami vat khai ebong se ruTi khay). Tree structure of the derivations for this example is presented in figure 3.10.

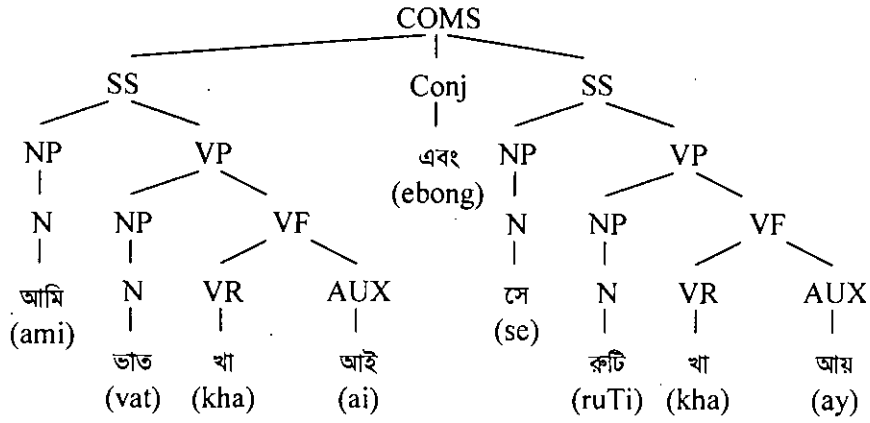


Figure 3.10: Tree structure for the rule COMS → SS Conj SS.

According to the rule COMS → SS Conj CS, a compound sentence can be composed of a simple sentence (SS) with a complex sentence (CS) through a conjunctive (Conj). We can give example of sentence using this rule as, “আমি পাঞ্জাবি পরব আর তুমি যদি স্যুট পর তবে তুমি টাই পরো” (ami panjabi poRbo ar tumi jodi syuT poRo tobe tumi Tai poRO). Tree structure of the derivations for this example is presented in figure 3.11.

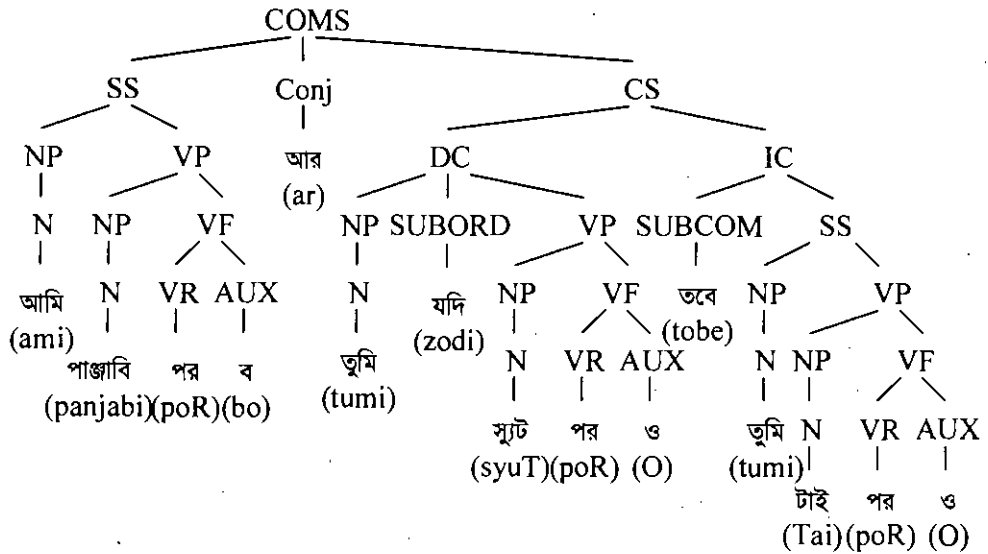


Figure 3.11: Tree structure for the rule COMS → SS Conj CS.

According to the rule COMS → CS Conj SS, a compound sentence can be composed of a complex sentence (CS) with a simple sentence (SS) through a

conjunctive (Conj). We can give example of sentence using this rule as, “যদি আমি ভাত খাই তবে সে ভাত খাবে নতুবা আমরা রুটি খাব” (zodi ami vat khai tobe se vat khabe notuba amora ruTi khabo). Tree structure of the derivations for this example is presented in figure 3.12.

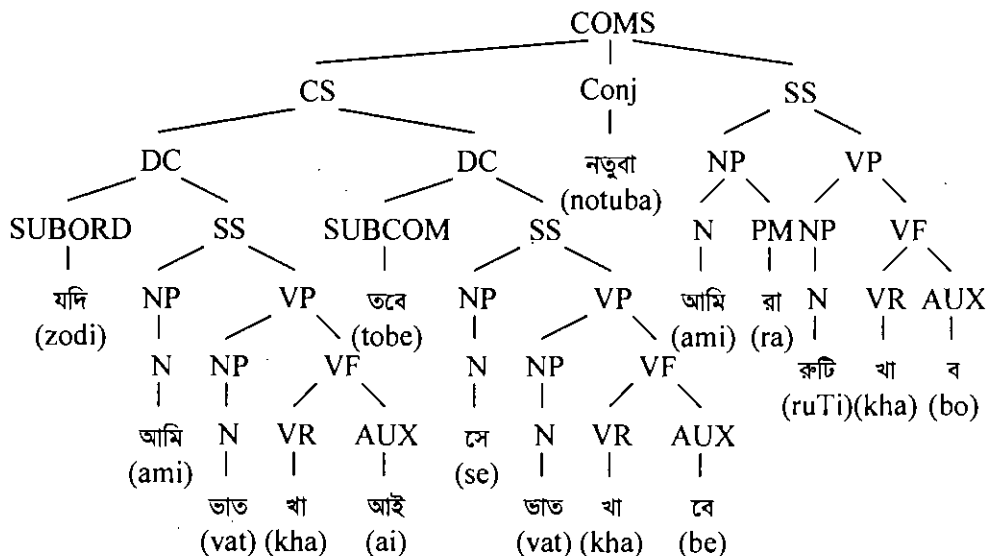


Figure 3.12: Tree structure for the rule COMS → CS Conj SS.

According to the rule COMS → CS Conj CS, a compound sentence can be composed of a complex sentence (CS) with another complex sentence (CS) through a conjunctive (Conj). We can give example of sentence using this rule as, “তুমি যদি ভাত খাও তবে আমি রুটি খাব আর তুমি যদি রুটি খাও তবে আমি ভাত খাব” (tumi zodi vat khao tobe ami ruTi khabo ar tumi zodi ruTi khao tobe ami vat khabo). Tree structure of the derivations for this example is presented in figure 3.13.

Like simple sentence and complex sentence, grammar for compound sentence is also ambiguous and does not support predictive parsing. Because, compound sentence is composed of several noun and verb phrases.

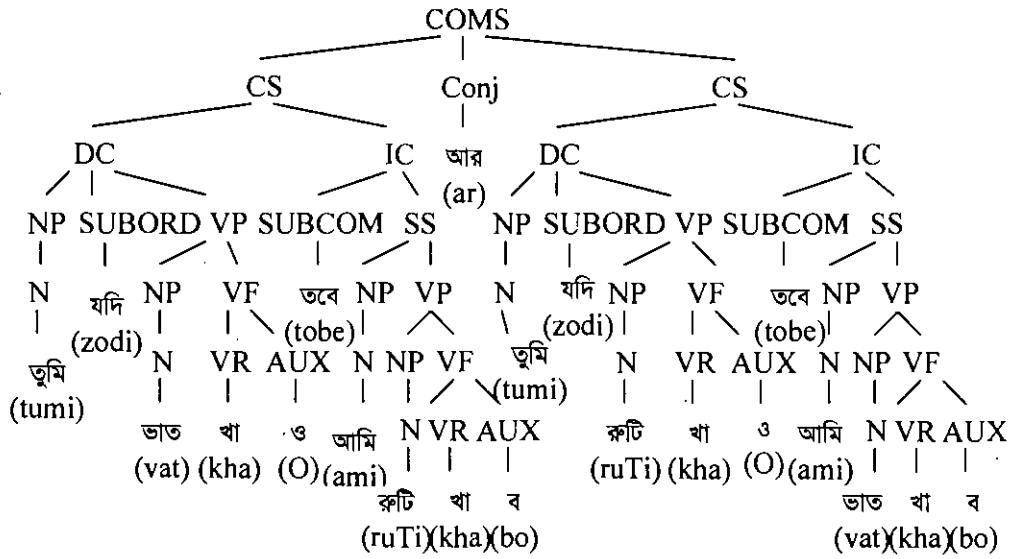


Figure 3.13: Tree structure for the rule COMS → CS Conj CS.

3.9 Context-free Grammar for Bangla Sentence

Context-free grammar for complex sentence in Bangla natural language is as follows:

$$S \rightarrow SS / CS / COMS$$

where, S indicates sentence, SS indicates simple sentence, CS indicates complex sentence and COMS indicates compound sentence.

Examples of all these tree types of sentences are already given in the previous three sections (3.6, 3.7, 3.8).

As a sentence can be simple or complex or compound and grammar for all these types of sentences is ambiguous and non-predictive, so context-free grammar for Bangla sentences is ambiguous and does not support predictive parsing.

3.10 Limitations of Present Context-free Grammar for Bangla Language

Using present context-free grammar for Bangla language, it is possible to parse almost all types of sentences of this language. Many limitations prevail in this grammar. We can mention the limitations as:

- Ambiguity
- Non-predictive
- Lacking error recoverability
- Unable to handle non-dictionary word
- Limited use of conjunctives
- Unable to handle numbers

3.10.1 Ambiguity

Ambiguity is found in many rules of present context-free grammar for Bangla language discussed in this chapter. Because of this ambiguous nature, a Bangla sentence can be parsed in different ways i.e., different parsing tree(s) exist for a single sentence. Main difficulty of ambiguous nature in a grammar is, this grammar is not LL(1) grammar. As a result, predictive parsing is not possible, which makes the task of parsing non-efficient.

For example, if we consider the sentence segment “আমার ভাইয়ের বন্ধু” (amar vaiyer bondhu) for parsing, two possible parsing is possible i.e., there will be two parsing trees. One parsing tree is left associative and another one is right associative. These two parsing trees are shown in figure 3.14.

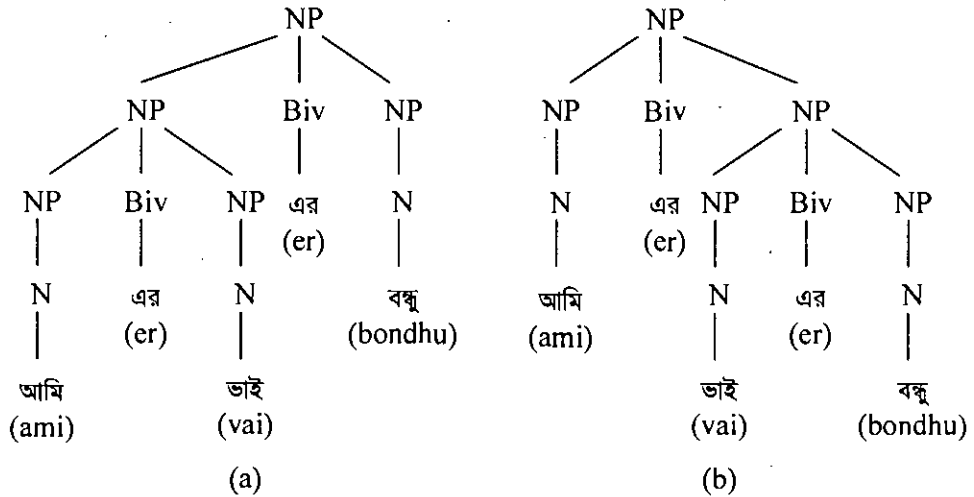


Figure 3.14: Ambiguity in the rule $NP \rightarrow NP \text{ Biv } NP$.

3.10.2 Non-predictive

Using present context-free grammar for Bangla language, it is not possible to find correct rule, when a word of a sentence is considered. So the parser tries all possible rules applicable to the word. For a correct sentence, among all possible rules, one rule is certainly correct. The correct rule is found, considering the next words. In brief, backtracking algorithm is required for parsing. As a result, parsing can not be done in linear time, rather it takes exponential time.

Because of ambiguous nature of present Bangla grammar, it can not be called as an LL(1) grammar. And predictive parsing table can not be build without LL(1) grammar. As, a result predictive parsing is not possible. Non-predictive parsing makes the parsing process inefficient.

For example, we can consider the sentence segment “আমার ভাইয়ের বন্ধু” (amar vaiyer bondhu). After lexical analysis and parts of speech tagging the sentence segment becomes $\langle \text{“আমি”, “N”} \rangle, \langle \text{“এর”, “Biv”} \rangle, \langle \text{“ভাই”, “N”} \rangle, \langle \text{“এর”, “Biv”} \rangle, \langle \text{“বন্ধু”, “N”} \rangle$. Here, first word is “আমি” (ami) which is a noun. Several rules can be applicable with a noun as a first word. These rules are, $NP \rightarrow N \text{ (DET)}$, $NP \rightarrow NP \text{ Biv (NP)}$, $NP \rightarrow N \text{ PM}$. When the word “আমি” (ami) is considered, it is not possible

to determine, which rule is applicable. As a result, each applicable rule has to be applied, to find which one fit the other words of the sentence segment. Similarly, same problem also occurs for the next words. This produces exponential run-time and makes the parsing process inefficient.

3.10.3 Lacking Error Recoverability

Existing Bangla grammar is designed in such a way, which does not support error recovery capability. In case of machine translation, it is not guaranteed that all sentences of source language are correct. Even, not all correct sentences can be fitted within a grammar. Because a sentence can be written in several possible ways.

Existing Bangla parsing methodology stops parsing, when an error is found, i.e., a word is found which can not be fitted within the grammar. But a standard parsing methodology should continue parsing when an error is found by skipping the error and an error report should be produced.

Existing Bangla grammar is designed in such a way that it can not detect the type of error. It can not determine how to continue parsing by skipping the error.

3.10.4 Unable to Handle Non-dictionary Word

Main difference between a parser of a compiler and a parser of a language translator is that in case of compiler, all the symbols are strictly defined. But in case of a language translator, all the words can be found in a dictionary i.e., parts of speech for all words can not be defined. It is quite usual that sentences contain name of people, name of places, cities, countries etc. Almost all these words are absent in a dictionary.

Existing Bangla grammar does not support such word not found in dictionary of the parser. Such unknown word is regarded as error and parsing halts when non-dictionary word is found. For example, if we want to parse the sentence “রবিন ঢাকা

যায়” (robin Dhaka zai), parsing fails, because “রবিন” (robin) and “ঢাকা” (Dhaka) are non-dictionary words.

3.10.5 Limited Use of Conjunctives

Existing Bangla grammar supports the use of conjunctive only in a compound sentence between two simple sentences. More than two simple sentences can not be joined to form a compound sentence. For example, “আমি খাই ও তুমি খাও ও সে খায়” (ami khai O tumi khao O se khay) is not possible to be parsed, because it has two conjunctives, but the rule $COMS \rightarrow SS \text{ Conj } SS$ does not support the use of multiple conjunctives.

Conjunctives are used not only in compound sentences, conjunctives may also occur in simple or complex sentences. For example, in the sentence “আমি ও তুমি কাজটি করব” (ami O tumi kajoTi korobo), “ও” (O) conjunctive is used but the sentence is a simple sentence.

3.10.6 Unable to Handle Numeric Words

Like all other languages, Bangla language sentences may contain numeric words like “১০” (10), “১৫” (15), “২২” (22) etc. Existing Bangla grammar and parsing methodology do not support these words considered for parsing.

Non-ambiguous Grammar for Simple Sentence

4.1 Non-ambiguous Grammar for Noun Phrase

Existing context-free grammar for noun phrase in Bangla natural language is written as:

$$\text{NP} \rightarrow \text{N (DET)}$$
$$\text{NP} \rightarrow (\text{DEMO}) (\text{SPR}) (\text{AP}) \text{NP}$$
$$\text{NP} \rightarrow \text{NP Biv (NP)}$$
$$\text{NP} \rightarrow \text{N PM}$$
$$\text{SPR} \rightarrow \text{QFR (PP)}$$
$$\text{DEMO} \rightarrow (\text{DD}) (\text{DO})$$

There are two problems using this grammar. Firstly, the grammar is ambiguous. Secondly, the grammar does not support predictive parsing. Ambiguity problem should be solved first. Then we can transform the grammar usable for predictive parser using the theory of left recursion elimination and left factoring (discussed in section 2.4.12.1 and 2.4.13).

This grammar is ambiguous because of the rule $\text{NP} \rightarrow \text{NP Biv (NP)}$. Using this rule same noun phrase can be parsed in several ways. An example of such problem is demonstrated in figure 4.1 and figure 4.2, where first one is left associative parsing and second one is right associative parsing.

If there is ambiguity in a grammar, it can not be called LL(1) grammar. As a result, predictive parsing table is not possible to build.

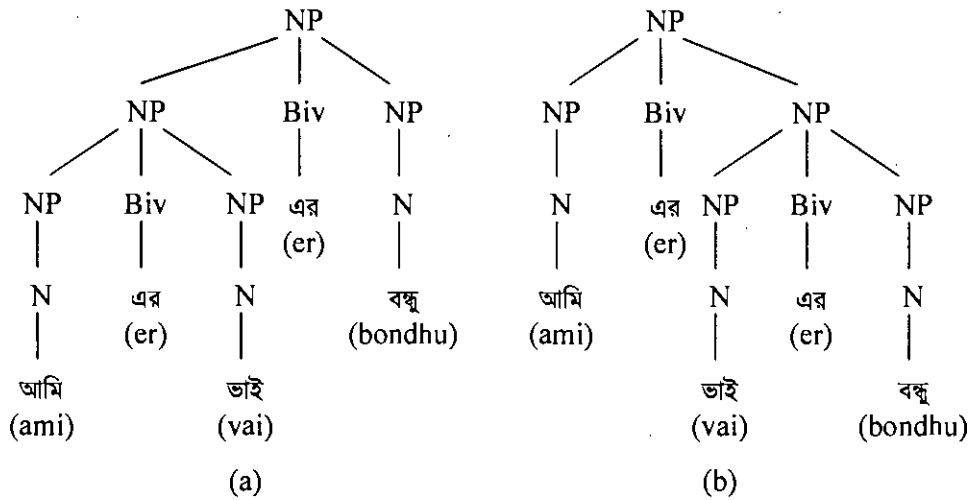


Figure 4.1: Ambiguity in the rule $NP \rightarrow NP \text{ Biv } (NP)$.

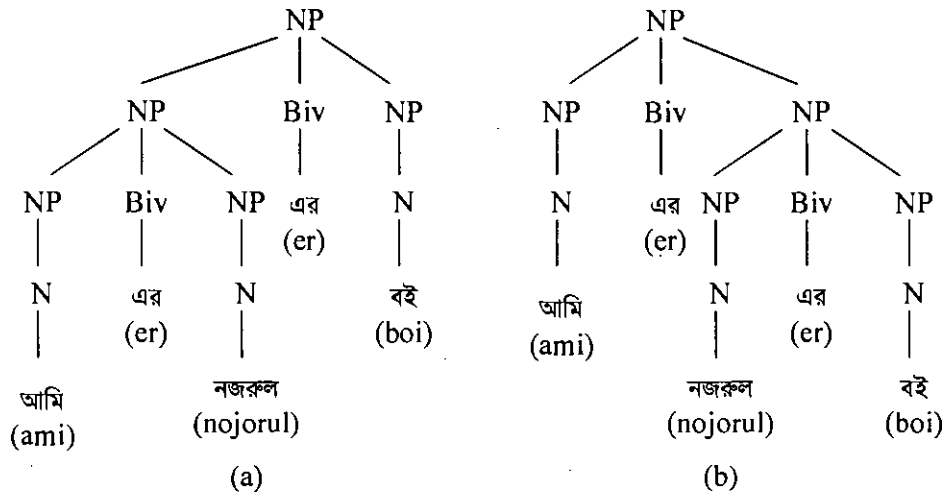


Figure 4.2: Ambiguity in the rule $NP \rightarrow NP \text{ Biv } (NP)$ (another example).

4.1.1 Ambiguity Elimination

To solve the ambiguity of the rule $NP \rightarrow NP \text{ Biv } (NP)$, we have a simple observation – there are two types of bivokti’s as extensive bivokti and non-extensive bivokti.

An extensive bivokti is a type of bivokti which extends a noun phrase, i.e., another noun phrase is expected after extensive bivokti. For example, in the sentence “আমি

এর ভাই ভাত খা আয়” (ami er vai vat kha ay), “এর” (er) is an extensive bivokti, where “এর” (er) follows another noun phrase “ভাই” (vai).

Again non-extensive bivokti is a type of bivoki which does not extend a noun phrase, i.e., another noun phrase is not expected after non-extensive bivokti. For example, in the sentence “আমি কে ভাত দা আও” (ami ke vat da ao), “কে” (ke) is a non-extensive bivokti, where “কে” (ke) follows a verb phrase “ভাত দা আও” (vat da ao).

We can denote extensive bivokti as BivE and non-extensive bivokti as Biv. Then we can re-write the rule $NP \rightarrow NP \text{ Biv} (NP)$ as:

$$NP \rightarrow NP \text{ BivE} NP \mid NP \text{ Biv}$$

Still the rule is ambiguous, because using the rule $NP \rightarrow NP \text{ BivE} NP$, different parsing of a single sentence is possible. We can remove ambiguity of the rule by posing non-extendable non-terminal to the left of BivE and Biv, and extendable non-terminal to the right of BivE and Biv.

Recalling from the rules $NP \rightarrow N (DET)$, $NP \rightarrow (DEMO) (SPR) (AP) NP$, $NP \rightarrow N PM$, we can define (DEMO) (SPR) (AP) as previous portion of noun phrase, denoted as PRE and N DET along with N PM as noun word, denoted as NW. Now, we can write rules for PRE and NW as:

$$\begin{aligned} \text{PRE} &\rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \\ &\quad \text{DEMO SPR AP} \\ \text{NW} &\rightarrow \text{N} \mid \text{N DET} \mid \text{N PM} \end{aligned}$$

Now it is quite clear that NW or PRE NW can represent any noun phrase having no bivokti. We can give an example of a noun phrase having no bivokti as “ঐ প্রথম এক টি সুন্দর ছেলে টি” (OI prothom ek Ti sundor chhele Ti). Parsing of the phrase using new rule for PRE and NW is given in figure 4.3.

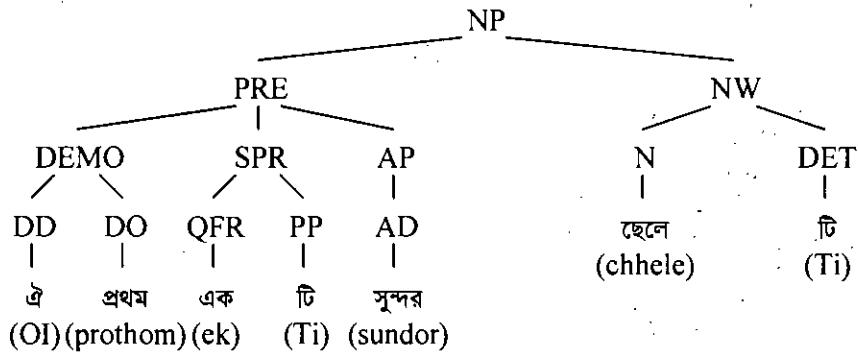


Figure 4.3: Tree derivation using the rules for PRE and NW.

As both the non-terminals PRE and NW are non-extensive i.e., no other inner noun phrase can be produced by PRE or NW, so we can place non-extensive non-terminals PRE and NW to the left side of bivokti in the rule $NP \rightarrow NP \text{ BivE } NP \mid NP \text{ Biv}$. Ultimately, the rule $NP \rightarrow NP \text{ Biv } (NP)$, previously modified as $NP \rightarrow NP \text{ BivE } NP \mid NP \text{ Biv}$, is replaced by the following rule:

$$NP \rightarrow NW \text{ BivE } NP \mid PRE \text{ NW BivE } NP \mid NW \text{ Biv} \mid PRE \text{ NW Biv}$$

Now the noun phrase “আমি এর ভাই এর বন্ধু” (ami er vai er bondhu), which was parsed in two possible ways, as demonstrated in figure 4.1, is now parsed in only one way using new equivalent rule, as demonstrated in figure 4.4.

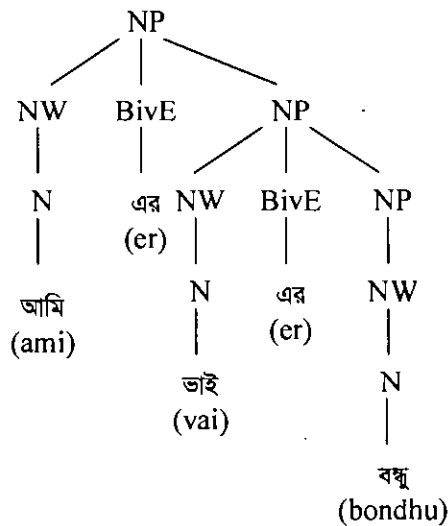


Figure 4.4: Tree derivation using non-ambiguous rule for NP.

Again, using the definition of NW, we can replace the rules $NP \rightarrow N$ (DET), $NP \rightarrow N$ PM by

$$NP \rightarrow NW$$

Similarly, using the definition of PRE, we can replace the rule $NP \rightarrow$ (DEMO) (SPR) (AP) NP by

$$NP \rightarrow PRE NW$$

Now, rule for NP can be written in a single line as:

$$NP \rightarrow NW \mid PRE NW \mid NW \text{ BivE } NP \mid PRE NW \text{ BivE } NP \mid NW \text{ Biv } \mid \\ PRE NW \text{ Biv}$$

The rule $SPR \rightarrow QFR$ (PP) can also be written as:

$$SPR \rightarrow QFR \mid QFR PP$$

without changing meaning.

Similarly, the rule $DEMO \rightarrow$ (DD) (DO) can be written as:

$$DEMO \rightarrow DD \mid DO \mid DD DO$$

without changing meaning.

Therefore, ambiguity in the grammar for noun phrase is eliminated. Existing Bangla grammar for noun phrase

$$NP \rightarrow N \text{ (DET)}$$

$$NP \rightarrow \text{(DEMO) (SPR) (AP) NP}$$

$$NP \rightarrow NP \text{ Biv (NP)}$$

$$NP \rightarrow N \text{ PM}$$

$$SPR \rightarrow QFR \text{ (PP)}$$

$$DEMO \rightarrow \text{(DD) (DO)}$$

is replaced by equivalent new non-ambiguous grammar as:

$$NP \rightarrow NW \mid PRE NW \mid NW \text{ BivE } NP \mid PRE NW \text{ BivE } NP \mid NW \text{ Biv } \mid \\ PRE NW \text{ Biv}$$

$PRE \rightarrow DEMO \mid SPR \mid AP \mid DEMO \ SPR \mid DEMO \ AP \mid SRR \ AP \mid$
 $DEMO \ SPR \ AP$
 $NW \rightarrow N \mid N \ DET \mid N \ PM$
 $SPR \rightarrow QFR \mid QFR \ PP$
 $DEMO \rightarrow DD \mid DO \mid DD \ DO$

4.1.2 Addition of Conjunctives

Existing Bangla grammar supports the use of conjunctives only in compound sentences to connect two simple or complex sentences. Existing Bangla grammar does not support the use of conjunctives within a noun phrase. But, conjunctives may also occur in a simple sentence as a part of noun phrase. For example, “আমি ও তুমি মার্কেটে যাই” (ami O tumi marrkeTe zai), “আমি ও আমার ভাইয়ের ছেলে মার্কেটে যাই” (ami O amar vaiyer chhele marrkeTe zai) etc, are simple sentences. But conjunctives are used in the sentences.

If we observe the use of conjunctives in a noun phrase, we can find, conjunctives connect some smaller noun phrases together to form a bigger noun phrase. In the first example, “আমি ও তুমি” (ami O tumi) is a noun phrase which is generated by two smaller noun phrases “আমি” (ami) and “তুমি” (tumi) connected by conjunctive “ও” (O). Similarly, in the second example, “আমি ও আমার ভাইয়ের ছেলে” (ami O amar vaiyer chhele) is a noun phrase which is generated by two smaller noun phrases “আমি” (ami) and “আমার ভাইয়ের ছেলে” (amar vaiyer chhele) connected by conjunctive “ও” (O).

Now, we are in a position to propose a new rule for noun phrase, which supports the use of conjunctive(s) within the noun phrase, as

$NP \rightarrow NP \ Conj \ NP$

This new rule supports the use of conjunctive within noun phrase. Now, it is possible to parse the above two sentences. Tree derivations of the sentences are demonstrated in figure 4.5.

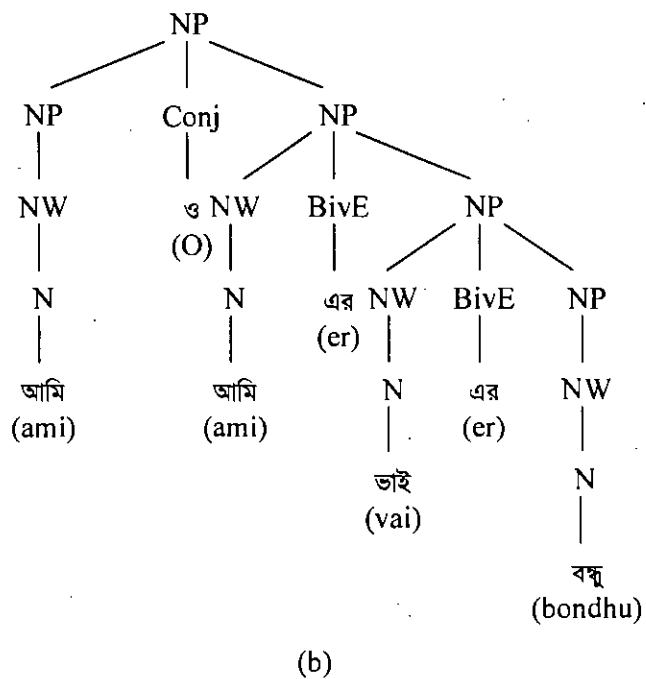
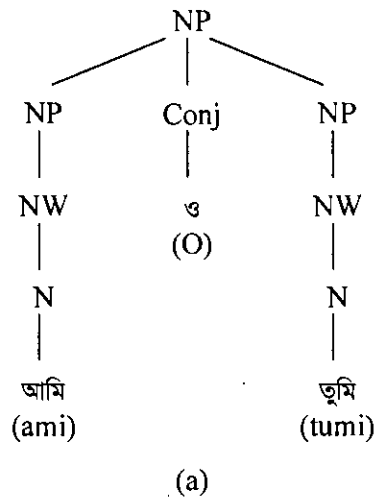


Figure 4.5: Tree derivation using the rule $NP \rightarrow NP \text{ Conj } NP$.

In both the examples, main noun phrase is composed of two smaller noun phrases connected by a conjunctive. Using the new rule, it is also possible to parse a noun phrase which contains more than two smaller noun phrases connected by conjunctives. We can give such example as “আমি, তুমি ও সে” (ami, tumi O se). Here,

three noun phrases “আমি” (ami), “তুমি” (tumi) and “সে” (se) are connected by conjunctives. Tree derivation of this sentence is demonstrated in figure 4.6.

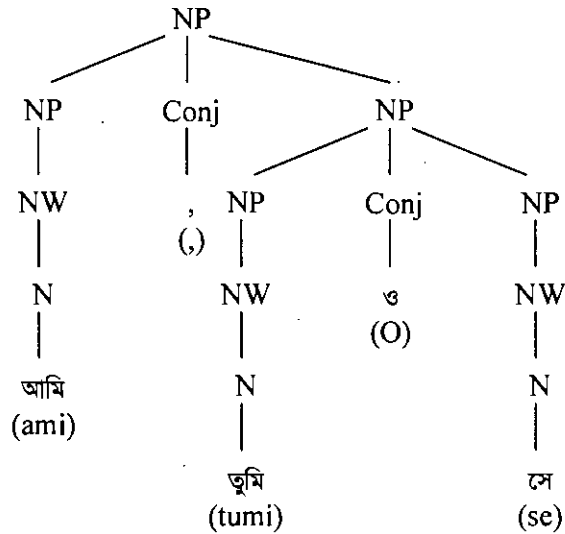


Figure 4.6: Tree derivation using the rule $NP \rightarrow NP \text{ Conj } NP$, where three smaller noun phrases are connected by conjunctives.

Unfortunately, ambiguity problem arises using this new rule. When a noun phrase is composed of two smaller noun phrases connected by conjunctive, ambiguity problem does not arise. But when a noun phrase is composed of more than two smaller noun phrases connected by conjunctives, there exists several possible derivations. For example, the noun phrase “আমি, তুমি ও সে” (ami, tumi O se) can be parsed in two possible ways, as demonstrated in figure 4.7.

The reason of ambiguity in this rule is, both left and right non-terminals NP are extensive. We can eliminate the ambiguity in this rule by applying same technique used in the previous section (4.1.1). We have to convert any non-terminal, either left or right, as non-extendable. In this case, non-extendable noun phrase does not contain two or more smaller noun phrases connected by conjunctive(s), whether an extendable noun phrase may contain two or more smaller noun phrases connected by conjunctive(s).

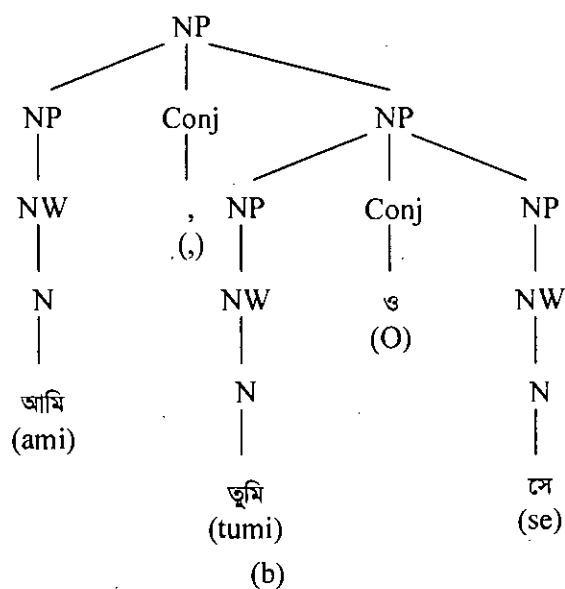
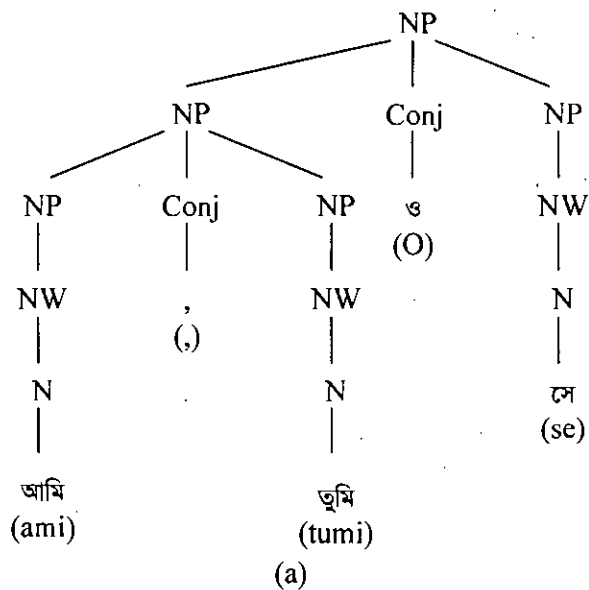


Figure 4.7: Ambiguity using the rule $NP \rightarrow NP \text{ Conj } NP$, where one is left associative (a), and another is right associative (b).

We can define a non-extendable noun phrase, which does not contain two or more smaller noun phrases connected by conjunctive(s), as NPU. Now, the rule for noun phrase is re-written as:

$$NP \rightarrow \text{NPU Conj } NP$$

where, ambiguity problem is solved because of non-extendable non-terminal NPU.

As use of conjunctive in a noun phrase is optional, we can re-write the rule for noun phrase again as:

$$NP \rightarrow NPU \mid NPU \text{ Conj } NP$$

In this situation, the rule for noun phrase, as proposed in the previous section (4.1.1), is now fit within the definition of NPU. So, rule for NPU becomes,

$$NPU \rightarrow NW \mid \text{PRE } NW \mid NW \text{ BivE } NP \mid \text{PRE } NW \text{ BivE } NP \mid NW \text{ Biv} \mid \text{PRE } NW \text{ Biv}$$

Therefore, use of conjunctives within noun phrase is achieved, along with ambiguity problem is also solved. Now, the noun phrase “আমি, তুমি ও সে” (ami, tumi O se) can be parsed in only one possible way as demonstrated in figure 4.8.

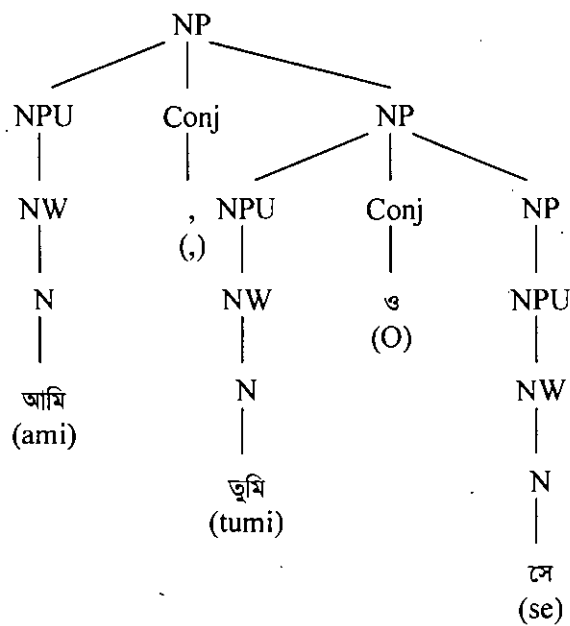


Figure 4.8: Elimination of ambiguity of the rule $NP \rightarrow NP \text{ Conj } NP$ using new rule $NP \rightarrow NPU \mid NPU \text{ Conj } NP$.

Finally, the grammar for noun phrase, as proposed in the previous section,

$$NP \rightarrow NW \mid \text{PRE } NW \mid NW \text{ BivE } NP \mid \text{PRE } NW \text{ BivE } NP \mid NW \text{ Biv} \mid \text{PRE } NW \text{ Biv}$$

$$\text{PRE} \rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \\ \text{DEMO SPR AP}$$

$$\text{NW} \rightarrow \text{N} \mid \text{N DET} \mid \text{N PM}$$

$$\text{SPR} \rightarrow \text{QFR} \mid \text{QFR PP}$$

$$\text{DEMO} \rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO}$$

is now replaced by new grammar,

$$\text{NP} \rightarrow \text{NPU} \mid \text{NPU Conj NP}$$

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \\ \text{PRE NW Biv}$$

$$\text{PRE} \rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \\ \text{DEMO SPR AP}$$

$$\text{NW} \rightarrow \text{N} \mid \text{N DET} \mid \text{N PM}$$

$$\text{SPR} \rightarrow \text{QFR} \mid \text{QFR PP}$$

$$\text{DEMO} \rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO}$$

which also preserves the features of noun phrase achieved before.

4.1.3 Unknown Word Handling

In case of machine translation, in source language sentences, many words are found, which are not available in dictionary. While lexical analysis, parts of speech tagging of such words is not possible. These non-dictionary words are regarded as unknown words.

Existing Bangla grammar is unable to handle non-dictionary words. Existing Bangla parsing methodology can not continue parsing, when any error occurs. As parts of speech tagging of unknown words is not possible, it is regarded as error, and parsing halts when unknown word occurs in the sentence to be parsed.

Unknown word handling is very important, because without this, parsing is very weak, as a result, quality, accuracy and effectiveness of machine translation is highly reduced.

If we analyze unknown or non-dictionary words in Bangla language sentences, we can see, most of the unknown words are name of man, place, company, organization, product etc. So, we can consider an unknown word like a noun. Very rarely, non-dictionary words are in other groups.

Though we can consider an unknown word as a noun, but in the grammar an unknown words can not immediately substitute a noun. Because, firstly, the non-terminal PRE may be applicable before a noun, but not applicable before an unknown word. For example, “ঐ প্রথম সুন্দর ছেলেটি মার্কেটে যায়” (OI prothom sundor chheleTi marrkeTe zay) is a meaningful sentence. But, “ঐ প্রথম সুন্দর রবিনটি মার্কেটে যায়” (OI prothom sundor robinoTi marrkeTe zay) is not a meaningful sentence.

Again, unlike noun, PM and DET is not applicable after an unknown word. For example, “ছেলেটি ফুটবল খেলে” (chheleTi fuTobol khele) is a meaningful sentence. But, “রবিনটি ফুটবল খেলে” (robinoTi fuTobol khele) is not a meaningful sentence.

Because of these differences, we can not tag an unknown word directly as the terminal N. We have to define a new terminal as UN, referring to unknown words. Now, we are in a position to re-write new rule for NPU to facilitate unknown word handling as,

$$\begin{aligned} \text{NPU} \rightarrow & \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \\ & \text{PRE NW Biv} \mid \text{UN} \mid \text{UN BivE NP} \mid \text{UN Biv} \end{aligned}$$

which also retains other features of NPU, achieved earlier.

Now, it is possible to parse noun phrases containing unknown words. For example, tree derivations of the sentences “রবিন” (robin) and “আমি ও রবিন এর বন্ধু” (ami O robin er bondhu) are given in figure 4.9.

Still, there is a limitation using the new rule. In many cases, two or more separate words together make a group and the group represent a significant meaning, not the separate unknown words. For example, name of a man can be “আমির খান” (amir

khan), a name of a place can be “বুয়েন্স আয়ার্স” (buyens ayarrs), a name of a company can be “বেঙ্গল গ্রুপ অব ইন্ডাস্ট্রিজ” (bengol grup ob inDasTrij) etc.

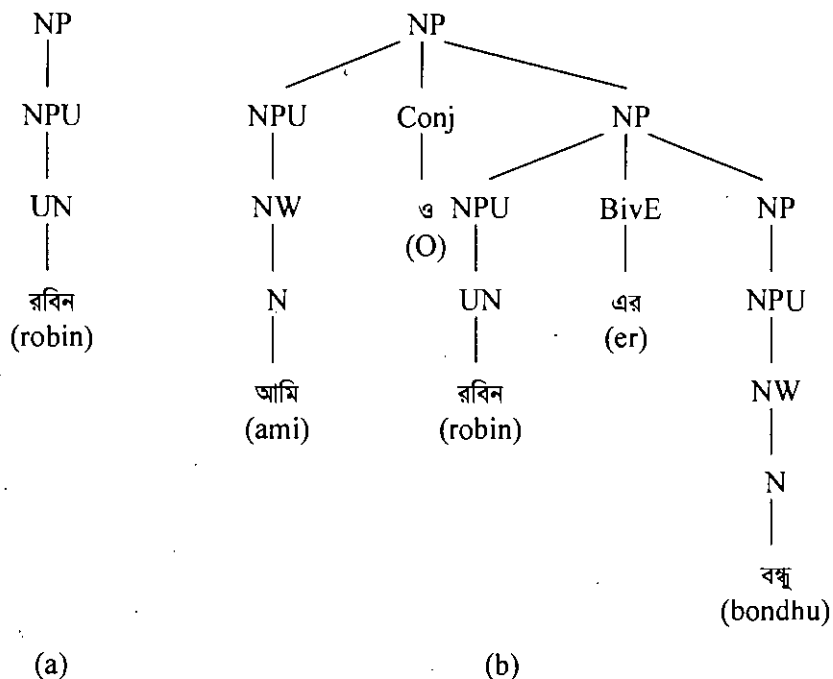


Figure 4.9: Tree derivation with grammar for noun phrase with unknown word (non-dictionary) handling.

It follows that, only use of the terminal UN is not sufficient. We can define a new non-terminal for unknown word group, denoted by UNG, which represents a group of unknown words. Now, we can replace UN by UNG in the rule for NPU and resultant rule becomes,

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \\ \text{PRE NW Biv} \mid \text{UNG} \mid \text{UNG BivE NP} \mid \text{UNG Biv}$$

which now supports multi-word unknown word groups.

As UNG represents one or more unknown words, we can define the rule for UNG as,

$$\text{UNG} \rightarrow \text{UN} \mid \text{UN UNG}$$

We can give example of noun phrase using multi-word unknown word group as “আমির খান” (amir khan), “আমি ও আমির খান এর বন্ধু” (ami O amir khan er bondhu) etc, for which tree derivations are demonstrated in figure 4.10.

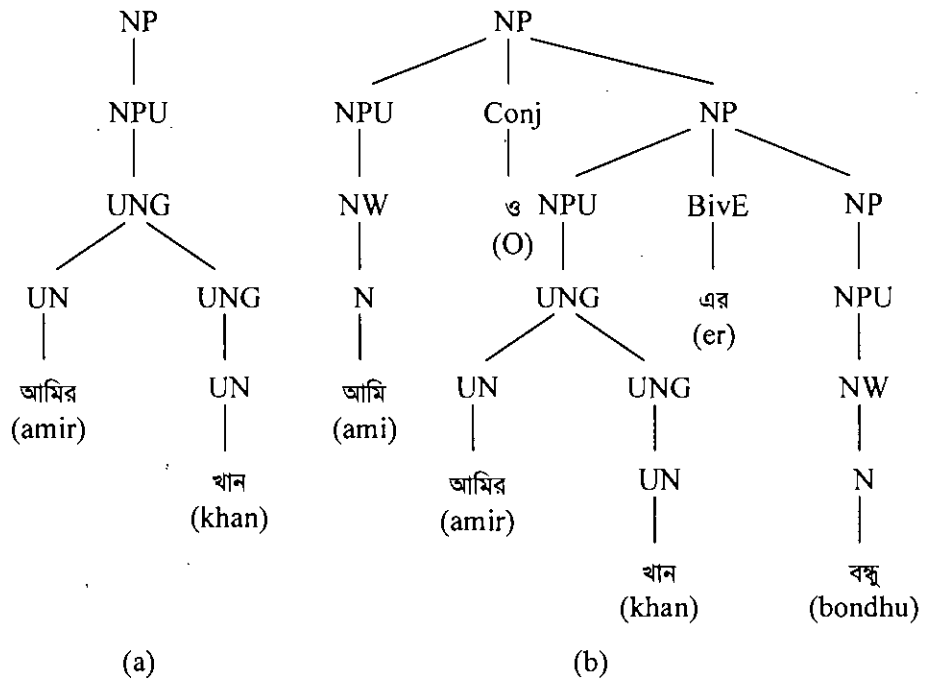


Figure 4.10: Tree derivation with grammar for noun phrase with both single (a) and multi-word (b) unknown word (non-dictionary) handling.

Finally, the grammar of noun phrase, as proposed in the previous section,

$$\text{NP} \rightarrow \text{NPU} \mid \text{NPU Conj NP}$$

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \text{PRE NW Biv}$$

$$\text{PRE} \rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \text{DEMO SPR AP}$$

$$\text{NW} \rightarrow \text{N} \mid \text{N DET} \mid \text{N PM}$$

$$\text{SPR} \rightarrow \text{QFR} \mid \text{QFR PP}$$

$$\text{DEMO} \rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO}$$

is now replace by new grammar as,

$$\text{NP} \rightarrow \text{NPU} \mid \text{NPU Conj NP}$$

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \text{PRE NW Biv} \mid \text{UNG} \mid \text{UNG BivE NP} \mid \text{UNG Biv}$$

$$\begin{aligned} \text{PRE} &\rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \\ &\quad \text{DEMO SPR AP} \\ \text{NW} &\rightarrow \text{N} \mid \text{N DET} \mid \text{N PM} \\ \text{SPR} &\rightarrow \text{QFR} \mid \text{QFR PP} \\ \text{DEMO} &\rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO} \\ \text{UNG} &\rightarrow \text{UN} \mid \text{UN UNG} \end{aligned}$$

which facilitate the use of unknown word retaining the features achieved earlier.

4.1.4 Left Factoring

In the previous sections (4.1.1, 4.1.2, 4.1.3), ambiguity problem for the grammar of noun phrase is eliminated and further, use of conjunctives and unknown word handling have been achieved. Now, we have a non-ambiguous grammar, but the grammar is not applicable to predictive parser, as most of the alternative rules for the non-terminals use common terminal or non-terminal as first symbol. As a result, it is not possible to determine the applicable rule immediately. We can eliminate such problem by applying the theory of left factoring.

Firstly, we have a non-ambiguous grammar for noun phrase as,

$$\begin{aligned} \text{NP} &\rightarrow \text{NPU} \mid \text{NPU Conj NP} \\ \text{NPU} &\rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid \\ &\quad \text{PRE NW Biv} \mid \text{UNG} \mid \text{UNG BivE NP} \mid \text{UNG Biv} \\ \text{PRE} &\rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid \\ &\quad \text{DEMO SPR AP} \\ \text{NW} &\rightarrow \text{N} \mid \text{N DET} \mid \text{N PM} \\ \text{SPR} &\rightarrow \text{QFR} \mid \text{QFR PP} \\ \text{DEMO} &\rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO} \\ \text{UNG} &\rightarrow \text{UN} \mid \text{UN UNG} \end{aligned}$$

Firstly, we can apply left factoring on the rule for NP,

$$\begin{aligned} \text{NP} &\rightarrow \text{NPU} \mid \text{NPU Conj NP} \\ &\Rightarrow \end{aligned}$$

$$\text{NP} \rightarrow \text{NPU E1}$$

where, $\text{E1} \rightarrow \text{Conj NP} \mid \varepsilon$

Then, we can apply left factoring on the rule for NPU,

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid$$

$$\text{PRE NW Biv} \mid \text{UNG} \mid \text{UNG BivE NP} \mid \text{UNG Biv}$$

$$\Rightarrow$$

$$\text{NPU} \rightarrow \text{NW E2} \mid \text{PRE NW E2} \mid \text{UNG E2}$$

where, $\text{E2} \rightarrow \text{BivE NP} \mid \text{Biv} \mid \varepsilon$

Then, we can apply left factoring on the rule for PRE,

$$\text{PRE} \rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid$$

$$\text{DEMO SPR AP}$$

$$\Rightarrow$$

$$\text{PRE} \rightarrow \text{DEMO E3} \mid \text{SPR E4} \mid \text{AP}$$

where, $\text{E3} \rightarrow \text{SPR} \mid \text{AP} \mid \text{SPR AP} \mid \varepsilon$

$$\text{E4} \rightarrow \text{AP} \mid \varepsilon$$

Then, we can apply left factoring on the rule for E3,

$$\text{E3} \rightarrow \text{SPR} \mid \text{AP} \mid \text{SPR AP} \mid \varepsilon$$

$$\Rightarrow$$

$$\text{E3} \rightarrow \text{SPR E4} \mid \text{AP} \mid \varepsilon$$

Then, we can apply left factoring on the rule for NW,

$$\text{NW} \rightarrow \text{N} \mid \text{N DET} \mid \text{N PM}$$

$$\Rightarrow$$

$$\text{NW} \rightarrow \text{N E5}$$

where, $\text{E5} \rightarrow \text{DET} \mid \text{PM} \mid \varepsilon$

Then, we can apply left factoring on the rule for SPR,

$$\text{SPR} \rightarrow \text{QFR} \mid \text{QFR PP}$$

$$\Rightarrow$$

$$\text{SPR} \rightarrow \text{QFR E6}$$

$$\text{where, E6} \rightarrow \text{PP} \mid \varepsilon$$

Then, we can apply left factoring on the rule for DEMO,

$$\text{DEMO} \rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO}$$

$$\Rightarrow$$

$$\text{DEMO} \rightarrow \text{DD E7} \mid \text{DO}$$

$$\text{where, E7} \rightarrow \text{DO} \mid \varepsilon$$

Then, we can apply left factoring on the rule for UNG,

$$\text{UNG} \rightarrow \text{UN} \mid \text{UN UNG}$$

$$\Rightarrow$$

$$\text{UNG} \rightarrow \text{UN E8}$$

$$\text{where, E8} \rightarrow \text{UNG} \mid \varepsilon$$

Ultimately, we can say, grammar for noun phrase proposed in the previous section,

$$\text{NP} \rightarrow \text{NPU} \mid \text{NPU Conj NP}$$

$$\text{NPU} \rightarrow \text{NW} \mid \text{PRE NW} \mid \text{NW BivE NP} \mid \text{PRE NW BivE NP} \mid \text{NW Biv} \mid$$

$$\text{PRE NW Biv} \mid \text{UNG} \mid \text{UNG BivE NP} \mid \text{UNG Biv}$$

$$\text{PRE} \rightarrow \text{DEMO} \mid \text{SPR} \mid \text{AP} \mid \text{DEMO SPR} \mid \text{DEMO AP} \mid \text{SRR AP} \mid$$

$$\text{DEMO SPR AP}$$

$$\text{NW} \rightarrow \text{N} \mid \text{N DET} \mid \text{N PM}$$

$$\text{SPR} \rightarrow \text{QFR} \mid \text{QFR PP}$$

$$\text{DEMO} \rightarrow \text{DD} \mid \text{DO} \mid \text{DD DO}$$

$$\text{UNG} \rightarrow \text{UN} \mid \text{UN UNG}$$

is now replaced by new grammar,

$$\text{NP} \rightarrow \text{NPU E1}$$

$$\text{E1} \rightarrow \text{Conj NP} \mid \varepsilon$$

$$\text{NPU} \rightarrow \text{NW E2} \mid \text{PRE NW E2} \mid \text{UNG E2}$$

$$\text{E2} \rightarrow \text{BivE NP} \mid \text{Biv} \mid \varepsilon$$

$$\text{PRE} \rightarrow \text{DEMO E3} \mid \text{SPR E4} \mid \text{AP}$$

$$\text{E3} \rightarrow \text{SPR E4} \mid \text{AP} \mid \varepsilon$$

- $E4 \rightarrow AP \mid \epsilon$
- $NW \rightarrow N E5$
- $E5 \rightarrow DET \mid PM \mid \epsilon$
- $SPR \rightarrow QFR E6$
- $E6 \rightarrow PP \mid \epsilon$
- $DEMO \rightarrow DD E7 \mid DO$
- $E7 \rightarrow DO \mid \epsilon$
- $UNG \rightarrow UN E8$
- $E8 \rightarrow UNG \mid \epsilon$

which is a non-ambiguous grammar for predictive parser. This grammar can also be called as LL(1).

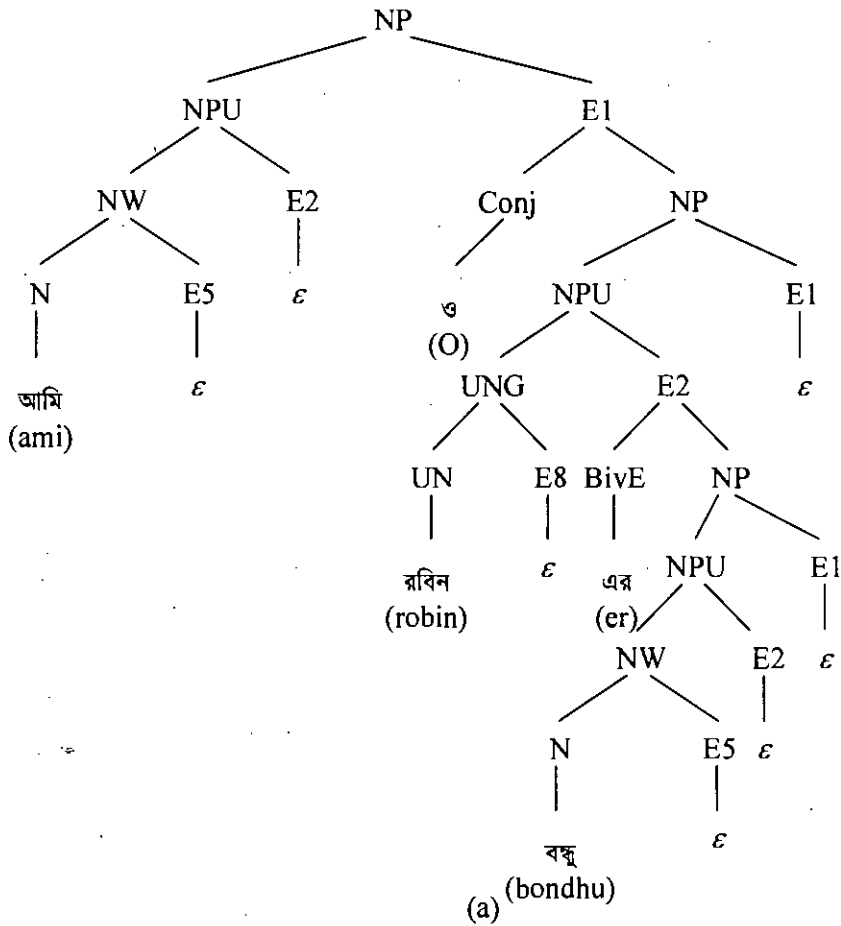
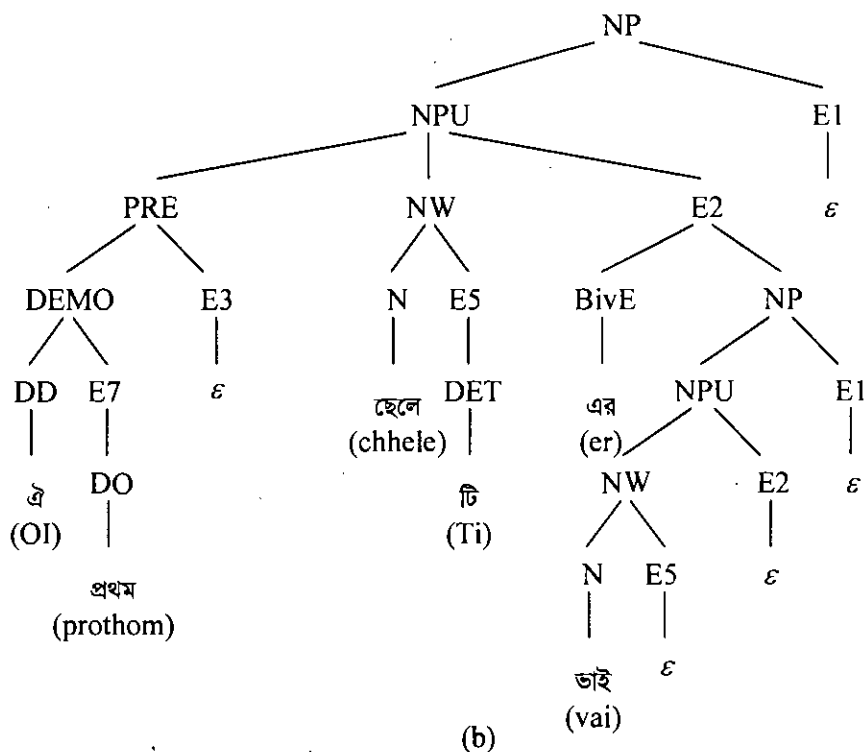


Figure 4.11: Tree derivation with non-ambiguous grammar for noun phrase with predictive parsing (continued).



Continued Figure 4.11: Tree derivation with non-ambiguous grammar for noun phrase with predictive parsing.

Now, we are in a position to give some examples of parsing using new grammar for noun phrase. Tree derivations for some noun phrases like, “আমি ও রবিন এর বন্ধু” (ami O robin er bondhu), “ঐ প্রথম ছেলেটির ভাই” (OI prothom chheleTir vai) are demonstrated in figure 4.11.

4.2 Non-ambiguous Grammar for Adjective Phrase

Existing context-free grammar for adjective phrase in Bangla natural language is written as:

$$AP \rightarrow AD / ADs$$

which indicates that, an adjective phrase is composed of one or more adjectives.

Existing grammar is not suitable for practical implementation. Thus, we can re-write the grammar as,

$$AP \rightarrow AD | AD AP$$

which is a non-ambiguous grammar.

For example, in the sentence “সে খুব ভাল গান গাইতে পারে” (se khub valo gan gaito pare), “খুব ভাল” (khub valo) is an adjective phrase. We can parse adjective using the non-ambiguous grammar as demonstrated in figure 4.12.

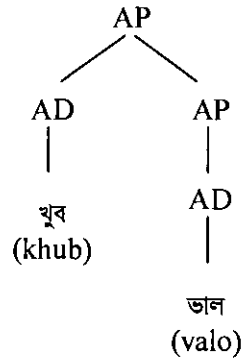


Figure 4.12: Tree derivation with non-ambiguous grammar for adjective phrase.

Difficulty using this grammar is, both alternative rules $AP \rightarrow AD$, $AP \rightarrow AD AP$ has AD as first symbol. As a result, this grammar does not support predictive parsing.

We can transform the grammar usable for predictive parsing by applying left factoring as follows,

$$AP \rightarrow AD \mid AD AP$$

\Rightarrow

$$AP \rightarrow AD F1$$

$$\text{where, } F1 \rightarrow AP \mid \varepsilon$$

Ultimately, non-ambiguous grammar for noun phrase with predictive parsing becomes,

$$AP \rightarrow AD F1$$

$$F1 \rightarrow AP \mid \varepsilon$$

Now, we can parse the adjective phrase “খুব ভাল” (khub valo) using new grammar, as demonstrated in figure 4.13.

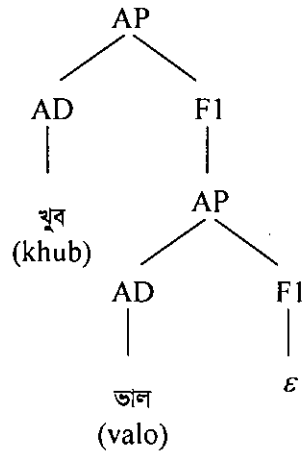


Figure 4.13: Tree derivation with non-ambiguous grammar for adjective phrase with predictive parsing.

4.2.1 Numeric Word Handling

Like all other languages, Bangla language sentences may contain numeric words like “১০” (10), “১৫” (15), “২২” (22) etc. Existing Bangla grammar and parsing methodology do not support these words considered for parsing.

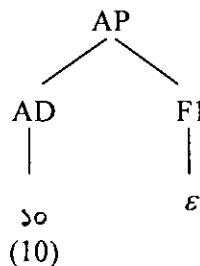


Figure 4.14: Tree derivation for a numeric word.

An important observation is that, numeric words are used as adjectives in Bangla sentences. Therefore, when any word containing only numeric digits is detected by the parser, it can be considered as adjective without looking up the lexicon. Then, parsing process continues after detection of such kind of word.

For example, we can parse the numeric word “১০” (10), as demonstrated in figure 4.14.

4.3 Non-ambiguous Grammar for Verb Phrase

Existing context-free grammar for verb phrase in Bangla natural language is written as:

$$\begin{aligned} VP &\rightarrow (NP) (AP) VF \\ VF &\rightarrow VR AUX \end{aligned}$$

We can re-write the rule for VP as,

$$VP \rightarrow VF \mid NP VF \mid AP VF \mid NP AP VF$$

This rule is not applicable in predictive parser. We can transform it to be usable for predictive parser by following derivations,

$$VP \rightarrow VF \mid NP VF \mid AP VF \mid NP AP VF$$

\Rightarrow

$$VP \rightarrow VF \mid AP VF \mid NP D1$$

$$\text{where, } D1 \rightarrow VF \mid AP VF$$

\Rightarrow

$$VP \rightarrow VF \mid AP VF \mid NPU E1 D1 \quad (\text{expanding rule for NP})$$

\Rightarrow

$$VP \rightarrow VF \mid AP VF \mid NW E2 E1 D1 \mid PR E NW E2 E1 D1 \mid UNG E2 E1 D1 \quad (\text{expanding rule for NPU})$$

\Rightarrow

$$VP \rightarrow VF \mid AP VF \mid NW E2 E1 D1 \mid DEMO E3 NW E2 E1 D1 \mid$$

$$SPR E4 NW E2 E1 D1 \mid AP NW E2 E1 D1 \mid UNG E2 E1 D1$$

(expanding rule for PRE)

\Rightarrow

$$VP \rightarrow VF \mid NW E2 E1 D1 \mid UNG E2 E1 D1 \mid AP D2 \mid$$

$$DEMO E3 NW E2 E1 D1 \mid SPR E4 NW E2 E1 D1$$

(left factoring)

$$\text{where, } D2 \rightarrow VF \mid NW E2 E1 D1$$

To resolve ambiguity problem of simple sentence (discussed in the next section), we have to separate UNG from other portions. For this purpose, we can define a rule,

$$D3 \rightarrow VF \mid AP \ D2 \mid NW \ E2 \ E1 \ D1 \mid DEMO \ E3 \ NW \ E2 \ E1 \ D1 \mid \\ SPR \ E4 \ NW \ E2 \ E1 \ D1$$

Using the new rule, rule for VP can be re-written as,

$$VP \rightarrow D3 \mid UNG \ E2 \ E1 \ D1$$

For simplification, we can define another rule as,

$$D4 \rightarrow NW \ E2 \ E1 \ D1$$

Therefore, the rules D2 and D3 becomes,

$$D2 \rightarrow VF \mid D4$$

$$D3 \rightarrow VF \mid AP \ D2 \mid D4 \mid DEMO \ E3 \ D4 \mid SPR \ E4 \ D4$$

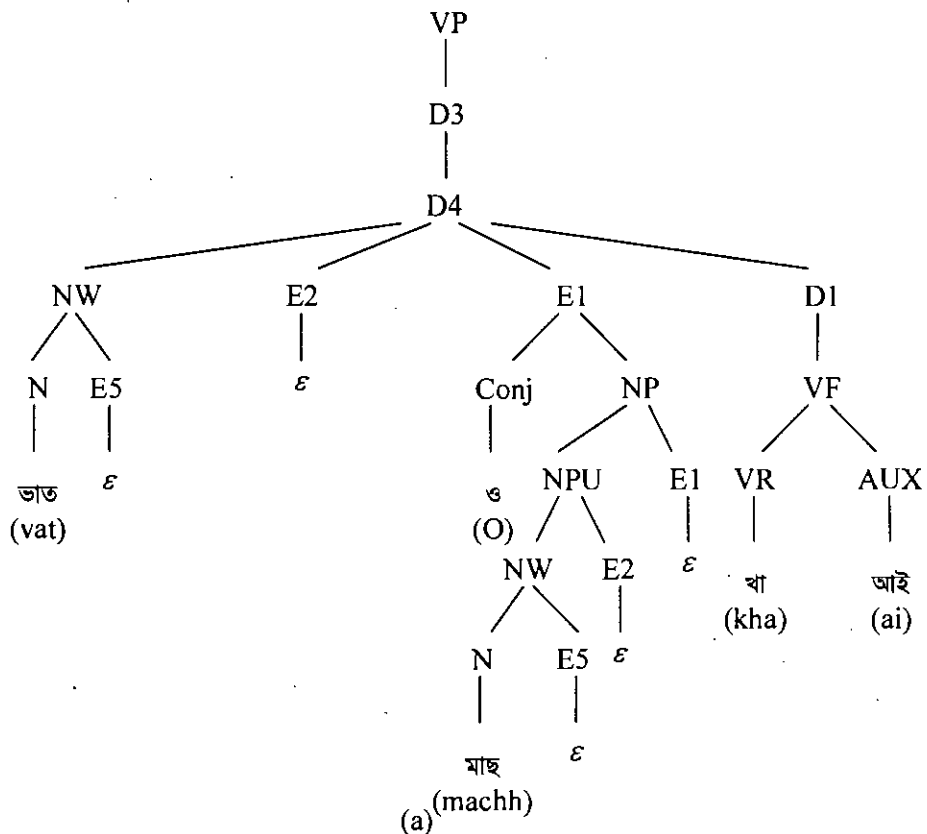
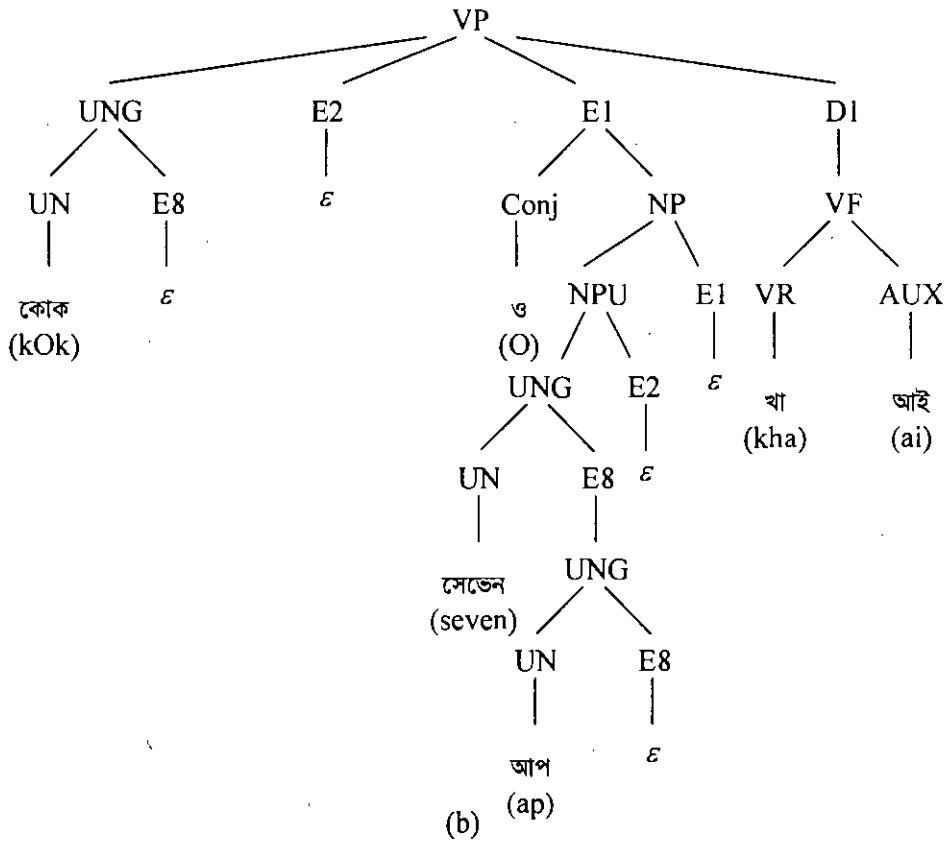


Figure 4.15: Tree derivation with non-ambiguous grammar for verb phrase with predictive parsing (continued).



Continued Figure 4.15: Tree derivation with non-ambiguous grammar for verb phrase with predictive parsing.

Ultimately, non-ambiguous grammar for verb phrase with predictive parsing becomes,

- VP → D3 | UNG E2 E1 D1
- D1 → VF | AP VF
- D2 → VF | D4
- D3 → VF | AP D2 | D4 | DEMO E3 D4 | SPR E4 D4
- D4 → NW E2 E1 D1
- VF → VR AUX

Now, we are in a position to give some examples of verb phrase as underlined in the sentences “আমি ভাত ও মাছ খাই” (ami vat O machh khai), “আমি কোক ও সেভেন আপ খাই” (ami kOk O seven up khai). Tree derivations of the examples are demonstrated in figure 4.15.

4.4 Non-ambiguous Grammar for Simple Sentence

Existing context-free grammar for simple sentence in Bangla natural language is written as:

$$SS \rightarrow NP VP$$

Difficulty using this rule is ambiguity. In non-dictionary word does not exist in a sentence, then the rule works fine. But ambiguity problem may occur for non-dictionary words, as NP may end with an unknown word and VP may start with an unknown word. In that case, problem arises, how to fit the unknown word.

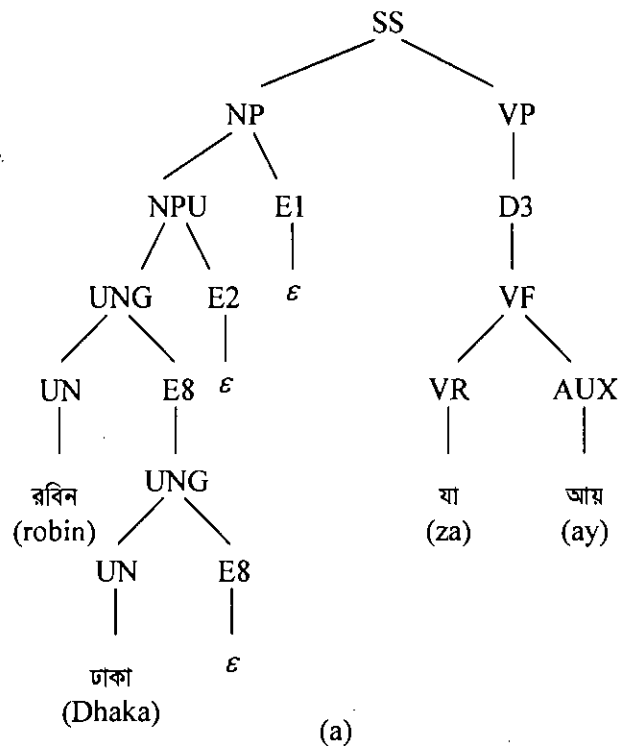
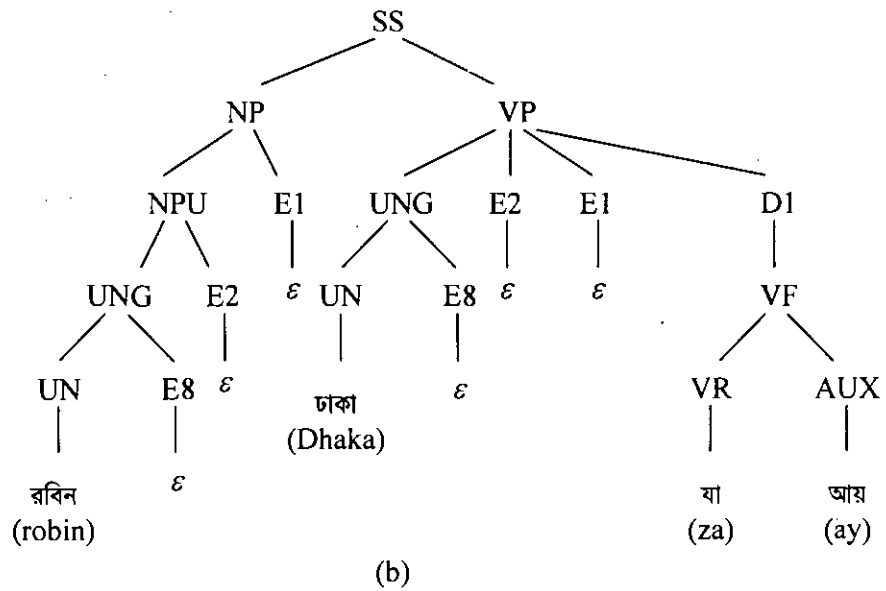


Figure 4.16: Ambiguity in the rule $SS \rightarrow NP VP$ (continued).



Continued Figure 4.16: Ambiguity in the rule $SS \rightarrow NP VP$.

For example, the simple sentence “রবিন ঢাকা যায়” (robin Dhaka zay) contains two non-dictionary words “রবিন” (robin) and “ঢাকা” (Dhaka). Here, obviously “রবিন” (robin) is a part of noun phrase and “যায়” (zay) is a part of verb phrase. But problem arises for the non-dictionary word “ঢাকা” (Dhaka), which can be fit both in noun phrase and verb phrase. As a result, the example sentence can be parsed in two possible ways, as demonstrated in figure 4.16.

The main reason of ambiguity problem is, we consider unknown words as a group, when some unknown words are found together in a sentence. But, in the real situation unknown words may be in different groups. In the sentence, “রবিন ঢাকা যায়” (robin Dhaka zay), conceptually “রবিন” (robin) is noun phrase and “ঢাকা যায়” (Dhaka zay) is verb phrase. In another sentence, “রবিন মিলফোর্ড যায়” (robin milofOrrD zay), “রবিন মিলফোর্ড” (robin milofOrrD) is noun phrase and “যায়” (zay) is verb phrase. It is not possible for a parser to decide whether these unknown words “ঢাকা” (Dhaka), “মিলফোর্ড” (milofOrrD) are belong to noun phrase or verb phrase, as these are unknown to the parser.

A possible solution is to make the unknown words left associative. In a unknown word group, if first word is identified as a part of noun phrase, then other words in the group will also be considered as part of noun phrase, and vice versa. This solution will certainly create some conceptual mistakes, like for the sentence “রবিন ঢাকা যায়” (robin Dhaka zay), “রবিন ঢাকা” (robin Dhaka) will be considered as noun phrase by the parser, though conceptually only “রবিন” (robin) is noun phrase.

Though conceptual mistake may occur, but it will be possible to continue parsing and ambiguity will be removed. We can apply some heuristics to minimize the conceptual mistakes. An idea may be to apply some learning mechanism. The learning mechanism will identify unknown word groups. If we apply such kind of learning mechanism in paragraph level, the parser may be able to identify that “রবিন” (robin) is a part of noun phrase and “ঢাকা” (Dhaka) is a part of verb phrase. Though it does not ensure 100% accuracy.

To eliminate ambiguity in SS, we have to merge UNG, for the case, when it is applicable both as last symbol of NP and first symbol of VP. To do this merging, firstly, we have to expand the rules of NP and VP as follows,

$$SS \rightarrow NP VP$$

$$\Rightarrow$$

$$SS \rightarrow NPU E1 VP \quad \text{(expanding rule for NP)}$$

$$\Rightarrow$$

$$SS \rightarrow NPU Conj NP VP \mid NPU VP \quad \text{(expanding rule for E1)}$$

$$\Rightarrow$$

$$SS \rightarrow NPU Conj NP VP \mid NW E2 VP \mid PRE NW E2 VP \mid UNG E2 VP \quad \text{(expanding rule for NPU)}$$

$$\Rightarrow$$

$$SS \rightarrow NPU Conj NP VP \mid NW E2 VP \mid PRE NW E2 VP \mid UNG BivE NP VP \mid UNG Biv VP \mid UNG VP \quad \text{(expanding rule for E2)}$$

$$\Rightarrow$$

$$SS \rightarrow NPU \text{ Conj } SS \mid NW \text{ E2 } VP \mid PRE \text{ NW } \text{ E2 } VP \mid UNG \text{ BivE } SS \mid \\ UNG \text{ Biv } VP \mid UNG \text{ VP}$$

(replacing NP VP by SS, as NP VP is ambiguous, but SS will be made non-ambiguous)

$$\Rightarrow$$

$$SS \rightarrow NPU \text{ Conj } SS \mid NW \text{ E2 } VP \mid PRE \text{ NW } \text{ E2 } VP \mid UNG \text{ BivE } SS \mid \\ UNG \text{ Biv } VP \mid UNG \text{ D3 } \mid UNG \text{ UNG } \text{ E2 } \text{ E1 } \text{ D1} \quad (\text{expanding rule for VP})$$

$$\Rightarrow$$

$$SS \rightarrow NPU \text{ Conj } SS \mid NW \text{ E2 } VP \mid PRE \text{ NW } \text{ E2 } VP \mid UNG \text{ BivE } SS \mid \\ UNG \text{ Biv } VP \mid UNG \text{ D3 } \mid UNG \text{ E2 } \text{ E1 } \text{ D1}$$

(replacing UNG UNG by UNG, as we are considering consecutive unknown words as a single group, as discussed earlier in this section)

$$\Rightarrow$$

$$SS \rightarrow NW \text{ E2 } \text{ Conj } SS \mid PRE \text{ NW } \text{ E2 } \text{ Conj } SS \mid UNG \text{ E2 } \text{ Conj } SS \mid \\ NW \text{ E2 } VP \mid PRE \text{ NW } \text{ E2 } VP \mid UNG \text{ BivE } SS \mid UNG \text{ Biv } VP \mid UNG \text{ D3 } \mid \\ UNG \text{ E2 } \text{ E1 } \text{ D1} \quad (\text{expanding rule for NPU})$$

$$\Rightarrow$$

$$SS \rightarrow NW \text{ E2 } C1 \mid PRE \text{ NW } \text{ E2 } C1 \mid UNG \text{ C2} \quad (\text{left factoring})$$

where,

$$C1 \rightarrow VP \mid \text{Conj } SS$$

$$C2 \rightarrow \text{E2 } \text{Conj } SS \mid \text{BivE } SS \mid \text{Biv } VP \mid \text{D3} \mid \text{E2 } \text{E1 } \text{D1}$$

Now, SS becomes non-ambiguous, at the same time usable for predictive parser. But C2 does not support predictive parsing. We can transform it usable for predictive parser by the following derivations,

$$C2 \rightarrow \text{E2 } \text{Conj } SS \mid \text{BivE } SS \mid \text{Biv } VP \mid \text{D3} \mid \text{E2 } \text{E1 } \text{D1}$$

$$\Rightarrow$$

$$C2 \rightarrow \text{BivE } NP \text{ Conj } SS \mid \text{Biv } \text{Conj } SS \mid \text{Conj } SS \mid \text{BivE } SS \mid \text{Biv } VP \mid \text{D3} \mid \\ \text{E2 } \text{Conj } NP \text{ D1} \quad (\text{expanding rule for E1 and E2})$$

$$\Rightarrow$$

$C2 \rightarrow \text{BivE NP Conj SS} \mid \text{Biv Conj SS} \mid \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv VP} \mid \text{D3} \mid$
 $\text{BivE NP Conj NP D1} \mid \text{Biv Conj NP D1} \mid \text{Conj NP D1}$

(expanding rule for E2)

\Rightarrow

$C2 \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv Conj SS} \mid \text{Biv VP} \mid \text{D3}$

(we can represent “BivE NP Conj SS | BivE SS | BivE NP Conj NP D1” by only BivE SS, because others are derivations of BivE SS, similarly, we can represent “Biv Conj SS | Biv Conj NP D1” by Biv Conj SS, again we can represent “Conj SS | Conj NP D1” by Conj SS)

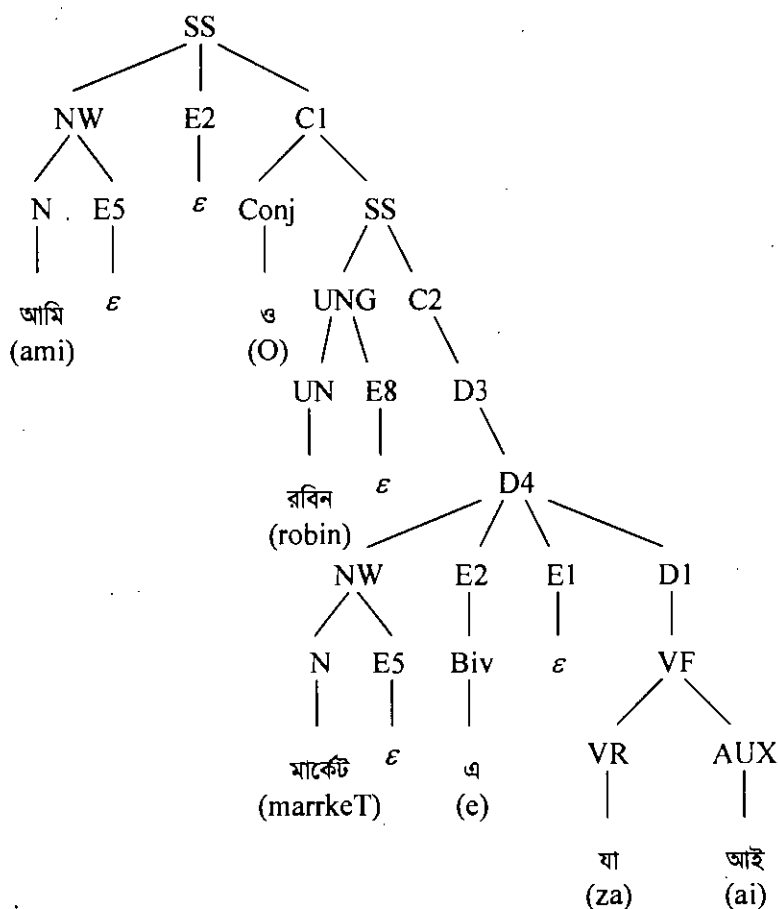


Figure 4.17: Tree derivation with non-ambiguous grammar for simple sentence with predictive parsing.

⇒

$C2 \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv C1} \mid D3$

(using the rule $C1 \rightarrow \text{VP} \mid \text{Conj SS}$, $\text{Biv VP} \mid \text{Biv Conj SS}$ can be merged into Biv C1)

Ultimately, non-ambiguous grammar for simple sentence with predictive parsing becomes,

$SS \rightarrow \text{NW E2 C1} \mid \text{PRE NW E2 C1} \mid \text{UNG C2}$

$C1 \rightarrow \text{VP} \mid \text{Conj SS}$

$C2 \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv C1} \mid D3$

Now, we are in a position to give some examples of simple sentence like “আমি ও রবিন মার্কেটে যাই” (ami O robin marrkeTe zai), “আমি ও আমার বন্ধু বসুন্ধরা সিটি তে যাই” (ami O amar bondhu bosundhora siTi te zai), for which tree derivations are demonstrated in figure 4.17.

As, a simple sentence is composed of noun phrase, verb phrase and optionally adjective phrase, we can present grammar for simple sentence all together as follows,

$SS \rightarrow \text{NW E2 C1} \mid \text{PRE NW E2 C1} \mid \text{UNG C2}$

$C1 \rightarrow \text{VP} \mid \text{Conj SS}$

$C2 \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv C1} \mid D3$

$VP \rightarrow D3 \mid \text{UNG E2 E1 D1}$

$D1 \rightarrow \text{VF} \mid \text{AP VF}$

$D2 \rightarrow \text{VF} \mid D4$

$D3 \rightarrow \text{VF} \mid \text{AP D2} \mid D4 \mid \text{DEMO E3 D4} \mid \text{SPR E4 D4}$

$D4 \rightarrow \text{NW E2 E1 D1}$

$\text{VF} \rightarrow \text{VR AUX}$

$\text{NP} \rightarrow \text{NPU E1}$

$E1 \rightarrow \text{Conj NP} \mid \epsilon$

$\text{NPU} \rightarrow \text{NW E2} \mid \text{PRE NW E2} \mid \text{UNG E2}$

$E2 \rightarrow \text{BivE NP} \mid \text{Biv} \mid \epsilon$

$$\text{PRE} \rightarrow \text{DEMO E3} \mid \text{SPR E4} \mid \text{AP}$$

$$\text{E3} \rightarrow \text{SPR E4} \mid \text{AP} \mid \varepsilon$$

$$\text{E4} \rightarrow \text{AP} \mid \varepsilon$$

$$\text{NW} \rightarrow \text{N E5}$$

$$\text{E5} \rightarrow \text{DET} \mid \text{PM} \mid \varepsilon$$

$$\text{SPR} \rightarrow \text{QFR E6}$$

$$\text{E6} \rightarrow \text{PP} \mid \varepsilon$$

$$\text{DEMO} \rightarrow \text{DD E7} \mid \text{DO}$$

$$\text{E7} \rightarrow \text{DO} \mid \varepsilon$$

$$\text{UNG} \rightarrow \text{UN E8}$$

$$\text{E8} \rightarrow \text{UNG} \mid \varepsilon$$

$$\text{AP} \rightarrow \text{AD F1}$$

$$\text{F1} \rightarrow \text{AP} \mid \varepsilon$$

4.5 Remarks

In this chapter, we have designed non-ambiguous Bangla grammar for simple sentences with predictive parsing. Moreover, we also added using of unknown words in Bangla grammar. This feature certainly directs to more effective parsing. Later this grammar will be merged as a part of comprehensive Bangla grammar.

Non-ambiguous Grammar for Complex Sentence

5.1 Existing Grammar for Complex Sentence

Recalling from chapter 3 (section 3.7), context-free grammar for complex sentence in Bangla natural language is presented as,

$$CS \rightarrow DC IC$$

$$CS \rightarrow IC DC$$

$$DC \rightarrow NP (SUBORD) VP$$

$$DC \rightarrow (SUBORD) SS$$

$$IC \rightarrow NP (SUBCOM) VP$$

$$IC \rightarrow (SUBCOM) SS$$

Main problems using the grammar for complex sentence are ambiguity and non-predictivity. A complex sentence is composed of several noun phrases and verb phrases. Ambiguity occurs in complex sentence, because of ambiguity in grammar of noun phrase and verb phrase. In the previous chapter, ambiguity problem of noun phrase and verb phrase is solved. Therefore, grammar for complex sentence is now non-ambiguous. But the grammar still does not support predictive parsing. For example, if we want to parse the complex sentence, “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi eso), for the first word “আমি” (ami), it is not possible to determine the rule applicable, without watching next words. Here, the two rules $CS \rightarrow DC IC$ and $DC \rightarrow NP (SUBORD) VP$ may be applicable, as these two rule allows “আমি” (ami) as first word. Again, two other rules $CS \rightarrow IC DC$ and $IC \rightarrow NP (SUBCOM) VP$ may also be applicable, as they also allows “আমি” (ami) as first word. Similarly, non-predictive nature also occurs for next words.

In this chapter, we will basically discuss how to transform the grammar for complex sentence usable for predictive parsing.

5.2 Identifying Complex Sentence Patterns

A complex sentence can be written in several possible ways. Following from existing grammar for complex sentence, a sentence “আমি গেলে তুমি এসো” (ami gele tumi esO) can be written in following possible ways,

- “আমি গেলে তুমি এসো” (ami gele tumi esO)
- “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO)
- “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO)
- “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO)
- “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO)
- “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO)
- “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO)
- “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO)
- “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO)
- “তুমি এসো আমি গেলে” (tumi esO ami gele)
- “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai)
- “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai)
- “তুমি তবে এসো আমি গেলে” (tumi tobe esO ami gele)
- “তুমি তবে এসো আমি যদি যাই” (tumi tobe esO ami zodi zai)
- “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai)
- “তবে তুমি এসো আমি গেলে” (tobe tumi esO ami gele)
- “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai)
- “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai)

Here, we can exclude following three patterns,

- “তুমি এসো আমি গেলে” (tumi esO ami gele)
- “তুমি তবে এসো আমি গেলে” (tumi tobe esO ami gele)
- “তবে তুমি এসো আমি গেলে” (tobe tumi esO ami gele)

as these patterns are unusual.

So, valid patterns are identified as follows,

- “আমি গেলে তুমি এসো” (ami gele tumi esO)
- “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO)
- “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO)
- “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO)
- “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO)
- “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO)
- “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO)
- “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO)
- “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO)
- “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai)
- “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai)
- “তুমি তবে এসো আমি যদি যাই” (tumi tobe esO ami zodi zai)
- “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai)
- “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai)
- “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai)

Now, we can re-write the grammar for complex sentence using the valid sentence patterns as,

- CS → SS SS | SS NP SUBCOM VP | SS SUBCOM SS |
- NP SUBORD VP SS | NP SUBORD VP NP SUBCOM VP |
- NP SUBORD VP SUBCOM SS | SUBORD SS SS |
- SUBORD SS NP SUBCOM VP | SUBORD SS SUBCOM SS |
- SS NP SUBORD VP | SS SUBORD SS |
- NP SUBCOM VP NP SUBORD VP | NP SUBCOM VP SUBORD SS |
- SUBCOM SS NP SUBORD VP | SUBCOM SS SUBORD SS

This rule is certainly non-ambiguous, though does not support predictive parsing. In the next section, we will derive grammar for complex sentence with predictive parsing.

5.3 Non-ambiguous Grammar for Complex Sentence

We will apply left factoring technique on the grammar for complex sentence, proposed in the previous section (5.2). If we apply left factoring on the rule for CS, the derivations will be as,

$$\begin{aligned} \text{CS} &\rightarrow \text{SS SS} \mid \text{SS NP SUBCOM VP} \mid \text{SS SUBCOM SS} \mid \\ &\text{NP SUBORD VP SS} \mid \text{NP SUBORD VP NP SUBCOM VP} \mid \\ &\text{NP SUBORD VP SUBCOM SS} \mid \text{SUBORD SS SS} \mid \\ &\text{SUBORD SS NP SUBCOM VP} \mid \text{SUBORD SS SUBCOM SS} \mid \\ &\text{SS NP SUBORD VP} \mid \text{SS SUBORD SS} \mid \\ &\text{NP SUBCOM VP NP SUBORD VP} \mid \text{NP SUBCOM VP SUBORD SS} \mid \\ &\text{SUBCOM SS NP SUBORD VP} \mid \text{SUBCOM SS SUBORD SS} \end{aligned}$$

⇒

$$\text{CS} \rightarrow \text{SS B1} \mid \text{NP B2} \mid \text{SUBORD SS B3} \mid \text{SUBCOM SS B4}; \text{ left factoring}$$

where,

$$\text{B1} \rightarrow \text{SS} \mid \text{NP SUBCOM VP} \mid \text{SUBCOM SS} \mid \text{NP SUBORD VP} \mid \text{SUBORD SS}$$

$$\begin{aligned} \text{B2} &\rightarrow \text{SUBORD VP SS} \mid \text{SUBORD VP NP SUBCOM VP} \mid \\ &\text{SUBORD VP SUBCOM SS} \mid \text{SUBCOM VP NP SUBORD VP} \mid \\ &\text{SUBCOM VP SUBORD SS} \end{aligned}$$

$$\text{B3} \rightarrow \text{SS} \mid \text{NP SUBCOM VP} \mid \text{SUBCOM SS}$$

$$\text{B4} \rightarrow \text{NP SUBORD VP} \mid \text{SUBORD SS}$$

⇒

$$\begin{aligned} \text{CS} &\rightarrow \text{NW E2 C1 B1} \mid \text{PRE NW E2 C1 B1} \mid \text{UNG C2 B1} \mid \text{NP B2} \mid \\ &\text{SUBORD SS B3} \mid \text{SUBCOM SS B4} \end{aligned} \quad (\text{expanding rule for SS})$$

⇒

$$\begin{aligned} \text{CS} &\rightarrow \text{NW E2 C1 B1} \mid \text{PRE NW E2 C1 B1} \mid \text{UNG C2 B1} \mid \text{NPU E1 B2} \mid \\ &\text{SUBORD SS B3} \mid \text{SUBCOM SS B4} \end{aligned} \quad (\text{expanding rule for NP})$$

⇒

$$\begin{aligned} \text{CS} &\rightarrow \text{NW E2 C1 B1} \mid \text{PRE NW E2 C1 B1} \mid \text{UNG C2 B1} \mid \text{NW E2 E1 B2} \mid \\ &\text{PRE NW E2 E1 B2} \mid \text{UNG E2 E1 B2} \mid \text{SUBORD SS B3} \mid \text{SUBCOM SS B4} \end{aligned} \quad (\text{expanding rule for NPU})$$

⇒

CS → NW E2 B5 | PRE NW E2 B5 | UNG B6 | SUBORD SS B3 |
SUBCOM SS B4 (left factoring)

where,

B5 → C1 B1 | E1 B2

B6 → C2 B1 | E2 E1 B2

Rule for CS is now usable for predictive parsing, but some of the derived rules do not support predictive parsing. If we consider the rule for B1, it does not support predictive parsing. We can transform it to be usable for predictive parsing by the following derivations,

B1 → SS | NP SUBCOM VP | SUBCOM SS | NP SUBORD VP |
SUBORD SS

⇒

B1 → SS | NP B7 | SUBORD SS | SUBCOM SS (left factoring)

where,

B7 → SUBORD VP | SUBCOM VP

⇒

B1 → NW E2 C1 | PRE NW E2 C1 | UNG C2 | NP B7 | SUBORD SS |
SUBCOM SS (expanding rule for SS)

⇒

B1 → NW E2 C1 | PRE NW E2 C1 | UNG C2 | NPU E1 B7 | SUBORD SS |
SUBCOM SS (expanding rule for NP)

⇒

B1 → NW E2 C1 | PRE NW E2 C1 | UNG C2 | NW E2 E1 B7 |
PRE NW E2 E1 B7 | UNG E2 E1 B7 | SUBORD SS | SUBCOM SS
(expanding rule for NPU)

⇒

B1 → NW E2 B8 | PRE NW E2 B8 | UNG B9 | SUBORD SS |
SUBCOM SS (left factoring)

where,

B8 → C1 | E1 B7

$$B9 \rightarrow C2 \mid E2 E1 B7$$

Rule for B2 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$\begin{aligned} B2 &\rightarrow \text{SUBORD VP SS} \mid \text{SUBORD VP NP SUBCOM VP} \mid \\ &\text{SUBORD VP SUBCOM SS} \mid \text{SUBCOM VP NP SUBORD VP} \mid \\ &\text{SUBCOM VP SUBORD SS} \end{aligned}$$

$$\Rightarrow$$

$$B2 \rightarrow \text{SUBORD VP B3} \mid \text{SUBCOM VP B4} \quad (\text{left factoring})$$

Rule for B3 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$B3 \rightarrow \text{SS} \mid \text{NP SUBCOM VP} \mid \text{SUBCOM SS}$$

$$\Rightarrow$$

$$\begin{aligned} B3 &\rightarrow \text{NW E2 C1} \mid \text{PRE NW E2 C1} \mid \text{UNG C2} \mid \text{NP SUBCOM VP} \mid \\ &\text{SUBCOM SS} \quad (\text{expanding rule for SS}) \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} B3 &\rightarrow \text{NW E2 C1} \mid \text{PRE NW E2 C1} \mid \text{UNG C2} \mid \text{NPU E1 SUBCOM VP} \mid \\ &\text{SUBCOM SS} \quad (\text{expanding rule for NP}) \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} B3 &\rightarrow \text{NW E2 C1} \mid \text{PRE NW E2 C1} \mid \text{UNG C2} \mid \text{NW E2 E1 SUBCOM VP} \mid \\ &\text{PRE NW E2 E1 SUBCOM VP} \mid \text{UNG E2 E1 SUBCOM VP} \mid \text{SUBCOM SS} \\ &\quad (\text{expanding rule for NPU}) \end{aligned}$$

$$\Rightarrow$$

$$B3 \rightarrow \text{NW E2 B10} \mid \text{PRE NW E2 B10} \mid \text{UNG B11} \mid \text{SUBCOM SS} \quad (\text{left factoring})$$

where,

$$B10 \rightarrow C1 \mid E1 \text{ SUBCOM VP}$$

$$B11 \rightarrow C2 \mid E2 E1 \text{ SUBCOM VP}$$

Rule for B4 supports predictive parsing. But rule for B5 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$B5 \rightarrow C1 B1 \mid E1 B2$$

$$\Rightarrow$$

$$B5 \rightarrow VP B1 \mid Conj SS B1 \mid E1 B2 \quad (\text{expanding rule for } C1)$$

$$\Rightarrow$$

$$B5 \rightarrow VP B1 \mid Conj SS B1 \mid Conj NP B2 \mid B2 \quad (\text{expanding rule for } E1)$$

$$\Rightarrow$$

$$B5 \rightarrow VP B1 \mid Conj B12 \mid B2 \quad (\text{left factoring})$$

where,

$$B12 \rightarrow SS B1 \mid NP B2$$

Rule for B6 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$B6 \rightarrow C2 B1 \mid E2 E1 B2$$

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE SS B1 \mid Biv C1 B1 \mid D3 B1 \mid E2 E1 B2 \quad (\text{expanding rule for } C2)$$

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE SS B1 \mid Biv C1 B1 \mid D3 B1 \mid BivE NP E1 B2 \mid Biv E1 B2 \mid E1 B2 \quad (\text{expanding rule for } E2)$$

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE SS B1 \mid Biv C1 B1 \mid D3 B1 \mid BivE NP E1 B2 \mid Biv E1 B2 \mid Conj SS B2 \mid B2 \quad (\text{expanding rule for } E1)$$

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE SS B1 \mid Biv C1 B1 \mid D3 B1 \mid BivE NP B2 \mid Biv E1 B2 \mid Conj SS B2 \mid B2$$

(replacing "NP E1" by "NP", as both are equivalent)

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE SS B1 \mid Biv C1 B1 \mid D3 B1 \mid BivE NP B2 \mid Biv E1 B2 \mid B2$$

("Conj SS B1" is superset of "Conj SS B2", so we can omit "Conj SS B2")

$$\Rightarrow$$

$$B6 \rightarrow Conj SS B1 \mid BivE B12 \mid Biv B5 \mid D3 B1 \mid B2 \quad (\text{left factoring})$$

Rule for B7 supports predictive parsing. But rule for B8 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$\begin{aligned}
 & B8 \rightarrow C1 \mid E1 B7 \\
 & \Rightarrow \\
 & B8 \rightarrow VP \mid Conj SS \mid E1 B7 \quad (\text{expanding rule for } C1) \\
 & \Rightarrow \\
 & B8 \rightarrow VP \mid Conj SS \mid Conj NP B7 \mid B7 \quad (\text{expanding rule for } E1) \\
 & \Rightarrow \\
 & B8 \rightarrow VP \mid Conj B13 \mid B7 \quad (\text{left factoring}) \\
 & \text{where,} \\
 & B13 \rightarrow SS \mid NP B7
 \end{aligned}$$

Rule for B9 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$\begin{aligned}
 & B9 \rightarrow C2 \mid E2 E1 B7 \\
 & \Rightarrow \\
 & B9 \rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3 \mid E2 E1 B7 \quad (\text{expanding rule for } C2) \\
 & \Rightarrow \\
 & B9 \rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3 \mid BivE NP E1 B7 \mid Biv E1 B7 \mid \\
 & E1 B7 \quad (\text{expanding rule for } E2) \\
 & \Rightarrow \\
 & B9 \rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3 \mid BivE NP E1 B7 \mid Biv E1 B7 \mid \\
 & Conj NP B7 \mid B7 \quad (\text{expanding rule for } E1) \\
 & \Rightarrow \\
 & B9 \rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3 \mid BivE NP B7 \mid Biv E1 B7 \mid \\
 & Conj NP B7 \mid B7 \quad (\text{replacing "NP E1" by "NP", as both are equivalent}) \\
 & \Rightarrow \\
 & B9 \rightarrow Conj B13 \mid BivE B13 \mid Biv B8 \mid D3 \mid B7 \quad (\text{left factoring})
 \end{aligned}$$

Rule for B10 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$\begin{aligned}
 & B_{10} \rightarrow C_1 \mid E_1 \text{ SUBCOM VP} \\
 & \Rightarrow \\
 & B_{10} \rightarrow \text{VP} \mid \text{Conj SS} \mid E_1 \text{ SUBCOM VP} \quad (\text{expanding rule for } C_1) \\
 & \Rightarrow \\
 & B_{10} \rightarrow \text{VP} \mid \text{Conj SS} \mid \text{Conj NP SUBCOM VP} \mid \text{SUBCOM VP} \\
 & \quad \quad \quad (\text{expanding rule for } E_2) \\
 & \Rightarrow \\
 & B_{10} \rightarrow \text{VP} \mid \text{Conj } B_{14} \mid \text{SUBCOM VP} \quad (\text{left factoring}) \\
 & \text{where,} \\
 & B_{14} \rightarrow \text{SS} \mid \text{NP SUBCOM VP}
 \end{aligned}$$

Rule for B_{11} does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$\begin{aligned}
 & B_{11} \rightarrow C_2 \mid E_2 E_1 \text{ SUBCOM VP} \\
 & \Rightarrow \\
 & B_{11} \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv } C_1 \mid D_3 \mid E_2 E_1 \text{ SUBCOM VP} \\
 & \quad \quad \quad (\text{expanding rule for } C_2) \\
 & \Rightarrow \\
 & B_{11} \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv } C_1 \mid D_3 \mid \text{BivE NP } E_1 \text{ SUBCOM VP} \mid \\
 & \text{Biv } E_1 \text{ SUBCOM VP} \mid E_1 \text{ SUBCOM VP} \quad (\text{expanding rule for } E_2) \\
 & \Rightarrow \\
 & B_{11} \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv } C_1 \mid D_3 \mid \text{BivE NP SUBCOM VP} \mid \\
 & \text{Biv } E_1 \text{ SUBCOM VP} \mid E_1 \text{ SUBCOM VP} \\
 & \quad \quad \quad (\text{replacing "NP } E_1" \text{ by "NP", as both are equivalent}) \\
 & \Rightarrow \\
 & B_{11} \rightarrow \text{Conj SS} \mid \text{BivE SS} \mid \text{Biv } C_1 \mid D_3 \mid \text{BivE NP SUBCOM VP} \mid \\
 & \text{Biv } E_1 \text{ SUBCOM VP} \mid \text{Conj NP SUBCOM VP} \mid \text{SUBCOM VP} \\
 & \quad \quad \quad (\text{expanding rule for } E_1) \\
 & \Rightarrow \\
 & B_{11} \rightarrow \text{Conj } B_{14} \mid \text{BivE } B_{14} \mid \text{Biv } B_{10} \mid D_3 \mid \text{SUBCOM VP} \quad (\text{left factoring})
 \end{aligned}$$

Rule for B12 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$B12 \rightarrow SS B1 \mid NP B2$$

$$\Rightarrow$$

$$B12 \rightarrow NW E2 C1 B1 \mid PRE NW E2 C1 B1 \mid UNG C2 B1 \mid NP B2$$

(expanding rule for SS)

$$\Rightarrow$$

$$B12 \rightarrow NW E2 C1 B1 \mid PRE NW E2 C1 B1 \mid UNG C2 B1 \mid NPU E1 B2$$

(expanding rule for NP)

$$\Rightarrow$$

$$B12 \rightarrow NW E2 C1 B1 \mid PRE NW E2 C1 B1 \mid UNG C2 B1 \mid NW E2 E1 B2 \mid$$

$$PRE NW E2 E1 B2 \mid UNG E2 E1 B2$$

(expanding rule for NPU)

$$\Rightarrow$$

$$B12 \rightarrow NW E2 B5 \mid PRE NW E2 B5 \mid UNG B6$$

(left factoring)

Rule for B13 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$B13 \rightarrow SS \mid NP B7$$

$$\Rightarrow$$

$$B13 \rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2 \mid NP B7$$

(expanding rule for SS)

$$\Rightarrow$$

$$B13 \rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2 \mid NPU E1 B7$$

(expanding rule for NP)

$$\Rightarrow$$

$$B13 \rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2 \mid NW E2 E1 B7 \mid$$

$$PRE NW E2 E1 B7 \mid UNG E2 E1 B7$$

(expanding rule for NPU)

$$\Rightarrow$$

$$B13 \rightarrow NW E2 B8 \mid PRE NW E2 B8 \mid UNG B9$$

(left factoring)

Rule for B14 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$B_{14} \rightarrow SS \mid NP \text{ SUBCOM } VP$
 \Rightarrow
 $B_{14} \rightarrow NW \ E2 \ C1 \mid PRE \ NW \ E2 \ C1 \mid UNG \ C2 \mid NP \ \text{SUBCOM } VP$
 (expanding rule for SS)
 \Rightarrow
 $B_{14} \rightarrow NW \ E2 \ C1 \mid PRE \ NW \ E2 \ C1 \mid UNG \ C2 \mid NPU \ E1 \ \text{SUBCOM } VP$
 (expanding rule for NP)
 \Rightarrow
 $B_{14} \rightarrow NW \ E2 \ C1 \mid PRE \ NW \ E2 \ C1 \mid UNG \ C2 \mid NW \ E2 \ E1 \ \text{SUBCOM } VP \mid$
 $PRE \ NW \ E2 \ E1 \ \text{SUBCOM } VP \mid UNG \ E2 \ E1 \ \text{SUBCOM } VP$
 (expanding rule for NPU)
 \Rightarrow
 $B_{14} \rightarrow NW \ E2 \ B_{10} \mid PRE \ NW \ E2 \ B_{10} \mid UNG \ B_{11}$ (left factoring)

Now all the rules of grammar for complex sentence supports predictive parsing. We can present grammar for complex sentence as follows,

$CS \rightarrow NW \ E2 \ B_5 \mid PRE \ NW \ E2 \ B_5 \mid UNG \ B_6 \mid SUBORD \ SS \ B_3 \mid$
 $SUBCOM \ SS \ B_4$
 $B_1 \rightarrow NW \ E2 \ B_8 \mid PRE \ NW \ E2 \ B_8 \mid UNG \ B_9 \mid SUBORD \ SS \mid$
 $SUBCOM \ SS$
 $B_2 \rightarrow SUBORD \ VP \ B_3 \mid SUBCOM \ VP \ B_4$
 $B_3 \rightarrow NW \ E2 \ B_{10} \mid PRE \ NW \ E2 \ B_{10} \mid UNG \ B_{11} \mid SUBCOM \ SS$
 $B_4 \rightarrow NP \ SUBORD \ VP \mid SUBORD \ SS$
 $B_5 \rightarrow VP \ B_1 \mid Conj \ B_{12} \mid B_2$
 $B_6 \rightarrow Conj \ SS \ B_1 \mid BivE \ B_{12} \mid Biv \ B_5 \mid D3 \ B_1 \mid B_2$
 $B_7 \rightarrow SUBORD \ VP \mid SUBCOM \ VP$
 $B_8 \rightarrow VP \mid Conj \ B_{13} \mid B_7$
 $B_9 \rightarrow Conj \ B_{13} \mid BivE \ B_{13} \mid Biv \ B_8 \mid D3 \mid B_7$
 $B_{10} \rightarrow VP \mid Conj \ B_{14} \mid SUBCOM \ VP$
 $B_{11} \rightarrow Conj \ B_{14} \mid BivE \ B_{14} \mid Biv \ B_{10} \mid D3 \mid SUBCOM \ VP$
 $B_{12} \rightarrow NW \ E2 \ B_5 \mid PRE \ NW \ E2 \ B_5 \mid UNG \ B_6$
 $B_{13} \rightarrow NW \ E2 \ B_8 \mid PRE \ NW \ E2 \ B_8 \mid UNG \ B_9$

B14 → NW E2 B10 | PRE NW E2 B10 | UNG B11

Now, we can give example of different variations of complex sentences like, “আমি গেলে তুমি এসো” (ami gele tumi esO), “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO), “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO), “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO), “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO), “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO), “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO), “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO), “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai), “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai), “তুমি তবে এসো আমি যদি যাই” (tumi tobe esO ami zodi zai), “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai), “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai), “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai), as demonstrated in figure 5.1 – 5.15.

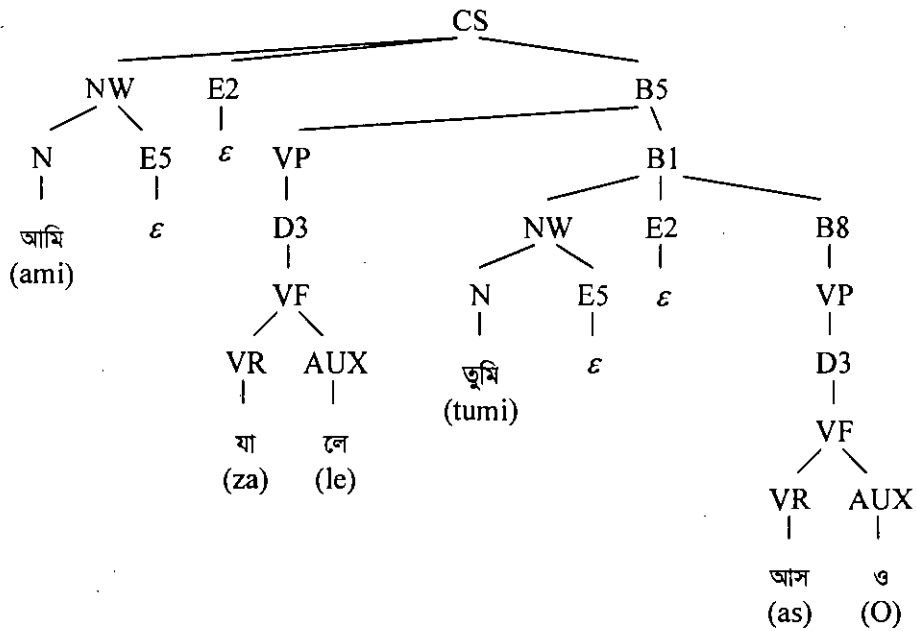


Figure 5.1: Tree derivation for the sentence “আমি গেলে তুমি এসো” (ami gele tumi esO) using non-ambiguous grammar with predictive parsing.

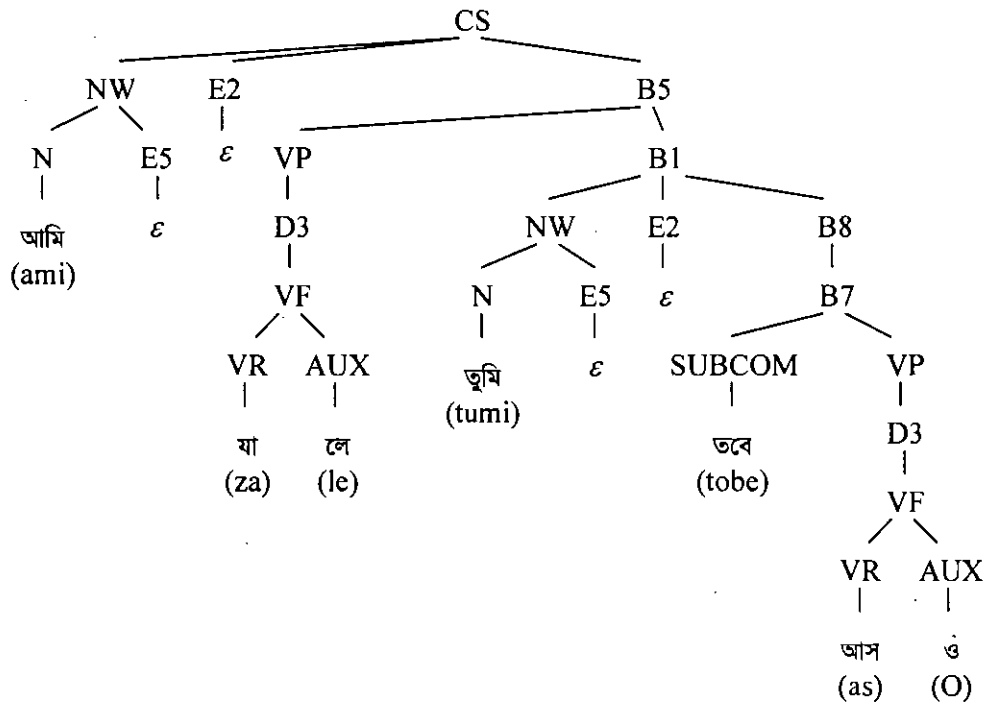


Figure 5.2: Tree derivation for the sentence “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO) using non-ambiguous grammar with predictive parsing.

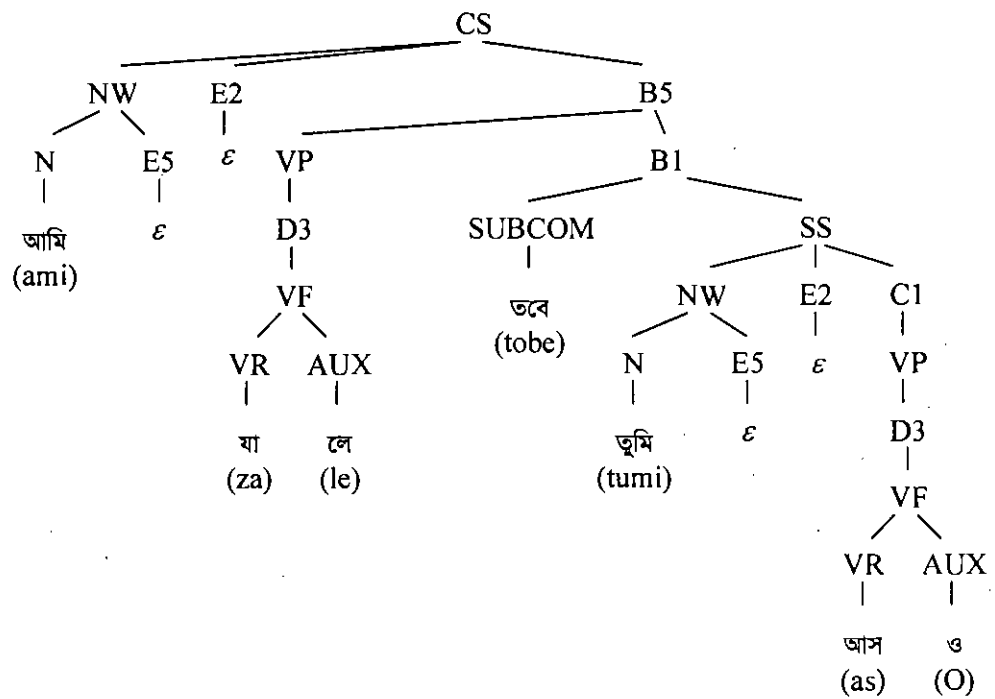


Figure 5.3: Tree derivation for the sentence “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO) using non-ambiguous grammar with predictive parsing.

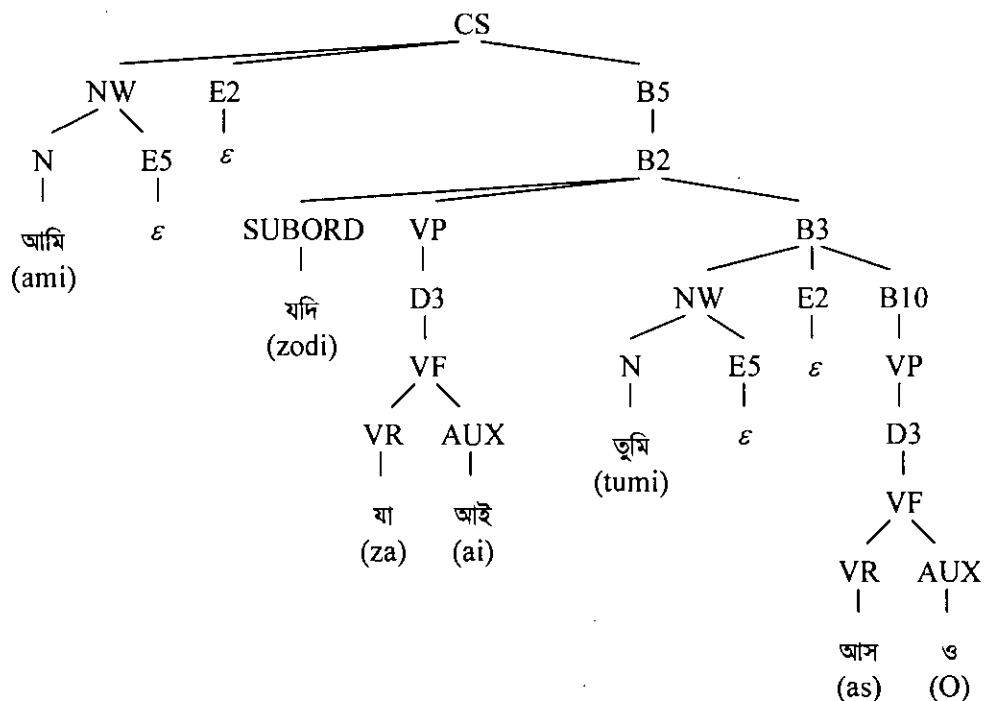


Figure 5.4: Tree derivation for the sentence “আমি যদি যাই তুমি এসো” (ami zodi zai tumi esO) using non-ambiguous grammar with predictive parsing.

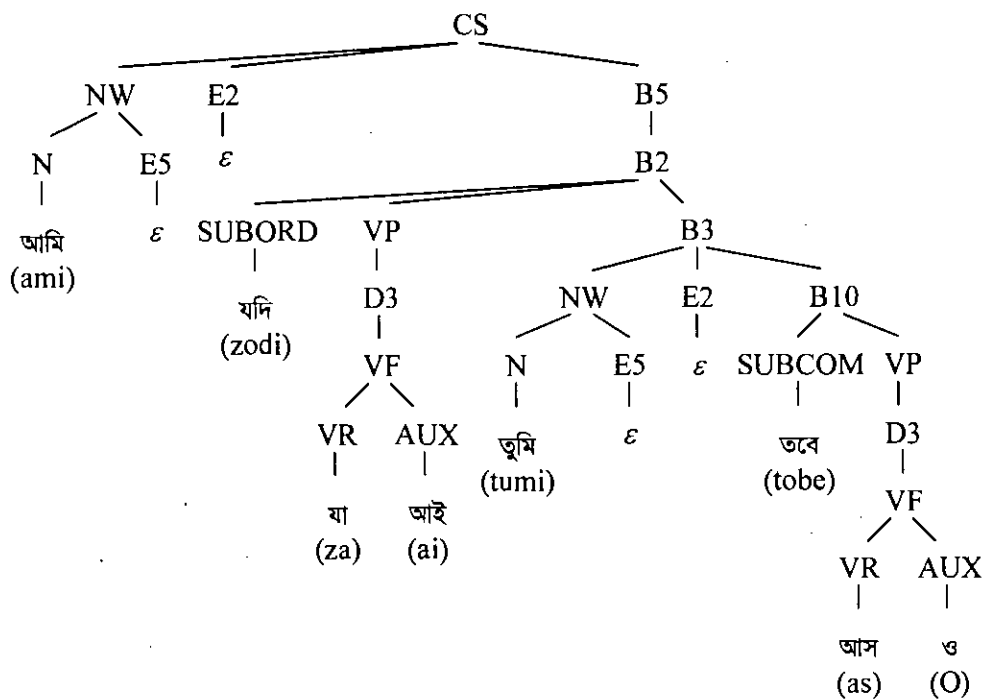


Figure 5.5: Tree derivation for the sentence “আমি যদি যাই তুমি তবে এসো” (ami zodi zai tumi tobe esO) using non-ambiguous grammar with predictive parsing.

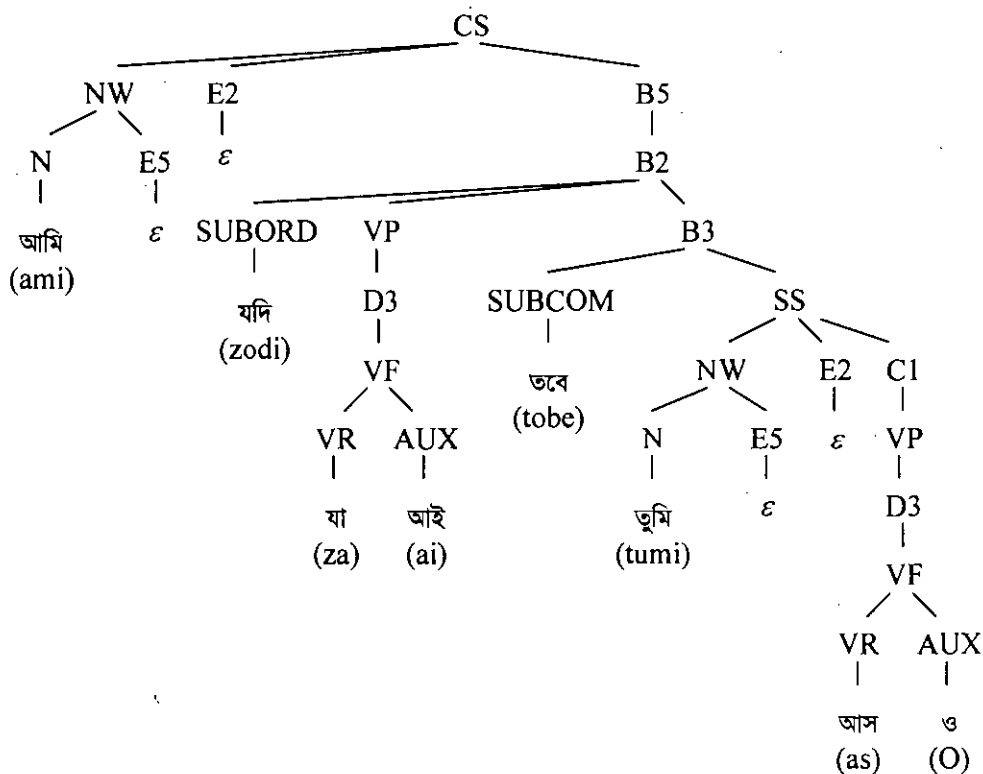


Figure 5.6: Tree derivation for the sentence “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO) using non-ambiguous grammar with predictive parsing.

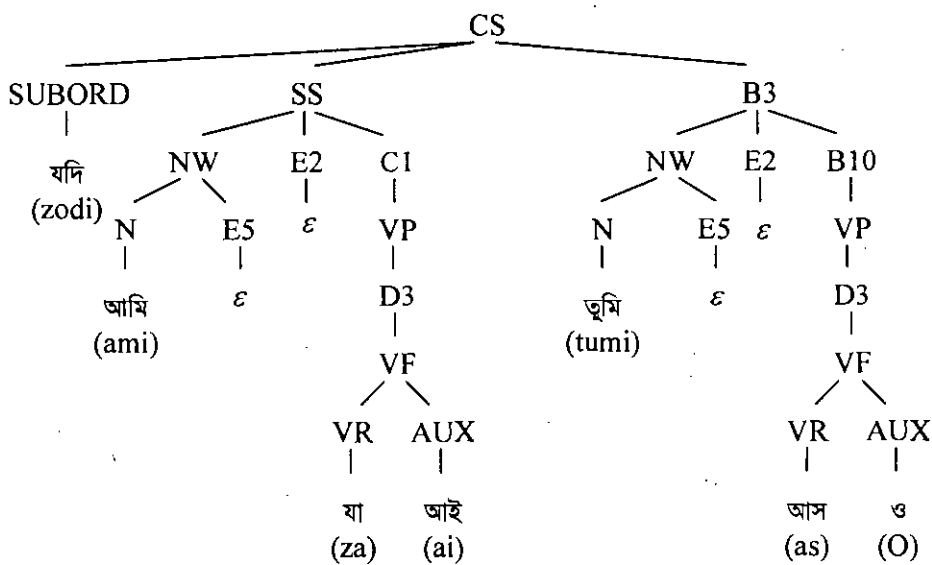


Figure 5.7: Tree derivation for the sentence “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO) using non-ambiguous grammar with predictive parsing.

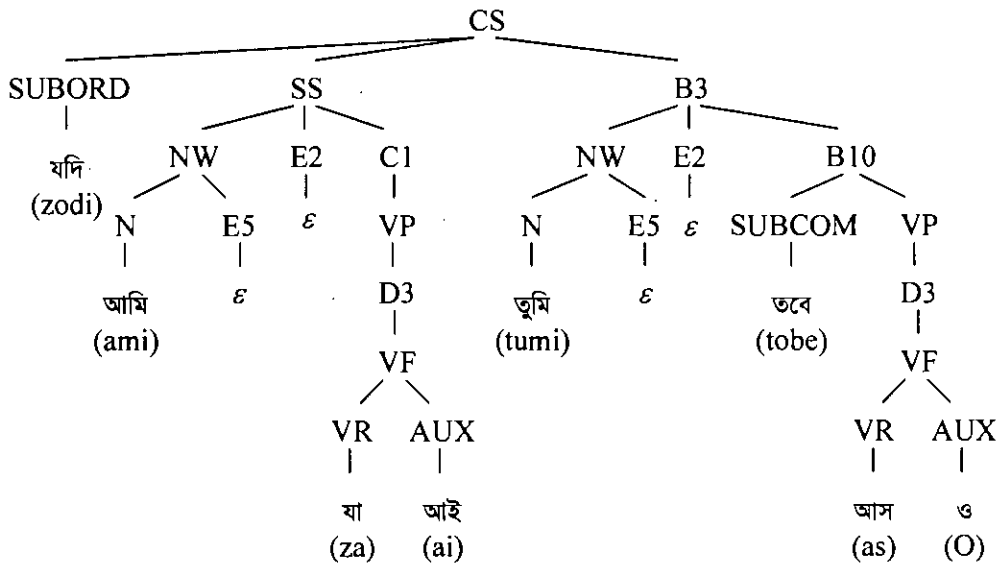


Figure 5.8: Tree derivation for the sentence “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO) using non-ambiguous grammar with predictive parsing.

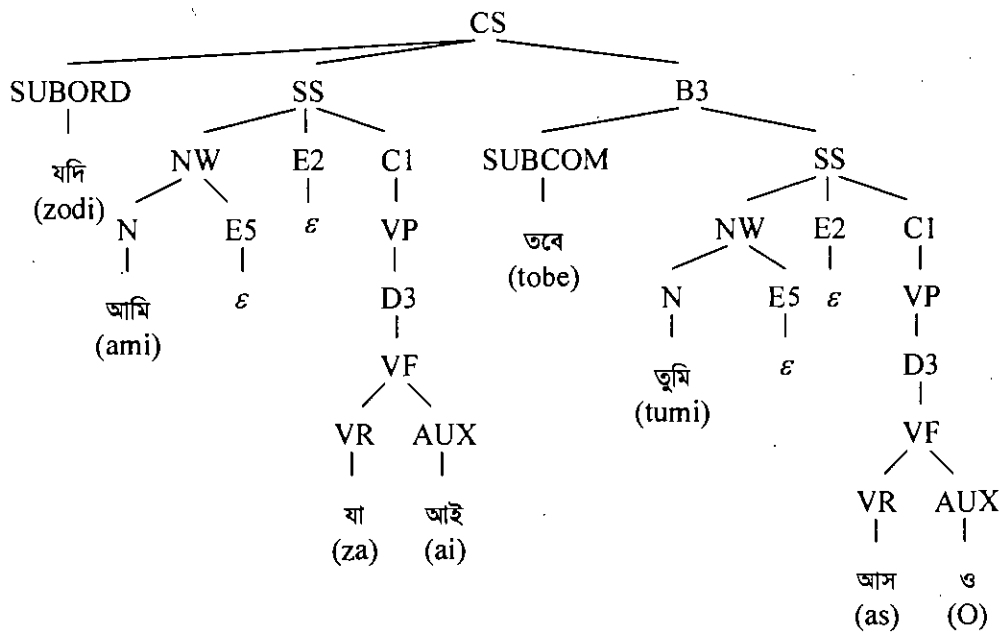


Figure 5.9: Tree derivation for the sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO) using non-ambiguous grammar with predictive parsing.

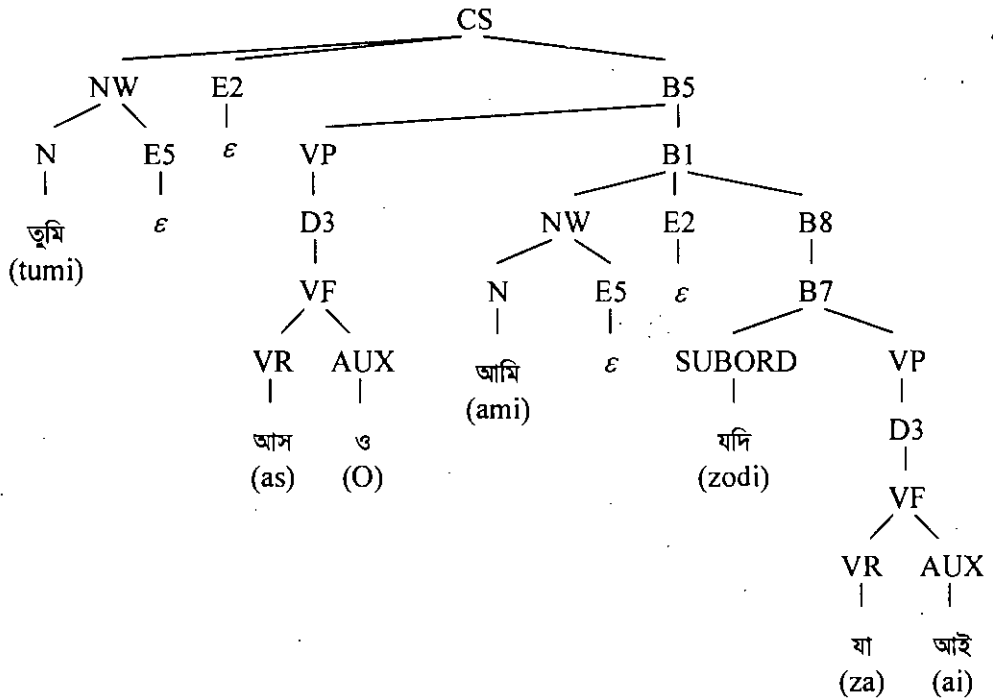


Figure 5.10: Tree derivation for the sentence “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai) using non-ambiguous grammar with predictive parsing.

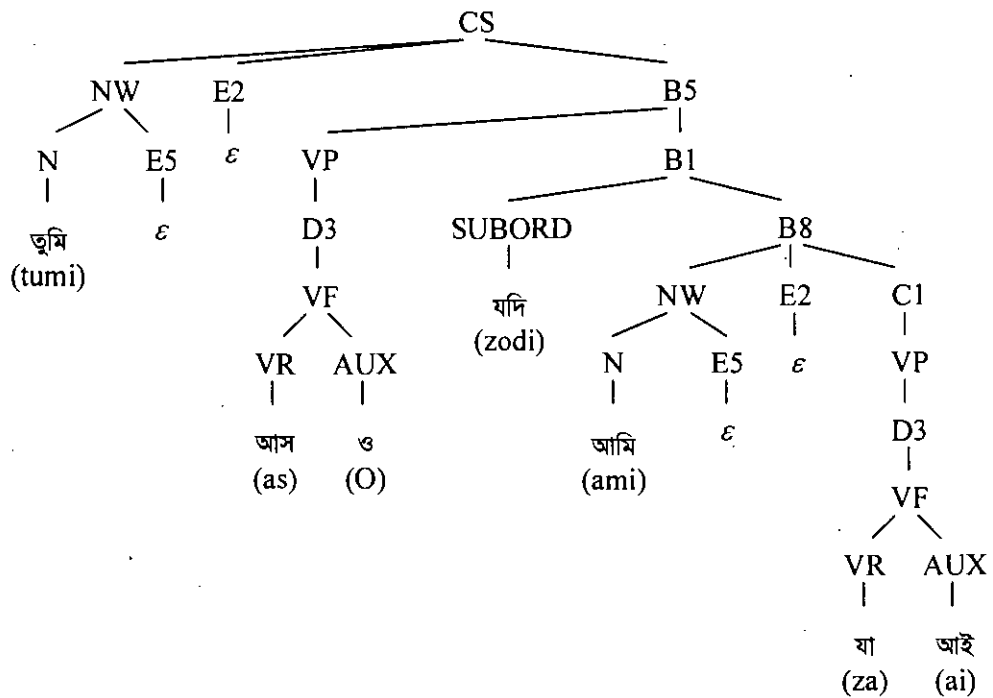


Figure 5.11: Tree derivation for the sentence “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai) using non-ambiguous grammar with predictive parsing.

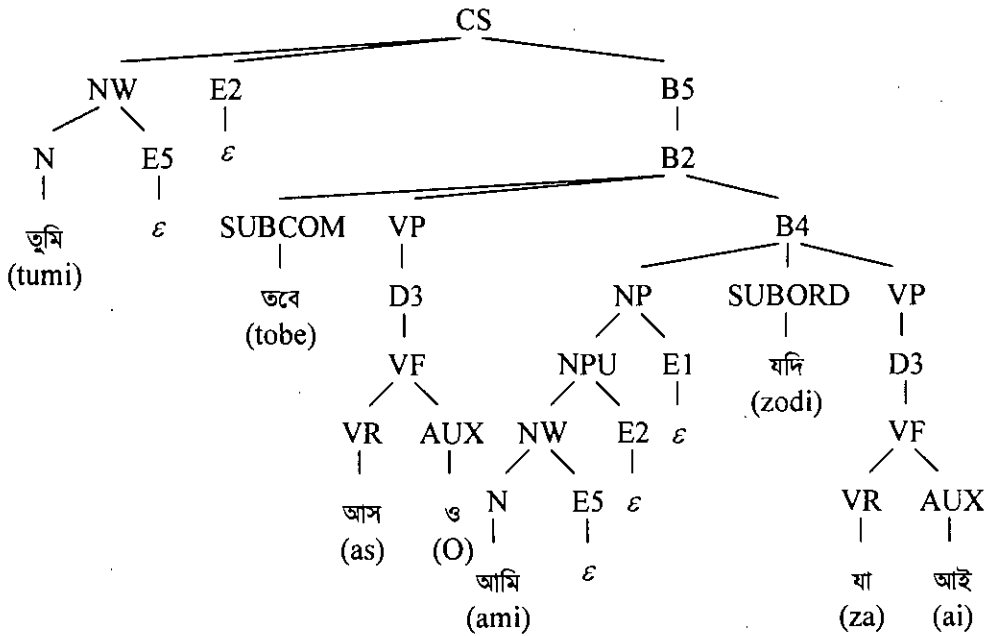


Figure 5.12: Tree derivation for the sentence “তুমি তবে এসো আমি যদি যাই” (tumi tobe esO ami zodi zai) using non-ambiguous grammar with predictive parsing.

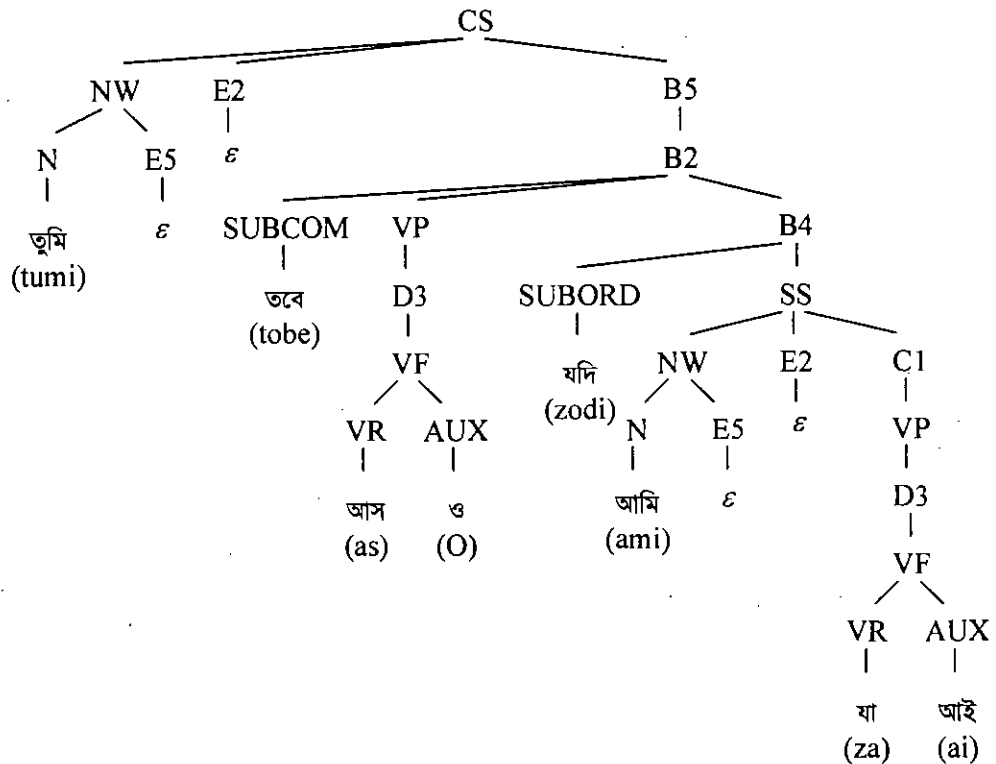


Figure 5.13: Tree derivation for the sentence “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai) using non-ambiguous grammar with predictive parsing.

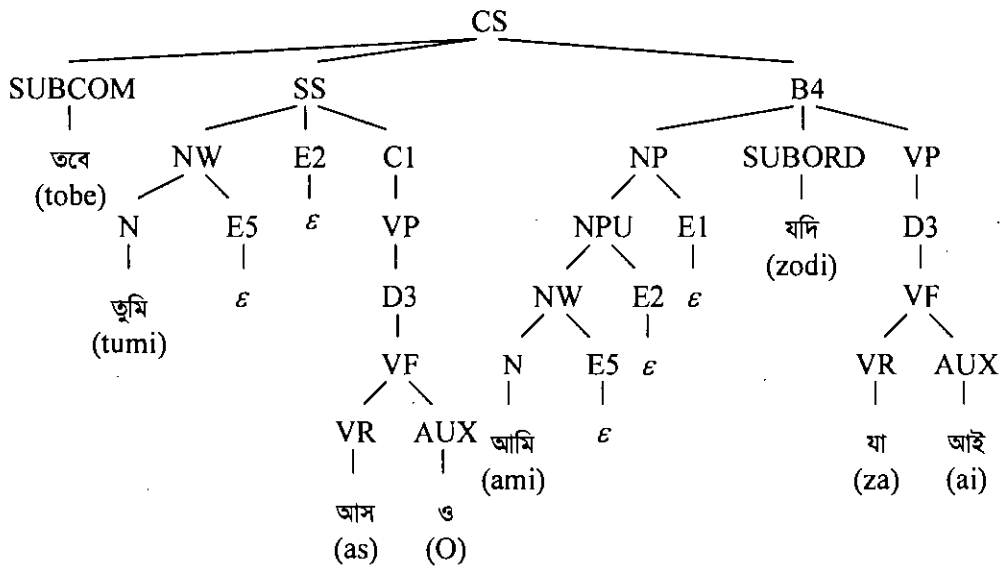


Figure 5.14: Tree derivation for the sentence “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai) using non-ambiguous grammar with predictive parsing.

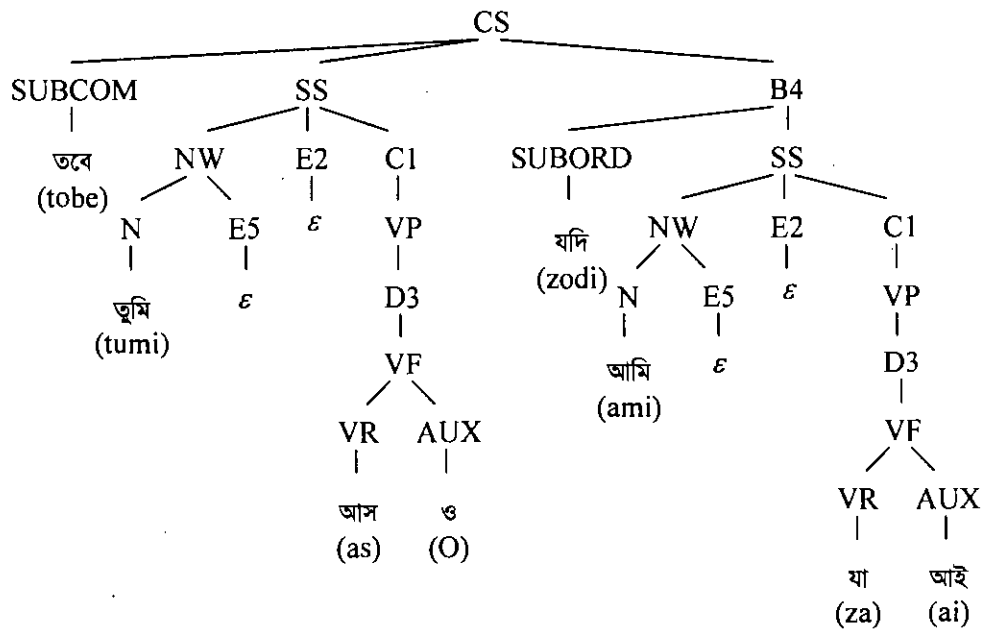


Figure 5.15: Tree derivation for the sentence “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai) using non-ambiguous grammar with predictive parsing.

5.4 Remarks

In this chapter, we have designed non-ambiguous predictive Bangla grammar for complex sentences. We have analyzed all types of complex sentences and designed grammar accordingly. Later this grammar will be merged as a part of comprehensive Bangla grammar.

Non-ambiguous Grammar for Compound Sentence

6.1 Existing Grammar for Compound Sentence

Recalling from chapter 3 (section 3.8), context-free grammar for compound sentence in Bangla natural language is presented as,

$$\text{COMS} \rightarrow \text{SS Conj SS}$$

$$\text{COMS} \rightarrow \text{SS Conj CS}$$

$$\text{COMS} \rightarrow \text{CS Conj SS}$$

$$\text{COMS} \rightarrow \text{CS Conj CS}$$

Like other types of sentences (simple sentence and complex sentence), ambiguity and non-predictivity problem is present in existing grammar for compound sentence. A compound sentence is composed of two simple sentence or complex sentence. Reason of ambiguity problem in compound sentence is, ambiguity problem in grammar for simple sentence and complex sentence. In previous two chapters, ambiguity problem of simple sentence (in chapter 4) and complex sentence (in chapter 5) is solved. Therefore, ambiguity problem will not occur in compound sentence, if new proposed grammar for simple sentence and complex sentence is used.

But still non-predictivity problem resides in grammar for compound sentence. For example, if we want to parse the compound sentence “আমি ঢাকা যাব এবং তুমি সিলেট যাবে” (ami Dhaka zabo ebong tumi sileT zabe), when first word “আমি” (ami) is considered, it is not possible to determine definitely, which rule is applicable. All four rules, $\text{COMS} \rightarrow \text{SS Conj SS}$, $\text{COMS} \rightarrow \text{SS Conj CS}$, $\text{COMS} \rightarrow \text{CS Conj SS}$, and $\text{COMS} \rightarrow \text{CS Conj CS}$ allows “আমি” (ami) as first word of a compound sentence. Similarly, such kind of non-predictivity to choose a rule occurs for rest of the words in the compound sentence.

There exists another problem in existing grammar for compound sentence. Existing grammar supports compound sentence having only two simple or complex sentences. It does not support compound sentence having more than two simple or complex sentences. More elaboration of such kind of problem and solution technique is discussed in next section (6.2).

6.2 Enhancement of Grammar for Compound Sentence

If we observe the rules of existing grammar for compound sentence, it is quite obvious that one compound sentence is composed of two simple sentence or complex sentence, connected by a conjunctive. Therefore, existing grammar supports only two simple or complex sentences in a compound sentence. For example, it is possible to parse the sentence “আমি ঢাকা যাব এবং তুমি সিলেট যাবে” (ami Dhaka zabo ebong tumi sileT zabe), because this sentence is composed of two simple sentences “আমি ঢাকা যাব” (ami Dhaka zabo) and “তুমি সিলেট যাবে” (tumi sileT zabe), connected by the conjunctive “এবং” (ebong). Tree derivation of the sentence is demonstrated in figure 6.1.

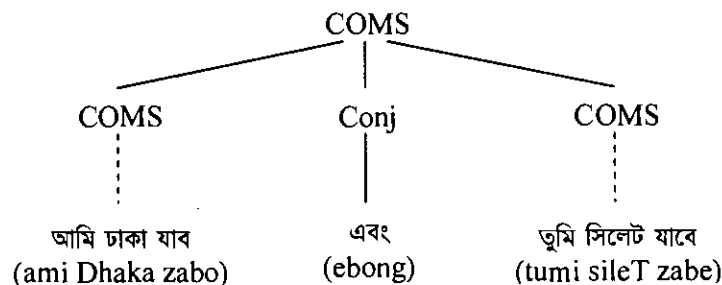


Figure 6.1: Tree derivation for the compound sentence “আমি ঢাকা যাব এবং তুমি সিলেট যাবে” (ami Dhaka zabo ebong tumi sileT zabe) using existing grammar for compound sentence.

But it is not possible to parse a compound sentence having more than two simple or complex sentences connected by conjunctives, using existing grammar for compound sentence. For example, we can not parse the compound sentence “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahI zabe).

We can facilitate use of more than two single or complex sentences connected by conjunctives by using the following grammar,

$$\begin{aligned} \text{COMS} &\rightarrow \text{S Conj S} \\ \text{S} &\rightarrow \text{SS} \mid \text{CS} \mid \text{COMS} \end{aligned}$$

where, S denotes sentence.

Using the new grammar, it is possible to use more than two single or complex sentences connected by conjunctives. For example, we can now parse the compound sentence “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahl zabe). Tree derivation for the sentence is demonstrated in figure 6.2.

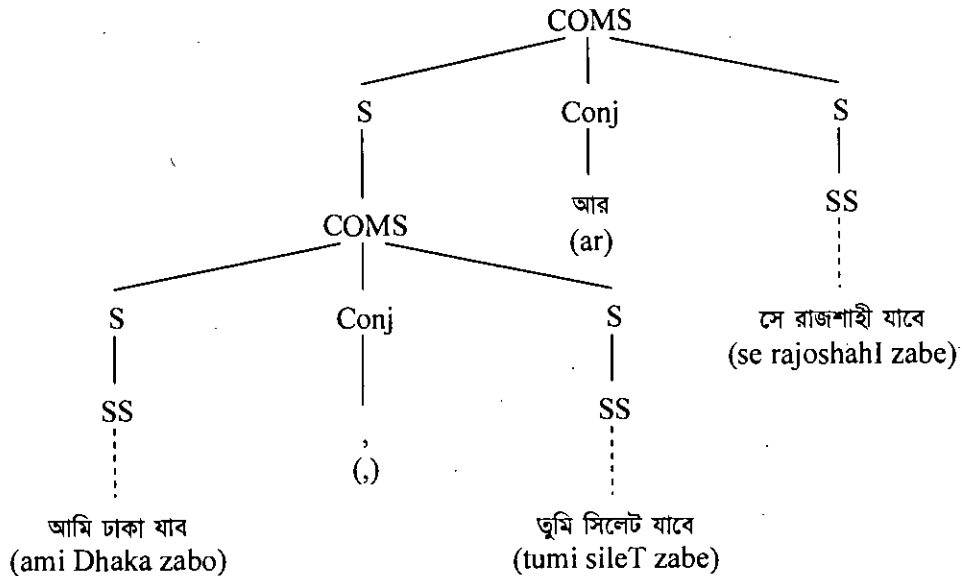
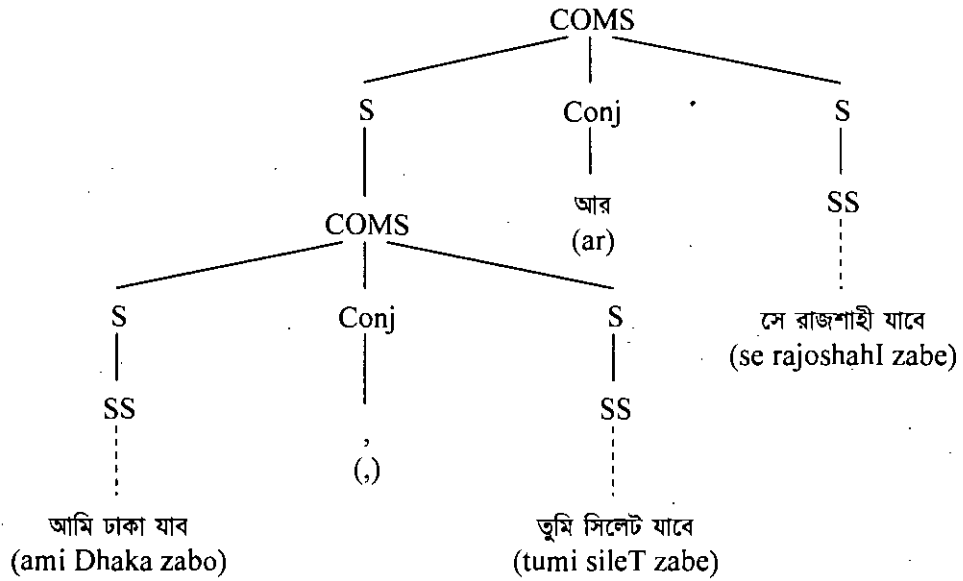
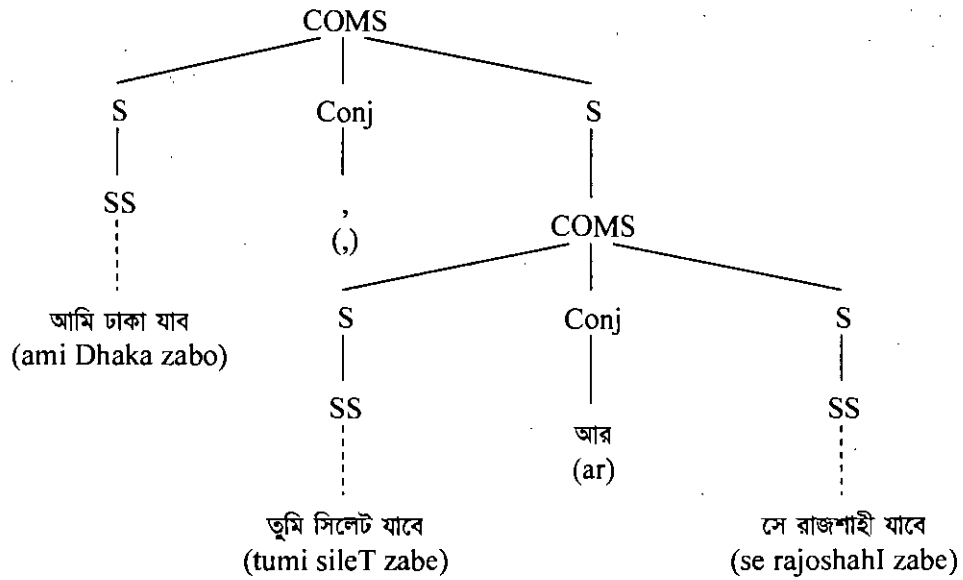


Figure 6.2: Tree derivation for the compound sentence “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahl zabe) using new grammar for compound sentence.

But the grammar is ambiguous, because of new arrival of the rule “COMS \rightarrow COMS Conj, COMS”. When more than two simple or complex sentences exist in a compound sentence, then multiple parsing trees are possible to be drawn. For example, there are two possible parsing trees for the compound sentence “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahl zabe), as demonstrated in figure 6.3.



(a)



(b)

Figure 6.3: Ambiguity in the new rule $COMS \rightarrow S \text{ Conj } S$.

Ambiguity occurs in the rule “ $COMS \rightarrow COMS \text{ Conj } COMS$ ”, because both side non-terminals of “Conj” are extensive. We can make the rule non-ambiguous, by making one non-terminal as non-extensive. We can make the left non-terminal non-extensive, by limiting it to be a simple sentence or complex sentence, but compound sentence is not allowed. Now we can define a non-terminal called “Basic Sentence” (BS in short), which refers a simple or complex sentence.

Now we can re-write the rule in non-ambiguous form as follows,

$$\text{COMS} \rightarrow \text{BS Conj COMS} \mid \text{BS Conj BS}$$

$$\text{BS} \rightarrow \text{SS} \mid \text{CS}$$

Using the new rule, only one parsing tree can be drawn for any compound sentence, for example “আমি ঢাকা যাব, তুমি সিলেট যাবে আর সে রাজশাহী যাবে” (ami Dhaka zabo, tumi sileT zabe ar se rajoshahI zabe), as demonstrated in figure 6.4.

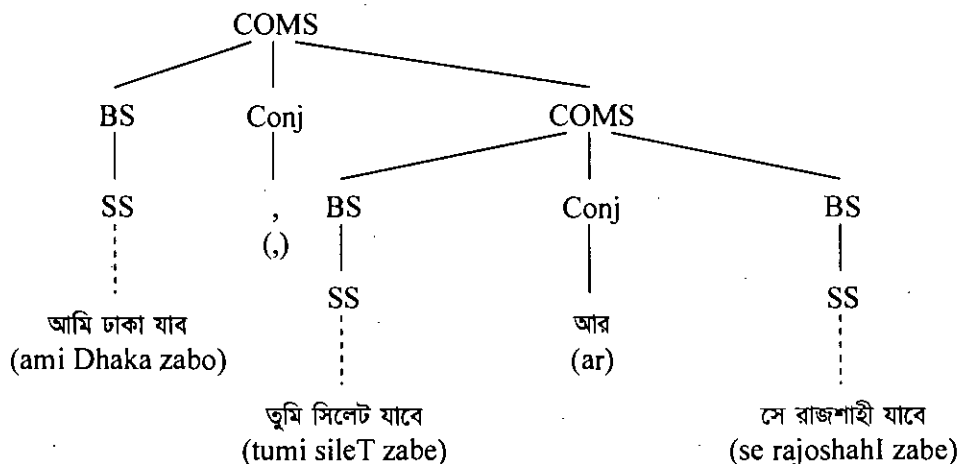


Figure 6.4: Tree derivation using non-ambiguous grammar for compound sentence.

6.3 Non-ambiguous Grammar for Compound Sentence

We can transform the rules of grammar for compound sentence to be usable for predictive parsing, by using the theory of left factoring. In the next chapter (chapter 7), we will see, a simple modification in the rule for COMS will work for the rule for a sentence, which also covers a compound sentence. So, we are not going to give effort of left factoring of rules for compound sentence in this chapter. Grammar for all kind of sentence with predictive parsing will be achieved in the next chapter (chapter 7).

6.4 Remarks

In this chapter, we have designed non-ambiguous Bangla grammar for compound sentences. Elimination of non-predictive nature is remained for the next chapter. We

have facilitated use of more than two basic sentences in a compound sentence. According to existing Bangla grammar, it is possible to use only two basic sentences in a compound sentence.

Non-ambiguous Comprehensive Grammar

7.1 Existing Grammar for Bangla Sentence

Recalling from chapter 3 (section 3.9), context-free grammar for Bangla sentence is presented as,

$$S \rightarrow SS / CS / COMS$$

where, S indicates sentence, SS indicates simple sentence, CS indicates complex sentence and COMS indicates compound sentence.

Problem using the grammar for Bangla sentence, this grammar is ambiguous and does not support predictive parsing. Ambiguity problem exists because of existence of ambiguity in the grammar for simple sentence, complex sentence and compound sentence. In the previous three chapters (chapter 4, 5, 6), ambiguity problem in grammar for simple sentence, complex sentence and compound sentence is solved.

Though ambiguity problem is solved, still grammar for Bangla sentence does not support predictive parsing. For example, the Bangla sentence “আমি ঢাকা যাব এবং তুমি সিলেট যাবে” (ami Dhaka zabo ebong tumi sileT zabe) is certainly a compound sentence. But when a parser looks up the first word “আমি” (ami), applying this grammar, the parser can not determine whether the sentence is simple or complex or compound. A backtracking algorithm requires for parsing the sentence using the existing grammar, which takes exponential runtime. In the next section (section 7.2), we will solve non-predictive nature of Bangla sentence.

7.2 Non-ambiguous Grammar for Bangla Sentence

In the previous chapter (chapter 6), we halted the task of transforming the grammar for compound sentence to be usable for predictive parser. In this section, we will

transform the grammar for Bangla sentence into a grammar with predictive parsing. Then predictive parsing of compound sentences will also be possible.

Existing Bangla grammar is written as,

$$S \rightarrow SS \mid CS \mid COMS$$

Again proposed grammar for compound sentence in previous chapter (chapter 6) is achieved as,

$$COMS \rightarrow BS \text{ Conj } COMS \mid BS \text{ Conj } BS$$

$$BS \rightarrow SS \mid CS$$

We can merge two grammars as follows,

$$S \rightarrow BS \mid BS \text{ Conj } S$$

$$BS \rightarrow SS \mid CS$$

Therefore, in the rule “ $S \rightarrow BS$ ”, all types of simple and complex sentence is achieved. Again in the rule “ $S \rightarrow BS \text{ Conj } S$ ”, all types of compound sentence is achieved.

Now, rule for S does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$S \rightarrow BS \mid BS \text{ Conj } S$$

\Rightarrow

$$S \rightarrow BS A1$$

(left factoring)

$$\text{where, } A1 \rightarrow \text{Conj } S \mid \varepsilon$$

Rule for BS does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$BS \rightarrow SS \mid CS$$

\Rightarrow

$$\begin{aligned} BS &\rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2 \mid NW E2 B5 \mid \\ &PRE NW E2 B5 \mid UNG B6 \mid SUBORD SS B3 \mid SUBCOM SS B4 \end{aligned}$$

(expanding rules for SS and CS)

 \Rightarrow

$$\begin{aligned} BS &\rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3 \mid SUBORD SS B3 \mid \\ &SUBCOM SS B4 \end{aligned}$$

(left factoring)

where,

 $A2 \rightarrow C1 \mid B5$ $A3 \rightarrow C2 \mid B6$

Rule for A1 supports predictive parsing. But rule for A2 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

 $A2 \rightarrow C1 \mid B5$ \Rightarrow $A2 \rightarrow VP \mid Conj SS \mid VP B1 \mid Conj B12 \mid B2$

(expanding rules for C1 and B5)

 \Rightarrow $A2 \rightarrow VP A4 \mid Conj A5 \mid B2$

(left factoring)

where,

 $A4 \rightarrow B1 \mid \epsilon$ $A5 \rightarrow SS \mid B12$

Rule for A3 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

 $A3 \rightarrow C2 \mid B6$ \Rightarrow

$$\begin{aligned} A3 &\rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3 \mid Conj SS B1 \mid BivE B12 \mid Biv B5 \mid \\ &D3 B1 \mid B2 \end{aligned}$$

(expanding rules for C2 and B6)

 \Rightarrow $A3 \rightarrow Conj SS A4 \mid BivE A5 \mid Biv A2 \mid D3 A4 \mid B2$

(left factoring)

Rule for A4 supports predictive parsing. But rule for A5 does not support predictive parsing. We can transform it to be usable for predictive parsing by following derivations,

$$A5 \rightarrow SS \mid B12$$

\Rightarrow

$$A5 \rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2 \mid NW E2 B5 \mid$$

$$PRE NW E2 B5 \mid UNG B6 \quad \text{(expanding rules for SS and B12)}$$

\Rightarrow

$$A5 \rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3$$

Finally, non-ambiguous grammar for Bangla sentence with predictive parsing is presented as,

$$S \rightarrow BS A1$$

$$BS \rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3 \mid SUBORD SS B3 \mid$$

$$SUBCOM SS B4$$

$$A1 \rightarrow Conj S \mid \varepsilon$$

$$A2 \rightarrow VP A4 \mid Conj A5 \mid B2$$

$$A3 \rightarrow Conj SS A4 \mid BivE A5 \mid Biv A2 \mid D3 A4 \mid B2$$

$$A4 \rightarrow B1 \mid \varepsilon$$

$$A5 \rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3$$

7.3 Comprehensive Bangla Grammar

In chapter 4, we have designed non-ambiguous grammar for simple sentence with predictive parsing. Then, in chapter 5, we have designed non-ambiguous grammar for complex sentence with predictive parsing and in chapter 6, we have designed non-ambiguous grammar for compound sentence. Finally, in this chapter, in previous section (section 7.2), we have designed non-ambiguous grammar for Bangla sentence with predictive parsing, which covers all three types of Bangla sentence.

We can merge all the rules of grammar for simple sentence, complex sentence, compound sentence and all sentence. Therefore, comprehensive Bangla grammar is found as,

$$S \rightarrow BS A1$$

$$BS \rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3 \mid SUBORD SS B3 \mid$$

$$SUBCOM SS B4$$

$$A1 \rightarrow Conj S \mid \varepsilon$$

$$A2 \rightarrow VP A4 \mid Conj A5 \mid B2$$

$$A3 \rightarrow Conj SS A4 \mid BivE A5 \mid Biv A2 \mid D3 A4 \mid B2$$

$$A4 \rightarrow B1 \mid \varepsilon$$

$$A5 \rightarrow NW E2 A2 \mid PRE NW E2 A2 \mid UNG A3$$

$$CS \rightarrow NW E2 B5 \mid PRE NW E2 B5 \mid UNG B6 \mid SUBORD SS B3 \mid$$

$$SUBCOM SS B4$$

$$B1 \rightarrow NW E2 B8 \mid PRE NW E2 B8 \mid UNG B9 \mid SUBORD SS \mid$$

$$SUBCOM SS$$

$$B2 \rightarrow SUBORD VP B3 \mid SUBCOM VP B4$$

$$B3 \rightarrow NW E2 B10 \mid PRE NW E2 B10 \mid UNG B11 \mid SUBCOM SS$$

$$B4 \rightarrow NP SUBORD VP \mid SUBORD SS$$

$$B5 \rightarrow VP B1 \mid Conj B12 \mid B2$$

$$B6 \rightarrow Conj SS B1 \mid BivE B12 \mid Biv B5 \mid D3 B1 \mid B2$$

$$B7 \rightarrow SUBORD VP \mid SUBCOM VP$$

$$B8 \rightarrow VP \mid Conj B13 \mid B7$$

$$B9 \rightarrow Conj B13 \mid BivE B13 \mid Biv B8 \mid D3 \mid B7$$

$$B10 \rightarrow VP \mid Conj B14 \mid SUBCOM VP$$

$$B11 \rightarrow Conj B14 \mid BivE B14 \mid Biv B10 \mid D3 \mid SUBCOM VP$$

$$B12 \rightarrow NW E2 B5 \mid PRE NW E2 B5 \mid UNG B6$$

$$B13 \rightarrow NW E2 B8 \mid PRE NW E2 B8 \mid UNG B9$$

$$B14 \rightarrow NW E2 B10 \mid PRE NW E2 B10 \mid UNG B11$$

$$SS \rightarrow NW E2 C1 \mid PRE NW E2 C1 \mid UNG C2$$

$$C1 \rightarrow VP \mid Conj SS$$

$$C2 \rightarrow Conj SS \mid BivE SS \mid Biv C1 \mid D3$$

$$VP \rightarrow D3 \mid UNG E2 E1 D1$$

- D1 → VF | AP VF
- D2 → VF | D4
- D3 → VF | AP D2 | D4 | DEMO E3 D4 | SPR E4 D4
- D4 → NW E2 E1 D1
- VF → VR AUX

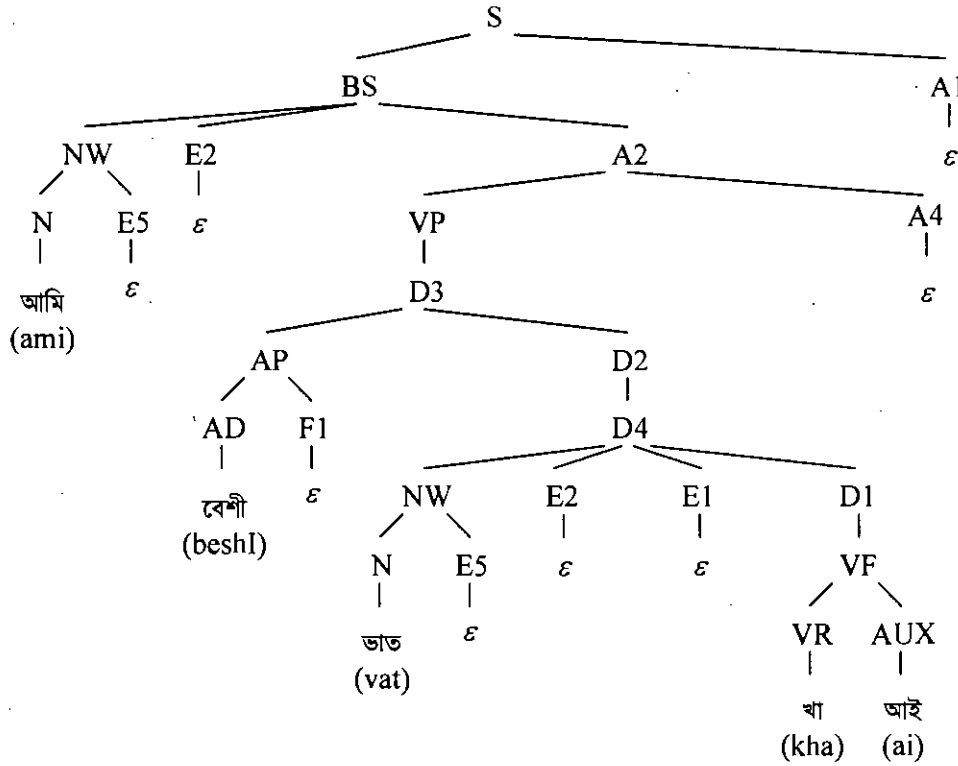


Figure 7.1: Tree derivation for the simple sentence “আমি বেশী ভাত খাই” (ami beshi vat khai).

- NP → NPU E1
- E1 → Conj NP | ε
- NPU → NW E2 | PRE NW E2 | UNG E2
- E2 → BivE NP | Biv | ε
- PRE → DEMO E3 | SPR E4 | AP
- E3 → SPR E4 | AP | ε
- E4 → AP | ε
- NW → N E5
- E5 → DET | PM | ε

- SPR \rightarrow QFR E6
- E6 \rightarrow PP | ϵ
- DEMO \rightarrow DD E7 | DO
- E7 \rightarrow DO | ϵ
- UNG \rightarrow UN E8
- E8 \rightarrow UNG | ϵ
- AP \rightarrow AD F1
- F1 \rightarrow AP | ϵ

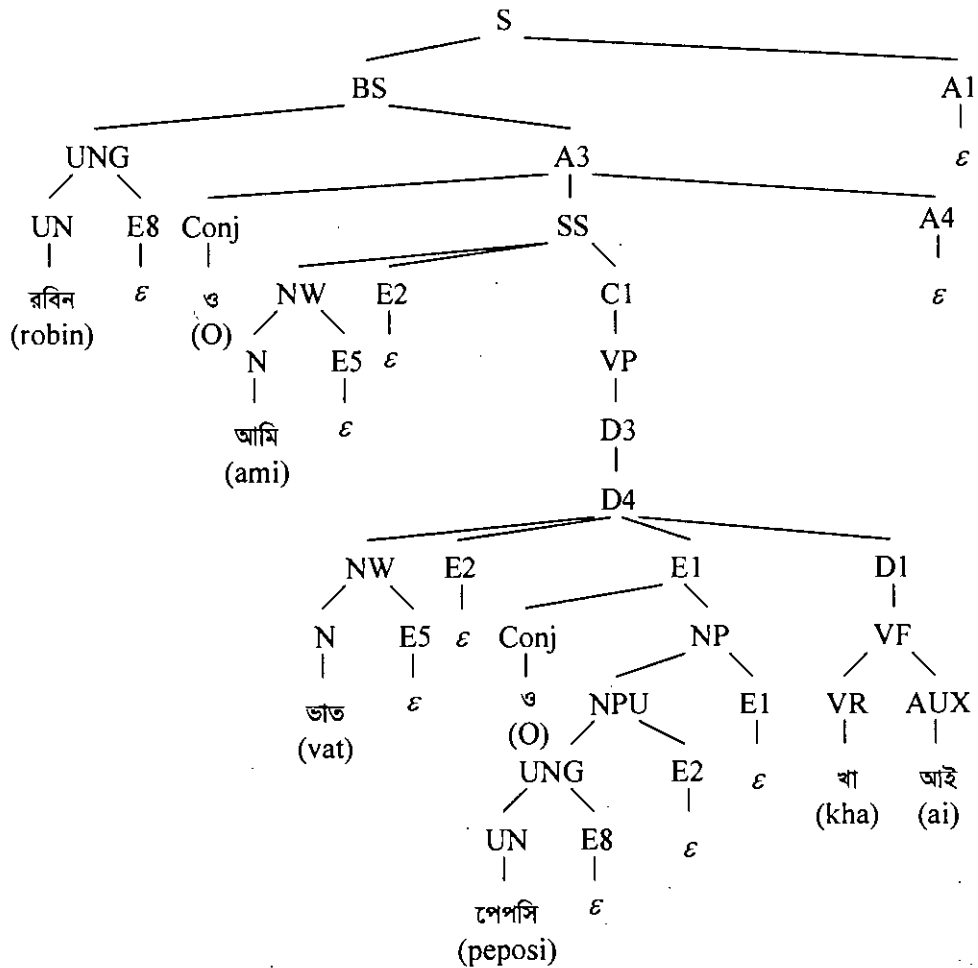


Figure 7.2: Tree derivation for the simple sentence “রবিন ও আমি ভাত ও পেপসি খাই” (robin O ami vat O peposi khai).

Now we can give some examples of different types of Bangla sentences and also watch their tree derivations. Firstly, we can give some examples of simple sentences

like “আমি বেশী ভাত খাই” (ami beshI vat khai), “রবিন ও আমি ভাত ও পেপসি খাই” (robin O ami vat O peposi khai), “ঐ ছেলেটি খেলে” (OI chheleTi khele). Tree derivations of these simple sentences are demonstrated in figure 7.1, 7.2 and 7.3.

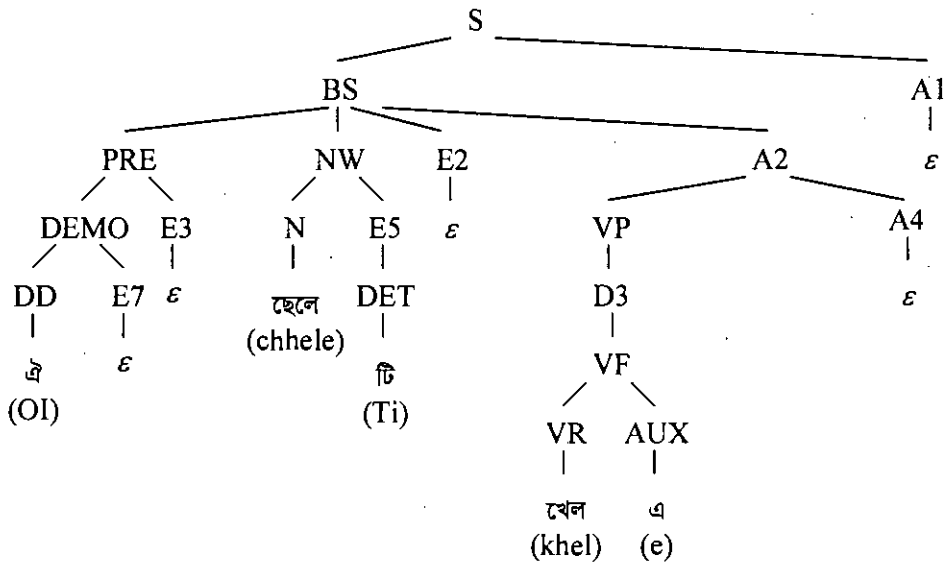


Figure 7.3: Tree derivation for the simple sentence “ঐ ছেলেটি খেলে” (OI chheleTi khele).

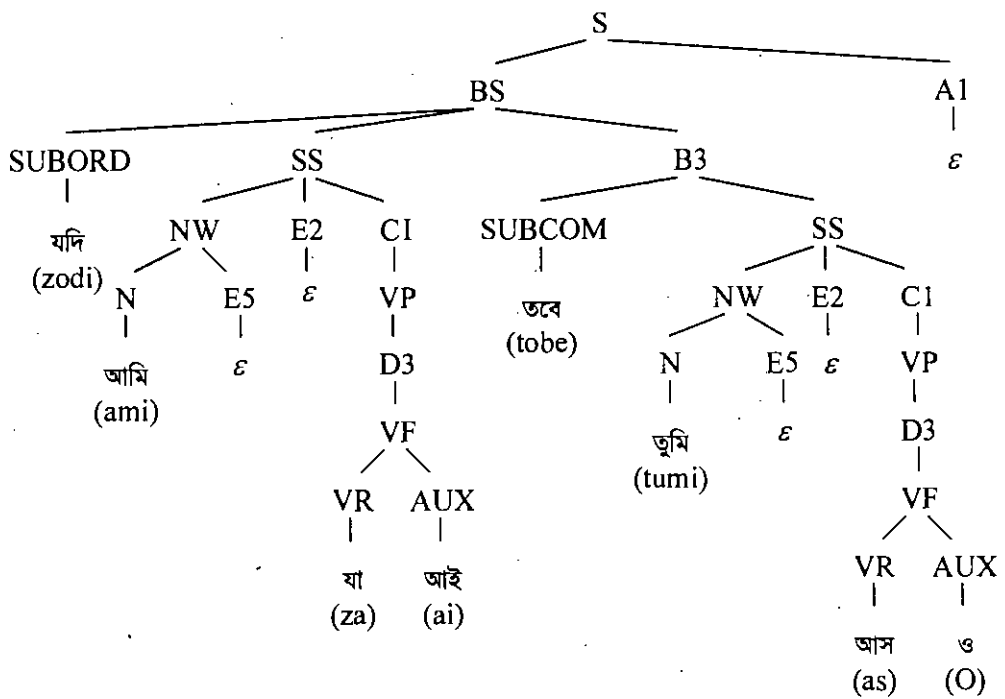


Figure 7.4: Tree derivation for the complex sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO).

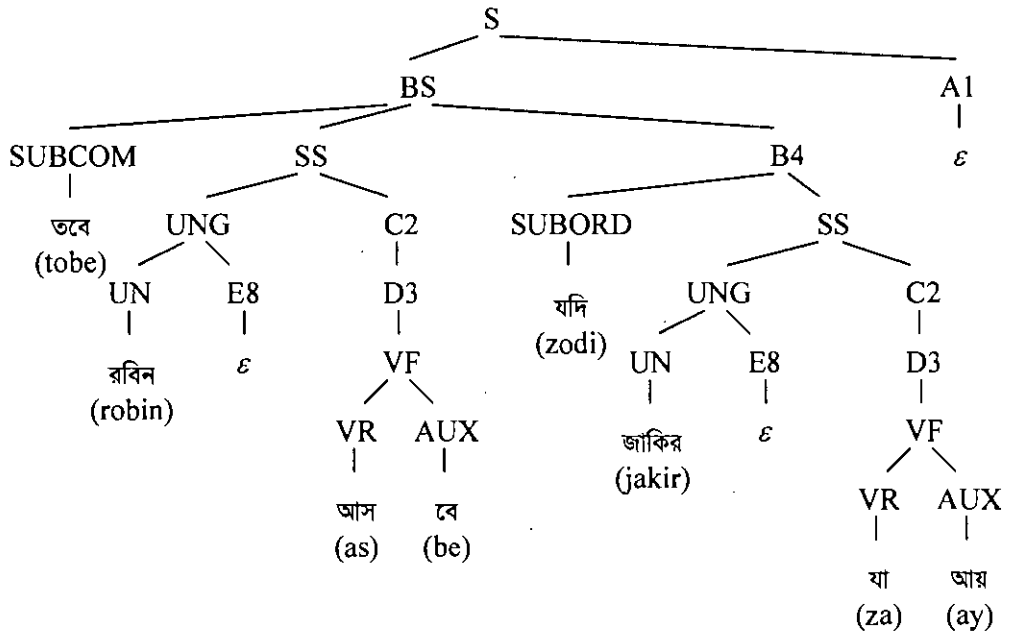


Figure 7.5: Tree derivation for the complex sentence “তবে রবিন আসবে যদি জাকির যায়” (tobe robin ashobe zodi jakir zay).

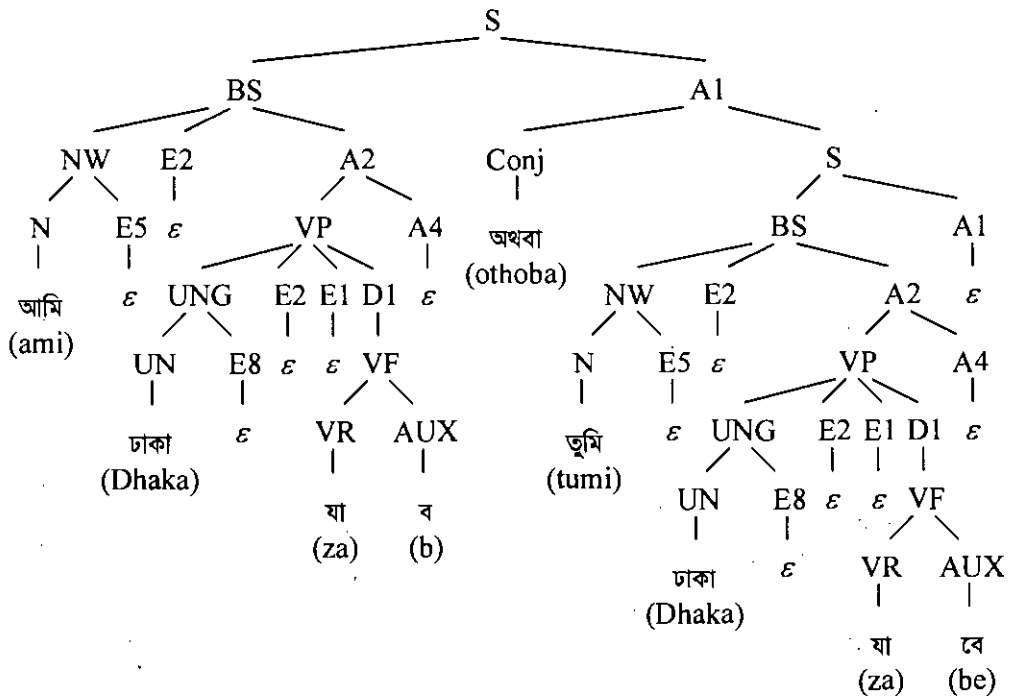


Figure 7.6: Tree derivation for the compound sentence “আমি ঢাকা যাব অথবা তুমি ঢাকা যাবে” (ami Dhaka zabo othoba tumi Dhaka zabe).

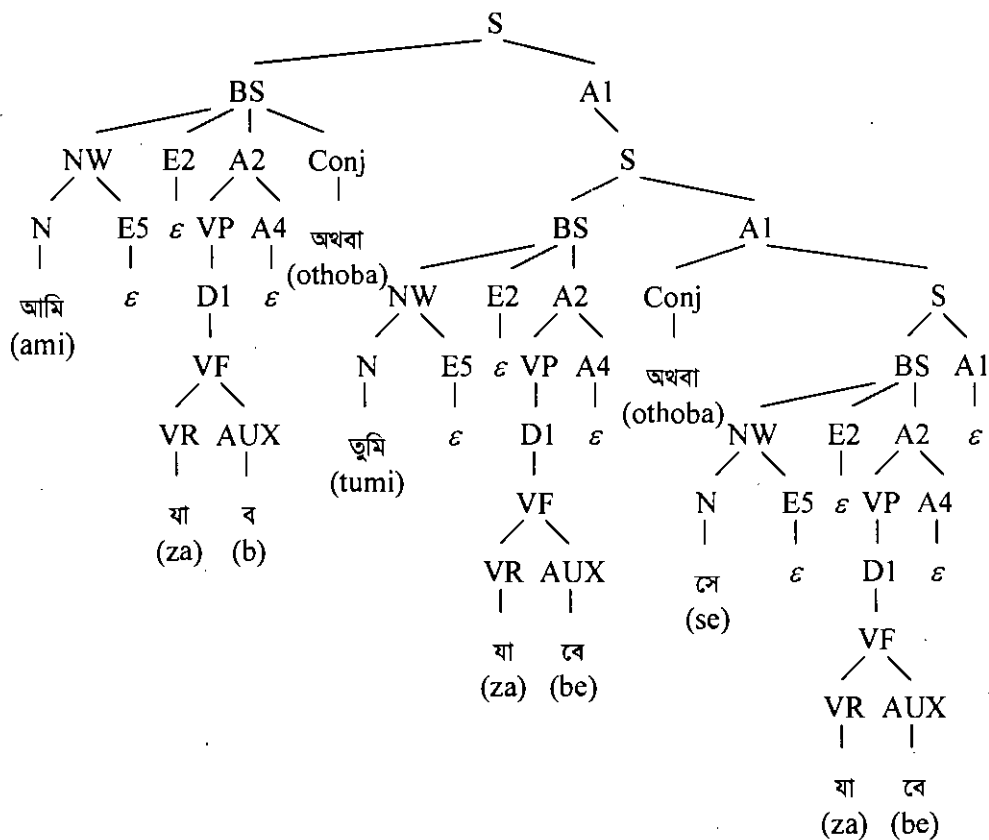


Figure 7.7: Tree derivation for the compound sentence “আমি যাব অথবা তুমি যাবে অথবা সে যাবে” (ami zabo othoba tumi zabe othoba se zabe).

Now, we can give some examples of complex sentences like “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), “তবে রবিন আসবে যদি জাকির যায়” (tobe robin asobe zodi robin zay). Tree derivations of these complex sentences are demonstrated in figure 7.4 and 7.5.

Finally, we can give some examples of compound sentences like, “আমি ঢাকা যাব আর তুমি সিলেট যাবে” (ami Dhaka zabo ar tumi sileT zabe), “আমি যাব অথবা তুমি যাবে অথবা সে যাবে” (ami zabo othoba tumi zabe othoba se zabe). Tree derivations of these compound sentences are demonstrated in figure 7.6 and 7.7.

7.4 Significance of Each Non-terminal of Proposed Bangla Grammar

From the previous section (section 7.3), it is found, there are 48 non-terminals in comprehensive Bangla grammar. Each non-terminal has its own significance. In other way, we can say, each non-terminal represents a word or a group of word of particular category. In this section, we will discuss significance of each non-terminal in comprehensive Bangla grammar.

We can divide the non-terminals into six categories. They are, non-terminals of sentence level (which also represents compound sentence level), complex sentence level, simple sentence level, verb phrase level, noun phrase level, adjective phrase level.

7.4.1 Significance of Non-terminals of Sentence Level

We have summarized the significance of non-terminals of sentence level as follows,

S: The first non-terminal “S” represents a sentence of Bangla language. The sentence may be simple or complex or compound. For example, the simple sentence “আমি ঢাকা যাই” (ami Dhaka zai), the complex sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), the compound sentence “আমি ঢাকা যাব অথবা তুমি ঢাকা যাবে” (ami Dhaka zabo othoba tumi Dhaka zabe), all three types of sentences are represented by “S”.

BS: The non-terminal “BS” indicates “basic sentence”. It may be either simple sentence or complex sentence. Therefore, “BS” indicates a simple sentence or a complex sentence or each independent part of a compound sentence, which may be either simple or complex sentence. For example, the simple sentence “আমি ঢাকা যাই” (ami Dhaka zai), the complex sentence “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), the independent

part of the compound sentence “আমি ঢাকা যাব অথবা তুমি ঢাকা যাবে” (ami Dhaka zabo othoba tumi Dhaka zabe), are represented by “BS”.

A1: The non-terminal “A1” is a factor in sentence level. In case of compound sentence having two independent parts, it represents last independent part along with the conjunctive. In case of compound sentence having three independent parts, it represents last one or two independent part along with a conjunctive as starting terminal. For example, the sentence segments “আমি যাব অথবা তুমি যাবে” (ami zabo othoba tumi zabe), “আমি যাব অথবা তুমি যাবে অথবা সে যাবে” (ami zabo othoba tumi zabe othoba se zabe), “আমি যাব অথবা তুমি যাবে অথবা সে যাবে” (ami zabo othoba tumi zabe othoba se zabe), are represented by “A1”. Use of “A1” is optional. In case of simple or complex sentence, “A1” represents null. In case of compound sentence, last used “A1” represents null.

A2: The non-terminal “A2” is a factor in sentence level. It represents words of a basic sentence, after noun phrase not consisting of unknown word as end symbol. The noun phrase mentioned may be actual or inner, but must occur first of the basic sentence. For example, the sentence segments “আমি ও তুমি ঢাকা যাই” (ami o tumi Dhaka zai), “আমি ও তুমি ঢাকা যাই” (ami O tumi Dhaka zai), “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO), “আমি ঢাকা যাব আর তুমি সিলেট যাবে” (ami Dhaka zabo ar tumi sileT zabe), are represented by “A2”.

A3: The non-terminal “A3” is a factor in sentence level. It represents words of a basic sentence, after noun phrase consisting of unknown word as end symbol. The noun phrase mentioned may be actual or inner, but must occur first of the basic sentence. For example, the sentence segments “রবিন ঢাকা যাবে” (robin Dhaka zabe), “রবিন ও আমি ঢাকা যাই” (robin O ami Dhaka zai), “রবিন এর বন্ধু আসবে” (robin er bondhu asobe), “রবিন যদি যায় তবে জাকির আসবে” (robin zodi zay tobe jakir asobe), are represented by “A3”.

Therefore, “A2” and “A3” has similar functionality, but “A2” occurs after noun phrase not ending with unknown word and “A3” occurs after noun phrase ending with unknown word.

A4: The non-terminal “A4” is a factor in sentence level. It represents second clause of a basic sentence. As we know, basic sentence is of two types – simple and complex. In case of complex sentence, “A4” is second clause, either independent or dependent, where first clause does not contain subordinate or subordinate complement. And, in case of simple sentence, “A4” is not applicable i.e. is null. For example, the clauses “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO), “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai) are represented by “A4”.

A5: The non-terminal “A5” is a factor in sentence level. It represents words of a basic sentence, after conjunctive of noun phrase, if the noun phrase does not consist of an unknown word. It also represents words of a basic sentence, after extensive bivokti of noun phrase, if the noun phrase consists of an unknown word. For example, the sentence segments “আমি ও তুমি ঢাকা যাব” (ami O tumi Dhaka zabo), “রবিন এর বন্ধু আসবে” (robin er bondhu asobe) are represented by “A5”.

7.4.2 Significance of Non-terminals of Complex Sentence Level

We have summarized the significance of non-terminals of complex sentence level as follows,

CS: The non-terminal “CS” represents a complex sentence of Bangla language. Recalling from chapter 5, we know, a complex sentence can be written in 15 ways. All the 15 ways can be represented by “CS”. For example, the complex sentences “আমি গেলে তুমি এসো” (ami gele tumi esO), “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO), “তবে তুমি এসো যদি

আমি যাই” (tobe tumi esO zodi ami zai) are represented by “CS”.

- B1:** The non-terminal “B1” is a factor in complex sentence level. It represents second clause of a complex sentence, which may be dependent or independent. First clause does not contain subordinate or subordinate complement. For example, the clauses “আমি গেলে তবে তুমি এসো” (ami gele tobe tumi esO), “তুমি এসো যদি আমি যাই” (tumi esO zodi ami zai) are represented by “B1”.
- B2:** The non-terminal “B2” is a factor in complex sentence level. It represents words after noun phrase of first clause of a complex sentence, which has subordinate or subordinate complement as starting word. For example, the sentence segments “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO), “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai) are represented by “B2”.
- B3:** The non-terminal “B3” is a factor in complex sentence level. It represents second clause of a complex sentence, which is independent. First clause must start with a subordinate. For example, the clauses “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO) are represented by “B3”.
- B4:** The non-terminal “B4” is a factor in complex sentence level. It represents second clause of a complex sentence, which is dependent. First clause must start with a subordinate complement. For example, the clauses “তবে তুমি এসো যদি আমি যাই” (tobe tumi esO zodi ami zai), “তবে তুমি এসো আমি যদি যাই” (tobe tumi esO ami zodi zai) are represented by “B4”.
- B5:** The non-terminal “B5” is a factor in complex sentence level. It represents noun phrase of first clause of a complex sentence, where noun phrase occurs first, it does not start with unknown word and it may be actual or

inner. First clause may be independent or dependent. For example, the sentence segments “আমি যদি যাই তবে তুমি এসো” (ami zodi zai tobe tumi esO), “তুমি তবে এসো যদি আমি যাই” (tumi tobe esO zodi ami zai) are represented by “B5”.

B6: The non-terminal “B6” is a factor in complex sentence level. It represents noun phrase of first clause of a complex sentence, where noun phrase occurs first, it consists of only unknown word and it may be actual or inner. First clause may be dependent or independent. For example, the sentence segments “রবিন যদি যায় তবে জাকির আসবে” (robin zodi zay tobe jakir asobe), “জাকির তবে আসবে যদি রবিন যায়” (jakir tobe asobe zodi robin zay) are represented by “B6”.

B7: The non-terminal “B7” is a factor in complex sentence level. It represents words of a complex sentence after first clause, not containing subordinate or subordinate complement and then, after noun phrase of second clause. “B7” starts with subordinate or subordinate complement. For example, the sentence segments “আমি গেলে তুমি তবে এসো” (ami gele tumi tobe esO), “তুমি এসো আমি যদি যাই” (tumi esO ami zodi zai) are represented by “B7”.

B8: The non-terminal “B8” is a factor in complex sentence level. It represents words of a complex sentence after first clause, not containing subordinate or subordinate complement and then, after noun phrase of second clause, not starting with unknown word. The noun phrase may be actual or inner. The second clause does not start with subordinate or subordinate complement. For example, the sentence segments “আমি গেলে তুমি এসো” (ami gele tumi esO), “তুমি গেলে আমি ও সে আসব” (tumi gele ami O se asobo) are represented by “B8”.

B9: The non-terminal “B9” is a factor in complex sentence level. It represents words of a complex sentence after first clause, not containing subordinate

or subordinate complement and then, after noun phrase of second clause, starting with unknown word. The noun phrase may be actual or inner. The second clause does not start with subordinate or subordinate complement. For example, the sentence segments “আমি গেলে রবিন আসবে” (ami gele robin asobe), “আমি গেলে রবিন ও জাকির আসবে” (ami gele robin O jakir asobe) are represented by “B9”.

- B10:** The non-terminal “B10” is a factor in complex sentence level. It represents words of a complex sentence after first clause, which is dependent and starts with subordinate, then after noun phrase of second clause. The noun phrase may be actual or inner and does not start with an unknown word. For example, the sentence segments “যদি আমি যাই তুমি এসো” (zodi ami zai tumi esO), “যদি আমি যাই তুমি তবে এসো” (zodi ami zai tumi tobe esO) are represented by “B10”.
- B11:** The non-terminal “B11” is a factor in complex sentence level. It represents words of a complex sentence after first clause, which is dependent and starts with subordinate, then after noun phrase of second clause. The noun phrase may be actual or inner and starts with an unknown word. For example, the sentence segments “যদি রবিন যায় জাকির আসবে” (zodi robin zay jakir asobe), “যদি রবিন যায় জাকির তবে আসবে” (zodi robin zay jakir tobe asobe) are represented by “B11”.
- B12:** The non-terminal “B12” is a factor in complex sentence level. If a complex sentence contains a conjunctive in the noun phrase of first clause and the noun phrase does not start with an unknown word, then “B12” represents words after the conjunctive. The first clause may be dependent or independent. For example, the sentence segments “আমি ও তুমি গেলে সে আসবে” (ami O tumi gele se asobe), “আমি ও তুমি যাব যদি সে আসে” (ami O tumi zabo zodi se ase) are represented by “B12”.

B13: The non-terminal “B13” is a factor in complex sentence level. If noun phrase of second clause of a complex sentence contains a conjunctive and does not start with an unknown word and first clause does not contain subordinate or subordinate complement, then “B13” represents words after the conjunctive. The second clause does not start with subordinate or subordinate complement. For example, the sentence segment “তুমি গেলে আমি ও সে আসব” (tumi gele ami O se asobo) is represented by “B13”.

B14: The non-terminal “B14” is a factor in complex sentence level. If first clause of a complex sentence is dependent and starts with subordinate and second clause starts with a noun phrase and the noun phrase contains a conjunctive and the noun phrase does not start with an unknown word, then words after the conjunctive is represented by “B14”. For example, the sentence segment “যদি তুমি আস আমি ও সে আসব” (zodi tumi aso ami O se asobo) is represented by “B14”.

7.4.3 Significance of Non-terminals of Simple Sentence Level

We have summarized the significance of non-terminals of simple sentence level as follows,

SS: The non-terminal “SS” represents a simple sentence in Bangla language. A simple sentence may be a sentence itself, or may be a clause or part of clause of a complex sentence, or may be a part of a compound sentence. For example, “আমি ঢাকা যাব” (ami Dhaka zabo), “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), “আমি ঢাকা যাব আর তুমি সিলেট যাবে” (ami Dhaka zabo ar tumi sileT zabe) are represented by “SS”.

C1: The non-terminal “C1” is a factor in complex sentence level. It represents words after noun phrase of a simple sentence. The noun phrase may be actual or inner and does not start with an unknown word. For example,

the sentence segments “আমি ঢাকা যাব” (ami Dhaka zabo), “আমি ও তুমি ঢাকা যাব” (ami O tumi Dhaka zabo) are represented by “C1”.

C2: The non-terminal “C2” is a factor in complex sentence level. It represents words after noun phrase of a simple sentence. The noun phrase must start with an unknown word. For example, the sentence segments “রবিন ঢাকা যায়” (robin Dhaka zay), “রবিন ও জাকির ঢাকা যায়” (robin O jakir Dhaka zay) are represented by “C2”.

7.4.4 Significance of Non-terminals of Verb Phrase Level

We have summarized the significance of non-terminals of verb phrase level as follows,

VP: The non-terminal “VP” represents a verb phrase in Bangla language. A verb phrase is present in all types of sentence – simple, complex and compound. For example, the verb phrases “আমি ঢাকা যাই” (ami Dhaka zai), “যদি আমি যাই তবে তুমি এসো” (zodi ami zai tobe tumi esO), “আমি ঢাকা যাব আর তুমি সিলেট যাবে” (ami Dhaka zabo ar tumi sileT zabe) are represented by “VP”.

D1: The non-terminal “D1” is a factor in verb phrase level. It represents words of verb phrase which occurs after noun phrase which is belong to the verb phrase. Therefore, “D1” may represent a verb form. It also may represent an adjective phrase belong to the verb phrase followed by a verb form. For example, the sentence segments “আমি ভাত খাই” (ami vat khai), “আমি তাকে ভালভাবে চিনি” (ami take valovabe chini) are represented by “D1”.

D2: The non-terminal “D2” is a factor in verb phrase level. If an adjective or a group adjectives first in a verb phrase, then “D2” represents words of verb phrase occurring after the adjective or the group of adjectives. For

example, the sentence segments “আমি বেশি খাই” (ami beshi khai), “আমি খুব বেশি ভাত খাই” (ami khub beshi vat khai) are represented by “D2”.

D3: The non-terminal “D3” represents a class of verb phrase, which does not start with an unknown word. For example, the verb phrases “আমি খাই” (ami khai), “আমি বেশি খাই” (ami beshi khai), “আমি বেশি ভাত খাই” (ami beshi vat khai) are represented by “D3”.

D4: The non-terminal “D4” is a verb phrase level non-terminal. If a noun, other than unknown word, occurs in a verb phrase and there is no adjective before the noun then the noun along with words of verb phrase after the noun is represented by “D4”. For example, the sentence segment “আমি ভাত খাই” (ami vat khai) is represented by “D4”.

7.4.5 Significance of Non-terminals of Noun Phrase Level

We have summarized the significance of non-terminals of noun phrase level as follows,

NP: The non-terminal “NP” represents a noun phrase in Bangla language. Noun phrase occurs in all three types of sentences – simple, complex and compound. A noun phrase may also contain some smaller noun phrases. For example, the noun phrases “আমি ঢাকা যাই” (ami Dhaka zai), “আমার বন্ধু ঢাকা যায়” (amar bondhu Dhaka zay), “আমার ভাইয়ের বন্ধু ঢাকা যায়” (amar vaiyer bondhu Dhaka zay), “আমি ও তুমি ঢাকা যাব” (ami O tumi Dhaka zabo) are represented by “NP”.

E1: The non-terminal “E1” is a factor in noun phrase level. If a conjunctive occurs in a noun phrase, it represents the conjunctive and rest of the words of the noun phrase. If conjunctive is not present in a noun phrase, it represents null. For example, the sentence segments “আমি ও তুমি ঢাকা যাই”

(ami O tumi Dhaka zai), “আমি ও আমার ভাইয়ের বন্ধু ঢাকা যাই” (ami O amar vaiyer bondhu Dhaka zai) are represented by “E1”.

NPU: The non-terminal “NPU” represents a type of noun phrase, which does not contain any inner noun phrase, but this may be contained in another noun phrase. For example, the noun phrases “আমি ঢাকা যাই” (ami Dhaka zai), “আমি ও তুমি ঢাকা যাই” (ami O tumi Dhaka zai), “রবিন এর বন্ধু ঢাকা যাবে” (robin er bondhu Dhaka zabe) are presented by “NPU”.

E2: The non-terminal “E2” is a factor in noun phrase level. If a bivokti occurs in a noun phrase, then it represents the bivokti and the rest of the words of the noun phrase. Otherwise, it represents null. In case of extensive bivokti, “E2” represents the bivokti followed by another noun phrase. But in case of non-extensive bivokti, it represents only the bivokti. For example, the sentence segments “আমি এর বন্ধু ভাত খায়” (ami er bondhu vat khay), “আমি এর ভাই এর বন্ধু ভাত খায়” (ami er vai er bondhu vat khay), “আমি কে ভাত দাও” (ami ke vat dao) are represented by “E2”.

PRE: The non-terminal “PRE” represents words of a smaller noun phrase occurring before the noun. In fact, “PRE” represents different combinations of demonstrator, specifier and adjectives. For example, the sentence segments “এঁ ছেলেটি ঢাকা যাবে” (OI chheleTi Dhaka zabe), “এঁ প্রথম ছেলেটি ঢাকা যাবে” (OI prothom chheleTi Dhaka zabe), “এঁ প্রথম একটি সুন্দর ছেলে ঢাকা যাবে” (OI prothom ekoTi sundor chhele Dhaka zabe) are represented by “PRE”.

E3: The non-terminal “E3” is a factor in noun phrase level. It represents a specifier or an adjective phrase or a specifier followed by an adjective phrase. If specifier or adjective phrase is not present in a noun phrase, “E3” represents null. For example, “একটি ছেলে ঢাকা যাবে” (ekoTi chhele Dhaka zabe), “সুন্দর ছেলেটি ঢাকা যাবে” (sundor chheleTi Dhaka zabe), “একটি

সুন্দর ছেলে ঢাকা যাবে” (ekoTi sundor chhele Dhaka zabe) are represented by “E3”.

- E4:** The non-terminal “E4” is a factor in noun phrase level. If adjective phrase is present in a noun phrase, then it represents the adjective phrase. Otherwise, it represents null. For example, “সুন্দর ছেলেটি ঢাকা যাবে” (sundor chheleTi Dhaka zabe), “খুব সুন্দর ছেলেটি ঢাকা যাবে” (khub sundor chheleTi Dhaka zabe) are represented by “E4”.
- NW:** The non-terminal “NW” is represented as “noun word”, which consists of a noun, optionally followed by a determiner or a plural marker. For example, “আমি ফুটবল খেলি” (ami fuTobol kheli), “ছেলেটি ফুটবল খেলে” (chheleTi fuTobol khele), “ছেলেরা ফুটবল খেলে” (chhelera fuTobol khele) are represented by “NW”.
- E5:** The non-terminal “E5” is a factor in noun phrase level. It represents a determiner or a plural marker or null, followed by a noun of a noun phrase. For example, “ছেলে টি ফুটবল খেলে” (chhele Ti fuTobol khele), “ছেলে রা ফুটবল খেলে” (chhele ra fuTobol khele) are represented by “E5”.
- SPR:** The non-terminal “SPR” is represented as “specifier”, which consists of a “quantifier”, optionally followed by a “post preposition”. For example, “একটি ছেলে ফুটবল খেলে” (ekoTi chhele fuTobol khele), “এক লোক ফুটবল খেলে” (ek lOk fuTobol khele) are represented by “SPR”.
- E6:** The non-terminal “E6” is a factor in noun phrase level. It represents a “post preposition” after a “quantifier” or null. For example, “এক টি ছেলে ফুটবল খেলে” (ek Ti chhele fuTobol khele), “এক জন লোক ফুটবল খেলে” (ek jon lOk fuTobol chele) are represented by “E6”.
- DEMO:** The non-terminal “DEMO” is represented is “demonstrator”. It consists

of a “demonstratic deictic” or “demonstratic ordinal” or both. For example, “এ ছেলেটি ফুটবল খেলে” (OI chheleTi fuTobol khele), “প্রথম ছেলেটি ফুটবল খেলে” (prothom chheleTi fuTobol khele), “এ প্রথম ছেলেটি ফুটবল খেলে” (OI prothom chheleTi fuTobol khele) are represented by “DEMO”.

E7: The non-terminal “E7” is a factor in noun phrase level. If “demonstratic deictic” is present in a “demonstrator”, then “E7” represents words of “demonstrator” occurring after “demonstratic deictic”. Therefore, “E7” represents “demonstratic ordinal” or null. For example, “এ প্রথম ছেলেটি ফুটবল খেলে” (OI prothom chheleTi fuTobol khele) is represented by “E7”.

UNG: The non-terminal “UNG” is represented as “unknown word group”. It consists of an unknown word or a group of unknown words together. For example, “বেকহ্যাম ফুটবল খেলে” (bekohyam fuTobol khele), “ডেভিড বেকহ্যাম ফুটবল খেলে” (DeviD bekohyam fuTobol khele) are represented by “UNG”.

E8: The non-terminal “E8” is a factor in noun phrase level. It represents words of an unknown word group occurring after an unknown word. Therefore, it represents an unknown word or an unknown word group or null. For example, “ডেভিড বেকহ্যাম ফুটবল খেলে” (DeviD bekohyam fuTobol khele) is represented by “E8”.

7.4.6 Significance of Non-terminals of Adjective Phrase Level

We have summarized the significance of non-terminals of adjective phrase level as follows,

AP: The non-terminal “AP” represents as “adjective phrase”. It consists of an adjective or a group of adjectives. For example, “আমি বেশি ভাত খাই” (ami beshi vat khai), “আমি খুব বেশি ভাত খাই” (ami khub beshi vat khai) are represented by “AP”.

F1: The non-terminal “F1” is a factor in adjective phrase level. It represents words of an adjective phrase occurring after an adjective. Therefore, it represents an adjective or a group of adjectives of null. For example, “আমি খুব বেশি ভাত খাই” (ami khub beshi vat khai) is represented by “F1”.

7.5 Remarks

In this chapter, we have designed non-ambiguous comprehensive Bangla grammar with predictive parsing. Application of the grammar into predictive parser is elaborated in the next chapter. In the later part of this chapter, we have discussed the significance of each non-terminals of new proposed Bangla grammar.

8.1 Predictive Parser Architecture

In the previous four chapters (chapter – 4, 5, 6, 7), we have designed non-ambiguous grammar for Bangla language with predictive parsing. With ambiguous grammar, it is not possible to build predictive parsing table. It causes multiple rules to fall in a single grid of parsing table. Similar problem arises in grammar without predictive parsing. When predictive parsing is not possible, backtracking algorithm requires to parsing which takes exponential runtime. In contrast, predictive parser takes linear runtime for parsing purpose.

In this chapter, we will design a predictive parser module for Bangla language using non-ambiguous grammar with predictive parsing. In general, the parser module consists of two stages – a lexical analyzer and a syntax analyzer. In fact, the syntax analyzer is called the parser, as it generates parsed text. The parser module has two other supporting sub-modules - a lexicon or dictionary and a parsing table. The full architecture is demonstrated in figure 8.1.

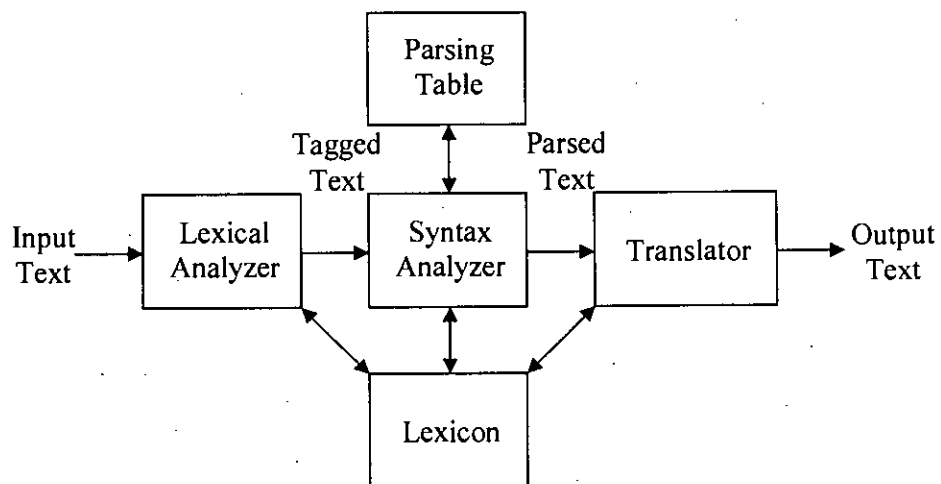


Figure 8.1: Architecture of a machine translator using predictive parser.

8.2 Lexicon

A lexicon can also be referred to be a dictionary. Generally, two types of information are stored in lexicon. First information is category of source language words. And, second information is translated word of destination language. For parsing stage, first information is necessary.

There are 16 terminals in our designed non-ambiguous grammar with predictive parsing. These terminals are – Conj, SUBCOR, SUBCOM, VR, AUX, N, UN, DD, DO, SPR, PP, BivE, Biv, DET, PM, AD. Out of these terminals, the terminal “UN” refers to an unknown word or non-dictionary word. Therefore, words in lexicon or dictionary are divided into 15 groups.

Table 8.1: Lexicon of Bangla language for a predictive parser.

Word	Category
অথবা	Conj
অনেক	QFR, AD
আই	AUX
আও	AUX
আমরা	আমি + রা
আমাকে	আমি + কে
আমার	আমি + এর
আমি	N
আস	VR
আস	আস + ও
ই	AUX
এ	AUX
এই	DD
এক	QFR
একটি	এক + টি
এবং	Conj
এর	BivE
ঐ	DD
ও	AUX, Conj, N

Here, another thing to be considered is, words may be simple or complex. Lexicon will maintain type of word for each simple word. Sometimes, a single word may be fit into several categories. For example, “ও” (O) may be AUX or Conj or N. Lexicon should maintain such information. Complex words consist of several simple words. Therefore, complex words can not be tagged with a category. Lexicon will maintain information how the complex words are formed. Table 8.1 shows a sample structure of lexicon.

There are some simple words in the lexicon which belong to single category. For example, “অথবা” (othoba) is belong to “Conj” category, “আই” (ai) is belong to “AUX” category, etc.

There are some simple words in the lexicon which belong to multiple categories. For example, “অনেক” (onek) is belong to “QFR” and “AD” category.

The rest of the words of lexicon are complex words, which can not be categorized into a particular category. This is because, they consist of several simple words. Categories of these simple words can be found by looking up the lexicon. For example, “আমরা” (amora) is a complex word which consists of two simple words “আমি” (ami) and “রা” (ra). Here, “আমি” (ami) is belong to “N” category and “রা” (ra) is belong to “PM” category.

8.3 Parsing Table

A parsing table is a table describing what action its parser should take when a given input comes while it is in a given state. It is a tabular representation that is generated from the context-free grammar of the language to be parsed. A predictive parser has a two dimensional parsing table – one dimension is non-terminal and another one is terminal. From current non-terminal and terminal, a particular rule can be looked up.

For building a predictive parsing table, *First* and *Follow* (as explained in section 2.8.1 and 2.8.2) of all non-terminals is needed to be calculated. Calculation of *First* and *Follow* is explained in next two sections (section 8.3.1 and 8.3.2).

8.3.1 Calculation of First

Using the definition of *First* (as discussed in section 2.8.1), we can calculate *First* of all non-terminals of Bangla grammar as follows,

$$\text{First}(S) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM \}$$

$$\text{First}(BS) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM \}$$

$$\text{First}(A1) = \{ \text{Conj}, \varepsilon \}$$

$$\text{First}(A2) = \{ VR, AD, N, DD, DO, QFR, UN, \text{Conj}, SUBORD, SUBCOM \}$$

$$\text{First}(A3) = \{ \text{Conj}, \text{BivE}, \text{Biv}, VR, AD, N, DD, DO, QFR, SUBORD, SUBCOM \}$$

$$\text{First}(A4) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \varepsilon \}$$

$$\text{First}(A5) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(CS) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM \}$$

$$\text{First}(B1) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM \}$$

$$\text{First}(B2) = \{ SUBORD, SUBCOM \}$$

$$\text{First}(B3) = \{ N, DD, DO, QFR, AD, UN, SUBCOM \}$$

$$\text{First}(B4) = \{ N, DD, DO, QFR, AD, UN, SUBORD \}$$

$$\text{First}(B5) = \{ VR, AD, N, DD, DO, QFR, UN, \text{Conj}, SUBORD, SUBCOM \}$$

$$\text{First}(B6) = \{ \text{Conj}, \text{BivE}, \text{Biv}, VR, AD, N, DD, DO, QFR, SUBORD, SUBCOM \}$$

$$\text{First}(B7) = \{ SUBORD, SUBCOM \}$$

$$\text{First}(B8) = \{ VR, AD, N, DD, DO, QFR, UN, \text{Conj}, SUBORD, SUBCOM \}$$

$$\text{First}(B9) = \{ \text{Conj}, \text{BivE}, \text{Biv}, VR, AD, N, DD, DO, QFR, SUBORD, SUBCOM \}$$

$$\text{First}(B10) = \{ VR, AD, N, DD, DO, QFR, UN, \text{Conj}, SUBCOM \}$$

$$\text{First}(B11) = \{ \text{Conj}, \text{BivE}, \text{Biv}, VR, AD, N, DD, DO, QFR, SUBCOM \}$$

$$\text{First}(B12) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(B13) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(B14) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(SS) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(C1) = \{ VR, AD, N, DD, DO, QFR, UN, Conj \}$$

$$\text{First}(C2) = \{ Conj, BivE, Biv, VR, AD, N, DD, DO, QFR \}$$

$$\text{First}(VP) = \{ VR, AD, N, DD, DO, QFR, UN \}$$

$$\text{First}(D1) = \{ VR, AD \}$$

$$\text{First}(D2) = \{ VR, N \}$$

$$\text{First}(D3) = \{ VR, AD, N, DD, DO, QFR \}$$

$$\text{First}(D4) = \{ N \}$$

$$\text{First}(VF) = \{ VR \}$$

$$\text{First}(NP) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(E1) = \{ Conj, \varepsilon \}$$

$$\text{First}(NPU) = \{ N, DD, DO, QFR, AD, UN \}$$

$$\text{First}(E2) = \{ BivE, Biv, \varepsilon \}$$

$$\text{First}(PRE) = \{ DD, DO, QFR, AD \}$$

$$\text{First}(E3) = \{ QFR, AD, \varepsilon \}$$

$$\text{First}(E4) = \{ AD, \varepsilon \}$$

$$\text{First}(NW) = \{ N \}$$

$$\text{First}(E5) = \{ DET, PM, \varepsilon \}$$

$$\text{First}(SPR) = \{ QFR \}$$

$$\text{First}(E6) = \{ PP, \varepsilon \}$$

$$\text{First}(DEMO) = \{ DD, DO \}$$

$$\text{First}(E7) = \{ DO, \varepsilon \}$$

$$\text{First}(UNG) = \{ UN \}$$

$$\text{First}(E8) = \{ UN, \varepsilon \}$$

$$\text{First}(AP) = \{ AD \}$$

$$\text{First}(F1) = \{ AD, \varepsilon \}$$

8.3.2 Calculation of Follow

Using the definition of *Follow* (as discussed in section 2.8.1), we can calculate *Follow* of all non-terminals of Bangla grammar as follows,

$\text{Follow}(S) = \{ \$ \}$
 $\text{Follow}(BS) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(A1) = \{ \$ \}$
 $\text{Follow}(A2) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(A3) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(A4) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(A5) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(CS) = \{ \}$
 $\text{Follow}(B1) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B2) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B3) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B4) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B5) = \{ \}$
 $\text{Follow}(B6) = \{ \}$
 $\text{Follow}(B7) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B8) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B9) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B10) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B11) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B12) = \{ \}$
 $\text{Follow}(B13) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(B14) = \{ \text{Conj}, \$ \}$
 $\text{Follow}(SS) = \{ N, DD, DO, QFR, AD, UN, SUBCOM, SUBORD, \text{Conj}, \$ \}$
 $\text{Follow}(C1) = \{ N, DD, DO, QFR, AD, UN, SUBCOM, SUBORD, \text{Conj}, \$ \}$
 $\text{Follow}(C2) = \{ N, DD, DO, QFR, AD, UN, SUBCOM, SUBORD, \text{Conj}, \$ \}$
 $\text{Follow}(VP) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$
 $\text{Follow}(D1) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$
 $\text{Follow}(D2) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$
 $\text{Follow}(D3) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$
 $\text{Follow}(D4) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$
 $\text{Follow}(VF) = \{ N, DD, DO, QFR, AD, UN, SUBORD, SUBCOM, \text{Conj}, \$ \}$

$\text{Follow}(\text{NP}) = \{ \text{SUBORD}, \text{VR}, \text{AD}, \text{N}, \text{DD}, \text{DO}, \text{QFR}, \text{UN}, \text{Conj}, \text{SUBCOM} \}$
 $\text{Follow}(\text{E1}) = \{ \text{VR}, \text{AD}, \text{SUBORD} \}$
 $\text{Follow}(\text{NPU}) = \{ \text{Conj}, \text{SUBORD} \}$
 $\text{Follow}(\text{E2}) = \{ \text{VR}, \text{AD}, \text{N}, \text{DD}, \text{DO}, \text{QFR}, \text{UN}, \text{Conj}, \text{SUBORD}, \text{SUBCOM} \}$
 $\text{Follow}(\text{PRE}) = \{ \text{N} \}$
 $\text{Follow}(\text{E3}) = \{ \text{N} \}$
 $\text{Follow}(\text{E4}) = \{ \text{N} \}$
 $\text{Follow}(\text{NW}) = \{ \text{BivE}, \text{Biv}, \text{Conj}, \text{SUBORD} \}$
 $\text{Follow}(\text{E5}) = \{ \text{BivE}, \text{Biv}, \text{Conj}, \text{SUBORD} \}$
 $\text{Follow}(\text{SPR}) = \{ \text{AD}, \text{N} \}$
 $\text{Follow}(\text{E6}) = \{ \text{AD}, \text{N} \}$
 $\text{Follow}(\text{DEMO}) = \{ \text{QFR}, \text{AD}, \text{N} \}$
 $\text{Follow}(\text{E7}) = \{ \text{QFR}, \text{AD}, \text{N} \}$
 $\text{Follow}(\text{UNG}) = \{ \text{Conj}, \text{BivE}, \text{Biv}, \text{VR}, \text{AD}, \text{N}, \text{DD}, \text{DO}, \text{QFR}, \text{SUBORD}, \text{SUBCOM} \}$
 $\text{Follow}(\text{E8}) = \{ \text{Conj}, \text{BivE}, \text{Biv}, \text{VR}, \text{AD}, \text{N}, \text{DD}, \text{DO}, \text{QFR}, \text{SUBORD}, \text{SUBCOM} \}$
 $\text{Follow}(\text{AP}) = \{ \text{VR}, \text{N} \}$
 $\text{Follow}(\text{F1}) = \{ \text{VR}, \text{N} \}$

8.3.3 Building of Parsing Table

Using the set of *First* and *Follow* of all non-terminals of Bangla grammar, we can build predictive parsing table using the technique of building of predictive parsing table as discussed in section 2.8.3. To achieve error recovery facility, we can put error recovery information into parsing table using the theory discussed in section 2.8.4.

Non-ambiguous Bangla grammar has 48 non-terminals and 16 terminals. It is not easy to display a 48×16 sized table. We can divide the parsing table into several

segments like – sentence portion, complex sentence portion, simple sentence portion, verb phrase portion, noun phrase portion and adjective phrase portion.

Predictive parsing table of sentence portion is shown in table 8.2, complex sentence portion in table 8.3, simple sentence portion in table 8.4, verb phrase portion in table 8.5, noun phrase portion in table 8.6 and adjective phrase portion in table 8.7.

Table 8.2 (a): Predictive parsing table for sentence portion.

	Conj	SUBORD	SUBCOM	VR	AUX
S	-	S → BS A1	S → BS A1	-	-
BS	sync	BS → SUBORD SS B3	BS → SUBCOM SS B4	-	-
A1	A1 → Conj S	-	-	-	-
A2	A2 → Conj A5	A2 → B2	A2 → B2	A2 → VP A4	-
A3	A3 → Conj SS A4	A3 → B2	A3 → B2	A3 → D3 A4	-
A4	A4 → ε	A4 → B1	A4 → B1	-	-
A5	sync	-	-	-	-

Table 8.2 (b): Predictive parsing table for sentence portion.

	N	UN	DD	DO	QFR	PP
S	S → BS A1	S → BS A1	S → BS A1	S → BS A1	S → BS A1	-
BS	BS → NW E2 A2	BS → UNG A3	BS → PRE NW E2 A2	BS → PRE NW E2 A2	BS → PRE NW E2 A2	-
A1	-	-	-	-	-	-
A2	A2 → VP A4	A2 → VP A4	A2 → VP A4	A2 → VP A4	A2 → VP A4	-
A3	A3 → D3 A4	-	A3 → D3 A4	A3 → D3 A4	A3 → D3 A4	-
A4	A4 → B1	A4 → B1	A4 → B1	A4 → B1	A4 → B1	-
A5	A5 → NW E2 A2	A5 → UNG A3	A5 → PRE NW E2 A2	A5 → PRE NW E2 A2	A5 → PRE NW E2 A2	-

Table 8.2 (c): Predictive parsing table for sentence portion.

	BivE	Biv	DET	PM	AD	\$
S	-	-	-	-	S → BS A1	sync
BS	-	-	-	-	BS → PRE NW E2 A2	sync
A1	-	-	-	-	-	A1 → ε
A2	-	-	-	-	A2 → VP A4	sync
A3	A3 → BivE A5	A3 → Biv A2	-	-	A3 → D3 A4	sync
A4	-	-	-	-	A4 → B1	A4 → ε
A5	-	-	-	-	A5 → PRE NW E2 A2	sync

Table 8.3 (a): Predictive parsing table for complex sentence portion.

	Conj	SUBORD	SUBCOM	VR	AUX
CS	-	CS → SUBORD SS B3	CS → SUBCOM SS B4	-	-
B1	sync	B1 → SUBORD SS	B1 → SUBCOM SS	-	-
B2	sync	B2 → SUBORD VP B3	B2 → SUBCOM VP B4	-	-
B3	sync	-	B3 → SUBCOM SS	-	-
B4	sync	B4 → SUBORD SS	-	-	-
B5	B5 → Conj B12	B5 → B2	B5 → B2	B5 → VP B1	-
B6	B6 → Conj SS B1	B6 → B2	B6 → B2	B6 → D3 B1	-
B7	sync	B7 → SUBORD VP	B7 → SUBCOM VP	-	-
B8	B8 → Conj B13	B8 → B7	B8 → B7	B8 → VP	-
B9	B9 → Conj B13	B9 → B7	B9 → B7	B9 → D3	-
B10	B10 → Conj B14	-	B10 → SUBCOM VP	B10 → VP	-
B11	B11 → Conj B14	-	B11 → SUBCOM VP	B11 → D3	-
B12	-	-	-	-	-
B13	sync	-	-	-	-
B14	sync	-	-	-	-

Table 8.3 (b): Predictive parsing table for complex sentence portion.

	N	UN	DD	DO	QFR	PP
CS	CS → NW E2 B5	CS → UNG B6	CS → PRE NW E2 B5	CS → PRE NW E2 B5	CS → PRE NW E2 B5	-
B1	B1 → NW E2 B8	B1 → UNG B9	B1 → PRE NW E2 B8	B1 → PRE NW E2 B8	B1 → PRE NW E2 B8	-
B2	-	-	-	-	-	-
B3	B3 → NW E2 B10	B3 → UNG B11	B3 → PRE NW E2 B10	B3 → PRE NW E2 B10	B3 → PRE NW E2 B10	-
B4	B4 → NP SUBORD VP	B4 → NP SUBORD VP	B4 → NP SUBORD VP	B4 → NP SUBORD VP	B4 → NP SUBORD VP	-
B5	B5 → VP B1	B5 → VP B1	B5 → VP B1	B5 → VP B1	B5 → VP B1	-
B6	B6 → D3 B1	-	B6 → D3 B1	B6 → D3 B1	B6 → D3 B1	-
B7	-	-	-	-	-	-
B8	B8 → VP	B8 → VP	B8 → VP	B8 → VP	B8 → VP	-
B9	B9 → D3	-	B9 → D3	B9 → D3	B9 → D3	-
B10	B10 → VP	B10 → VP	B10 → VP	B10 → VP	B10 → VP	-
B11	B11 → D3	-	B11 → D3	B11 → D3	B11 → D3	-
B12	B12 → NW E2 B5	B12 → UNG B6	B12 → PRE NW E2 B5	B12 → PRE NW E2 B5	B12 → PRE NW E2 B5	-
B13	B13 → NW E2 B8	B13 → UNG B9	B13 → PRE NW E2 B8	B13 → PRE NW E2 B8	B13 → PRE NW E2 B8	-
B14	B14 → NW E2 B10	B14 → UNG B11	B14 → PRE NW E2 B10	B14 → PRE NW E2 B10	B14 → PRE NW E2 B10	-

Table 8.3 (c): Predictive parsing table for complex sentence portion.

	BivE	Biv	DET	PM	AD	S
CS	-	-	-	-	CS → PRE NW E2 B5	-
B1	-	-	-	-	B1 → PRE NW E2 B8	sync
B2	-	-	-	-	-	sync
B3	-	-	-	-	B3 → PRE NW E2 B10	sync
B4	-	-	-	-	B4 → NP SUBORD VP	sync

B5	-	-	-	-	B5 → VP B1	-
B6	B6 → BivE B12	B6 → Biv B5	-	-	B6 → D3 B1	-
B7	-	-	-	-	-	sync
B8	-	-	-	-	B8 → VP	sync
B9	B9 → BivE B13	B9 → Biv B8	-	-	B9 → D3	sync
B10	-	-	-	-	B10 → VP	sync
B11	B11 → BivE B14	B11 → Biv B10	-	-	B11 → D3	sync
B12	-	-	-	-	B12 → PRE NW E2 B5	-
B13	-	-	-	-	B13 → PRE NW E2 B8	sync
B14	-	-	-	-	B14 → PRE NW E2 B10	sync

Table 8.4 (a): Predictive parsing table for simple sentence portion.

	Conj	SUBORD	SUBCOM	VR	AUX
SS	sync	sync	sync	-	-
C1	C1 → Conj SS	sync	sync	C1 → VP	-
C2	C2 → Conj SS	sync	sync	C2 → D3	-

Table 8.4 (b): Predictive parsing table for simple sentence portion.

	N	UN	DD	DO	QFR	PP
SS	SS → NW E2 C1	SS → UNG C2	SS → PRE NW E2 C1	SS → PRE NW E2 C1	SS → PRE NW E2 C1	-
C1	C1 → VP	C1 → VP	C1 → VP	C1 → VP	C1 → VP	-
C2	C2 → D3	sync	C2 → D3	C2 → D3	C2 → D3	-

Table 8.4 (c): Predictive parsing table for simple sentence portion.

	BivE	Biv	DET	PM	AD	\$
SS	-	-	-	-	SS → PRE NW E2 C1	sync
C1	-	-	-	-	C1 → VP	sync
C2	C2 → BivE SS	C2 → Biv C1	-	-	C2 → D3	sync

Table 8.5 (a): Predictive parsing table for verb phrase portion.

	Conj	SUBORD	SUBCOM	VR	AUX
VP	sync	sync	sync	VP → D3	-
D1	sync	sync	sync	D1 → VF	-
D2	sync	sync	sync	D2 → VF	-
D3	sync	sync	sync	D3 → VF	-
D4	sync	sync	sync	-	-
VF	sync	sync	sync	VF → VR AUX	-

Table 8.5 (b): Predictive parsing table for verb phrase portion.

	N	UN	DD	DO	QFR	PP
VP	VP → D3	VP → UNG E2 E1 D1	VP → D3	VP → D3	VP → D3	-
D1	sync	sync	sync	sync	sync	-
D2	D2 → D4	sync	sync	sync	sync	-
D3	D3 → D4	sync	D3 → DEMO E3 D4	D3 → DEMO E3 D4	D3 → SPR E4 D4	-
D4	D4 → NW E2 E1 D1	sync	sync	sync	sync	-
VF	sync	sync	sync	sync	sync	-

Table 8.5 (c): Predictive parsing table for verb phrase portion.

	BivE	Biv	DET	PM	AD	\$
VP	-	-	-	-	VP → D3	sync
D1	-	-	-	-	D1 → AP VF	sync
D2	-	-	-	-	sync	sync
D3	-	-	-	-	D3 → AP D2	sync
D4	-	-	-	-	sync	sync
VF	-	-	-	-	sync	sync

Table 8.6 (a): Predictive parsing table for noun phrase portion.

	Conj	SUBORD	SUBCOM	VR	AUX
NP	sync	sync	sync	sync	-
E1	E1 → Conj NP	E1 → ϵ	-	E1 → ϵ	-
NPU	sync	sync	-	-	-
E2	E2 → ϵ	E2 → ϵ	E2 → ϵ	E2 → ϵ	-
PRE	-	-	-	-	-

E3	-	-	-	-	-
E4	-	-	-	-	-
NW	sync	sync	-	-	-
E5	$E5 \rightarrow \epsilon$	$E5 \rightarrow \epsilon$	-	-	-
SPR	-	-	-	-	-
E6	-	-	-	-	-
DEMO	-	-	-	-	-
E7	-	-	-	-	-
UNG	sync	sync	sync	sync	-
E8	$E8 \rightarrow \epsilon$	$E8 \rightarrow \epsilon$	$E8 \rightarrow \epsilon$	$E8 \rightarrow \epsilon$	-

Table 8.6 (b): Predictive parsing table for noun phrase portion.

	N	UN	DD	DO	QFR	PP
NP	$NP \rightarrow$ NPU E1	$NP \rightarrow$ NPU E1	$NP \rightarrow$ NPU E1	$NP \rightarrow$ NPU E1	$NP \rightarrow$ NPU E1	-
E1	-	-	-	-	-	-
NPU	$NPU \rightarrow$ NW E2	$NPU \rightarrow$ UNG E2	$NPU \rightarrow$ PRE NW E2	$NPU \rightarrow$ PRE NW E2	$NPU \rightarrow$ PRE NW E2	-
E2	$E2 \rightarrow \epsilon$	$E2 \rightarrow \epsilon$	$E2 \rightarrow \epsilon$	$E2 \rightarrow \epsilon$	$E2 \rightarrow \epsilon$	-
PRE	sync	-	$PRE \rightarrow$ DEMO E3	$PRE \rightarrow$ DEMO E3	$PRE \rightarrow$ SPR E4	-
E3	$E3 \rightarrow \epsilon$	-	-	-	$E3 \rightarrow$ SPR E4	-
E4	$E4 \rightarrow \epsilon$	-	-	-	-	-
NW	$NW \rightarrow$ N E5	-	-	-	-	-
E5	-	-	-	-	-	-
SPR	sync	-	-	-	$SPR \rightarrow$ QFR E6	-
E6	$E6 \rightarrow \epsilon$	-	-	-	-	$E6 \rightarrow$ PP
DEMO	sync	-	$DEMO \rightarrow$ DD E7	$DEMO \rightarrow$ DO	sync	-
E7	$E7 \rightarrow \epsilon$	-	-	$E7 \rightarrow$ DO	$E7 \rightarrow \epsilon$	-
UNG	sync	$UNG \rightarrow$ UN E8	sync	sync	sync	-
E8	$E8 \rightarrow \epsilon$	$E8 \rightarrow$ UNG	$E8 \rightarrow \epsilon$	$E8 \rightarrow \epsilon$	sync	-

Table 8.6 (c): Predictive parsing table for noun phrase portion.

	BivE	Biv	DET	PM	AD	S
NP	-	-	-	-	$NP \rightarrow$ NPU E1	-
E1	-	-	-	-	$E1 \rightarrow \epsilon$	-
NPU	-	-	-	-	$NPU \rightarrow$ PRE NW E2	-

E2	E2 → BivE NP	E2 → Biv	-	-	E2 → ε	-
PRE	-	-	-	-	PRE → AP	-
E3	-	-	-	-	E3 → AP	-
E4	-	-	-	-	E4 → AP	-
NW	sync	sync	-	-	-	-
E5	E5 → ε	E5 → ε	E5 → DET	E5 → PM	-	-
SPR	-	-	-	-	sync	-
E6	-	-	-	-	E6 → ε	-
DEMO	-	-	-	-	sync	-
E7	-	-	-	-	E7 → ε	-
UNG	sync	sync	-	-	sync	-
E8	E8 → ε	E8 → ε	-	-	E8 → ε	-

Table 8.7 (a): Predictive parsing table for adjective phrase portion.

	Conj	SUBORD	SUBCOM	VR	AUX
AP	-	-	-	sync	-
F1	-	-	-	F1 → ε	-

Table 8.7 (b): Predictive parsing table for adjective phrase portion.

	N	UN	DD	DO	QFR	PP
AP	sync	-	-	-	-	-
F1	F1 → ε	-	-	-	-	-

Table 8.7 (c): Predictive parsing table for adjective phrase portion.

	BivE	Biv	DET	PM	AD	\$
AP	-	-	-	-	AP → AD F1	-
F1	-	-	-	-	F1 → AP	-

8.4 Lexical Analyzer

The main purpose of lexical analyzer is “parts of speech tagging”. There are different policies of tagging for simple and complex words. Some simple words are belong to single category and some others belong to multiple category. Again, there are some non-dictionary words in an input sentence. Tagging policies are described as follow,

- Simple words, which are belong to single category, are tagged immediately. For example, “আমি” (ami) is tagged immediately to “N” and <“আমি”, “N”> is generated.
- Simple words, which are belong to multiple categories, can not be tagged in this stage. Tagging is remained for syntax analyzer. To indicate multiple categories, “MULTI” tagging is done. For example, “ও” (O) is tagged to “MULTI” and <“ও”, “MULTI”> is generated.
- For complex words, three cases occur, as follows,
 - Firstly, the complex word may be found in the lexicon. Then, its forming simple words are retrieved and tagged. For example, the complex word “আমার” (amar) is found in lexicon, then its forming simple words “আমি” (ami) and “এর” (er) are retrieved. “আমি” (ami) is tagged to “N” and “এর” (er) is tagged to “BivE” and finally <“আমি”, “N”>, <“এর”, “BivE”> are generated.
 - Then, the complex word may not be found in the lexicon. Then, an algorithm analyzes the word, whether its segments can form some meaningful words found in lexicon. If the word can be broken into some other meaningful words, then these words are tagged. For example, the complex word “বন্ধুরা” (bondhura) is not found in lexicon. Algorithm tries to break the word and finds “বন্ধু” (bondhu) and “রা” (ra) within it. Then, “বন্ধু” (bondhu) is tagged to “N” and “রা” (ra) is tagged to “PM” and finally <“বন্ধু”, “N”>, <“রা”, “PM”> are generated.
 - Finally, the complex word may not be found in the lexicon and forming words may not be found breaking the complex word. In that case, we can apply some heuristics. There are some common patterns of common words like, “যাই” (“যা” + “আই”), “খাই” (“খা” + “আই”), “যায়” (“যা” + “আয়”), “খায়” (“খা” + “আয়”), “ছেলের” (“ছেলে” + “এর”), “মেয়ের” (“মেয়ে” + “এর”) etc. These words may not be found in lexicon and forming simple words can not be retrieved simply by breaking the word. So, using heuristic is effective here.

- For an unknown word or non-dictionary word, the word is tagged as “UN”. For example, “ঢাকা” (Dhaka) is tagged to “UN” and <“ঢাকা”, “UN”> is generated.

Table 8.8: A sample lexicon.

Word	Category
আমার	আমি + এর
আমি	N
এবং	Conj
এর	BivE
ও	N, Conj
বে	AUX
বন্ধু	N
ভাই	N
রা	PM
যা	VR

For example, the sentence “আমি, আমার ভাই, রবিন এবং ওর ভাইয়ের বন্ধুরা ঢাকা ও সিলেট যাবে” (ami, amar vai, robin ebong Or vaiyer bondhura Dhaka O sileT zabe) is to be parsed. Firstly, the sentence is passed to lexical analyzer for “parts of speech tagging”. We are considering the sample lexicon shown in table 8.8 here. The simple words “আমি” (ami), “,” (,), “ভাই” (vai), “এবং” (ebong) are tagged immediately. The simple word “ও” (O) is a multi category word. Therefore, it is tagged as “MULTI”. The complex word “আমার” (amar) can be broken and tagged from the lexicon. The complex words “বন্ধুরা” (bondhura), “যাবে” (zabe) can be broken using algorithm and tagged from the lexicon. The complex words “ওর” (Or), “ভাইয়ের” (vaiyer) is broken using heuristic and tagged from the lexicon. The rest of the words are “রবিন” (robin), “ঢাকা” (Dhaka) and “সিলেট” (sileT), which are not found in lexicon and can not be broken. These words are tagged as “UN”. Finally, the tagged text becomes <“আমি”, “N”>, <“,”, “Conj”>, <“আমি”, “N”>, <“এর”, “BivE”>, <“ভাই”, “N”>, <“,”, “Conj”>, <“রবিন”, “UN”>, <“এবং”, “Conj”>, <“ও”, “MULTI”>, <“এর”, “BivE”>, <“বন্ধু”, “N”>, <“রা”, “PM”>, <“ঢাকা”, “UN”>, <“ও”, “Conj”>, <“সিলেট”, “UN”>, <“যা”, “VR”>, <“বে”, “AUX”>. The work flow of lexical analyzer module is demonstrated in figure 8.2.

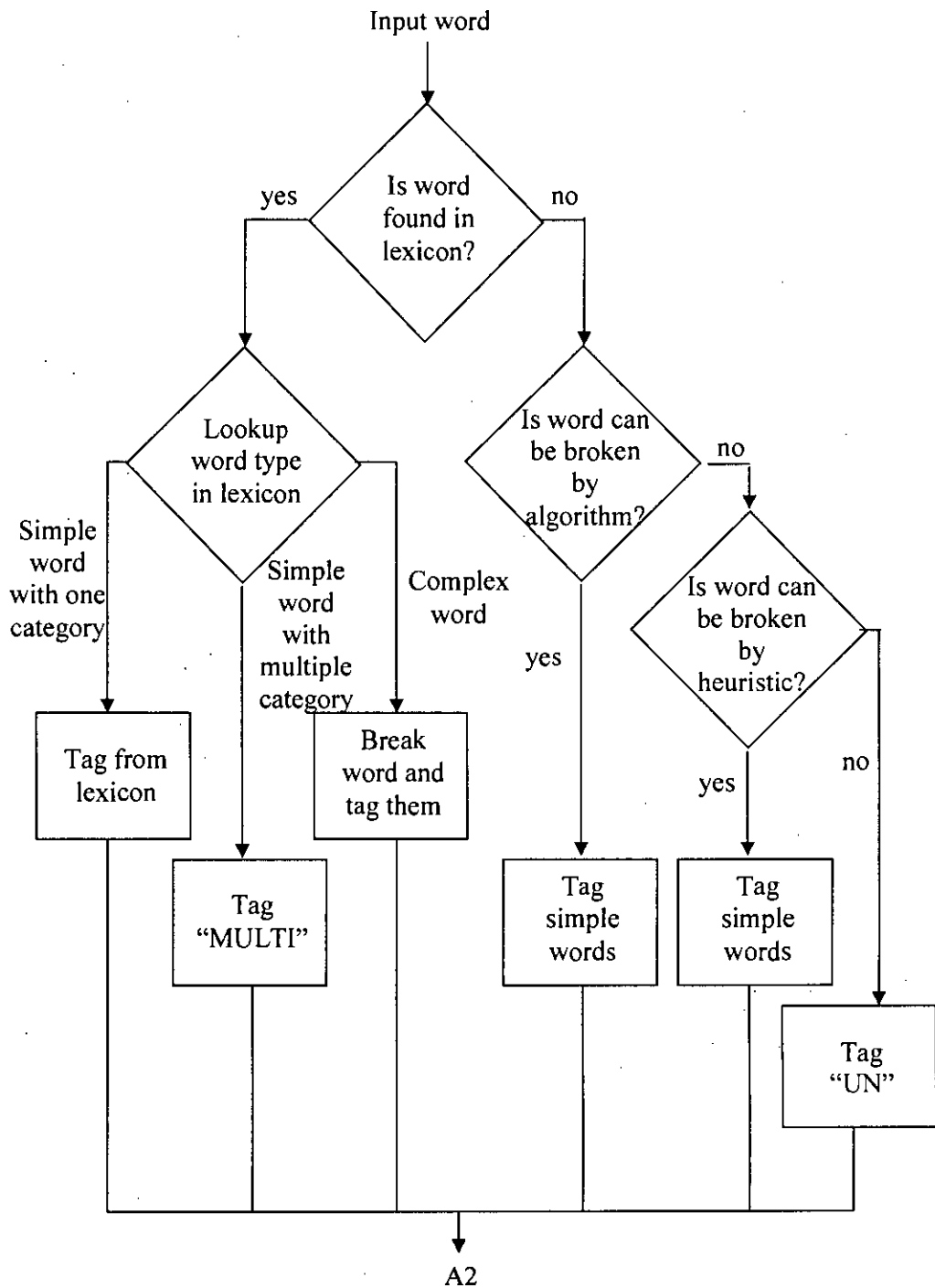


Figure 8.2: Work flow of lexical analyzer of predictive parser.

8.5 Syntax Analyzer

Syntax analyzer can also be called as parser. Our proposed grammar is designed for a top-down predictive parser. Therefore, top-down predictive parsing algorithm is deployed in the syntax analyzer. Architecture of syntax analyzer or parser was discussed in section 2.8 and top-down predictive parsing algorithm was discussed in algorithm 2.3. In the next section (section 8.6), we will see, how a correct Bangla sentence is parsed using the algorithm.

A modification is applied on algorithm 2.3 for dynamic resolve of “MULTI” tagged word. When a multi-tagged word arises, parser looks up lexicon for probable choices of tags. Then it tries to match the tag for probable applicable rule. If any probable applicable rule matches with tag of probable choices, the tag is assigned to the word. In the example of next section, we will watch how dynamic resolve works.

8.6 Parsing of Correct Sentence

In this section, we will watch how top-down predictive parsing algorithm works on an example sentence. Suppose, we want to parse the sentence “আমি, আমার ভাই, রবিন এবং ওর ভাইয়ের বন্ধুরা ঢাকা ও সিলেট যাবে” (ami, amar vai, robin ebong Or vaiyer bondhura Dhaka O sileT zabe). After lexical analysis, the input sentence becomes <“আমি”, “N”>, <“,”, “Conj”>, <“আমি”, “N”>, <“এর”, “BivE”>, <“ভাই”, “N”>, <“,”, “Conj”>, <“রবিন”, “UN”>, <“এবং”, “Conj”>, <“ও”, “MULTI”>, <“এর”, “BivE”>, <“বন্ধু”, “N”>, <“রা”, “PM”>, <“ঢাকা”, “UN”>, <“ও”, “Conj”>, <“সিলেট”, “UN”>, <“যা”, “VR”>, <“বে”, “AUX”>, <“\$”, “\$”>. The last symbol “\$” indicates an end-marker, which lexical analyzer places to indicate end of an input sentence.

The parser maintains a stack. Initially, an end-marker “\$” is pushed into the stack. Then starting non-terminal “S” is pushed into the stack. Then, the parsing process runs in following way,

Stack	Current symbol		Production
	Word	Tag	
\$ S	আমি	N	
\$ A1 BS	আমি	N	S → BS A1
\$ A1 A2 E2 NW	আমি	N	BS → NW E2 A2
\$ A1 A2 E2 E5 N	আমি	N	NW → N E5
\$ A1 A2 E2 E5	আমি	N	N → আমি
\$ A1 A2 E2	,	Conj	E5 → ε
\$ A1 A2	,	Conj	E2 → ε
\$ A1 A5 Conj	,	Conj	A2 → Conj A5
\$ A1 A5	,	Conj	Conj → ,
\$ A1 A2 E2 NW	আমি	N	A5 → NW E2 A2
\$ A1 A2 E2 E5 N	আমি	N	NW → N E5
\$ A1 A2 E2 E5	আমি	N	N → আমি
\$ A1 A2 E2	এর	BivE	E5 → ε
\$ A1 A2 NP BivE	এর	BivE	E2 → BivE NP
\$ A1 A2 NP	এর	BivE	BivE → এর
\$ A1 A2 E1 NPU	ভাই	N	NP → NPU E1
\$ A1 A2 E1 E2 NW	ভাই	N	NPU → NW E2
\$ A1 A2 E1 E2 E5 N	ভাই	N	NW → N E5
\$ A1 A2 E1 E2 E5	ভাই	N	N → ভাই
\$ A1 A2 E1 E2	,	Conj	E5 → ε
\$ A1 A2 E1	,	Conj	E2 → ε
\$ A1 A2 NP Conj	,	Conj	E1 → Conj NP
\$ A1 A2 NP	,	Conj	Conj → ,
\$ A1 A2 E1 NPU	রবিন	UN	NP → NPU E1
\$ A1 A2 E1 E2 UNG	রবিন	UN	NPU → UNG E2
\$ A1 A2 E1 E2 E8 UN	রবিন	UN	UNG → UN E8
\$ A1 A2 E1 E2 E8	রবিন	UN	UN → রবিন
\$ A1 A2 E1 E2	এবং	Conj	E8 → ε
\$ A1 A2 E1	এবং	Conj	E2 → ε
\$ A1 A2 NP Conj	এবং	Conj	E1 → Conj NP
\$ A1 A2 NP	এবং	Conj	Conj → এবং
\$ A1 A2 E1 NPU	ও	MULTI	NP → NPU E1
\$ A1 A2 E1 NPU	ও	N	
\$ A1 A2 E1 E2 NW	ও	N	NPU → NW E2
\$ A1 A2 E1 E2 E5 N	ও	N	NW → N E5
\$ A1 A2 E1 E2 E5	ও	N	N → ও
\$ A1 A2 E1 E2	এর	BivE	E5 → ε
\$ A1 A2 E1 NP BivE	এর	BivE	E2 → BivE NP
\$ A1 A2 E1 NP	এর	BivE	BivE → এর
\$ A1 A2 E1 E1 NPU	ভাই	N	NP → NPU E1
\$ A1 A2 E1 E1 E2 NW	ভাই	N	NPU → NW E2
\$ A1 A2 E1 E1 E2 E5 N	ভাই	N	NW → N E5

\$ A1 A2 E1 E1 E2 E5	ভাই	N	N → ভাই
\$ A1 A2 E1 E1 E2	এর	BivE	E5 → ϵ
\$ A1 A2 E1 E1 NP BivE	এর	BivE	E2 → BivE NP
\$ A1 A2 E1 E1 NP	এর	BivE	BivE → এর
\$ A1 A2 E1 E1 E1 NPU	বন্ধু	N	NP → NPU E1
\$ A1 A2 E1 E1 E1 E2 NW	বন্ধু	N	NPU → NW E2
\$ A1 A2 E1 E1 E1 E2 E5 N	বন্ধু	N	NW → N E5
\$ A1 A2 E1 E1 E1 E2 E5	বন্ধু	N	N → বন্ধু
\$ A1 A2 E1 E1 E1 E2 PM	রা	PM	E5 → PM
\$ A1 A2 E1 E1 E1 E2	রা	PM	PM → রা
\$ A1 A2 E1 E1 E1	ঢাকা	UN	E2 → ϵ
\$ A1 A2 E1 E1	ঢাকা	UN	E1 → ϵ
\$ A1 A2 E1	ঢাকা	UN	E1 → ϵ
\$ A1 A2	ঢাকা	UN	E1 → ϵ
\$ A1 A4 VP	ঢাকা	UN	A2 → VP A4
\$ A1 A4 D1 E1 E2 UNG	ঢাকা	UN	VP → UNG E2 E1 D1
\$ A1 A4 D1 E1 E2 E8 UN	ঢাকা	UN	UNG → UN E8
\$ A1 A4 D1 E1 E2 E8	ঢাকা	UN	UN → ঢাকা
\$ A1 A4 D1 E1 E2	ও	MULTI	E8 → ϵ
\$ A1 A4 D1 E1 E2	ও	Conj	
\$ A1 A4 D1 E1	ও	Conj	E2 → ϵ
\$ A1 A4 D1 NP Conj	ও	Conj	E1 → Conj NP
\$ A1 A4 D1 NP	ও	Conj	Conj → ও
\$ A1 A4 D1 E1 NPU	সিলেট	UN	NP → NPU E1
\$ A1 A4 D1 E1 E2 UNG	সিলেট	UN	NPU → UNG E2
\$ A1 A4 D1 E1 E2 E8 UN	সিলেট	UN	UNG → UN E8
\$ A1 A4 D1 E1 E2 E8	সিলেট	UN	UN → সিলেট
\$ A1 A4 D1 E1 E2	যা	VR	E8 → ϵ
\$ A1 A4 D1 E1	যা	VR	E2 → ϵ
\$ A1 A4 D1	যা	VR	E1 → ϵ
\$ A1 A4 VF	যা	VR	D1 → VF
\$ A1 A4 AUX VR	যা	VR	VF → VR AUX
\$ A1 A4 AUX	যা	VR	VR → যা
\$ A1 A4	বে	AUX	AUX → বে
\$ A1	\$	\$	A4 → ϵ
\$	\$	\$	A1 → ϵ

8.7 Error Recovery Policy

To recover from an error, firstly we have to detect when an error occurs. While parsing, if a terminal is found from stack and it does not match with input word tag,

then error occurs. Again, if a non-terminal is found from stack and no production is found for the non-terminal from parsing table then error occurs.

For error recovery purpose, we have pushed synchronizing set of many non-terminals into the parsing table. We have selected *Follow* set as synchronizing set. *Follow* set of a non-terminal indicates terminals occurring after the derivation of the non-terminal. Therefore, when a production is not found in the parsing table against the non-terminal, then synchronizing set is looked up. If matches with a member of synchronizing set, then a missing word is detected and parsing continues by popping top non-terminal from the stack.

In our non-ambiguous predictive parser following recovery policies are adopted when an error is detected,

1. If stack becomes empty and still some words are remained to be parsed, then parsing stops at that point.
2. If an unknown word is found in input, the word is skipped and next word will be considered to continue parsing.
3. If the non-terminal has a “null” production, the production is considered and parsing continues.
4. Words are skipped until a valid production or synchronizing set is found in the parsing table.
 - a. If valid production found, the production is considered and parsing continues.
 - b. If synchronizing set found, missing word is detected. Therefore, a missing word is reported and parsing continues.
 - c. If neither valid production nor synchronizing set is found, “null” production is considered if “null” production exists, otherwise missing word is reported and parsing continues.

Such error recovery policy allows to continue parsing when error found in input sentence, which was absent in previous Bangla parsing methodology [22][23].

Now, we are in a position to re-write algorithm 2.3, with error recovery policy. Such an algorithm is written in algorithm 8.1.

Algorithm 8.1: Non-recursive predictive parsing with error recovery policy.

Input. A string w and a parsing table M for Grammar G .

Output. If w is in $L(G)$, a leftmost derivation of w ; otherwise, an error indication.

Method.

Parse (w, M)

Push "\$" into stack "ST", where "\$" is end-marker
 Push "S" into stack "ST", where "S" is starting symbol
 Add "\$" after the sentence " w "
 set ip to point to the first symbol of w \$

repeat

let X be the top stack symbol and a the symbol pointed to by ip

if X is a terminal or \$ **then**

if $X = a$ **then**

 pop X from the stack and advance ip

else // symbol not matches

 Recover(ST, w, ip)

end if

else // X is a non-terminal

if $M[X, a] = X \rightarrow Y_1Y_2\dots Y_k$ **then**

 pop X from the stack

 push Y_k, Y_{k-1}, \dots, Y_1 onto the stack (Y_1 on top)

 output the production $X \rightarrow Y_1Y_2\dots Y_k$

else // rule not found on table

 Recover(ST, w, ip)

end if

end if

until $X = \$$ // stack is empty

end function

Recover (ST, w, ip)

let X be the top stack symbol and a the symbol pointed by ip

if $X = \$$ **then** // stack is empty

return

end if

```

if  $a = \text{"UN"}$  then // unknown symbol found
    error report on "unknown symbol"
    advance  $ip$  // symbol skipped
    return
end if

if  $X$  has "null" rule in  $M$  then
    pop  $X$  from the stack
    output the production  $X \rightarrow \epsilon$ 
    return
end if

while  $M[X][a]$  contains neither rule nor sync do
     $a$  is the symbol pointed by  $ip$ 
    if  $a = \$$  then
        break
    end if
    error report on "skipped symbol"
    advance  $ip$  // symbol skipped
end loop

if sync found in  $M[X][a]$  then
    pop  $X$  from the stack
    error report on "missing symbol"
else if rule not found in  $M[X][a]$  then
    if  $X$  has "null" rule in  $M$  then
        pop  $X$  from the stack
        output the production  $X \rightarrow \epsilon$ 
    else
        pop  $X$  from the stack
        error report on "missing symbol"
    end if
end if
end function

```

8.8 Parsing of Erroneous Sentence

In this section, we will watch, how error recovery routine, discussed in previous section (section 8.7) works on an erroneous Bangla sentence.

Firstly, we try to parse the sentence "আমি ঢাকা যা" (ami Dhaka za), which is erroneous. An auxiliary is expected after "যা" (za). After parsing using algorithm 8.1, the following output is found,

- $S \rightarrow BS A1$
- $BS \rightarrow NW E2 A2$
- $NW \rightarrow N E5$
- $N \rightarrow \text{আমি}$
- $E5 \rightarrow \epsilon$
- $E2 \rightarrow \epsilon$
- $A2 \rightarrow VP A4$
- $VP \rightarrow UNG E2 E1 D1$

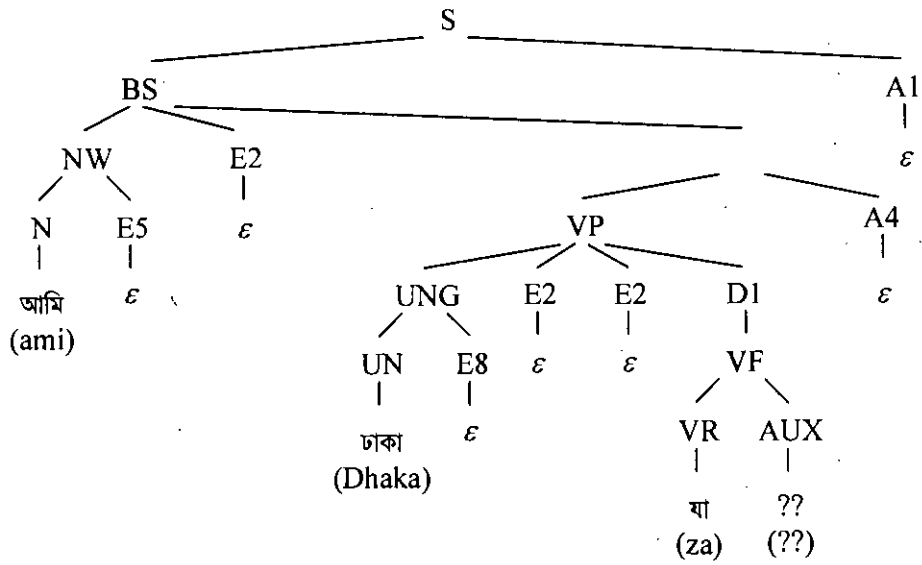


Figure 8.3: Tree derivation of erroneous sentence “আমি ঢাকা যা” (ami Dhaka za).

- $UNG \rightarrow UN E8$
- $UN \rightarrow \text{ঢাকা}$
- $E8 \rightarrow \epsilon$
- $E2 \rightarrow \epsilon$
- $E1 \rightarrow \epsilon$
- $D1 \rightarrow VF$
- $VF \rightarrow VR AUX$
- $VR \rightarrow \text{যা}$
- $AUX \rightarrow ??$
- $A4 \rightarrow \epsilon$

$$A1 \rightarrow \varepsilon$$

From the output, the production “AUX \rightarrow ??” indicates, algorithm detect an auxiliary is expected. The output can be represented as tree in figure 8.3.

Then, we can consider another erroneous sentence “ঐ টি ছেলে ভাত খায়” (OI Ti chhele vat khay) for parsing. Following output is found using the same algorithm,

$$S \rightarrow BS A1$$

$$BS \rightarrow PRE NW E2 A2$$

$$PRE \rightarrow DEMO E3$$

$$DEMO \rightarrow DD E7$$

$$DD \rightarrow ঐ$$

$$E7 \rightarrow \varepsilon$$

$$E3 \rightarrow \varepsilon$$

< symbol skipped: টি >

$$NW \rightarrow N E5$$

$$N \rightarrow ছেলে$$

$$E5 \rightarrow \varepsilon$$

$$E2 \rightarrow \varepsilon$$

$$A2 \rightarrow VP A4$$

$$VP \rightarrow D3$$

$$D3 \rightarrow D4$$

$$D4 \rightarrow NW E2 E1 D1$$

$$NW \rightarrow N E5$$

$$N \rightarrow ভাত$$

$$E5 \rightarrow \varepsilon$$

$$E2 \rightarrow \varepsilon$$

$$E1 \rightarrow \varepsilon$$

$$D1 \rightarrow VF$$

$$VF \rightarrow VR AUX$$

$$VR \rightarrow খা$$

AUX → আয়

A4 → ϵ

A1 → ϵ

In this example, an extra symbol “টি” (Ti) is found, which is skipped by recovery routine. This output can be presented like a tree as in figure 8.4.

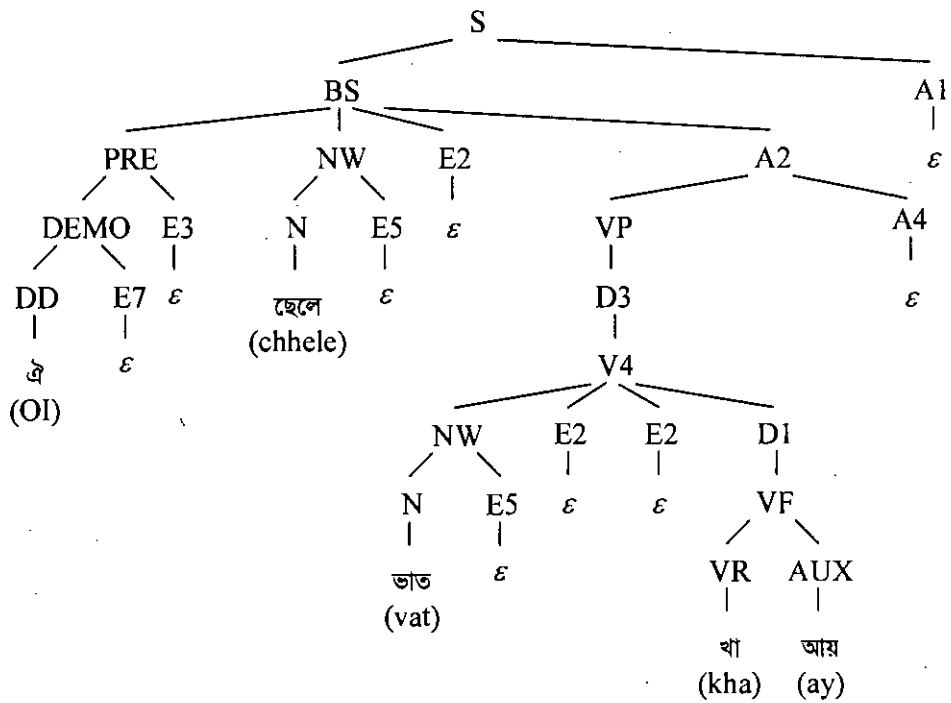


Figure 8.4: Tree derivation of erroneous sentence “এ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).

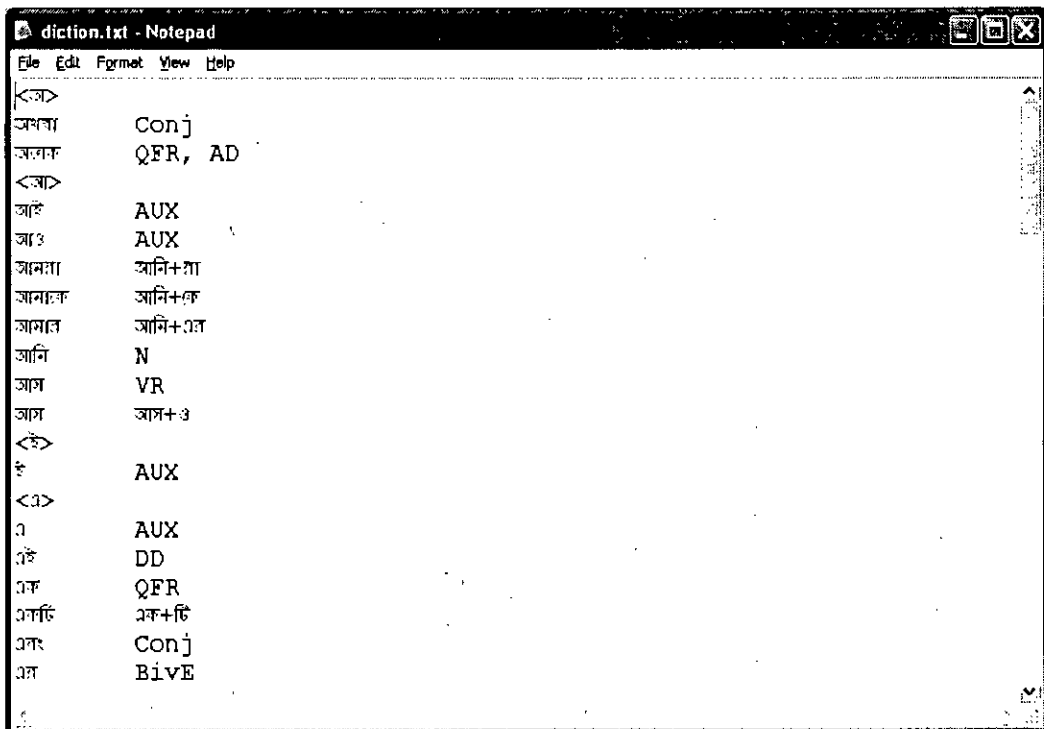
Therefore, error recovery mechanism allows a Bangla sentence to be parsed after occurring error, which was absent previous Bangla sentence parsing methodology. In a machine translation system such error recovery system is very fruitful, because it is very unlikely to assume all the sentences of source language are correct. Even not all correct sentences can be possible to fit in a grammar. In a grammar of a language, we only define most likely and common structure of sentences. It is an NP-complete problem to design a grammar to fit all possible types of sentences of source language.

8.9 Remarks

Error recovery mechanism proposed in algorithm 8.4, does not recover an error correctly all the time. Sometimes, it may skip a correct symbol, as it considers only one word at a time. A “global correction” mechanism could be more effective, but also takes exponential runtime. As machine translation software runs on a large volume of data, exponential runtime is very costly. In that perspective, our error recovery mechanism is efficient, and recovers error correctly most of the time. Most importantly, it allows a sentence to be parsed after the occurrence of an error.

9.1 Overview

In this chapter, we will discuss about architecture and functionality of our simulation program used to verify functionality and correctness of our proposed non-ambiguous predictive Bangla grammar. We will also discuss about parsing method and error recovery strategy of the simulation program.



Word	Tag
<অ>	
অথবা	Conj
অথক	QFR, AD
<আ>	
আই	AUX
আঃ	AUX
আনোনা	আনি+ণা
আনোনা	আনি+ক
আমার	আনি+ও
আনি	N
আগ	VR
আগ	আগ+ঃ
<ই>	
ই	AUX
<ঊ>	
ঊ	AUX
ঊই	DD
ঊক	QFR
ঊকটি	ঊক+টি
ঊক	Conj
ঊক	Bive

Figure 9.1: Lexicon used for simulation program.

9.2 The Lexicon

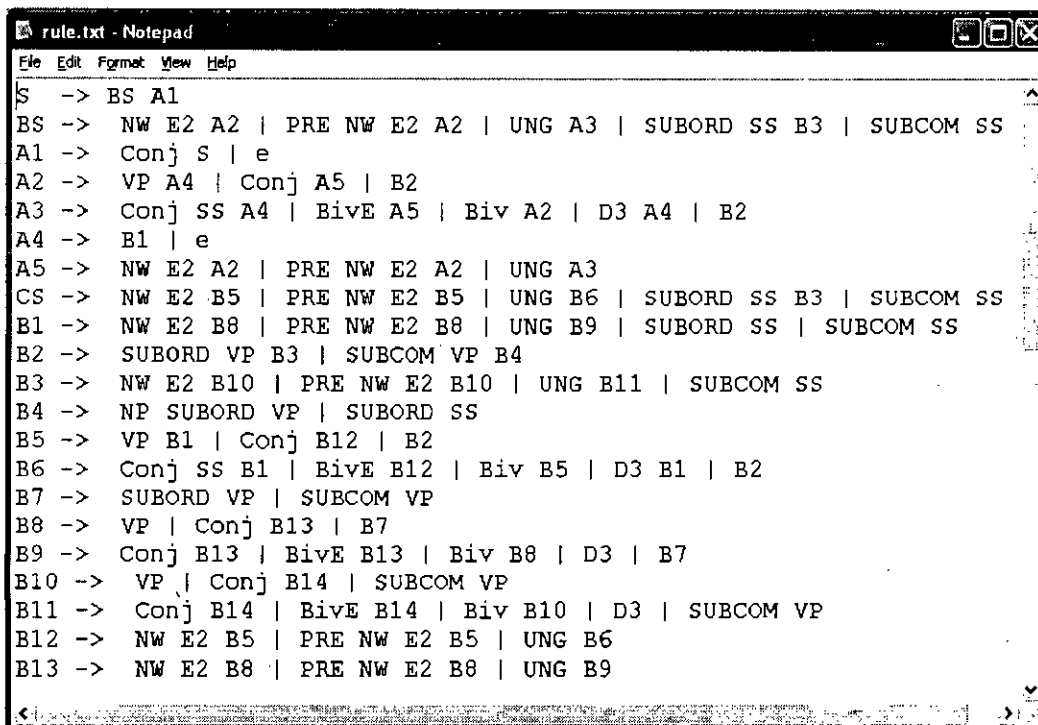
The simulation program uses a lexicon for “parts of speech tagging” purpose. The lexicon represents set of words in Bangla language and their corresponding terminal or tag. The lexicon may be stored as a text file or in a database. In commercial machine translation software, the lexicon represents a large volume of data.

Therefore, a database should be used. Though in our simulation program, we have stored the lexicon as a text file for simplicity purpose.

We have built the lexicon using the architecture proposed in section 8.2. So, each simple Bangla word is corresponding to one of fifteen terminals or tags. In case of complex words no terminal or tag is applicable. In that case, lexicon stores how complex words are built. Figure 9.1 shows the text file we have used as lexicon.

9.3 Simulation program

The functionality of our simulation program can be divided into different stages. First stage can be called as initialization stage, which is used to build the parsing table. After the parsing table is initialized, the input sentences are parsed one by one. Each sentence is firstly passed through lexical analyzer and then syntax analyzer. Then parsed text is found as output.



```

rule.txt - Notepad
File Edit Format View Help
S -> BS A1
BS -> NW E2 A2 | PRE NW E2 A2 | UNG A3 | SUBORD SS B3 | SUBCOM SS
A1 -> Conj S | e
A2 -> VP A4 | Conj A5 | B2
A3 -> Conj SS A4 | BivE A5 | Biv A2 | D3 A4 | B2
A4 -> B1 | e
A5 -> NW E2 A2 | PRE NW E2 A2 | UNG A3
CS -> NW E2 B5 | PRE NW E2 B5 | UNG B6 | SUBORD SS B3 | SUBCOM SS
B1 -> NW E2 B8 | PRE NW E2 B8 | UNG B9 | SUBORD SS | SUBCOM SS
B2 -> SUBORD VP B3 | SUBCOM VP B4
B3 -> NW E2 B10 | PRE NW E2 B10 | UNG B11 | SUBCOM SS
B4 -> NP SUBORD VP | SUBORD SS
B5 -> VP B1 | Conj B12 | B2
B6 -> Conj SS B1 | BivE B12 | Biv B5 | D3 B1 | B2
B7 -> SUBORD VP | SUBCOM VP
B8 -> VP | Conj B13 | B7
B9 -> Conj B13 | BivE B13 | Biv B8 | D3 | B7
B10 -> VP | Conj B14 | SUBCOM VP
B11 -> Conj B14 | BivE B14 | Biv B10 | D3 | SUBCOM VP
B12 -> NW E2 B5 | PRE NW E2 B5 | UNG B6
B13 -> NW E2 B8 | PRE NW E2 B8 | UNG B9

```

Figure 9.2: Grammar used for simulation program.

9.3.1 Initialization

When the simulation program starts, at first it goes through the initialization stage. It analyzes the grammar stored in a text file, as shown in figure 9.2 and builds the parsing table using the theory discussed in section 2.8.3. The program also puts error recovery information into the parsing table using the theory discussed in section 2.8.4. If we use non-ambiguous Bangla grammar, then the parsing table derived in section 8.3.3 will be generated. Input sentences will be parsed using the information of the parsing table. For erroneous sentences, error recovery will also be achieved using the information of the parsing table.

9.3.2 Lexical Analyzer

The functionality of the lexical analyzer is “parts of speech tagging”. For this purpose, it uses the lexicon. As tagging is possible only for simple words, the complex words are broken into simple words. Complex words are attempted to be broken firstly using the lexicon, then using an algorithm, and then heuristics are applied. Then the simple words are tagged using the lexicon. If any word is found in the lexicon, it is tagged as “unknown”. In brief, the lexical analyzer is implemented using the theory discussed in section 8.4.

9.3.3 Syntax Analyzer

The functionality of the syntax analyzer is to generate parsed text as output. Output parsed text is represented as a tree structure. The syntax analyzer uses top-down predictive parsing algorithm, as discussed in algorithm 8.1. The algorithm also adopts error recovery policy. When any error is found in the input sentence, the syntax analyzer does not stop parsing. Rather it reports the type of error (like extra word or missing word) and continues parsing.

9.4 Input of the Program

The simulation program takes sentences as input from a text file. Bangla input sentences are stored in UNICODE Bangla font. As a result, each Bangla letter is recognizable to the simulation program. Each UNICODE character takes 2 bytes of storage, contrasts with ASCII letter which takes 1 byte but does not support multi-lingual characters. First byte of UNICODE character represents character code and second byte represents language code. Code for Bangla language is 9. ASCII code has also equivalent representation in UNICODE format. In that case, language code is 0 and character code is same as ASCII code. UNICODE for Bangla characters is described in Appendix A.

Input file containing Bangla sentences to be parsed is shown in figure 9.3.

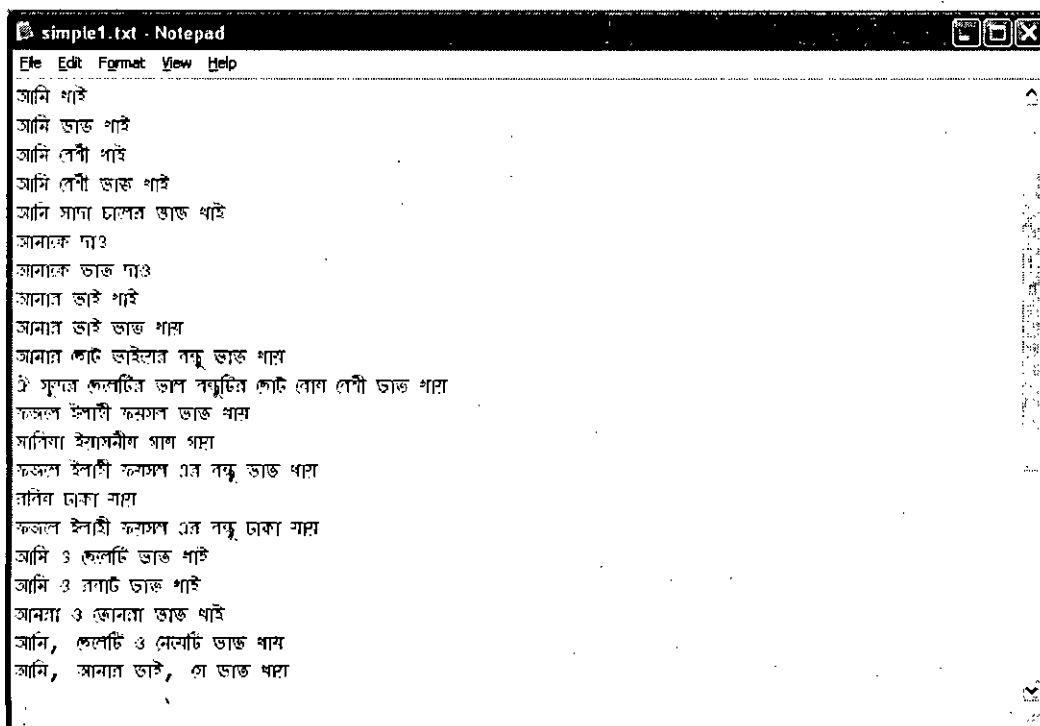
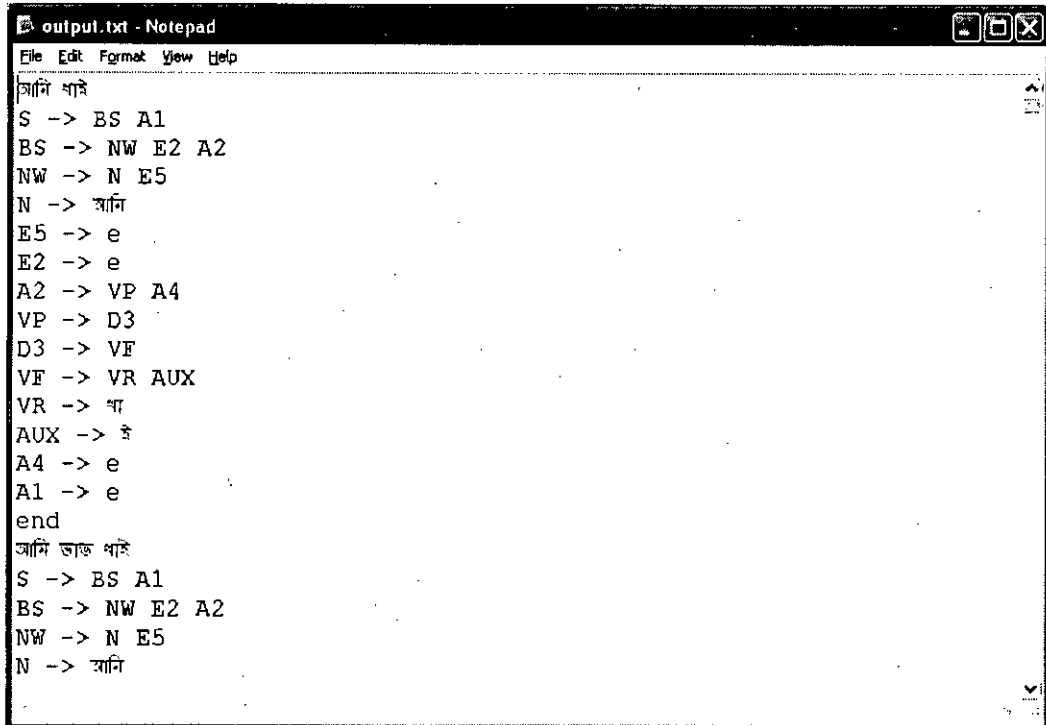


Figure 9.3: Input file used for simulation program.

9.5 Output of the Program

After parsing, the simulation program generates productions which represent the parsed text for an input sentence. Corresponding productions of a parsed text is stored in a text file, as shown in figure 9.4.



```

output.txt - Notepad
File Edit Format View Help
আমি খাই
S -> BS A1
BS -> NW E2 A2
NW -> N E5
N -> আমি
E5 -> e
E2 -> e
A2 -> VP A4
VP -> D3
D3 -> VF
VF -> VR AUX
VR -> খা
AUX -> ই
A4 -> e
A1 -> e
end
আমি ভাত খাই
S -> BS A1
BS -> NW E2 A2
NW -> N E5
N -> আমি

```

Figure 9.4: Output file showing parsed text by simulation program.

To visualize the parsed text output as text file, an another program is written in Java, which displays tree representation of each parsed text. Reason for choosing Java for display program is, Java is UNICODE supported language, and it is possible to show UNICODE Bangla words in a Java program. A tree representation of a parsed text using the Java based display program is shown in figure 9.5.

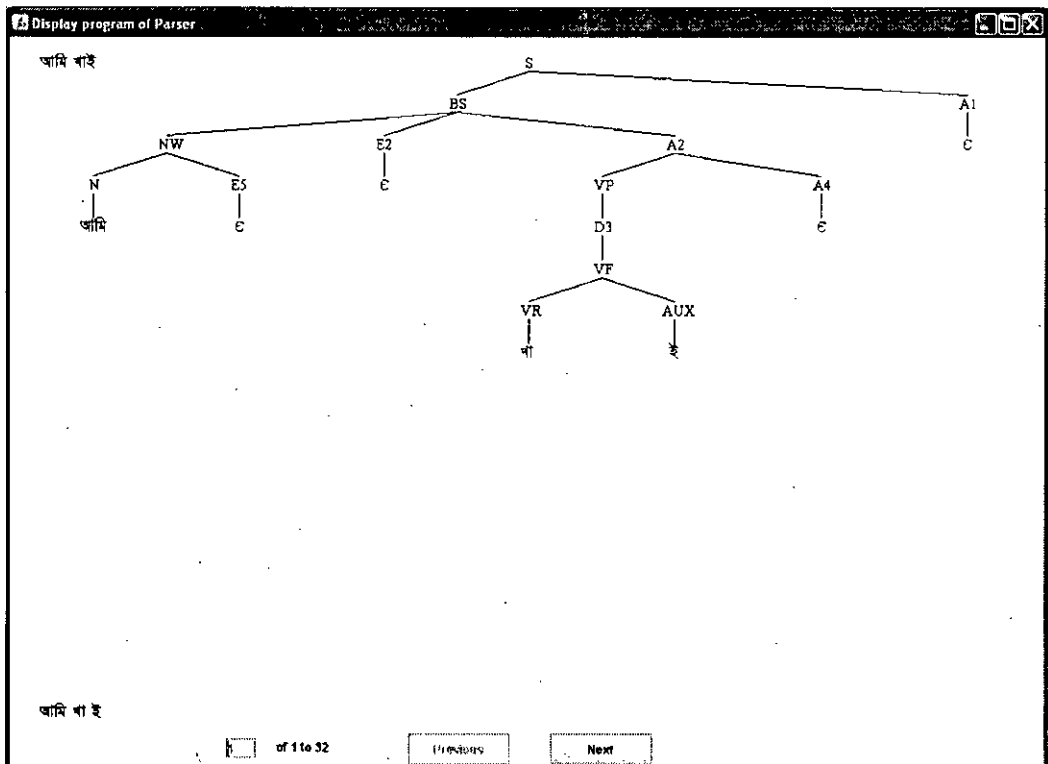


Figure 9.5: Output of display program using Java showing tree representation of parsed text.

9.6 Error Recovery

The parser program recovers from error if any error occurs in the input sentence and continues parsing. At the point of error, the parser reports the type of error.

For example, while parsing the erroneous sentence “আমি ঢাকা যা” (ami Dhaka za), the parser detects a missing auxiliary. Then the parser reports missing of auxiliary and continues parsing. Output productions are shown in figure 9.6, where missing auxiliary is reported using the production “AUX → ??”. Tree representation is shown in figure 9.7.

```

output.txt - Notepad
File Edit Format View Help
আমি ঢাকা যা
S -> BS A1
BS -> NW E2 A2
NW -> N E5
N -> আমি
E5 -> e
E2 -> e
A2 -> VP A4
VP -> UNG E2 E1 D1
UNG -> UN E8
UN -> ঢাকা
E8 -> e
E2 -> e
E1 -> e
D1 -> VF
VF -> VR AUX
VR -> যা
AUX -> ??
A4 -> e
A1 -> e
end
    
```

Figure 9.6: Output productions for the sentence “আমি ঢাকা যা” (ami Dhaka za).

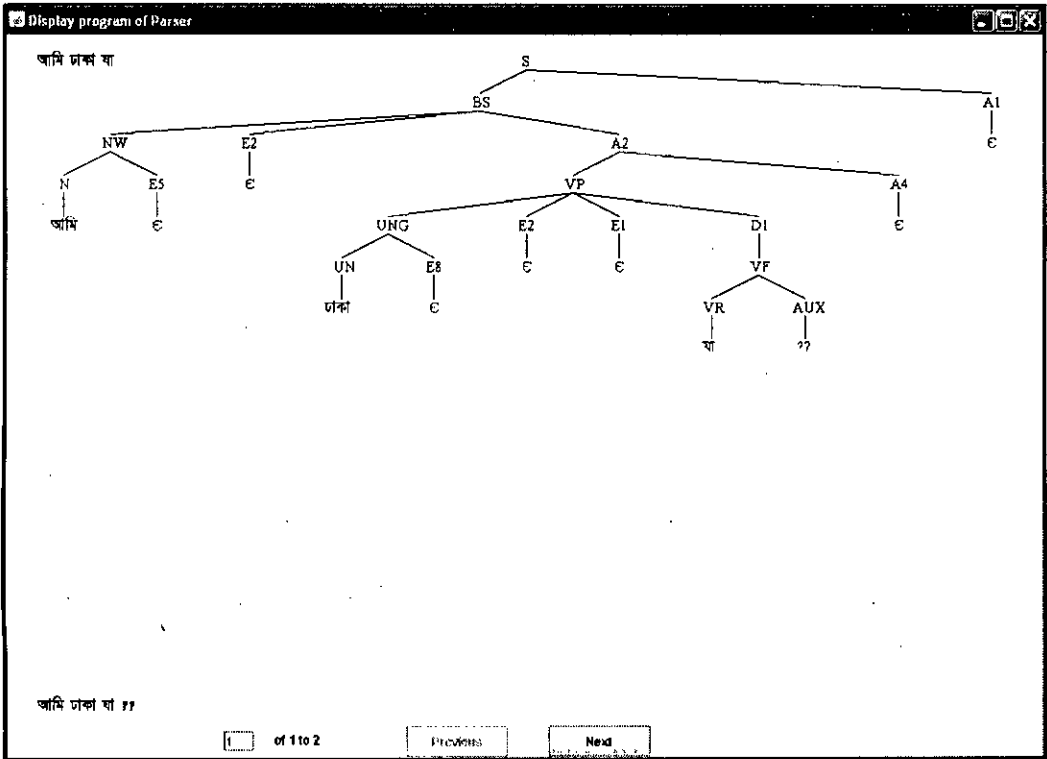


Figure 9.7: Tree representation for the sentence “আমি ঢাকা যা” (ami Dhaka za).

Again, while parsing the erroneous sentence “এ টি ছেলে ভাত খায়” (OI Ti chhele vat khay), the parser detects an extra word “টি” (Ti). Then the parser reports the detection of that extra word, skips it and continues parsing. Output productions are shown in figure 9.8, where detection and skipping of extra word is reported as “symbol skipped: টি”. Tree representation is shown in figure 9.9.

```

output.txt - Notepad
File Edit Format View Help
এ টি ছেলে ভাত খায়
S -> BS A1
BS -> PRE NW E2 A2
PRE -> DEMO E3
DEMO -> DD E7
DD -> এ
E7 -> e
E3 -> e
< symbol skipped: টি >
NW -> N E5
N -> ছেলে
E5 -> e
E2 -> e
A2 -> VP A4
VP -> D3
D3 -> D4
D4 -> NW E2 E1 D1
NW -> N E5
N -> ভাত
E5 -> e
E2 -> e
E1 -> e
D1 -> VF
VF -> VR AUX
VR -> খা
AUX -> খায়
A4 -> e
A1 -> e
end

```

Figure 9.8: Output productions for the sentence “এ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).

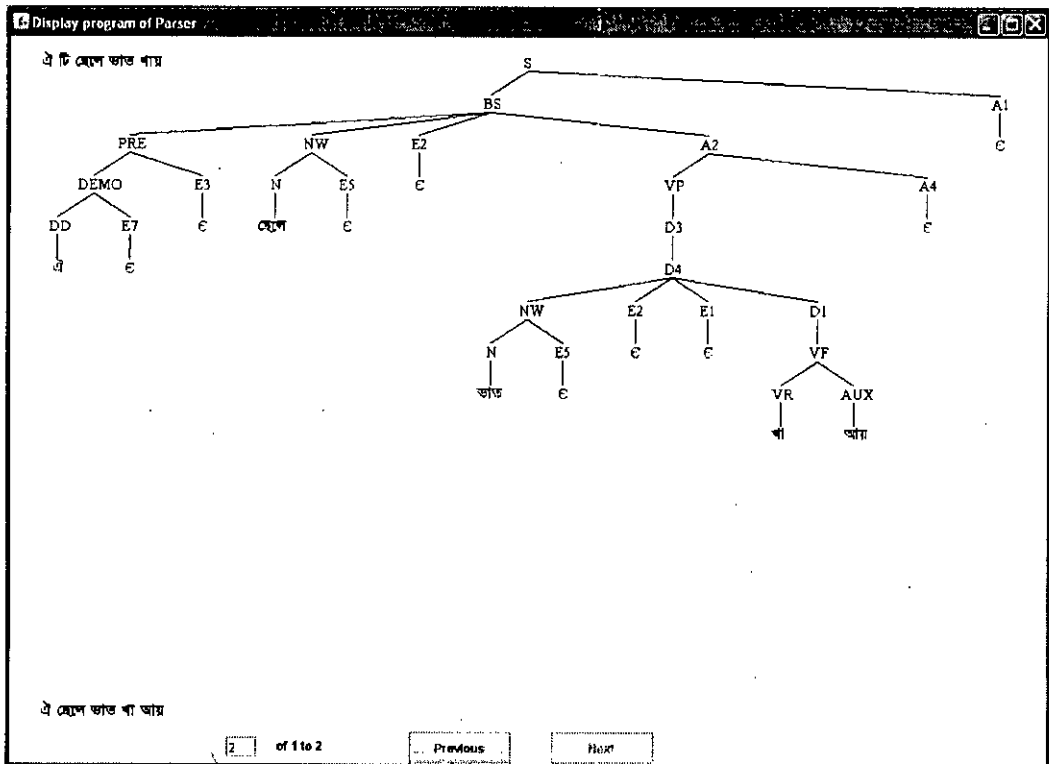


Figure 9.9: Tree representation for the sentence “এ টি ছেলে ভাত খায়” (OI Ti chhele vat khay).

9.7 Remarks

In this chapter, we have discussed about the simulation program, where non-ambiguous Bangla grammar, predictive parsing algorithm and error recovery mechanism is simulated by several Bangla input sentences. Therefore, justification of non-ambiguous predictive Bangla grammar and error recovery mechanism is analyzed.

10.1 Achievements of This Thesis

Machine translation is a very important and popular field in computer science over the world because of its social, commercial and scientific applications. Whether human translation is very slow, machine translation enables swift translation of literary works, product catalogs, official documents, scientific works from one language to another language. Machine translation can play a vital role in international business, like swift translation of product manuals and catalogs to language of target market and then fulfilling the market demands. Though quality of translation may not be up to the mark, specially for the case of literary works, quality may be much denigrated. But manual translators can take the output of machine translation system as ground work, and can enhance the quality of translation by re-ordering and using more appropriate words. Raw translation using machine translation system makes the task of translation much easier for a manual translator other than translating from the root.

There are not so many works in the field of machine translation in Bangla. There are some works in translating from English to Bangla, like ANIRBAN. But in translating from Bangla to other languages, there is no commercial machine translation software. This is because, there are very few research works in the field of machine translation using Bangla as source language. It has been analyzed and justified that syntax based machine translation technique is one of the most suitable technique for machine translation in Bangla. Syntax based machine translation is composed of two stages – parsing and translating. Of the two stages, parsing is most challenging, as translation is directed by the parsed text.

Parsing methodology in Bangla was developed in 2005 [22][23]. But the grammar for parsing was ambiguous and did not support predictive parsing. Furthermore,

error recovery mechanism was absent in that work. This thesis has overcome such limitations and gained following achievements.

1. **Non-ambiguous grammar:** In this thesis, we have designed non-ambiguous grammar for Bangla natural language sentences. Non-ambiguity of grammar is very important, as it ensures at most one entry in each grid position of parsing table, whereas ambiguous grammar allows several entries in some grid positions of parsing table, not making the parsing table fully decisive.
2. **Predictive parser:** In this thesis, we have designed non-ambiguous Bangla grammar for predictive parser. Predictive parsing is very important, as it allows parsing in linear time. Previous grammar was not applicable for predictive parsing, resulting the parser to analyze all possible combinations of rules to match an input sentence and taking exponential run time. Generally, machine translation software runs on a large volume of data. Therefore, exponential run time is very costly and impractical. In this thesis, we have designed grammar for predictive parser. When such grammar is applied on a parser, it runs in linear time. As a result, it meets the demand of commercial machine translation software.
3. **Non-dictionary word handling:** In this thesis, we have designed an efficient and effective mechanism to handle non-dictionary words in an input sentence. Previous Bangla parsing methodology can not parse a sentence containing non-dictionary words and parsing halts when a non-dictionary word occurs in input sentence. Non-dictionary word handling is very important, as non-dictionary word occurs frequently in Bangla sentences, like any other natural language. Non-dictionary word handling mechanism designed in this thesis has been proved very effective and increased accuracy of parsing greatly.
4. **Enhancement of Bangla grammar:** In this thesis, we have enhanced previous Bangla grammar, which has increased the accuracy of parsing. We

have added rule in Bangla grammar to allow use of conjunctive in noun phrase, use of more than two independent sentences in a compound sentence and use of numeric word in input sentence.

5. **Error recovery mechanism:** In this thesis, we have designed an error recovery mechanism in Bangla parsing. In previous works of Bangla parsing, the idea of error recovery was absent. A parser having no error recovery mechanism stops parsing when an error occurs in input sentence, whereas a parser having error recovery mechanism can continue parsing when an error occurs skipping the error. Error recovery mechanism in this thesis has improved the accuracy of parsing. In case of previous works, number of unparsed sentence was high, as probability or error in natural language is high. This is because, not all the correct sentences can be fitted in grammar. Allowing all possible combinations of sentences grows the size of grammar exponentially. Therefore, we allow only most frequent and common structures in grammar. To recover from error, we have applied some heuristics. Use of heuristic does not ensure correct recovery in all cases, though in most of the cases recovery is correct. But most importantly, error does not stall the task of parsing and parsing continues after occurrence of error.
6. **Lexicon structure:** In this thesis, we have designed lexicon structure for Bangla natural language. The structure is very similar to Bangla dictionary structure. Therefore, the lexicon can easily be built from Bangla dictionary. Moreover, the structure of lexicon is very efficient for the functionality of lexical analysis.
7. **Lexical analysis:** In this thesis, we have designed a lexical analyzer for Bangla parser. Though, the main purpose of lexical analyzer is parts-of-speech tagging, we have brought out some new and efficient ideas. We have showed an idea of part-of-speech tagging for complex words by breaking

through lexicon or algorithm or heuristic. We also have showed idea of handling of simple words having multiple parts-of-speech.

10.2 Future Works

Research works are not discrete, rather continuation of works. Such as, we have enhanced previous Bangla parsing methodology and added many features. There are also lots of research scopes in direction of Bangla parsing methodology. We have mentioned some scopes of future works in this area as follows,

- Use of negative sentences in Bangla predictive parser with error recovery capability. For example, predictive parsing of “আমি ঢাকা যাব না” (ami Dhaka zabo na) is not possible.
- Use of interrogative sentences in Bangla predictive parser with error recovery capability. For example, predictive parsing of “তুমি কেন ঢাকা যাবে না?” (tumi keno Dhaka zabe na?) is not possible.
- Use of two consecutive verbs in verb phrase in Bangla predictive parser with error recovery capability. For example, “আমি চলে এসেছি” (ami chole esechi) is not possible to parse using our proposed Bangla grammar.
- Bangla grammar can be designed in paragraph level, whether our proposed Bangla grammar is designed in sentence level.
- Learning method may be included when an unknown word or a group of unknown word occurs in a sentence.
- Error recovery method may be enhanced using more heuristics.
- Consideration of idioms in Bangla grammar.

Unicode for Bangla Characters

UNICODE stands for universal character code. Bangla characters also included into UNICODE. Table A.1 shows UNICODE for most common Bangla characters.

Table A.1: Code in UNICODE for Bangla characters.

Bangla Character	Code in UNICODE	
	1st byte	2nd byte
ঐ	133	9
ঐা	134	9
ঐা	135	9
ঐা	136	9
ঐা	137	9
ঐা	138	9
ঐা	139	9
ঐা	143	9
ঐা	144	9
ঐা	147	9
ঐা	148	9
ঐা	149	9
ঐা	150	9
ঐা	151	9
ঐা	152	9
ঐা	154	9
ঐা	155	9
ঐা	156	9
ঐা	157	9
ঐা	158	9
ঐা	159	9
ঐা	160	9
ঐা	161	9
ঐা	162	9
ঐা	163	9
ঐা	164	9
ঐা	165	9
ঐা	166	9
ঐা	167	9
ঐা	168	9

প	170	9
ফ	171	9
ব	172	9
ভ	173	9
ম	174	9
য	175	9
র	176	9
ল	178	9
শ	182	9
ষ	183	9
স	184	9
হ	185	9
আ কার	190	9
ঈ কার	191	9
ঐ কার	192	9
ঊ কার	193	9
ঋ কার	194	9
এ কার	199	9
ঐ কার	200	9
ও কার	203	9
ঔ কার	204	9
ং	206	9
ড়	220	9
ঢ	221	9
য়	223	9
০	230	9
১	231	9
২	232	9
৩	233	9
৪	234	9
৫	235	9
৬	236	9
৭	237	9
৮	238	9
৯	239	9

Bibliography

- [1] T. McEney, and A. Wilson, *Corpus Linguistics*, 2nd Edition, Edinburgh University Press, 2001.
- [2] U. Muegge, "An Excellent Application for Crummy Machine Translation: Automatic Translation of a Large Database", in *Proceedings of the Annual Conference of the German Society of Technical Communicators*, (Stuttgart, Germany), pp. 18-21, 2006.
- [3] P. Lenssen, "Google Translator: The Universal Language", 2005.
- [4] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase based translation", in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, (Edmonton, Canada), vol. 1, pp. 48-54, 2003.
- [5] D. Chiang, "A Hierarchical Phrase-Based Model for Statistical Machine Translation", in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL) 2005*, (Michigan, USA), pp. 263-270, 2005.
- [6] M. Nagao, "A framework of a mechanical translation between Japanese and English by analogy principle", in *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, (Lyon, France), pp. 173-180, 1984.
- [7] J. Hutchins, "Milestones in Machine Translation – No.6: Bar-Hillel and the Nonfeasibility of FAHQT", *International Journal of Language and Documentation*, no.1, pp. 20-21, 1999.
- [8] Y. Bar-Hillel, "The Present Status of Automatic Translation of Languages", *Advances in Computers*, vol. 1, pp. 91-163, 1960.
- [9] F. Och, "The Machines do the Translating", 2005.
- [10] D. Geer, "Statistical Translation Gains Respect", *IEEE Computer*, pp. 18-21, 2005.
- [11] E. Ratcliff, "Me Translate Pretty One Day", *WIRED Magazine*, Issue 14.12, Dec. 2006.
- [12] "NIST 2006 Machine Translation Evaluation Official Results", Nov. 2006.
- [13] "Machine Translation Controlled Language Translation Standards", available at <http://muegge.cc> (last access on Oct. 2008).
- [14] "In-Q-Tel", available at <http://www.iqt.org> (last access on Oct. 2008).
- [15] W. Jackson, "Air Force wants to build a Universal Translator", *Government Computer News (GCN)*, 2003.
- [16] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd Edition, Addison Wesley, 2007.
- [17] N. Chomsky, "Three Models for the Description of Language", *IRE Transactions on Information Theory*, vol. 2, no. 2, pp. 113-124, Sep. 1956.
- [18] "The Penn Treebank Project", available at <http://www.cis.upenn.edu/~treebank> (last access on Oct. 2008).
- [19] P. Adriaans, H. Fernau, and M. V. Zaanen, *Grammatical Inference: Algorithms and Applications*, 1st Edition, Springer, 2002.
- [20] C. D. Manning, and H. Schuetze, *Foundations of Statistical Natural Language Processing*, 1st Edition, MIT Press, 1999.

- [21] "Anubadok Online: The Bengali Machine Translator", available at <http://bengalinux.sourceforge.net/cgi-bin/anubadok/index.pl> (last access on Oct. 2008).
- [22] M. M. Hoque, and M. M. Ali, "A Parsing Methodology for Bangla Natural Language Sentences", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 2005*, (Dhaka, Bangladesh), pp. 277-282, 2003.
- [23] M. M. Hoque, "A Parsing Methodology for Bangla Natural Language Sentences", Master's thesis, Department of CSE, BUET, 2005.
- [24] M. M. Murshed, "Parsing of Bangla Natural Language Sentences", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 1998*, (Dhaka, Bangladesh), pp. 185-189, 1998.
- [25] M. R. Selim, and M. Z. Ikbal, "Syntax Analysis of Phrases and Different Types of Sentences in Bangla", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 1999*, (Sylhet, Bangladesh), pp. 175-186, 1999.
- [26] M. M. Ali, and M. M. Ali, "Development of Machine Translation Dictionaries for Bangla Language", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 2002*, (Dhaka, Bangladesh), pp. 267-271, 2002.
- [27] L. Mehedy, S. M. Arefin, and M. Kaykobad, "Bangla Syntax Analysis: A Comprehensive Approach", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 2003*, (Dhaka, Bangladesh), pp. 287-293, 2003.
- [28] M. M. Asaduzzaman, and M. M. Ali, "A Knowledge Based Approach to Bangla-English Machine Translation for Simple Assertive Sentences", *International Journal of Translation*, vol. 15, no. 2, pp. 77-97, 2003.
- [29] K. J. Alam, and M. A. Rahman, A. K. M. M. Haque, T. Islam, and M. T. Irfan, "Structure-based Bangla to English Machine Translation", in *Proceedings of the International Conference on Computer and Information Technology (ICCIT) 2005*, (Dhaka, Bangladesh), 2005.
- [30] M. Z. H. Sarkar, S. Rahman, and M. A. Mottalib, "Bottom-up Parsing Algorithms for Bengali Parser Applying Context-sensitive Transformation Rules to Maintain the Freeness of Word Order", in *Proceedings of the National Conference on Computer Processing of Bangla (NCCPB) 2005*, (Dhaka, Bangladesh), 2005.
- [31] M. Z. H. Sarkar, S. Rahman, and M. A. Mottalib, "Parsing Algorithms for Bengali Parser to Handle Affirmative Sentences", *Asian Journal of Information Technology (AJIT)*, vol. 5, no. 5, pp. 504-511, 2006.
- [32] E. T. Irons, "A Syntax Directed Compiler for ALGOL 60", *Communications of the ACM*, vol. 4, no. 1, pp. 51-55, Jan. 1961.
- [33] P. M. Lewis, and R. E. Stearns, "Syntax-directed Transduction", *Journal of the ACM (JACM)*, vol. 15, no. 3, pp. 465-488, Jul. 1968.
- [34] A. V. Aho, and J. D. Ullman, *The Theory of Parsing, Translation and Compiling (Volume 1: Parsing)*, Prentice Hall, 1972.
- [35] P. O. Hearn, and R. Tennent, *Algol-like Languages*, 1st Edition, Birkhauser Boston, 1996.

- [36] A. Nijholt, *Context-free Grammars: Covers, Normal Forms, and Parsing*, 1st Edition, Springer, 1980.
- [37] J. Levin, *A Madman Dreams of Turing Machines*, Anchor, 2007.
- [38] M. Ganapathi, and C. N. Fischer, "Affix Grammar Driven Code Generation", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 7, no. 4, pp. 560-599, Oct. 1985.
- [39] H. Alblas, and B. Melichar, *Attribute Grammar, Applications and Systems*, 1st Edition, Springer, 1991.
- [40] B. Keller, and D. Weir, "A Tractable Extension of Linear Indexed Grammars", in *Proceedings of the Seventh Conference on European Chapter of the Association for Computational Linguistics*, (Dublin, Ireland), pp. 75-82, 1995.
- [41] S. Shieber, "Evidence Against the Context-freeness of Natural Language", *Linguistics and Philosophy*: 8, pp. 333-345, 1985.
- [42] G. K. Pullum, and G. Gazdar, "Natural Languages and Context-free Languages", *Linguistics and Philosophy*: 4, pp. 471-504, 1982.
- [43] C. Culy, "The Complexity of the Vocabulary of Bambara", *Linguistics and Philosophy*: 8, pp. 345-351, 1985.
- [44] R. Frost, and R. Hafiz, "A New Top-Down Parsing Algorithm to Accommodate Ambiguity and Left Recursion in Polynomial Time", *ACM SIGPLAN Notices*, vol. 41, no. 5, pp. 46 - 54, 2006.
- [45] R. Frost, R. Hafiz, and P. Callaghan "Modular and Efficient Top-down Parsing for Ambiguous Left-recursive Grammars", in *Proceedings of the 10th International Workshop on Parsing Technologies (IWPT)*, (Prague, Czech Republic), pp. 109-120, Jun. 2007.
- [46] R. Frost, R. Hafiz, and P. Callaghan, "Parser Combinators for Ambiguous Left-recursive Grammars", in *Proceedings of the 10th International Symposium on Practical Aspects of Declarative Languages (PADL)*, (San Francisco, USA), pp. 167-181, Jan. 2008.
- [47] M. Sipser, *Introduction to the Theory of Computation*, 2nd Edition, Course Technology, 2005.
- [48] P. Norvig, "Techniques for Automatic Memoization with Applications to Context-free Parsing", *Journal of Computational Linguistics*, vol. 17, no. 1, pp. 91-98, Mar. 1991.
- [49] M. Tomita, *Efficient Parsing for Natural Language*, 1st Edition, Springer, 1985.
- [50] D. Grune, and C. J. H. Jacobs, *Parsing Techniques: A practical Guide*, 2nd Edition, Springer, 2007.

