7.1 Introduction

The chapter discusses the design and development of the parser module of the prototype translation system. It clearly describes the working of the parser algorithm with sample inputs. The chapter also discusses the set of part of speech tags, chunk tags and the chunk dependencies which were derived for the parser. The selected syntactic structure transfer rules required in the translation of Malayalam sentences to the corresponding English sentence are also discussed in this chapter.

7.2 Selection of the parser model

Of the different models available for parsing natural language sentences the bottom up parsers generate subtrees that have no hope of leading to the start state. But a top down approach never wastes time exploring trees that cannot result from the start state. The algorithms based on dynamic programming approaches like Earley parsing and CYK parsing algorithms overcome all problems of standard top down and bottom up parsing. Also there are parsers based on finite state cascades which improves the speed of the search process [37]. A top down approach has been chosen as the base model for the translation system considering many aspects. Frost, Hafiz and Callaghan [115] have proved that the time complexity of a top down parser to accommodate any form of CFG to produce all parses is polynomial time $\Theta(n^4)$ for left-recursive grammars and $\Theta(n^3)$ for non left-recursive grammars where $n$ is the
number of input symbols. The top down parsers may fall into infinite loop for left recursive grammars. So the grammar rules are to be non left recursive. The derived grammar rules for Malayalam for the parser were non left recursive. The prototype system uses a small dictionary which further reduces the problems of multiple traversals in the case of standard top down approach.

7.3 Formation of rules used by the parser

It was decided to include the word sense disambiguation and the syntactic structure from source parse tree to target parse tree into the parser module. For this the parser uses two sets of rules: i) the syntax rules of source language for source parse tree creation ii) the syntactic structure transfer rules for target parse tree creation. The syntax of the source language was arrived at by the following steps i) selection of appropriate primitive tagset ii) selection of chunk tagset iii) identification of hierarchical dependencies between chunks.

7.3.1 Selection of POS Tags

First step in deriving the syntactic structure of Malayalam sentences was the identification of set of word categories in a Malayalam sentence called part of speech tags. Since we found that a morpheme based parsing was appropriate for a highly agglutinative language like Malayalam we decided to give a unique tag mane for each morpheme category. The inflectonal and derivational suffixes were given separate tag names. The set of tags identified for our problem are listed in Table 7.1.

I. N - Nouns

Nouns are words used as the name of a person, place or thing. Nouns are
further classified into a) common nouns b) proper nouns c) collective nouns d) abstract nouns.

a) Common nouns - Common nouns are names given in common to every person.


(kutti/child), (maram/tree)

b) Proper nouns - Proper nouns is the name of some particular place or person.

(sita/sita), (delhi/Delhi)

c) Collective nouns - Collective noun is the name of a number of persons or things taken together and spoken of as one whole.

(janam/people), (kutumbam/family)

d) Abstract nouns – Abstract nouns is the name of a quality, action or state considered apart from the object to which it belongs.

Quality- (krooratha/cruelty), (dhairyam/bravery)

Action – (chiri/smile), (karachchil/cry)

State – (kuttikkalam/childhood)

Abstract nouns are usually formed from adjectives, verbs and common nouns with proper modifications to root.

(kuttikkalam/childhood) formed from noun (kutt/child)
Another classification for nouns are a) countable nouns b) uncountable nouns

a) Countable nouns

Countable nouns are names of objects, people etc. that we can count.

(\textit{kara} / verb root for cry)

\textit{pEna} (pen), \textit{kuta} (umbrella)

b) Uncountable nouns

Uncountable nouns are names of things which we can not count.

\textit{paalz} (milk), \textit{eNNa} (oil)

Nouns have gender, number and case information attached to them. The Penn tag NN has been adopted as such for common nouns. Penn tagset makes a distinction between noun singular and noun plural. As mentioned earlier, distinct tags based on grammatical information were avoided. This reduces the number of tags and hence helps achieve simplicity and consistency. The same tag \textit{N} had been used for all kinds of nouns. The tagset derived at IIIT for Hindi language includes a separate tag for proper nouns. Such is not required in languages like English as the Proper nouns in the text are marked by capital letters. This is not the case with many of the Indian languages as:

a) Indian languages, unlike English, do not have any specific marker for proper nouns in orthographic conventions. English proper nouns begin with a capital letter which distinguishes them from common nouns.
b) all proper nouns in Indian languages are otherwise used as common nouns containing a lexical meaning. For example, the names വിശാലം / viSaalam, വെന്ന് / vEnu etc. can occur as proper nouns and also common noun. This poses a problem during translation. The common nouns should be translated to their equivalent target language words whereas the proper nouns should be used as such.

1. രാജുവിന്റെ മനസ്സ്‌വിശാലം ആണ്

   raajuvininte manassu (viSaalam) aanu

   Correct translation: Raju’s mind is (broad)

2. വിശാലം നല്ല കുട്ടിയാണ്

   (viSaalam) nallakuttiaanu

   Correct translation: (ViSaalam) is a good student.

In sentence 1, the word വിശാലം viSaalam occurs as a common noun and needs translation. In sentence 2, the same word occurs as proper noun and should not be translated. The identification of proper nouns can be better achieved by named entity filters. We have not used a separate tag for proper nouns as we have assumed that the proper nouns are distinct from common nouns. IIIT tag set includes a separate tag for proper nouns for manual annotation and ignores it for machine learning algorithms. This tag is also similar to the Penn tagset and IIIT tag set uses NNP as the tag for proper nouns.

2. V Verb Finite

V is used to mark a verb root. A verbal construction such as the following is finite:

കുട്ടികള് വീട്ടില് കളി-ക്കുന്നു (kuttikaL veetttil kaLi-kkunnu)
children are playing in the house

Here the verb root kali is given the tag verb.

Marking the finiteness or non-finiteness as in IIT tag set was unnecessary for our problem. All verb roots are tagged as V whether it is part of a finite verb or non finite verb.

3. VA Verb Auxiliary

All auxiliary verbs which marks inflections for verb roots due to tense, aspect and mood will be marked as VA. VAUX is the tag which has been adopted in Penn tag set and IIT tagset.

‌

4. VNN Gerund suffix

A separate tag, VNN, for gerunds has been kept as functionally they are more like nouns and take on the nominal postpositions. This distinction is made in order to preserve the information that this word is a form of a verb. Every verb is capable of taking its own arguments in a sentence, even when it occurs in a nominalised form. The suffix അത്/athu forms the gerund suffix.

a) എനിക്ക്‌നീന്തുന്നത്‌ഇഷ്ടമാണ്‌(enikku neenthunnathu ishtamaanu)

I like swimming

ഇഷ്ടമാണ്‌(neenthunnathu/swimming) functions as a noun in the sentence.

b) എനിക്ക്‌പഴം‌കഴിക്കുന്നത്‌ഇഷ്ടമാണ്‌(I like eating banana)

enikku pazham kazhikkunnathu ishtamaanu
The gerund takes an argument in the above sentence. Noun പഴം / pazham is an object of the verb കഴിക്കുന്നത് / kazhikkunnathu and has no relation to the main verb. Therefore, in order to be able to show the exact verb-argument structure in the sentence, it is essential that this crucial information of a noun derived from a verb is preserved. The verbs having അവന് / avan and അവള് / aval as suffix are also marked as VNN.

5. **INFA verb infinitive suffix**

This tag is to mark the infinitival verb form. Infinitive form of the verbs ends with aan. In Hindi the gerunds and infinitives end with –na. Since in Hindi both behave functionally in a similar manner, the distinction is not very clear.

ആന് (Ramu went to hit Raju)

ramerajuvine atikk – aan- pOyi

Here verb root കഴിക്ക് / atikk suffixed by ആന് / aan forms the infinitive.

6. **PA Adjective**

A word used with a noun to describe the person, animal, place or thing which the noun names or to tell the number or quantity is called an adjective. Adjectives are of following types:

i) **adjectives of quality**: this shows quality of a person or thing.

ദൽഹി വലിയ നഗരം (delhi valiya nagaramaNz)

Delhi is a big city

Here വലിയ (valiya / big) is the adjective of quality.

ii) **adjective of quantity**: This shows how much of a thing is meant.
Ramu had some food

In this sentence കുറച്ച് (kuRachchu / some) is the adjective of quantity.

iii) *Number adjectives*: this shows how many persons, things etc. are meant.

ഖരാഞ്ച ആഹാരം കഴിച്ച്

_kaiyil –anchu- viralukaluntu._

The word ആച്ച അന്വ is the number adjective here.

iv) *Demonstrative adjectives*: point out which person or thing is meant. They are ആ / aa and ഏ / ee.

_ഈ (ee) saari puthiyathaanu_

This sari is new

Here ഏ / ee is the demonstrative adjective which qualifies സാരി / saari.

In Malayalam the words for demonstratives are different from words for pronouns. Some languages like Hindi uses the same word as demonstratives and pronouns.

Hindi: _vaha ladaK merA bhAI hE_(vaha is used as demonstrative)

Hindi: _vaha merA bhAI hE_ (vaha is used as pronoun)

It was decided to use the same tag PA to all these kinds of adjectives. This tag is named as JJ in Penn tagset and IIIT tagset. Penn tagset also makes a
distinction between comparative and superlative adjectives. This has not been considered for our problem.

7. PAV Adverb

A word that modifies a verb, an adjective or another adverb is called an adverb. Adverbs are further classified into:

a) Adverbs of time: This shows when the event has happened.

\[ \text{My leg was hurt yesterday} \]

\[ \text{(innale/yesterday)} \]

b) Adverbs of frequency: This shows how often the event happens.

\[ \text{The child cries even now} \]

\[ \text{(eppOzhum/always)} \]

c) Adverbs of place: It shows where it has happened.

\[ \text{Raju came here} \]

\[ \text{(ivite/here)} \]

d) Adverbs of manner: This shows how the action is done.
He slept nicely

He slept nicely (nannaayi/nicely) is the adverb of manner.

e) Adverbs of affirmation

Seetha will definitely come (Seetha will definitely come)

seetha – theerchchayaayum – varum.

Theerchayaayum (definitely) is the adverb of affirmation.

All these classes of adverbs are given the tag PAV. This is RB tag of Penn tagset and IIIT tagset. Penn tagset also makes a difference between comparative and superlative adverbs, which is not adopted in our case. This is in accordance with our philosophy of coarseness in linguistic analysis.

8. NA Postposition

All Indian languages have the phenomenon of postpositions. Some languages like Hindi separate the post positions from the noun. All postpositional suffixes are given the tag NA. All case markers are tagged as PSP by IIIT tagset.

Raju gave the cash to Madhu

Here the case marker inu (inu/to) is attached to the noun preceding it.

9. C Conjunctions (co-ordinating and subordinating)
The tag C will be used for both co-ordinating and subordinating conjuncts. The Penn tagset has used IN tag for prepositions and subordinating conjuncts. The rationale behind this is that subordinating conjuncts and prepositions can be distinguished because subordinating conjuncts are followed by a clause and a prepositions by a noun phrase. All connectors other than prepositions will be marked as C. The conjuncts are ഉം / um and ഔ ല് / loo.

( Raaman and Madhu went to shop)

raaman-um-madhuvum katayil pOyi

ൂ / um is the conjunct used here.

10. **QW Question Words**

The Penn tagset makes a distinction between various uses of 'wh-' words and marks them accordingly (WDT, WRB, WP, WQ etc). The 'wh' words in English can act as questions, as relative pronouns and as well as determiners. However, for Indian languages we need not keep this distinction. Therefore, we tag the question words as QW.

( aaraaNu  madhuvine atichchathu?)

Who hit madhu?

( aaraaNu/who) is the question word.

11. **NEG Negative**

The words under this tag are ഇല്ല / illa, ആല്ല / anta, ആല്ല / alla etc.

In Hindi Negatives like 'nahIM', 'na', etc. will be marked as NEG. Penn tagset does have a separate tag for this.
Chapter - 7

116

(Raju did not come)

Here (illa/did not) is the word in this category.

(Raju should not go to shop)

Here (antal/should not) following (poku) is the NEG.

12. PL Plural marker

The plural markers for nouns we have considered are kaL and maar.

(kutti-kaL/children), (naari-maar/ladies)

13. NCA Noun clause marker

The noun clause marker considered are athu/ and ennzu.

Seetha told that Raman had gone

Here (Raman poyi ennzu) is the noun clause which is marked by the suffix (ennzu/that).

I liked what Ramu told
Here രാമു പരഞ്ഞാത്ത് *raamu paranjnathu* is the noun clause which is marked by the suffix അടു (athu/what).

14. ADJA Adjectival suffix

The adjectival suffixes join nouns to form adjectives. The adjectival affixes considered are ഉടെ uṭe, അയാ / aaya and ഉല്ല /ulla. The nouns with suffixes ഇലെ /ile is treated differently than the other suffixes during the final word alignment phase.

രാമു ദരിദ്രൻ അയാ പനം കോടത്തു (raamu daridran aaya mohanu paNam kotuththu)

Ramu gave money to poor Mohan

In this the suffix അയാ / aaya following the word ദരിദ്ര് (daridran/poor) makes the word adjective which qualifies the noun മോഹൻ /mohan which follows it.

15. ADVA Adverbal suffix

These are suffixes which forms adverbs when joined with nouns. Two examples are കുട യായി / aayi and ഇവലക്ക് /ilEkkz.

രാമു താറയ് (nannz–aayi-–) kazhuki.

Ramu washed the floor neatly

In this the suffix യായി (aayi / ly) added to the noun നറ്റ / (nannz / neat) forms the adverb which qualifies the verb കൽ kazhuki.
16. **RP Relative participle**

Relative participles form the clause markers for adjective clauses. Some of the words forming the relative participles are `paranjja`, `etuthha`, `kotuthha` etc.

(Seetha ate the banana which Raju gave) (raaju-kotuththa-) pazham seetha kzhichchu.

In the sentence `kotuthha` is the word which belongs to this category which marks the adjective clause `raaju kotuththa` which qualifies the noun `pazham`.

17. **ADVCA Adverbal clause suffix**

These form the marker suffixes in adverb clauses. The suffixes in this category are `appOL`, `kazhinju`, `athinuSesham`, `athininumpu`, `athuthottz`, `athumuthal`, `athinuventi`, `athukontu`, `athukaaranam`, `aal`, `Athinekkal` and `athupOle`.

(raamu viLichhathukontunjaan pOyi) I went because Ramu called.

118
In the above sentence അതുകണ്ട (athukont/because) marks the end of the adverb clause രാമു നില്പിക്കുമെന്താണ് / raamu vilichathukontu.

ബാല് നിലപാട് കാലി പകര് (raamu vilichaal njaan pOkum).

I will go if Ramu calls

In the above sentence the suffix ആല് (aal/if) marks the end of the adverb clause രാമു നിലപാട് / Raamu vilichaal.

18. LOC Locative suffix

The locative suffix for nouns ഇല് (ile/in) is given a separate tag as the noun with this suffix has to be treated differently compared to possessive suffix during the syntactic transfer.

7.3.2 Selection of Chunk Tags

After selection of POS tags in sentences the chunk tags were identified. The chunks that are to be rearranged for the translation from Malayalam to English were identified and given a unique tag name for each chunk. The tagset includes all of the tags in IIIT tagset and also some additional tags to handle
Table 7.1  POS tags

<table>
<thead>
<tr>
<th>No.</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PL</td>
<td>Plural suffix</td>
</tr>
<tr>
<td>3</td>
<td>NA</td>
<td>Postposition</td>
</tr>
<tr>
<td>4</td>
<td>PA</td>
<td>Adjective</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
<td>Noun</td>
</tr>
<tr>
<td>6</td>
<td>V</td>
<td>Verb</td>
</tr>
<tr>
<td>7</td>
<td>ADJA</td>
<td>Adjectival suffix</td>
</tr>
<tr>
<td>8</td>
<td>ADVA</td>
<td>Adverbal suffix</td>
</tr>
<tr>
<td>9</td>
<td>PAV</td>
<td>Adverb</td>
</tr>
<tr>
<td>10</td>
<td>VN</td>
<td>Verbal Noun</td>
</tr>
<tr>
<td>11</td>
<td>V RP</td>
<td>Verbal Relative participle</td>
</tr>
<tr>
<td>12</td>
<td>NCA</td>
<td>Noun clause suffix</td>
</tr>
<tr>
<td>13</td>
<td>ADVCA</td>
<td>Adverbal clause suffix</td>
</tr>
<tr>
<td>14</td>
<td>INFA</td>
<td>Infinitive</td>
</tr>
<tr>
<td>15</td>
<td>DJ</td>
<td>Disjunction</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>Conjunction</td>
</tr>
<tr>
<td>17</td>
<td>LOC</td>
<td>Locatives</td>
</tr>
<tr>
<td>18</td>
<td>VA</td>
<td>Verbal suffix</td>
</tr>
</tbody>
</table>

higher level constructs like clauses. The list of chunk tags is shown in Table 7.2. A chunk tag is allotted for each of the morpheme group found in the hierarchical structure. The tags were so chosen that it helps to identify the morpheme groups to be used in the reordering process to generate the target language parse tree.
Table 7.2 Chunk tags

<table>
<thead>
<tr>
<th>No.</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NP</td>
<td>Noun Group</td>
</tr>
<tr>
<td>2</td>
<td>VG</td>
<td>Verb Group</td>
</tr>
<tr>
<td>3</td>
<td>NC1</td>
<td>Noun clause</td>
</tr>
<tr>
<td>4</td>
<td>ADVC</td>
<td>Adverb clause</td>
</tr>
<tr>
<td>5</td>
<td>ADJC</td>
<td>Adjective clause</td>
</tr>
<tr>
<td>6</td>
<td>NPC</td>
<td>Conjunct Noun</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>Sentence</td>
</tr>
<tr>
<td>8</td>
<td>CS</td>
<td>Compound sentence</td>
</tr>
<tr>
<td>9</td>
<td>CMPN</td>
<td>Compound noun</td>
</tr>
<tr>
<td>10</td>
<td>ADJCNP</td>
<td>Adjectival clause + Noun</td>
</tr>
<tr>
<td>11</td>
<td>ADJG</td>
<td>Adjective group</td>
</tr>
<tr>
<td>12</td>
<td>INFSG</td>
<td>Infinitive + verb group</td>
</tr>
<tr>
<td>13</td>
<td>INF</td>
<td>Infinitive</td>
</tr>
<tr>
<td>14</td>
<td>ADVG</td>
<td>Adverb group</td>
</tr>
<tr>
<td>15</td>
<td>VGC</td>
<td>Compound verb</td>
</tr>
<tr>
<td>16</td>
<td>VA</td>
<td>Verbal suffix</td>
</tr>
<tr>
<td>15</td>
<td>ADJLOC</td>
<td>Locative adjective</td>
</tr>
</tbody>
</table>

7.3.3 Hierarchical dependency rules between chunks

The parser needs the hierarchical dependencies among the chunks for the creation of parse tree and also for the final reordering required to build the target parse tree. The sentences of Malayalam language were carefully
analysed to identify these dependency rules and the syntactic structure of sentences in Malayalam were formulated.

Malayalam belongs to Indo- Dravidian family of languages and it is an relatively free word order language like other Dravidian languages. Malayalam is an S-O-V language. The default or unmarked order of constituents is Subject first, then the Object and finally the verb. However, Malayalam, being a relatively free word order language, permits substantial amount of freedom in the order of constituents although normally the verb remains in the sentence final position. Word order is less important mainly because noun groups are marked for cases and the verb agrees with the subject in gender, number and person. Subjects and objects are often dropped. The subject of a sentence is expressed by a noun group in the nominative case in most of the sentences [74]. Normally all modifiers precede the modified.

There are a variety of subordinate clauses. Subordinate clauses also precede the main clause. They typically involve special non-finite forms of verbs which occur invariably in the clause final position and mark the right hand boundary of the respective clauses. All these assertions were used to form the syntax rules. There are exceptional situations where deviations from these rules are possible. Also, most of these rules apply not only to Malayalam but to Dravidian languages in general.

1. ഞാന്‍ നാന്റള്‍വരം (njaan nnaLe varum)
   I will come tomorrow

2. രാമന്‍ കാട്ടില്‍വപായി (raaman kaattil pOyi)
   Raman went to forest
The nominative case marker is empty in sentences 1 and 2. Malayalam also permits dative subject constructions where the understood subject is indicated by a noun group in dative case whereas the surface subject appears in the nominative case.

1. എനിക്ക് മിഠായി ഇഷ്ടമാണ് (I like sweets)
   
   enikku mittayi ishtamaanz

2. സീതക്ക് നാല് പൂച്ചകളുണ്ട് (seethakkz naalupoochakaL untz)
   
   Seetha has four cats

Malayalam has postpositions unlike prepositions in English. The genitive precedes the head noun in the genitive phrase and the complementiser follows the embedded clause. Adjectives, participial adjectives and free relatives precede the head noun. There is no person, number and gender (PNG) agreement between the subject and the verb as in the case of English. Based on the above facts the hierarchical dependencies among the chunks in a Malayalam sentence are shown in Figure 7.1. The rules are shown as a finite state diagram. The circles shown are the states and the transitions between states are POS tags or chunk tags.

Rules for forming chunks are given below with examples.

1) Start - Highest level chunk

1. S - A simple sentence
   
   രാമു പഠിക്കുന്നു (raamu paThikkunnu)
   
   Ramu is studying
Figure 7.1 Hierarchical dependency rules
Figure 7.1 Hierarchical dependency rules Contd....
Figure 7.1 Hierarchical dependency rules Contd....
Figure 7.1 Hierarchical dependency rule Contd....
2. CS – Complex sentence

(Raamupadichaal)(ADVC) (pareekshayiljayikkum)(S)

If Ramu studies he will pass in the examination

2) CS - Complex sentence
1. An adverb clause followed by a simple sentence

(Raamupadichaal)(ADVC) (pareekshayiljayikkum)(S)

If Ramu studies he will pass in the examination

2. A noun clause followed by a complex sentence

(Raaman Mohane adichchennu)(NC) (ramaye kandappOL seetha paRanjnuu)(CS)

When Seetha saw Rama she told that Raman hit Mohan

3. An adverb clause followed by a complex sentence

(raamu padichaal)(ADVC)(pareekshayil jayikkumennu mohan paranju) (CS)

Mohan told that if Mohan studies he will pass in the examination
4. A noun clause followed by a simple sentence

(രാമന് വമാഹന്റന അടിച്ചന്ന്(NC) സീത പറഞ്ഞു(S))

(seethaparanju)(S)

Seetha told that Raman ad hit Mohan

3) S - Simple sentence

One or more noun groups followed by a verb group.

NP(രാമന്) NP(വമാഹന്റന) VG(അടിച്ചു) (Raman hit Mohan)

NP(raaman) NP(mohane) VG(aticchu)

കുട്ടികള് ക്ലാസ്സില് വപാകാന് വപാകുന്നു (Children are about to go to class)

NP(kuttikaL) NP(cLaSSil) VG(pOkaanpOkunnu)

4) ADVC - Adverb clause

A simple sentence followed by adverb clause marker.

S(രാമുവ) CONDP(ആല) (S(raamu vann) CONDP(aal))

if Ramu comes

5) NC1 - Noun clause

A sentence followed by the clause marker നുന്ന / ennz forms noun clause.

(രമ എന്നു മഹന്റന പറഞ്ഞില് (srama vannu)(S) ennu(NCE1) (mOhan paRanjnju)(S))

(Mohan told that Rama had come)
6) **NPC - Noun Conjunct**

A noun group followed by the conjunct suffix ഉം/um forms a conjunct noun.

രമ / rama(NP) – ഉം/um(C) രവി / ravi (NP)– ഉം/um (C) (Rama and Ravi)

7) **ADJC - Adjective clause**

A sentence followed by relative participle forms an adjective clause.

(സീത‍പ്)(ADJC) കഥ‍രമക്കിഷ്ടന്റെട്ടു (seetha paRanjnja)(ADJC) kadha Ramakkuishtappettu

(Rama liked the story which Seetha told)

8) **NP - Noun chunk**

1. A noun alone.

ഉം (raaman / Raman)

2. A noun followed by a case marker

ഉം - ഓ (raaman-Odu / to Raman)

3. A noun followed by a plural marker and a case suffix

കള് - ഓ (kutti-kaL-Odu / to children)

4. A noun preceded by an adjectival clause

(അ എന്റെ കഥ) (the story which Raman told)

rama paRanjnja kaTha

9) **CMPN - Compound noun**

A noun followed by another noun.
ADJCNP - Noun preceded by an adjective clause
The adjective clause and the noun it qualifies are grouped as they are to be treated as a single unit during structure transfer from Malayalam to English.

ADJG - Adjective chunk
1. A pure adjective
   നല്ല (nalla / good), കുന്ററ (kure / some)
2. A derived adjective formed by a noun followed by adjectival suffixes.
   ഭംഗി (bhangi / beautiful) – ഉള്ള (ulla)

VG - verb group
1. Zero or more adverb group followed by a verb, verb and inflectional suffixes, verb, inflectional suffix and question tag.
   വപായി (V)(pOyi/went), വപാക് (V)(pOk) – ഉന്ന് (VA)(unnu /is going)
2. A Compound verb i.e. a verb followed by another verb
   ചാടി/chaadi (V) കയറി/kayar(V)(climbed jumping), ഓടി/Odi(V)
   വപായി / pOyi(V)(went running)
3. Infinitive followed by a verb
   വപാക്/pOk(V)– ആന്ന്/aan-(INFA) വപായി /pOyi(V)(went to go)

INFSG - Infinitive followed by a verb group
The infinitive and the verb following it are grouped.
14) **INF- Infinitive**

A verb followed by the suffix ആന് / aan is taken as infinitive.

ബാക് / pOkaan(INF) ആന് / thutangi(V)(started to go),

ബായി / vaangaan(INF) ബായി / pOyi(V)(went to by)

15) **ADVG - Adverb group**

1. Pure adverb (PAV)

പതുന്റക്ക / pathukke(slowly), ന്റപന്റട്ടന്ന് / pettennu(quickly)

2. Noun followed by adverbial suffix

ഭംഗി / bhangi(N) – ആയി / aayi(ADVA)(beautifully)

16) **VGC- Compound verb**

A verb followed by another verb are grouped to form a compound verb.

ചാടി / chaati(V) – കയറി / kayaRi(V), നടന്ന് / natannu(V) – ബായി / pOyi(V)

### 7.3.4 Syntactic structure difference between Malayalam and English

The different classes of sentences in Malayalam and their English translation were carefully analysed and the set of reordering rules required in Malayalam to English translation process were derived. It was found that the reordering occurs in the word level and also in the chunk level. 13 rules wee identified for reordering which are given below along with examples. The set of chunk transfer rules are shown in Table 7.3.
1. When an adverb clause is encountered the adverb clause marker is moved to the beginning of the clause. In the following sentence the clause marker ല് (appOL/when) in the end of the clause is moved to the beginning of the clause.

ന് മഹാന്റെ സീതയെ അടുക്കി (appOL)

mohan seethaye adichch-(appOL)

(when) Mohan hit Seetha

2. When a noun clause is found, the noun clause marker is moved to the beginning of the clause. In the following example the clause marker ന് (ennu/that) is moved to the beginning of the noun clause.

ന് മഹാന്തടെ വീട്ടിൽ പോയ് (ennu)

mohan veettil pOy-(ennu)

(that) Mohan went to house

3. When a noun group with a noun and a case suffix is found position of both are interchanged. In the following example positions of വീട്ട് (veett/house) and the case marker ല് (il/to) are interchanged.

വീട്ട് ല്

veett-il

to house
4. When a simple sentence is found the verb is placed after the subject. If the sentence does not have a subject it is placed before the object. In the example the verb ന്ന്‌ (pOyi/went) is positioned after the subject ന്ന്‌ മOhan.

   ന്ന്‌ മOhan veettil (pOyi).
   Mohan (went) house to.

5. When an adjective clause is found the adjectival marker is placed before the clause. The marker is replaced by who or that depending on whether the noun it qualifies is human or nonhuman. In the following example the adjective clause marker ത്ത / a (which/who/what) is moved to the beginning of the clause.

   ന്ന്‌ കുത്ത‌- രാമന
   raman kotuthth-a
   (which) raaman gave

6. When an adjective clause is followed by a noun group the noun group is placed in the beginning of the clause. In the example given below the noun following the adjective clause കട (kuta/umbrella) is moved to the beginning of the clause.

   ന്ന്‌ കട (കട)
   raman kotuththa (kuta)
   (umbrella) which raman gave
7. When complex sentence with one subordinate clause and main clause is found the main clause is written first. In the given example the main clause പറഞ്ഞു / seetha paRanju is written first.

mOhan veettiil pOyennu (seethe paRanjju)
(Seetha told) that Mohan went to house

8. When an infinitive is found the infinitive stem and suffix are exchanged. The infinitive marker ആ (aan/to) and the verb root പഠിക്കു (paThikkz/study) are interchanged.

padhikk-aan

to study

9. When a verb group is encountered the inflection suffix and root are exchanged. Proper modifications are made to the verb root. In the following example the verb തോട്ട /Ot(z/run) and the suffix കൊണ്ട് /ikkonrunnu(had been) are interchanged.

ot-ikkondirunnu.

had been running

10. When a verb follows an infinitive they are interchanged. In sentences with objects the infinitives are placed after the object. In the following example the infinitive അടിക്കാൻ (adikkaan/to hit) and the verb following it are interchanged.
11. The adjective group i.e. a noun with adjective derivation markers ( ويم / ile) and the noun group following it should be interchanged. In the following example तरातली / maraththile is a derived adjective formed from तरातली / maram(tree) and the marker य (ile / in). The adjective and the noun it qualifies are interchanged.

तरातली य / बीली

maraththile kili

bird in tree

12. Adverb and verb following it should be interchanged. In sentences with objects the adverb is placed after the object. In the following example the verb य (pOyi/went) and the adverb बीली / pettennz/quickly) are interchanged.

य / बीली

pettennz pOyi

went quickly

7.4 Parser model

A top down depth first search method was chosen as the parsing strategy [93]. The parser tries to find all valid parser trees for the source sentence. The parser creates a source parse tree using the syntax rules given in Figure 7.1. It uses synchronous tree adjoining grammar (STAG) approach to create the target
parse tree. Synchronous grammars for TAGs were introduced by Shieber and Schabes to characterize correspondences between tree adjoining languages [111,112]. Synchronous TAGs are used for relating TAGs for two different languages for the purpose of machine translation or generation or semantic analysis [113].

The model keeps both source and target tree pair, and performs operations simultaneously while traversing through the tree nodes. Thus the syntax

### Table 7.3 Structure transfer rules

<table>
<thead>
<tr>
<th>No</th>
<th>Source structure</th>
<th>Target structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S ADVCA</td>
<td>ADVCA S</td>
</tr>
<tr>
<td>2</td>
<td>S NCA</td>
<td>NCA S</td>
</tr>
<tr>
<td>3</td>
<td>N NA</td>
<td>NA N</td>
</tr>
<tr>
<td>4</td>
<td>NP* VG</td>
<td>NP VG NP*</td>
</tr>
<tr>
<td>5</td>
<td>S ADJA</td>
<td>ADJA S</td>
</tr>
<tr>
<td>6</td>
<td>ADJC NP</td>
<td>NP ADJC</td>
</tr>
<tr>
<td>7</td>
<td>ADVC S</td>
<td>S ADVC</td>
</tr>
<tr>
<td>8</td>
<td>V INFA</td>
<td>INFA V</td>
</tr>
<tr>
<td>9</td>
<td>V VA</td>
<td>VA V</td>
</tr>
<tr>
<td>10</td>
<td>INF V</td>
<td>V INF</td>
</tr>
<tr>
<td>11</td>
<td>N LOC</td>
<td>LOC N</td>
</tr>
<tr>
<td>12</td>
<td>ADJLOCN</td>
<td>N ADJLOC</td>
</tr>
<tr>
<td>13</td>
<td>ADVG V</td>
<td>V ADVG</td>
</tr>
</tbody>
</table>
structure transfer from Malayalam to English is integrated into the parser [114]. The syntax rules were given in the regular expression form. The longest rules are listed first on the right hand side of each production rule so that longer chunks are recognized first. Regular expression notation helps for a compact representation of language syntax and also it provides easy modification of the syntax rules. Some examples regular expressions for language syntax and production rules are given below. The compete set of syntax rules in the context free grammar notation is given in Table 7.4.

1. S -> NP*VP
The rule states that a simple sentence is a sequence of noun chunks followed by a verb chunk.

2. NG -> ADJ*N
The rule states that a noun chunk consists of a set of adjectives followed by a noun.

3. VG -> ADV*V|INF V
The rule states that a verb chunk consist of a sequence of adverbs followed by a verb or an infinitive followed by a verb.

7.5 Algorithm of the parser
The algorithm works in a recursive manner. The search begins with the start symbol of the set of grammar rules and works in a top down manner. As each symbol is recognized the sub tree is added to the source parse tree and the subtree with the required change in order is added to target parse tree. Each
Table 7.4 Grammar rules in regular expression form

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Production rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>START=&gt;S</td>
</tr>
<tr>
<td>2</td>
<td>CS=&gt;ADVC S</td>
</tr>
<tr>
<td>3</td>
<td>ADVC=&gt;S ADVCA</td>
</tr>
<tr>
<td>4</td>
<td>NC1=&gt;S NCE1</td>
</tr>
<tr>
<td>5</td>
<td>S=&gt;NP* VG</td>
</tr>
<tr>
<td>6</td>
<td>NPC1=&gt;NP C</td>
</tr>
<tr>
<td></td>
<td>NPC=&gt;NPC1 NPC1</td>
</tr>
<tr>
<td>7</td>
<td>ADJC=&gt;NP* VRP</td>
</tr>
<tr>
<td>8</td>
<td>ADJG=&gt;PA</td>
</tr>
<tr>
<td></td>
<td>ADJLOC= N LOC</td>
</tr>
<tr>
<td>9</td>
<td>NP=&gt;ADJG* N</td>
</tr>
<tr>
<td></td>
<td>ADJLOCN=&gt;ADJLOC N</td>
</tr>
<tr>
<td>10</td>
<td>VG=&gt;ADV* V NE</td>
</tr>
<tr>
<td>11</td>
<td>ADVG=&gt;PAV</td>
</tr>
<tr>
<td>12</td>
<td>INFSG=&gt;INF V</td>
</tr>
<tr>
<td>13</td>
<td>INF=&gt;V INFA</td>
</tr>
</tbody>
</table>

rule on the right side of the production is tried with the following algorithm.

1. Initialise two trees to store the parse trees for the source and target parse trees identified.
2. For each possible untried symbol in the grammar rule do steps 3-7

3. If the symbol is a primitive morpheme tag then if it matches with the word in input move to the next symbol in the rule and advance the input pointer to get the next morpheme. Create a node with the matched morpheme. Add it to current parse tree. Go to step 2.

4. Else if there is a mismatch then the next rule for the nonterminal is tried. The input pointer is reset to the initial position.

5. Else if the transition is on a chunk tag then search for the symbol recursively. If it is a successful parse add the sub tree returned by the call to the current parse tree and the reordered tree to the target parse tree. Go to step 2. If it is a failure then the next rule for the nonterminal is tried. The input pointer is reset to the initial position.

6. Else if all the symbols for the current rule are satisfied then the parse tree for the non terminal is returned.

7. Return failure.

8. Stop.

### 7.6 Creation of source parse tree

A sample grammar is given below. The working of the top down parsing algorithm for the sample input ‘Mohan paatunnu’ is shown in Figure 7.2.

S-> NP VP | VP
   NP-> Adj Nominal | Noun
   Nominal -> Noun | Noun Nominal
VP -> NP Verb | Verb  
Verb -> uRangunnu | padhikkunnu | natakkunnu | paatunnu  
Noun -> raam | mOhan

In level 3, in substitution 1 the first terminal adj does not match with the first input which is a noun Mohan. So the next alternative for NP is tried. Now noun matches with the first input. So the same procedure is repeated for the VP part of level 3. If at any level all the options fail then control goes back to the previous level and there other options are tried and a depth first search starts with the new substitution. A successful parse corresponds to a tree which matches exactly with the words in the input sequence.

7.7 Creation of target tree

A parse tree for a Malayalam sentence and its target tree in English created by the parser are shown in Figure. 7.3.

রവി പുസ്തകം വാങ്ങാൻ യി (ravi pusthakam vaangngaan pOyi)

Ravi went to buy book

More sample results are given in Appendix 3. The parser uses rule no. 8, 10 and 4 of the structure transfer rules described in this chapter for parsing the above sentence. The target parse tree is created along with the source tree in a depth first manner. When the subgroups are formed during parsing appropriate rearrangement rules are applied to the subgroup in the target parse tree. For
Figure 7.2 parse tree creation in top down parsing
example in the above sentence, when the group INF is recognized, reordering rule no. 8 is applied to interchange the suffix and the verb root. After that when VG group is found rule no.10 is applied to interchange infinitive and finite verb went. Finally when the simple sentence S is recognized, the sub tree for the verb group VG is placed after the first noun group(i.e. subject) according to rule 4. Thus when the parsing of the source sentence is complete the target parse tree is created for the target sentence generator module.
7.8 Conclusion

The chapter discussed the design and development of the parser module of the prototype translation system. It clearly described the working of the parser algorithm using sample inputs. The chapter also discussed the set of part of speech tags, chunk tags and the chunk dependencies derived to be used by the parser. Only a subset of the tagset developed by IIIT was identified for the application. The tagset can be extended to handle other sentence classes. The selected set of syntactic structure transfer rules required in the translation of Malayalam sentences to the corresponding English sentence were also discussed in this chapter.