19

Context Free Grammar for Natural Language constructs – An implementation for Venpa class of Tamil Poetry

Bala Sundara Raman L, Ishwar S, Sanjeeth Kumar Ravindranath

Indian Institute of Information Technology, "Innovator" Towers, ITPL,
Whitefield Road, Bangalore 560066, India.
{balasundaraman.l, ishwar.s, sanjeeth.kumar}@iiitb.ac.in

Abstract:

Much of linguistic research has gone into formalising natural languages. Such formalisation depends on the nature and structure of the language. In this paper we formally represent the grammar of a poetry construct of the Tamil language as a context free grammar (CFG). This representation is aided by the manner in which the grammar for the language has been defined. The definition of the grammar, which dates back to the 5th century B.C, follows a pattern very similar to CFG, the notion of which was formalised in the latter part of the 20th century.

A class of Tamil poetry called as "Venpa" has been represented as a context free grammar. An application has also been built that checks if the given piece of poetry adheres to the rules defined by the grammar for Venpa and also provides suggestions whenever there are violations.

Keywords: Context free grammar, natural language grammar, linguistics, phonetics

1. Introduction

Formalisation of natural languages has been a topic of interest among linguistic researchers for years. But such formal representations depend much on the nature of the language. This paper deals with formally describing the grammar for a particular class of Tamil[9] poetry called "*Venpa"* by exploiting the inherently structured nature of Tamil language. We have expressed Venpa as a Context Free Grammar (CFG), a notation that has been used extensively for defining the syntax of programming languages. One of the interesting features is that Tholkaappiyar[1], who is believed to have lived during the 5th century B.C, has defined the rules for the Venpa in a manner very similar to CFG constructs. In fact, he has defined the grammar for the entire Tamil language in this manner.

Using modern computational notations to express natural language constructs makes development of applications that parse natural language inputs easier. We have developed a

[9] Pronunciation of 'l' in the word Tamil does not correspond to the way 'l' is pronounced in English. To denote the difference it is customary to write Tamil as Thamizh.

parser called "Visai Neri"[10] that checks if the entered text complies with the rules of Venpa. Visai Neri helps in both writing and analyzing Venpa type of poetry. Visai Neri correctly identifies text that adheres to the rules of Venpa and also detects anomalies if present.

2. Background

2.1 Context Free Grammar (CFG)

Chomsky[2,3] proposed the notion of CFG as a model for describing natural languages. When applied to the English language, CFG was successful in describing only a small subset of simple sentences. Backus and Naur [5] proposed the Backus Naur Form (BNF) as a formal notation for representing the syntax of programming languages and this was used in defining the syntax of ALGOL 60. But BNF was found to be equivalent to CFG[4] with some minor changes in notations and format.

CFG has been widely used for defining programming languages rather than natural languages. A CFG involves the following four quantities:

1) Terminals:

Terminals define the basic symbols of which strings in the language are composed.

2) Non-terminals:

Non-terminals are special symbols that denote the set of strings of the language. Non-terminals are described recursively in terms of each other and terminals.

3) Productions:

Productions are rules that define the ways in which non-terminals may be built from one another and from terminals. Production rules are represented as follows:

$A \rightarrow \alpha$

where A is a non-terminal and α is a string of terminals and non-terminals.

4) Start symbol:

Start Symbol is a special non-terminal from which all other strings are derived. It signifies the language being defined.

Formally a CFG is a quadruple (N, T, P, S), where N and T are finite sets of non-terminals and terminals respectively; P is the finite set of productions, each production is of the form $A->\alpha$, where A belongs to N and α is a string of symbols from $(N \cup T)^*$; S is the start symbol. Figure 1 gives the grammar for simple arithmetic expressions as a CFG. E and O are non-terminals and E is the start symbol. The remaining symbols are terminals.

[10] Visai means force and Neri means framework in Tamil. Visai Neri provides a framework for the creative force of poetry.

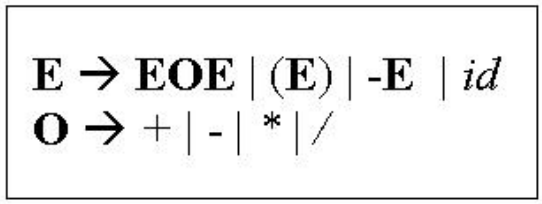


Figure 1.A CFG for simple arithmetic expressions

2.2 Tholkaappiyam

Tholkaappiyar[1] defines the grammar for the entire Tamil language. The description of the grammar follows a structure very similar to CFG. He defines formal production rules and definitions to describe the grammar. Starting with the set of symbols that constitute the valid character set of the language, he goes on to describe the rules that define the ordering of these symbols into strings and their ordering into valid language constructs. Though this paper primarily deals with Venpa, this structured description could be exploited to formalise broader classes of the language.

3. Venpa and CFG

3.1 Venpa

Classical Tamil poetry has been classified into various categories based on phonetics. Venpa is one such class of poetry. Venpa class of poetry ranges between two to twelve lines following specific rules. All the 1330 poems of Thiruvalluvar[8] are examples of Venpa. The rules for Venpa, as specified by Tholkaappiyar are shown in Figure 2.

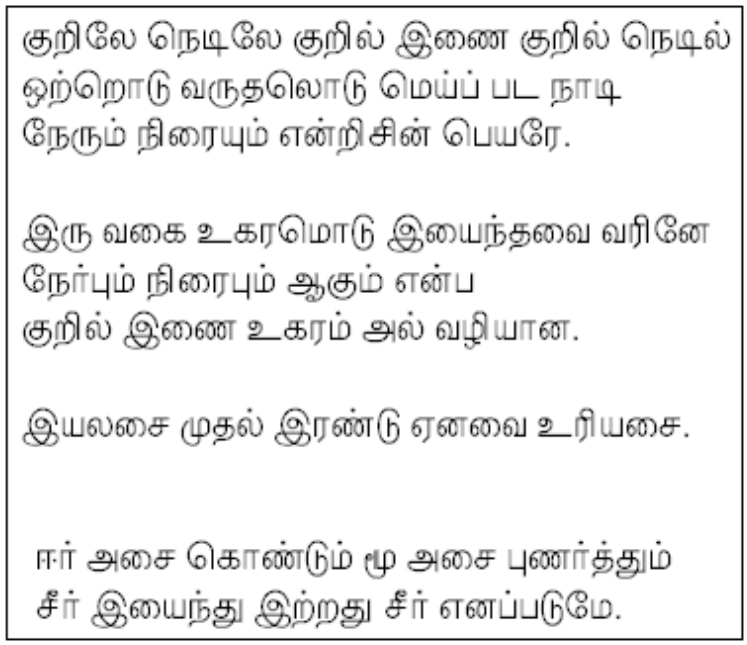
The alphabets in Tamil language can be divided into three categories based on phonetics:

a. Nedil

b. Kuril

c. Otru

Any Tamil alphabet is either a vowel, consonant or compound type (formed from vowels and consonants). Otru are the consonants. Vowels and the compound types can be categorized into kuril and nedil. The difference between kuril and nedil lies in the duration of their sounds. The duration of the sound of nedil alphabets is longer than that of kuril alphabets. The rules of Venpa define a particular ordering of alphabets and words that enforce a rhythm, characteristic of all Venpa poems.


குறிலே நெடிலே குறில் இணை குறில் நெடில்
ஒற்றொடு வருதலொடு மெய்ப் பட நாடி
நேரும் நிரையும் என்றிசின் பெயரே.

இரு வகை உகரமொடு இயைந்தவை வரினே
நேர்பும் நிரைபும் ஆகும் என்ப
குறில் இணை உகரம் அல் வழியான.

இயலசை முதல் இரண்டு ஏனவை உரியசை.

ஈர் அசை கொண்டும் மூ அசை புணர்த்தும்
சீர் இயைந்து இற்றது சீர் எனப்படுமே.

Figure 2. Excerpt from Tholkaappiyar's definition of Venpa.

3.2 Venpa as a CFG

The grammar for Venpa is expressed as a CFG in Figure 3.

In addition to the rules specified in the figure, Tholkaappiyar has defined the notion of Thalai. He describes Thalai not as a component of the poetry schema, but as a relationship between components. Poems are composed of lines (ADI), which in turn are composed of words (CHEER), which in turn are composed of further sub-components. Thalai is a set of rules that define the association of words. These are the rules that are responsible for providing the "*phonetic sugar*" to the poem by enforcing a rhythm characteristic of Venpa[11]. These rules are represented as a CFG in Figure 4. Based on the rules followed, Thalai can be classified as:

1. Iyarcheer Vendalai
2. Vencheer Vendalai

It should be noted that an equivalent regular expression could be written for the CFG shown in Figure 3. But the intricacies of Venpa such as Thalai can be captured only by a CFG (The CFG shown in Figure 4 does not have an equivalent regular expression). Hence we have defined even the general grammar for Venpa as a CFG for the sake of uniformity, ease of implementation and analysis.

[#] {} is not a standard notation in BNF but is a widely accepted Perl notation.
[11] The characteristic rhythm of Venpa is termed "Cheppalosai" in Tamil and translates to "sermonizing tone".

```
<VENPA>          ::= <ADI>{1-11}#<EETRADI>
<ADI>            ::= <CHEER> <CHEER> <CHEER> <CHEER>
<EETRADI>        ::= <CHEER> <CHEER> <EETRU CHEER>
<CHEER>          ::= <EERASAI> | <MOOVASAI>
<EETRU CHEER>    ::= <NAAL> | <MALAR> | <KAASU> | <PIRAPPU>
<EERASAI>        ::= <THEMA> | <PULIMA> | <KARUVILAM> | <KOOVILAM>
<MOOVASAI>       ::= <THEMANGAI> | <PULIMANGAI> | <KARUVILANGAI> | <KOOVILANGAI>
<THEMA>          ::= <NER> <NER>
<PULIMA>         ::= <NIRAI> <NER>
<KARUVILAM>      ::= <NIRAI> <NIRAI>
<KOOVILAM>       ::= <NER> <NIRAI>
<THEMANGAI>      ::= <THEMA> <NER>
<PULIMANGAI>     ::= <PULIMA> <NER>
<KARUVILANGAI>   ::= <KARUVILAM> <NER>
<KOOVILANGAI>    ::= <KOOVILAM> <NER>
<NAAL>           ::= <NER>
<MALAR>          ::= <NIRAI>
<KAASU>          ::= <NER> <NER>
<PIRAPPU>        ::= <NIRAI> <NER>
<NER>            ::= <KURIL> | <NEDIL> | <KURIL> <OTRU> | <NEDIL> <OTRU>
<NIRAI>          ::= <KURIL> <KURIL> | <KURIL> <NEDIL> | <KURIL> <KURIL> <OTRU>
                   | <KURIL> <NEDIL> <OTRU>
```

Figure 3. CFG for Venpa

```
<VENPA> ::= <X> | <Y>
<X>     ::= <THEMA> <Y>
<X>     ::= <KOOVILAM> <X>
<Y>     ::= <PULIMA> <Y>
<Y>     ::= <KARUVILAM> <X>
<X>     ::= <NULL>
<Y>     ::= <NULL>

[ If the above rules are conformed to, the thalai is said to be Iyarcheer vendalai]

<X>     ::= <THEMANGAI> <X>
<X>     ::= <KOOVILANGAI> <X>
<Y>     ::= <PULIMANGAI> <X>
<Y>     ::= <KARUVILANGAI> <X>

[ If the above rules are conformed to, the thalai is said to be Vencheer vendalai]
```

Figure 4. CFG for Thalai.

3.3 Example

Figure 5 shows a Venpa taken from Thirukkural. To demonstrate how the poetry con-forms to the rules of Venpa, it has been represented as a derivation tree in Figure 6. In the figure, 0 represents Kuril, 1 represents Nedil and 2 represents Otru. While the derivation tree helps in appreciating the structure of the poetry, one cannot resist appreciating the imagination of the poet for giving such an apt analogy to describe the trite “A friend in need is a friend indeed”. G. U. Pope et al[9] provide the English translation of the poem as follows:

“As hand of him whose vesture slips away,
Friendship at once the coming grief will stay.
(True) friendship hastens to the rescue of the afflicted (as readily) as the hand of one whose garment is loosened (before an assembly).”

உடுக்கை இழந்தவன் கைபோல ஆங்கே
இடுக்கண் களைவதாம் நட்பு

Figure 5. Example of a Venpa

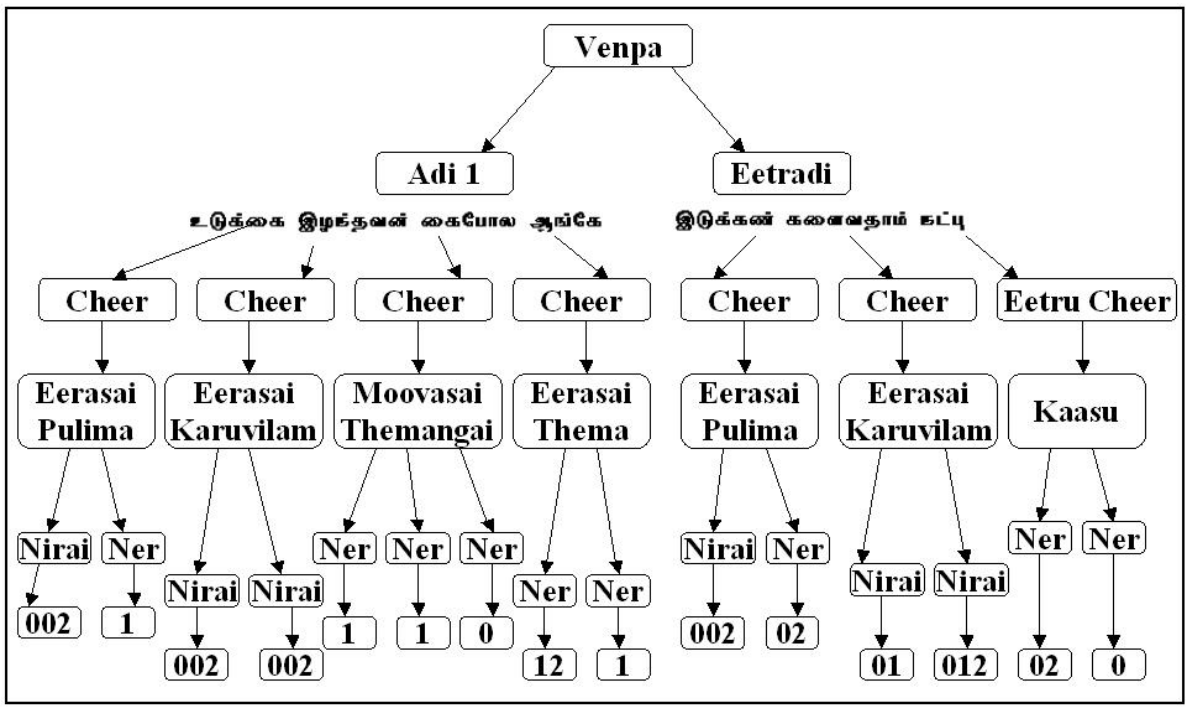


Figure 6. Derivation tree for the Venpa shown in Figure 5.

The set of productions that enforce the rules of Thalai for the Venpa is shown in Figure 7. It should be noted that this particular Venpa contains both Iyarcheer Vendalai and Vencheer Vendalai.

<Venpa> ::=<Y>
<Y> ::=<Pulima><Y>
<Y> ::=<Karuvilam><X>
<X> ::=<Themangai><X>
<X> ::=<Thema><Y>
<Y> ::=<Pulima><Y>
<Y> ::=<Karuvilam><X>
<X> ::=<Thema><X>
<X> ::=<Null>

Figure 7. The set of productions enforcing the rules of Thalai for the Venpa shown in Figure 5

4. Implementation

The application "Visai Neri" that we have built verifies if a given piece of poetry satisfies the rules of Venpa and offers suggestions for the poet whenever there are violations. The broad architecture for "Visai Neri" is shown in Figure 8. Visai Neri consists of three components:

1. Tokeniser
2. Parsers (PASS 1 and PASS 2)
3. Report Engine

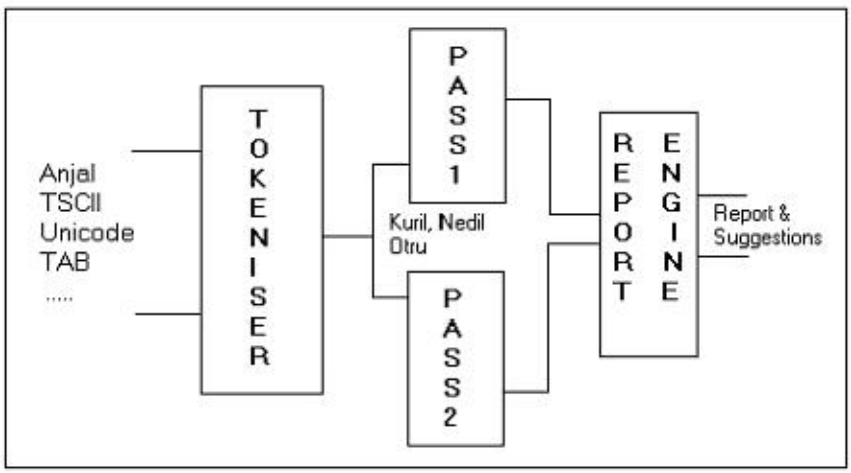


Figure 8.Architecture of Visai Neri.

4.1 Tokeniser

The input to the tokeniser can either be in TSCII, Unicode, TAB or Anjal encoding. Tokeniser maps each character of the input to a token and generates a sequence of tokens. A token can be a Kuril, Nedil or Otru.

4.2 Parsers

The two parsers are Push Down Automata[6,7] that check for conformance with Venpa rules. The first parser checks if the basic rules of Venpa as shown in Figure 3 are adhered to. The second level parser checks for conformance to Thalai rules specified in Figure 4.

The parsers have been implemented in Java using the JavaCC utility [10]. JavaCC was chosen over yacc-like tools because JavaCC generates top-down (recursive-descent) parsers as opposed to bottom-up parsers generated by YACC. This allows the use of more general grammars, although left-recursion is disallowed. Top-down parsers have a bunch of other advantages such as being easier to debug, having the ability to parse to any non-terminal in the grammar, and also having the ability to pass values (attributes) both up and down the parse tree during parsing. The two parser programs (PASS 1 and PASS 2) were generated by feeding the CFGs to the JavaCC utility. The parsers output exceptions that were met while parsing the input text.

4.3 Report Engine

The Report Engine generates a report, which specifies the violations, if present and also suggestions for the expected tokens at the places of violation.

The architecture of Visai Neri is scalable in that, more parts of Tamil grammar can be written in the 'so called' BNF[12] and added. This would help establish the 'structuredness' of Tamil language and also in the development of Tamil Word processors. Similarly more tokenisers can be plugged-in to accommodate other encoding. The implementation is portable across platforms, as it has been developed in Java. We also plan to offer this as a free web service, if we get enough resources.

5. Conclusion

Representation of Venpa as a CFG provides a pointer towards the possibility of representing broader classes of Tamil grammar in a formal way. This formal representation establishes that the Tamil grammar is highly structured. Extensive study of Tamil grammar also reveals that the notations used in it are analogous to modern computational notations. It is also astonishing to note that the poets who have composed thousands of Venpas and other poetry constructs were able to do so without committing even a single '*syntactic error*' even in the absence of tools to assist them.

The structured nature of the language should be exploited to completely formalise the language. In fact, the authors have the vision that a BNF can be written for the entire Tamil grammar if taken up as an open source effort. If the CFG for the entire Tamil grammar is written, applications such as semantic parsers for Tamil Word processors will become possible. Visai Neri is the first step in building such applications (though acting on a smaller subset of the grammar) and demonstrates that classification of classical Tamil poetry can be automated.

6. Bibliography

1. Tholkaappiyar's Tholkaappiyam

   http://www.tamil.net/projectmadurai/pub/pm0100/tolkap.pdf

[12] Though the concept of BNF was proposed in the latter part of the 20th century to describe programming languages, the notion of terminals, non-terminals and productions existed in the grammar of natural languages like Tamil and Sanskrit[11].

2. N. Chomsky, Three models for the description of language, *IRE Trans. Info. Theory* 2 (3) (1956), 113-124.

3. N. Chomsky, On certain formal properties of grammars, *Information and Control* 2 (3) (1959), 137-167.

4. S. Ginsburg, H. G. Rice, Two families of languages related to ALGOL, *Journal of the ACM* 9:3,350-371.

5. Peter Naur et al. Revised report on the algorithmic language Algol 60, *Communications of the ACM* 6(1): 1-17, January 1963.

6. Oettinger, Automatic syntactic analysis and the pushdown store, *Proc. Symposia in Applied Math.* 12, American Mathematical Society, Providence, R.I.

7. Schutzenberger, On context-free languages and pushdown automata, *Information and Control* 6:3,246-264.

8. Thiruvalluvar's Thirukkural *http://www.tamil.net/projectmadurai/pub/pm0001/trkrl1.pdf*

9. G. U. Pope et al. English Translation and Commentary of Thirukkural

   http://www.tamil.net/projectmadurai/pub/pm0153/trkrlpop.pdf

10. JavaCC Documentation

    http://trese.cs.utwente.nl/prototypes/composeJ/wichman/JavaCC/index.html

11. Ganakastadhyayi: A Software on Panini's sutras

    http://www.taralabalu.org/panini/