A Case Study on Ambiguity Detection Method Using Some Set of Grammars.

Hari Mohan Pandey, M.Tech.

Faculty Department of Computing, Middle East College of Information Technology, Muscat, Oman.

E-mail: hari04top@yahoo.co.in
harip@mecit.edu.om

ABSTRACT

This paper details the study of an existing ambiguity detection algorithm for grammars. This algorithm is sound as well as complete to detect ambiguous and unambiguous grammars. In this paper we are exploring ambiguity, existing approaches to deal with ambiguity, their comparisons, challenges, recent trends, etc., and cite the important literature on these elements. The author tested one of the existing algorithms using a set of eighty four grammars includes ambiguous, unambiguous, and grammars derive empty strings. The observation has been made available to show the variations in execution time in finding ambiguous grammars and unambiguous grammar by the studied algorithms.

(Keywords: ambiguity, context free grammars, degree of ambiguity, Chomsky normal form, horizontal ambiguity, vertical ambiguity)

INTRODUCTION

One of the problems with context free grammar revolves around whether a given context free grammar (CFG) is ambiguous. In [1] Michael Kruse proposed a theorem for un-decidability of the ambiguity problem for context free grammars. Ambiguity problems hamper the working of technical languages. Although, no algorithm exists which solves the ambiguity for every CFG, there are algorithms available for some grammars.

Using degree of ambiguity, one can distinguish ambiguous grammar, which is the maximal number of a derivation trees for the words generated by them. However, if there is not any upper bound given then a grammar G is known as ambiguous of finite degree. In [11] Maurer presents example of context free languages with inherent degree of ambiguity d_a for each $d_a \in \mathbb{N}$.

The application areas where we can use context free grammar typically need a unique structure for each sentence that a grammar generates. The context free grammars are mainly used to design compiler to check the signature/syntax of computer program. But the definition of context free grammar does allow for the possibility of having more than one structure for a given sentence, this problem, known as ambiguity. The ambiguity problem is a serious problem and may make the meaning of a sentence unclear.

Aim of the paper: The aim of this paper is to project the work conducted by the author on major grammars along with detailed descriptions about ambiguity and ambiguity detection methods.

***Outline of the paper*:** This paper shows the related work done so far in the studied area. It also explains the problem of ambiguity in learning grammar or languages along with details about the degree of ambiguity. The author has presented a case study that is the method studied in this paper. A detailed comparison of studied method with other existing methods has also been given. Further, experimental setup, results, analysis, observations, and conclusions are given. Lastly, a reference section has been presented.

RELATED WORK

Saul Gorn [4] described a Turing Machine to generate all possible strings of a grammar. Searching (string generated before or not generated) starts after generation of each new string. If the string exists before then the string

has multiple derivations, the grammar is ambiguous. The searching process for the string is a simple Breadth First Search.

In [5] Cheung and Uzgalis give a searching method with pruning of all possible derivations of a grammar. This method is nothing but an optimization of [4]. In addition to [4], this method can also detect non-ambiguity for some languages with an infinite language.

Schroer [7] developed an ambiguity detection tool "AMBER". To generate all possible strings and for finding the duplicates it uses Earley parser. The way of derivation of a string of a grammar is similar to the methodology given in [4] except for some variations in the search method which helps in improving the search timing and comparing the strings.

LR (k) parsing algorithms described by Knuth [9] makes decisions based on k-input symbols of look-ahead. It is based on the bottom-up parsing technique. LR (k) grammars can be deterministically parsed using this algorithm. For applying this method, a parse table has to be maintained which helps in identifying the action (shift, reduce, accept or error) to be performed. LR (k) testing is very simple for detecting ambiguity: if the parse table contains no conflicts then the grammar is LR (k). Similarly, the test can be used for ambiguity detection.

The approach presented by Brabrand, Giegerich, and Møller [10] is a new approach for detecting ambiguity. They divided the ambiguity detection problem into two sets of problems of similar types called as Vertical and Horizontal ambiguity. In this method, languages of the productions are approximated to make the intersection and overlap problem decidable. By approximating, we can represent the language into regular grammar; this process is called as conservative approximation. By doing this we can compute the regular approximations of vertical and horizontal ambiguity.

INTRODUCTION TO AMBIGUITY

This section covers detailed description about the Ambiguity, Degree of Ambiguity, and Detection of Ambiguity.

Context Free Grammar

In formal language theory, a context-free grammar (CFG) is a grammar in which every production rule is of the form [12]:

$$\alpha \rightarrow \beta \qquad \textbf{(1)}$$

Where,

α = single non-terminal symbol, and
β = string of terminals and/or non-terminals (possibly empty)

What is Ambiguity?

There is no bijective relation between input words and parse tree. While every parse tree corresponds to a single word, the reverse does not hold necessarily. A Context Free Grammar where a word with more than one parse tree exists is called ambiguous.

An example of a grammar with more than one parse tree is Grammar [2] as given below:

$$E \rightarrow E + E / E * E / id \qquad \textbf{(2)}$$

It recognizes the word $id + id * id$, which has two different leftmost (rightmost) derivations and parse trees.

$$\begin{aligned} E &\Rightarrow_{LM} E + E \\ E &\Rightarrow_{LM} id + E \\ E &\Rightarrow_{LM} id + E * E \\ E &\Rightarrow_{LM} id + id * E \\ E &\Rightarrow_{LM} id + id * id \end{aligned} \qquad \textbf{(3)}$$

Another leftmost derivation:

$$\begin{aligned} E &\Rightarrow_{LM} E * E \\ E &\Rightarrow_{LM} E + E * E \\ E &\Rightarrow_{LM} id + E * E \\ E &\Rightarrow_{LM} id + id * E \\ E &\Rightarrow_{LM} id + id * id \end{aligned} \qquad \textbf{(4)}$$

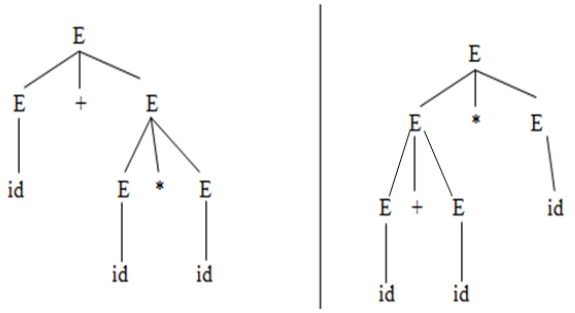


Figure 1: Two Parse Trees of the Word "id + id * id" using Grammar (2) and Derivations (3 and 4).

Now, having seen Figure-1, formally we can define Ambiguous Context Free Grammar as [1]: A Context Free Grammar $G = (V, \sum, P, S)$ is ambiguous; *iff* a word $w \in \sum^*$ exists, which has multiple leftmost derivations $S \Rightarrow^*_{LM} w \in G$.

For the derivation (2) and (3), the leftmost derivation operation has been used, using rightmost derivation or parse tree instead result in equivalent definition. We can call a grammar unambiguous if it is not ambiguous.

Degree of Ambiguity

The degree of ambiguity is defined as the total number of different parse tree of a string w. For example degree of ambiguity for Grammar (1) is 2 for left most derivation (2) and (3) for *w = id + id * id*.

We can classify ambiguity in grammars using the degree of ambiguity. If the number of distinct parse trees for each string increases with the length of strings generated by a grammar, it is possible that the degree of ambiguity of that grammar is infinite.

STUDIED ALGORITHMS: CASE STUDY

Jampana [8] presented an algorithm to detect ambiguity. This method is based on the observation that "non-trivial grammars define non-finite languages in a recursive manner. Based on this observation this method divide sentential form of a grammar into two groups: those with one or more repetition of one or more patterns and the rest of the strings. For the first set it has been assumed that all ambiguities of a grammar in Chomsky Normal Form (CNF) will occur in derivations in which every live production (A→BC) is used at most once. The assumption about the repletion of the application of production is incorrect in this method which leads to failing to search string with repetitive derivations will leave ambiguities undetected hence this algorithm might report false negatives.

COMPARISON OF METHODS

Table 1 shows the comparative study of all the ambiguity detection methods M1…..M6 studied in this paper under certain factors/properties.

Table 1: Comparative Summary of the existing Ambiguity Detection Methods

Methods / Factors	M1	M2	M3	M4	M5	M6
Termination	N	N	N	N	Y	Y
Ambiguity	Y	N	Y	Y	Y	Y
Unambiguity	N	Y	Y	N	Y	Y
False Positive	N	-	N	N	N	Y
False Negative	-	N	N	-	Y	N
Scalable	N	N	N	Y	N	Y
Optimize	N	N	Y*	Y	Y**	Y

M1: Gorn's Method M2: Knuth's Method M3: Cheung and Uzgalis Method
M4: Schorer's Method M5: Jampana's Method M6: Brabrand's, Giegrich's and Moller's Method
Y: Yes N: No Y*: Optimize to M1 Y**: Optimized but report false negative

EXPERIMENT AND RESULTS

This section is given for the implementation of the studied algorithms on various grammars. We tested the algorithm on a set of 84 grammars.

Test Data

The experiment for the study has been done on the set of various grammars listed in Table 2. For the test we included grammars of different sizes ambiguous grammars, unambiguous grammars, grammars that generate known inherently ambiguous languages and the grammars which derives empty strings hence there is no CNF. In this case study we took the set of 84 grammars and the distribution of varieties of grammar is shown in Table 2.

Table 2: Grammar Tested [3].

Grammar #1 S->AB S->a A->SB A->b B->BA B->a	**Grammar #2** E->E+E E->E*E E->(E) E->a	**Grammar #3** E->E*T E->T T->T*F T->F F->(E) F->a	**Grammar #4** S->AB S->C A->aAb A->ab B->cBd B->cd C->aCd C->aDd D->bDc D->bc	**Grammar #5** S->AB S->CD S->EF A->a B->b C->a D->b E->a F->b	**Grammar #6** S->ictS S->ictSeS S->a	**Grammar #7** S->M S->U M->ictMeM M->a U->ictS U->ictMeU	**Grammar #8** S->aSa S->bSb S->c
Grammar #9 S->AB S->ASB A->a B->b	**Grammar #10** S->AB S->CA A->a B->BC B->AB C->aB C->b B->b	**Grammar #11** S->AB S->BC A->BAD A->a B->CC B->bD C->AB C->c D->d	**Grammar #12** S->bA S->aB A->bAA A->aS A->a B->bBB B->b B->SB	**Grammar #13** S->AA S->a A->SS A->b	**Grammar #14** S->AB S->BC A->BA A->a B->CC B->b C->AB C->a	**Grammar #15** S->ABCD A->CS B->bD D->SB A->a B->b C->c D->d	**Grammar #16** S->N1V1 S->N2V2 N1->DN3S1 N2->DN4 V1->V3N2 V2->V4S1 V2->stink S1->CS D->the N3->fact N4->cats N4->dogs C->that V4->think V3->amazes V3->bothers
Grammar #17 S->N1V1 S->N2V2 N1->DN3S1 N2->DN4 V1->V3N2 V2->V4S1 V2->s S1->CS D->t N3->f N4->c N4->d C->h V4->i V3->a V3->b	**Grammar #18** S->PQ P->ROT P->a R->MP O->a O->ab T->b T->bb M->a Q->CeD C->a C->ab D->d D->ed	**Grammar #19** S->ABD S->ABC A->AE A->a B->SE B->b D->d C->c E->dc	**Grammar #20** S->BC S->AB A->CF C->c F->ged B->ab B->aC	**Grammar #21** S->U S->V U->TaU U->TaT V->TbV V->TbT T->aTbT T->bTaT T->	**Grammar #22** S->AA A->AAA A->bA A->Ab A->a	**Grammar #23** S->ACA A->aAa A->B A->C B->bB B->b C->cC C->c	**Grammar #24** T->ABC T->ABD A->AD A->AC A->a B->AB B->b C->c D->d

Table 2: Grammar Tested [3] (continued).

Grammar #25	**Grammar #26**	**Grammar #27**	**Grammar #28**	**Grammar #29**	**Grammar #30**	**Grammar #31**	**Grammar #32**
A->AS S->AB S->BB B->b A->bA A->a	S->AB B->bb B->bB A->a A->aAb	A->a A->B A->CA B->bD B->b D->d D->dD D->ad C->bc C->c C->CC	Z->aXY Z->bXZ Z->z X->aY X->az X->y Y->y	S->NVNDJ N->a N->h N->PJN N->PN V->f V->e P->f P->p D->r D->v J->b J->g J->PJ	S->AB S->CA A->a B->BC B->AB C->aB C->b B->b	S->AB S->BC A->BAD A->a B->CC B->bD C->AB C->c D->d	S->bA S->aB A->bAA A->aS A->a A->bBB B->b B->SB
Grammar #33	**Grammar #34**	**Grammar #35**	**Grammar #36**	**Grammar #37**	**Grammar #38**	**Grammar #39**	**Grammar #40**
S->T S->a T->A T->b T->T A->ab A->aab	S->t S->e S->he S->S S->eH S->eHS S->H H->hH H->h H->ht	S->WCT W->while C->bconditionb T->bRb R->statement R->statementR	S->WCT W->w C->bcb T->bRb R->s R->sR	S->wbcbbRb R->s R->sR **END	S->SS S->AS S->a S->b A->AA A->AS A->a	S->b S->Tb S->TQ T->baT T->caT T->aT T->ba T->ca T->a Q->bc Q->bcQ Q->caQ Q->ca Q->a Q->aQ	T->rXr T->rXrT X->text X->C C->dtextd C->dtextdC
Grammar #41	**Grammar #42**	**Grammar #43**	**Grammar #44**	**Grammar #45**	**Grammar #46**	**Grammar #47**	**Grammar #48**
S->UVPO S->UVCPO U->house U->houseflies V->flies V->like C->like P->a P->the O->banana	S->MN S->PQ M->aM M->Mc M->b N->Nc N->bN N->c P->Pd P->cP P->d Q->ad	A->BDE B->cA B->c B->a D->cD D->d D->aB E->e E->de E->ce	S->SAB S->ASB S->b A->ab A->Ba B->b B->bB	L->AND L->GA A->a A->aA A->ab N->ab D->ba D->Da G->bG G->baG G->ba	S->NVN S->NVNVingN N->dogs N->NVingN V->eat	A->SB A->AS S->ab B->b B->bB B->SB	P->PRQ P->p R->pr R->r R->rR Q->q Q->qQ Q->rq
Grammar #49	**Grammar #50**	**Grammar #51**	**Grammar #52**	**Grammar #53**	**Grammar #54**	**Grammar #54**	**Grammar #55**
S->ABSB S->ASB S->a A->aA A->a B->ab B->AB B->b B->bB	S->AC S->BA A->Ba A->aB A->a C->CB C->c B->Bc B->b B->bc	S->E E->E+E E->a	S->A S->B A->a B->a	A->a A->b A->aA A->Ab	A->B A->C A->BC B->a B->aB C->b C->bC	A->B A->C A->BC B->a B->aB C->b C->bC	S->aSbSc S->aSb S->bSc S->d

Table 2: Grammar Tested [3] (continued).

Grammar #56 S->A S->B A->aA A->a B->Ba B->a	**Grammar #57** S->aSa S->a	**Grammar #58** S->AB A->aA A->B->bB B->	**Grammar #59** S->A S->B A->aA A->B->bB B->	**Grammar #60** S->N1 V S->S P N1->N2 N1->a N2 N1->N1 P N2->i N2->man N2->home P->at N1 V->see N1	**Grammar #61** S->A A->B A->BC B->b C->c C->	**Grammar #62** S->aLALa L-> L L->A->Ab A->	**Grammar #63** A->Ab A->BA B->BA A->a B->b
Grammar #64 S->aB S->bA A->a A->aS A->baA A->bbAAa B->b B->bS B->aBB	**Grammar #65** S->I S->A I->ifEthenS A->D=E E->D=D D->if D->then	**Grammar #66** S->E S->D E->I*I D->T*I I->a T->a	**Grammar #67** S->(S) S->.S S->S. S->SS S->	**Grammar #68** S->(P) S->.S S->S. S->SS S->P->(P) P->S	**Grammar #69** S->(S) S->.L S->R. S->LS L->(S) L->.L R->R. R->	**Grammar #70** S->.S S->T S->T->T. T->(S) T->T(S)	**Grammar #71** S->.S S->(S)S S->
Grammar #72 S->LS S->L L->(F) L->. F->(F) F->LS	**Grammar #73** S->(P) S->.L S->R. S->LS L->(P) L->.L R->R. R->P->(P) P->(N) N->.L N->R. N->LS	**Grammar #74** S->.S S->T S->T->T. T->(P) T->T(P) P->(P) P->(N) N->.S N->T. N->T(P)	**Grammar #75** S->AA A->xAx A->y	**Grammar #76** R->cRg R->gRc R->aRu R->uRa R->gRu R->uRg R->	**Grammar #77** P->(P) P->(O) O->LP O->PR O->SPS O->H L->.L L->. R->.R R->. S->.S S->. H->.H H->...	**Grammar #78** S->aBc S->aDd B->b D->b	**Grammar #79** S->aBc S->aDd S->Bd B->b D->b
Grammar #80 S->AC S->BCb A->a B->a C->cCb C->cb	**Grammar #81** S->aSa S->bSb S->a S->b S->	**Grammar #82** S->L=R S->R L->*R L->a R->L	**Grammar #83** S->aC S->aCb C->cCb C->cb	**Grammar #84** S->AB A->xa A->x B->ay B->y			

Results

In [8], different algorithms are presented by Jampana, to deal with ambiguity detection. For the experiments on a set of 84 grammars we tested two variations of algorithms showed the correct results. Table 3 shows the execution times of the implementations of the two variations of the presented algorithm.

Table-3: Execution Time (in seconds) for Studied Method to Determine Ambiguity.

Grammar Type	Ambiguous	Unambiguous
CNF Grammar (Sentences)	0.639067	16.8119
CNF Grammar (Sentential Forms)	0.0474333	10.9554

OBSERVATION

The following observation has been made after implementation of the algorithms and running it on a set of grammar given in Table 2. The author identified the following different sets of grammar listed in Table 4.

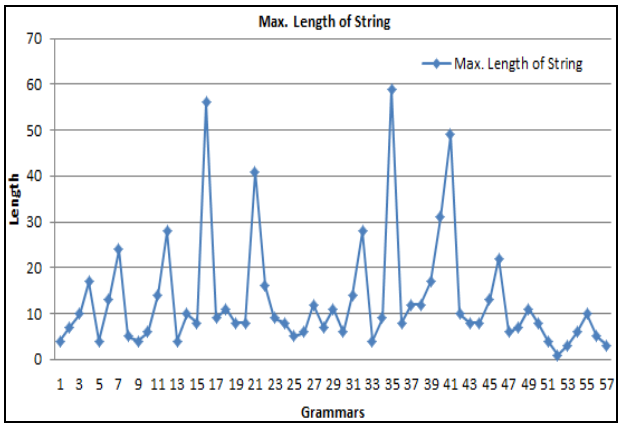
Table 4: Type of Grammar Used for Experiments.

Grammar Types	Total Number
Total Grammars	84
Ambiguous Grammars	30
Unambiguous Grammars	27
Grammar Derive Empty String	27

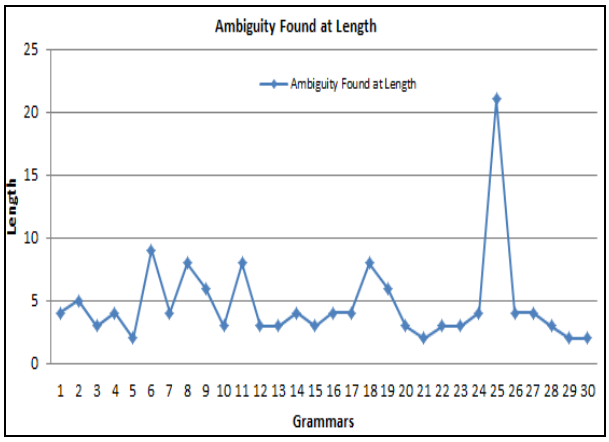
The author also identified that the time consumed for detecting ambiguity by a grammar in CNF is independent of the degree of ambiguity of the first ambiguous string. The productivity of the grammar was also independent in finding the ambiguous string. Table 2 (grammar-58 to 84) shows the grammar which derives an empty string and for these grammars the studied algorithm will not work because we cannot get the CNF form. The noticeable point of this method is that the process of ambiguity detection does not depend on the rate a grammar produces a string.

Using Table 1, we can understand that this method terminates in any case where CNF grammar with no duplicates is always finite. The assumption made for the algorithms for the repetition of production is incorrect. This approach fails in searching strings with repetitive derivation. Hence, there may be the possibility of ambiguity undetected. Therefore, the algorithm might report false negatives.

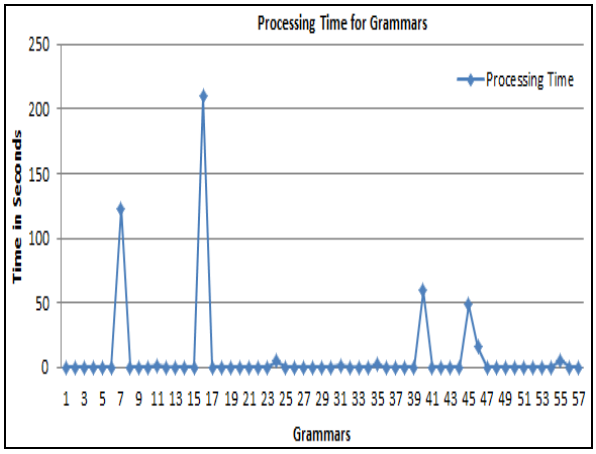
Table 3 shows that the execution time for detecting ambiguity and unambiguity in CNF grammar (sentential) is faster than the execution time by CNF grammar (sentence).



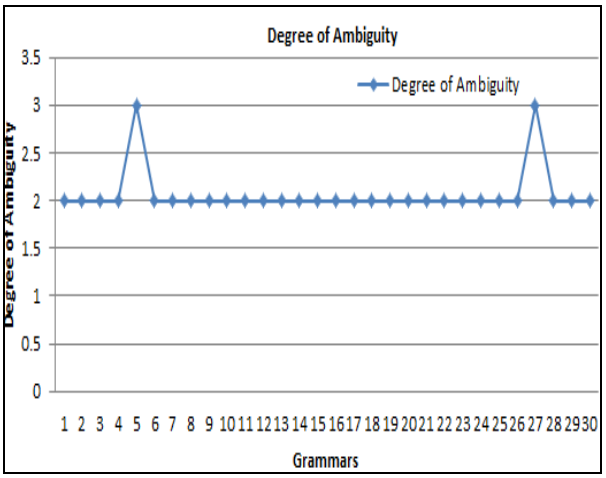
A) Chart for Max Length of String for Grammar.



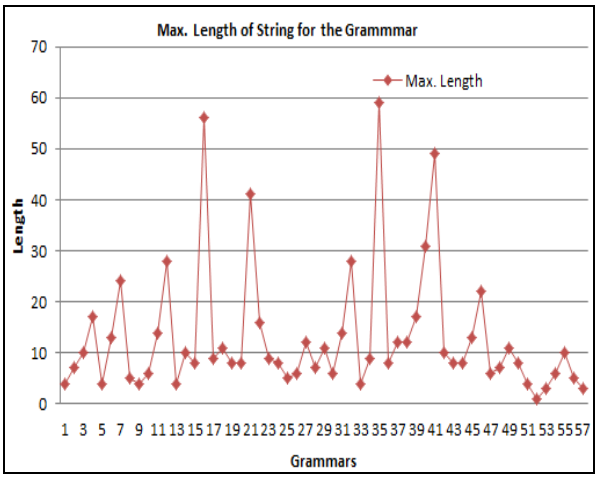
B) Chart for Ambiguity found at Length.



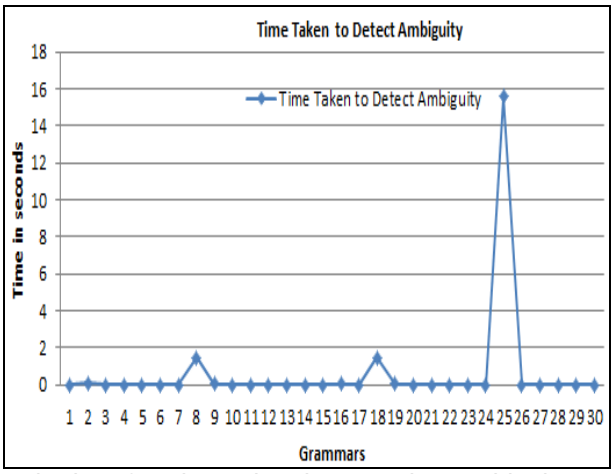
C) Chart for Processing Time for Grammars.



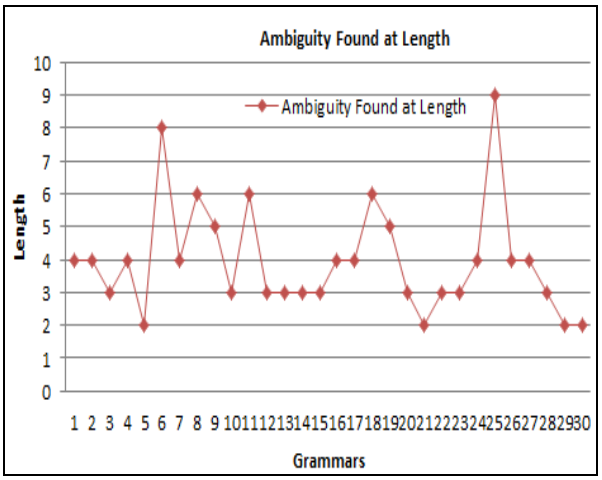
D) Chart for Degree of Ambiguity for Ambiguous Grammar.



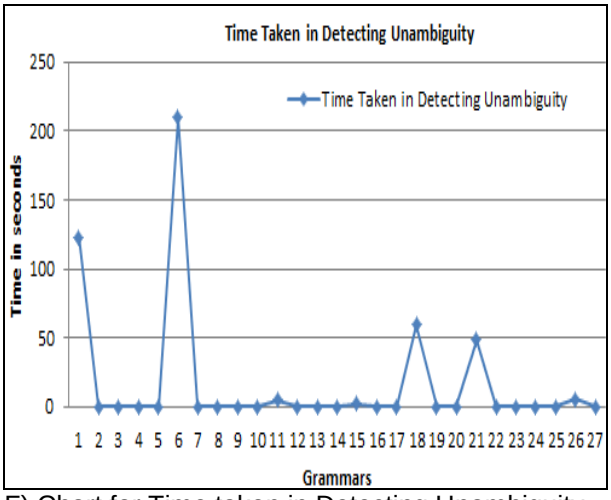
A) Chart for Max Length of String for Grammar.



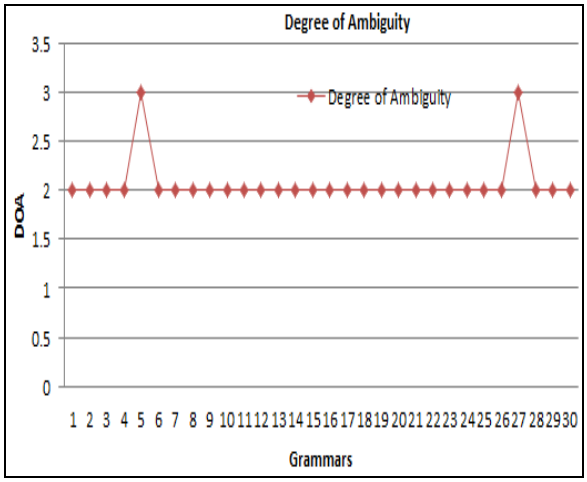
E) Chart for Time taken in Detecting Ambiguity.



B) Chart for Ambiguity found at Length.

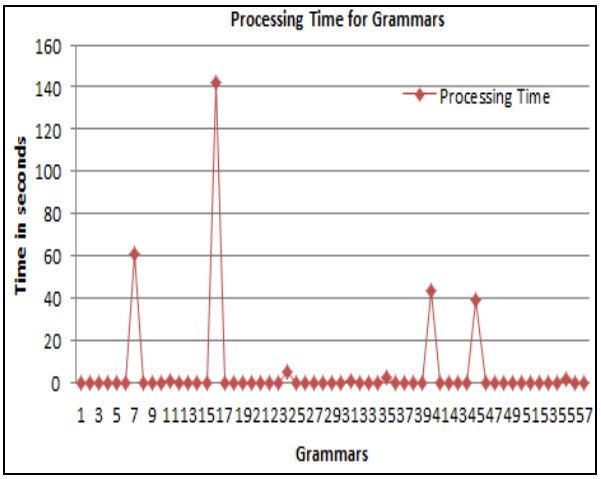


F) Chart for Time taken in Detecting Unambiguity.

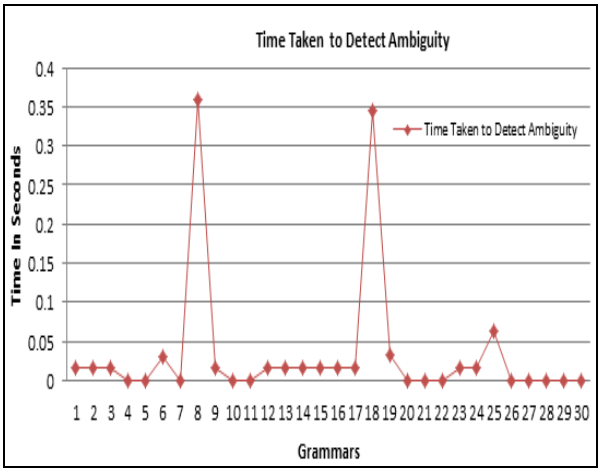


C) Chart for Degree of Ambiguity for Ambiguous Grammar.

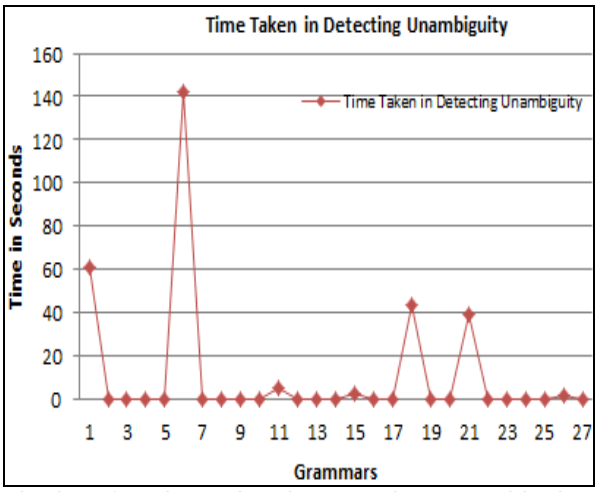
Figure 2: Analysis charts for CNF Grammar (Sentence) from (A) through (F).



D) Chart for Processing Time for Grammars.



E) Chart for Time taken in Detecting Ambiguity.



F) Chart for Time taken in Detecting Unambiguity.

Figure 3: CNF Grammar (Sentential Forms) from (A) through (F).

CONCLUSION

We have found very few articles for ambiguity detection from the formal grammars. This paper mainly covers the study of existing algorithms, their comparisons, degree of ambiguity, and other factors. The author tested one of the existing algorithms on set of 84 grammars such as ambiguous grammars, unambiguous grammars, and the grammar which derives empty strings.

While preparing the subject matter of the present paper, the author's fundamental objective was to show the detailed introduction about the active and challenging area grammars and ambiguity detection methods.

Some algorithms have been presented in the introduction section of this paper for detecting ambiguity of sentences. The author conducted tests on various grammars to detect ambiguous strings in context free grammars and the results are given. The targeted audiences are mainly beginners or someone who may be interested in starting research related activities in a similar field.

REFERENCES

1. Kruse, M. 2008. "Ambiguity Detection for Context-Free Grammars in Eli". Bachelor's Thesis, University of Paderborn: Germany.

2. Hopcroft, J.E., R. Motwani, and J.D. Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education Asia, Inc.: Delhi, India.

3. Basten, H.J.S. 2007. "Ambiguity Detection Methods for Context-Free Grammars". Master's Thesis. University of Amsterdam: The Netherlands.

4. Gorn, S. 1963. "Detection of Generative Ambiguities in Context-Free Mechanical Languages". *JACM*. 10(2):196–208. http://doi.acm.org/10.1145/321160.321168.

5. Cheung, B.S.N. and R.C. Uzgalis.1995. "Ambiguity in Context-Free Grammars". In" *SAC'95: Proceedings of the 1995 ACM Symposium on Applied Computing*. 272–276. ACM Press: New York, NY. http://doi.acm.org/10.1145/315891.315991.

6. Cheung, B.S.N. 1994. "A Theory of Automatic Language Acquisition". Ph.D. Thesis. University of Hong Kong: China.

7. Schr¨oer, F.W. 2001. “AMBER: An Ambiguity Checker for Context-Free Grammars”. Technical report, compilertools.net. http://accent.compilertools.net/Amber.html.

8. Jampana, S. 2005. “Exploring the Problem of Ambiguity in Context-Free Grammars”. Master’s Thesis. Oklahoma State University: Stillwater, OK. http://e-archive.library.okstate.edu/dissertations/AAI1427836/.

9. Knuth, D.E. 1965. “On the Translation of Languages from Left to Right”. *Information and Control*. 8(6):607–639.

10. Brabrand, C., R. Giegerich, and A. Møller. 2007. “Analyzing Ambiguity of Context Free Grammars”. In: M. Balik and J. Holub (eds.). *Proc. 12th International Conference on Implementation and Application of Automata, CIAA ’07*. July 2007. http://www.brics.dk/~brabrand/grambiguity/

11. Maurer, H. 1968. “The Existence of Context-Free Languages Which are Inherently Ambiguous of any Degree”. Department of Mathematics, Research Series. University of Calgary: Canada.

12. Pandey, H.M. 2010. “Context Free Grammar Induction Library Using Genetic Algorithms”. International Conference on Computer & Communication Technology ICCCT’10.

ABOUT THE AUTHOR

Hari Mohan Pandey, B.Tech. (Computer Science and Engineering), M.Tech. (Computer Engineering), was educated at U.P. Technical University, Lucknow and Narsee Monjee Institute of Management Studies, Mumbai, India. Presently, he is working as a faculty member in the Department of Computing at Middle East College of Information Technology. He is a life time member of the International Journal IJERIA (India), International Association of Computer Science and Information Technology (Singapore), and International Association of Engineers (IAENG, Hong Kong). He has written many books in the areas of Computer Science & Engineering for McGrawHill, Pearson Education, and University Science Press. Apart from writing books he has presented and published research papers at national and international conferences and journals. He is associated with various international journals as a reviewer/editorial board member. His research interests are in artificial intelligence, machine learning, evolutionary computing, natural language processing, principles of programming languages, and other related topics.

SUGGESTED CITATION

Pandey, H.M. 2012. “A Case Study on Ambiguity Detection Method Using Some Set of Grammars”. *Pacific Journal of Science and Technology*. 13(1):277-286.

Pacific Journal of Science and Technology