Journal of Applied Intelligent System (e-ISSN : 2502-9401 | p-ISSN : 2503-0493)
Vol. 2 No. , April 2017, pp. 37 – 51

GenTree A Tool For Analysis Of Syntactic Algorithms Based On Younger Cocke Kasami

Wijanarto*[1], Ajib Susanto[2], Desi Purwanti Kusumaningrum[3]
[1,2,3]Faculty of Computer Science, University of Dian Nuswantoro, Jl. Imam Bonjol 207, tel.(+6224)3517261/fax. (+6224)3569684
*e-mail: *[1]wijanarto@dsn.dinus.ac.id, [2]ajib.susanto@dsn.dinus.ac.id, [3]desi.purwanti.kusumaningrum@dsn.dinus.ac.id*
**Corresponding author*

Abstract - Analysis of syntactic is a series of processes in order to validate a string that is received by a language. Understand the process of decline of the rules to be a tree is the hard part. This paper presents the results of the design tools to automate an input string into a decline in the rule until the tree is in the visualize the image either in the form of a file or display, performance evaluation tools and analysis of understanding of students against a tool with algorithm Cocke Younger Kasami (CYK) was chosen as one of the cases for the technique of parsing in a Context Free Grammar (CFG) in the form of Chomsky Normal Form (CNF). The results of this study showed that the model was successfully implemented into the application named genTree (Tree Generator), the performance of the application obtained a significant number of measurements against complexity of the variations of the input string of grammar and 29.13% with the complexity of 7 and 8.50% with the complexities of 20, while for the length of the input string against the algorithm processing time can be a value of 3.3 and 66.98% 6.19% 29 and as well as , also obtained the distinction ability of the t-test on the student group control group experiments with value t calculate = 5.336 with df 74, p-value of 0.001, in extent significant 0.05% (5%). Also, trap at the increase in the percentage of correct answers amounted to 58% of the variation is difficult, 83% of the moderate and easy variation. Sebalikanya a decrease of 60% wrong answers on difficult variations, 100% variation being and 57% on easy variation. The last change occurred a decrease in the percentage of students who do not practice by 60% on difficult variations, 44% of medium and 13% variation on a variation of easy to conclude that the applications run efficiently and optimally but also can effectively improve the understanding of the students in be ajar CYK algorithm with automata.

Keywords —*Tools, Analysis, Syntax, Algorithms*, *Tree*

1. INTRODUCTION

Syntactic Analysis is a series of processes in order to validate a string that is received by a language. Linear representation of a rules or grammar is a technique that ultimately produces a tree can be represented in a language understood by the computer. The theory of language and automata (TBO) it is a part which was instrumental in engineering computation on the current computer. On the other hand the theory of languages and automata difficult to grasp manually because the algorithms and approaches are very complicated to study [1]. Grammar is part of the theory of languages and automata, which are represented in linear and is known by the term parsing, which results in a tree (*parse tree*) for the analysis of syntactic [2, 3, 4, 5]. This technique will be the focus for implemented in a paper, with the model of the language into a form of Chomsky normal form [6]. In addition to known very complicated and slow Its development, both in terms of algorithms, theory as well as implementation, the existence of TBO is still very central in the computer science, because without it the computer can't be developed as currently.

Theory and language automata are the science about device software computer or abstract machines [5], which useful and great role in applied natural sciences [7]. Within a decade, the TBO much useful in particular from mathematics, Although not exclusive as in mathematical physics or applied mathematics. In Science Math, he serves an important part of the functional aspects, the concept of *finite automata* and *formal grammar* used in design and software construction. The main concept of the generator is producing a way of automation on a method or model, and generators are in the intent in this paper was routine or special modules that can be used to control the behavior of the loop or iteration, the translation process is the input of a code or text into a particular good that can be executed or not [2, 6], through the analysis of the lexical and syntactic, generated by a grammar. Grammar is a contextual rule syntax with semantics contained therein of a formal language. The syntax that is used in the paper is the *Chomsky-Norm Form* (CNF) of *a Context Free Grammar* (CFG) [6, 3, 8], where there is a set amount of *terminal symbols*, *the non-terminal*, *early* and *production rules* $N \rightarrow \alpha \mid \beta$, with **N** is *a non-terminal* , $\rightarrow$ means *consisting of a* and **a** is *a string of terminal* or *non-terminal* that may be empty and symbols **|** defined as the choice or alternatively, the set of this called a *context-free grammar*, the *grammar* in a nutshell. Language or grammar determine abstraction *syntactic* in a set of *Abstract Syntax Tree* (AST), each a nonterminal symbol from AST label rules applicable product and grammar do not produce a sentence to a terminal symbol not instrumental in abstraction *syntactic*. In General, the level of grammar can classify into4 (four) according to production rules in *Chomsky-hierarchy of grammar* [4], and this paper using CFG type 2 (two). Production rules in the CFG are on the left side there should only be one non-terminal symbol as follows below.

$$S \rightarrow AB$$
$$A \rightarrow AA \mid a$$
$$B \rightarrow BB \mid b$$

Figure 1. Production rules in the CFG

The formal definition is important in TBO there are 3, *the alphabet* (set symbol), *string* (sequence of symbols from the set of the alphabet), and *the language* (a set of strings) [3, 5, 6]. The alphabet is the set of symbols are limited and are not empty. In General, the notation used is "Σ" (sigma), which symbolizes an alphabet. The string is a set of symbols is limited from some alphabet. The notation of string written with "w" while the length of the string notation with with "| w |". Language is a collection of strings that have no rules (Σ *) and symbolized by "L". Furthermore, another formal definition used in this paper is *parsing* i.e. process analyst with syntactic of input data, which is provided as a text, for determines the structure of a *grammar* [9]. *The output* from the process of *parsing* i.e. to answer whether string text it contains the given language that describes a specific *grammar*. The additional *Output* resulting from the process of *parsing* is *parsing tree,* the representation of the shape of the tree *(tree)* of the process of decline *(derivation)*[4, 10]. P*arsing tree* perhaps more important is used in *compilers*, where data structure the represents the translation of *source* program into executable code by following the natural functions and recursive to support the process of translation, as Figure 2 below.

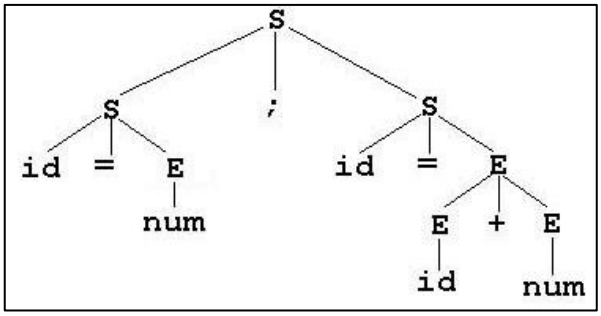

Figure 2: Parsing Tree

There are two techniques for conducting the process of *parsing*, *top-down parsing* and *bottom-up parsing* [6]. *Top-down parsing* is a process of formation of the *parse tree* starting from the root *(root)* headed to the leaves *(leaf)*. *Bottom-up parsing* is the process of formation of the *parse tree* starting from the leaves *(leaf)* headed to the root *(root)*. While in the process of decline *(derivation)* there are two approaches, namely *leftmost derivation* and *rightmost derivation* [6, 10, 11].

This Paper focus on *Cocke Younger Kasami (CYK) algorithm that* uses grammatical *Context Free Grammar*. The weakness of the algorithm of *Cocke Younger Kasami (CYK)* is in the language, must be in the form of *Chomsky Normal Form (CNF)*, there is no empty strings, production units, and the production of *useless* [6]. But this issue not become obstacles due to the *ContextFree Grammar* can be transformed into a form already *Chomsky Normal Form (CNF)* [4]. The algorithm of *Cocke Younger Kasami (CYK)* represents the structure of the data table *(array)* or 2 dimensional matrix shaped triangles, where each *entry* hold results *parsing* for a while, then will be used for the next stage until all the inputted string have been completed [12, 13, 14, 15]. The following is *pseudocode for* the algorithm of *Cocke Younger Kasami (CYK),*

```
input:G=(N,Σ,S,→)dalamCNF, string w=a1..anϵΣ+
CYK(G , w) =
1  for i=1…n do
2    Ti,i := {Aϵ N | A → ai}
3  for j=2…n do
4    for i=j-1…1 do
5       Ti,j:=∅;
5       for h=i…j-1 do
6          for all A → BC
7              if B ∈ Ti,h and C ∈ Th+1,j then
8                 Ti,j:=Ti,j∪ {A}
10 if S ∈ T1,n then return yes else return no
```

Figure 3: Algorithm Cocke-Younger-Kasami (CYK)

Figure 3 can be explained as follows, input grammar ***G*** in the form of CNF and a string *w* top terminal *alphabet* of *G*, its output in the form of a table *T* that contains substring every *v* of strings *w*, the set of non-terminal derivatives of *v*, for example, the property syntactic. At any given moment *T* showed us that *w* is the string from *G*. Thus, using the notation in the show in Figure 3, which shows a decrease of the nonterminal from a substring *with* $a_i r...a_j$ from the *w* at the store in T_{aj}. Syntactic properties *of w* can be calculated from the whole of the property syntactic a substring of the string *v w* with dynamic programming approach. By storing the non-terminals that scaled back by the *v* in the table T_v and counting $T_w o$ with combined entries stored on T_v according the rules of *G* and all possibility of separation from *w* into *m* which is a substring *of v*, where *m* is the maximum number of characters on the right side of the rules of grammar.

Thus given the complexity of the analysis of the syntactic subset with the TBO on the CYK algorithm, then this paper proposes a model of generator to generate the tree from the input string and the CFG in the form CNF automatically, then realizing the model into a tool that can generate a decline rules and trees in the form of textual and graphic, as well as measuring the performance of these tools by providing a variation of input strings and grammar to see the accuracy of the length of the string the complexities of grammar, and the processing time to generate output that is in want, and analysis of the level of understanding by doing a trial against a second-year student at the Faculty of computer science.

2. RESEARCH METHOD

The main objective of this paper is the *First*, making the model and framework of thought to produce tools that generates tree and decline rules automatically based on input grammar and CYK algorithms with string, *Second*, methods development system used to build models and software architecture involves the structure of the high level of abstraction of system software, by using decomposition and composition, with the attributes of the style and quality of architecture. A software architecture design must comply with the functions and performance of the main requirements of the system, as well as meet the non-functional requirements such as reliability, scalability, portability, and availability [16], which is implemented with the Unified Modeling Language (UML) and the tool will generate in the made with java via eclipse IDE (http://www.eclipse.org/) and the dot [17], as well as Application Programming Interface (API) GraphViz [18] to connect the dot with java. *Third,* this paper also attempted, will evaluate the performance of the resulting tools and test with the variation of the input string and grammar with particular complexity and timescale process. *The fourth*, to gauge the level of understanding of students toward mastery of the material of the lecture automata and language theory section of the CYK algorithm, using the method of experimental, which is in use to evaluate whether application implementation results improve understanding for students who use it. Evaluation methods in the implementation of selected is a method of experimental with the design of a control group without protest (*Posttest Only with Control Group*) of the sample population of the two groups (experiment and control) each of 40 second-year student of computer science faculty University Dian Nuswantoro, Semarang.

2.1. Model and framework of Thought

The standard model of grammar notation in the paper is the main frame of the application that will be generated. A notation in the model select model notation is Chomsky Norm Form (CNF) [9]. In general, the model architecture of grammar in use are as in Figure 4 as follows,

Figure 4. Model Tree Generators

Model is composed of three steps namely Notation CNF and string input, which consists of grammar (context-free grammar) which is already in the form of the input string and the CNF can be received by grammar, both is the input that will be processed by tree generator and will produce trees, both visually (image/*.png) and textual (*.txt) and the source code tree for can be customized in the format *.dot . From the General model like Figure 4 above, may be more developed in never achieved within the framework of thought in Figure 5 below:

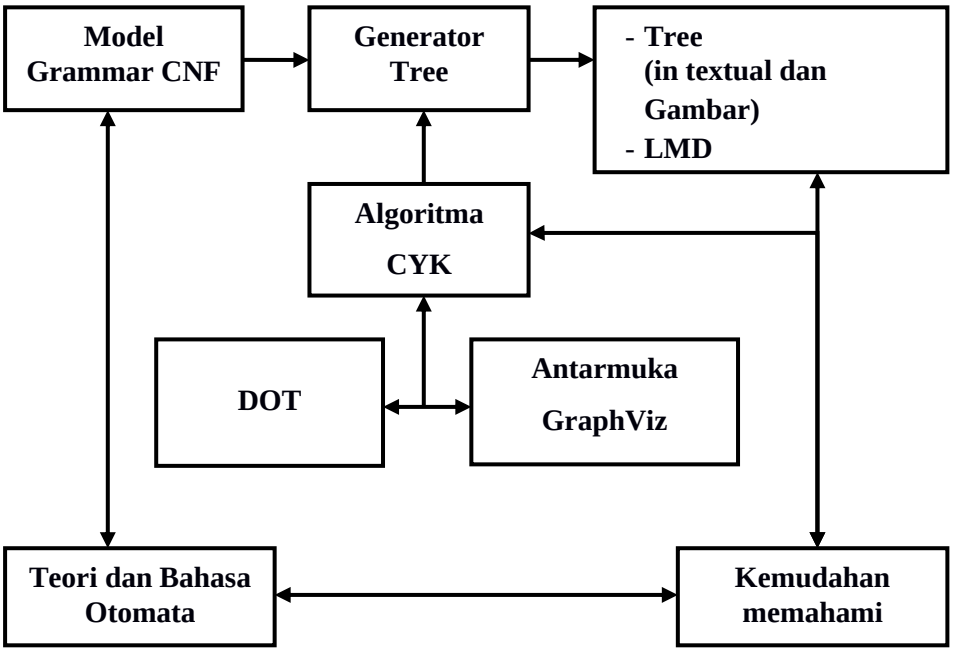

Figure 5. The Framework Of Thought Tree Generator

The framework of thought above in Figure 5 describes the method or framework of thought that will be in use in resolving problems as well as how to conduct an evaluation of the level of understanding the user (in this case a second year student of the Faculty of computer science), against subjects automata and language theory, with the case of Cocke Younger Kasumi algorithm in terms of decrease in the rules, the determination of the abstract tree syntax both visually as well as textual.

2.2. System Development Method

Applications system developed with a method or a diverse way, this research will use the approach in developing applications that are Rapid Application Development (RAD), the harness approach the object-oriented programming with the implementation using the Unified Modeling Language. Besides, because of its convenience, this technique is also very fast in building medium-scale systems. As for the design of architecture in general like the picture 6 as follows,

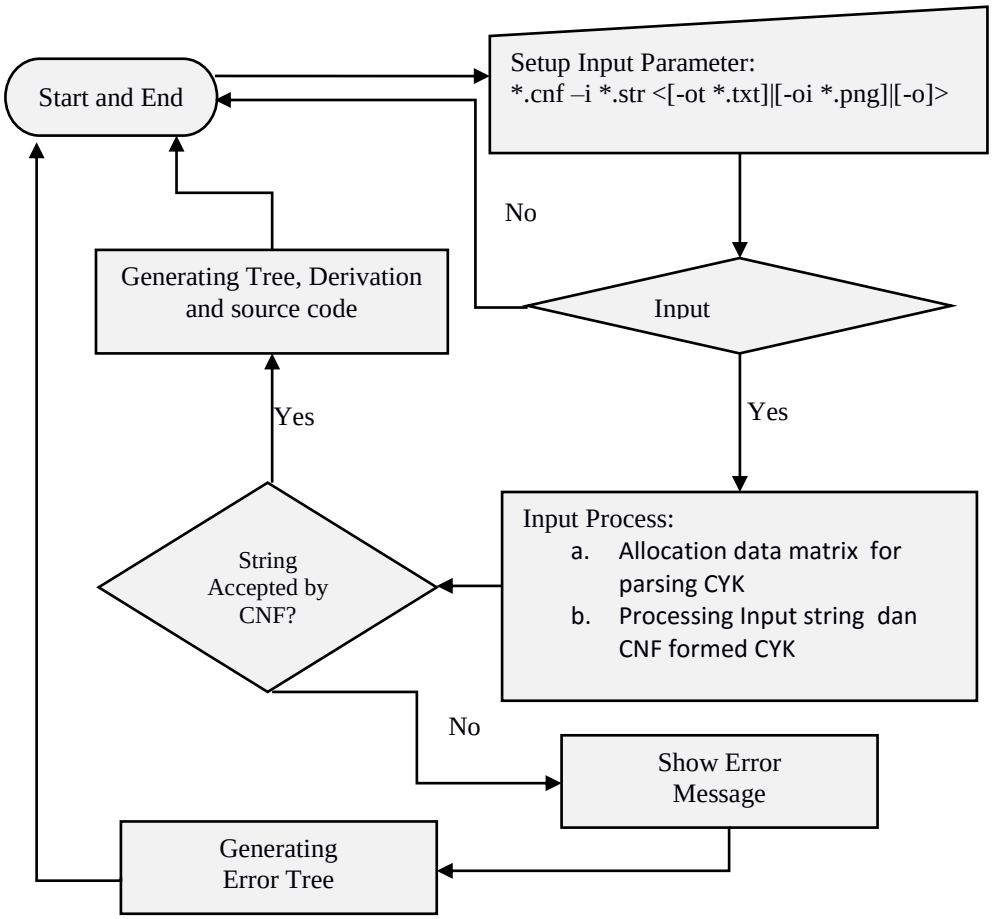

Figure 6. Tree Generator Architecture

Figure 6 can be described that the input in the form of file grammar in *Chomsky Norm Form* and input strings (*.cnf, *.str) will be in the read or in the CYK algorithm with parsing. If all the input does not match (input validation) then the program finishes. If input validation passes then the next step is to allocate the matrix or CYK algorithm according to the table. After the input string in check and accepted by the grammar then it will generate a tree, rules in decline and the source code for the visual tree. If the input string not in receipt by the grammar, then an error message will be displayed and will remain in generate tree contained these errors.

2.3. Performance evaluation methods

To evaluate the performance of an application that will do an experiment to measure the performance of your application in this regard how accurate and fast application can handle variations in two primary inputs (the input string and grammar) in such a way so as to produce valid output with processing time. To do this, then created a grammar complexity classification specified on the basis of table 1 below.

Table 1. Criteria and the complexity of grammar

The complexity of the	**The characteristics of the**	**Description**
Low	rule ≤ 5; psi ≤ 5	Low complexity means "**easy**"
Medium	5 ≤ rule ≤ 15; psi ≤ 10	Medium complexity means "**medium**"
High	5 ≤ rule ≤ ∞; psi ≥ 10	High complexity means "**Hard**"

While the number of input strings varies as in table 2 below,

Table 2. The variation of the input string

Variation	Length | w |	Description
Easy	3 ≤ **|w|** ≤ 10;	The input string with easy variations in length at least 3 or at most 10
Medium	5 ≤ **|w|** ≤ 15;	The input string with easy variations in length at least 3 or at most 15
Hard	5 ≤ **|w|** ≤ ∞;	The input string with hard variations in length at least 3 or more

The purpose of the evaluation with the above variation is to know the performance of applications based on the complexity of grammar, length input and speed the process, which is a performance API Kasi in General.

2.4. Analysis method of understanding

First in the analysis, did to find quantitative data through experimentation, trial results using design of experiments without protest (*posttest only with control group*) [19], because of its convenience in giving special treatment against experiments with test generator tree to work on problems in supply and control the other groups, as in the following table 3 design,

Table 3. The subjects of the experiment and control Groups Randomly

The subject of the	Group	Variable bound	Posttest
A	Experiment	X	Ye
A	Control	-	Yk

Table 2 will be assigning each subject group of experimental and control group at random (A). Then just carry out the treatment on experimental group (mark X) whereas in the control group were not in treatment (sign), and then will carry out experimental group posttest (Ye) and control group (Yk), to determine the difference in the average value in the earn on Ye and Yk with statistical methods (test T), so it can determine the significance of the difference in the two groups. *Second*, simpler method, in used to support the method [20] before, by measuring student comprehension of the scale with variations of low, medium, high in answering the question that was given as part of the experiment, as in table 4 below,

Table 4. The student's level of understanding Scale

Level Of Understanding	The Percentage Of The Number Of Students Who Answered Correctly
High	76%-100%
Medium	60%-75%
Low	0%-59%

To get the value in the scale above it needs to be done for Computing Group based on your answers, the correct answer being students wrong or not answered (absent) in the form of percentages, with the formula in table 5 below,

Table 5. Formula calculations percentage

The Percentage Of Answers	The formula	Description
Correct (P_b)	$\frac{X_b}{N}X100\%$	X_b: the frequency of Students answered

Wrong (P_s)	$\frac{X_s}{N}X100\%$	Correctly X_s: the frequency of Students answered Wrong X_a: the frequency of Students answered Absent N: Total sample of students
Absent (P_a)	$\frac{X_a}{N}X100\%$	

3. RESULTS AND DISCUSSION

3.1.Generator Tree

Based on the model and framework of thought that exist, it can generate in a system application generator tree which is the implementation of architectural on the previous figure 6, where generator tree is actually a kernel from the model as the main processor, which consists of 5 blocks are interconnected with one another, the application interface, GraphViz DOT, CYK Algorithm, Tree Data structure and Error correction of Grammar. The output of the model form 4 file that consists of a text file (*.txt) and the image (*.png) to store the tree in produce, text file (*.lmd) to store the tree decline in the order of *leftmost derivation,* as well as the source code files from the image file (*.dot) that can be customized later if in need. The following figure 7, which is the *building block* of the kernel from the model tree generators are in the box with a red outline.

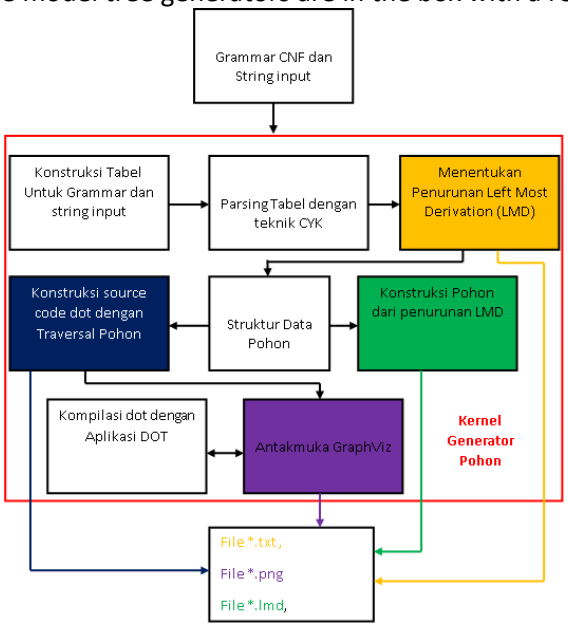

Figure 7. The Kernel Of Generator Tree

Based on the above, the kernel then obtained as a result of implementation in the form of diagram *Use-case* System Tree Generator is an application in the name as ***genTree*** which stands for Generator Tree, and Figure 8 below is the main model of system ***genTree*** built in the form of a use case diagram.

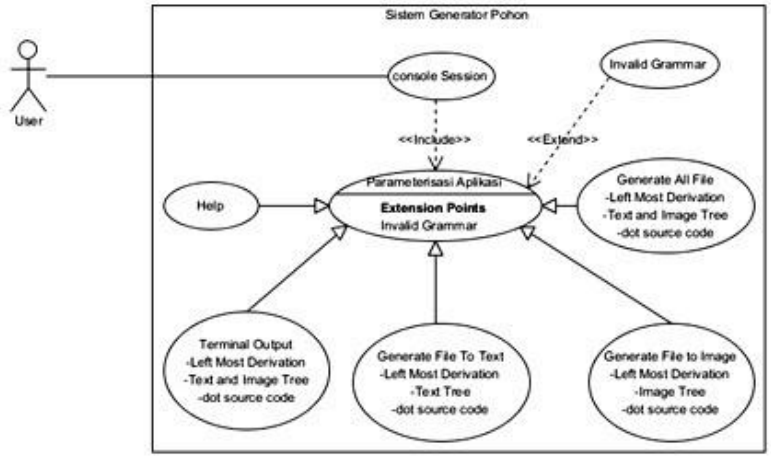

Figure 8. Use Case of Generator Phone

The result of the design of the application will be in the mentioned above in this section. **First**, the draft was successfully implemented dang terminal-based applications generate a console with the name **genTree. jar** which can be run as shown in Figure 9 below, with output as desired.

(a) Perintah dan input genTree.jar (b) Output di terminal konsol

(c)

Figure 9. Example of a genTree.jar application

In Figure 9 is an example of a genTree.jar application usage can be explained as follows, images (a), is an example of usage of **genTree.jar** with two *input* parameters grammar *mudah1.cfg* and input the string *imudah11.str*, other parameters namely – *c* the intended will issue *the output* in the terminal console like the picture (b) and window and display an image (c).

3.2.Performance Analysis

While the analysis of the performance of the application tested based on variations of the input with **the complexity of** grammar refers to table 1 and 2 which consists of, **easy**, **medium** and **difficult** to retrieved **the processing time** of the application for each input as in table 6 below.

Table 6. The Variation And Complexity Of Input

Grammar			
Variation	**G1**	**G2**	**G3**
Easy	S -> BS | DS | a | b. A -> b. B -> SA. C -> c. D -> SC.	S -> AB. A -> CA | a. B -> DB | b. C -> a. D -> b.	S -> AB | BC. A -> BA | a. B -> CC | b. C -> AB | a.
Input String	aca aba abb	aab aabb aaaaabbbbb	Bbab babaa aabaabab
Medium	S -> CB | DA. A -> a | CS | EA. B -> b | DS | FB. C -> a. D -> b. E -> DA. F -> CB.	S -> ED | CB | DA. A -> a | CS | EA. B -> b | DS | FB. C -> a | DC | FD. D -> b | ED. E -> c. F -> DA. G -> CB. H -> ED.	S -> XY | a | b. T -> AB | XB. X -> YS | a | c. Y -> SX | b | c. A -> a. B -> b.
Input String	aaabbabbba aaabbbaaaaabbbbb abaaaaabbbbb	cbaba ccccbaba ccccbbababa	abbaaccbb cccbab aaabcab
Hard	S -> EF | AF | EB | AB. X -> AY | BY | a | b. Y -> AY | BY | a | b | c. E -> AX. F -> BX. A -> a. B -> b.	S -> a | EF | AF | EB | AB | AS. X -> BY | SB | SY | a | b. Y -> AY | BS | a | b | c. E -> AX. F -> BX. A -> a. B -> b.	S -> AB | AA | AS | AD | b. A -> CC | a | c. B -> BC | b. C -> CB | BA | c. D -> SA | AS | a.
Input String	aaabbbb abaab ababbbb	aabcb aaaababababab aaaaaacbcbcbcb	abbc aabbaac abbbaaac

Performance test done by using the following hardware and software, hardware in the form of an Intel Pentium processor Dual Core E5500 2.80 GHz CPU, 4 GB RAM, 1 TB hard drive and software in the form of operating system Windows 7 Professional 32 bit, JDK 1.1, Eclipse Kepler, can describe as in table 7 is the result of research by using a variation of the grammar and input as shown in the previous table 1.

Table 7. The processing time to the length of the input string

Grammar								$\bar{t}$	$\bar{s}$	$\bar{t}/\bar{s}$
Variasi		**G1**	**|s|**	**G2**	**|s|**	**G3**	**|s|**			
Easy	1	2.16	3	2.723	3	1.815	4	2.23	3.3	66.98%
	2	1.851	3	2.191	4	2.076	5	2.04	4.0	50.98%

	3	2.032	3	2.28	10	2.303	8	2.21	7.0	31.50%
Medium	1	2.356	10	2.293	5	2.979	9	2.54	8.0	31.78%
	2	1.841	16	1.575	8	1.95	6	1.79	10.0	17.89%
	3	1.653	12	1.778	11	1.638	7	1.69	10.0	16.90%
Hard	1	1.56	7	1.498	5	1.248	4	1.44	5.3	26.91%
	2	1.591	5	2.231	11	1.279	7	1.70	7.7	22.18%
	3	1.934	7	2.184	14	1.263	8	1.79	29.0	6.19%

There are three grammar which in tested are G1, G2, G3 with each input string test three variations so that there are 9 (nine) variations of the test input strings with length (***/ s /***) also varies. From the input grammar and test against the input string on the overall variation obtained interesting facts that to turn out for the input grammar *is easy* with the mean of the input string (character variation) 2.1 ***easy 1***, in execution with the average processing time **( )** 2.23, while for grammar is difficult with the average length of the input string 29.0 variation ***3 is hard***, it can be executed with a speed of 1.79 seconds. The above indicates that the input string length is inversely proportional to the length of time the process for work done by algorithms based on the CYK variations. In addition, the average time was late in getting by the variation ***medium 1*** of 2.54 seconds and fastest gain variations ***hard 1*** with a time of 1.44 sec. So in general it can be said that the average processing time is precisely the algorithm can run fast and can be concluded while the smaller percentage comparison of average processing time (the average length of the input) with a string **( )**, in the lowest percentage, gained 6.19% which means the algorithm can process variation of grammar with the variation ***is hard*** and the input string is the longest and highest of 66.98% with ***easy*** .

Meanwhile in the analysis based on the complexity of the Grammarly a, as in the present in table 8 below, at the got the mere fact that a comparison between the average length of time the process ***(t)*** to the amount of average complexity number rule ***(r)*** and production rule option ***(op)***, thus **c = r + op** as at present in table 7 as follows below :

Table 8. Time the process of with the complexity of grammar

Grammar						
Variation					**c**	
		$\bar{t}$	**|r|**	**|op|**	**|r|+|op|**	$\bar{t}/c$
Easy	1	2.23	5	3	8	27.91%
	2	2.04	5	2	7	29.13%
	3	2.21	4	4	8	27.56%
Medium	1	2.54	7	5	12	21.19%
	2	1.79	9	9	18	9.94%
	3	1.69	6	7	13	13.00%
Hard	1	1.44	7	10	17	8.44%
	2	1.70	7	13	20	8.50%
	3	1.79	5	11	16	11.21%

Table 8, can be read as follows, the speed of the process (t) are in variation ***1 hard*** 1.44 sec, with time to work on the complexity of 17, while the longest time on the variation ***medium 1*** of 2.54 seconds with the amount of complexity. In general the complexity of work done with the lowest percentage of 29.13% with great complexity of 7, while the highest variation in complexity of the difficult 2, by 20% 8.50 value worked out with, which means that the algorithm can handle the grammar with time as its complexity, the greater percentage the more appropriate algorithm in handling Grammarly a vice versa.

3.1. Analysis of Understanding

There are two analysis in this section as a result of the research in this paper, pour in the first, the analysis is done by looking at the difference in test results do with experimental techniques to figure out the understanding of students on courses with a subchapter TBO CYK algorithm in the get as in the mentioned above at the bottom here.

3.1.1. T-Test

The t-test to find out the difference in the ability to resolve problems in the syntactic analysis of algorithms based on variation of CYK have been made against the student experiments using tree generator with students who do not use this control is carried out three times on the basis of the level of complexity of the variations of the problem , easy, medium and hard. T-test after test the ability to troubleshoot analysis of syntactic on the algorithm of CYK control group and the experimental group performed to figure out the difference in the ability to resolve problems with analysis of syntactic on the algorithm of CYK students after being given the treatment.

The calculation results show that magnitude *t* count was 5.336 with *df*=74 obtained *p* of 0.001. *P* value less than 0.05 significance level ($p = 0.001 < 0.05$). Thus, the t-test results show that passes the ability to troubleshoot analysis of syntactic on the algorithm of CYK group control and experimental groups produce different data and there is a significant difference. The difference in the average score increase data the ability to troubleshoot analysis of syntactic on the algorithm of CYK control group and the experimental group performed to find out the score increases the ability to troubleshoot analysis of syntactic on the algorithm of CYK between experiments using generator tree with a control group who did not use tree generator. The results of calculation show that the average score on the abilities of the group control is 62.14 while in the group experiment was 77.38.

Thus, there is a difference in the average of the 15, showed a rise in score ability resolve problems analysis of syntactic on significant CYK algorithm between a group of experiments using generators and a control group who did not use tree generator. This indicates that after being given the treatment with the experimental group tree generator more increase when compared with the control group. Increased ability to resolve problems with analysis of syntactic on the experimental group of CYK algorithm shown their answers based on the analysis amplifier is understanding of the interpretation of this section.

3.1.2. Percentage Of Understanding

While the percentage of the level of understanding of students ' answers to the questions made in reference to table 6 with variations of complexity and input the string as shown in table 1 and 2, then gained the following results in table 9 below,

Table 9. Presentation answers the student control and experiment

Variation	Control			Experiment		
	Correct	Wrong	Absent	Correct	Wrong	Absent
Hard	14	10	15	24	6	9
	35%	25%	38%	60%	15%	23%
Medium	25	5	9	30	5	4
	63%	13%	23%	75%	13%	10%
Easy	24	7	8	34	4	1
	60%	18%	20%	85%	10%	3%

Table 9 can read as follows, there are two groups of subjects, namely the control groups of students (students learn CYK without tree generator) and the experimental groups of students (students who study CYK with tree generator). Test results against students control on the variation of the problem

is difficult, there are 14 students who can answer the question correctly, 10 and 15 abstentions from a total of students. While the question is being answered right there by 25, 5 and 9 abstentions, while for a matter easy of 24 right, 7 and 8 abstentions. Being on the Group experiment perilous (24.6 and 9), medium (30.5 and 4) and easy (34.4 and 1).

By calculating the difference between the results obtained in the student's answer against variations of questions made earlier, which is to be tested on students prior been a treatment (control) and after the students were given a treatment (experiment), then get the result as mentioned above on where in table 10 below.

Table 10. The change of Student Understanding

Variation	Correct	%	Wrong	%	Absent	%
Hard	10	58%	4	60%	6	60%
Medium	5	83%	0	100%	5	44%
Easy	10	83%	3	57%	7	13%

Imaginary distinction in the hard matter of variation of 10 (58%), 4 (60%) and 6 (60%) to *correct*, *wrong* and *absent*. While variations *are* there is a difference of 5 (83%), 0 (100%) and 5 (44%), while variations *easily* produce a difference of 10 (83%), 3 (57%) and 7 (13%). Need to take note that the difference in the answers *is true* to its **positive** meaning, there is **increasing,** while in the *wrong* and *absent* its nature **negative** means **decreasing**.

4. CONCLUSION

From exposure to problems, methods and results and analysis in the previous section in this paper the author while can conclude that:

1. The Model proved to be implemented and produce applications with efficient performance which the algorithm can work on variations of the complexity of grammar (29.13% with the complexity of 7 and 8.50% with the complexities of 20) and input strings, for a length of 3.3 and 66.98% and 6.19% 29 and, compared with the time of processing.
2. Based on analysis of T-Test score after test between the control group and the experimental group performed with the help of the program SPSS and the calculation of earned value t calculate = 5.336 with df 74, in extent significance 0.05% (5%). T-Test results after test CYK algorithm ability to resolve problems in the control group and experimental group produces the value p of 0.001. P value less than 0.05 significance level ($p = 0.001 < 0.05$), so you can conclude there is a significant difference in the students prior to using the application generator trees with after using it.
3. On the basis ability to troubleshoot syntactic analysis on the algorithm of CYK then it can be proved that there is an increase in the ability of 58% at the correct answer with hard variations, 83% of the moderate and easy variation of the problem. Whereas as a consequence happens a decrease of 60% of students work on hard variations, variations are 100 and 57% on easy variation. The last change occurred a decrease in the percentage of students who do not practice by 60% on difficult variations, 44% of medium and 13% variation on a variation of easy. So we can conclude that there is a significant change in the understanding of the views of the students answer presentation control of student experiments in working on the issue given a particular variation, the increase in the percentage of *correct* answers and a decrease in the percentage of answers are *wrong* or *absent.*

REFERENCES

[1] S. H. Wantah Satria, "Pembuatan Media Pembelajaran Untuk Proses Konversi Pada Finite Automata Berbasis Multimedia," *Jurnal Sarjana Teknik Informatika e-ISSN: 2338-5197,* vol. 1, no. 1, 2013.

[2] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation and Compiling, New York: Prentice Hall Englewood Cliffs, 1973.

[3] A. W. Appel and M. Ginsburg, Modern Compiler Implementation In C, New York: CAMBRIDGE UNIVERSITY PRESS, 1998.

[4] D. Grune and C. J. Jacobs, Parsing Techniques - A Practical Guide, New York: Springer, 2008.

[5] J. E. Hopcroft, R. Motwani and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, New York: Addison-Wesley, 2001.

[6] A. V. Aho, M. S. Lam, R. Sethi and J. D. Ullman, Compiler: Principles, Techniques, and Tools, New York: Pearson Educating Addison Wesly, 2007.

[7] J. V. Neumann, "The General and Logical Theory of Automata," *Cerebral Mechanisms in Behavior,* vol. 1, no. 51, pp. 288-326, 1951.

[8] D. A. Watt and D. F. Brown, Programming Language Processors in Java, Compiler and Intepreter., New York: Pearson Education, Addison Wesly, 2000.

[9] P. Skrzypczak, "Parallel Parsing of Context-Free Grammars," Master Thesis MCS, Blekinge Institute of Technology, Karlskorna, 2011.

[10] T. Parr, Language Implementation Patterns Create Your Own Domain-Specific and General Programming Languages, Raleigh, North Carolina Dallas, Texas: The Pragmatic Bookshelf, 2010.

[11] T. Parr and K. Fisher, "LL(*): the foundation of the ANTLR parser generator," *ACM SIGPLAN Notices - PLDI,* vol. 11, 2011.

[12] A. Shamshad , "CYK Algorithm," *International Journal of Scientific Research Engineering & Technology (IJSRET) ,* vol. 1, no. 5, pp. 1-4, 2012.

[13] J. Cocke and J. T. Schwartz, "Programming Languages And Their Compilers," Courant Institute of Mathematical Sciences, New York University, New York, April 1970.

[14] K. T and K. Torii, "A Syntax-Analysis Procedure For Unambiguous Context-Free Grammars," *Journal Of The Acm (JACM),* vol. 16, no. 3, 1969.

[15] D. M. Younger , "Recognition And Parsing Of Context-Free Languages In Time n3," *Information And Control,* vol. 10, pp. 189-208, 1967.

[16] I. Somerville, Software Engineering, Boston, Massachusetts.: Pearson Education, Addison-Wesly, 2011.

[17] E. Gansner, "Graphviz - Graph Visualization Software, Envisioning connections," AT&T Labs-Research, - - -. [Online]. Available: http://www.graphviz.org. [Accessed 2 April 2016].

[18] L. Szathmary, "GraphViz Java API," GraphViz Java API, 4 December 2003. [Online]. Available: https://github.com/jabbalaci/graphviz-java-api. [Accessed 2 April 2016].

[19] Sugiyono, Metode Penelitian Kuantitatif, Kualitatif dan R &F, Bandung: Alfabeta, 2010.

[20] A. Sudijono, Pengantar Statistik Pendidikan, Jakarta: Rajawali Press, 2010.

[21] M. Lange and H. Leiß, "To CNF or not to CNF? An Efficient Yet," *Informatica Didactica,* vol. 8, 2009.

[22] N. Bodenstab, "Efficient Implementation Of The CKY Algorithm," Computational Linguistics, Final Project Paper, 2009.

[23] S. A. Blythe, M. C. James, and S. H. Rodger, "LLparse and LRparse: Visual and Interactive Tools for Parsing," in *Proceedings of the Twenty-fifth SIGCSE Technical Symposium on Computer Science Education*, 1994.