مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(١١) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (11) 2023

زيادة استطاعة اللغة الشكلية المتولدة باستخدام النحو الشكلي النظامي(TYPE 3)

لودي يعرب ديوب*

(تاريخ الإيداع 5/2/ 2023 – تاريخ النشر3/12/ 2023)

□ ملخّص □

ان اللغة الشكلية المتولدة باستخدام النحو الشكلي النظامي (Type3)محدودة وضعيفة وبالتالي فان عدد سلاسل الدخل المحرفية الممكن توليدها صغير .وبغية تنفيذ التحليل النحوي لعبارات الدخل فان ذلك يتطلب وجود عدد كبير من قواعد الاشتقاق مختلفة البنية الذي يتطلب بدوره زمن تحليل نحوي كبير جدا "مما حد من استخدامه، ولحل هذه المشكلة تم اقتراح استخدام قواعد اشتقاق متعددة المستويات المشتركة وبعدة محارف مما يسمح بتوليد عدة سلاسل محرفية مختلفة البنى والتعقيد مما يسمح بزيادة استطاعة اللغة الشكلية المتولدة.

الكلمات المفتاحية:النحو –قواعد الاشتقاق – التحليل.

---

*لودي يعرب ديوب الحاصلة على شهادة الماجستير. هندسة تقانة المعلومات –كلية هندسة تكنولوجيا المعلومات والاتصالات. جامعة طرطوس – طرطوس–سورية.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(١١) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (11) 2023

Increasing the power of generated formal language for regular formal grammar (type-3)

Lude Yaroub Dayoub*

(**Received 5/2/2023.Accepted 3/12/2023)**

□**ABSTRACT** □

The generated formal language using regular formal grammar (Type3) is very poor and limited, in result can using it syntaxis very small number of input string series. For executing the syntaxis analysis of large input string series it's requires using very large number of production rules which requires big time in result this restricts the using of this formal grammar type. In this paper, it was recommended to use production rules with multi-hierarchy levels, with many shared characters, which permits generation different string series with different structures which permits increasing the power of generated language.

Keywords:Formal grammar, production rule, syntaxes.

---

*Lude Yaroub Dayoub. Master of science, Department of Technology Engineering, Faculty of Technology Engineering of information and Communication, Tartous University, Tartous- Syria

1.Introduction

Last time it was used different modeling scientific approaches to solve different scientific problems enhancement, and for solving this problems it was used Colored Petre Net CPN Neural Petre Net ,clouding computing, Formal grammars Formal generated linguistic languages..etc[1][8].

The using of the formal grammars Automata category (type3) regarding to Chomsky classification wasn't unfortunately successfully because it's required very big number of production rules and so it's required very big time for executing syntaxis analysis. This formal grammars generates very limited poor formal language and isn't powerful so enough for specifying modeling controlled objects.

Using formal grammar with it's production rule permits substitute each secondary element at left side (of production rule) by one terminal element or by set of terminal and secondary elements. Generated formal language very poor and isn't so rich for specifying different input string series[8][4]. For increasing the power of generated language (type3),it was recommended using together some additional production rules free-context type with multi-hierarchy levels [5],which permits generate very rich and powerful formal languages by using finite set of production rules recommended type .

For eliminating above mentioned problems and optimizing the required for syntaxsis analysis time it's important to minimizing the number of applied production rules where each production rule leads to use the related suitable child production rule at certain level and use it in any condition in result it will be reduce the total number of faulty applied production rules in executing syntaxsis process, in result it's possible to minimize the number of recommended to use production rules and so reducing the number of the total number of production rules required for executing the syntaxis analysis of input string series[2].

2.Importance and aim of this work:

For increasing the power of generated formal languages it was inserted some production rules free-context type beside the using of production rules (Automata type) and using the hierarchy production levels so each production rule is pointed to select any child production rule recommended to use, so each hierarchy level has a special set of child production rules .This approach permits to increase the number hierarchy levels of production rules , so increasing the power of generated formal language and so very large number of different input string series can matched[6].

3.Search methods and materials:

Regarding to Chomsky classification there are four main formal grammar types, mainly all types have the same structure except production rules, where each type has its special production rules type. Let's demonstrate the types of formal grammar:

1.Free Formal Grammar (Type 0-free type)[1]:

This type can generate very big rich and powerful formal languages which sufficient for executing syntaxsis analysis of nature languages, non -terminal and terminal elements can located at right or left side of production rules which can be defined as follow:

$G_1 = \{V_T, V_N, S, P\}$ where:

V_T / V_N-finite set of terminal /non-terminal(secondary) elements(characters),

$V = V_T \cup V_N$ –dictionary , $V = V_T \cap V_N =$, SV_N-start symbol(element),

P- Finite set of grammar's productions rules have the following format $A \rightarrow a\beta$ that means left side of production rule "A" can be substitute by right side "aβ",

$A, \beta \in V_N$, V^*_T(*-zero, one or more occurrence)

.

2. **non-free context formal grammar** (type1)

Productions rules have the following format :

$\xi_1 A \xi_2 \quad \xi_1 \beta \xi_2 \quad (1)$

That means the left side of production rule element $A \in V_N$ can be substitute by right side element β and in the context defined by ξ_1 and ξ_2 where :

$\xi_2 , \beta \neq \lambda \subset V^* \quad , \quad |\xi_1 \beta \xi_2| \geq |\xi_1 A \xi_2| \xi_1$

3. **free context formal grammar** (Type2).

The production rule of Type2 has the next format $A \rightarrow \beta$ where:

$A \in V_N , \beta \in V^+$ that means the left side of production rule element $A \in V_N$ can be substitute by right side element β where will be necessary and eliminates the context factor.

4. **free context formal grammar**(Automata-Type3)

The production rule of this type3 has the next format:

$$A \rightarrow a\beta , A \rightarrow b \quad (2)$$

where:

$A, \beta \in V_N , a, b \in V_T$

that means the left side of production rule element $A \in V_n$ can be substitute by right side element "aβ" or by "b" where will be necessary and eliminates the context factor.

All languages $L(G_i) , \overline{i=0,3}$ generated by using formal grammar satisfied the following relation:

$L(G_3) \subseteq L(G_2) \subseteq L(G_1) \subseteq L(G_0) \quad (3)$

From this relation it's clear that the generated by formal grammar (Tye3-Regular grammar) is subset of languages all other types, in other word it's the poorest generated formal language.

In spite of this disadvantage this type has unique category of production rule where an secondary element can be substitute with only one terminal element, (A b) ,so no more iteration (no more child vertex),in other word the syntaxsis analysis of input string can be terminated at selected vertex by using like this production rule.

Another fact the free context formal grammar (type 2), has production rule where each secondary element at left side of production rules can substitute by set of secondary and terminal elements which permits generate powerful languages.

It's very important to increase the power of generated formal language by using the free context type of production rule(Type2) together with automate formal grammar(Type3) depending on the inserted hierarchy level.

The main idea is inserting sum production rules related to previous hierarchy level at any next hierarchy level, so it's possible to cross-over some levels and eliminates them from being in syntaxsis analysis tree[2].

4. **Discussion:**

It's important to notice that is the speed of executing the syntaxsis analysis of input string so far depending on the correct selection of the corresponding production rule and the power of generated formal language proportional with the number of used

production rules so, it's necessary to compensate between formal grammars type2 and type3 [6].

At each hierarchy level it's possible to be include some production rules which permits transfer the syntaxsis analysis from one hierarchy level to another, so reduce the number of given vertex necessary to visit.

The regular formal grammar is defined as four tuples as follow:

$G_r = \{V_T, V_N, P, S\}$ Where:

P-finite set of production rules with next format: $A \xrightarrow{\beta}$

Generally it's possible recognizing the following hierarchy different levels:

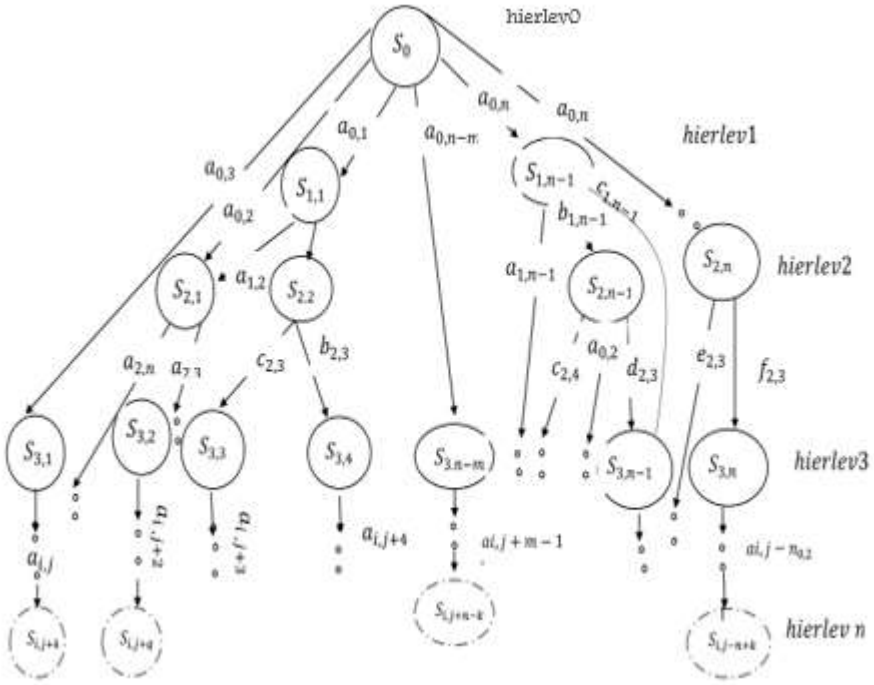

figure.1.multi hierarchic levels crossover transitions

$S_{i,j}$ -Vertexes recommended to visit at given row i= $\overline{1,n}$ and column j=$\overline{1,m}$ (figure.1).

$\{\alpha_{i,y}, b_{i,j}\ d_{i,j}\ e_{i,j}, c_{i,j}, f_{i,j}\}$ -terminal elements represents the weight of the related edge between tow vertexes recommended at given row i= $\overline{1,n}$ and column j=$\overline{1,m}$ where:

$\{\alpha_{0,2}, \alpha_{0,3}, \alpha_{0,2}, \alpha_{0,n}, \alpha_{2,n}, \alpha_{1,n-1}, c_{1,n-1}\}$–the weight of given hierarchic crossover transitions .

$\{S_{i,j+K}, S_{i,j+q}, S_{i,j+n-K}, S_{i,1-n+K}\} \subseteq F$ -set of final vertexes, k, q-constants.

hierlev i-hierarchy level with sequence serial number "i".

oriented edges permits transitions between vertexes at different hierarch levels.

a)**initial** hierarchy level of production rules can be consisting the following list:

$a_{0,1}S_{1,1} \longrightarrow$ //right part is containing terminal and non-terminal elements. 1. S_0

$a_{0,2}S_{2,1} \longrightarrow$ 2. S_0

$a_{0,3}S_{3,1} \longrightarrow$ 3. S_0

$b_{0,n}S_{1,n} \longrightarrow$ 4. S_0

$a_{0,n-1}S_{3,n-1} \longrightarrow | \varepsilon$ // production rule with empty string 5. S_0
6. S $bS_{3,1} \longrightarrow | \varepsilon$
7. $S_0 \longrightarrow b$ //right part is consisting of only terminal elements
8. $S_{1,1} \longrightarrow a$
$bS_{n,1} \longrightarrow$ 9. S
Fig.2. **initial** hierarchy level
Where:
This production rules list reflects the possible passes starting at initial level and continue at first hierarchy level (production rules number 1,4),second hierarchy level (production rules number 2,5),..,n^{-th} hierarchy levels(production rules number is 9).

B)first hierarchy level can contain the following list of production rules.
$a_{1,2}S_{2,1} \longrightarrow$ // production rule cross-over type to 2^{-d} hierarchy 10. $S_{1,1}$ level.
$a_{1,3}S_{3,4} \longrightarrow$// production rule cross-over type to 3^{-th} hierarchy 11. $S_{1,1}$ level.
$b_{1,2}S_{2,2} \longrightarrow$12. $S_{1,1}$
$a_{1,2}S_{3,1} \longrightarrow$ 13. $S_{3,1}$
$bS_{4,1} \longrightarrow$ // production rule cross-over type to 4^{-th} hierarchy 14. $S_{2,1}$ level.
$aS_{5,1} \longrightarrow$// production rule cross-over type to 5^{-th} hierarchy 15. $S_{2,1}$ levels
b $\longrightarrow$ //right part is containing only terminal elements 16. $S_{2,1}$
$a \longrightarrow$17. $S_{1,1}$
$aS_{n-1,1} \longrightarrow$ 18. $S_{2,1}$
$bS_{n,1} \longrightarrow$ 19. $S_{2,1}$
$\varepsilon \longrightarrow$ 20. $S_{3,2}$
Fig.3. first hierarchy level's production list .

This production rules list reflects the possible passes starting at first and second hierarchy level and continue, at third hierarchy level (production rule number 13),fourth hierarchy level. (production rule number 14), .., n^{-th} hierarchy levels(production rules number 19).
c)second hierarchy level can contain the following list of production rules.
$a_{2,3}\ S_{3,\ 1} \longrightarrow$ 21. $S_{2,3}$
$b_{2,3}\ S_{3,2} \longrightarrow$ 22. $S_{2,3}$
23. $S_{2,3} \longrightarrow a_{2,3}\ S_{3,4}$
$b_{2,3}\ S_{3,5} \longrightarrow$ 24. $S_{2,3}$
$a_{2,j+1}\ S_{i,j+1} \longrightarrow$25. $S_{2,3}$
$a \longrightarrow$//right part is containing only terminal element. 26. $S_{3,2}$
$a\ S_{n-1,1} \longrightarrow$27. $S_{3,2}$
$bS_{n,1} \longrightarrow$28. $S_{3,2}$
$\varepsilon , \longrightarrow$ ε-empty string series. 29. $S_{3,2}$
Fig.4. Second hierarchy level 's production list.
These production rules list reflects the possible passes starting at third hierarchy level and continue at level 4^{-th} (production rule number 21),five^{-th} hierarchy level(production rule number 22),6-th(production rule number 23) an so on then n^{-th} hierarchy level (production rule number 28).

d)The(n-mk)$^{-th}$ hierarchy level can contain the following list of production rules.

29. $S_0 \longrightarrow a_{0,n}S_{1,n}$

$a_{0,n-1}S_{3,n-1} \longrightarrow$ 30. S_0

31. $S_0 \longrightarrow b_{0,n}S_{2,n}$

$a_{1,n}S_{3,n-1} \longrightarrow$ 32. $S_{1,n}$

$b_{1,n-1}\, S_{2,n-1} \longrightarrow$ 33. $S_{1,n}$

$c_{1,n-1}S_{2,n} \longrightarrow$ 34. $S_{n-m4,1}$

$c_{2,3}\, S_{3,n-1} \longrightarrow$ 35. $S_{2,n-1}$

$c_{2,j+n-1}\, S_{i,j+n-1} \longrightarrow$ 35. $S_{2,n-1}$

$d_{2,3}\, S_{3,n} \longrightarrow$ 35. $S_{2,n-1}$

$a_{n-1,\, j+n-1}\, S_{i,j+n-1} \longrightarrow$ 35. $S_{2,n-1}$

$e_{2,3,\, j+n-1}\, S_{3,\, n+1} \longrightarrow$ 35. $S_{2,n}$

$f_{2,3}\, S_{3,n} \longrightarrow$ 35. $S_{2,n-1}$

36. $S_{n-mk,1} \longrightarrow a$ //right part is containing only terminal element

$\varepsilon \longrightarrow$ //inserting empty string 37. $S_{n,1}$

Fig.5. n-mk^{-th} hierarchy level 's production list

This production rules list reflects the possible passes starting at (n-2)$^{-th}$ hierarchy level to last hierarchy "n" level or transfer the syntaxis analysis from any hierarchy level with number "n-mk" to any next higher hierarchy level "(n-mk)+C " , where : n,m,k ,c-positive integer constant.

As seen in result it will be eliminate all intermediate hierarchy levels from syntaxsis analysis.

From above mentioned contractures, it's clear that each secondary element located at left side of production rules can substitute by any string at right side of recommended subset of production rules started with the same secondary element at left side, so the power of generated language will be increased by the number of this subset of secondary elements production rules, also the number of hierarchy levels so far can define the power of generated formal language at different hierarchy levels[7].

5.**multi -hierarchy levels formal production rules**

Suppose the number of production rules at first hierarchy level with one secondary element at right side of production rules $N_{i,j,k}$ $\overline{i=1,m1}$, $\overline{j=1,m2}$, $\overline{k=1,m3}$, where:

i-Hierarchy level's sequence serial number , j- vertex's serial sequence number
k-Child vertex's sequence serial number. m_1,m_2,m_n-Constraint constants .

At 2^{-nd} hierarchy level each production rule has $N_{2,j,k}$ (production rules with two secondary element at right side of production rules) in this case the total number of possible to generate different string series $N_{hier.2}$ is calculated by the next forma:

$$N_{hier.2} = N_{1,j,k} * N_{2,j,k} \qquad (4)$$

At the (n-1)$^{-th}$ hierarchy level the total number N_{n-1} possible to use production rules for generation different string series is calculated by the next formula:

$N_{n-1}= N_{1,j,k} * N_{2,j,k} *\ldots* N_{i,j,k}$ (5)

From this relation it's clear the number of possible generated very rich and powerful language which permits modeling very complex object's motion and by using the production rules with multi-hierarchy levels permits eliminate many production rules from using in syntaxsis analysis and so far optimizing the required time for executing syntaxis of input string series .
Production rules with maximum two secondary elements at right side using any hierarchy production rule at 2^{-nd} level permits discard 8 vertexes(fig.3) from visit list and the maximum number $N_{max.h.2}$ of discarded vertexes from visit list($N_{h2.rule}$-the recommended to visit production rules) is given as follows:

$$N_{max.h.2}=N_{h2.rule}^{2}-1 \qquad (6)$$

at 3^{-d} level using any hierarchy production rule permits discard 12 production rules (vertexes) from visit list and the maximum number of discarded vertexes from visit list is given as follows:

$$N_{max.h.3}=N_{h.rule}^{3}-1 \qquad (7)$$

Generally The total number of vertexes recommended to visit at each hierarchy levels is 2^n (n-hierarchy level sequence number),in result the discarded vertex's number from visit list is given by this formula:

$$N_{v.disc}=2^{n}-1 \qquad (8)$$

6. **The actual coefficient of the utilizing hierarchy production rules.**

Suppose visit time one vertexes is T_i , so the maximum total time $T_{max\cdot total}$ required to visit recommended child vertexes $N_{ch.ver}$ is depending on the number of hierarchy production levels $N_{h.lev}$ and is given by this formula :

$$T_{max\cdot total} =T_i * N_{h.lev} * N_{ch.ver} \qquad (9)$$

Using multi –hierarchy levels permits optimizing the possible very big time required for executing the syntaxes analysis of input string.
Suppose "γ_i", $\overline{i=1,m1}$, then the discarded vertex's number at production rules hierarchy levels "i" so the max the hierarchical maximum total time $T_{h.max\cdot total}$ required to visit recommended child hierarchical vertexes $N_{ch.ver}$ depending on the number of hierarchy production levels $N_{h.lev}$ and is given by this forma :

$$T_{h.max\cdot total} =T_i*(N_{h.lev}- \gamma_i)* N_{ch.ver} \qquad (10)$$

The optimized time is given by the following forma:

$$T_{optimized} = T_{h.max\cdot total} /T_{max\cdot total}= T_i*(N_{h.lev}- \gamma_i)* N_{ch.ver} \ / \ T_i* N_{h.lev}* N_{ch.ver}$$
$$= (N_{h.lev}- \gamma_i)/ N_{h.lev} \qquad (11)$$

Example:
To illustrate the proposed approach suppose the number of hierarchy levels $N_{lev}=3$,
Lev0-start hierarchy level , Lev1. Lev2, Lev3-first, second ,third hierarchy level respectively, (see fig.6).

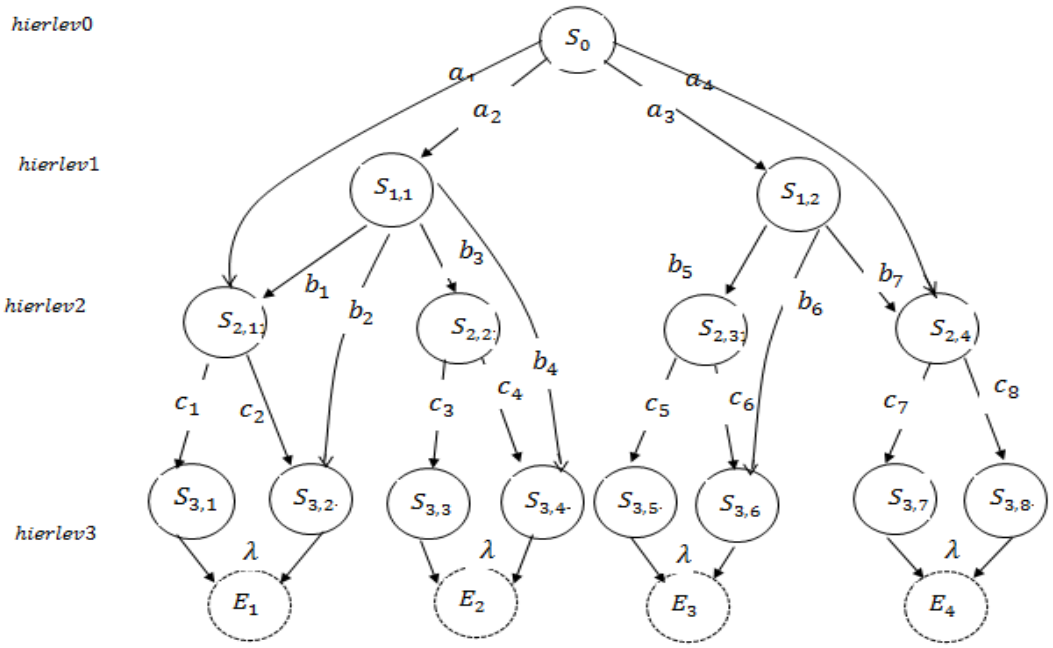

Fig.6.Crosover transition between different hierarchy levels

The used grammar for executing the syntaxis analysis of input string defined as follow:

$G=(S_O, V_t, V_n, P)$ where:

S_0-start element.

$V_t=\{\alpha_1,..,\alpha_4, b_1,b_2,..,b_7,c_1,c_2,..,c_8\}$-set of terminal elements.

$V_n=\{S_{1,1}, S_{1,2},..,S_{2,1}, S_{2,4}, S_{3,1},..., S_{3,8}, E_1,..,E_4\}$- set of non-terminal(secondary) elements.

P-set of production rules represented by oriented arrows in the given graph, where each secondary element can be replaced by an terminal element followed child vertex name(non secondary).

The path from vertex S_0 to vertex $S_{1,1}$ with the weight "a_2 " ,so the corresponding (fig.8) production rule has the following format $S_0 \xrightarrow{\quad} a_2 S_{1,1}$.

For simplicity suppose no recursion and the process of syntaxis analysis has oriented edges.

At any hierarchy level the maximum number of neibour child vertexes is 2 .

The crossover hierarchy will be using starting at first level vertex S_0 to vertex $S_{2,1}$ and $S_{2,4}$ (2^{-nd} hierarchic level).

Also the crossover hierarchy will be using at first level vertex $S_{1,1}$ to vertex $S_{3,2}$ and $S_{3,4}$ $S_{i,j}$ -Vertexes (secondary element)recommended to visit at given row $i=\overline{1,3}$ and column $j=\overline{1,8}$ (figure.1).

a_1,a_4 -terminal elements represents the weight of the related crossover transition between vertex S_0 and $S_{2,1}$, $S_{2,4}$ respectively.

b_2, b_4 -terminal elements represents the weight of the related crossover transition between vertex $S_{1,1}$ and $S_{3,2}$, $S_{3,4}$ respectively.

b_6- terminal element represents the weight of the related crossover transition between vertex $S_{1,2}$ and $S_{3,6}$.

$\{\alpha_2, \alpha_3, b_1, b_3, b_5, b_7, c_1, c_2, .., c_4\}$–the weights(set of terminal elements) of given ordinary transitions between recommended to visit child vertexes .
$\{E_1, E_2, E_3, E_4\} \subseteq F$ -set of final vertexes ,E_4-the target final vertex.
hierlev i-hierarchy level with sequence serial number "i=3". λ-input empty string.
Depending on the number of visited vertexes getting the target final vertex through one of the recommended path let's calculate the improvement factor μ_{impro} of using production rules (represent the recommended vertexes to be visit)which used in syntaxis analysis of input string where it's can recognize the minimum value(number of possible to visit vertexes at first level as follows:

$N_{Tot.min}= N_{v.min} /N_{Tot.min}=2/4=1/2$ (12)

The improvement factor's maximum value is given as follows:

$N_{Tot.max}= N_{v.max} /N_{Tot.max}=2/4=1/2$ (13)

By the same way we calculate the improvement factor μ_{impro} for other levels.
In generally the general improvement factor is given as follows:

$\mu_{impro}=1/2^n$

Tab1.The gain of using recommended approach

hierLevel sequence number	N_{recom}	$N_{her.prod}$	N_{visit}		$N_{Total.pass}$		μ_{impro}	
			$N_{v.min.}$	$N_{v.max}$	$N_{Tot.min}$	$N_{Tot.max}$	$\mu_{imp.min}$	$\mu_{imp.max}$
hierLev.0	2	0	2	2	4	4	1/2	1/2
hierLev.1	4	4	4	15	16	60	1/4	1/4
hierLev.2	8	8	3	7	24	56	1/8	1/8
hierLev.3	16	16	2	3	32	48	1/16	1/16

Fig.7. The relationship between the traditional and recommended approach

Conclusions:

The hierarchy multilevel structure of production rule free –context type permits the following operations:

-Crossover "n" syntaxis levels ,so minimize the time required for syntaxsis analysis.
-Inserting some production rules free- context type gives the constructed formal grammar more power and strength.
-Using the hierarchy types of production rules permits select the suitable hierarchy production rules with the suitable number of recommended to visit child vertex.
-Minimized structure of production leads to minimize the required time for executing the syntaxis analysis of input string.
-The hierarchy types production rules(type2) permits executing syntaxsis analysis of
different input series string.
-generated formal languages has dynamic flexible structure ,which permits modeling many scientific problems.
-It's possible to use in different scientific fields(pattern recognition ,compilers, automaton etc.).

References:

1.Symbolic *Rewgister Automat for Complex Event Recognition and Forcasting*,2021.
2.Formal grammar and languages.A Petrrossi-*Automat theory and Formal springer*- languages,2022.

3.Simulating derivations of context free grammar. Researches sguare.com- k vayadande-2022.
4.&field-author=Luis+M+Augusto&text=Luis+M+Augusto&sort=relevancerank&search-alias=books"Augusto"Luis.M.HYPERLINK "%26field-author=Luis+M+Augusto%26text=Luis+M+Augusto%26sort=relevancerank%26search-alias=books%22Luis%20M%20HYPERLINK%20%22https://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Luis+M+Augusto&text=Luis+M+Augusto&sort=relevancerank&search-alias=books"Augusto"Augusto.*LanguagesMachines,andClassicalComputation Paperback* February 4, 2019.
5.Luis M. Augusto, *Languages, machines, and classical computation, London: College Publications*, 2019. ISBN -5%22978184890300-5"978HYPERLINK "5%22978-1-84890-300-5"-HYPERLINK "5%22978-1-84890-300-5"1HYPERLINK "5%22978-1-84890-300-5"-HYPERLINK "5%22978-1-84890-300-5"84890HYPERLINK "5%22978-1-84890-300-5"-HYPERLINK "5%22978-1-84890-300-5"300HYPERLINK "5%22978-1-84890-300-5"-HYPERLINK "5%22978-1-84890-300-5"5. Web page.
6.A.Bombarda,A.Gargantini,*"An Automata-Based Generation Method For Combinatorial Sequence Testing of Finite State Machines*",2020
7.E.Muskardin,IPill,Mtartin Tappler2,1 and Bernhard K.Aicherning,"*Automata learning Enabling model based Diagnosi"*,2021.
8.Automated *Requirements-Based Testing of Black-Box Reactive system*. http://link.springker.com/chapter/HYPERLINK "http://link.springker.com/chapter/10"10.