2020سلسلة العلوم الهندسية المجلد (4) العدد (9) _ مجلة جامعة طرطوس للبحوث والدراسات العلمية
Tartous University Journal for Research and Scientific Studies- Engineering Sciences Series Vol. (4) No. (9) 2020

تسريع التحليل النحوي باستخدام النحو الشكلي الاحتمالي المعدل

د.يعرب شحاده ديوب*

(تاريخ الإيداع 22 /9 / 2020 . قُبل للنشر 28 / 10 / 2020)

□ الملخّص □

ان سرعة تنفيذ التحليل النحوي لعبارات الدخل يعتمد الى حد بعيد على نوع النحو الشكلي المستخدم كالنحو الشكلي الاحتمالي الذي يستطيع توليد لغات غنية وقوية كفاية لتنفيذ التحليل النحوي لمختلف عبارات الدخل ،حيث تخصص وفق هذا النوع من النحو الشكلي قواعد الاشتقاق بقيم احتمال استخدامها اعتمادا" على الخبرة العملية الذي قد يتسبب في توليد الاخطاء، وبالتالي زيادة زمن تنفيذ التحليل النحوي وانخفاض دقته بالإضافة لعدم امكانية انشاء اشجار فرعية ، وانطلاقا" من نفس العنصر المنطلق وبالتالي انخفاض قوة اللغة المتولدة. يتطرق هذا البحث لتقديم حل للمشاكل المطروحة اعلاه وزيادة سرعة التحليل النحوي المعتمد على النحو الاحتمالي وزيادة قوة اللغة المتولدة.

الكلمات المفتاحية: النحو الشكلي– قاعدة اشتقاق– شجرة – تحليل نحوي.

* استاذ مساعد في قسم هندسة تقانة المعلومات –كلية هندسة تكنولوجيا المعلومات والاتصالات
جامعة طرطوس – طرطوس– سورية.

مجلة جامعة طرطوس للبحوث والدراسات العلمية_ سلسلة العلوم الهندسية المجلد (4) العدد (9) 2020
Tartous University Journal for Research and Scientific Studies- Engineering Sciences Series Vol. (4) No. (9) 2020

SPEEDING UP THE SYNTAXSIS ANALYSIS USING MODEFIED STOCHASTIC FORMAL GRAMMAR TYPE

Dr. Yaroub Dayoub*

(Received 22 / 9 / 2020 . Accepted 28 / 10/ 2020)

□ ABSTRACT □

The speed of syntaxis analysis, of input series depends, so far on the type of used formal grammar as stochastic type, which can generate very rich and powerful languages, for executing syntaxis analysis of different complex input string, but the everisticaly given applied probability for each production rule can reduce the resolution and increase the required time for executing the syntaxis analysis of input string, which can case infinite syntaxis analysis ,in addition to there is possibility to construct only one tree starting with, the start element. In this paper was introduced an approach to sole above mentioned problems and increasing the of the generated formal grammar.

Keywords: Applied probability, formal, grammar, rule, tree, syntaxis, series, analysis.

*ph.d.Yaroub Dayoub. assistant, Department of Technology Engineering, Faculty of Technology Engineering of Information and Communication, Tartous University, Tartous- Syria.

1. Introduction:

While executing some operations as data transmission, transaction, correction or measuring physical parameters of different phonemes, it is necessary to, recognize any form of the studied objects, this can cause infinite process.

Because the used for specifying the object's state with or without raised noise, can leads to find some objects belong to many different classes, t.e one string series can be generated by different formal grammar in result can be raised unsolved problems like what is the type of used formal grammar for specifying given object?.

2. Importance and aim of this work:

The stochastic formal grammar with, the specified everisticaly applied probability value for each production rule make this type is powerful and can generate very rich formal languages so enough for executing syntaxs analysis for big variety of input series, but the applied probability value given for production rule is selected everisticaly ,this method can causes may problems ,t.e increasing the required time and in result the efficiency analysis factor became very low.

In some cases the low unsuitable applied probability value prevents using a production rule where this necessary to make and this leads to gets fault results and where the big unsuitable applied probability value wouldn't be use.

3.Search methods and materials:

For modeling different controlled objects it was used the stochastic formal grammar where not wishes production rules are given very low probability expected value. But the value of using production rules probability staid without limitation and defined everisticaly, which can leads to fault and without result execution.

Stochastic formal grammar is defining as four tuples:

$G_S = (V_N, V_T, P_S, S)$ (1)

Where:

V_N / V_T - finite set of non-terminal / terminal elements respectively.

$S \in V_N$ - start symbol (element).

P_S: finite set of stochastic production rules, which has two next forms:

$$\alpha_i \xrightarrow{Pij} \eta_{ij}, \quad j = \overline{1,n} \ , \ i = \overline{1,k} \qquad (2)$$

Where:

$\alpha i \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*, \eta_{I,J} \in (V_N \cup V_T)^*$

P_{ij} . probability using of related production rule in condition

$0 < P_{ij} \le 1$, $\sum_{i=1}^{k} \sum_{j=1}^{n} P_{ij} = 1$ (3)

Production rule (2) means string series "αi" can directly substitute by η_{ij} and if there is a string series $X = \varsigma_1 \alpha_i \varsigma_2$ it takes the form $X = \varsigma_1 \eta_{ij} \varsigma_2$ with using of production rule probability P_{ij} .its necessary to notice the next relation:

$$\alpha_i = \varphi_1 , \eta_i = \varphi_1 + 1 , \varphi_i \overset{P}{\Rightarrow} \varphi_i + 1 , i = \overline{1,n} \quad (4)$$

This means string series "α_i" generates "η_i" with probability "P" which defined as the multiplication probability of all used production rules, i.e.

$$P = \prod_{i=1}^{n} P_i \quad (5)$$

it is clear that relation reflectively and transitively.

The stochastic generated formal language $L(G_S)$ is defined as follows:

$$L(G_S) = \{ (x, P(x)) \mid x \in V_T^* , \alpha_i \xrightarrow{P_{ij}} \eta_i , i = \overline{1,k} , P(x) = \sum_{i=1}^{k} P_i \} \quad (6)$$

Where:

k . the number of generated string series، P_i . probability generation of string series.

$$L(G_S)_{Type3} \subseteq L(G_S)_{Type2} \subseteq L(G_S)_{Type1} \subseteq L(G_S)_{Type0} \quad (7)$$

The stochastic generated language $L(G_S)_{Type3}$ is a stochastic language of any type $L(G_S)_{Type2}$ and $L(G_S)_{Type2}$ is a stochastic language $L(G_S)_{Type1}$ and the last one is stochastic language $L(G_S)_{Type0}$ at the same time[1,7,6].

Stochastic formal grammar restricts the using of production rule, through specializing each production rule with its applying probability value, so the applying probability of each set of production rules started with the same (non-terminal) left side element and different right side defined as follow:

$$P = \sum_{i}^{n} \sum_{j}^{n} P_{ij} = 1 \quad (8)$$

Where:

i . the sequence serial number of the non-terminal. j . the serial number of the production rule related to a set $S_k , k = \overline{1,n}$.

As seen from above the applying probability of production rules is defined everisticaly, i.e there is possibility to arise some errors, caused faulty results (from one side). From other side all the production rules related to one set must have different terminal at right side, this eliminates the possibility to has different production rules with the same at right side (non-terminal) element but different followed set of non-terminal which decrease the power of generated stochastic language, which leads to increasing the required time for executing syntaxs analysis and low efficiency of using like this grammar[1,4,,6,8].

Using this grammar makes the constructing of syntaxes tree with multi-sub-tree is very difficult, so the constructed syntaxs tree starting with one start element, discard the possibility to minimized the structure of constructed syntaxes tree, so more faulty using of production rules and in result more expired time in best case(the input string was correctly) , but in worst condition(the input string was faulty constructed) it will use all possible production rules , this leads to many backtracking to parent vertexes and increase the required for syntaxes analysis time. In result small effect of using this type of formal grammar[2,5].

Generated string series(by using stochastic formal grammar) can belonged to any type of formal grammars, which makes the definition of class of the generated series belongs to is very difficult, and so arise serious problems [4,7,8]how to define the class to which the generated string series belongs to?

Let's suppose we have the stochastic (context-free) formal grammar G_s is defined as follows:

$G_S=(V_T,V_N,P_S,S)$,where:

$V_N=\{A_1,A_2,\ldots,A_K,B_1,\ldots,B_K\}$ – finite set of non-terminal elements.

$V_T=\{\alpha_1 \ldots \alpha_m\}$ – finite set of terminal elements ,$S=A_i$-Start symbol , $i=\overline{1,k}$

(9)

$$A_1 \xrightarrow{P_{ij}^1} \alpha_1 A$$

…….

$$A_1 \xrightarrow{P_{ij}^{m1}} \alpha_{m2} B_1$$

$$B_1 \xrightarrow{P_{ij}^{m1}} \alpha_{m1} B_1$$

$$\sum_{j=1}^{k}\sum_{l=1}^{m}P_{ij}^{l}=1$$

$$\sum_{j=1}^{k}\sum_{c=1}^{m1}P_{ij}^{c}=1$$

From above it is clear these facts:

- If there are some production rules with the same using probability, so the randomly selection of next recommended to use production rule will be use, and this leads to many faulty applying of production rules.
- The production rules with the same start element has applying probability “p”, so no possibility to construct any sub-tree starting with the same start element, in result the syntaxis tree will grows horizontally and decrease the power of possibility constructed syntaxis tree, and the possibility to generate string language will be restricted, and using this type of formal grammar doesn’t permit executing syntaxis analysis of wide range of input string series.
- The given for each production rule using probability ″ P_i ″ everisticaly is defined, i.e, there is a possibility to make some mistakes, which causing faulty applying of production rules.
- It is very important to have some production rules with the same value of applying *probability which permits to construct many different sub-tree starting with any number* of child vertexes, but without using randomly selection of proposed to apply production rule.
- No inference of next expected to use in applying process production rule, where it is necessary to use given applying probability to estimate the next possible to use production rule.

4.Discussion:

Through studying above mentioned problems, raised question how to eliminate them?, increasing the grammar power and decreasing the expired spend for executing syntaxis analysis time.

If we specify each production rules with the weight W_i , $i=\overline{1,k}$, which consisting of two parameters p_i , r_i , $\overline{i=1,k}$,where:

p_i –probability values of using production rules, R_i– Boolean expression

The production rule (2) takes the following format:

$$\alpha_i \xrightarrow{W_i(P_i,r_i)} \eta_{ij} \qquad (10)$$

Where:

$$i=\overline{1,k}\ ,\ j=\overline{1,k}\ ,\ \alpha_i \in V_N\ ,\ \eta_{ij} \in V^*$$

The production rule with probability P_{ij} will be using in applying if the Boolean expression r_i , $i=\overline{1-k}$ is in true condition and in result the symbol α_i would replace by η_{ij} , otherwise it will be searching for new production rule with Boolean expressions r_{i+k} , $\overline{k=1-k}$ in condition

" true" if all Boolean expressions in condition $r_i = `false`$,so the syntaxis analysis will issues error message the input string series was faulty constructed this structure of recommended formal grammar permits having many production rules with the same value of applying probability P_i, t.e. possibility to construct syntaxis analysis tree with many sub-trees starting at any child vertex, so supports with a wide range of different input string series and impact the number of possible to use production rules.

If we have "m_1" production rules with the same applying probability " P_i "and set of different Boolean expressions r_i , $\overline{i=1-k}$, so it possible to build "(m_1-1)" additional sub-tree.

In condition the formal grammar consisting of set of "n_1 "different production rules with the different applying probability "P_i "and various Boolean expression" r_i " (various production rules can have same Boolean expression), so it is possible to construct "£" different sub-tree which can calculate by this formula:

$$\mu = n_1(m_1-1) \qquad (11)$$

Let`s suppose each parent vertexes contains "k_1" child vertex , so the total possible to construct string series is given using the next form:

$$£=\mu * k_1 \qquad (12)$$

Using this type of production rule permits constructing many sub-tree with same start symbol(see fig.1).

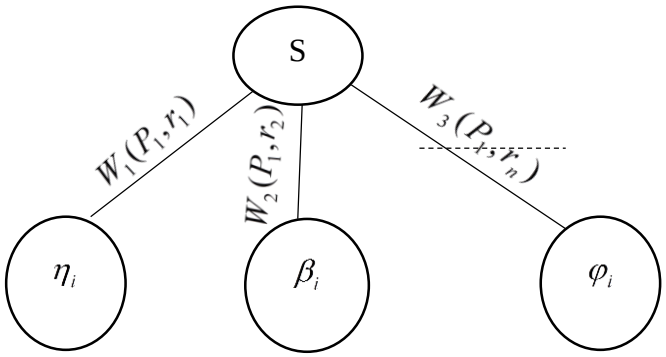

Fig.1.syntaxs tree with "n" subtree using one start element.

$$S \xrightarrow{W_1(P_1,r_1)} \eta_i$$
$$S \xrightarrow{W_2(P_1,r_2)} \beta_i \qquad (13)$$
$$\cdots\cdots\cdots\cdots$$
$$S \xrightarrow{Wn(P_1,r_n)} \varphi_i$$

Suppose we have the number of production rules with the same start symbol is `$\ell$` each production rule with start element `S` can construct one sub-tree with number possible to use production rule, so the number of possible to use production rules is $N_{star} = \ell * c$ regarding to formal (13) and (14) the power "ρ" of recommended formal grammar is increased as follows:

$$\rho = \ell + N_{star}$$

$$\rho = n_1 * k_1 (m_1 - 1) + \ell * c \qquad (14)$$

Providing a set of production rules with everisticaly applying probability values of using production rules which supported with related Boolean expression, eliminates the possibility to make any error because the applying of production rules depends on two different parameters

p_i , r_i ,(see fig.2) ,where the Boolean expression delete the possibility of using production rule.

This kind of formal grammar permits having many production rules with the same left side and different right side increasing the possibility to construct additional sub-tree, in result generates very rich powerful formal languages which suitable for different purpose.

let's suppose it's required to make syntaxs analysis of the following given digital series as

X ="10110.110" and string series Y="ab&ba" ,now for executing syntaxis analysis of X

we construct the necessary stochastic formal grammar with the following set production rules P_i:

1:$S_0 \xrightarrow{w_1}$< integP >< . >< floatP > W_1: p_1=0.5, r_1: integP ! = ""

2:$S_0 \xrightarrow{W2}$< 0 >< . >< floatP > W_2: p_1=0.5, r_2: integP ! = 0

3:$S_0 \xrightarrow{W3}$< 1 >< . >< floatP > W_3: p_1=0.5, r_3: integP ! = 1

4:$S_0 \xrightarrow{W4}$< integP >< e >< floatP > W_4: p_1=0.5, r_4: integP *contains e*

5:$S_0 \xrightarrow{W5}$< integP >< E >< floatP > W_5: p_1=0.5, r_5: integP *contains E*

6: < integP >$\xrightarrow{W6}$< digit >< digits > W_6: p_6=0.5, r_6: integP *is digit*?

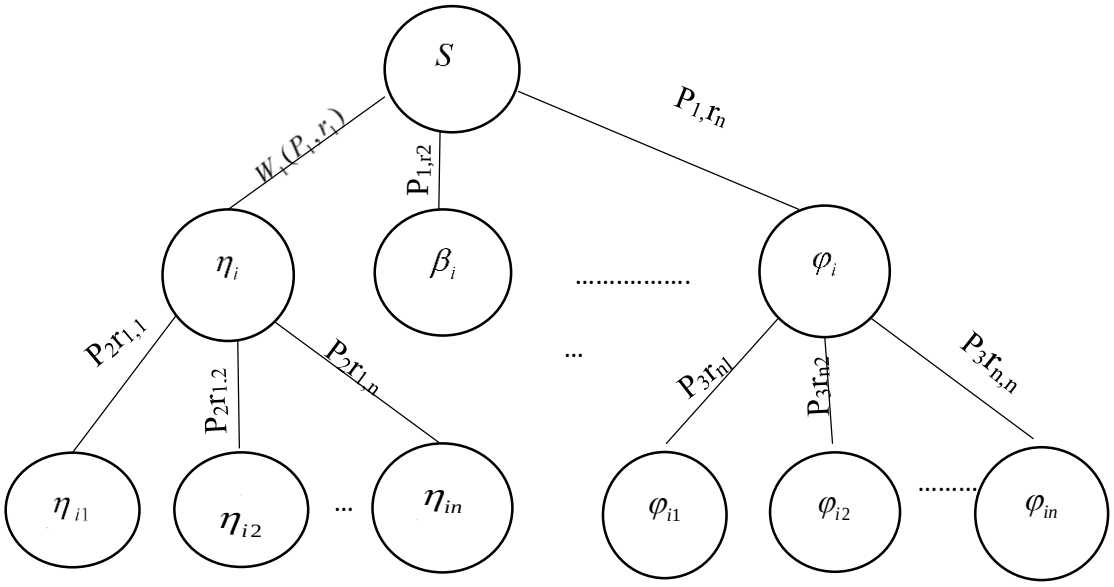

Fig.2.syntaxs tree with "n" child subtree.

7:< digit >$\xrightarrow{W7}$< digits >< digit > W_7: p_7=0.4, r_7: integP *is* ≥ 2?

8:< digits >$\xrightarrow{W8}$< digit >< digit > W_8: p_8=0.6, r_8: integP *is* ≤ 2?

9:< floatP >$\xrightarrow{W9}$< leftP >< rightP > W_9: p_9=0.3,r_9: integP *contains* "." *point*?

10:< floatP >$\xrightarrow{W10}$< leftP >< floatP > W_{10}: p_{10}=0.4, r_{10}: integP *is* $\geq$ 5 *digit*?

11:< floatP >$\xrightarrow{W11}$< Lfirs >< Lsec > W_{11}: p_{11}=0.3, r_{11}: floatP ≥ 2 $digit$?

12 :< Lfirs >$\xrightarrow{W12}$< digit >< digits > W_{12}: p_{12}=0.4, r_{12}: Lfirs > 2 $digit$?

13:< Lsec >$\xrightarrow{W13}$< 0 >< digit > W_{13}: p_{13}=0.3, r_{13}: digit = 0?

14:< Lsec >$\xrightarrow{W14}$< 1 >< digit > W_{14}: p_{14}=0.3, r_{14}: digit = 1?

15:< rightP >$\xrightarrow{W15}$< digits >< digit > W_{15}: p_9=0.5, r_{15}: rightP < 10?

16:< rightP >$\xrightarrow{W16}$< 1 >< digits > W_{16}: p_9=0.5, r_{16}:digits $starts\ with$ 1?

17:< leftP > $\xrightarrow{W17}$ < digits >< digit > W_{17}: p_{17}=0.6, r_{17}:leftP ≥ 2 1?

18:< leftP >$\xrightarrow{W18}$< 1 >< digit > W_{18}: p_{18}=0.4, r_{18}:leftP ≥ 2?

19: < digit > $\xrightarrow{W19}$ ε W_{19}: p_{19}=0.4, r_{19}:digit = ""?

20: < digit >$\xrightarrow{W20}$ 1 W_{20}: p_{20}=0.3, r_{20}:digit = 1?

21: < digit >$\xrightarrow{0.3}$ 0 W_{21}: p_{21}=0.3, r_{21}:digit = 0?

Let's calculate the probability of executing the syntaxs analysis of digital series

X ="10110.110" which will be done as follows:

$$S_0 \xrightarrow{1} P_1(10110).P_2(110)$$

$$p_{\text{int}}(101101) = 1\times0.5\times0.6\times0.6\times0.6\times0.6\times0.2\times0.2\times0.2\times0.2\times0.2\times0.2 = \underline{4.1472x10^{-6}}$$

We will calculate the after floating point (decimal)part, where $P_2(110)$ using the next set of stochastic formal production rules(here indicated only the values of probability "P"):

$S_0 \xrightarrow{1}$< integP >< . >< floatP >

$\xrightarrow{0.5}$< floatP > $\xrightarrow{0.5}$< leftP >< rightP >

$$\xrightarrow{0.5}< \text{leftP} >\xrightarrow{0.5}< 1 >< \text{digit} >< \text{rightP} >$$
$$\xrightarrow{0.5}< 1 >< 1 >< \text{digit} >< \text{rightP} >$$
$$\xrightarrow{0.3}< 1 >< 1 >< \text{digit} >< \text{rightP} >$$
$$\xrightarrow{0.3}< 1 >< 1 >< \varepsilon >< \text{rightP} >$$
$$\xrightarrow{0.3}< 1 >< 1 >< \varepsilon >< \text{digit} >< \text{digits} >$$
$$\xrightarrow{0.3}< 1 >< 1 >< 0 >< \text{digits} >$$
$$\xrightarrow{0.3}< 1 >< 1 >< 0 >< \varepsilon >$$

$$p_{float}(110) = 1\times 0.5\times 0.6\times 0.5\times 0.5\times 0.5\times 0.2\times 0.2\times 0.2\times 0.2\times 0.2 = \underline{1.2x10^{-7}}$$

The total value of the applying probability of production rules is for executing the syntaxs analysis for digital constants with floating point is calculated as follows(see.fig.3):

$$p(x) = P(10110.110) = p_{\text{int}}(101101)\ \text{X}\ p_{float}(110) = \underline{4.1472x10^{-6}}\ \text{X}\ \underline{1.2x10^{-7}} = \boxed{4.9766}\ \boxed{x10^{-13}}$$

b)syntaxis analysis for input string series x_2="ab&ba" .

Let's construct the set of stochastic production rules P_{ss} for executing syntaxs analysis string series $P_{SS}(Y)= P_{SS}$ ("ab&ba") as follows:

22:$< S_0 > \xrightarrow{W22} < \text{ser1} >< \text{ser2} >$ W_{22}: p_1=0.25, r_1: integP ! = ""

23: $< S_0 > \xrightarrow{W23} < \text{digit} >< \& >< \text{ser2} >$ W_{23}: p_1=0.25, r_1: S_0 contains "&"

24:$< S_0 > \xrightarrow{W24} < \text{char} >< | >< \text{chars} >$ W_{24}: p_1=0.25, r_2: S_0 contains "|"

25:$< S_0 > \xrightarrow{W25} < \text{digits} >< \&\& >< \text{chars} >$ W_{25}: p_1=0.25, r_2: S_0 contains "&&"

26:$< S_0 > \xrightarrow{W26} < \text{char} >< || >< \text{digit} >$ W_{26}: p_1=0.25, r_2: S_0 contains "||"

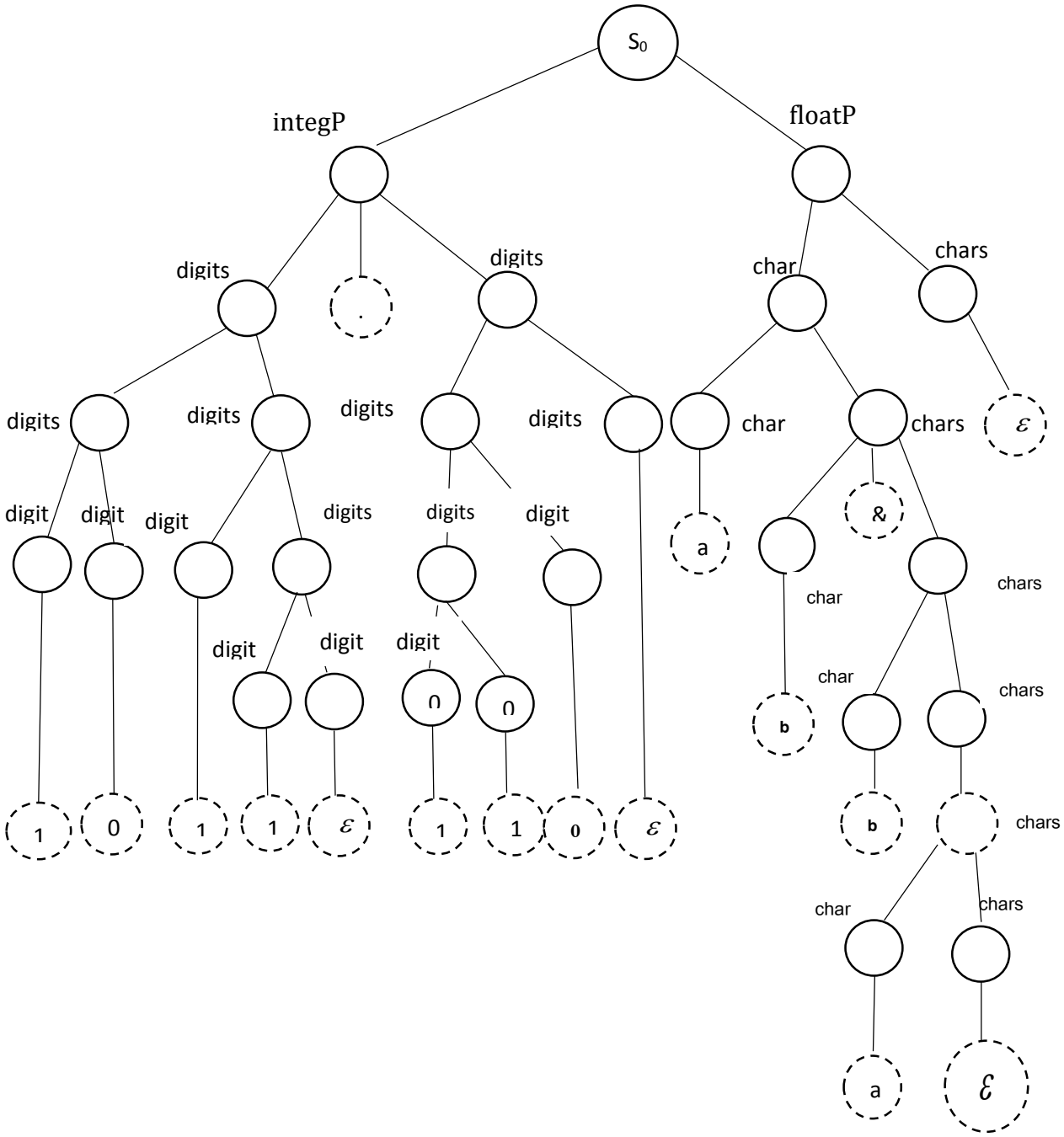

Fig.3.syntaxis tree of X&Y

27:< ser1 >$\overset{W27}{\longrightarrow}$< char >< chars > W_{27}: p_1=0.7, r_2: Nchars > 2

28:< chars >$\overset{W28}{\longrightarrow}$< chars >< char > W_{28}: p_1=0.3, r_3: fir = b

29:< char >$\overset{W29}{\longrightarrow}$< char >< chars > W_{29}: p_1=0.3, r_3: fir = a

30:< ser2 >$\overset{W30}{\longrightarrow}$< b >< chars > W_{30}: p_1=0.3, r_3:ser2.$fir = b$

31:< ser2 >$\overset{W31}{\longrightarrow}$< a >< chars > W_{31}: p_1=0.3, r_3: ser2.$fir = a$

32:< char >$\overset{W32}{\longrightarrow}$ a W_{32}: p_1=0.2, r_3: curr = a

33:< char >$\overset{W33}{\longrightarrow}$ b W_{33}: p_1=0.2, r_3: curr = b

34:< char >$\overset{W34}{\longrightarrow}$ ε W_{34}: p_1=0.1, r_3:char = ε

35:< ser2 >$\overset{W35}{\longrightarrow}$ ε W_{35}: p2=0.1, r4: ser2 = ε

36:< ser1 >$\overset{W36}{\longrightarrow}$ ε W_{36}: p_1=0.1, r_3: ser1 = ε

37:< ser1 >$\overset{W27}{\longrightarrow}$< digits >< digits > W_{27}: p_1=0.7, r_2: ser1 is digit?

Let's calculate The total value of the applying probability of stochastic production rules for executing the syntaxs analysis for string constant Y="ab&ba" is calculated as follows:

$< S_0 >\overset{0.25}{\longrightarrow}$< ser1 >< & >< ser2 >

$\overset{\mathbf{0.7}}{\rightarrow}$< char >< chars >< & >< ser2 > $\overset{\mathbf{0.3}}{\rightarrow}$< a >< chars >< & >< ser2 >

$\overset{\mathbf{0.3}}{\rightarrow}$< a >< b >< chars >< & >< ser2 >

$\overset{\mathbf{0.1}}{\rightarrow}$< a >< b >< ε >< & >< ser2 > $\overset{\mathbf{0.3}}{\rightarrow}$< a >< b >< ε >< & >< b >< chars >

$\overset{\mathbf{0.3}}{\rightarrow}$< a >< b >< ε >< & >< b >< a >< chars >

$\overset{\mathbf{0.1}}{\rightarrow}$ ab&ba ε → ab&ba

So the total value of the applying probability of stochastic production rules $P_{SS}(Y)$for executing the syntaxs analysis for string constant(see fig.3) is given as:

$$P_{SS}(Y)= P_{SS}("ab\&ba")=0.25x0.7x0.3x0.3x0.1x0.3x0.3x0.1=\boxed{1.417x10^{-5}}$$

If the input string series was correctly constructed the probability value of executing the syntaxs analysis for string constant can represent as:

$$P_{S.trad}(Y)= P_{S.trad}("ab\&ba")=0.25^5x0.7x0.3^4x0.2^2x0.1^3=\boxed{2.214x10^{-10}}$$

By using the traditional stochastic formal production rules the total value of the applying probability of production rules required for executing the syntaxs analysis for both **digital** and **string** series is calculated as follows:

$$P_{trad}= P_{SS}("ab\&ba")XP_{sd}("10110.110")=4.9766\,x10^{-13}X1.417x10^{-5}=\boxed{7.0518X10^{-18}}$$

Using production rules $1\sim5$ (stochastic formal grammar) permits to build 4 additional subtree for executing syntaxs analysis of different types structures of input strings(integer, float, string series).

The related probability values was defined everistically, where for each secondary element

$\rho_{total}=U_{i=1}^{k}$, $\rho_i \le 1$ $\quad Whrere: i-serial\ sequence\ number\ of\ production\ rule.$

From above it`s clear, only the production rule with start element has the applying probability *P=1*, and it is possible to generate only on type of series started with ay terminal element related to *LP* secondary element and there isn`t any possibility to generate for example string series like "ab & ba" (see fig.3).

4.INCREASING THE POWER OF MODEFIED STOCHASTIC FORMAL FORMAL GRAMMAR:

Let increasing the power of modified formal grammar by include some(suppose 3 rules) production stochastic rules with the same start element "S_0" and applying probability " P_i" add different right side and applying Boolean expressions "r_i" , where each production with start element can construct one specific subtree, so the power of grammar can increasing by the increasing the

number of production rules with start element lets add 3 new forms of production stochastic rules as follows:

a) $$S_0 \xrightarrow{W_1(p1,r1)} < Lp >< . >< Rp > \qquad (20)$$

This type can be used in executing syntaxs analysis of floating numbers with floating point or similar expression if $W_1(p1,r1)$ (set of attributes).

b) $$S_0 \xrightarrow{W_2(p1,r2)} < S_1 >< \& >< S_2 > \qquad (21)$$

Where:

S_1,S_2–string series, r_i–Boolean expression(set of conditions), P_i –applying probability.

This type can be used in executing syntaxs analysis of concatenated string expressions if the condition $W_2(p1,r1)$ (set of attributes) is yield.

c) $$<S_0> \xrightarrow{w3(p1,r3)} <ser1> <||><ser2>, \qquad (22)$$

This type can be used in executing syntaxs analysis of concatenated string expressions using Boolean logic operator if the condition $W_3(p1,r3)$ (set of attributes) is yield.

As seen it is possible to use any operator(logical, relational, arithmetic, Boolean logical) and it's possible to add any number of production rules with the start element keeping the structure of the formal grammar simple as possible.

In generally using different types of expressions forms and operators it is possible to generate many different structures of string.

Suppose we have "n_i" different structure of expressions where can be used "m_1 "operators, so the possibility to generate string series is increased by *value* "β" as follows:

$$\beta=(n_i*m_1)-1 \qquad (23)$$

for only starting element and at the same time the constructed formal grammar can generate very wide and powerful formal language which contains numeric and string series, so increasing the number of possible to generate string series.

For using the old everisticaly values "p_i" of applying probability of production rules, it possible to keep the values of p_i and specify using Boolean expression $r_i \quad ,i=\overline{1-n}$

for each production rule, in condition set of production rules with the same left secondary element has primary using Boolean expression $r_i, i=\overline{1-n}$ and the same applying probability $P_j, j=\overline{1-n}$.

Suppose the iteration number of production rules is 5 and taking in consideration the relations $20,\ 21,\ 22$ start symbol can modified as follows (suppose the iteration number of production rule is 1(see fig.2):

As seen from above using the same applying probability P_i we have "4" additional production rules which can generate all types of constants (integer, float, char, string series, identifiers) starting with/without using exponential function with the same start symbol.(see fig.3). By the same way the second production rules can possible modified as follows:

Now let's see how to construct the syntaxs tree of the input string X="ab&ba" depending on the stochastic formal grammar where the set of production rules is defined as follows: While P_{str} using traditional stochastic formal grammar can

$$P_{sstr}=(0.25)^4\times(0.4)^8\times(0.2)^7\times(0.5)^6\times(0.3)^2=\underline{4.3038\times10^{-14}}$$

For demonstration purpose suppose it's necessary to make syntaxs analysis of identifier also it's necessary to add the next production rules set to the structured above mentioned stochastic formal grammar(see fig.4).

38: $S_0 \xrightarrow{w38(p1,r38)} < digitSer >< stringSer >$ r_{38}: $digitConst$ is digit ? , p=0.25 .

39: $< digitSer > \xrightarrow{w39(p1,r39)} < digit >< digits >$ r_{39}: $digits = \varepsilon$? , P=0.3.

40: $< digits > \xrightarrow{w40(p1,r40)} < digit >< digits >,$ r_{40}: $digits \le 2$? ,P=0.25.

41: $< digits > \xrightarrow{w41(p1,r41)} < digits >< digit >,$ r_{41}: $digits > 2$? ,P=0.25.

42: $< digits > \xrightarrow{w42(p1,r42)} < digit >< digit >,$ r_{42}: $digits < 1$? ,P=0.25.

43: $< digits > \xrightarrow{w43(p1,r43)} < char >< + >< digit >,$ r_{43}: $char\ no\ digit$? ,P=0.25.

44: $< stringSer > \xrightarrow{w39(p1,r39)} \varepsilon$ r_{44}: $digitSer\ gets\ end?$, P=0.2.

Using syntaxs tree for digital constants (see fig.4) we will construct syntaxs tree for analyzing additional string series as follows:

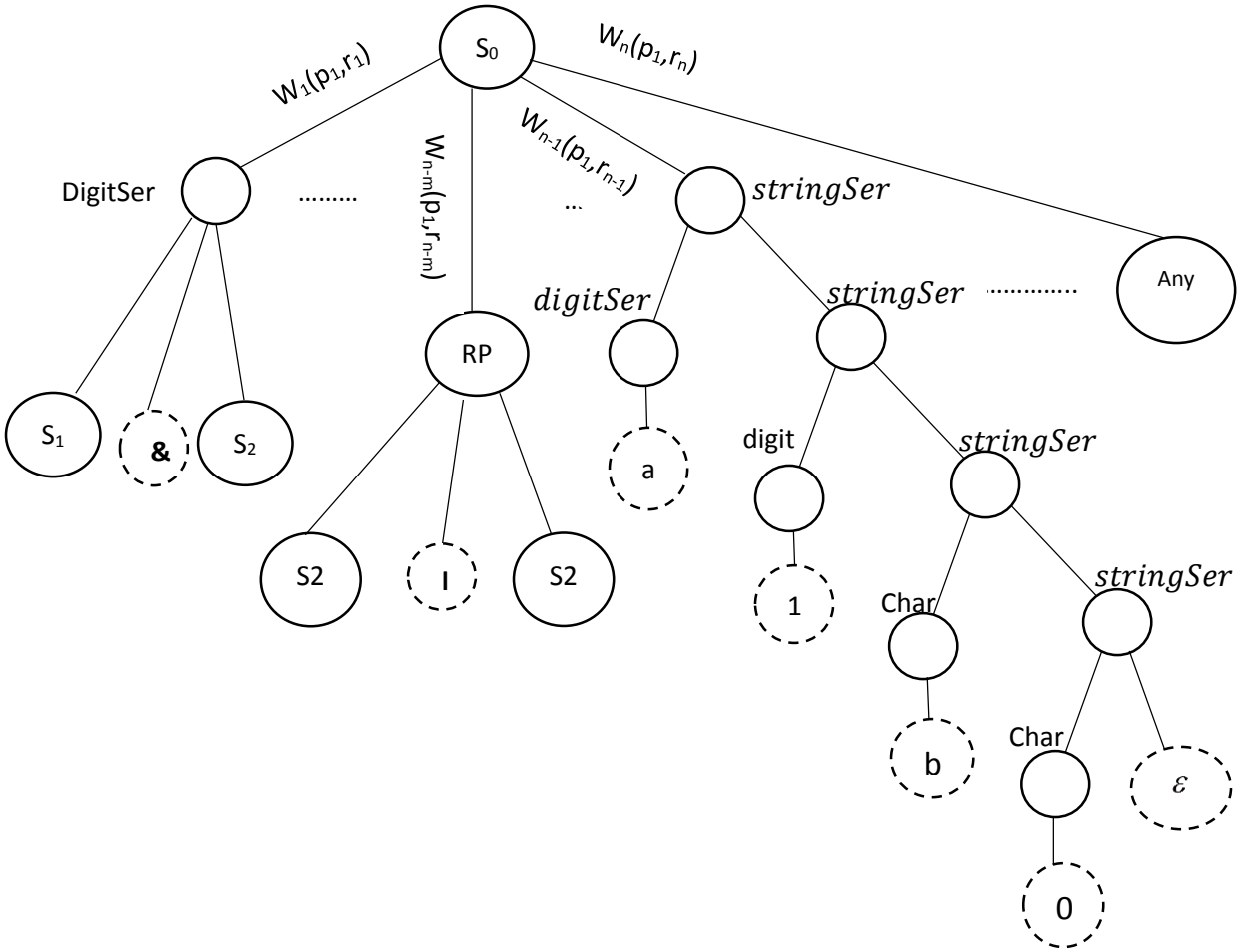

Fig.4.syntaxs tree for digital input series

Suppose the iteration of each production rule is only one ,so the total number of expected to use production rules is N_{total}=44.

The result of syntaxs analysis can recognize two conditions:

a)The syntaxs input string was successfully constructed, then the possible number of production rules to be use is calculated as follows:

1)For executing syntaxis analysis the digital series X_1="*1101.11*" the number of used production rules is $N_{cd}=27$ so the minimized number of production rules is calculated as follows: $N_{cd.min}= N_{total}-N_{cd}=44-27=17$

The ratio of using production rule is $Pcd.\min = \frac{27}{44} = 0.61$,t.e the required time for syntaxs analysis of correct constructed input digital series T_{cd} was minimized by $T_{cd}.min=17/44=0.38$.

2. For executing syntaxs analysis the string series X_2 ="*ab&ba*" the number of used production rules is $N_{cs}=11$ so the minimized of production rules is calculated as follows:

$N_{cs\cdot min}= N_{total}-N_{cs}=44-11=33$

the ratio of using $Pcs.\min = \frac{11}{44} = 0.25$ and the required time for syntaxs analysis of correct input string series T_{cs} was minimized by $T_{cs.min}=33/44=0.75$

b)The syntaxs input string was faulty constructed, then the possible number of production rules to be use for both types string and digital input series is has the same value t.e(see fig.4) $N_{FD}=N_{FS}=44$,so minimized number of production rules is calculated as follows:

$N_{cs\cdot min}= N_{total}-N_{cs}=44-44=0$

$N_{ds\cdot min}= N_{total}-N_{ds}=44-44=0$

It means no minimization in both conditions and the required time for syntaxs analysis takes the maximum value.

The number of production rules with the same and different right side element is 5, so it will be used for executing syntaxs analysis for digit string X_1 only one time, in result the ratio of using production rule with start symbol is $P_{start}=1/5=0.2$

So it is possible to calculate the speed factor " h " for syntaxis analysis digital series X_1 (suppose h_d)and string series X_2 (suppose h_S)as the deference between the number of used production rules in faulty constructed N_{fs} condition and in correctly constructed N_{cs} condition for digital and string series follow:

a)The speed factor h_S for string input series:

$h_S = \Delta_{string} = N_{fs} - N_{cs} = 44 - 11 = 33$

b)The speed factor h_d for digital input series:

$h_d = \Delta_{digit} = N_{fd} - N_{cd} = 44 - 27 = 17$

and the enhancing factor of using production rule ratio as:

$$h_{rs} = 1 - \frac{11}{44} = \underline{0.75} \qquad For\ input\ Y$$

$$h_{rd} = 1 - \frac{27}{44} = \underline{0.3863} \qquad For\ input\ X$$

We summarized all these result s in tab1.

Table 1. applying number of prod. Rules of stochastic &modified stochastic formal grammar

	Modified stochastic formal grammar		Stochastic formal grammar	
	N_{cs}	N_{cd}	N_{fs}	N_{fd}
$N_{used\ prod.\ number}$	11	27	44	44
λ (using ratio)	0.25	0.61	1	1
$N_{minimized}$	33	17	0	0
$T_{minimized}$	0.75	0.386	0	0

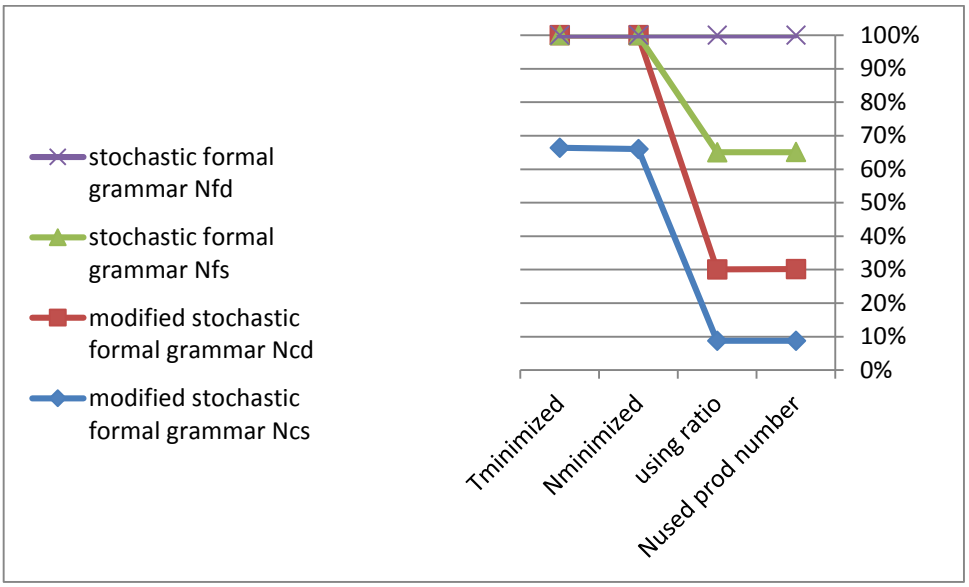

Fig.5.Relationship between stochastic &modified stochastic formal grammar

5.Conclusions:

-The recommended approach gives the possibility to construct syntaxs analysis tree for different input series with n various subtree started with the same start element which increase the power of generated formal languages.

-Supporting the ability to consist many parent different production rules with the same left side and various right side ,possible to diagnose and find the best way for executing the syntaxis analysis.

-through the syntaxs analysis it's possible to discard the set of production rules with the weight in "false" condition $W_i, \overline{i=1,n}$ =" false " so we get the maximum minimized number of used production rules in syntaxs analysis.

- Using the minimized number of production rules permits minimizing the required time for executing the syntaxis analysis of input string.

- minimizing the number of applied production rules and discard the production rules from using in syntaxs analysis permits decrease the required time for syntaxs analysis of different input string.

REFERENCES:

1.Luis M Augusto. Languages, Machines, and Classical Computation Paperback – February 4, 2019.

2.C Mag Staff ."Encyclopedia Definition of Compiler". PCMag.com. Retrieved 2017

3. Sun, Chengnian ; Le, Vu; Zhang, Qirun; Su, Zhendong (2016). "Toward Understanding Compiler Bugs in GCC and LLVM". ACM.

4.C Mag Staff ."Encyclopedia Definition of Compiler". PCMag.com. Retrieved 2017

5. Silberztein, Max (2013). "NooJ Computational Devices". Formalizing Natural Languages with NooJ. pp. 1–13. ISBN 978-1-4438-4733-9.

6. 35المجلد 2013،مجلة جامعة البعث" نمذجة المحادثة التعليمية المؤتمتة مع الكائن" د.يعرب ديوب

7.Dayoub.y,Predictive adaptive dynamic object's traversals control,Tartous,Volume3 ,N:6 2019.

8. Dayoub.y, converting indexed program formal grammar into free-context grammar ,Tartous. Volume 5,N:4 ,2020.