Analysis of Valid Closure Property of Formal Language

Chen Wenyu
School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: cwy@uestc.edu.cn

Wang Xiaobin
School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: xbwang@uestc.edu.cn

Cheng Xiaoou
School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: carlyhawk@gmail.com

Sun Shixin
School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: sxsun@uestc.edu.cn

***Abstract*—This paper focuses on the basic operations of Chomsky's languages. The validity and the effectiveness of some closure operations, such as union operator, product operator and Kleene Closure operator, are discussed in detail. The crosstalk problems in Context-Sensitive Languages (CSL) and Phrase Structure Languages (PSL) are analyzed, and a valuable method to solve this problem is presented by suing the alphabet of the operating languages. In addition, according to the valid closure property of regular languages (RL), a simple method to create a regular expression (RE) is proposed. The closure property of the permutation operator in Context-Free Languages (CFL) is proved and tested. In conclusion, by using our proposed methods, the exact type of a given language can be proved theoretically. By the way, the grammar to produce complex language can be created easy. Finally ,the constructing ε -NFA with the closure property is proved.**

***Index Terms*—language operation, valid closure property, crosstalk, context-free permutation**

I. Introduction

Given alphabet ∑,Ψ is a type of language of ∑, language $L_1,L_2 \in \Psi$,let α be a binary operation of the language:

$$(L_1,L_2)\rightarrow\alpha(L_1,L_2)$$

βis the unary operation of the language:

$$L_1\rightarrow\beta(L_1)$$

If for any language of Ψ, L_1 and L_2,$\alpha(L_1,L_2)$ is also a language of Ψ, then we say Ψ is closed on the operation ofα; if for any language L_1 of Ψ,$\beta(L_1)$ is also a language of Ψ, then we say that Ψ is closed on the operation ofβ[1].

For grammars generating languages, given a specified operation, the grammar of the same type language can be created, then the language is effectively closed on that operation.

The closure issue of language operation is important in language research and has significant value in both theory and practice[1,2,3].

Linz Peter proposed the crosstalk problem of context-dependent language and the corresponding solutions[1] without discussing the crosstalk problem of 4 types languages of Chomsky theory by Kleene closure operation. Prof. Jiang Zongli and Prof. Chen Youqi proved that for different alphabets, regular language and context-free language are effectively closed by basic language operations[2,3]. From the standpoint of automata, especially Turing machine, Michael Sipser discussed the valid closure property issue of language operations[4,5].

From the view of formal language, the paper proves that 4 types languages of Chomsky theory are effectively closed by join, product and Kleene closure operations. The paper proposes solution to crosstalk problem of context-dependent language and phrase structure language by product and Kleene closure operations and discusses the valid closure property of context-free language by context-free in-placement.

II. Language Classification

doi:10.4304/jcp.5.2.322-329

For any grammar $G=(\sum,V,S,P)$, G is type-0 grammar, or PSG(Phrase Structure Grammar). G generates type-0 language, or Phrase Structure Language correspondingly.

Grammar G, if for any $\alpha\rightarrow\beta\in P$, we have $|\alpha|\leq|\beta|$, then G is type-1 grammar, or Context-Sensitive Grammar(CSG). G generates type-1 language, or Context-Sensitive Language correspondingly.

If for any $\alpha\rightarrow\beta\in P$, we have $\alpha|\leq|\beta|$ and $\alpha\in V$, then G is type-2 grammar, or Context-Free Grammar(CFG). The language generated by G is type-2 language or Context-Free Language(CFL).

If for any $\alpha\rightarrow\beta\in P,\alpha\rightarrow\beta$, we have forms like $A\rightarrow w$ or $A\rightarrow wB$, in which, $A,B\in V$, $w\in\sum^+$,then G is type-3 grammar, or Regular Grammar(RG). Correspondingly, the language generated by G is type-3 language or Regular language(RL).

The basic principle to classify grammar disallows ε-formula in type-1, type-2 and type-3 grammars; if S is not on the right side of any formula of the grammar, and if G is type-1, type-2 or type-3 grammar, then

$$G'=(\sum,V,S,P\cup\{S\rightarrow\varepsilon\})$$
$$G''=(\sum,V,S,P-\{S\rightarrow\varepsilon\})$$

are still type-1, type-2 or type-3 grammars, and the languages correspondingly generated are also type-1, type-2 or type-3 languages.

III. BASIC LANGUAGE OPERATIONS S

Languages L_1 and L_2 are based on alphabet $\sum_1$ and $\sum_2$ respectively, the union operation of L_1 and L_2 is:

$$L_1\cup L_2$$
$$=\{w|w\in L_1 \text{ or } w\in L_2\}$$

the product operation of L_1 and L_2 is:

$$L_1L_2$$
$$=\{ w \mid w=w_1w_2, w_1\in L_1, w_2\in L_2\}$$

the Kleene closure operation (or Star operation) of L_1 is

$$L_1{}^*$$
$$=\{w|w=w_1w_2\ldots w_m, w_i\in L_1, m\geq 0\}$$
$$=\cup L_1{}^n \quad \text{对 } n\geq 0$$

IV. THE VALID CLOSURE PROPERTY OF LANGUAGE ON OPERATIONS

The valid closure property can be described as following: given same type grammars G_1 and G_2

$$L_1=L(G_1)$$
$$L_2=L(G_2)$$

Same type grammar G must be created to satisfy

$$L(G)=\alpha(L_1,L_2)$$

or

$$L(G)=\beta(L_1)$$

V. THE VALID CLOSURE PROPERTY OF BASIC OPERATIONS IN 4 TYPES OF LANGUAGES

Let language L_1 and L_2 attribute to the languages of alphabet $\sum_1$ and $\sum_2$ respectively, grammar G_1 generates language L_1

$$G_1=(\sum_1,V_1,S_1,P_1)$$

grammar G_2 generates language L_2

$$G_2=(\sum_2,V_2,S_2,P_2)$$

Then

$$S_1=>\alpha=>^*w_1\in L_1$$
$$S_2=>\beta=>^*w_2\in L_2$$

Suppose

$$\textstyle\sum_1\cap\sum_2=\Phi;\ V_1\cap V_2=\Phi;\ S\notin V_1;\ S\notin V_2$$

Set

$$\textstyle\sum=\sum_1\cup\sum_2$$
$$V=V_1\cup V_2\cup\{S\}$$

A. The Valid Closure Property on Union Operation

Create grammar

$$G_3=(\sum,V,S,P_3)$$

in which

$$P_3=\{S\rightarrow S_1\}\cup\{S\rightarrow S_2\}\cup P_1\cup P_2$$

For i=0, 1, 2, if G_1 and G_2 are type-i grammar, then G_3 is the same type grammar.

G_3 could use

$$S=>S_1=>\alpha=>^*w_1\in L_1$$

to obtain L_1; or use

$$S=>S_2=>\beta=>^*w_2\in L_2$$

to obtain L_2, that is

$$L(G_3)=L_1\cup L_2$$

So, languages of type-0,1,2 are effectively closed on union operation.

For example, type-2 grammar G_1 is

$$S_1\rightarrow aS_1a$$
$$S_1\rightarrow bS_1b$$
$$S1\rightarrow cS1c$$
$$S1\rightarrow a|b|c$$
$$S1\rightarrow aa|bb|cc$$

so L_1 is

$$\{x|x=x^T,x\in\{a,b,c\}^+\}$$

and type-2 grammar G_2 is

$$S_2\rightarrow AC$$
$$A\rightarrow 0A1$$
$$A\rightarrow 01$$
$$C\rightarrow 2|2C$$

so L_2 is

$$\{0^n1^n2^m|n.m>0\}$$

Set type-2 grammar G_3 is

$$S\rightarrow S_1$$
$$S\rightarrow S_2$$
$$S_1\rightarrow aS_1a$$
$$S_1\rightarrow bS_1b$$
$$S_1\rightarrow cS_1c$$
$$S_1\rightarrow a|b|c$$
$$S_1\rightarrow aa|bb|cc$$
$$S_2\rightarrow AC$$
$$A\rightarrow 0A1$$
$$A\rightarrow 01$$
$$C\rightarrow 2|2C$$

so L_3 is

$$\{x|x=x^T,x\in\{a,b,c\}^+\}\cup\{0^n1^n2^m|n.m>0\}$$

that is

$$L_3=L_1\cup L_2$$

If G_1 and G_2 are type-3 grammar while G_3 is not type-3 grammar, then create type-3 grammar

$$G_4=(\sum,V,S,P_4)$$

in which

$$P_4=\{S\rightarrow\alpha|S_1\rightarrow\alpha\in P_1\}$$

$$\cup \{S\rightarrow\beta | S_2\rightarrow\beta \in P_2\}$$
$$\cup P_1 \cup P_2$$

then G_4 is a type-3 grammar.

G_4 could use

$$S \Rightarrow \alpha \Rightarrow^* w_1 \in L_1$$

to obtain L_1; or use

$$S \Rightarrow \beta \Rightarrow^* w_2 \in L_2$$

to obtain L_2, that is,

$$L(G_4)=L_1 \cup L_2$$

So, language of type-3 is effectively closed by union operation.

The method to create G_4 can also be used to create grammars of type-0,1,2.

For example, type-3 grammar G_1 is

$$S_1\rightarrow aS_1|aA$$
$$A\rightarrow bA|bB$$
$$B\rightarrow cB|c$$

so L_1 is

$$a^+b^+c^+$$

and type-3 grammar G_2 is

$$S_2\rightarrow 0|0C$$
$$C\rightarrow 0|1|0C|1C$$

so L_2 is

$$0(0+1)^*$$

Set type-3 grammar G_4 is

$$S\rightarrow aS_1|aA$$
$$S\rightarrow 0|0C$$
$$S_1\rightarrow aS_1|aA$$
$$A\rightarrow bA|bB$$
$$B\rightarrow cB|c$$
$$S_2\rightarrow 0|0C$$
$$C\rightarrow 0|1|0C|1C$$

so L_4 is

$$a^+b^+c^+0\{0+1\}^*$$

that is

$$L_4=L_1 \cup L_2$$

B. The Valid Closure Property on Product Operation

Create grammar

$$G_5=(\Sigma,V,S,P_5)$$

in which

$$P_5=\{S\rightarrow S_1S_2\} \cup P_1 \cup P_2$$

For i=0, 1, 2, if G_1 and G_2 are grammar of type-i, then G_5 is also type-i grammar.

G_5 uses

$$S \Rightarrow S_1S_2 \Rightarrow \alpha\beta \Rightarrow^* w_1w_2 \in L_1L_2$$

to obtain the product of L1 and L2.

That is, $L(G_5)=L_1L_2$

So, type-0, type-1, type-2 languages are closed on product operation.

For example, type-2 grammar G_1 is

$$S_1\rightarrow aS_1a$$
$$S_1\rightarrow bS_1b$$
$$S_1\rightarrow cS_1c$$
$$S_1\rightarrow dS_1d$$
$$S_1\rightarrow aa|bb|cc|dd$$

so L_1 is

$$\{xx^T \mid x \in \{a,b,c,d\}^+\}$$

and type-2 grammar G_2 is

$$S_2\rightarrow AC$$
$$A\rightarrow 0A|0$$
$$C\rightarrow 1C2|12$$

so L_2 is

$$\{0^n1^m2^m | n.m>0\}$$

Set type-2 grammar G_5 is

$$S\rightarrow S_1S_2$$
$$S_1\rightarrow aS_1a$$
$$S_1\rightarrow bS_1b$$
$$S_1\rightarrow cS_1c$$
$$S_1\rightarrow dS_1d$$
$$S_1\rightarrow aa|bb|cc|dd$$
$$S_2\rightarrow AC$$
$$A\rightarrow 0A|0$$
$$C\rightarrow 1C2|12$$

so L_5 is

$$\{xx^T \mid x \in \{a,b,c,d\}^+\}\{0^n1^m2^m | n.m>0\}$$

that is

$$L_5=L_1L_2$$

If G_1 and G_2 are type-3 grammar while G_5 is not type-3 grammar, create type-3 grammar,

$$G_6=(\Sigma,V_1 \cup V_2,S_1,P_6)$$

in which

$$P_6=\{A\rightarrow wS_2 | A\rightarrow w \in P_1\}$$
$$-\{A\rightarrow w\}$$
$$\cup P_1 \cup P_2$$

For every formula like

$$A\rightarrow w$$

rewritten as

$$A\rightarrow wS_2$$

Grammar G_6 uses

$$S_1 \Rightarrow^+ r_1r_2\ldots r_kA$$
$$\Rightarrow r_1r_2\ldots r_kwS_2$$
$$\Rightarrow^* w_1w_2 \in L_1L_2$$

in which, $r_1r_2\ldots r_kw \in L_1$, that is,

$$L(G_6)=L_1L_2$$

So,language of type-3 is effectively closed on product operation.

For example, type-3 grammar G_1 is

$$S_1\rightarrow aS_1|aA$$
$$A\rightarrow bA|cA|bB|cB$$
$$B\rightarrow dB|d$$

so L_1 is

$$a^+(b+c)^+d^+$$

and type-3 grammar G_2 is

$$S_2\rightarrow 0C$$
$$C\rightarrow 0|1|0C|1C$$

so L_2 is

$$0\{0+1\}^+$$

Set type-3 grammar G_6 is

$$S_1\rightarrow aS_1|aA$$
$$A\rightarrow bA|cA|bB|cB$$
$$B\rightarrow dB$$
$$B\rightarrow dS_2$$
$$S_2\rightarrow 0C$$

$$C \to 0|1|0C|1C$$

so L_6 is

$$a^+(b+c)^+d^+0\{0+1\}^+$$

that is

$$L_6=L_1L_2$$

C. The crosstalk of Product operation

G_1 and G_2 are type-0 or type-1 grammar, if

$$\Sigma_1 \cap \Sigma_2 \neq \Phi \ (\Sigma_1=\Sigma_2 \text{ is possible})$$

the grammar G_5 is not always correct. For example:

Grammar G_1:

$$S_1 \to a$$

Grammar G_2:

$$S_2 \to aS_2$$
$$aS_2 \to bc$$

then

$$L_1=\{a\}, L_2=\{a^*bc\}$$
$$L_1L_2= a^+bc$$

However, if G_5 uses

$$S \Rightarrow S_1S_2 \Rightarrow aS_2 \Rightarrow a^+S_2 \Rightarrow^+ a^+bc$$

there can also be

$$S \Rightarrow S_1S_2 \Rightarrow aS_2 \Rightarrow bc$$

the language generated by grammar G_5 is

$$a^*bc \neq L_1L_2= a^+bc$$

The crosstalk between sentence patterns generated by S_1 and S_2 is the reason why G_5 is not we want sometimes. Namely, the sentence pattern generated by S_1 might take for the sentence generated by S_2 as the following text, while the sentence generated by S_2 might take for the sentence generated by S_1 as the preceding text; and the crosstalk could only be caused by the terminal symbol.

To solve the problem above, copy Σ as Σ' and Σ''

$$\Sigma' =\{x' \mid x \in \Sigma\}$$
$$\Sigma'' =\{x'' \mid x \in \Sigma\}$$

Replace x in P_1 by x' and then obtain P', replace x in P_2 by x'' and then obtain P'', the process is to distinguish the terminator symbols between G_1 and G_2 in deduction. Finally, x' and x'' need to be restored to the original terminator symbols.

Create grammar

$$G_7=(\Sigma, V \cup \Sigma' \cup \Sigma'', S, P_7)$$

in which

$$P_7= \{ S \to S_1S_2 \} \cup P' \cup P''$$
$$\cup \{x' \to x \mid x \in \Sigma\}$$
$$\cup \{x'' \to x \mid x \in \Sigma\}$$

G_7 uses

$$S \Rightarrow S_1S_2 \Rightarrow^+ w_1' \ w_2'$$
$$\Rightarrow^+ w_1w_2 \in L_1L_2 \Rightarrow \in L_1L_2$$

to obtain the product of L_1 and L_2, that is,

$$L_7=L_1L_2$$

thus, the crosstalk problem is solved.

In the example above,

Grammar G_1:

$$S_1 \to a$$

Grammar G_2:

$$S_2 \to aS_2$$
$$aS_2 \to bc$$

P_7 is

$$S \to S_1S_2$$
$$S_1 \to a'$$
$$S_2 \to a'' S_2$$
$$a'' S_2 \to b'' c''$$
$$a' \to a$$
$$a'' \to a$$
$$b'' \to b$$
$$c'' \to c$$

G_7 uses

$$S \Rightarrow S_1S_2$$
$$\Rightarrow a' \ S_2 \quad \text{//Can't use } a'' S_2 \to b'' c''$$
$$\Rightarrow a' \ a'' S_2$$
$$\Rightarrow^+ a' \ a''{}^* b'' c''$$
$$\Rightarrow^+ a^+bc$$

to create the product language a^+bc of L_1 and L_2.

D. The Valid Closure Property on Kleene Closure operation

The generation of sentence ε and any number of products must be considered in Kleene Closure operation.

Adding a formula

$$S \to \varepsilon \mid SS_1$$

to generate empty sentence and any number of products of L_1.

Since S is on the right side of the formula, which is not satisfied the principle of closure, and can generate other extra strings so we add a new non-terminal symbol to solve the problem.

Rewrite the newly added formula,

$$S \to \varepsilon \mid S'$$
$$S' \to S_1 \mid S_1S'$$

then only ε and $S_1^n (n \geqslant 1)$ can be deducted from S.

Create grammar

$$G_8=(\Sigma, V_1 \cup \{S, S'\}, S, P_8)$$

in which

$$P_8=\{S \to \varepsilon \mid S'\} \cup \{S' \to S_1 \mid S_1S'\} \cup P_1$$

If G_1 is type-2 grammar, then G_8 is also type-2 grammar and

$$L(G_8)=L_1^*$$

So, language of type-2 is closed on Kleene Closure. If G_1 is type-0 or type-1 grammar, grammar G_8 may also has crosstalk problem. That because

$$S \Rightarrow^+ S_1 \cdots S_1S_1 \cdots S_1$$

each S_1 could only generate sentence of L1 from the formula of P_1, and the sentence patterns generated by any two consecutive S_1 might be following and preceding text with each other, then crosstalk is appear.

To avoid crosstalk, copy Σ as Σ' and Σ'', create P' and P''; rewrite S_1 as S', create grammar

$$G' =(\Sigma, V_1 \cup \Sigma' \cup \{S'\}-\{S_1\}, S', P')$$

Rewrite S_1 as S'', create grammar

$$G'' = (\Sigma, V_1 \cup \Sigma'' \cup \{S''\}-\{S_1\}, S'', P'')$$

Create grammar

$$G_9=(\Sigma, V_1 \cup \Sigma' \cup \Sigma'' \cup \{S', S'', S_1, S_2\}, S, P_9)$$

in which

$$P_9= \{S \to \varepsilon \mid S_1 \mid S_2\}$$

$$\cup \{ S_1 \to S' \mid S' S_2\}$$
$$\cup \{ S_2 \to S'' \mid S'' S_1\}$$
$$\cup P' \cup P''$$
$$\cup \{x' \to x \mid x \in \Sigma\} \cup \{x'' \to x \mid x \in \Sigma\}$$

To avoid crosstalk itself, S' and S'' must be alternated to satisfy:

$$S \Rightarrow S_1 \Rightarrow S' S'' S' S'' \cdots S' S''$$

or

$$S \Rightarrow S_1 \Rightarrow S' S'' S' S'' \cdots S'$$

and

$$S \Rightarrow S_2 \Rightarrow S'' S' S'' S' \cdots S'' S'$$

or

$$S \Rightarrow S_2 \Rightarrow S'' S' S'' S' \cdots S''$$

then the consecutive S_1 are replaced by alternated S' and S'', each S' and S'' could only deduce from the formula of P' or P'' respectively, and crosstalk is avoided.

S' and S'' each generates language of alphabet Σ' and Σ'' (The sentence structures are equal to the sentence structure of L_1), then after restoration, L_1* is obtained, that is

$$L(G_9)=L_1^*$$

So, language of type-0 and type-1 are closed on Kleene Closure operation.

For Example, type-1 grammar G_1 is

$$S_1 \to aS_1BC$$
$$S_1 \to aBC$$
$$CB \to BC$$
$$aB \to ab$$
$$bB \to bb$$
$$bC \to bc$$
$$cC \to cc$$

so L_1 is

$$\{a^n b^n c^n \mid n>0\}$$

Set Σ' is

$$\{ a', b', c' \}$$

Set Σ'' is

$$\{a'', b'', c'' \}$$

Set type-1 grammar G' is

$$S' \to a' S' B' C'$$
$$S \to a' B' C'$$
$$C' B' \to B' C'$$
$$a' B' \to a' b'$$
$$b' B' \to b' b'$$
$$b' C' \to b' c'$$
$$c' C' \to c' c'$$

Set type-1 grammar G'' is

$$S'' \to a'' S'' B'' C''$$
$$S'' \to a'' B'' C''$$
$$C'' B'' \to B'' C''$$
$$a'' B'' \to a'' b''$$
$$b'' B'' \to b'' b''$$
$$b'' C'' \to b'' c''$$
$$c'' C'' \to c'' c''$$

Set type-1 grammar G_9 is

$$S \to \varepsilon \mid S_1 \mid S_2$$
$$S_1 \to S' \mid S' S_2$$
$$S_2 \to S'' \mid S'' S_1$$
$$S' \to a' S' B' C'$$
$$S \to a' B' C'$$
$$C' B' \to B' C'$$
$$a' B' \to a' b'$$
$$b' B' \to b' b'$$
$$b' C' \to b' c'$$
$$c' C' \to c' c'$$
$$S'' \to a'' S'' B'' C''$$
$$S'' \to a'' B'' C''$$
$$C'' B'' \to B'' C''$$
$$a'' B'' \to a'' b''$$
$$b'' B'' \to b'' b''$$
$$b'' C'' \to b'' c''$$
$$c'' C'' \to c'' c''$$
$$a' \to a$$
$$b' \to b$$
$$c' \to c$$
$$a'' \to a$$
$$b'' \to b$$
$$c'' \to c$$

so L_9 is

$$\{a^n b^n c^n \mid n>0\}^*$$

that is

$$L_9=L_1^*$$

If G_1 is type-3 grammar while G_8 is not type-3 grammar, add new starting symbol S and

$$S \to \varepsilon$$

ε is generated,add

$$S \to r$$

in which

$$S_1 \to r \in P_1$$

to deduce (r=wB or r=w)。

For every formula like $A \to w$, add

$$A \to wS_1 \text{ (} A \to w \text{ is not deleted)}$$

from S, the sentence pattern could be deduced,

$$r_1 r_2 \ldots r_k A$$

in which

$$r_1, r_2, \cdots, r_k \in L_1$$

Stop deduction when

$$r_1 r_2 \ldots r_k w$$

is deduced or having deduced another sentence from

$$r_1 r_2 \ldots r_k w S_1$$

until L_1*.

G_1 is type-3 grammar, create -3type grammar,

$$G_{10}= (\Sigma, V_1 \cup \{S\}, S, P_{10})$$

in which

$$P_{10}= \{S \to \varepsilon \} \cup (P_1 - \{S_1 \to \varepsilon \})$$
$$\cup \{ S \to r \mid S_1 \to r \in P_1\}$$
$$\cup \{A \to wS_1 \mid A \to w \in P_1\}$$

then

$$L(G_{10})=L_1^*$$

So, language of type-3 is closed on Kleene Closure operation.

For example, type-3 grammar G_1 is

$$S_1 \to aS_1|bS_1$$
$$S_1 \to aA|bB$$
$$A \to aA|bA$$
$$A \to aC$$
$$B \to aB|bB$$
$$B \to bC$$
$$C \to a|b$$

so L_1 is

$$(a+b)^*a(a+b)^* a(a+b)+ (a+b)^*b(a+b)^* b(a+b)$$

Set type-3 grammar G_{10} is

$$S \to \varepsilon$$
$$S \to aS_1|bS_1$$
$$S \to aA|bB$$
$$A \to aA|bA$$
$$A \to aC$$
$$B \to aB|bB$$
$$B \to bC$$
$$C \to a|b$$
$$C \to aS_1|bS_1$$

so L_{10} is

$$((a+b)^*a(a+b)^*a(a+b)+ (a+b)^*b(a+b)^*b(a+b))^*$$

that is

$$L_{10}=L_1^*$$

Therefore, whether alphabet

$$\Sigma_1 \cap \Sigma_2 = \Phi$$

or

$$\Sigma_1 \cap \Sigma_2 \neq \Phi \quad (\Sigma_1=\Sigma_2 \text{ is included})$$

language of type-0,type-1, type-2 and type-3 are closed on union, product and Kleene Closure operations.

VI. THE CREATION OF REGULAR EXPRESSION

For regular language, regular expression can be generated as the method above.

R_1 and R_2 are regular expressions of language L_1 and L_2.

Suppose

$$L=L_1 \cup L_2$$

regular expression of L is $(R_1)+(R_2)$

$$L=L_1L_2$$

regular expression of L is $(R_1)(R_2)$

$$L=L_1^*$$

regular expression of L is $(R_1)^*$

VII. CFL IS EFFECTIVELY CLOSED TO CONTEXT-FREE IN-PLACEMENT

For context-free language, there is another useful operation, that is in-place operation[1].

Suppose X and Y are alphabets, mapping

$$g: X^* \to Y^*$$

if

$$g(\varepsilon) = \varepsilon$$

and for any $n \geqslant 1$

$$g(x_1x_2\cdots x_n)=g(x_1)g(x_2)\cdots g(x_n)$$

in which

$$x_i \in X$$
$$g(x_i)=y \in Y^*$$

or

$$g(x_i)=\{y_1, y_2, \cdots\}$$

then g is a context-free in-placement.

If L is a language of alphabet X, then

$$g(L)=\cup g(w)$$

in which

$$w \in L$$

Context-free grammar G=（X,V,S,P）, generates context-free language L , g is a context-free in-placement:

$$g(x)= L_x$$

in which

$$x \in X$$

Copy X as X′

$$X' =\{x' | x \in X\}$$

for every formula of P, replace the terminal symbol x on the right side by x′ , and P′ is obtained.

Rewrite G as:

$$G' = (Y,V \cup \Sigma', S,P')$$

The language generated by grammar G is based on alphabet X, and the language generated by grammar G′ is based on alphabet X′ . The sentence structures of the languages are all the same. (Only differ in alphabet.)

For every x′ , add a group of context-free formulas to satisfy:

$$x' \Rightarrow^+ L_x$$

P″ is obtained.

Create context-free grammar, #

$$G'' = (Y,V \cup \Sigma', S,P'')$$

Grammar G generates

$$x_1x_2\ldots x_n$$

Grammar G″ first uses P′ to generate

$$x_1' x_2' \cdots x_n'$$

and then uses the new formulas to obtain

$$L_{x_1} L_{x_2} \cdots L_{x_n}$$

Language g(L) generated by grammar G″ is also context-free. For example,

Context-free grammar G generates a^nb^n for

$$S \to aSb$$
$$S \to ab$$

Suppose context-free in-placement is:

$$g(a)=0^+= L_a$$
$$g(b)=101^*= L_b$$

Create grammar G′

$$S \to a' S b'$$
$$S \to a' b'$$

$a'^n b'^n$ is generated

Add formula

$$a' \to 0|0a'$$

0^+ is generated。

Add formula

$$b' \to 10|10A$$

$$A \rightarrow 1|1A$$

101* is generated

Create G″

$$S \rightarrow a' \ S \ b'$$
$$S \rightarrow a' \ b'$$
$$a' \rightarrow 0|0 \ a'$$
$$b' \rightarrow 10|10A$$
$$A \rightarrow 1|1 \ A$$

language $0^+(101^*)^+$ is generated. .

VIII. Constructing NFA with the Closure Property

Suppose L1, L2 be two type-3 languages, the DFA which receive these two languages is

$$M_1=(Q_1,\sum_1,\delta_1,q_1,\{f_1\})$$

and

$$M_2=(Q_2,\sum_2,\delta_2,q_2,\{f_2\})$$

Suppose Q_1 and Q_2 not be intersect.

Construct

$$\varepsilon\text{-NDA}=(Q_1 \cup Q_2 \cup \{q_0,f_0\}, \sum_1 \cup \sum_2,\delta,q_0,\{f_1\} \cup \{f_2\})$$

function δ is

$$\delta(q_0,\varepsilon) = q_1$$
$$\delta(q_0,\varepsilon) = q_2$$

to all states $q \in Q_1, a \in \sum_1 \cup \{\varepsilon\}$

$$\delta(q,a) = \delta_1(q,a)$$

to all states $q \in Q_2, b \in \sum_2 \cup \{\varepsilon\}$

$$\delta(q,b) = \delta_2(q,b)$$

This can be shown visually as Fig.1.

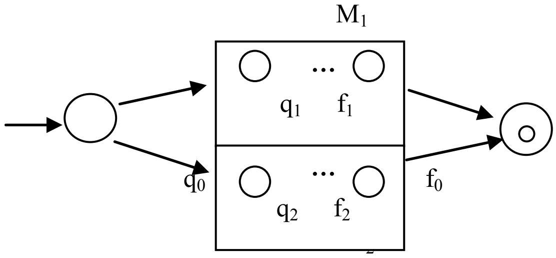

Figure 1ε -NDA for union operator

This ε -NDA concludes all function δ of M_1 and M_2,and adds 4δ functions that scan ε ,then we get: setting out from the ε -NDA beginning appearance, passing twoε actions:

$$\delta(q_0,\varepsilon) = q_1$$

and

$$\delta(q_0,\varepsilon) = q_2$$

it can arrive the beginning appearance q_1 or q_2 of M_1 or M_2,then, with the usage of own δ function that belong to M_1 or M_2, it can reach the only receiving states f_1 or f_2, finally, enter the only receiving states f_0.

Obviously, the language thatε -NDA receive is union of $L(M_1)$ and $L(M_2)$.

Construct

$$\varepsilon\text{-NDA} = (Q_1 \cup Q_2,\sum_1 \cup \sum_2,\delta,q_1,\{f_2\})$$

function δ is :

to all states $q \in Q_1\text{-}\{f_1\}, a \in \sum_1 \cup \{\varepsilon\}$

$$\delta(q,a) = \delta_1(q,a)$$
$$\delta(f_1,\varepsilon) = \{q_2\}$$

to all states $q \in Q_2\text{-}\{f_2\}$,b i$\in \sum_2 \cup \{\varepsilon\}$

$$\delta(q,b) = \delta_2(q,b)$$

This can be shown visually as Fig.1.

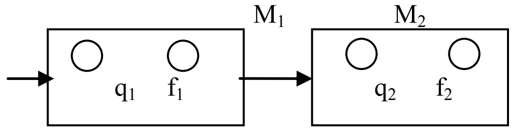

Figure 2ε -NDA for product operator

This ε -NDA concludes all function δ of M_1 and M_2,and adds one δ functions that scan ε ,then we get: setting out from the beginning states q_1 of M_1, with the usage of its ownδ function, can reach the only receiving state f_1,then, using the new added function

$$\delta(f_1,\varepsilon) = \{q_2\}$$

it get the beginning state q_2 of M_2, as the same ,with the own δ function of M_2, it can reach the only receiving appearance f_2(it is also the only receiving state of ε -NDA), then receive strings from language $L(M_2)$.

Obviously, the language that ε -NDA receive is product of languages $L(M_1)$ and $L(M_2)$.

Construct

$$\varepsilon\text{-NDA}=(Q_1 \cup \{q_0,f_0\},\sum_1,\delta,q_0,\{f_0\})$$

functionδ is：

$$\delta(q_0,\varepsilon) = q_1$$
$$\delta(q_0,\varepsilon) = f_0$$
$$\delta(f_1,\varepsilon) = \{q_0,f_0\}$$

to all appearance $q \in Q_1\text{-}\{f_1\}, a \in \sum_1 \cup \{\varepsilon\}$

$$\delta(q,a) = \delta_1(q,a)$$

This can be shown visually as Fig.3.

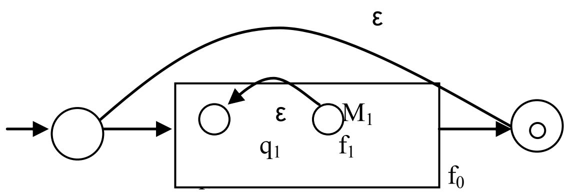

Figure 3ε -NDA for Kleene Closure operator

Thisε -NDA concludes all function δ of M_1,and adds 4 δ functions that scan ε ,then we get: setting out from theε -NDA beginning appearance, passing twoε actions:

$$\delta(q_0,\varepsilon) = q_1$$

and

$$\delta(q_0,\varepsilon) = f_0$$

it can straightly reach the only receiving states f_0(in order to receive null stringε),or reach the beginning state q_1 of M_1,then, setting out from the beginning state q_1,using the own δ function of M_1,it can reach the only receiving state f_1, at that time, pass twoε actions, it straightly get the receiving state f_0 so that it can finish this receiving process; also ,this state can be changed to the beginning sate q_1 of M_1, in order to receiving strings.

Obviously, the language thatε -NDA receive is the Kleene Closure of $L(M_1)$.

IX. CONCLUSION

Usually, complex language could be decomposed into several simple languages of the same type and re-composed by union, product and Kleene closure operations. The paper proves that the 4 types of language of Chomsky theory are effectively closed on the above three operations, and proposes a general method to create grammar of complex languages.

The valid closure property of positive closure operation can be referred to the effective closure of Kleene closure without considering the generation of sentenceε .

The closure of other operations, like intersection and complementary operations are not discussed in this paper.

We can construct NFA with the closure of language calculation.

ACKNOWLEDGMENT

This paper is supported by the the Si Chuan science and technology (2006J13-068).

REFERENCES

[1] Peter Linz .An introduction to formal languages and automata[M]. Boston :Jones and Bartlett,2001.

[2] JIANG Zong-Li and JIANG Shou-Xu, Theory of Formal languages and Automata[M], Tsinghua University Press, Beijing, China, 2007 (in Chinese).

[3] CHEN You-Qi, Formal Languages and Automata[M], Nankai University Press, Tianjing, China, 1999 (in Chinese).

[4] Michael Sipser. Introduction to the theory of computation[M]. New York: Thomson, 1996.

[5] J.E. Hopcroft and J.D. Ullman.Introduction to Automata Theory, Languages and Computation [M]. 2nd ed. Reading, MA: Addison-Wesleg, 2000

Chen Wenyu, male, born in 1968. He is an associate professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interest include: compiling technique, pattern recognition, formal language and automata.

He received the B.S. and M.S. degrees in computer science and engineering from University of Electronic Science and Technology of China(UESTC),Chengdu, China, in 1990 and 1993, respectively. He was a researcher at the School of Computer Science and Engineering,UESTC. From 2004, he was a associate professor at the School of Computer Science and Engineering, UESTC. His research interests include computer Language and compile, formal language and automation, and neural networks.

He is a mentor for postgraducate student;a grade college teacher (for undergraduate);a quality outstanding young teacher (for postgraducate).

He completed a number of research tasks, both through the Ministry, the provincial identification; publish over 20 papers and 6 teaching materials.

Associate prof. Chen is a senior member of China Computer Federation(E200011786S).

Wang Xiaobin, male, born in 1964.

He is an associate professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interests include data structure, computer Language and compile, software engineering, neural networks, and optimization.

He received the B.S. and M.S. degrees in computer science and engineering from University of Electronic Science and Technology of China, Chengdu, China, in 1985 and 1988, respectively. He is currently working toward the Ph.D. degree in Computational Intelligence Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China. From 1988 to 1998, he was a researcher at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. From 1998, he was a Professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He is currently the dean of ChengDu College, University of Electronic Science and Technology of China.

Cheng Xiaoou, female, born in 1984.

She is candidate for Master's degree in Software Engineering, School of Software, University of Electronic Science & Technology of China. Her research interest include: compiling technique, theory of software.

Sun Shixin, male, born in 1940.

He is a Ph.D supervisor and professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interest include: theory of Computer Science, Parallel Computer Systems, Numerical Algorithm & Non-Numerical Algorithm (including Parallel Algorithm).

Prof Sun 1966. B.A. in Mathematics,Sichuan University, Chengdu, China;1984-1987. visiting scholar at the Grenoble University, France,;1990. doing research at Roma University, Italy and Grenoble;University, France.