GENERATIVE GRAMMARS IN ACTIONS AGGREGATION

Vasile CRĂCIUNEAN[1]

ABSTRACT: The grammar generating a context free language gives us an internal hierarchy of the language. When such a type two grammar is used to model a manufacturing process, each nonterminal will be naturally associated with an intermediate resource and each rule will be an action in the production process. The key to mastering the complexity of large complex production processes is hierarchical modularization, i.e. group actions in cohesive blocks with minimal interaction. Everyone agrees that a production process must be decomposed into hierarchical blocks to be driven and optimized. The derivation tree of the generative grammar that models the production process may underlie solving these problems because it provides a hierarchical structure of both resources and actions of the production process. At each hierarchical level we have a lot of resources, a lot of actions and a lot of rules for generating. For this we consider that the semantics of the language coincides with the production process that shapes it.

KEY WORDS: action of aggregation, aggregation block of actions, application of aggregation, application of disaggregation.

1 INTRODUCTION

The grammar generating a context free language gives us an internal syntactic and semantic hierarchy of the language elements. For this we consider the semantics of the language coincides with the production process.

Basically each production process is associated with a generative grammar as follows

i)Each non-terminal symbol of the grammar represents a resource.

ii)Each terminal symbol of the grammar is an action.

ii)Each production is associated with an action, i.e. if $A \rightarrow \alpha \in P$ and $\alpha = r_1 r_2 ... r_n a$, then the action a associated with this production operates on the resources $r_1, r_2, ..., r_n$ and produces the resource represented by the nonterminal symbol A.

In this paper we consider that the atomic unit of a production process is the action.

Actions and the resources on which the actions are produced are distributed for concurrent execution in a virtual network of companies working together. Actions are very inhomogeneous beginning with actions of design, procurement, logistics, manufacturing, quality control etc. There will be software actions that will store and dynamically process all data on the evolution of the production process.

Each component embeds in it a certain degree of intelligence. It will know how to check if it has provided the necessary context for starting, for example it will know to check if it has the necessary resources for the smooth conduct. If this conditions are not satisfied it will wait until these conditions are met. The wait will last until it gets a message about updating the input resources and then it will restart the process with context verification. The action will know how to decrement stock, for example, with consumed resources and increment the stock with the resource created by it and also sending the corresponding event. An action of aggregation also does a very important thing that is real-time update of developments related to various data such as start time, the actual length, waiting time, costs, etc.

2 PRELIMINARY NOTIONS AND NOTATIONS

Let X be a set. The family of subsets of X is denoted by $P(X)$. The cardinality of X is denoted by $|X|$. The set of natural numbers, {0, 1, 2, ...} is denoted by N. The empty set is denoted by Ø. An alphabet is a finite nonempty set of abstract symbols. For an alphabet V we denote by V^* the set of all strings of symbols in V. The empty string is denoted by λ. The set of nonempty strings over V, that is $V^*-\{\lambda\}$, is denoted by V^+. Let Sub(w) denote the set of subwords of w. Each subset of V^* is called a language over V. The length of a string $x \in V^*$ (the number of symbol occurrences in X) is denoted by $|x|$. The number of occurrences of a given symbol $a \in V$ in $x \in V^*$ is denoted by $|x|_a$. Let Symb(w) denote the set of symbol occurrences in w.

[1] "Lucian Blaga" University of Sibiu, Victoriei Blvd, no. 10, Sibiu, 550024, Romania

E-mail: craciunean@sln.ro;

A Chomsky generative grammar (shortly grammar) (Salomaa, A.(1973)) is a quadruple $G=(V_N,V_T,S,P)$, where V_N and V_T are alphabets of nonterminal respectively terminal symbols; $S\in V_N$ is the starting symbol or axiom and P is a finite set of pairs of words from $(V_N\cup V_T)^*$, $P=\{(u_i,v_i)|1\le i\le m\}$, so that any word u_i contains at least one nonterminal symbol. The pairs (u_i,v_i) are called *derivation rules* or production rules or simple *productions* and will be denoted by $u_i\to v_i$.

The linguistic construction of the aggregation model starts from the basic actions that we call actions of aggregation and make up the atomic elements of the production process.

An action of aggregation (Crăciunean V.(2014)) *a* is a construct of the form

$$a = (I,O,M) \quad (1)$$

where

$I=\{(r_i,c_i,e_i) \mid i=1,2,...m$; r_i is a resource, c_i is the quantity of that resource, e_i is an event$\}$;

The resource r_i is called an input resource and represents the necessary resource in the c_i quantity for the action a, and e_i is called an input event.

$O=\{(r_o,c_o,e_o) \mid r_o$ is a resource or a final product, c_o is the quantity of the resource, e_o is an event$\}$;

If r_o is the resource then it's called the output resource and c_o is the quantity of that resource, else r_o is the final output product. The event e_o is the output event.

M is a set of data and associated metadata such as duration of action, the minimum allowed stock, costs, technical data about the action, software components, etc.

We consider the set A of actions of aggregation involved in a production process, then we define the precedence actions by two functions, namely: a bottom-up function of precedence $f:A\to\mathcal{P}(A)$, defined as follows:

$$f(a) = \{b\in A \mid O_a \in I_b\},\ (\forall)\ a\in A \quad (2)$$

and a top-down function of precedence $g:A\to\mathcal{P}(A)$, defined as follows:

$$g(b) = \{a\in A \mid O_a \in I_b\},\ (\forall)\ b\in A. \quad (3)$$

The symbols from the A set, represent real elementary actions and therefore the semantic load of the elements of the vocabulary is well defined.

An aggregation system of actions (Crăciunean V.(2014)) is a construct of the form

$$S = (A, f, g, \{L_X \mid X\in\mathcal{P}(A)\}) \quad (4)$$

where

A is a finite set of actions called action of aggregation;

f is a function $f: A \to \mathcal{P}(A)$ named bottom-up function of precedence;

g is a function $g: A \to \mathcal{P}(A)$ named top-down function of precedence;

L_X is a language over A, $(\forall)\ X\in\mathcal{P}(A)$. It is obvious that L_X can be an empty language for some $X\in\mathcal{P}(A)$.

Based on the function f we define the application of aggregation φ as follows:

An application of aggregation (Crăciunean V.(2014)) is an application $\varphi: \mathcal{P}(A) \to \mathcal{P}(A)$ defined as follows:

$$\varphi(X) = \cup_{x\in X} f(x) \quad (\forall)X\in\mathcal{P}(A). \quad (5)$$

In this conditions $\varphi^n(X)$ can be defined as:

$\varphi^0(X) = X \quad (\forall)\ X\in\mathcal{P}(A)$;

$\varphi^n(X) = \varphi(\varphi^{n-1}(X))$.

Similarly, we define the application of disaggregation ψ:

An application of disaggregation (Crăciunean V.(2014)) is an application $\psi: \mathcal{P}(A) \to \mathcal{P}(A)$ defined as follows:

$$\psi(X) = \cup_{x\in X} g(x) \quad (\forall)X\in\mathcal{P}(A). \quad (6)$$

Similarly, $\psi^n(X)$ can be defined as:

$\psi^0(X) = X \quad (\forall)\ X\in\mathcal{P}(A)$;

$$\psi^n(X) = \psi(\psi^{n-1}(X)). \quad (7)$$

3 GENERATIVE GRAMMARS ASSOCIATED WITH RESOURCES

When a context free grammar is used to model a manufacturing process, each nonterminal i will be naturally associated with an intermediate resource and each generating rule will be an action in the production process. As a result the derivation tree of this grammar will give us both a hierarchical structure of resources and actions of the production process.

At each hierarchical level we have to deal with a lot of resources, a lot of actions and a lot of rules for generating defined as above.

The grammar generating a context free language gives us an internal hierarchy of the language. For this we consider that the semantics of the language coincides with the production process obtained in the following way: each non-terminal symbol of the grammar is a resource, each terminal symbol of the grammar notes an action and each production is associated with an action, i.e. if

$A \rightarrow \alpha \in P$ and $\alpha = r_1 r_2 ... r_n a$, then the action a associated with this production operates on the resources $r_1, r_2, ..., r_n$ and produces the resource represented by the nonterminal symbol A.

We now consider an aggregation system of actions:

$$S = (A, f, g, \{L_X \mid X \in \mathcal{P}(A)\}) \quad (8)$$

defined as above and an action of aggregation a :

$$a = (I, O, M) \quad (9)$$

defined as above.

We denote with R the set of all intermediate resources involved in our production process. We will consider, to simplify the presentation, among resources also the final products resulting from the manufacturing process.

$$R = \bigcup_{a \in A} \{r \mid r \in O_a\}; \quad (10)$$

Obviously R is a finite set. Let $R=\{r_1, r_2, ... r_n\}$ be the enumeration of intermediate resources that are involved in our production process. We associate to each resource $r_i \in R$ a nonterminal R_i and we denote these nonterminal with $R_S=\{R_1, R_2, \ldots, R_n\}$.

We further consider the set A of actions associated with the aggregation system S. For every action $a \in A$ we will defined in the following one generation rule.

Let $r_i \in R$ be the output resource of the action and $\{r_1, r_2, \ldots, r_k\} = \bigcup_{x \in \psi(a)} \{r \mid r \in O_x\}$ the set of output resources of the actions from $\psi(a)$. Then the generation rule associated with the action $a \in A$ is: $R_i \rightarrow R_1 R_2 ... R_k a$. We denote by P_S the set of all these generation rules.

We note that for the initial actions for which $\psi(a) = \varnothing$ the production appropriate for the action a is $R_i \rightarrow a$. We also notice that we get a set of R_i-productions because there is, generally, a set of actions that have as output the same resource.

We will further associate to each intermediate resource $r \in R$ one generative grammar that will shape the hierarchical structure of the production subprocess witch determines this resource. Therefore we have to extract from the set of actions A only the actions involved in producing resource r, from the set of nonterminals R_S only those associated with the resources involved in producing resource r and from the set of generating rules P_S those involved in producing resource r.

We consider a context free grammar $G=(V_N, V_T, S, P)$ and the intermediate resource r represented by the nonterminal $R \in R_S$. If this grammar shapes the production subprocess of the intermediate resource r than $S=R$, $V_N \subset R_S$, $V_T \subset A$ and $P \subset P_S$

We first consider a hierarchy of nonterminal symbols constructed as:

i) $V_N^0 = \{S=R\}$;

$V_T^0 = \{a \in \mid O_a = r\}$ where a is the action with the resource r as output;

$P^0 = \{p \in P_S \mid p = X \rightarrow \alpha \text{ si } X = S = R\}$;

ii) $V_N^{i+1} = V_N^i \cup \{Y \in R_S \cap Symb(\alpha) \mid$

$X \rightarrow \alpha \in P_S \text{ si } X \in V_N^i\}$;

$V_T^{i+1} = V_T^i \cup \{x \in A \cap Syb(\alpha) \mid X \rightarrow \alpha \in P_S \text{ si } X \in V_N^i\}$;

$P^{i+1} = P^i \cup \{ p \in P \mid X \rightarrow \alpha \in P_S \text{ si } X \in V_N^i \}$.

Because R_S is a finite set $\Rightarrow \exists\ k \geq 0$ so that $V_N^k = V_N^{k+1}$, $V_T^k = V_T^{k+1}$ and $P^{i+1} = P^i$. Then $V_N = V_N^k$ because V_N^k contains all nonterminal symbols associated with the subresources involved in the production process of resource r. Likewise $V_T = V_T^k$, $P = P^i$ and the grammar G is completely determined.

If the context free grammar $G=(V_N, V_T, S, P)$ associated with the resource r is defined as above then the internal hierarchy of this grammar reflects the hierarchy of the production process of the resource r, i.e. the hierarchy of subresources and actions that contribute to the production of resource r. We further give the algorithms that help determine these hierarchies.

Given the context free grammar $G=(V_N, V_T, S, P)$ associated with the resource r as above, we present an algorithm that helps determine the hierarchy of resources involved in the production of resource r.

Input: A context free grammar $G=(V_N, V_T, S, P)$ associated with the resource r.

Output: Sets V_N^i of nonterminal symbols that represent the subresources involved in the production of resource r.

Method: is to build a string of sets $\{ V_N^i \mid i \in N\}$ with the property $V_N^0 \subset V_N^1 \subset \ldots \subset V_N^{k-1} = V_N^k = \ldots$. As we can see V_N^k will contain all nonterminal symbols witch by derivation will generate all words from V_T^*.

Obviously if $S \notin V_N^k$ then the language is empty.

Algorithm 3.1. 01

i ⟵ 0; 02

V_N^0 ={X∈ V_n | X→ α ∈P, α ∈ V_T^*} 03

DO UNTIL (V_N^i= V_N^{i-1}) 04

V_N^{i+1}= V_N^i ∪ { X∈ V_N | X→ $X_1X_2...X_n$a∈P and $X_1,X_2,...,X_n ∈ V_N^i$} 05

i ⟵ i+1; 06

ENDDO 07

STOP 08

Obviously, whatever the context free grammar G=(V_N,V_T,S,P) will be, there is a finite number of hierarchical levels k.

We now build an internal hierarchy of the generation rules. Given the context free grammar G=(V_N,V_T,S,P) associated with the resource r, we present an algorithm that helps determine the hierarchy of actions involved in the production of resource r.

Input: A context free grammar G=(V_N,V_T,S,P) associated with the resource r.

Output: The sets P^i of generation rules associated with actions involved in the production of resource r.

Method: is to build a string of sets { P^i |i∈N} with the property $P^1 \subset P^2 \subset \ldots \subset P^{k-1} = P^k = \ldots$. As we can see P^k will contain all useful productions of grammar G.

Algorithm 3.2. 01

i ⟵ 1; 02

P^1={p∈P|p= X→α ∈P, $\alpha \in V_T^*$ }; 03

DO UNTIL (P^i = P^{i-1}) 04

P^{i+1}=P^i ∪ { p∈P| p=X→ $X_1X_2...X_n$a∈P and $X_1,X_2,...,X_n ∈ V_N^i$ } 05

i ⟵ i+1; 06

ENDDO 07

STOP 08

Whatever the context free grammar G=(V_N,V_T,S,P) associated with a resource r will be, there is a finite number k of levels of generation rules and therefore there is a finite number of levels of actions involved in the production process of resource r.

4 GENERATIVE GRAMMARS ASSOCIATED WITH AGGREGATION BLOCKS OF ACTIONS

We now consider an aggregation system of actions:

$$S = (A, f, g, \{L_X \mid X \in \mathcal{P}(A)\}) \quad (11)$$

as defined above, together with this applications:

$$\varphi: \mathcal{P}(A) \to \mathcal{P}(A)$$

$$\psi: \mathcal{P}(A) \to \mathcal{P}(A)$$

defined as above.

An action of aggregation block of level k initiated by X is a construct of the form:

$$\mathcal{B}_k(X) = (I_k, O_k, L_k(X), B_k) \quad (12)$$

where

$$L_k(X)=L_{\varphi^k(X)}$$

$$I_k = \bigcup_{\alpha \in B_k(X)} I_a$$

$$O_k = \bigcup_{\alpha \in B_k(X)} O_a$$

B_k is an object, named block-board, that can store information of the form (resource, quantity, event). B_k is used by the actions of aggregation for communication.

So X∈$\mathcal{P}$(A) is a subset of actions defined as above and $L_k(X)=L_{\varphi^k(X)}$ is a language over the alphabet $\varphi^k(X)\subset$A. We consider that there is a context free generative grammar that generates the language $L_k(X)=L_{\varphi^k(X)}$ which we denote by:

$$G(X,k)=(V_N(X,k),V_T(X,k),S(X,k),P(X,k)). \quad (13)$$

Also, it follows that for every resource r_i there is a context free grammar

$$G(r_i)=(V_N(r_i),V_T(r_i),R_i,P(r_i)) \quad (14)$$

that generates the production process of the resource r_i .

Based on the grammar G(X,k) and the grammars {$G(r_i)$ | $r_i \in O_k$} we build the grammar

$$G^k(X)=(V_N^k(X), V_T^k(X),S(X,k),P^k(X)) \quad (15)$$

where :

V_N^k= $V_N(X,k) \cup (\cup_{r_i \in O_k} V_N(r_i))$;

V_T^k= $V_T(X,k) \cup (\cup_{r_i \in O_k} V_T(r_i))$;

$P^k=(\cup_{r_i \in O_k} P(r_i)) \cup P$

where P is obtained from P(X,k) by replacing the terminals in the productions with symbols associated with the output resources from the appropriate actions.

If $\alpha \in$ P(X,k) and a∈Symb(α) then R_j will replace a, if $r_j \in O_a$.

Therefore for each aggregated block of actions on the one hand we have a grammar $G^k(X)=(V_N^k, V_T^k, S^k, P^k)$ which gives us an overview of the production process at an aggregated block level and on the other hand we have a grammar $G(r_i)=(V_N(r_i),V_T(r_i),R_i,P(r_i))$ associated with each intermediate resources that gives us a picture on the aggregate production subprocess of this resource. From the way the grammar $G^k(X)$ is constructed, we notice that every word generated by it is a combination of words generated by the $G(r_i)$ grammars. Specifically grammar $G^k(X)$ is actually grammar G(X,k) in which the alphabet of terminals are replaced by the set of words generated by the $G(r_i)$ grammars. Therefore the grammars $G(r_i)$ and $G^k(X)$ can be substituted by one grammar $G_k(X)=(V_N^k(X), V_T^k(X), S^k(X), P^k(X))$ associated with the block in which we replace the start symbol with a set of start symbols

$S^k(X)=\{S(X,k)\}\cup\{R_i | r_i\ r_i \in O_k\}$.

Aggregation and disaggregation are useful techniques for reducing computational complexity in management and optimization issues of production processes. In general problems of planning and optimization of aggregated production process include at least three steps, namely:

i)The aggregation of the production process up to a convenient level, i.e., the replacement of variables and conditions of the process with aggregate variables and conditions.

ii)Data processing at the aggregated level.

iii)The results agreed at the aggregated level are projected to the detail level.

In our model the maximum level of aggregation is given by the aggregation block of actions $\mathcal{B}_k(X)=(I_k,O_k,L_k(X),B_k)$. The sublevels of aggregation are given by the internal hierarchy of the grammar $G_k(X)=(V_N^k(X), V_T^k(X), S^k(X), P^k(X))$, defined above. Most often the aggregation is done on value and time, and the disaggregation on quantities.

We now give a general algorithm for aggregation and disaggregation based on the $G_k(X)$ grammar associated with a block of actions $\mathcal{B}_k(X)$. For this we consider that every action a has a software component with the same name a as the action associated and admits as input an object μ in which the disaggregated data is and outputs another object ν which contains the aggregated data. Also, we assume that we have a software component ag that aggregates the data from multiple objects into one, and an another software component dg which disaggregates the data from one object into several objects.

Algorithm 4.1.

Given a context free grammar $G=(V_N,V_T,S,P)$ we present a general algorithm for aggregation and disaggregation of information at the start symbol level.

Input: a context free grammar $G=(V_N,V_T,S,P)$.

Output: is the data aggregated at home or broken symbol in the action.

Method: The algorithm is recursive and is based on the derivation model using the generation rules.

StartProcess(Y, μ)

Let $P(Y)=\{p_1,p_2,\ldots,p_l\}$ be the set of Y-productions from P.

DO WHILE($k\leq l$)

IF($cd(p_k)$)

$p \longleftarrow p_k$

$p : A \longrightarrow A_1A_2...A_n$;

$(\mu_1, \mu_2,... \mu_n)=dg(\mu)$;

$i \longleftarrow 1$;

DO WHILE($i\leq n$)

CASE (A_i) OF

CASE : $A_i \in V_N$

$Task(A_i) \longleftarrow StartProcess(A_i, \mu_i)$;

CASE : $A_i \in V_T$

$Task(A_i) \longleftarrow StartComponent(A_i,\mu_i)$;

ENDCASE

$Task(A_i).START$;

$i \longleftarrow i+1$;

ENDDO

$i \longleftarrow 1$;

DO WHILE($i\leq n$)

$\nu_i \longleftarrow Task(A_i).Results$;

$i \longleftarrow i+1$;

ENDDO

$\nu \longleftarrow ag(\nu_1,\nu_2,..., \nu_n)$;

ENDIF

$k \longleftarrow k+1$;

ENDDO

RETURN ν.

Starting up the process will be done by activating the StartProcess(S, μ) method where S is the start symbol of the grammar and μ contains the

aggregated data. The components associated to the actions have access to disaggregated data such as duration of actions, costs of actions etc.

In our case the grammar

$$G_k(X)=(V_N^k(X), V_T^k(X), S^k(X), P^k(X))$$

associated with an aggregated block of actions has a lot of start symbols, namely a symbol associated with the block and one symbol associated with each intermediate resource produced in the block. Therefore the algorithm will start with a convenient original symbol, depending on the objective.

Another important aspect of our model is related to the integration of software components that make junction with component-oriented programming, a highly topical software technology.

5 CONCLUSION

The model provides real support for the leadership, management and optimization of complex production processes.

The model provides real support for the leadership, management and optimization of complex production processes. The internal hierarchy of context free grammars provides a true model for hierarchical production processes.

Aggregation and distribution of actions in this model enables accurate calculation of costs, time and many other parameters that depend on the development of basic actions.

The model reflects an accurate picture of the complexity of production systems. We can define measures of production systems as functions proportionally dependent on the complexity of context free grammars shaping the production system. So we can compare the production process to each other.

The objectives of the current increase in standards of quality and efficiency in manufacturing processes can be achieved only through the best combination of inputs, independent of spatial distance between them. Electronic communication is to solve the distance problem between resources.

As a natural consequence achieving these objectives is solved by virtual production processes.

Because of the complexity of these production processes they cannot be mastered only with the help of accurate and complex models that reflect in real-time the evolution of aggregated and disaggregated physical and logical actions focused on achieving the objectives.

It is clear that software technologies are an essential condition for virtual process management but on the other hand the real need of managing virtual processes creates the impetus for the creation of appropriate technologies. Component-oriented programming is one of the major hopes for the development of software applications as flexible as virtual processes.

6 REFERENCES

►Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, (2001) *Compilers: Principles, Techniques, and Tools*, Addison Wesley.

►Calude Ch. S., Păun Gh.(2001), *Computing with Cells and Atoms an introduction to quantum, DNA and membrane computing*, Taylor & Francis.

►Crăciunean, V. (2014), *A Linguistic Model Of Aggregation In Virtual Manufacturing Processes* - Academic Journal Of Manufacturing Engineering, Vol. 12, Issue 3/2014 124

►Crăciunean, V. (2014), *Proiectarea Translatoarelor*, Sibiu, Editura Universității Lucian Blaga Sibiu, ISBN 978-606-12-0842-5

►Crăciunean, V.(2007), Fabian R. Popa E. M. ,*Total Fuzzy Grammar Driven Architecture for Intelligent Software Sytems*– WSEAS Transactions on Computers, Issue 2, Volume 6, pag. 248-253, ISSN 1109-2750.

►Dassow J., Păun Gh.(1999), *On the power of membrane computing*, J. Universal Computer Sci., 5, 2, 33-49.

►Matthias Dehmer,(2013) Abbe Mowshowitz, and Frank Emmert-Streib - *Advances in Network Complexity* - Wiley-VCH Verlag GmbH & Co. KGaA,Boschstr. 12, 69469 Weinheim, Germany.

►Marvin Scheaffer,(1973) *A Mathematical Theory of Global Program Optimization*, Prentice-Hall.

►Mowshowitz A.(1994), *Virtual Organization: A vision of Management in the Information Age*, The Information.

►Păun Gh.(1988), *Mecanisme generative ale proceselor economice*, Ed Tehnică, Bucureşti.

►Salomaa, A.(1973) , *Formal Languages*, New York, Academic Press.