33 A FORMAL THEORY OF BM VIRTUAL ENTERPRISES STRUCTURES

Rui Sousa; Goran Putnik
Production and Systems Engineering Department
University of Minho, School of Engineering
Guimarães, PORTUGAL
Email: rms@dps.uminho.pt
Email: putnikgd@dps.uminho.pt

Formalisms are a tool commonly used in many engineering areas and, as expected, are also being used on virtual enterprises research. However the use of formalisms is not enough to ensure correctness and ambiguities absence on the developed projects. Only with a background formal theory is possible to achieve that goal. This paper presents a formal theory of the structural aspect of virtual enterprises according to the BM_Virtual Enterprise Architecture Reference Model (BM_VEARM) developed at University of Minho – Portugal. The theory is generated and represented by an attributed context-free formal grammar accepting some pre-requisites as input and producing as output canonical structures of virtual enterprises compliant to those pre-requisites. The formal theory of BM_VEARM virtual enterprises structures is in fact the formal language generated by the defined grammar.

1. INTRODUCTION

Contrarily to some speculations the use of formalisms doesn't mean that a formal theory is behind. For the case of first-order theories this claim is proved in (Sousa, 2003), using mathematical first-order logic concepts, and implies that formalisms by their own are not enough to ensure correctness and to avoid ambiguities.

It is commonly accepted that only with solid theories it is possible to achieve the desired rigour on developing projects. Research on virtual enterprises (VEs) is an area of investigation whose importance is rapidly increasing as VEs are seen, especially by the scientific community, as the new paradigm for the factories/enterprises of the future. It is obvious that a formal theory of VEs would be of extreme importance for the investigation on this area.

This paper introduces a formal theory, generated by an attributed context-free formal grammar, of the structural aspects of VEs according to BM_Virtual Enterprise Architecture Reference Model (BM_VEARM).

The concept of theory is rigorously defined by mathematical logic as a set of some formulas with some special characteristics (Mendelson, 1987; Ebbinghaus *et al.*, 1996; Keisler, 1996). Those formulas are obtained from a given alphabet of symbols, using some derivation rules (calculus of formulas) and they constitute a language. Thus a theory is a language but, obviously, a special language. The formal

grammar presented in this paper was specially developed to synthesize strings of symbols which are formulas compliant to the theory definition coming from mathematical logic. Hence, this grammar generates a language which is in fact a theory. The theory concept has as background other formal concepts from mathematical logic involving not only the syntactical viewpoint, but also the semantic perspective (e.g. structure, interpretation and model). With the developed grammar, and given some pre-requisites, it is possible to generate canonical structures of VEs compliant to BM_VEARM reference architecture.

The paper is intended to be introductory and self-contained regarding the grammatical principles involved, and its structure is as follows. Section 2 provides the basics of formal grammars, arising from theory of languages. A generic definition of formal grammar and the Chomsky's classification for formal grammars are presented. Attributed grammars are also referred as they are the truly powerful grammars. A simple example, already interpretable in the manufacturing systems structural aspects area, is provided. The fundamentals of BM_VEARM developed at the Production and System Engineering Department, University of Minho, Portugal (Putnik, 2000), are provided on section 3. Comprehensively more emphasis is dedicated on the structural aspects of VEs. On section 4 it is introduced the attributed context-free formal grammar G_{BM} responsible for the generation of the formal language L_{BM} which is a formal theory of BM_Virtual Enterprises structures. On section 5 some conclusions are outlined along with some perspectives of future work.

2. FORMAL GRAMMARS

A grammar is usually known as a set of rules allowing the creation of words and sentences over a given alphabet. The formal grammar concept goes a bit further by including the alphabet itself on the definition. Many similar definitions can be found in literature (Salomaa, 1973; Denning *et al.*, 1978; Hopcroft and Ullman, 1979; Lewis and Papadimitriou, 1981; Mikolajczak, 1991; Révész, 1991; Pittman and Peters, 1992), all based on Chomsky's definition (Chomsky, 1959). Adapted to the notation used in this paper we have:

Definition 1: A formal grammar G is a four-tuple $G=(V_T,V_N,S,R)$ where V_T is a finite set of terminal symbols, V_N a finite set of non-terminal symbols ($V_T \cap V_N=\varnothing$), S is the initial symbol ($S \subseteq V_N$) and R is a finite set of rewriting rules.

Each rewriting rule, or production, is an ordered pair (α,β) usually denoted as $\alpha \rightarrow \beta$ showing how the word $\alpha \in (V_T \cup V_N)^+$ can be rewrite as $\beta \in (V_T \cup V_N)^*$. The word α must contain at least one non-terminal symbol. Recall that if V is an alphabet then V^* represents the set of all the words, including the empty word λ, that can be constructed with the symbols of V and $V^+=V^* \setminus \{\lambda\}$.

Example 1: Consider a grammar $G=(V_T,V_N,S,R)$ where $V_T=\{m, \mapsto, //,), (\}$, $V_N=\{S\}$ and $R=\{S \rightarrow m,\ S \rightarrow S \mapsto S,\ S \rightarrow S//S,\ S \rightarrow (S)\}$.

Two possible words of terminal symbols generated by this grammar are:

$$S \Rightarrow S \mapsto S \Rightarrow m \mapsto S \Rightarrow m \mapsto (S) \Rightarrow m \mapsto (S//S) \Rightarrow m \mapsto (m//S) \Rightarrow m \mapsto (m//m) \quad (1)$$

$$S \Rightarrow S \mapsto S \Rightarrow (S) \mapsto S \Rightarrow (S//S) \mapsto S \Rightarrow ((S)//S) \mapsto S \Rightarrow ((S \mapsto S)//S) \mapsto S \Rightarrow$$
$$\Rightarrow ((m \mapsto S)//S) \mapsto S \Rightarrow ((m \mapsto m)//S) \mapsto S \Rightarrow ((m \mapsto m)//m) \mapsto S \Rightarrow$$
$$\Rightarrow ((m \mapsto m)//m) \mapsto m \quad (2)$$

Each symbol $\Rightarrow$ represents a derivation step and corresponds to the application of one of the available productions. A derivation process ends when all the symbols of the word are terminal symbols. From the manufacturing systems structures perspective, words obtained by derivations (1) and (2) can de interpreted as different machine compositions (see Figure 1).

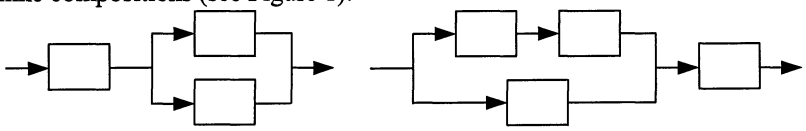

Figure 1 - Machine compositions generated by G grammar

Based on their productions type, formal grammars are classified in four classes: unrestricted (type 0), context-sensitive (type 1), context-free (type 2) and regular (type 3). This classification is known as Chomsky's hierarchy. To overcome some limitations of formal grammars the concept of attributed grammar was introduced by (Knuth, 1968). In an attributed grammar each symbol may have none, one or more attributes, addressing thus, besides syntactical aspects, the semantic viewpoint. Consequently the definition of each production must be extended with assertions or predicates about the involved attributes. To illustrate this concept the grammar G from example 1 will be transformed into an attributed grammar G'. The distinction between different machines can be accomplished using a finite number i of m_i symbols, instead of a single symbol m. Each machine can be characterized by its production rate p_r in parts/h. Thus p_r will be an attribute of each m_i symbol and also of symbol S which represents the entire system (see Table 1). This attribute is not applicable to the remaining alphabet symbols.

Table 1 - Symbols with attributes for grammar G'

Symbol	*Description*	*Attribute*	*Description*
m_i	machine i	p_r	machine production rate
S	system	p_r	system production rate

Now the productions of the new grammar G' are completed with assertions about the p_r attribute (see Table 2). Note that superscript identifiers are introduced whenever symbol instance distinction is necessary.

Table 2 - Productions and assertions for grammar G'

Production	*Assertion*
$S^{(1)} \rightarrow m_i$	$p_r(S^{(1)})=p_r(m_i)$
$S^{(1)} \rightarrow S^{(2)} \mapsto S^{(3)}$	$p_r(S^{(1)})=\min(p_r(S^{(2)}), p_r(S^{(3)}))$
$S^{(1)} \rightarrow S^{(2)} // S^{(3)}$	$p_r(S^{(1)})=p_r(S^{(2)})+ p_r(S^{(3)})$
$S^{(1)} \rightarrow (S^{(2)})$	$p_r(S^{(1)})=p_r(S^{(2)})$

Example 2: Consider the attributed grammar $G'=(V_T, V_N, S, R)$ where $V_T=\{m_1, \ldots, m_i, \mapsto, //,), (\}$, $V_N=\{S\}$ and R contains the productions of Table 2.

Recalling the derivation (1) of example 1, but including now instance identifiers and distinct m_i symbols we may have:

$$S^{(1)} \Rightarrow S^{(2)} \mapsto S^{(3)} \Rightarrow m_1 \mapsto S^{(3)} \Rightarrow m_1 \mapsto (S^{(4)}) \Rightarrow m_1 \mapsto (S^{(5)} // S^{(6)}) \Rightarrow m_1 \mapsto (m_2 // S^{(6)}) \Rightarrow m_1 \mapsto (m_2 // m_3) \quad (3)$$

Besides the showed generation of machine compositions, G' can also determine the production rate p_r of the generated system, based obviously on the individual machines production rates which in this case are set, for instance, to 20, 18 and 16 parts/h for m_1, m_2 and m_3, respectively. Formally system p_r calculation is done using the assertions associated to the applied productions, starting from the last derivation step because system p_r is an synthesized attribute (Pittman and Peters, 1992; Sousa, 2003). Thus in the last derivation step it is used the production $S^{(6)} \rightarrow m_3$ implying that $p_r(S^{(6)})=p_r(m_3)=16$ parts/h. The previous derivation step applies production $S^{(5)} \rightarrow m_2$ and thus $p_r(S^{(5)})=p_r(m_2)=18$ parts/h. The fourth derivation step uses production $S^{(4)} \rightarrow S^{(5)} // S^{(6)}$ leading to $p_r(S^{(4)})=p_r(S^{(5)})+ p_r(S^{(6)})=18+16=34$ parts/h. The third derivation step applies $S^{(3)} \rightarrow (S^{(4)})$ and thus $p_r(S^{(3)})=p_r(S^{(4)})=34$ parts/h. The second derivation step uses the production $S^{(2)} \rightarrow m_1$ implying that $p_r(S^{(2)})=p_r(m_1)=20$ parts/h. Finally the first derivation step applies $S^{(1)} \rightarrow S^{(2)} \mapsto S^{(3)}$ and consequently the production rate of the generated system is $p_r(S^{(1)})=\min(p_r(S^{(2)}), p_r(S^{(3)}))=\min(20,34)=20$ parts/h. Although simple this example illustrates the high potential of attributed grammars when compared with traditional grammars.

The language generated by a grammar is the set of all the words of terminal symbols generated by that grammar.

Definition 2: The language generated by a formal grammar $G=(V_T, V_N, S, R)$ is $L(G) = \left\{ p \in V_T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} p \right\}$.

Symbol $\overset{*}{\underset{G}{\Rightarrow}}$ denotes derivation in many steps according to the productions of G.

Mathematical logic defines language as the set of all the formulas obtained from a given alphabet according to a set of rules (calculus of formulas). From all those formulas some, under certain circumstances, may constitute a theory (Mendelson,

1987; Ebbinghaus *et al.*, 1996; Keisler, 1996). Thus, and without further justification, we can say that a mathematical logic language may potentially include one or more theories. Hence if a formal grammar generates words that can be considered as formulas, then that grammar is a potential theory generator. This subject is deeply investigated in (Sousa, 2003).

3. FUNDAMENTALS OF BM_VEARM ARCHITECTURE

The BM_Virtual Enterprise Architecture Reference Model (Putnik, 2000) is based on a multilevel hierarchical model (Mesarovic *et al.*, 1970) and supports four crucial characteristics for VEs: integrability, distributivity, agility and virtuality. To achieve the first characteristic BM_VEARM includes an integration mechanism concept. The use of wide area networks supports the distribution of the VE resources. Agility and virtuality are provided in BM_VEARM through the broker concept. Figure 2(a) represents the elementary hierarchical BM_VEARM structure which works as a building unit in the synthesis process of VEs. Figure 2(b) shows an example of a VE structure synthesized according to BM_VEARM. Both diagrams on Figure 2 are logical representations with a high abstraction level. From the implementation viewpoint, integration mechanisms are usually embedded in the adjacent blocks (i.e. control level and resources management).

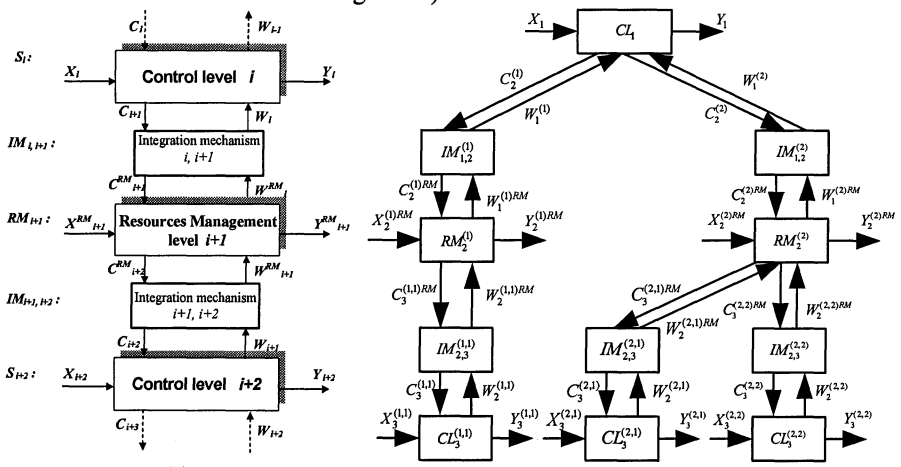

Figure 2 - BM_VEARM (a) elementary structure (Putnik, 2000) (b) VE instance

Based on this perception the incoming grammar for VE synthesis may include only two types of basic blocks: c_i - control level and r_j resources management. The complete description of BM_VEARM can be found in (Putnik, 2000).

4. A FORMAL THEORY OF BM_VEARM VIRTUAL ENTERPRISES

This section presents a context-free attributed grammar, denoted as G_{BM}, able to

generate VEs structures according to BM_VEARM. As seen before two fundamental terminal symbols are necessary: c_i – to represent control level blocks and r_j – for resources management blocks (see Figure 3). Due to space limitations is not possible to include here all the symbols, attributes and assertions of G_{BM}. This is the reason why definition 3 and derivation 4 only refer to the syntactical aspects of G_{BM}. However the entire development process can be found in (Sousa, 2003).

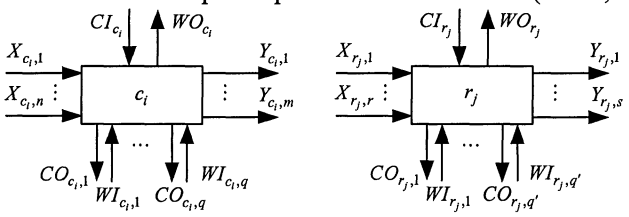

Figure 3 - Basic blocks for G_{BM} (Sousa, 2003)

Definition 3: $G_{BM}=(V_T,V_N,S,R)$ is an attributed context-free grammar where $V_T=\{c_1,\ldots,c_{n_c},r_1,\ldots,r_{n_r},S_{eq},\equiv,\ \downarrow\uparrow,),(\}$, $V_N=\{S,A,B\}$ and $R=\{S\rightarrow c_i(\downarrow\uparrow A)\equiv S_{eq}$, $A\rightarrow r_i(\downarrow\uparrow B)$, $A\rightarrow AA$, $B\rightarrow c_i(\downarrow\uparrow A)$, $B\rightarrow BB$, $B\rightarrow c_i\}$ dedicated to the synthesis of virtual enterprises according to BM_VEARM.

Figure 4(a) represents the so-called "BM_VEARM minimal system". The VE instance of Figure 2 (b), now with embedded integration mechanisms, is shown in Figure 4(b) and can be synthesized from the following G_{BM} derivation:

$$\begin{aligned} S &\Rightarrow c_1(\downarrow\uparrow A)\equiv s_{eq} \Rightarrow c_1(\downarrow\uparrow AA)\equiv s_{eq} \Rightarrow c_1(\downarrow\uparrow r_1(\downarrow\uparrow B)A)\equiv s_{eq} \Rightarrow c_1(\downarrow\uparrow r_1(\downarrow\uparrow c_2)A)\equiv s_{eq} \\ &\Rightarrow c_1(\downarrow\uparrow r_1(\downarrow\uparrow c_2)r_2(\downarrow\uparrow B))\equiv s_{eq} \Rightarrow c_1(\downarrow\uparrow r_1(\downarrow\uparrow c_2)r_2(\downarrow\uparrow BB))\equiv s_{eq} \Rightarrow \\ &\Rightarrow c_1(\downarrow\uparrow r_1(\downarrow\uparrow c_2)r_2(\downarrow\uparrow c_3c_4))\equiv s_{eq} \end{aligned} \quad (4)$$

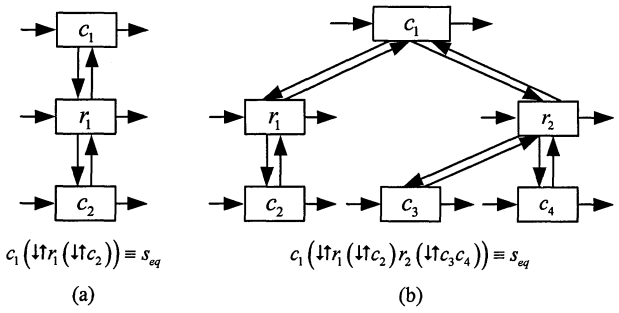

Figure 4 - BM_VEARM (a) minimal system (b) VE instance

As seen before the set of all the words generated by a grammar is a language.
Definition 4: $L_{BM}=L(G_{BM})$ is a formal language for VEs structures compliant to BM_VEARM.

Every word generated by G_{BM} ends with '$\equiv S_{eq}$' being thus a formula. Therefore the language L_{BM} is a set of formulas. According to mathematical logic if those formulas are satisfiable by a given interpretation (and closed under consequence) then they will constitute a theory (Ebbinghaus *et al.*, 1996). The structural interpretation of G_{BM} terminal symbols as control level blocks, resources management blocks, hierarchical connection, etc., satisfies all the formulas of L_{BM}. Thus we can claim that L_{BM} is a formal theory of VEs structures compliant to BM_VEARM.

5. CONCLUSIONS

The importance of the virtual enterprise (VE) paradigm at present and near future seems to be obvious. It seems also consensual that investigation on this area must have a solid theoretical background otherwise sustainable research won't be possible. Following this line of thought this paper presents an important contribution to the establishment of the referred theoretical base.

It is shown how formal grammars, and specially attributed grammars, can be used to deal with some aspects of VEs – structural aspects in this case - in a completely rigorous manner.

It is presented the attributed context-free formal grammar G_{BM} responsible for the generation of the formal language L_{BM}. L_{BM} is not just another formal representation language used, in this case, in the VEs area. Due to the development process of G_{BM}, the language L_{BM} can be used to represent VEs structures but it is also a theory of VEs structures compliant to the BM_Virtual Enterprise Architecture Reference Model (BM_VEARM), providing other potentialities. Although not detailed here, due to space limitations, the inclusion of attributes associated to the symbols of G_{BM} grammar constitutes the true power of this approach. For example we can define how many blocks (control level and resources management) are available and how many inputs and outputs each one of them has, and let G_{BM} synthesize VEs instances compliant to those predefined requisites. Furthermore with simple modifications G_{BM} can be used not only to synthesize VEs structures but also to recognize that kind of structures.

The exploitation of the equivalence grammars-automata will lead to the specification of a pushdown automaton equivalent to the context-free attributed grammar G_{BM}, allowing thus the development of application tools. This work is already running and a very simple prototype tool (not yet based on G_{BM}) was already developed by two computer science students.

The Formal Theory (FT) presented in this paper is not a general FT of VEs, but only the FT of a specific aspect of VEs - the structural aspect - compliant to BM_VEARM. BM_VEARM is a reference model and others may exist. The grammatical approach proposed could be applied to other reference models, implying that a specific grammar should be constructed for each model. What to do with these FT of particular VE models and aspects? Unify them in a more general FT or leave them as they are resolving only specific problems? These, and other related issues, are open questions. This paper is also a contribution to these questions.

6. REFERENCES

1. Chomsky, N. (1959). "On Certain Properties of Grammars." *Information and control* **2**: 137-167.
2. Denning, P. J., Dennis, J. B. and Qualitz, J. E. (1978). *Machines, Languages and Computation*, Prentice-Hall, Inc.
3. Ebbinghaus, H. D., Flum, J. and Thomas, W. (1996). *Mathematical Logic*, Springer.
4. Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company.
5. Keisler, H. J. (1996). *Mathematical Logic and Computability*, McGraw-Hill International Editions.
6. Knuth, D. E. (1968). "Semantics of Context-free Languages." *Mathematical Systems Theory* **2**: 127-145.
7. Lewis, H. R. and Papadimitriou, C. H. (1981). *Elements of the Theory of Computation*, Prentice-Hall International Editions.
8. Mendelson, E. (1987). *Introduction to Mathematical Logic*, Chapman & Hall.
9. Mesarovic, M. D., Macko, D. and Takahara, Y. (1970). *Theory of Hierarchical, Multilevel, Systems*, Academic.
10. Mikolajczak, B., Ed. (1991). *Algebraic and Structural Automata Theory*, North-Holland.
11. Pittman, T. and Peters, J. (1992). *The Art of Compiler Design - Theory and Practice*, Prentice-Hall International, Inc.
12. Putnik, G. (2000). BM_Virtual Enterprise Architecture Reference Model, *in Agile Manufacturing: 21st Century Manufacturing Strategy (A. Gunasekaran)*, Elsevier science Publ: 73-93.
13. Révész, G. E. (1991). *Introduction to Formal Languages*, Dover Publications, Inc.
14. Salomaa, A. (1973). *Formal Languages*, Academic Press, Inc.
15. Sousa, R. (2003). Contribuição para uma Teoria Formal de Sistemas de Produção. Tese PhD. Departamento de Produção e Sistemas, Universidade do Minho.