Modelling the Formation of Virtual Buying Cooperatives with Grammars of Regulated Rewriting

Suna Bensch[1], Sigrid Ewert[2] and Mpho Raborife[2]
[1]*Department of Computing Science, Umeå University, Umeå, Sweden*
[2]*School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa*

Keywords: Coalition Formation, Virtual Buying Cooperative, Regulated Rewriting, Random Permitting Context Grammar, Random Context Grammar.

Abstract: In this paper we model virtual buying cooperatives (VBC) with grammars of regulated rewriting and show that, if VBC relevant information is distributed over several successive VBC processes and must, in a later stage, be synchronised and co-ordinated, the formal grammar needs to be very powerful with respect to mode of derivation and thus generative capacity. In particular, we show how to model the supplier phase, invitation phase, and declaration phase of a VBC with random permitting context grammars and the VBC reservation phase with random context grammars under a special kind of leftmost derivation. If we use random permitting context grammars for all processes, we can only model a VBC formation during which information is introduced and processed locally and successively rather than being spread over different VBC processes.

1 INTRODUCTION

Cooperatives are generally goal-directed and have a short lifespan; they are formed with a purpose and dissolve when that purpose no longer exists and the agents leave the cooperative (Horling and Lesser, 2005). They are most useful in situations where a single organisation or business cannot perform a particular task or the efficiency of the task is increased if more than one organisation or business performs it. We consider cooperatives formed by physically distributed enterprises with the purpose of purchasing items from a supplier as a single entity in a virtual marketplace. The authors in (Ngassam and Raborife, 2013) term such a cooperative a *virtual buying cooperative (VBC)*. Since a VBC acts as a single entity and can buy a larger amount than an individual enterprise, the VBC can negotiate favourable pricing.

In this paper we divide the VBC formation process into four phases, *supplier phase*, *invitation phase*, *declaration phase* and *reservation phase* and propose that the reservation phase is guided by a purchasing strategy that either promotes close business associations or a wide association distribution. The VBC formation is basically the process during which enterprises invite their associates to participate in a VBC. The VBC itself is the finally formed cooperative, that is, the enterprises that purchase items from a supplier.

We model the formation of a VBC and the VBC itself with grammars and strings that these grammars generate. Grammars or rewriting systems are generative devices that generate infinitely many strings in a so-called language but are themselves finite representations. Grammars are a suitable tool for modelling VBCs because they

- describe the formation of a VBC as a generative process reflecting the joint effort of all involved enterprises,
- generate infinite possibilities of VBC structures (i.e. VBCs),
- can easily be implemented, and
- shed light on the structural conditions of a VBC and thus can guide the implementation and reduce the costs for the actual technological development.

An implementation of our grammar model should enable the enterprises participating in a VBC, to

- form the VBC cooperatively and according to their needs and networks,
- decide on a purchasing strategy for the VBC,
- be able to see the current state of the VBC formation (e.g. the participating enterprises, the quantity provided by the supplier), and

Bensch, S., Ewert, S. and Raborife, M.
Modelling the Formation of Virtual Buying Cooperatives with Grammars of Regulated Rewriting.
DOI: 10.5220/0006403700450055
In *Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2017)*, pages 45-55
ISBN: 978-989-758-265-3

- have proof of how many items an enterprise will obtain once the overall quantity is purchased from the supplier.

A technological realisation should take into account that in rural areas Internet is not always accessible, but a text message based implementation would let the isolated small enterprises communicate and form the VBC via text messages. The content of such text messages should contain information about the current state of the VBC formation which is reflected in our model as sentential forms during a derivation process (explained in more detail later). We believe that an implementation of our VBC model would enable such small businesses that operate in isolation to access markets, to exploit current market opportunities in a virtual marketplace whilst maintaining their autonomous business operations in the real world.

We model VBCs with grammars of regulated rewriting and show that, if VBC relevant information is introduced in different VBC formation phases and must during the reservation phase be co-ordinated, the grammars need to be very powerful with respect to derivation mode and thus generative capacity. In particular, we show that the reservation phase requires random context grammars under a special kind of leftmost derivation, whereas supplier phase, invitation phase, and declaration phase of a VBC can be modelled with random permitting context grammars (without leftmost restriction). Finally we show that, if we only use random permitting context grammars to model all four VBC phases, the VBC model will be restricted, since information has to be processed "locally" and successively and cannot be spread across the four phases.

2 RELATED WORK

The authors in (Ngassam and Raborife, 2013) consider VBCs, that is, cooperatives formed by geographically distributed enterprises in a virtual marketplace. These enterprises meet at a virtual marketplace and form cooperatives as and when needed based on the items they are interested in. The enterprises pool their buying power and negotiate a favourable pricing based on the number of items that they will purchase. Once they have made the purchase, the cooperative is disbanded and another cooperative can be formed.

A VBC is especially beneficial to very small enterprises (VSEs) where the owners usually work in isolation and are not connected to economically strong regions and markets. Due to their small sizes, VSEs do not buy significantly large amounts of goods and this hinders their ability to fully aggregate demand and negotiate discounted prices from their suppliers. In emerging economies such as the Republic of South Africa, VSEs are essential in driving economic growth and creating employment (Ngassam and Raborife, 2013). These businesses are crucial to emerging economies but are usually operated in informal environments which are typically characterised by poor infrastructure, poor inventory management, bad working habits, and lack of direct access to markets. This leads to the exploitation of such enterprises by their suppliers (Merz et al., 2007) (Merz, 2010).

The authors in (Mashkov et al., 2015) model coalition formation as unselfish agents using Petri Nets. We model the formation of a VBC and the VBC itself with random context grammars and one of their variants. Various regulated rewriting grammars and their languages and the motivations thereof are extensively described in (Dassow and Păun, 1989). We choose random context grammars to model the VBC formation since many actions during the formation are context-dependent and the grammar formalism is a relatively simple extension of the easy-to-handle context-free grammars. Random context grammars (rcgs) (van der Walt, 1972) belong to the class of context-free grammars with regulated rewriting (Dassow and Păun, 1989), i.e., the productions of a grammar are context-free, but are applied in a non-context-free manner.

In rcgs, the application of a production at any step in a derivation depends on the set of symbols that appear in the sentential form of the derivation at that step. As opposed to context-sensitive grammars, the context may be distributed in a random manner in the sentential form. Context is classified as either permitting or forbidding: permitting context enables the application of a production, while forbidding context inhibits it. When a grammar uses permitting context only or forbidding context only, it is called a random permitting context grammar (rPcg) or random forbidding context grammar (rFcg), respectively. The corresponding generated languages are called random permitting context languages (rPcls) and random forbidding context languages (rFcls).

The authors in (Dassow and Păun, 1989) showed that rcgs without erasing productions lie strictly between the context-free and context-sensitive grammars. When erasing productions are allowed, rcgs are as powerful as the recursively-enumerable grammars. It is not known whether rFcgs without erasing production rules have an erasing equivalent. However, every rPcg with erasing production rules has a non-erasing equivalent (Zetzsche, 2010).

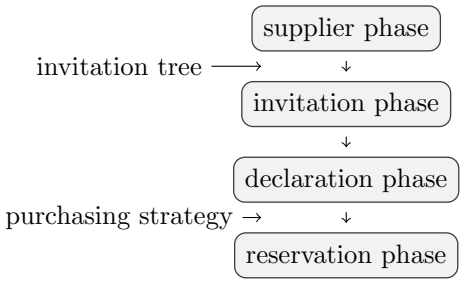

Figure 1: An illustration of the VBC formation process divided into supplier phase, invitation phase, declaration phase and reservation phase. The invitation phase is regulated by an invitation tree and the reservation phase is regulated by a chosen purchasing strategy.

3 VBC FORMATION

The formation of a VBC involves an enterprise (termed the *initiator enterprise*) approaching a supplier with the intent to purchase items (Ngassam and Raborife, 2013). The supplier in turn replies with the overall available quantity of the requested items. The initiator enterprise then purchases items, and invites selected associates, who in turn invite their associates, etc., to join the VBC in order to purchase the items from the supplier. In our model of a VBC formation we deviate from (Ngassam and Raborife, 2013) as follows. The initiator enterprise approaches a supplier that replies with the overall quantity of purchasable items as in (Ngassam and Raborife, 2013) but the initiator enterprise does not necessarily get to purchase before all other enterprises. Our model of VBC formation, which is illustrated in Figure 1 consists of the four following successive phases.

The first phase is the *supplier phase* in which the supplier provides the quantity of goods that can be purchased to a favourable pricing. The supplier phase is followed by the *invitation phase* in which all enterprises that belong to a network of associates and want to participate in the VBC are invited. In the subsequent *declaration phase* all enterprises declare how many items they would want to purchase. The last phase is the *reservation phase* where enterprises reserve the quantity of goods they will obtain after the overall quantity is purchased from the supplier. We suggest that the reservation phase (i.e. the actual purchasing) in the VBC is regulated by a *purchasing strategy* that is chosen after the declaration phase. We propose two different strategies:

1. (Early strategy or E-strategy) Enterprises that were invited at an earlier time step can purchase items before enterprises that were invited at a later time step. This reflects a purchasing strategy where close associations or first-degree associations are promoted.
2. (Maximal strategy or M-strategy) Maximize the number of enterprises that purchase. This strategy reflects a network-wide distribution of purchasing opportunities to as many enterprises as possible.

One can think of other possible purchasing strategies, including a mixture between E- and M-strategy. In this paper we assume that a purchasing strategy is chosen outside our model. Regardless of the purchasing strategy, the difference between the number of items provided by the supplier and the number of items purchased by the VBC should be as small as possible. This is in order to guarantee that the VBC can negotiate a favourable pricing. Moreover, the total number of items purchased by members of the VBC cannot be more than the quantity made available to them by the supplier. In our model the four phases of a VBC and the results thereof are explicitly modelled as generative processes. We will construct grammar fragments for all four phases. The supplier phase is part of our model and represents the supplier providing the available overall quantity of goods. For the subsequent invitation phase we assume the following:

- The VBC formation starts with a given network graph. A network graph represents a network of associates and is represented as a graph. A network graph is finite and predetermined at the start of a VBC formation. Network graphs can be extended by adding new enterprises to the network (but not during a VBC formation).
- Enterprises can invite as many associates from their network as desired, regardless of whether the invited associates are interested in purchasing or not, or whether they invite other associates or not.
- An enterprise may only be invited once.

During the invitation phase an invited enterprise can either invite other enterprises from its network or not invite any further enterprises. During the declaration phase an invited enterprise can either declare the amount it wants to purchase or opt out of the VBC by not declaring items. During the reservation phase and after a purchasing strategy has been chosen, we model the binding allocation of goods to the enterprises that have chosen to purchase. The reservation phase is the part of the coalition formation that requires grammars with high generative capacity to model, since the coordination of information must be modelled with devices that have the capacity to keep track of information distributed over several phases.

4 DEFINITIONS AND PRELIMINARIES

Let $\mathbb{N}$ denote the integers, and $\mathbb{N}_+ = \{1,2,\ldots\}$. An alphabet is a finite set of symbols. A word over an alphabet Σ is a finite ordered list of symbols chosen from the set Σ. A language L is a set of words over some alphabet (Xavier, 2005).

Definition 1. *(Ewert and Van der Walt, 2002) A random context grammar (rcg) is a quadruple $G = (V_N, V_T, P, S)$, where*

1. *V_N is a finite set of non-terminals,*
2. *V_T is a finite set of terminals,*
3. *P is a finite set of productions of the form*

$$A \to x\ (\mathcal{P}; \mathcal{F}),$$

where $A \in V_N$, $x \in (V_N \cup V_T)^*$ and $\mathcal{P}, \mathcal{F} \subseteq V_N$, and

4. *$S \in V_N$ is the start symbol.*

Let V denote $V_N \cup V_T$. For two strings $y_1, y_2 \in V^*$ and a production $A \to x\ (\mathcal{P}; \mathcal{F})$ in P, we may write $y_1 A y_2 \Longrightarrow y_1 x y_2$ if every $B \in \mathcal{P}$ is in the string $y_1 y_2$ and no $B \in \mathcal{F}$ is in the string $y_1 y_2$. We refer to $y_1 A y_2 \Longrightarrow y_1 x y_2$ as a derivation step and to the strings $y_1 A y_2$, $y_1 x y_2$ as sentential forms. The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\overset{*}{\Longrightarrow}$. The language generated by a grammar G is defined as

$$L(G) = \{w \mid w \in V_T^* \text{ and } S \overset{*}{\Longrightarrow} w\}.$$

A random permitting context grammar (rPcg) is a random context grammar $G = (V_N, V_T, P, S)$, where for each production $A \to x\ (\mathcal{P}; \mathcal{F}) \in P$, $\mathcal{F} = \emptyset$. A random forbidding context grammar (rFcg) is a random context grammar $G = (V_N, V_T, P, S)$, where for each production $A \to x\ (\mathcal{P}; \mathcal{F}) \in P$, $\mathcal{P} = \emptyset$. A context-free grammar (cfg) is a random context grammar $G = (V_N, V_T, P, S)$, where $\mathcal{P} = \mathcal{F} = \emptyset$ for each production $A \to x\ (\mathcal{P}; \mathcal{F}) \in P$. A language is context-free, random permitting context, random forbidding context, or random context if it is generated by a context-free, random permitting context, random forbidding context, or random context grammar respectively. In the remainder of this paper, we write $A \to x$ instead of $A \to x(\emptyset; \emptyset)$, if the permitting and forbidding context are empty sets. If a context consists only of one element, we write A instead of $\{A\}$.

Example 1. *Let $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, where P is given by*

$$\{S \to ABC,$$
$$A \to aA'(B; \emptyset),$$
$$B \to bB'(C; \emptyset),$$
$$C \to cC'(A'; \emptyset),$$
$$A' \to A(B'; \emptyset),$$
$$B' \to B(C'; \emptyset),$$
$$C' \to C(A; \emptyset),$$
$$A \to a(B; \emptyset),$$
$$B \to b(C; \emptyset),$$
$$C \to c\}.$$

The generated language is $L(G) = \{a^n b^n c^n \mid n \geq 1\}$. The derivation starts with rewriting S with the first rule and we obtain the sentential form ABC. Now the second and third are applicable, each of which introduces an A' or B' into the sentential form. After, for example, applying the second and third rule we obtain the sentential form $aA'bB'C$. Then the fourth rule is applied (since there is an A' in the sentential form) and we obtain the sentential form $aA'bB'cC'$. Note that at this point of the derivation process the fifth rule as well as the sixth rule can be applied. If we, at this point, would apply the sixth rule we would obtain the sentential form $aA'bBcC'$ and the fifth rule would not be applicable since the symbol B' does not occur in the sentential form. In fact, such a derivation would block and never complete since no rule could be applied. So, after obtaining the sentential form $aA'bB'cC'$ as described above, we apply the fifth, sixth, seventh rule and obtain $aA'bB'cC' \overset{*}{\Rightarrow} aAbBcC$. After this we start the first cycle again (applying second, third, fourth rule again) or terminate the derivation by using the last three rules. We can generate all strings consisting of an equal number of occurrences of a, b, and c.

5 RANDOM PERMITTING CONTEXT GRAMMARS

In this section we construct successively fragments of an rPcg for the supplier phase, invitation phase, declaration phase and reservation phase and show that the reservation phase cannot be described by an rPcg. We illustrate the modelling with the Examples 2, 3, 4, and 5.

Let $q \geq 1$ be the number of enterprises participating in the VBC. The language that we want to generate is L_{VBC} and reflects the outcomes of the supplier phase, invitation phase, declaration phase, and reservation phase. We divide a string in L_{VBC} by the designated symbol \$, where to the left of the \$ symbol invitation and declaration are represented and to the right of \$ the reservation and supplier phase are represented. L_{VBC} is defined as:

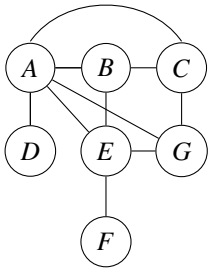

Figure 2: The graph represents a network of associates. Each node label corresponds to a name of an enterprise and an edge connecting a node n_1 with a node n_2 represents that the enterprises at n_1 and n_2 are associates.

$$L_{\text{VBC}} = \{a^{n_1}b^{n_2}c^{n_3}\dots q^{n_q}\hat{\$}a^{m_1}b^{m_2}c^{m_3}\dots q^{m_q}x^k \mid \text{for } 1 \le i \le q, m_i = n_i \text{ or } 0, n_i \ge 1, k \ge 0\}.$$

The non-context-free language L_{VBC} reflects the VBC formation and the VBC itself (that is, the enterprises that will purchase goods and were selected by a purchasing strategy). The letters $a, b, \dots, q$ represent different enterprises and the number of their occurrences (n_1 occurrences of a, n_2 occurrences of b, etc.) represent the quantity of goods they declared (i.e. would want to buy). To the right of the $\hat{\$}$ symbol we represent the enterprises that will purchase goods, possibly followed by some letters x representing items provided by the supplier that the VBC will not buy. In particular, a string in the language represents the result of the invitation phase and declaration phase to the left of the symbol $\hat{\$}$ and the reservation to the right of the symbol $\hat{\$}$. In what follows, we describe how a random permitting context grammar G is constructed to describe the phases up to the reservation phase.

We assume there exists a network of associates represented by a graph. Let the network graph in Figure 2 represent such a network of associates, where A is the name of the initiator enterprise. Each node label corresponds to a name of an enterprise and an edge connecting a node n_1 and a node n_2 represents that the enterprises at nodes n_1 and n_2 are associates. In Figure 2, for example, enterprise A is linked with enterprises B, C, D, E and G and enterprise B is linked with its associates A, C and E, etc.

Assume the network graph N in Figure 2 as given. We start with modelling the initiator enterprise approaching the supplier and the supplier providing the overall quantity that the VBC can purchase. To this end, we construct a fragment of our rPcg $G = (V_N, V_T, P, S)$ as follows.

Example 2. We construct rules of the following form: $S \to A'A_{\text{inv}}X'$, $X' \to X'X$, $X' \to \$$, where $A', A_{\text{inv}}, X', X, \$ \in V_N$. The first rule rewrites the start symbol and is applied once, representing the initiator approaching the supplier. The second rule is applied as many times as the total number of purchasable items the supplier is providing and the third rule is applied once, indicating that the supplier provided a total number of items and that the supplier phase ended. After applying these three initial rules and assuming that the supplier provides ten items, we have a sentential form of the following form (representing the result of the supplier phase):

$$A'A_{\text{inv}}\$XXXXXXXXXX.$$

Now the invitation phase begins where the initiator enterprise begins inviting its associates. Note that a network graph represents the entire network of enterprises, not necessarily the enterprises that are invited to join a VBC. Moreover, note that from a network graph one cannot see when and which enterprises invited which enterprises. Therefore, given a network graph N, we construct an invitation tree θ according to the sequence of invitations and then construct production rules for G. Since the initiator enterprise and all other invited enterprises must not be invited again (by other enterprises at a later time), the set of node labels (i.e. enterprises) in an invitation tree θ are distinct. We construct the invitation tree such that a node labelled with a nonterminal symbol is never introduced more than once. The Algorithm 1 constructs an invitation tree θ from a given network graph N according to the sequence of invitations.

Algorithm 1. `Input`: *A network graph N.* `Output`: *An invitation tree* θ *with n tree layers for the sequence of invitations.* `Method`: *Construct an invitation tree* θ *as follows:*

- *Let the initiator A label the root of* θ *and denote it tree layer* L_0.
- *For each tree layer* L_i, $0 \le i \le n$ *and for each symbol* $A_1, A_2, \dots, A_k$ *in* L_i *do*
  - *for* A_j, $1 \le j \le k$ *inviting associates* $B_1, B_2, \dots, B_{l_j}$ *do*
  - *for all* B_l, $1 \le l \le l_j$ *not occurring in any tree layer* L_g, $0 \le g \le i$ *do*
  - *let* $B_1, B_2, \dots, B_{l_j}$, $1 \le l \le l_j$ *be the children of* A_j.
- *Denote the resulting layer* L_{i+1}.

As an example, assume that enterprise A invites B and C and that B invites E and C invites G. Then θ is as illustrated in Figure 3.

Given an invitation tree θ we construct the rules for the invitation phase. The invitation rules are of the form $A \to \hat{\hat{A}}B_AC_A\dots G_AA_{\text{inv}}B_{\text{inv}}\dots G_{\text{inv}}$. This rule reflects that enterprise A invited enterprises $B, C, \dots, G$

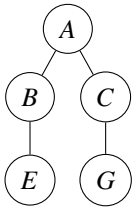

Figure 3: An invitation tree θ representing that enterprise A invites B and C, that B invites E and that C invites G.

by the subscript A on the nonterminals. The nonterminals with the subscript "inv" give the respective enterprises the opportunity to invite their associates. The Algorithm 2 constructs the invitation rules with permitting context.

Algorithm 2. `Input`: *An invitation tree* θ. `Output`: *Invitation rules with permitting context.* `Method`: *Construct the invitation rules as follows:*

- *If the root of* θ *is labelled by A and* $A_1, A_2, \ldots, A_k$ *occur in layer* L_1 *as children nodes of A do*
  - *add the rule*
    $$A_{\text{inv}} \to \hat{\hat{A}} A_{1_A} A_{2_A} \ldots A_{k_A} A_{1_{\text{inv}}} A_{2_{\text{inv}}} \ldots A_{k_{\text{inv}}} (\$;\emptyset)$$
    to P.
- *For all tree layers* L_i, $1 \leq i \leq n$ *and all symbols* A_j, $1 \leq j \leq l$ *occurring in* L_i *do*
  - *for* $j = 1$ *add a rule*
    $$A_{1_{\text{inv}}} \to \hat{\hat{A}}_1 B_{1_{A_1}} B_{2_{A_1}} \ldots B_{p_{A_1}} B_{1_{\text{inv}}} B_{2_{\text{inv}}} \ldots B_{p_{\text{inv}}} (\hat{\hat{A}};\emptyset)$$
    to P, where B_f, $1 \leq f \leq p$, *are the children of node* A_1 *in* L_{i+1} *and* $\hat{\hat{A}}$ *is (the rightmost)* A_l *occurring in layer* L_{i-1}.
  - *for* $j = 2, \ldots, l$ *add a rule*
    $$A_{j_{\text{inv}}} \to \hat{\hat{A}}_j C_{1_{A_j}} C_{2_{A_j}} \ldots C_{r_{A_j}} C_{1_{\text{inv}}} C_{2_{\text{inv}}} \ldots C_{r_{\text{inv}}} (\hat{\hat{A}}_{j-1};\emptyset)$$
    to P, where C_g, $1 \leq g \leq r$, *are the children of node* A_j *in* L_{i+1}.
- *If a node A has no children we add the rule*
  $$A_{\text{inv}} \to \hat{\hat{A}}(\hat{Y},\emptyset)$$
  to P, where $\hat{Y}$ *is either (the rightmost)* A_l *occurring in the previous layer* L_{i-1} *or* $\hat{\hat{A}}_{j-1}$.
- *For the rightmost Z in the last layer* L_n *that has no children add the rule*
  $$Z_{\text{inv}} \to \hat{\hat{Z}}\#(\hat{\hat{Y}},\emptyset)$$
  to P, where $\hat{Y}$ *is either (the rightmost)* A_l *occurring in the previous layer* L_{i-1} *or* $\hat{\hat{A}}_{j-1}$.

The nonterminal symbols of the form $\hat{\hat{A}}$ are introduced in order to keep track of the order of the invitations, because no enterprise invited at a later stage should be able to act prior to an enterprise that was invited before. Note how the permitting context of $\hat{\hat{A}}$ nonterminals is simply carried over to the next layer in a tree or to a sister node to the right in a layer. The symbol # indicates that the invitation phase ended.

For our example, the following rules would be constructed.

Example 3. *We construct*

$$A_{\text{inv}} \to \hat{\hat{A}} B_A C_A B_{\text{inv}} C_{\text{inv}} (\$;\emptyset),$$
$$B_{\text{inv}} \to \hat{\hat{B}} E_B E_{\text{inv}} (\hat{\hat{A}};\emptyset),$$
$$C_{\text{inv}} \to \hat{\hat{C}} G_C G_{\text{inv}} (\hat{\hat{B}};\emptyset),$$
$$E_{\text{inv}} \to \hat{\hat{E}} (\hat{\hat{C}};\emptyset),$$
$$G_{\text{inv}} \to \hat{\hat{G}}\#(\hat{\hat{E}};\emptyset).$$

Applying the rules in our running example we obtain the sentential form:

$A' \hat{\hat{A}} B_A C_A \hat{\hat{B}} E_B \hat{\hat{E}} \hat{\hat{C}} G_C \hat{\hat{G}} \# \$XXXXXXXXXX.$

In the declaration phase all invited enterprises that want to buy items declare how much they would want to buy. For this, we introduce for all nonterminal symbols with a subscript and for A', simple recursive rules that are applied as many times as the number of the items that the respective enterprise wants to purchase. For A' we introduce $A' \to A'A(\#;\emptyset)$ and $A' \to A(\#;\emptyset)$ and for a nonterminal symbol A_B we introduce rules of the form $A_B \to A_B A_B$. For an enterprise A_B that does not want to purchase anything, we introduce a rule of the form $A_B \to A_{B_{\text{OO}}}$ the subscript OO standing for "opt out". The rules for the declaration phase (except the initiator rules) are context-free, since a (leftmost) ordering on the declaration of goods does not matter.

Now we introduce rules that delete all nonterminals A in the sentential form that are marked with $\hat{\hat{A}}$ or the subscript OO. Note that this is a simple design choice that can be made as desired. We make this design choice to simplify the formalisation and for better readability of the paper. In case terminal symbols a are marked with $\hat{a}$ or the subscript OO, the language L_{VBC} must be changed accordingly. We introduce for all nonterminals of the form $\hat{\hat{A}}$ and $A_{B_{\text{OO}}}$ the rules $\hat{\hat{A}} \to \lambda(\#;\emptyset)$ and $A_{B_{\text{OO}}} \to \lambda(\#;\emptyset)$, respectively.

Example 4. *In our example we assume that the enterprises A, B and E want to purchase five items each and that enterprises C and G opt out. Applying the rules, that we construct as explained above, we obtain the following sentential forms:*

$$A'\hat{\hat{A}}B_AC_A\hat{\hat{B}}E_B\hat{\hat{E}}\hat{\hat{C}}G_C\hat{\hat{G}}\#\$XXXXXXXXXX \stackrel{*}{\Rightarrow}$$

$$A^5B_A^5C_{A_{OO}}\hat{\hat{B}}E_B^5\hat{\hat{E}}\hat{\hat{C}}G_{C_{OO}}\hat{\hat{G}}\#\$XXXXXXXXXX \stackrel{*}{\Rightarrow}$$

$$A^5B_A^5E_B^5\#\$XXXXXXXXXX.$$

After the declaration phase a purchasing strategy is chosen outside our model. The chosen purchasing strategy determines which enterprises may purchase items. After the declaration phase (and deleting the marked nonterminals) we obtain sentential forms of the form: $A\ldots AB\ldots BC\ldots C\ldots\#\$X\ldots X$ indicating the quantity each enterprise would want to purchase to the left of the symbol \$ and the quantity provided by the supplier to the right of the symbol \$.

Note that in order to obtain a string in the language L_{VBC}, we would have to rewrite the leftmost nonterminal on the left side of the symbol \$ and then we would have to rewrite the leftmost nonterminal X on the right side of the symbol \$. We basically would need to walk across the sentential form and back. In the following we argue that such a mode of derivation cannot be encoded in the permitting context, but that we need forbidding context too. This kind of rcg with a special kind of leftmost restriction is investigated in (Dassow and Păun, 1989) (leftmost restriction II, page 54). In this leftmost restriction at each step of a derivation the leftmost occurrence of a nonterminal in a sentential form which can be rewritten has to be rewritten. It is shown that the generative capacity of rcg with this kind of leftmost restriction is considerably increased (they generate recursively-enumerable languages) (see Theorem 1.4.4 in (Dassow and Păun, 1989)). If no deleting rules are used, rcg with this leftmost restriction generate context-sensitive languages. A simpler form of leftmost restriction in which at each step of a derivation the leftmost occurrence of a nonterminal has to be rewritten, decreases the generative capacity to context-free.

Lemma 1. *Given a sentential form of the form $w = A\ldots AB\ldots BC\ldots C\ldots\#\$X\ldots X$, there is no rPcg that alternates between rewriting the leftmost nonterminal of the left hand side of* \$ *and the leftmost nonterminal of the right hand side of* \$.

Proof idea 1. *Assume that there exists such an rPcg. Then there exists a rule of the form $A \to \alpha(X;\emptyset)$, $\alpha \in (V_N \cup V_T)^*$ to rewrite the leftmost nonterminal in w. Then there exists a rule the second nonterminal A in w from being replaced. One way to do this is a rule of the form $A \to \alpha'(Y;\emptyset)$, where Y is a nonterminal that is introduced by a rule that rewrites X, i.e. $X \to \hat{\alpha}Y(K;\emptyset)$ for some $K \in V_N$. Then we have the following rules*

$$r_1 : A \to \alpha(X;\emptyset) \text{ and } r_2 : A \to \alpha'(Y;\emptyset).$$

Thus, the first two A in the sentential form w must be distinct, otherwise r_1 is applicable to the second nonterminal A immediately after the first A has been replaced (i.e. not the derivation we want). Extending this discussion in a similar fashion, we conclude that all nonterminals to the left hand side of \$ *must be distinct. A similar discussion holds for the nonterminals X to the right of* \$. *Then w consists of distinct nonterminals and we can build a regular grammar that rewrites w into a regular string $w' \in L_{\mathrm{VBC}}$, which shows the contradiction.*

6 RANDOM CONTEXT GRAMMARS

Thus, at this point we know that random permitting grammars are not powerful enough to model the reservation phase. We can however construct a random context grammar and impose the leftmost restriction II described above in order to model the reservation phase. The Algorithm 3 constructs rcg production rules describing the reservation phase.

Algorithm 3. `Input`: *An invitation tree* θ. `Output`: *Reservation rules with permitting and forbidding context.* `Method`: *Construct the reservation rules as follows:*

- *Let $A_1, A_2, \ldots, A_k$ be the enterprises that have been chosen by the E-strategy or M-strategy.*
- *For every A_i, $1 \le i \le k$ do*
- *Construct for A_1 the following rules:*

$$A_1 \to a_1\hat{A}_1(X;\{\hat{A}_1,\hat{X}_{A_1}\})$$
$$X \to a_1\hat{X}_{A_1}(\hat{A}_1;\hat{X}_{A_1})$$
$$\hat{A}_1 \to \lambda(\hat{X}_{A_1};\emptyset)$$
$$\hat{X}_{A_1} \to \lambda(\emptyset;\hat{A}_1)$$

  where $\hat{A}_1$ and $\hat{X}_{A_1}$ are newly introduced nonterminal symbols in V_N.
- *Construct for all other A_i, $j = 2,\ldots,k$, the following rules*

$$A_i \to a_i\hat{A}_i(X;\{A_{i-1},\hat{A}_{i-1},\hat{X}_{A_{i-1}},\hat{A}_i\})$$
$$X \to a_i\hat{X}_{A_i}(\hat{A}_i;\hat{X}_{A_i})$$
$$\hat{A}_i \to \lambda(\hat{X}_{A_i};\emptyset)$$
$$\hat{X}_{A_i} \to \lambda(\emptyset;\hat{A}_i)$$

  where $\hat{A}_i$ and $\hat{X}_{A_{i-1}}$ are newly introduced nonterminal symbols in V_N.

- *For all other nonterminals B on the left hand side of \$ representing enterprises that were not chosen by a purchasing strategy create the following rules:*

$$B \to b.$$

- *Add $X \to x(\emptyset;\gamma)$ to P to finally replace possible nonterminals X, where $\gamma = \{A_1, A_2, \ldots, A_k, \hat{A}_1, \hat{A}_2, \ldots, \hat{A}_k, \hat{X}_{A_1}, \hat{X}_{A_2}, \ldots, \hat{X}_{A_k}\}$.*
- *Add the rules $\# \to \lambda(\$;\emptyset)$ and $\$ \to \hat{\$}(\emptyset;\#)$ to P, $\hat{\$} \in V_T$.*

Example 5. *Suppose that, in our running example, the E-strategy determined that the two enterprises A and B were chosen to purchase items. Then, the following rules would be constructed:*

$$A \to a\hat{A}(X;\{\hat{A},\hat{X}_A\}),$$
$$X \to a\hat{X_A}(\hat{A};\hat{X_A}),$$
$$\hat{A} \to \lambda(\hat{X_A};\emptyset),$$
$$\hat{X_A} \to \lambda(\emptyset;\hat{A}),$$
$$B_A \to b\hat{B_A}(X;\{A,\hat{A},\hat{X}_A,\hat{B}_A\}),$$
$$X \to b\hat{X}_{B_A}(\hat{B}_A;\hat{X}_{B_A}),$$
$$\hat{B}_A \to \lambda(\hat{X}_{B_A};\emptyset),$$
$$\hat{X}_{B_A} \to \lambda(\emptyset;\hat{B}_A),$$
$$X \to x(\emptyset;\{A,B,\hat{A},\hat{B},\hat{X}_A,\hat{X}_{B_A}\}),$$
$$E_B \to e.$$

Corollary 1. *A rcg modelling a VBC and its four phases has to work in leftmost restriction II.*

Figure 4 illustrates the information that we obtain after each phase. The information is represented in the sentential forms that our grammar generates. In Figure 4 the sentential forms are generalised to the following two parts: left of the \$ symbol and right of the \$ symbol.

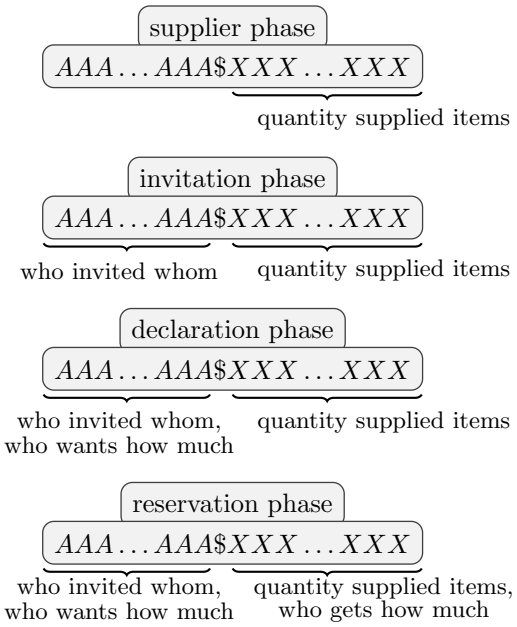

Figure 4: The information that we obtain after each phase represented as generalised sentential forms. The sentential forms are divided into left of \$ and right of \$ and are depicted as $AAA\ldots AAA\$XXX\ldots XXX$. Beneath each left and right part we explain what kind of information is conveyed.

7 VBC MODEL CHANGE

If we want to generate L_{VBC} with random permitting context grammars, we have to change the ordering of the phases in our VBC model and thus the VBC model itself. We generate the information to the left of the \$ symbol and the information to the right of \$ successively with chain rules. In order to be able to control the rewriting we have to have a constant number of nonterminals in each sentential form and we have to build chain rules accordingly. By our previous discussion we know that the occurrences of the nonterminals X cannot be matched against the occurrences of the nonterminals to the left of \$ with only permitting context. Therefore, we cannot initiate a VBC with letting the supplier provide the quantity of purchasable items. Instead the phases of the changed VBC model are the following. Each enterprise participating in the VBC (starting with the initiator)

- declares the quantity of goods it wants to purchase to the left and to the right of \$ symbol, and then
- invites its associates from its network.

After the declaration and invitation phases, the purchasable quantity provided by the supplier is inquired and a purchasing strategy is chosen accordingly outside the grammar model. Once the enterprises are determined by the E-strategy or M-strategy a string in L_{VBC} is generated. In this alternative VBC model the declaration phase and invitation phase alternate for each enterprise that is invited to join the VBC. This process is illustrated in Figure 5 for n enterprises. Moreover, Figure 5 shows that after all enterprises declared and invited their associates, the supplier provides purchasable items. Depending on the formed VBC and chosen purchasing strategy, the process finally leads to a deal or non-deal.

Algorithm 4 generates the fragments of the rPcg for the declaration phase and invitation phase. Some of the design choices can be made differently but the

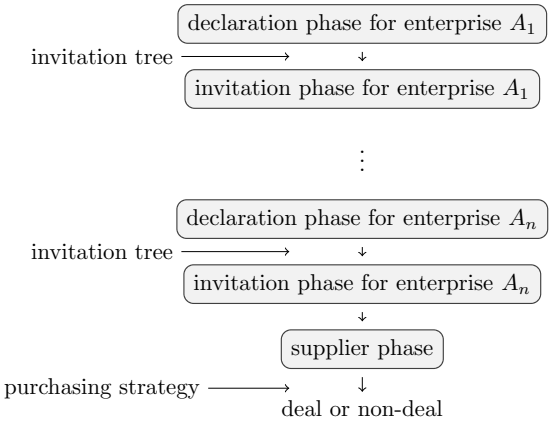

Figure 5: An illustration of the individual processes of the alternative and restricted VBC model.

quintessence is that information cannot be distributed over several phases and co-ordinated later, but has to be introduced rather locally.

Algorithm 4. `Input`: *An invitation tree* θ. `Output`: *Production rules for the declaration and invitation phase for each enterprise participating in the alternative VBC model.* `Method`: *Construct the rules as follows:*

- *Let* $S \to A\$X_A$ *be a rule.*
- *For each tree layer* L_i, $0 \leq i \leq n$ *and for each symbol* $A_1, A_2, \ldots, A_k$ *in* L_i *do*
  - *for* A_j, $1 \leq j \leq k$ *construct declaration rules as follows*

$$A_j \to A_j A'_j(X_{A_j};\emptyset)$$

$$X_{A_j} \to X_{A_j} X'_{A_j}(A'_j;\emptyset)$$

$$A'_j \to A_j(X'_{A_j};\emptyset)$$

$$X'_{A_j} \to X_{A_j}(A_j;\emptyset)$$

  - *for* A_j, $1 \leq j \leq k$ *inviting associates* $B_1, B_2, \ldots, B_{l_j}$ *construct the following rules*

$$A_j \to B_1 B_2 \ldots B_{l_j}$$

$$X_{A_j} \to X_{B_1} X_{B_2} \ldots X_{B_{l_j}}(\{B_1 B_2 \ldots B_{l_j}\};\emptyset)$$

  - *for the leaf nodes A in* θ *construct the following rules*

$$A \to \lambda$$

Note that the rules have to be applied in the order they are constructed in Algorithm 4, since otherwise the derivation blocks. After applying rules constructed in Algorithm 4 we obtain a sentential form of the form (here simplified for better readability):

$$A^{n_1} B^{n_2} \ldots \$X_A^{n_1} X_B^{n_2} \ldots$$

Once a purchasing strategy is chosen after consultation with the supplier about the quantity of purchasable goods, context-free rules can be generated as outlined in Algorithm 5.

Algorithm 5. `Input`: *A sentential form w resulting from the declaration and invitation phase and the chosen enterprises by the purchasing strategy.* `Output`: *Production rules for terminating the derivation.* `Method`: *Construct the rules as follows:*

- *Let* $w = A^{n_1} B^{n_2} \ldots Z^{n_p} \$X_A^{n_1} X_B^{n_2} \ldots X_Z^{n_p}$.
- *Construct for all nonterminals* A_j *occurring to the left of* \$ *rules of the form*

$$A_j \to a_j$$

- *Construct for all chosen enterprises* B_j *rules of the form*

$$X_{B_j} \to b_j$$

- *Construct for all other nonterminals X occurring on the right hand side of* \$ *the following rules:*

$$X \to \lambda$$

8 CONCLUSIONS

We showed how to model the supplier phase, invitation phase, and declaration phase of a VBC with random permitting context production rules. Then we argued that to model the VBC reservation phase we need random context grammars under a special kind of leftmost derivation. If we interpret the modelling from an information distribution and cooperation view, then we showed that if VBC relevant information is distributed over supplier phase, invitation phase and declaration phase and must during the reservation stage be synchronised and co-ordinated, the formal grammar needs to be very powerful with respect to mode of derivation and thus generative capacity. From a perspective of representing the enterprises and the supplier as agents, this model is adequate as the VBC formation is a process in which several agents cooperate and synchronise information. In particular, we need rcgs that work under a special kind of leftmost restriction and generate

recursively-enumerable languages (see (Dassow and Păun, 1989)).

If we use random permitting context grammars for all processes, we can only model a VBC formation during which information is introduced and processed locally and successively rather than being spread over the different VBC processes. Thus, supplier phase, invitation phase, declaration phase and reservation phase have to be altered and have to be synchronised in a more local and successive way. In this alternative model, the supplier is not part of the model, which may have its benefits, as the enterprises act instead of react to a supplier offer. A disadvantage is that VBCs may be formed without making a purchase in the end (e.g. if the quantity the VBC wants to purchase does not coincide with the quantity provided by the supplier). Finally, the alternative VBC model discussed in Section 7 does not achieve that all enterprises are able to see what is happening and has happened during the VBC formation (e.g. the amount of available goods, the enterprises participating, etc.), which is an unfavourable position for the very small enterprises working in isolation.

We see the two main benefits of our approach, that is, modelling the VBC and the VBC formation with grammars of regulated rewriting, as the following. One is that grammars model generative processes and the formation of a VBC is a generative process that involves several enterprises, which can actively participate in the formation. A model of such a generative process can shed light on the structural conditions of a VBC and can thus guide an implementation (as we have shown in this paper).

The second main benefit is that once an adequate model is found, one can draw many implications from the rich mathematical theory behind regulated rewriting. One example is the observation that we showed in this paper, namely if we restrict the grammar model we cannot model the four successive phases of a VBC formation and if we want an adequate model we have to have more powerful grammars (in terms of lifting the rewriting mode to a specific leftmost restriction). Another example is parsing. A parsing algorithm decides first if an input string w is in a certain formal language L and then assigns a structural representation to w (i.e. a derivation tree). A parsing algorithm can be useful in a VBC context if several enterprises $a_1, \ldots a_l$ want to form a VBC (independent of the four successive VBC formation phases). These enterprises would, for instance, input a string $w = a_1^{n_1} \ldots a_l^{n_l} \$ a_1^{n_1} \ldots a_l^{n_l}$ in which the quantity of the items that each participating enterprise wants to purchase is represented and let the parsing algorithm decide whether $w \in L$ (representing the question whether it is possible for those enterprises to form that specific VBC). Structural representations from parsing might give further insights into the VBC structure. This is a more bottom-up approach differing from the top-down approach that we described in this paper. An interesting observation in this context is that the restricted VBC model, even though it does not model the four successive phases, might be more useful due to more efficient parsing algorithms.

Other mathematical properties such as closure properties for certain formal languages may be useful in situations in which two networks of different enterprises want to combine into a larger network. Can we in such a situation use the same grammar formalisms? Or does joining two networks and their respective grammars lead beyond the grammar model?

Apart from investigating mathematical properties relevant for VBCs (e.g. parsing, closure), future work will also focus on implementations of the two VBC model approaches presented in this paper and on evaluations with respect to efficiency of the VBC formation process, number of closed deals, and user (i.e. enterprise) satisfaction. In the context of implementation, one interesting question is how to translate natural language text into the respective grammar rules or sentential forms and vice versa.

REFERENCES

Dassow, J. and Păun, G. (1989). *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag.

Ewert, S. and Van der Walt, A. (2002). A pumping lemma for random permitting context languages. *Theoretical Computer Science*, 270:959–967.

Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316.

Mashkov, V., Barilla, J., Simr, P., and Bicanek, J. (2015). Modeling and simulation of coalition formation. In *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2015)*, pages 329–336.

Merz, C. (2010). Incubating micro enterprises in rural South Africa - The use case of virtual buying cooperatives. In *ICT for the Next Five Billion People*, pages 35–45, Berlin, Germany. Springer Berlin Heidelberg.

Merz, C., de Louw, R., and Ullrich, N. (2007). Collaborative working environments for enterprise incubation: The Sekhukhune rural living lab. In *Proceedings of IST Africa*, pages 173–186, Maputo, Mozambique.

Ngassam, E. and Raborife, M. (2013). Virtual Buying Cooperative: A procurement model for improving the sustainability of very small retailers in emerging

economies. In *Proceedings of IST Africa*, Nairobi, Kenya.

van der Walt, A. P. J. (1972). Random Context Languages. *Information Processing*, 71:66–68.

Xavier, S. (2005). *Theory of Automata, Formal Languages and Computation*. New Age International Pvt Ltd Publishers.

Zetzsche, G. (2010). On erasing productions in random context grammars. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Proceedings, Part II*, pages 175–186.