Comments on Csuhaj-Varjú

Mark-Jan Nederhof
University of Groningen
The Netherlands
markjan@let.rug.nl

Abstract. We investigate some of the definitions in the paper commented upon, and we propose some extensions.

1 Colonies versus context-free grammars

Here we investigate Definition 3.1, which introduces colonies. A colony is a system of regular grammars generating finite languages; each such regular grammar is called a *component* of the colony. Because a terminal symbol of one colony may also be the start symbol of another, the components may be activated to derive a string in cooperation.

There are two ways of having the components cooperate. The first is called b-mode, which means that one component after the other is used to rewrite one particular symbol of the sentential form. The second is called t-mode, which differs from the b-mode in that in each iteration a particular component is to rewrite *all* occurrences of the corresponding start symbol.

The rewriting process starts with one particular start symbol S, and may be finished when the sentential form consists only of symbols in a certain set of terminals T.

We will present colonies in a slightly different form, in order to allow comparison with context-free grammars. First we will consider only the b-mode, and in Section 2 we will show that the t-mode increases the generative capacity.

Looking at the definition in the paper, we see that the regular grammars, i.e. the components, are to generate finite languages.[1] This means that we can enumerate all ways that the start symbol can be rewritten to a string. For example, we may have

$$\begin{array}{ccc} S_i & \rightarrow & \alpha_1 \\ S_i & \rightarrow & \alpha_2 \\ & \vdots & \\ S_i & \rightarrow & \alpha_p \end{array}$$

where S_i is the start symbol of the i-th component, and α_1, ..., α_p are the strings that S_i can be rewritten to. The enumeration of the p ways to rewrite S_i can be given as p context-free rules, as we have done above. The purpose of this notation is to make clear that colonies can be seen as context-free grammars: If we collect all such context-free rules from all components, then a context-free grammar results.

A few small details need to be settled. First, the start symbol of the context-free grammar naturally is S.

Second, we need to distinguish between terminals and nonterminals. Some symbol may be treated differently by different components: a terminal symbol of one may be the start symbol of another. The solution is to treat *all* symbols in the components as nonterminals in the context-free grammar. This means that for the rules $S_i \rightarrow \alpha_j$ that we derive from each component, all symbols in α_j will be considered to be nonterminals. The terminals will now be chosen to be fresh symbols, one for each symbol in T.

More precisely, if the symbols in T are $A_1, \ldots, A_m$, then we introduce fresh terminal symbols $a_1, \ldots, a_m$. We then add the rules

$$\begin{array}{ccc} A_1 & \rightarrow & a_1 \\ A_2 & \rightarrow & a_2 \\ & \vdots & \\ A_m & \rightarrow & a_m \end{array}$$

Now we have a full context-free grammar that generates the same language as the colony it was derived from, provided this colony is interpreted in b-mode. (For the purpose of comparing the two languages, we need to identify the fresh terminals a_i with the corresponding symbols A_i.)

The opposite construction is also interesting: Can we find a colony for each context-free grammar? The answer is yes for the acceptance styles (i), (ii) and (iv), as Theorem 3.1 already suggests.

The construction is as follows. Consider a context-free grammar.

We construct the colony by taking each context-free rule separately as a component. That is, for each rule $A \rightarrow \alpha$ we construct a component, as a regular grammar with start symbol A generating α as only string.

The start symbol of the colony is the start symbol of the context-free grammar, and the set T is the set of all terminals in the context-free grammar. With regard to the acceptance styles (Definition 3.3), the terminal symbols of the components need to be chosen with some care. In the case of style (i), *"arb"*, and style (iv), *"all"*, we can take the set of terminals for each component to be the set of all grammar symbols,

[1] Actually, nowhere is any use made of the assumption that the components are regular grammars. Only the finiteness of the generated languages is relevant to the discussion. In Section 3 we will investigate what happens when the constraint on finiteness is dropped.

© 1996 M.J. Nederhof
Proceedings of the ECAI 96 Workshop
Extended Finite State Models of Language
Edited by A. Kornai.

in which case $T \subseteq \cup_{i=1}^{n} T_i$ and $T \subseteq \cap_{i=1}^{n} T_i$ obviously hold.[2]

In the case of style (ii), *"one"*, the problem is solved by furthermore introducing one dummy component, with a fresh symbol as start symbol, generating the empty language, and having T as terminal set. Then obviously $T = T_i$, for some i, viz. for the i corresponding to that dummy component.

We cannot make the construction in the case of style (iii), *"ex"*, since in general, the set of terminals for each component should contain at least the symbols in the right-hand side of the context-free rules from which it is derived, which generally contains more than the terminals of the grammar.

It seems that the class of languages $\mathcal{L}(Col, b, ex)$ corresponds to those languages that can be described by context-free grammars in which the terminal symbols occur only in a subset of the rules of the form:

$$\begin{array}{lcl} A_1 & \rightarrow & a_1 \\ A_2 & \rightarrow & a_2 \\ & \vdots & \\ A_m & \rightarrow & a_m \end{array}$$

where $A_1, \ldots, A_m$ is a list of all nonterminals in the grammar without duplicates, and $a_1, \ldots, a_m$ is a list of all terminals in the grammar without duplicates. In other words, there is a one-to-one mapping between nonterminals and terminals.

My conjecture is that for any context-free language L, there is a grammar with the above constraint on the occurrences of the terminal symbols, such that the intersection of the generated language and the language Σ^*, some alphabet Σ, is L. In other words, by taking a language from $\mathcal{L}(Col, b, ex)$, and eliminating those strings containing unwanted terminals, we may obtain any context-free language.

2 Colonies in t-mode

If we want to see colonies as context-free grammars then we need an additional concept in order to allow modeling of t-mode derivations.

Consider for example the following context-free grammar.

$$\begin{array}{lcll} S & \rightarrow & ABABA & (1) \\ A & \rightarrow & CA & (2) \\ A & \rightarrow & C & (3) \\ B & \rightarrow & b & (4) \\ C & \rightarrow & c & (5) \end{array}$$

This may have been derived from a colony by the process described above. The rules (1), (2) and (3) may have come from 3 distinct components, and rules (4) and (5) may have been introduced because B and C were terminals in T.

The colony in t-mode would generate the language $\{c^k b c^k b c^k \mid k \in \{1, 2, \ldots\}\}$. Modeling this in the context-free grammar above requires that at each derivation step, all occurrences of a certain nonterminal need to be rewritten using context-free rules originating from the same component of the colony. In the running example, this means that either all occurrences of A need to be rewritten by means of rule (2) or all by means of rule (3), but rules (2) and (3) should not be applied simultaneously.

In the general case, for each nonterminal A we need to partition the grammar rules defining A into a number of sets, each of which corresponds to one particular component of the colony.

3 A generalization of colonies

It is well-known that the generative capacity of context-free grammars is not increased when right-hand sides of rules are generalized to be regular expressions instead of linear strings over grammar symbols. Such grammars are known under the name "extended context-free grammars". (See for example Purdom&Brown, 1981.)

For example, consider a rule $A \rightarrow (b \cup c)^*$. This rule can be rewritten to a number of traditional context-free rules which together have the same meaning:

$$\begin{array}{lcl} A & \rightarrow & \epsilon \\ A & \rightarrow & bA \\ A & \rightarrow & cA \end{array}$$

The general case can be handled by considering that regular expressions can be transformed into finite automata, and from the transitions of these finite automata, the linear context-free rules can be easily derived. (Note that the resulting grammar is regular, apart from epsilon rules.)

In light of this fact, one may wonder if the same generalization can be applied to colonies. By Definition 3.1, the languages generated by components of colonies are not only restricted to be regular, they are even constrained to be finite, and dropping this restriction on finiteness is what we are interested in here.

Suppose therefore that the languages of the components can be described by means of regular expressions. By generalizing the transformation from Section 1 from colonies to context-free grammars we obtain an easy proof that preservation of the generative capacity of context-free grammars under extension with regular expressions carries over to colonies in b-mode: Colonies in which the components generate arbitrary regular languages, as opposed to finite languages, can describe the same class of languages, provided the b-mode is applied.

The remaining question is whether the situation is different for t-mode. My conjecture is that the generative capacity does indeed increase. Consider for example

$$\begin{array}{lcl} S & \rightarrow & ABABA \\ A & \rightarrow & C^* \\ B & \rightarrow & b \\ C & \rightarrow & c \end{array}$$

At the three occurrences of A in the sentential form $ABABA$ where rewriting to a number of C's takes place, there is no "internal synchronization" between the three rewriting processes, so that a distinct number of C's can be produced for each occurrence of A. Although in this case the same language could be described by a traditional kind of grammar derived from a colony, viz.

$$S \rightarrow A_1 B A_2 B A_3$$

[2] There is no reason why the set of terminals of a grammar may not contain some that do not occur in any terminal string.

$$\begin{array}{rcl}
A_1 & \rightarrow & CA_1 \\
A_1 & \rightarrow & \epsilon \\
A_2 & \rightarrow & CA_2 \\
A_2 & \rightarrow & \epsilon \\
A_3 & \rightarrow & CA_3 \\
A_3 & \rightarrow & \epsilon \\
B & \rightarrow & b \\
C & \rightarrow & c
\end{array}$$

my hunch is that it may not be possible in all cases.

In this limited commentary, we will not take into consideration the more advanced concepts presented in the paper, such as extended colonies and structured colonies. Possibly, generalizations similar to those presented above can be applied to these concepts as well.

REFERENCES

[1] P.W. Purdom, Jr. and C.A. Brown, 'Parsing extended LR(k) grammars', *Acta Informatica*, **15**, 115–127, (1981).