1

1965 International Conference on Computational Linguistics

SOME QUESTIONS OF LANGUAGE THEORY

S. Abraham

53, Uri u., Budapest I., Hungary
Computing Centre of the Hungarian Academy of Sciences

ABSTRACT. It is shown that the assumption that language is non-finite involves the use of a constructive logic which leads to some restrictions on language theory and to the fact that the only possible definition of language is that proposed by generative grammars. Generative grammars can be formulated as normal /Markov/ algorithms and thus their study can be reduced to the study of such algorithms of a special type. A new type of generative grammar is defined, called matrix grammar. It is shown that a language generated by a context-restricted grammar can be also generated by a matrix grammar. Some properties of matrix grammars are shown to be decidable. The problem of the explicative power of generative grammars is discussed.

1. Language metatheory, as indeed any metatheory, must exactly specify the operations allowed in building up the theory /of language/. This may be done by choosing the logic of the theory.If language is considered a non-finite set,a constructive logic /Kolmogorov/ must be choosen. This entails some restrictions on the notions and methods to be used in language theory. Namely, we can not speak of actually infinite sets, and we can not use the quantifiers "there exists" and "all". Thus we can not include in language theory the notion of 'language' itself in the usual way, as the set of all /grammatically or semantically/ correct sentences. Similarly, we can not make use of "distributional analysis" /at least without any restrictions/ , as it generally has the form:

/the sentence/ s_1 has the property R_1 , if there exists /a sentence/ s_2 with the property R_2 /not necessarily $R_1 \neq R_2$ /.

It follows that the single way of defining language is that proposed by generative grammars. These grammars are in fact devices that produce /generate/ the sentences of a language /and only those/, one after the other. So, at every moment we have generated a finite set of sentences, and at the same, if the grammar is properly constructed, at evry moment we can generate a sentence not yet generated before. So, in fact the language /the set of all the sentences of the language/ is a potentially infinite set and the abovementioned difficulties do not arise. The restrictions to be respected within generative grammars as to the /logically correct/ notions and operations are precisely formulated /it may be interesting to note that Chomsky does not respect all of them/.

2. Most of the properties /and possibly even the most important ones/ of generative grammars are obtained by constructing automata, equivalent to

different generative grammars, and in this way using the results of automata theory. It is shown that a more natural /and easy/ way to study generative grammars is to formulate them as normal /Markov/ algorithms [7], [1]. So, if given a Phrase Structure Grammar G it can be given a finite set of normal algorithms $\bar{G} = \{\mathfrak{A}_i\}$ so that by applying the algorithms to the initial strings we obtain the language generated by G.

The algorithms $\mathfrak{A}_i$ have the properties:

(i) each rule /of the algorithm/ rewrites at once only one symbol;

(ii) by applying a rule to a string the length of the string is not diminished.

For constructing $\bar{G}$ we must be able to compose the normal algorithms so that these properties should be preserved. The composition rule given by Markov does not fulfil this condition. So the following composition rule is proved and used:

If $\mathfrak{A}_{V_p}$, $\mathfrak{B}_{V_p}$ are two normal algorithms with the properties (i) and (ii) then for every $\sigma \in \Sigma$ /the set of initial strings/ we have

$$\mathfrak{L}(\sigma) = (\mathfrak{A} \circ \mathfrak{B})(\sigma) = \mathfrak{B}(\mathfrak{A}(\sigma))$$

where $\mathcal{L}$ is a normal algorithm with the scheme

$$\begin{cases} \bar{\xi}\eta \longrightarrow \bar{\xi}\bar{\eta} \\ \bar{\mathfrak{L}} \\ \xi\bar{\eta}\# \longrightarrow \cdot\,\xi\eta\# \\ \bar{\xi} \longrightarrow \xi \\ \mathfrak{A} \\ \xi \longrightarrow \bar{\xi} \end{cases}$$

where $\xi, \eta \in V_p$; $\bar{\xi}, \bar{\eta}$ are symbols put in one-to-one correspondence to the symbols from V_p /and different from them and between them/; $\bar{\mathfrak{L}}$ is the list of the rules of the algorithm $\mathfrak{L}$ with every ξ changed to $\bar{\xi}$. Evidently $\mathcal{L}$ has the properties (i) and (ii) .

It is shown that to a set of algorithms $\bar{G} = \{\mathfrak{A}_i\}$ a single algorithm $\mathfrak{A}$ corresponds if Σ /the set of the initial strings/ is properly enlarged, so that $L(G) = \mathfrak{A}(\Sigma)$. Thus the study of PSG is reducible to the study of normal algorithms of the type of $\mathfrak{A}$ /the rewriting rules of which are, in fact, context-restricted rules/. The sufficient and necessary conditions are established for generating a non-finite language /by different generative grammars/.

It is shown that each singular transformation /Chomsky/ can be formulated as an algorithm of type $\mathfrak{A}$.

The most studied generative grammars are the context-free grammars /CFG/ and the context-restricted grammars /CRG/. Some properties of these grammars are considered to be undecidable. In this respect they are also different. The differences are formulated in Table 1 [6]:

Property	CFG	CRG
1. is the language generated by a grammar empty ?	D	U
2. is the language generated by a grammar infinite ?	D	U
3. for any strings Φ , ψ can some string including ψ be derived from Φ in a grammar ?	D	U

where D indicates that the property in question is decidable, U that it is undecidable.

The CF grammars have not the necessary generative power to model natural languages. The CR grammars may have this power /altough this problem has not been cleared up/ but the undecibality of the properties 1 - 3 /especially, 3/ makes highly doubtful their fitness for modeling natural languages.

A new type of generative grammars is proposed under the name of matrix grammars /MG/ [2].

A matrix grammar is a quintuple

$$G = (V, V_t, \Sigma, F, F^*)$$

where

$$\bar{G} = (V, V_t, \Sigma, F)$$

is a context-free frammar and F^* is a finite set of matrices /called matrix rules/ defined as follows:

(1) f^* is a matrix rule if it has the form

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$f_i \in F$ $(1 \leq i \leq n)$ and not necessarily $f_i \neq f_j$;

(2) f^* is a matrix rule if it has the form

$$\begin{bmatrix} f_1^* \\ \vdots \\ f_n^* \end{bmatrix}$$

where f_i^* are matrix rules or belong to F.

To apply a matrix rule f^* to a string x means to apply to x all the context-free rules which form it, in the given order /to apply a CF rule to a string means to replace the first occurence of its left-side with its right-side/. If at least one of these context-free rules can not be applied to x, we say that f^* can not be applied to x.

It is shown that for any context-restricted grammar G it is possible to construct a /strongly/ equivalent matrix grammar.

For instance, the /not context-free/ language

$$L = \{ a^n b^n c^n \}$$

is generated by the matrix grammar

$$G = (V, V_t, \Sigma, F)$$

with

$$V = \{ S, X, Y, Z, a, b, c \} \; ; \; V_t = \{ a, b, c \} \; ; \; \Sigma = \{ S \}$$

$$F: [S \rightarrow abc]$$
$$[S \rightarrow aXbYcZ]$$
$$\begin{bmatrix} X \rightarrow aX \\ Y \rightarrow bY \\ Z \rightarrow cZ \end{bmatrix}$$
$$\begin{bmatrix} X \rightarrow a \\ Y \rightarrow b \\ Z \rightarrow c \end{bmatrix}$$

It is shown that the properties 1,2,3 are decidable for matrix grammars. So the statement that they are undecidable for the CR grammars is erroneous /the erroneousness of the proof of the undecidability of property 3 given in [5] can be easily shown/. So the fitness of these grammars for modeling natural languages is most likely.

As we have mentioned, for each singular transformation a normal algorithm can be constructed which contains only context-restricted rules. Departing from this, it can be shown that for a transformational grammar /containing only singular transformations, see [4] / a weakly equivalent matrix grammar can be constructed.

The matrix grammars can be formulated as a normal algorithm, too.

Since any normal /Markov/ algorithm can be reversed, it is possible to devise a method for the construction of a recognition grammar corresponding to any given generative grammar. As the matrix grammar corresponding to a transformational grammar is, in general, only weakly equivalent to the latter, and in automatic /natural/ language processing /and especially in machine translation/ the adequate analysis is a crucial requirement, the to strong requirement of Chomsky to derive the structure of a generated sentence from the way it is generated, is dropped, and the matrix grammar is completed with a definitional apparatus /DA/ that makes it possible to assign to a generated sentence the same structure /analysis/ as is assigned by a transformational grammar /details see in [3] /. By constructing the recognition grammar corresponding to a given generative grammar, the DA of the generative grammar is taken over.

3. Some examples are shown how the above considerations can be applied to automatic processing of natural languages.

BIBLIOGRAPHY

[1] Abraham,S., A Formal Study of Generative Grammars I, Computational Linguistics II, pp.5-18, 1964

[2] Abraham,S., Some Questions of Phrase Structure Grammars I,/under press in Computational Linguistics IV/

[3] Abraham,S., Kiefer,F., Some Questions of Language Theory /under press in Acta Hungarica/

[4] Chomsky,N., Categories and Syntactic Theory, MIT Press, 1964

[5] Chomsky,N., Formal Properties of Grammars in Handbook of Mathematical Psychology, J.Wiley and Sons,Inc. New York,NY,vol.2, pp.323-418, 1963

[6] Landweber,P.S., Decision Problems of Phrase Structure Grammars, IEE Trans.,vol.EC-13, pp.354-362, 1964

[7] Markov,A.A., Teorija algorifmov, Trudy Mat. Inst. AN SSSR, Moscow, 1954