

NODES AND NETS: EXPLORATIONS IN RELATIONAL NETWORK NOTATION

by *Rüdiger Schreyer*

INTRODUCTION

A theory of language, like any other theory, must be coherent and consistent. Vague theoretical statements make it impossible to determine coherence or consistency. One way of escaping the danger of vagueness is to formalize theoretical statements. The strong trend towards formalized linguistic models that we have been witnessing in the past two decades is at least partly due to a serious attempt to improve the quality of linguistic models. True, the prestige of formalization has produced "a bull-market in pseudo-mathematical and quasi-scientific gadgetry" (Mey 1972:116), but it has also led to the creation of some formal linguistic models of great sophistication. In the minds of some linguists, Stratificational Grammar (SG) is one of them. Sampson (1970:10), for instance, calls it "the most highly developed theory . . . of communicational descriptions of language" known to him. This praise of SG is in strange contrast with Mey's (1972:115) statement that SG "is rejected by most of the theoretical grammarians on purely formal grounds." The question whether SG is or is not a *formal* model of language, is, perhaps, not easy to answer. It is hoped that the following examination of some fundamental aspects of the relational network notation used in SG will contribute to a clearer understanding of the principles underlying the formal mechanism employed. I shall raise some questions to which I have no answer. Whether they will lead to the creation of better networks in a formal sense remains to be seen.

I. STRATIFICATIONAL THEORY AND NOTATION

Students of Stratificational Grammar consider language as a code that enables the conversion of non-linear semantic structures into linear phonetic structures and vice versa. This code is described as a network of relationships

Rüdiger Schreyer is Akademischer Oberrat at the Institut für Anglistik, Aachen, West Germany.

connecting meaning and sound. Like its original, the relational network model is a highly complex information-processing system. It is constructed by combining a limited number of nodes, each representing a basic relationship. Although more than just two relationships are involved, "the basic dichotomy is between AND and OR" (Lamb 1974:215). The nets of SG are somewhat reminiscent of the graphic representation of nerve nets, but there is no suggestion that today's stratificational networks are models of real nerve nets. Lamb (1974:195) believes that "in the actual *mental structures* involved we will some day find the same kind of neural configurations." On the other hand, there are warnings "NOT to interpret the relational network approach as a neurological theory of language" but rather "as a formal system within which a few neurologically motivated restrictions can be incorporated, but which must stand and fall entirely with its ability to handle linguistic and psycholinguistic data" (Reich 1973a:90-91). Virtually the same point is made by Christie (1977:18). In addition to relational network notation, SG uses an algebraic notation. Lamb (1974:215) does, however, regard the former as the principal mode of representation, and he emphasizes that "there is no reason graphic networks cannot be formalized to just as high a degree of refinement as algebraic notation."

Stratificationists take notation seriously. They refuse to believe that the only thing that "really counts are the thoughts and ideas, and that the means of expressing them is relatively unimportant" (Lamb 1974:203). This concern with notation finds expression in the changes of the overall structure of networks, in the addition of new types of nodes, in the discussion of node properties, in the development of a simplicity measure of linguistic description based on these properties, and in micronode analysis. Relational networks are not just notational devices; they are intended to be models of language. This is the reason why the notation has been devised, and, one might add, is being changed "in such a way that it is as close as possible to the actual structure we are trying to represent" (Lamb 1974:203). Stratificationists, then, do aim at a structural similarity between their networks and the linguistic code. For SG the question "whether graphic representation occupies in linguistic theory the place of models in scientific theory" (Stewart 1976:110) must be answered in the affirmative. Furthermore, researchers do explore the "neutral analogy" offered by the model by, for instance, extending it to describe linguistic change (Christie 1977). The following statement certainly fits Stratificational Grammar, at least in its intention:

A notion may be fleeting, doubtful, vague; once it is codified as a method of representation, a figure, it must be reckoned with in all its implacable detail. Graphic representation influences scientific theory in two ways: in its extension and in its development. The extension of a theory is its application to more kinds of data than that for which it originated; the development of a theory is its articulation in progressively greater detail, and its modification. (Stewart 1976:82)

Lamb (1974:203) notes that notation influences thinking, and that this is necessarily so: "One cannot think without having the information stored in some way. Information has to be in some kind of *medium* and any scientist uses his notation as a primary medium in which he does his thinking." On the other hand, if "one thinks in terms of the notation that one uses" (Lamb 1974:203) there is a very real danger of the notational representations replacing reality.

... it may be the case that the operations that have been "established" (or, rather, assumed) to prop up the intuitive statements do not seem to be connected with the latter except by the author's wishful thinking. So, upon encountering this kind of "formalized" statement, the well-advised reader should keep in mind the age-old truth that "Sayin' it don't make it so." (Mey 1972:112)

Necessary and useful as it may be, model-building is not without its hazards. To avoid the pitfalls outlined above, the model (whether algebraic or graphic) must be tested against the data both in its conception and in its subsequent detailed development, and particularly in its extension. In linguistic modeling this poses a host of theoretical and practical questions. But they will have to be tackled lest linguistic theory be nothing but "a product of its representation" (Stewart 1976:81).

In this study I shall take up the point made by Mey: i.e., that it must be ensured that the suggested formalization is, in actual fact, a formalization of the intuitive statements to be represented. If a network model of the linguistic code is to be of any value, its building blocks—the relationships represented by individual nodes—must be well-defined. In other words, there must be a systematic connection between the verbal definition and the relationship to be defined. This also means that there must be a uniform definitional principle for all nodes.

Most works on SG contain definitions of the nodes employed, but even a superficial examination will reveal that not all definitions have been executed with the same clarity and that not all nodes are equally well-defined. In many definitions the proponents apparently try to kill two birds with one stone: they attempt to define a given relation while at the same time explaining its linguistic relevance. This is often done only at the cost of definitional clarity and consistency. Even a study that specifically addresses the problem of node definition (Sullivan 1977:395-399) is not exempt from this criticism (cf. Schreyer Ms.).

Along the crooked path of progress of Stratificational Linguistics, more and more nodes have been introduced to cope with the exigencies of linguistic description or with the performance requirements of the network (Sampson 1970:17; Lockwood 1972:31-73; Reich 1973a; Johannesson 1976:100-105; Christie 1977:4-13). It is, however, not always easy to decide whether the new nodes improve the relational network description, or whether they only complicate it. It would be advisable to follow the principle that relationships should not be multiplied beyond necessity. But how can we determine whether that

necessity exists? I suggest that no new node should be introduced that can be defined by a combination of the basic nodes.¹

In sum, there seem to be several good reasons for studying the relational networks of SG as formal systems, and it seems to me useful to begin by examining in detail the node definitions offered by stratificational linguists.

II. DEFINITION OF THE BASIC NODES

In this section, I examine the verbal definitions of the relationships represented by the four nodes commonly accepted as basic: the ordered and unordered OR and AND.² My goal is to establish as accurately as possible the relations they are meant to stand for. I do not distinguish between upward and downward nodes, since this distinction does not increase the basic types of relationship. But as SG nodes are considered two-way nodes that operate in a downward direction in encoding (meaning into sound) and in an upward direction in decoding (sound into meaning), one must distinguish between an upward and a downward definition for each node.

Networks can be described by their inputs and outputs; Lamb's (1966:10, 42) dynamic interpretation of a network assumes that the lines connecting the nodes can be activated, an activation being the movement of an impulse along that line:

If t is a top line of a linguistic graph G , then a **DOWNWARD OUTPUT OF t THROUGH G** is a specific combination of (one or more) line-activations of bottom lines of G that can result from a downward impulse along t . Similarly, if b is a bottom line of G , then an **UPWARD OUTPUT OF b THROUGH G** is a specific combination of (one or more) line activations of top lines of G that can result from an upward impulse along b . (Lamb 1966:42)

Nodes being elementary networks, they can also be understood as input-output systems. In fact, Lamb (1966:10) suggests that the relationships symbolized by the nodes "are perhaps most easily understood in terms of the dynamic interpretation," and his are still the most systematic definitions with reference to inputs and outputs.³ This type of definition is, in my opinion, the clearest that has been suggested so far. I shall show that at least three of the basic nodes' input-output relations allow us to identify them with certain logical formulae.

I. Unordered OR—Simple disjunction



FIG. A.

Downward definition. Lamb's (1966:10) definition states that a goes to b or c . He emphasizes that the node represents an exclusive OR (XOR). Lockwood's (1972:34-36) examples and Christie's (1977) and Sullivan's (1977:395) verbal definitions, as well as Sampson's (1970:25) mathematical definition, suggest the same.

Upward definition. According to Lamb (1966:10), b or c goes to a . Lockwood (1972:34-36), Johannesson (1976:102)

and Christie (1977:4-13) do not give a behavioral upward definition, but if the node is assumed to behave like the upward unordered OR in encoding, it must be concluded that we are again dealing with an XOR. Sullivan's (1977:395) and Sampson's (1970:25) definitions confirm this conclusion.

We arrive thus at the following input-output relations:

Downward definition. An input at *a* will produce an output either at *b* or *c*, but not at both *b* and *c*.

Upward definition. An input at *b* or *c*, but not at both *b* and *c*, will produce an output at *a*.

2. Ordered OR—Precedence disjunction

Downward definition. In Lamb's (1966:10) definition, *a* goes to *b* if possible, or else to *c*. He explains, "that which comes first takes priority over the second if both are possible." Lockwood (1972:48-49) points out that the node handles conditioned choice. He stresses that the conditions governing the choice must be explicitly provided for.⁴ Christie (1977:5-7) and Sullivan (1977:397-398) agree with this view. Johannesson (1976:103) actually incorporates the conditioning factor in his diagram.

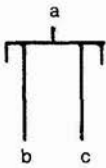


FIG. B.

Upward definition. According to Lamb (1966:10), *b* or *c* goes to *a*. Neither Lockwood nor Johannesson offers a behavioral definition. Sullivan (1977:398) states explicitly that the node behaves like the unordered OR. Christie's downward definition of the upward ordered OR points to the same conclusion: either *a* or *b* is realized as *a*.

This leads to the following definitions:

Downward definition. Under condition *a_y*, an input at *a_x* will produce an output at *b*. In the absence of *a_y*, an input at *a_x* will produce an output at *c*.

Upward definition. If there is an input at either *b* or *c* there will be an output at *a_x*.

3. Unordered AND—Conjunction

Downward definition. Lamb (1966:10), Johannesson (1976:100) and Christie (1977:5-7) offer the following definition: an impulse at *a* leads to the simultaneous output of impulses at *b* and *c*. Lockwood (1972:31) and Sullivan (1977:396) do not describe the outputs as necessarily simultaneous, but as in no specified (discernible) order. The point seems to be that the order is irrelevant.



FIG. C.

Upward definition. According to Lamb (1966:10) *b* and *c* together go to *a*. Johannesson (1976:100) stipulates that the inputs at *b* and *c* must be simultaneous. Lockwood, Sullivan, and Christie do not have a definition, but their

downward definition of the upward unordered AND suggests that the order of inputs at *b* and *c* is irrelevant (Lockwood 1972:31; Sullivan 1977:396; Christie 1977:5). The relation between inputs and outputs may, therefore, be described as follows:

Downward definition. If there is an input at *a* there will be an output at both *b* and *c*.

Upward definition. If there is an input at both *b* and *c* there will be an output at *a*.

4. Ordered AND—Concatenation

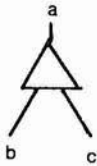


FIG. D.

Downward definition. In Lamb's (1966:10) definition, *a* goes to *b* and then (afterwards) to *c*. Lockwood's (1972:33), Johannesson's (1976:100), Sullivan's (1977:396-397), and Christie's (1977:5-7) definitions, although phrased differently, all suggest a temporal sequence of the outputs: first there is an output at *b*, and then there is an output at *c*.

Upward definition. According to Lamb (1966:10), *b*, and after it *c*, goes to *a*. Lockwood, Johannesson, and Christie do not give a definition. Their downward definitions of the upward ordered AND indicate that if *b* is followed by *c*, they are together realized as *a*. This is confirmed by Sullivan (1977:397).

We arrive, then, at the following definitions:

Downward definition. If there is an input at *a*, there will be an output at *b* at time *t*, followed by one at *c* at time *t+x*.

Upward definition. If there is an input at *b* at time *t*, followed by one at *c* at time *t+x*, there will be an output at *a*.

5. Conclusion

The examination of the verbal definitions of the four basic nodes reveals certain differences in the phrasing, a few inconsistencies in the definitional method, and some omissions in the upward definitions. But there is enough similarity to make possible an input-output definition if we assume the upward definition of downward nodes to be identical with the downward definition of upward nodes. As regards precedence disjunction, scholars now state clearly that the factor conditioning the choice of *b* must be made explicit. But there seems to be some disagreement about the simultaneity of the inputs (outputs) at the plural terminal of the conjunction node. There appears to be a tendency to consider temporal order as irrelevant. In concatenation, temporal order is clearly important. From this vantage point, one may recognize another fundamental distinction between the nodes discussed. On the one hand we have the combinatorial node type (disjunction, precedence disjunction, conjunction) for which timing is inconsequential, and on the other we have the sequential

type, represented only by concatenation, for which time is essential. Since SG nets are made up of both types of nodes, they must be classed as mixed combinatorial and sequential networks.

III. NODES INTERPRETED AS INPUT-OUTPUT SYSTEMS

Figure 1.1 is a general representation of the typical SG node. It has a singular side (one terminal) and a plural side (two terminals). The node is bidirectional. In encoding, inputs are from the top (figure 1.2), in decoding from the bottom (figure 1.3). This is why behavioral definitions divide a bidirectional node into two unidirectional nodes. For either of these nodes, the output is defined as a function of the input(s). In SG, inputs and outputs can assume two values only. The value is 1 when there is an input or output, and 0 when there is not.

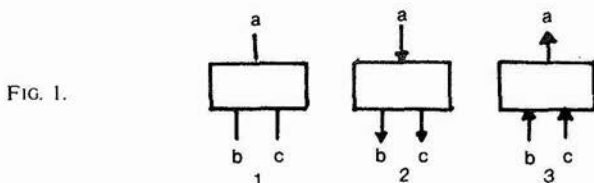


FIG. 1.

1. One input terminal—two output terminals

If the input is from the singular side, we have one independent input variable and two dependent output variables. In this case figure 2.1-4 and table 1 provide an exhaustive account of all possible relations between input and outputs.

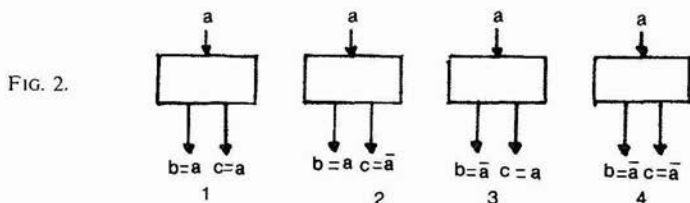


FIG. 2.

TABLE 1

	Input <i>a</i>	Outputs <i>b</i> <i>c</i>	
Figure 2.1	0	0	0
	1	1	1
Figure 2.2	0	0	1
	1	1	0
Figure 2.3	0	1	0
	1	0	1
Figure 2.4	0	1	1
	1	0	0

Outputs may have the same value as the input (figure 2.1), they may have the opposite value (figure 2.4), and last, one output may have the same, the other the opposite value of the input (figures 2.2, 2.3). As all output values depend solely on the input value of a , they can be written as functions of a (see the formulae under figures 2.1-4, where a in the output stands for the same, and \bar{a} for the opposite value of the input).

2. Two input terminals—one output terminal

If the input is from the plural side of the node and the output at the singular side, output is a function of two independent variables. The sixteen possible functions are represented in table 2.

TABLE 2

Inputs		Outputs															
b	c	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Not all functions are equally important: a_1 and a_{16} are evidently independent of the input variables, and $a_{13}(=b)$, $a_4(=\bar{b})$, $a_{11}(=c)$ and $a_6(=\bar{c})$ depend on one variable only. The remaining functions depend on both variables.

IV. UPWARD DEFINITIONS OF DISJUNCTION, CONJUNCTION, AND INTERJUNCTION

Table 2 lists all possible functions of two input variables. Consequently the upward definitions of all combinatorial SG nodes discussed here must match one of these functions.

It is easily recognized that a_7 corresponds to the upward definition of both the ordered and the unordered OR (figure 3, table 3). In logic this function is commonly called disjunction. The OR is often referred to as XOR (exclusive OR) in contradistinction to adjunction (inclusive OR). The symbol $\underline{\vee}$ will be used as a representation of XOR. Thus the upward definition of the node depicted in figure 3 reads $a : b \underline{\vee} c$.

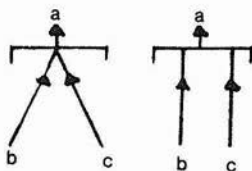


FIG. 3.

TABLE 3

b	c	a_7
0	0	0
0	1	1
1	0	1
1	1	0



FIG. 4.

TABLE 4

<i>b</i>	<i>c</i>	<i>a₉</i>
0	0	0
0	1	0
1	0	0
1	1	1

The upward definition of the unordered AND corresponds to function a_9 (figure 4, table 4). This function will be represented by the symbol \wedge : AND defined as $a : b \wedge c$.

We have seen that the three upward definitions of the nodes considered so far can be expressed by two functions. If we assume that these are not sufficient to express all input-output relations necessary in a relational network model of natural language, how can any other functions be represented? The answer is that AND and XOR may be combined to produce some, but not all, of the functions in table 2. As the nodes are nothing but graphic representations of AND and XOR, any logical representation employing \vee and \wedge must also be representable by a corresponding network of AND and XOR nodes.

By way of example, I shall present a definition of interjunction (see Schreyer 1977:142) using AND and XOR. It has the following upward definition: there is an output at a except when there is no input at both b and c (figure 5).



FIG. 5.

TABLE 5

<i>b</i>	<i>c</i>	<i>a₁₅</i>
0	0	0
0	1	1
1	0	1
1	1	1

Table 5 shows that the upward definition of interjunction corresponds to adjunction (inclusive OR) in logic. This is the designation of function a_{15} , which is often represented by the symbol \vee : Adjunction definition $a : b \vee c$. Adjunction may be represented by the formula $(b \vee c) \vee (b \wedge c)$, as table 6 will prove.

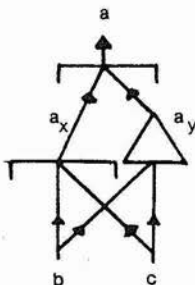


FIG. 6.

TABLE 6

1	2	3	4	5
<i>b</i>	<i>c</i>	$(b \vee c)$	\vee	$(b \wedge c)$
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	0	1	1

Columns 1 and 2 give the input values for the independent variables b and c , 3 and 5 the output values of the two bracketed expressions, and 4 the output values for the combination of these values. It will be seen that column 5 is identical with a_{15} . The formula can easily be converted into the network of figure 6, which also represents adjunction as a combination of AND and XOR: input at b will produce output at a along a_x , input at c will do the same. Input at both b and c will produce an output at a along a_y . Zero input will not produce any output. Because of the logical properties of AND and XOR, different formulae and different nets may also be expressions of the inclusive OR. The structural principle is, however, always the same: the output of one node provides the input of another. Other functions expressible by a combination of AND and XOR are:

$$a_5 : (b \wedge c) \vee b; (b \vee c) \wedge b$$

$$a_3 : (b \wedge c) \vee c; (b \vee c) \wedge c.$$

The number of functions to be expressed by combining AND and XOR is, however, limited: functions a_3 , a_5 , a_7 , a_9 , and a_{15} exhaust all combinatorial possibilities. The question whether they are sufficient to build a network model of natural language is an empirical one and can only be answered with reference to the data.

V. DIGRESSION: BOOLEAN ALGEBRA AND THE CHOICE OF PRIMARY FUNCTIONS

Two of the functions listed in table 2 hold a special position: a_2 and a_8 . Each by itself is able to express all other functions in the table. Boolean algebra, the most elaborate system to express all functions, employs AND (conjunction), OR (adjunction), and negation. It is widely used in computer science, and is isomorphic with switching logic, a network representation of the same functions. Boolean algebra is therefore a genuine alternative to a system employing AND and XOR. It may even be considered superior, as it is capable of representing all possible functions of two variables.

It is important to recognize that the logic underlying SG is not the only one possible. A different choice of primary functions can be used to construct formulae and networks that have the same input-output relations as SG networks, although their internal structure is different.⁵ For the same reason Boolean logic may be used to define combinatory SG nodes. The next sections will be devoted to a definition of these nodes using terms of Boolean algebra.

VI. THE DOWNWARD DEFINITIONS OF CONJUNCTION, SIMPLE DISJUNCTION, AND PRECEDENCE DISJUNCTION

In the downward definitions of section II, the nodes under discussion are regarded as consisting of one input and two output terminals. Thus they ought

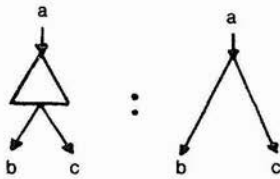


FIG. 7.

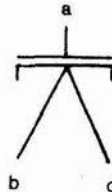


FIG. 8.

to correspond to one of the general types analyzed in section III.1. The downward definition of the unordered AND clearly attributes the node to the type represented in figure 1.1: input at *a* produces outputs at *b* and *c*. This will be represented by a simple branching of lines (figure 7).

For the downward definition of the unordered OR, none of the figures 1.1-4 seems to be an obvious choice. In the definition it is stipulated that for an input at *a* there must be an output at either *b* or *c*, but not at both. The definition does not specify where the output will actually be. Sullivan (1977:395-396) recognizes that, without any additional information, the output is indeterminate. But his solution of simply changing the (logical) definition avoids the problem and, furthermore, creates a discrepancy between his verbal and his logical definition. The solution suggested by Reich (Schreyer 1977:139-141) seems to me the better approach. Reich proposes the introduction of a new node, the so-called random choice disjunction, to take care of the possibility of free variation among the outputs (figure 8). It is evident that this node will produce new definitional problems. As we have enough of these already I will not discuss the node here, although I can see no other way of integrating free variation into the network. For cases of conditioned variation among outputs, Reich suggests the use of additional ANDs (figures 9.1-3) which are to provide for the conditioning factors. This solution is based on the assumption that the presence or absence of an output at the *b* or *c* terminal of a node depends on more than one input variable. Nevertheless Reich's solution does not solve all the problems. In figure 9.1, if there is an input at *a_y* and *a_x*, there will be not only the desired output at *b* but also the undesired output at *c*. Or, in figure 9.3, if there is an input at all pertinent terminals, how can the node decide whether to put out *b* or *c*?

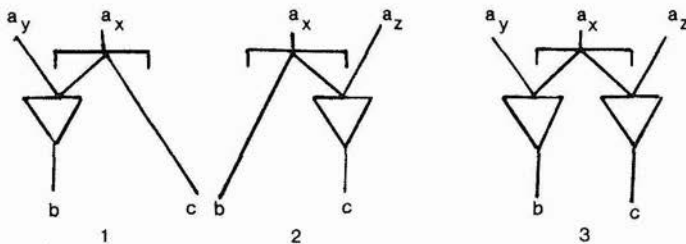


FIG. 9.

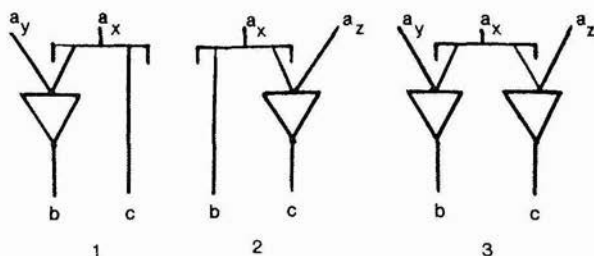


FIG. 10.

The network as it stands is incapable of handling these difficulties. One obvious solution consists in specifying which terminal takes precedence over the other in such cases of indeterminacy.⁶ This means that in cases of conditioned choice, the simple disjunction node would have to be replaced by the precedence disjunction node (figures 10.1-3). To give a logical definition of the networks of figure 10, it is useful to set down the conditions governing the choice of the b and the c terminals in an input-output table (truth table). Table 7 corresponds to figure 10.1, table 8 to figure 10.2, and table 9 to figure 10.3. A look at tables 7 and 8 shows that the output columns are the same for both, with the only difference that b and c are interchanged. For our purpose it will therefore be sufficient to construct the logical formula for one table. If there is only a small number of input variables this can be done by using Veitch diagrams or by converting the table into its so-called adjunctive normal form, which can then be simplified by applying the laws of Boolean algebra. As the networks are very simple I shall use the latter method.

In table 7 the output values for b show that there is an output only if there is an input at both a_x and a_y . This corresponds to function a_9 of table 2: $a_x \wedge a_y$. One can obtain the same formula by joining the two variables responsible for an output of 1 at the b terminal by an AND: $a_x \wedge a_y$. For the c terminal the output is 1 only if a_y is negative (0) and a_x positive (1). In this case then, we arrive at the formula $a_x \wedge \bar{a}_y$.

a_y	a_x	b	c
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

a_z	a_x	b	c
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

a_x	a_y	a_z	b	c
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

The foregoing discussion of precedence disjunction leads to the following downward definition:

$$b : a_x \wedge a_y$$

$$c : a_x \wedge \bar{a}_y$$

This analysis of precedence disjunction fulfills all the specifications of the input-output definition. There will be no output unless there is an input at a_x . If there is an additional input at a_y , output will be at b , otherwise at c . Table 7 makes plain that the choice of the output terminal depends solely on the presence or absence of an input at a_y . The formulae for b and c show that a_y furnishes the positive condition for b and the negative condition for c .

We have seen that precedence disjunction admits of a downward definition if the conditioning factor is included. We have also seen that it may be represented by a combination of AND and negation. There is, however, no direct conversion of the Boolean formula into an SG network, since SG has no negator element. But I cannot see any objection to its being introduced (at least for the sake of argument). If we represent negation by a small circle at the beginning or end of a line, the network of figure 11 would be a direct conversion of the Boolean definition of precedence disjunction. The net should be interpreted as follows: Input at a_x will go down the two branches to AND 1 and AND 2. Input at a_y will also go down to AND 1 and AND 2, but the signal down the AND 2 branch is negated (or inverted) before it reaches the node: the line will have zero output. Thus only AND 1 has the correct input, and the output will be at b . If there is an input at a_x and not at a_y , only AND 2 has the correct input: the negative input signal from a_y will be inverted. There are, then, two positive signals at AND 2 that will produce an output at c .

Note that figure 11 employs only those input-output relations allowed in section III.1: a_x is equivalent to figure 1.1, a_y to figure 1.2. The fact that outputs are uniquely determined by the input is represented by simple branching of lines. A separate node is not needed.

Larger networks can also be defined and simplified in the same fashion. Table 9, for instance, lists the input-output relations of the net of figure 10.3

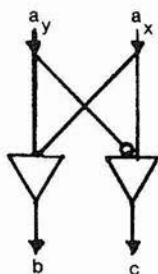


FIG. 11.

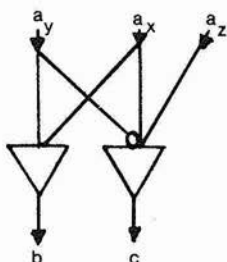


FIG. 12.

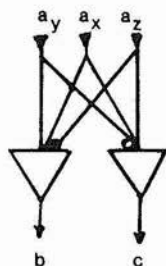


FIG. 13.

(3 input terminals, 2 output terminals). From this we derive the adjunctive normal form for b :

$$b: (a_x \wedge a_y \wedge \bar{a}_z) \vee (a_x \wedge a_y \wedge a_z).$$

By applying the laws of Boolean algebra, we may simplify this formula, and we obtain: $b: a_x \wedge a_y$. For c no simplification of the adjunctive normal form is necessary: $c: a_x \wedge \bar{a}_y \wedge a_z$. Figure 12 is a graphic representation of this definition. The problem of simple disjunction discussed at the beginning of this

TABLE 10

a_x	a_y	a_z	b	c
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	?	?

section (figure 9.3) becomes much clearer when we try to establish its input-output table (table 10). There is no difficulty until we reach the last row; then the question marks have to be eliminated. They cannot both be replaced by 1. This would contradict the verbal definition of the node. If we replace one question mark by 1, the other by 0, we have a case of precedence disjunction as in table 9. The only alternative left is to substitute a 0 for each question mark. There would be no output if there was an input at both a_y and a_z . The emended table 10 converts to the following definitions:

$$b: a_x \wedge a_y \wedge \bar{a}_z$$

$$c: a_x \wedge \bar{a}_y \wedge a_z$$

These are represented in the diagram of figure 13.

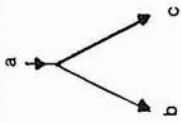

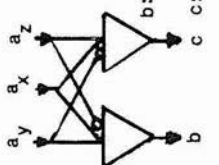
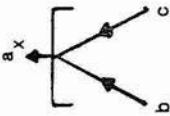
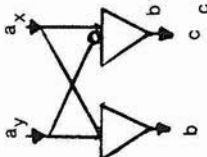
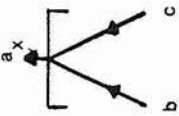
VII. SUMMARY: THE DEFINITIONS OF COMBINATORY SG NODES

The results of the foregoing analysis of conjunction, disjunction, and precedence disjunction are collected in table 11.

The downward definitions employ only the functions allowed in section III. Choice between particular branches is made by the conditioning factors a_y or a_z , which must be present in the logical definition and its graphic representation. If we assume that the output is completely determined by the inputs (i.e., if we ignore the possibility of random selection), no branch of an XOR must be without the selection mechanism provided by the upward AND and the negator. The upward definitions employ the familiar symbols with arrows indicating the direction. These nodes suffice as long as it is stipulated that there be no upward outputs along the wires leading to the conditioning factors.

In accordance with the verbal definitions of the ordered and unordered XORs, there is no upward distinction between them. It is, however, obvious that the node is not a direct representation of the definition formula: in Boolean algebra XOR is not a primary function and has, therefore, a complex structure. It must be expressed as a combination of AND, OR, and negator. This can also be done in SG if interjunction is used as a primary function. As I mentioned, inter-

TABLE 11

	DOWNWARD DEFINITION	UPWARD DEFINITION
CONJUNCTION	 <p>b: a c: a</p>	 <p>a: $b \wedge c$</p>
DISJUNCTION	 <p>b: $a_x \wedge a_y \wedge \bar{a}_z$ c: $a_x \wedge \bar{a}_y \wedge a_z$</p>	 <p>a: $(b \wedge \bar{c}) \vee (\bar{b} \wedge c)$</p>
PRECEDENCE DISJUNCTION	 <p>b: $a_x \wedge a_y$ c: $a_x \wedge \bar{a}_y$</p>	 <p>a: $(b \wedge \bar{c}) \vee (\bar{b} \wedge c)$</p>

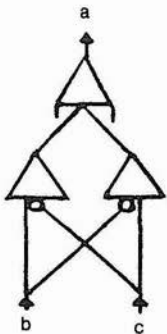


FIG. 14.

junction (figure 5) is identical with Boolean adjunction and can therefore be employed for directly converting the XOR formula into a network (figure 14).

VIII. THE DEFINITION OF CONCATENATION (Ordered AND)

The logical definition of the ordered AND poses very specific problems. The following discussion should, therefore, be considered as an attempt to define the problem, rather than an attempt to define the node.

The definitions of the unordered AND and of both the ordered and the unordered OR have no specific reference to time. This is why they were considered as purely combinatory nodes. The ordered AND, on the other hand, is clearly sequential. Without this node the linearization of non-linear structures would not be possible. In the downward definition an output at *c* can occur only *after* there has been an output at *b*. In the upward definition an output at *a* is possible only if the input at *c* occurs *after* the input at *b*. Any logical definition that ignores these sequential requirements is therefore bound to be wrong. Christie's (1978:248) criticism of Sullivan's (1977:396-397) definition using propositional calculus makes precisely this point:

The predicate [sic] calculus is by its very nature an atemporal system. As such it is inherently incapable of describing the difference between simultaneous and successive occurrences of conjuncts. Since natural language operates in time, with differences between simultaneous and successive occurrences, it is inadequate as a description of the operation of a linguistic system.

I do not share Christie's belief that Sullivan's logical definitions were intended as a mere description of the competence of the system. Sullivan (1977:397, 299, 403 n. 5) certainly makes an effort to show how his definitions of concatenation can produce the desired sequential outputs. For this purpose he introduces specific performance considerations, which are, however, extraneous to the laws of predicate calculus (see Schreyer 1979). Furthermore, it should be recognized that a model of competence completely divorced from performance is alien to the aims of SG. The aim of stratificational linguistics has always been the construction of "a competence model that can serve as a basis of a model of performance" (Christie 1978:248). Thus, a network model that makes no explicit provision for temporal sequence must be inadequate.

The nodes discussed so far are, by definition, atemporal and can therefore be defined both in Fitch's logic (Fitch 1952) and in Boolean algebra; the ordered AND cannot. This does not mean that propositional calculus is a bad tool for defining SG nodes; it only means that it is not rich enough to define all nodes. It is certain that at least for concatenation some kind of timing device must be incorporated into the network. The following definitions are an attempt to include the time factor:

Upward definition. $a : b(t) \wedge \bar{b}(t+1 \dots x) \wedge \bar{c}(t) \wedge c(t+1 \dots x)$

Downward definition. $b(t) : a$
 $c(t+1 \dots x) : a$

The ordered AND must have a more complicated internal structure than the combinatorial nodes, as there must be a time delay between the outputs (and inputs) at the b and the c terminals of the node. If one assumes that c must follow b with a delay of one time unit, it will be sufficient to insert a delay node (delay time = 1 unit) in the appropriate places (figure 15). The delay element would insure the sequentiality of downward output and of upward input. Simple as it may be, this solution will probably not work, since it is hard to believe that inputs (outputs) will always follow each other at intervals of one time unit, whatever its definition. Delay time may be different for different concatenation nodes in the network, but also for the same node, depending on which downward path the signals from the b and c terminals are taking.

A second way of dealing with sequentiality is by internal feedback. In the downward processing of a concatenation node N , an input at a will produce an output at b . The node will then change from an initial state 0 to state 1. In this state it will not accept any input from a . When all outputs along b have been completed, a feedback signal to N will trigger the outputs down c . N now changes to state 2. When all outputs down path c have been completed, a second feedback signal comes up line c . This causes N to put out a feedback signal at a . At the same time the node is switched back to its initial state 0. The node is now ready to process new input from a (figure 16). This is basically Reich's (1969:835) analysis of the ordered AND. He regards the node as a finite

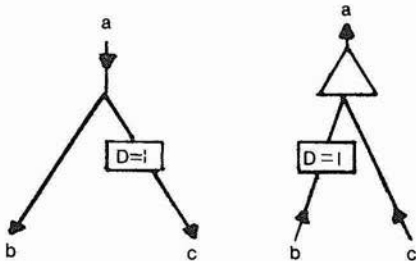


FIG. 15.

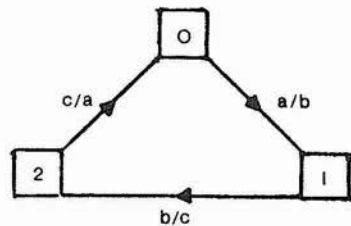


FIG. 16.

state device with three states.⁷ Clearly, such a node must “remember” the state it is in: its micronotation must be given a memory.⁸ This is the solution adopted in Christie’s micronotation (1977:11-12; 1978:252) of which figure 17 is a replica. The circles marked 1 are memory nodes. They can be switched on by a signal along the lines marked “plus”; they can be switched off by a signal along the lines marked “minus.”

In downward processing input at *a* leads to output at *b* and excites the bottom memory node. Feedback from *b* produces an output down *c*, switches off the bottom memory node and excites the top one. Feedback from *c* causes feedback at *a* and switches off the top memory node.

In upward processing, *N* must be able to distinguish between the sequence *bc* (output at *a* = 1) and *cb* (output at *a* = 0); it must not process an input at *c* unless it is preceded by an input at *b*. Figure 17 fulfills this requirement on the assumption that the feedback lines are also used in decoding. An input at *b*, and only at *b*, excites the top memory node, without which no upward output is possible. Subsequent input at *c* then causes an output at *a*.

By introducing memory nodes, Christie’s microanalysis of concatenation provides for the sequencing of inputs and outputs. These nodes keep alive signals until they are needed at a later time. This analysis is compatible with the verbal definition of the ordered AND, and it avoids the difficulties a definition in terms of propositional calculus runs into.⁹

It should be pointed out that Christie’s micronotation bears a strong resemblance to Boolean algebra. The cells with a threshold of 2 are equivalent to conjunction, those with a threshold of 1 are equivalent to adjunction, and the inhibitory terminals are related to negation (see George 1962:119-123;

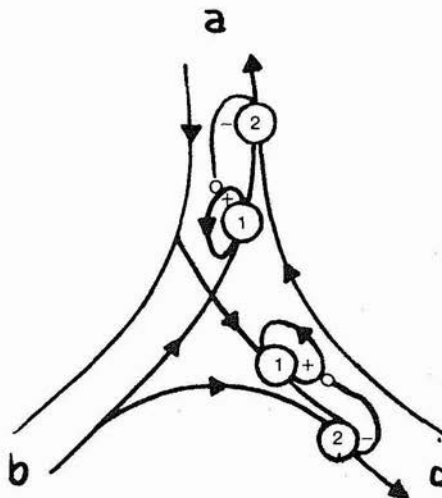


FIG. 17.

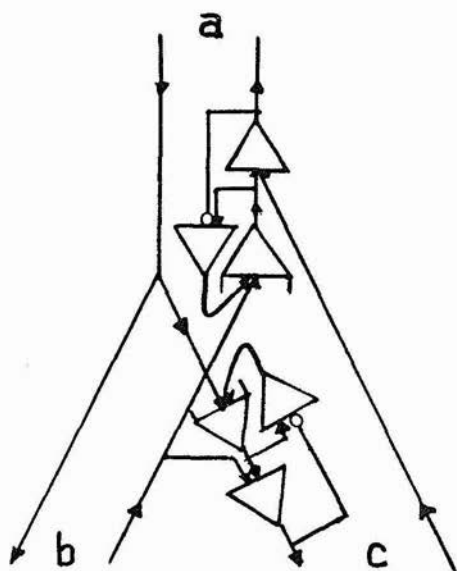


FIG. 18.

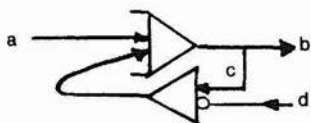


FIG. 19.

Klaus 1966:129-137). The only real difference seems to be the introduction of the memory cell, which, once activated, keeps firing itself. If we add the memory element to a network notation that employs AND (conjunction), OR (adjunction, interjunction), and negation, we obtain the network of figure 18, which is equivalent to that of figure 17.

In this notation the memory node has the structure depicted in figure 19: input at *a* produces outputs along lines *b* and *c*. The *c* output loops back by way of the AND node and provides the input of the OR. Thus the OR will produce outputs until an input from *d* interrupts the process.

IX. SUMMARY

This study presents a necessarily preliminary analysis of SG nodes as input-output systems. It offers a more rigorous definition of the combinatorial nodes by establishing input-output tables that allow a systematic conversion into logical expression in Boolean algebra. To set up these tables, a bidirectional SG node is divided into two unidirectional nodes, and separate definitions are suggested for the upward and downward processing of signals. Outputs are regarded and defined as functions of the inputs. These definitions are valid only as long as sequentiality does not come into play. The ordered AND is clearly sequential, a fact that can be accounted for only by introducing a timing element into its algebraic and network representation.

There are many problems that the present study only touches upon. Further detailed analysis of the performative aspects of SG networks will no doubt

bring improvements in the construction and simplification of the sequential networks SG is working with.

NOTES

1. In some cases it may be expedient to replace a well-defined constellation of basic nodes by one new, secondary node, particularly if it makes the network easier to read. But it is imperative that the equivalence between the secondary node and the network replaced by it be expressly stated; for unless the higher complexity of a secondary node is taken into consideration in calculating the structural complexity of larger networks, one will arrive at the wrong result, and the algorithms for simplifying networks may no longer work (see Schreyer 1977:131-132, 142-143).

2. The question as to which of the remaining nodes are basic and which are derived is still open and will not concern us here. It is hoped that a clearer understanding of the basic nodes will shed some light on that problem.

3. Gleason also suggested a definition using inputs and outputs: "concatenation, AND-OR, and disjunction can all be thought of as the same function of two variables—the minimum number of successful downward outputs, and the maximum number of same. In concatenation the minimum and the maximum are both 2. In AND-OR the minimum is 1 and the maximum is 2. In disjunction the minimum and the maximum are both 1" (Reich 1973b:115, n. 8). Sampson's mathematical definitions seem to implement this suggestion for some of the nodes.

4. Lockwood's (1972:63) enabler was specifically introduced to incorporate conditioning environment into the relational network.

5. If the same input-output relation can be represented by logical expressions and networks employing different primary functions, a new question arises: which functions should be chosen for a model of the linguistic code and which criteria determine that choice?

6. Other solutions to the problem of figure 9.3: one could introduce random choice, or one could ensure further up in the network that a_1 and a_2 cannot both occur. For yet another solution, see page 176.

7. For a more sophisticated analysis, which also accounts for looping, see Reich (1969) and Christie (1976).

8. Furthermore, the network must have a device that triggers the feedback signals. In other words, it must be specified at which point in the network the downward output counts as completed.

9. Nevertheless, more research into the internal structure of the concatenation node will be necessary. How, for instance, does the network of figure 17 react to input at a while the node is still processing the previous input?

REFERENCES CITED

- Christie, William M. 1976. Evidence concerning limits on central embeddings in English. *Forum Linguisticum* 1:25-37.
- . 1977. *A Stratificational View of Linguistic Change*. Lake Bluff, IL: Jupiter Press.
- . 1978. Atemporal logic, temporal logic, and natural language. *Forum Linguisticum* 2:247-253.
- Fitch, Frederic B. 1952. *Symbolic Logic—An Introduction*. New York: The Ronald Press Company.

- George, Frank H. 1962. *The Brain as a Computer*. Oxford: Pergamon Press.
- Johannesson, Nils-Lennart. 1976. *The English Modal Auxiliaries: A Stratificational Account*. Stockholm: Almqvist and Wiksell International.
- Klaus, Georg. 1966. *Moderne Logik*. 3rd edition. Berlin: VEB Deutscher Verlag der Wissenschaften.
- Lamb, Sydney M. 1966. *Outline of Stratificational Grammar*. Washington, D.C.: Georgetown University Press.
- . 1974. Sydney M. Lamb. In Herman Parret, ed., *Discussing Language*, pp. 179-219. The Hague: Mouton.
- Lockwood, David G. 1972. *Introduction to Stratificational Linguistics*. New York: Harcourt Brace Jovanovich.
- Mey, Jacob. 1972. A note on formalization in the sciences, with special reference to linguistics. *Norsk Tidsskrift for Sprogvidenskap* 26:111-119.
- Reich, Peter A. 1969. The finiteness of natural language. *Language* 45:831-843.
- . 1973a. Competence, performance, and relational networks. In Adam Makkai and David G. Lockwood, eds., *Readings in Stratificational Linguistics*, pp. 84-91. University: University of Alabama Press.
- . 1973b. Symbols, relations, and structural complexity. In Adam Makkai and David G. Lockwood, eds., *Readings in Stratificational Linguistics*, pp. 92-115. University: University of Alabama Press.
- Sampson, Geoffrey. 1970. *Stratificational Grammar: A Definition and an Example*. The Hague: Mouton.
- Schreyer, Rüdiger. 1977. On structural complexity in relational networks. *Canadian Journal of Linguistics* 22:125-143.
- . 1979. The Definition of Nodes in Stratificational Linguistics. To appear in H. Izzo and W. McCormack, eds., *The Sixth LACUS Forum 1979*. Columbia: Hornbeam Press.
- Stewart, Ann H. 1976. *Graphic Representation of Models in Linguistic Theory*. Bloomington: Indiana University Press.
- Sullivan, William J. 1977. Toward a logical definition of linguistic theory. In Robert J. DiPietro and Edward L. Blansitt, eds., *The Third LACUS Forum 1976*, pp. 393-404. Columbia, S.C.: Hornbeam Press.