MEMORANDUM
RM-4087-PR
JULY 1964

DEPENDENCY THEORY: A FORMALISM AND SOME OBSERVATIONS

David G. Hays

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-4087-PR
JULY 1964

DEPENDENCY THEORY: A FORMALISM AND SOME OBSERVATIONS

David G. Hays

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-700 monitored by the Directorate of Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

PREFACE

Linguistic theory is a topic of considerable interest just now, partly because new applications such as machine translation and information retrieval demand more precise and comprehensive theory than ever before, and partly because recent work has greatly enlarged both the variety and the formal precision of theories that are available.

One relatively new branch of linguistics is called dependency theory, with The RAND Corporation one of the principal centers of its development. The present Memorandum surveys dependency theory primarily from the viewpoint of the professional linguist. This is the first general treatment of the subject, in which results were collected from numerous sources, and the motivation as well as the structure of the theory are examined in some detail.

SUMMARY

Dependency grammars characterize the class of context-free languages, assigning to each sentence of a characterized language a tree structure with minimal syntactic units at the nodes. Both production and recognition procedures are given. Either transformational or stratified linguistic systems can be constructed on the basis of dependency theory; more attention is given to the latter possibility. Semantic and psychological considerations are cited as motivating specific features of the theory, but they are no more necessary as justifications for this theory than for others.

CONTENTS

DEPENDENCY THEORY: A FORMALISM AND SOME OBSERVATIONS

1. Introduction

In the general theory of linguistics, dependency theory competes for the same role as immediate-constituent (IC) theory: Each formalizes a view of tactic relations among elementary units in utterances. In each theory, a finite apparatus is taken to characterize (or generate) a set of utterances that may be infinite. In addition, each theory specifies a structure, or perhaps alternative structures, for each utterance it generates.

The essential insight formalized by IC theory was stated clearly by Harris in 'From morpheme to utterance'.[1] There are composite expressions (phrases) that have the same distribution as minimal syntactic units; for example, there are noun phrases that have the same distribution as some nouns. This insight leads to a principle of empirical research, since the integrity of a phrase in an utterance is confirmed by the existence of a minimal unit substitutable for it. This principle is not sufficient, for two reasons. In the first place, some phrases have distributions that differ from all those of minimal units. Secondly, there are indefinitely many varieties of mutually substitutable stretches of utterances, and therefore the principle does not lead to discovery of a finite

characterization. Nevertheless, the principle is significant for practical linguistics. The same insight leads also to a formal calculus of phrase types or classes, namely Chomsky's theory of context-free phrase-structure grammars.[2] The structures assigned by grammars of this type will be called P-markers. The importance of IC theory and the use of P-markers in linguistics is entirely independent of the empirical principle of substitution; characterization theories and research methods (so-called discovery procedures) must be studied separately.

The insight formalized by dependency theory is that particular occurrences of minimal units are directly related to one another. Occurrence of an adjective before a noun, for example, is more closely related to the noun occurrence than to anything else. In describing the distribution of a class of adjectives, therefore, one naturally refers to the classes of nouns with which the adjectives occur. Similarly, an occurrence of a preposition is generally most directly related to two other occurrences, its object and the unit modified by the prepositional phrase. Prepositions are therefore regarded as connectors, and their distributions are most easily stated by reference to classes of objects and classes of governing or modified units. This kind of observation

about language is sometimes expressed in terms of the valences of units for one another.[3]

Some of Hjelmslev's empirical principles are closely related to the insight behind dependency theory,[4] but empirical dependency in his sense cannot be identified with abstract dependency in the sense of the present paper, since he explicitly differentiates dependencies from other kinds of relations, whereas the present theory intends to be complete, i.e. to account for all relations among units of utterances.

Dependency theory is actually a characterization theory, not necessarily associable with any empirical method or principle. It is a theory of grammars, with abstract mechanisms for characterizing sets of utterances and for assigning to them certain structural descriptions, which will be called D-trees.

The present paper is an introductory account of the theory, and a survey of its current state of development. Section 2 presents a formalism for the theory, identifying the components of any dependency grammar. Next, a generative procedure is described (Sec. 3), and some decision procedures (Sec. 4). These are alternative and equivalent explications of the relation between finite grammars and infinite sets of utterances. IC theory and dependency theory are compared in Sec. 5, which also gives

some attention to their ability to assign similar structures to utterances in a given set. Section 7 discusses extensions of dependency theory by addition of transformations or by embedding in a multi-stratal system. Finally, in Sec. 8, the problem of empirical application is examined.

2. Formal dependency theory

In the more familiar IC grammars, two alphabets, terminal and nonterminal, are used.[5] One interpretation makes the terminal alphabet a lexicon or morpheme list. The nonterminal alphabet is a set of names or symbols for phrase types, perhaps including phrases of a single element.

A dependency grammar also uses two alphabets, and one is terminal in exactly the same sense: the utterances to be characterized are strings of elements of the terminal alphabet. The nonterminal, auxiliary alphabet is a set of names or symbols for types of occurrences of terminal-alphabet characters. Each element of the terminal alphabet is a minimal syntactic unit—for example, a morph, morpheme, or word. But the theory allows minimal syntactic units to be syntactically ambiguous; a given morpheme can act as a verb stem in one utterance, as a noun stem in another. An auxiliary character is, in this sense, unambiguous; a terminal character can correspond to any number of

different auxiliaries, each with fixed syntactic properties. On the other hand, an auxiliary character can correspond to a long list of different terminals, namely, all those that can occur with its syntactic properties. Bar-Hillel has called the many-many correspondence between auxiliary and terminal elements the assignment function of a grammatical system.[6]

A dependency rule is a statement about the valence of one kind of syntactic unit. It consists of one auxiliary character as governing element, and any finite number of auxiliary characters as dependent elements; terminal characters never appear in dependency rules. A convenient notation (due to Gaifman[7]) is

$$X_i(X_{j_1}, X_{j_2}, \ldots, *, \ldots, X_{j_n}). \qquad (1)$$

Here X_i is the governing element; $X_{j_1}, \ldots, X_{j_n}$ are dependent elements, n in number. The rule implicitly orders all of the elements in it, by the order of the elements within the parentheses. The asterisk marks the position of the governor among its dependents. The valence of X_i given by this rule is just the set of dependent elements stated. The order given the elements by the rule is the relative order of corresponding terminal characters in utterances, but other terminals can be interspersed among them.

A dependency rule can be regarded as a statement that an occurrence of a syntactic unit corresponding to X_i can govern simultaneously occurrences of units corresponding to X_{j_1}, to X_{j_2}, and so on up to X_{j_n}. As an example, take the hypothetical English rule

$$V_\alpha(N_{pl}, *, N, D_\beta),$$

where V_α is a class of verb morphemes, N_{pl} a class of plural nouns, N a noun class, and D_β a class of adverbs—say, of manner. This rule could be used in connection with utterances like children eat candy neatly.

The last example illustrates the nearest approach in dependency theory to classification of phrases. In general, minimal syntactic units are the elements of dependency calculations, but whenever necessary an auxiliary character like N_{pl} can be introduced. Interpreting the minimal units of the theory as morphemes, we must understand N_{pl} as corresponding to occurrences of noun-stem morphemes governing plural morphemes. In this fashion, a dependency grammar can take into account distant influences such as agreements between subject and predicate complement of a copulative verb.

Formally, a dependency grammar must contain two other kinds of rules. A rule of the form

$$X_i(*) \qquad (2)$$

states that a terminal corresponding to X_i can occur without dependents. Likewise, a rule

$$*(X_i) \qquad (3)$$

states that a unit corresponding to X_i can occur without being governed. Such a unit is the central or main element of the utterances in which it so occurs.

Thus a dependency grammar consists of a terminal alphabet, an auxiliary alphabet, an assignment function linking the two, and a set of rules including some of each of the three forms specified. The utterances characterized by the grammar are strings over the terminal alphabet. The assignment function matches terminal characters with syntactic types. A rule of form (3) marks the center of an utterance, rules of form (2) mark the outer limits, and rules of form (1) mark the valence, or dependency, relations among the units of the utterance. How such a grammar characterizes a set of utterances is described in Secs. 3 and 4.

3. Generative characterization

An utterance is a finite string of minimal syntactic units. A grammar characterizes a set of utterances; as

Chomsky rightly argues, a collection of statements about a language that does not characterize its utterances does not deserve to be called a grammar.[8] (Incidentally, this argument has the terminological consequence that no special descriptive term is needed for the class of grammars that do characterize sets of utterances, there being no grammars that do not.)

Classically, mathematical logic offers at least two approaches to the characterization of sets: generative functions and decision procedures. Postal[9] is mistaken in arguing that the former approach is logically more fundamental than the latter, and Chomsky[10] is almost equally wrong in asserting that the former are overwhelmingly predominant in the history of linguistics. Chomsky is correct, however, in differentiating between (a) the generation-decision distinction in characterization theory and (b) the production-recognition distinction in theories of behavior or, as Chomsky puts it, performance.[11] Although the distinctions are different, the difference is susceptible of overstatement; a practical parsing device for a class of sets of utterances is a fortiori a decision procedure for the class.

A class of grammars was defined in Sec. 2, but only up to the point of a mechanism for generation of utterances or a decision procedure. The first kind of

abstract mechanism, given a dependency grammar, generates an infinite list of strings: those of the language characterized by the grammar. A decision procedure, given the grammar, decides for each string over the terminal alphabet whether it does or does not belong to a certain set; that set is also said to be the language characterized by the grammar. Naturally, it is essential to prove, for a class of grammars, that the set generated by a certain procedure is the same as the set accepted by a given decision procedure whenever the same grammar is supplied to both.

In the present section, a generative procedure is described; Sec. 4 presents a decision procedure.

The generative procedure has two stages; the first stage is a sequence of steps, with all except the first carried out on the same plan. The first step is the choice of a central element. For this step, a rule of form (3) is chosen; it contains a single auxiliary element, X_i. The second step is to choose a rule of form (1) or (2) with the same X_i as governing element. If the rule chosen has the form $X_i(*)$, the first stage of the procedure is complete. Otherwise, the rule has dependent elements $X_{j_1}, X_{j_2}, \cdots, X_{j_n}$. Take the first and choose a rule with X_{j_1} as governing member. The new rule may specify dependent members; if it does, they are added to the

object being constructed. The construction continues until a rule has been chosen for each character introduced.

Certain information must be recorded each time a rule is chosen. A notation is the following. Let $*(X_i)$ be the first rule chosen, and $X_i(X_{j_1}, X_{j_2}, \ldots, *, \ldots, X_{j_n})$ the next. Let $X_{j_1}(*)$ be the rule chosen for X_{j_1}. Then, after the first choice, record

$$*(X_i).$$

After the second,

$$*(X_i(X_{j_1}, X_{j_2}, \ldots, *, \ldots, X_{j_n})).$$

After the third rule choice, record

$$*(X_i(X_{j_1}(*), X_{j_2}, \ldots, *, \ldots, X_{j_n})).$$

This expression shows clearly that rules have been chosen for X_i and X_{j_1}, but not for X_{j_2} and the rest. When this stage of the procedure is complete, the expression contains a parenthesized term after each auxiliary character, but of course some parenthesized terms are of the form $(*)$.

The second stage of the generative procedure replaces each auxiliary character with a terminal one and orders the terminal characters in a string. The replacement is controlled by the assignment function of the grammar; each auxiliary corresponds to a list of terminals, of which one is chosen.

The order of the units in each dependency rule is given as part of the rule; these various orderings are combined in accordance with a principle shown by the following example.[12]

Let the expression developed with dependency rules be

$$*(X_i(X_{j_1}(X_{k_1}(*), *), *, X_{j_2}(*), X_{j_3}(X_{p_1}(*), *))).$$

The six auxiliary characters must be ordered. The main element, X_i, is governing member in the rule $X_i(X_{j_1}, *, X_{j_2}, X_{j_3})$. Assign order numbers to these four characters in the obvious way, that is, number 1 to X_{j_1}, number 2 to X_i, number 3 to X_{j_2}, and number 4 to X_{j_3}. Now, X_{j_1} governs in $X_{j_1}(X_{k_1}, *)$. Taking the number 1 already assigned to X_{j_1}, expand it: Assign 1.1 to X_{k_1} and 1.2 to X_{j_1}. The number finally

assigned to X_{j_3} is 4.2, and to X_{p_1}, 4.1. Taking these numbers in order gives:

$$
\begin{aligned}
1.1 &: X_{k_1} \\
1.2 &: X_{j_1} \\
2 &: X_i \\
3 &: X_{j_2} \\
4.1 &: X_{p_1} \\
4.2 &: X_{j_3}
\end{aligned}
$$

Thus the sentence is ordered by composition of rule orders.

Any dependency grammar, then, characterizes the set of sentences that can be generated by this procedure, using the rules and assignment function of the grammar. Since all the lists in a dependency grammar are finite, this generative procedure obviously requires only a finite number of finite operations for any finite string that it can generate.

4. A decision procedure

Consider the class of all finite strings of elements of a given terminal alphabet, and suppose that a dependency grammar is specified. A decision procedure must assign each string to one of two sets: one is the set of strings characterized by the grammar, and the elements of this set are said to be accepted by the decision

procedure; the other set consists of strings of elements that are rejected by the procedure.

The first step in the procedure to be described here is to list all auxiliary characters corresponding to each minimal syntactic unit of a given string. Next, consider all rules of the grammar that match substrings of auxiliaries. For example, suppose the string under examination consists of four minimal syntactic units, corresponding to auxiliary characters as given in Table 1.

Minimal syntactic unit	1	2	3	4
Corresponding auxiliary characters	X_{11} X_{12}	X_2	X_{31} X_{32} X_{33}	X_{41} X_{42}

Table 1. Assignment of auxiliaries to syntactic units

If there are rules $X_{11}(*)$, $X_{32}(*)$, and $X_{41}(*)$ in the grammar, add them to the list (Table 2).

Minimal syntactic unit	1	2	3	4
Corresponding auxiliary	X_{11} X_{12}	X_2	X_{31} X_{32} X_{33}	X_{41} X_{42}
One-character rules	$X_{11}(*)$		$X_{32}(*)$	$X_{41}(*)$

Table 2. Application of one-character rules.

Now the following two-character rules, if they exist, match substrings: $X_2(X_{11}, *)$; $X_2(*, X_{32})$; $X_{31}(*, X_{41})$; $X_{32}(*, X_{41})$; $X_{33}(*, X_{41})$; $X_{41}(X_{32}, *)$; and $X_{42}(X_{32}, *)$. Note that a character can be matched as dependent member in a rule, that the order in a rule must match order in the string, and that no jumps are allowed, e.g., $X_{11}(*, X_{32})$ does not match, since it jumps unit 2.

Suppose, for the sake of the example, that the only two-character rule in the grammar, among those that could match, is $X_2(*, X_{32})$. Turning to three-character substrings, we note that there are just two spans of three characters, namely, the first, second, and third units and the second, third, and fourth. The first of these matches any of the following rules: $X_2(X_{11}, *, X_{32})$; $X_{11}(*, X_2)$; $X_{12}(*, X_2)$. The latter rules are matchable if we take X_2 as governing X_{32}. The second span matches any of the following rules: $X_2(*, X_{32}, X_{41})$; $X_{41}(X_2, *)$; $X_{42}(X_2, *)$. No jump is involved in the latter rules, since the third syntactic unit is taken with the second.

Suppose that the grammar actually includes the rules $X_{11}(*, X_2)$ and $X_2(*, X_{32}, X_{41})$. Then the tabulation to date has the appearance of Table 3.

Minimal syntactic unit	1	2	3	4
Corresponding auxiliary character	X_{11} X_{12}	X_2	X_{31} X_{32} X_{33}	X_{41} X_{42}
One-character rules	$X_{11}(*)$		$X_{32}(*)$	$X_{41}(*)$
Two-character rules		$X_2(*,X_{32}(*))$		
Three-character rules	$X_{11}(*,X_2(*,X_{32}(*)))$	$X_2(*,X_{32}(*),X_{41}(*))$		

Table 3. Application of rules spanning two or three units

A rule covering the whole string of four minimal syntactic units will account for the string in terms of the grammar. The following rules would match: $X_{11}(*, X_2)$; $X_{12}(*, X_2)$; $X_{11}(*, X_2, X_{41})$; $X_{12}(*, X_2, X_{41})$; $X_2(X_{11}, *, X_{32}, X_{41})$; $X_{41}(X_{11}, *)$; $X_{42}(X_{11}, *)$; $X_{41}(X_{11}, X_2, *)$; and $X_{42}(X_{11}, X_2, *)$. The first of these rules would lead to the expression

$$X_{11}(*, X_2(*, X_{32}(*), X_{41}(*))).$$

The third leads to

$$X_{11}(*, X_2(*, X_{32}(*)), X_{41}(*)).$$

Similarly, each rule leads to an expression corresponding to the whole string and built out of subexpressions already tabulated.

If any rule covering the whole string exists in the grammar, the final step is to test whether the first character of the expression obtained appears in a rule $*(X_i)$. If it does, the decision procedure accepts the string; it belongs to the set characterized by the grammar.

The set of strings accepted by this procedure, given a certain grammar, is almost obviously identical with the set generated by the procedure of Sec. 3, given the same

grammar. Gaifman[13] proves equivalence for a similar pair of procedures.

The procedure just described is almost (but not quite) an outline of a practical algorithm for recognition, as required, for example, in such branches of applied linguistics as machine translation. It is closely related to procedures suggested by Cocke[14] and Lecerf.[15] The differences are simplifications for the sake of clarity.

5. Structures

The structure of an utterance is a representation of its parts and their properties and interrelationships. It is clear that any structure must reveal the application of a grammar to a particular string, and also that ambiguous utterances must have alternative structures, at least insofar as their ambiguities are grammatical. Other attributes and uses of structures are subject to debate; further remarks appear in later sections. Here it is shown that dependency theory permits assignment of structures to utterances—structures that differ considerably from constituency diagrams.

Dependency structures are in fact represented by the expressions already used in Secs. 3 and 4. In the example of the previous section, a possible expression for the complete string was the following:

$$*(X_{42}(X_{11}(*),\ X_2(*,\ X_{32}(*))),\ *).$$

According to this expression, the fourth minimal syntactic unit of the string, interpreted as type X_{42} rather than X_{41}, is the syntactically central unit. It governs the first and second units of the string, those units being given syntactic interpretations X_{11} and X_2, respectively. Finally, the second unit governs the third, interpreted as X_{32}.

In an expression of this form, the assignment function is not represented. Replacing each asterisk (except the first) with a terminal character yields an expression which corresponds to exactly one terminal string and shows precisely how the dependency grammar accounts for that string. In cases of ambiguity, two or more such expressions correspond to the same terminal string. For a terminal string that does not belong to the set characterized by the given grammar, no such expression can be written.

Returning to an earlier example, consider Children eat candy neatly. An expression for this utterance might be as follows:

$$*(V_{\alpha}(N_{pl}(\underline{childr},\ S(\underline{-en})),\ \underline{eat},\ N(\underline{candy}),\ D_{\beta}(\underline{neat},\ L(\underline{-ly}))).$$

The terminal string, in this interpretation, is a string of morphs: childr -en eat candy neat -ly. How such a string might be converted into a string of phonemes or graphemes is discussed in Sec. 7. For an irregular noun like children, assignment of childr to a syntactic category N_{pl} is not especially unnatural, but to assign book to N_{sg} when it occurs without a plural affix, and to N_{pl} when the affix is present, does seem unnatural. Such double assignments are by no means inherent in dependency theory; in Sec. 7, various techniques for avoiding them are proposed.

The expressions used here for dependency structures are linearized expressions for labeled, rooted trees.[16] A tree is a set of elements on which a relation is defined; in effect, the relation connects every element of the set directly to at least one other, and there is exactly one sequence of connections from any element to any other. A tree is rooted if one element, the origin, is distinguished from the rest. A tree is labeled if properties of its elements—other than the defining relation—are noted.[17] In dependency structures, the origin of the tree is the syntactically central unit; the relation is that of direct dependency, and the labels are auxiliary characters and order indicators.

Trees are often presented in diagrams like that of Fig. 1; the structure illustrated there is the one represented by

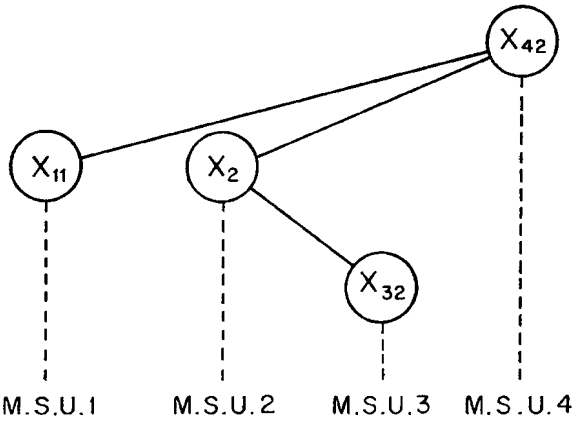

Fig. 1—Diagram of a dependency structure
(M.S.U. = minimal syntactic unit)

the expression just analyzed. The root element of the tree is at the top in the diagram; solid lines stand for dependency connections, the lower element depending on the higher. Each element is labeled explicitly with an auxiliary character, implicitly with an order indicator. Below the tree is a string of minimal syntactic units, each joined to the corresponding element of the tree; the broken lines map the assignment function. When such a diagram is accurately constructed, there are no intersections of connection lines with one another or with projection lines; this follows directly from the ordering rules given earlier as part of the construction procedure. This attribute, called projectivity,[18] is analogous to non-discontinuity of consitutents in immediate-constituent theory.

Tree diagrams and linear expressions correspond biuniquely; in passing from one to the other, there is no opportunity for variation. Thus the dependency structure of a sentence can be understood by reference to either without ambiguity.

6. Comparison of dependency and IC theories

Two theories of grammar can be compared on divers grounds. Formally, two issues arise: (i) Can every set of strings characterizable by a grammar of one type be characterized by a grammar of the other type? Two

grammars are said to be weakly equivalent if they characterize the same set of strings; let us say that two theories are weakly equipotent if every grammar of either type is weakly equivalent to some grammar of the other type. (ii) Can the structures assigned to strings by grammars of the two types be set into correspondence in such a way that corresponding structures are attributed to the same strings? In other words, choose a grammar of one type and consider whether there exists a grammar of the other type that not only characterizes the same set of strings, but also attributes, to each string, structures corresponding to those attributed to the same string by the first grammar. Two grammars are said to be strongly equivalent if they have this property; let us say that two theories are strongly equipotent if every grammar of either type is strongly equivalent to some grammar of the other type.

In addition to these formal questions, there are others. The relative economy of grammars of the two types might be discussed; does one theory generally require shorter lists of rules, or smaller auxiliary alphabets? Intuitive appeal, especially with respect to the form of the structures assigned, is likely to influence the personal choices of investigators. If the theories in question are partial theories of linguistic behavior, to

be included in systems with, for example, transformations or multiple strata, one theory may be considerably more appropriate than the other for integration into the larger system. If empirical evidence beyond sentencehood is admitted, still further bases of comparison may be found.

Dependency theory is weakly equipotent to IC theory. The proof is due to Gaifman,[19] and is too lengthy to present here. The consequence of Gaifman's theorem is that the class of sets of utterances characterizable as in Sec. 3 or 4 is Chomsky's class of context-free languages.[20]

A demonstration of strong equipotence rests on a pre-established formal definition of correspondence between structures. Gaifman studies two definitions. According to his own definition, dependency theory is strongly equipotent to an essentially trivial subclass of IC grammars. The other definition is taken from Hays,[21] and will now be stated.

An IC grammar cannot differentiate governors from dependents, but it can differentiate degrees of closeness of relation. Thus a dependency grammar for English might treat verbs as governors of their subjects and objects, whereas an IC grammar might treat verb and object as partners in a constituent, the latter in turn being one partner in a subject-object construction. Since the two theories emphasize different aspects of linguistic

patterning, it is reasonable to adopt a definition of structural correspondence that allows several dependency structures to correspond to one IC structure, and vice versa.

A subtree is a connected subset of a tree. A complete subtree consists of some element of a tree, plus all others connected to it, directly or indirectly, and more remote from the origin of the tree. The tree in Fig. 1 includes four complete subtrees: (X_{11}), (X_2, X_{32}), (X_{32}), and all four elements (since the tree is a subtree of itself).

An IC structure and a dependency structure, both defined over the same string, correspond relationally if every constituent is coextensive with a subtree and every complete subtree is coextensive with a constituent. (Two structural entities are coextensive if they refer to the same elements of a terminal string.) The sense of relational correspondence is plainly to fix requirements for similarity of unlabeled structural diagrams; this is the extent of Hays's definition.

Gaifman adds a restriction on labeling; its necessity is apparent from Fig. 2, where an ambiguous sentence is shown with two IC diagrams and two dependency trees. Structures (a) and (c) take are flying as a compound verb; structures (b) and (d) take flying planes as a noun phrase. Yet structure (c) corresponds relationally to both (a)

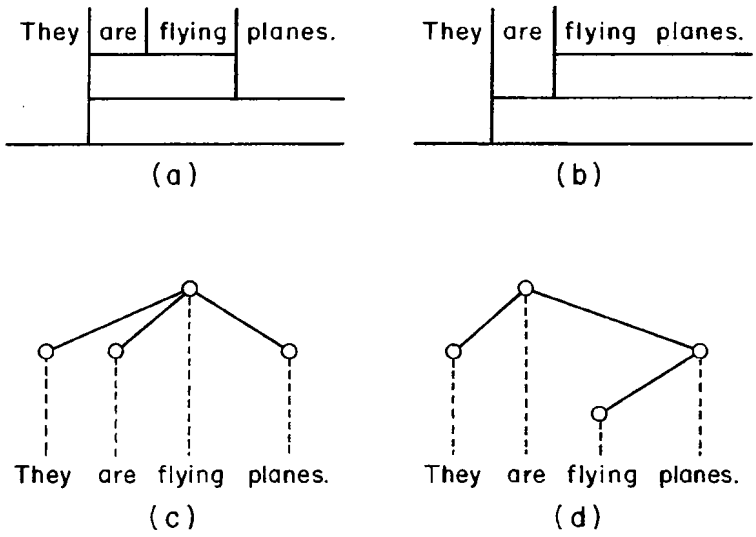

Fig.2—IC and dependency structures for an ambiguous sentence

and (b). The requirement of label correspondence avoids such counterintuitive matches.

In dependency theory, a string derives from the auxiliary that labels the root node in a corresponding tree. Thus every string in the set characterized by a dependency grammar derives from one of the auxiliaries X_i such that $*(X_i)$ is a rule in the grammar. In IC theory, a string derives from the auxiliary that labels it as a constituent. These definitions are valid for all strings covered by complete subtrees, on the one hand, or by constituents, on the other. An auxiliary accounts for the set of strings that derive from it; two auxiliaries are substantively equivalent if they account for the same set of strings.

An IC structure and a dependency structure correspond if (i) they correspond relationally, and (ii) every complete subtree has an origin labeled with an auxiliary that is substantively equivalent to the auxiliary labeling the coextensive constituent.

A dependency grammar is strongly equivalent to an IC grammar if (i) they have the same terminal alphabet, and (ii) for every string over that alphabet, every structure attributed by either grammar corresponds to a structure attributed by the other. It follows that the grammars are weakly equivalent, since if one assigns no structure to a string, the other must do likewise.

Gaifman shows that every dependency grammar is strongly equivalent to an IC grammar, but that IC grammars with a property he establishes are not strongly equivalent to any dependency grammars. Hence the two theories are not strongly equipotent. The property of some IC grammars that cannot be imitated in dependency theory cannot be stated in a few words; it seems most unlikely that linguistic applications would call for grammars with the property, and if that be true, dependency theory is in that respect preferable: The weakest empirically adequate theory is always best, other criteria failing to distinguish. However, the empirical point is unsettled.

As for the relative economy of dependency and IC theories, nothing is presently known. The intuitive appeals of the two theories cannot be discussed, since intuitions are personal and irrational. Other bases for comparison are adduced in the following sections.

7. Extensions of dependency theory

In contemporary linguistic theory, two major avenues for extension of syntactic models have been opened. One is transformation theory, which extends context-free IC grammars by admitting additional kinds of rules. The other is stratification theory, which extends IC grammars by combining them in sequences. Proponents of both systems claim greater power, economy, and intuitive acceptability.

Both avenues do guarantee greater formal power. Dependency and IC theories, weakly equipotent to one another, can characterize only a subclass of the sets of strings characterizable by transformational or stratified systems. Economy, in a sense, can be improved by following either avenue, although discussions of this point generally fail to mention the loss of economy incurred by admitting more powerful rules or sequences of grammars. Economy, in any case, is often a matter of notational conventions, achievable without a change in power.

A transformation is a rule that replaces an ordered k-tuple of IC structures, or P-markers, with a single structure. One trite example is the combination of structures associated with the old man and the tall man to yield the structure associated with the tall old man. The argument for use of a transformation rule in this case is that an IC grammar must overspecify the structure by making old man a constituent of tall old man; otherwise, an IC rule would be needed for a noun with one adjective, another rule for a noun with two, another for three, and so on with no obvious limit. Other transformations are proposed for passive voice, for conjunction, and so on.

Rules that compose k-tuples of dependency trees into single trees can be defined in like manner. This proposal

does not fall within the scope of Chomsky's strictures[22] on transformation rules applied to k-tuples of strings, since dependency trees carry structural information. Casual examination suggests that there would be little difference between transformation of dependency trees and transformation of IC structures, but no definite investigation has been undertaken.

In a stratified grammar, each stratum has its own alphabet, tactic rules, and so on; Lamb[23] postulates four strata, which he calls phonemic, morphemic, lexemic, and sememic. A string over the alphabet of any stratum has three kinds of properties. Firstly, it does or does not belong to a set characterized by a grammar for its stratum. Secondly, it is related, by a transduction, to a single string or set of alternative strings on the adjacent higher stratum. And thirdly, it is similarly related to a string or set of strings on the adjacent lower stratum. (Naturally, exceptions must be made for strings on the highest and lowest levels.)

Empirically, the set of strings on the lowest stratum is of primary interest, since these strings, if any, are testable for sentencehood. The set of principal interest is not merely the set characterized by the lowest stratal grammar (i.e. the phonemic), but the subset that maps, by transduction, onto a certain set of morphemic strings.

Again, the set at issue is not the whole set characterized by a morphemic grammar, but a subset. In short, a terminal string must satisfy independent phonemic, morphemic, lexemic, and sememic criteria.

The application of several grammars is tantamount to taking the intersection of several languages. Let us suppose, for example, that the grammars of the four strata are context free, and that the transductions, or conversions, from stratum to stratum are finite state. Then the conversions have no effect on the power of the system, since a finite-state transduction maps any context-free language into a context-free language.[24] The system thus characterizes the intersection of four context-free languages; the intersection, not necessarily context free, is context sensitive.[25] The result is the same even if only two of the grammars are context free; those of the phonemic and morphemic strata could be taken as finite state with no loss in power, and indeed both Hays[26] and Lamb[27] have made such suggestions.

Transductions are not necessarily finite state, although such operations as consultation of a dictionary are readily formalized with finite-state machines.[28] Pushdown transducers are essentially more powerful; one possible application in stratification theory is in generative systems or decision procedures with vacuous grammars for all but one of the strata. A sequence of finite-state

transductions up to the sememic stratum, followed by application of a context-free grammar, is a decision procedure weakly equipotent to direct application of a context-free grammar on the lowest stratum. (Questions of economy are not being considered here.) On the other hand, a sequence including at least one pushdown transduction and ending with application of a context-free grammar is more powerful.

A significant aspect of stratification theory is its demand for transcription of the text on each stratum. If Lamb's four strata are adopted, four alphabets must be posited. An element of the alphabet on any stratum is said to be represented by an element or combination of elements on the next lower stratum. On each stratum, moreover, the law of composition of alphabetic elements can be chosen freely. In linguistics, the tradition is to deal always with strings, associating structural descriptions (e.g., P-markers) with them. On the phonemic stratum, and even perhaps on the morphemic stratum in Lamb's model, this tradition need not be altered. On higher strata, other composition laws are attractive. Instead of concatenating lexemic elements into strings, we can concatenate them, by a different law, into trees. In particular, we can regard dependency trees not as structural descriptions of strings but as the fundamental transcriptions of utterances on their stratum. For the sememic stratum, a still more complex law is wanted; the transcription there takes the form of a

generalized graph, or network.

If this view is adopted, the transduction from the morphemic stratum to the lexemic can be regarded as having two stages. The first, to be understood intuitively as the consultation of a dictionary, passes from a string of morphemic elements to one of lexemic units. The second, very much like a decision procedure for a dependency grammar, passes from a lexemic string to a lexemic tree. A similar account could be given of transduction to the sememic stratum, but here trees must be mapped onto graphs.

If a dependency tree is to be the transcription of an utterance, it must be labeled at each node with sufficient information to identify the corresponding phonemic transcription, at least up to the point of free lexemic, morphemic, and phonemic variation. Yet the amount of information required at each node is far less than that given by the corresponding element of a lexemic string; some of the information given by the identity and order of elements in the string is also given by the grammatical labels and dependency relations in the tree. To record only the string is to leave implicit dependency relations which, explicitly recorded, are helpful in conversion to the next higher stratum; to record both tree and string introduces redundancy. To take the tree as lexemic transcription, with only the necessary supplementary labeling at each node, resolves the dilemma.

In fact, it is possible to construct a dependency grammar in such a fashion that the order of elements in lexemic strings, and all so-called grammatical elements (such as affixes of agreement), are determined from the labels of nodes and the dependency relations during transduction downward.

A grammar composed in this manner eliminates the inefficiency, mentioned in Sec. 2, of assigning a unit such as book- to two categories, N_{sg} and N_{pl}. The double assignment was made only for the sake of agreement with verbs. According to the new plan, only the subject (or only the verb) would have fixed grammatical number in the lexemic tree transcription. The relation of subject to verb would be differentiated from that of object to verb. In the transduction to the morphemic stratum, the order of these elements would be determined and the number of the unmarked element would be decided in accordance with the agreement rule, now a rule of transduction.

8. Empirical questions

The distinction between formal models and classes of empirical data, or procedures for data collection, is sometimes lost in linguistic debates. If the data of linguistics are restricted to observations of grammaticality or sentencehood, then no empirical reason for preferring the dependency model to that of phrase structure—or vice versa—can be given, since the two are weakly equipotent.

If instead the range of data to be explained by linguistic theory is widened, empirical grounds for choice can perhaps be found. This statement is still quite far removed from any associating dependency theory with a specific interpretation.

One line of interpretation would make dependency a semantic theory, justifying the valences in any grammar by reference to meaningful relations among elements. Naturally, in order to give this interpretation an empirical basis, techniques for collecting data about meaningful relations would have to be devised. As Garvin[29] has pointed out, translation and paraphrase give at least indirect evidence about meaning; the task of obtaining semantic data is not hopeless.

As an argument favoring adoption of a dependency model, this one is potentially interesting. It can be put in terms of simplifying the transduction between two strata (Lamb's lexemic and sememic). It provides a rationale for counting co-occurrences of elements. If phrase-structure theory is used, the only pairs of elements that can be said to co-occur in text are those that occur as partners in simple constructions. When one of the partners is complex, the elementary partner co-occurs with the complex, not with any of its elementary constituents. In the dependency structure of a sentence, every element co-occurs with its governor and with its dependents, if any.

Another line of interpretation makes dependency a psychological relation. Empirically, this interpretation would require data about the structural intuitions of native speakers; dependency theory would be supported if many or most native speakers could agree on the central (governing) element in each utterance presented to them, and on the connections binding elements together. Phrase—structure theory would be supported if they agreed on containments, e.g., that object—verb relations are closer than subject—verb relations, so that predicates are contained in sentences.

Psycholinguistic data are notoriously hard to manage. Naive native speakers of Standard Average European are almost extinct; readily obtainable data are bound to be contaminated with school-learned rules that in no way affect the speaker's language system outside the observation room. The linguist need not conclude, however, that psycholinguistic data are totally forbidden to him, but only that their use requires proper controls.

Like the semantic argument, the psychological justification for dependency theory is potentially interesting. Since neither kind of data is presently available, there is no reason to attach dependency theory to either. Recent formal work in linguisitcs has drawn attention to the inadequacy of sentencehood data; at best, use of such data must leave the linguist with a plurality of theoretical models, each capable of explaining his facts. Turning to

other varieties of data seems, at this time, to offer the best hope of deciding relatively delicate issues like the choice between IC and dependency theories, or between transformation and stratification.

9. Acknowledgments

The author is especially grateful to Charles F. Hockett, Martin J. Kay, Sydney M. Lamb, and Yves Lecerf for discussions of the subject of this paper.

REFERENCES

[1]Zellig S. Harris, Lg. 22.161–83 (1946).

[2]Noam Chomsky, 'Three models for the description of language', I.R.E. Trans. PGIT2.113–24 (1956).

[3]L. N. Iordanskaya, 'O nekotorykh svojstvakh pravil'noj sintaksicheskoj struktury', Vopr. Yaz. 4.102–12 (1963).

[4]Louis Hjelmslev, Prolegomena to a theory of language 24 (Madison, 1961).

[5]Noam Chomsky and George A. Miller, 'Introduction to the formal analysis of natural languages', Handbook of mathematical psychology (R. Duncan Luce, Robert R. Bush, and Eugene Galanter, eds.) 2.292–3 (New York, 1963).

[6]Y. Bar-Hillel, M. Perles, and E. Shamir, 'On formal properties of simple phrase structure grammars', Zeit. Phonetik, Sprachwiss. und Kommunikationsforsch. 14.143–72(1961).

[7]Haim Gaifman, 'Dependency systems and phrase structure systems', Info. and Control, in press.

[8]Chomsky and Miller, 283.

[9]Eric P. Hamp et al., 'The transformation theory, panel', Georgetown Univ. monograph series on lang. and ling. 4.36 (Washington, 1964).

[10]Chomsky and Miller, 283.

[11]Noam Chomsky, 'Formal properties of grammars', Handbook of math. psychol. 2.326–31.

[12]David G. Hays, 'Basic principles and technical variations in sentence-structure determination', Information theory (Colin Cherry, ed.) 367–376 (Washington, 1961). The procedure was suggested by Thomas Mullikin.

[13]Gaifman, op. cit.

[14]David G. Hays, 'Automatic language—data processing', Computer applications in the behavioral sciences (Harold Borko, ed.) 394—421 (Englewood Cliffs, N. J., 1962).

[15]Y. Lecerf, 'Programme des conflits, modèle des conflits', la Traduction automatique 1:4.11—20, 1:5.17—36.

[16]Saul Gorn, 'Specification languages for mechanical languages and their processors—a baker's dozen', Comm. of the ACM 4.535—7 (1961).

[17]Claude Berge, Théorie des graphes et ses applications (Paris, 1959).

[18]P. Ihm and Y. Lecerf, Eléments pour une grammaire générale des langues projectives (Bruxelles, 1963).

[19]Gaifman, op.cit.

[20]Chomsky, 'Three models'.

[21]David G. Hays, 'Grouping and dependency theories', Proc. nat. symp. mach. trans. (H. P. Edmundson, ed.) 258—66 (Englewood Cliffs, N. J., 1961).

[22]Noam Chomsky, 'On the notion "rule of grammar" ', Proc. Symp. Appl. Math. 12.18 (1961).

[23]Sydney M. Lamb, 'The sememic approach to structural semantics', Transcultural studies in cognition (Roy d'Andrade and A. Kimball Romney, eds.) in press.

[24]Seymour Ginsburg and G. F. Rose, 'Operations which preserve definability in languages', J ACM 10.175—95 (1963).

[25]Sheila A. Greibach, personal communication.

[26]David G. Hays, 'Linguistic methodology and the theory of strata', presented at a meeting of the Philol. Assoc. of the Pac. Coast, November, 1961.

[27]Sydney M. Lamb, 'Stratificational linguistics as a basis for mechanical translation', Proc. US—Japan seminar on Mech. trans., to appear.

[28]Sydney M. Lamb and William H. Jacobsen, Jr., 'A high—speed, large—capacity dictionary system', Mech. trans. 6.76—107 (1961).

[29]Paul L. Garvin, 'A descriptive technique for the treatment of meaning', Lg. 34.1—32 (1958).