Asymmetries in Word Order and Syntactic Structure as Lexical Graph

Miyoko Yasui

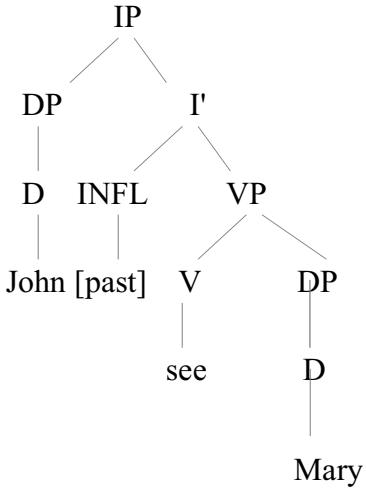
Dokkyo University

February, 2004

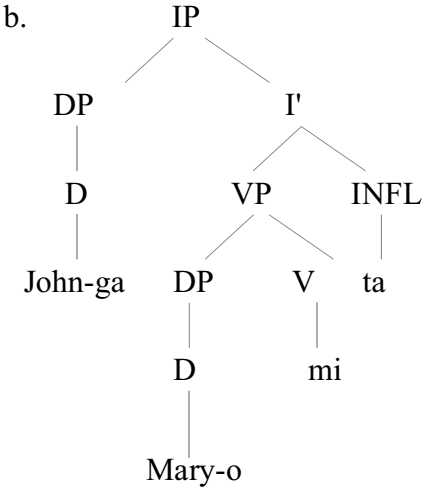
1. Word Order Variation Among and Within Languages

It has been widely assumed that head-initial languages like English and head-final languages like Japanese are hierarchically the same, and their contrast in word order has been described by the head-parameter. For instance, the English prepositional phrase *see Mary* and its Japanese counterpart have the structures in (1a,b) after the head parameter value is fixed for each language:

(1) a.



b.



In both (1a,b), V asymmetrically c-commands D, and other hierarchical relations among the non-terminal nodes are the same; only the linear orderings of terminal nodes and of sister projection nodes are different. This accords our intuition that linear properties are not syntactically significant and hierarchical properties are universal. For example, a reflexive in object position can refer to the subject, whether it follows or precedes the verb, which shows that the antecedent-reflexive relation is essentially hierarchical.

Kayne (1994) challenges the above view, advocating the universal word order rather than the universal hierarchical structure. According to Kayne's theory, linear ordering is mapped from asymmetric c-command relations that hold between non-terminal nodes; thus,

distinct word orderings should reflect distinct hierarchical structures. It follows that (1b) cannot be the structure for the Japanese sentence in question since it involves exactly the same set of asymmetric c-command relations as those in (1a) and its terminal nodes should be ordered just as the corresponding nodes in (1a). Kayne (1994, 2003) replaces (1b) with a head-initial underlying structure with abstract functional categories, to which movement operations apply so as to derive the surface head-final order.[1] Chomsky (1995) points out some weaknesses of Kayne's theory such as its crucial reliance on non-branching nodes to deduce surface order. The most unattractive aspect, however, seems to be its rejection of hierarchically parallel analyses of head-initial and head-final languages exemplified in (1a,b).

Another type of word order variation that is not strongly linked to clear syntactic or semantic differences has been referred to under the name of scrambling. (2) is as acceptable as (1a) without a heavy stress on the initial constituent or a pause after it:

(2) Mary-o John-ga mi-ta 'Mary, John saw.'
-Acc -NOM go-Past

One dominant approach represented by Saito (1985) and subsequent work is to regard (1b) as basic and derive (2) by the syntactic operation called scrambling. Scrambling produces an adjoined structure and the moved constituent leaves a trace in its original position. Thus, (1b) and (2) differ not only in their orderings of the words but also in their hierarchical properties. Saito presents syntactic evidence on the structural distinction between (1b) and (2) but also recognizes that unlike wh-movement and topicalization, clause-internal scrambling is semantically vacuous in failing to establish a operator-variable relation.[2] In contrast to Saito's position, Hale (1980, 1981), Farmer (1980, 1984) and others claim that the spec-complement-head (SCH) order exemplified by (1b) is only a tendency. Hale analyzes (2) on a par with (1b), giving them flat structures without distinguishing spec and complement hierarchically. This approach can capture the word order variation in Japanese straightforwardly, but it amounts to claiming that Japanese and English are different not only in word order but also in their hierarchical properties, unlike the analysis in (1a,b). The main purpose of this paper is to offer a theory of phrase structure that will assign the same

hierarchical structure to (1a,b) and (2) without specifying their linear differences syntactically, and to formulate algorithms that can deduce multiple PF interpretations from the shared syntactic structure.

The flexibility in word order or the multiplicity of PF interpretation appears to be attested only in head-final languages. English, which is head-initial, allows a certain amount of word order freedom by shifting a heavy NP rightward as shown in (3a,b), but a light constituent like the pronoun *it* cannot be shifted rightward, as shown in (3d):

(3) a. They brought the beautiful dress into my room,

b. They brought into my room the beautiful dress. (Fukui (1993: 410))

c. They brought it into my room.

d. *They brought into my room it.

On the other hand, Japanese scrambling moves a constituent leftward, whether it is light or heavy, as shown in (4a,b):

(4) a. sono utukusii doresu-o karera-wa watasi-no heya ni mottekita.
that beautiful dress-Acc they-Top I-no room to brought

b. sore-o karera-wa watasi-no heya ni mottekita.
it-Acc they-Top I-no room to brought

It can be said that head-final languages allow wider variation in word order than head-initial languages, and the variation in the former is always leftward; rightward word flexibility is non-existent or highly limited.[3] Another important asymmetry between head-initial and head-final languages is pointed out by Greenberg (1964), which is cited by Bach (1970) and Bresnan (1972) among others. Bresnan (1972: 42) states that only languages with clause-initial COMP permit a COMP-attraction transformation. Logically possible but non-existent are languages with a clause-final COMP that attracts a *wh*-phrase rightward.

In this connection, Fukui (1993: 400) proposes an interesting account of the correlation between a value of the head-parameter and word order flexibility based on (5):

(5) A grammatical operation (Move alpha, in particular) that creates a structure that is inconsistent with the value of a given parameter in a language is costly in the language,

whereas one that produces a structure consistent with the parameter value is costless. According to (5), scrambling in Japanese is costless since it moves a constituent leftward and does not destroy the head-finality, whereas heavy-NP shift in English, which moves a constituent rightward and preserves its head-initiality, is likewise costless. (5) predicts that Japanese shows mirror-image properties of those of English, but this is not strictly correct, as has been shown by the contrast between (3) and their Japanese counterparts in (4). Moreover, (5) incorrectly predicts that at least some head-final languages should have rightward costly operations that are mirror-images of English wh-movement and comparable obligatory leftward movements in languages with the SHC order. The asymmetries can be restated in terms of Fukui's terminology as (6a,b) :

(6) a. A language has a costless optional movement (or shows flexible word order) only if it is head-final, and the operation produces a structure consistent with the head-final value (i.e., it is leftward).

b. A language has a costly obligatory movement only if it is head-initial, and the operation produces a structure inconsistent with the head-initial value (i.e., it is leftward).

Setting aside the existence of heavy NP shift, I will assume that (6a,b) largely hold. (6a,b) accord with half of Fukui's theory and deny the rest. To account for (6a,b), I will propose to regard syntactic structure as consisting solely of lexical items (or terminal nodes in the standard phrase structure tree); multiple PF-interpretations are deduced from it according to modified versions of tree traversal algorithms studied in graph and algorithm theories, which apply to a given graph and yield more than one ordering of its nodes. This conception of PF-interpretation will incorporate Fukui's insight on the correlation between a value of the head-parameter and the directionality of movement and Kayne's ingenious attempt to deduce word order from configurational properties.

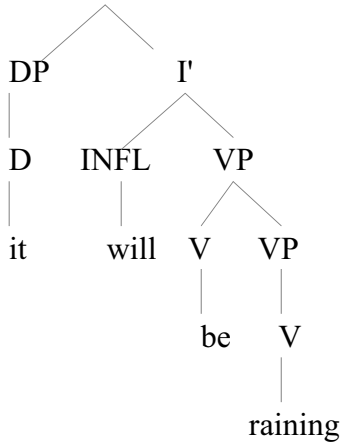
The rest of this paper is organized as follows: In Section 2, I will replace a standard tree representation of syntactic structure with a graph that consists solely of lexical items, which is to be referred to as lexical graph. A lexical graph is a pair (V, E), where V is the set of

lexical items, and E is the set of ordered pairs of lexical items that express the syntactic relation holding between the nodes. I will argue that this conception of syntactic structure best satisfies the Inclusiveness Condition proposed by Chomsky (1995) and assumed in subsequent work. In Section 3, I will claim that the nodes of a lexical graph are indexed based on the theory of crash-proof structure-building proposed by Frampton and Gutmann (1999, 2000): they are indexed in ascending order as they are introduced into the derivation in a crash-proof manner. In Section 4.1, I will outline three kinds of standard tree traversal algorithms, which are crucially based on the linear distinction of sister nodes: preorder, inorder, and postorder. In Section 4.2, I will first eliminate the linear distinction from the inorder and postorder algorithms by the use of the crash-proof indexing and the overall configuration of a lexical graph. I will then argue that the rigid SHC order of English and the fairly flexible head-final order of Japanese can be deduced by applying the modified algorithms to bilingual graphs shared by the languages. In Section 4.3, the proposed analysis will be formalized in the programming language C. Section 5 will examine distinct traversals of lexical graphs from the viewpoint of economy. Head movement and some other remaining issues will be briefly discussed in the final section.

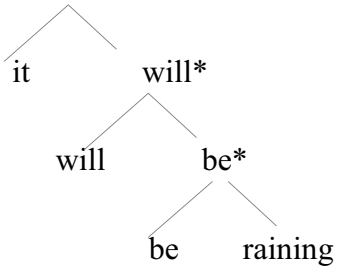
2. The Inclusiveness Condition and Syntactic Structures without Projection Nodes

A standard phrase structure representation contains projection nodes for minimal projections such as D and INFL, for intermediate projections such as V' and INFL' and for maximal projections such as VP and IP. Chomsky (1995: Chapter 4) argues, based on the Inclusiveness Condition, that minimal projection nodes are redundant and can be naturally replaced by lexical items they immediately dominate. Moreover, intermediate and maximal projections can be likewise eliminated without significant loss of syntactic information as long as they are not branching. For instance, (7a) can be simplified as (7b):

(7) a.



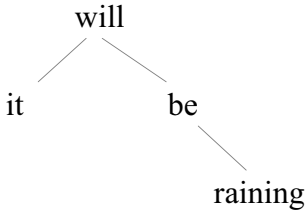
b.



Among the seven nodes in (3b), the asterisked ones correspond to the three branching nodes in (3a): the IP, the I' and the upper VP. They have no PF effects but are widely posited to define fundamental syntactic relations. In particular, the node *be** expresses the following: *be* and *raining* are sisters; they form a constituent; and the constituent they form is the same type as *be* rather than *raining*. The lower *will** has analogous functions. The upper *will** captures the fact that *it* forms a larger structure with the constituent that is of the same type as *will*, and that the resultant structure is also the same type as *will*. The question arises whether these syntactic properties can be captured without the be* and will*s.

Notice that *raining* is immediately dominated by *be** since it is selected by *be*, while *it* is immediately dominated by the upper *will** since it feature-agrees with *will*. Let us remove the asterisked nodes from (7b) and express selection and spec-head agreement in terms of edges from the head to its complement/spec as follows:

(8)



A tree like (8) will be referred to as lexical graph hereafter. In (8), the left-directed edge expresses the spec-head agreement, whereas the right-directed edges expresses the selections by *will* and *be* of their respective complements; this linear distinction will be eliminated in Section 4. The four nodes in (8) correspond to the four minimal projections in (7b). The upper will* and be* in (7b), which are IP and the upper VP in (7a), correspond to the

subgraphs rooted by *will* and *be* in (8), respectively. More generally, a constituent of the type X can be defined as X itself or a subgraph that consists of all the nodes X dominates. One constituent that falls out of this definition is the lower *will** in (7b), which corresponds to the intermediate projection I' in (7a). This is a welcome result, since an intermediate projection is syntactically invisible as Chomsky (1994:10) claims.

Graph-theoretically, a lexical graph like (8) is nothing more than a finite set of nodes (V) and a finite set of ordered pairs of the nodes (E) that express immediate domination relations. (8) is equivalent to (9):

(9) V={it, will, be, raining}

E={<will, it>, <will, be>, <be, raining>}

V corresponds to the initial lexical array . The ordered pairs in E express the selectional and agreement relations. The conception of syntactic structure as a pair of V and E best satisfies the Inclusiveness Condition in that E is formed solely by set-theoretic operations on the initial lexical array, which is V.

3. Building a Lexical Graph and Indexing its Nodes

The PF-interpretation of a standard syntactic tree is obtained by ignoring its non-terminal nodes and pronouncing its terminal nodes from left to right, and the ordering of terminal nodes comes from a value of the head parameter set for the language in question. Obviously, a lexical graph requires a different PF-interpretive algorithm, since all its nodes need to be pronounced. Does a lexical graph possess the linearity required for its PF interpretation? I will claim that it originates in its overall configuration.

Lexical items are introduced into a syntactic derivation one by one, producing a larger structure in a bottom-up fashion according to the cyclicity. Frampton and Gutmann (1999, 2000) make this point clear in their theory of crash-proof computation: lexical items are introduced automatically in the right order, and no crash caused by incorrect order of selection is possible. Then, it is natural to expect the right order of structure-building to be reflected in the PF word order. Part of their theory relevant to the discussion here can be

summarized as follows:

(10) (i) Each step of structure-building is triggered by the introduction of a new head.

(ii) If the new head is subject to selectional/agreement requirements, some of them must be satisfied immediately in its entry to the derivation before another lexical item is introduced.

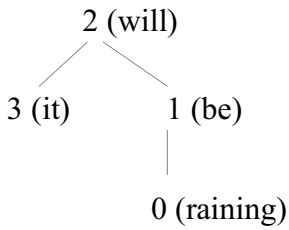
(iii) Selection takes priority to agreement.

It follows from (10i-iii) that only a lexical item that selects nothing can start the derivation; if a lexical item that selects a complement were introduced first, its selectional requirement could never be satisfied by itself and no other lexical items could be subsequently introduced.

V in (9a) contains four lexical items. *Raining* selects nothing and thus can start the derivation. The next item to be introduced is necessarily *be*, which selects a progressive verb. The third item is the modal *will*, which selects a bare verbal form. Lastly, the agreement requirement of *will* is satisfied by merging it with the expletive *it*. Suppose the four lexical items are indexed in ascending order as they are introduced into the derivation. Replacing the lexical items in V and E in (9a) with their indices results in (11):

(11) V={0, 1, 2, 3}

E={<1, 0>, <2, 1>, <2, 3>}



The content of each node is: 0=raining, 1=be, 2=will, and 3=it. This information will be explicitly represented in Section 4.2. The PF-interpretation of (11) can be obtained simply by putting the nodes in the descending order of their indices, In other words, the last-in-first-out pronunciation of the stack of the lexical items can yield the SHC order in the case of (11).

Frampton and Gutmann illustrate their theory with the sentence in (12), which involves movement:

(12) Men arrived.

(12) consists of three lexical items: *men*, [past], and *arrive*. *Arrive* selects nominal, [past]

selects verbal and agrees with nominal, and *men* selects nothing. As argued in connection with (6i-iii), *men* is the first lexical item to be introduced into the derivation. *Men* can be selected by the verb *arrive* and it can agree with the tense [past]. Given (6iii), [past] cannot agree with *men* before satisfying its selectional requirement. Thus, *arrive* is to be introduced next as a head selecting *men*. The remaining lexical item is [past], and it can select *arrive* and agree with *men*. Given (6iii), [past] is introduced as a head selecting *arrive*. The structure constructed so far is as follows:

(13) V={0, 1,2} 2 ([past])

E={<1,0>,<2,1>} 1 (arrive)

0=men,1=arrive, 2=[past] 0 (men)

All the three nodes have been indexed. Finally, the agreement requirement of [past] is satisfied by its internal Merge with *men*. I argued in Yasui (2003) that internal Merge is not different from external Merge in adding one ordered pair to E. The difference is that in external Merge, one member of a pair is a new lexical item introduced from the lexicon, while two of the lexical items already introduced form a pair in internal Merge. Suppose that the ordered pair <2,0> is just added to E as in (14):

(14) V={0, 1,2} 2 ([past])

E={<1,0>,<2,1>, <2,0>} 1 (arrive)

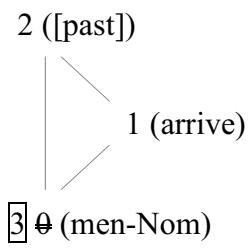
0=men,1=arrive, 2=[past] 0 (men)

Pronouncing the nodes in the descending order of their indices will yield the incorrect order: [past]-*arrive-men*. What is problematic with (14) is that addition of the edge <2,0> does not alter the index of *men*, which remains to be 0, though it is the last element affected. The bottom-up structure building can be naturally expressed by updating the index of *men* to 3 and adding <2,3> to E as follows:[4]

(15) V={1,2, 3}

E={<1,3>,<2,1>, <2,3>}

1=arrive, 2=[past], 3=men-Nom



The first ordered pair in E of (14) is <0,1>, and it is replaced by <1,3> in (15). To facilitate the understanding of this reindexing process, the original index is struck out and the new index is boxed in the lexical graph. The underlined parts in (15) are added or changed from (14). Pronouncing the three nodes in the descending order of their indices will produce the correct word order: *men-Nom*-[past]-*arrive*.

The transitive structure construction can be analyzed in a number of ways. One standard assumption is that a verb merges with the object argument before merging with the subject argument (mediated by the light verb).[5] Then, (10iii) is to be refined as follows:

(16) Selection of internal argument takes priority to that of external argument, and the latter takes priority to agreement.

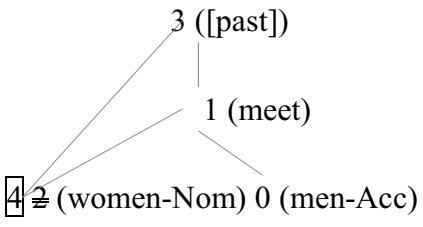
I will also assume, following Lopez (2001), that a non-ECM object is Case-checked in-situ by the lexical verb that theta-marks it rather than the light verb. Translating these assumptions into my framework and applying it to (17) results in (18):

(17) Women met men.

(18) V={0, 1, 3, 4}

E={<1,0>, <1,4>, <3,1>, <3,4>}

0=men-Acc, 1=meet,
3=[past], 4=women-Nom



The licensing theory of Case is adopted in (18); the two nouns come into the syntactic derivation with Case-features. In the graph given in (18), the index of the subject *women*-Nom is changed from 2 to 4 for the reason discussed above. Pronouncing the four lexical items in the descending order of their indices will result in the correct word order: *women*-[past]-*meet-men*.[6]

According to Kayne's theory, spec precedes head and complement since some non-

terminal node dominating the former asymmetrically c-commands some non-terminal nodes dominating the latter. The theory outlined in this section offers an alternative account: lexical items are indexed in ascending order as they are introduced into or affected in the syntactic derivation and the SHC order simply reflects the indexing.

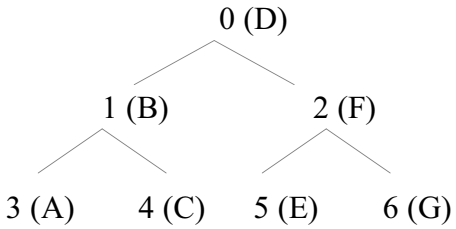
This conception of word order, however, is too simple to capture the word order flexibility of head-final languages. If SHC order is derived from the descending order, head-final order is expected to be obtainable by pronouncing the input lexical items in the ascending order of their indices (or in the first-in-first-out manner). Associating the head-finality with the ascending order of nodes' indices appears to be on the right track, but such a simple algorithm can define only one head-final ordering for a single lexical graph, contrary to the generalization in (6a). Remember that this paper has been aiming to deduce the SHC ordering in (1a) and the two head-final orderings in (1b) and (2) from essentially the same lexical graph. I will, therefore, turn to more sophisticated algorithms, which fall under traversal of graphs, or more specifically of acyclic connected graphs (i.e., trees).[7] It will be shown that the algorithms at stake do not require the total ordering of nodes imposed by the crash-proof syntax; they only need to distinguish spec from complement.

4. Traversal and the Head Parameter

4.1 Standard Tree Traversal Algorithms

Tree traversal algorithms can be classified into two major categories: depth-priority and width-priority traversals. What seems to be relevant to traversal of natural languages is the former: starting from the root, we go as deep as possible until reaching some leaf node, typically the leftmost one; we move back to its mother node and visit the other children if any; the remaining nodes are traversed in the same manner. The process can be illustrated by the binary-branching tree in (19), where the PF value of each node is given in parentheses:

(19)



The nodes are traversed in the following order: 0-1-3-1-4-1-0-2-5-2-6-2-0. If the right child is given priority at each branching node, they are traversed in the opposite order. In each case, non-terminal nodes are visited more than once; node 1, for example, is visited three times. Each node, however, are pronounced just once. There are three kinds of depth-priority algorithms that differ in the time at which each node is pronounced:

(20) a. Preorder: Pronounce a given node before its left child and right child are visited.

b. Inorder: Pronounce a given node after its left child is visited and before its right child is visited

c. Postorder: Pronounce a given node after its left child and right child are visited.

Traversing (15) according to (20a-c) results in (21a-c), respectively:

(21) a. 0-1-3-1-4-1-0-2-5-2-6-2-0 ==> "D B A C F E G"

b. 0-1-3-1-4-1-0-2-5-2-6-2-0 ==> "A B C D E F G"

c. 0-1-3-1-4-1-0-2-5-2-6-2-0 ==> "A C B E G F D"

In the three kinds of traversal, the nodes are traversed in the same order but they are pronounced in different positions, which are marked by underlines. This convention with underlines will be adopted hereafter. The important point here is that three PF-interpretations can be obtained from a single tree.

Before converting (20a-c) into algorithms in some programming language like C, the properties of (19) need to be represented explicitly. Each node is typically associated with three kinds of attributes: its pronunciation and its left child and right child. (19) is represented in C as follows:

(22) a. node[0].pfvalue="D"; node[0].left=node[1]; node[0].right=node[2];

/* node[0] is pronounced as “D”; its left child is node[1] and its right child is node[2] */

b. node[1].pfvalue="B"; node[1].left=node[3]; node[1].right=node[4];

c. node[2].pfvalue="F"; node[2].left=node[5]; node[2].right=node[6];

d. node[3].pfvalue="A"; node[3].left=NULL; node[3].right=NULL;

e. node[4].pfvalue="C"; node[4].left=NULL; node[4].right=NULL;

f. node[5].pfvalue="E"; node[5].left=NULL; node[5].right=NULL;

g. node[6].pfvalue="G"; node[6].left=NULL; node[6].right=NULL;

The statements enclosed with /* */ are annotations and irrelevant to the program. The inorder algorithm is recursively defined as follows:

(23) Inorder Traversal

```
1.      void in_order(int p){
2.          if(p!=NULL){                                    /* If there is a node    */
3.                  in_order(node[p].left);                 /* Visit its left child    */
4.                   printf("%s",node[p].pfvalue);          /* Print or pronounce the node */
5.                   in_order(node[p].right;}}              /* Visit its right child */
```

The root of the whole tree in (19) is node[0]. The application of the function *in_order* to node[0] first triggers the application of *in_order* to its left child, node[1], which in turn triggers the application of *in_order* to node[3]. Since node[3] has no children, lines 3 of (23) has been vacuously executed; and line 4 is carried out, which results in the pronunciation of "A". Line 5 is vacuously executed, too. At this stage, line 3 of the previous application of *in_order* to node[1] has been executed; thus, the next step is line 4, which results in the pronunciation of "B". Then, line 5 is carried out, which eventually results in the pronunciation of "C". At this stage, line 3 of the initial application of *in_order* to node[0] has been executed; so line 4 is executed, which results in the pronunciation of "D". The remaining three nodes of its right subtrees are traversed and pronounced in similar manners. The final result is (21b). The standard postorder algorithm can be obtained by swapping lines 4 and 5 of (23) , which yields (21c) from (19). Executing line 4 before lines 3 and 5 defines the preorder algorithm, which derives (21a) from (19).

Given the three algorithms, the inorder and postorder traversals seem to be first approximations to deduce SHC and SCH orders, respectively. If the spec and complement

of each node are represented as its left child and right child, (23) can properly deduce SHC order. (11), for example, can be represented as in (24):

(24) a. node[0].pfvalue="raining"; node[0].left=NULL; node[0].right=NULL;

b. node[1].pfvalue="be"; node[1].left=NULL; node[1].right=0;

c. node[2].pfvalue="[past]"; node[2].left=3; node[2].right=1;

d. node[3].pfvalue="it"; node[3].left=NULL; node[3].right=NULL;

Applying (23) to node[2] yields the correct surface order. As for (15), node[3] is doubly-connected, and it would be pronounced twice as the left child (spec) of node[2] and the right child (complement) of node[1]: *men-Nom [past] men-Nom arrive*. The same problem arises with (18). This can be circumvented by adding to each node's specification an attribute on its pronunciation status and revising (23) as (25):

(25)

```
1.      void in_order(int p){
2.          if(p!=NULL && word[p].pronounced==0){    /* If p is not pronounced */
3.                  in_order(node[p].left);                    /* Visit its left child    */
4.                  printf("%s",node[p].pfvalue);       /* Print or pronounce the node */
5.                  word[p].pronounced=1;                    /* Mark it as pronounced */
6.                  in_order(node[p].right;}}                 /* Visit its right child */
```

(15) should be represented as (26), where node[k].pronounced=0 means that node[k] is not pronounced:

(26) a. node[1].pfvalue="arrive"; node[0].left=NULL; node[0].right=3;
node[0].pronounced=0;

b. node[2].pfvalue="will"; node[2].left=3; node[2].right=1; node[2].pronounced=0;

c. node[3].pfvalue="men"; node[3].left=NULL; node[3].right=NULL;
node[3].pronounced=0;

Node[3] is the left child of node[2] and the right child of node[1], but lines 2 and 5 of (25) guarantee that it is pronounced just once as the left child of node[2].

One might argue that the head-final order of Japanese can be obtained by executing

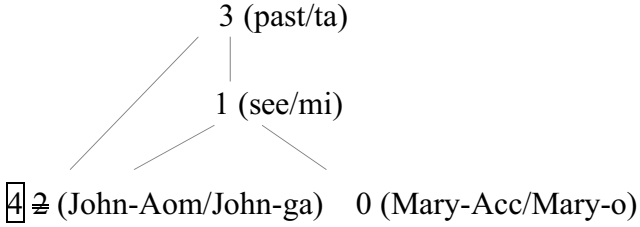
line 6 of (25) before lines 4; in fact, it yields one head-final ordering from a given lexical graph. As has been observed in Section 1, however, Japanese word order is flexible as long as the head-finality is respected. One way to derive multiple PF-interpretations is to choose at a branching node either of its child nodes rather than always giving priority to its left child. Another possibility is to start a postorder traversal from any leaf node and proceed towards the root generally in the ascending order of the nodes. This is reasonable since all the leaf nodes are pronounced before the internal nodes in postorder, as has been exemplified in (21c). The root of a graph is unique in dominating all the other nodes and that leaf nodes are also unique in the opposite sense; they dominate no other nodes. If a postorder traversal starts from a leaf node, there should be as many head-final pronunciations as the number of leaf nodes. I will revise the standard inorder and postorder algorithms along this line.

4.2 Revising the Standard Inorder and Postorder Traversal Algorithms

A standard representation of a graph such as (22) consists of information on each node's child nodes. This type of data representation is not appropriate if a graph is traversed in the opposite direction, from a node to its mother node. Moreover, the indexing system proposed in Section 3 has enough information to distinguish the left child (spec) of a branching node and its right child (complement); the index of the former is larger than that of the latter. I will attempt to formulate bidirectional traversal algorithms based on the indexing system.

Let us first consider the English and Japanese sentences in (1a,b) and (2), which can be associated with the bilingual lexical graph below:

(27)



(27) is essentially the same as the transitive construction in (18). To obtain the English SHC order in (1a), we should start from the root. The root immediately dominates two nodes and the one with the larger index is node 4; so it is visited next. Node 4 has no children; so we

go back to the root. Giving priority to a node with a larger index again would end up with a hopelessly circular traversal. This can be avoided by traversing only unpronounced nodes. When should a node be pronounced, then? In the standard formalizations, the timing of each node's pronunciation is regulated by referring to its left and right child, as stated in (20a-c). In an inorder traversal, a node is pronounced after its left child is visited and pronounced, but before its right child is visited and pronounced; it is pronounced after one of its two child nodes has been pronounced. Let us revise the standard inorder algorithm as (28), which does not refer to the left/right distinction of sister nodes:

(28) Inorder: Pronounce a given node when it immediately dominates no more than one unpronounced node.

A binary-branching node is pronounced after one of its two child nodes has been pronounced. A non-branching internal node is pronounced in its first traversal since it has just one unpronounced child node originally. A leaf node is also pronounced in its first traversal since it immediately dominates no nodes by definition.[8]

In (27), the root is not pronounced in its first traversal since it immediately dominates two unpronounced nodes. Node 4 is visited next according to the larger-index-priority strategy, and it is pronounced right away since it dominates no unpronounced nodes. We go back to the root. The root immediately dominates just one unpronounced node at this stage; so it is pronounced. Next, node 1 is visited, which is the only unpronounced node of the root. One of node 1's child nodes has been pronounced, so it is pronounced. Node 0 is visited and pronounced finally. The sequence of nodes traversed until they are all pronounced is given below:

(29) 3-4-3-1-0 ==> (1a) 'John-Nom [past] see Mary-Acc'

(29) corresponds to (1a).

It has been argued in Section 3 that SHC order can be derived directly from the descending order of nodes' indices. In the analysis proposed above, the indices have a more limited role of resolving path ambiguities at a binary-branching node, giving priority to its unpronounced child node with the larger index, which is its spec. What regulates the overall

pronunciation is mainly the changing number of each node's unpronounced children. Thus, swapping the indices of nodes 1 and 0 in (27) does not affect the result at all; the verb *see* is pronounced when it immediately dominates one unpronounced node, after its subject has been pronounced and before its object is pronounced.[9] For ease of exposition, however, I will continue to assume the indexing system proposed in Section 3.

Let us next analyze the Japanese counterpart of (27) . In the standard postorder traversal stated in (20c), a node is pronounced after all its left and right child nodes have been pronounced. This can be restated as (30):

(30) Postorder: Pronounce a given node when it immediately dominates no unpronounced node.

A leaf node is pronounced in its first traversal since it immediately dominates no nodes by definition. A non-terminal node is pronounced after all its child nodes have been pronounced. Traversing (27) just as in (29) but pronouncing its nodes according to (30) yields (31):

(31) 3-4-3-1-0-1-3 ==> (1b) 'John-ga Mary-o mi-ta'

This corresponds to (1b).

There is one more PF-interpretation of (27) given in (2). Note that pronunciation of all the nodes requires five steps in (29) but 7 steps in (31). In (31), the non-terminal nodes are not pronounced at their first traversals. It is more economical to start a postorder traversal from a leaf node and proceed in the depth-priority manner. (27) has two leaf nodes: nodes 0 and 4. Let us start from node 0. Node 0 is pronounced right away according to (30). Next, node 1 is visited, as it is the only node connected to node 0. Node 1 is not pronounced at this stage since it immediately dominates node 4, which is not yet pronounced. Node 4 is visited next and pronounced at once as it dominates no unpronounced nodes. We go back to node 1, and it is pronounced this time, since both its child nodes have been pronounced. Node 3 is visited and pronounced finally. Starting from node 4 yields (1b). The results are given below:

(32) a. 0-1-4-1-3 ==> (2) 'Mary-o John-ga mi-ta'

b. 4-1-0-1-3 ==> (1b) 'John-ga Mary-o mi-ta'

(32b) gives the same PF-interpretation as (31) does, but it does more economically.

It has been demonstrated that one head-initial PF interpretation and two head-final interpretations have been deduced from the lexical graph in (27). More generally speaking, there is only one head-initial PF-interpretation but are as many head-final PF-interpretations as the number of leaf nodes. This captures the generalization in (6a): A language has a costless optional movement (or shows flexible word order) only if it is head-final, and the operation produces a structure consistent with the head-final value (i.e., it is leftward).

Let us turn to wh-movement to see if the present theory can account for (6b): a language has a costly obligatory movement only if it is head-initial, and the operation is leftward. (33a,b) show that wh-movement must have phonetic effects in English, while (33c,d) are both allowed in Japanese:

(33) a. I don't know what John bought.

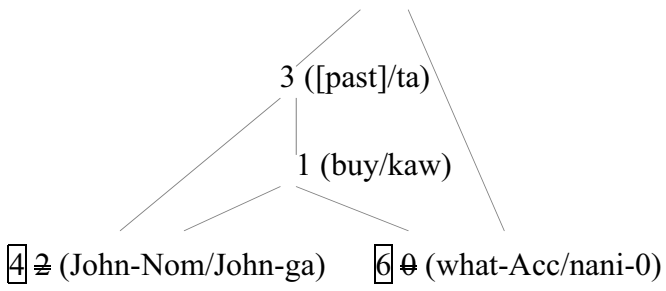
b. *I don't know John bought what.

c. watasi-wa John-ga nani-o kaw-ta ka sir-anai.
I-Top -Nom what-Acc buy-Past Q know-not

d. watasi-wa nani-o John-ga katta ka sir-anai.

According to the theory here, the embedded interrogative clauses of (33a,c,d) are analyzed as sharing the bilingual lexical graph in (34):

(34)



An inorder traversal starts from node 5. According to the larger-index-priority strategy, node 6 is chosen at node 5 as the next node to visit, and node 4 is chosen at node 3. Traversing (34) in this way and pronouncing its nodes according to (28) results in the sequence: 5-6-5-3-4-3-1. This corresponds to the embedded clause of (33a). The wh-

phrase is pronounced initially since it is directly connected to the root and has the largest index. It is also connected to the verb, but we do not go through he edge between them since it has become pronounced earlier in the traversal. This captures the overt property of English wh-movement. On the other hand, there are two postorder traversals, starting from the two leaf nodes. The embedded clause of (32c) is obtained by starting from node 4 and traversing in accordance with (30): 4-1-6-1-3-5. That of (33d) results if node 6 is the starting node: 6-1-4-1-3-5. In this traversal, node 6 is pronounced clause-initially on a par with the object R-expression in (2).

The differences in PF-interpretation have been attributed to the distinct set of edges traversed, as described below:

(35) a. English (5-6-5-3-4-3-1) b. Japanese (4-1-6-1-3-5/6-1-4-1-3-5)

(34) is not a tree, with two cycles: 3-1-4-3 and 5-6-1-3-5. (35a,b) are two of its spanning trees with the original root node retained. <1,4> and <1,6> have been introduced by external Merge and they are ignored in (35a).[10] In (35b), <3,4> and <5,6> are bypassed, which have been added by Move or internal Merge. Nodes are pronounced generally downward from the root in an inorder traversal and upwards from a leaf node in a postorder traversal. A moved phrase is connected to a functional head agreeing with it and a lexical head selecting it. The former is nearer the root than the latter. Since a given node is pronounced only once, the moved phrase is pronounced near the functional head in an inorder traversal, while it is pronounced near the lexical head in a postorder traversal. In this way, the overt/covert distinction of movement is contingent on the direction of traversal, which is a version of the head parameter assumed in the present theory. This can capture (6b).

According to the traditional analysis, the embedded clause of (33c) is the canonical

word order, while that of (33d) results from clause-internal scrambling of the object. Both are regarded here on a par, deriving from (34), which is also shared by the English sentence in (33a). Besides Saito's theoretical arguments, Miyamoto (2003) and others claim there is experimental evidence against this conclusion; a scrambled sentence is more difficult to process than a canonical one.[11] What this paper has been concerned with, however, is the contrast observed between (3) and (4) as well as the outright ungrammaticality of (32b) rather than a subtle processing difference between (33c,d), if any.

It is beyond the scope of this paper to explore all the properties of wh-movement, but the present theory can offer a simple account of the restrictions on piedpiping observed by Kayne (1994: 24-25). Consider the following examples:[12]

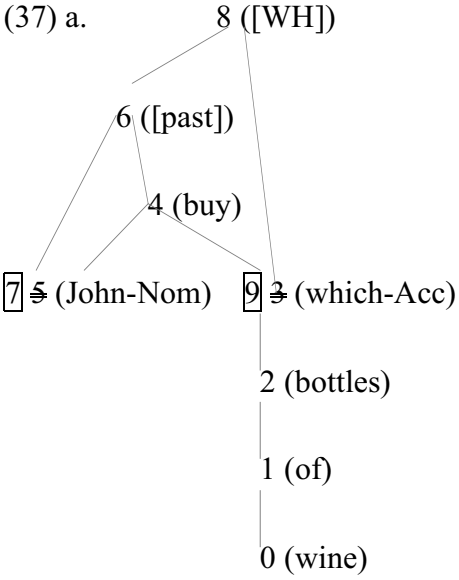
(36) a. I wonder which bottles of wine John bought.

b. *I wonder bottles of what John bought.

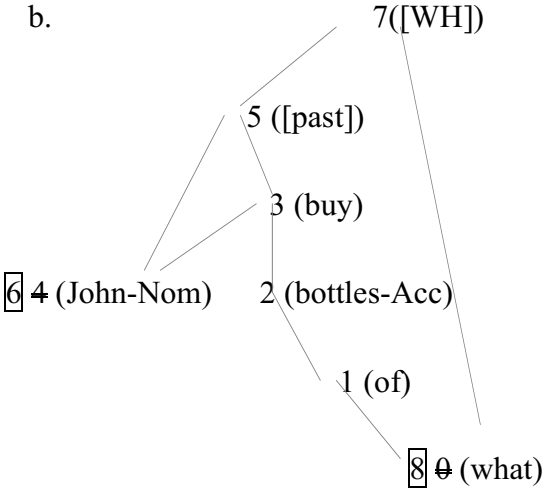
c. ?I wonder what John bought bottles of.

What in (33a) is replaced by more complex wh-phrases in (36a-c). A wh-word appears as the determiner head in (36a), while it is part of the complement in (36b,c). The embedded clauses of (36a) and (36b,c) are to be analyzed as (37a,b), respectively:

(37) a.



b.



Intuitively, wh-movement connects the functional category [WH] with the nearest node corresponding to a wh-word, which is *which* in (37a) and *what* in (37b). In (37a), the movement connects *which* to [WH] and changes the index of the former from 3 to 9; the

inorder traversal results in: 8-9-2-1-0-1-2-9-8-6-7-6-4. Note that all the nodes dominated by *which* continue to have the original indices after the wh-movement, but they are pronounced right after it in a depth-priority traversal. The embedded clause of (36c) is obtained similarly by the inorder traversal of (37c): 7-8-7-5-6-5-3-2-1. Since *what* does not dominate the rest of the object noun phrase, (36b) is not derivable.

As expected, a clause-initial wh-phrase in Japanese is not subject to the above restriction on piedpiping. A fronted wh-phrase can be complex and a wh-word can appear anywhere in it as follows:

(38) a. [dare-ga sagasitei -ta wain-o] John-wa kaw-ta ka.
who-Nom looking-for-pastwine-Acc -Top buy-past Q
Lit. 'The bottle of wine that who had been looking for did John buy ?'

b. [Mary-ga doko-de non-da wain-o] John-wa kaw-ta ka.
-Nom where-at drink-past wine-Acc -Top buy-past Q
Lit. 'The bottle of wine that Mary had drunk where did John buy ?'

The wh-phrases in (38) are pronounced initially not because they contain a wh-word but because they contain a leaf node . The functional category [WH] is presumably connected to the wh-word rather than the root of the whole initial constituent, but the edge in question is bypassed, unlike the cases in (33a).

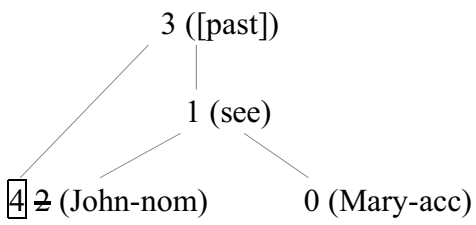
4.3 Formalization

To formalize the ideas presented in Section 4.2, it is necessary to revise standard data representations that are based on the left/right distinction of nodes such as (22). Remember that a lexical graph is defined as a pair of V and E in the previous section. The English version of the lexical graph in (27) is equivalent to (39):

(39) V={0,1,3,4}

E={<1,0>, <1,4>, <3,1>, <3,4>}

0=Mary-Acc, 1=see, 3=[past], 4=John-nom



It is natural to represent the structure of (39) in terms of the set E. (39) can be translated in C as (40) and (41):

(40) word[0].pfvalue="Mary-Acc"; word[0].pronounced=0;

word[1].pfvalue="see"; word[1].pronounced=0;

word[3].pfvalue="[past]"; word[3].pronounced=0;

word[4].pfvalue="John-Nom"; word[4].pronounced=0;

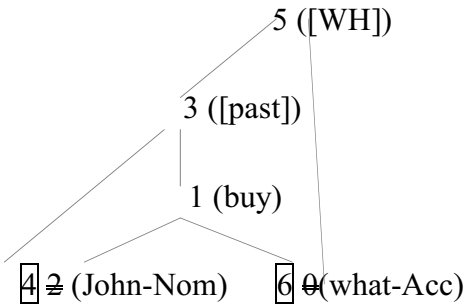
(41) rel[1][0]=1; rel[1][4]=1; rel[3][1]=1; rel[3][4]=1;

The members of V with their PF values are listed in (40), and they are initially unpronounced (=0). The ordered pairs in E correspond to the equations in (41), where 1 means that the relation holds. An obvious advantage of (41) is that information on both child and mother nodes is directly expressed; rel[1][0] means that node[1] is a mother node of node[0] and node[0] is a child node of node[1]. This enables a traversal to proceed naturally in both directions, from child to mother and from mother to child.

The English version of (34) and the three examples analyzed in (11), (15) and (18) are likewise represented as (42)-(45), respectively:

(42) (I don't know) what John bought

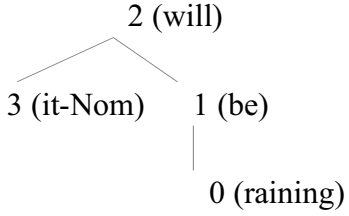
word[1]="buy";
word[3]="[past]";
word[4]="John-Nom";
word[5]="[wh]";
word[6]="what-Acc";



rel[1][6]=1; rel[1][4]=1; rel[3][1]=1; rel[3][4]=1; rel[5][3]=1; rel[5][6]=1;

(43) It will be raining.

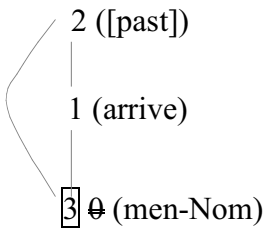
word[0].pfvalue="raining";
word[1].pfvalue="be";
word[2].pfvalue="will";
word[3].pfvalue="raining";



rel[1][0]=1; rel[2][1]=1; rel[2][3]=1;

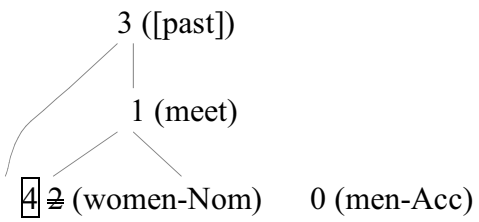
(44) Men arrived.

word[1].pfvalue="arrive";
word[2].pfvalue="[past]";
word[3].pfvalue="men-Nom";
rel[1][3]=1; rel[2][1]=1; rel[2][3]=1;



(45) Women met men.

word[0].pfvalue="men-Acc";
word[1].pfvalue="meet";
word[3].pfvalue="[past]";
word[4].pfvalue="women-Nom;



rel[1][0]=1; rel[1][4]=1; rel[3][1]=1; rel[3][4]=1;

Each node's pronunciation status is omitted above.

It has been discussed that the SHC ordering can be derived from a lexical graph according to (28), specifically controlled by the number of unpronounced nodes each node immediately dominates, together with the indexing of all its nodes. Let us call it hierarchical inorder traversal to distinguish it from the standard algorithm that is based on the linear distinction of child nodes. (28) can be formulated in C as in (46):

(46) Hierarchical Inorder Traversal

```
1.  void h_in_order(int p){
2.      int i, m=0, nextchild=-1;
3.      if(word[p].pronounced==0){              /* If word[p] has not been pronounced */
4.          for(i=0; i<=sp-1;i++)                          /* Increment counter i */
5.              if (rel[p][i] == 1 && word[i].pronounced==0){   /* Count word[p]'s child */
6.                  m = m + 1;                          /* nodes that have not been pronounced */
7.                  nextchild=i;}                       /* Store the unpronounced child node */
8.          if(m<=1){          /* If word[p] has no more than one unpronounced child node */
9.              printf("%s ",word[p].data);                        /* Pronounce word[p]  */
10.             word[p].pronounced =1; }                /* Mark word[p] as pronounced */
11.     h_in_order(nextchild);          /* Visit word[p]'s child node with the largest index */
```

12. h_in_order(p);}} /* Backtrack to word[p] */

The function *h_in_order* takes one argument *p*, which is the index of a current node. If word[p] has not been pronounced, it is subject to lines 4-11. They are divided into three major parts: the for-loop on lines 4-7, the conditional on lines 8-10, and calls to *h_in_order* on lines 11 and 12. The for-loop counts *m*, the number of unpronounced child nodes of word[p] and stores their indices in the variable *nextchild*. The value of *nextchild* is updated if another child node is found. When the for-loop is over, *nextchild* holds the largest index that satisfies the condition since the counter *i* in the for-loop is incremented up to *sp*-1, which is the largest index of the input lexical item.[13] This imposes the larger-index-priority traversal strategy locally at each node. The conditional on lines 8-10 says that if word[p] immediately dominates no more than one unpronounced node, word[p] is to be pronounced; otherwise, nothing is done. When word[p] is pronounced, its pronunciation status changes from unpronounced (=0) to pronounced (=1), which will change *m* of its mother node and enable it to be pronounced later. Line 11 guides the traversal to word[p]'s unpronounced child node with the largest index, and line 12 directs the traversal back to word[p]. The inorder traversals of (39), (42)-(45) are as follows:

(47) (39) => 3-4-3-1-0

(42) => 5-6-5-3-4-3-1

(43) => 2-3-2-1-0

(44) => 2-3-2-1

(45) => 3-4-3-1-0

Readers are invited to verify the results by running (46) with the initial values as specified in (42)-(45).[14]

The idea of hierarchical postorder traversal given in (30) can be implemented by replacing line 8 of (46) repeated as (48a) with (48b):

(48) a. if(m<=1)

b. if(m<=0)

(48b) says that each node is pronounced when it immediately dominates no unpronounced

nodes. (48a,b) correspond to (28) and (30), respectively. Two possible approaches to deduce multiple PF-interpretations have been pointed out in Section 4: (i) starting from the root and proceeding in either direction at each branching node and (ii) starting a traversal from any leaf node. I will attempt to formalize (ii). This calls for further revision of (47), since it does not let a traversal proceed from leaf node to mother node. Information on the mother nodes of each node can be obtained within the for-loop of (46) as (49):

(49) if (rel[i][p] == 1 && word[i].pronounced==0){
nextmother=i; }} /* Store the unpronounced mother node */

Let us name the function (46) modified with (48b) and (49) as *h_post_order* (hierarchical postorder traversal). Correspondingly, lines 11 and 12 of (46) are to be replaced by (50a,b):

(50) a. h_post_order(nextchild);
b. h_post_order(nextmother);

Changing (48b) back to (48a) in *h_post_order* just defined will give the function that is virtually the same as the original inorder algorithm given in (46). (48a,b) are two options of the head-parameter, and the rest is virtually the same between *h_in_order* and *h_post_order*. One more minor revision is needed on line 4 of (46), where the counter *i* is incremented to find the child node with the largest index. If the starting node is a leaf, which has a smaller index than those dominating it, it is natural to give priority to a node with a smaller index; nodes are traversed and pronounced basically in the ascending order of their indices. The counter *i* on line 4 of (46) should be incremented or decremented depending on the distance between the starting node and the root.

Going back to the Japanese version of in (39), which has been given in (27), applying *h_post_order* to its two leaf nodes, word[4] and word[0], produce (51a,b):

(51) a. 4-1-0-1-3
b. 0-1-4-1-3

(51a,b) recapitulate the conclusions in Section 4.2. The function *h_post_order* works properly with the other cases discussed so far.

5. Economy in Traversal

If multiple PF interpretations can be deduced from a single lexical tree, some of them should be more economical than the others in the sense that the former take fewer steps than the latter. This point can be illustrated by the following data:

(52) John says that Mary bought books.

(53) a. [John-ga [Mary-ga hon-o kaw-ta to] iw-ru]

NOM NOM book-ACC buy-Past COMP say-Present

'John says that Mary bought the book.'

b. [John-ga [hon-o Mary-ga kaw-ta to] iw-ru]

Lit. 'John says that the book, Mary bought.'

c. [[Mary-ga hon-o kaw-ta to] John-ga iw-ru]

Lit. 'That Mary bought the book, John says.'

d. [[hon-o Mary-ga kaw-ta to] John-ga iw-ru]

Lit. That the book, Mary bought, John says.'

The English sentence in (52) corresponds to the Japanese sentence in (53a) straightforwardly but (53b-d) are also possible without any special suprasegmental features. (53b-d) have been analyzed as involving clause-internal scrambling: (53b) by clause-internal scrambling of the embedded object and (53c,d) by scrambling of the embedded clause in front of the matrix subject in (53a,b), respectively. According to the present theory, (52) and (53a-d) share the bilingual representation below:

(54) word[0].pfvalue="books-Acc/hon-o"; 8 ([present]/ru)

word[1].pfvalue="buy/kaw"; 6 (say/iw)

word[4].pfvalue="Mary-Nom/Mary-ga"; 9 ~~7~~ (John-Nom/John-ga) 5 (that/to)

word[5].pfvalue="that/to"; 3 ([past]/ta)

word[6].pfvalue="say/iw"; 1 (buy/kaw)

word[9].pfvalue="John-Nom/John-ga"; 4 ~~2~~ (Mary-Nom/Mary-ga) 0 (books-Acc/hon-o)

word[8].pfvalue="[present]/ru";

rel[1][0]; rel[1][4]; rel[3][1]; rel[3][4]; rel[5][3]; rel[6][5]; rel[6][9]; rel[8][6]; rel[8][9];

The indices of the embedded and matrix subjects are reindexed from 2 and 7 to 4 and 9, respectively. The inorder traversal of (54) with word[8] as the starting node results in (55):

(55) 8-9-8-6-5-3-4-3-1-0 ==> (52)

(54) has three leaf nodes: word[0], word[4], and word[9]. Applying the function *h_post_order* to word[0], word[4], and word[9] eventually yield (56a-c), respectively. The remaining sentence in (53a) can be obtained by starting from word[9] and giving priority to a node with a larger index when path ambiguities arise, as in (56d):

(56) a. 0-1-4-1-3-5-6-9-6-8 ==> (53d)

b. 4-1-0-1-3-5-6-9-6-8 ==> (53c)

c. 9-6-5-3-1-0-1-4-1-3-5-6-8 ==> (53b)

d. 9-8-6-5-3-4-3-1-0-1-3-5-6-8 ==> (53a)

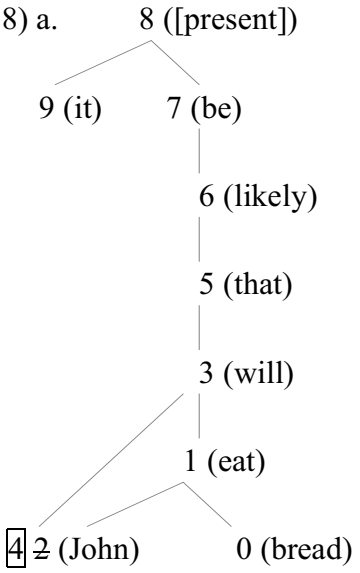
Clearly, the postorder traversals in (56d,c), which start from the leaf nearest to the root, take more steps in pronouncing all the nodes than those in (56a,b) and the inorder traversal in (55). This conclusion might be a problem since (53a) has been standardly supposed to be the basic word order. Actually, quite a few speakers prefer (53c,d) to (53a,b) due to the center-embedding in the latter. If a sentence involves two complement clauses, its 'basic' order according to the traditional analysis necessarily has three consecutive nominative phrases and double center-embedding, and it is extremely hard to understand. Preferred starting nodes in postorder, then, can be said to be those distant from the root, which have relatively small indices.

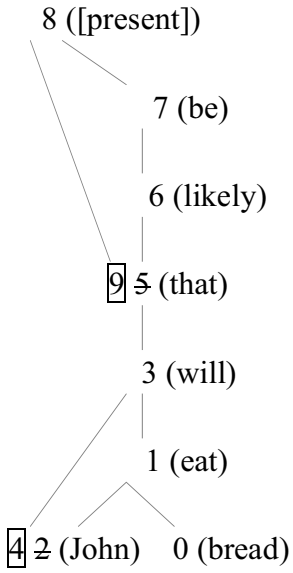
A similar contrast is found in the pair of English sentences in (57a,b):

(57) a. It is likely that John will eat bread.

b. That John will eat bread is likely.

As Koster (1978) claims, sentential subjects are non-existent or at least highly marked in SHC languages. The inorder traversal of (57a) is more economical than that of (57b) if they are analyzed as (58a,b), respectively:

(58) a. 

b. 

I assume in (58b) that the sentential argument is selected by the adjective *likely* and later raised into the specifier of the matrix tense (or some functional category like TOP). (58a,b) are traversed and pronounced as (59a,b), respectively:

(59) a. 8-9-8-7-6-5-3-4-3-1-0

b. 8-9-3-4-3-1-0-1-3-9-8-7-6

After all the nodes within the sentential subject are pronounced, the traversal in (59b) inevitably involves a few steps to return to the root and pronounce the matrix predicate, which is not economical.[15]

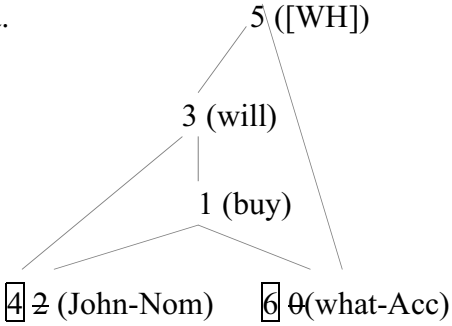
6. Head Movement and Other Remaining Issues

The conception of syntactic structure proposed in this paper is radically different from the standard one so that more problems have been produced than solved. One of them is head-movement. If head-movement is analyzed as involving two adjacent nodes, how is it defined graph-theoretically and how is its PF effect deducible? Move (or internal Merge) is defined as connecting two nodes that have not been connected in the structure formed. This definition does not apply to head-movement since it involves two nodes that have already been connected. I suggested in Yasui (2003) that head-movement might be analyzed as edge-contraction defined in graph theory, which changes the configuration radically.[16] An alternative is to remove the PF value of the moving head and append it to that of the target head without changing the graph configuration.

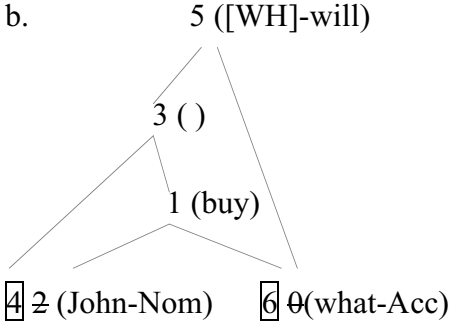
Let us explore the second possibility by applying it to the matrix English question in (60), which is to be analyzed as (61b):

(60) What will John buy?

(61) a.



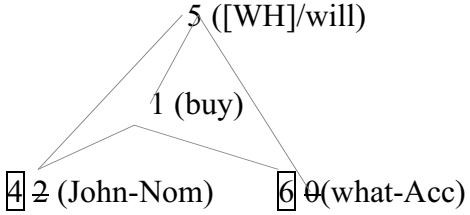
b.



Will is assumed to be moved into the position occupied by [WH]. Removing the PF value of node 3 and appending it to that of node 5 results in (61b). Since a target of head-movement is typically a functional category with no or little phonetic value like [WH] in (61), this conception of head-movement is reasonable. The sequence of nodes traversed and pronounced in (61b) is exactly the same as that in (61a): 5-6-5-3-4-3-1. Distinct PF-interpretations are obtained due to the differences in the PF value of nodes 3 and 5.[17]

If the idea of edge-contraction is applied to (61a), it is converted to (62):

(62)



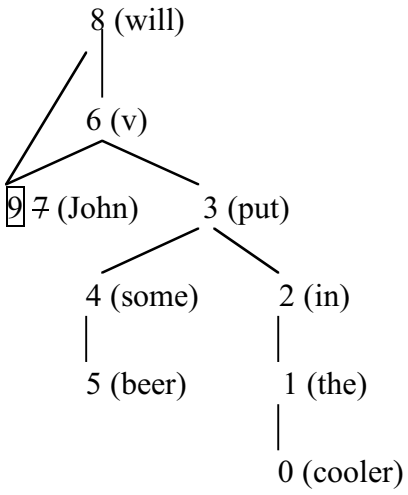
The edge <5,3> is contracted into node 5; subsequently, the nodes connected to node 3 are reconnected to node 5 and multiple edges are simplified. The inorder traversal of (62) produces the incorrect sequence as a matrix question: 5-6-5-4-5-1 (*What John will buy?). It is because node 5 has three child nodes, and it does not satisfy the pronunciation condition (i.e., m<=1) after node 6 has been pronounced. (62) should be rejected as an analysis of (60).

Head-movement as edge-contraction, however, might be a possibility for head-final languages. In Section 3, the transitive verb construction was analyzed without the light verb. Larson (1988) and Hale and Kayser assume the light verb for constructions involving three

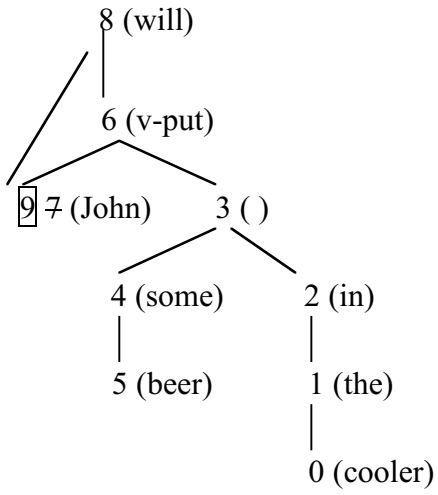
arguments like (63). (63) can be analyzed as (64a) here, with its details set aside:

(63) John will put some beer in the cooler.

(64) a.

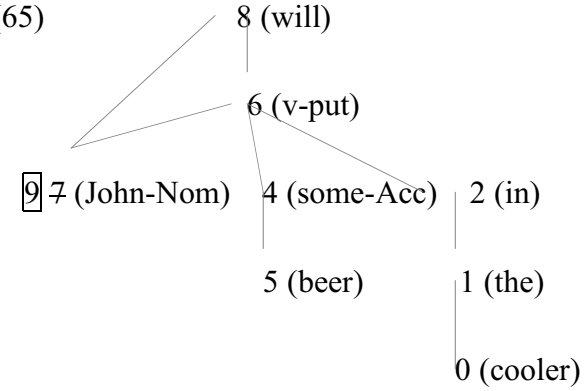


b.



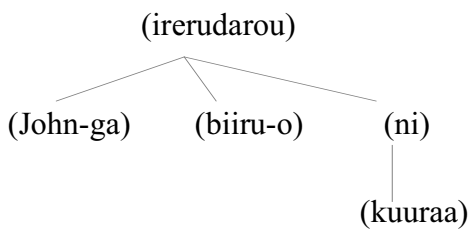
Removing the PF value of node 3 and appending it to that of node 6 results in (64b). The inorder traversal of (64b) correctly produces (63): 8-9-8-6-3-4-5-4-3-2-1-0. If the edge <6,3> in (64a) is contracted into node 6, the overall configuration is radically altered as follows:

(65)



Node 6 has become ternary-branching, which causes the inorder traversal of (65) to lead to the incorrect sequence, 8-9-8-6-4-5-4-6-2-1-0 (*John will some beer put in the cooler). The two internal arguments are sisters in (64b), to which the subject is added as the third sister in (65). The subgraph rooted by the contracted node 6 is quite similar to the flat structure assumed for Japanese by Hale (1980, 1981). Consider the Japanese counterpart of (63) given in (66):

(66)



Japanese and many other head-final languages are characterized as agglutinated in that a verbal base is affixed with a number of bound morphemes that semantically correspond to modal auxiliary verbs like *will*. The root in (66) has the meaning *will put*, and it presumably results from multiple edge-contractions to the Japanese counterparts of the edges consisting of *put, v* and *will* in (64). The three arguments are hierarchically non-distinct in (66), which appears better fitted to derive the six permutations of them given below:

(67) a. John-ga biiru-o kuuraa-ni irerudarou.
-Nom beer-Acc cooler-in will-put
'John will put beer in the cooler.'

b. John-ga kuuraa-ni biiru-o irerudarou.

c. biiru-o kuuraa-ni John-ga irerudarou.

d. kuuraa-ni biiru-o John-ga irerudarou.

e. biiru-o John-ga kuuraa-ni irerudarou.

f. kuuraa-ni John-gabiiru-o irerudarou.

The hierarchical postoder traversal algorithm proposed in Section 4 can derive (67a-d) from the highly configurational lexical graph corresponding to (64b) with some modifications, but it in principle cannot produce (67e,f). The crucial difference is that the two internal arguments are pronounced consecutively in the former but they are interrupted by the subject in the latter. If the flat structure in (66) is posited instead, all the six sentences in (67) are more naturally derivable from the hierarchical postoder traversal algorithm with slight modifications

Besides the treatment of head movement, quite a few issues remain to be explored. I will take up two of them. First, modified versions of the inorder and postorder tree traversal algorithms have been argued to yield SHC and SCH order of natural languages from common

lexical graphs, respectively. A question arises what languages, if any, are derived by preorder traversal. The root should be a natural starting node in a preorder traversal just as in an inorder traversal. Each node is pronounced in its first visit regardless of the number of unpronounced child nodes it has; so the conditional in line 10 of (40) is not necessary. If nodes are traversed strictly in the descending order of their indices as in an inorder traversal, the preorder traversal of a given lexical graph yields a single ordering of its nodes. Alternatively, since there is no restriction on the pronunciation timing, traversing of nodes itself might be unrestricted; it may proceed in any direction at a branching node. Then, more than one ordering would be derivable through the preorder traversal of a single lexical item tree. Bearing on Chung (1990), Fukui (1993: 414-416) argues that Chamorro is underlyingly VOS (or HCS) and its surface VSO order is derived by optional leftward adjunction of spec to the right of the verb, which does not destroy its head-initiality. The second conception of preorder traversal is consistent with Fukui's analysis of Chamorro.

Second, there are no places for adjuncts in the present theory, where a lexical item is connected to another item in terms of selection or agreement only. Lebeaux's analysis of introducing adjuncts by generalized transformation rather than base-generation is one possibility. I explore another possibility along the line of Phillips (2003) in Yasui (2004). I will leave the two issues and many others for future research.

References

Antonakos, J. L. and K. C. Mansfield, Jr. (2000) *Practical data structures using CC++*. Upper Saddle River, N.J.: Prentice-Hall.

Bach, Emmon (1971) "Questions." *Linguistic Inquiry* 2, 153-166

Balakrishnan, R. and K. Ranganathan (2000) *A textbook of graph theory.* New York, N.Y.: Springer.

Bresnan, Joan (1972) *Theory of Complementation in English Syntax*. Doctoral dissertation, MIT.

Chomsky, Noam (1994) *Bare phrase structure*. MIT occasional papers in

linguistics *5*. Department of Linguistics and Philosophy, MIT. Cambridge, Mass.

_____ (1995) *The minimalist program*.. Cambridge, MA: MIT Press.

_____ (1999) *Derivation by phase.* MIT occasional papers in linguistics 18. Department of Linguistics and Philosophy, MIT. Cambridge, Mass.

_____ (2000) Minimalist inquiries: the framework. In *Step by step: essays on minimalist syntax in honor of Howard Lasnik*, eds. Roger Martin, David Michaels, and Juan Uriagereka:89-155 Cambridge, MA: MIT Press.

_____ (2001) *Beyond explanatory adequacy.* MIT occasional papers in linguistics 20. Department of Linguistics and Philosophy, MIT. Cambridge, Mass.

Chung, Sandra (1990) VP's and verb movement in Chamorro. *Natural language & linguistic theory* 8:559-619.

Diestel, Reinhard (2000) *Graph Theory.* New York: Springer.

Farmer, A. (1980) *On the interaction of morphology and syntax*. Doctoral dissertation, MIT.

_____ (1984) *Modularity in Syntax: A Study of Japanese and Syntax*. Cambridge, MA: MIT Press.

Frampton, John and Sam Gutmann (1999). Cyclic computation, a computationally efficient minimalist syntax. *Syntax* 2, 1-27.

_____ (2000) Crash-proof syntax. " *Derivations and explanation in the minimalist program*, eds. Sam Epstein and Daniel Seely:90-105. Oxford and Malden, Mass.: Blackwell.

Fukui, Naoki (1993) "Parameters and Optionality." *Linguistic Inquiry* 24, 399-420.

Hale, Kenneth (1980) Remarks on Japanese Phrase Structure: Comments on the Papers on J apanese Syntax. In *Theoretical Issues in Japanese Linguistics : MIT Working Papers in Linguistics* Vol. 2, eds. Yukio Otsu and Ann Farmer. MIT.

_____ (1981) *On the Position of Warliri in a tTpology of the Base*. Bloomington, IN: IULC.

Haegeman, L. (1994) *Introduction to Government and Binding Theory.* Oxford and Cambridge, Mass: Blackwell.

Holte, Robert C.(2002) Data Structures Course Lecture Notes. University of Ottawa (http://www.csi.uottawa.ca/~holte/T26/top.html)

Hopcroft, John E. and Jeffrey D. Ullman (1979) *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Ikawa, Hajime (1996) *Overt Movement as a Reflex of Morphology.* Doctoral dissertation, University of California, Irvine.

Kayne, R. S. (1984) *Connectedness and binary branching*. Dordrecht: Foris.

_____ (1994) *The antisymmetry of syntax*. Cambridge, Mass: MIT Press.

_____ (2003) Antisymmetry and Japanese. *English Linguistics* 20:1-40

Koster, J. (1978) "Why subject sentences don't exist." In Kayer, S. J. (ed.) *Recent Transformational Studies in European Languages*. Cambridge, Mass: The MIT Press.

Larson, Richard (1988) On the distribution of object constructions. *Linguistic Inquiry* 19:335-392.

_____ (1990). Double objects revisited: reply to Jackendoff. *Linguistic Inquiry* 21: 589-632.

Lebeaux, David (1988) *Language acquisition and the form of the grammar.* Doctoral dissertation, University of Massachusetts, Amherst.

Lopez, L. (2001) "On the (Non)complementarity of q-theory and Checking Theory." *Linguistic Inquiry* 32 (4), 694-716.

Marantz, A (1984) *On the Nature of Grammatical Relations*. Cambridge, Mass: The MIT Press.

Miyamoto, Edson T. (2003). Processing alternative word orders in Japanese. Ms.

Phillips, Colin (2003) Linear order and constituency. *Linguistic inquiry* 34:37-90.

Sahni, Sartaj (2001) *Data Structures, Algorithms and Applications in JAVA*. McGraw-Hill

Saito, Mamoru (1985) *Some Asymmetries in Japanese and Their Theoretical Implications*. Doctoral dissertation, MIT.

_____ (1992) Long Distance Scrambling in Japanese. *Journal of East Asian Linguistics* 1, 69-118.

Saito, M. and N. Fukui (1998) Order in phrase structure and movement. *Linguistic Inquiry* 29: 439-474.

Takahashi, Daiko (1993). Movement of wh-phrases in Japanese. *Natural language & linguistic Theory* 11: 655-678.

Yasui, Miyoko (2002) A graph-theoretic reanalysis of bare phrase structure and wh-movement. Paper presented at Linguistics and Phonetics: Typology and Linguistics Theory. Meikai University. September 2002.

_____ (2003) A graph-theoretic reanalysis of bare phrase structure theory and its implications on parametric variation. In *Proceeding of LP2002 On-Line*. (http://www.wata-net.com/proceedings/MiyokoYasui/yasui.pdf) Meikai University, Urayasu. Also available at http://kasiwagi.dokkyo.ac.jp/~myasui/general/ profile/profile.html

_____ (2004) Syntactic structure without projection labels. Ms., Dokkyo University, Soka. (http://kasiwagi.dokkyo.ac.jp/~myasui/general/ profile/publications/nolabel.pdf)

Department of English
Faculty of Foreign Languages
Dokkyo University
1-1 Gakuencho, Soka
Saitama 340-0042
myasui@dokkyo.ac.jp

---

[1] In Kayne (1994), a case-particle is analyzed as a head taking a sentential complement, from which a DP moves to its specifier position. A postposition is assumed to be a head taking a DP complement, which moves to its specifier position in Kayne (1994) or the specifier position of some higher abstract head in Kayne (2003). In this way, head-final languages are not analyzed as mirror-images of head-initial languages.

[2] As Takahashi (1993) shows, long-distance scrambling is semantically non-vacuous. I

will deduce cases of local scrambling from the structure that underlies the so-called basic word order sentence, assuming that long-distance scrambling is triggered by some functional category that does not exit in the underlying structure of its non-scrambled counterpart.

[3] If heavy NP shift resists for multiple applications, the possibility mentioned by Fukui (1993: note 10), it differs from scrambling in this respect, which can apply iteratively.

[4] If V in (15) continues to have node 0, this does not matter, as <1,0> in E has been replaced by <1,3>.

[5] The subject/object (or external/internal) argument asymmetry has been amply justified by Marantz (1984) and subsequent work. A more refined version of (16) is given by Larson (1988, 1990). (16) should not be regarded as a syntactic constraint but as originating in our lexical knowledge

[6] Frampton and Gutmann (1999, 2000) do not discuss examples involving Ds such as (i):

(i) The woman met the man.

An N should be introduced before a D in the case of the object but a D should be introduced before an N in the subject according to their theory. This difference in indexing does not cause any problem in my theory.

[7] See Antonakos and Mansfield (2000) and others.

[8] In the common inorder algorithm, nodes of a left-linear subgraph are pronounced from the leaf toward the root but those of a right-linear subgraph are pronounced in the opposite direction. In the inorder traversal proposed here, nodes are pronounced from the root in both cases.

9 Thus, (10i,ii) are not necessary; (16) as an extension of (10iii) is crucial in distinguishing spec from complement. See Yasui (2004).

[10] If a node with a smaller index is given priority, the postorder traversal of (34) that starts from the root produces another sequence: 5-3-1-4-1-6-1-3-5. This corresponds to the embedded clause of (33c) and the set of edges traversed is the same as (35a).

Ignoring edges such as <3,4> and <1,6> produces other spanning trees of (34). They

appear to correspond to postorder traversals that start from the root and proceed either way at the three branching nodes of (34). Removing <3,1> and <5,4> yields a non-tree; node 5 can no longer dominate nodes 1 and 6.

[11] They use declarative sentences without wh-phrases.

[12] Kayne (1994: 24) discusses the contrast between *whose article* and *articles by who(m)*. If *whose* is analyzed as a determiner rather than the specifier of the genitive *-s*, the account offered in the text applies to Kayne's examples.

[13] (46) is part of the larger algorithm that contains those mentioned in note 5. The variable *sp* appears in the latter. The complete script of *h_in_order* and that of *h_post_order* to be given shortly, with the structure-building system, are available at http://kasiwagi.dokkyo.ac.jp/~myasui/general/profile/profile.html.

[14] (38), for instance, has the following form in C:

```
...
int sp=4;
struct tfield word[4]={{"men-Nom",0},{"arrive",0},{"will",0},{"men-Nom",0}};
int rel[4][4]; ...
main(){ rel[1][3]=1; rel[2][1]=1; rel[2][3]=1; ...
```

"Men-Nom" is listed twice as word[0] and word[3], but it is not problematic since index 0 does not appear in any of rel[i][j]s. See note 3.

[15] There are also several returning steps at the end of the sequence in (59a), but they can be eliminated easily by stopping the traversal immediately when all the nodes have been pronounced

[16] See Diestel (2000: 16) and others.

[17] "What did John buy?" is likewise derivable if *did* is base-generated like *will* in (61).