Modeling Q-feature movement in Japanese

Jason Ginsburg
Sandiway Fong
University of Arizona

We discuss how we use the Japanese version of PAPPI, a multilingual parsing engine in the Principles-and-Parameters framework (Chomsky 1981), to computationally model a theory that accounts for the grammaticality of certain Japanese *wh*-constructions. In this theory, a Q-feature is base generated within a *wh*-phrase and raises to C, where it checks an uninterpretable Q-feature. Q-feature movement can be blocked by an intervening quantificational head; an effect due to the Minimal Link Condition (Chomsky 1995). Furthermore, certain multiple *wh*-questions in which Q-feature movement is not subject to the Minimal Link Condition are accounted for in terms of the Principle of Minimal Compliance (Richards 2001), which is basically the notion that once a constraint is satisfied it may be subsequently ignored. In this paper, we describe modifications to the parsing engine of PAPPI necessary to implement this Q-feature movement theory.

1. Introduction

In this paper, we use a computational parser to model a theory that accounts for the distribution of certain *wh*-questions in Japanese. In this theory, a Q-feature associated with a *wh*-phrase that has scope must raise to C (cf. Ginsburg 2006). The grammaticality of *wh*-constructions relies on whether or not this Q-feature movement is able to occur.

A direct computational implementation can formally validate certain important properties of a linguistic theory. First, a working computer implementation, i.e. one where legitimate derivations do not crash, demonstrates that the theory is formally sound. Furthermore, an implementation of a theory within a larger framework, as is the case described in this paper, demonstrates that the theory is logically consistent and compatible with the already implemented framework.

We utilize PAPPI, a multilingual parsing engine in the Principles-and-Parameter framework (Chomsky 1981). At the heart of the parser is a core set of pre-defined principles expressing structural constraints from Government-and-Binding (GB) theory as described in Lasnik and Uriagereka (1988), together with parameterized extensions specific to various languages. The parsing engine is implemented in Prolog, a Horn clause logic-based programming language originally designed for natural language processing (Colmerauer et al. 1973).

Coyote Papers 15, 18-39.
Psycholinguistic and Computational Perspectives

Computation in this framework proceeds by first constructing candidate phrase structures for input sentences in accordance with X'-theory plus the effects of (phrasal and head) movement through substitution and adjunction. Next, the ill-formed structures and their counterparts at various levels of syntactic representation (D-structure, S-structure and LF) are subsequently eliminated by parser operations that implement the GB structural constraints defined at each level. Surviving LF structures are deemed to be well-formed parses.

We assume as our starting point the Japanese language version of PAPPI, described in Fong (2001), that implements the basic language features such as head-final word order and scrambling, plus a variety of specific constraints and phenomena such as anti- superiority, indirect passives and potentials, *o*/*ni*-causative and dative subject constructions, and the double-*o* constraint. In this paper, we describe the modifications to the parsing engine necessary to implement the Q-feature movement theory.

2. Why *wh*-phrases can remain in-situ

In a Japanese matrix *wh*-question, a *wh*-phrase may remain in-situ, as shown in (1). Miyagawa (2001) accounts for this fact by arguing that Q-feature movement allows a *wh*-phrase to remain in-situ.

(1) Kare-ga **dare-ni** atta no?
he-NOM who-DAT met Q
'Who did he meet?'

Miyagawa follows the proposal by Hagstrom (1998) that the Japanese Q-particle can be base generated next to a *wh*-phrase and then raise to C, where it satisfies an uninterpretable Q-feature. According to this proposal, in Japanese, a *wh*-word contains a *wh*-feature and a question particle contains a Q-feature, so the two features are contained on separate lexical items. The question particle then moves to C, where it checks an uninterpretable Q-feature, as shown below (adapted from Miyagawa 2001: 314).

(2) $[_{CP}\ [_{C}\text{'} \ldots [_{vP}$ *wh*-phrase $t_1]$ ... $ka_1]]$

Miyagawa argues that movement to CP in both Japanese and English is motivated by the EPP-feature, in accord with Chomsky (2000). That is, in English the EPP motivates movement of a *wh*-phrase to [Spec, CP], and in Japanese, the EPP motivates head movement of a Q-particle. Notably, the idea that the EPP can

motivate head movement differs from the traditional notion that the EPP motivates XP movement.[1]

Because movement of a Q-particle satisfies the Q-feature in C, there is no need for overt *wh*-movement in Japanese.[2] For example, in (3) below, the Q-feature in Comp is checked by overt movement of the Q particle *no* and the *wh*-phrase *nani-o* 'what-ACC' remains in-situ.

(3) [$_{CP}$ [[$_{TP}$ John-ga$_1$ [vP t_1 [nani-o **t $_2$**] katta]] **no$_2$**]]?
John-NOM what-ACC bought Q
'What did John buy?' (Miyagawa 2001)

In summary, Miyagawa and Hagstrom provide an explanation for why there does not need to be overt *wh*-movement in Japanese; a Q-particle checks a Q-feature in Comp and a *wh*-feature does not need to be checked by overt movement.

3. Data

Evidence for Q-feature movement can be seen in constructions in which Q-feature movement appears to be blocked. Notably, *wh*-constructions in Japanese do not appear to show NP-island effects, whereas they show *wh*-island effects. *Wh*-island effects can be ameliorated with the addition of a *wh*-phrase outside of an embedded clause. Furthermore, constructions in which a *wh*-phrase is c-commanded by a quantifier or negative polarity item (NPI) are ill-formed, but well-formedness results when the *wh*-phrase is scrambled. The Q-feature movement theory accounts for these types of constructions.

A *wh*-phrasal argument contained within a complex NP can obtain matrix scope. For example, in (4), the *wh*-phrase *dare-o* 'who-ACC' has scope over the matrix clause.

(4) Kimi-wa [$_{DP}$ **dare-o** egaita hon]-o yomimashita ka?
you-TOP who-ACC described book-ACC read Q
'You read a book such that it described who?' (Nishigauchi 1999:274)

[1] See Alexiadou & Anagnostopoulou (1998) for an argument that the EPP can be satisfied by head movement.

[2] This leaves open the issue of why a *wh*-phrase does not need to move to check a *wh*-feature. One possibility, argued for by Miyagawa (2001), is that in Japanese, a *wh*-feature is checked on T and movement of a *wh*-phrase to [Spec, TP] can check a *wh*-feature. However, a *wh*-phrase does not have to move if a non-*wh*-DP satisfies the EPP feature on T.

Notably, there is no NP-island effect in this construction,[3] unlike in English. In (5), a *wh*-phrase moves out of an NP-island, resulting in ungrammaticality.

(5) ??**Who$_1$** is he reading [$_{DP}$ a book that criticizes **t$_1$**]? (Watanabe 2003:205)

We argue that in the Japanese example (4), the Q-feature associated with *dare-o* 'who-ACC' is able to raise to the matrix Comp because there is no intervening element in the embedded Comp of the relative clause to block its movement.

Wh-island effects appear to occur in certain Japanese constructions such as (6).[4]

(6) ??[$_{CP}$ John-wa [Mary-ga **nani-o** katta ka-doo-ka] Tom-ni
John-TOP Mary-NOM what-ACC bought whether Tom-DAT
tazuneta no]?
asked Q
'What did John ask Tom whether Mary bought?' (Watanabe 2003:208)

In this case, movement of the Q-feature associated with *nani-o* 'what-ACC' is blocked by the intervening *ka-doo-ka* 'whether' in the embedded C. Assuming that *ka-doo-ka* 'whether' contains a Q-feature, ill-formedness can be accounted for under the Minimal Link Condition (MLC) (Chomsky 1995), which requires attraction of the closest element of the relevant type.

(7) Minimal Link Condition (MLC)
K attracts α if there is no β, β closer to K than α, such that K attracts β. (Chomsky 1995:311)

In (6), the Q-feature associated with *ka-doo-ka* 'whether' in the embedded Comp is closer to the matrix Comp than is the Q-feature associated with the *wh*-phrase. Therefore, the MLC prevents the Q-feature associated with *nani-o* 'what-ACC' from moving to the matrix C, and (6) cannot be interpreted as a matrix *wh*-question.

When there is an additional *wh*-phrase in the matrix clause of a construction such as (6), a *wh*-island effect can be avoided; this is a phenomenon known as an

[3] Note that there appear to be NP-island effects in Japanese *wh*-constructions in which a *wh*-adjunct occurs within an NP-island. See Saito (1994), Maki (1995), Richards (2000), among others for discussion of this phenomenon.

[4] Watanabe (1992b:12) writes that "the degree of unacceptability" of this type of construction "varies among different speakers."

additional *wh*-effect (Saito 1994; Hagstrom 1998). For example, (8) is identical to (6) except that *Tom* in the matrix clause has been replaced by the *wh*-phrase *dare* 'who'. Notably, the *wh*-island effect of (6) disappears, as shown in the following example.[5]

(8) [$_{CP}$ John-wa [Mary-ga **nani-o** katta ka-doo-ka] **dare-ni**
John-TOP Mary-NOM what-ACC bought whether who-DAT
tazuneta no]?
asked Q
'Who did John ask whether Mary bought what?' (Watanabe 2003:208)

Both *wh*-phrases can have matrix scope, which is an indication that something associated with the *wh*-phrase *nani-o* 'what-ACC' is able to move out of the embedded clause.

The well-formedness of (8) can be accounted for in terms of the Principle of Minimal Compliance (PMC), defined below, which is basically the notion "...that a given constraint only has to be satisfied once in a certain domain (Richards 2001:197)".

(9) Principle of Minimal Compliance (PMC)
(a) If the tree contains a dependency headed by H which obeys constraint C, any syntactic object G which H ''immediately c-commands" can be ignored for purposes of determining whether C is obeyed by other dependencies.
(b) A immediately c-commands B iff the lowest node dominating A dominates B and there is no C such that A asymmetrically c-commands C and C asymmetrically c-commands B. (Richards 2001:199)

In (8) the Q-feature associated with *dare-ni* 'who-DAT' raises to the matrix Comp to check an uninterpretable Q-feature, and this Q-feature is overtly pronounced as *no*. Movement of *no* 'Q' satisfies the MLC because there is no intervening head to block its movement. Furthermore, *no* 'Q' heads a well-formed dependency. Since *no* 'Q' is c-commanded by C, as it is in C, it may be ignored with respect to the

[5] Although Japanese allows the word order in which an indirect object precedes a direct object, for some speakers, a construction in which an additional *wh*-phrase c-commands a *wh*-island is ill-formed (cf. Watanabe 1992 a, b, Saito 1994, Richards 2001, among others). For example, whereas (8) is fine when the *wh*-island c-commands the additional *wh*-phrase *dare-ni* 'who-DAT', ill-formedness results when *dare-ni* 'who-DAT' c-commands the *wh*-island.

MLC. The matrix Comp may then attract a lower Q-feature. The next closest Q-feature is the Q-feature associated with *ka-doo-ka* 'whether' in the embedded C. However, this Q-feature already satisfies the uninterpretable Q-feature in the embedded Comp and so it has no reason to move. Therefore, the matrix Comp is able to attract the lower Q-feature associated with *nani-o* 'what-ACC'.

Notably, ill-formedness results in certain *wh*-constructions in which a negative polarity item (NPI) or quantifier c-commands a *wh*-phrase; this is a phenomenon known as an intervention effect (Beck & Kim 1997). In (10a), the NPI *sika* 'only' c-commands the *wh*-phrase *nani-o* 'what-ACC', and in (11a), the quantifier phrase *daremo-ga* 'everyone-NOM' c-commands the *wh*-phrase *nani-o* 'what-ACC'; both constructions are ill-formed. Well-formedness results when the *wh*-phrase is scrambled over the quantifier or NPI, as shown in the (b) examples.

(10) (a) *?Hanako-**sika** **nani-o** yoma-**nai** no?
Hanako-only what-ACC read-NEG Q

(b) **Nani-o$_1$** Hanako-**sika** **t$_1$** yoma-**nai** no?
what-ACC Hanako-only read-NEG Q
'What did only Hanako read?' (adapted from Tanaka 1999, per Pesetsky 2000, per Karimi & Taleghani, to appear)

(11) (a) *? **Daremo-ga** **nani-o** katta no?
everyone-NOM what-ACC bought Q

(b) **Nani-o$_1$** **daremo-ga** **t$_1$** katta no?
what-ACC everyone-NOM bought Q
'What did everyone buy? (Watanabe 2003:215)

In these constructions the NPI and quantifier appear to block Q-feature movement. This can be accounted for if a quantifier feature associated with an NPI or quantifier blocks Q-feature movement. When the *wh*-phrase is scrambled over the NPI or quantifier, well-formedness results, indicating that scrambling of a *wh*-phrase over an intervening quantificational element allows a Q-feature to move to Comp without an MLC violation occurring.

4. Implementation

The standard Japanese PAPPI implementation (Fong 2001) employs LF movement of quantificational NPs to positions in Comp indicating scope. We altered PAPPI to

implement the Q-feature movement theory described here, in which quantificational NPs remain *in situ* and LF scope is indicated by Q-feature movement to the head of Comp.

4.1 The lexicon

Q-feature movement applies to nouns that have a lexical feature *qf(q)*. The lexical entries for *nani* 'what', *dare* 'who' and *doko* 'where' are shown in (12).

(12) *lex(nani,n,[a(-),p(-),agr([[],[],n]),wh(+),qf(q)]). % what*
lex(dare,n,[a(-),p(-),agr([[],[],[m,f]]),wh(+),qf(q)]). % who
lex(doko,n,[a(-),p(-),agr([[],[],n]),wh(+),qf(q)]). % where

For example, the lexical entry for *nani* 'what' first identifies it as a noun (*n*) to the parser. The category label *n* is followed by a list of lexical features to be inserted into phrase structure at parse time. These include Binding theory-relevant features [-anaphoric] *(a(-))* and [-pronominal] *(p(-))*, formal agreement (*agr*), the LF-feature *wh(+)*, and the feature that triggers Q-feature movement *(qf(q))*. In the course of parsing, the Q-feature for a *wh*-NP such as *nani* 'what' will move to a compatible element in Comp. This will be simulated by adding a copy of *qf (q)* to Comp and deleting the original in the lexical feature array for *nani* 'what'.
Lexical entries for Japanese complementizers such as the question particles *ka* 'Q' and *ka-doo-ka* 'whether' are given in (13). Compatibility with respect to Q-feature movement will be checked by a Spell-Out condition (to be described later).

(13) *lex(ka,c,[wh(+),yn]). % Q*
lex(no,c,[wh(+),yn])]). % only in matrix clause
lex(ka_doo_ka,c,[wh(+),merge(yn)]). % Q - yes/no (whether or not)}
%lex(to,c,[wh(-),selR([not(feature(inf(_)))])]). % that

The complementizer *ka* can be ambiguously interpreted as either a [+wh] question particle (indicated by *wh(+)*) or a yes/no question particle (indicated by *yn*). When it is used as a [+wh] question particle (in computational terms, this occurs when Q-feature movement has taken place to *ka*), the feature *yn* will be cancelled, indicating that the yes/no question interpretation is no longer available. Similar considerations apply to matrix clause-only *no* 'Q' and *ka-doo-ka* 'whether'.

Also subject to feature movement at LF are lexical items that are marked with feature *qf(quant)*. For example, quantifiers such as *daremo* ‘everyone’ will possess the lexical feature *qf (quant)* as shown in (14).

(14) *lex(daremo,n,[a(-),p(-),agr([[],[],[m,f]]),op(+),qf(quant)]).*
% everyone

Similarly, nouns marked by the (suffix) particle *-sika* ‘only’ will have *qf(quant)* added to their lexical entries.

(15) *lex(sika,mrkr,[left(n,[],[qf(quant)])]).*

In (15) *-sika* ‘only’ has the category label *mrkr* (marker). Markers constitute a special category of PF-only elements in PAPPI. Unlike regular lexical or functional categories, markers do not project any phrase structure; instead, they are assumed to be affixes thatare the Spell-Out, i.e. PF realization, of one (or possibly more) features from the lexical entry of the head that they attach to.[6] The rest of the lexical entry in (15) specifies that, during parsing, *-sika* disappears after marking the noun (*n*) to its left by appending the feature *qf(quant)* to the noun’s lexical entry.

4.2 Q-feature movement

The parser operation *moveQfeature* that implements Q-feature movement is defined as follows:

(16) *moveQfeature in_all_configurations CP*
where cat(CP,c2) then moveQ(CP).

(16) states a universally-quantified condition that is applied to all candidate phrase structures considered by the parser. It states that *moveQ* must be applied to every (sub)tree *CP* that has category label *c2*. *moveQ* is defined as follows:

[6] A canonical case of feature realization is genitive Case realization in examples such as *proud of him* (Chomsky 1986a). In the current implementation, this example is handled by having *of* be a marker that realizes the feature *case(gen)* instantiated by genitive Case assignment from the predicate *proud* to the object *him*.

(17) *moveQ(CP) :- % optionally iterates if NP[Q] is found*
X complement_of CP,
C0 head_of CP,
(addDeleteQ(C0,NP,F), moveQ(CP)
; true)
if findQ(NP,X,C0,F).

(17) states that if some NP with a Q-feature of type *F* (*F = q* or *quant*) can be found, i.e. if *findQ* is true, in a search beginning with the subtree *X* (the complement of CP), the parser may carry out one of two possible operations:

1. Either, perform the Q-feature movement (*addDeleteQ*) and then recursively apply the operation *moveQ*, or (*;* = logical disjunction)

2. do nothing (*true*), i.e. do not perform Q-feature movement.

In other words, Q-feature movement is optional and may be iterated (if more than one NP with a Q-feature can be found) at this point.

4.2.1 *addDeleteQ*

addDeleteQ performs the actual operation of adding and removing features to simulate *q*/*quant*-feature movement.

(18) *addDeleteQ (C0,NP,F) :-*
spelloutCheck(F,C0),
deleteQ(NP,F).

spelloutCheck(quant,C) :-
\+ C has_feature yn,
addFeature(qf(quant),C).
spelloutCheck(q,C0) :-
addFeature(goal(yn,fail),C0),
addFeature(qf(q),C0).

There are two cases for *addDeleteQ* to consider.

1. In the case when *F=q*, *addDeleteQ* in (17) simulates Q-feature movement by adding the feature *qf(q)* to the head of Comp (*C0*) (via

spelloutCheck) and removing it (via *deleteQ*) from the NP identified by *findQ*. After Q- feature movement has taken place, it also simulates the deletion of any yes/no question (*yn*) interpretation for *C0*.

2. In the case when *F=quant*, *spelloutCheck* first checks to see that *C0* has no yes/no question feature (*yn*). Coupled with the behavior of *addDeleteQ* when *F=q* and the fact that the complementizer *ka* has feature *yn*, this implies that *quant* feature movement cannot precede *q* feature movement. This forces obligatory scrambling of a *wh*-phrase above a quantifier. (For example, compare the screenshots in Figures 7 and 8).

The logic engine at the heart of PAPPI allows features to be created in the course of derivation but does not permit already-created features to be destroyed (for reasons concerning logic consistency). Thus *deleteQ* in (19) simulates Q-feature removal by adding an extra-grammatical feature *goal* that is triggered whenever *qf(F)* is referenced and returns *fail*.

(19) *deleteQ(NP,F) :- addFeature(goal(qf(F),fail),NP).*

In other words, after the execution of *deleteQ*, all accesses to *qf(F)* will fail, i.e. as if *qf(F)* no longer existed. This shows up in the screenshots in section 5, as a matched pair of features, e.g. *qf(f)* and *goal(qf(f),fail)*, indicating that the relevant feature, e.g. *qf(f)*, has been cancelled.

4.2.2 *findQ*

For completeness, the operation *findQ* that digs for a Q-feature NP is given in (20). The predicate *findQ* holds if a quantificational NP can be identified during a recursively-defined search of the complement of the CP that it is presented with (by *moveQ*).

(20)
```
findQ(NP,NP,_,F) :- % findQ is local to the CP
        quantificationalNP(NP,F).
findQ(NP,CF,C0,F) :-
        \+[7] stopFindQ(CF,C0),
        CF has_constituent CF2,
        findQ(NP,CF2,C0,F).
```

[7] \+ is the negation operator in Prolog.

findQ digs for a quantificational NP, roughly speaking NPs that require LF scope identification, defined in (21) below as any NP with category label *np* and lexical feature *qf(_)*. *Wh*-phrases such as *dare* 'who' in (12), quantifiers such as *daremo* 'everyone' in (14), and *sika*-marked nouns will satisfy these requirements.

(21) *quantificationalNP(NP,F) :-*
cat(NP,np),
NP has_feature qf(F).

4.2.3 *stopFindQ*

The stopping condition for the search is given by the predicate *stopFindQ* in (22).

(22) *stopFindQ(CF,C0) :-*
cat(CF,c2), % blocks at lower CP boundary
C head_of CF,
\+ ec(C),
\+ C0 has_feature qf(_).

Normally, the search does not proceed beyond the local CP boundary, i.e. the first matching condition is that the category label is CP (*c2*). Two other possibilities are encoded here:

1. If the complementizer is an empty category, i.e. if *ec(C)* is true, as in the case of (Japanese) relative clause, *stopFindQ* fails and search may continue down into the relative clause itself.

2. If the complementizer fails the empty category test, e.g. in the case of the overt question particle *ka*, *stopFindQ* checks to see if feature movement has already taken place to Comp, encoded by the test *C0* has feature *qf(_)*. If *qf(_)* is present, this licenses *findQ* to dig inside the clause.

4.3 Check Q-feature

The *moveQ* parser operation described in the previous section implements Q-feature movement as optional, iterated feature addition and deletion. The *CheckQ* operation specified in (23) below is applied after *moveQ* as a well-formedness

check to ensure that all quantification NPs have undergone feature movement to determine scope.

(23) *checkQ in_all_configurations CF*
where quantificationalNP(CF,_) then noQF(CF)
else whMinusComp(CF) then noQF(CF).

whMinusComp(C) :- cat(C,c), C has_feature wh(-).

noQF(NP) :- \+ NP has_feature qf(_).

Separating the movement operation from the check allows the parser to (conveniently) represent (and display) both ill-formed and well-formed structures (see section 5 for examples.) Taken together, the two operations constrain the admissible LF phrase structures to only those that satisfy the Q-feature theory.

The universally quantified *CheckQ* condition triggers a search for quantificational NPs, defined previously in (21), for all phrase structures to be examined by the parser. Each of these NPs must satisfy the condition *noQ*, which states that the NP must not have a *qf(_)* feature. Since the lexical entries of the NPs in (12)-(14) all contain the *qf(_)* feature, the absence of the feature indicates that feature movement has taken place. Finally, *checkQ* also imposes the same condition on all [-wh] complementizers (given by *whMinusComp*).

5. Example parses

5.1 Long distance Q-feature movement

Figure 1 illustrates the parse produced for (4), repeated below as (24), when a *wh*-phrase *dare-o* ‘who-ACC’ obtains matrix scope despite originating from within an object relative clause (headed by *hon-o* ‘book-ACC’). The predicate *stopFindQ*, given in (22), licenses the Q-feature movement because the intermediate Comp is an empty category. The displayed features indicate that *qf(q)* has moved from *dare-o* ‘who-ACC’ to *ka* ‘Q’. Also, the yes/no interpretation possibility for *ka* has been deleted.

(24) Kimi-wa [$_{DP}$ **dare-o** egaita hon]-o yomimashita ka?
you-TOP who-ACC described book-ACC read Q
‘You read a book such that it described who?’

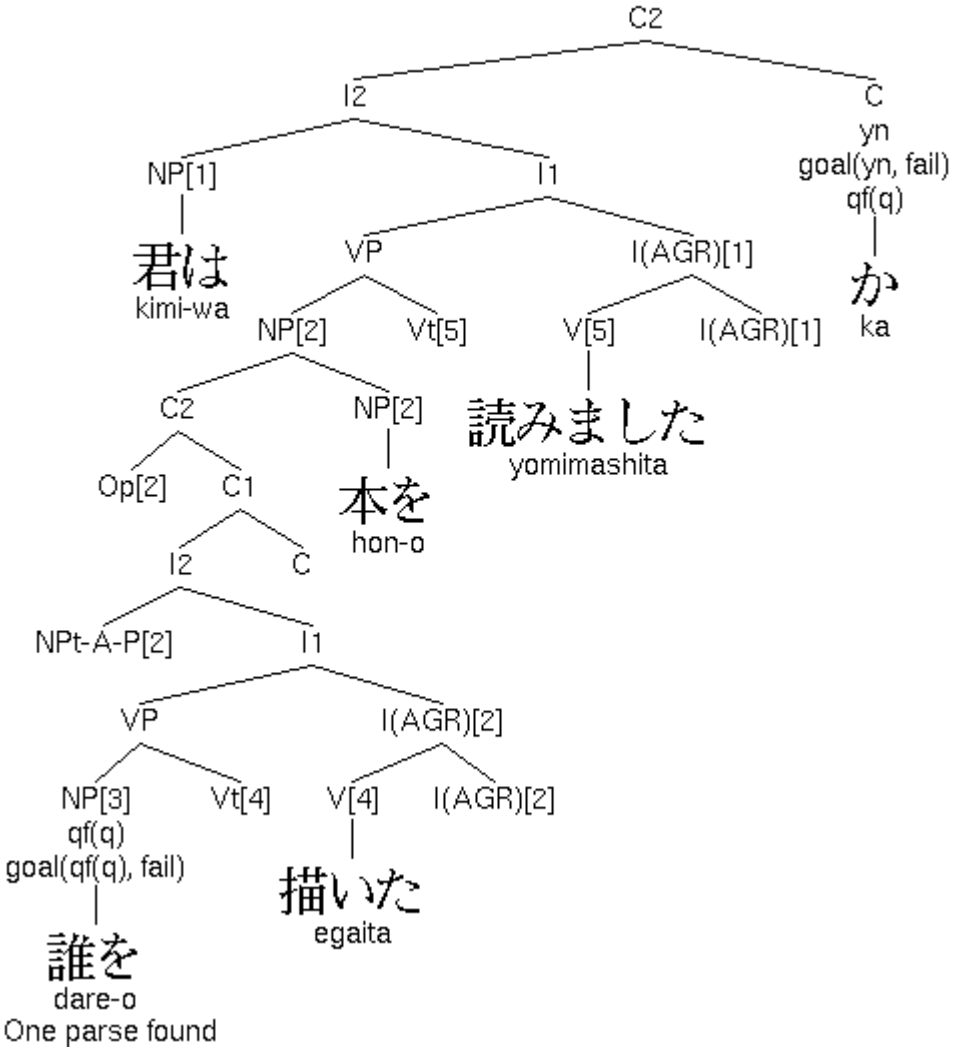

Figure 1: *Wh*-noun in object relative clause

5.2 Local Q-feature Movement

Figure 2 illustrates the parse produced for (6), repeated as (25), which contains an embedded complementizer *ka-doo-ka* 'whether'. Figure 2 shows that only local Q-feature movement applies to *qf(q)* for *nani-o* 'what-ACC'. This is because *addDeleteQ*, the component of *moveQ* defined in (18), transfers the *qf(q)* from *nani-o* 'what-ACC' to the embedded Comp. As a result, (25) is interpreted as a matrix yes/no question. This example models how the matrix *wh*-question interpretation is not allowed because movement of the Q-feature associated with *nani-o* 'what-ACC' out of the embedded clause is blocked by the MLC.

(25) ??[$_{CP}$ John-wa [Mary-ga **nani-o** katta ka-doo-ka] Tom-ni
John-TOP Mary-NOM what-ACC bought whether Tom-DAT
tazuneta no]?
asked Q
'What did John ask Tom whether Mary bought?'

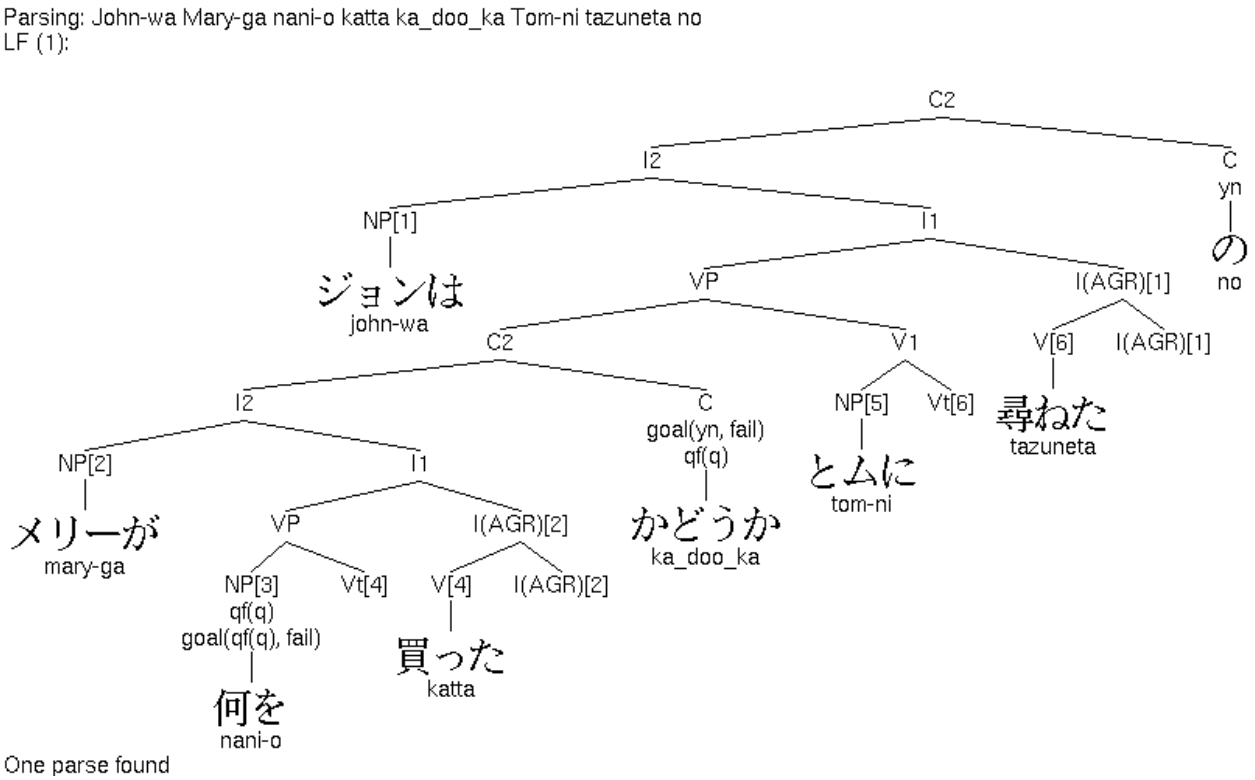

Figure 2: *Wh*-noun in complement clause headed by *ka-doo-ka*

5.3 Q-feature movement with an additional *wh*-phrase

Figures 3 and 4 illustrate the parses produced for (8), repeated as (26), containing both an embedded *wh*-phrase (*nani-o* 'what-ACC') and a matrix *wh*-phrase (*dare-ni* 'who-DAT').

(26) [$_{CP}$ John-wa [Mary-ga **nani-o** katta ka-doo-ka] **dare-ni**
John-TOP Mary-NOM what-ACC bought whether who-DAT
tazuneta no]?
asked Q
'Who did John ask whether Mary bought what?'

Unlike for (25) above, a matrix *wh*-question interpretation is permitted. The possible interpretations produced by PAPPI are:

1. Embedded scope for *nani-o* ‘what-ACC’ and matrix scope for *dare-ni* ‘who-DAT’: see Figure (3).

2. Matrix scope for both *nani-o* ‘what-ACC’ and *dare-ni* ‘who-DAT’: see Figure (4).

These interpretations are permitted because *moveQ*, defined in (17), can be applied recursively. *moveQ* must be applied because *findQ*, defined in (20), is true, as the NP *dare-ni* ‘who-DAT’ contains a *qf(q)* feature. Since *findQ* is true, *addDeleteQ* must be applied, and *addDeleteQ* moves this *qf(q)* to the matrix Comp. Then the operation *moveQ* can be applied again, thereby allowing *addDeleteQ* to move the *qf (q)* feature of the lower *wh*-phrase *nani-o* ‘what-ACC’. *addDeleteQ* then has two options: the first is to move this *qf(q)* to the embedded Comp, as shown in figure (3), and the second is to allow this *qf(q)* to be transferred to the matrix Comp, as shown in Figure (4). This latter example models the PMC effects, whereby satisfaction of the MLC by the Q-feature associated with the matrix *wh*-phrase allows the MLC to be ignored so that the more distant Q-feature associated with the embedded *wh*-phrase can also raise to the matrix Comp.

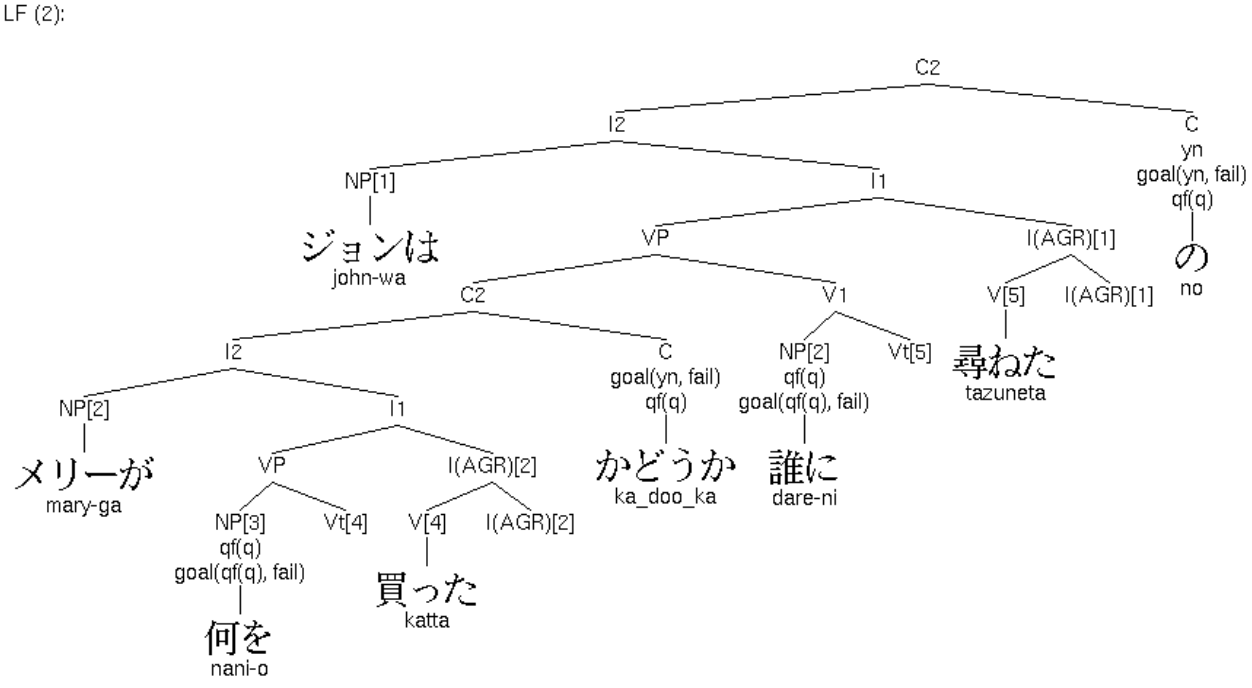

Figure 3: Multiple *wh*-phrases (part 1)

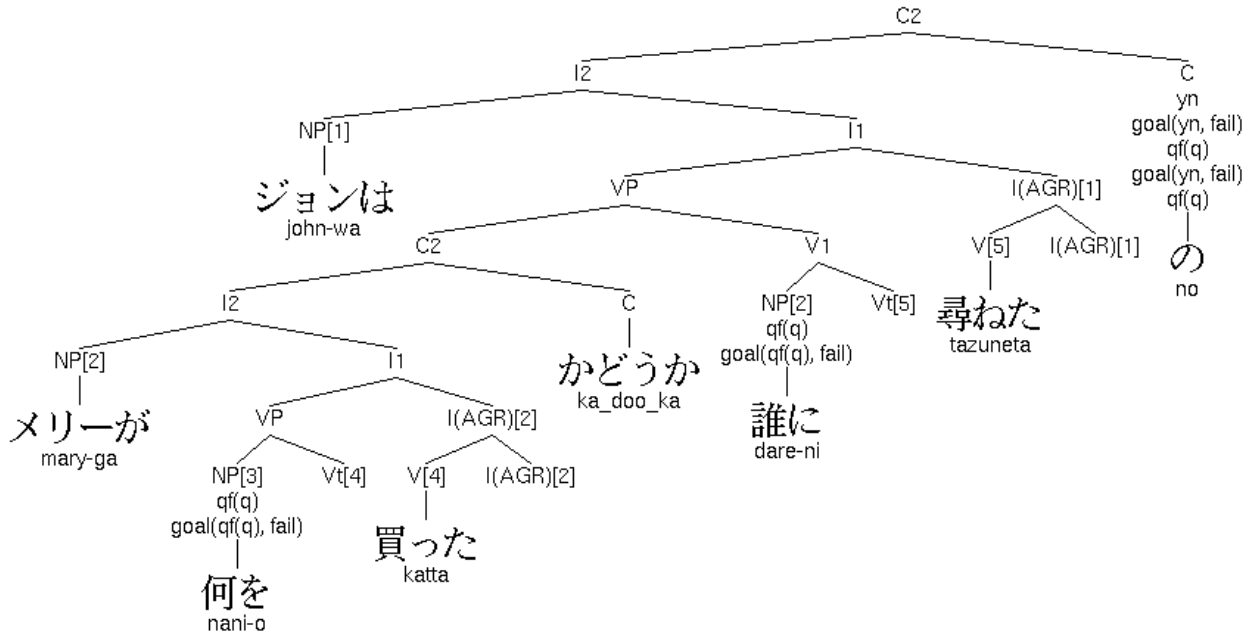

Figure 4: Multiple *wh*-phrases (part 2)

5.4 *sika*-marked NPs and Q-feature movement

Figures 5 and 6 illustrate the parses rejected and accepted for examples (10a-b), repeated as (27a-b), respectively.

(27) (a) *?Hanako-**sika** **nani-o** yoma-**nai** no?
Hanako-only what-ACC read-NEG Q

(b) **Nani-o$_1$** Hanako-**sika** **t$_1$** yoma-**nai** no?
what-ACC Hanako-only read-NEG Q
'What did only Hanako read?'

In either case, the lexical entry for the marker *-sika*, given previously in (15), instructs it to mark *Hanako*, the noun immediately to its right, with the feature *qf(quant)*. Next *moveQ* applies with different results for figures (5) and (6):

- In the case of (27a), in which no scrambling has taken place, *addDeleteQ* blocks the movement of *qf(quant)* on the basis of the *yn* feature present on the matrix question marker *no*. As a result, *CheckQ* (the parser operation defined in (23) that checks to see whether all quantificational NPs have

undergone movement to determine scope) will block the parse as shown in Figure 5.

- By contrast, in (27b) *nani-o* 'what-ACC' has been scrambled above the *-sika*-marked NP. The operation *moveQ* applies first to *nani-o* 'what-ACC', resulting in *qf(q)* appearing on the question marker *no*, and therefore identifying *no* as a *wh-* question. The possibility of *no* functioning as a yes/no-question is canceled by deleting the feature *yn*. Next, *moveQ* applies to the lower quantificational NP *Hanako-sika*. *moveQ* succeeds this time since *qf(q)* has been deleted during the prior Q-feature move. Hence, the parser admits the structure shown in Figure 6.

In this manner, the parser models blocking of Q-feature movement by an intervening quantifier. Scrambling of the *wh*-phase over the intervening NPI allows the Q-feature to avoid an MLC violation and move to Comp.

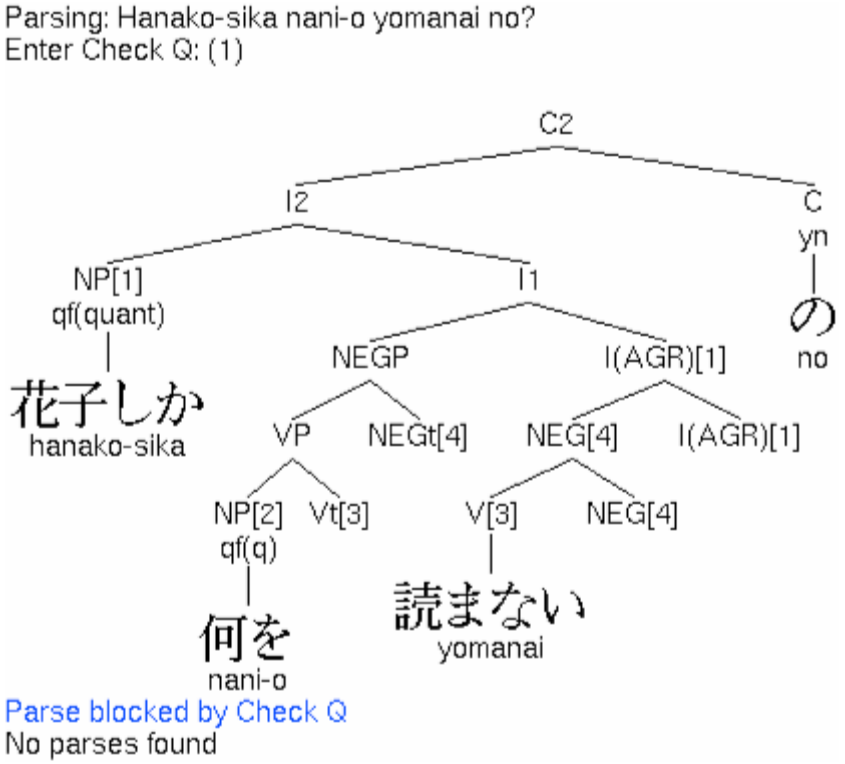

Figure 5: Canonical word order

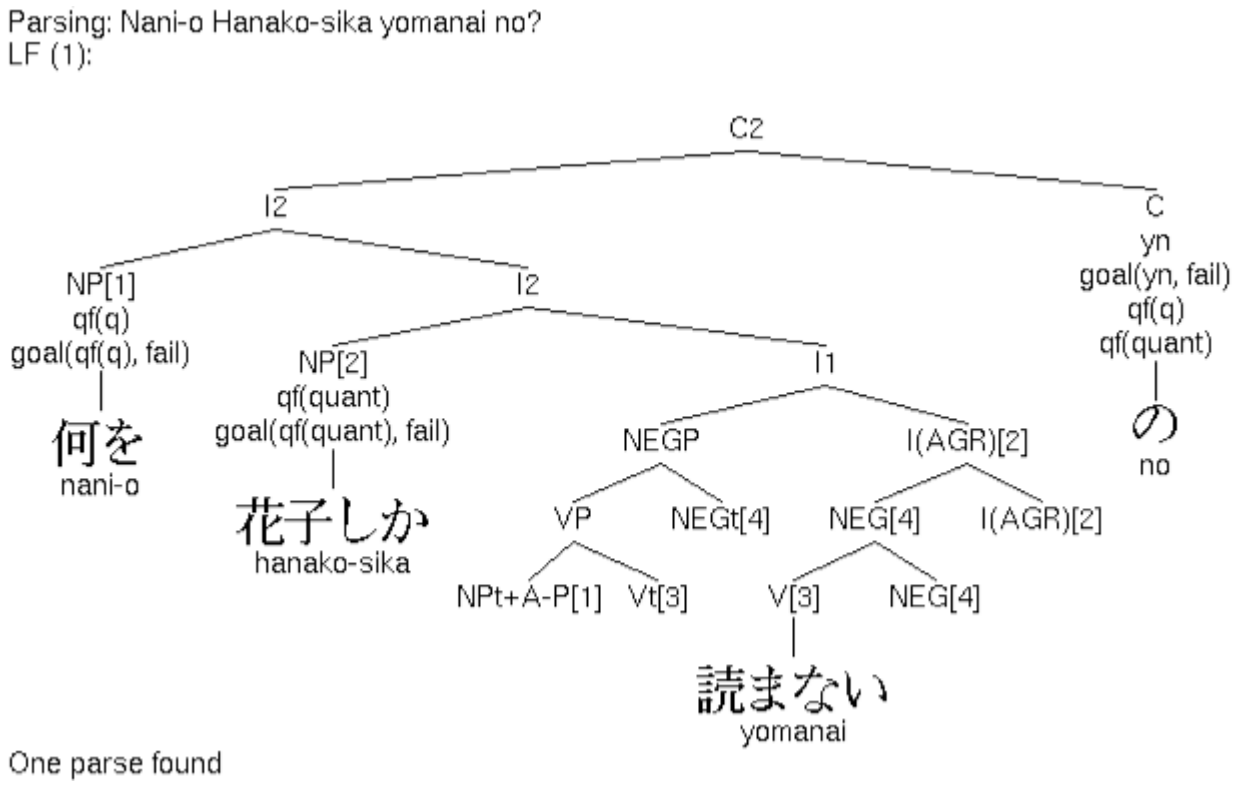

Figure 6: Scrambled word order

5.5 Quantifiers and Q-feature movement

Figures 7 and 8 illustrate the parses rejected and accepted for examples (11a-b), repeated as (28a-b), respectively.

(28) (a) *? **Daremo-ga** **nani-o** katta no?
everyone-NOM what-ACC bought Q

(b) **Nani-o$_1$** **daremo-ga** **t$_1$** katta no?
what-ACC everyone-NOM bought Q
'What did everyone buy?'

These examples are accounted for in the same manner as (27a-b) discussed in the previous section. The lexical entry for *daremo* 'everyone', given in (14), has the feature *qf(quant)*. *moveQ* applies as follows:

- In (28a), in which no scrambling has taken place, *addDeleteQ* blocks movement of *qf(quant)* on the basis of the *yn* feature present on the

matrix question marker *no*. Then *CheckQ* blocks the parse, as shown in Figure 7.

- In (28b), *nani-o* 'what-ACC' has been scrambled over the quantifier *daremo-ga* 'everyone-NOM'. The operation *moveQ* applies first to *nani-o* 'what-ACC', resulting in *qf(q)* appearing on the question marker *no*, which identifies *no* as a *wh*-question. The feature *yn* is deleted, thereby preventing *no* from functioning as a yes/no-question. Then, *moveQ* applies to the lower quantificational NP, *daremo* 'everyone'. *moveQ* succeeds since *qf(q)* has been deleted during the prior Q-feature move, and the parser admits the structure shown in Figure 8.

The parser therefore models blocking of Q-feature movement by an intervening quantifier. Scrambling of the *wh*-phase over the quantifier allows the Q-feature to move to Comp without violating the MLC.

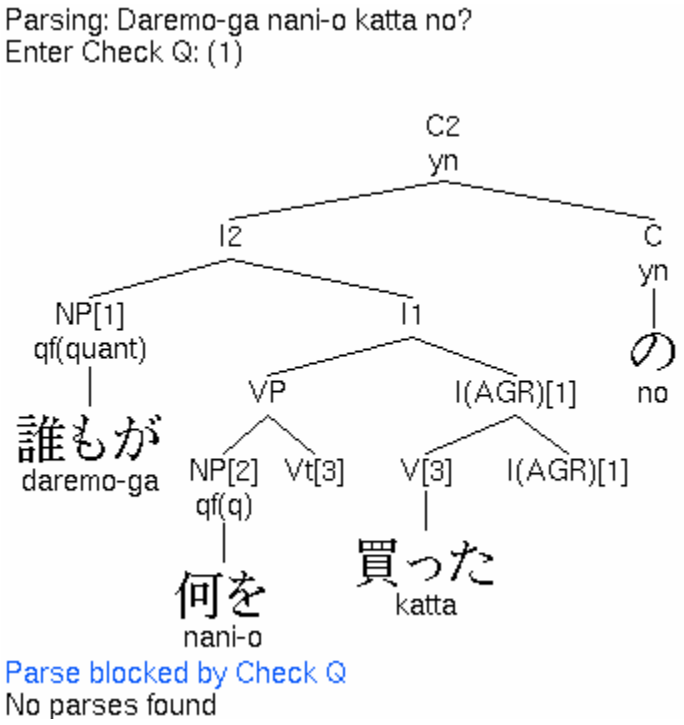

Figure 7: Canonical word order

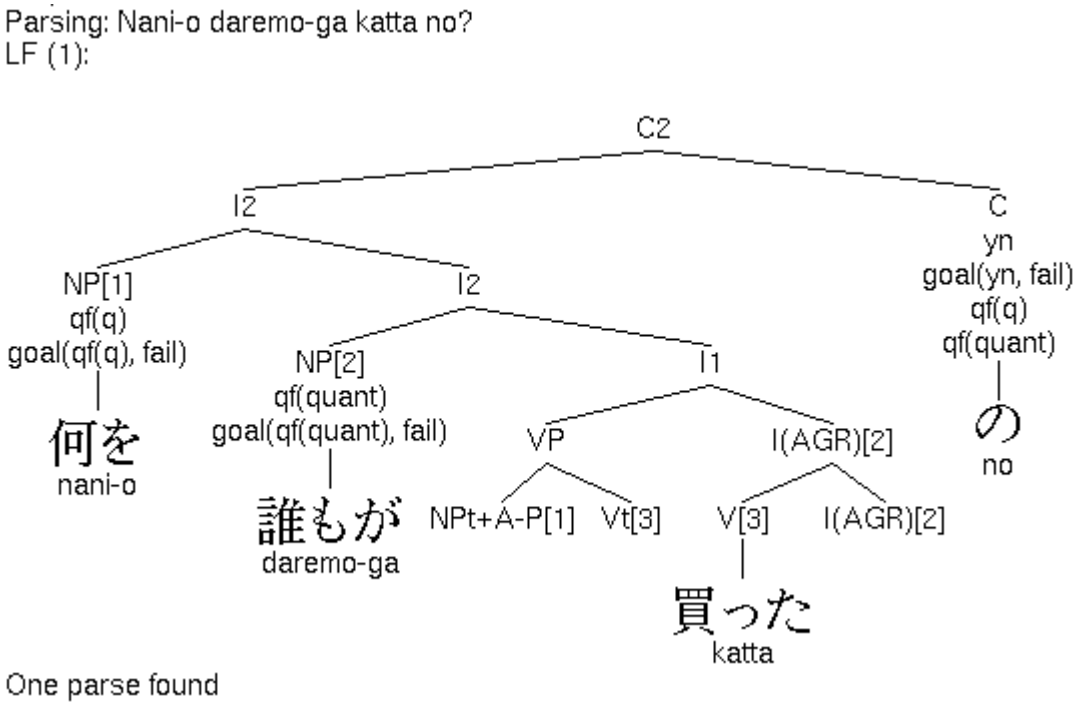

Figure 8: Scrambled word order

6. Conclusion

In this paper, we showed how we modified PAPPI to implement a theory of Q-feature movement. This theory partially accounts for the structure of certain *wh*-constructions. A *wh*-phrasal argument may occur in an NP-island because movement of a Q-feature to Comp may proceed without violating the MLC. In a single *wh*-construction, a *wh*-phrase may not occur in a *wh*-island because Q-feature movement is blocked by an intervening Q-feature. This type of *wh*-island effect can be ameliorated when an additional *wh*-phrase occurs in the matrix C. Q-feature movement in the matrix clause licenses C, via the PMC, to attract a Q-feature associated with an embedded *wh*-phrase. Lastly, this theory also accounts for why a *wh*-phrase must be scrambled over an intervening quantificational phrase, such as an NPI or quantifier; scrambling enables a Q-feature to move to Comp without violating the MLC.

By successfully modeling the Q-feature movement theory discussed here, we have shown that it is compatible with the GB theory implemented by PAPPI. This is because in order for a parse to be successful, it must be permitted by all of the principles utilized by PAPPI. In this way, computational modeling can show whether or not a proposal that accounts for a particular syntactic phenomenon is compatible with other proposals within a particular syntactic theory.

In conclusion, we have shown how a computer program can model a linguistic theory. If the theory is representative of what is occurring in the human mind, then PAPPI may be modeling certain aspects of how humans parse language, thereby providing insight into how syntactic processes occur in the human mind.

References

Alexiadou, Artmeis, and Elena Anagnostopoulou. 1998. Parameterizing word order, V-movement, and EPP-checking. *Natural Language and Linguistic Theory* 16: 491-539.

Beck, Sigrid, and Shin-Sook Kim. 1997. On *wh*- and operator scope in Korean. *Journal of East Asian Linguistics* 6: 339-384.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Chomsky, Noam. 1986. *Knowledge of language: Its nature, origin, and use*. New York: Praeger.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In *Step by step*, eds. Roger Martin, David Michaels, and Juan Uriagereka, 89-155. Cambridge, MA: MIT Press.

Colmerauer, A., Kanoui, H., Pasero, R., and P. Roussel. 1973. *Une système de communication homme-machine en Français*. Report, Artificial Intelligence Group. Université d'Aix-Marseille II.

Fong, Sandiway. 2001. Japanese PAPPI. *Researching and Verifying an Advanced Theory of Human Language, COE Report* (5), 445-464. Kanda University of International Studies (KUIS), Chiba, Japan.

Ginsburg, Jason. 2006. Overt Q-feature movement in Japanese *wh*-constructions. Ms., University of Arizona.

Hagstrom, Paul. 1998. *Decomposing questions*. Doctoral Dissertation, MIT.

Karimi, Simin, and Azita Taleghani. To appear. Wh-movement, interpretation, and optionality in Persian. University of Arizona.

Lasnik, Howard, and Juan Uriagereka. 1988. *A course in GB syntax: Lectures on Binding and Empty Categories*. MIT Press.

Maki, Hideki. 1995. *The syntax of particles*. Doctoral Dissertation, University of Connecticut.

Miyagawa, Shigeru. 2001. EPP, scrambling and *wh*-in-situ. In *Ken Hale: A life in language*, ed. Michael Kenstowicz, 293-338. Cambridge, MA: MIT Press.

Nishigauchi, Taisuke. 1999. Quantification and *wh*-constructions. In *The handbook of Japanese linguistics*, ed. Natsuko Tsujimura, 269-296. Malden, MA: Blackwell.

Pesetsky, David. 2000. *Phrasal movement and its kin*. Cambridge, MA: MIT Press.

Richards, Norvin. 2000. An island effect in Japanese. *Journal of East Asian Linguistics* 9: 187-205.

Richards, Norvin. 2001. *Movement in language: Interactions and architectures*. New York: Oxford University Press.

Saito, Mamoru. 1994. Additional-*wh* effects and the adjunction site theory. *Journal of East Asian Linguistics* 3: 195-240.

Tanaka Hidekazu. 1999. LF *wh*-islands and the minimal scope principle. *Natural Language and Linguistic Theory* 17: 371-402.

Watanabe, Akira. 1992a. Subjacency and S-structure movement of *wh*-in-situ. *Journal of East Asian Lingustics* 1: 255-291.
Watanabe, Akira. 1992b. *Wh*-in-situ, Subjacnecy and chain formation. *MIT Occasional Papers in Linguistics* 2.
Watanabe, Akira. 2003. *Wh*-in-situ languages. In *The handbook of contemporary syntactic theory*, eds. Mark Baltin and Chris Collins, 203-225. Malden, MA: Blackwell.

Jason Ginsburg
Department of Linguistics
University of Arizona
Douglass Building, 200E
Tucson, AZ 85721
jginsbur@email.arizona.edu

Sandiway Fong
Department of Linguistics
University of Arizona
Douglass Building, 200E
Tucson, AZ 85721
sandiway@email.arizona.edu