Association of Floating Quantifiers with NPs: A Linear Order Perspective

Masahiro KOBAYASHI[†] Kei YOSHIMOTO[‡]

Graduate School of International Cultural Studies, Tohoku University
[†]mkoba@insc.tohoku.ac.jp [‡]kei@intcul.tohoku.ac.jp

1 Introduction

It is widely known that in many languages quantifiers float from NPs they modify. Many disccussions of floating quantifiers can be found in the abundant syntactic literature (Shibatani 1977, Miyagawa 1989). In addition, the necessity of identification of the NPs modified by floating quantifiers is also recognized in machine translation systems (Bond, F., D. Kurz, and S. Shirai 1998). This paper investigates syntactic and semantic constraints on quantifiers floating in Japanese, and proposes a way to specify which NPs are associated with floating quantifiers. The formal framework proposed is based on Dynamic Syntax (Kempson, M.-Viol and Gabbay 2001) in which logical representations are built up incrementally.

2 Quantifier Floating

Before getting into the detailed discussion, we briefly illustrate the floating quantifier constructions in Japanese. In Japanese numeral quantifiers are usually attached to classifiers (CL) with the genitive case marker *no*, and placed in the position immediately preceding the NPs they modify, as seen in (1a). By constrast, the quantifier in (1b) is moved to a position immediately followed by the verb, dropping the genitive case marker *no*. The quantifier in (1b) is called a floating quantifier (or *FQ* hereafter).

(1) a. San-satsu-no hon-o John-ga
three-CL-GEN book-ACC John-NOM
katta.
buy-PAST
'John bought three books.'

b. Hon-o John-ga san-satsu
book-ACC John-NOM three-CL
katta.
buy-PAST

Miyagawa (1989) claims that FQs and the NPs they modify (or *antecedents* hereafter) must c-command each other[1]. It seems that (1b) does not satisfy the condition of Miyagawa (1989): the position of the antecedent NP *hon-o* is too high to be c-commanded by the FQ. In spite of this fact, he claims that (1b) is well-formed because the antecedent NP moves from the adjacent position to that of FQ, and the FQ and the trace left by the movement of the antecedent are in a c-command relation.

Takami (1998) points out that there exist examples which are not subject to Miyagawa's claim. (2a, b) are quoted from Takami (1998).

(2) a. Gakusei-ga sono shinkan zasshi-o
student-NOM the newly-published magazine-ACC
go-nin kainikita.
five-CL buy-PAST and come-PAST
'Four students came and bought the newly-published magazine.'

b. Mai-toshi nada-kō-no gakusei-wa
every-year Nada-school-GEN student-TOP
tōdai-o 50-nin-izyō
Univ. of Tokyo-ACC 50-CL-more than
zyukensuru.
take entrance exams
'Every year more than 50 students of Nada high school take entrance exams of Univ. of Tokyo.'

Although the FQs and antecedent the NPs do not c-command each other, in other words, the FQs are not adjacent to the NPs or their traces, (2a, b) are grammatical. Takami (1998) claims that the subject NP and subject-oriented FQ can be separated when the object NP inserted between them is definite as seen in (2a, b).

Appropriately capturing the charateristics of FQs in Japanese, this paper is meant to provide a formal framework to associate FQs with the antecedent

[1]The standard definition of c-command (Reinhart 1981) is provided as follows:
Node A c-commands node B iff
(i) A does not dominate B and B does not dominate A; and
(ii) the first braning node dominating A also dominates B.

NPs. The problems we address here are that there are two different types of constructions associated with FQs. In this paper, the two inconsistent examples from Miyagawa (1989) and Takami (1998) can be interpreted in a principled way: the informal idea of the definiteness of the object NP by Takami (1998) can be straightforwardly formalized and the c-command based account by Miyagawa (1989) can be accommodated into the framework proposed in this paper.

3 Formal Framework

The formal framework we adopt is based on Dynamic Syntax (Kempson et al. 2001) in which tree representations are built up on a left-to-right basis. What differentiates this formalism from others is the explicit use of underspecification in syntax. In Dynamic Syntax the trees are regarded as sets of nodes. As each word is processed, the tree constructions grow: when there is no word to be processed, the resulting tree must be binary branching. Since the information of a node within a tree, e.g, addresses, types, may change as a tree grows, we need to divide the information of each node into two parts: one part describes what is already fixed (done part), the other is for what is to be achieved or to be satisfied until all words are processed (requirement part).

To illustrate the parsing process, consider (1b). The starting point of parsing is defined as (3).

(3) $\{Tn(a), ?Ty(t), \Diamond\}$

The annotation $Tn(a)$ in (3) represents the address of this node. (3) is the initial state, and the relative position within the tree is expected to change as the parsing proceeds, so the position is described as the variable a in early stages. $?Ty(t)$ indicates that this node may be of type t but it is not specified yet. The question mark '?' means that this annotation is to be achieved in a later stage, and called a requirement. '$\Diamond$' is called a pointer, and rules are applied to the node which is highlighted by this pointer.

The first word to be read in (1b) is *hon* and the lexical specification of *hon* takes the form in (4).

(4) **IF** $\{?Ty(t)\}$
THEN $\mathtt{make}(\langle\downarrow_*\rangle)$; $\mathtt{put}(Fo(\epsilon, x, Hon(x)), Ty(e))$
ELSE ABORT

To begin with, a common noun *hon* tests for the existence of $?Ty(t)$ at the pointed node (3), then make a new node. New annotations associated with semantics and type information are added to a created node. Hence the application of (4) to (3) leads to the following tree description (5).

(5) $\{Tn(\mathrm{a}), ?Ty(t)\}$

$\{\langle\uparrow_*\rangle Tn(\mathrm{a}), Fo(\epsilon, x, Hon(x)), Ty(e), ?\exists \mathrm{x} Tn(\mathrm{x}), \Diamond\}$

The nodes in (5) are connected by a dotted line: this means that the position of a newly created node is not fixed. This is ensured by the action $\mathtt{make}(\langle\downarrow_*\rangle)$ in (4). It makes a new node connected downward by the dotted line. The subsequent step is the processing of the accusative case marker *o*, and this results in the new tree in (6).

(6) $\{Tn(\mathrm{a}), ?Ty(t), \Diamond\}$

$\{\langle\uparrow_*\rangle Tn(\mathrm{a}), Fo(\epsilon, x, Hon(x)), Ty(e), ?\exists \mathrm{x} Tn(\mathrm{x}), \langle\uparrow_0\rangle Ty(e \rightarrow t)\}$

The added annotation $\langle\uparrow_0\rangle Ty(e \rightarrow t)$ indicates that this node is an argument daughter of a node with a type $Ty(e \rightarrow t)$. The annotation associated with a type is used to merge a node with other nodes in later stages. As a next step, the reading of the NP *John-ga* yields the following tree (7). The detailed illustrations are abridged.

(7) $\{Tn(\mathrm{a}), ?Ty(t), \Diamond\}$

$\{Fo(\epsilon, x, Hon(x)), Ty(e), \langle\uparrow_0\rangle Ty(e \rightarrow t)\}$ $\{Fo(John), Ty(e), \langle\uparrow_0\rangle Ty(t)\}$

The annotation $\langle\uparrow_0\rangle Ty(t)$ in (7) means that the node will be an argument daughter of a type $Ty(t)$ node. The annotations $\langle\uparrow_0\rangle Ty(e \rightarrow t)$ and $\langle\uparrow_0\rangle Ty(t)$ are added by the case particles *o* and *ga* respectively.

The FQ *san-satsu* introduces the annotation $FQ(+)$ to the top node (root node), and the verb *katta* introduces the subtree which has three arguments decorated by meta-variables, as seen in Figure 1. In Fig. 1 the nodes decorated by meta-variables V, W and X have their fixed positions. By contrast, the nodes connected by the dotted lines lack their fixed positions. Therefore the nodes decorated by the object *hon-o* and the subject *John-ga* are merged with the nodes decorated by $Fo(W)$ and $Fo(V)$ respectively because of the type matching in Fig. 1. The floating quantifier $Fo(san\text{-}satsu)$ is merged with the meta-variable X.

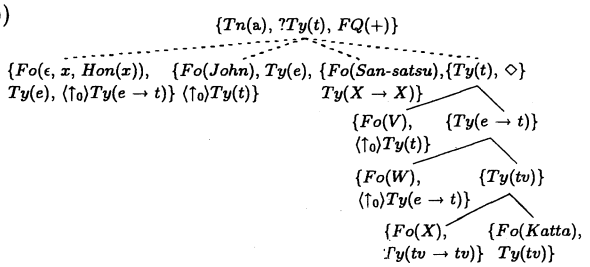

Figure 1: The tree descriptions of (1b)

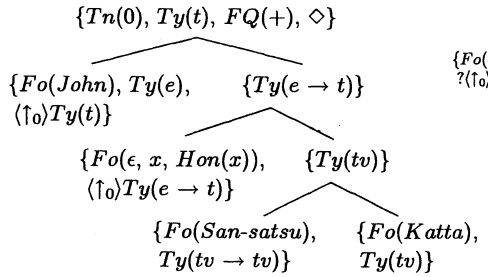

Figure 2: The final tree description of (1b)

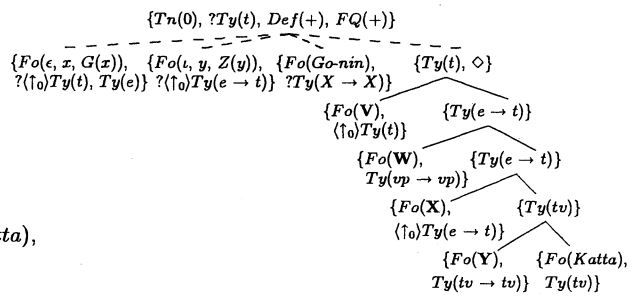

Figure 3: The tree description of (2a)

The type $Ty(X \rightarrow X)$ indicates that this type takes an arbitrary type X and returns the same type X. The final step merges the top node introduced by the verb with the root node, and this yields the final complete binary tree as seen in Figure 2. As seen in Fig. 2, the antecedent NP *hon-o* and the FQ are adjacent even though they are separated in the surface word order as in Fig. 1. All FQs should do is to quantify the NPs adjacent to themselves.

Each node is decorated by the pair of syntax and semantics, *labels* and *formulae*. On the label side, the mode of combination is determined by types, i.e., modus ponens, and on the formula side the functional application allows us to build meanings compositionally. The semantic formula of (1b) is illustrated in (8). 'ϵ' corresponds to the existential quantifier of the familiar first order predicate logic.

(8) $Fo(San\text{-}satsu(Katta)(\epsilon, x, Hon(x))(John))$

(8) is reduced to the pre-final representation in (9).

(9) $\exists x(Hon(x) \wedge San\text{-}satsu(Katta)(x)(John))$

We define the rule (10) by which we interpret the sentences with FQs.

(10)

$$\frac{\{Ty(t), Fo(\exists x \phi[\gamma FQ_{\mathcal{N}}(\psi)]), \ldots, \diamond\}}{\{Ty(t), Fo(\exists x[\mid x \mid = \mathcal{N} \wedge x \in CL \wedge (\gamma)(\psi)]), \ldots, \diamond\}}$$

(9) is reduced to (11) by the application of (10) and (11) appropriately expresses the truth-condition of (1b).

(11) $\exists x(|x| = 3 \wedge x \in satsu \wedge Hon(x) \wedge Katta(x)(John))$

We now turn to example (2a) where the FQ and the antecedent subject NP are not adjacent. Figure 3 illustrates the tree description of (2a). In Fig. 3 NPs and the FQ are connected to the root node by the dotted lines. Takami (1998) claims that if the object NP is definite, the subject NP and the subject-oriented FQ can be separated. His informal statement can be formalized straightforwardly by the framework in this paper: the annotation $Def(+)$ in the root node is induced by the definite NP *sono-shinkan-zasshi* 'the newly-published magazine'. It means that the object NP is definite. The verb *katta* 'bought' proves the existence of $Def(+)$ in the root node, and introduces the verb-frame which has four argument places decorated by meta-variables V, W, X and Y. The node with $Fo(V)$ is merged with that with $Fo(\epsilon, x, G(x))$ which represents *gakusei* 'student' because they have the same types $\langle\uparrow_0\rangle Ty(t)$ and $Ty(e)$. In the same way, the node with $Fo(X)$ is merged with that with $Fo(\iota, y, Z(y))$ because they have the same types $\langle\uparrow_0\rangle Ty(e \rightarrow t)$ and $Ty(e)$. The FQ $Fo(Go\text{-}nin)$ is of type $X \rightarrow X$. This means that it takes an arbitrary type as an argument and returns the same category. Hence there are two possibile arguments to be merged with the FQ $Fo(Go\text{-}nin)$: one is the node with $Fo(W)$ and $Ty(vp \rightarrow vp)$, and the other is that with $Fo(Y)$ and $Ty(tv \rightarrow tv)$. If the FQ is merged with the node with $Fo(Y)$, then it is eliminated by the semantic constraint $y \in nin$. This turns out to mean that the magazine is meaningfully counted by the classifier *nin*, which is supposed to be attached to a noun refering to people. Consequently the FQ is forced to be merged with the node with $Fo(W)$, and the node with $Fo(Y)$ is seman-

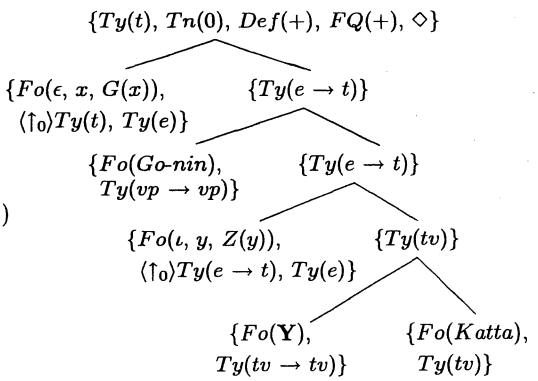

Figure 4: The final tree description of (2a)

```
IF    {?Ty(t)}
THEN make(⟨↓*⟩); put(?Ty(t), ?∃xTn(x));
     make(⟨↓0⟩); put(Fo(V), Ty(e), ?∃xFo(x)); go(⟨↑⟩);
     make(⟨↓1⟩); put(?Ty(e → t)); go(⟨↑⟩); gofirst↑(?Ty(t));
     IF    {FQ(+), Def(+)},
     THEN gofirst↓(?Ty(t)); go(⟨↓1⟩); make(⟨↓1⟩);
          put(Fo(W), Ty(vp → vp), ?∃xFo(x)); go(⟨↑⟩);
          make(⟨↓0⟩); put(Ty(e → t));
          make(⟨↓0⟩); put(Fo(X), Ty(e)); go(⟨↑⟩);
          make(⟨↓1⟩); put(Ty(tv));
          make(⟨↓1⟩); put(Fo(Y), Ty(tv → tv)); go(⟨↑⟩);
          make(⟨↓0⟩); put(Fo(Katta), Ty(tv));
          gofirst↑(?Ty(t))
     ELSE IF    {FQ(+)},
          THEN gofirst↓(?Ty(t)); go(⟨↓1⟩);
               make(⟨↓0⟩); put(Fo(W), ?Ty(e)); go(⟨↑⟩);
               make(⟨↓1⟩); put(Ty(tv));
               make(⟨↓1⟩); put(Fo(X), Ty(tv → tv)); go(⟨↑⟩);
               make(⟨↓0⟩); put(Fo(Katta), Ty(tv));
               gofirst↑(?Ty(t))
          ELSE gofirst↓(?Ty(t)); go(⟨↓1⟩);
               make(⟨↓0⟩); put(Fo(W), Ty(e), ?∃xFo(x)); go(⟨↑⟩);
               make(⟨↓1⟩); put(Fo(Katta), Ty(tv));
               gofirst↑(?Ty(t))
ELSE ABORT
```

Figure 5: The lexical specification of *Katta*

tically vacant. The final tree description is illustrated in Figure 4. As seen in Fig. 4, the antecedent NP and FQ are adjacent even though they are separated in actual surface word order in the same way we argued in Fig. 2. (12) is the logical formula we get in the root node in Fig. 4.

(12) $Go\text{-}nin(Katta(\iota, y, Z(y)))(\epsilon, x, G(x))$

(12) is reduced to the pre-final version as seen in (13).

(13) $\iota y(Z(y) \wedge \exists x(G(x) \wedge Go\text{-}nin(Katta(y)(x))))$

In order to reduce (13) to the familiar first order predicate logic representation (14), we do not need any additional rules: the rule we defined in (10) also allows us to get the final representation.

(14) $\iota(Zasshi(y) \wedge \exists(|x| = 5 \wedge x \in nin \wedge Gakusei(x) \wedge Katta(y)(x)))$

As we have shown, the underspecification allows us to cope with two different constructions in a principled way, and interpret them through the simple rule we have defined. only one lexical specification of transitive verbs in Figure 5 is sufficient to introduce two types of subtrees in Fig. 1 and 3.

4 Discussion

Shibatani (1977) claims that FQs are floated only from *ga*-marked NPs (nominative NPs) and *o*-marked NPs (accusative NPs). Unlike Shibatani (1977), in this paper the possible antecedent NPs of FQs are restricted to argument NPs. We used the text data to verify the adequacy of the formalization proposed in this paper. We chose *Aozora Bunko*, literary works with expired copyright accesible on the website. The total amount of data is 4,588,819 bytes. Out of data, we found that 196 sentences contain the FQs: 115 are floated from *ga*-marked NPs, 76 from *o*-marked NPs and 5 from *wa*-marked NPs (Topic NPs). They all function as arguments of verbs, therefore the results support the formalization in this paper.

5 Conclusion

In this paper we defined a rule to associate FQs with the NPs they quantify. The use of underspecification and incremental tree construction enable us to reduce two different types of constructions to a unique semantic type. The FQs are associated with argument NPs, and the data supports the formalization presented in this paper.

Acknowledgements

This study was supported in part by the 21st Century Center of Excellence (COE) Program (Ministry of Education, Culture, Sports, Science and Technology) entitled "A Strategic Research and Education Center for an Integrated Approach to Language and Cognition" (Graduate School of International Cultural Studies, Tohoku University). I thank Daniela Caluianu and Tsuneyoshi Ishihara, who helped with some of the material presented in this paper.

References

Bond, F., D. Kurz, and S. Shirai. 1998. Anchoring Floating Quantifiers in Japanese-to-English Machine Transltion. in *17th International Conference on Computational Linguistics: COLING-98*. pp.152-159.

Kempson, R., W. M.-Viol and D. Gabbay. 2001. *Dynamic Sytnax: The Flow of Language Understanding*. Oxford: Blackwell Publishers.

Miyagawa, S. 1989. *Syntax and Semantics 22: Structure and Case Marking in Japanese*. San Diego: Academic Press.

Reinhart, T. 1981. Definite NP Anaphora and C-command Domains. *Linguistic Inquiry*. 12. pp.605-635.

Shibatani, M. 1977. Grammatical Relations and Surface Case. *Language*. 53: 4. pp.789-809.

Takami, K. 1998. Nihongo-no sūryōshi yūri-ni tsuite: Kinōron teki bunseki. (On quantifier floating in Japanese: A functional analysis.) *Gengo*. 27: 3. pp.99-107.