# Two Types of Nominalization in Japanese as an Outcome of Semantic Tree Growth

**Tohru Seraku**
St. Catherine's College, University of Oxford
Manor Road, Oxford, UK.
OX1 3UJ
tohru.seraku@stcatz.ox.ac.uk

## Abstract

The particle *no* in Japanese exhibits two types of nominalization: "participant" and "situation" nominalization. Despite several motivations for a uniform account, only a few attempts have been made to address *no*-nominalization uniformly. In this paper, I shall develop a unified account within the formalism Dynamic Syntax, and show that a number of properties of the phenomenon follow from the analysis.

## 1 Introduction

The particle *no* in Japanese displays two types of nominalization: "participant" nominalization (1) and "situation" nominalization (2).

(1) [*Akai no*]-*o Tom-ga nagu-tta.*
[red NO]-ACC Tom-NOM hit-PAST
'Tom hit a/the red one.'

(2) [*Mary-ga kireina no*]-*o*
[Mary-NOM beautiful NO]-ACC
*Tom-ga shi-tteiru.*
Tom-NOM know-PRES
'Tom knows that Mary is beautiful.'

In participant nominalization, the particle *no* turns a preceding clause into a nominal that denotes an object or a person. In situation nominalization, the particle *no* turns a preceding clause into a nominal that denotes an event or a proposition. A case of ambiguity is presented in (3).

(3) [*Nai-ta no*]-*o Tom-ga mi-ta.*
[cry-PAST NO]-ACC Tom-NOM see-PAST
a. 'Tom saw someone who cried.'
b. 'Tom saw the event of someone's having cried.'

Participant nominalization is exemplified by (3a), and situation nominalization by (3b).[1]

One issue that immediately arises is whether *no* in (1, 2, 3) should be treated uniformly. In other words, does *no* in (1, 2, 3) form a single item or are there two *no*s one of which appears in (1, 3a) and the other of which appears in (2, 3b)? Seraku (in press) defends a uniform analysis based on several motivations (e.g. methodological, cross-linguistic, functional, diachronic). Despite these motivations, a unified analysis of *no* has been largely untouched (e.g. Kitagawa, 2005; Kitagawa and Ross, 1982; Murasugi, 1991; Shibatani, 2009; Tonoike, 1990).

Against this background, the aim of the present paper is twofold as follows. First, I shall articulate a unified analysis of *no*-nominalization within the grammar formalism Dynamic Syntax (Cann et al., 2005; Kempson et al., 2001). Second, I shall show

[1] Seraku (in press) summarizes diachronic data that give credence to the exclusion of such data as (i) from the analysis to be developed in this paper.

(i) *Tom-no*
Tom-NO
'Tom's'

26th Pacific Asia Conference on Language,Information and Computation pages 153–162

that the analysis captures a range of characteristics of the phenomenon.

## 2 Dynamic Syntax

Dynamic Syntax (DS) is a formalism that models "knowledge of language", which is conceived as a set of constraints on language use (Cann et al., 2005; Kempson et al, 2001). Language use consists of production and comprehension. DS is shown to model production (Cann et al., 2007; Purver et al., 2006), but this paper focuses on comprehension. DS is then said to provide a set of constraints on how a parser builds up an interpretation gradually as it processes a string word-by-word online.

DS models gradual growth of an interpretation as successive updating of a semantic tree. A string of words is directly mapped onto a semantic tree; in this view, a separate level of syntactic structures is not postulated. The initial state of semantic tree growth is specified by the AXIOM, which sets out an initial node to be subsequently developed.

(4) AXIOM

?t, ◇

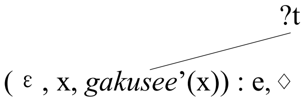
?t is a requirement that this node be of type-t. That is, DS tree growth is goal-driven, the goal being to construct a type-t formula. This requirement must be satisfied before tree transitions come to an end. The pointer ◇ indicates a node under development. Once the initial node in (4) is set out, it is gradually updated by a combination of general, lexical, and pragmatic actions.

For illustration, consider the string (5).

(5) *Gakusee-ga* *nai-ta.*
student-NOM cry-PAST
'A/the student cried.'

The initial state (4) is updated into (6) by the parse of *gakusee-ga* (= 'student-NOM'). First, the general action LOCAL *ADJUNCTION introduces an unfixed node, and the lexical actions encoded in *gakusee* decorate the node with semantic content and type. This unfixed node is fixed as a subject node by the lexical actions of the nominative case particle *ga*. ("Unfixed nodes" is a central DS mechanism, but it is not directly relevant to the present paper.)

(6) Parsing *Gakusee-ga*



The content of *gakusee* is ( ε , x, *gakusee*'(x)), a type-e term expressed in the Epsilon Calculus.

In the Epsilon Calculus, every quantified noun is mapped onto a type-e term defined as a triple: an operator, a variable, and a restrictor. Syntactically, these type-e terms correspond to arbitrary names in natural-deduction proofs in predicate logic. So, the quantified noun *gakusee* (= 'a student')[2] is mapped onto the epsilon term (7), a type-e term consisting of the existential operator ε , the variable x, and the restrictor *gakusee*'(x).

(7) ( ε , x, *gakusee*'(x))

If the term (7) is combined with the predicate *gakusee*', as in (8), the equivalence relation holds for (8) and the predicate-logic formula (9).

(8) *gakusee*'( ε , x, *gakusee*'(x))

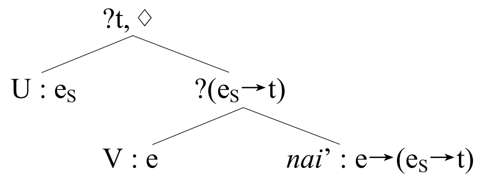
(9) ∃x.*gakusee*'(x)

Semantically, the term (7) stands for an arbitrary witness of the predicate logic formula (9).

The next item to be parsed is *nai* (= 'cry'). As Japanese is fully pro-drop (i.e. arguments do not have to be explicitly uttered), a predicate builds up a template for a propositional structure. In the case of *nai*, it builds up an open propositional structure, where a subject node is decorated with a place-holding variable. Moreover, à la Davidson (1967), it is claimed that all predicates take a type-e event term as an argument (Gregoromichelaki, 2011). So, the predicate *nai* constructs an open propositional structure with the argument slots for a subject term and an event term, as in (10). The subject node is decorated with the place-holding variable V, and the event node with the place-holding variable U. In order to distinguish event terms from non-event terms, the type for event terms is notated as $e_S$, where "s" stands for a "situation".

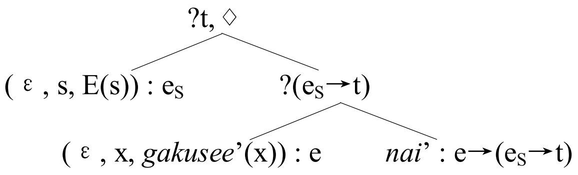
[2] Japanese lacks determiners, and the quantificational force of a bare noun is contextually inferred (cf. §4.2).

(10)



Notice that a subject node has already been created in (6). Thus, the subject node in (6) and that in (10) collapse. The content at the subject node in (10) is the place-holding variable V, and it is weaker than the content at the subject node in (6). Therefore, the collapse of the two subject nodes is harmless. At this stage, the tree (6) is updated into (11).

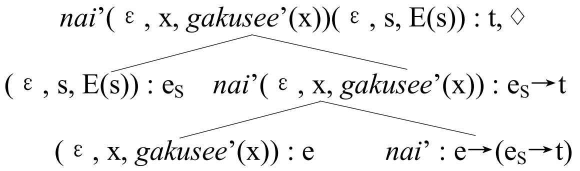
(11) Parsing *Gakusee-ga nai*



U is now replaced with the event term $(\varepsilon, s, E(s))$, where E is an event predicate. For discussion of event predicates, see Cann (2011).

As two daughter nodes are specified for content and type, functional application and type-deduction may occur. These processes are formalized as the general action ELIMINATION. Thus, the tree (11) is updated into (12) after ELIMINATION is run twice.

(12) ELIMINATION


nai'( ε , x, gakusee'(x))( ε , s, E(s)) : t, ◇
( ε , s, E(s)) : e_S
nai'( ε , x, gakusee'(x)) : e_S→t
( ε , x, gakusee'(x)) : e
nai' : e→(e_S→t)

Notice that the requirement ?t has been deleted at the root node in (12) since the type-t formula has appeared at this node.

Finally, the parse of the past tense suffix *ta* adds tense information to the tree. Tense is represented as a restrictor within an event term (Cann, 2011), but this issue is disregarded in this paper. Thus, for the sake of simplicity, I take it that (12) is the final state of the tree transitions for the string (5).

The proposition in (12) contains two terms, and their scope relation needs to be explicated.[3] In a fully articulated tree, a top node of a propositional structure is decorated with a "scope statement", which is incrementally constructed as a string is parsed. The detail is not pertinent; what is at stake is that once tree transitions come to a final state, a proposition at the root node and a complete scope statement are subject to QUANTIFIER EVALUATION (Q-EVALUATION). Through this process, each term in the proposition is enriched so as to explicate the scope dependencies in the whole proposition. For illustration, consider the schematic formula (13).

(13) $\phi(\varepsilon, x, \psi(x))$

Firstly, the predicates $\phi$ and $\psi$, with the term "a" whose content is worked out below, are connected. The type of a connective is determined by the type of an operator; for the existential operator $\varepsilon$, the connective & is employed.

(14) $\phi(a) \& \psi(a)$

Secondly, "a" is constructed so that it reflects the predicates in the whole proposition.

(15) $\phi(a) \& \psi(a)$

$a = (\varepsilon, x, \phi(x) \& \psi(x))$

Now, let us return to the proposition in (12), which is repeated here as (16).

(16) $\mathit{nai}'(\varepsilon, x, \mathit{gakusee}'(x))(\varepsilon, s, E(s))$

Suppose that the scope statement declares that the event term out-scopes the non-event term. In this case, a parser first evaluates the non-event term.

(17) Evaluating the non-event term

$\mathit{gakusee}'(a) \& \mathit{nai}'(a)(\varepsilon, s, E(s))$

$a = (\varepsilon, x, \mathit{gakusee}'(x) \& \mathit{nai}'(x)(\varepsilon, s, E(s)))$

[3] In (12), different scope relations do not affect the truth-conditional content, because only existential quantifications are involved. But the issue is not trivial when different types of quantifications are involved.

Next, the event term in (17) is evaluated.

(18) Evaluating the event term

$$E(b)\&[\textit{gakusee}'(a_b)\&\textit{nai}'(a_b)(b)]$$

$$b = (\varepsilon, s, E(s)\&[\textit{gakusee}'(a_s)\&\textit{nai}'(a_s)(s)])$$
$$a_b = (\varepsilon, x, \textit{gakusee}'(x)\&\textit{nai}'(x)(b))$$
$$a_s = (\varepsilon, x, \textit{gakusee}'(x)\&\textit{nai}'(x)(s))$$

The technical detail is not germane; what should be noted is that the event term "b" and the non-event term "$a_b$" explicate the scope dependencies in the whole formula. ("$a_s$" is not a full-blown term since the variable "s" is not bound in the term; "$a_s$" is just part of "b".) The formula (18) represents the indefinite reading of (5): 'A student cried.'

To sum up, DS models the incremental nature of language use; a parser progressively constructs an interpretation in context on the basis of word-by-word parsing. This exegesis has not mentioned the mechanism of LINK, a core machinery of DS. This is illustrated in the next section since it is essential for the analysis of the particle *no*.

## 3 A Uniform Analysis

### 3.1 Proposal

A novel feature of DS tree transitions is a pair of structures that are connected by a LINK relation. A LINKed structure is an adjunct structure to a main structure, and their relation is guaranteed by the presence of a shared element.

Cann et al. (2005: p.285) analyze the particle *no* as a LINK-inducing device.

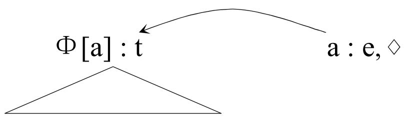
(19) Lexical entry of *no*

```
IF      t
THEN    IF      Φ[a]
        THEN    make(L^-1); go(L^-1); put(a : e)
        ELSE    abort
ELSE    abort
```

In general, every lexical item encodes a constraint on tree growth. The IF-line specifies a condition; if the condition is met, a parser looks at the THEN-line; otherwise the ELSE-line. In (19), "abort" is an action that quits tree transitions, in which case a string is said to be ungrammatical. "make(L)" is an action that introduces a LINK relation, "go(L)" is an action that moves the pointer ◇ to a LINKed node, and "put(a : e)" is an action that decorates a node with "a : e". In plain English, the entry of *no* amounts to the constraint (20); the corresponding tree-update is shown in (21).

(20) If a current node is decorated with a type-t proposition, a parser copies a type-e term in the evaluated proposition and pastes it at a type-e node across a LINK relation.

(21)



In (21), a parser copies the type-e term "a" in the evaluated version of the proposition Φ and pastes it at a type-e node across a LINK relation. The LINK relation is shown by the curved arrow.

Given the entry of *no* in (19), my proposals are formulated as (22).

(22) The two types of *no*-nominalization can be reduced to a parser's choice of what type-e term it copies in processing *no*.

a. Copying of a non-event term gives rise to participant nominalization.

b. Copying of an event term gives rise to situation nominalization.

### 3.2 Participant Nominalization

Let us start with the participant nominalization (1), reproduced here as (23).

(23) [*Akai no*]-*o* *Tom-ga* *nagu-tta*.
[red NO]-ACC Tom-NOM hit-PAST
'Tom hit a/the red one.'

The initial state is determined by the AXIOM:

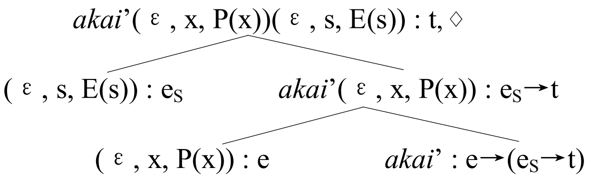
(24) AXIOM

?t, ◇

The predicate *akai* (= 'red') in (23) constructs a propositional template with subject and event slots.

The event node is decorated with ( ε , s, E(s)), and the subject node is decorated with ( ε , x, P(x)), where P is an abstract restrictor (Kempson and Kurosawa, 2009: p.65). Then, the general action ELIMINATION is conducted twice, and the tree (24) is updated into (25).

(25) Parsing *Akai*


*akai*'( ε , x, P(x))( ε , s, E(s)) : t, ◇
( ε , s, E(s)) : $e_S$
*akai*'( ε , x, P(x)) : $e_S \to t$
( ε , x, P(x)) : e
*akai*' : $e \to (e_S \to t)$

Once a proposition emerges, it is subject to Q-EVALUATION. As the proposition in (25), repeated here as (26), involves two terms, Q-EVALUATION is conducted twice.

(26) *akai*'( ε , x, P(x))( ε , s, E(s))

Let us suppose that the scope statement declares that the non-event term out-scopes the event term; in this case, the event term is evaluated first.

(27) Evaluating the event term ( ε , s, E(s))

E(a)&*akai*'( ε , x, P(x))(a)

a = ( ε , s, E(s)&*akai*'( ε , x, P(x))(s))

The formula (27) still contains a type-e term. This term is evaluated as follows:

(28) Evaluating the non-event term ( ε , x, P(x))

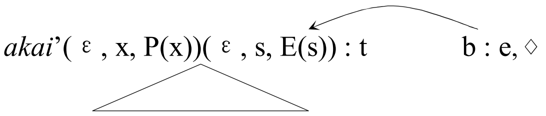
$P(b)\&[E(a_b)\&\textit{akai}'(b)(a_b)]$

$b = (\varepsilon, x, P(x)\&[E(a_x)\&\textit{akai}'(x)(a_x)])$
$a_b = (\varepsilon, s, E(s)\&\textit{akai}'(b)(s))$
$a_x = (\varepsilon, s, E(s)\&\textit{akai}'(x)(s))$

The formula (28) is the final representation for the interpretation of the pre-*no* clause *akai*.

Now, it is time to parse *no*; a parser copies a type-e term and pastes it at a type-e node across a LINK relation. In (29), what is copied is the non-event term "b" in the evaluated proposition.[4]

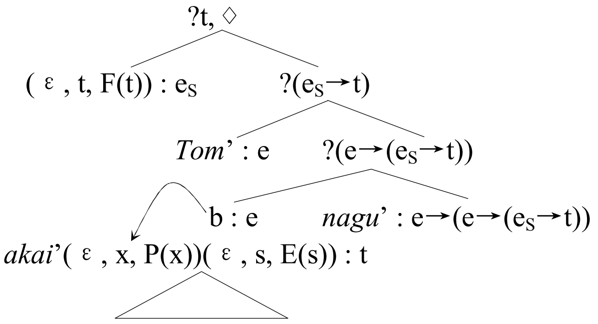
(29) Parsing *Akai no*



$b = (\varepsilon, x, P(x)\&[E(a_x)\&\textit{akai}'(x)(a_x)])$
$a_x = (\varepsilon, s, E(s)\&\textit{akai}'(x)(s))$

The node decorated with "b" becomes an object node by the lexical actions of the accusative case particle *o*. Then, the matrix predicate *nagu* (= 'hit') constructs a propositional template; in (30), the event node is decorated with ( ε , t, F(t)), the subject node is decorated with *Tom*', and the object node is decorated with "b". (As for the object node, the node decorated with "b" in (29) collapses with the object node introduced by *nagu*.)

(30) Parsing [*Akai no*]-*o Tom-ga nagu*

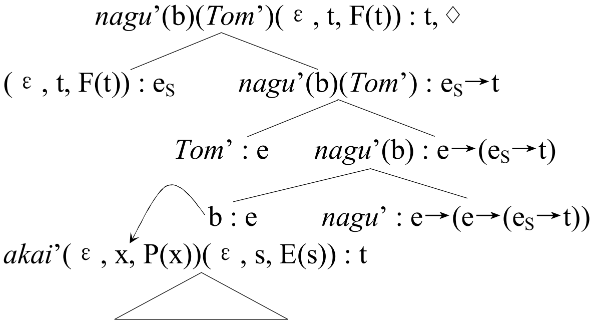


$b = (\varepsilon, x, P(x)\&[E(a_x)\&\textit{akai}'(x)(a_x)])$
$a_x = (\varepsilon, s, E(s)\&\textit{akai}'(x)(s))$

Finally, the general action ELIMINATION is run three times. The past tense marker *tta* being set aside, the tree (31) is the final state, and the top node represents the indefinite reading of the string (23): 'Tom hit a red one.' (For the definite reading of (23), see Section 4.2.)

[4] A parser could copy the event term "$a_b$" but it leads to tree transition crash, since the matrix predicate *nagu* (= 'hit') cannot take an event term as an argument. As for "$a_x$", a parser cannot copy it, since it is not a full-blown term in that the variable "x" is not bound within the term; "$a_x$" is part of the evaluated non-event term "b".

(31) ELIMINATION



$b = ( \varepsilon , x, P(x)\&[E(a_x)\&akai'(x)(a_x)])$
$a_x = ( \varepsilon , s, E(s)\&akai'(x)(s))$

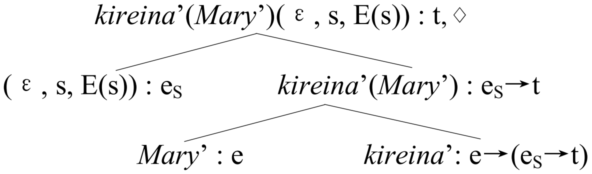
### 3.3 Situation Nominalization

Let us move on to situation nominalization. The example (2) is repeated here as (32).

| | | | |
|---|---|---|---|
| (32) | [*Mary-ga* | *kireina* | *no*]-*o* |
| | [Mary-NOM | beautiful | NO]-ACC |
| | *Tom-ga* | *shi-tteiru*. | |
| | Tom-NOM | know-PRES | |
| | 'Tom knows that Mary is beautiful.' | | |

As always, the initial state of tree transitions is set out by the AXIOM. Given the tree transitions in the last sub-section, the parse of (32) prior to *no* yields the tree (33).

(33) Parsing *Mary-ga kireina*



The lexical actions of *kireina* (= 'beautiful') builds up a propositional structure with two slots. The event slot is filled by the event term ( ε , s, E(s)), and the subject slot collapses with the node that has been created by the parse of *Mary-ga*.

The top node in the tree (33) is decorated with the proposition, which is re-cited here as (34). This proposition is subject to Q-EVALUATION, and the proposition (35) is engendered.

(34) *kireina*'(*Mary*')( ε , s, E(s))

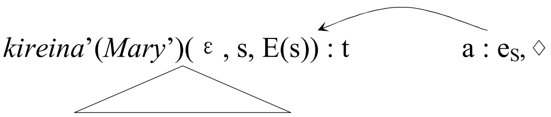
(35) Evaluating the event term ( ε , s, E(s))

E(a)&*kireina*'(*Mary*')(a)

a = ( ε , s, E(s)&*kireina*'(*Mary*')(s))

Next, *no* copies the evaluated event term "a" and pastes it at a node across a LINK relation.[5]

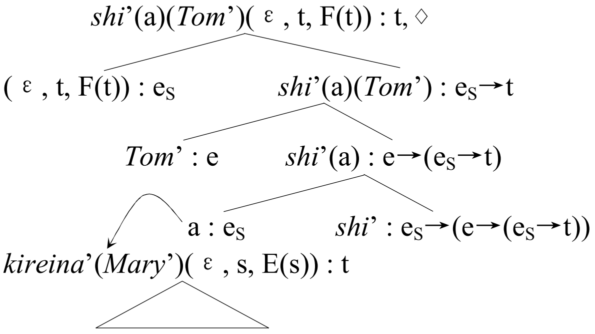
(36) Parsing *Mary-ga kireina no*



a = ( ε , s, E(s)&*kireina*'(*Mary*')(s))

The current node in (36) is fixed as an object node by the accusative case particle *o*, and the parse of *Tom-ga* creates a subject node. These two nodes collapse with the nodes introduced by the predicate *shi* (= 'know'). After ELIMINATION is run three times, the tree (36) is updated into (37).

(37) Parsing [*Mary-ga kireina no*]-*o Tom-ga shi-tteitu*



a = ( ε , s, E(s)&*kireina*'(*Mary*')(s))

This is a final state of the tree transitions, and the root node represents the interpretation of the string (32): 'Tom knows that Mary is beautiful.'

[5] A parser could copy another type-e term: the evaluated term for *Mary*. (For this purpose, *Mary* is mapped onto an iota term.) In fact, copying of this term leads to Cann et al.'s (2005) analysis of head-internal relatives. However, the string in question cannot be so interpreted due to the Relevancy Condition (Kuroda, 1992: p.147).

## 4 Consequences

### 4.1 *No* as a Dependent Item

Makino (1968: p.51) observes that *no* cannot stand on its own. Compare (38) with (1)/(23).

| | | | |
|---|---|---|---|
| (38) | **No-o* | *Tom-ga* | *nagu-tta.* |
| | NO-ACC | Tom-NOM | hit-PAST |

Makino considers only participant nominalization, but it is also true of situation nominalization. (39) should be compared with (2)/(32).

| | | | |
|---|---|---|---|
| (39) | **No-o* | *Tom-ga* | *shi-tteiru.* |
| | NO-ACC | Tom-NOM | know-PRES |

These data are amenable to my analysis. The entry of *no* requires that a proposition should have been constructed before the parse of *no*. Formally, this requirement is expressed in the two IF-clauses in the entry of *no* in (19). In (38, 39), however, no items precede *no* in the strings, and a parser cannot build up a proposition before processing *no*.

### 4.2 Indeterminacy of Denotation

Denotation of the *no*-headed part is indeterminate in two respects. Firstly, as shown in (1), repeated here as (40), it is indeterminate with regard to the definiteness of the denotation.

| | | | | |
|---|---|---|---|---|
| (40) | [*Akai* | *no*]-*o* | *Tom-ga* | *nagu-tta.* |
| | [red | NO]-ACC | Tom-NOM | hit-PAST |
| | ‘Tom hit a/the red one.’ | | | |

In Section 3.2, it was argued that the parse of *Akai no* yields the epsilon term (41).

(41) ( ε , x, P(x)&*akai*’(x))

Since DS is not encapsulated in Fodor’s (1983) sense, pragmatics comes in during DS tree growth. For the model of pragmatics, I assume Relevance Theory (Sperber and Wilson, 1995). Thus, if it is inferable that the speaker has in mind a definite entity, a parser may strengthen the epsilon operator ε in (41) as the iota operator ι , as in (42).

(42) ( ι , x, P(x)&*akai*’(x))

This models the definite reading of the string (40) à la Russellian treatment of definite descriptions (Russell, 1905).

Secondly, the content of the *no*-headed part is indeterminate. So, when it is pragmatically inferred that a speaker has in mind a specific entity, say, a red person, the term (41) may be enriched as (43), where *hito*’ is the content of *hito* (= ‘person’).

(43) ( ε , x, P(x)&[*akai*’(x)&*hito*’(x)])

These two types of indeterminacies are captured in my analysis, since pragmatic inference interacts with DS structure building.

### 4.3 Expressivity

It is well known that if the *no*-headed part denotes a human in participant nominalization, derogatory expressivity is observed (Kitagawa, 2005: p.1259). Consider (1, 2, 3), repeated here as (44, 45, 46); expressivity is found in participant nominalization (44, 46a), but not in situation nominalization (45, 46b).

| | | | | |
|---|---|---|---|---|
| (44) | [*Akai* | *no*]-*o* | *Tom-ga* | *nagu-tta.* |
| | [red | NO]-ACC | Tom-NOM | hit-PAST |
| | ‘Tom hit a/the red one.’ | | | |

| | | | |
|---|---|---|---|
| (45) | [*Mary-ga* | *kireina* | *no*]-*o* |
| | [Mary-NOM | beautiful | NO]-ACC |
| | *Tom-ga* | *shi-tteiru.* | |
| | Tom-NOM | know-PRES | |
| | ‘Tom knows that Mary is beautiful.’ | | |

| | | | | |
|---|---|---|---|---|
| (46) | [*Nai-ta* | *no*]-*o* | *Tom-ga* | *mi-ta.* |
| | [cry-PAST | NO]-ACC | Tom-NOM | see-PAST |

a. ‘Tom saw someone who cried.’

b. ‘Tom saw the event of someone’s having cried.

What has not been reported in the literature is that expressivity is not always derogatory. To take (44) as an example, if the denoted person’s face turns red after a pint of beer and the speaker hits the person in jest, expressivity may be “affectionate familiarity with the denoted person”. Any adequate account of *no* must model this context-dependency of expressivity (Yuji Nishiyama, p.c.).

To account for the above data, I shall posit the constraint that the denotation of the *no*-headed part should be an object (rather than a human), the idea being that if the *no*-headed part denotes a human, expressivity emerges through pragmatic inference.[6]

First, in (44), given the predicate *nagu* (= 'hit'), a parser expects that *akai no* denotes a human, and constructs, say, the term (47), which denotes a red person (cf. §4.2).

(47) ( ε , x, P(x)&[*akai*'(x)&*hito*'(x)])

That the term (47) denotes a human indicates that the speaker treats a denoted person as if s/he were a thing, which has a pragmatic implication that the speaker does not treat the person respectfully. This pragmatic inference yields derogatory expressivity.

This pragmatic analysis naturally accounts for the context-dependence of expressivity. Consider the context where the speaker is a good friend of the denoted person. In this context, that the term (47) denotes a human indicates that the speaker frankly describes a person, which has a pragmatic implication that the speaker shows a sign of close friendship. In this case, the type of expressivity is affectionate familiarity with the denoted person. This pragmatic analysis is extendable to (46a).

It is predicted that if the *no*-headed part denotes a non-human, expressivity should be absent:

(48) [*Akai* *no*]-*o* *Tom-ga* *tabe-ta.*
[red NO]-ACC Tom-NOM eat-PAST
'Tom ate a/the red one.'

In (48), due to the predicate *tabe* (= 'eat'), the term copied by *no* denotes a non-human (e.g. apple). So, the pragmatic inference mentioned above is not triggered, and expressivity is not engendered.

Next, how about the absence of expressivity in (45, 46b)? In these cases, *no* copies an event term (cf. §3.3). Since an event is not a human, the pragmatic inference mentioned above does not take place, and expressivity does not emerge.

The present account has some implications for a cross-linguistic study of nominalization. Consider (49), the Korean counterpart of (46).

(49) [*Wu-nun* *kes*]-*ul*
[cry-MOD KES]-ACC
*Tom-i* *pwa-ss-ta.*
Tom-NOM see-PAST-DECL
a. *'Tom saw someone who cried.'
b. 'Tom saw the event of someone's having cried.

While (49b) is acceptable, (49a) is not[7]. Of note is that, unlike *no*, the nominalizer *kes* derived from the noun *kes* meaning 'thing', and that this lexical meaning somehow persists in the nominalizer *kes* (Horie, 2008: p.178). So, the restriction that the denoted entity be an object is stronger in *kes* than in *no*; this is why the reading (46a) in Japanese is possible but the reading (49a) in Korean is not.

In closing, let me examine some previous works that are relevant to the present discussion. Firstly, McGloin (1985) also suggests, albeit very briefly, a pragmatic analysis of expressivity. However, in her analysis, neither situation nominalization nor the context-dependency of expressivity is treated.

Second, from the perspective of the Principles-and-Parameters Theory, Kitagawa (2005) suggests that expressivity emerges only if the external-head pro has an indefinite referent. However, suppose that (50) is uttered with a pointing gesture; further, the demonstrative *sono* (= 'that') is used in order to ensure that the small pro has a definite referent.

(50) *Sono* [*akai* *no*]-*o*
that [red NO]-ACC
*Tom-ga* *nagu-tta.*
Tom-NOM hit-PAST
'Tom hit that red one.'

In (50), expressivity is still observed, contrary to what Kitagawa (2005) would predict. My analysis

[6] This constraint may be modeled along the lines with Cann and Wu's (2011) analysis of the *bei* construction in Chinese. They argue that *bei* marks the pre-*bei* item as the locus of affect; *bei* projects a propositional structure where the Locus-of-Affect (LoA) predicate takes as an internal argument the content of the pre-*bei* item, and as an external argument the content of the rest of the string. In their analysis, the LoA predicate is underspecified for the type of affect, and thus it fits well with the context-dependency of *no*-expressivity. I shall assume that the entry of *no* has a constraint that if a term to be copied does not denote an object, it projects a structure involving the LoA predicate. This ramification of the entry of *no* is not attempted in this paper.

[7] The degraded status of (49a) does not mean that *kes* lacks participant nominalization. In fact, if *wu-nun* in (49) is replaced with *kkayeci-nun* (= break-MOD), the string exhibits the participant-nominalization reading: 'Tom saw something (e.g. machine) that was being broken.'

postulates neither a null element nor an external-head position; the presence of expressivity in (50) is expected as a result of pragmatic inference.

## 4.4 Nature of Denotation

In Kamio (1983) and McGloin (1985), it is stated that *no* in participant nominalization cannot refer to abstract entities. Consider the contrast between (51) and (52) (Kamio, 1983: p.82).

(51) [[*katai shinnen*]-*o* *motta*] *hito*
[[solid belief]-ACC have] person
'a person who has a solid belief'

(52) *[[*katai no*]-*o* *motta*] *hito*
[[solid NO]-ACC have] person
Int. 'a person who has a solid belief'

The string (52) is acceptable if the *no*-headed part is meant to denote some non-abstract entity, such as a stone.

It seems, however, that the above generalization is suspicious. In (52), the use of the predicate *katai* (= 'solid') is metaphorical; it drives the interpreter to look for a physical object to which the predicate *katai* normally applies (e.g. stone). This is why it is hard to get the intended interpretation in (52). If a predicate that is congruous with an abstract object is used, such as *settokutekina* (= 'convincing'), the *no*-headed part may denote an abstract entity:

(53) [*gakkai-de* [*settokutekina no*]-*o*
[conference-at [convincing NO]-ACC
*teijishita*] *hito*
presented] person
'a person who presented a convincing
one (e.g. argument) at a conference'

Given my unitary analysis of *no*, it is expected that if the *no*-headed part may denote an abstract entity in participant nominalization, it should also hold of situation nominalization. This expectation is confirmed. First, consider (54).

(54) *Tom-wa* [[*ni tasu ni*]-*ga*
Tom-TOP [[2 plus 2]-NOM
*yon dearu no*]-*o* *shitteiru*
4 COPULA NO]-ACC know
'Tom knows that 2 plus 2 equals 4.'

In this example, the *no*-headed part denotes the abstract proposition that 2 plus 2 equals 4. Second, as pointed out by an anonymous reviewer, modal statements, which seem to denote propositions, can be nominalized by *no*. This is illustrated in (55).

(55) [*Mary-ga kuru kamoshirenai*
[Mary-NOM come might
*no*]-*o* *omoidashita.*
NO]-ACC remembered
'I remembered that Mary might come.'

But there is some indication that *no* in situation nominalization tends to denote a perceptible event. Kuno (1973: p.222) notes that in (56), if *no* is used, it denotes Tom's death as a tangible event, whereas if the situation nominalizer *koto* is employed, it denotes Tom's death as a less tangible event. (See also Watanabe (2008).)

(56) [*John-ga shinda no*/*koto*]-*wa*
[John-NOM died NO/KOTO]-TOP
*tashika desu.*
certain COPULA
'It is certain that John has died.'

I contend that this difference between *no* and *koto* reflects the origins of these two items. As noted in Horie (2008: p.174), there are no confirmed lexical origins for *no*, but *koto* is a diachronically bleached development of the noun *koto*, meaning 'matter' or 'event'. It may then be assumed that *koto* retains the property of denoting an event as a matter, and that this lexical residue is encoded as a constraint in the nominalizer *koto* (but not in the nominalizer *no*). Then, the difference in (56) can be analyzed as the difference in the encoded constraints of *koto* and *no*. But this reasoning raises another problem: as shown below, *koto* does not exhibit participant nominalization; compare (57) with (44).

(57) *[*Akai koto*]-*o* *Tom-ga nagu-tta*.
[red KOTO]-ACC Tom-NOM hit-PAST

As stated above, the nominalizer *kes* in Korean, which also derived from the noun meaning 'thing', allows not only situation but also participant nominalization. This functional difference between *koto* and *kes* is a remaining issue.

## 5 Conclusion

This article has proposed an integrated analysis of *no*-nominalization within Dynamic Syntax, and has accounted for a number of characteristics of the phenomenon. The particle *no* is assigned a single lexical entry, and the participant/situation divide boils down to an outcome of semantic tree growth, more specifically, a parser's choice of what type-e term it copies. In this account, incrementality is a key notion, as the participant/situation distinction arises at the timing of processing *no*.

## Acknowledgments

I would like to thank Ronnie Cann, David Cram, Stephen Horn, Ruth Kempson, Jieun Kiaer, Yuji Nishiyama, and the anonymous PACLIC reviewers for helpful suggestions. I am grateful to Aimi Kuya and Eun Hyuk Chang for discussion of Korean examples. Any remaining inadequacies are solely my own.

## References

Cann, Ronnie. 2011. Towards an Account of the Auxiliary System in English. In Kempson, R. et al. (eds.) The Dynamics of Lexical Interfaces. CSLI, Stanford.

Cann, Ronnie, Kempson, Ruth, and Marten, Lutz. 2005. The Dynamics of Language. Elsevier, Oxford.

Cann, Ronnie, Kempson, Ruth, and Purver, Matthew. 2007. Context-dependent Well-formedness. Research on Language and Computation, 5: 333-358.

Cann, R. and Wu, Y. 2011. The *Bei* Construction in Chinese. In Kempson, R. et al. (eds.) The Dynamics of Lexical Interfaces. CSLI, Stanford.

Davidson, Donald. 1967. The Logical Form of Action Sentences. In Rescher, N. (ed.) The Logic of Decision and Action. University of Pittsburgh Press, Pittsburgh.

Fodor, Jerry. 1983. The Modularity of Mind. The MIT Press, Cambridge, MA.

Gregoromichelaki, Eleni. 2011. Conditionals in Dynamic Syntax. In Kempson, R. et al. (eds.) The Dynamics of Lexical Interfaces. CSLI, Stanford.

Horie, K. 2008. The Grammaticalization of Nominalizers in Japanese and Korean. In López-Couso, M. J. and Seoane, E. (eds.) Rethinking Grammaticalization. John Benjamins, Amsterdam.

Kamio, Akio. 1983. Meeshiku no Koozoo. (Structure of Noun Phrases) In Inoue, K. (ed.) Nihongo no Koozoo. (Structure of Japanese) Sanseido, Tokyo.

Kempson, Ruth and Kurosawa, Akiko. 2009. At the Syntax-Pragmatics Interface. In Hoshi, H. (ed.) The Dynamics of Language Faculty. Kuroshio, Tokyo.

Kempson, Ruth, Meyer-Viol, Wilfried, and Gabby, Dov. 2001. Dynamic Syntax. Blackwell, Oxford.

Kitagawa, Chisato. 2005. Typological Variants of Head-internal Relatives in Japanese. Lingua, 115: 1243-1276.

Kitagawa, Chisato and Ross, Claudia. 1982. Prenominal Modification in Chinese and Japanese. Linguistic Analysis, 9: 19-53.

Kuno, Susumu. 1973. The Structure of the Japanese Language. The MIT Press, Cambridge, MA.

Kuroda, Shigeyuki. 1992. Japanese Syntax and Semantics. Kluwer, Dordrecht.

Makino, Seiichi. 1968. Some Aspects of Japanese Nominalization. Tokai University Press, Kanagawa, Japan.

McGloin, Naomi. 1985. *No*-pronominalization in Japanese. Papers in Japanese Linguistics, 10: 1-15.

Murasugi, Keiko. 1991. Noun Phrases in Japanese and English. Ph.D. Dissertation, UConn.

Purver, Matthew, Cann, Ronnie, and Kempson, Ruth. 2006. Grammars as Parsers. Research on Language and Computation, 4: 289-326.

Russell, Bertrand. 1905. On Denoting. Mind, 14: 479-493.

Seraku, Tohru. in press. An Incremental Semantics Account of the Particle *No* in Japanese. Proceedings of the Western Conference on Linguistics 2011.

Shibatani, Masayoshi. 2009. Elements of Complex Structures, where Recursion Isn't. In Givon, T. and Shibatani, M. (eds.) Syntactic Complexity. John Benjamins, Amsterdam.

Sperber, Dan and Wilson, Deirdre. 1995. Relevance, 2nd edn. Blackwell, Oxford.

Tonoike, Shigeo. 1990. *No* no Ronrikeeshiki. (LF Representation of *No*) Meijigakuin Ronsou, 467: 69-99.

Watanabe, Yukari. 2008. Bunhogohyooshiki "Koto" "No" no Imitekisooi nikansuru Kenkyuu. (Study of Semantic Differences between Complementizer "Koto" and "No") Keisuisya, Hiroshima, Japan.