IEICE TRANS. INF. & SYST., VOL.E99–D, NO.9 SEPTEMBER 2016

PAPER

Incremental Semantic Construction Based on Combinatory Categorial Grammar

Yoshihide KATO[†a)], *Member and* **Shigeki MATSUBARA**[††], *Senior Member*

SUMMARY This paper proposes a method of incrementally constructing semantic representations. Our method is based on Steedman's Combinatory Categorial Grammar (CCG), which has a transparent correspondence between syntax and semantics. In our method, a derivation for a sentence is constructed in an incremental fashion and the corresponding semantic representation is derived synchronously. Our method uses normal form CCG derivation. This is the difference between our approach and previous ones. Previous approaches use most left-branching derivation called incremental derivation, but they cannot process coordinate structures incrementally. Our method overcomes this problem.

key words: *incremental interpretation, normal form derivation, λ-calculus, incremental parsing*

1. Introduction

By incremental interpretation, we mean that a sentence is analyzed from left to right, and a fully-connected semantic representation is assigned to each initial fragment of the sentence[*]. These properties enable NLP systems to analyze unfinished sentences. Moreover, incremental interpretation is useful for incremental dialogue systems [5]–[8]. Furthermore, in the field of psycholinguistics, incremental interpretation has been explored as a human sentence processing model.

This paper[**] proposes a method of constructing a semantic representation for each initial fragment of a sentence in an incremental fashion. The proposed method is based on Combinatory Categorial Grammar (CCG) [9]. CCG has a transparent correspondence between syntax and semantics, and represents the syntactic process as a derivation which is a tree structure. Our method constructs a CCG derivation by applying operations used in incremental phrase structure parsing. Each intermediate data structure constructed by the operations represents partial information of some derivation. We formally define how to construct a semantic representation from the intermediate structure. Our method can obtain a semantic representation for any initial fragments. Since the obtained semantic representations conform to the CCG semantic construction, we can expect that incremental semantic interpretation is realized by applying a CCG-based semantic analysis such as the literature [10].

Manuscript received September 3, 2015.
Manuscript revised March 31, 2016.
Manuscript publicized June 2, 2016.

[†]The author is with Information & Communications, Nagoya University, Nagoya-shi, 464–8601 Japan.
[††]The author is with Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.
a) E-mail: yoshihide@icts.nagoya-u.ac.jp

DOI: 10.1587/transinf.2015EDP7355

This paper is organized as follows: Sect. 2 briefly explains Combinatory Categorial Grammar. Section 3 gives an overview of previous work on CCG-based incremental parsing and discusses its problem. Section 4 proposes our CCG-based method of incrementally constructing semantic representations. Section 5 reviews related work and Sect. 6 concludes this paper.

2. Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) [9] is a grammar formalism which has a transparent correspondence between syntax and semantics. Syntactic information is represented using basic categories (e.g., S, NP) and complex categories. Complex categories are in the form of X/Y or $X\backslash Y$, where X and Y are categories. Intuitively, each category in the form of X/Y means that it receives a category Y from its right and returns a category X. In the case of the form $X\backslash Y$, the direction is to left. For example, the category of a transitive verb is (S\NP)/NP, which receives an object NP from its right and returns a category S\NP. The category S\NP corresponds to a verb phrase. It receives a subject NP from its left and the result is a sentence S. Formally, categories are combined using CCG rules such as the ones shown in Fig. 1. Each rule means that, when the elements of the left-hand side of the arrow are combined in this order, the result is the right-hand side. The symbol with which the arrow is subscripted designates its rule type. Each element consists of a syntactic category and a semantic representation which is separated by a colon. A semantic representation is a λ-term. Each combination of syntactic categories has a corresponding semantic composition of their semantic representations. Figure 2 shows an example of CCG derivation, which is taken from the literature [9][***]. Here, we write

[*]The term "incremental" has been used in different ways. For example, in the literatures [1], [2] the term means the first property only. Here, we use the term to refer to *strong incrementality* [3] or *strict incrementality* [4], that is, the second property is required.

[**]A preliminary version of this paper was presented at the 4th Joint Conference on Lexical and Computational Semantics (*SEM 2015) as "Incremental semantic construction using normal form CCG derivation" by the same authors.

[***]For simplicity, we use a symbol for a semantic representation of a word. Note that it is allowed to use complex semantic representations. For example, by assigning $\lambda px.\Diamond(px)$ ($\Diamond$ is possibility operator.) and $\lambda pq.p \wedge q$ to "might" and "and" respectively, we can obtain a modal logic formula $\Diamond(\mathsf{marry}'\mathsf{manny}'\mathsf{anna}') \wedge \mathsf{meet}'\mathsf{manny}'\mathsf{anna}'$.

Forward function application:	$X/Y : f$	$Y : a$	$\Rightarrow_{>}$	$X : fa$
Backward function application:	$Y : a$	$X\backslash Y : f$	$\Rightarrow_{<}$	$X : fa$
Forward function composition:	$X/Y : f$	$Y/Z : g$	$\Rightarrow_{>B}$	$X/Z : \lambda x.f(gx)$
Backward function composition:	$Y\backslash Z : g$	$X\backslash Y : f$	$\Rightarrow_{<B}$	$X\backslash Z : \lambda x.f(gx)$
Backward crossed substitution:	$Y/Z : g$	$(X\backslash Y)/Z : f$	$\Rightarrow_{<S_\times}$	$X/Z : \lambda x.fx(gx)$
Forward type-raising:	$X : a$		$\Rightarrow_{>T}$	$T/(T\backslash X) : \lambda f.fa$
Backward type-raising:	$X : a$		$\Rightarrow_{<T}$	$T\backslash(T/X) : \lambda f.fa$
Coordination:	$X : f$	$CONJ : b \quad X : g$	$\Rightarrow_{<\Phi>}$	$X : \lambda \ldots b(g \ldots)(f \ldots)$

Fig. 1 CCG rules

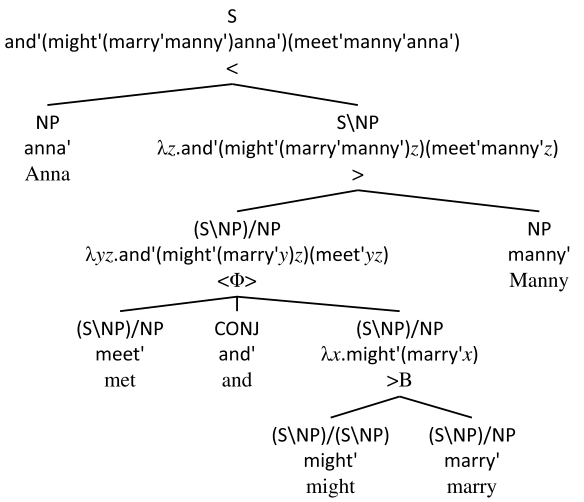

Fig. 2 An example of CCG derivation.

$\lambda x_1 x_2 \cdots x_n.M$ and $M_1 M_2 M_3 \cdots M_n$ to abbreviate λ-terms $(\lambda x_1.(\lambda x_2.(\cdots(\lambda x_n.M)\cdots)))$ and $((\cdots((M_1 M_2)M_3)\cdots)M_n)$, respectively. In this example, each node has three labels: a syntactic category, a semantic representation and the rule type which is used to derive this node. For each leaf node, a word is assigned instead of a rule type.

3. Incremental Parsing Based on CCG

Incremental parsing methods based on CCG have been proposed so far [11]–[13]. By using the property that CCG allows non-standard constituents, previous CCG-based incremental parsers assign a syntactic category to each initial fragment of an input sentence. The obtained derivations are most left-branching ones which are called *incremental derivations*. Figure 3 shows two examples of incremental derivations. In Fig. 3 (a), the fragment "Anna met" is a non-phrase, but it has a syntactic category `S/NP`.

However, Demberg [4] has demonstrated that some kinds of sentences cannot have strictly left-branching derivations. This means that previous approaches have the case where the parser cannot assign any syntactic categories to an initial fragment. This also means that such initial fragments do not have any semantic representations.

A typical example is coordinate structure. In CCG, a coordinate structure is derived by combining conjuncts and a conjunction using coordination rule. This prevents the first conjunct from combining with its left constituent. As an example, let us consider the incremental derivation shown in Fig. 3 (b). Here, the word "met" is the first conjunct of "met and might marry" and cannot be combined with "Anna". If we assign the category `S/NP` to initial fragment "Anna met" as shown in Fig. 3 (a), the word "met" cannot be treated as a conjunct. This example demonstrates that sentences including coordinate structures cannot be represented by any strictly left-branching derivations. That is, incremental derivation approaches cannot achieve a word-by-word incremental interpretation.

4. Incremental Semantic Construction Based on CCG

This section proposes a method of constructing semantic representations in an incremental fashion. To overcome the problem described in the previous section, our method adopts a different approach. Our method needs not to use incremental derivations. For each initial fragment of a sentence, our proposed method obtains a semantic representation from the normal form derivation. A normal form derivation is defined as the one which uses type-raising and function composition only if they are required[†]. We consider a derivation as a parse tree and construct it based on incremental phrase structure parsing. For each initial fragment of a sentence, incremental parsing can construct a partial parse tree which connects all words in the fragment. Our method obtains a semantic representation from the partial parse tree. In the constructed partial parse tree, some parts of the derivation are underspecified. Our method introduces variables to denote underspecified parts of the semantic representation. These variables are replaced with semantic representations as soon as they are determined. In the rest of this section, we first describe incremental parsing which is the basis of our method. Next, we explain how to obtain a semantic representation from a partial parse tree constructed by incremental parsing.

4.1 Incremental Construction of CCG Derivation

Our method considers a CCG derivation as a tree structure. We call this *parse tree*. Our method constructs a parse tree according to an incremental parsing formalism proposed by Kato and Matsubara [16]. This formalism extends Collins and Roark's incremental parsing [17] by introducing adjoining operation used in Tree Adjoining Grammar [18]. The

[†]Several variants of normal form have been presented. For example, see the literatures [14], [15].

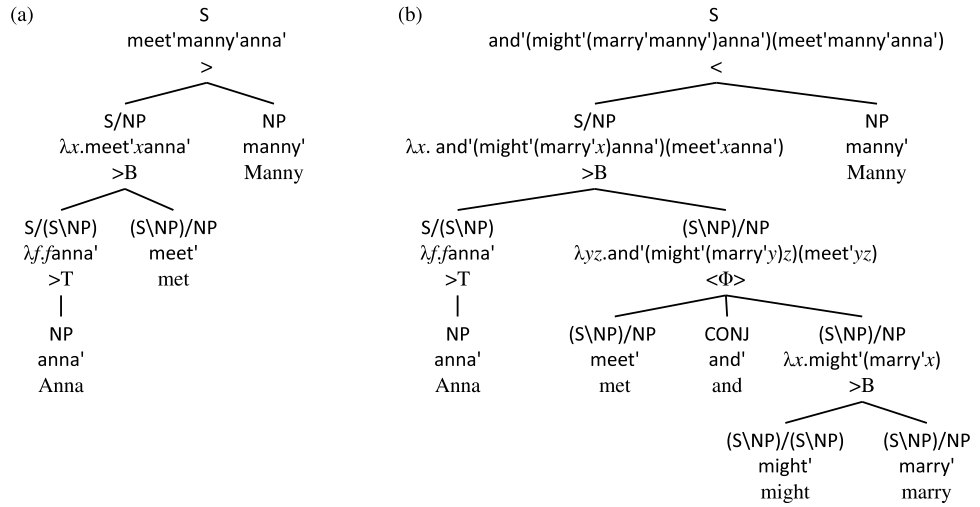

Fig. 3 Incremental derivations.

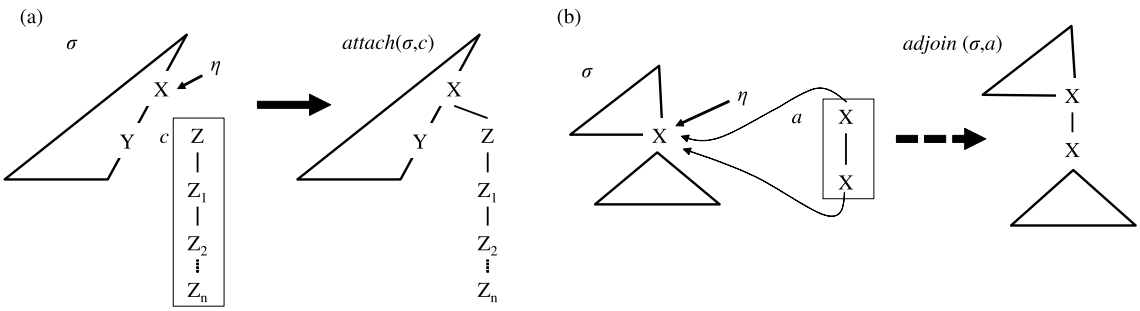

Fig. 4 Attaching operation and adjoining operation.

incremental parsing assigns partial parse trees for any initial fragments of a sentence. Adjoining operation reduces local ambiguity caused by left-recursive structure, and improves the parsing accuracy (for more detail, see the literature [16]). Furthermore, in the field of psycholinguistics, adjoining operation is introduced to a human sentence processing model [19]–[21].

4.1.1 A Formal Description of Incremental Parsing

This section gives a formal description of Kato and Matsubara's incremental parsing. The parsing grammar consists of three types of elements: *allowable tuples*, *allowable chains* and *auxiliary trees*. Each allowable tuple is a 3-tuple $\langle X, Y, Z\rangle$ which means that the grammar allows a node labelled with Z to follow a node labelled with Y under its parent labelled with X. Each allowable chain is a sequence of labels. This corresponds to a sequence of labels on a path from a node to its leftmost descendant leaf in a parse tree[†]. Each auxiliary tree consists of two nodes: a root and a foot. The label of a root is the same as that of its foot.

[†]Allowable chain is similar to *spine* in spinal parsing [2], [22]. A spine is also a sequence of labels which represents a head projection path. However, spinal parsing does not achieve strict incrementality.

A parse tree is constructed by applying two operations: *attaching* and *adjoining*. Attaching operation combines a partial parse tree and an allowable chain. The operation is defined as follows:

attaching: Let σ be a partial parse tree and c be an allowable chain. Let η be the attachment site of σ. $attach(\sigma, c)$ is the result of attaching c to η as the rightmost child (see Fig. 4 (a)).

Let X, Y and Z be the label of η, the label of the rightmost child of η and the label of the root of c. If a grammar does not have allowable tuple $\langle X, Y, Z\rangle$, $attach(\sigma, c)$ is not allowed by the grammar. Next, we give the definition of adjoining operation. Adjoining operation inserts an auxiliary tree into a partial parse tree. The operation is defined as follows:

adjoining: Let σ be a partial parse tree and a be an auxiliary tree. Let η be the adjunction site of σ. $adjoin(\sigma, a)$ is the result of splitting σ at η and combining the upper tree of σ with the root of a and the lower tree of σ with the foot of a (see Fig. 4 (b)). If the label of η is not the same as that of the foot of a, $adjoin(\sigma, a)$ is undefined.

Here, we give the definitions of attachment site and adjunction site. These sites are defined in order to construct a parse tree from left to right. We say that a node η is *complete*

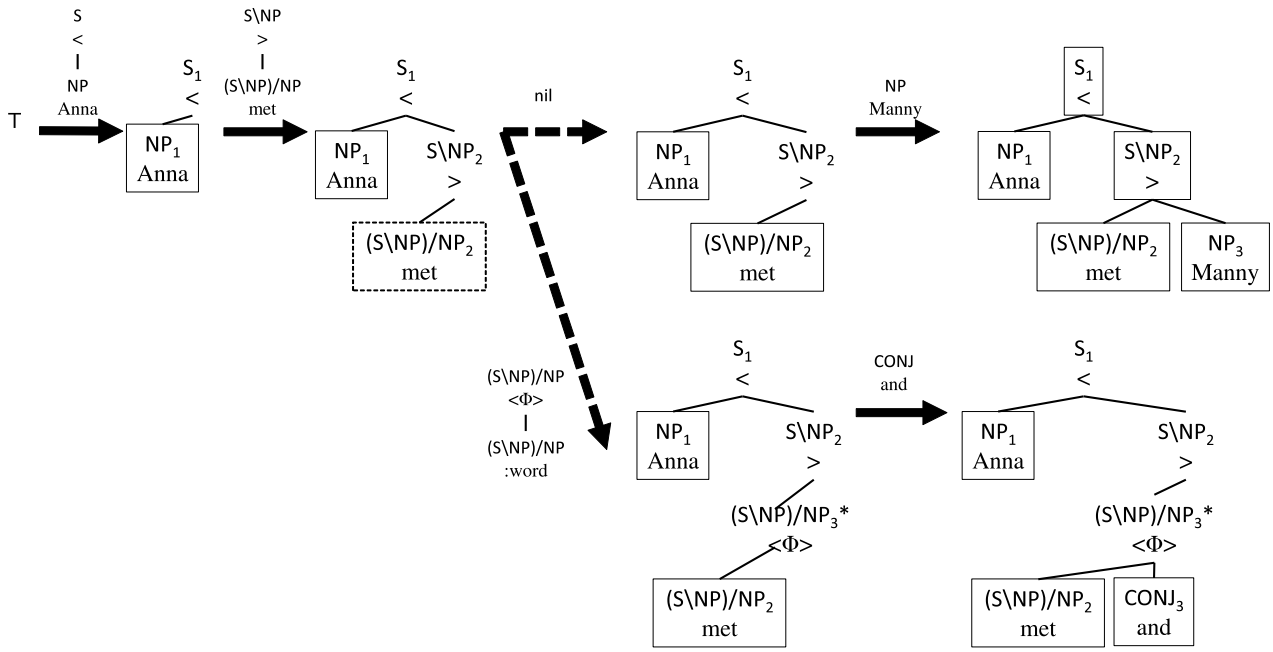

Fig. 5 Incremental constructing process of CCG derivations.

if η satisfies the following conditions:

- All children of η are instantiated and complete[†].
- Adjoining operation is not applicable to η. By the term "applicable", we mean that the grammar has an auxiliary tree whose foot label is identical to that of η and adjoining operation has not been applied to η yet.

The attachment site of σ is defined as the node η satisfying the following conditions:

- Not all children of η are instantiated.
- All instantiated children of η are complete.

The adjunction site of σ is defined as the node η satisfying the following conditions:

- All children of η are instantiated and complete.
- Adjoining operation is applicable to η.

Finally, we introduce *nil-adjoining operation* which changes not a partial parse tree, but node states. When the operation is applied to a node, we deem that adjoining operation is applied to the node. This affects whether or not each node in the partial parse tree is complete. The symbol nil designates the operation.

4.1.2 Constructing CCG Derivations

First of all, we show an example of incremental constructing process of CCG derivations in our proposed method. See Fig. 5. Attaching operation is represented as a solid arrow labelled with an allowable chain. Adjoining operation is represented as a dotted arrow labelled with an auxiliary tree. The subscript i of a node indicates that the node is instantiated at the point when i-th word w_i is consumed. The solid boxes mean that the nodes are complete. The dotted box represents that adjoining operation is applicable to the node. The symbol '$*$' means that the annotated node is introduced by adjoining operation (This node corresponds to the root of the auxiliary tree.). We call it *adjoined node*. Each node in a partial parse tree is labelled with a syntactic category and a rule type (or a word). No semantic representations are assigned. This is because each partial parse tree includes underspecified parts and it is impossible to determine their contents. This example demonstrates that each initial fragment has a partial parse tree, which connects all the words in the fragment.

Next, we consider the parsing grammar for CCG derivation. We do not need any allowable tuples, since the CCG rules determine the syntactic category of the node which follows a node. For example, when a parent node is labelled with category S and rule type <, and its leftmost child is labelled with category NP, the following node must be labelled with S\NP. The rule type is arbitrary. Of course, we can also define allowable tuples to restrict the rule type.

Each node of the allowable chains and the auxiliary trees is also labelled with a category and a rule type as shown in Fig. 5. When an auxiliary tree a is adjoined to a partial parse tree at a node η, the label of η must be the same as that of the foot of a. That is, $cat(\eta) = cat(foot(a))$ and $rule(\eta) = rule(foot(a))$ hold. Here, we write $cat(\eta)$ and $rule(\eta)$ for the category and the rule type of a node η, re-

[†]In incremental phrase structure parsing, to identify whether or not all children are instantiated, [17] and [16] use a special symbol which means end of constituent. All children of η are instantiated if and only if the rightmost child of η is labelled with this special symbol. In CCG derivation, it can be identified by counting the number of children, since the number is uniquely determined by the rule type of η.

spectively. $foot(a)$ is the foot node of an auxiliary tree a.

4.2 Incremental Semantic Construction

This section presents our incremental semantic construction procedure. For each initial fragment, our method derives a semantic representation from the partial parse tree obtained by the incremental constructing process. The semantic representation is composed as follows:

- Construct a function t_i which adds the information about the word w_i to the semantic representation s_{i-1} for $w_1 \cdots w_{i-1}$. The function is obtained from the nodes which are instantiated at the point when the word w_i is consumed.
- Apply the function t_i to the semantic representation s_{i-1}. That is, the semantic representation for $w_1 \cdots w_i$ is $s_i = t_i(s_{i-1})$.

We call the function t_i *semantic transition function* (or *transition function* for short). The key point is how to construct the semantic transition function for a word. In the following, we explain it.

To construct a semantic transition function t_i, our method assigns a pair $\langle \alpha, M \rangle$ to each node $\eta \in N_i(\sigma)$ where $N_i(\sigma)$ is the set of the nodes in a partial parse tree σ which are instantiated at the point when i-th word w_i is consumed. Here, α is a sequence of variables and M is a semantic representation. The variables in α occur in M and represent underspecified parts of the semantic representation M. The semantic representation M conveys information about the word w_i. The variables are expected to be specified in the order of α. A transition function is obtained from a pair.

4.2.1 Semantic Construction without Adjoining Operation

For ease of explanation, we first describe the construction of transition function in the case where adjoining operation is not used. Below, $arity(R)$ is the number of the elements of the left-hand side of rule R. $C_R[M_1, \ldots, M_n]$ is the result of combining semantic representations M_1, …, M_n using rule R where n must be equal to $arity(R)$. The procedure of constructing a transition function is as follows:

1. For the leaf node $\eta \in N_i(\sigma)$, if $cat(\eta) : M$ is a lexical entry for w_i, assign $\langle \varepsilon, M \rangle$ to η.
2. Let η be an inner node in $N_i(\sigma)$. Let $\langle \alpha, M \rangle$ be the pair assigned to the child of η. Assign $\langle \alpha x_2 \cdots x_n, C_{rule(\eta)}[M, x_2, \ldots, x_n] \rangle$ to η, where $n = arity(rule(\eta))$ and $x_2, \ldots, x_n$ are fresh variables.
3. Let $\langle \alpha, M \rangle$ be the pair assigned to the highest node in $N_i(\sigma)$. The semantic transition function t_i is defined as follows:

   $$\lambda s \alpha.sM$$

   where s is a fresh variable.

By applying semantic transition functions, our method realizes incremental semantic construction. All semantic representations for initial fragments are in the form of $\lambda x \alpha'.M'$ where $x\alpha'$ is a sequence of variables designating underspecified parts in a semantic representation M' (x is the first variable.). By applying semantic transition function $\lambda s \alpha.sM$, we obtain the following semantic representation:

$$(\lambda s \alpha.sM)(\lambda x \alpha'.M') \twoheadrightarrow_\beta \lambda \alpha \alpha'.M'[x := M]$$

where $\twoheadrightarrow_\beta$ means the reflexive transitive closure of β-reduction and $M'[x := M]$ is the capture-avoiding substitution of M for x in M'. The result is in the same form. The underspecified part designated by the variable x is replaced with M which is specified by the word w_i.

As an example of our incremental semantic construction, let us consider a sentence "Anna met Manny." Figure 6 shows examples of semantic transition functions. The initial semantic representation is the identity function $\lambda x.x$. For the word "Anna", the transition function shown in Fig. 6 (a) is constructed. By applying this function to the initial semantic representation, we obtain the following semantic representation for the initial fragment "Anna":

$$(\lambda s x.s(x\mathsf{anna}'))(\lambda x.x) \twoheadrightarrow_\beta \lambda x.x\mathsf{anna}' \quad (1)$$

Next, by applying the semantic transition function for "met" which is shown in Fig. 6 (b) to the semantic representation (1), the following one is obtained for the initial fragment "Anna met":

$$(\lambda s x.s(\mathsf{meet}'x))(\lambda x.x\mathsf{anna}') \twoheadrightarrow_\beta \lambda x.\mathsf{meet}'x\mathsf{anna}' \quad (2)$$

This semantic representation captures the predicate-argument relation between anna' and meet'. Finally, by applying the semantic transition function $\lambda s.s\mathsf{manny}'$ to the semantic representation (2), we can obtain the following one:

$$\mathsf{meet}'\mathsf{manny}'\mathsf{anna}' \quad (3)$$

This semantic representation is the same as that of the normal form derivation.

4.2.2 Semantic Construction Using Adjoining Operation

In this section, we extend the transition function construction procedure to allow adjoining operation.

For $\eta \in N_i(\sigma)$ which is a node of an allowable chain, we modify steps 1 and 2 in the transition function construction procedure as follows:

- Let $\langle \alpha, M \rangle$ be the pair assigned to η in the version without adjoining operation. If adjoining operation is applicable to η, assign the pair $\langle \alpha r, rM \rangle$ to η instead of $\langle \alpha, M \rangle$ where r is a fresh variable.

Figure 6 (c) shows an example of constructing the transition function where adjoining operation is applicable to the node (S\NP)/NP. The variable r is utilized for updating a semantic representation when adjoining operation is applied to η.

For an adjoined node $\eta' \in N_i(\sigma)$, the modified procedure assigns a pair to η' in the following way:

- Let $\langle \alpha, M \rangle$ be the pair assigned to the root node of the

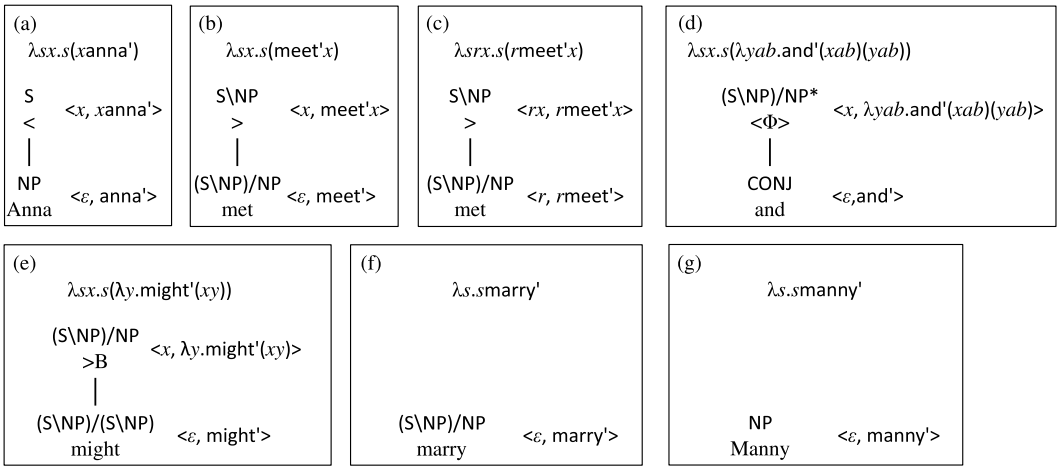

Fig. 6 Examples of semantic transition function construction.

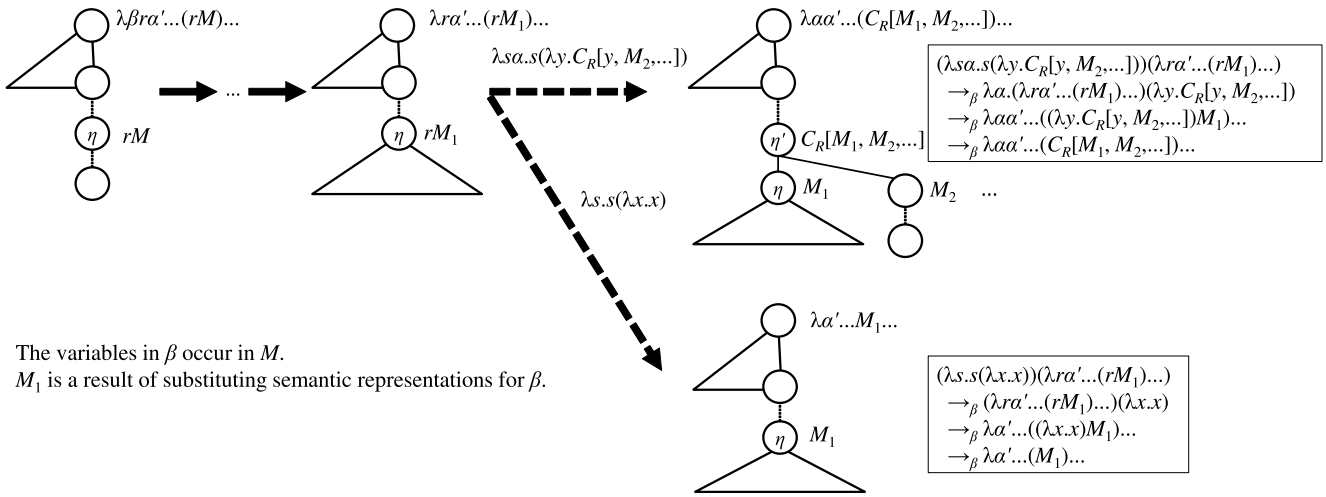

Fig. 7 Updating a semantic representation by adjoining operation.

allowable chain which is attached under η'. Let R be $rule(\eta')$ and n be $arity(R)$. If adjoining operation is applicable to η', assign the following pair to η':

$$\langle \alpha x_3 \ldots x_n r, \lambda y.rC_R[y, M, x_3, \ldots x_n] \rangle$$

Otherwise, assign the following pair to η':

$$\langle \alpha x_3 \ldots x_n, \lambda y.C_R[y, M, x_3, \ldots x_n] \rangle$$

Here, x_3,...,x_n, y and r are fresh variables.

Figure 6 (d) shows an example of constructing the transition function where the node (S\NP)/NP is an adjoined node.

The pair assignment for a node η to which adjoining operation is applicable and the one for an adjoined node η' work cooperatively (see Fig. 7). If adjoining operation is applicable to the node η, a fresh variable r is introduced to the semantic representation. At a later stage, when adjoining operation is applied to η, the variable r is replaced with a function in the form of $\lambda y.C_R[y, M_2, \ldots]$ which receives a semantic representation M_1 of η and returns the semantic representation $C_R[M_1, M_2, \ldots]$ which is the result of semantic composition. When nil-adjoining operation is applied to η, the variable r is replaced with the identity function $\lambda x.x$.

Table 1 Incremental semantic construction of "Anna met and might marry Manny."

word	#	semantic representation
Anna	1	$\lambda x.x$anna′
met	2	$\lambda rx.r$meet′$x$anna′
and	3	$\lambda yx.$and′(yxanna′)(meet′$x$anna′)
might	4	$\lambda yx.$and′(might′(yx)anna′)(meet′$x$anna′)
marry	5	$\lambda x.$and′(might′(marry′x)anna′)(meet′$x$anna′)
Manny	6	and′(might′(marry′manny′)anna′)(meet′manny′anna′)

The transition function is applied in the same way as the version without adjoining operation. Table 1 shows an example of the semantic representations constructed by our method.

As an example, let us consider the initial fragment "Anna met ..." By applying the transition function shown in Fig. 6 (c) to the semantic representation (1), we obtain the semantic representation #2 shown in Table 1.

In the case where the next word is "Manny", nil-

Table 2 Semantic representations assigned by incremental derivations.

word	semantic representation
Anna	anna′
met	–
and	–
might	–
marry	λx.and′(might′(marry′x)anna′)(meet′$x$anna′)
Manny	and′(might′(marry′manny′)anna′)(meet′manny′anna′)

adjoining operation is applied to the node (S\NP)/NP, that is, the function $\lambda s.s(\lambda x.x)$ is applied to #2. The result is identical to the semantic representation (2), therefore, we obtain the semantic representation (3) for "Anna met Manny".

Next, let us consider the case where the word "and" follows the initial fragment "Anna met." In this case, the derivation is constructed as shown in the lower side of Fig. 5. The semantic transition function for the word "and" is constructed as shown in Fig. 6 (d). By applying the function to the semantic representation #2, we obtain the semantic representation #3. Furthermore, if the word sequence "might marry Manny" follows this initial fragment, the semantic representations #4, #5 and #6 are obtained by applying the functions (e), (f) and (g) shown in Fig. 6. This example demonstrates that our method can incrementally construct semantic representations for sentences including coordinate structures. In comparison with our incremental semantic construction, incremental derivation approaches have the case where no semantic representations are assigned to initial fragments. Table 2 shows semantic representations which are assigned using incremental derivations. There exist initial fragments which have no semantic representations as discussed in Sect. 3[†].

5. Related Work

Our incremental semantic construction is based on the λ-calculus. There have been several methods of incremental semantic construction using the λ-calculus. Pulman [23] has developed an incremental parser which uses context-free rules annotated with semantic representations. The parsing process proceeds on a word-by-word basis, but its intermediate structure is a stack, that is, the parser does not assign a fully-connected semantic representation to each initial fragment. Milward [24] has proposed an incremental semantic construction method based on Categorial Grammar. The method uses two types of transition functions: state-application and state-prediction. Our semantic transition function is similar to these functions. However, our method is more general than that of Milward. Milward's method cannot produce CCG derivations, since it can deal with only function application.

There are other approaches to incremental semantic construction, which use different formalism. Purver et al. [7] have developed a dialogue system based on Dynamic Syntax (DS) [25], which provides an incremental framework of constructing semantic representations. Peldszus and Schlangen [8] have proposed incremental semantic construction based on Robust Minimal Recursion Semantics (RMRS) [26]. Sayeed and Demberg [27] have proposed incremental semantic construction for PLTAG [21]. It is unclear how to construct a wide coverage grammar (with semantic annotation) in these frameworks[††]. On the other hand, our method can use CCG-based lexicon (e.g., [29]) directly. Although our method requires a set of allowable chains and auxiliary trees in addition to such a lexicon, we can easily extract it from CCGbank [30] by using the method proposed in the literature [16].

6. Conclusion

This paper proposed a CCG-based method of incrementally constructing semantic representations. Our formalization is based on normal form CCG derivation unlike existing alternatives. The main contribution of our paper is to formally define incremental semantic construction which is guaranteed to obtain semantic representations for any initial fragment of an input sentence. The previous CCG-based methods did not have this property.

Although in this paper the proposed method uses normal form CCG derivation, it can be applied to any kind of CCG derivation including incremental derivation. We will further study what kind of derivation is appropriate for our method. Furthermore, we will implement an incremental semantic parser based on our method and evaluate its performance. To evaluate the usefulness of the intermediate semantic representations directly, we will investigate a task-oriented evaluation in a similar manner as the literature [31], where two incremental semantic construction methods [7], [8] were compared in an incremental reference resolution task [32].

Another important issue is how to interpret intermediate semantic representations for initial fragments. To our knowledge, there is little work on this issue. In future work, we will explore a model-theoretic approach to this problem.

Acknowledgements

This research was partially supported by the Grant-in-Aid for Scientific Research (B) (No.26280082) of JSPS.

[†]The initial fragment "Anna met" can have the semantic representation λx.meet′$x$anna′ as shown in Fig. 3 (a). However, the derivation which has this semantic representation is not a partial structure of incremental derivation shown in Fig. 3 (b). That is, the derivation is not consistent with that of "Anna met and might marry Manny."

[††]DS grammar induction method [28] was only applied to a small artificial corpus (200 sentences, max sentence length is 6.). Peldszus and Schlangen [8] manually assigned semantic annotations to a small set of context-free rules (30 rules). Sayeed and Demberg [27] only provided small examples.

References

[1] L. Huang and K. Sagae, "Dynamic programming for linear-time incremental parsing," Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, pp.1077–1086, July 2010.

[2] L. Shen and A. Joshi, "Incremental LTAG parsing," Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada, pp.811–818, Oct. 2005.
[3] V. Lombardo and P. Sturt, "Incremental processing and infinite local ambiguity," Proceedings of the 19th Annual Conference of the Cognitive Science Society, pp.448–453, 1997.
[4] V. Demberg, "Incremental derivations in CCG," Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 11), pp.198–206, 2012.
[5] J. Allen, G. Ferguson, and A. Stent, "An architecture for more realistic conversational systems," Proceedings of International Conference of Intelligent User Interfaces, Santa Fe, New Mexico, USA, pp.1–8, Jan. 2001.
[6] G. Aist, J. Allen, E. Campana, C.G. Gallo, S. Stoness, M. Swift, and M.K. Tanenhaus, "Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods," Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue, ed. R. Artstein and L. View, Trento, Italy, pp.149–154, June 2007.
[7] M. Purver, A. Eshghi, and J. Hough, "Incremental semantic construction in a dialogue system," Proceedings of the Ninth International Conference on Computational Semantics, pp.365–369, Association for Computational Linguistics, 2011.
[8] A. Peldszus and D. Schlangen, "Incremental construction of robust but deep semantic representations for use in responsive dialogue systems," Proceedings of the Workshop on Advances in Discourse Analysis and its Computational Aspects, pp.56–76, 2012.
[9] M. Steedman, The Syntactic Process, The MIT Press, 2000.
[10] J. Bos, "Wide-coverage semantic analysis with Boxer," Proceedings of the 2008 Conference on Semantics in Text Processing - STEP '08, pp.277–286, 2008.
[11] D. Reitter, J. Hockenmaier, and F. Keller, "Priming effects in combinatory categorial grammar," Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp.308–316, 2006.
[12] H. Hassan, K. Sima'an, and A. Way, "A syntactic language model based on incremental CCG parsing," Spoken Language Technology Workshop, 2008. SLT 2008. IEEE, pp.205–208, 2008.
[13] A. Hefny, H. Hassan, and M. Bahgat, "Incremental combinatory categorial grammar and its derivations," Computational Linguistics and Intelligent Text Processing, pp.96–108, Springer, 2011.
[14] J. Eisner, "Efficient normal-form parsing for combinatory categorial grammar," Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz, California, USA, pp.79–86, June 1996.
[15] J. Hockenmaier and Y. Bisk, "Normal-form parsing for combinatory categorial grammars with generalized composition and type-raising," Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Beijing, China, pp.465–473, Aug. 2010.
[16] Y. Kato and S. Matsubara, "Incremental parsing with adjoining operation," IEICE Transactions on Information and Systems, vol.E92-D, no.12, pp.2306–2312, 2009.
[17] M. Collins and B. Roark, "Incremental parsing with the perceptron algorithm," Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, Barcelona, Spain, pp.111–118, July 2004.
[18] A.K. Joshi, "Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description?," in Natural Language Parsing, ed. D.R. Dowty, L. Karttunen, and A.M. Zwicky, pp.206–250, Cambridge University Press, 1985.
[19] P. Sturt and V. Lombardo, "Processing coordinated structures: Incrementality and connectedness," Cognitive Science, vol.2, no.29, pp.291–305, 2005.
[20] A. Mazzei, V. Lombardo, and P. Sturt, "Dynamic TAG and lexical dependencies," Research on Language and Computation, vol.5, no.3, pp.309–332, Sept. 2007.
[21] V. Demberg, F. Keller, and A. Koller, "Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar," Computational Linguistics, vol.39, no.4, pp.1025–1066, 2013.
[22] M. Ballesteros and X. Carreras, "Transition-based spinal parsing," Proceedings of the Nineteenth Conference on Computational Natural Language Learning, Beijing, China, pp.289–299, July 2015.
[23] S.G. Pulman, "A parser that doesn't," Proceedings of the Second Conference on European Chapter of the Association for Computational Linguistics, pp.128–135, 1985.
[24] D. Milward, "Incremental interpretation of categorial grammar," Proceedings of the Seventh Conference on European Chapter of the Association for Computational Linguistics, pp.119–126, 1995.
[25] R. Kempson, W.M. Viol, and D. Gabbay, Dynamic Syntax: the Flow of Language Understanding, Blackwell, 2001.
[26] A. Copestake, "Semantic composition with (robust) minimal recursion semantics," Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing, Prague, Czech Republic, pp.73–80, June 2007.
[27] A. Sayeed and V. Demberg, "Incremental Neo-Davidsonian semantic construction for TAG," Proceedings of 11th International Workshop on Tree-Adjoining Grammars and Related Formalisms, pp.64–72, 2012.
[28] A. Eshghi, M. Purver, and J. Hough, "Probabilistic induction for an incremental semantic grammar," Proceedings of the 10th International Conference on Computational Semantics, pp.107–118, 2013.
[29] J. Bos, "Towards a large-scale formal semantic lexicon for text processing," Proceedings of the Biennal GSCL Conference From Form to Meaning: Processing Texts Automatically, pp.3–14, 2009.
[30] J. Hockenmaier and M. Steedman, "CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank," Computational Linguistics, vol.33, no.3, pp.355–396, 2007.
[31] J. Hough, C. Kennington, D. Schlangen, and J. Ginzburg, "Incremental semantics for dialogue processing: Requirements, and a comparison of two approaches," Proceedings of the 11th International Conference on Computational Semantics, London, UK, pp.206–216, April 2015.
[32] D. Schlangen, T. Baumann, and M. Atterer, "Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies," Proceedings of the SIGDIAL 2009 Conference, London, UK, pp.30–37, Sept. 2009.

Yoshihide Kato received the B.E. degree, the M.E. degree, and the Dr. of Engineering degree in information engineering from Nagoya University, in 1997, 1999 and 2003, respectively. He was an Assistant Professor in Graduate School of International Development, Nagoya University, from 2003 to 2009. He was a Researcher at Information Technology Center, Nagoya University, from 2009 to September 2011, and a Designated Assistant Professor, from October 2011 to March 2012. He is currently an Associate Professor at Information & Communications, Nagoya University. His research interests include natural language processing and information retrieval. He is a member of the IPSJ, the NLP and the ACL.

Shigeki Matsubara received the B.E. degree in electrical and computer engineering from Nagoya Institute of Technology in 1993, and the M.E. degree, and the Dr. of Engineering from Nagoya University in 1995 and 1998, respectively. After becoming an Assistant Professor at Nagoya University, he became an Associate Professor at Information Technology Center in 2002, and he is currently an Associate Professor in Graduate School of Information Science. His research interests include natural language processing, information retrieval and digital library. He is a member of the IPSJ, the JSAI and the NLP.