A Formal Proof of Strong Equivalence for a Grammar Conversion from LTAG to HPSG-style

Naoki Yoshinaga,† Yusuke Miyao,† and Jun'ichi Tsujii†‡
† *University of Tokyo* ‡ *CREST, JST (Japan Science and Technology Corporation)*

1. Introduction

This paper presents a sketch of a formal proof of strong equivalence,[1] where both grammars generate equivalent parse results, between any LTAG (Lexicalized Tree Adjoining Grammar: Schabes, Abeille and Joshi (1988)) G and an HPSG (Head-Driven Phrase Structure Grammar: Pollard and Sag (1994))-style grammar converted from G by a grammar conversion (Yoshinaga and Miyao, 2001). Our proof theoretically justifies some applications of the grammar conversion that exploit the nature of strong equivalence (Yoshinaga *et al.*, 2001b; Yoshinaga *et al.*, 2001a), applications which contribute much to the developments of the two formalisms.

In the past decades, LTAG and HPSG have received considerable attention as approaches to the formalization of natural languages in the field of computational linguistics. Discussion of the correspondences between the two formalisms has accompanied their development; that is, their linguistic relationships and differences have been investigated (Abeillé, 1993; Kasper, 1998), as has conversion between two grammars in the two formalisms (Kasper *et al.*, 1995; Tateisi *et al.*, 1998; Becker and Lopez, 2000). These ongoing efforts have contributed greatly to the development of the two formalisms.

Following this direction, in our earlier work (Yoshinaga and Miyao, 2001), we provided a method for converting grammars from LTAG to *HPSG-style*, which is the notion that we defined according to the computational device that underlies HPSG. We used the grammar conversion to obtain an HPSG-style grammar from LTAG (The XTAG Research Group, 2001), and then empirically showed strong equivalence between the LTAG and the obtained HPSG-style grammar for the sentences in the ATIS corpus (Marcus, Santorini and Marcinkiewicz, 1994). We exploited the nature of strong equivalence between the LTAG and the HPSG-style grammars to provide some applications such as sharing of existing resources between the two grammar formalisms (Yoshinaga *et al.*, 2001b), a comparison of performance between parsers based on the two different formalisms (Yoshinaga *et al.*, 2001a), and linguistic correspondence between the HPSG-style grammar and HPSG. As the most important result for the LTAG community, through the experiments of parsing within the above sentences, we showed that the empirical time complexity of an LTAG parser (Sarkar, 2000) is higher than that of an HPSG parser (Torisawa *et al.*, 2000). This result is contrary to the general expectations from the viewpoint of the theoretical bound of worst time complexity, which is worth exploring further. However, the lack of the formal proof of strong equivalence restricts scope of the applications of our grammar conversion to grammars which are empirically attested the strong equivalence, and this prevents the applications from maximizing their true potential. In this paper we give a formal proof of strong equivalence between any LTAG G and an HPSG-style grammar converted from G by our grammar conversion in order to remove such restrictions on the applications.

2. Grammar conversion

We start by stating our definition of an HPSG-style grammar, and then briefly describe our algorithm for converting grammars from LTAG to HPSG-style. We hope that the reader will refer to the cited literature (Yoshinaga and Miyao, 2001) for a more detailed description.

We defined *an HPSG-style grammar*, the form of the output of our conversion, according to the computational architecture which underlies HPSG (Pollard and Sag, 1994). An HPSG-style grammar consists of *lexical entries* and *ID grammar rules*, each of which is described with typed feature structures (Carpenter, 1992). A lexical entry for a word must express the characteristics of the word, such as its subcategorization frame and grammatical category. An ID grammar rule must represent the constraints on the configuration of immediate constituency, and

* This research was funded by JSPS Research Fellowships for Young Scientists.

1. Chomsky (1963) first introduced the notion of strong equivalence between grammars, where both grammars generate the same set of structural descriptions (e.g., parse trees). Kornai and Pullum (1990) and Miller (1999) used the notion of isomorphism between sets of structural descriptions to provide the notion of strong equivalence across grammar formalisms, which we have adopted in our research.

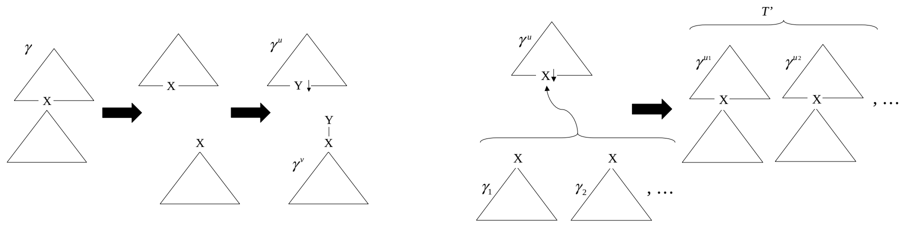

Figure 1: Sketch for the division transformation (left) and the substitution transformation (right)

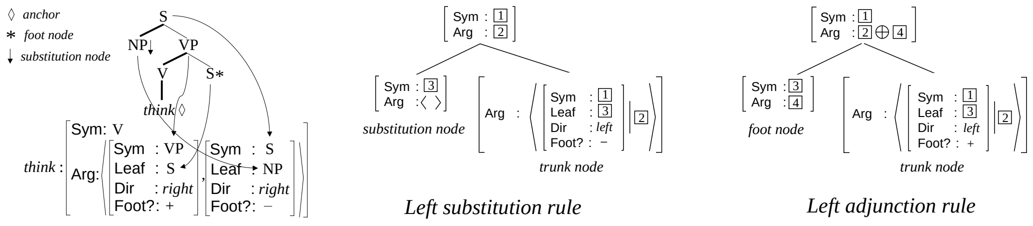

Figure 2: A conversion from a canonical elementary tree to an HPSG lexical entry (left) and grammar rules: the substitution rule (center) and adjunction rule (right)

not be a construction-specific rule specified by lexical characteristics. The formal definition of an HPSG-style grammar converted from LTAG G is given later in Section 3.3.

Our conversion algorithm consists of two kinds of conversion; i) a conversion from LTAG into *canonical LTAG*, LTAG which consists only of *canonical elementary trees*, and ii) a conversion from the canonical LTAG into an HPSG-style grammar. Canonical elementary trees are tree structures satisfy the following conditions; Condition 1: A tree must have only one anchor, and Condition 2: Every branching structure in a tree must contain trunk nodes. *Trunk nodes* are nodes on *a trunk* which is a path from an anchor to the root node. We call a subtree of depth $n(\geq 1)$ that includes no anchor *a non-anchored subtree*. Elementary trees which violate Condition 1 are converted into canonical ones by dividing them into single-anchored parts (*the division transformation*: the left-hand side of Figure 1). Elementary trees which violate Condition 2 are initially divided into multiple subtrees by the division transformation, each of which has at most one anchor, and then converted into canonical ones by substituting the deepest nodes in the non-anchored subtrees with every initial tree (*the substitution transformation*: the right-hand side of Figure 1). We give the formal definition of these transformations later in Section 3.2. Conversion of a canonical elementary tree is straightforward; that is, we traverse the trunk of a canonical elementary tree from its anchor to root, regard the leaf nodes as the anchor's arguments, and store the symbols of the leaf nodes and the trunk nodes as Leaf and Sym features respectively in a stack (Arg feature in the left-hand side of Figure 2), where Dir and Foot? features are the direction of the leaf node relative to the trunk and the type of the leaf node, respectively. A set of pre-determined rules manipulates the stack to emulate substitution and adjunction; namely, substitution rules (the center of Figure 2) and adjunction rules (the right-hand side of Figure 2).

3. A formal proof of strong equivalence

The whole proof consists of two pieces, each of which respectively proves that strong equivalence is guaranteed before and after the two conversions mentioned in the previous section.

3.1. Definitions

We first define LTAG, according to the definition of TAG given by (Vijay-Shanker, 1987). We then define *a derivation tree*, which is a structural description of LTAG, and introduce the notion of strong equivalence.

We hereafter denote a tree as a set of pairs $\langle p, X\rangle$ where $p \in \mathcal{N}^*$, which is a free monoid of the set of natural numbers, and $X \in V$, which is a finite set of alphabets (Gorn, 1962). For example, a tree in the left-hand side of Figure 2 is denoted as $\{(\epsilon, S), (\epsilon \cdot 1, NP)(\epsilon \cdot 2, VP), (\epsilon \cdot 2 \cdot 1, V), (\epsilon \cdot 2 \cdot 2, S), (\epsilon \cdot 2 \cdot 1 \cdot 1, think)\}$. An inequality $p \leq q$ is satisfied if and only if there is a $r \in \mathcal{N}^*$ such that $q = p \cdot r$. Another inequality $p < q$ is satisfied if and only if $p \leq q$ and $p \neq q$.

Definition 3.1 (Lexicalized Tree Adjoining Grammar (LTAG)) *Lexicalized Tree Adjoining Grammar G*[2] *is a quintuplet (Σ, NT, S, I, A) where Σ and NT are a finite set of terminal symbols and a finite set of nonterminal symbols respectively, S is a distinguished nonterminal symbol called the start symbol, and I and A are a finite set of initial trees and a finite set of auxiliary trees respectively.*

Here, an elementary tree $\gamma \in A \cup I$ is a tree whose leaf nodes are labeled by $X \in NT \cup S$ or $x \in \Sigma$, and whose internal nodes are labeled by $X \in NT \cup S$. The symbol of one leaf node in an auxiliary tree $\beta \in A$ is identical to that of its root node, and is specially marked for a foot node. Note that more than one leaf nodes called anchors in an elementary tree γ are labeled with $x \in \Sigma$, and leaf nodes other than anchors and foot nodes are marked for substitution nodes.

We denote adjunction and substitution of several trees $\gamma_1, \ldots, \gamma_k$ into a tree γ at k distinct addresses $a_1, \ldots, a_k$ by $\gamma' \rightarrow \gamma[a_1, \gamma'_1] \ldots [a_k, \gamma'_k]$ where $k \geq 1$, and $[a_i, \gamma'_i]$ indicates substitution at a_i of γ'_i in the case where a_i is a substitution node, or indicates adjunction at a_i of γ'_i in the case where a_i is an internal node. We call this production as *a derivation* for γ if all of the addresses of the substitution nodes in γ are included in $a_1, \ldots, a_k$. A derivation for γ without substitution and adjunction is denoted as $\gamma' \rightarrow \epsilon$.

We use the above notations to define *a derivation tree*, which represents the history of combinations of trees and is a structural description of LTAG.

Definition 3.2 (Derivation trees) *A derivation tree Υ_G for LTAG $G = (\Sigma, NT, S, I, A)$ is defined by a set of derivations as follows:*

$$\Upsilon_G = \{\gamma'_i \rightarrow \epsilon \mid 1 \leq i \leq m, \gamma_i \in A \cup I\} \bigcup D_G$$

where $D_G \subset \{\gamma'_i \rightarrow \gamma_i[a_1, \gamma'_{i_1}] \ldots [a_k, \gamma'_{i_k}] \mid k \geq 1, i > m, \gamma_i, \gamma_{i_j} \in A \cup I\}$. The derivation tree Υ_G must satisfy the following condition: γ'_i can appear once respectively in the left-hand side and the right-hand side of derivations except that one distinguished elementary tree γ_S, which is the root of the derivation tree Υ_G and γ'_S can appear once in the left-hand side of the derivation, because γ_i can adjoin or substitute once.[3] *Note that the inequality $i > i_j \geq 1$ is necessary to avoid cyclic applications of substitution and adjunction among elementary trees.*

Finally, we give the definition of strong equivalence between two given grammars G_1 and G_2.

Definition 3.3 (Strong equivalence) *Two given grammars G_1 and G_2 are strongly equivalent if and only if there is a bijective (i.e., one-to-one and onto) function which maps a set of structural descriptions of G_1, $T_D(G_1)$, to a set of structural descriptions of G_2, $T_D(G_2)$.*

In what follows, we assume that structural descriptions of LTAG are derivation trees in which the root node of γ_S is labeled by the start symbol S in the definition 3.2.

3.2. Proof of strong equivalence for the two tree transformations

In this section we give a proof that strong equivalence is guaranteed for grammars before and after the two tree transformations. In this abstract, We omit the proof of the substitution procedure, because the substitution transformation is exactly the same as the one that Schabes and Waters (1995, pp. 494–495) defined and proved in their strong lexicalization procedure of CFG into Lexicalized Tree Insertion Grammar.

The division transformation is formalized in the following lemma.

Lemma 3.1 (The division transformation) *Let $G = (\Sigma, NT, S, I, A)$ be LTAG. Let $\gamma \in A \cup I$ be an elementary tree and let μ be an internal node with address p of γ that is labeled by X and be not on the spine. We divide γ at μ and obtain two trees γ^u, γ^v as follows. Let γ^u be a subtree except that a node labeled by $Y \notin NT \cup S$ is added to its root node, and let γ^v be a supertree, except that the symbol of μ is relabeled by the symbol $Y \notin NT$ and by marking it for substitution as shown in Figure 1. Define $G' = (\Sigma, NT \cup \{Y\}, S, I', A')$ where I' and A' are created as follows:*

If $\gamma \in I$ then $I' = (I - \{\gamma\}) \cup \{\gamma^u, \gamma^v\}$ and $A' = A$
If $\gamma \in A$ then $I' = I \cup \{\gamma^v\}$ and $A' = (A - \{\gamma\}) \cup \{\gamma^u\}$

Then, G' is strongly equivalent to G; that is, there is a one-to-one onto mapping from the set of derivation trees $T_D(G')$ generated by G' to the set of derivation trees $T_D(G)$ generated by G for the same sentence.

2. Due to limitations of space, we omit the notion of adjoining constraints and the proof including the notion in this abstract, and then assume all internal nodes take selective adjoining constraints.
3. The condition implies that no trees can substitute or adjoin to two different nodes.

Proof We show that there is a one-to-one mapping from a derivation tree $\Upsilon_{G'} \in T_D(G')$ to a derivation tree $\Upsilon_G \in T_D(G)$.

Assume each derivation tree $\Upsilon_{G'}$ consists of elementary trees $\{\gamma_1, \ldots, \gamma_n\}$, $\gamma_j \in A \cup I$ for $1 \leq j \leq n$. Then, we can represent the derivation tree $\Upsilon_{G'}$ by the set of derivations as shown in the definition 3.2.

Since we assume that a derivation tree is rooted by an elementary tree whose symbol of the root node is S, every occurrence of γ^v in $\Upsilon_{G'}$ must always accompany with γ^u and vice versa. In the following procedure, we construct a one-to-one mapping from $\Upsilon_{G'}$ to Υ_G by replacing every occurrence of γ^u which takes a substitution of γ^v with γ in derivations in $\Upsilon_{G'}$.

1. When $\gamma^u \notin \{\gamma_1, \ldots, \gamma_n\}$ or $\gamma^v \notin \{\gamma_1, \ldots, \gamma_n\}$, $\Upsilon_{G'}$ includes neither γ^u nor γ^v. $\Upsilon_{G'}$ therefore consists of $\gamma_i \in (A \cup I - \{\gamma\}) \subset A \cup I$, there is exactly the same derivation tree Υ_G in $T_D(G)$.
2. When $\gamma^u \in \{\gamma_1, \ldots, \gamma_n\}$, we can construct one derivation tree Υ_G from Υ'_G as follows.
   (a) We first replace every occurrence of γ'^u in the right-hand side of derivations with γ'.
   (b) We next replace every derivation whose left-hand side is either γ'^u or γ'^v.
      i. When a root node with address ϵ of γ^v takes substitution or adjunction, a pair of two derivations whose left-hand side is γ'^u and γ'^v is denoted as $\gamma'^u \rightarrow \gamma^u[a_1, \gamma'_1] \ldots [a_{h-1}, \gamma'_{h-1}][p, \gamma'^v]$ and $\gamma'^v \rightarrow \gamma^v[\epsilon, \gamma'_h][b_{h+1}, \gamma'_{h+1}] \ldots [b_k, \gamma'_k]$, where $k > h \geq 1$. Here we assume $a_i \not\geq p$ for $1 \leq i < h$ without loss of generality. We replace these two derivations with the following derivation:

$$\gamma' \rightarrow \gamma[a_1, \gamma'_1] \ldots [a_{h-1}, \gamma'_{h-1}][p, \gamma'_h][p \cdot 1 \cdot b_{h+1}, \gamma'_{h+1}] \ldots [p \cdot 1 \cdot b_k, \gamma'_k]$$

      ii. If a root node with address ϵ in γ^v takes neither adjunction nor substitution, we can also replace a pair of two derivations whose left-hand side are respectively γ'^u and γ'^v with one derivation whose left-hand side is γ' in a similar way as above.
   (c) By repeating the above replacements at most the number of pairs of two derivations for γ^u and γ^v, we can obtain a set of derivations Υ_G without γ'^u and γ'^v. The replacement in (a) is valid since γ^u includes both root node and foot node of γ, and thus γ can substitute or adjoin every node at which γ^u does. In the procedure (b), we replace exactly the same number of γ'^u as the procedure (a). The resulting derivations including γ' is valid in G because γ' appear only once in the right-hand side and the left-hand side of the derivations, respectively.

The resulting derivation tree Υ_G is the same as $\Upsilon_{G'}$ except that every occurrences of γ^u which takes a substitution of γ^v with γ. Since γ^u which takes a substitution of γ^v is the same as γ except that one internal node is added, this does not cause effect on the frontier string. Also, when $\Upsilon^1_{G'}, \Upsilon^2_{G'}$ are mapped to $\Upsilon^1_G, \Upsilon^2_G$ and Υ^1_G and Υ^2_G are equivalent, $\Upsilon^1_{G'}$ and $\Upsilon^2_{G'}$ are also equivalent owing to the formulation of the above mapping.

On the other side, we can also construct a one-to-one onto mapping from Υ_G to $\Upsilon_{G'}$ by replacing every occurrence of γ in Υ_G by γ^u which takes a substitution of γ^v. Due to limitations of space, we omit the proof here.

In this way, we can construct a one-to-one onto mapping from a derivation tree $\Upsilon_{G'} \in T_D(G')$ to a derivation tree $\Upsilon_G \in T_D(G)$ for the same sentence. This indicates that G is strongly equivalent to G'. □

3.3. Proof of strong equivalence for the conversion from canonical LTAG to HPSG-style

In this section, we prove that strong equivalence is guaranteed for the latter part of our grammar conversion, that is, a conversion from canonical LTAG G to an HPSG-style grammar G'. In the following proof, we first introduce the notion of *origination* for every Sym and Leaf feature in HPSG lexical entries. We next define *an HPSG parse,* which is a structural description of an HPSG-style grammar. We then prove the strong equivalence by giving a one-to-one onto mapping from a derivation tree by G to an HPSG parse by G'.

Definition 3.4 (An HPSG-style grammar converted from LTAG) *Given canonical LTAG $G = (\Sigma, NT, S, I, A)$, an HPSG-style grammar G' converted from G is denoted by quituplet $(\Sigma, NT, S, \Delta, R)$ where $\delta_i \in \Delta$ is a lexical entry converted from $\gamma_i \in A \cup I$ and R is substitution and adjunction rules. δ_i is denoted as follows: $\delta_i = (s_0, (s_1, l_1, d_1, t_1), \ldots, (s_k, l_k, d_k, t_k))$ where $k \geq 1$, s_0 is the symbol of the mother node of the anchor in γ_i, and $s_j \in \Sigma \cup NT, l_j \in \Sigma \cup NT, d_j \in \{right, left\}, t_j \in \{+, -\}$ are values of Sym, Leaf, Dir, Foot? features in the j-th element of the Arg feature in δ_i. When the length of the Arg feature of δ_i is 0, δ_i is denoted as $\delta_i = (s_0, \phi)$.*

First, we introduce the notion of *origination* for the Sym and Leaf features in HPSG lexical entries in order to define *an HPSG parse,* which represents the histories of rule applications to lexical entries and is a structural description of an HPSG-style grammar. We hereafter assume that each HPSG lexical entry δ_i is converted from a canonical elementary tree γ_i. We define the origination of the feature in δ_i as $\langle p, \gamma_i \rangle$, which indicates that the value of the feature originates from the symbol of a node with address p in γ_i.

Next, we define *a rule history* for δ_i, which is a history of rule applications to a lexical entry δ_i in the parse tree. We then follow the parse tree from an anchor of δ_i to root, and then assign each rule application as an element of the rule history for δ_i if and only if the applied rule pops an element which originates from an element of the Arg feature in δ_i. Assume that δ_i is denoted as the one given in the definition 3.4. A rule history for δ_i is denoted as follows, where the origination of l_j and the feature unified with l_j are $\langle a_j, \gamma_i \rangle$ and $\langle b, \gamma_{i_j} \rangle$, respectively.

1. When $\gamma_i \in I$, no application of the adjunction rule is assigned to δ_i as an element of the rule history for δ_i. The rule history is then denoted as $\delta'_i \rightarrow \delta_i[a_1, \delta'_{i_1}] \ldots [a_k, \delta'_{i_k}]$.

2. When $\gamma_i \in A$, one application of the adjunction rule is assigned to δ_i as an element of the rule history for δ_i. The rule history for δ_i is then denoted as $\delta'_i \rightarrow \delta_i[a_1, \delta'_{i_1}] \ldots [a_{h-1}, \delta'_{i_{h-1}}][b, \delta_{i_h}][a_{h+1}, \delta'_{i_{h+1}}] \ldots [a_k, \delta'_{i_k}]$ where $t_h = +$.

When the length of the Arg feature of δ_i is 0, a rule history for δ_i is denoted by $\delta'_i \rightarrow \epsilon$.

Definition 3.5 (HPSG parses) *Given canonical LTAG $G = (\Sigma, NT, S, I, A)$ and an HPSG-style grammar $G' = (\Sigma, NT, S, \Delta, R)$ converted from G, an HPSG parse $\Psi_{G'}$ is denoted by a set of rule histories for $\delta_i \in \Delta$ as follows:*

$$\Psi_{G'} = \{\delta'_i \rightarrow \epsilon \mid 1 \leq i \leq m, \gamma_i \in I\} \bigcup A_{G'} \bigcup B_{G'}$$

where $A_{G'}$ is a set of rule histories for δ_i converted from $\gamma_i \in I$, and $B_{G'}$ is a set of rule histories for δ_i converted from $\gamma_i \in A$, and elements in $A_{G'}$ and $B_{G'}$ are denoted as the ones in the above paragraph where $i > m$.

Since the above HPSG parse Ψ_G must uniquely correspond to the parse tree, we require some conditions on Ψ_G. First, δ'_i where $\gamma_i \in I$ can appear once respectively in the left-hand side and the right-hand side of rule histories except that one distinguished lexical entry δ_S where δ'_S appears once in the left-hand side of the rule history for δ_S. Second, δ'_i where $\gamma_i \in A$ must appear only once in the left-hand side of the rule history for δ_i. Third, $1 \leq i_j < i$ for the rule history for δ_i where $\gamma_i \in I$. Fourth, $1 \leq i_j < i$ where $j \neq h$, and $i_h > i$, for the rule history for δ_i where $\gamma_i \in A$. The third and fourth conditions are necessary to avoid cyclic applications of grammar rules to lexical entries.

Lemma 3.2 *Let $G = (\Sigma, NT, S, I, A)$ and G' be LTAG and an HPSG-style grammar converted from G, respectively. Then, we can map a derivation tree Υ_G by G one-to-one onto to an HPSG parse $\Psi_{G'}$ by G'.*

Proof In the following proof, we first show a mapping from $\Psi_{G'}$ to a set of derivations $\Upsilon_{G'}$, and then show that $\Upsilon_{G'}$ is a valid derivation by G.

Suppose an HPSG parse denoted as the one given in the definition 3.5. We can map it to a set of derivations $\Upsilon_{G'}$ in the following procedure. For each δ_i where $\gamma_i \in A$, we eliminate $[b, \delta_{i_h}]$, which corresponds to an application of the adjunction rule, and add the element $[b, \delta'_i]$ to the right-hand side of the rule history for δ_{i_h}. Then, we obtain a set of derivations $\Upsilon_{G'}$ by replacing δ_{i_j} and δ'_{i_j} with γ_{i_j} and γ'_{i_j} in the rule history for δ_i and by assigning it as the derivation for γ_i. This mapping is one-to-one because a pair operation of an elimination of $[b, \delta_{i_h}]$ and an addition of $[b, \delta'_i]$ is one-to-one mapping.

Following the definition 3.2, we show that $\Upsilon_{G'}$ is a valid derivation tree by G. First, every substitution and adjunction in the derivations in $\Upsilon_{G'}$ must be valid in G. Since the substitution and adjunction rules preserve the order of the elements in the Arg feature of δ_i, substitution rules always unify the symbol of the substitution node with the symbol of the root node of γ_{i_j}, which represents the same constraint as the one on which substitution imposes. We can give the similar argument for an adjunction rule. The substitution and adjunction in the derivations in $\Upsilon_{G'}$ are then valid in G. Second, all addresses in the substitution nodes of γ_i must be included in its derivation. This is apparently guaranteed by the definition of the rule history for δ_i. Third, γ'_i can appear only once respectively in the right-hand side and the left-hand side of the derivations. This is apparently guaranteed for γ'_i where $\gamma_i \in I$ by the definition 3.5, and is guaranteed for γ'_i where $\gamma_i \in A$ because δ'_i does not appear in the right-hand side of rule histories, $[b, \delta_{i_h}]$ appears only once in the rule history for δ_i, and the elimination of $[b, \delta_{i_h}]$ accompanies the addition of $[b, \gamma'_i]$ once to the right-hand side of the derivation for γ_{i_h}. Fourth, the elements in the right-hand side of the derivation for γ_i must be $[a_j, \gamma'_{i_j}]$ where $i_j < i$. This is apparently guaranteed for γ'_i where $\gamma_i \in I$ by the definition 3.5, and is guaranteed for γ'_i where $\gamma_i \in A$ because the addition of $[b, \gamma'_i]$ for the derivation for γ'_{i_h} satisfy $i_h > i$ due to the definition 3.5.

The frontier string is preserved before and after this mapping from $\Psi_{G'}$ to $\Upsilon_{G'}$, because δ_i stores the same LP constraints between δ_i and δ_j for $i \neq j$ as the constraints between γ_i and γ_j. Then, an HPSG parse $\Psi_{G'}$ by G' mapped one-to-one to a derivation tree $\Upsilon_{G'}$ which is valid in G.

On the other side, we can construct a mapping from Υ_G to an HPSG parse Ψ_G as the inverse procedure for the above mapping from $\Psi_{G'}$ to $\Upsilon_{G'}$. The obtained Ψ_G is a valid HPSG parse by G' because we can give a similar argument for the validity of the rule histories in Ψ_G. $\square$

Hence, strong equivalence is guaranteed for a conversion from canonical LTAG to an HPSG-style grammar. The two proofs given here and in the previous section prove the strong equivalence between any LTAG G and an HPSG-style grammar converted from G by our grammar conversion.

4. Conclusion

In this research, we proved that strong equivalence is guaranteed between any LTAG grammar G and an HPSG-style grammars converted from G by our grammar conversion. Our proof theoretically justifies some applications of the grammar conversion that exploit the nature of strong equivalence (Yoshinaga *et al.*, 2001b; Yoshinaga *et al.*, 2001a), applications which contribute much to the developments of the two formalisms.

References

Abeillé, Anne. 1993. *Les nouvelles syntaxes: grammaires d'unification et analyse du français*. Armanda Colin. in French.

Becker, Tilman and Patrice Lopez. 2000. Adapting HPSG-to-TAG compilation to wide-coverage grammars. In *Proc. of TAG+5*, pages 47–54.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press.

Chomsky, Noam. 1963. Formal properties of grammar. In R. D. Luce, R. R. Bush and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume II. John Wiley and Sons, Inc., pages 323–418.

Gorn, Saul. 1962. Processors for Infinite Codes of Shannon-Fano type. In *Proc. of the Symposium on Mathematical Theory of Automata*, pages 223–240.

Kasper, Robert. 1998. TAG and HPSG. Talk given in the tutorial session at TAG+4.

Kasper, Robert, Bernd Kiefer, Klaus Netter and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proc. of ACL 1995*, pages 92–99.

Kornai, A. and G. K. Pullum. 1990. The X-bar Theory of Phrase Structure. *Language*, 66:24–50.

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Miller, Philip H. 1999. *Strong Generative Capacity*. CSLI publications.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Sarkar, Anoop. 2000. Practical Experiments in Parsing using Tree Adjoining Grammars. In *Proc. of TAG+5*, pages 193–198.

Schabes, Yves, Anne Abeille and Aravind K. Joshi. 1988. Parsing strategies with 'Lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. of COLING 1992*, pages 578–583.

Schabes, Yves and Richard C. Waters. 1995. Tree Insertion Grammar: A Cubic-Time Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Tree Produced. *Computational Linguistics*, 21(4):479–513.

Tateisi, Yuka, Kentaro Torisawa, Yusuke Miyao and Jun'ichi Tsujii. 1998. Translating the XTAG English Grammar to HPSG. In *Proc. of TAG+4*, pages 172–175.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. http://www.cis.upenn.edu/~xtag/.

Torisawa, Kentaro, Kenji Nishida, Yusuke Miyao and Jun'ichi Tsujii. 2000. An HPSG Parser with CFG Filtering. *Natural Language Engineering*, 6(1):63–80.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

Yoshinaga, Naoki and Yusuke Miyao. 2001. Grammar conversion from LTAG to HPSG. In *Proc. of the Sixth ESSLLI Student Session*, pages 309–324.

Yoshinaga, Naoki, Yusuke Miyao, Kentaro Torisawa and Jun'ichi Tsujii. 2001a. Efficient LTAG parsing using HPSG parsers. In *Proc. of Pacific Association for Computational Linguistics (PACLING 2001)*, pages 342–351.

Yoshinaga, Naoki, Yusuke Miyao, Kentaro Torisawa and Jun'ichi Tsujii. 2001b. Resource sharing among HPSG and LTAG communities by a method of grammar conversion from FB-LTAG to HPSG. In *Proc. of ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education*, pages 39–46.