Lenient Default Unification for Robust Processing within Unification Based Grammar Formalisms

Takashi NINOMIYA,[†‡] **Yusuke MIYAO,**[†] and **Jun'ichi TSUJII**[†‡]

[†] Department of Computer Science, University of Tokyo

[‡] CREST, Japan Science and Technology Corporation

e-mail: {ninomi, yusuke, tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper describes new default unification, lenient default unification. It works efficiently, and gives more informative results because it maximizes the amount of information in the result, while other default unification maximizes it in the default. We also describe robust processing within the framework of HPSG. We extract grammar rules from the results of robust parsing using lenient default unification. The results of a series of experiments show that parsing with the extracted rules works robustly, and the coverage of a manually-developed HPSG grammar for Penn Treebank was greatly increased with a little overgeneration.

1 Introduction

Parsing has often been considered to be crucial for natural language processing, thus, efficient and wide coverage parsing has been extensively pursued in natural language literature. This study aims at robust processing within the Head-driven Phrase Structure Grammar (HPSG) to extend the coverage of manually-developed HPSG grammars. The meaning of 'robust processing' is not limited to robust processing for ill-formed sentences found in a spoken language, but includes robust processing for sentences which are well-formed but beyond the grammar writer's expectation.

Studies of robust parsing within unification-based grammars have been explored by many researchers (Douglas and Dale, 1992; Imaichi and Matsumoto, 1995). They classified the errors found in analyzing ill-formed sentences into several categories to make them tractable, e.g., constraint violation, missing or extra elements, etc. In this paper, we focus on recovery from the constraint violation errors, which is a violation of feature values. All errors in agreement fall into this category. Since many of the grammatical components in HPSG are written as constraints represented by feature structures, many of the errors are expected to be recovered by the recovery of constraint violation errors.

This paper proposes two new types of default unification and describes their application to robust processing. Default unification was originally studied to develop a system of lexical semantics to deal with the default inheritance in a lexicon, but it is also desirable for the recovery of such constraint violation errors due to the following merits: i) default unification is always well-defined, and ii) a feature structure is relaxed such that the amount of information is maximized. From the viewpoint of robust processing, an amount of lost information can be regarded as a cost (i.e., penalty) of robust processing. In other words, default unification tries to minimize the cost. Given a strict feature structure F and a default feature structure G, default unification is defined as unification that satisfies the following (written as $F \overset{<}{\sqcup} G$): 1) It is always defined. 2) All strict information is preserved. That is, $F \sqsubseteq (F \overset{<}{\sqcup} G)$. 3) It reduces to standard unification in the case of F and G being consistent. That is, $(F \overset{<}{\sqcup} G) = (F \sqcup G)$ if $F \sqcup G$ is defined. With these definitions, Douglas' relaxation technique can be regarded as a sort of default unification. They classify constraints into necessary constraints and optional constraints, which can be regarded as strict information and default information in the definition of default unification.

Carpenter (1993) gave concise and comprehensive definitions of default unification. However, the problem in Carpenter's default unification is that it tries to maximize the amount of information in a default feature structure, not the result of default unification. Consider the case where a grammar rule is the default feature structure and the daughters are the strict feature structure. The head feature principle can be described as the structure-sharing between the values of the head feature in a mother and in a head daughter. The set of constraints that represent the head feature principle consists of only one element. When we lose just one element in the head feature principle, a large amount of information in the daughter's substructure is not propagated to its mother. As Copestake (1993) mentioned, another problem in Carpenter's default unification is that the time complexity for finding the optimal answer of default unification is exponential because we have to verify the unifiability of the power set of constraints in a default feature structure.

Here, we propose *ideal lenient default unifica-*

tion, which tries to maximize the amount of information of a result, not the amount of default information. Thus, the problem of losing a large amount of information in structure-sharing never arises. We also propose *lenient default unification* whose algorithm is much more efficient than the ideal one. Its time complexity is linear to the size of the strict feature structure and the default feature structure. Instead, the amount of information of a result derived by lenient default unification is equal to or less than that of the ideal one.

We apply lenient default unification to robust processing. Given an HPSG grammar, our approach takes two steps; i) extraction of grammar rules from the results of robust parsing using lenient default unification for applying the HPSG grammar rules (offline parsing), and ii) runtime parsing using the HPSG grammar with the extracted rules. The extracted rules work robustly since they reflect the effects of recovery rules applied during offline robust parsing and the conditions in which they are applied.

Sections 3 and 4 describe our default unification. Our robust parsing is explained in Section 5. Section 6 shows a series of experiments of robust parsing with default unification.

2 Background

Default unification has been investigated by many researchers (Bouma, 1990; Russell et al., 1991; Copestake, 1993; Carpenter, 1993; Lascarides and Copestake, 1999) in the context of developing lexical semantics. Here, we first explain the definition given by Carpenter (1993) because his definition is both concise and comprehensive.

2.1 Carpenter's Default Unification

Carpenter proposed two types of default unification, *credulous default unification* and *skeptical default unification*.

(Credulous Default Unification)

$$F \overset{<}{\sqcup}_c G = \left\{ F \sqcup G' \;\middle|\; \begin{array}{l} G' \sqsubseteq G \text{ is maximal such that} \\ F \sqcup G' \text{ is defined} \end{array} \right\}$$

(Skeptical Default Unification)

$$F \overset{<}{\sqcup}_s G = \bigsqcap (F \overset{<}{\sqcup}_c G)$$

F is called a *strict feature structure*, whose information must not be lost, and G is called a *default feature structure*, whose information might be lost but as little as possible so that F and G can be unified. A credulous default unification operation is greedy in that it tries to maximize the amount of information it retains from the default feature structure. This definition returns a set of feature structures rather than a unique feature structure.

Skeptical default unification simply generalizes the set of feature structures which results from credulous default unification. The definition of skeptical default unification leads to a unique result. The default information which can be found in every result of credulous default unification remains. Following is an example of skeptical default unification.

$$[\text{F: } \mathbf{a}] \overset{<}{\sqcup}_s \begin{bmatrix} \text{F: } \boxed{1}\mathbf{b} \\ \text{G: } \boxed{1} \\ \text{H: } \mathbf{c} \end{bmatrix} = \bigsqcap \left\{ \begin{bmatrix} \text{F: } \mathbf{a} \\ \text{G: } \mathbf{b} \\ \text{H: } \mathbf{c} \end{bmatrix}, \begin{bmatrix} \text{F: } \boxed{1}\mathbf{a} \\ \text{G: } \boxed{1} \\ \text{H: } \mathbf{c} \end{bmatrix} \right\} = \begin{bmatrix} \text{F: } \mathbf{a} \\ \text{G: } \bot \\ \text{H: } \mathbf{c} \end{bmatrix}$$

2.2 Forced Unification

Forced unification is another way to unify inconsistent feature structures. Forced unification always succeeds by supposing the existence of the top type (the most specific type) in a type hierarchy. Unification of any pair of types is defined in the type hierarchy, and therefore unification of any pair of feature structures is defined. One example is described by Imaichi and Matsumoto (1995) (they call it cost-based unification). Their unification always succeeds by supposing the top type, and it also keeps the information about inconsistent types. Forced unification can be regarded as one of the toughest robust processing because it always succeeds and never loses the information embedded in feature structures. The drawback of forced unification is the postprocessing of parsing, i.e., feature structures with top types are not tractable. We write $F \sqcup_f G$ for the forced unification of F and G.

3 Ideal Lenient Default Unification

In this section, we explain our default unification, *ideal lenient default unification*. Ideal lenient default unification tries to maximize the amount of information of the result, subsuming the result of forced unification. In other words, ideal lenient default unification tries to generate a result as similar as possible to the result of forced unification such that the result is defined in the type hierarchy without the top type. Formally, we have:

Definition 3.1 *Ideal Lenient Default Unification*

$$F \overset{<}{\sqcup}_i G = \bigsqcap \left\{ F \sqcup G' \;\middle|\; \begin{array}{l} G' \sqsubseteq_f (F \sqcup_f G) \textit{ is maximal} \\ \textit{such that } F \sqcup G' \textit{ is defined} \\ \textit{without the top type} \end{array} \right\}$$

where $\sqsubseteq_f$ is a subsumption relation where the top type is defined.

From the definition of skeptical default unification, ideal lenient default unification is equivalent to $F \overset{<}{\sqcup}_s (F \sqcup_f G)$ assuming that skeptical default unification does not add the default information that includes the top type to the strict information.

Consider the following feature structures.

$$F = \begin{bmatrix} \text{F: } \begin{bmatrix} \text{F:}\mathbf{a} \\ \text{G:}\mathbf{b} \\ \text{H:}\mathbf{c} \end{bmatrix} \\ \text{G: } \begin{bmatrix} \text{F:}\mathbf{a} \\ \text{G:}\mathbf{a} \\ \text{H:}\mathbf{c} \end{bmatrix} \end{bmatrix}, G = \begin{bmatrix} \text{F:}\boxed{1} \\ \text{G:}\boxed{1} \end{bmatrix}$$

In the case of Carpenter's default unification, the results of skeptical and credulous default unification become as follows: $F \overset{<}{\sqcup}_s G = F, F \overset{<}{\sqcup}_c G = \{F\}$. This is because G is generalized to the bottom feature structure, and hence the result is equivalent to the strict feature structure.

With ideal lenient default unification, the result becomes as follows.

$$F \overset{<}{\sqcup}_i G = \begin{bmatrix} \text{F:} & \begin{bmatrix} \text{F:}\boxed{1}\mathbf{a} \\ \text{G:}\mathbf{b} \\ \text{H:}\boxed{2}\mathbf{c} \end{bmatrix} \\ \text{G:} & \begin{bmatrix} \text{F:}\boxed{1} \\ \text{G:}\mathbf{a} \\ \text{H:}\boxed{2} \end{bmatrix} \end{bmatrix} \sqsubseteq_f \begin{bmatrix} \text{F:}\boxed{1} & \begin{bmatrix} \text{F:}\mathbf{a} \\ \text{G:}\top \\ \text{H:}\mathbf{c} \end{bmatrix} \\ \text{G:}\boxed{1} & \end{bmatrix}$$

Note that the result of ideal lenient default unification subsumes the result of forced unification.

As we can see in the example, ideal lenient default unification tries to keep as much information of the structure-sharing as possible (ideal lenient default unification succeeds in preserving the structure-sharing tagged as $\boxed{1}$ and $\boxed{2}$ though skeptical and credulous default unification fail to capture it).

4 Lenient Default Unification

The optimal answer for ideal lenient default unification can be found by calculating $F \overset{<}{\sqcup}_s (F \sqcup_f G)$. As Copestake (1993) mentioned, the time complexity of skeptical default unification is exponential, and therefore the time complexity of ideal lenient default unification is also exponential.

As other researchers pursued efficient default unification (Bouma, 1990; Russell et al., 1991; Copestake, 1993), we also propose another definition of default unification, which we call *lenient default unification*. An algorithm derived for it finds its answer efficiently.

Given a strict feature structure F and a default feature structure G, let H be the result of forced unification, i.e., $H = F \sqcup_f G$. We define $topnode(H)$ as a function that returns the fail points (the nodes that are assigned the top type in H), $fpnode(H)$ as a function that returns the fail path nodes (the nodes from which a fail point can be reached), and $fpchild(H)$ as a a function that returns all the nodes that are not fail path nodes but the immediate children of fail path nodes.

Consider the following feature structures.

$$F = \begin{bmatrix} \text{F:F:} & \begin{bmatrix} \text{F:F:}\mathbf{a} \\ \text{G:G:}\mathbf{b} \\ \text{H:H:}\mathbf{c} \end{bmatrix} \\ \text{G:} & \begin{bmatrix} \text{G:} & \begin{bmatrix} \text{F:F:}\mathbf{a} \\ \text{G:G:}\mathbf{a} \\ \text{H:H:}\mathbf{c} \end{bmatrix} \\ \text{H:H:}\mathbf{a} & \end{bmatrix} \end{bmatrix}, G = \begin{bmatrix} \text{F:F:}\boxed{1}\bot \\ \text{G:G:}\boxed{1} \end{bmatrix}$$

Figure 1 shows F, G and H in the graph notation. This figure also shows the nodes that correspond to $topnode(H)$, $fpnode(H)$ and $fpchild(H)$.

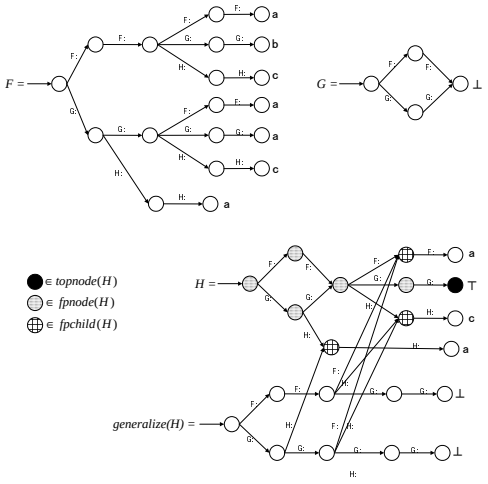

Figure 1: F, G and H in the graph notation

$F \sqcup H$ fails because some of the path value in H conflict with F, or some of the path equivalence in H cause inconsistencies. The basic ideas are that i) the inconsistency caused by path value specifications can be removed by generalizing the types assigned to the fail points in H, and that ii) the inconsistency caused by path equivalence specifications can be removed by unfolding the structure-sharing of fail path nodes in H.

Let H be $\langle Q_H, \bar{q}_H, \theta_H, \delta_H \rangle$, where Q is a set of a feature structure's nodes, $\bar{q}$ is the root node, $\theta(q)$ is a total node typing function, and $\delta(\pi, q)$ is a partial function that returns a node reached by following path π from q. We first give several definitions to define a generalized feature structure.

Definition 4.1

$$\begin{aligned}
&toppath(H) = \{\pi | \exists q \in topnode(H).(q = \delta_H(\pi, \bar{q}_H))\} \\
&failpath(H) = \{\pi | \exists q \in fpnode(H).(q = \delta_H(\pi, \bar{q}_H))\} \\
&ss(H) = \{\pi | \exists q \in fpchild(H).(q = \delta_H(\pi, \bar{q}_H))\} \\
&I(H) = \bigsqcup(\{F | \exists \pi \in toppath(H).(F = PV(\pi, \bot))\}) \\
&I'(H) = \bigsqcup\left(\left\{ F \;\middle|\; \begin{array}{l} \exists \pi \in failpath(H). \\ (F = PV(\pi, \theta_H(\delta_H(\pi, \bar{q}_H)))) \end{array} \right\}\right) \\
&I''(H) = I(H) \sqcup I'(H) \\
&PV(\pi, \sigma) = \left\{ \begin{array}{l} \text{the least feature structures where} \\ \text{path value of } \pi \text{ is } \sigma \end{array} \right.
\end{aligned}$$

Let $I(= \langle Q_I, \bar{q}_I, \theta_I, \delta_I \rangle)$ be $I''(H)$. The definition of the generalized feature structure is given as follows:

Definition 4.2 *(Generalization of H)*

$$\begin{aligned}
&generalize(H) = \langle Q_{H'}, \bar{q}_I, \theta_{H'}, \delta_{H'} \rangle \text{ where} \\
&Q_{H'} = Q_H \cup Q_I \\
&\theta_{H'}(q) = \left\{ \begin{array}{ll} \theta_H(q) & \text{if } q \in Q_H \\ \theta_I(q) & \text{if } q \in Q_I \end{array} \right. \\
&\delta_{H'}(f, q) = \left\{ \begin{array}{ll} \delta_H(f, q) & \text{if } q \in Q_H \\ \delta_I(f, q) & \text{if } q \in Q_I \end{array} \right. \\
&\delta_{H'}(\pi, \bar{q}_I) = \delta(\pi, \bar{q}_H) \text{ for all } \pi \in ss(H)
\end{aligned}$$

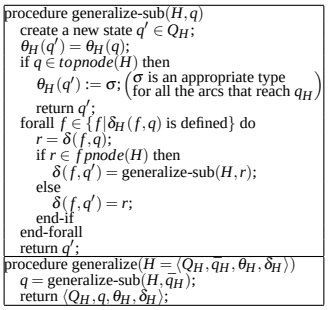

Figure 2: An algorithm that makes H more general

Figure 1 also shows the result of $generalize(F \sqcup_f G)$.

Finally, lenient default unification $F \overset{<}{\sqcup} G$ is defined as follows:

Definition 4.3 *(Lenient Default Unification)*

$$F \overset{<}{\sqcup} G = F \sqcup generalize(F \sqcup_f G)$$

For F and G depicted in Figure 1, $F \overset{<}{\sqcup} G$ becomes as follows:

$$F \overset{<}{\sqcup} G = \begin{bmatrix} \text{F:F:} & \begin{bmatrix} \text{F:}\boxed{1}\mathbf{a} \\ \text{G:G:}\mathbf{b} \\ \text{H:}\boxed{2}\mathbf{c} \end{bmatrix} \\ \text{G:} & \begin{bmatrix} \text{G:} & \begin{bmatrix} \text{F:}\boxed{1} \\ \text{G:G:}\mathbf{a} \\ \text{H:}\boxed{2} \end{bmatrix} \\ \text{H:H:}\mathbf{a} & \end{bmatrix} \end{bmatrix}$$

Both ideal and non-ideal lenient default unification satisfy the following desiderata: 1) It is always defined (and produces a unique result). 2) All strict information is preserved. That is, $F \sqsubseteq (F \overset{<}{\sqcup} G) \sqsubseteq_f (F \sqcup_f G)$. 3) $F \overset{<}{\sqcup} G$ is defined without the top type. 4) It reduces to standard unification in the case F and G are consistent (unifiable). That is, $F \overset{<}{\sqcup} G = F \sqcup G$ if $F \sqcup G$ is defined.

Algorithm

Our algorithm for lenient default unification proceeds in the following steps. 1) Calculate forced unification of F and G (let H be $F \sqcup_f G$). 2) Find *fail points* and *fail path nodes* in H. 3) Generalize H so that $F \sqcup H$ can be unified.

Figure 2 describes the algorithm that generalizes the result of forced unification.[1] The time complexity of the algorithm for finding $F \overset{<}{\sqcup} G$ is linear to the size of feature structures F and G because the time complexity of each algorithm (the algorithm for finding fail points, finding fail path nodes, and generalization) is linear to their size.

[1] In this paper, we assume acyclic feature structures. Our algorithm never stops if a cyclic part in a feature structure is to be generalized. Acyclicity is easily enforced by requiring that no path has a proper extension that leads to the same node. We also assume outputs of default unification are not necessarily totally-well typed since constraints of appropriateness conditions of types can propagate a node to its subnodes, and this behavior makes the definitions of default unification complex. Instead, totally-well typedness can be easily enforced by the total type inference function.

Comparison

The difference between ideal lenient default unification and lenient default unification can be exemplified by the following example.

(skeptical default unification)

$$\begin{bmatrix} \text{F: }\mathbf{a} \\ \text{G: }\mathbf{a} \\ \text{H: }\mathbf{b} \end{bmatrix} \overset{<}{\sqcup_s} \begin{bmatrix} \text{F: }\boxed{1} \\ \text{G: }\boxed{1} \\ \text{H: }\boxed{1} \end{bmatrix} = \begin{bmatrix} \text{F: }\boxed{2}\mathbf{a} \\ \text{G: }\boxed{2} \\ \text{H: }\mathbf{b} \end{bmatrix}$$

(ideal lenient default unification)

$$\begin{bmatrix} \text{F: }\mathbf{a} \\ \text{G: }\mathbf{a} \\ \text{H: }\mathbf{b} \end{bmatrix} \overset{<}{\sqcup_i} \begin{bmatrix} \text{F: }\boxed{1} \\ \text{G: }\boxed{1} \\ \text{H: }\boxed{1} \end{bmatrix} = \begin{bmatrix} \text{F: }\boxed{2}\mathbf{a} \\ \text{G: }\boxed{2} \\ \text{H: }\mathbf{b} \end{bmatrix}$$

(lenient default unification)

$$\begin{bmatrix} \text{F: }\mathbf{a} \\ \text{G: }\mathbf{a} \\ \text{H: }\mathbf{b} \end{bmatrix} \overset{<}{\sqcup} \begin{bmatrix} \text{F: }\boxed{1} \\ \text{G: }\boxed{1} \\ \text{H: }\boxed{1} \end{bmatrix} = \begin{bmatrix} \text{F: }\mathbf{a} \\ \text{G: }\mathbf{a} \\ \text{H: }\mathbf{b} \end{bmatrix}$$

In the example above, the results of ideal lenient default unification and skeptical default unification are the same. In the case of lenient default unification, all the information embedded in the default is removed because all structure-sharings tagged as $\boxed{1}$ are on the paths that lead to the fail points in the result of forced unification. Lenient default unification is much more suspicious in removing information from the default than the ideal one. Lenient default unification may remove structure-sharings that are irrelevant to unification failure. Another defect of lenient default unification is that the bottom type is assigned to nodes that correspond to the fail points in the result of forced unification. The type assigned to their nodes should be more specific than the bottom type as the bottom type has no feature, i.e., all the arcs that go out from the fail point are cut.

Though lenient default unification seems to have many defects, lenient default unification has the advantage of efficiency. As we are thinking to use default unification for practical robust processing, the efficiency is of great importance. Furthermore, the result of lenient default unification can be more informative than that of skeptical default unification in many cases of practical applications. For example, suppose that the grammar rule R and the daughters DTR are given as follows.

$$R = \begin{bmatrix} \text{MOTHER:HEAD: }\boxed{1} \\ \text{DTRS: } \begin{bmatrix} \text{H: }\boxed{2} & [\text{HEAD: }\boxed{1}\mathbf{head}] \\ \text{NH:SPR: }\boxed{2} & \end{bmatrix} \end{bmatrix}$$

$$DTR = \begin{bmatrix} \text{DTRS: } \begin{bmatrix} \text{H:HEAD:CASE: }\mathbf{obj} \\ \text{NH:SPR:HEAD:CASE: }\mathbf{nom} \end{bmatrix} \end{bmatrix}$$

Suppose also that the type **head** has PHON:, CASE:, INV: and TENSE: as its features, and the type **sign** has HEAD: and VAL:. The result of skeptical default unification $DTR \overset{<}{\sqcup_s} R$ becomes DTR. This is because

all structure-sharings embedded in R are relevant to unification failure. However, the result of lenient default unification is more informative.

$$DTR \overset{<}{\sqcup}_i R = DTR \overset{<}{\sqcup} R =$$

$$\begin{bmatrix} \text{MOTHER:HEAD:} & \begin{bmatrix} \textbf{head} \\ \text{PHON: } \boxed{3} \\ \text{CASE: } \bot \\ \text{INV: } \boxed{4} \\ \text{TENSE: } \boxed{5} \end{bmatrix} \\ \text{DTRS:} & \begin{bmatrix} \text{H:} & \begin{bmatrix} \textbf{sign} \\ \text{HEAD:} & \begin{bmatrix} \textbf{head} \\ \text{PHON: } \boxed{3} \\ \text{CASE: } \textbf{obj} \\ \text{INV: } \boxed{4} \\ \text{TENSE: } \boxed{5} \end{bmatrix} \\ \text{VAL: } \boxed{6} \end{bmatrix} \\ \text{NH:SPR:} & \begin{bmatrix} \textbf{sign} \\ \text{HEAD:} & \begin{bmatrix} \textbf{head} \\ \text{PHON: } \boxed{3} \\ \text{CASE: } \textbf{nom} \\ \text{INV: } \boxed{4} \\ \text{TENSE: } \boxed{5} \end{bmatrix} \\ \text{VAL: } \boxed{6} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The information of structure-sharing is preserved as much as possible. In the example above, the structure-sharing tagged as $\boxed{2}$ in the original grammar rule R is decomposed into the structure-sharings $\boxed{3}, \boxed{4}, \boxed{5}, \boxed{6}$. That is, the structure-sharing tagged as $\boxed{2}$ is preserved except HEAD:CASE:.

5 Offline Robust Parsing and Grammar Extraction

This section describes a new approach to robust parsing using default unification. Given an HPSG grammar, our approach takes two steps; i) extraction of grammar rules from the result of *offline robust parsing* using default unification for applying the HPSG grammar rules, and ii) *runtime parsing* using the HPSG grammar with the extracted rules. Offline parsing is a training phase to extract grammar rules, and runtime parsing is a phase where we apply the extracted rules to practice. The extracted rules work robustly over corpora other than the training corpus because the extracted rules reflect the effects of default unification that are applied during offline parsing. Given an annotated corpus, our algorithm extracts grammar rules that make the coverage of the HPSG grammar wider.

In the offline parsing, constituents are generated by default unification of daughters and grammar rules of the HPSG grammar[2], where a head daughter and a grammar rule are strict feature structures and a non-head daughter is a default feature structure. With this construction, the information in a grammar rule and a head daughter is strictly preserved and the information in a non-head daughter is partially lost (but, as little as possible). The ideas behind this construction are that (i) we had better construct a mother node without the information of the non-head daughter rather than construct nothing (i.e., we had better construct a mother node by unifying only a head-daughter and a grammar rule), (ii) we had better construct a mother node with the maximal information of a non-head daughter rather than have no information of the non-head daughter added. Parse trees can be derived even if a parse tree cannot be derived by normal unification.

Offline robust parsing is based on A* algorithm, but we generate only parse trees which meet the following conditions, 1) a generated parse tree must be consistent with an existing bracketed corpus, and 2) the parsing cost of a generated parse tree must be minimum. This means that i) we can limit a search space, and that ii) the parsing result is valid in the sense that it is consistent with the existing corpus. The cost of a parse tree can be calculated by adding the cost of lenient default unification, which is the amount of information that is lost by lenient default unification. We regard it as the difference between the number of path values and structure-sharing in the results of a lenient default unification and a forced unification.

Grammar extraction is very concise. When we find a mother M in the result of offline parsing that cannot be derived by using unification but can be derived by default unification, we regard $M \rightarrow L, R$ as a new rule, where L and R are the daughters of the mother. The rules extracted in such a way can reconstruct the mothers as does default unification, and they reflect the condition of triggering default unification, i.e., the extracted rules are not frequently triggered because they can be applied to feature structures that are exactly equivalent to their daughter's part. By collecting a number of such rules,[3] a grammar becomes wide-coverage with some overgeneration. They can be regarded as exceptions in a grammar, which are difficult to be captured only by propagating information from daughters to a mother.

This approach can be regarded as a kind of explanation-based learning (Samuelsson and Rayner, 1991). The explanation-based learning method is recently attracting researcher's attention (Xia, 1999; Chiang, 2000) because their parsers are comparative to the state-of-the-art parsers in terms of precision and recall. In the context of unification-based grammars, Neumann (1994) has developed a parser running with an HPSG grammar learned by explanation-based learning. It should be also noted that Kiyono and Tsujii (1993) exemplified the grammar extraction approach using offline parsing in the

[2] In HPSG, both constituents and grammar rules are represented by feature structures.

[3] Although the size of the grammar becomes very large, the extracted rules can be found by a hash algorithm very efficiently. This tractability helps to use this approach in practical applications.

	Training Corpus	Test Set A	Test Set B
# of sentences	5,903	1,480	100
Avg. length of sentences	23.59	23.93	6.63

Table 1: Corpus size and average length of sentences

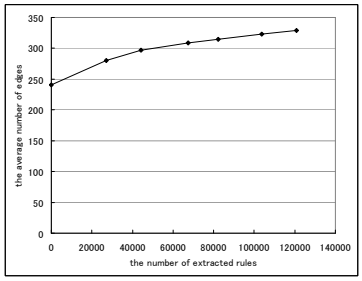

Figure 4: The average number of edges when *TestSetB* was parsed

Phenomena	(A)	(B)	(C)	(D) (%)
lack of lexical entry	118	32	86	72.9
inconsistency between XHPSG and Penn Treebank	44	13	31	70.5
punctuation, quotation, parenthesis	36	15	21	58.3
coordination	21	8	13	61.9
apposition	16	6	10	62.5
compound noun/adjective/adverb	14	3	11	78.6
Adv modifying PP	12	2	10	83.3
relative clause	12	5	7	58.3
topicalization	11	0	11	100.0
noun modifier	10	2	8	80.0
omission	8	2	6	75.0
parenthetical expression	7	3	4	57.1
verb saying	7	1	6	85.7
expression of frequency, NP + a + N	7	0	7	100.0
present participle construction	6	3	3	50.0
idiom	5	2	3	60.0
violation of agreement	3	0	3	100.0
adverbial noun	3	0	3	100.0
present progressive, be + Adv + present progressive	3	1	2	66.7
sentence modification from the beginning of a sentence	2	0	2	100.0
be + complement sentence	2	0	2	100.0
nominalization of adjective	1	0	1	100.0
double numerals (NP + roughly + double + NP)	1	1	0	0.0
total	349	99	250	71.6

(A) ... frequency of phenomena that the XHPSG grammar fails to analyze
(B) ... frequency of phenomena that the XHPSG grammar with the extracted rules fails to analyze
(C) = (A) – (B) ... frequency of phenomena that cannot be analyzed by the XHPSG grammar but can be analyzed by the XHPSG grammar with the extracted rules

(D) = (C)/(A) ... the ratio of phenomena that are recovered

Table 2: Analysis of phenomena that are recovered

context of explanation-based learning.

Finally, we need to remove some values in the extracted rules because they contain too specific information. For instance, a value of PHONOLOGY: represents a list of phoneme strings of a phrasal structure. Without removing them, extracted rules cannot be triggered until when completely the same strings appear in a text.[4]

6 Performance Evaluation

We measured the performance of our robust parsing algorithm by measuring coverage and degree of overgeneration for the Wall Street Journal in the Penn Treebank (Marcus et al., 1993). The training corpus consists of 5,903 sentences selected from the Wall Street Journal (Wall Street Journal 00 – 02), and we prepared two sets of test corpora, *TestSetA* and *TestSetB*. *TestSetA* consists of 1,480 sentences (Wall Street Journal 03) and is used for measuring coverage.[5] *TestSetB* consists of 100 sentences and is used for measuring the degree of overgeneration. The sentences of *TestSetB* are the shortest 100 sentences in *TestSetA*. Table 1 shows the average sentence length of each corpus. Here, 'coverage' means the ratio of 'the number of sentences that are covered by a grammar' to 'the number of all sentences'. Here, we say 'a sentence is covered' when a sentence can be analyzed by a parser and the result includes trees that are consistent with brackets and POS tags annotated in the Penn Treebank.

Grammar rules were extracted by offline parsing with the XHPSG grammar (Tateisi et al., 1998), which is a translation into HPSG of the manually-developed XTAG English grammar (The XTAG Research Group , 1995). The growth of the number of extracted rules is shown in the left of Figure 3. The average cost per sentence in offline parsing was 8.11. This means the total number of nodes and structure-sharing that are removed was less than 9 for each sentence. The coverage for the training corpus by offline parsing was 95.4%.

The coverage was measured by using the XHPSG grammar with the extracted rules. The coverage for *TestSetA* and *TestSetB* is illustrated in the middle and right of Figure 3, respectively. As seen in the figure, the coverage for the Wall Street Journal grew from 24.7% to 65.3% for *TestSetA* and from 64% to 88% for *TestSetB*.

We measured the degree of overgeneration by measuring the number of edges, using a parser based on A* algorithm. Figure 4 shows the average number of edges when *TestSetB* was parsed. From this figure and Figure 3, we can observe that the coverage grew from 64% to 88% by generating just 87.99 more edges (the number of edges grew from 240.68 to 328.67 in average).

From the experiments, we can say that our approach is effective in extending coverage with a little overgeneration.

We have analyzed the phenomena that cannot be analyzed by the original XHPSG grammar but can be analyzed by the extracted rules in the first 200 sentences in Wall Street Journal 03 of the test set. Among the 200 sentences, the original XHPSG grammar can cover 38 sentences (19% of the sentences) and the XHPSG grammar with the extracted rules can analyze 131 sentences (65.5% of the sentences). Table 2 shows the number of each

[4] In the experiment, we removed the values that correspond to the phoneme strings of phrasal structures, some of syntactic constraints, and semantics of phrasal structures

[5] There is no overlap between the training and test corpus.

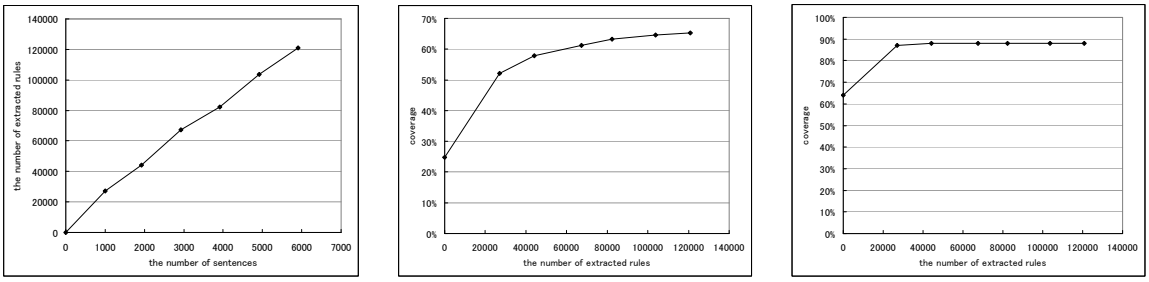

Figure 3: The number of extracted rules (left), coverage for *TestSetA* (middle) and *TestSetB* (right)

phenomenon that the original grammar fails to analyze ((A) in the table), and also shows the number of each phenomenon that the XHPSG grammar with the extracted rules still fails to analyze ((B) in the table). As seen in the table, more than 70% of phenomena that the original grammar cannot analyze were analyzed by our method. Note that most of the phenomena that cannot be analyzed with the extracted rules were lack of lexical entry, inconsistency between the grammar and the treebank, and complicated phenomena that are currently open problems in the field of linguistics.

Most of the lack of lexical entries failures were caused by the lack of 'apostrophe s.' This means that just by adding lexical entries for 'apostrophe s', we can cover almost half of this type of error. Among the words listed in the table, the XHPSG grammar has no lexical entry for 'itself' and 'as (Adv)'. As our method is only concerned with grammar rules, our method cannot recover words that have no lexical entry. This means that if a sentence includes the word 'itself', the sentence cannot be recovered by our method.

7 Conclusion

We proposed two new types of default unification, ideal and non-ideal lenient default unification. Ideal lenient default unification is desirable in that it maximizes the amount of information in the result, while other existing types of default unification maximize the amount of information in the default. Although non-ideal lenient default unification gives a less informative result than the ideal one, it works efficiently and retains the desiderata the ideal one satisfies.

We also proposed a new approach to extend the coverage of a grammar. We extracted grammar rules from the results of robust parsing using lenient default unification. A series of experiments showed that the extracted rules work robustly, and the coverage of the XHPSG grammar for Penn Treebank greatly increased with a little overgeneration.

References

G. Bouma. 1990. Defaults in unification grammar. In *Proc. of ACL-1990*, pages 165–172.

B. Carpenter, 1993. *Inheritance, Defaults, and the Lexicon*, chapter Skeptical and credulous default unification with applications to templates and inheritance, pages 13–37. Cambridge University Press, Cambridge.

D. Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proc. of ACL-2000*, pages 456–463.

A. Copestake. 1993. *The representation of lexical semantic information*. Ph.D. thesis, University of Sussex.

S. Douglas and R. Dale. 1992. Towards robust PATR. In *Proc. of COLING-1992*, pages 468–474.

O. Imaichi and Y. Matsumoto. 1995. Integration of syntactic, semantic and contextual information in processing grammatically ill-formed inputs. In *Proc. of IJCAI-1995*, pages 1435–1440.

M. Kiyono and J. Tsujii. 1993. Linguistic knowledge acquisition from parsing failures. In *Proc. of EACL-1993*, pages 222–231.

A. Lascarides and A. Copestake. 1999. Default representation in constraint-based frameworks. *Computational Linguistics*, 25(1):55–105.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

G. Neumann. 1994. Application of explanation-based learning for efficient processing of constraint-based grammars. In *Proc. of the 10th IEEE Conference on Artificial Intelligence for Applications*, pages 208–215.

The XTAG Research Group. 1995. A Lexicalized Tree Adjoining Grammar for English. Technical Report 95-03, IRCS, University of Pennsylvania.

G. Russell, J. Carroll, and S. Warwick-Armstrong. 1991. Multiple default inheritance in a unification-based lexicon. In *Proc. of ACL-1991*, pages 215–221.

C. Samuelsson and M. Rayner. 1991. Quantitative evaluation of explanation-based learning as an optimization tool for a large-scale natural language system. In *Proc. of IJCAI-1991*, pages 609–615.

Y. Tateisi, K. Torisawa, Y. Miyao, and J. Tsujii. 1998. Translating the XTAG English grammar to HPSG. In *Proc. of TAG+4*, pages 172–175.

F. Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proc. of NLPRS-1999*, pages 398–403.