An Empirical Study on Rule Granularity and Unification Interleaving Toward an Efficient Unification-Based Parsing System

Masaaki NAGATA
ATR Interpreting Telephony Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
nagata@atr-la.atr.co.jp

Abstract

This paper describes an empirical study on the optimal granularity of the phrase structure rules and the optimal strategy for interleaving CFG parsing with unification in order to implement an efficient unification-based parsing system. We claim that using "medium-grained" CFG phrase structure rules, which balance the computational cost of CFG parsing and unification, are a cost-effective solution for making unification-based grammar both efficient and easy to maintain. We also claim that "late unification", which delays unification until a complete CFG parse is found, saves unnecessary copies of DAGs for irrelevant subparses and improves performance significantly. The effectiveness of these methods was proved in an extensive experiment. The results show that, on average, the proposed system parses 3.5 times faster than our previous one. The grammar and the parser described in this paper are fully implemented and used as the Japanese analysis module in SL-TRANS, the speech-to-speech translation system of ATR.

1 Introduction

Unification-based framework has been an area of active research in natural language processing. Unification, which is the primary operation of this framework, provides a kind of constraint-checking mechanism for merging various information sources, such as syntax, semantics, and pragmatics. The computational inefficiency of unification, however, precludes the development of large practical NLP systems, although the framework has many attractive theoretical properties.

The efforts made to improve the efficiency of a unification-based parsing system can be classified into four categories.

- CFG parsing algorithm
- Graph unification algorithm
- Grammar representation and organization
- Interaction between CFG parsing and unification

There have been well-known efficient CFG parsing algorithms such as CKY [Aho and Ullman, 77], Earley [Earley, 70], CHART [Kay, 80], and LR [Aho and Ullman, 77] [Tomita, 86]. There have also been several recent in-depth studies into efficient graph unification algorithms, whose main concerns have been either avoiding irrelevant copies of DAGs [Karttunen and Kay, 85] [Pereira, 85] [Karttunen, 86] [Wroblewski, 87] [Godden, 90] [Kogure, 90] [Tomabechi, 91] [Emele, 91], or the exhaustive expansion of disjunctions into their disjunctive normal forms [Kasper, 87] [Eisele and Dörre, 88] [Maxwell and Kaplan, 89] [Dörre and Eisele, 90] [Carter, 90] [Nakano, 91].

There has, however, been little discussion regarding the optimal representation of a grammar, or linguistic knowledge, in the unification-based framework, from the engineering point of view. Grammar organization is highly flexible, as the unification-based framework uses two different forms of knowledge representation; atomic phrase structure rules and feature structure descriptions. Method selection greatly affects both the computational efficiency and the maintenance cost of the system. There has also been little discussion regarding optimal interaction between the CFG parsing process and the unification process in unification-based parsing, which also greatly affects overall performance.

Here we introduce the notion of *granularity*, and suggest *medium-grained* phrase structure rules, in which morpho-syntactic specifications in the feature descriptions are expanded into phrase structure rules. We claim that it reduce the computational loads of unification without intractably increasing the number of rules, and it is optimal in the sense that it satisfies both efficiency and maintainability. We also suggest *late unification* as another solution to the copying problem, as it avoids unnecessary copies of irrelevant subparses by delaying unification until a complete CFG parse is found.

In the following sections, the design and implementation of the medium-grained phrase structure rules is explained, then the implementation of the late unification is illustrated, and finally the effectiveness of the proposed methods is proven in experiments.

Granularity of Phrase Structure Rules	Constraints in Phrase Structure Rules	Constraints in Feature Descriptions	Number of Phrase Structure Rules
Extremely-Coarse-Grained	weak	very strong	$1 \sim 10$
Coarse-Grained	medium	strong	$10 \sim 100$
Medium-Grained	strong	medium	$100 \sim 1000$
Fine-Grained	very strong	weak	$1000 \sim$

Table 1: Granularity of phrase structure rules characterized by the number of rules and the strength of linguistic constraints in the phrase structure rules and the feature descriptions

2 The Granularity of Phrase Structure Rules

2.1 Granularity

Phrase structure rule granularity has been introduced to refer to the amount of linguistic constraints specified in the atomic CFG phrase structures rules without annotations. The rule granularity spectrum has been classified into four categories as shown in Table 1, using the number of grammar rules as a measure.

Unification-based grammars, in general, are characterized by a few general annotated phrase structure rules, and a lexicon with specific linguistic descriptions. This is especially true for HPSG [Pollard and Sag, 87] and JPSG [Gunji, 87], which are to be categorized as extremely-coarse grained, as they drastically reduce the number of phrase structure rules into two for English and one for Japanese, respectively. In these frameworks, the only role of the phrase structure rules is to provide a device for combining a head with its complement. Most linguistic constraints are stored in the feature descriptions.

Coarse-grained rules have been characterized as a grammar consisting of atomic phrase structure rules with medium constraints, and feature descriptions with strong constraints. Medium-grained rules have been characterized as a grammar consisting of atomic phrase structure rules with strong constraints, and feature descriptions with medium constraints. Medium-grained rules differ from coarse-grained rules in that they include morpho-syntax in the phrase structure rules, while coarse-grained rules include them in the feature descriptions. This means that medium-grained rules are strong enough to derive syntactic structures from atomic phrase structure rules without feature descriptions.

Grammars for conventional NLP systems using simple or augmented CFG fall into the category of fine-grained rules, which represent most of linguistic constraints as CFG phrase structure rules, and the number of rules usually amounts to an intractable number of several thousands for practical applications.

2.2 Maintainability and Efficiency

In unification-based framework, a linguistic constraint can either be described as atomic context-free phrase structure rules, or as feature descriptions in annotations and lexical entries. As the number of atomic phrase structure rules decreases, the number of feature descriptions increases.

It is true that the lexico-syntactic approach makes the grammar modular and improves its maintainability by reducing the number of rules . However, it must be noted that the computational cost of disjunctive feature structure unification, in the worst case, is exponential in the number of disjunctions [Kasper, 87], whereas the cost of CFG parsing is $o(N^3)$ in the input length N. Therefore, extreme rule reduction results in inefficiency. This overwhelms the benefits of the maintainability of the reduced number of rules since grammar development is essentially a trial-and-error process and requires a short turn-around time. However, the cost for CFG parsing also increases as the number of rules increases. Therefore, we must chose the granularity so that the reduction in unification cost outweighs the increase in CFG parsing cost, in order to gain overall efficiency.

3 The HPSG-Based Japanese Grammars

In this section, we illustrate the difference between "coarse-grained" rules and "medium-grained" rules using our HPSG-based spoken-style Japanese grammars as an example.

We have developed two unification-based grammars with different granularity[1], which are essentially based on HPSG and its application to Japanese (JPSG), for the analysis module [Nagata and Kogure, 90] of an experimental Japanese-to-English speech-to-speech translation system (SL-TRANS) [Morimoto et al. , 90].

We have selected the "secretarial service of an international conference registration" as our task domain, in which a conversation between a secretary and a questioner is carried out. The Japanese grammars, however, are not task-specific, but rather general-purpose ones, which cover a wide range of phenom-

[1] Historically speaking, we first developed coarse-grained rules and then we manually transformed them into medium-grained rules for efficiency.

ena at many linguistic levels from syntax, and semantics, to pragmatics using typed feature structure descriptions. The linguistic phenomena covered in these grammars include:

- *Fundamental Constructions*: causative, passive, benefactive, negation, interrogative, etc.,
- *Control and Gaps*: subject/object control,
- *Unbounded Dependencies*: topic, relative,
- *Word Order Variation and Ellipsis.*

3.1 Coarse-Grained Rules vs. Medium-Grained Rules

The coarse-grained HPSG-based Japanese grammar has about 20 generalized phrase structure rules, while the medium-grained grammar has about 200 phrase structure rules. Both grammars use the same lexicon with a vocabulary of about 400. [2]

In the coarse-grained grammar, phrase structure rules only refer to the relative position between the five basic syntactic categories for Japanese: verb (V), noun (N), adverb (ADV), postposition (P), and attributive (ATT). Most of the specific linguistic information is encoded as feature descriptions in either the annotation of the phrase structure rules or the lexical entries. In principle, there is no distinction as to whether a constituent is lexical or phrasal, and no subcategories of the 5 basic categories. This contributes greatly to the reduction in the number of phrase structure rules, which results in better grammar maintainability. We present all the phrase structure rules of the coarse-grained Japanese grammar in Appendix A.

It has been noticed that the extensive use of disjunctions in feature descriptions, which results from the reduction of the number of phrase structure rules, is the main cause of inefficiency in the coarse-grained version of the grammar. The three major sources of disjunctions are, morpho-syntactic specifications for diverse expressions in the final part of the sentence, free word order and ellipsis of verb complements (subcat slash scrambling), and semantic interpretation of deep case and aspect, where the first two particularly are the problems in spoken-style Japanese.

We have manually converted the coarse-grained phrase structure rules into medium-grained rules to reduce the computational cost of unification. First, we divided each of the basic categories into several subcategories. Then, we divided the coarse-grained phrase structure rules according to the subcategories. To keep the grammar readable, however, we choose to leave the subcat slash scrambling and the semantic interpretation undone, and made extensive efforts to expand the morpho-syntactic specifications.

3.2 Example: Medium-Grained Rules for Predicate Verb Phrases

In this section, we illustrate the process of transformation using a predicate verb phrase production rule as an example. Japanese predicate phrases consist of a main verb followed by a sequence of auxiliaries and sentence final particles. There is an almost one-dimensional order of verbal constituents such as in Figure 1, which reflects the basic hierarchy of the Japanese sentence structure.

Kernel verbs occur first in a predicate phrase sequence. Voice auxiliaries precede all other auxiliaries, and within this category, the causative auxiliary *(sa)seru* precedes the passive auxiliary *(ra)reru*. Aspect auxiliaries, such as the progressive auxiliary *(te)iru* precede modal auxiliaries and follow voice auxiliaries. Modal auxiliaries are classified into two groups with respect to the relative order of negative and tense auxiliaries. Mood1 includes the optative auxiliaries, such as *tai* (want), *beki* (should/must), etc. Mood2 includes the evidential or inferential auxiliaries such as *rashii* (seem/look), *kamoshirenai* (may), etc. Negative auxiliaries *nai, n* (not) follow voice, aspect, and mood1 auxiliaries, and precede tense and mood2 auxiliaries. Tense auxiliaries *ta, da* (-ed) show irregular behavior. They follow the voice, aspect, mood1, and negative auxiliaries, and precede the mood2 auxiliaries. They also can follow the mood2 auxiliaries.

In the coarse-grained grammar, we provide a single phrase structure rule for the phenomena.

$$V \rightarrow (V \quad AUXV) \tag{1}$$

The order constraints between auxiliaries are specified in the annotation of rule (1) and each lexical entry by the combination of the syntactic features, such as the syn|head|subcat for preceding constituents, the syn|head|coh for following constituents, and the syn|head|mod1 for the position of the constituents in the verb phrase hierarchy. For example, the causative auxiliary verb *seru* has the following feature bundles in its syn|head|mod1 feature.

```
[[CAUS +][DEAC -][ASPC -][DONT -][OPTT -]
 [NEGT -][PAST -]
 [EVID -][TENT -][POLT -][POLT-AUX -][INTN -]
 [SFP-1 -][SFP-2 -][SFP-3 -]]
```

In converting the rule, first we have classified the verbal phrasal categories according to the hierarchy, e.g. V-kernel, V-aspect, V-mood1, V-negt, V-mood2, and V-tense, then we have subcategorized the auxiliaries as shown in Table 2. Thus, the coarse-grained phrase structure rule (1) is converted to the 32 medium-grained grammar rules in Appendix B.

[2] We also have another version of the grammar for the same subcorpus, which is used for the continuous speech recognition module [Takezawa et al. , 91]. It only uses atomic CFG rules, and the number of rules amounts to more than 2,000. It is, therefore, categorized as a fine-grained grammar in our definition.

```
kernel < voice < aspect < mood1 < negate < tense < mood2 < tense
       (sa)seru (te)iru   tai     nai      ta      rasii   ta
       (ra)reru (te)morau tagaru  n        da      desu    u
                          masu                    darou
```

Figure 1: The predicate hierarchy of Japanese

AUXV-caus	causative auxiliary: *(sa)seru*
AUXV-deac	passive auxiliary: *(ra)reru*
AUXV-aspc	aspect auxiliary: *(te)iru, (te)aru*
AUXV-dont	benefactive auxiliary: *(te)morau*
AUXV-optt	optative auxiliary: *tai, beki*
AUXV-negt	negative auxiliary: *nai, n*
AUXV-tense	tense auxiliary: *ta, da*
AUXV-evid	evidential auxiliary: *rashii, darou*
AUXV-copl	copulative auxiliary: *da, desu*

Table 2: Subcategories of auxiliaries in the medium-grained grammar

4 Interleaving CFG Parsing and Unification

4.1 Strategies for Evaluating Feature Descriptions

Unification is an expensive operation, so the point of evaluating feature descriptions during CFG parsing has serious affects on the overall performance. We have implemented two strategies for feature description evaluation:

Early Unification (Step-by-step Strategy) Feature descriptions are evaluated step-by-step, at each rule invocation in the CFG parsing.

Late Unification (Pipeline Strategy) Feature descriptions are evaluated when a complete CFG parse is found. The "well-formedness" of a parse derived from atomic CFG rules is verified by evaluating associated feature descriptions.

The granularity of the phrase structure rules is closely related to the proper selection of the evaluation strategy. Since the atomic phrase structure rules in the coarse-grained grammar are not so strong as to constrain syntactic structures, we have to employ the early unification to avoid a number of irrelevant subparses which should have been eliminated by the evaluation of annotations. However, since the atomic rules in the medium-grained grammar have detailed morpho-syntax specifications, they should be able to avoid irrelevant copies by using the late unification.

4.2 Implementing the Evaluation Strategies

We have implemented the various evaluation strategies by doing additional housekeeping in the underlying parser. The parser used here is called the Typed Feature Structure Propagation Parser (TFSP Parser) [Kogure, 89], which is based on the active chart parsing algorithm [Kay, 80] and typed feature structure unification [Aït-Kaci, 86].

```
procedure AcpContinue(chart)
begin
 if SatisfySuspendingCondition?(chart)
  then return chart
 else
  AcpContinue(AcpOneStep(chart))
end

procedure AcpOneStep(chart)
begin
 pendingedge = GetPendingEdge(chart)
 AddEdge(pendingedge)
 if EdgeActive?(pendingedge) then
  TryToContinueActiveEdge(pendingedge, chart)
 else
  TryToContinueInactiveEdge(pendingedge, chart)
 ProposeProductions(pendingedge)
 return chart
end
```

Figure 2: Iterative Rule Invocation in an Active Chart Parsing Algorithm

The active chart parser and the unification algorithm are implemented in C on Sun4, which is a 10-MIPS work station. The unification algorithm is based on nondestructive graph unification [Wroblewski, 87], which we extend to treat negation, loop, type symbol subsumption relationships, and disjunction. Successive approximation [Kasper, 87] is used for disjunctive feature structure unification.

The Active chart parsing algorithm basically consists of chart initialization and iterative rule invocation. The basic part of the iterative rule invocation is shown in Figure 2. AcpContinue checks the suspending condition and calls rule invocation recursively. AcpOneStep carries out a cycle of rule invocation which consists of getting a new pending edge (GetPendingEdge), adding it to the chart (AddEdge), combining active and inactive edges (TryToContinueActiveEdge/TryToContinueInactiveEdge), and proposing new edges (ProposeProductions). The parser stops (SatisfySuspendingCondition?) when it finds an inactive edge whose starting and ending vertex are the left-most and right-most vertex of the chart, respectively, and whose label is the start symbol of the grammar.

In early unification, the feature descriptions are evaluated when the edges are combined, while in late unification, they are evaluated in the chart suspending condition check only if the chart suspending condition holds. Delaying unification is implemented by adding a slot edge.parse to the edge structure, which keeps a list of the pair of active and inactive edges constructing the edge. If either or both of the argument feature structures of the unification have not been evaluated, they are recursively evaluated to get the target feature structure.

It has to be noted that some derivations that terminate when feature descriptions are evaluated, may not terminate if they are ignored. For example, it is possible to write a rule for unbounded dependency like (2), in which an element in the subcat feature is moved to the slash feature, to introduce slashed categories dynamically[3].

$$\alpha \rightarrow (\alpha) \qquad (2)$$

Ignoring feature descriptions in the rule may cause an infinite loop. Therefore, feature descriptions are forced to be evaluated, when rules that cause a loop are encountered in late unification.

5 Experiment

The effectiveness of the strategies proposed in this paper can be judged by observing their behavior in practice. We have tested the time behavior of parsing with respect to rule granularity and interleaving strategy of CFG parsing and unification. 85 sample sentences are used. These are selected from the sample subcorpus of ATR's dialogue corpus whose task domain is the "secretarial service of an international conference". The average length of the sample sentences is 11.0 characters, and their maximum and minimum length are 2 and 28 characters, respectively.

We have developed two Japanese grammars of different granularity with almost the same coverage. The *coarse-grained rules* consist of 22 generalized phrase structure rules with detailed feature description in their annotations, while the *medium-grained rules* consist of 164 detailed phrase structure rules with less detailed feature descriptions. Both grammars use the same lexicon with about 400 lexical entries. We have also implemented two different feature description evaluation modes in the active chart parser. The *early unification evaluation mode* evaluates the feature descriptions at each rule application (the step-by-step strategy). The *late unification evaluation mode*, on the other hand, delays unification until a complete syntactic structure is found by using the atomic phrase structure rules only (the pipeline strategy).

The average parsing time is shown in Table 3. It shows that, on average, the medium-grained grammar rules are 1.7 times more efficient than the coarse-grained rules in the early unification mode, and that the late unification mode is 2.0 times more efficient than the early unification mode with the medium-grained grammar. Moreover, when the medium-grained grammar rules and the late unification mode are combined, the new parser runs 3.5 times faster than the previous one using the coarse-grained grammar rules and the early unification.[4]

[3] In our implementation, for efficiency reasons, we generate all the appropriate combinations of subcat and slash in advance, and keep them as a disjunctive feature structure

Rule Granularity	Coarse	Medium	Medium
Unification Mode	Early	Early	Late
Average Runtime	30.2 sec	17.8 sec	8.7 sec
Relative Speed	1.0	1.7	3.5

Table 3: Average parsing time with respect to granularity and unification mode

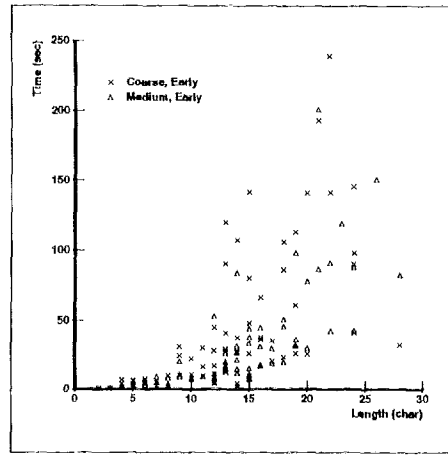

Figure 3: Comparison of Coarse-Grained Rules and Medium-Grained Rules

6 Discussion

The relationship between input length and parsing time with respect to grammar granularity is shown in Figure 3. In general, the medium-grained rules performed better than the coarse-grained rules. This tendency becomes clearer, as the sentence length increases. This results from the reduction of disjunctive feature descriptions whose computational cost in-

[4] The approach of saving unnecessary copies for irrelevant subparses in a parsing process by late unification is orthogonal to the approaches of saving unnecessary copies within a unification process, such as [Tomabechi, 91]. Therefore, the effects of speed up can be multiplied. We have already implemented his quasi-destructive graph unification, and the preliminary experiment result shows that the parser with new unifier runs almost twice as fast as the one reported in this paper.

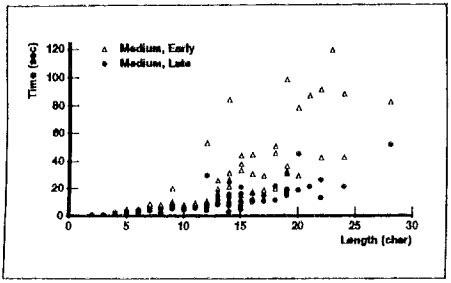

Figure 4: Comparison of Early Unification and Late Unification

creases exponentially in the number of disjunctions. However, we occasionally encounter sentences which are parsed faster using coarse-grained rules rather than medium-grained rules. This is because the increase in the atomic CFG parsing cost exceeds the reduction of the unification cost.

The relationship between input length and parsing time with respect to unification evaluation mode is shown in Figure 4. This shows that the late unification mode is significantly more efficient than the early unification mode. It also shows that the parsing time in the late unification mode seems to be polynomial (not exponential) in the input length, while that in the early unification mode varies widely and irregularly. This is because the parsing time in the late unification mode is mainly predominated by the cost of atomic CFG parsing by delaying unification, whereas the parsing time in the early unification mode is mainly predominated by the cost of unification.

We have demonstrated the effectiveness of combining medium-grained phrase structure rules with late unification. Experiment results suggest that new prospective techniques for speeding up unification-based parsing exist.

The first is automatic transformation of phrase structure rules, which converts disjunctions in the feature descriptions into atomic phrase structure rules. Some disjunctions such as subcat slash scrambling are so regular that it seems possible to expand them into a set of CFG rules using formal procedures. If the grammar compiler can perform this kind of transformation automatically, we can gain efficiency without losing grammar maintainability.

The second is feature-sensitive lazy unification. Unification is used for both building up a structure using information-propagation and blocking rule application using constraint-checking. If the grammar compiler can separately output those features for constraint-checking such as syntactic features, and those for information-propagation such as semantic representations, irrelevant subparses can be pruned efficiently by evaluating the constraint-checking features first. Unification is an associative and commutative operation, so the same results from the feature-sensitive lazy unification are assured.

The third is parallel implementation of a unification-based parser based on late unification (pipe-line strategy). In early unification (step-by-step strategy), it is hard to perform parsing in parallel because the CFG parsing process and the unification process depend strongly on each other. However, both processes are completely separated in the pipe-line strategy . Therefore, it is easy to introduce the existing parallel algorithms to both CFG parsing and unification. It is estimated that most feature descriptions can be evaluated in parallel, at least, at the lexical level, because unification-based grammars such as HPSG derive phrase structure by iteratively propagating the local constraints.

7 Conclusion

In this paper, we have proposed two techniques for implementing an efficient unification-based parsing system, which, when combined, significantly improve the overall performance. The first is changing the granularity of the context-free phrase structure rules into medium-grained rules. This enables us to reduce the amount of unification for feature descriptions without intractably increasing the number of phrase structure rules. The second is late unification in which the unification for feature descriptions is delayed until a complete CFG parse is found. This saves unnecessary copies of feature structures which are wasted for irrelevant subparses.

We have tested the time behavior of the parsing system using two grammars of different granularity (coarse/medium) and two different strategies for invoking unification (early/late). It is proved that, on average, late unification using medium-grained rules parses 3.5 times faster than the previous early unification using coarse-grained rules.

Acknowledgments

The author would like to thank Dr. Kurematsu, and all the members of ATR Interpreting Telephony Research Labs. for their constant help and fruitful discussions.

References

[Aho and Ullman, 77] Aho, A. and Ullman, J., *Principles of Compiler Design* Addison-Wesley, 1977.

[Aït-Kaci, 86] Aït-Kaci, H., "An Algebraic Semantics Approach to the Effective Resolution of Type Equations," *Journal of Theoretical Computer Science 45*, 1986.

[Carter, 90] Carter, D., "Efficient Disjunctive Unification for Bottom-Up Parsing," *Proc. of COLING-90*, 1990.

[Dörre and Eisele, 90] Dörre and Eisele, "Feature Logic with Disjunctive Unification," *Proc. of COLING-90*, 1990.

[Earley, 70] Earley, J., "An Efficient Context-Free Parsing Algorithm," *ACM, 13, 2*, 1970.

[Eisele and Dörre, 88] Eisele, A. and Dörre, J., "Unification of Disjunctive Feature Descriptions," *Proc. of the 26th ACL*, 1988.

[Emele, 91] Emele, M., "Unification with Lazy Non-Redundant Copying," *Proc. of the 29th ACL*, 1991.

[Godden, 90] Godden, K., "Lazy Unification," *Proc. of the 28th ACL*, 1990.

[Gunji, 87] Gunji, T., *Japanese Phrase Structure Grammar — A Unification-Based Approach*, Dordrecht, Reidel, 1987.

[Kasper, 87] Kasper, R., "A Unification Method for Disjunctive Feature Descriptions," *Proc. of the 25th ACL*, 1987.

[Karttunen, 86] Karttunen, L., D-PATR — A Development Environment for Unification-Based Grammars, CSLI-86-91, CSLI, 1986.

[Karttunen and Kay, 85] Karttunen, L. and Kay, M., "Structure Sharing with Binary Trees," *Proc. of the 23rd ACL*, 1985.

[Kay, 80] Kay, M., *Algorithm Schemata and Data Structures in Syntactic Processing*, Technical Report CSL-80-12, Xerox PARC, 1980.

[Kogure, 89] Kogure, K., "Parsing Japanese Spoken Sentences based on HPSG," *Proc. of the IWPT*, 1989.

[Kogure, 90] Kogure, K., "Strategic Lazy Incremental Copy Graph Unification," *Proc. of COLING-90*, 1990.

[Maxwell and Kaplan, 89] Maxwell, J. and Kaplan, R., "An Overview of Disjunctive Constraint Satisfaction," *Proc. of the IWPT*, 1989.

[Morimoto et al., 90] Morimoto,T. et al., "Integration of Speech Recognition and Language Processing in Spoken Language Translation System (SL-TRANS)," *Proc. of the ICSLP*, 1990.

[Nagata and Kogure, 90] Nagata, M. and Kogure, K., "HPSG-Based Lattice Parser for Spoken Japanese in a Spoken Language Translation System," *Proc. of the 9th ECAI*, 1990.

[Nakano, 91] Nakano, M., "Constraint Projection: An Efficient Treatment of Disjunctive Feature Descriptions," *Proc. of 29th ACL*, 1991.

[Pereira, 85] Pereira, F., "A Structure-Sharing Representation for Unification-Based Formalisms," *Proc. of the 23rd ACL*, 1985.

[Pollard and Sag, 87] Pollard, C. and Sag, I., *An Information-Based Syntax and Semantics*, CSLI Lecture Notes No. 13, CSLI, 1987.

[Takezawa et al., 91] Takezawa, T. et al., "Linguistic Constraints for Continuous Speech Recognition in Goal-Directed Dialogue," *Proc. of ICASSP-91*, 1991.

[Tomabechi, 91] Tomabechi, H., "Quasi-Destructive Graph Unification," *Proc. of 29th ACL*, 1991.

[Tomita, 86] Tomita, M., *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*, Kluwer Academic Publishers, 1986.

[Wroblewski, 87] Wroblewski, D., "Nondestructive Graph Unification," *Proc. of the 6th AAAI*, 1987.

Appendix

A Coarse-grained Grammar Rules for Japanese

The name of each rule is shown in the comment, where the suffix *-ah*, *-ch*, *-coord* means adjunction, complementation, and coordination, respectively.

```
;;; Start Symbol
start -> (V)                    ; start-rule
;;; Noun Phrase
n -> (att n)                    ; att-n-ah
n -> (p n)                      ; p-n-ah
n -> (v fn)                     ; v-fn-ch
n -> (v n)                      ; v-n-ah
n -> (p n)                      ; n-n-coord
;;; Complex and Compound Noun
n -> (nprefix n)                ; nprefix-n-ah
n -> (n n)                      ; n-n-ah
n -> (n nsuffix)                ; n-nsuffix-ch
;;; Postpositional Phrase
p -> (n postp)                  ; n-postp-ch
p -> (v postp)                  ; v-postp-ch
p -> (p postp)                  ; p-postp-ch
;;; Adverbial Phrase
adv -> (adv postp)              ; adv-postp-ch
adv -> (v fadv)                 ; v-fadv-ch
adv -> (n fadv)                 ; n-fadv-ch
;;; Verb Phrase
v -> (p v)                      ; p-v-any
v -> (adv v)                    ; adv-v-ah
;;; Predicate Verb Phrase
v -> (v auxv)                    ; v-auxv-ch
v -> (n auxv)                   ; n-auxv-ch
v -> (adv auxv)                 ; adv-auxv-ch
;;; Verb Inflection
v -> (vstem vinfl)              ; v-stem-infl
auxv -> (auxvstem vinfl) ; auxv-stem-infl
```

B Medium-grained Rules for Japanese Verb Phrases

The coarse-grained grammar rule (1) is converted to the following 32 medium-grained grammar rules.

```
V-voice -> (V-kernel  AUXV-voice)
V-aspect -> (ADV  AUXV-aspc)
V-aspect -> (ADV  AUXV-dont)
V-mood1 -> (V-kernel  AUXV-optt)
V-mood1 -> (V-voice  AUXV-optt)
V-mood1 -> (V-aspect  AUXV-optt)
V-negt -> (V-kernel  AUXV-negt)
V-negt -> (V-voice  AUXV-negt)
V-negt -> (V-aspect  AUXV-negt)
V-negt -> (V-mood1  AUXV-negt)
V-tense -> (V-kernel  AUXV-tense)
V-tense -> (V-voice  AUXV-tense)
V-tense -> (V-aspect  AUXV-tense)
V-tense -> (V-mood1  AUXV-tense)
V-tense -> (V-negt  AUXV-tense)
V-mood2 -> (V-kernel  AUXV-evid)
V-mood2 -> (V-voice  AUXV-evid)
V-mood2 -> (V-aspect  AUXV-evid)
V-mood2 -> (V-mood1  AUXV-evid)
V-mood2 -> (V-negt  AUXV-evid)
V-mood2 -> (V-tense  AUXV-evid)
V-tense -> (V-mood2  AUXV-tense)
AUXV-voice -> (AUXV-caus)
AUXV-voice -> (AUXV-deac)
AUXV-voice -> (AUXV-caus  AUXV-deac)

V -> (V-kernel)
V -> (V-voice)
V -> (V-aspect)
V -> (V-mood1)
V -> (V-negt)
V -> (V-tense)
V -> (V-mood2)
```