Vol. 35 No. 5 Transactions of Information Processing Society of Japan May 1994

Regular Paper

An Empirical Study on Rule Granularity and Unification Interleaving in Unification-Based Parsers

MASAAKI NAGATA†,* and TSUYOSHI MORIMOTO†,**

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 balances the computational cost of CFG parsing and unification, and is a cost-effective solution for making unification-based grammar both efficient and easy to maintain. We also claim that "late unification," which dalays unification until a complete CFG parse is found, reduces unnecessary copying of DAGs for irrelevant subparses and so improves performance significantly. The effectiveness of these methods is proven by an extensive experiment. The results show that, on average, the proposed system parses 2.1 times faster than our previous one. The grammar and the parser described in this paper have been fully implemented and used as the Japanese analysis module in SL-TRANS, the speech-to-speech translation system of ATR.

1. Introduction

The 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. This is unfortunate because the framework has many attractive theoretical properties.

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

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

There are several well-known efficient CFG parsing algorithms such as CKY,[1] Earley,[3] CHART,[11] and LR.[1),24] Several recent studies have examined efficient graph unification algorithms in depth. The main concerns have been either avoiding irrelevant copying of DAGs,[25),7),13),5),23] or the exhaustive expansion of disjunctions into their disjunctive normal forms.[10),4),14),19]

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 the optimal interaction between the CFG parsing process and the unification process in unification-based parsing. Since the theoretical and empirical computational behavior of the two components is significantly different, the design of the interface between them also greatly affects overall performance.

We introduce the notion of *granularity*, which refers to the amount of linguistic specification contained within an atomic phrase structure rule. This paper recommends *medium-grained* phrase structure rules, in which morpho-syntactic specifications in the feature descriptions are expanded into atomic phrase structure rules. We claim that use of such rules reduces the computational loads of unification without intractably increasing the number of rules, and

† ATR Interpreting Telephony Research Laboratories
* Presently with NTT Network Information Systems Laboratories
** Presently with ATR Interpreting Telecommunications Research Laboratories

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.

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	1～10
Coarse-grained	medium	strong	10～100
Medium-grained	strong	medium	100～1000
Fine-grained	very strong	weak/none	1000～

that this strategy is optimal in the sense that it achieves both efficiency and maintainability.

We also introduce the notion of an *interleaving strategy* with reference to the interface between CFG parsers and unifiers, more specifically, the point at which feature descriptions are evaluated. Research shows the superiority of *late unification*, which avoids unnecessary copying of irrelevant subparses by delaying unification until a complete CFG parse is found. This reduces the number of unnecessary copies of DAGs made for irrelevant subparses, and improves the performance significantly, especially when the interleaving strategy is combined with medium-grained phrase structure rules.

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

2. Granularity of Phrase Structure Rules

2.1 Variations of Granularity

The term "granularity of phrase structure rules" refers to the amount of linguistic constraints specified in the atomic CFG phrase structures rules excluding annotations. The rule granularity spectrum can be classified into four categories as shown in **Table 1**, using the number of phrase structure rules as the measure.

Unification-based grammars, such as LFG[9] and GPSG,[6] are characterized by a few, general annotated phrase structure rules, and a lexicon with specific linguistic descriptions. They are implicitly assumed to be "coarse-grained" because their phrase structure rules only refer to the possible combinations of major constituents (NP, VP, etc.), and a detailed linguistic specification is given in the feature descriptions. This is especially true for HPSG[20] and JPSG.[8] They can be categorized as "extremely coarse-grained," as they drastically reduce the number of phrase structure rules into two for English and one for Japanese.

On the other hand, grammars for conventional NLP systems using simple or augmented CFG fall into the category of "fine-grained" because they represent most linguistic constraints in atomic CFG phrase structure rules. They usually contain several thousands of rules, an intractable number for practical applications.

In this paper, we will suggest an intermediate organization for phrase structure rules, which we call "medium-grained." Compared with "coarse-grained" rules, they are characterized by atomic phrase structure rules with more detailed structural descriptions (strong constraints), and feature descriptions with less linguistic information (medium constraints).* Medium-grained rules are designed to be strong enough to derive syntactic structures from atomic phrase structure rules without feature descriptions.

2.2 Maintainability and Efficiency

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

As a first order approximation, it may be ture that it is easier to maintain the grammar as the number of phrase structure rules decreases. Therefore, it may also be true that the lexico-

* It is very difficult to give an objective measure for the strength of linguistic specification in phrase structure rules and feature descriptions. However, we will roughly compare them by the success rate of unification in pruning edges during parsing, as described in Section 5.

syntactic approach, such as in HPSG, 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,[10] 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.** Therefore, we believe that medium-grained rules make a unification-based parsing system both efficient and easy-to-maintain.

3. The HPSG-Based Japanese Grammars

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

We have developed two unification-based grammars with different granularities.*** They are essentially based on HPSG and its application to Japanese (JPSG), for the analysis module[18] of an experimental Japanese-to-English speech-to-speech translation system (SL-TRANS).[16]

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. The vocabulary in the lexicon is about 400 for both grammars.*

We selected "secretarial services 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 are not task-specific, but are rather general-purpose ones and cover a wide range of phenomena 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

In the coarse-grained grammar, phrase structure rules, in principle, only refer to the relative position between the five basic syntactic categories for Japanese: verb (V), noun (N), adverb (ADV), postposition (P), and adnominal (ADN). There is, moreover, no distinction as to whether a constituent is lexical or phrasal.** Most of the specific linguistic information is encoded as feature descriptions in either the annotation of the phrase structure rules or the lexical entries. This enables us to reduce the number of phrase structure rules, which results in better grammar maintainability.

We present some representative coarse-grained phrase structure rules for Japanese in the following:

* You may find this discussion strange because the complexity of a problem must be the same regardless of the form of knowledge representation. Strictly speaking, the complexity of finding the *first* parse is polynomial. If the grammar is ambiguous and if we try to find *all* parses, the worst case complexity is exponential. On the other hand, the worst case complexity of finding *all* feature structures that satisfy the disjunctive feature description is exponential. So, the true problem lies in the fact that, at the moment, we have no good polynomial algorithm that can find the *first* and, preferably, the most plausible answer for disjunctive unification.

** However, note also that the cost for CFG parsing increases with the number of rules. Therefore, we must choose the right degree of granularity so that the reduction in unification cost outweighs the increase in CFG parsing cost.

*** Historically speaking, we first developed coarse-grained phrase structure rules and then manually transformed them into medium-grained phase structure rules for efficiency.

* We also have another version of the grammar for the same subcorpus, which is used in a continuous speech recognition module.[22] It uses only atomic CFG rules, and the number of rules amounts to more than 2,000. It is, therefore, categorized as fine-grained by our definition.

** For practical use, there are some exceptions to these principles. For example, verbals are subcategorized as phrasal verbals (V) and lexical auxiliaries (AUXV), and postpositionals are subcategorized as postpositional phrase (P) and lexical postposition (POSTP).

Table 2 Grammar statistics : the coarse-grained grammar vs. the medium-grained grammar.

		count	node	arc	conjunction	disjunction	atomic value	variable
Coarse	PS rules	22	91.1	113.9	6.5	1.9	8.7	25.2
	lexicon	438	93.4	101.8	5.5	1.9	28.9	12.9
Medium	PS rules	163	63.2	88.7	2.3	0.5	3.3	24.7
	lexicon	504	83.5	83.5	4.9	1.7	22.7	12.8

$$START \rightarrow (V) \quad (1)$$
$$V \rightarrow (P \quad V) \quad (2)$$
$$P \rightarrow (N \quad POSTP) \quad (3)$$
$$V \rightarrow (ADV \quad V) \quad (4)$$
$$N \rightarrow (ADN \quad N) \quad (5)$$

where *START* is the start symbol of the grammar. Rule (1) states that any verb phrase is accepted as a sentence in this grammar, reflecting the flexibility of ellipsis in spoken-style Japanese. Rule (2) and rule (3) stipulate the basic construction of Japanese, where a noun followed by a postpositional particle constitutes a postpositional phrase, and the predicate verb takes it as an argument. Rule (4) and rule (5) describe that a verb is modified by an adverb, while a noun is modified by an adnominal.

In the coarse-grained grammar, we 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. 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 in particular are problems in spoken-style Japanese.

We have manually converted the coarse-grained phrase structure rules into medium-grained ones to reduce the computational cost of unification. First, we divided each of the basic categories into several subcategories. We then 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.

Table 2 shows the average numbers of nodes, arcs, conjunctions, disjunctions, atomic values, and variables of the feature description in a phrase structure rule and a lexical entry, for the coarse-grained grammar and the medium-grained grammar.* The number of nodes and arcs represent the size of the feature structure (DAG), while the number of conjunctions and disjunctions represent the complexity of feature description (AND-OR Tree). Moreover, the number of atomic values and variables roughly corresponds to the amount of linguistic specification encoded in the feature structure.

In Table 2, we can see that the size of the feature structure associated with a phrase structure rule in the medium-grained grammar is reduced to about 70% of that of the coarse-grained grammar, and that the number of conjunctions, disjunctions and atomic values is also reduced accordingly. The size of the feature structure for lexical entries in the medium-grained grammar is also reduced, reflecting the descriptional change in the phrase structure rules. However, the number of variables is almost the same, because variables are used for implementing co-reference, which is mainly used in the specification for the feature propagation in HPSG, according to the HEAD Feature Principle and the SUBCAT Feature Principle, etc.

3.2 An Example of Changing Granularity

There are four major points that we focussed on in subdividing the coarse-grained phrase structure rules into medium-grained ones.

- word order in predicate phrases, reflecting the predicate hierarchy of Japanese.
- syntax of adverbials, in line with the predicate hierarchy.
- word order of adjacent postpositional particles, especially between case particles and topicalizing particles.

* The coarse-grained grammar has 22 phrase structure rules and 438 lexical entries, while the medium-grained grammar has 163 phrase structure rules and 504 lexical entries. The difference in the number of lexical entries is nominal, because it mainly results from the subdivision of the categories.

```
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
```

Fig. 1 The predicate hierarchy of Japanese.

- structuring of complex and compound nouns, such as name, address, and date.

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: see **Fig. 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.*

$$V \rightarrow (V \quad AUXV) \qquad (6)$$

The order constraints between auxiliaries are specified in the annotation of rule (6) and lexical entries using syntactic features, which must be combined properly. We use **syn|head|subcat** for preceding constituents, **syn|head|coh** for following constituents, and **syn|head|modl** for the position of the constituents in the verb phrase hierarchy.* For example, if we straightforwardly specify the order constraints of the causative auxiliary verb, such as *seru*, we must have the following feature bundles in its **syn|head|modl** 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 classify the verbal phrasal categories according to the hierarchy, e.g. V-kernel, V-aspect, V-mood1, V-negt, V-mood2, and V-tense, we then subcategorize the auxiliaries as shown in **Table 3**. Thus, the coarse-grained phrase structure rule (6) is converted to the 32 medium-grained grammar rules as shown in Appendix A.***

4. Interleaving CFG Parsing and Unification

4.1 Strategies for Evaluating Feature Descriptions

There seems to be an implicit assumption that feature description must be evaluated at every rule invocation in order to prune irrelevant subparses, and this strategy is used almost with-

* From the surface form of the rule (6), the left daughter of the righthand side *V* seems to be the head constituent. In JPSG, however, the right daughter *AUXV* is considered to be the head, because the syntactic behavior of the mother is mainly stipulated by the right daughter.

* In order to refer to a particular subpart of a feature structure, we use the notation like **syn|head|subcat**, which enumerates the sequence of label names up to the subpart.

** In fact, we can dispense with part of the feature bundle in the **modl** of auxiliaries, if we can guarantee that there are no erroneous input sentences that violate this hierarchy. We still need, however, the feature bundle in the **modl** of adverbials in order to properly specify the constituent that a adverbial modifies.

*** We do not intend to claim that the phrase structure rules listed here completely specify all the Japanese morph-syntactic constraints. What we intend to do is to illustrate the difference between coarse-grained and medium-grained rules.

Table 3 Subcategories of auxiliaries in the medium-grained grammar.

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*

out question in previous unification-based parsing systems. Unification is, however, an expensive operation, so the point at which feature descriptions are evaluated during CFG parsing seriously impacts the overall performance. In this paper, we will think of unification-based parsing as two mutually interleaving processes, namely, CFG parsing and unification. From that viewpoint, we distinguish two strategies for interleaving:

Early Unification (Step-by-step Strategy) Feature descriptions are evaluated step-by-step, at each rule invocation under the CFG parsing. This is the conventional strategy.

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 early unification to avoid creating a number of irrelevant subparses which will be eliminated by the evaluation of annotations. However, since the atomic rules in the medium-grained grammar have detailed morpho-syntax specifications, we don't have to evaluate feature descriptions during CFG parsing.

4.2 Implementing the Evaluation Strategies

We have implemented the various evaluation strategies with additional house-keeping in the underlying parser. The parser used here is called the Typed Feature Structure Propagation Parser

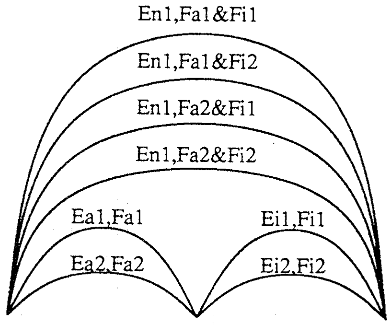


Fig. 2 Non-edge-sharing augmentation.

(TFSP Parser),[12] which is based on the active chart parsing algorithm[11] and typed feature structure unification.[2]

The basic active chart parsing algorithm consists of chart initialization and iterative rule invocation. A cycle of rule invocation consists of getting a new pending edge, adding it to the chart, combining active and inactive edges, and proposing new edges. The parser stops when it finds an inactive edge whose starting and ending vertices are the left-most and right-most vertex of the chart, respectively, and whose label is the start symbol of the grammar.

Augmenting a chart parser to handle feature descriptions is not as easy as you might think! There are at least two methods, which differ in computational properties.

The first method is straightforward augmentation, in which edges are augmented so that they also record the feature description associated with a constituent. We call this *non-edge-sharing augmentation.* In the context free case, two edges are equivalent if they span the same substring, try to identify the same category, and have exactly the same categorial requirements for further matching. However, if edges are augmented with feature descriptions, two edges are equivalent if they meet the above requirements and have the same feature descriptions. The polynomial bound on the cost for parsing context-free grammars comes from the fact that equivalence does not depend on the internal structure of the constituent. Since feature descriptions which come from different combinations of daughters may result in different feature descriptions for mothers, non-edge-sharing augmentation may require an exponential amount of computation, as illustrated in **Fig. 2**.

By using early unification (step-by-step strategy), we can prune the exponential number of edges that might have been constructed from edges with unsatisfiable feature descriptions. This might be the reason why early unification is used without further consideration in most systems. However, it must be noted that non-edge-sharing augmentation is still exponential in the worst case, even if we employ early unification.

The second method merges edges that span the same substring, try to identify the same category, and have exactly the same categorial requirements for further matching. This is the method we employed in our parser. We call this *edge-sharing augmentation*. In this implementation, the combinations of the feature description of the daughter edges are recorded as a list in the edge, as illustrated in **Fig. 3**. Each combination is implemented as a list of the two immediate daughter edges* and the resulting feature description. We call such a list an **edge-internal**.

Edge-sharing augmentation preserves the polynomial nature of CFG parsing in unification-based parsing. In an edge-sharing augmented chart parser, however, you must be careful when you add a new feature description to an existing edge. If there are edges that have

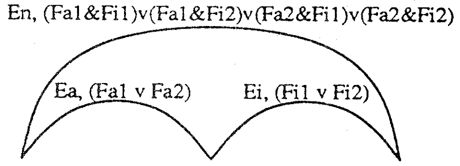


Fig. 3 Edge-sharing augmentation.

* More precisely, pointers to the **edge-internals** in the immediate daughter edges.

the edges as a daughter, you must also add the new feature description to its mother edges, considering the possible combinations of feature descriptions with its sister edges. This addition must be propagated recursively.* For this purpose, we have an upward link in each edge, which is a list of the edge itself, its sister edge, and its mother edge.[12]

In early unification, the feature descriptions are evaluated when the active edge(s) and the inactive edge(s) are to be combined. If there is more than one edge combination whose feature descriptions are satisfiable, a new edge is created and all the solutions are recorded in the list of **edge-internals**. In late unification, first, normal CFG parsing proceeds with the position for feature description in the **edge-internal** marked as "undefined." Next, if the suspending condition of chart parsing holds, that is, if the chart parser finds an inactive edge which spans the entire input string and whose label is the start symbol, the feature descriptions for the root category are recursively evaluated following the downward and upward links between edges. If the CFG parse has no feature descriptions that are satisfiable, the chart parsing process is resumed. The parser continues this generate-and-test procedure until it finds a CFG parse that has more than one satisfiable feature description.

Note that some derivations that terminate when feature descriptions are evaluated may not terminate if the feature descriptions are ignored. For example, in order to introduce slashed categories dynamically, it is possible to write a seemingly self-looping rule like (7), whose feature description specifies that an element in the subcat feature is moved to the slash feature.**

$$V \rightarrow (V) \qquad (7)$$

Ignoring feature descriptions in the rule may cause an infinite loop. Therefore, as a special-case precaution in late unification, we always evaluate feature descriptions as soon as we encounter rules which could cause looping.

It is worth mentioning that the combination of edge-sharing augmentation and late unification

* Hence, we call our parser the Typed Feature Structure Propagation Parser.

** 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.

may eliminate the exponential amount of computation that would have been required by the combination of non-edge-sharing augmentation and early unification. This is another advantage of 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 relative to rule granularity and interleaving strategy.

The experiment used 28 test sentences, examples are shown in Appendix B. They are selected from ATR's dialogue corpus whose task domain is "international conference registration." The average length of the test sentences is 18.1 characters, and their minimum and maximum lengths are 4 and 31 characters, respectively.

We tested the two Japanese grammars of different granularity with almost the same coverage described in Section 3. The *coarse-grained rules* consist of 22 generalized phrase structure rules with detailed feature descriptions in their annotations, while the *medium-grained rules* consist of 163 detailed phrase structure rules with less detailed feature descriptions.

We implemented two different interleaving strategies in the active chart parser. *Early unification* evaluates the feature descriptions at each rule application (the step-by-step strategy). *Late unification*, on the other hand, delays unification until a complete syntactic structure is found by using the atomic phrase structure rules (the pipeline strategy).

We also implemented two different graph unification algorithms, namely, non-destructive graph unification (NDGU) and quasi-destructive graph unification with structure-sharing (QDSS). NDGU[25] avoids irrelevant copying of DAGs by using the incremental copying technique. QDSS[23] is an improved version of NDGU, which can eliminate a lot of copies by delayed copying and structure sharing, as well as incremental copying. Both algorithms were extended to treat negation, loop, and type symbol subsumption relationships. In order to handle disjunction in feature descriptions, a disjunctive constraint solver using successive approximation[10] was implemented on top of a graph unifier (either NDGU or QDSS).

The active chart parser and the unifier were implemented in Common Lisp, and tested on a Sun Sparc Station 2, which is a 28.5-MIPS work station.

The average parsing time over the 28 test sentences is shown in **Table 4**. The results show that, with regard to average parsing time, it is better to use the early unification strategy for coarse-grained rules, and the late unification strategy for medium-grained rules. It also shows that, on average, the combination of medium-grained rules and late unification reduced the parsing time to 47% of the time required with coarse-grained rules and early unification, when QDSS is used for graph unification.

Note that the number of steps in CFG chart parsing increases 4.41 times, whereas the number of nodes created in graph unification and the number of calls to graph unifier (**unify-fs**) decreased by 68% (51046.6→16128.3) and 74% (1881.3→495.7), respectively, comparing the combination of coarse-grained rules and early unification with that of medium-grained rules and late unification. We can conclude that the improvement of parsing time is obtained by reducing the need for unification even though there is more CFG parsing.

The importance of selecting the best interleaving strategy with respect to the restrictive power of the atomic CFG phrase structure rules is clearly indicated in the number of calls and the success rate of **unify-desc**.* When we compare

Table 4 Rule granularity and unification interleaving strategy.

	time (sec)	chart steps	dag nodes	unify-fs		unify-desc	
				calls	success	calls	success
coarse.qdss.early	16.6 (1.00)	179.8 (1.00)	51046.6 (1.00)	1881.3	81.1%	103.1	65.3%
coarse.qdss.late	29.1 (1.75)	338.4 (1.88)	47791.8 (0.94)	1607.0	82.0%	75.6	73.0%
medium.qdss.early	18.0 (1.08)	638.5 (3.55)	48082.8 (0.94)	1628.8	82.6%	107.0	83.7%
medium.qdss.late	7.8 (0.47)	792.2 (4.41)	16128.3 (0.32)	495.7	83.2%	34.4	91.0%

the combination of coarse-grained rules and early unification with that of medium-grained rules and late unification, the number of calls to **unify-desc** is reduced from 103.1 to 34.4, and the success rate is increased from 65.3% to 91.0%. This means that few wasteful unifications were made in case of medium-grained rules and late-unification because the medium-grained rules are powerful enough to constrain parse trees by themselves. This decrease in unification cost results in an increase of total performance.**

6. Discussion

We have already discussed the reduction of unnecessary copying during parsing using late unification. The reduction gained using QDSS[23] and similar structure sharing is independent. Thus we can multiply the gain by combining the two techniques.

Table 5 shows the effect in reducing copies of the two methods, QDSS and late unification, individually and when combined. The same 28 test sentences were used in this experiment.*** The baseline parser uses coarse-grained rules using NDGU as a unifier and early unification as the interleaving strategy. When the unifier is changed from NDGU to QDSS, the parsing time is reduced to 33% of that of the original combination, and the number of nodes created is reduced to 11%, by the effect of delayed copying and structure sharing. Similarly, when the grammar granularity is changed from coarse to medium and the interleaving strategy is changed from early to late, the parsing time is reduced to 34%, and the number of nodes created is reduced to 25%. When all of these techniques are combined, the parsing time is reduced to 16%, and the number of nodes is reduced to 3% of the values of the original combination.

Table 5 Delaying copy within unifier and/or between parser and unifier.

	time (sec)	nodes
coarse.ndgu.early	46.0 (1.00)	465211.1 (1.00)
coarse.qdss.early	15.2 (0.33)	51046.6 (0.11)
medium.ndgu.late	15.8 (0.34)	114315.0 (0.25)
medium.qdss.late	7.2 (0.16)	16128.3 (0.03)

In order to more closely compare the performance of coarse-grained rules using early unification with that of medium-grained rules using late unification, we measured the parsing time with respect to input length, as shown in **Fig. 4**. The figure shows that the medium-grained rules using late unification are significantly more efficient than the coarse-grained rules using early unification. This tendency becomes clearer as the sentence length increases. It also shows that the parsing time of medium-grained rules using late unification increases slowly, i.e. polynomially (not exponential) in the input length. On the other hand, the parsing time of coarse-grained rules using early unification varies widely. This is because the parsing time with late unification is mainly dominated by the cost of atomic CFG parsing, whereas the parsing time with early unification is mainly dominated by the cost of unification.

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

The first is automatic transformation of phrase structure rules, converting 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

* **unify-desc** is a unifier of disjunctive feature descriptions in which the graph unifier **unify-fs** is called as a primitive operation. What the chart parser calls in combining the feature descriptions of an active edge and an inactive edge is **unify-desc**, so its number of calls and its success rate show the number of trial and the success rate of combining two edges in chart parsing.

** Maxwell and Kaplan[15] and the authors[17] independently obtained the same results that the pipeline strategy is significantly better than the step-by-step strategy under certain circumstances. Their "interleaved pruning" corresponds to our early unification under non-edge-sharing augmentation, and their "non-interleaved pruning" corresponds to our late unification under edge-sharing augmentation. Moreover, their discussion about "selective feature movement" is closely related to our discussion about rule granularity. They worked on LFG-based English parser, while we worked on HPSG-based Japanese parser. We believe that this coincidence demonstrates the necessity, the effectiveness, and the generality of the techniques presented in this paper.

*** The average parsing times shown in Table 5 are a little faster than those shown in Table 4 for the same test conditions. This is because we embedded a lot of metering routines in the experiment shown in Table 4.

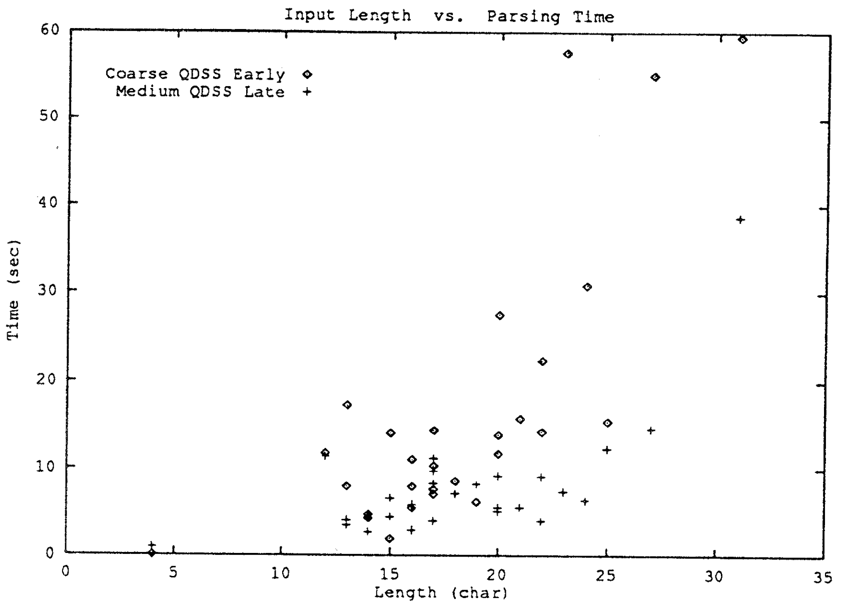


Fig. 4 Comparison of coarse.early.qdss and medium.late.qdss.

formal procedures. If the grammar compiler can perform this kind of transformation automatically, we can improve efficiency without losing grammar maintainability.

If the number of features and their values is finite, we can automatically move information in annotations toward phrase structure rules, since the domain of complex-valued nonterminals is finite. However, in general unification-based grammars, no automatic transformation procedure is known because the domain of nonterminals is infinite. Shieber[21] proposed a technique called restriction, to dynamically classify the infinite nonterminal domain into a finite set of equivalence classes. By using restriction, the parsing algorithm does not require a context-free backbone in unification-based formalisms. We think that it may be possible to automatically transform the granularity of phrase structure rules, using restriction or some similar kind of technique, although we assume that hand-crafted rules are more efficient than automatically transformed rules.

The second possibility is feature-sensitive lazy unification. In general, unification is used for both building up a structure using information-propagation and blocking rule application using constraint-checking. If the grammar compiler can extract features for constraint-checking, that is, features that are significant for guiding the parsing process, irrelevant subparses can be pruned efficiently by evaluating those features first. Unification is an associative and commutative operation, so the same results as with the feature-sensitive lazy unification are assured.

In fact, we can think of a variety of interleaving strategies other than the "early" and "late," described in the paper. For example, in order to prune irrelevant subparses during CFG parsing, it is possible to evaluate feature descriptions whenever some major constituents, such as noun phrases and verb phrases, are found. Moreover, if we only want to roughly check the validity of the mother edge in Fig. 3, we can stop evaluation when we find a satisfiable combination of feature descriptions associated with the daughter edges, and we don't have to evaluate the remaining combinations.

7. Conclusion

This paper has proposed two techniques for implementing an efficient unification-based parsing system, which, when combined, significantly improve the overall performance. The first technique is to make the granularity of the context-free phrase structure rules finer, until it becomes what we call medium-grained. This enables us to reduce the amount of unification for feature descriptions without intractably increasing the number of phrase structure rules. The second technique is late unification, in which the unification for feature descriptions is delayed until a complete CFG parse is found. This reduces the number of unnecessary copies of feature structures which are wasted on 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 interleaving unification (early/late). It has been proved that, on average, the combination of medium-grained rules and late unification is more than twice as fast than the combination of coarse-grained rules and early unification.

Acknowledgments We would like to thank Dr. Kurematsu, and all the members of ATR Interpreting Telephony Research Laboratories for their constant help and fruitful discussions. We also would like to thank the two anonymous referees, whose comments helped to make this a better paper.

References

1) Aho, A and Ullman, J.: *Principles of Compiler Design*, Addison-Wesley, Reading (1977).
2) Aït-Kaci, H.: An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *J. Theor. Comput. Sci.*, Vol. 45, pp. 293-351 (1986).
3) Earley, J.: An Efficient Context-Free Parsing Algorithm, *Comm. ACM,* Vol. 13, No. 2, pp. 94-102 (1970).
4) Eisele, A. and Dörre, J.: Unification of Disjunctive Feature Descriptions, *Proc. ACL-88*, pp. 286-294 (1988).
5) Emele, M.: Unification with Lazy Non-Redundant Copying, *Proc. ACL-91*, pp. 325-330 (1991).
6) Gazdar, G., Klein, E. and Sag, I.: *Generalized Phrase Structure Grammar*, Basil Blackwell, Oxford (1985).
7) Godden, K.: Lazy Unification, *Proc. ACL-90*, pp. 180-187 (1990).
8) Gunji, T.: *Japanese Phrase Structure Grammar—A Unification-Based Approach*, D. Reidel, Dordrecht (1987).
9) Kaplan, R. M. and Bresnan, J.: Lexical-Functional Grammar: A Formal System for Grammar Representation, in Bresnan, J. (ed.), *The Mental Representation of Grammatical Representations*, pp. 173-281, MIT Press, Cambridge (1982).
10) Kasper, R.: A Unification Method for Disjunctive Feature Descriptions, *Proc. ACL-87*, pp. 235-242 (1987).
11) Kay, M.: Algorithm Schemata and Data Structures in Syntactic Processing, Technical Report CSL-80-12, Xerox PARC (1980).
12) Kogure, K.: Parsing Japanese Spoken Sentences based on HPSG, *Proc. IWPT-89*, pp. 132-141 (1989).
13) Kogure, K.: Strategic Lazy Incremental Copy Graph Unification, *Proc. COLING-90*, pp. 223-228 (1990).
14) Maxwell, J. and Kaplan, R.: An Overview of Disjunctive Constraint Satisfaction, in Tomita, M. (ed.), *Current Issues in Parsing Technology*, Kluwer, Boston, pp. 172-190 (1991).
15) Maxwell, J. and Kaplan, R.: The Interface between Phrasal and Functional Constraints, *Computational Linguistics*, Vol. 19, No. 4, pp. 571-590 (1993).
16) Morimoto, T., Suzuki, S., Takezawa, T., Kikui, G., Nagata, M. and Tomokiyo, M.: A Spoken Language Translation System: SL-TRANS2, *Proc. COLING-92*, pp. 177-183 (1992).
17) Nagata, M.: An Empirical Study on Rule Granularity and Unification Interleaving toward an Efficient Unification-Based Parsing System, *Proc. COLING-92*, pp. 177-183 (1992).
18) Nagata, M. and Morimoto, T.: A Unification-Based Japanese Parser for Speech-to-Speech Translation, *IEICE Trans. Inf. & Syst.*, Vol. E76-D, No. 1, pp. 51-61 (1993).
19) Nakano, M.: Constraint Projection: An Efficient Treatment of Disjunctive Feature Descriptions, *Proc. ACL-91*, pp. 307-314 (1991).
20) Pollard, C. and Sag, I.: *An Information-Based Syntax and Semantics*, CSLI Lecture Notes No. 13, CSLI (1987).
21) Shieber, S. M.: Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms, *Proc. ACL-85*, pp. 145-152 (1985).
22) Takezawa, T., Kita, K., Hosaka, J. and Morimoto, T.: Linguistic Constraints for Continuous Speech Recognition in Goal-Directed Dialogue, *Proc. ICASSP-91*, pp. 801-804 (1991).
23) Tomabechi, H.: Quasi-Destructive Graph Unification with Structure-Sharing, *Proc.*

COLING-92, pp. 440-446 (1992).
24) Tomita, M.: *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*, Kluwer, Boston (1986).
25) Wroblewski, D.: Nondestructive Graph Unification, *Proc. AAAI-87*, pp. 582-587 (1987).

Appendix

A. Medium-grained Rules for Japanese Verb Phrases

```
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)
```

B. Examples of Test Sentences

会議に申し込みたいのですが

I'd like to apply for the conference.

どのような手続きをすればよろしいのでしょうか

What kind of procedure should I follow?

それでは登録用紙をお送りいたします

Then, I'll send you the registration form.

会議の参加料について教えていただきたいのですが

Please tell me about the attendance fee of the conference.

来月お申し込みになりますと四万円です

If you apply for it next month, it's forty thousand yen.

今回は割引を行なっておりません

We won't make a discount this time.

会議は八月二十二日から二十五日まで京都国際会議場で開催されます

The conference will be held at the Kyoto International

Conference Center from August 22nd to 25th.

失礼ですがお名前とご住所をお願いいたします

Excuse me, your name and your address, please.

住所は大阪市東区玉造二丁目二十七の七です

The address is Osaka Higashi ku Tamatsukuri two twenty seven - seven.

そうです

That is right.

登録料を払い戻していただけますか

Can you refund the registration fee?

九月二十七日以後の取り消しに対する払い戻しはできません

We can't make a refund for the cancellation after September 27th.

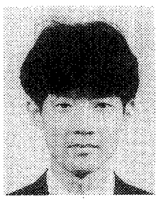
では誰かがわたしの代わりに参加することはできますか

Then, will it be possible for anybody to attend instead?

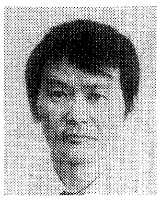
代理人が決まりましたらお知らせいたします

If the substitute is determined, we'll inform you about it.

(Received February 9, 1993)
(Accepted November 16, 1993)



Masaaki Nagata received the B.E. and M.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively. In 1987, he joined NTT Communications and Information Processing Laboratories. From 1989 to 1993, he worked at ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, and was engaged in the research and development of speech-to-speech translation systems. In 1993, he moved to NTT Network Information Systems Laboratories. His current research interests include speech recognition, natural language processing, and the integration of the two. He is a member of the IPSJ, the Japanese Society for Artificial Intelligence, and Association for Computational Linguistics.



Tsuyoshi Morimoto received the B.E. and M.E. degrees in Electronics Engineering from Kyushu University, Fukuoka, Japan, in 1968 and 1970, respectively. In 1970, he joined the Electrical Communication Laboratories of NTT. He was engaged in the research and development of operating system and database retrieval. In 1987, he moved to ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, and is currently with ATR Interpreting Telecommunications Research Laboratories. He is the Head of the Department-4. His research interests are the integration of speech recognition and language processing, and natural language understanding. He is a member of the IPSJ, and the Japanese Society for Artificial Intelligence.