# N A T U R A L  L A N G U A G E

## Parsing and Morphology

# Coordinated Morphological and Syntactic Analysis of Japanese Language

Tsunenori Mine, Rin-ichiro Taniguchi
and Makoto Amamiya

Department of Information Systems, Kyushu University
6-1, Kasuga-koen, Kasuga-shi, Fukuoka 816 JAPAN
email: mine@is.kyushu-u.ac.jp

## Abstract

A method for parallel morphological and syntactic analysis of Japanese language is proposed. Parallel syntactic analysis is based on an efficient parallel LR parsing algorithm for general context-free grammars. It handles syntactic features as constraints. Each syntactic feature is defined by a verbal subcategorization and attached to a special set of phrases called bunsetsu in Japanese. The bunsetsu is used as a processing unit for both analyses. All processes act asynchronously, and are coordinated on a P-RAM(Parallel Random Access Machine).

## 1 Introduction

In order to speed up natural language processing, efficient parallel processing methods must be developed. Recently, we have proposed an efficient parallel parsing algorithm for general context-free grammars which recognizes an input string of length *n* in *O(n)* time with $O(n^2)$ processors and memory spaces [MiTaAm90b ]. This algorithm is optimal in the sense that, in general, almost $0(n^3)$ steps are required to recognize a context-free language on a sequential machine [Ear70, Val75 ]. Our algo-] rithm is based on an LR parsing scheme [Knu65 ]. This scheme offers two advantages: high speed analysis due to the compiled LR parsing table and the additional capacity of left-to-right on-line parsing[Tom87 ]. However, pure context-free grammars are not enough for natural language processing. It is often desirable for each symbol in the grammar rule to have attributes[Tom87 ] and for each grammar rule to allow an unrestricted word order. Particularly, in order to process a Japanese sentence, a parser must be able to handle an unrestricted word order.

A morphological analysis, whose processes are themselves performed in parallel, must be performed in parallel with a syntactic analysis, and both analyses must be coordinated, so that the syntactic analysis may work up to capacity and they may perform together lexical disambiguations using bits of information provided by the syntactic analysis.

In this paper, a coordinated, parallel, morphological and syntactic analysis method is proposed. The parallel syntactic analysis performs a shift and a reducing action in parallel by using an LR transition diagram[MiTaAm90b ] which is derived from a pure context-free grammar and controls unrestricted word order by using syntactic features. The morphological analysis uses a finite state automaton called a *morphological network.* The morphological grammar and the syntactic grammar are integrated hierarchically. This integrated grammar is called a *two-level grammar.* By using this grammar, a terminal symbol of the syntactic grammar is used as a processing unit for both analyses. Requests for constructing a terminal symbol are passed from a syntactic process to a morphological process, and constructed candidates of terminal symbols are returned to the syntactic process. The terminal symbol stands for a special set of words called bunsetsu in Japanese, which can be regarded as a minimal semantic element in a Japanese sentence.

Section 2 describes the two-level grammar and its constituents: the syntactic grammar and the morphological grammar. Section 3 describes the parallel morphological analysis method, and section 4 describes the parallel syntactic analysis method.

## 2 Two-Level Grammar

### 2.1 Notations of Features

- BS: BS stands for bunsetsu which can be regarded as a minimal semantic element in a Japanese sentence. BS is denoted as follows:

$$\begin{aligned} \mathrm{BS} &:= \mathrm{SI\ JI} \\ \mathrm{SI} &:= \mathrm{N} \mid \mathrm{V} \mid \mathrm{AC} \\ \mathrm{JI} &:= (\mathrm{AX})^{*}\mathrm{P}^{*} \end{aligned}$$

N stands for a noun and nouns, V for a verb, verbs and an adjective, P for a postpositional particle, AC for an adverb and a conjunction, and AX for

an auxiliary verb. '|' means 'or', and '*' means more than 0 iterations. BS denotes relations of structural dependency: renyou 'dependent of V, rental 'dependent of N' and danshi 'governor of V'[Haga82].

- GR: GR stands for the grammatical relation of some of the postpositional phrases. The main particle of the phrases is called kakujoshi 'postpositions with a grammatical function'.
- SC: SC stands for subcategorizations. Each SC features can take a set of GRs as its value. The value is a list of GKs for which the verbal particle subcategorizes.
- PS: PS stands for a part of speech.

## 2.2 Significance of the Two-Level Grammar

The construction of our two-level grammar is shown in figure 1.

A two-level grammar is a grammar whose constituents (a morphological grammar and a syntactic grammar) are integrated in hierarchical structures. The syntactic grammar consists of context-free rules and the morphological grammar consists of regular rules. A terminal symbol of the syntactic grammar stands for BS and it is constructed by using the morphological grammar. Since a morphological analysis can be performed by using a finite state automaton[Ama78 ], without stacks, this hierarchical construction is significant from a computational viewpoint. Also, from a semantic viewpoint, this construction is important to describe and apply local constraints for each analysis.

## 2.3 Syntactic Grammar

The syntactic grammar consists of context-free rules. A terminal symbol in the grammar denotes a BS. The following is a subset of Japanese grammar:

| | | | |
|---|---|---|---|
| 1: S | $\rightarrow \mathbf{v}_d$ | 6: $\mathbf{N}_t$ | $\rightarrow \mathbf{n}_t$ |
| 2: $\mathbf{V}_d$ | $\rightarrow \mathbf{NV}_y\ \mathbf{V}_d$ | 7: $\mathbf{N}_y$ | $\rightarrow \mathbf{n}_y$ |
| 3: $\mathbf{V}_d$ | $\rightarrow \mathbf{N}_y\ \mathbf{V}_d$ | 8: $\mathbf{NV}_y$ | $\rightarrow \mathbf{nv}_y$ |
| 4: $\mathbf{N}_y$ | $\rightarrow \mathbf{N}_t\ \mathbf{N}_y$ | 9: $\mathbf{V}_d$ | $\rightarrow \mathbf{v}_d$ |
| 5: $\mathbf{NV}_y$ | $\rightarrow \mathbf{N}_y\ \mathbf{NV}_y$ | | |

The suffixes $y$, $t$ and $d$ are connective forms between N and V; $y$ stands for renyou, $t$ for rentai, $d$ for danshi. $\mathbf{v}_x$, $\mathbf{n}_x$ and $\mathbf{nv}_x$ ($x$ stands for $y$, $t$ or $d$) mean BS. Figure 2 shows the LR otransition diagram derived from this syntactic grammar, si in the goto field of this diagram means 'shift and goto state $i$', and ri means 'reduce and go to the state $i$' The rule field means a rule number of the syntactic grammar. The info field means information used by a shift or a reducing action.

Our LR transition diagram is similar to the LR parsing table [AhU172), but with two additional advantages:

| state | goto | info | rule | state | goto | info | rule |
|---|---|---|---|---|---|---|---|
| 0 | s3 | $\mathbf{v}_d$ | | 7 | r5 | (0,4,5,12) | 7 |
| | s6 | $\mathbf{nv}_y$ | | | r10 | (8) | 7 |
| | s7 | $\mathbf{n}_y$ | | 8 | s7 | $\mathbf{n}_y$ | |
| | s9 | $\mathbf{n}_t$ | | | s9 | $\mathbf{n}_t$ | |
| 1 | acc | $ | | 9 | r8 | (0,4,5,8,12) | 6 |
| 2 | r1 | (0) | 1 | 10 | r5 | (0,4,5,12) | 4 |
| 3 | r11 | (4,12) | 9 | | r10 | (8) | 4 |
| | r13 | (5) | 9 | 11 | r11 | (4,12) | 3 |
| | r2 | (0) | 9 | | r13 | (5) | 3 |
| 4 | s3 | $\mathbf{v}_d$ | | | r2 | (0) | 3 |
| | s6 | $\mathbf{nv}_y$ | | 12 | r4 | (0,4,12) | 5 |
| | s7 | $\mathbf{n}_y$ | | | r12 | (5) | 5 |
| | s9 | $\mathbf{n}_t$ | | | s3 | $\mathbf{v}_d$ | |
| 5 | s3 | $\mathbf{v}_d$ | | | s6 | $\mathbf{nv}_y$ | |
| | s6 | $\mathbf{nv}_y$ | | | s7 | $\mathbf{n}_y$ | |
| | s7 | $\mathbf{n}_y$ | | | s9 | $\mathbf{n}_t$ | |
| | s9 | $\mathbf{n}_t$ | | 13 | r2 | (0) | 2 |
| 6 | r4 | (0,4,12) | 8 | | r11 | (4,12) | 2 |
| | r12 | (5) | 8 | | r13 | (5) | 2 |

Figure 2: The LR transition diagram derived from a Japanese grammar.

it has a natural form as an automaton, and it can perform both a reducing action and a goto action as one action[MiTaAm90b ].

## 2.4 Morphological Grammar

The morphological grammar consists of a set of regular rules producing BS. The grammar is transformed into a finite state automaton called a *morphological network.* A state of a morphological network denotes either a part of speech or a word itself. A link stands for a connective relation between words. The morphological network consists of two networks: a nominal network, which constructs $\mathbf{n}_x$ and a verbal network, which constructs $\mathbf{v}_x$ and $\mathbf{nv}_x$. The following is an example of production rules for $\mathbf{n}_x$, $\mathbf{v}_x$ and $\mathbf{nv}_x$:

| BS | rule | |
|---|---|---|
| $\mathbf{n}_x$ | $N_y \rightarrow N\ p_{case1}$ | $N_y \rightarrow N\ p_{case1}\ p_{add1}$ |
| | $N_y \rightarrow N\ p_{pre}$ | $N_t \rightarrow N\ p_{case2}$ |
| | $N_t \rightarrow N\ p_{add1}$ | $N_t \rightarrow N\ p_{add2}$ |
| | $N_t \rightarrow N\ n$ | $N \rightarrow n$ |
| $\mathbf{v}_x$ | $V_d \rightarrow v_d$ | $V_y \rightarrow v_y$ |
| $\mathbf{nv}_x$ | $NV_y \rightarrow v_t\ p_{case2}\ p_{case1}$ | $NV_y \rightarrow v_d\ p_{conj}$ |

( $p_{case1}$: **ga, wo, ni, de** ... )
( $p_{add1}$: **no, dake, bakari** ... )
( $p_{pre}$: **ha, mo, sae, koso** ...)
( $p_{case2}$: **to, ni, ya, ka, no** ... )
( $p_{add2}$: **dano, yara, nari** ... )
( $p_{conj}$: **ga, kara, to** ... )

n and $v_x$ stands for a noun and a verb, respectively. Both $p_{case1}$ and $p_{case2}$ stands for a postpositional particle with a grammatical function. A $p_{case1}$ is a dependent of V. A $p_{case2}$ is a dependent of N.

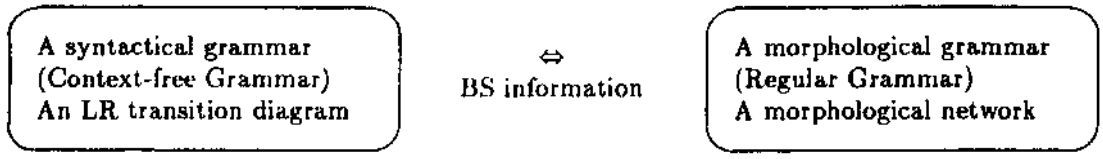


Figure 1: The construction of a two-level grammar

## 3 Parallel Morphological Analysis

The morphological analysis corresponds to a shift action of a syntactic analysis. In the morphological analysis, a process is created by a construction-request for BS issued from a parsing process. The request has the form $< N, i >$, where TV is a state name of an LR transition diagram, and *i* is a pointer to an input symbol called an *input pointer* for short. Extracting BS is performed simply by a process moving from the initial state to a final state in a morphological network. Since a morphological network is, in general, a *non-deterministic* finite state automaton, when a morphological process arrives at a final state, but it can still go on to to other final states, then, it creates a new process that resumes the syntactic analysis and returns BS information to the parsing process which had requested the production of the BS. So, the morphological process proceeds from one final state to another. BS information has the form: $< N, PS, word, gr, sem, i, j >$, where *word* is a BS extracted from a given input string, *gr* is a GR. sera is an item of semantic information. Both *i* and *j* are positions of the *word* in the input string. In order to avoid duplicated analysis, each word, which is extracted during the morphological analysis, is stored in a table, and its morphological information is accessed directly by using an input pointer.

The coordination mechanism for morphological and syntactic analyses is shown in figure 3. Both processes with $< N_3, 4 >$ and $< N_2, 4 >$ access 4, then a morphological process is created and the morphological analysis begins, because the BS information has not been held as an entry, namely, the tag of the table is off.

## 4 Parallel Syntactic Analysis

### 4.1 Parallel Parsing Model

Our parallel parsing model is called a DAG(Directed Acyclic Graph) array parsing model[MiTaAm90b ]. This model has a CREW (Concurrent-Read Exclusive-Write) type three dimensional shared memory, called a DAG memory. A cell of the DAG memory denotes an element of *DAG* called *a DAG node.* The DAG node has the form:$< N, PS, i, j >$. Each cell of a DAG memory can be directly accessed by a DAG node name used as an index. A processor is assigned to each DAG node. All processors act asynchronously. The DAG array parsing model is shown in figure 4.

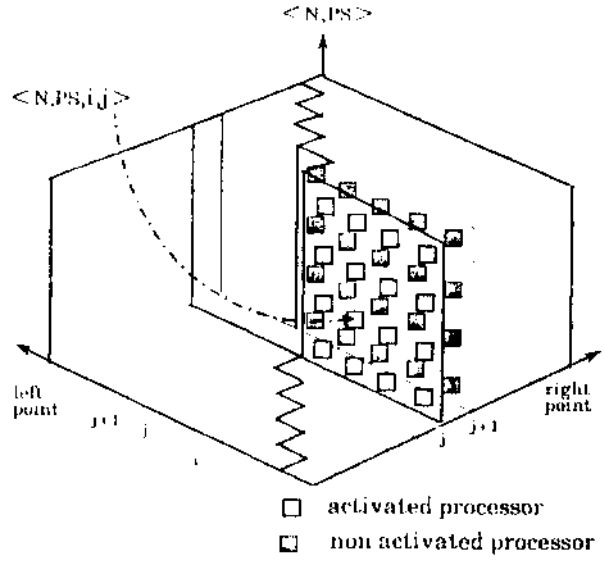


Figure 4: DAG array parsing model

### 4.2 Parallel Parsing for Context-free Grammars

Our parallel parsing method[MiTaAm90b ] is called a generalized parallel LR parsing scheme. It extends an LR parsing method for general context-free grammars by using a DAG which allows multiple stack entries instead of a conventional stack [MiTaAm89, MiTaAm90b ]. Similar parallel parsing schemes have been proposed by Tomita[Tom87 ], Yasutome[YaAo88 ], and Numazaki[NuTaTa89, NuTa90 ].

In our parsing method, parallel shift-reducing actions are controlled by means of an LR transition diagram [MiTaAm90b ]. A DAG node is produced by a shift or a reducing action, and then the processor, which is assigned to a newly produced DAG node, is activated. The DAG node is held as an entry in a DAG memory cell to avoid producing the same DAG nodes. If there are several conflicts between shift and reducing actions at any state, all those actions are performed in parallel, because one different processor is activated for each action. Thus, all actions which can be performed at any state, do not have to be synchronized with an input symbol. In order to inhibit the exponential increase of activated processors, and to avoid redundant actions, we use the following strategy:

- Only processors assigned to a DAG node, which has not been held in an DAG memory entry, are activated.

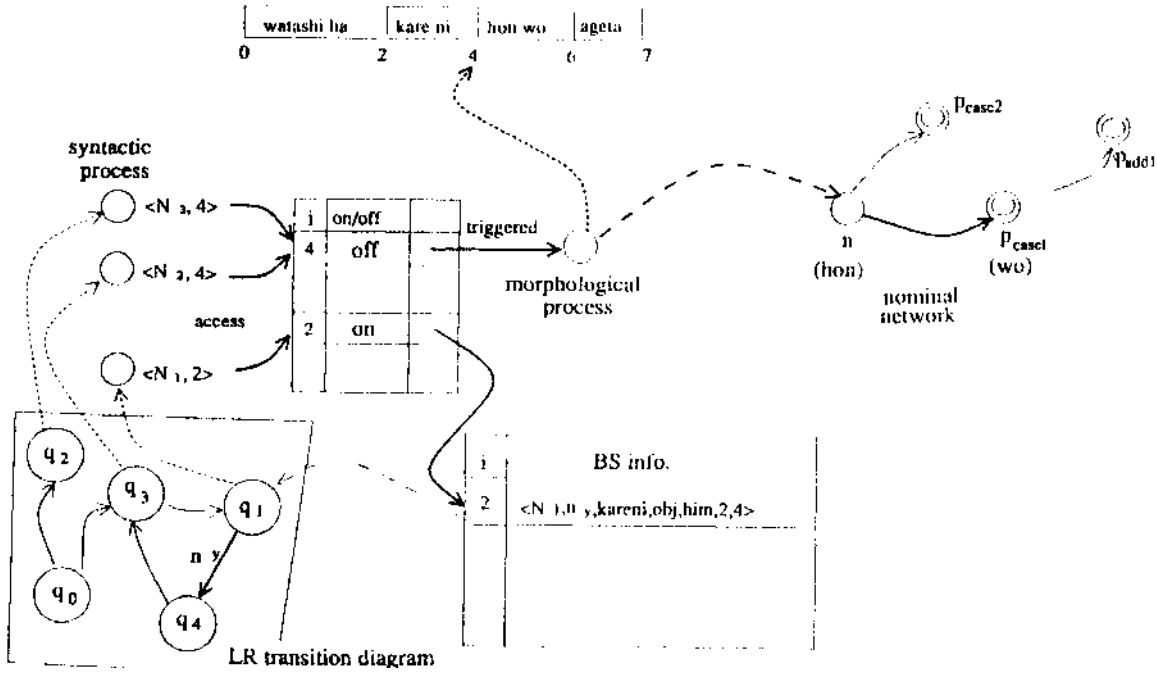


**Figure 3: Coordination mechanism of both morphological and syntactic analyses**

- **The results of shift/reducing actions performed by any processors are stored, so they can be shared by processors about to perform those actions.**

According to this strategy, first, all parsing trees with the same top vertex are packed, and their vertices are unified into one vertex. This strategy is controlled by checking whether a DAG node has been held as an entry in the DAG memory or not. Then, only one symbol for each grammar rule is assigned to each reducing action, which is therefore called a *one-symbol reduction.* The one symbol reduction consists of two actions: a *reduce 1* and a *reduce2* action. A reducel action is a one-symbol reduction for the rightmost symbol in a grammar rule, and a reduce2 action is for the other symbols, i.e., for a reduction by $A \rightarrow B\ C$, a reducel action performs a one-symbol reduction for C, making a rule-with-marker $[A \rightarrow B \cdot C]$. A reduce2 action performs for *B* and completes the reduction. The result of each one-symbol reduction is called *reduced information* which has the form $< [A \rightarrow \alpha \cdot \beta], i, N >$. $[A \rightarrow \alpha \cdot \beta]$ stands for reductions of $\beta$ from $[A \rightarrow \alpha\beta \cdot]$. *i* is an input pointer. TV is a state where the reduction by $A \rightarrow \alpha\beta$ was executed. A table in which reduced information is stored is called a *reducing table.* Each consists of 2 tables: a reference table and an entry table, which are both CREW type shared memories. A reference table has a field, which is used to check whether reduced information has been held as an entry or not. An entry table has a field used to store reduced information, which is written in the field from top to bottom. A reducing table is assigned to each DAG node produced by a shift action. Reduced information is passed through a reducing table between DAG nodes. Reduced information is then processed in pipeline-like processing through reducing tables. Our parallel parsing method is described as follows:

**[Parallel Parsing Method][1]**

1. **Start: A processor assigned to the DAG node $< N_0, nil, 0, 0 >$ starts a shift action.**
2. **Activate a processor: If a processor produces, in a shift or a reducing (reducel/reduce2) action, a DAG node dn, for which no entry exists, then the processor makes an entry *dn* and activates a new processor assigned to *dn.***
3. **Execute a shift/reducel action: a newly activated processor performs shift and reducel actions.**
4. **Execute a reduce2 action: After performing a shift action, a processor *P* executes a reduce2 action. If there is no reduced information in a field accessed by P, *P* waits until some reduced information is put to an entry. Then *P* reads it and performs a further one-symbol reduction according to it. *P* also activates a new processor *P',* which is newly reassigned to the same DAG node previously assigned to *P* and reads the next reduced information[2].**

[1] For brevity's sake, the linkage of a DAG node to reduced information, which is actually required, is not described. This linkage is performed by reducel/reduce2 actions.

[2] This newly activated processor is necessary to link a DAG node to reduced information.

5. Accept/Reject: If the processor assigned to $< N_0, S, 0, n >$($N_0$ and S are a start state and a start symbol, respectively.) is activated by a reducel or reduce2 action, and this processor reads the terminal symbol '$'(the end mark), by a shift action, then the input string is accepted. If there are no processors performing a shift and a reducel action, the input string is rejected.

### 4.3 Handling Syntactic Features

In order to extend the syntactic analysis method of 4.2 so that it can handle syntactic features, reduced information is improved as follows:

[Definition 1] reduced information: Reduced information is defined as $< [A \rightarrow \alpha \cdot \beta], sc, i, N >$, where *sc* means a list of SC features which has the form:({ sbj, obj1, ... }).

The reducing table is also extended for the newly defined reduced information. Table 1 shows the extended reducing table. By handling these syntactic features, a reduce2 action follows the constraints:

Constraints for a reduce2 action

- Inhibition of duplicated reference: In performing reduce2 actions, if an element of SC in reduced information is in agreement with the GR component of a DAG node, the element is marked with V. If, during a reduce2 action, a GR component, which matches a marked element is found, the reduce2 action aborts.
- Inhibition of wrong type reference: If, during a reduce2 action, a GR component does not match any element of SC in reduced information, the reduce2 action aborts.

As mentioned above, the syntactic analysis is allowed to handle an unrestricted word order by using SC features as a list, since GR is not attached to PS. This method is also adopted by JPSG[Gun87 ].

In figure 5, the parsing of Japanese sentence Watashi ha Kare ni Australia ni Iku to Itta(I told him that I would go to Australia.) is shown as an example. The form of the DAG node used in this example is $< N, PS, fs >$. The *fs* has the form [GR][{SC}], where "[ ]"means optional. No marked SC feature means a gap(missing constituent). In the figure, sbj which is a SC feature of the verb Iku, stands for a gap. The reduction by $V_d \rightarrow N_y\ V_d$ with both $< 5, N_y, obj2 >$ and $< 5, V)d, \{sbj, obj1, obj3*, \{sbj, obj2\}\} >$ failed because obj2 is not in agreement with any subcategorization elements in the list{sbj, obj1, obj3*}.

Reference table $Rt_{dn}$; $(Rt_{dn}.F[ri])$

| | *ri* | *boole* |
|---|---|---|
| $F[ri]$ | $< [A \rightarrow \alpha \cdot \beta], sc_1, i, N_j >$ | *on* |
| | $< [A \rightarrow \alpha \cdot \beta], sc_2, i+1, N_j >$ | *off* |
| | ... | *off* |
| | $< [A \rightarrow \alpha \cdot \beta], sc_3, m, N_j >$ | *off* |
| | ... | *off* |

Entry table $Et_{dn}$;$(Et_{dn}.F[i])$

| | *i* | reduced information |
|---|---|---|
| $F[i]$ | 0 | $< [A \rightarrow \alpha \cdot \beta], sc_1, i, N_j >$ |
| | 1 | $< [A \rightarrow \alpha \cdot \gamma], sc_2, i, N_k >$ |
| | | ... |
| | k | $< [A \rightarrow \alpha \cdot \delta], sc_4, m, N_h >$ |
| | $k+1$ | *nil* |
| | | ... |
| | $m$ | *nil* |

Table 1: A reducing table assigned to a DAG node *dn*

## 5 Conclusion

We proposed a coordinated parallel processing method for morphological and syntactic analyses of Japanese language. The proposed parallel parser is extended from the previous parser[MiTaAm90b ] so that it can handle syntactic features. From the computational viewpoint, it is significant to use a two-level grammar with a morphological and a syntactic grammar integrated hierarchically, using bunsetsu as a processing unit for both analyses. We are now implementing this method on Sequent Symmetry S-2000, a 20-CPU multi-processor system.

It must be noted that this method is not complete in the sense that it does not handle semantic features. This method, however, can be extended to include a semantic processing component, which performs, in coordination with morphological and syntactic analyses components, parallel semantic and pragmatic analyses on a semantic network, based on the situation semantics theory.

## References


[AhU172] A.V.Aho and J.D.Ulmann: The Theory of Parsing, Translation and Compiling. Volume 1, Parsing, *Prentice-Hall,* 1972.

[Ama78] M. Amamiya: *Zukei wo Taishou to shita Nihongo niyoru Shitsumon Ohtou Kei no Kcnkyuu* (Question Answering System by Japanese Language on Figure Manipulations). Ph.D. Thesis, Kyushu University, 1978.(in Japanese)

[Ear70] J. Earley: An efficient context-free parsing algorithm. *Comm.ACM,* 13(2):95-10, 1970.

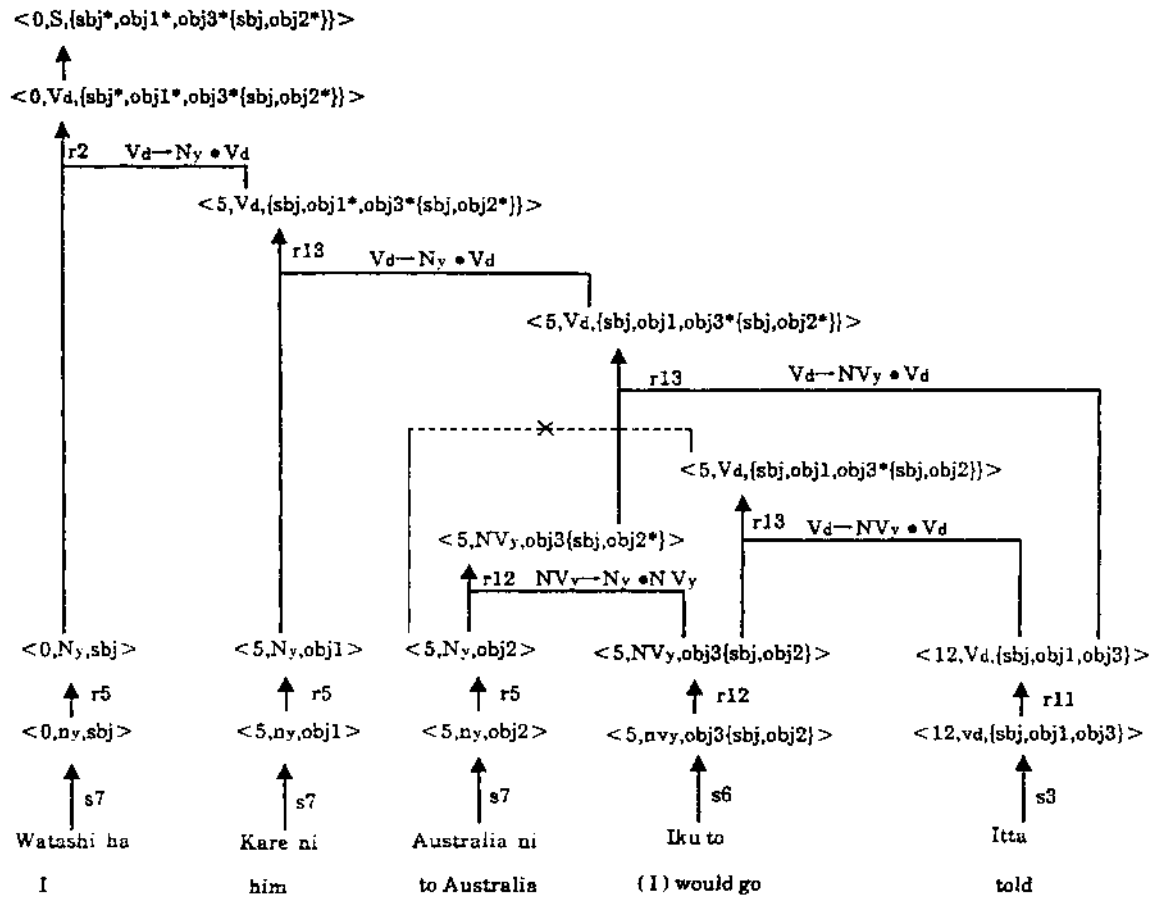


**Figure 5: An example of Japanese sentence parsing**


[Gun87] T. Gunji: Japanese Phrase Structure Grammar. D.Reidel, Dordrecht, 1987.

[Knu65] D.E.Knuth: On the translation of languages from left to right. *Information and Control,* 18(6):607-639, 1965.

[MiTaAm89] T. Mine, It. Taniguchi and M. Amamiya: A Parallel Parsing for Context-Free Grammars. *IPS of japan, NL* , 1-8, 6 1989. (in Japanese)

[MiTaAm90a] T. Mine: An Efficient Parallel Parsing for General Context-free Grammars. *Proceedings of 41th Conference of IPS J, 3-( 121-122), 1990.(in* Japanese)

[MiTaAm90b] T. Mine, R. Taniguchi and M. Amamiya: An Efficient Parallel Parsing Algorithm for Context-Free Grammars. *Proceedings of Pacific Rim International Conference on Artificial Intelligence '90,* 239-244, 1990.

[NuTaTa89] H. Numazaki and M. Tamura and H. Tanaka: An implementation of Generalized LR-Parsing Algorithm based on Parallel Logic Programming Language. *Natural Language Processing,* 74(5) 9 1989. (in Japanese)

[NuTa90] H.Numazaki and H.Tanaka: An Efficient Parallel Generalized LR Parsing based on Logic Programming. *Proceedings of The Logic Programming Conference '90,* 191-198 1990. (in Japanese)

[Haga82] Y. Haga: *Nikon Bunpou Kyousitsu(*Japanese Grammar Class), *Kyouiku Shuppan,* 1982. (in Japanese)

[Tom87] M.Tomita: An efficient augmented-context-free parsing algorithm. *Computational Linguistic,* 13(1-2), 1-6 1987.

[Val75] L.G.Valiant: General Context-Free Recognition in Less than Cubic Time. *Journal of Computer and System Sciences,* 10:308-315,1975

[YaAo88] S. Yasutome, and J. Aoe: Parallel Processing of Ambiguous Syntax Analysis for Natural Language. *IPS of japan, SIG-PL* 19-12 1988. (in Japanese)