Pergamon

Information Processing & Management Vol. 34, No. 6, pp. 761–773, 1998

Printed in Great Britain
0306-4573/98/$19.00

PII: S0306-4573(98)00036-3

A FAST RETRIEVING ALGORITHM OF HIERARCHICAL RELATIONSHIPS USING TRIE STRUCTURES

MASAFUMI KOYAMA, KAZUHIRO MORITA, MASAO FUKETA and JUN-ICHI AOE*

Department of Information Science and Intelligent Systems, The University of Tokushima, 2-1 Minami-Josanjima-Cho, Tokushima-Shi 770-8506, Japan

(Received 17 December 1997; accepted 1 September 1998)

Abstract—Case structures are useful for natural language systems, such as word selection of machine translation systems, query understanding of natural language interfaces, meaning disambiguation of sentences and context analyses and so on. The case slot is generally constrained by hierarchical concepts because they are simple knowledge representations. With growing hierarchical structures, they are deeper and the number of concepts to be corresponded to one word increases. From these reasons, it takes a lot of cost to determine whether a concept for a given word is a sub-concept for concepting the case slot or not. This paper presents a faster method to determine the hierarchical relationships by using trie structures. The worst-case time complexity of determining relationships by the presented method could be remarkably improved for the one of linear (or sequential) searching, which depends on the number of concepts in the slot. From the simulation result, it is shown that the presented algorithm is 6 to 30 times faster than linear searching, while keeping the smaller size of tries. © 1998 Elsevier Science Ltd. All rights reserved

1. INTRODUCTION

Hierarchical classification (McArthur, 1981; Kirkpatrick, 1987) is a simple and useful representation in many applications, such as selection of word translation in machine translation systems (Takamatsu & Nishida, 1981; Nakaiwa & Ikehara, 1993; Nagao *et al.*, 1996), knowledge representation of natural language analysis (Shimazu, Naitoh, & Nomura, 1986; Okumura & Tanaka, 1989; Hakomori, Sagawa, Ohnishi, & Sugie, 1992) and disambiguiation of meaning (Makino & Kizawa, 1981; Miyazaki & Ooyama, 1981; Ohshima, Abe, Yuura, & Takeichi, 1986) and so on (Oishi & Matsumoto, 1995; Nagao, 1996). In these applications, some of the constraints for case slots in case structures (Fillmore, 1968) are used and the basic operations depend on the determination of hierarchical relationships in terms of concepts.

This paper presents a fast method for determining a hierarchical relationship by introducing trie structures instead of a linear storage of a concept code. The worst-case time complexity of determining the relationships by the trie structures becomes constant and the presented method remarkably improves the time complexity of the linear storage, depending on the number of concepts in the slot.

Section 2 describes the outline of case structures and constraints in their case slots. A fast retrieval algorithm of determining hierarchical concepts is presented in Section 3. Section 4 proposes a method of dividing a set of verbs without degrading the compactness of resulting tries. The data structure, called a double-array, for the

*Corresponding author. Tel.: +81-886-56-7486; fax: +81-886-55-4424; e-mail: aoe@is.tokushima-u.ac.jp.

presented method is also introduced in Section 5. From theoretical and experimental observations, the presented method is evaluated. Finally, Section 6 concludes this paper and describes some of remaining problems.

2. CASE STRUCTURES AND HIERARCHICAL RELATIONSHIPS

2.1. Case structures

Case structures are a simple knowledge representation for natural language understanding and they have semantic constraints, or roles, for a noun phrase in the slot for a verb. Generally, the semantic constraints can be determined by hierarchical relationships between concepts of nouns and the input concepts.

Let C be a concept of a noun phrase and let A be a case attribute, such as 'agent(AGE)', 'object(OBJ)' and so on (Fillmore, 1968; Nagao, 1996). The case structure for verb V is denoted by a set CONCEPT_SET(V), which includes a pair (C, A).

Consider an example of case structures for verbs 'take', 'breed', 'fit' and 'meet', where INS stands for a case attribute 'instrument', SOU for 'source' and LOC for 'location'. Note that 'take' means 'buy' in this example. Concept names are denoted by the notation of < >.

CONCEPT_SET('take') = {(< human >, AGE), (< organization >, AGE), (< concrete object >, OBJ), (< shop >, LOC), (< place name >, LOC), (< human >, SOU), (< organization >, SOU)},

CONCEPT_SET('breed') = {(< human >, AGE), (< organization >, AGE), (< animal >, OBJ), (< breeding instrument >, INS), (< building >, LOC), (< place name >, LOC)},

CONCEPT_SET('fit') = {(< clothing >, AGE), (< color vision >, AGE), (< human >, OBJ), (< organization >, OBJ), (< clothing >, OBJ)},

CONCEPT_SET('meet') = {(< human >, AGE), (< organization >, AGE), (< human >, OBJ), (< organization >, OBJ), (< building >, LOC), (< place name >, LOC)}.

Concept X on hierarchical structures can be denoted by the concept code STRING(X), generally being a string of values (Aoe, 1990). Figure 1 shows an example of hierarchical structures of concepts with value strings for the above CONCEPT_SET. The hierarchical relationship between concepts X and Y can be determined by the prefix comparison of STRING(X) with STRING(Y). For example, STRING(< breeding instrument >) = '13111' is the prefix of STRING(< fish tank >) = '131111'.

For pair (X, K) of concept X and attribute K in a given sentence, if CONCEPT_SET(V) includes the pair (C, K) such that C is the superconcept of X, then we can say that the pair (X, K) successfully matches for CONCEPT_SET(V). Consider an input sentence 'A doctor breeds tropical fishes in the fish tank'. It is clear that pairs (< doctor >, AGE), (< tropical fish >, OBJ) and (< fish tank >, INS) obtained from the sentence successfully match for (< human >, AGE), (< animal >, OBJ), (< breeding instrument >, INS) in CONCEPT_SET('breed'), respectively. Therefore, the input sentence satisfies the constraint of the case structure. However, 'A doctor meets tropical fishes in the fish tank', does not satisfy the constraint of the case structure because (< tropical fish >, OBJ) could not match for (< human >, OBJ) in CONCEPT_SET('meet').

A Japanese verb [AU] has two meanings of English verb 'fit' and 'meet', i.e. [AU] is a homophones (Mizutani *et al.*, 1983; Ohno & Hamanishi, 1981; Koizumi, Funaki, Honda, Nitta, & Tsukamoto, 1989). By using the case matching of CONCEPT_SET('meet') and CONCEPT_SET('fit'), verb [AU] of Japanese sentence '[DOKUTA-HA(a doctor)] [AU] [KANGOFU-NI(a nurse')] can be understood as

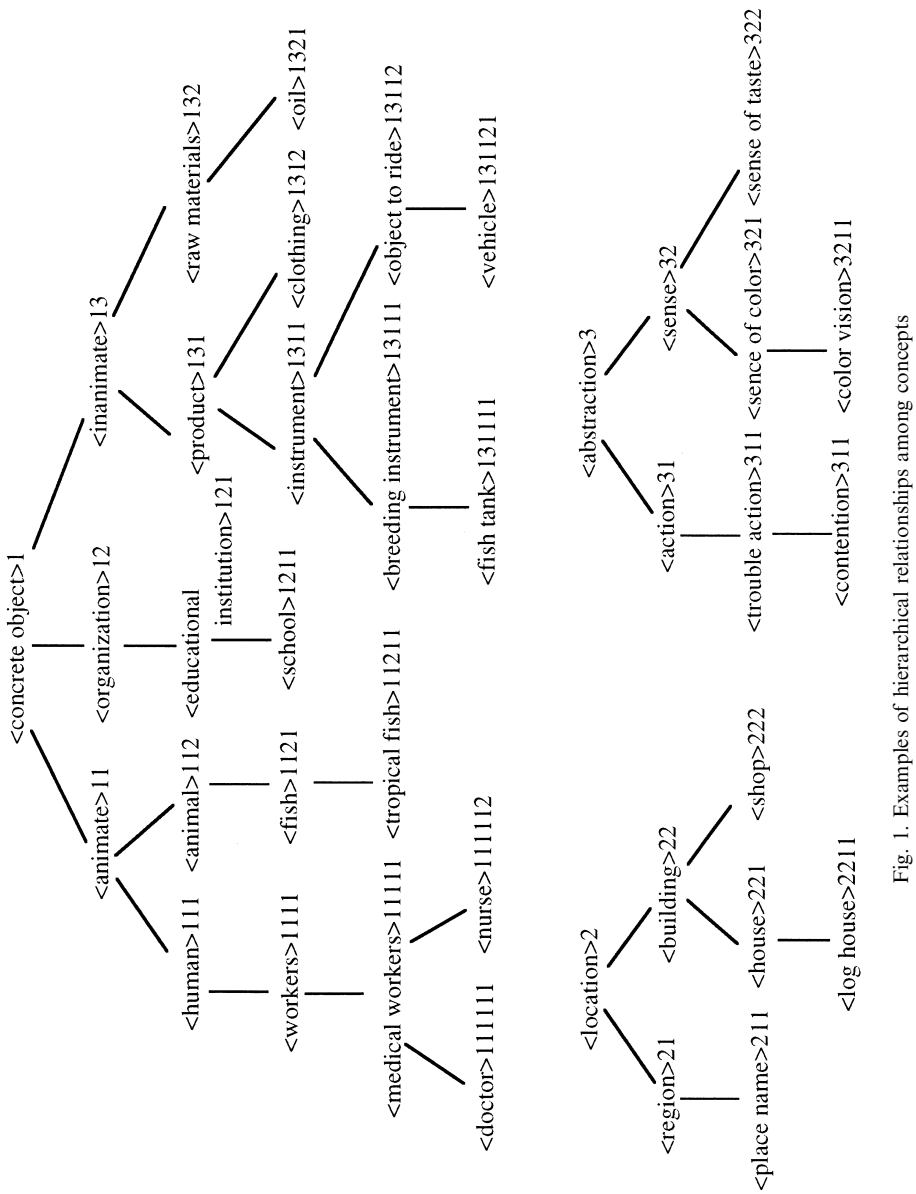


Fig. 1. Examples of hierarchical relationships among concepts

'meet' and [AU] of [AO-GA('blue')] [AU] [FUKU-NI('the clothing')] can be done as 'fit'.

For word selection of the English-to-Japanese machine translation (Takamatsu & Nishida, 1981; Nagao, Tsujii, & Nakamura, 1986; Nakaiwa & Ikehara, 1993), consider 'A doctor takes tropical fishes in a pet shop'. It is clear that (<doctor>, AGE), (<tropical fish>, OBJ) and (<pet shop>, LOC) obtained from the sentence with successful match for (<human>, AGE), (<concrete object>, OBJ), (<shop>, LOC) in CONCEPT_SET('take'), respectively. The meaning of 'take' can be understood as 'buy' and the corresponding target word [KAU] in Japanese could be selected.

2.2. Determination of hierarchical relationships

Let V be the input verb, let X be the input concept and let K be the input attribute. The function GET_SUPERCONCEPT of retrieving superconcept C of X can be shown as follows:

```
function GET_SUPERCONCEPT(V, X, K);
begin
    Retrieve CONCEPT_SET(V) for V;
    for each concept C such that A = K for (C, A) in CONCEPT_SET(V) do
            if C is the superconcept of X then append C into TEMP;
    return TEMP;
end;
```

For an algorithm of retrieving CONCEPT_SET(V) from key V, it is easy to implement with one of key search techniques such as hashing, binary tree searching and so on (Knuth, 1973; Standish, 1980). Problems arise in superconcept checking when a linear storage approach is used, because the worst-case time complexity becomes $O(kn)$ for the number n of concepts and for the length k of STRING(X). This complexity makes the whole performance of case analysis inefficient for the following reasons:

(R1) GET_SUPERCONCEPT operation is increased because multiple concepts correspond to a noun.

(R2) Since combinations of nouns and case structures grows in the complex sentence including one more verbs, GET_SUPERCONCEPT operation is also increased.

(R3) The length of codes grows by a larger hierarchical structure for a more strict semantic constraint.

A method of solving these problems will be proposed in Section 3.

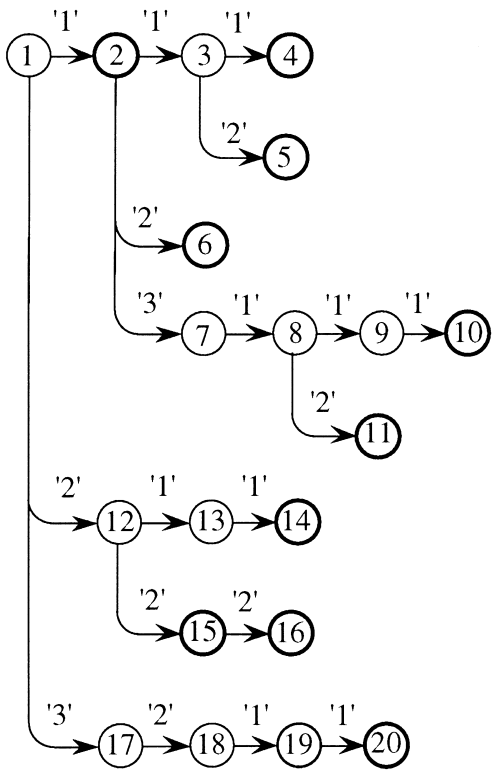
3. DETERMINATION OF HIERARCHICAL RELATIONSHIPS

3.1. A trie representation of case structures

A trie (Fredkin, 1960; Knuth, 1973; Standish, 1980; Aoe, 1991; Dundas, 1991) is a tree structure which merges common prefixes of strings. It is useful for a retrieval method of natural language dictionaries. In the trie, if an arc label a is defined from node s to node t, it indicates that $g(s, a) = t$. Otherwise, it is denoted by $g(s, a) =$ fail.

Let VERB_SET be a set of verbs V_i, $1 \leqslant i \leqslant h$. In this paper, a trie TRIE(VERB_SET) is constructed from concept codes in CONCEPT_SET(V_i) for V_i in VERB_SET. For TRIE(VERB_SET), ATTR_SET(m) is defined as follows:

Let y be a sequence of arc labels from the initial node 1 to node m. ATTR_SET(m) includes an element consisting of a pair (K, V_i) for K and for V_i of CONCEPT_SET(V_i)



ATTR_SET(2)={(OBJ;"take")}

ATTR_SET(4)={(AGE;"take","breed", "meet"),(OBJ;"fit","meet"), (SOU;"take")}

ATTR_SET(5)={(OBJ;"breed")}

ATTR_SET(6)={(AGE;"take","breed","meet"),(OBJ;"fit", "meet"),(SOU;"take")}

ATTR_SET(10)={(INS;"breed")}

ATTR_SET(11)={(AGE;"fit"), (OBJ;"fit")}

ATTR_SET(14)={(LOC;"take","breed","meet")}

ATTR_SET(15)={(LOC;"breed", "meet")}

ATTR_SET(16)={(LOC;"take")}

ATTR_SET(20)={(AGE;"fit")}

Fig. 2. Examples of trie representation

such that (X, K) in CONCEPT_SET(V_i) and STRING(X) = y. If ATTR_SET contains (K, V_i) and (K, V_j), it is denoted by $(K; V_i, V_j)$.

Figure 2 shows TRIE(VERB_SET) for VERB_SET = {'take', 'breed', 'fit', 'meet'} shown in Section 2.1. Nodes circled by a bold line means that ATTR_SET is defined. For instance, STRING(< human >) = '111' and STRING(< organization >) = '12' are contained in both CONCEPT_SET('take') and CONCEPT_SET('breed') as (< human >, AGE) and (< organization >, AGE), therefore ATTR_SET(4) contains (AGE; 'take', 'breed') for node 4 such that $g(1, '111') = 4$.

For the trie of Fig. 2, consider matching of (< tropical fish >, OBJ). The prefix '112' of STRING(< tropical fish >) = '11211' is defined on the path from the initial node 1 to node 5 and ATTR_SET(5) contains (OBJ, 'breed'). Thus, matching of the superconcept (< animal >, OBJ) is successful for CONCEPT_SET('breed'). At the halfway node 2, ATTR_SET(2) contains the concept (OBJ, 'take'), thus matching of (< concrete object >, OBJ) is successful too.

In the case of the trie, it is possible to retrieve all of the superconcept with one traversal by confirming ATTR_SET(m) for each node m on the path.

3.2. A retrieval algorithm using a trie structure

Suppose that a retrieval algorithm takes as the input TRIE(VERB_SET), case attribute K and concept X and produces pair (LEN, SUB_SET) as the output, where LEN is the length of a matchable prefix of STRING(X) and SUB_SET is a subset of VERB_SET. But if LEN = 0, then SUB_SET becomes the empty set. Let POS be the current code position of STRING(X) = C_1, C_2, $\cdots$, C_n and let NODE be the current node number of TRIE(VERB_SET). A retrieval algorithm using a trie is presented as follows:

```
[Retrieval Algorithm]
begin
Step (R1): {Initialization}
        POS is initialized to 1;
        NODE is initialized to 1;
Step (R2): {Retrieval of the trie}
        if g(NODE, C_POS) ≠ fail then Set g(NODE, C_POS) to NODE
        else Terminate this algorithm;
Step (R3): {Producing the output}
        if ATTR_SET(NODE) is defined for the input attribute K
        then begin
      SUB_SET is initialized to the empty set;
      Append all verbs V such that (K, V) is in ATTR_SET(NODE) into SUB_SET;
      Set POS to LEN;
      Produce (LEN, SUB_SET);
  end;
    Step (R4): {Advance an arc}
      Increment POS;
      if POS > n then Terminate this algorithm else go to Step (R2);
end;
```

For a sentence 'A doctor takes tropical fishes in the fish tank', (<doctor>, AGE), (<tropical fish>, OBJ) and (<fish tank>, INS) are retrieved on the trie shown in Fig. 2. Consider STRING(<tropical fish>) = '11211' of (<tropical fish>, OBJ). (1, 'take') from LEN = POS = 1 and SUB_SET = {'take'} is obtained for the first prefix '1' at step (R3) because g(1, '1') = 2 and ATTR_SET(2) = {OBJ; 'take')}. The next output is (3, {'breed'}) for the prefix '112' detected by g(1, '112') = 5 and ATTR_SET(5) = {OBJ; 'breed')}. By the same manner, (3, {'take', 'breed', 'meet'}) for STRING(<doctor>) = '111111' of (<doctor>, AGE) and (5, {'breed'}) for STRING(<fish tank>) = '131111' of (<fish tank>, INS) are obtained. It is clear that the input sentence satisfies the constraint of the case structure for CONCEPT_SET('breed').

4. A COMPRESSION ALGORITHM OF TRIES

When all of the concept codes for VERB_SET are merged into one trie, the resulting trie becomes more compact, but it is really inefficient in the cost for retrieving a long verb list of (K, V) in the ATTR_SET. On the other hand, when some of the tries are constructed for the divided VERB_SET, the retrieval time of the verb list becomes fast, but inefficient at a compression rate. This paper presents a method of combining these

approaches. This method is to define the upper bound for the verb list of ATTR_SET without degrading the compactness of the resulting tries. In order to obtain a shorter verb list, the maximum number of verbs to be stored into one trie is restricted by MAX_VERB. The formal definition is as follows:

A superimposing degree DEGREE(S_1, S_2, ..., S_n) is defined by $n \times \alpha$, where n is the number of sets for the intersection COMMON = $S_1 \cap S_2 \cap \ldots \cap S_n$, and α is the total number of elements of COMMON.

The compression rate of the restricted trie depends on the degree, so it is better to choose an intersection with a larger DEGREE. The intersection is determined by a first-fit approach in the presented method. The following merging algorithm takes the whole set U of CONCEPT_SET(V) as the input and produces a subset MERGE_SET of U to be stored into one trie as the output. CONCEPT_SET(V) is denoted by S, for simplicity. Let TEMP_COMMON be the current intersection set, let TEMP_SET be the current subset of U and let DEGREE_MAX be the current maximum degree. Suppose that MAX_VERB = 16 and $\alpha > 1$.

[Merging Algorithm]
begin
Step (M1): {Find the initial TEMP_COMMON}
  Set U to TEMP_U;
  if There are S and S' in TEMP_U such that α = NUM($S \cap S'$) > 1 is the maximum
  then begin
    Set $S \cap S'$ to TEMP_COMMON;
    Set {S, S'} to TEMP_SET and MERGE_SET, respectively;
    Set $2 \times \alpha$ to DEGREE_MAX;
    Remove S and S' from TEMP_U;
  end;
  else begin {There are no sets S and S' in TEMP_U such that α = NUM($S \cap S'$) > 1}
    Set TEMP_U to MERGE_SET; {All remaining elements are merged}
    Produce MERGE_SET; {The final output}
    Terminate this algorithm;
  end;
Step (M2):{Determine set S for the next merging}
  if NUM(TEMP_SET) > MAX_VERB **then go to** Step (M4);
  if There is S in TEMP_U such that α = NUM(TEMP_COMMON $\cap$ S) > 1 is the maximum
  then go to Step (M3)
  else {There is no S in TEMP_U such that α = NUM(TEMP_COMMON $\cap$ S) > 1}
  go to Step (M4);

Step (M3): {Update of TEMP_COMMON}
  Set TEMP_COMMON $\cap$ S to TEMP_COMMON;
  Remove S from TEMP_U;
  Add S to TEMP_SET;
  if NUM(TEMP_SET) $\times \alpha$ > DEGREE_MAX
  then begin
    Set TEMP_SET to MERGE_SET;
    Set NUM(TEMP_SET) $\times \alpha$ to DEGREE_MAX;
  end
  go to step (M2);

Step (M4): {Output}

Produce MERGE_SET;
Set U-MERGE_SET to TEMP_U;{All elements in MERGE_SET are removed from U.}
if U is not the empty set **then go to** step (M1)
else Terminate this algorithm;
end;

Figure 3(a) shows a universal set $U = \{S_1, S_2, S_3, S_4\}$ for CONCEPT_SET introduced in Section 2.1 and Fig. 3(b) shows the merging process, where S_1, S_2, S_3 and S_4 correspond to CONCEPT_SET for 'take', 'breed', 'fit' and 'meet'. $C_1 \sim C_{15}$ correspond to the following pairs.

$C_1 = (<\text{human}>, \text{AGE})$.
$C_2 = (<\text{organization}>, \text{AGE})$.
$C_3 = (<\text{concrete object}>, \text{OBJ})$.
$C_4 = (<\text{shop}>, \text{LOC})$.
$C_5 = (<\text{place name}>, \text{LOC})$.
$C_6 = (<\text{human}>, \text{SOU})$.
$C_7 = (<\text{organization}>, \text{SOU})$.
$C_8 = (<\text{animal}>, \text{OBJ})$.
$C_9 = (<\text{breeding instrument}>, \text{INS})$.
$C_{10} = (<\text{building}>, \text{LOC})$.
$C_{11} = (<\text{clothing}>, \text{AGE})$.
$C_{12} = (<\text{color vision}>, \text{AGE})$.
$C_{13} = (<\text{human}>, \text{OBJ})$.
$C_{14} = (<\text{organization}>, \text{OBJ})$.
$C_{15} = (<\text{clothing}>, \text{OBJ})$}.

As shown in Fig. 3(b), step (M1) determines $S = S_2$ and $S' = S_4$ from the maximum number $\alpha = \text{NUM}(S_2 \cap S_4) = 4$ for TEMP_U $= \{S_1, S_2, S_3, S_4\}$, and prepares TEMP_COMMON $= S_2 \cap S_4$, TEMP_SET = MERGE_SET $= \{S_2, S_4\}$, DEGREE_MAX $= 2 \times 4 = 8$ and TEMP_U $= \{S_1, S_3\}$. In step (M2), S_1 is selected as the next candidate because $\alpha = \text{NUM}(\text{TEMP\_COMMON} \cap S_1) = 3$ and $\alpha = \text{NUM}(\text{TEMP\_COMMON} \cap S_3) = 0$, Therefore, TEMP_COMMON $= S_1 \cap S_2 \cap S_4$, TEMP_SET $= \{S_1, S_2, S_4\}$, TEMP_U $= \{S_3\}$ are obtained and DEGREE_MAX and MERGE_SET are changed by 9 and $\{S_1, S_2, S_4\}$, respectively, since NUM(TEMP_SET) $\times \alpha = 3 \times 3 = 9 >$ DEGREE_MAX $= 8$. The next step (M2) can not find the new merging set S for TEMP_U $= \{S_3\}$ because TEMP_COMMON $= S_1 \cap S_2 \cap S_3 \cap S_4$ is the empty set. The first output MERGE_SET $= \{S_1, S_2, S_4\}$ is produced at step (M4) and U is changed by U-MERGE_SET $= \{S_3\}$. For TEMP_U $= \{S_3\}$, step (M1) produces the final MERGE_SET $= \{S_3\}$. Accordingly, VERB_SET is divided into two sets {'take', 'breed', 'meet'} and {'fit'}.

If tries are constructed for the divided VERB_SET, then it is clear that the worst-case cost retrieving the verb list of ATTR_SET on the algorithm in Section 3.2 is constant because the longest verb list can be restricted by MAX_VERB.

5. EVALUATION

5.1. Data structure and theoretical observations

The trie TRIE(VERB_SET) and the verb list of ATTR_SET can be implemented by a double-array structure (Aoe, 1991). The double-array uses two one-dimensional arrays BASE and CHECK to represent trie's arcs (Fig. 4). The index of BASE and CHECK corresponds to the trie's node. Suppose that the index number is the same as the node number. The double-array represents the transition $g(s, a) = t$ of a trie as follows:

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
S_1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
S_2	1	1	0	0	1	0	0	1	1	1	0	0	0	0	0
S_3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
S_4	1	1	0	0	1	0	0	0	0	1	0	0	1	1	0

(a) An universal set U={S_1,S_2,S_3,S_4 }

																α	DEGREE
$S_1 \cap S_2$	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	3	6
$S_1 \cap S_3$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$S_1 \cap S_4$	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	3	6
$S_2 \cap S_3$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$S_2 \cap S_4$	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	4	8
$S_3 \cap S_4$	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	2	4
$S_1 \cap S_2 \cap S_4$	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	3	9
$S_2 \cap S_3 \cap S_4$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Intersection of a merging process.

Fig. 3. Examples of merging method

$$t = \text{BASE}[s] + a \quad \text{and} \quad \text{CHECK}[t] = s, \quad \text{where the symbol } a \text{ is a numerical value.}$$

Namely, the value of $g(s, a)$ is mapped into CHECK[t] = s to represent an arc labeled a from node s to node t. The worst-case time complexity of retrieving an arc is O(1), thus, the worst-case complexity of retrieving STRING(X) becomes O(k) for length k of STRING(X).

The attribute part of ATTR_SET can be also included by the trie representation based on the double-array, where arc labels use the first character of attribute names, 'A' for AGE, 'O' for OBJ and so on. For example, the modified trie for the original trie shown in Fig. 2 is depicted in Fig. 5. The modified trie has arc lables for attribute names and their node numbers correspond to the index numbers of the double-array shown in Fig. 6(b), where Fig. 6(a) indicates the numerical codes (internal values) of arc labels. Actually, two-byte codes like Japanese are divided into one-byte arcs. In this example, t = BASE[1] + '2' = 3, CHECK[3] = 1 then g(1, '2') = 3 can be retrieved. Attribute OBJ of ATTR_SET(2) is retrieved as follows:

$$t = \text{BASE}[2] + \text{'O'} = 4 + 5 = 9, \qquad \text{CHECK}[9] = 2.$$

By including the attribute part of ATTR_SETT into the modified trie, the verb list of ATTR_SET can be represented by a bit vector for sets shown in Fig. 5. The bit vectors are shown in Fig. 6(c) as an array VECTOR. A set {'take'} for node 9 in the modified trie can be retrieved by BASE[9] = -1 and VECTOR[1], where a minus integer of the BASE value indicates that it is the index of the array VECTOR. The length of the bit vector is easy restricted by the maximum number MAX_VERB as 2, or 4 bytes words, so the worst-case time complexity of searching the verb list becomes constant.

In the linear search, the worst-case time complexity of determining hierarchical relationships in the case slots becomes approximately O(nk) for the number n of concepts and for the length k of STRING(X) corresponding the input concept X. The presented method can be improved to constant time complexity.

For the number m of sets in U and for h = MAX_VERB, the worst-case time complexity of step (M1) in the merging method in Section 4 is O(m^2) by selecting two sets from m sets in U. The worst-case time complexity for one operation of steps (M2)

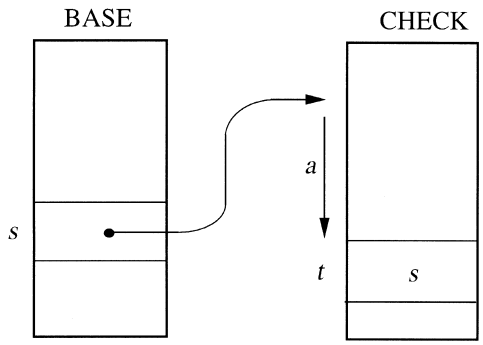


Fig. 4. An illustration of the arrays BASE and CHECK of the double-array structure

and (M3) is O(m) and the maximum number of repeating these steps is h, so the total cost is O(mh). Since mh can be neglected for m^2, the worst-case time complexity of the presented method is O(m^2). On the other hand, if the largest set of intersection is required for an universal set U, then DEGREE must be calculated for all of subsets of U. Accordingly, the complexity becomes O(2^m) for the number 2^m-1 of subsets.

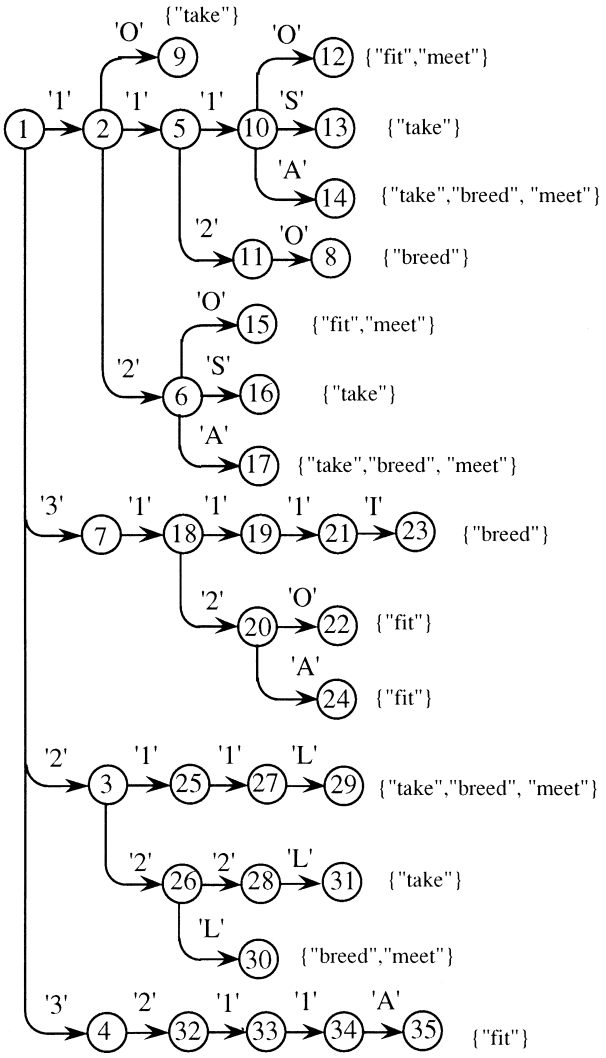


Fig. 5. An example of the double-array

Arc lables	'1'	'2'	'3'	'L'	'O'	'S'	'A'	'I'
Numerical codes	1	2	3	4	5	6	7	8

(a) Numerical codes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
BASE	1	4	24	30	9	10	17	-4	-1	7	3	-3	-1	-2	-3	-1	-2	18
CHECK	0	1	1	1	2	2	1	11	2	5	5	10	10	10	6	6	6	7

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
BASE	20	17	15	-5	-4	-5	26	26	25	27	-2	-6	-1	32	33	28	-5
CHECK	18	18	19	20	21	20	3	3	25	26	27	26	28	4	32	33	34

(b) The double-array

	"take"	"breed"	"fit"	"meet"
VECTOR[1]	1	0	0	0
VECTOR[2]	1	1	0	1
VECTOR[3]	0	0	1	1
VECTOR[4]	0	1	0	0
VECTOR[5]	0	0	1	0
VECTOR[6]	0	1	0	1

(c) Bit Vectors

Fig. 6. The modified trie

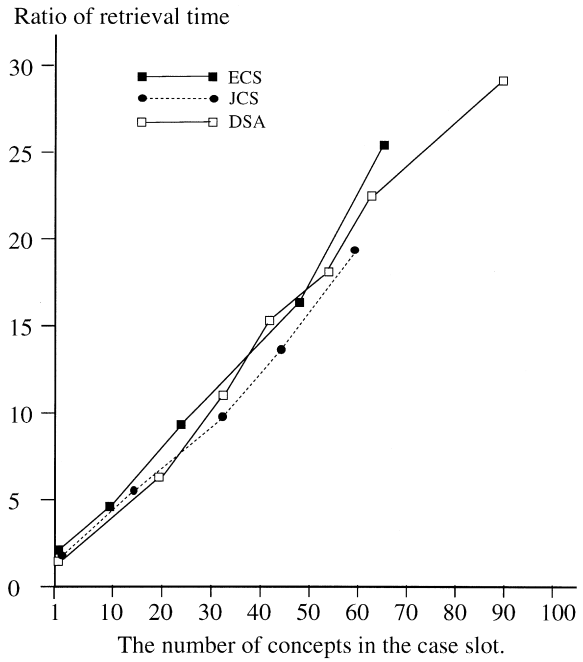


Fig. 7. Comparison of retrieval time for trie and linear storage

Table 1. Information about case structures and hierarchical concepts

	ECS	JCS	DSA
Information about case structures			
Number of verbs	6,431	12,068	6,588
Number of concepts for an attribute			
Average	3.2	3.2	23
Maximum	62	59	96
Information about hierarchical concepts			
Number of concepts	402,028	402,028	1,843
Depth of hierarchical structures			
Average	8.1	8.1	5.8
Maximum	16	16	11
Number of concepts for a noun			
Average	1.8	1.5	2.2
Maximum	63	56	18

5.2. Empirical observations

In this simulation, DELL OptiPlex (PentiumPro 200 MHz) computer has been used and the program has been written in C + + . The experimental data has been prepared as follows:

1. ECS: English case structures and word selection data (Kirkpatrick, 1987; EDR, 1993) for the English-to-Japanese machine translation system.
2. JCS: Japanese case structures and word selection data (Mizutani *et al.*, 1983; EDR, 1993) for the Japanese-to-English machine translation system.
3. DSA: Japanese case structures and disambiguation rules (for homophones (Ohno & Hamanishi, 1981; Mizutani *et al.*, 1983; Koizumi *et al.*, 1989).

The details of experimental data are indicated in Table 1 and the results of retrieval time are shown in Fig. 7. In Fig. 7, the horizontal axis indicates the number of concepts in the case slot, and the vertical one indicates the speeding-up ratio of retrieval time of the presented method to that of linear searching. For example, the retrieval time of the presented method becomes about 20 times faster than that of linear searching for 70 concepts in the case slot. From Fig. 7, it turns out that the retrieval time is improved remarkably for a large number of concepts.

Table 2 shows a comparison of storages for the presented method with linear searching, where the size of tries includes STRING(X) and attribute information of

Table 2. Simulation results

	ECS	JCS	DSA
Information about tries			
Number of tries	1,056	1,672	908
Number of nodes	18,568	31,498	13,899
Depth of ties			
Average	5.7	4.6	4.5
Maximum	15	14	9
Number of the merged verbs			
Average	8.1	8.3	4.9
Maximum	31	32	13
Storages (K-bytes)			
Tries	199	269	147
VECTOR	21	23	16
Linear Searching	188	233	134

ATTR_SET, the size of VECTOR includes a bit vector representation of the verb list. From the result, it turns out that the presented method keeps compactness.

The merging algorithm runs in 1.2 (ECS), 1.1 (JCS) and 0.3 (DSA) h, where MAX is 32 (ECS), 32 (JCS) and 16 (DSA). That is, the number of bytes for representing the verb list of ATTR_SET becomes 4 (ECS), 4 (JCS) and 2 (DSA), respectively. We can say that this time is practical because the data to be merged is static.

6. CONCLUSION

This paper has presented a speeding-up method for the determination of hierarchical concepts in the case structures. The trie structures has been introduced to represent hierarchical structures and the presented algorithm can retrieve a hierarchical relationship in a constant time. The presented approach has been evaluated by theoretical analysis and it has been supported by the experimental results. The future study is to apply this approach for machine translation systems.

REFERENCES

Aoe, J. (1990). A compact representation of hierarchical relations using decimal notations. *Int. J. Comput. Math., 33,* 37–54.

Aoe, J. (1991). An efficient digital search algorithm by using a double-array structure. *IEEE Trans. Software Eng., 15*(9), 0.

Dundas, J. A. (1991). Implementing dynamic minimal prefix tries. *Software Prac. Exp., 21*(10), 1027–1040.

EDR (Electronic Dictionary Research Institute) (1993). *Specification for EDR electronic dictionary* (in Japanese).

Fillmore, C. J. (1968). The case for case. In *Universals in linguistic theory*. New York: Hort, Reinhart and Winston.

Fredkin, E. (1960). Trie memory. *Commun. ACM, 3*(9), 490–500.

Hakomori, S., Sagawa, Y., Ohnishi, N., & Sugie, N. (1992). A basic research on an evaluation system of modification structure in Japanese text. *Trans. IPSJ, 33*(2), 153–161 (in Japanese).

Kirkpatrick, B. (1987). *Roget's thesaurus of English words and phrases* (new ed.). Longman.

Knuth, D. E. (1973). *The art of computer programming* (Vol. 3, pp. 481–505).

Koizumi, T., Funaki, M., Honda, K., Nitta, Y., & Tsukamoto, H. (1989). *Dictionary about usage of Japanese basic verbs*. Taisyukan-Syoten (in Japanese).

Makino, H., & Kizawa, M. (1981). An automatic translation system of non-segmented kana sentences into kanji-kana sentences and its homonym analysis. *Trans. IPSJ, 22*(1), 59–67 (in Japanese).

McArthur, T. (1981). *Longman lexicon of contemporary English*. Longman.

Miyazaki, M., & Ooyama, Y. (1981). Automatic reading method for kanji homographs using hierarchical semantics. *Trans. IEICE, J68-D*(3), 392–399 (in Japanese).

Mizutani, S. *et al.* (1983). *Grammar and meaning I*. Asakura-Syoten (in Japanese).

Nagao, M., Tsujii, J., & Nakamura, J. (1986). Machine translation from Japanese into English. *Proc. IEEE, 74*(7), 993–1012.

Nagao, M. (1996). *Natural language processing*. Iwanami-Syoten (in Japanese).

Nakaiwa, H., & Ikehara, S. (1993). Zero pronoun resolution in a Japanese to English machine translation system using verbal semantic attributes. *Trans. IPSJ, 34*(8), 1705–1715 (in Japanese).

Ohno, S. & Hamanishi, M. (1981). *Synonym new dictionary*. Kadokawa-Syoten (in Japanese).

Ohshima, Y., Abe, M., Yuura, K., & Takeichi, N. (1986). A disambiguation method in kana-to-kanji conversion using case frame grammar. *Trans. IPSJ, 27*(7), 679–687 (in Japanese).

Oishi, A. & Matsumoto, Y. (1995). A method for deep case acquisition based on surface case pattern analysis. *Proceedings of the 3rd Natural Language Processing Pacific Rim Symposium* (pp. 678–684).

Okumura, M., & Tanaka, H. (1989). Incremental disambiguation model of natural language analysis. *J. JSAI, 4*(6), 687–694 (in Japanese).

Shimazu, A., Naitoh, S., & Nomura, H. (1986). A method of Japanese language semantic analysis with structure prediction. *Trans. IPSJ, 27*(2), 165–176 (in Japanese).

Standish, T. A. (1980). *Data structure techniques* (ch. 3). Reading, MA: Addison-Wesley.

Takamatsu, S., & Nishida, F. (1981). English–Japanese translation based on verb patterns and their case-structures. *Trans. IEICE, J64-D*(9), 815–822 (in Japanese).