格構造解析における概念階層の効率的判定アルゴリズム

小山雅史[†] 泓田正雄[†]
岡田真[†] 青江順一[†]

シソーラスに代表される階層化された分類体系は，非常にシンプルな知識表現であり，その応用範囲は非常に広い．特に，自然言語文の格構造解析では，格スロットの制約条件として，この階層化された概念体系（以後概念階層と呼ぶ）による上位と下位関係の判定がよく利用される．しかしながら，階層が深くなり，また解析文が複雑になると，この判定コストが増加するので，判定処理の高速化は重要な課題である．本論文では，概念階層のデータ構造にトライ構造を導入して，判定効率を向上させる手法を提案する．また，本手法では複数の用言の格構造を1つのトライに併合して格納するので，トライ検索後に目的とする用言を効率的に確定する手法も提案する．同音語とEDR[1]の共起データに対する実験結果により，格スロットに概念20から96個を列挙する線形格納法に比べて，本手法は6から25倍高速化されることが分かった．また，トライを併合圧縮した記憶量は線形格納法とほぼ同等となり，十分に実用的な結果が得られた．

An Efficient Algorithm for Determining Hierarchical Concepts of Case Structures

Masafumi Koyama,[†] Masao Fuketa,[†] Makoto Okada[†] and Jun-ichi Aoe[†]

Case structure is a well-known approach for natural language analysis of Kana-Kanji transformation, machine translation and so on. The case slot is generally restricted by hierarchical concepts, because they are simple knowledge representations. With growing hierarchical structures, the number of concepts increases and the length of numerical codes representing concepts becomes long. The frequency of determinations is very high for complex sentences. Multiple concepts correspond to one word. From these reasons, it costs a lot to determine whether a concept for a given word is a sub-concept for concepts of the case slot or not. This paper presents a fast method for determining the sub-concept relationship by introducing trie structures instead of a sequential storage of concept codes. The worst-case time complexity of the determination process by the presented method remarkably improves the one of the sequential storage, which depends on the number of concepts in the slot. The trie structures can be compressed by merging common transitions into one trie. From the simulation results, it is shown that the presented algorithm is 6 to 25 times faster than the sequential storage, while keeping a smaller size of tries.

1. まえがき

シソーラス[2),3)]に代表される階層化された分類体系は，人間にとっても理解しやすく，また機械へのインプリメントも容易であるので，その利用価値は高い．また，この階層表現（以後，概念階層と呼ぶ）は，2つの概念の上位と下位の関係に基づく制約知識として利用され，意味解析や曖昧性の解消などの基礎的研究[4),5)]；仮名漢字変換の同音語判定[6)~8)]；機械翻訳システムの意味処理に対する単語の意味分類[9),10)]；文書校正処理[11)]；同形語の読み分け[12)]など多くの応用分野を有する．特に，階層の判定は自然言語文の格構造解析と連携して行われる場合が多いので，本論文では，格構造解析[13),14)]における格スロットの意味制約で利用される概念階層の効率的な記憶検索手法を提案する．

概念階層の各ノードの概念を10進分類法（decimal notation）で表現したものを概念コードとするとき，格スロットの意味制約は，概念コードを列挙する格納法（線形格納法と呼ぶ）で表現するのが一般的である．しかし，1つの単語に対応する概念は複数存在し，ま

† 徳島大学工学部知能情報工学科
Department of Information Science and Intelligent Systems, Tokushima University

た概念階層が大きく（深く）なると，格スロットの制約条件にも多くの概念が使用され，またそれぞれの概念コードも長くなるので，概念階層による上位下位の判定（階層判定と呼ぶ）回数は非常に多くなる．複数の用言を含む複雑な自然言語文の解析では，この判定回数はさらに増加するので，階層判定の効率化は重要な課題となってくる．これに対して，青江ら[15]は，階層関係を表現する概念コードの圧縮方法を提案したが，階層判定の高速化は，議論していない．

本論文では，概念コードの記憶検索にトライ構造を導入して，階層判定を高速化する手法を提案する．本手法では，複数の用言の格構造をトライに併合するので，トライ検索後に目的とする用言を効率的に確定する手法も提案する．以下，2章では格構造と概念階層を説明し，線形格納法に基づく階層判定法の概要と問題点を説明し，3章ではトライ構造を利用した判定アルゴリズムを提案する．4章では，トライの併合手法を提案し，5章では，実験により提案手法を評価する．6章では，本手法のまとめと今後の課題について触れる．

2. 概念階層における検索手法の概要

2.1 格構造と概念階層

名詞句の概念Cと格属性（主格，目的格[13),14)]など）Aの組を(C,A)で表すとき，用言Vに対する格構造を(C,A)を要素とする集合CASE_SET(V)で表す．入力文から得られる名詞句の概念Xとその格属性Kによる組(X,K)に対して，概念Xの上位概念Cである要素(C,K)を格構造CASE_SET(V)が含むとき，組(X,K)の照合がCASE_SET(V)で成功したことになる．概念Xに対する概念コードCODE(X)として，10進分類記法がよく利用され，XとYの階層判定はCODE(X)とCODE(Y)の接頭辞関係の判定と等価になる[15]．

用言“買う”，“飼う”および“合う”，“会う”に対する格構造の例を**表1**に示す．ただし，AGEは主格AGENT，OBJは目的格OBJECT，LOCは場所格LOCATION，SOUは源泉格SOURCEを表し，概念名は＜＞で囲んで示す．また，概念コードを含む概念階層の例を**図1**に示す．たとえば，CODE(＜無生物＞)=13は，＜水槽＞のコード131111の接頭辞であるので，＜無生物＞は＜水槽＞の上位概念であることが分かる．例文“医者が水槽で熱帯魚をかう”に対して，“かう”の同音語を決定する場合，名詞句の概念と格属性候補の組(＜医者＞, AGE)，(＜水槽＞, INS)，(＜熱帯魚＞, OBJ)をCASE_SET(“飼う”)とCASE_SET(“買う”)の要素と照合する．図1の概念階層より，(＜医者＞, AGE)と(＜熱帯魚＞, OBJ)は，CASE_SET(“飼う”)の要素(＜人間＞, AGE)と(＜具体物＞, OBJ)にそれぞれ照合が成功し，同様にCASE_SET(“買う”)の要素(＜人間＞, AGE)と(＜動物＞, OBJ)に成功する．しかし，(＜水槽＞, INS)は，CASE_SET(“買う”)の要素とは照合が成功せず，CASE_SET(“飼う”)の要素(＜飼育道具＞, INS)としか照合が成功しないので，“飼う”が選択できる．

表1 格構造CASE_SET(V)の例
Table 1 Examples of case structuresCASE_SET(V).

用言V	CASE_SET(V)の要素
買う	(＜人間＞, AGE) (＜組織＞, AGE) (＜具体物＞, OBJ) (＜店＞, LOC) (＜地名＞, LOC) (＜人間＞, SOU) (＜組織＞, SOU)
飼う	(＜人間＞, AGE) (＜組織＞, AGE) (＜動物＞, OBJ) (＜飼育道具＞, INS) (＜建物＞, LOC) (＜地名＞, LOC)
合う	(＜衣類＞, AGE) (＜色彩＞, AGE) (＜人間＞, OBJ) (＜組織＞, OBJ) (＜衣類＞, OBJ)
会う	(＜人間＞, AGE) (＜組織＞, AGE) (＜人間＞, OBJ) (＜組織＞, OBJ) (＜建物＞, LOC) (＜地名＞, LOC)

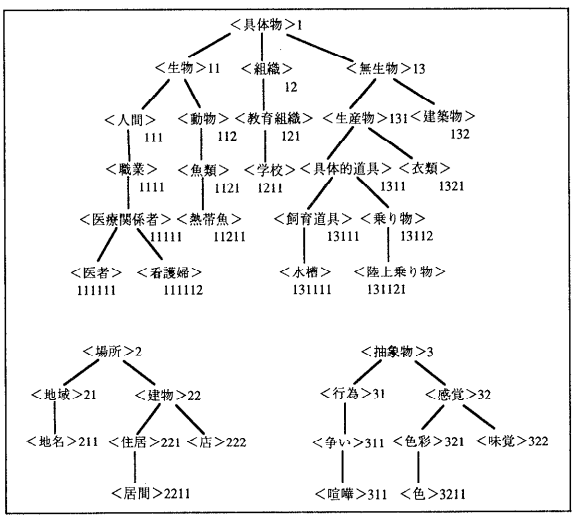


図1 概念階層の例
Fig. 1 Examples of hierarchical relationships among concepts.

2.2 概念間の距離を加味した照合結果

前節の例文では，(＜水槽＞, INS)により“飼う”が選択できたが，例文“医者が熱帯魚をかう”に対しては，両用言の格構造で照合が成功す

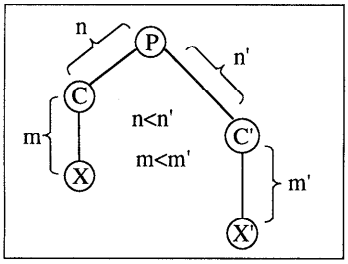


図2 概念コードの距離関係の例
Fig. 2 Examples of distance among concept codes.

るので区別できない．しかし，下位概念ほど抽象化レベルは低く，実体（instance）を制約する条件が強くなるので，＜具体物＞と＜動物＞の両方の下位概念である＜熱帯魚＞は，＜具体物＞より＜動物＞に近いといえる．したがって，照合が成功した中で優先順位を付けたい場合，概念の近さを加味した照合結果が有効である．

名詞には複数の概念が対応するので，近さの判定はもう少し複雑である．**図2** の名詞に対応する2つの概念 X と X' とそれぞれの上位概念 C と C' を考える．n, n'(n<n'); m, m'(m<m') は各概念の距離（概念間のパスの長さ，すなわち概念コード長の差）を表すとき，

$$n/(n+m) < n'/(n'+m') \qquad (1)$$

ならば，C と X の距離 m は，概念 C' と X' の距離 m' より近いが，概念 C' が概念 C より相対的に下位にある．この距離の利用形態は，応用分野により異なるが，判定アルゴリズムでは，この情報を出力できるようにしておく必要がある．

以上の観点より，本論文で議論する照合アルゴリズムでは，次を考慮する．

(1) 入力概念 X の上位概念で概念 X に最も近い概念 C を見つける．

(2) 式 (1) に示した概念距離の相対的な比較を行うために，出力情報は照合できた最長の概念コード CODE(C) の長さを持つ．

2.3 線形格納法による概念階層の検索

格構造 CASE_SET(V) に対して組 (X,K) の照合を行う手続き MATCH(V,X,K) を次のように定義する．

【手続き MATCH(V,X,K)】

手順 (1)：CASE_SET(V) を検索する．

手順 (2)：CASE_SET(V) 中の要素 (C,A) で A=K なるすべての概念 C に対して，次の手順3を実行する．

手順 (3)：概念 C が概念 X の上位概念であれば，C を出力する． （手続き終）

手順 (1) は用言 V をキーとする検索技法が利用でき，手順 (2) の概念 C も格属性 A を見出しとしてまとめて格納すれば容易に検索できる．したがって，問題になるのは手順 (3) の概念階層の判定コストである．表1のように概念列を列挙する線形格納法では，手順 (3) の計算量が概念数 n に対して $O(n)$ となり，この計算量は，次の要因でより深刻な問題となる．

（要因1）名詞に対応する概念数が多くなると照合回数が増加する．たとえば，“学校” は＜場所＞，＜組織＞，＜建物＞などの複数の概念に対応する．

（要因2）複数の用言を含む複雑な文では，複数の格構造で照合されるので，照合回数はさらに増加する．

（要因3）2.2節で述べたように，近い上位概念の検出では，概念コードと部分一致する接頭辞の長さの情報を残すための処理が必要となる．

（要因4）最も近い上位概念の検出では，上位概念が1つ発見されても，概念列の最後までの照合が必要となり，途中終了による効率化が期待できない．

さらに，より厳密な意味制約を行うためには，概念階層が複雑になる．この場合，概念コード列は長くなり，名詞に対応する概念数や格スロットの概念数も増加するので，上記のすべての要因による効率低下を加速させることになる．

2.4 格構造の利用形態の分類

以上は，1つの用言に対する格構造とその照合方法について議論したが，格構造の構築と照合法は利用分野により異なるので，この議論を先にしておく．

A) 限定された用言集合の曖昧性の解消

代表例は，表1の同じ読み “かう” で限定される同音語集合 {“買う”, “飼う”} の曖昧性の解消である．また，機械翻訳でも，“上げる” の英語訳で限定される集合は，“raise”, “lift”, “look up”, “improve”, “increase” などの要素を含む．同形語の判定[12]もこの範疇である．これらの応用では，限定された集合の中から適切な語彙を決定するために格スロットの制約条件を利用するので，限定された用言に対する格構造をまとめて構成し，その照合法を3章で提案する．

B) 限定されない用言集合の曖昧性の解消

この範疇として，係り受け構造の解析が考えられる．

(a) “洋服が東京で買った靴に合う”

(b) “私が東京で買った靴に合う”

(c) “私は東京で別れた彼に会う”

以上の例で，(a) では，“洋服が” は，用言 “合う” の主格であるが，(b) の “私が” は，“買う” の主格である．また，(c) では，“私は” と “東京で” が，それぞれ用言 “別れる” と “会う” の主格と場所格になりうるが，この文の情報だけではどちらの用言の格かは決

定できない．このような係り受け構造の解析では，用言集合を事前に限定できないので，4章では記憶量の軽減の観点から，この集合を限定する手法を提案する．

3. 階層判定の高速化アルゴリズム

3.1 格構造情報のトライ表現

用言集合 V_SET が $1 \leqq i \leqq h$ なる用言 V_i で限定されるとき，$1 \leqq i \leqq h$ なるすべての V_i の CASE_SET(V_i) に含まれる概念コード列からトライ[16),17)]（以後，TRIE(V_SET) で表す）を構成する．トライとは概念コードの共通接頭辞を共有アークとして併合した木構造であり，自然言語辞書の検索技法として利用されている．トライにおいて，ノード s から t に対してアークラベル a が定義されていることを関数 g を用いて，g(s,a)=t で表す．

本手法では，トライのノード m に対して，次の集合 ATTR_SET(m) を定義する．

TRIE(V_SET) の初期ノード 1 から m までのアークラベルの連鎖 t を PATH(1,m)=t で表すとき，CODE(X)=t なる要素 (X,K) を含む CASE_SET(V_i) のすべての V_i に対して，ATTR_SET(m) は，要素 (K,V_i) を含む．

ATTR_SET が同じ格属性 K の要素 (K,V_i) と (K,V_j) を含む場合，これを (K; V_i,V_j) と記述するとき，表 1 の用言集合 V_SET={“買う”, “飼う”} に対する TRIE(V_SET) を図 **3** に示す．ただし，図中の “喧嘩” に対する破線のアークと，( ) 内の番号は後に説明を加える．太線の○は，ATTR_SET が定義されているノードを示す．たとえば，CODE(＜人間＞)=111 と CODE(＜組織＞)=12 は CASE_SET(“買う”) と CASE_SET(“飼う”) に，(＜人間＞, AGE), (＜組織＞, AGE) として含まれるので，ATTR_SET(4)={AGE; “買う”; “飼う”} が定義できる．

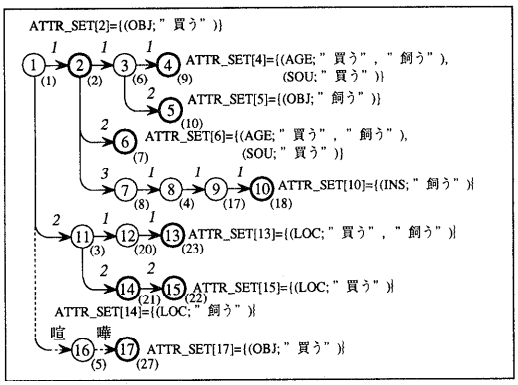


図 3 格構造のトライ表現例

Fig. 3 Examples of trie representation for case structers.

図 3 のトライで (＜熱帯魚＞, OBJ) の照合を考える．CODE(＜熱帯魚＞)=11211 より，初期ノード 1 から最初のコード 1 のアークラベルをたどりノード 2 へ進む．同様に，残りの 2 コード分 12 をたどると，ノード 5 へ到着する．ここで，ATTR_SET(5) は (OBJ, “飼う”) を含むので，用言 “飼う” の格属性 OBJ の上位概念＜動物＞に対して照合は成功することが分かる．また，探索途中のノード 2 でも (OBJ, “買う”) が ATTR_SET(2) に含まれるので，用言 “買う” の格属性 OBJ の上位概念＜具体物＞とも照合したことになる．

したがって，トライの探索では任意のノード m へ到達したとき，ATTR_SET(m) を確認することで，照合可能なすべての上位概念が 1 回のパスの走査で探索可能となる．形式的な検索アルゴリズムは次節に与える．

3.2 トライの検索アルゴリズム

トライによる検索アルゴリズムを次に示す．

【検索アルゴリズム 1】

[入力] TRIE(V_SET), 格属性 K, 概念 X

[出力] TRIE(V_SET) 上で，CODE(X) と照合できた最長の接頭辞の長さ LEN とそのときの用言 V_SET の部分集合 SUB_SET の組 (LEN,SUB_SET) を出力情報とするが，照合できる接頭辞がなければ，LEN=0, SUB_SET は空集合となる．

[方法]

手順 (1-1)：{初期化}

概念 X の概念コード CODE(X)=C_1 C_2 … C_n に対して，コード位置を示す変数 POS を 1 にセットする．トライ TRIE(V_SET) に対して，処理中のノードを表す変数 NODE を初期ノード 1 とする．接頭辞の長さを表す変数 LEN を 0 に初期化する．

手順 (1-2)：{アークの判定}

g(NODE,C_{POS}) ≠ fail ならば次の手順 (1-3) へ進む．fail ならば手順 (1-5) へ進む．

手順 (1-3)：{状態アークの進行}

NODE に g(NODE,C_{POS}) をセットし，POS をインクリメントする．ただし，POS が n を超えれば，手順 (1-5) へ進む．

手順 (1-4)：{最長接頭辞の確認}

ATTR_SET(NODE) が (K,V) なる要素を含むならば，SUB_SET を空集合に初期化し，すべての用言 V を要素として加える．そして，LEN に POS をセットする．手順 (1-2) へ戻る．

手順 (1-5)：{出力}

組 (LEN,SUB_SET) を出力する．

（アルゴリズム終）

SUB_SET は手順 (1-4) で初期化されるが，これはつねに最長の接頭辞に対応する用言 V を格納するために，古い（短い接頭辞）情報を消すことを意味する．

“医者が熱帯魚をかう” の “かう” の同音語を決定する場合，図3のトライ上で，(<医者>, AGE) と (<熱帯魚>, OBJ) を検索する．(<医者>, AGE) の検索結果 (LEN,SUB_SET) は，(3, {“買う”, “飼う”}) となり，(<熱帯魚>, OBJ) の出力は，(3, {“飼う”}) となり，2つの出力 SUB_SET の共通要素 “飼う” が制約を満足する同音語となる．

3.3 慣用表現による共起関係

“喧嘩を買う”，“油を売る” 等の慣用表現に対しては，名詞句の概念ではなく，(“喧嘩”, OBJ) のように，名詞表記自体を登録しなければならない．この場合にも，図3（破線で示したアーク参照）のトライに名詞句の表記を登録することは容易であり，しかも，検索コストも増加しない．

4. トライの併合アルゴリズム

4.1 併合手法の概要

対象となる用言が限定されない場合，V_SET をすべての用言集合と見なして，1つのトライ TRIE(V_SET) を構築する方法が考えられる．この方法は，トライの圧縮効率の点で優れているが，ATTR_SET の要素 (K,V) の用言列 V が非常に長くなり，トライ検索後の用言の探索コストが増加する．逆に，各用言 V に対して小さなトライを数多く構成する手法では，ATTR_SET の要素 (K,V) の V の情報は不要になるので，トライ検索後の ATTR_SET の要素判定は高速になるが，トライの細分化により，圧縮効果が薄れる．

本論文では，これらの融合手法を提案する．すなわち，ATTR_SET の用言列の長さの上限値を設定し，その上限値を超えない範囲で用言集合を限定し，トライの圧縮効率を上げる手法である．

4.2 併合アルゴリズム

ATTR_SET の用言列の長さの上限を MAX 以下にするには，V_SET の V の要素数を MAX 以下にすればよい．またトライの圧縮効率を考えるうえで，次の重なり度数を定義する．

用言 V の CASE_SET(V) を S とするとき，n 個の用言の格構造集合 $S_1, S_2, ..., S_n$ の重なり度数 DEGREE $(S_1, S_2, ..., S_n)$ を，積集合 COMMON=$S_1 \cap S_2 \cap ... \cap S_n$ の要素数 α=NUM(COMMON) に対して，$n \times \alpha$ と定義する．

併合したトライの圧縮率は重なり度数に比例するため，重なり度数が大きい組合せを優先して併合すればよい．したがって，これらの基準を満足した限定集合の決定手法を提案する．

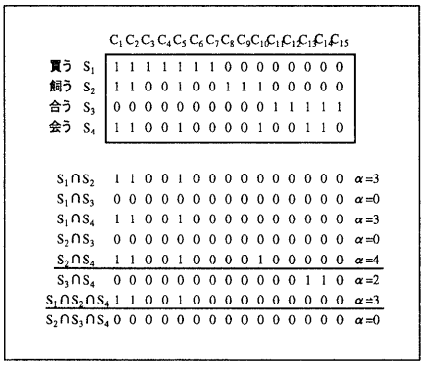


図4 併合手法の例
Fig. 4 An example of the merging method.

【併合アルゴリズム 2】
[入力] 格構造集合の全体集合 U
[出力] 併合する格構造の組合せ集合 PILE_SET
[方法]
手順 (2-1)：{準備}
格構造集合の積集合を一時的に格納する集合 TEMP を空集合にセットする．
手順 (2-2)：{重なり集合の検出}
TEMP が空集合でない場合，U から NUM(TEMP ∩ S) が最大となる集合 S を検出して次へ進む．空集合の場合，U からすべての2集合 S, S′ に対して，NUM(S ∩ S′) が最大になる組合せを検出し，PILE_SET={S, S′} をセットし，U から S, S′ を除く．そして TEMP を S ∩ S′ にセットして次に進む．また，重なり集合が存在しない場合，手順 (2-4) に進む．
手順 (2-3)：{重なり領域の更新}
NUM(PILE_SET)<MAX なる条件が成立するならば，PILE_SET に S を追加し，U から S を除く．TEMP に TEMP ∩ S をセットして手順 (2-2) に帰る．
手順 (2-4) {出力}
PILE_SET を出力して終了する．
（アルゴリズム終）

制約条件 MAX を 3 として，図4に，表1の4つの格構造の集合を併合する場合の例を示す．図4の S_1〜S_4 には表1の上から順に “買う” から “会う” を，C_1〜C_{15} には (<人間>, AGE) から (<衣類>, OBJ) を，順に割り当てている．手順 (2-2) では，積集合の要素数が最大になる組合せ S_2, S_4 を検出し，PILE_SET=$\{S_2, S_4\}$ をセットし，TEMP を $S_2 \cap S_4$ にセットする．次に，TEMP と集合 S_1, S_3 との積集合をとり，重なり度数の大きい S_1 を選択する．

NUM(PILE_SET)=2<MAX なので，PILE_SET に S_1 を追加し，TEMP に $S_1 \cap S_2 \cap S_4$ をセットする．続いて，残る集合 S_3 との積集合をとり，S_3 を検出するが，NUM(PILE_SET)=MAX=3 なので PILE_SET=$\{S_1, S_2, S_4\}$ を出力して終了する．したがって，V_SET は {“買う”, “飼う”, “会う”} に限定される．

最も容量が圧縮される組合せを求めるためには，全体集合 U の要素数 n に対して，すべての部分集合 2^n 個の重なり度数を調べねばならず，検出は困難である．したがって，本アルゴリズムでは，要素数の多い集合を優先することにより，これを $O(n \times MAX)$ に改善している．また MAX の値は実際の格構造集合に影響を受けるが，目安として 1 つの用言中の CASE_SET の要素数の平均値が考えられる．

4.3 併合されたトライの検索アルゴリズム

係り受け構造の解析では，概念 X と格属性 K の組 (X,K) と指定された用言 V の格構造との照合が必要であるので，ここでは検索アルゴリズム 1 を変更したアルゴリズムを提案する．

【検索アルゴリズム 3】

[入力] 用言 V，トライ TRIE(V_SET)，格属性 K，概念 X．ただし，V_SET は V を含む．

[出力] TRIE(V_SET) 上で，CODE(X) の接頭辞 t が存在し，PATH(1,m)=t なるノード m に対して，ATTR_SET(m) が (K,V) を要素として含むとき，接頭辞 t の長さ LEN を出力する．この接頭辞 t が発見できない場合 LEN=0 を出力する．

[方法]

手順 (3-1) から (3-3) は手順 (1-1) から (1-3) に同じ．

手順 (3-4)：{最長接頭辞の確認}

ATTR_SET(NODE) が (K,V) なる要素を含むならば，LEN に POS をセットする．手順 (3-2) へ帰る．

手順 (3-5)：{出力}

LEN を出力する． （アルゴリズム終）

5. 実装手法と評価

5.1 トライと ATTR_SET のデータ構造

トライ構造は青江ら[16]の提案したダブル配列法を使用することで，概念コードは概念数に依存しない（概念コードの長さに比例した）一定時間で探索できる．

ダブル配列法では（図 5），2 つの配列 BASE, CHECK でトライのアークを表現する．ダブル配列上のインデックス番号はトライのノードと 1 対 1 に対応しており，以下の説明では簡単のため，両者の値を一致させて説明する．ダブル配列はアーク $g(s,a)=t$ に対して，記号 a を内部表現値とするとき，次式を満足する．

$$t = BASE[s] + a \quad (2)$$

$$CHECK[t] = s \quad (3)$$

すなわち，$g(s,a)$ の値は式 (2) の値 t で決定され，式 (3) の CHECK[t] には，ノード t に入るアークが s から引かれていることを示す情報を格納する．ダブル配列によるノードのアークは上の 2 つの式を確認するだけであるため，つねに $O(1)$ で行われ，きわめて高速である．たとえば，図 3 のトライは，ダブル配列を用いて図 6 のように表現できる．図 6 の (a) は，アークラベルの内部コードを示す．この例では便宜上，各アークラベルに対して単一の内部表現値を割り当てているが，実際は 2 バイトコードには，1 バイト単位のアークに分割されて実装される．

図 6 の (b) がダブル配列であり，ノード番号は図 3 の（）内の番号に対応する．たとえば，t = BASE[1]+‘2’= 3，CHECK[3] = 1 より，$g(1(1),‘2’)=11(3)$（図 3 における（）内のノード番号も（）付きの番号で表す）が検索できる．ただし，ATTR_SET の格属性部分は，AGE を ‘A’，OBJ を

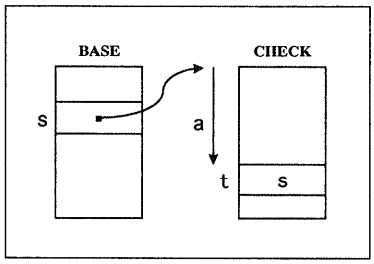


図 5 ダブル配列の説明
Fig. 5 A diagram of a double-array.

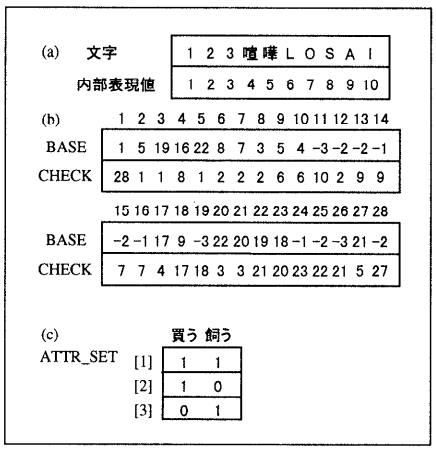


図 6 ダブル配列の例
Fig. 6 An example of the double-array.

'O' のようにその先頭文字をアークラベルとしてトライ上に組み込む．たとえば，図3の ATTR_SET(5(10)) の格属性 OBJ に対しては，t = BASE[10]+'O'= 4+7 = 11，CHECK[11] = 10 と定義される．ここで，ATTR_SET への参照は，ダブル配列内でのアークと区別するために，BASE に負の値で格納され，その値に-1 をかけた番号で ATTR_SET を参照する．したがって，BASE[11]=−3<0 より，格属性 OBJ によるアークは ATTR_SET[3] を参照する．

図6の(c)は "買う" と "飼う" をビット位置に対応させた ATTR_SET のビットベクトル表現である．たとえば，ATTR_SET[3] は "飼う" を示し，全体として ATTR_SET(5(10))=(OBJ, "飼う") と一致する．

限定された用言集合に対するトライでは，ATTR_SET の格属性に対応する用言数は多くならない．また，限定されない用言集合でも4章で提案した併合方法により，用言集合の上限要素数 MAX が制約できるので，用言列をビットベクトルで表現することで，トライ検索後の用言の確定コストも一定時間で行える．

5.2 検索効率の評価

n 個の概念が列挙されて格納される線形格納法では，階層判定の最悪の時間計算量が $O(n)$ となるが，提案手法では，5.1節の実装方法により最悪の時間計算量が概念コードの長さに比例したコストに改善でき，さらに ATTR_SET の要素判定のコストも一定時間に高速化できる．

次に，実験による具体的評価を示す．実験データは，文献 18)~20) を参考にして構築された仮名漢字変換の実用規模の同音語処理と，EDR[1]による共起処理データに対して行われた．実験データの詳細を**表2**に示し，検索時間結果を**図7**に示す．図7の横軸は，属性に対応する概念列（慣用句を含む）の数であり，縦軸は線形探索法に対する速度改善率を示す．検索時間は，データをすべて主記憶上に置いた状態で測定した．なお，使用した計算機は DELL OptiPlex (PentiumPro200 MHz) であり，開発言語は C++ である．

図7から，概念数の増加とともにトライ法を用いた速度改善率が著しく向上することが分かる．特に，判定回数は，名詞に対する概念数，格属性の数，文に含まれる用言数に比例して増加するので，提案手法による階層判定の高速化の効果は非常に大きいといえる．

5.3 記憶効率の評価

表3に，同音語データと EDR の共起データに対する線形格納法と提案手法の記憶量を比較する．

表3の結果より，提案手法は線形格納法とほぼ同等であり，記憶量も大きく増大しないことが分かる．これは，トライ構造自身の統合効果と4.2節で提案したトライの併合手法によるものである．

併合アルゴリズムは，同音語と EDR のデータに対して，ATTR_SET の要素 (K,V) の V の最大長 MAX を 16 と 32 に設定し，2 byte と 4 byte のビットベクトルで表現した場合，それぞれ 0.3，1.1 時間で計算を

表2 格構造と概念階層の情報
Table 2 Information about case structures and hierarchical concepts.

	同音語	EDR
格構造情報		
用言数	6,588	12,068
格属性に対する概念数		
平均（最大）	23(96)	3.2(55)
格属性に対する慣用語数		
平均（最大）	2.3(33)	0(0)
概念階層の情報		
概念数	1,843	402,028
階層の深さ 平均（最大）	5.8(11)	8.1(16)
名詞に定義されている概念数		
平均（最大）	2.2(6)	1.5(87)

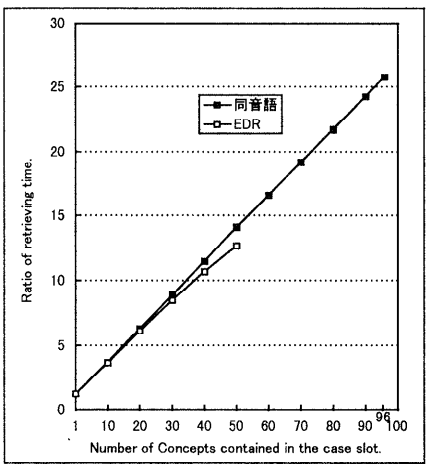


図7 検索時間の比較
Fig. 7 Comparison of retrieving time for trie and linear storage methods.

表3 記憶量の実験結果
Table 3 Simulation results of storage.

	同音語	EDR
トライの情報		
トライの総数	1,672	10,056
総ノード数	31,498	292,607
トライの深さ 平均（最大）	4.5(9)	5.6(14)
併合用言情報 平均（最大）	4.9(13)	8.3(32)
記憶量の比較		
トライの記憶量	269 KB	1.4 MB
ATTR_SET の記憶量	23 KB	161 KB
線形格納法の記憶量	233 KB	1.3 MB

終了した．したがって，本手法は実行時間においても実用的である．

6. むすび

以上，格構造解析を対象として，概念階層における階層の判定の高速化手法を提案し，提案手法の理論評価と実験評価により有効性を実証した．自然言語処理で取り扱う概念階層の規模はますます大きくなるので，提案した記憶検索法は意義があると思われる．

なお，実験で取り上げられなかった機械翻訳での訳語の制約に関する実験評価を行うことが，今後の課題である．

参考文献

1) 日本電子化辞書研究所：EDR 電子化辞書仕様説明書 (1993).
2) JIS X 0901：シソーラスの構成及びその作成方法 (1981).
3) 田中穂積，仁科喜久子：上位/下位関係シソーラス ISAMAP1 の作成 [I]，自然言語処理，Vol.64, No.4, pp.25–44 (1987).
4) 島津　明，内藤昭三，野村浩郷：構造予測を用いた日本語文の意味解析法，情報処理学会論文誌，Vol.27, No.2, pp.165–176 (1986).
5) 奥村　学，田中穂積：自然言語解析における意味的曖昧性を増進的に解消する計算モデル，人工知能学会誌，Vol.4, No.6, pp.687–694 (1989).
6) 大島義光，阿部正博，湯浦克彦，武市宣之：格文法による仮名漢字変換の多義解消，情報処理学会論文誌，Vol.27, No.7, pp.679–687 (1986).
7) 山本喜大，久保田淳市：共起グループを用いたかな漢字変換，第 44 回情報処理学会全国大会論文集，No.3, pp.189–190 (1992).
8) 牧野　寛，木澤　誠：べた書き文のかな漢字変換システムとその同音語処理，情報処理学会論文誌，Vol.22, No.1, pp.59–67 (1981).
9) 中岩浩巳，池原　悟：日英翻訳システムにおける用言意味属性を用いたゼロ代名詞照応解析，情報処理学会論文誌，Vol.34, No.8, pp.1705–1715 (1993).
10) 高松　忍，西田富士夫：動詞パターンと格構造に基づく英日機械翻訳，信学論（D），J64–D, No.9, pp.815–822 (1981).
11) 箱守　聰，佐川雄二，大西　昇，杉江　昇：日本語の修飾構造を評価する添削支援システムを実現するための基礎研究，情報処理学会論文誌，Vol.33, No.2, pp.153–161 (1992).
12) 宮崎正弘，大山芳史：階層的単語属性を用いた同形語の自動読み分け方，信学論（D），J68–D, No.3, pp.392–399 (1985).
13) Fillmore, C.J.，田中ほか（訳）：格文法の原理，三省堂 (1975).
14) 長尾　真（編）：自然言語処理，岩波書店 (1996).
15) 青江順一，山内美加子：階層構造をもつ知識表現の効率的記憶検索法，信学論（D），J70–D, No.2, pp.269–277 (1986).
16) 青江順一：キー検索技法—トライ法とその応用，情処学会誌，Vol.34, No.2, pp.244–251 (1993).
17) Dundas, J.A.: Implementing Dynamic MinimalPrefix Tries, *Softw. Pract. Exper.*, Vol.21, No.10, pp.1027–1040 (1991).
18) 小泉　保ほか：日本語基本動詞用法辞典，大修館書店 (1989).
19) 水谷静夫：文法と意味，朝倉書店 (1983).
20) 大野　晋，浜西正人：類語新辞典，角川書店 (1981).

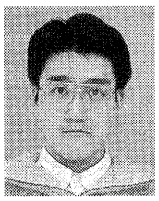
(平成 9 年 7 月 28 日受付)
(平成 10 年 1 月 16 日採録)

小山　雅史（学生会員）

昭和 46 年生．平成 6 年徳島大学工学部知能情報工学科卒業．平成 8 年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．情報検索，自然言語処理の研究に従事．

泓田　正雄（学生会員）

昭和 46 年生．平成 5 年徳島大学工学部知能情報工学科卒業．平成 7 年同大学院博士前期課程修了．現在同大学院博士後期課程在学中．情報検索，自然言語処理の研究に従事．



岡田　　真（学生会員）

昭和 49 年生．平成 8 年徳島大学工学部知能情報工学科卒業．現在同大学院博士前期課程在学中．情報検索，自然言語処理の研究に従事．

青江　順一（正会員）

昭和 26 年生．昭和 49 年徳島大学工学部電子工学科卒業．昭和 51 年同大学院修士課程修了．同年同大学工学部情報工学科助手．現在同大学工学部知能情報工学科教授．工学博士．電子情報通信学会，人工知能学会，日本認知科学会，日本機械翻訳協会，IEEE, ACM, AAAI, ACL 各会員．