自然言語処理 112－1
（1996．3．14）

トライ構造による概念階層の高速判定アルゴリズム

小山雅史　林淑隆　獅々堀正幹　青江順一
徳島大学工学部　知能情報工学科

概念階層は，最もシンプルな知識表現の一つであり，自然言語処理システムの多くの知識判定モジュールで利用されている．例えば，かな漢字変換における同音語共起判定，形態素解析での複合語処理，日本語文読み上げにおける同型語の読み分け，機械翻訳システムにおける訳文の決定，格構造解析における格の判定，及び文書検索等々数限りない応用分野がある．特にシステムが高度化するにつれて，階層の判定回数は非常に多く要求されるので，階層判定の高速化は重要な課題である．

本稿では，対象として格構造解析における共起概念のマッチングについて考察を行い，その上で，概念パターンの高速検索手法を提案する．また実験により，本手法の有効性を理論的，具体的に示す

A Fast Retrieval Algorithm for Hierarchical Concepts by Using Trie Structures

Masafumi Koyama, Yoshitaka Hayashi, Masami Shishibori, Jun-ichi Aoe
Department of Infomation Science and Intelligent Systems,
University of Tokushima

Hierarchical notations of concepts are useful for simple semantic representation of natural language processing systems such as Kana-Kanji transformation, analysis of compound words, constraints of case slots and so on. With the complexity of semantic information, a faster determination of hierarchical relationships is required. This paper presents a method to determine hierarchical relationships by using tree or trie structures. The approach has the following desirable features: (1) The worst-case time complexity of hierarchical decision is proportional to the length of the notations, (2) All concepts of slots for case-structures are confirmed by only one traversal of the tries. From the simulation results, it turned out that the presented method is about ten times faster than a linear search scheme.

1．はじめに

概念相互が上位／下位，全体／部分関係等で結ばれた知識の体系（以後概念階層と呼ぶ．）は，最もシンプルな知識表現の一つであり，自然言語処理システムの多くの知識判定モジュールで利用されている．例えば，かな漢字変換における同音語共起判定，形態素解析での複合語処理，日本語文読み上げにおける同型語の読み分け，機械翻訳システムにおける訳文の決定，格構造解析における格の判定，及び文書検索等々数限りない応用分野がある[1]～[4]．特にシステムが高度化するにつれて，階層の判定回数は非常に多く要求されるので，階層判定の高速化は重要な課題である．

本稿では，対象として格構造解析における共起概念のマッチングについて考察を行い，その上で，概念パターンの高速検索手法を提案する．また実験により，本手法の有効性を示すと共に理論面からの評価も行う．最後に，今後の課題及び本手法の応用性について触れる．

2．概念階層における検索手法の概要

格構造解析における名詞句の役割は，概念関係子と共起関係[5]となる概念とのマッチングにより判定を行う．図１を例に手順を示す．図中のagent，place等[6]は概念関係子を表し，これはＥＤＲの概念辞書の仕様に従っている．入力文”太郎に会う”の場合，前処理を経て

[会う goal 11(人間)]

の形で共起判定される．判定の手順は，まずデータ中から関係子goalを探し，その後goalの概念リストと線形マッチングを行うことになる．関係子が一意に決められない場合も考えられ，マッチングの計算量は共起概念数をｎとしたとき$O(n)$となり，判定回数や共起概念数が多い場合には速度の低下は避けられない．

従って本稿では，各見出し語の共起概念のデータ構造を線形に並べるのではなく，木構造（図２参照）で構成する．即ち，各見出し語の概念階層は全体の概念階層の部分木になっている．

本手法では概念階層を効率的に実現するために，トライを用いる[7]．以後見出し語の共起概念のトライを分類トライ(Classification Sub-Trie)と呼ぶ．トライに登録するキーは概念間を繋ぐノードにそれぞれ記号を割り振った記号列（以後概念コード（図１）と呼ぶ．）で，これ自体が階層構造内の各概念の位置を表している．従って，トライを用いることにより成功または不成功のみの照合結果だけではなく，階層レベルの判定も行うことができる．

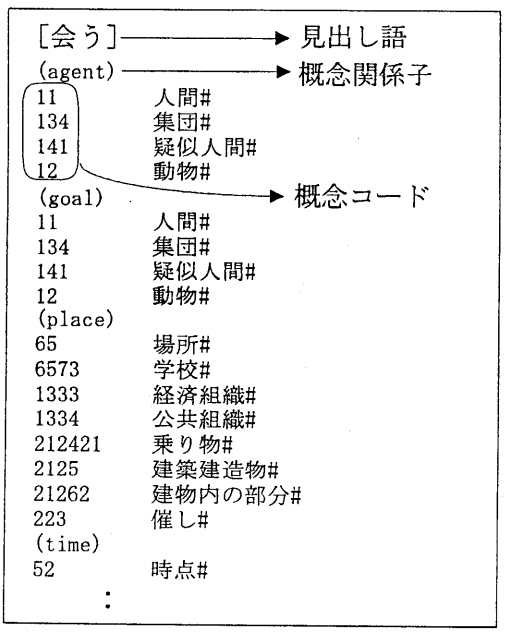

図１　格構造の例

しかし，ここで問題となるのは関係子の扱いで，特に図１中の

[agent 11(人間)], [goal 11(人間)]

の様に同一概念が複数の関係子によって共起されている場合である．本手法では，トライに登録されるキーの末尾に付加される終端記号#（endmarker）が格納される状態では，次の状態への遷移を表すnextフィールド（以後末尾フィールド）が未使用であることを利用し，関係子をビットフラグにして格納することにより解決している．

この様にして図１のデータにトライを用いた場合の構造を示す（図２）．尚，以後

agent = 0x01,
place = 0x02,
time = 0x04,
goal = 0x08

と定義する．図１と同様に[会う goal 11(人間)]を判定した場合，トライの状態番号０→１→２→３と遷移を辿り，関係子のフラグ(0x09)とgoal(0x08)の論理積が０でないことにより検索が

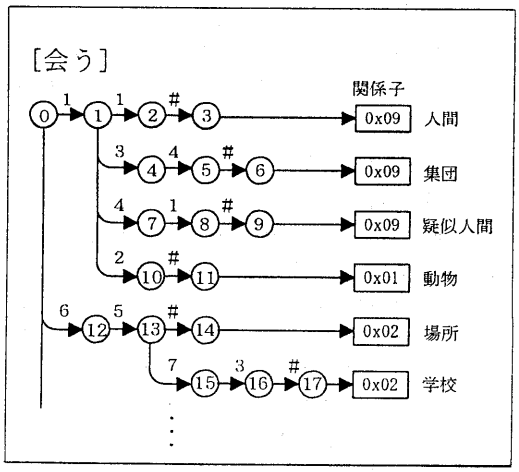

図２　トライによる格構造の例

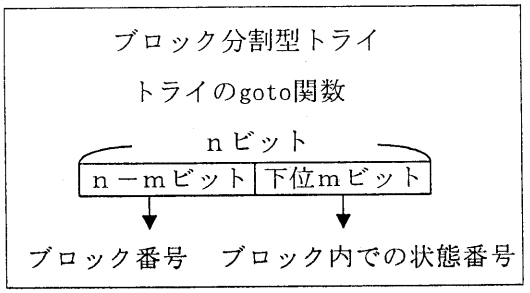

図３　ブロック分割型トライのgoto関数

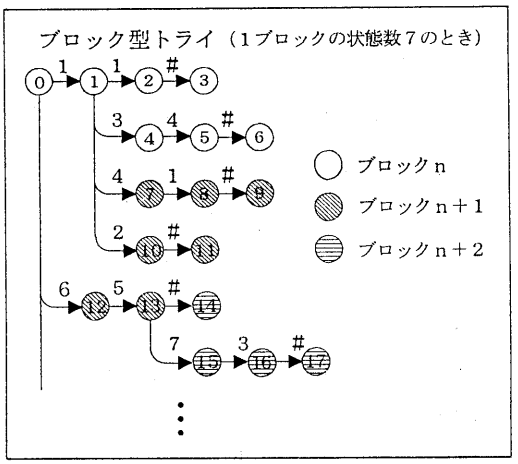

図４　ブロック型トライ

成功する．

もう一つの問題は，作成された各トライの管理の問題である．各見出し語の共起概念数は大小様々であるが，１トライに１ブロックを割り当てる手法では，容量が最大のトライに合わせてブロック容量を割り当てる為，未使用領域が多くなり非常に効率が悪い．そこで本手法では，次の状態への状態遷移を表すgoto関数を単なる配列の要素番号から以下の様に変更する（図３参照）．

状態遷移がｎビットで表されているとき，

（１）下位ｍビットをブロック内での状態番号に割り当てる．

（２）上位ｎ－ｍビットをブロック番号に割り当てる．

これにより，次の状態がどこのブロックに格納されていても遷移を辿ることが可能になる．従って一つのトライを複数のブロックで張り合わせて構成することが可能である（図４参照）．また，複数のトライを扱う場合にも，各トライの容量を管理する必要がなく，各トライの状態０の格納位置を記憶するだけで全てのトライを参照できる[8]．

容量の面でも，ブロックの容量を小さくすることで，未使用領域を最小限に抑えることができる．

以上の構造を用いることで，トライの特長の入力概念数に左右されない高速な検索が可能になると思われる．よって次節では本データ構造の構築及び検索アルゴリズムを示す．

３．構築と検索アルゴリズム

３．１　構築アルゴリズム

分類トライは前節で示したブロック型トライを用いて構築される．ブロック型トライでは新規状態の割り当てもgoto関数と同様の手法に変更されるが，アルゴリズム的には通常のトライとの差異を意識する必要はなく，構築アルゴリズムは基本的にはトライへのキー追加アルゴリズムに準じる．よってここでは変更部分を中心に説明する．

[定義]

関数*g*はgoto関数を表す．

見出し語の表記情報をHとする．

Hから分類トライの格納ブロックを返す関数indexを導入する．

共起概念数をKMAXとする．

共起概念KEY_k(k=1, 2, …, KMAX)の概念コード文字列を$code_k = k_0k_1\cdots k_nk_{n+1}$とする．

共起概念KEY_kの概念関係子を$Kflag_k$とする．

POSは入力文字列の現在の位置を示す．

STATEは状態番号を表す．

現在の全ブロック数を保持するNBLKを０に初期化し，全ての見出し語に対し以下の手続きを行う．

[手続きM_CTRIE]

手順（１）：{分類トライの更新}

NBLK←NBLK+1, k←0;

index(H)にNBLKを登録する．

手順（２）：{分類トライの作成}

k←k+1;
k ＞ KMAXならばの分類トライの作成終了で
return(TRUE);
手順（３）：{キーの検索}
手順（３－１）：{初期化}
POS←0, STATE←0;
手順(３－２）：{遷移の判定}
g(STATE, k_{POS})がfailならば
残りの$k_{POS}\cdots k_n k_{n+1}$をトライに追加し，
末尾フィールドに$Kflag_k$を格納し，追加終了
で手順（２）へ戻る．
そうでなければ手順（３－３）へ
手順（３－３）：{状態遷移}
POS ＞ n+1ならばキーは登録済みで
$Kflag_k$と末尾フィールドの論理和をとり
末尾フィールドに格納し，追加終了で
手順（２）へ戻る．
そうでなければ以下を実行する．
STATE←g(STATE, k_{POS});
POS←POS+1;
手順（３－２）へ戻る．

（アルゴリズム終）

［構築例］

図５の上のトライに
[agent 12], [goal 11]
を追加する場合の例を示す．
[agent 12]の追加
（１）状態０→１と遷移を辿ったところで，検索が失敗する．
（２）図５の下のように状態１０，１１の遷移を作成する．
（３）末尾フィールドにagent(0x01)を格納する．
[goal 11]の追加
（１）状態０→１→２→３と遷移を辿り，末尾フィールドを得る．
（２）末尾フィールド(0x01)とgoal(0x08)の論理和をとり(0x09)，末尾ノードに格納する．

３．２　検索アルゴリズム

本手法の検索アルゴリズムは基本的にはトライの

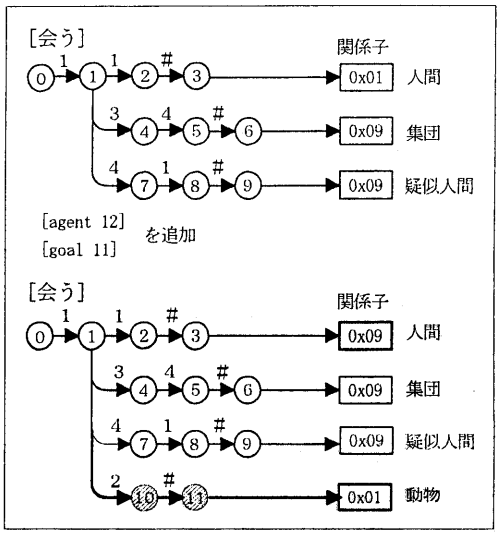

図５　構築例

キー検索アルゴリズムに準じる為，ここでは相違点を中心に説明する．

［定義］

分類トライの先頭ブロック番号index(H),
概念関係子Kflag,
検索概念の概念コードcode ＝ $k_0k_1\cdots k_nk_{n+1}$
を引数とする関数
search(index(H), Kflag, code)と表す．
返値は分類トライ上で[Kflag code]の共起判定を行い，概念群Ｋ中に含まれているならばTRUEを，含まれていない場合には，検索できた階層の深さを返す．
[関数search(index(H), Kflag, code)]
code ＝ $k_0k_1\cdots k_nk_{n+1}$を分類トライから検索する．
手順（１）：{初期化}
index(H)の分類トライを読み込む．
POS←0, STATE←0;
手順(２）：{遷移の判定}
g(STATE, k_{POS})がfailならば検索失敗で
return(-POS);
そうでなければ手順（３）へ
手順（３）：{状態遷移}
POS <= n+1ならば
STATE←g(STATE, k_{POS}), POS←POS+1, 手順（２）へ戻る．

そうでなければ手順（４）へ

手順（４）：{概念関係子の判定}

末尾フィールドとKflagとの論理積が０でなければ検索成功でreturn(TRUE);

そうでなければ検索失敗でreturn(-POS);

（アルゴリズム終）

[検索例]

図５下のトライにおいて”会う agent 人間”，”会う agent 組織”の検索例.

手続きsearch(index(会う), agent, 11);

手順（１）

index(会う)より分類トライを取得.

手順（２）

分類トライで11を検索する．状態０→１→２→３と遷移を辿り，末尾ノードを得る.

手順（３）

末尾ノード(0x09)とagent(0x01)の論理積が０でないため判定が成功し，return(TRUE);

手続きsearch(index(会う), agent, 133)の場合，手順（２）において状態０→１→４と遷移を辿ったところで失敗し，検索できた深さ２を返す.

４．　評価

４．１　理論的評価

提案したデータ構造を構築した場合の，キーの検索・更新の最大時間計算量（worst-case time complexity：以後，計算量と略記）を求める．本構造はトライ構造の為，検索の計算量はトライに準じ，検索キーの長さkのとき検索の計算量は$O(k)$となる．従って検索時間は，見出し語の共起概念数には影響されない.

入力記号の要素数をeとすると，トライにおいて１つの遷移の追加は$O(e)$となるため，キーの追加の計算量は$O(k \cdot e)$となる．index関数への登録に関しては，index関数の実装手法によって異なる為，ここでは評価しない．よってキー追加の計算量は$O(k \cdot e)$となる.

同様に本データ構造の最大辞書容量をトライを基にして求める．トライの容量が最大になるのは，登録キー群が全く共通接頭辞を持たない場合であり，リスト型トライの辞書容量は(総キーワード長S+キー総数K+終端記号分1)×(１つの遷移を構成する構造体の容量L)になる.

４．２　具体的評価

４．２．１　共起概念数による検索時間の比較

ここでは，見出し語の共起概念数の大小による本手法と線形探索の場合の検索時間の比較を行う．実験用データにはＥＤＲの概念辞書から抜粋した約７００概念（概念コード長１～９，平均５．３）（以後概念データと呼ぶ）からなる概念階層を使用した．実験は，概念データから指定数抜き出して作成した共起概念に対して概念データ抜き出して作成した共起概念に対して概念データの全概念の検索を行い，その平均時間を求めることにより行った.

共起概念数を横軸に，本手法と線形探索の検索時間をグラフにしたものを図６に示す.

図より，線形探索の場合は概念数に比例して検索時間が増大しているのに対し，本手法では見出し語の共起概念数によらずほぼ一定時間であることが判る．理論的に見ても，共起概念数ｎに対し線形探索の検索時間は$O(n)$で４．１節と併せ

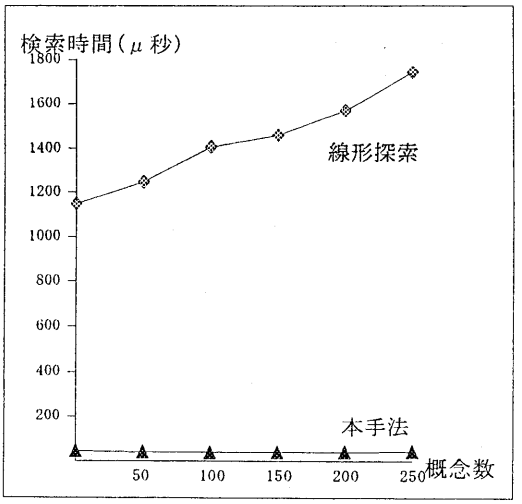

図６　検索時間の比較のグラフ

表1　平均検索時間

総検索回数　1,163,340回

平均検索時間（μ秒）

本手法	線形探索
119	1,285

表2　辞書容量

	容量(Kbyte)	比率
線形格納手法	344.3	−
分類トライ	724.1	2.1

て，この実験結果は理論的評価にも合致する．

以上より，本手法は概念数が増大しても変わらず高速な検索が可能であると云える．

4．2．2　総合評価

本研究室で作成した約1，700件の見出し語（動詞，形容詞，形容動詞）による格構造解析用共起辞書（共起概念数1～86，平均9．5）に本手法を用いた場合での実験データを基に総合的な評価を行う．

まず検索時間については，index関数をトライで実装し，2．での共起判定の形式で，4．2．1と同様に概念データの全ての概念について検索を行い，その1回あたりの平均検索時間を求めた．従って検索時間は，index関数の検索時間と辞書読み込み時間を含めた実際の検索時間になる．実験結果を表1に示す．

4．2．1で共起概念数が多い場合に，本手法が有利であることを示したが，この結果より全検索手順を通しての検索においても約10倍高速であり，本手法が概念を高速に検索する上で有効であることが示せた．

次に実験により作成された辞書の容量を表2に示す．表中の比率は，線形格納手法に対する容量比を表す．表より，容量比約2．1倍であるが，これは通常の（ノードの圧縮を行わない）トライ構造にしては比較的小さく収まっているといえる．要因としては概念コードが特性上，共通接頭辞が多く含まれていることと，概念関係子をフラグ化することで同一概念の吸収を行っていること，ブロック型トライの利用により未使用領域を最小限に抑えていることによると考えられる．概念検索においては高速性が最重要であり，その意味からもこの程度の辞書容量ならば充分許容範囲であると思われる．

5．　まとめ

本稿では概念検索を高速に行う手法として，トライ構造を用いる手法を提案した．また実験によりその有効性と理論的妥当性を確認した．

今後は，本手法を格文法による日本語解析，同音語判定，同型語の読み分け等に利用していく予定である．これらの場合には，トライに概念関係子以外のレコードが付加されることも予想され，容量の圧縮と共に効率化を図りたい．

また，本手法の動的構成法の実現により概念辞書の保守ツールとしても利用していきたい．

参考文献

（1）大島，阿部，湯浦，武市：”格文法による仮名漢字変換の多義解消”，情処学論，Vol 27，No. 7，pp. 679-687（1986）．

（2）宮崎，大山：”階層的単語属性を用いた同型語の自動読み分け方”，信学論（D），J68-D，3，pp. 392-399（1985）．

（3）高松，西田：”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，9，pp. 815-822（1981）．

（4）牧野，木澤：”べた書き文のかな漢字変換システムとその同音語処理”，情処学論，Vol. 22，No. 1，pp. 59-67（1981）．

（5）山本，久保田：”共起グループを用いたかな漢字変換”，情処学会第44回全国大会，pp. 4-11（1992）．

（6）EDR電子化辞書仕様説明書，（株）日本電子化辞書研究所（1993）．

（7）青江：”キー検索技法－トライ法とその応用”，情報処理学会誌，Vol. 34，No. 2，pp. 244-251（1993）．

（8）森本，入口，青江：”二つのトライを用いた辞書検索アルゴリズム”，信学論（D-II），Vol. J76-D-II，No. 11，pp. 2374-2383（1993）．