

7H-3

バイパス付き trie による  
S式集合の unification 検索

渋谷 克智 田中 譲  
北海道大学 工学部

1 はじめに

汎用的な知識ベースの構築のために、知識表現の基本的なデータ構造として用いられている S式集合をデータベース化することが提案されている[2]。当研究室ではこの S式のデータベースに S-expr Base[1]と名付け、3年程前から研究を続けている。今回の発表では、大量の S式集合に対し unification による検索を行う時の効率のよい方法について述べる。

2 S式の unification

S式を2進木で表現しそのリーフを変数、定数の2種類に区別する(以降、変数は\*を前に付けて示す)。変数には任意の S式を代入することができるものとする。また、1つの S式中で2ヶ所以上に同じ変数名が現れるとき、これらの変数には同じ S式が代入されなければならない。2つの S式を考え、それらに含まれる変数にそれぞれ適当な S式を代入することにより同一の S式にすることができるとき、それらの S式は(互いに)unification に成功するという。

S-expr Base に対する検索要求のありかたは様々考えられるが「あるキーとなる S式と unification に成功する S式を全て探す」という検索は柔軟な検索要求を提供する。例えば、Prolog のルールの形態をとる知識表現や非正規形の表の形を持つ知識表現はこの検索により扱うことができる。

3 S式の trie 化

unification 検索を実現するために trie を用いて S式を格納することを考える。trie は多数の string を格納するデータ構造であるのでまず S式を string で表現することを考える。Lisp で用いられている印字表現は S式を string 化した典型的な例であり、これを trie 化し unification 検索を実現することができる。しかしここでは cons セルを明記した prefix notation で記述し、これを用いることにする。すなわち string の各要素は cons セル(以降 & の記号で示す)、定数、変数の3種類のいずれかである。

いくつかの S式の集合を trie 化した例を図1に示す。S式の追加や削除は string の長さのオーダーで行われる。

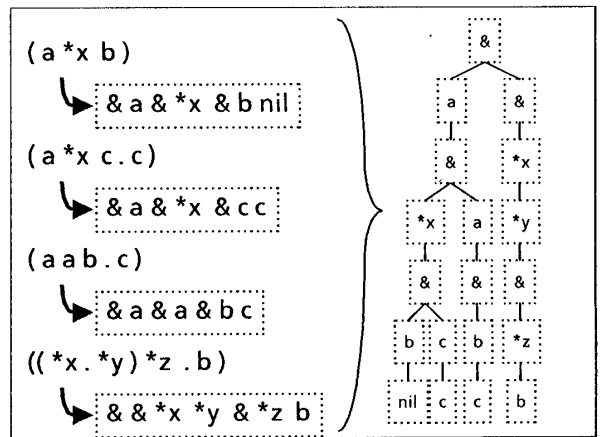


図1

1つの S式中に同じ変数が複数現れることがない場合、unification 検索は次の手順で行われる。まず、キーとなる S式を string 化する(以降 keystring と呼ぶ)。そして、trie のルートノードと keystring の第一要素を表1に示すマッチング規則に従ってマッチングを行い、成功するならばその下の各ノードと keystring の次の要素でマッチングを行う。それを繰り返し、keystring の最後の要素まで全てマッチングに成功した時の trie のノードが示す S式は unification に成功するものである。

	変数	定数	&
変数	○	○	○ その&が示すサブリストを読みとばす
定数	—	同じ定数なら○ さもなければ×	×
&	—	—	○

表1

表1で変数と & の組合せの項の「その & が示すサブリストを読みとばす」というのは keystring または trie 中の string を次々と読んでゆき、(出現した & の個数+1)個の変数または定数を読みとばすことにより実現される。

複数の同じ変数を持つS式を考慮した方法は、この方法の拡張として現在検討中である。

4 バイパス付き trie

trie中の&のノードについて、読みとばし先へのポインタ(のリスト)をあらかじめ持たせておくことによりデータベースへのアクセス回数を大幅に減らすことができると考えられる(図2)。

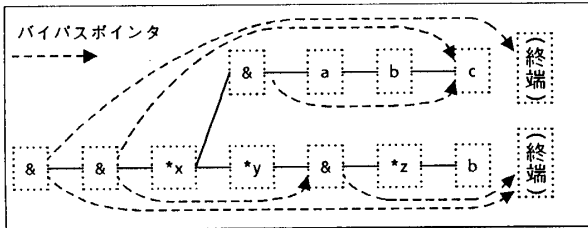


図2

&をtrieに格納する際にそのノードのアドレスをスタックにつみ、サブリストの格納が済んだ後にスタックからアドレスを取り出してバイパスポインタを登録するというを行う。これにより、オーダーを悪化させることなくS式の追加が行われる。

5 実験例

バイパスポインタを持たせることによりどの程度の速度向上が望めるかを調べるため、実際にプログラムを作成して実験を行った。プログラムは富士通S-3300+FACOM αのシステムの上でUTILISPを用いて書かれている。trieは連想リストを再帰的に利用することで実現してある。consセルのcar部、cdr部が分岐する確率をそれぞれcar-p、cdr-p、car部またはcdr部が分岐しない時に変数が現れる確率をvar-pとし、それらを設定してランダムなS式を作り実験に使用している。定数については使用するシンボルの個数を設定するようになっている。

100~10000個のランダムなS式をtrie及びバイパス付きtrieを用いてデータベース化し、ランダムなS式をキーとする検索を行った時の結果を図3に示す。横軸はデータベース中のS式の個数、縦軸は1000回の検索に要した時間の総和で、それぞれ10を底とするlogで示してある。データベース中のS式のcar-p、cdr-p、var-pはそれぞれ0.1、0.8、0.05で使用する定数は10種類、100種類、及び1000種類である(図中ATOMS=で示してある)。検索のキーとなるS式のcar-p、cdr-p、定数の種類はデータベース中のS式と同じであるがvar-pは0.2としてある。両対数グラフであることからこのグラフの傾きをaとするとS式の個数Nに対する検索時間は $O(N^a)$ であることがわかる。これを表2にまとめる。

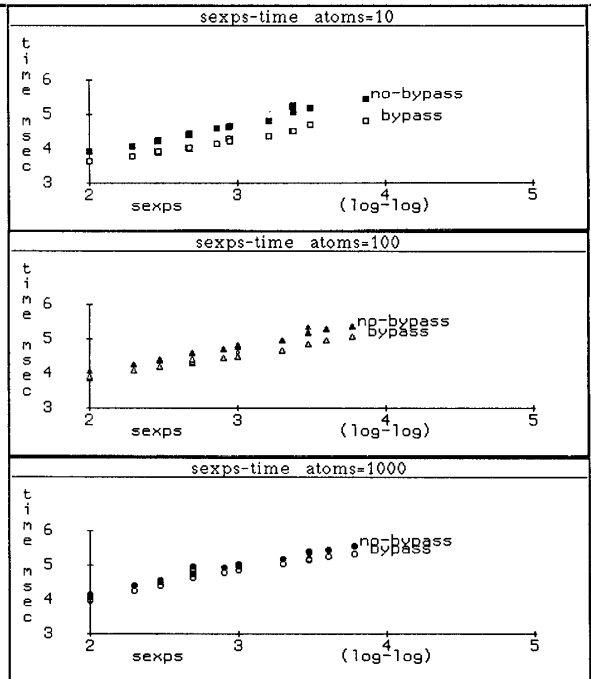


図3

ATOMS=	10	100	1000
バイパス無し	a=0.89	0.80	0.83
バイパス付き	a=0.66	0.66	0.77

表2

同じ実験でバイパスを用いることによって検索速度が何倍向上したかを示したグラフが図4である。

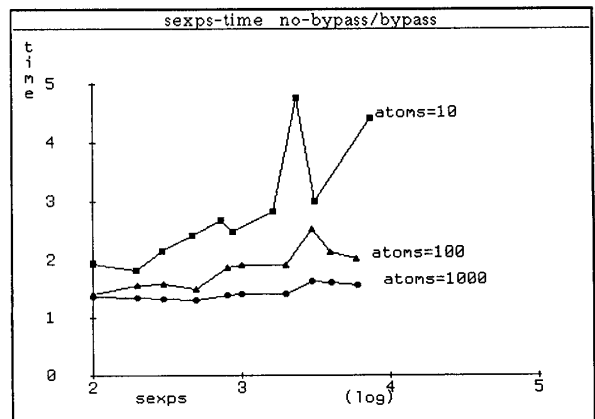


図4

参考文献

[1] 小林哲雄、田中讓: LISP BASE S式による知識表現を扱うDBMS  
データベース・システム 57-2(1987.1)

[2] 渋谷克智、田中讓: S-expr BaseにおけるS式の格納構造と検索機構  
情報処理学会全国大会講演論文集 pp1623-1624 (1987.9)