5G-1

重複する変数を考慮した 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[1]。当研究室ではこのS式のデータベースにS-expr Base[2]と名付け、研究を続けている。unificationによる検索をbypass付きtrieを用いて効率よく行う方法を第37回全国大会において発表したが[3]、その際課題となっていた重複変数を考慮した検索について以下に述べる。

2 bypass付きtrieによるS式のunification

consセルを明記したprefix notation(cons-expression[4])でS式をstring化する。そして複数のS式のstringをtrie化する。trieのノード中consセルに対しては、そのサブリストの読みとばし先ノードへのポインタ(のリスト)を持たせる。このポインタをbypass-pointerと呼ぶ。いくつかのS式の集合をbypass付きtrieにより表現した例を図1に示す。図中ではconsセルを&の記号で示し各stringの末端に終端記号$を付けてある。

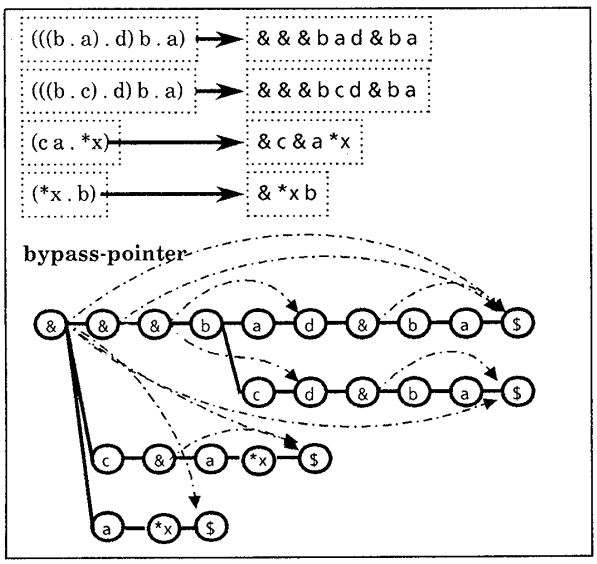

図1

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key-stringと呼ぶ)。trieのルートとkey-string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、成功するならその下の各ノードとkey-stringの次の要素でマッチングを行う。それを繰り返し、key-string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	○	○	○ bypass pointerを使いサブリストを読みとばす
定数	———	同じ定数なら○ さもなくば×	×
&	———	———	○

表1

3 重複する変数を持つS式のunification

1つのS式中に同じ変数が複数個存在する場合(重複する変数を持つ場合)、unificationに成功するためにはそれらの変数に同じS式が代入されなければならない。データベース中のS式に含まれる重複変数に対してはこのチェックは次の様に行うことができる。初出変数に対してはその変数と対応するkey-string中のノードを記憶し、次のマッチングに移る。既出変数に対してはkey-string中の対応するノードと先に記憶しておいた初出時の対応ノードとの間で表1に従いマッチングを行う。両ノードがconsセルである場合はそのサブリスト間でunificationに成功するかどうかがわかるまでマッチングを繰り返す。マッチングに成功する(サブリストがunificationに成功する)時には次のマッチングに移り、さもなければunification失敗とわかる。

4 重複変数処理の問題点と解決案

key-stringに含まれる重複変数に対しても同様の手法を用いることが考えられるが、trieの枝の分岐のため初出変数に対応するサブリストがわからなくなることがある。すなわちbypass-pointerで読みとばした間に分岐があると、どの枝をたどるとbypass先に行けるのかが不明になってしまう(図2)。

Unification-based retrieval for a large set of S-expressions including the duplicate variables
Katsutoshi SHIBUYA, Yuzuru TANAKA
Hokkaido Univ.

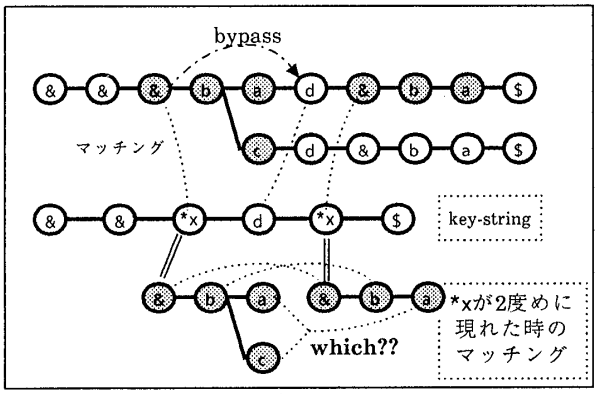

図2

そこでtrieの各ノードに親ノードを示すポインタ(parent-pointer)を付加し、子から親へたどり戻せるようにする。検索時には初出変数に対して対応するノードとbypassした先のノードを記憶する。その変数の2度目の出現からはbypass先からbypass元までをたどり戻して変数に対応するサブリストを得ることができる(図3)。

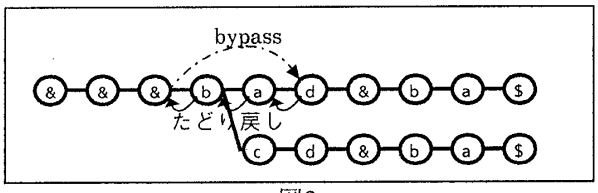

図3

この方法はtrie格納のためのstorageのオーダー、及び新しいcons-expressionをtrieに追加する時間のオーダーを変えることなく実現できる。

5 重複変数処理の改良

上記の方法によると、2度以上出現する変数に対してはbypass-pointerで読みとばしたノード全てをアクセスすることになり、bypass-pointerの効果が極端に下がってしまう。枝の分岐の無い部分では全てのノードをたどり戻さなくともサブリストを構成する経路がわかる。trieの性質より、分岐の無い(子を1つしか持たない)ノードが大多数を占め、複数の子を持つノードはごく一部であることが期待できる。そこで、子を1つしか持たないノードはたどり戻しの際に読みとばせるようにすることによりたどり戻し方式を改良する。具体的には各ノードのparent-pointerに「子ノードのリストの第1要素をたどってゆけばそのノードに到達できる」最も親となるノードの番号を格納しておく。もしそのノードが子ノードリストの第2要素以下に登録されているならばparent-pointerには親ノード番号を格納する。また、同じく各ノードにルートからの深さも格納しておく(図4)。

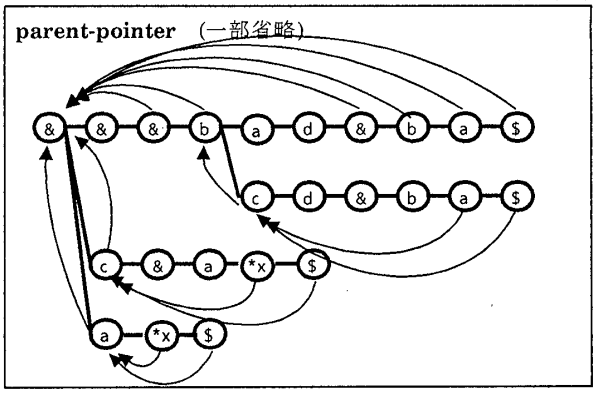

図4

たどり戻しの際にはbypass元のノードの深さを下回らない範囲でparent-pointerをたどってゆく(図5)。

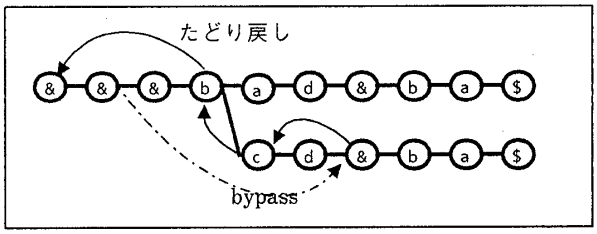

図5

新しいcons-expressionをtrieに追加する時に子ノードリストの先頭要素を変えないアルゴリズムを用いると、この方法もstorage及び追加時間のオーダーを変えることなく実現できる。

6 実験

現在富士通FACOM α上のUTILISPを用いてこれらの方式の実験を行っている。リーフの数が10~30のランダムなS式を200~1000個格納したデータベースに対し、重複変数を含むランダムなS式をキーにunification検索を繰り返すという実験を行った。このとき、たどり戻しの際にアクセスするノード数は上記改良により半分以下に減ることがわかった。

参考文献

[1] 小林哲夫、田中譲: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、田中譲:S-expr Baseにおけるs式の格納構造と検索機構
第35回情報処理学会全国大会講演論文集 pp1623-1624(1987.9)
[3] 渋谷克智、田中譲:バイパス付きtrieによるS式集合のunification検索
第37回情報処理学会全国大会講演論文集 pp1351-1352(1988.9)
[4] Charniak,E.,et al. : Artificial Intelligence Programming pp157-161
Lawrence Erlbaum Associates, Publishers