ソフトウェア工学 94－9
(1993. 9. 21)

述語論理型仕様を用いた部品検索とその評価

栗野 俊一‡ 松澤 裕史† 深澤 良彰§
‡ 日本大学 メディア科学研究室
† 日本アイ・ビー・エム株式会社 東京基礎研究所
§ 早稲田大学 理工学部

ソフトウェア部品の再利用を行なうためには、部品を蓄積したデータベースとそれを管理するシステムが必要である。このシステムの重要な機能は、部品という複雑な対象に対して、利用者の要求を満たす部品を正確にかつ高速に検索することである。

本論文では、まず、形式的仕様を用いる検索手法を示し、これを用いることによって、部品データベースシステムにおける、最も重要な機能である正確な検索が行なえることを示す。そして、この検索を高速に実行するための手法を示す。また、検索手法の応用として、部品の自動的な合成が検索と同じ枠組で実行可能であることを示す。最後に、これの手法を実験的なデータベースに適用した結果について報告する。

Components Searching by Formal Specification and its Evaluation

Shun-ichi KURINO‡, Hirofumi MATSUZAWA†, Yoshiaki FUKAZAWA§
‡Media Science Laboratory, Nihon University,
†IBM Research, Tokyo Research Laboratory,
§School of Science and Engineering, Waseda University

In order to reuse software components, a database which contains many components and its management system are necessary. Most important function of this kind of systems is to retrieve a component correctly and speedily which satisfies user's requests. Since programs are very complex objects as data in database, this function is indispensable. In this paper, a retrieval method which utilizes a formal specification is described. By this method, correct retrieval can be realized. Also, a speedy retrieval algorithm is shown. An automatic component synthesis method is reported as the application of this retrieval method. Lastly, experimental results are described.

1 はじめに

ソフトウェア工学の目的の一つに、生産性の向上があり、そのための手段として、部品化とその再利用が有効であることが知られている [1][2][3]。

既存部品の再利用を行なうためには、再利用に備えて作成された部品をデータベースに蓄積すると共に、データベースから部品を検索するためのシステムが必要である。特に検索を正確に行なうためには、次の 2 つの性質が成立していることが望ましい。

完全性 データベース中に要求する部品が含れていれば、必ず検索できる。

健全性 検索システムが出力した部品は、要求を満足する。

現在、キーワードによる検索システムや、ブラウザ [4] などが使われているが、利用者の要求する対象を正確に検索するという点で、これらのシステムは十分ではない。なぜならキーワードは、部品を表現するための記述力に乏しく、その結果として完全性や健全性をもちえない、あるいは、これらの性質を要求すると、貧弱なデータベースしか扱えなくなってしまう。さらに、ブラウザに到っては、ブラウザの設計者による部品の分類法を理解してなければ、検索そのものがままならない。

これに対して、我々は正確な検索を行なうシステムとして、一階述語論理型の形式的仕様を用いた検索システムについて提案してきた [5] [6]。

これは、部品とその形式的仕様の対が納められたデータベースを用意し、検索の際、要求する部品の仕様を与えることによって、検索を行なうものである。この手法においては、与えられた仕様とデータベースに納められている仕様から一階述語の命題を作成し、これが真であることの証明を試みることによって検索が行なわれる。したがって、検索の正確性は、証明系によって保証されることになる。

しかし、本システムのように定理証明を利用する場合には、どうしても決定不能の問題を避けることができない。この問題の対処法は、時間による打ち切りを行なうことである。これは、検索の完全性を失うことになる。しかし、検索の健全性に関しては他の検索システムとは異なり成立する。これが本手法の利点となる。

一方、この手法の欠点は、仮に検索対象がデータベースに含まれており、証明が成功し、部品が検索できる場合であっても、キーワード検索等と比較して、非常に効率が悪いということである。これは、この手法において時間による打ち切りが避けられないことを考えると、大きな問題となってしまう。

本論文では、述語論理型仕様を用いた部品検索手法を紹介し、さらにその欠点を補うための高速化手法について述べる。次に、同じ枠組によって、部品の自動合成が可能であることを示す。そして、さらにこれを unix のフィルタ系のコマンドからなるデータベースに適用した結果について報告する。

2 述語論理型仕様を用いた部品検索手法

筆者らの提案している部品検索手法について述べる。

2.1 仕様の形式

ユーザの要求する部品を正確に検索するためには、キーワードのように類似する部品が同一の記述になってしまうような検索情報をもつものは避ける必要がある。即ち、検索用情報の記述から一意に部品が決定できるような記述形式が必要になる。

本手法では、部品を指定するための検索情報として、部品の機能を記述した形式的仕様を用いる。

仕様の形式は一階の述語論理に基づく前提終了条件表記である。ここで、前提条件は、部品の入力となるデータが満たすべき条件であり、終了条件は、その結果として部品から出力されたデータが満たしている性質を表す。

以後、一階述語論理式で記述された部品 A の仕様の前提・終了条件のそれぞれを A_{pre}, A_{post} と記す。

2.2 汎用部品

データベースには、部品とその仕様を蓄積しておく。ユーザは、要求仕様として前提条件 R_{pre}、終了条件 R_{post} を入力するものとする。

このとき、データベース中のある部品 S の仕様が要求仕様に対して下記の関係になっているとき、部品 S は要求する部品に対して汎用であると言う。

- 前提条件に関して、汎用部品に対し要求仕様の方が厳しく、要求仕様を汎用部品の仕様が満たす
- 終了条件に関して、汎用部品に対し要求仕様の方がゆるく、要求仕様が汎用部品の仕様を満たす

そして、この汎用部品 S の仕様と要求仕様の関係を論理式で表したのが式 (1) である。ただし、$\rightarrow$ は包含関係を表し、$\wedge$ は論理積を表す。

$$(R_{pre} \rightarrow S_{pre}) \wedge (S_{post} \rightarrow R_{post}) \qquad (1)$$

この汎用部品は、要求仕様と必ずしも同一の部品ではないが、代用することのできる部品となる。

2.3 汎用部品の判定と検索法

データベース中の部品 S が、要求部品 R に対して汎用かどうかを検査するには、式 (1) が成立することを調べれば良い。すなわち、式 (1) の証明を試みれば良い。

そして、証明に成功した場合は部品 S が要求されている部品 R の汎用部品であることが言える。

この枠組では、部品データベースの中から汎用部品を選択するには、データベース中の全ての部品 S に対して、それぞれ式 (1) を立て、証明を試みればよい。そして、証明できた部品を出力することが部品検索となる。

従って、この検索法によって検索された部品は、次のような性質を満たす。

- 検索された部品は、必ず機能の上で要求を満たす。
- データベースに要求と一致する部品がなくても、代用可能な部品があれば検索することができる

一階述語の証明では、証明が可能な場合は必ず有限時間で停止するが、可能でない場合は証明が永久に止まらない可能性がある。そこで、時間による打ち切りを行なう必要がある。

この打ち切り時間の導入には次の二つの形式が考えられる。

- 個々の部品の証明に対し、打ち切り時間を決める。これは、各部品毎に証明の打ち切り時間を設け、一つずつ調べる方法である。
- 検索時間全体に、打ち切り時間を決める。これは、一つずつ調べると、永久に止まらない部品に当ってしまった時点で必ず打ち切られてしまうので、全ての部品を並行に証明する必要がある。

今回の実験では、後に述べる様に後者を選択している。これは、将来、並列で実行する証明に対して、スケジューリングを行ない、有望な証明 (例えば、論理式の長さが短いなど) を優先的に実行することにより、発見の可能性を高めることを考えているためである。前者に関しては、このような将来性はない。

3 検索の高速化

3.1 高速化の必要性

部品検索にかかるコストとして時間的なコストがある。要求されている部品の仕様の入力から目的の部品が見つかるまで、あるいは、発見できないという結論がでるまでにかかる時間を検索時間とする。ソフトウェアを部品を再利用して構築する場合、生産コストを下げるために、検索時間を低く抑える必要がある。

本手法では、述語論理式の証明が頻繁に試みられる。ところが、一般に述語論理式の証明は非常に時間を消費する。また、最悪の場合には、証明そのものが停止しないことを考慮すると、検索の高速化は実用の為の大きな条件となる。

3.2 ふるい落し

通常、多数の部品を持つデータベースにおいては、要求仕様を満たす汎用部品はごく一部であり、その他は、要求仕様とはほとんど無関係である。すなわち、検索のための証明は、データベース中の部品の大部分において失敗する。従って、この大半を占める無関係な部品に対して証明を試みることは、多大な時間を浪費することになる (図 1)。

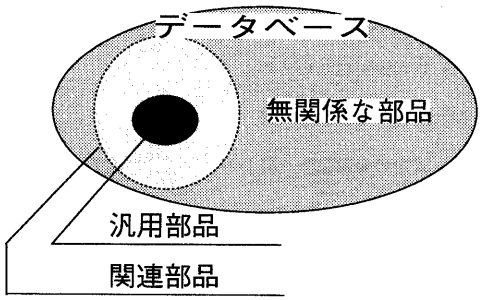

図 1: 部品の分類

そこで、検索時間を短縮するために、あらかじめ要求された部品と無関係と判断可能な部品のふるい落しを行なう。もし、このふるい落としが高速に実行可能であれば、残った部品に関してのみ厳密に証明を試みるだけで済むので、検索時間の短縮を図ることができる。

以下に、このふるい落しを行なう方法を述べる。

3.2.1 ふるい落しの原理

本システムでは、式 (1) の証明を試みることで、検索が行なわれる。ところが、データベース中に含まれるほとんどの部品に対して、この試みは成功しない。

そこで、明らかに証明に失敗するような部品をふるい落す方法を考える。そのために、まず、以下の二つの性質を満たすような、論理式 R^1_{pre}, S^2_{pre} の存在を仮定する。

性質 1 $R^1_{pre} \rightarrow R_{pre}, S_{pre} \rightarrow S^2_{pre}$ を満たす

性質 2 $R^1_{pre} \rightarrow S^2_{pre}$ の証明は $R_{pre} \rightarrow S_{pre}$ に対し決定的かつ高速に実行可能

すると、性質 1 から $R^1_{pre} \rightarrow S^2_{pre}$ が証明できなければ、$R_{pre} \rightarrow S_{pre}$ が証明できないことがわかるので、この部品 S はこの時点でふるい落して良いことがわかる。その一方、性質 2 から、この証明は決定的に行なえるため、証明に失敗する場合でも、安全に試みることができる。特に、式 (1) に比較して高速であれば、ふるい落とす意義がある。

従って、それぞれ R^1_{pre}, S^2_{pre} を求め、$R^1_{pre} \rightarrow S^2_{pre}$ の証明を試みる。そして、証明できなかった部品を検索対象から取り除く。

この作業がふるい落しとなる。

3.2.2 命題化操作

ふるい落しを行なうためには、R^1_{pre}, S^2_{pre} の存在を仮定しなければならない。ここでは、その存在と、具体的な論理式を求める方法 (命題化操作) を示す。

①命題化操作

述語論理式に対し、以下の操作を命題化操作と呼ぶ。

(1) 冠頭連言標準形にする

(2) $\exists$ を $\forall$ に置き換える

(3) 式中の変数出現を異なる変数記号に変える

この操作を R_{pre} に適用して得られた論理式 R^1_{pre} について $R^1_{pre} \rightarrow R_{pre}$ が成り立つ。また、$S_{pre} \rightarrow S^2_{pre}$ となる S^2_{pre} は、上の操作の **(2)** を **(2')** とすればよい。

(2') $\forall$ を $\exists$ に置き換える

②命題化操作後の述語を用いた証明

通常の述語論理式の証明は、導出原理 [7] を用いて、以下の手順で行なう。

(1) 節集合の形にする。

(2) 同じ名前の述語で正 (α) と負 (β) のリテラルをそれぞれ含むような 2 つの節の新しい組合せを探す。なければ、証明は失敗する。

(3) α と β が単一化できるか調べる。

(4) 単一化が可能なら、その単一化代入 θ を求める。

(5) 2 つの節を合わせ、α と β を取り除く。この結果、空節になれば、証明は成功する。

(6) (5) の節に θ を適用し整理し、これを新しい節として、節集合に加える。

(7) (2) へ行く。

この中で **(3)(4)(6)** の操作が最も時間を消費する。

ところが、命題化操作を施した述語に対しては、**(1)(2)(5)** と **(7)** の操作を行なうだけで良い。したがって、最も時間を消費する操作を省略して導出が行なえる。

例えば、本来の手順では、以下の (2) 式と (3) 式から代入 $\theta = \{x/y, f(x)/z\}$ を求めて、これを適用することにより、(4) 式が導出される。

$$\phi(x) \vee \frac{\psi(x, f(x))}{\alpha} \tag{2}$$

$$\neg \frac{\psi(y, z)}{\beta} \vee \tau(y, z) \tag{3}$$

$$\phi(x) \vee \tau(x, f(x)) \tag{4}$$

ところが、述語に命題化操作を施すと、**(3)(4)(6)** の操作は次のようになる。

(3) α, β の引数は、全て異なる変数になっているため必ず単一化できる

(4) θ は、すべて違う変数に対する代入となっている

(6) **(4)** より、これを適用しても式の形が変わらないので、適用する必要がない

従って、**(3)(4)(6)** の操作は行なう必要がなくなる。故に、上の例は以下のように単純化される。まず、(2),(3) 式は、命題化操作により次の (5),(6) 式のようになる。

$$\phi(x_1) \vee \frac{\psi(x_2, x_3)}{\alpha} \tag{5}$$

$$\neg \frac{\psi(x_4, x_5)}{\beta} \vee \tau(x_6, x_7) \tag{6}$$

代入 θ を求めると、$\theta = \{x_2/x_4, x_3/x_5\}$ が得られるが、(7) 式を導出する際に、適用する必要がない。

$$\phi(x_1) \vee \tau(x_6, x_7) \tag{7}$$

また、変数間に依存関係が存在しないので、述語名 (とそのアリティ) とその組み合わせ以外の可能性がなく、これらは有限なので、この手続きが必ず終了すること、すなわち、決定性を持つことがわかる。

① ②から、命題化によって得られた論理式は、性質 1, 性質 2 のいずれも満たしているので、これを利用することによって、ふるい落しが行なえる。

これは $R_{post} \leftarrow S_{post}$ についても同様である。

4 部品合成手法

4.1 部品の結合

本検索手法の特徴として、次のことが挙げられる。

(1) 部品の仕様に関して述語論理の証明で検索を行なっている。

(2) 部品の仕様を前提条件と終了条件に分けている。

この特徴を利用することにより、単一の部品を検索するだけでなく、複数の部品を結合した合成部品も検索できることになる。

この合成部品は、部品 A と部品 B があたえられており、かつ、A_{post} が B_{pre} を満たす場合、与えられた入力をとりあえず、部品 A に与え、その出力をそのまま、部品 B へ入力し、その結果を全体の出力とするものであり、合成部品 [A ▷B] と表記する。

図2にその概念図を示す。

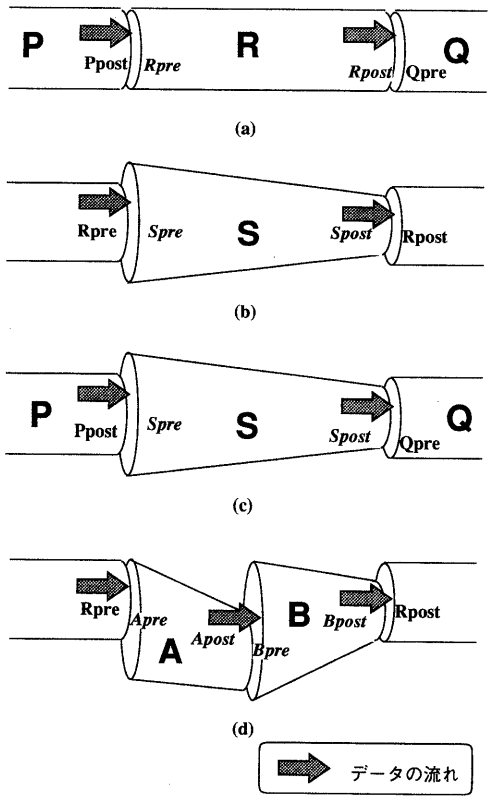

図 2: 部品の結合

図 (a) に示す部品 R は、要求仕様を実現する部品であり、他の部品との接続を考えると、部品 P の出力と部品 R への入力は一致し、部品 R の出力と部品 Q への入力が一致している。このとき、式 (8) が成立する。

図 (a) に対し図 (b) は部品 R の要求仕様をシステムに入力したとき、部品 S がその代用となれることを示している。部品 S は、本検索システムが検索対象とする汎用部品であり、要求仕様 R と部品 S の仕様の間には、式 (1) が成立する。

$$(P_{post} \leftrightarrow S_{pre}) \wedge (S_{post} \leftrightarrow Q_{post}) \tag{8}$$

図 (c) についても図 (a) と同様に考えることができ、部品 P からの出力を部品 S への入力へ接続し、部品 S からの出力を部品 Q への入力へと接続する。このとき、P、S、Q の仕様の間には、式 (9) が成立する。

$$(P_{post} \rightarrow S_{pre}) \wedge (S_{post} \rightarrow Q_{post}) \tag{9}$$

つまり、要求仕様の前提条件は、部品 P の出力に関する制約であり、これが P の終了条件とみなすことができる。同様に、要求仕様の終了条件は、部品 Q への入力に関する制約であり、これを Q の前提条件とみなすことができる。従って、部品が S の汎用部品であることを証明することは、部品 P に部品 S が結合できるかどうかを調べていることになる。

この概念を利用して、図 (d) を考えることができる。これは、要求を満たす部品が部品 A と B からなる合成部品 ($[A \triangleright B]$ と表記する) で実現できることを示している。このとき、A、B、R の仕様の間のには、式 (10) が成立する。

$$\underbrace{(R_{pre} \rightarrow A_{pre})}_{\alpha} \wedge \underbrace{(A_{post} \rightarrow B_{pre})}_{\beta} \wedge \underbrace{(B_{post} \rightarrow R_{post})}_{\gamma} \tag{10}$$

したがって、以下の手順により、$[A \triangleright B]$ を検索することが可能になる。

(1) 要求仕様の前提条件について、α の証明を試みる

(2) (1) が成功したならば、終了条件について $A_{post} \rightarrow R_{post}$ の証明を試みる

(3) (2) が成功したならば、その部品を出力する

(4) β 式が成立する部品 B を検索する

(5) (4) の部品 B の終了条件 γ の証明を試みる

(6) (5) の証明が成功したら、$[A \triangleright B]$ を出力する

従って、検索システムが証明すべき対象は、常に前提あるいは終了条件に関する証明を行なうことであり、本検索手法と同様の方法で行なうことができる。

4.2 合成部品の検索例

検索例をここで示す。要求仕様として以下の仕様を与える。ここでは説明のため自然言語を用いているが、実際の記述は述語論理による形式的な記述である。

R_{pre} 文字列のリスト

R_{post} 同一の文字列を含まないリスト

これに対し、データベース中に部品 A、B がデータベース中に存在すると仮定する。ここで、A、B は unix におけるフィルタコマンドである sort と uniq に相当するものである。

A_{pre} 文字列のリスト

A_{post} ソートされた文字列のリスト

B_{pre} ソートされた文字列のリスト

B_{post} 同一の文字列を含まないリスト

このとき、以下の式が成り立ち、合成部品 [A ▷B] が検索できる。

$$R_{pre} \rightarrow A_{pre} \tag{11}$$

$$A_{post} \rightarrow B_{pre} \tag{12}$$

$$B_{post} \rightarrow R_{post} \tag{13}$$

すなわち、要求仕様に対して、[uniq ▷sort] という合成部品が検索できた。

5 システムの実現

5.1 システムの構成

我々は、上記の手法を実現したプロトタイプを作成している。このシステムは、次のような構成になっている (図 3)。

部品データベース unix のフィルタ系のコマンドを部品とする実験的なデータベースである。これらの部品の入力はファイル (複数の入力ストリームの存在が仮定されている)、または標準入力からのバイトストリームであり、出力も同様である。

検索部 検索部は証明系と証明系のドライブ部の二つからなる。証明系は、新たには作成せず、既存の定理証明系 Otter[8] を利用した。また、証明系のドライブ部とは、与えられた要求仕様とデータベースの各仕様から、証明しなければならない命題をそれぞれ作成し、サブプロセスとしてを生成した Otter に、その命題を与える。そして、証明が終了するか、時間による打ち切りが生じるまでスリープする。なお、命題は前提条件、終了条件を別々に生成し、一方が証明できた時点で、合成部品の検索のサブプロセスを起動する。各プロセスは、unix 上のサブプロセスとなるため、互いに並行して証明を行なう。

仕様変換系 入力された要求仕様を、Otter の入力形式に変換する簡単な処理系である。

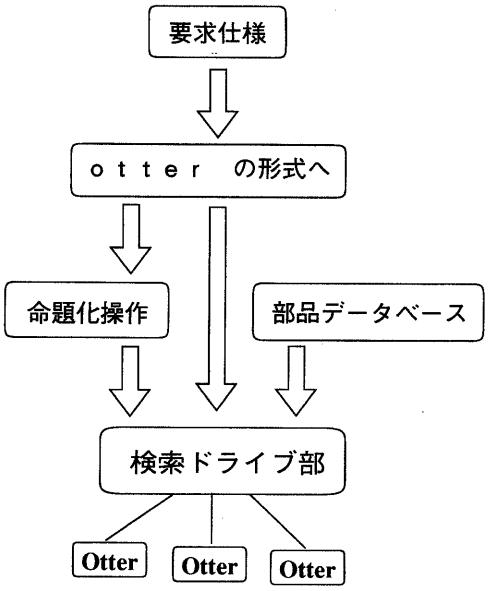

図 3: システム図

命題化操作系 入力された論理式から、命題化された論理式を求めるシステムである。

5.2 仕様記述言語

5.2.1 記述形式

要求仕様は前提・終了条件の二つの論理式を与える。ただし、今回は、対象がバイトストリーム (英数字からなるテキストファイル) であることから、利用者の便宜を図り、次のようなプリミティブを用意している。

line[n], line'[n] それぞれ入出力ストリームの n 番目の行を表す。ここで、line, line' のように 「'」 は入力の値に対する出力の値を表す。以下同様である。

word[n], word'[n] n 番目の単語を表す。

char[n], char'[n] n 番目の文字を表す。

NL, NL', NW, NW', NC, NC' それぞれ行数、単語数、文字数を表す。

比較演算 上記の各々に対して比較演算が可能である。

正則表現 文字列の集合を表現するために正則表現が利用可能である。

5.2.2 記述例

以下に仕様記述例を示す。

```
/* uniq : 連続した同じ行を一つにまとめる */

Pre:
        /* 前提条件はない */

Post:
        [ uniq(i,j) <-> [
                [ i = 0 AND j = 0 ]
                        OR
                [ line[i] = line'[j] AND
                        [ [     i > 1
                                AND
                        line[i-1] = line[i]
                                        AND
                                uniq(i-1,j)
                          ]     OR
                                uniq(i-1,j-1)
                        ]
                ]
        ]       AND
        uniq(N,N')
```

5.2.3 記述上の注意

今回のデータベースを記述するうえで、次の点に注意した。

オプションの扱い unix のコマンドはオプションの指定次第で異なる振舞いをする。特に、引数にプログラムの書けるコマンド (awk, sed など) があり、これを仕様化することは難しい。そこで、オプションのついたコマンドと、ついていないコマンドは異なる部品として登録した。とくに、典型的なオプションの付加されたコマンドに関して (例えば、子文字を大文字に変換する "tr a-z A-Z" とその逆など) も、それぞれ別々な部品として登録してある。

性質の保存 入力が部品の前提条件以外の性質をもっていた場合、通常は明示的には記述されていないが、その部品がその性質を保存したり、破壊したりすることがある。例えば、uniq は sort の出力の性質である行が整列しているという性質を保存するし、逆に sort は、uniq の出力の性質 (隣あった行は異なる) を破壊することがある。

そこで、保存する性質を利用したい場合は、もとの仕様の前提・終了条件の双方に保存する性質を付加した仕様と同じ部品を組にして異なる部品として登録した。

6 評価

以下に実験の結果とその評価について述べる。

6.1 実験環境に関して

この実験は以下のような環境で行なった。

CPU SunSparc Station Classic

Memory 16M

OS Solaris 2.1

Prover Otter ver 2.0

Time Limit 30 minutes

6.2 命題操作の効果

命題化操作により述語論理式の証明時間がどの程度、短縮されるかを実例を用いて評価した。以下に示す論理式を定理証明系 Otter を用いて証明した結果である。なお、実験は下線部の X を Q とした場合と P とした場合について行なった。

$$\exists x \exists x_1 \forall y \exists z \exists z_1 [\ (\neg P(y,y) \vee P(x,x) \vee \neg S(z,x)) \wedge (S(x,y) \vee \neg S(y,z) \vee Q(z_1,z_1)) \wedge (\underline{X}(x_1,y) \vee \neg Q(y,z_1) \vee S(x_1,x_1))\ ]$$

	通常の証明		命題化操作後の証明	
	所要時間	証明結果	所要時間	証明結果
$X = Q$	2.34 sec.	成功	0.22 sec.	成功
$X = P$	2.91 sec.	失敗	0.26 sec.	失敗

図 4: 命題化操作の効果

この結果から、命題化操作を用いて、ふるい落としを行なうことによって、検索時間の短縮が可能であることが解る。この例では、一桁程度速いが、他の例でも同程度の効果が見込まれる。したがって、後はこの操作の結果、どの程度無関係な部品をふるい落とすことができるかが問題となる。

6.3 サイズの影響

データベース中に含まれる部品の仕様の一つを選び、その仕様を要求仕様としてあたえる。この時、データベースのサイズによる検索時間の違いを調べた。この結果は以下のようになった。

サイズ	ふるい落とし	
	あり	なし
1	3.44 sec.	3.72 sec.
10	89.64 sec.	73.11 sec.
20	119.25 sec.	103.43 sec.
30	147.67 sec.	118.66 sec.

図 5: サイズによる検索の所要時間の比較

この結果、データベースのサイズに対する検索時間は、一次関数的に振舞うことが解る。これは、検索の為の証明を各データベース中の部品に対して、並列に行ない、汎用部品の発見ができた時点で打ち切りを行なっているからである。この結果、単独での証明にかかる時間にデータベースの部品数を掛けた時間が消費されることになる。

また、ふるい落としが思ったより効果を上げていないことも解った。これは、データベース中の仕様に、再帰的に定義された述語がある場合、これを命題化すると、恒真や恒偽の命題になってしまい。全く無関係な部品であっても、ふるい落しに失敗してしまうからである。

これは、証明を行なう際、導出の対象となる 2 つの節を、必ず、要求仕様とデータベース中の仕様からそれぞれ一つずつ選ぶことによって回避できると思われる。しかし、これを行なうためには、命題化された式専用の証明系を用意する必要がある。

7 おわりに

本論文では、部品の健全性のある検索方法とその高速化について述べた。また、この応用として自動的な部品合成が可能であることを示し、最後に、実際に適用した場合の結果を示した。

この結果、実験的なシステムでは、それなりに利用可能であるということが解った。しかし、複雑な部品が含まれていたり、検索の対象となるデータベースが大きくなった場合は、現在の状態では実用にならない。これは本質的に、証明系の速度に依存しているものと思われる。

今回は証明系の部分として汎用なもの利用したが、今回のように分野が限定されれば、それに特化した証明系も可能だと思われる。

逆に、あらかじめ頻繁に利用されるものを抽出し、データベースを洗練しておくという方法も考えられる。今回の実験の対象となった、 unix のフィルタ関係のコマンドも、数多くのコマンドを駆使するというよりも、少数のコマンドをうまく組み合わせるて使われる例である。したがって、むしろデータベースを拡充するよりは、合成部品の検索のほうを重視すべきかもしれない。

また、今回はライブラリに関しては何の工夫もなさなかったが、予めライブラリに納められた仕様に対し、前処理を行なえば、高速化が可能であると思われる。仕様記述の形式に関しても工夫の余地がある。例えば、今回は、オプションの有無に関しては、異なる部品として扱っている。この結果、検索の利用者からみると、あたかもオプションの自動設定までおこなったかのように感じられる。しかし、その一方これらの存在が、データベースの肥大を招き、検索時間に多大な影響を及ぼしている。したがって、本来の機能仕様とオプション仕様を分離する記述を工夫し、それに対応した検索法を用意することが考えられる。部品の保存する性質に関しても、同様なことがいえる。特に部品合成を行なう場合、単に接続できるかどうかよりも、一つ前の部品の性質を次の部品が保存し、性質が蓄積されることが本質的なことが多い。しかしながら、この場合うまく合成できるかどうかは、データベースの構築者にかかっている。頻繁に利用されるような性質 (整列されているかどうかなど) に関しては、あらかじめ、リストアップしておき、それらの性質が保存されるかどうかを仕様の一部として含め、検索するような工夫が必要であろう。

参考文献

[1] James W.Hooper and Rowena O.Chester, *"Software Reuse"*, Plenum Press, 1991.

[2] 片岡雅憲著,「ソフトウェア・モデリング－ソフトウェア再利用のための設計パラダイム－」, 日科技連出版社, 1988.

[3] 古宮誠一・原田実,「部品合成による自動プログラミング」, 情報処理, Vol.28, No.10, pp.1329-1345, 1987.

[4] Adele Goldberg 原著, 相磯 秀夫 監訳,「SMALLTALK-80 －対話型プログラミング環境－」, オーム社, 1986.

[5] 松澤・栗野・深澤・門倉,「機能仕様と品質を用いた部品検索のための一手法」, 情報処理学会第 43 回全国大会予稿集, 5-175, 1991.

[6] 松澤 裕史、栗野 俊一、深澤 良彰、門倉 敏夫：部品検索システムの持つべき性質とその実現法の提案, 情報処理学会, ソフトウェア再利用技術シンポジウム論文集, pp.53-62, 1992.

[7] 長尾 真,「論理と意味」, 情報科学－ 7 , 岩波書店, 1983.

[8] William W.McCune, Otter 2.0 Users Guide, ANL-90/9, Argonne National Laboratory, Argonne. Ill., March 1990.