

形式的表記によるソフトウェア部品の検索手法に関する研究

情報システム学 9931076 馬 薫

指導教官: 渡辺成良 教授 西野哲朗 助教授

1 序論

ソフトウェア開発における生産性の向上のために、すでに開発済みのソフトウェア部品の再利用が有効である。再利用率向上のために提案されている膨大な数の部品を管理できるソフトウェアリポジトリシステム [1] において、管理されている数多くの部品の中から要求部品を検出する検索機構が必要となる。

部品を検索する際、記述力が高くかつ記述量が少なく、さらに部分的に一致する部品や類似する部品も検出できることが要求される。現在、自然言語で記述された仕様をキーワード方式の検索キーで検索する方法が主流であるが [2]、文字列の有無のみで一致を判定し、論理的な構造を考慮しないため、要求部品の本質を正確に表現できず記述力や読解性、実用性に関して多くの欠点を持ち、部品の検索法として満足できるものではない。その他にも図的表現 [3] や仕様記述言語を検索キーとする研究 [4] がなされている。しかし、図的表現では記述力に限界があり、対応できる部品が少ない。一方、仕様記述言語による検索キーでは、仕様を完全に記述するので部品を設計したのと等価になるため、両方とも部品の検索キーとしてふさわしくない。

上記の欠点を解消するためには、検索に形式的表記を用いる必要がある。また、ソースコードの再利用のみではなく、形式的仕様記述言語により記述された部品の仕様の再利用が将来重要になってくると考えられる。本研究は部品仕様の再利用も視野に入れ、キーワード検索とは異なる新しい部品検索機構を構築することを最終目標としている。そのために、登録部品は仕様記述言語 CafeOBJ で記述した仕様を持つものと仮定し、CafeOBJ を抽象化した検索キーと、登録部品の仕様との類似性および部分的な一致を評価する手法を提案する。

2 形式的表記と CafeOBJ

形式手法に基づいた記述を形式的表記といい、ある文字列が与えられた時、意味を離れて純粋に文法の見地から正誤が判別でき、かつ文法上の正誤のみでその文字列が正当な記述単位かどうかが決まるという特徴を持つ。

CafeOBJ は形式的表記の一種であり、代数仕様の思考体系に由来し、実行可能である。例として CafeOBJ で記述された銀行の当座預金の例を図 1 に示す。

3 CafeOBJ を抽象化した形式的表記

本研究で提案している検索キーと登録部品の仕様との類似性および部分的な一致を評価する手法 (以下本評価手法)

```

1. module CURRENT-ACCOUNT {
2.   protecting (NAME)
3.   protecting (INT-EX)
4.   class Account {
5.     name : Name
6.     balance : Int
7.   }
8.   signature {
9.     op debit : ObjectId Int -> Message
10.    op credit : ObjectId Int -> Message
11.  }
12.  axioms {
13.    var A : ObjectId
14.    var N : Name
15.    vars I I' : Int
16.    rule credit(A,I) < A : Account | name = N, balance = I' > =>
17.      < A : Account | name = N, balance = (I' + I) > .
18.    crule debit(A,I) < A : Account | name = N, balance = I' > =>
19.      < A : Account | name = N, balance = (I' - I) > .
20.    :if I < I' = true .
21.  }
22. }

```

図 1: CafeOBJ による銀行の当座預金の仕様

では、検索キーを記述するために、新たに CafeOBJ を抽象化した形式的表記 (以下本表記と略す) を導入している。さらにこの表記を強化するために、使用頻度が高くある程度まとまった機能を持つ関数を登録したライブラリを用意する。利用者が検索時必要に応じてライブラリ関数を検索キー取り込み記述を行う。ライブラリ関数は、本表記と異なる表現形式を取っている。

3.1 ライブラリ

3.1.1 ライブラリ関数の表現形式の全体像

ライブラリには関数の具体的なアルゴリズムを登録するのではなく、データの性質や入出力の関係だけを記述する。関数の機能を実現するには複数のアルゴリズムが存在し、やり方や人によっていろんな書き方ができる。しかし、関数の入出力の間に成り立つ依存関係や束縛関係は常に一定であるため、その関係さえ的確に表せば関数を特定できる。本研究では、関数のこの性質に着目し、データの性質および入出力間関係を演算子、識別子、関数などを使って表した関係式でライブラリ関数を表現する。こうすることで、具体的なアルゴリズムなしで関数を抽象化できる。

3.1.2 関係式の記述項目

ライブラリ関数を定義するための関係式には以下のような項目を記述する。

- 関数の適用前後での入出力リストの長さの変化
- 関数の適用前後での入出力の要素の包含関係
- 関数の適用前後での要素の順序の変化
- 関数の適用前後での入出力間の要素の大小関係
- 関数の適用前後での要素自体の変化

3.1.3 ライブラリ関数を表現するのに必要な記述子

ライブラリ関数を定義するための関係式およびデータの性質を表現するには、他のライブラリ関数、識別子および演算子という三種類の記述子を用いる。識別子には L_i で

表す入力リスト識別子や a_i, b_j, \dots で表すリストの要素識別子などがある。演算子には大小関係演算子、包含関係演算子などが含まれる。

3.1.4 ライブラリ関数の表現形式とその例

本研究のライブラリ関数の記述は以下の書式に従う。

```
Function : <関数の名前>
Input    : <引数の並び>
Output   : <出力>
Relation : <入出力間の関係式の並び>
```

ここで、引数と出力は前に定義した各識別子で表す。Relationにある関係式は他のライブラリ関数、または識別子それに一個以上の演算子の組合わせである。

この書式で作成したライブラリ関数の例を以下に示す。

- 最大値


```
Function : max
Input    : L(ai); 1 ≤ i ≤ n
Output   : b
Relation : b ∈ L. (1)
           b ≥∨ ai. (2)
```
- 最小値


```
Function : min
Input    : L(ai); 1 ≤ i ≤ n
Output   : b
Relation : b ∈ L. (a)
           b ≤∨ ai. (b)
```

3.2 本表記による検索キー

本表記は CafeOBJ の文法を部分的に継承し、モジュール単位で要求仕様を記述し、必要に応じて階層的に一つ一つのモジュールを書き、それぞれのモジュールがお互い参照可能である。

3.2.1 本表記の記述項目

本表記で検索キーを作成する際、以下の項目を記述する。

- モジュール単位で記述する
- モジュールの初期状態
- モジュールに変化をもたらす入力データ
- モジュールに変化をもたらす関数
- それらの関数によるモジュールの状態の変化の結果

CafeOBJによる記述と比べて、以下の項目が省略される。

- 関数の引数の種類と数に関する定義
- 関数の戻り値の種類と数に関する定義
- 変数などの定義
- モジュールの状態の変化に関係ない関数
- モジュールの状態の変化の過程と具体的な実現方法

3.2.2 本表記に用いる記述子

本表記に用いる記述子は、ライブラリ関数、利用者が定義した関数、識別子、演算子の四種類である。識別子は A, B, \dots で表す状態クラス識別子や a, b, \dots で表すデータ識

別子などがあり、演算子には大小関係演算子、算数演算子などが含まれる。

3.2.3 本表記の文法と記述例

本表記は以下の書式にしたがう。

```
Module : <モジュールの名前> {
State   : <モジュールの初期状態クラス>
Data    : <必要データ>
Action  : <状態変化をもたらす関数の働き> }
```

ここで、Stateにある初期状態クラスは定義済みの要素識別子と状態識別子の組合わせであり、Dataにある必要データはデータ識別子で表される。Action部では関数ごとに状態変化の式を一つずつ記述する。条件によってその関数がモジュールに対する影響が異なる場合、それぞれの条件での変化を一つずつの式で記述する。関数はライブラリ関数または利用者自身が定義する関数である。関数の引数は一個以上の状態クラス識別子または要素識別子とデータ識別子である。変化の結果は変化後状態クラス識別子と条件式で構成され、それぞれ要素識別子またデータ識別子と演算子の組合わせとなっている。

例として、本表記による銀行の当座預金という部品の検索キーを示す。定義した記述子を用い、必要な記述項目を文法に従って記述すると、例えば図2のように書ける。

```
Module CurrentAccount {
State : A(n,b)
Data  : m
Action :
  credit(A, m) => A'(n,b + m).
  debit(A, m) => A'(n,b - m) :if m < b.
}
```

図2: 当座預金の検索キーの例

4 本評価手法について

4.1 検索キーと仕様との比較方針

本評価手法では、部品の機能を Actionの部分で関数単位で記述し、部品仕様の関数とそれぞれ一致または類似を評価する。また、検索キーのモジュールの階層構造が、部品仕様のそれと異なった際、検索キーと部品仕様の両方を一つの階層に展開してから検索を行う。

4.2 一致および類似の評価基準

検索キーの Action部分と部品仕様の axioms部分のそれぞれの関数に関する式を比較し、以下の項目を満たせば、それらの式が等価と判断する。

- 関数の引数の数が等しい
- 関数の引数の型が等しい
- 関数によるモジュールの状態クラスの変化が等しい

検索キーと部品仕様との間ではモジュール、関数、状態クラス、入力データの名前の違いは無視するが、内部では異なるものと判断する。

また、以下のような相違点を類似として許容する。

- 四則演算の違い

- 不等号の向きの違い

4.3 検索キーと仕様との評価手順

検索キーと部品仕様の評価は以下の手順に沿って行う。

手順 1

検索キーの State に現れる状態クラスの種類 (要素識別子の数など) が部品仕様の class と var の部分の記述と一致する場合手順 2 へと進む。しかし、検索キーにしか含まれない識別子が存在する場合、モジュールの適用対象が異なると考えられるので不一致と判断し、評価を終了する。

手順 2

検索キーの Data 部にあるデータ識別子と同種類のデータ識別子が部品仕様の vars 部に存在するのなら手順 3 へと進む。そうでない場合不一致と判断する。

手順 3

検索キーの Action の中の各記述項目と部品仕様の中のものに対して、以下の操作を繰り返す。

- ライブラリ関数があれば、次節の方法に従う
- ライブラリ関数以外の関数では、一致の基準を満たせば一致
- =>の右側において、違いがあった場合、上で挙げた類似として許容できるものなら類似、そうでなければ不一致と判断する

手順 4

評価の結果、検索キーと部品仕様が一致すれば一致と、類似なら仕様中の差を利用者に提示する。

4.4 ライブラリ関数同士の評価

検索キーにライブラリ関数を使用されている場合、部品仕様にそのライブラリ関数に相当する記述のある部品仕様を探し、見つからない時、そのライブラリ関数に類似したライブラリ関数をライブラリから探し出し、それらの類似ライブラリ関数に相当する記述のある部品仕様を検索する。こうして得られる部品が元の要求に類似している部品として利用者に提示する。このように、ライブラリ関数同士の評価は、要求部品に類似する部品を検索するのに役立つ。

4.4.1 ライブラリ関数同士の評価方針

ライブラリ関数同士の評価は関係式単位で行われる。以下の相違点があった場合、それらの関係式が類似していると判断する。

- 否定記号の有無
- 不等号の向きの違い
- 論理和と論理積の違い
- 四則演算の違い

以上に挙げた違い以外に相違点があった場合、それらの関係式が不一致と判断する。

4.4.2 ライブラリ関数同士の評価手順と評価例

ライブラリ関数同士 (関数 A と関数 B とする) の評価は次の手順に従う。

手順 1

関数 A の Input にある入力データの性質が関数 B のそれと一致すれば次の手順に進む。そうでない場合、関数の適用対象が異なると考えられるので評価を中止する。

手順 2

関数 A の Output にある出力データの性質が関数 B のそれと一致すれば次の手順に進む。

手順 3

関数 A と B の Relation 部分にあるそれぞれの関係式に対して、下にある操作を繰り返し、一致また類似を判定する。

- 表現が同じなら一致
- 許容できる相違点なら類似
- それ以外は不一致

手順 4

手順 3 まで評価を行った結果、関数 A と関数 B に不一致の関係式が存在しない場合、関数 A と関数 B が類似していると判断する。

例として、前で挙げたライブラリ関数 max と min をこの手順で比較すると、お互い類似していると判断できる。

4.5 検索キーと部品仕様の評価例

本評価手法による検索キーと CafeOBJ による部品仕様との評価例として、前章で示した銀行の当座預金のモジュールを使用する (図 1 と図 2)。

1. 手順 1 では、検索キーの State 部分と部品仕様の 4~7 行を評価し、一致しているので手順 2 へ進む
2. 手順 2 では検索キーの Data 部分と部品仕様の 15 行を評価し、一致しているので手順 3 へ進む
3. 手順 3 では、Action 部分にある二つの関数 credit と debit それぞれに対して、部品仕様の axioms 部分の 16~20 行と評価する。今回の場合、二つの関数ともに引数の種類が一致している。また、=>の右側もそれぞれ一致している
4. 手順 4 で、部品仕様を検索キーの要求を満たしていると判断できる

5 考察

5.1 本評価手法の効果

数多くの部品を管理するソフトウェアリポジトリシステムから要求部品を検索するために満たすべき要件を、本評価手法が満たしているかを考察する。

5.1.1 記述力の高さと言語量

本評価手法は、CafeOBJ に基づき、さらに独自の演算子、識別子そしてライブラリ関数を定義した形式的表記で検索キーを記述するため、CafeOBJ よりも抽象度の高いレベルで部品の機能を記述できる。そのため、キーワード方式や図的表現による検索キーよりも高い記述力が得られた。また、仕様記述言語と比べ、記述力は低い、部品の検索キーとして十分だと考えられる。

また、部品モジュールの状態、入力データ、状態に変化をもたらす関数などの検索に必要な最小限の情報のみを検索キーに記述するため、その記述量は CafeOBJ などの仕様記述言語より少ない。しかしキーワード方式よりは多い。

5.1.2 階層的な表現

本評価手法では、要求仕様をモジュール単位でしかも複数の階層に分けて記述できるため、今日の部品開発形態に合致している。また、検索キーのモジュールの階層構造が部品仕様のそれと異なった場合、両方ともに一つの階層に展開した後、評価を行う。

5.1.3 部分的な記述と部分的な一致

本評価手法では、モジュールに変化をもたらす関数一つずつ記述し、検索も関数単位で行うため、部分的な記述では検索を行うことが可能であり、部分的な一致が判定でき、要求部品の一部を満たす部品を検索できる。

5.1.4 類似部品の検索

本評価手法では検索時、状態の変化をもたらす関数を記述した式において、四則演算と不等号の違いを類似として認めている。さらに、検索キーにライブラリ関数が使用されている場合、部品仕様とそのライブラリ関数に相当する記述のある部品仕様がないとき、そのライブラリ関数に類似したライブラリ関数をライブラリから探し出し、そしてそれらの類似ライブラリ関数に相当する記述のある部品仕様を検出することで、要求部品に類似する部品を利用者に提示する。

5.2 論理構造のみによる一致判定の利点と欠点

本評価手法は従来の自然言語によるキーワード方式の検索法とはまったく逆の立場に立ち、検索キーを形式的な表記で記述し、文字列の意味から離れて純粋に論理的な文法の見地から一致判定を行っている。このため、問題の規模や種類によってはキーワード方式のほうが簡単に要求部品を検出できたりする。しかし、本評価手法は部品の再利用において極めて重要だと考えられる類似部品の検出も可能で、それができないキーワード方式より優れている。

5.3 本評価手法の問題点と今後の課題

5.3.1 本表記法の完成度

現時点での本評価手法の完成度はまだ十分とは言えず、より複雑で規模の大きい部品に対応するために、さらなる識別子と演算子の追加やライブラリの充実、類似性評価アルゴリズムの強化が必要である。

5.3.2 検索結果の提示

検索の結果、要求部品に一致または類似する部品を検出した場合、それをどういう形で利用者に提示するかを検討しなければならない。類似している部品の場合、その類似が仕様中のどの部分であるかを示し、利用者がカスタマイズしやすい形で提示する必要がある。

5.3.3 検索対象の絞り込み

本研究が対象としているソフトウェアリポジトリシステムには膨大な数の部品が管理されている。そのためすべての部品仕様を比較対象にするのは効率的でない。管理されている部品のうち、まったく無関係な部品を比較の早い段階で除外する必要がある。

6 結論

本研究は部品仕様の再利用も視野に入れ、キーワード検索とは異なる新しい部品検索機構を構築することを最終目標としている。そのために、登録部品は仕様記述言語 CafeOBJ で記述した仕様を持つものと仮定し、CafeOBJ を抽象化した検索キーと、登録部品の仕様との類似性および部分的な一致を評価する手法を提案した。

この中で、検索キーを記述するために、新たに CafeOBJ を抽象化した形式的表記を導入した。さらにこの表記を強化するために、使用頻度が高くある程度まとまった機能を持つ関数を登録したライブラリも用意した。またそれらのそれぞれの表現形式と文法などを定義し、検索キーの作成手順及び作成例も示した。また、ライブラリ関数同士そして検索キーと部品仕様間のそれぞれの評価アルゴリズムを提案し、評価手順と評価例も示した。これらによって、本評価手法はキーワード方式における記述力と実用性に関する欠点のある程度解消できた。登録部品に CafeOBJ による仕様を持たせたのは、将来重要になると考えられる仕様の再利用も考慮したためである。現時点ではまだ一般化されていない形式的な仕様記述言語による仕様の部品への添付に近い将来一般的なものになると考えられるので、本研究の手法が利用者とシステムの負担になることはない。

今後は本手法の完成度を高め、考察した問題点を解決し、最終的にはキーワード方式に取って代わる検索機構として完成し、部品の再利用を通して部品の生産性をの向上に貢献したいと考えている。

謝辞

本研究を進めるにあたり、終始温かい御指導をいただきました渡辺成良教授、並びに西野哲朗助教授、織田健助手に厚く御礼申し上げます。

参考文献

- [1] 雲切 啓太, “多組織による共有を考慮した細粒度ソフトウェアリポジトリ”, 平成 11 年度修士学位論文, 1999
- [2] 堤富士雄, 篠原靖志, “キーワードを 2 次元平面に配置する文書検索システム”, コンピュータソフトウェア 15-4, pp2-15 1998
- [3] 大岡俊彦, 長谷英明, “メンタルモデルを機能表現に用いた関数検索システム”, 日本ソフトウェア科学会 FOSE'96, pp186-189 1996
- [4] 栗野俊一, 松澤裕史, 深澤良彰, “ソフトウェア部品検索における形式的仕様の活用法”, 日本ソフトウェア科学会 FOSE'94, pp129-136 1994
- [5] 中川中, 谷津弘一, “形式手法への誘い—代数仕様とその周辺”, CafeOBJ HomePage, 1996