データベースシステム 125－39
（２００１．　７．１８）

サンプルを用いた Visual Basic 関数検索システムの開発

掛下 哲郎　　川口 雄司　　釣本 勝之

佐賀大学 理工学部 知能情報システム学科
佐賀市本庄町 1 番地
kake@is.saga-u.ac.jp

大規模化・複雑化するソフトウエアの開発コストを低減するためには、ルーチンやモジュール等のコンポーネントを再利用することが不可欠である。このためにはリポジトリから所望のコンポーネントを検索する機構が必要になるが、我々は仕様に基づいた検索機構としてサンプルを用いる方法を提案している。本稿ではサンプルを用いた検索方式の具体例として Visual Basic (VB) 関数を対象とする検索システムを開発する。VB 関数の仕様は引数/戻り値のデータ型および値の組を要素とする (無限) 集合で表現できる。利用者がこれらの要素を用いて (有限の) サンプルを記述すると、システムは該当する関数を検索する。利用者のサンプル記述を支援するために要素辞書、要素辞書内での要素検索、重複度別分類の各機構を実現した。これにより、利用者はリポジトリに格納された任意の関数を特定するサンプルを構築できる。

A Sample-Based Retrieval System for Visual Basic Functions

Tetsuro Kakeshita　　Yuji Kawaguchi　　Katsuyuki Tsurimoto

Department of Information Science,
Faculty of Science and Engineering, Saga University

It is essential to reuse software components, such as routine and module, in order to reduce software development cost. Component retrieval mechanism becomes necessary to realize component resuse. We have proposed a sample-based approach for this purpose. In this paper, we develop a retrieval system for Visual Basic functions using the sample-based approach. Specification of a VB function is represented by an infinite set of two types of elements: (1) data type of an argument or a return value and (2) tuple of actual argument values and corresponding return values. A user specifies a finite sample using these elements. Then the system retrieves functions using the sample. We also implemented user support mechanism for sample construction. These include element dictionary, element retrieval in the dictionary and weight calculation of each element. These mechanism enables to construct a sample to identify an arbitrary function in the repository.

1　まえがき

ソフトウエアの開発コストは非常に高い。例えば、Windows 95 は 85,000 Function Point (FP) の規模を持つが、10〜20 万円/FP の開発コストを考慮すると少なくとも 85 億円の開発費が必要になる [1]。金融ビッグバンに伴う銀行のオンラインシステムの開発費用は 100 億円規模であり、セブンイレブンの最新 POS システムの開発・導入費は 600 億円規模になる。このような開発費の高騰を抑えるためにソフトウエアの再利用技術 (コンポーネントウエア) が注目されている [2]。

コンポーネント (ルーチン、モジュール、クラス、デザインパターン等) を再利用するためには、リポジトリに格納されたコンポーネントを検索する必要がある。このとき、利用者の利便性を考慮するとコンポーネント仕様に基

づいた検索方式が望ましい。

我々は仕様に基づいたコンポーネント検索方式としてサンプルを用いた方式を提案している [3]。本方式では集合を用いてコンポーネント仕様を表現する。利用者は集合内の要素を用いてサンプルを作成する。作成されたサンプルはリポジトリに対する検索条件に対応し、検索するコンポーネントの概略仕様を表現する。また、利用者がサンプルを作成する際の支援機構として、要素辞書とサンプル作成支援機構が提供されている。利用者は、要素辞書に格納されている要素だけを用いて、任意のコンポーネントを特定するサンプルを構成できる [4]。また、サンプル作成支援機構を用いると、コンポーネントを特定するサンプルのサイズ (要素数) を平均的に $\log n$ 程度 (n はコンポーネント総数) に抑えられる。

本稿では、サンプルを用いた検索方式の具体例として Visual Basic (VB) 関数を対象とする検索システムを開発した。VB 関数の仕様は、(1) 引数や戻り値のデータ型 (入出力データ型)、(2) 引数/戻り値の値で構成される組 (入出力値) を要素とする無限集合で表現される。サンプルはこれらの要素に正例/負例の別を指定した有限集合によって指定される。検索の際には、サンプル内の全ての正例を仕様に含み、1 つの負例も含まない VB 関数が検索される。本検索システムでは、上記の検索機構の他にも、(1) リポジトリに対する VB 関数の追加と削除、(2) 要素辞書の管理、(3) サンプル作成機構、(4) リポジトリ内の任意の VB 関数の差を示す要素の提示、等の機能を実現した [6, 7]。実現に当たっては Microsoft Access 97 を用いた。

本稿は以下のように構成されている。2 節および 3 節ではサンプルを用いた検索方式、要素辞書、サンプル作成支援機構について紹介する。4 節では集合を用いて Visual Basic 関数の仕様を定義する。5 節では VB 関数検索システムの機能および利用者インタフェースについて示す。6 節では検索システムのモジュール設計およびデータ構造設計について述べる。

2　サンプルを用いた検索方式

サンプルを用いたコンポーネント検索方式では、集合を用いてコンポーネントの仕様を表現する。コンポーネント c_i の仕様を表現する集合を $g(c_i)$ と呼ぶ。

リポジトリに格納されているコンポーネント集合を $C = \{c_1, \cdots, c_n\}$ とする。要素 $e \in \cup_{i=1}^{n} g(c_i)$ に対して、$\langle e \rangle$, $\langle \overline{e} \rangle$ をリテラルと呼ぶ。以下、紛らわしくない限り $\langle e \rangle$ を e、$\langle \overline{e} \rangle$ を $\overline{e}$ と表記する。リテラル e を正例、$\overline{e}$ を負例と呼ぶ。サンプル S はリテラルの集合である。サンプル S とコンポーネント $c \in C$ について、以下の条件が全て満足されるとき S は c を検索するという。

- 全ての正例 $e \in S$ が $e \in g(c)$ を満足する。
- 全ての負例 $\overline{e} \in S$ が $e \notin g(c)$ を満足する。

サンプルは複数リテラルを用いることで AND 条件を表し、負例を用いることで NOT 条件を表す。

コンポーネント集合 C とサンプル S について、$Set(C, S)$ を以下のように定義する。

$$Set(C, S) = \{c | c \in C, S \text{ は } c \text{ を検索する。}\}$$

$Set(C, S)$ は、S を用いた C に対する検索結果に対応する。$Set(C, S) = \{c\}$ となるとき、S はコンポーネント c を特定するという。

サンプルを用いたコンポーネント検索は以下の手順に従って行われる。ここで、太字で示されたステップは利用者が行う。

(1) サンプル S を空にする。

(2) コンポーネント集合 C をリポジトリ内の全コンポーネントとする。

(3) $|C| \neq 1$ である限り以下の処理を繰り返す。

(3-1) 利用者がリテラル e (または $\overline{e}$) をサンプルに追加する。

(3-2) $C \leftarrow Set(C, \{e\})$ (または $C \leftarrow Set(C, \{\overline{e}\})$)

3　サンプル作成支援機構

2 節で示したコンポーネント検索アルゴリズムを用いただけでは、ループを繰り返してもコンポーネント集合 C の要素数が減少しなくなる場合がある。また、検索対象コンポーネントを特定するために必要なサンプルへのリテラル追加回数が非常に大きくなることもある。サンプル作成支援機構を用いると、リポジトリ中で任意のコンポーネントを特定するサンプルに含まれるリテラル数 (サンプルサイズ) が高々 $n-1$ かつ平均的には $\log n$ となる。ここで、n はコンポーネント総数である。

サンプル作成支援機構として要素辞書、要素検索、要素の重複度が提供されている。要素辞書はサンプルサイズを高々 $n-1$ にするために用いる。要素検索と要素の重複度を要素辞書と組み合わせて用いると、平均のサンプルサイズを $\log n$ にできる [3]。要素検索は検索対象コンポーネント数が比較的多い場合に用いる。一方、要素の重複度は検索対象コンポーネント数が少なく、候補コンポーネントが互いに類似している場合に有効である。

3.1 要素辞書

コンポーネント集合 $C = \{c_1, \cdots, c_n\}$ の要素辞書 D_C は、任意の $c, c' \in C$ について、以下の条件を満足する $\cup_{i=1}^{n} g(c_i)$ の部分集合である。

$$g(c) \neq g(c') \text{ ならば } g(c) \cap D_C \neq g(c') \cap D_C$$

要素辞書 D_C には以下の性質がある [4]。

1. D_C には互いに異なる任意の 2 つのコンポーネントの差を表す要素が必ず含まれている。
2. D_C の要素のみを用いて任意の $c \in C$ を特定するサンプルが構成できる。
3. コンポーネント集合 C において、各コンポーネントの仕様が互いに異なるならば、要素数が高々$n-1 = |C|-1$ の要素辞書 D_C が多項式時間で構成できる。
4. 要素数最小の要素辞書構成問題は NP 困難である。

以上より、コンポーネント検索を行うサンプルは要素辞書内の要素を用いて作成する。また、利用者が指定した 2 つのコンポーネント間の違いを説明するための要素も要素辞書から検索する。

リポジトリにコンポーネントが追加または削除されるたびに、要素辞書の変更が必要になる。我々はこれを多項式時間で行うアルゴリズムを提案している。

3.2 要素検索

要素検索は要素辞書内でサンプルへの追加に適した要素を検索するための機構である。システムは利用者に対して正例、負例いずれのリテラルを追加した方が候補コンポーネントの絞り込みが進行する可能性が高いかをアドバイスする。利用者はアドバイスに従って、正例 (または負例) を要素辞書から検索するための条件を指定する。システムは利用者が指定した条件を用いて要素辞書内で検索を行い、検索された要素をサンプルに追加する。

システムは、以下のアルゴリズムに従って利用者にアドバイスを与える。m 個の候補コンポーネントが検索されているとする。サンプル S に正例を追加することで候補コンポーネント数が m' に絞り込まれたとしよう。$\alpha = m'/m$ は、S に正例を追加した場合の候補コンポーネント数削減割合 (期待値) である。一方、サンプル S に負例を追加することで候補コンポーネント数が m' に絞り込まれた場合には、$\alpha = 1 - m'/m$ によって α の値を推定できる。α が $1/2$ より小さいならば、正例を追加するよう利用者にアドバイスする。逆に、$\alpha > 1/2$ ならば、負例を追加するよう利用者にアドバイスする。

アドバイスに従って利用者がサンプルにリテラルを追加すれば、n 個のコンポーネントを含むリポジトリに対して、平均的に $\log_2 n$ 個のリテラルを含むサンプルを構成すれば、任意のコンポーネントを特定できる。利用者は、システムがアドバイスした正例 (または負例) を検索するための条件を指定する。従って、条件を満たすリテラルが要素辞書に複数個含まれていた場合には、利用者はより少ない条件指定で目的のコンポーネントを特定できる。

要素検索を用いた場合、要素辞書中で検索対象となる要素の数は候補コンポーネント数にほぼ比例して減少する。このため、候補コンポーネント数が少なくなると、条件指定が難しくなる。また、要素辞書中の検索対象要素を一覧できれば、条件指定を行う必然性は薄くなる。従って、要素検索は候補コンポーネント数が多い場合に有効な方法である。

3.3 要素の重複度

候補コンポーネント数が少なくなると、要素検索は有効には機能しなくなる。要素の重複度を用いた支援機構は、この問題点を補うために提案された。

コンポーネント集合を C、サンプルを S、$C' = Set(C, S)$、C' の要素辞書を $D_{C'}$ とする。要素 $e \in D_{C'}$ の重複度 W_e を以下の式で定義する。

$$W_e = |\{c | c \in C' \text{かつ } e \in g(c)\}|$$

要素 e を正例として S に追加すると、候補コンポーネント数は W_e に減少する。e を負例として S に追加すると、候補コンポーネント数は $|C'| - W_e$ に減少する。従って、重複度の小さい要素は正例として、重複度の大きい要素は負例としてサンプルに追加するのが効果的である。

システムは、$D_{C'}$ の各要素について重複度を計算し、重複度順に要素を並べ換えて利用者に提示する。利用者は提示されたリストを見ながら、サンプルに追加すべき要素を決定する。サンプルにリテラルを追加するたびに C' は変化するので、$D_{C'}$ だけでなく重複度順の要素リストを再計算する。

要素の重複度を利用した上記の方法は、重複度計算のために候補コンポーネント数に比例する手間がかかる。従って、候補コンポーネント数が少ない場合に有効な方法である。また要素検索とは違い、利用者が条件指定を行う必要はない。利用者が常に適切なリテラルをサンプルに追加すれば、n 個のコンポーネントを含むリポジトリに対して $\log_2 n$ 個以下のリテラルを含むサンプルによって任意のコンポーネントを特定できる。ここで、適切なリテラルとは、(1) 重複度が $|C'|/2$ 以下の要素を正例とする、また

表 1: 引数を表現する集合要素

要素	引数の種類
通し番号:データ型:In	入力用引数
通し番号:データ型:Out	出力用引数
通し番号:データ型:InOut	入出力用引数

は (2) 重複度が $|C'|/2$ 以上の要素を負例とする、のいずれかに該当するものである。

4 集合を用いた VB 関数仕様の表現

サンプルを用いた検索方式を実現するためには、集合を用いてコンポーネント仕様を表現する必要がある。本節では集合を用いて VB 関数の仕様を表現する。

VB 関数は一般に 1 個以上の引数を取り、戻り値を返す。関数によっては、実行時にシステム状態 (例：システムクロック、Windows レジストリ) を参照または変更することもある。議論を単純化するために、本稿では以下の条件を満足する関数を対象として議論する。

1. システム状態を参照または変更しない。
2. 引数および戻り値は、単純データ型 (VB に組み込まれているデータ型) または単純データ型を要素とする配列である。

第 2 の条件により、ポインタ、構造体の配列、グラフィックスオブジェクト等を引数 (戻り値を含む) とする関数は議論から除外される。

VB 関数の仕様は、各引数 (戻り値を含む) を表現する要素 (引数要素) と具体的な入出力値の組を表現する要素 (入出力値要素) からなる集合で定義される。具体的な入出力値の組は無限に存在するため、一般にこの集合は無限集合である。

関数の引数は関数に対する入力、出力、入出力を行う 3 種類の引数に分類される。また、各引数毎にデータ型を持つ。関数の戻り値は出力用引数の特別な場合と考えられる。以上から、関数のシグネチャは各引数の通し番号、種類、データ型から構成される組を要素とする集合で表現できる (表 1 を参照のこと)。引数の通し番号は、種類とデータ型の両方が一致する引数を区別するために導入する。

VB 関数の中には仕様が異なるにも関わらずシグネチャが同一のものが含まれている。例えば、`Left` 関数と `Right` 関数のシグネチャを表現する集合はいずれも以下のようになる。

{1:String:In, 2:Integer:Out, 3:String:Out}

この場合、両者を区別するためには、具体的な入力用引数 (入出力用引数を含む) を与えて関数を実行し、出力用引数 (入出力用引数を含む) の値を比較する必要がある。これを実現するために、具体的な入出力値 (引数および戻り値) の組で定義される要素を考える。

本稿では以上で定義した引数要素と入出力値要素を用いて関数の仕様を定義する。具体例を表 2 に示す。入出力値要素は引数の通し番号の順に並べることで、引数との対応付けが可能になる。VB 関数の仕様を表現する集合は一般に無限集合になる。しかし、n 個の VB 関数の集合 C が与えられた時、要素辞書 D_C は高々 $n-1$ 個の要素しか含まない。そのため、有限の集合を対象とした検索が可能になる。

5 VB 関数検索システムの仕様

5.1 機能

VB 関数検索システムは以下に列挙する 5 つの機能を持つ。利用者 (管理者を含む) は図 1 に従ってこれらの機能を利用する。

関数の追加 システム管理者が入力した VB 関数をリポジトリに追加する。関数追加の際には要素辞書を再構成する。必要に応じて、既存の VB 関数との差を表す要素を管理者に入力させる。

関数の削除 管理者が指定した VB 関数をリポジトリから削除する。関数削除に伴い要素辞書を再構成する。

サンプルの作成 サンプルの初期化およびサンプルへのリテラル追加を行う。サンプル作成支援機構として要素検索と重複度別分類を提供する。要素検索では、正例/負例のいずれを追加すべきか利用者にアドバイスする。また、利用者が指定した条件を用いて要素辞書を検索する。検索された要素を表示する際には重複度を併記する。利用者は重複度を参考にしてリテラルを指定する。

関数の検索 利用者が作成したサンプルを用いて VB 関数を検索する。サンプルの条件を満たす VB 関数をリポジトリから絞り込む。

補助 リポジトリ、検索された VB 関数の一覧、サンプルをそれぞれ表示する。サンプルを初期化する機能も提供する。利用者が指定した 2 つの VB 関数について、両者を区別する要素を要素辞書から検索する。

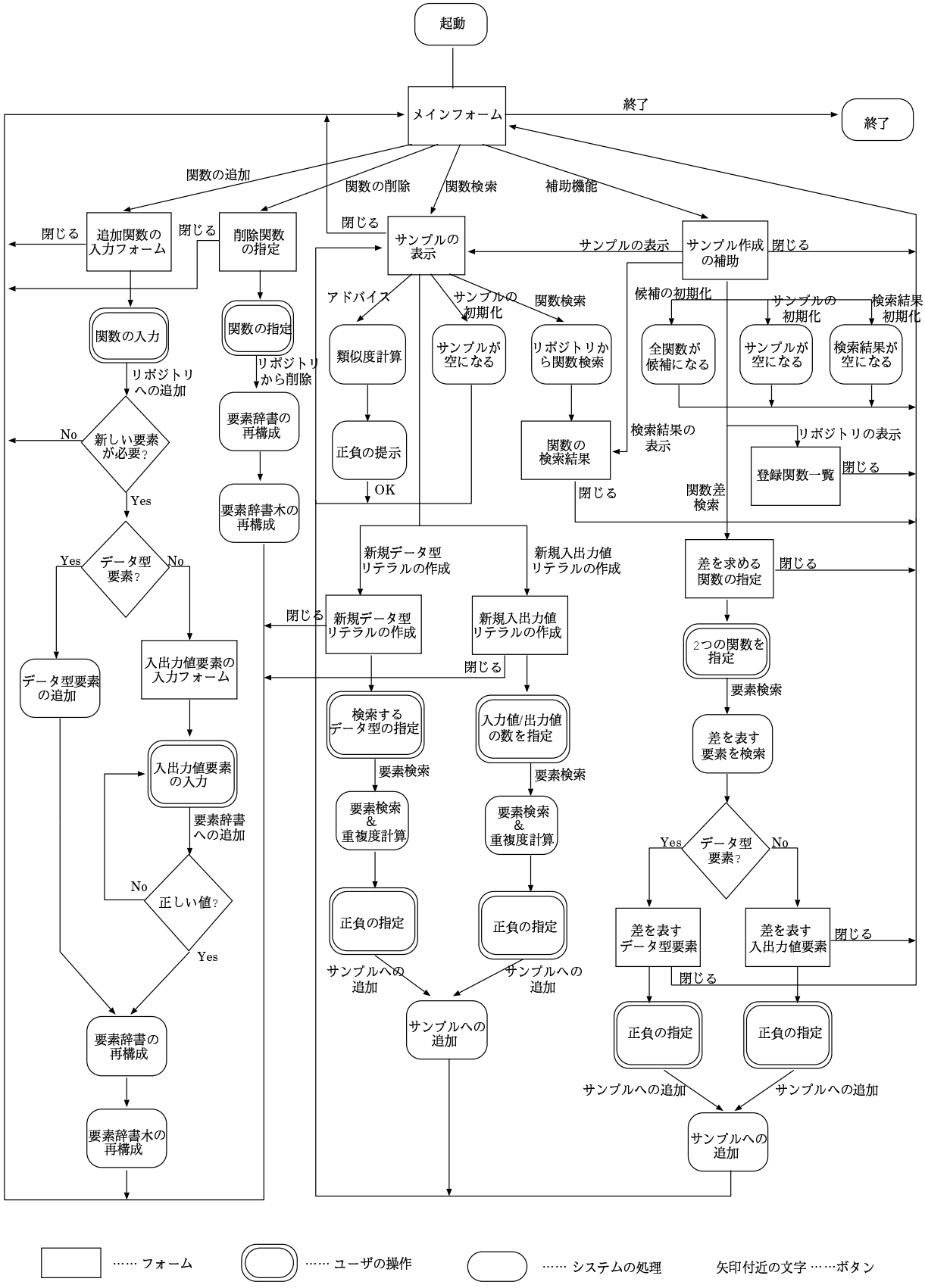

図 1: 利用者による操作の流れ

表 2: 集合を用いた VB 関数仕様の表現例

関数	仕様を表現する集合
Subtring	{ 1:String:In, 2:Variant:In, 3:Variant:In, 4:String:Out, ⟨"abc",2,2,"bc"⟩, ⟨"a",1,1,"a"⟩, ··· }
Right	{ 1:String:In, 2:Integer:In, 3:String:Out, ⟨"abc",2,"bc"⟩, ⟨"a",1,"a"⟩, ··· }
Replicate	{ 1:String:In, 2:Integer:In, 3:String:Out, ⟨"abc",2,"abcabc"⟩, ⟨"a",1,"a"⟩, ··· }

5.2 利用者インタフェース

本節では、サンプル作成および関数検索の流れに従って VB 関数検索システムの利用者インタフェースを紹介する。紙数の関係で関数の追加/削除、補助の各機能や詳細については文献 [7] に譲る。

サンプルの表示 利用者がサンプルを表示/編集する際には「サンプルの表示」ウインドウを用いる。「サンプルの表示」ウインドウに設定された 2 つのタブでは、データ型要素の追加 (図 2) と入出力値リテラルの追加 (図 3) を行える。利用者が「アドバイス」ボタンを押すと、システムは 3.2 節の方式に基づいて正例/負例のいずれを追加するのが適切かをアドバイスする。「サンプルの初期化」ボタンはサンプルを空にするためのものである。

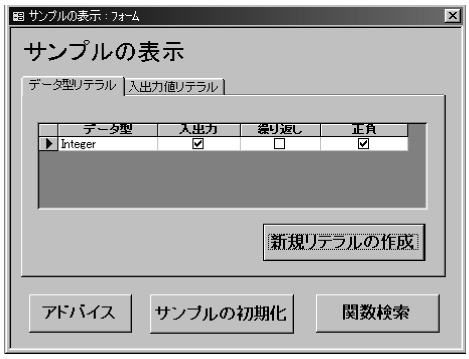

図 2: サンプルの表示 (データ型リテラル)

リテラルの追加 「サンプルの表示」ウインドウで利用者が「新規リテラルの作成」ボタンを押すと、要素辞書から要素を検索するためのウインドウが開かれる (図 4、5)。

図 4 はデータ型リテラルを作成するためのウインドウである。利用者は列挙された各データ型について「含む」フィールドをチェックすることで検索条件を指定する。利用者が複数のデータ型を指定した場合には、システムは OR 条件として解釈する。「要素検索」ボタンを押すと要素辞書から条件を満たす要素が検索され、繰り返し (配列ならば Yes) および重複度とともに表示される。利用者が要素に対して正負の別を指定して「サンプルへの追加」ボタンを押すと、当該要素がリテラルとしてサンプルに追加される。

図 3: サンプルの表示 (入出力値リテラル)

図 5 は入出力値リテラルを作成するためのウインドウである。利用者は検索条件として入出力値リテラルに含まれる入力値と出力値の数を指定する。利用者は検索された要素に正負の別を指定した上でサンプルに追加できる。

関数の検索結果 「サンプルの表示」ウインドウで利用者が「関数検索」ボタンを押すと、システムは現在のサンプルを用いて関数検索を実行する。関数の検索結果を図 6 に示す。検索された関数の 1 つを利用者が選択すると、システムは要素辞書から当該関数の引数 (戻り値) および入出力値を検索して表示する。

6 VB 関数検索システムの実装

VB 関数検索システムは表 3 に示すテーブルを用いて必要なデータを格納している。表 5 および表 7 で指定できるデータ型は単純データ型のみである。また、表 8 を用い

図 4: データ型リテラルの作成

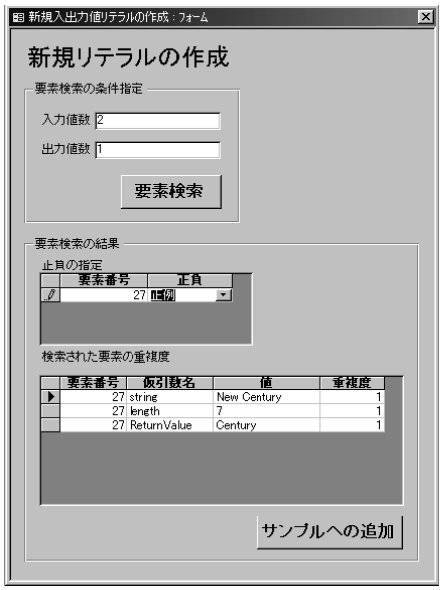

図 5: 入出力値リテラルの作成

表 3: VB 関数検索システムのテーブル一覧

	説　明
表 4	リポジトリ (関数の一覧)
表 5	リポジトリに格納されている関数の引数
表 6	要素辞書 (データ型要素と入出力値要素の一覧)
表 7	データ型要素のデータ型情報
表 8	入出力値要素の各フィールド値
表 9	サンプル (リテラル一覧)

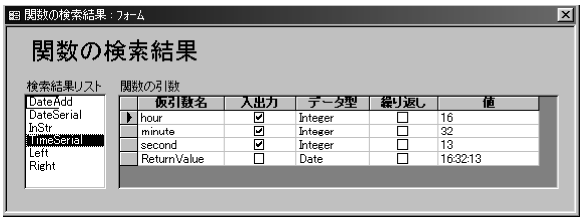

図 6: 関数の検索結果

ることで、引数の数やデータ型が異なる関数の入出力値要素を単一のスキーマに格納している。

利用者が指定したリテラルがデータ型を表現していた場合には、表 7 を参照することで関数が検索されるか否かを容易に判定できる。利用者が入出力値要素をリテラルとして指定した場合には、関数を動的に実行する必要がある。これは、関数呼出しを含む SQL 文を生成し、それを実行することで実現する。

7　むすび

現在、我々は VB 関数検索システムの評価実験を進めている。実験の結果は別途報告を予定している。4 節で示したように、現時点の VB 関数検索システムにはいくつかの制限事項があるため、これらの点に対する拡張作業が今後の課題として挙げられる。

参考文献

[1] C. Jones 著、鶴保、富野監訳、“ソフトウエア開発の定量化手法”、共立出版 (発行 構造計画研究所)、1998.

[2] 青山、中所編、“コンポーネントウェア”、共立出版、1998.

[3] T. Kakeshita, M. Murata, “Specification-based component retrieval by means of examples”, International Symposium on Database Applications in Non-Traditional Environments (DANTE'99), pp. 389–398, 1999.

[4] 村田 美友紀、掛下 哲郎、“集合間の相違を明確にする要素辞書”、情報処理学会論文誌 : データベース、Vol. 40、No. SIG3 (TOD1)、pp. 60–67、1999.

[5] 掛下 哲郎、村田 美友紀、“仕様に基づいた RDB クエリ検索システム”、情報処理学会論文誌 : データベース、Vol. 41、No. SIG6 (TOD7)、pp. 37–45、2000.

[6] 川口 雄司、“サンプルを用いた VB 関数検索システムの実装”、佐賀大学 知能情報システム学科卒業論文、平成 12 年 3 月.

[7] 釣本 勝之、“サンプルを用いた Visual Basic 関数検索システムの開発 – 入出力値を用いた検索条件の指定”、佐賀大学 知能情報システム学科卒業論文、平成 13 年 3 月.

フィールド名	キー	データ型	説明
関数番号	*	数値型	関数を識別する番号
関数名		テキスト型	
候補		Yes/No 型	検索候補か否か

表 4: リポジトリのスキーマ定義

フィールド名	キー	データ型	説明
関数番号	*	数値型	
仮引数名	*	テキスト型	
入出力		Yes/No 型	入力引数ならば Yes。出力引数 (戻り値を含む) ならば No。
データ型		テキスト型	引数のデータ型
繰り返し		Yes/No 型	引数が配列ならば Yes。

表 5: 関数の引数のスキーマ定義

フィールド名	キー	データ型	説明
要素番号	*	数値型	各要素を識別する番号
種類		Yes/No 型	データ型要素ならば Yes。

表 6: 要素辞書のスキーマ定義

フィールド名	キー	データ型	説明
要素番号	*	数値型	
入出力		Yes/No 型	入力引数ならば Yes。出力引数 (戻り値を含む) ならば No。
データ型		テキスト型	引数のデータ型
繰り返し		Yes/No 型	引数が配列ならば Yes。

表 7: データ型要素のスキーマ定義

フィールド名	キー	データ型	説明
要素番号	*	数値型	
仮引数名	*	テキスト型	値と対応する仮引数名
値		テキスト型	引数または戻り値の値

表 8: 入出力値要素のスキーマ定義

フィールド名	キー	データ型	説明
リテラル番号	*	数値型	要素辞書の要素番号を指定する。
正負		Yes/No 型	正例ならば Yes。負例ならば No。

表 9: サンプルのスキーマ定義