データベースシステム　122－16
(2000. 7. 26)

サンプルを用いた検索方式における仮想的なシステム状態の構築

村田 美友紀† 掛下 哲郎††
† 八代工業高等専門学校 情報電子工学科
†† 佐賀大学理工学部 知能情報システム学科
〒866-8501 熊本県八代市平山新町 2627
TEL: 0965(53)1301, E-mail: m-murata@as.yatsushiro-nct.ac.jp

あらまし

我々は，サンプルを用いたコンポーネント検索方式を提案した．本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．集合はコンポーネントの実行結果に対応するためシステム状態に依存する．そこで運用中のシステム状態とは独立した仮想的なシステム状態を導入する．n 個のコンポーネントからコンポーネントを特定するためには，集合が互いに異なるような仮想システム状態が必要である．これを満足する単一の仮想システム状態は，常には存在しない．複数の仮想システム状態を導入すると，集合が互いに異なるような仮想システム状態の集合を構築できる．このとき仮想システム状態集合の要素数は n を超えない．また仮想システム状態を導入するために本方式を拡張する．

キーワード データベース，情報検索，コンポーネントウエア，コンポーネント仕様

Construction of Virtual System State for Sample based Retrieval Mechanism

Miyuki Murata† Tetsuro Kakeshita††
†Department of Information and Electronics Engineering,
Yatsushiro National College of Technology
††Department of Information Science, Saga University
2627 Hirayama-shimnachi, Yatsushiro. Kumamoto. 866-8501 JAPAN
TEL: 0965(53)1301, E-mail: m-murata@as.yatsushiro-nct.ac.jp

Abstract

We proposed the sample based component retrieval mechanism. The specification of a component is represented using a set. The set corresponds to result of a component. We introduce virtual system state independent from current system state. In order to identify a target from n components, a virtual system state in which each sets representing the specifications of components are different is necessary. Such single virtual system state always does not exist. Thus, we introduce multiple virtual system state. We are able to construct a collection of virtual system states in which each sets are different. Here the collection has less than n members. We extend our mechanism to introduce the virtual system state into our mechanism.

key words database, information retrieval, component ware, component specification

1 はじめに

ソフトウエア工学の分野ではアプリケーション開発コストを低減するために，関数やクラス，モジュールなどの再利用が行われている．再利用の対象となるものをコンポーネントと呼ぶ．データベース (DB) の分野においても，レポートやフォームを作成する際に，多くのクエリやビューが作成される．これらの中には，入れ子クエリや関数呼出を含むクエリなど，SPJ 質問だけでは作成が困難なクエリも含まれる．クエリを再利用することで，DB アプリケーションの開発コストが低減される．

コンポーネントを再利用するためには，コンポーネントの格納とそれに対する検索機構が必要である．我々はこれまでの研究で仕様に基づいたコンポーネント検索方式 [1] を提案している．本方式は，サンプルを用いたコンポーネント検索機構，コンポーネント間の差を明確にする要素辞書 [2]，サンプル作成支援機構から構成されている．本方式では，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 の仕様を表現する集合 $g(c)$ の要素は，c の実行例に対応する．コンポーネントがクエリであれば，射影属性やクエリによって検索される組が要素に対応する．コンポーネント検索の際には，要素を用いてサンプルを作成する．

コンポーネントを再利用するためには，所望のコンポーネントを特定できることが必要である．このためには，コンポーネントの仕様を表現する集合が互いに異なることが必要である．集合が互いに異なるならば，コンポーネント間の違いを説明できる．コンポーネント c の仕様を表現する集合 $g(c)$ は，コンポーネント実行時におけるシステム状態に依存する．よって，集合が互いに異なるようなシステム状態が必要である．このようなシステム状態を仮想システム状態と呼ぶ．仮想システム状態は運用中のシステム状態とは独立しているため，運用中のシステムに対する操作とコンポーネント検索を独立に実行できる．一般にシステムの運用中には，システム状態は変更される．これにともない，仮想システム状態の再構築が必要になる．また，仮想システム状態をサンプルを用いたコンポーネント検索方式に適用する．

本稿は以下のように構成されている．2 節では，集合を用いてコンポーネントの仕様を表現する．3 節では，n 個のコンポーネントの仕様を表現する集合が，互いに異なるような仮想システム状態について議論する．このとき単一の状態のみを扱う場合と複数の状態を扱う場合を考える．4 節では，仮想システム状態を導入するために，サンプルを用いたコンポーネント検索方式を再定義する．5 節では，コンポーネント集合の更新にともなう仮想システム状態と要素辞書の再構築アルゴリズムを提案する．最後にまとめと今後の課題について述べる．

2 集合を用いたコンポーネント仕様の表現

本節では，集合を用いてコンポーネントの仕様を表現する．実行可能なコンポーネントの仕様は，その実行結果によって表現できる．コンポーネントは一般に引数などの入力値やシステム状態を読み込んで，出力値を計算する．システム状態は，コンポーネントの出力値に影響するデータの中で，入力値以外のデータの状態である．ルーチン (関数および手続き) におけるシステム状態としては大域変数，モジュール変数，ファイル，各種システム設定などがあげられる．またクエリの場合は DB の状態が対応する．

コンポーネント c の仕様は以下の集合を用いて表現できる．ここで，I は入力値から構成される組である．O は出力値から構成される組である．A はコンポーネント実行前のシステム状態である．

$$\{\langle I, O, A\rangle\}$$

本稿で議論するコンポーネント集合は，それに含まれるコンポーネントの仕様が互いに異なると仮定する．仕様が等しい 2 個のコンポーネントの差を明確にするためには，実行時間やアルゴリズムを比較する手法がある．

実行可能なコンポーネントには，副作用を持つ関数や更新クエリなどシステム状態を変更するコンポーネントも存在する．このようなコンポーネントについても集合を用いて仕様を表現できる [3]．本稿では，議論をシンプルにするために，システム状態を変更しないコンポーネントを対象にする．

以下にクエリ仕様を表現する集合の例を示す．クエリの仕様は，射影属性，クエリが検索する組，組のペアの 3 種類の要素から構成される．組のペアは検索された組の表示順序を表現するために用いる．

例 1 図 1(A) に示す 2 個のテーブルから構成される DB を考える．これらに対して，図 1(B) のクエリ q_1 を実行した結果が 図 1(C) である． q_1 の仕様は以下の集合を用いて表現できる．ここで，図 1(A) の DB 状態を仮想システム状態 A_1 とする．

$\{\langle$ 学生, $A_1\rangle, \langle$ 科目, $A_1\rangle, \langle$ 点, $A_1\rangle,$
$\langle\langle$ 森, 英語, 43$\rangle, A_1\rangle, \langle\langle$ 森, 英会話, 49$\rangle, A_1\rangle,$
$\langle\langle$ 鈴木, 英会話, 45$\rangle, A_1\rangle,$
$\langle[\langle$ 森, 英語, 43$\rangle, \langle$ 森, 英会話, 49$\rangle], A_1\rangle,$
$\langle[\langle$ 森, 英語, 43$\rangle, \langle$ 鈴木, 英会話, 45$\rangle], A_1\rangle,$
$\langle[\langle$ 森, 英会話, 49$\rangle, \langle$ 鈴木, 英会話, 45$\rangle], A_1\rangle\ \}$

□

講義

科目	教官	時間数
英語	山田	16
DB	山下	16
SW	田中	15
英会話	山田	10

成績

学生	科目	点	欠課数
森	英語	43	0
斎藤	英語	90	0
鈴木	英語	70	2
森	DB	58	5
斎藤	DB	65	3
森	英会話	49	0
鈴木	英会話	45	2

(A)

q_1

```
SELECT   学生, 科目, 点
FROM     成績, 講義
WHERE    (講義. 科目 = 成績. 科目)
         AND (教官 = '山田')
         AND (点 < 50)
```

(B)

学生	科目	点
森	英語	43
森	英会話	49
鈴木	英会話	45

(C)

図 1: サンプルスキーマ

3 仮想システム状態

3.1 単一の仮想システム状態

一般にシステム運用中においてシステム状態は変更される．システム状態が変更されると，実行結果も変化するためコンポーネントの仕様を表現する集合も変化する．このため，利用者はコンポーネント検索のたびに現在のシステム状態を理解しなければならない．サンプルを作成するためには，要素 (I, O, A) がコンポーネントに含まれるか否かの判断が必要なためである．これは，利用者にとって大きな負担である．そこで，コンポーネント実行前のシステム状態を仮定する．仮定したシステム状態は運用中のシステム状態とは独立しており，コンポーネント実行の影響を受けない．仮定したシステム状態を仮想システム状態と呼ぶ．仮想システム状態は，集合を用いてコンポーネントの仕様を表現するときやサンプルを用いてコンポーネントを検索するときに用いる．

一般に仮想システムの状態は複数存在する．ここで，仮想システム状態 A_i におけるコンポーネント実行結果に対応する集合を $g(A_i, c)$ と表記する．また，$g(c) = \cup g(A_i, c)$ を定義する．

2 個のコンポーネント c, c' について，以下の定理が成り立つ．ただし，$\mathit{diff}(A, c, c') = (g(A, c) - g(A, c')) \cup (g(A, c') - g(A, c))$ である．

定理 1 コンポーネント c, c' の仕様が異なるとき，$\mathit{diff}(A, c, c') \neq \phi$ を満足する仮想システム状態 A が少なくとも 1 個存在する． □

[証明] c, c' の仕様が異なると仮定すると，$g(c) \neq g(c')$ が成り立つ．ここで，任意の仮想システム状態 A について $\mathit{diff}(A, c, c') = \phi$ が成立するならば，$g(c) = g(c')$ となり仮定に反する． □

$g(A, c)$, $g(A, c')$ のいずれか一方にのみ含まれる要素を用いると c, c' の差を明確にできる．よって，$\mathit{diff}(A, c, c')$ は c, c' $(\in C)$ の差を明確にする要素の集合である．

コンポーネント集合について完全な仮想システム状態を以下に定義する．

定義 1 コンポーネント集合 C と仮想システム状態 A を考える．任意の 2 個のコンポーネント $c_i, c_j (\in C)$ について以下の式が成り立つとき，仮想システム状態 A は C について完全である．

$$\mathit{diff}(A, c_i, c_j) \neq \phi$$

□

C について完全な仮想システム状態 A が存在するとき，任意のコンポーネント c_i, c_j $(\in C)$ の差を明確にする要素が存在する．これらを組み合わせることによって任意のコンポーネント c_i を常に特定できる．

任意のコンポーネント集合について以下の補題が成り立つ．

補題 1 任意のコンポーネント集合 $C = \{c_1, \cdots, c_n\}$ を考える．c_i $(\in C)$ の仕様を表現する集合 $g(c_i)$ が互いに素であるならば，任意の仮想システム状態 A は C について完全である． □

[証明] $g(c_i) \cap g(c_j) = \phi$ より，任意の仮想システム状態 A について $\mathit{diff}(A, c_i, c_j) \neq \phi$ が成り立つことは自明である． □

補題 2 任意のコンポーネント集合 $C = \{c_1, \cdots, c_n\}$ を考える．$|\cup_{i=1}^{n} g(c_i)| < \log_2 n$ が成り立つならば，C について完全な仮想システム状態 A は存在しない． □

[証明] 要素集合 E を用いて集合 $G = \{g_i | g_i \subseteq E\} (i = 1, \cdots, 2^{|E|} + 1)$ を作成する．G の中には，互いに等しい集合が少なくとも 2 個存在する．g_i をコンポーネント c_i $(\in C)$ の仕様を表現する集合 $g(A, c_i)$ と考えると，$\mathit{diff}(A, c_i, c_j) = \phi$ を満足する 2 個のコンポーネント

システム状態 A_1 におけるクエリ実行結果

q_2：DB の成績順に学生をソートする．

学生
斎藤
森

q_3：平均点が高い順に学生をソートする．

学生
斎藤
鈴木
森

q_4：英語の成績順に学生をソートする．

学生
斎藤
鈴木
森

システム状態 A_2 におけるクエリ実行結果

q_2

学生
鈴木
斎藤
森

q_3

学生
鈴木
斎藤
森

q_4

学生
斎藤
鈴木
森

図 2: クエリの例

$c_i, c_j (\in C)$ が必ず存在する．よって，A は定義 1 の条件を満足しない． □

補題 1 は，コンポーネント集合 C について完全な仮想システム状態が常に存在する十分条件を示している．補題 2 は，C について完全である仮想システム状態が常に存在しない十分条件を示している．

コンポーネント集合 C について完全な仮想システム状態 A が存在すると仮定する．C に新たなコンポーネント c を追加する．仮想システム状態によっては，$\mathit{diff}(A, c, c_k) = \phi$ となるコンポーネント c_k 存在する場合がある．このとき c_k は高々1 個である．これは，$\mathit{diff}(A, c, c_k) = \phi$ を満足する 2 個のコンポーネント c_i, c_j が存在するとき，$\mathit{diff}(A, c_i, c_j) = \phi$ が成り立つことから証明できる．そこで，$C \cup \{c\}$ について完全な仮想システム状態 A' を構築する必要がある．

$\mathit{diff}(A', c, c_k) \neq \phi$ が成立するように仮想システム状態 A' を構築したとき，任意のコンポーネント $c_i, c_j (\in C)$ について $\mathit{diff}(A', c_i, c_j) \neq \phi$ が成り立つならば，A' は $C \cup \{c\}$ について完全である．しかし一般には，システム状態が更新されると他のコンポーネント c_i の仕様を表現する集合 $g(A', c_i)$ も変更される．よって，$\mathit{diff}(A', c_i, c_j) = \phi$ が成立する場合がある．このとき，A' は $C \cup \{c\}$ について完全でない．このため，仮想システム状態を更新するときに c, c_k に注目するだけでは，定義 1 の条件を満足する A' が構築できない．また，補題 2 より完全な仮想システム状態が存在しないコンポーネント集合も存在する．したがって，任意のコンポーネント集合 C に対して，完全な仮想システム状態を構築できるとは限らない．

完全な仮想システム状態が存在するコンポーネント集合に，新規コンポーネントを追加する場合の例を示す．

例 2 図 1(A) に示す DB 状態を仮想システム状態 A_1 とする．また，図 2 のクエリ q_2, q_3 から構成される集合を考える．$\mathit{diff}(A_1, q_2, q_3) \neq \phi$ より，A_1 は $\{q_2, q_3\}$ について完全である．$\{q_2, q_3\}$ にクエリ q_4 を追加すると，q_3, q_4 について $\mathit{diff}(A_1, q_3, q_4) = \phi$ が成立する．そこで，q_3, q_4 を区別するためにテーブル 成績に $\langle$ 鈴木, DB, 90$\rangle$ を追加する．追加後の仮想システム状態を A_2 とする．A_2 を用いると $\mathit{diff}(A_2, q_3, q_4) \neq \phi$ が成り立つ．しかし，q_2, q_3 について，$\mathit{diff}(A_2, q_2, q_3) = \phi$ が成立する．よって，$\{q_2, q_3, q_4\}$ について A_1, A_2 はともに完全でない． □

3.2 仮想システム状態の集合

仮想システム状態が単一である場合には，コンポーネント集合について完全である仮想システム状態が常に存在するとは限らない．しかし，所望のコンポーネントを特定するためやコンポーネント間の差を明確にするためには，完全な仮想システム状態を構築することが必要である．

定理 1 より，2 個のコンポーネント c, c' について $\mathit{diff}(A, c, c') \neq \phi$ を満足するシステム状態 A が少なくとも 1 個存在する．よって，仮想システム状態の集合を導入した場合には，以下の定理が成り立つ．

定理 2 コンポーネント集合 $C = \{c_1, \cdots, c_n\}$ についてコンポーネント $c_i (\in C)$ の仕様が互いに異なるならば，高々 $n-1$ 個の仮想システム状態から構成される集合 $\boldsymbol{A}$ が存在し，任意の 2 個のコンポーネント $c_i, c_j (\in C)$ についていずれかの仮想システム状態 $A_l (\in \boldsymbol{A})$ において $\mathit{diff}(A_l, c_i, c_j) \neq \phi$ が成立する． □

[証明] n に関する帰納法を用いる．$n = 1$ の場合には，仮想システム状態は不要なので自明である．$n = k$ の場合に定理 2 が成立したと仮定する．仮定より，すべての仮想システム状態 A で $\mathit{diff}(A, c_{k+1}, c_i) = \phi$ となるコンポーネント $c_i \in C$ は高々 1 個である．したがって，$\mathit{diff}(A', c_{k+1}, c_i) \neq \phi$ を満足する新たな仮想システム状態 A' を作成すると，$\boldsymbol{A} \cup \{A'\}$ は定理 2 の条件を満足

する．□

コンポーネント集合について完全な仮想システム状態の集合を以下に定義する．

定義 2 コンポーネント集合 C と仮想システム状態の集合 $\boldsymbol{A}$ を考える．任意のコンポーネント $c_i, c_j (\in C)$ について以下の式を満足する仮想システム状態 $A_l (\in \boldsymbol{A})$ が存在するとき，$\boldsymbol{A}$ は C について完全である．

$$diff(A_l, c_i, c_j) \neq \phi$$

□

コンポーネント集合 C について $\boldsymbol{A}$ が完全であるとき，任意のコンポーネント $c_i, c_j (\in C)$ の差を明確にする要素が常に求まる．これらの要素を組み合わせると，任意のコンポーネントを特定できる．

コンポーネント集合について完全な仮想システム状態の例を示す．

例 3 図 1(A) に示す DB 状態を仮想システム状態 A_1 とする．テーブル 成績に ⟨鈴木, DB, 90⟩ を追加した後の仮想システム状態を A_2 とする．また，図 2 に示すクエリ集合 $\{q_2, q_3, q_4\}$ 考える．A_1 を用いると $diff(A_1, q_2, q_3) \neq \phi$, $diff(A_1, q_2, q_4) \neq \phi$ が成立する．また，A_2 を用いると $diff(A_2, q_2, q_3) \neq \phi$ が成立する．よって，$\{q_2, q_3, q_4\}$ について，$\{A_1, A_2\}$ は完全である．□

4 サンプルを用いたコンポーネント検索の拡張

本節では，仮想システム状態を適用するためにサンプルを用いたコンポーネント検索方式を拡張する．まず，サンプルを定義した後，要素辞書，サンプル作成支援機構を再定義する．

4.1 サンプルを用いたコンポーネント検索

コンポーネント集合 $C = \{c_1, \cdots, c_n\}$ について完全な仮想システム状態の集合を $\boldsymbol{A} = \{A_1, \cdots, A_m\}$ とする．ここで，$g(\boldsymbol{A}, c) = \cup_{l=1}^{m} g(A_l, c)$ と定義する．このとき，$g(\boldsymbol{A}, c) \subseteq g(c)$ である．

サンプルを定義する．サンプル S はリテラルの集合である．リテラルは，要素 $\langle I, O, A_l\rangle \in \cup_{i=1}^{n} g(\boldsymbol{A}, c_i)$ に対して，正例または負例のいずれかを明示したものである．サンプル S がコンポーネント c を検索するとき，すべての正例 $\langle I, O, A_l\rangle (\in S)$ に対し $\langle I, O, A_l\rangle \in g(A_l, c)$，かつすべての負例 $\overline{\langle I, O, A_l\rangle} (\in S)$ に対し $\langle I, O, A_l\rangle \notin g(A_l, c)$ が成り立つ．S がただ 1 個のコンポーネント c を検索するとき，S は c を特定するという．

以下にサンプルを用いたクエリ検索の例を示す．

クエリ	クエリを特定するサンプル
q_2	$\{\langle\langle$ 鈴木 $\rangle, A_1\rangle\}$
q_3	$\{\overline{\langle\langle$ 鈴木 $\rangle, A_1\rangle}, \langle[\langle$ 斎藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\}$
q_4	$\{\overline{\langle\langle$ 鈴木 $\rangle, A_1\rangle}, \langle[\langle$ 斎藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\rangle\}$

表 1: 要素辞書

例 4 例 3 で用いた仮想システム状態の集合 $\boldsymbol{A}$= $\{A_1, A_2\}$ とクエリ集合 $C = \{q_2, q_3, q_4\}$ を考える．サンプル $\{\overline{\langle\langle$ 鈴木 $\rangle, A_1\rangle}\}$ を用いて検索を行うと，クエリ q_3, q_4 が検索される．さらにサンプル $\{\overline{\langle\langle$ 鈴木 $\rangle, A_1\rangle}, \langle[\langle$ 斎藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\rangle\}$ を用いるとクエリ q_4 が特定できる．□

4.2 要素辞書

コンポーネント集合 C から所望のコンポーネントを特定するサンプルを作成するためには，所望のクエリと他のクエリを区別するリテラルをサンプルに追加する必要がある．このようなリテラルを利用者が直接指定するのは困難である．我々は，C に含まれる任意のコンポーネント間の差を明確にするための要素辞書 [2] を提案している．要素辞書を用いると，所望のコンポーネントを特定するサンプルが常に作成できる．また，コンポーネント間の違いに関する質問にも答えられる．

定義 3 コンポーネント集合 $C = \{c_1, \cdots, c_n\}$ について完全な仮想システム状態の集合を $\boldsymbol{A} = \{A_1, \cdots, A_m\}$ とする．C の要素辞書 D_C は 要素 $\langle I, O, A_l\rangle (\in \cup_{i=1}^{n} g(\boldsymbol{A}, c_i))$ の集合である．任意のコンポーネント c_i, c_j $(c_i \neq c_j)$ に対して $g(\boldsymbol{A}, c_i) \cap D_C \neq g(\boldsymbol{A}, c_j) \cap D_C$ が成立する．□

例 5 例 3 で用いた仮想システム状態の集合 $\boldsymbol{A}$= $\{A_1, A_2\}$ とクエリ集合 $C = \{q_2, q_3, q_4\}$ を考える．C の要素辞書は $\{\langle\langle$ 鈴木 $\rangle, A_1\rangle, \langle[\langle$ 斎藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\rangle\}$ である．クエリとクエリを特定するサンプルを表 1 に示す．□

要素辞書に含まれる要素数 h について，$m \leq h$ が成立する．$m = h$ が成立するとき，要素辞書に含まれるすべての要素 $\langle I, O, A_l\rangle$ について A_l は互いに異なる．

4.3 サンプル作成支援機構

サンプル作成支援機構は，要素辞書を用いて効率的にサンプルを作成するために用いられる．サンプル作成支援機構は，フィルタリング，要素検索，重複度別分類から構成されている．これらを組み合わせると，n 個のコンポーネントから所望のコンポーネントを特定するサンプルが作成できる．このとき，サンプルの要素数はほとんどの場合 $\log_2 n$ になる．

フィルタリング

コンポーネントの仕様を表現する集合には，複数の種類の要素が混在する場合がある．例えば，クエリの仕様を表現する集合は，射影属性，組，組のペアの3種類の要素を含む．フィルタリングは，これらの種類の中から目的の種類の要素のみを抽出するために用いる．

要素検索

本手法は，フィルタリングで求めた要素の中から利用者が作成するクエリを用いてサンプルに追加するリテラルを検索する．作成するクエリを要素検索クエリと呼ぶ．要素検索クエリは，コンポーネントの種類によって記述法が異なる．要素検索クエリは，仮想システム状態 A_l についても条件を指定できる．指定できる条件は，$A_l\ \theta\ X$ および $A_l\ \varphi\ \boldsymbol{X}$ である．θ は $=, \neq$ のいずれかである．φ は $\in, \notin$ のいずれかである．X は仮想システム状態，$\boldsymbol{X}$ は仮想システム状態の集合である．

重複度別分類

本手法は，正例としてサンプルに追加したときの絞り込みの効果を用いて要素を分類する．重複度別分類を用いると，絞り込み効果が最も高い要素をリテラルとしてサンプルに追加できる．以下に要素の重複度を定義する．

定義 4 コンポーネント集合 C について完全な仮想システム状態の集合を $\boldsymbol{A}$ とする．C に対する要素 $\langle I, O, A_l\rangle$ の重複度は $|\{c_i \in C | \langle I, O, A_l\rangle \in g(A_l, c_i)\}|$ である． □

システムは，要素検索により検索された要素について重複度を計算し，それらを重複度順にソートしたリストを利用者に提示する．利用者は提示されたリストの中から要素を選択する．絞り込み効果が高い要素を選択するためには，正例を追加する場合は重複度が小さいものを選択し，負例を追加する場合は重複度が大きいものを選択すればよい．

5 要素辞書および仮想システム状態集合の再構成

コンポーネント集合が変更されたとき，仮想システム状態および要素辞書の再構築が必要になる場合がある．本節では，コンポーネント追加，削除時における仮想システム状態および要素辞書の再構成アルゴリズムを提案する．これに先ち，要素辞書木を定義する．

5.1 要素辞書木

要素辞書再構築のために，要素辞書木を導入する．要素辞書木の定義を以下に示す．

定義 5 完全な仮想システム状態の集合 $\boldsymbol{A}$ が存在するコンポーネント集合 $C = \{c_1, \cdots, c_n\}$ について，要素辞書木 T_C は以下の条件を満足する木である．

1. 各中間節点 v は要素 $\langle I, O, A_l\rangle (\in \cup_{i=1}^{n} g(\boldsymbol{A}, c_i))$ に対応しており，2個の子節点 v_+,v_- を持つ．枝 (v, v_+) のラベルは $\langle I, O, A_l\rangle$ であり，枝 (v, v_-) はラベルを持たない．
2. 各葉節点 v は $c_i\ (\in C)$ と一対一に対応している．根節点から v に至る中間節点に対応する要素の集合を E_v とすると，根節点から v に至る経路上のラベル集合は $g(\boldsymbol{A}, c_i) \cap E_v$ である．

□

要素辞書木 T_C の中間節点に対応する要素の集合は C の要素辞書 D_C であり，$|D_C| \leq n-1$ である．

5.2 コンポーネント追加にともなう再構成

完全な仮想システム状態の集合 $\boldsymbol{A}$ が存在するコンポーネント集合 C と C の要素辞書 D_C を考える．C に新たなコンポーネント c を追加したとき，D_C では c と他のコンポーネント $c_k \in C$ の差を明確にできない場合がある．このような場合には，c, c_k の差を明確にする要素を D_C に追加する必要がある．ところが，$C \cup \{c\}$ について $\boldsymbol{A}$ が完全であるとは限らないため，c, c_k の差を明確にする要素が存在しない場合がある．このような場合には，新たな仮想システム状態を作成し，$\boldsymbol{A}$ に追加する必要がある．本節では，コンポーネントの追加にともなう要素辞書および仮想システム状態の再構成アルゴリズムを提案する．

要素辞書および仮想システム状態集合の再構成アルゴリズム（コンポーネント追加時）

入力：要素辞書木 T_C，要素辞書 D_C，仮想システム状態集合 $\boldsymbol{A}$，コンポーネント c

出力：要素辞書木 $T_{C'}$，要素辞書 $D_{C'}$，仮想システム状態集合 $\boldsymbol{A}'$

(1) 追加するコンポーネント c に対して，$g(\boldsymbol{A}, c) \cap E_k = g(\boldsymbol{A}, c_k) \cap E_k$ を満足するコンポーネント c_k を検索する．ここで，E_k は T_C の根節点から $g(\boldsymbol{A}, c_k)$ に対応する葉節点に至る中間節点に対応する要素の集合である．

(2) $\langle I, O, A_l\rangle \in \mathit{diff}(\boldsymbol{A}, c, c_k)$ を検索する．

(3) $\langle I, O, A_l \rangle$ が存在するならば，$\boldsymbol{A}' \leftarrow \boldsymbol{A}$.

(4) $\langle I, O, A_l \rangle$ が存在しないならば，以下の処理を実行する．

(4-1) $diff(A', c, c_k) \neq \phi$ を満足する仮想システム状態 A' を作成し，$\boldsymbol{A}' \leftarrow \boldsymbol{A} \cup \{A'\}$ とする．

(4-2) $\langle I, O, A_l \rangle \in diff(\boldsymbol{A}', c, c_k)$ を検索する．

(5) $D_{C'} \leftarrow D_C \cup \{\langle I, O, A_l \rangle\}$.

(6) $g(c_k)$ に対応する葉節点 v_k を中間節点とし，要素 $\langle I, O, A_l \rangle$ を対応づける．

(7) v_k の子節点 v_+, v_- を作成する．枝 (v_k, v_+) のラベルを $\langle I, O, A_l \rangle$ とする．$\langle I, O, A_l \rangle \in g(\boldsymbol{A}', c)$ ($\langle I, O, A_l \rangle \notin g(\boldsymbol{A}', c)$) ならば，$v_+$ と v_- をそれぞれ c および c_k(c_k および c) と対応づける． □

ステップ (1) において，c_k を求めるアルゴリズムを以下に示す．

要素辞書木を用いたコンポーネント検索アルゴリズム

入力：要素辞書木 T_C，コンポーネント c, 仮想システム状態の集合 $\boldsymbol{A}$

出力：$g(\boldsymbol{A}, c) \cap E_k = g(\boldsymbol{A}, c_k) \cap E_k$ を満足するコンポーネント c_k ($\in C$)

(1) 節点 v を T_C の根節点とする．

(2) v が葉節点ならば v に対応する集合を $g(c_k)$ とする．

(3) v に対応する要素 $\langle I, O, A_l \rangle$ について，$\langle I, O, A_l \rangle \in g(A_l, c)$ ならば $v \leftarrow v_+$，そうでなければ $v \leftarrow v_-$ とする．

(4) 2へ戻る． □

上記のアルゴリズムを用いると c_k がただ1個検索されることが保証されている [2]．ステップ (2) で要素が検索できるならば，$\boldsymbol{A}$ は C について完全であるため，新たな仮想システム状態を作成しなくてもよい．要素が検索できなければ，ステップ (4-1) で仮想システム状態を作成し，仮想システム状態の集合 $\boldsymbol{A}'$ を構築する．検索される c_k はただ1個であるため，高々1個の仮想システム状態を作成すればよい．よって，$\boldsymbol{A}'$ に含まれる仮想システム状態の数は $n-1$ を越えない．$\boldsymbol{A}'$ は定義2の条件を満足するため，$C \cup \{c\}$ について完全である．ステップ (4-1) において，新たな仮想システム状態を作成するために以下の方法が考えられる．

1. 運用中のシステム状態 A' 上で c, c_k を実行し，$diff(A', c, c_k) \neq \phi$ が成り立つならば，A' を新たな仮想システム状態とする．
2. c, c_k を利用者に与え，利用者が作成する．

$\boldsymbol{A}'$ は，$C \cup \{c\}$ について完全であるため，ステップ (4-2) において，条件を満足する要素が必ず存在する．ステップ (5) では，要素辞書を再構築する．ここで，C に含まれる任意のコンポーネントは D_C を用いて特定できるため，追加する要素は高々1個でよい．ステップ (6), (7) では，要素辞書木を再構築する．

以下にコンポーネント追加に伴う要素辞書および仮想システム状態集合の再構成の例を示す．

例 6 図 1(A) に示す DB 状態を仮想システム状態 A_1 とする．仮想システム状態の集合 $\boldsymbol{A}$= $\{A_1\}$ と図 2 のクエリ q_2, q_3 から構成される集合 C を考える．C の要素辞書を $D_C = \{\langle\langle$ 鈴木 $\rangle, A_1\rangle\}$ とする．C にクエリ q_4 を追加する．要素辞書および仮想システム状態集合の再構成アルゴリズム (コンポーネント追加時) のステップ (1) において q_3 が検索される．ステップ (2) では，$\langle I, O, A_l \rangle$ が検索されないため，$diff(A_2, q_3, q_4) \neq \phi$ が成立する仮想システム状態 A_2 を作成し，$\boldsymbol{A}$ に追加する (ステップ (4-1))．ここでは，テーブル 成績に組 $\langle$ 鈴木, DB, 90$\rangle$ を追加した DB 状態を A_2 とする．ステップ (4-2) において，要素 $[\langle$ 斉藤 $\rangle, \langle$ 鈴木 $\rangle] \in diff(A_2, q_3, q_4)$ が検索できる．これを用いて要素辞書を再構成すると，$D_{C'} = \{\langle\langle$ 鈴木 $\rangle, A_1\rangle, \langle[\langle$ 斉藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\rangle\}$ が構成できる (ステップ (5))．

再構成された仮想システム状態 $\boldsymbol{A}'$ は，$C \cup \{c\}$ について完全である．再構成された要素辞書 $D_{C'}$ を用いると任意のクエリ $q_i (\in C)$ を特定するサンプルが作成できる．また，$\boldsymbol{A}'$ と $D_{C'}$ の要素数はそれぞれ $|C \cup \{c\}|$ を超えない． □

5.3 コンポーネント削除にともなう再構成

完全な仮想システム状態の集合 $\boldsymbol{A}$ が存在するコンポーネント集合 $C = \{c_1, \cdots, c_n\}$ と C の要素辞書 D_C を考える．C からコンポーネント c を削除する場合を考える．このとき，$C - \{c\}$ に含まれる任意のコンポーネントを特定するのに不要な要素 $\langle I, O, A_l \rangle$ が D_C に含まれる場合がある．要素辞書のサイズを $n-1$ 以下にするためには，この要素を D_C から削除する必要がある．再構成後の要素辞書には，仮想システム状態 A_l を用いて構成される要素 $\langle I, O, A_l \rangle$ が含まれない場合がある．このとき，任意のコンポーネント c_i, c_j は，A_l 以外のいずれかの仮想システム状態 $A_{l'}$ ($\in \boldsymbol{A}$) について，$diff(A_{l'}, c_i, c_j) \neq \phi$ が成立する．よって $\boldsymbol{A} - \{A_l\}$ は $C - \{c\}$ について完全であるため，A_l は不要である．また仮想システム状態の数を $n-1$ 以下にするためにも，A_l を $\boldsymbol{A}$ から削除する必要がある．

以下に C からコンポーネント c を削除した場合の要素辞書および仮想システム状態集合の再構成アルゴリズム

を提案する．

要素辞書および仮想システム状態集合の再構成アルゴリズム（コンポーネント削除時）

入力：要素辞書木 T_C，要素辞書 D_C，仮想システム状態 A，コンポーネント c

出力：要素辞書木 $T_{C'}$，要素辞書 $D_{C'}$，仮想システム状態 A'

(1) 削除する集合 c に対応する葉節点を v とし，その親節点を v_p とする．

(2) v_p に対応する要素 $\langle I, O, A_l \rangle$ と同一の要素が対応する節点が T_G 中に存在しないならば $D'_C \leftarrow D_C - \{\langle I, O, A_l \rangle\}$．

(3) A_l を用いて構成される他の要素 $\langle I', O', A_l \rangle$ が $D_{C'}$ に存在しないならば $A \leftarrow A \cup \{A_l\}$ とする．

(4) v 以外の v_p の子節点を v' とする．

(5) 節点 v_p, v および枝 (v_p, v)，(v_p, v') を削除する．

(6) v_p が T_C の根節点ならば v' を新たな根節点とする．

(7) v_p が根節点でなければ，v_p の親節点 v_{gp} について枝 (v_{gp}, v_p) を (v_{gp}, v') に変更する． □

ステップ (1) で削除の候補となる要素に対応する節点を検索する．ステップ (2) において，検索された要素が削除可能であれば要素辞書から削除する．ここで，削除される要素は高々1個である．ステップ (3) において $\langle I', O', A_l \rangle$ が要素辞書に存在しなければ，A' は $C - \{c\}$ について完全である．よって，A_l を削除できる．また，$\langle I', O', A_l \rangle$ が存在する場合は，A に含まれる仮想システム状態数 m と C に含まれるコンポーネントの数 n について $m < n - 1$ が成立する．以下のステップでは，要素辞書木を再構成する．

例 7 例 3 で用いた仮想システム状態の集合 $A = \{A_1, A_2\}$ とクエリ集合 $C = \{q_2, q_3, q_4\}$ を考える．C の要素辞書を $\{\langle\langle$ 鈴木 $\rangle, A_1\rangle, \langle[\langle$ 斎藤 $\rangle, \langle$ 鈴木 $\rangle], A_2\rangle\}$ とする．C からクエリ q_2 を削除する．要素辞書および仮想システム状態集合の再構成アルゴリズム (コンポーネント削除時) のステップ (1) において求められた v_p に対応する要素は，$\langle\langle$ 鈴木 $\rangle, A_1\rangle$ である．$\langle\langle$ 鈴木 $\rangle, A_1\rangle$ は $g(A_1, q_3)$, $g(A_1, q_4)$ のいずれにも含まれないため，q_3, q_4 を区別するのに不要である．よって，これを要素辞書から削除する．再構築された要素辞書は，A_1 を用いて構成される要素を含まない．よって，A_1 を A から削除する．再構成後の要素辞書および仮想システム状態集合の要素数はいずれも $|C - \{c\}|$ を越えない． □

6 おわりに

本稿では，コンポーネントの仕様を集合を用いて表現するために，仮想的なシステム状態を仮定した．コンポーネントの仕様は，仮想システム状態上でコンポーネントを実行したときの実行結果を用いて表現する．ところが，単一の仮想システム状態を用いた場合は，任意のコンポーネント集合 C について，完全である仮想システム状態が存在するとは限らない．仮想システム状態の集合を導入すると，任意のコンポーネント集合に対して完全である仮想システム状態の集合が常に存在することが示された．そこで任意のコンポーネント集合 C について，完全な仮想システム状態集合 A を構築するためのアルゴリズムを提案した．A に含まれる仮想システム状態数は，$|C| - 1$ を越えない．また，仮想システム状態を適用するために，コンポーネントを用いた検索方式を拡張した．

我々は，サンプルを用いたクエリ検索システムを構築している [4]．今後は，仮想システム状態を適用するためのシステムの拡張を行う予定である．

参考文献

[1] Murata, Kakeshita, "Specification-based component retrieval by means of examples", The Proc. the 1999 international symposium on database applications in non-traditional enverонments", pp. 411-420, 1999.

[2] 村田, 掛下, "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.3, pp.60–67, 1999

[3] 村田, "仕様に基づいたコンポーネント検索に関する研究", 佐賀大学大学院工学系研究科博士論文, 2000.

[4] 村田, 掛下, "仕様に基づいた RDB クエリ検索システム", 情報処理学会データベースシステム研究会, 119–49, pp.291–296, 1999