設計データベースのための導出関連ビューの実現

徐 海燕[†] 古川哲也[††] 上林弥彦[†††]

種々の工業製品の高度化・大規模化に伴い，設計過程で生成される大量の設計データを設計データベースで統一的に管理することが極めて重要となってきている．設計データが複雑に関連しているため，設計データは，設計オブジェクトを節点に，データ関連を枝に対応させた有向グラフによって表す必要がある．共同設計作業において，各設計者は一般に一部の設計オブジェクトしか扱わないが，各自の作業環境（ビュー）中のデータ関連は，概念スキーマ上の複数の関連で構成される長さが不定の経路から写像される必要がある．従来の関係代数は推移律を表現できないため，この種のビューを実現できない．簡単には演繹データベースのDatalogなどの言語を利用する方法が考えられるが，本問題は有向グラフ中の経路探索と考えることができ，それに特有の効率化を図るため，本論文では，正規表現による実現方法を提案する．具体的には，正規表現に基づいて導出関連ビューと質問の定義方法を導入し，ビュー上の質問を概念スキーマ上の等価な質問へ変換する方法と概念スキーマ上の質問の探索空間を減らすことに基づいて効率化を図った処理方法を示す．

Realization of Derived Relationship Views for Design Database Systems

Haiyan Xu,[†] Tetsuya Furukawa[††] and Yahiko Kambayashi[†††]

In a design database system, complicated data representing design objects as well as various relationships among them should be handled. If we restrict relationships to be binary, we can represent design data by a graph whose vertices and edges correspond to design objects and relationships, respectively. In order to support the cooperation of many designers, flexible views are essential. There are views which cannot be defined by subgraphs, since derived relationships not explicitly expressed must be considered. We introduce such view functions using regular expressions. Utilizing the property of regular expressions, we then show the algorithm to translate queries on views into the equivalent queries on their underlying schemata and the algorithm to process queries on the conceptual schema efficiently.

1. はじめに

種々の設計過程において，大量の設計データが生成される．しかし，各設計者は一般に一部の設計しか担当しないので，必要なデータを自分にとって適切な形にして扱う必要がある．このため，設計データを扱う種々のデータベース（DB）であるCAD用DB[1),3),7),10)]やエンジニアリングDB[2),6)]（本論文ではそれらを総称して設計DBという）は，各種のデータを統一的に管理するだけでなく，各々の設計者が必要とする作業環境も提供できなければならない．さらに，設計中のデータは頻繁に変更されるので，すべての設計者に共通な概念スキーマ上の変更を各設計者の作業環境に即時に反映できなければならない．DBのビュー機能は，このような作業環境を提供するための適切なツールであるが[9)]，設計DBにおいては，下記に示すような高水準ビュー機能を必要とする．

設計データの間に存在する履歴関連（更新の過程を記述）や集約関連（部品階層に関する情報を記述）等の種々の関連は，設計データを利用するために極めて重要な情報となる．各種のデータ関連を2項関連で記述すると，概念スキーマやビュー上の設計データは，節点で設計オブジェクト，枝でそれらの間の関連を表した有向グラフによって表すことができる．一般に各設計者は概念スキーマ上の一部の設計オブジェクトしか扱わないが，それらの間の関連を完全に把握するためには，概念スキーマ上の長さが一定でなく，かつ

† 福岡工業大学工学部電子工学科
Department of Electronics, Fukuoka Institute of Technology
†† 九州大学大型計算機センター
Computer Center, Kyushu University
††† 京都大学工学部高度情報開発実験施設
Integrated Media Environment Experimental Laboratory, Faculty of Engineering, Kyoto University

1種類以上の枝からなるような経路に対応する枝をビューの中で提供する必要がある.

例えば,あるVLSI回路設計プロジェクトにおいて,レイアウト設計を担当するグループは,論理設計を担当するグループによって作成された設計オブジェクトのうち,論理シミュレーションが終了したものを入力として操作を行うとする.このとき,**図1**(a)に示されているような設計オブジェクトのうち,v_i と v_{i+4} のみが論理シミュレーションを通過しているとすると,レイアウトグループは v_{i+1}, v_{i+2} と v_{i+3} を操作する必要はないが,修正過程を把握するためには,推移的な履歴関連 (v_i, v_{i+4}) をレイアウトグループの作業環境に提供する必要がある(図1(b)).

従来の関係代数は推移律を表現できないため[9],このようなビューを定義することができない.簡単には演繹DBで利用される推移律を表現できるDatalogなどの言語を利用する方法が考えられるが,本問題のための効率化を必要とする.Datalogプログラムのすべてのクラスに有効な質問処理の効率化が難しく,各々の限定されたクラスに対して効率化を行っているのが現状である.一方,従来の設計DBにおいては,図1(b)のような導出関連を物理的に記憶していた[3),7)].したがって,データ関連の変更は不整合を引き起こす可能性がある.本論文では,正規表現の次のような性質を利用して図1(b)のような導出関連ビューの効率良い実現を図る.

(1) 正規表現は,推移的関連や複数の種類の関連による導出関連を簡単に表現できる.

(2) 正規表現は,代入などの多くのよく使われる演算に関して閉じている[5].

(3) 任意の正規表現に対して,それの表す集合を受理する有限オートマトンが存在する[5].このため,既存のグラフ関係のアルゴリズムを利用できる.

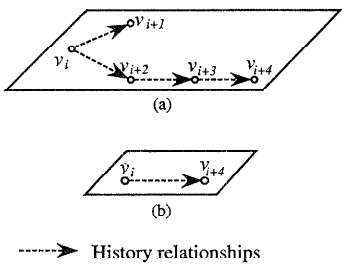


図1 レイアウトグループが必要とする作業環境
Fig. 1 The working environment required by the layout group.

グラフに基づくデータモデル上で正規表現を使用した質問言語 G^+ [8], GraphLog[4]などが提案されている.これらの研究では質問の複雑さや記述能力を対象としており,G^+ で記述される質問の答も本論文と異なる.本論文では,正規表現を用いて導出関連ビューと質問を定義する.さらに,性質(2)を利用してビュー上の質問を概念スキーマ上の等価な質問へ変換する方法を与える.概念スキーマ上の質問処理は,概念スキーマと質問の表す集合を受理する有限オートマトンとの交差グラフを検索することによって行うことができる[8]が,本論文では,それを行う $O(n+e)$ の質問処理アルゴリズムを示す.ただし,n と e はそれぞれ概念スキーマ内の節点の数と枝の数である.

ビューには,一般的にビュー上の更新を元のDBに反映される方法が唯一とはならないビュー更新問題が存在するが,データ関連の変更が設計オブジェクトの変更により生じ,かつ導出関連ビュー上の設計オブジェクトが概念スキーマ上の設計オブジェクトの部分集合であるため,本論文の提案する導出関連ビューには更新問題が存在しない.

本論文は,以下のように構成されている.2章では,設計データ間の関連の基本的な性質をまとめる.データ関連としては,履歴関連,集約関連,対応関連と交替関連について検討する.3章では,設計DBのための正規表現を定義し,それに基づく導出関連ビューとビュー上の質問を定義する.ビュー上の質問を概念スキーマ上の等価な質問へ変換する方法は4章で示す.概念スキーマ上の質問の効率良い処理方法は5章で与える.6章は全体のまとめを行う.

2. 設計データグラフを用いたデータ関連表現

設計データには設計結果の実体を記述する設計オブジェクトと,設計オブジェクト間のつながりを記述する関連が存在する.本論文では,すべての関連を2項関連で表す.n 項関連も変数を導入することにより,2項関連で表せるため,一般性は失われない.すべての関連を2項関連で表せば,設計データはラベルつき有向グラフによって表現できる.

定義 1 設計データグラフ G は (V, E) によって記述される.ただし,$V=\{v_i \mid v_i$ は設計オブジェクトのID$\}$, $E=\{v_i R v_j \mid v_i$ から v_j への種類 R の関連が存在する$\}$.

設計結果の実体は,仕様設計,論理回路設計,レイ

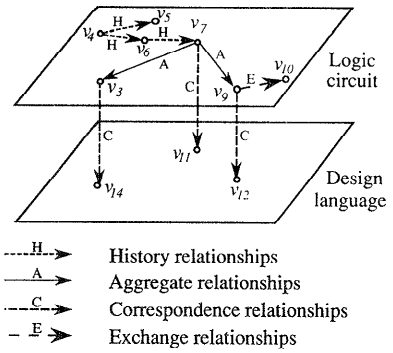


図 2 設計データの有向グラフによる表現
Fig. 2 Representation of design data by directed graphs.

アウト設計といったような多段階の表現レベル中のどれかに属し，かつ種々のテストもなされるので，設計結果の実体を記述する設計オブジェクトには，属する表現レベル，テスト結果，作成者，作成時刻などの情報も含まれる．

図 2 にラベルつき有向グラフによって表される設計データの例を示す．各種のデータ関連は以下のことを意味する．

(1) 履歴関連 v_iHv_j (History relationship).
設計オブジェクト v_j は設計オブジェクト v_i を更新したものであるという履歴情報を記述する．

(2) 集約関連 v_iAv_j (Aggregate relationship).
設計オブジェクト v_j は設計オブジェクト v_i の部品オブジェクトであるという部品階層に関する情報を記述する．

(3) 対応関連 v_iCv_j(Correspondence relationship).
v_i と v_j は同じ設計目標を表す上下表現レベルにある設計オブジェクトであるという表現レベルに関する情報を記述する．

(4) 交替関連 v_iEv_j (Exchange relationship).
設計オブジェクト v_i は v_j と交替できる設計オブジェクトであるという情報を記述する．

4種類の関連とも推移的である．例えば，設計オブジェクト v_k が v_j を更新したものであり，v_j が v_i を更新したものである時，v_k は v_i を更新したものでもある．さらに，複数のデータ関連によって新しいデータ関連も導出できる．**図 3** は，対応関連と交替関連や履歴関連によって導出されるデータ関連の一つの例を示している．同じ表現レベルにある二つの設計オブジェクトが交替関連 (図3(a))あるいは履歴関連 (図3(b))を持つなら，対応関連で関連している下位の同一表現レベルにある設計オブジェクト間にも同種類の関連が存在する．すなわち，図3(a)と(b)においては，v_3 から v_4 へ，または，v_3' から v_4' へ，導出される交替関連または履歴関連が存在する．

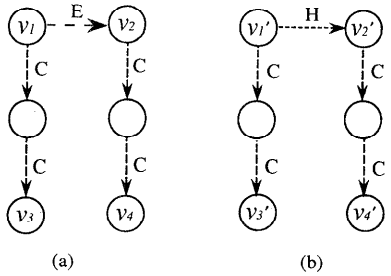


図 3 対応関連と他の関連による導出関連
Fig. 3 New relationships derived by correspondence and other relationships.

あるデータ関連の集合 T に対して，閉包 T^* は T を含み T から導かれるすべての関連の集合である．$U \subseteq T$, かつ $U^* = T^*$ の時，U は T の被覆といい，被覆の中で要素の数が最小のものを極小被覆という．一般に，極小被覆は唯一でないが，Tから導かれるすべての関連を表すための最も簡単なものである．

データ関連の閉包を物理的に記憶すると，データ量が増えるだけでなく，設計オブジェクトの変更に伴うデータ関連の変更のオーバヘッドも大きい．これはデータ関連の変更時に，閉包内の基本的なデータ関連とそれらにより計算できるデータ関連間の一貫性の保全を行わなければならないためである．DB の空間の節約とデータ関連変更のオーバヘッドの減少のため，本論文では，概念スキーマに各関連の極小被覆しか記憶しないという仮定の下で検討を行う．

3. 導出関連ビューとビュー上の質問

本章では，正規表現に基づいて，設計データグラフのための質問と導出関連ビューを定義する．

定義 2[5] Σ をアルファベットとする．Σ 上の正規表現とその表現の表す集合を，次のように帰納的に定義する．

(1) ϕ は正規表現で，その表す集合は空集合である．

(2) ε は正規表現で，その表す集合は $\{\varepsilon\}$ である．

(3) Σ の各要素 a は正規表現で，その表す集合は $\{a\}$ である．

(4) 正規表現 r と s の表す集合をそれぞれ R と S

とした時，$(r+s)$, $(r\cdot s)$ と (r^*) は正規表現で，それぞれ集合 $R\cup S$, RS (連接)，R^* (閉包) を表す．

以後 $\|r\|$ で正規表現 r の表す集合を記述する．また，集合 $S=\{s_1, s_2, \cdots, s_n\}$ に対して，正規表現中の集合記号 S は $(s_1+s_2+\cdots+s_n)$ を記述する．さらに，正規表現に必要でない括弧は外す．

設計データグラフのための正規表現を定義する前に，ある性質を満たす節点集合を記述するために，節点ラベル関数を導入する．

定義 3 設計データグラフ $G=(V, E)$ とアルファベット Σ に対して，節点ラベル関数 L は，V から Σ への関数であり，V の要素に対する条件 $Cond_i$ と Σ の要素 γ_i の組の集合 $\{(\gamma_i, Cond_i)\}$ で表す．ただし，

(1) $L(v_j)=\gamma_i$, if v_j が条件 $Cond_i$ を満たす．

(2) v_j が $Cond_i$ を満たすなら，v_j は $Cond_k$ $(k\neq i)$ を満たさない．

(3) $\forall v_i\in V\ (\exists \gamma_k\in\Sigma$ s.t. $L(v_i)=\gamma_k)$.

設計データグラフ上の経路は節点と枝が交互に現れるので，そのような経路集合を表す正規表現を定義する．

定義 4 設計データグラフ $G=(V, E)$ と節点ラベル関数 L における正規経路表現 q は，$\Sigma_V\cup\Sigma_E$ 上の正規表現であり，かつ条件 $\|q\|\subseteq\|(\Sigma_V\cdot\Sigma_E)^*\Sigma_V\|$ を満たす．ただし，$\Sigma_{VL}=\{L(v_i)|v_i\in V\}$，$\Sigma_V=V\cup\Sigma_{VL}$，$\Sigma_E=\{R|v_iRv_j\in E\}$ である．

すなわち，正規経路表現が表す集合の要素には，節点と枝のラベルが交互に現れる．例えば，v_0 を V 内の要素，節点ラベル関数 L を $\{(\gamma_a$, 作成者=「斉藤」$)$, $\cdots\}$，H を Σ_E 内の要素とすると，$v_0H\gamma_a$ は，v_0 を始点，ラベル γ_a を持つ節点を終点とするラベル H の枝を表す正規経路表現である．もし，節点ラベル関数を用いないと，「斉藤」によって作成されるすべての設計オブジェクトの名前を列挙しなければならない．節点ラベル関数はこのような場合に有効である．

設計データブラフ $G=(V, E)$ 内の v_{i1} から v_{ij} への経路 $P: v_{i1}R_{i1}v_{i2}, v_{i2}R_{i2}v_{i3}, \cdots, v_{ij-1}R_{ij-1}v_{ij}(v_{ik}, v_{ik-1}\in V,\ v_{ik-1}R_{ik-1}v_{ik}\in E)$ に対して，$(v_{i1}+L(v_{i1}))R_{i1}(v_{i2}+L(v_{i2}))R_{i2}\cdots R_{ij-1}(v_{ij}+L(v_{ij}))$ をその経路の表す正規表現 q_P という．

定義 5 設計データグラフ $G=(V, E)$ と節点ラベル関数 L での質問は，正規経路表現 q であり，(G, L, q) で表す．質問の答は，集合 $\{(v_i, v_j)|\|q_P\|\cap\|q\|\neq\phi$ であるような v_i から v_j への経路 P が存在する$\}$ である．

定義 6 ビューは，(G, L, S_V, Q_E, L_E) によって定義される設計データグラフ G' である．ただし，

(1) $G=(V, E)$ は設計データグラフ，

(2) L は G の節点ラベル関数，

(3) $S_V\subseteq\Sigma_V$,

(4) $Q_E=[q_1, \cdots, q_m]$ は G と L における正規経路表現からなるリスト，

(5) $L_E=[l_1, \cdots, l_m]$ は枝ラベルのリスト．

$G'=(V', E')$ は下記のように構成される．$V'=\{v_i|v_i\in V\wedge((v_i\in S_V)\vee(L(v_i)\in S_V))\}$，$E'=\{v_iRv_j|(v_i, v_j\in V')\wedge((v_i, v_j)$ が質問(G, L, q_k) の答に含まれる$)\wedge(R=l_k)\ (1\leq k\leq m)\}$．

ビューは設計データグラフであるため，ビュー上に新たなビューを定義できる．ビューを定義する設計データグラフをそのビューの基底スキーマという．次では，例を用いてビューの定義法を示す．

例 1 図1の例におけるレイアウト設計を担当するグループの論理設計を担当するグループによって作成されたデータに対するビューは，以下のように定義することになる．

G: 概念スキーマ (ビューの基底スキーマ)，

$$L=\left\{\begin{array}{l}(X,\ \text{テスト結果}=\text{“ok”}),\\(Y,\ \text{テスト結果}\neq\text{“ok”})\end{array}\right\}$$

$S_V=\{X\}$,

$Q_E=[XH(YH)^*X]$,

$L_E=[H']$.

図1(a)中の v_i と v_{i+4} はテストを通過しているので，ラベル X，その他の節点はテストを通過していないので，ラベル Y と付けられる．$XH(YH)^*X$ はラベル X を持つ二つの節点を接続する経路を定義しており，経路中の枝のラベルが H であり，節点があれば，そのラベルは Y である．それらの経路はビューの上で H' とラベル付けられた枝となる．図1(a)中の v_i から v_{i+4} への経路 P の表す正規表現 q_P は，$\|q_P\|\cap\|XH(YH)^*X\|\neq\phi$ であるため，その経路がビュー中の v_i から v_{i+4} へのラベル H' を持つ枝に写像される．

例 2 図3(b)のためのビュー定義 (C^r は逆方向の対応関連を表している)．

G: 概念スキーマ (ビューの基底スキーマ)，

$$L=\left\{\begin{array}{l}(T_u,\ \text{表現レベル}>v_3\ \text{の表現レベル}),\\(T,\ \text{表現レベル}=v_3\ \text{の表現レベル}),\\(T_d,\ \text{表現レベル}<v_3\ \text{の表現レベル})\end{array}\right\},$$

$S_V=\{T\}$,

$Q_E=[T(C^rT_d)^*H(T_dH)^*(T_dC)^*T]$,

$L_E=[H']$.

ここで，$T(C^rT_d)^*H(T_dH)^*(T_dC)^*T$ の意味は以下のとおりである．

(1) T: T 中の節点により開始し，

(2) $(C^rT_d)^*$: 表現レベルを上げてゆき，

(3) $H(T_dH)^*$: H あるいは推移的な H による関連があり，

(4) $(T_dC)^*$: 表現レベルを下がり，

(5) T: ラベル T を持つ節点で終了する．

すなわち，上位表現レベルに存在する履歴関連が下位レベルに写像される．

4. 導出関連ビュー上の質問処理

3章では，導出データ関連を抽出できるビュー機能を導入した．ビューは物理的に存在しないため，ビュー上の質問を，ビューが物理的に存在した場合と同じ答を出す概念スキーマ上の等価な質問に変換しなければならない．従来のDBにおいては，質問をビュー定義と合併すれば良いが，導出関連ビューはその基底スキーマの部分グラフではないため，従来の方法はそのまま適用できない．本章では，正規表現の性質を利用したビュー上の質問の変換方法を示す．

定義 7[5] **代入** f は，Σ_1 から Σ_2^* への関数であり，$a\in\Sigma_1$ を $f(a)$ で置き換える操作である．これを Σ_1^* 上の関数に次のように拡張する：$f(\varepsilon)=\varepsilon$, $f(ax)=f(a)f(x)$.

正規表現が代入に関して閉じており，任意の正規表現 r に対して $\|r\|$ を受理する非決定性有限オートマトン (NFA) が存在し，かつ NFA で受理される集合は正規表現で表される[5]．それらの性質を利用した，ビュー上の質問をその基底スキーマ上の等価な質問に変換する方法を，まず例で示す．

例 3 ビュー G' は次のような (G, L, S_V, Q_E, L_E) より定義されるとする．

G=概念スキーマ（ビューの基底スキーマ），

$$L=\left\{\begin{matrix}(X,\ \text{設計者}=\text{“斉藤”}),\\(Y,\ \text{設計者}\neq\text{“斉藤”})\end{matrix}\right\},$$

$S_V=\{X\}$,

$Q_E=[X(HY)^*HX]$,

$L_E=[H']$.

上記のように構成されるビュー G' において，ある設計者がテストを通過した設計オブジェクト間の推移的な履歴関連を検索したいとする．答をできるだけ小さくするため，他の関連から推移的に得られる関連は答から除くようにする．したがって，テストを通過した節点からテストを通過していない任意個の節点と H' の枝を介してつながるテストを通過した節点の組を答とする．この要求は下記のようなビュー G' 上の質問 (G', L', q') によって表現される．

$$L'=\left\{\begin{matrix}(T,\ \text{テスト結果}=\text{“ok”}),\\(N,\ \text{テスト結果}\neq\text{“ok”})\end{matrix}\right\},$$

$q'=T(H'N)^*H'T$.

ビュー上の質問をその基底スキーマ上の等価な質問へ変換する過程が**図4**と**5**に示されている．図4(b)と(a)はそれぞれビューを定義するための正規経路表現の表す集合を受理するNFAと，ビュー上の質問の正規経路表現の表す集合を受理するNFAを示している．図4全体は代入を示している．H' を $X(HY)^*HX$ で置き換える代入によって，関連 H' がそれの対応する基底スキーマ上の経路 $X(HY)^*HX$ に変換される．

代入の結果が，図5(a)に示されている ε-動作を含むNFAである．等価なNFAでε-動作を含まないものが図5(b)に示されている．図5(b)からは，それぞれ Σ_V と $\Sigma_{V'}$ に属している節点ラベルを持つ状態遷移が隣接していることがわかる（例えば，s_0 から出る T と X という節点ラベルを持つ二つの状態遷移）．ビュー上の正規経路表現をその基底スキーマ上の正規経路表現へ変換するためには，質問の節点ラベル関数 L' とビュー定義の節点ラベル関数 L を合併しなければならない．この例における合併された新たな節点ラベル関数 L'' は以下のようになる．

$$L''=\left\{\begin{matrix}(XN,\ \text{設計者}=\text{“斉藤”}\wedge\text{テスト結果}\neq\text{“ok”}),\\(XT,\ \text{設計者}=\text{“斉藤”}\wedge\text{テスト結果}=\text{“ok”}),\\(YN,\ \text{設計者}\neq\text{“斉藤”}\wedge\text{テスト結果}\neq\text{“ok”}),\\(YT,\ \text{設計者}\neq\text{“斉藤”}\wedge\text{テスト結果}=\text{“ok”})\end{matrix}\right\}$$

上記の合併された新たな節点ラベル関数に従って，節点ラベルを持つ遷移の合併を行うと，ε-動作を含ま

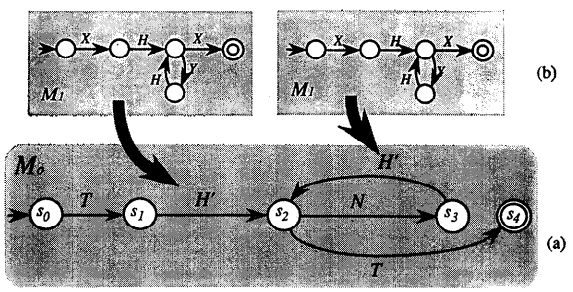
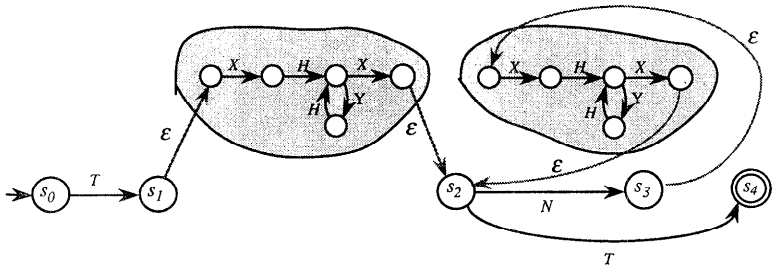
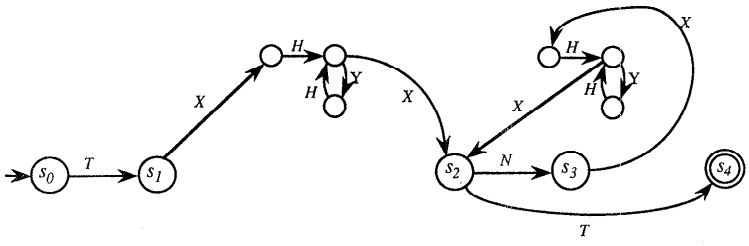


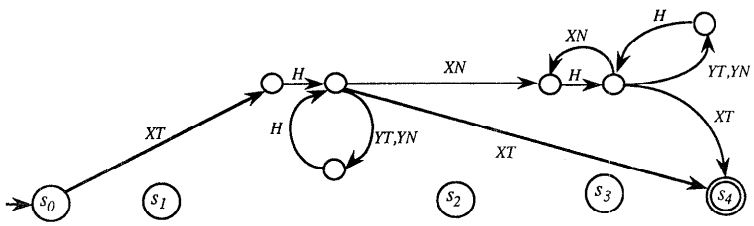
図 4 例3における代入
Fig. 4 The substitution of Example 3.



(a)



(b)



(c)

図 5 ビュー上の質問の処理過程
Fig. 5 The processes of conversion of the query on a view.

ない NFA (図5(b)) は図5(c)のような NFA に変換される．すなわち，基底スキーマ上の正規経路表現の表す集合を受理する NFA を構成できた．

アルゴリズム 1：ビュー上の質問の変換．

入力：ビュー G' の定義：(G, L, S_V, Q_E, L_E)；
ビュー G' の質問：(G', L', q')．

出力：G 上の質問：(G, L'', q'')

計算：/*代入 f を構成*/

for each l_i in L_E **do** $f(l_i) := q_i$；

for each vertex label $x \in \Sigma_{V'}$ **do** $f(x) := x$；

$f(q')$ を計算；$L'' = \phi$；

$\|f(q')\| - \{\varepsilon\}$ を受理する NFA$=(S, \Sigma, \delta, s_0, F)$ を構成；

その NFA に対して，下記のような節点ラベルを持つ遷移を置き換える操作を行う．

(1) s_0 から出る節点ラベル x と y を持つ隣接する遷移，または $s_f \in F$ で終了する節点ラベル y と x $(x \in \Sigma_{V'}, y \in \Sigma_V)$ を持つ隣接する遷移を，下記のように得られる $\Sigma_V^* = V \cup \{L''(v_i) | v_i \in V\}$ 内の新たな節点ラベル r を持つ遷移と置き換える．

if $((x, Cond_1) \in L) \wedge ((y, Cond_2) \in L')$ **then**
begin
$L'' := L'' \cup \{(a\ new\ label,\ Cond_1 \wedge Cond_2)\}$；
r :=the new label；
end
if $(x \in V) \wedge ((y, Cond) \in L') \wedge$ (x が条件 $Cond$ を満たす)

then $r:=x$;
if $((x, Cond)\in L)\wedge(y\in V')\wedge($$y$が条件 $Cond$を満たす)
then $r:=y$;
if $(x\in V)\wedge(y\in V')\wedge(x-y)$ **then** $r:=x$;

次に，Σ_V, $\Sigma_{V'}$ と Σ_V に属する節点ラベル u, y と w を持つ隣接する遷移は，u と w から下記のように x を計算してから，上記のように x と y から得られる Σ_V^q 内の新たな節点ラベル r を持つ新たな遷移と置き換える．

if $u=w$ **then** $x=u$ **else**
begin
if $(u\in V)\wedge((w, Cond)\in L)\wedge($$u$が条件$Cond$を満たす)
then $x=u$;
if $(w\in V)\wedge((u, Cond)\in L)\wedge($$w$が条件$Cond$を満たす)
then $x=w$;
end

(2) Σ_E, Σ_V と Σ_E に属するラベルを持つ隣接する遷移中の Σ_V に属するラベルを持つ遷移を，Σ_V^q 内の同一条件を表すラベル列を持つ遷移と置き換える．

最後に，NFA からそれの受理する集合を表す正規表現 q'' を生成する． □

定理 1 アルゴリズム1は，(G, L, S_V, Q_E, L_E) によって定義されるビュー G' 上の質問 (G', L', q') を，正しく基底スキーマ G 上の等価な質問 (G, L'', q'') に変換する．

証明：アルゴリズム1は，次の二つの操作を行っている．

(1) 代入 $f(q')$: q' 内の各枝ラベルを基底スキーマ G 上の対応する経路を表す正規経路表現で置き換える操作と，

(2) $\|f(q_i')\|$ を受理する NFA$=(S, \Sigma, \delta, s_0, F)$ に，隣接する節点ラベルを持つ遷移を，それらの節点ラベルの対応する条件の論理積からなる条件で定義される Σ_V^q 内のラベルを持つ遷移で置き換え，Σ_V に属する節点ラベルを持つ遷移を，同一条件を表す Σ_V^q 内のラベル列を持つ遷移で置き換える操作．

ビュー G' が物理的に存在している場合に，(v_{i1}, v_{im}) が質問 (G', L', q') の答に属するための必要十分条件は，G' に $\|q_p\|\cap\|q'\|\neq\phi$ であるような v_{i1} から v_{im} への経路 P: $v_{i1}l_{i1}v_{i2}, v_{i2}l_{i2}v_{i3}, \cdots, v_{im-1}l_{im-1}v_{im}$ が存在することである．さらに，ビュー G' に $v_{ik}l_{ik}v_{ik+1}$ が存在するための必要十分条件は，ビューの基底スキーマ G に $\|q_{Tk}\|\cap\|q_{ik}\|\neq\phi$ であるような v_{ik} から v_{ik+1} への経路 T_k が存在することである．したがって，(v_{i1}, v_{im}) が質問 (G', L', q') の答に属するための必要十分条件は，Gに v_{i1} から v_{im} への v_{ik} $(1\leq k\leq m)$ を順次通過する経路が存在し，その中の各 v_{ik} から v_{ik+1} への部分経路 T_k は $\|q_{Tk}\|\cap\|q_{ik}\|\neq\phi$ $(1\leq k\leq m-1)$ であり，かつ各 v_{ik} は G' 上の正規経路表現 q' によって表される節点に対する条件も満たすことである．アルゴリズム1の行う操作より，(v_{i1}, v_{im}) が質問 (G', L', q') の答に属するための必要十分条件は，Gに $\|q_{P'}\|\cap\|q''\|\neq\phi$ であるような v_{i1} から v_{im} への経路 P' が存在することとなる． □

5. 概念スキーマ上の質問処理

概念スキーマ G 上の質問 (G, L, q) の処理は，従来の言語理論の結果を利用すると，次のように行うことができる[8]．

(a) $\|q\|-\{\varepsilon\}$ を受理する NFA$=(S, \Sigma, \delta, s_0, F)$ を構成する．

(b) NFA と $G^c=(V^c, E^c)$ の交差グラフ☆を構成する．ただし，$V^c=\{v_i^b\,|\,v_i\in V, b=1, 2\}$,
$E^c=\{v_i^2Rv_j^1\,|\,v_iRv_j\in E\}$
$\cup\{v_i^1Rv_i^2\,|\,R=L(v_i)\}$
$\cup\{v_i^1v_iv_i^2\,|\,v_i\in V\}$.

(c) 交差グラフにおいて，(v_i^1, s_0) から始まり，(v_j^2, s_f) $(s_f\in F)$ で終了する経路を検索する．そのような経路が存在するならば，(v_i, v_j) が質問の答に属する．かつ (v_i, v_j) が質問の答に属するのは，その時に限る．

例えば，**図6**(a)のGに対する $G^c=(V^c, E^c)$ は，図6(b)に示されている．ただし，ここで，$L(v_0)=X$, $L(v_1)=John$ とする．質問を表した NFA を図6(c)とすると，それと図6(b)の G^c との交差グラフは図6(d)に示した．

G 内の節点の数を n，NFA 内の状態数を m とすると，交差グラフ内の節点の数は $n\times m$ になる．節点間のすべての経路を検索できる $O((n\cdot m)^3)$ のアルゴリズムが知られているので[5]，上記の質問処理は少なくとも $O(n^3)$ 時間内に完成できる $(m\ll n$ とする$)$．

☆ $G_1=(V_1, E_1)$ と $G_2=(V_2, E_2)$ から構成される交差グラフ $G=(V, E)$ は，$V=V_1\times V_2$, $E=\{((v_i, v_j), (v_k, v_t))\,|\,(v_i, v_k)\in E_1\wedge(v_j, v_t)\in E_2\}$ である．

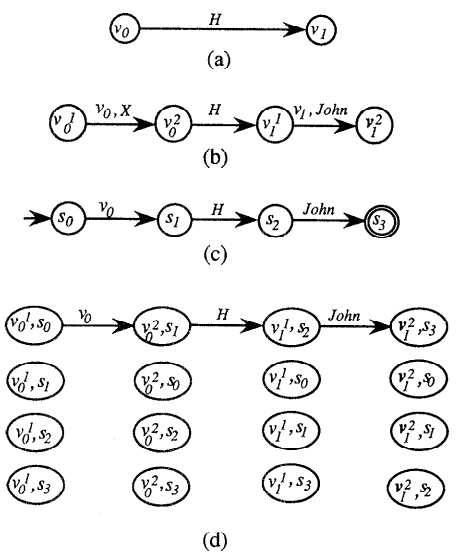


図 6 交差グラフの構成例
Fig. 6 An example of how to construct the intersection graphs.

本章では，良く知られる深さ優先探索を拡張して，$O(n+e)$ 時間内で交差グラフから質問の答となる経路を検索できる効率的なアルゴリズムを与える．深さ優先探索は，すべての節点あるいは枝を1回のみ検索するので，まずそれを交差グラフ内の各節点 $[v_i^b, s_k]$ に対して，次のような検索結果を記憶するように拡張する．

$$result[v_i^b, s_k] = \left\{ v_j^2 \middle| \begin{array}{l}((v_i^b, s_k) \text{ から深さ優先探索で } (v_j^a, s_l)\\ \text{に到着可能}) \wedge (s_l \in F) \wedge (a=2).\end{array} \right\}.$$

$$up[v_i^b, s_k] = \left\{ (v_j^a, s_l) \middle| \begin{array}{l}(v_i^b, s_k) \text{ と } (v_j^a, s_l) \text{ が同じループに}\\ \text{属し，かつ } (v_j^a, s_l) \text{ の深さ優先探}\\ \text{索番号がそのループにおいて最小}\\ \text{である．ただし，} (v_i^b \neq v_j^a) \vee (s_k \neq s_l)\end{array} \right\}.$$

グラフが非巡回なら，$result[v_i^b, s_k]$ が節点 (v_i^b, s_k) から到着可能なすべての節点から得られる結果を記憶することになるが，巡回グラフにおいては，$result[v_i^b, s_k]$ は答の真部分集合としかならない．例えば，x と y からなる巡回部分と x から z への枝を含む有向グラフにおいて，x，y，z という順で行われる深さ優先探索では，y から z へ到達可能であることを直接得ることができず，このような経路になる始終点の組が含まれていない．この問題を解決するために，深さ優先探索の後，各ループ内の最小深さ優先探索番号を持つ節点 (v_j^b, s_l) の $result[v_j^b, s_l]$ を，そのループ内の他の節点 (v_i^b, s_k) の $result[v_i^b, s_k]$ に合併する．すなわち，全体の処理は次のような四つのステップによって行われる．

(1) 交差グラフを構成するステップ：質問を表した NFA と G から得られる G^c との交差グラフを構成する．

(2) 深さ優先探索：交差グラフに対して通常の深さ優先探索を行いながら，$result[v_i^b, s_k]$，$up[v_i^b, s_k]$ という各節点 (v_i^b, s_k) に対する探索結果を記憶する．

(3) 補正ステップ：$up[v_i^b, s_k]$ を利用して，深さ優先探索の不完全な結果 $result[v_i^b, s_k]$ を補正する．

(4) 最終結果計算ステップ．

アルゴリズム 2 概念スキーマ上の質問処理

入力：質問 (G, L, q)
出力：$result[v_i] = \{v_j\}$．$i=1, 2, \cdots,$
計算：
G と L から $G^c=(V^c, E^c)$ を作成；
$\|q\| - \{\varepsilon\}$ を受理する $NFA=(S, \Sigma, \delta, s_0, F)$ を構成；
NFA と G^c の交差グラフを構成；
交差グラフに対して深さ優先探索を行いながら，各節点 $[v_i^b, s_k]$ に対して，$result[v_i^b, s_k]$ と $up[v_i^b, s_k]$ を記憶；

```
                    /*補正*/
VN = {(v_i^b, s_k) | v_i^b ∈ V^c, s_k ∈ S} ;
while VN ≠ φ do
begin
  ∃(v_i^b, s_k) ∈ VN に対して,
  for each (v_j^b, s_l) ∈ up[v_i^b, s_k] do
  begin
    revision(v_j^b, s_l) ;
    result[v_i^b, s_k] := result[v_i^b, s_k] ∪ result[v_j^b, s_l] ;
  end
  VN := VN − {(v_i^b, s_k)}
end
                    /*最終結果の計算*/
for earh v_i ∈ V do
  result[v_i] := result[v_i^1, s_0] ;
procedure revision(v_i^b, s_k) ;
for each (v_j^b, s_l) ∈ up[v_i^b, s_k] do
begin
  revision(v_j^b, s_l) ;
  result[v_i^b, s_k] := result[v_i^b, s_k] ∪ result[v_j^b, s_k] ;
```

end

$VN := VN - \{(v_i^b, s_k)\}$;

end procedure □

補正操作は再帰的に行われるが，ある節点が複数のループに属している場合でも終了する．また，(v_j^b, s_t) がそのような節点でつながるループの節点中の最小深さ優先探索番号を持つ節点とすると，$result[v_j^b, s_t]$ は必ず完全である．$result[v_j^b, s_t]$ を $result[v_i^b, s_k]((v_j^b, s_t) \in up[v_i^b, s_k])$ に合併する補正操作によって，そのループによって生じる不完全結果が全部補正できる．上記のように補正操作を繰り返していれば，ループにより生じるすべての不完全結果は全部補正され，かつ補正操作が終了する．

すなわち，アルゴリズム2の出力 $result[v_i]$ に対して，$v_j \in result[v_i]$ の必要十分条件は，(v_i, v_j) が質問の答に属することである．

定理 2 アルゴリズム2の出力は，質問の答である．

さらに，アルゴリズム2の効率も次のように定量化できる．グラフ G 内の節点の数を n，枝の数を e，NFA 内の節点の数を m とすると，交差グラフ内の節点の数は $n \times m$ で，枝の数は高々 $m^2 \times e$ である．$m \ll n$ のため，交差グラフ構成のコストおよび深さ優先探索のコストとも $O(n+e)$ である．補正操作は，各ループに対して，ループ内の節点の数−1(＝ループ内の枝の数) 回行うが，ループは深さ優先探索によって見つけているため，異なるループには共通な枝を含まない．このため，補正操作のコストは，$O(e)$ 以下である．最終結果を計算するステップは，$O(n)$ であるため，アルゴリズム2のコストは，$O(n+e)$ である．グラフ内の特定の性質を満たす経路を検索するためには，少なくとも各節点や枝を1回検索する必要があるので，それ以上の効率化は不可能であることが容易に分かる．

6. おわりに

本論文では，設計 DB における導出関連ビューの必要性を示し，そのためのビュー定義機能を導入し，ビュー上の質問をその基底スキーマ上の等価な質問へ変換する方法と概念スキーマ上の効率良い質問処理アルゴリズムを示した．一般に，設計者は一部の設計オブジェクトしか扱わないが，特定の性質を満たす導出データ関連を必要とする．効率化を図るため，本論文では，正規言語を用いて効率良い導出関連ビューの実現方法を提案した．さらに，ビュー機能を用いてこのような作業環境を実現すると，既存の方法がかかえていたいくつかの重要な問題を解決できることも示した．

さらに，本論文の最も重要な貢献は，設計 DB 上の問題をグラフ上の問題として整理できたことである．本手法を応用することにより，設計 DB におけるデータ関連による各種の検索やデータ関連更新時の一貫性の保持というような問題も，既存のグラフアルゴリズムやその簡単な拡張で効率良く解くことができる．

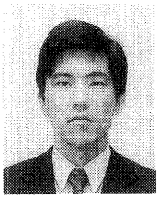
参考文献

1) Barghouti, N. S. and Kaiser, G. E.: Concurrency Control in Advanced Database Applications, *ACM Comput. Surv.*, Vol. 23, No. 3, pp. 269-317 (1991).
2) Bernstein, P. A.: Database Systems Support for Software Engineering—An Extended Abstract, *Proc. 9th Software Eng. Conf.*, pp. 160-178 (1987).
3) Chou, H.-T. and Kim, W.: A Unifying Framework for Version Control in a CAD Environment, *Proc. 12th Int. Conf. on Very Large Data Bases*, pp. 336-344 (1986).
4) Consens, M. P. and Mendelzon, A. O.: GraphLog: a Visual Formalism for Real Life Recursion, *Proc. of the 9th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Syst.*, pp. 404-416 (1990).
5) Hopcroft, J. E. and Ullman, J. D.: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).
6) Katz, R. H.: Toward a Unified Framework for Version Modeling in Engineering Databases, *ACM Comput. Surv.*, Vol. 22, No. 4, pp. 375-408 (1990).
7) Katz, R. H., Chang, E. and Bhateja, R.: Version Modeling Concepts for Computer-Aided Design Databases, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 379-386 (1986).
8) Mendelzon, A. O. and Wood, P. T.: Finding Regular Simple Paths in Graph Databases, *Proc. 15th Int. Conf. on Very Large Data Bases*, pp. 185-194 (1989).
9) Ullman, J. D.: *Principle of Databases and Knowledge-Base Systems*, Vol. 1, Computer Science Press (1988).
10) 宇田川佳久：オブジェクト指向データベースのCADへの応用，情報処理，Vol. 32, No. 5, pp. 585-592 (1991).

(平成4年8月10日受付)
(平成4年12月10日採録)

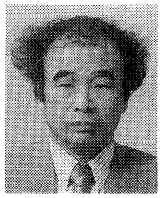
徐　海燕（正会員）

1962年生．1983年中国復旦大学理学部計算機科学科卒業．1990年九州大学大学院博士後期課程修了．工学博士．同年福岡工業大学電子工学科講師，現在に至る．エンジニアリングデータベースの質問処理論，並行処理制御などの研究に従事．電子情報通信学会，日本ソフトウェア科学会各会員．



古川　哲也（正会員）

1959年生．1983年京都大学工学部情報工学科卒業．1985年同大大学院修士課程修了．1988年九州大学大学院博士後期課程修了．工学博士．1988年九州大学工学部助手，1989年同大大型計算機センター講師，現在同助教授．データベースの設計理論，質問処理論などの研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM各会員．



上林　弥彦（正会員）

1943年生．1965年京都大学工学部電子工学科卒業．1970年同大大学博士課程修了．工学博士．京都大学助手，イリノイ大学リサーチアソシエイト，京都大学助教授，九州大学教授を経て1990年より京都大学工学部教授．論理回路・オートマトン，データベースの研究に従事．この間，カナダマッギル大学，クウェート大学，中国武漢大学の客員教授．「Database—A Bibliography」(Computer Science Press)「データベース」(昭晃堂)など．1980年米沢賞，1983年丹羽賞，1987年電子情報通信学会著述賞．本会元理事．電子情報通信学会，日本ソフトウェア科学会，ACM各会員．IEEEシニア会員．