DEWS2003 8-B-05

グラフィカルな問合せ言語処理系における巡回グラフ問合せの実現

Cyclic Graph Query in an Interpreter of the Graphical Query Language

宮崎　則雄† 寶珍　輝尚†† 原　章広†† 都司　達夫†† 樋口　健††

† 福井大学大学院工学研究科　〒 910–8507 福井県福井市文京 3–9–1
†† 福井大学工学部情報・メディア工学科　〒 910–8507 福井県福井市文京 3–9–1
E-mail: †{miyazaki,hochin,hara,tsuji,higuchi}@pear.fuis.fukui-u.ac.jp

あらまし　本論文では，ラベル付き有向グラフで表現されたデータベースに対する問合せ言語 DUO のインタプリタにおける巡回グラフ問合せの実現について述べる．DUO 言語では要素 (点, 枝, 括弧) によって問合せを記述する．従来の DUO 用インタプリタでは，問合せの中から 1 つ点を見つけ，それを問合せグラフとし，問合せグラフの両端の要素のどちらかにつながる要素があればそれをつなげて問合せグラフとするという方法で構文解析を行っていた．この方法では，問合せが巡回を含む場合に対応できない．そこで，問合せ中の枝と括弧のそれぞれに対して，つながる要素を求め，それらを集めて問合せグラフとするという方法に採用した．これにより，巡回を含む問合せの処理が可能になった．

キーワード　グラフィカル言語, グラフデータベース

Norio MIYAZAKI†, Teruhisa HOCHIN††, Akihiro HARA††, Tatsuo TSUJI††, and Ken HIGUCHI††

† Graduate School of Engineering, Fukui University　Bunkyo 3–9–1, Fukui-shi, Fukui 910–8507 Japan
†† Faculty of Engineering, Fukui University　Bunkyo 3–9–1, Fukui-shi, Fukui 910–8507 Japan
E-mail: †{miyazaki,hochin,hara,tsuji,higuchi}@pear.fuis.fukui-u.ac.jp

Abstract　Processing of the cyclic graph query in an interpreter for DUO is described in this paper. DUO is a graphical query language for the database based on labeled directed graphs. A query in DUO is described by elements(nodes, edges and parentheses). The parsing process of the interpreter for DUO was based on the method of finding a node in the query, regarding it as a query graph, and connecting the query graph to an element connected to it. As this method can not process the cyclic graph query, the parsing process is based on another method. The new method is that of seeking elements connected to edges and parentheses, collecting those edges and parentheses, and regarding them as a query graph.

Key words　Graphical language, Graph databases

1. はじめに

コンピュータの世界ではソフトウェア構築のために様々なプログラミング言語が開発され，ソフトウェアの発展に貢献してきた．従来のプログラミング言語は，様々な文字列を記述することによりプログラミングを行う．このような言語を文字列言語と呼ぶことにする．文字列言語は我々が普段用いる自然言語とは異なり，プログラミング特有の独特な文字列・記号の列挙であり，使用される文字列も英語やその短縮形であることが多い．また，プログラミング言語特有の独特の構文やアルゴリズムを修得しなければならない．そのためプログラミング初心者などにとっては，修得が困難であるという問題がある．また，文字列を線型に記述しているため，複雑な構造 (3 次元配列など次元数の高い配列，線型リストや木) を持つデータの扱いが難しく，読解も非常に困難になっている．

そこで，これらの問題点を解決するものとしてヴィジュアル

言語が提案されている．これは視覚的にプログラミングを行うことができるため，修得，読解が比較的容易であり，複雑なデータも容易に記述することができる．

このようなヴィジュアル言語のひとつとして，グラフィカル問い合わせ言語 DUO [1] が提案されている．DUO は，有向グラフとして表現されるデータに対して，グラフィカルな一種の正規表現を用いて問合せを行なうための言語である．

著者らは，グラフィカル問合せ言語 DUO の文法を完全にサポートし，十分な機能を持つインターフェースを備えた DUO インタプリタの実現を目指して検討を行ってきた [2]．試作中の DUO インタプリタはインターフェースと本体に大きく分けられる．本体は，記号解析部，構文解析部，問合せ変換部, 結果変換部から構成されている．構文解析部は多次元言語パーザ自動生成システム NAE [4] を用いて作成されている．実際のデータベース処理には，ICOT(新世代コンピュータ技術開発機構) で研究 · 開発された演繹オブジェクト指向データベースシステム Quixote [6] を使用している．しかし，これまで作成してきた DUO インタプリタでは，巡回を含む問合せができないという問題があった [2].

そこで本論文では，DUO インタプリタにおける巡回問合せの実現について述べる．採用する方法は問合せ中の枝と括弧のそれぞれに対して，つながる要素を求め，それらを集めて問合せグラフとするという方法である．これにより，巡回問合せを処理することが可能となった．

以下，2. ではグラフィカル問合せ言語 DUO について概説する．3. では多次元言語パーザ自動生成システム NAE について概説する．4. では DUO インタプリタのシステム構成について述べる．5. では DUO インタプリタにおける巡回問合せの処理について述べる．6. では DUO インタプリタにおける巡回問合せの実行例を示す.

2. グラフィカル問合せ言語 DUO

グラフィカル問い合わせ言語 DUO はラベル付き有向グラフで表されるデータに対してグラフィカルな正規表現を用いて問合せを行なうための言語である．DUO は以下に挙げるような特徴を備えている．

- 問合せグラフと問合せ表現

DUO は 2 段階の問合せの表現方法を持っている．簡単な問合わせを宣言的に行う問合せグラフと複雑な問合せを手続き的に行う問合せ表現である．問合せグラフは点, 枝, 括弧, スコープが連結された物である．点と枝はそれが答を求めたいものである場合には黒い線で，そうでないものはグレイの線で記述する．

- 括弧による正規表現

括弧を使うことにより再帰指定等がグラフィカルに表現できる．

- スコープによる否定の表現

矩形によりスコープの範囲を指定でき，これを利用して否定を表現できる．

- 矩形領域の重なりによる集合演算等の表現

集合演算や集合比較演算を矩形領域の重なりによりグラフィカルに表現できる．

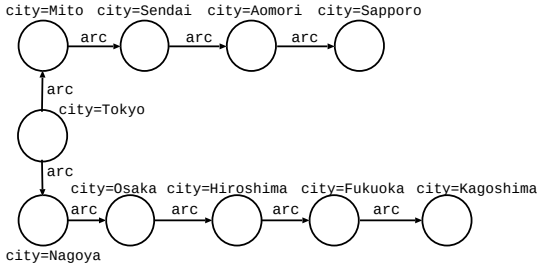

図 1　問合せの対象となるデータの例

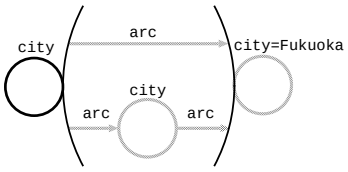

図 2　問合せグラフの例

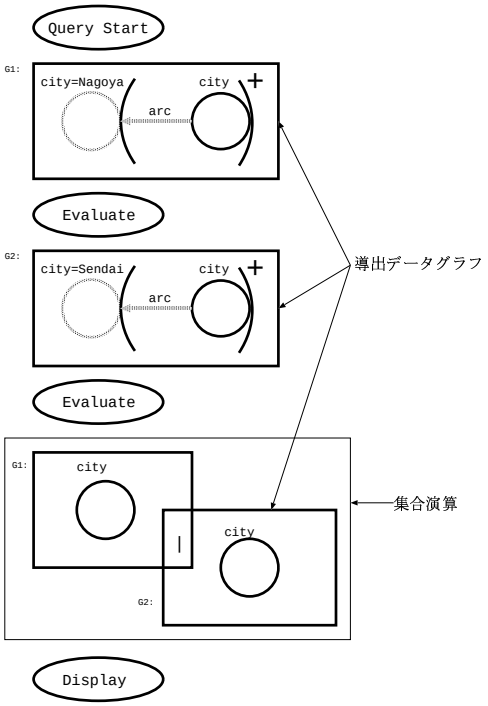

図 3　問合せ表現の例

図 2 は，図 1 に示したデータに対して，”Fukuoka” という値を持つ city に，パス arc, または，パス arc,city,arc でつながっ

ている city を求めるための問合せである．図の最左の点は黒線で描かれているので，この点が問合せ結果として利用者に返却される．

また，図 3 は，nagoya の上り方面，または，sendai の上り方面の city を求める” という問合せ表現である．ここでは，nagoya の上り方面を求め，次に sendai の上り方面を求め，最後にそれらの city の和を求めている．問合せ表現は複数の導出データグラフと 3 種類のラベルから構成される．3 種類のラベル “Query Start”，“Evaluate”，“Display” はそれぞれ，” 問合せを開始する”，“問合せグラフを評価する”，“結果を表示する” ことを示す．

3. 多次元言語パーザ自動生成システム NAE

3.1 概　　要

NAE(N-dimention syntax Analysis Environment genarator) とは，制約・関係マルチセット文法 (Constraint Relation Multiset Grammar : CRMG) を用いた多次元言語の定義から自動的に多次元言語のパーザを生成するシステムである [4].

制約・関係マルチセット文法 (CRMG) [4] とは，Constraint Multiset Grammar [5] に記号間の関係を導入した文法である．これにより，制約を簡潔に表現することができ，生成規則も簡潔なものとなる．Constraint Multiset Grammar [5] は，ヴィジュアル言語の構文解析を行うための文法であり，文脈自由文法の拡張である．記号の属性間の制約を用いて記号間の関係を表記するということを特徴としている．

3.2 システム構成

NAE は，多次元言語の定義を入力として受け取り，多次元言語パーザの C++ソースプログラムを自動的に生成するシステムである．NAE は，記号の型変換部，関係変換部，生成規則変換部，共通パーザの 4 つの要素から構成される．NAE のシステム構成を図 4 に示す．

- 記号の型変換部

記号の型の定義を C++のクラスへ変換する．

- 関係変換部

記号間の関係の定義を制約を満たすかどうか (真か偽か) を返す C++の関数へと変換する．

- 生成規則変換部

生成規則の定義を生成規則を処理するための C++の関数へ変換する．

- 共通パーザ

多次元言語を構文解析するための構文解析 (パーザ) 関数である．各変換部で生成されたクラス，関係関数，生成規則関数を用いて構文解析を行う．

3.3 多次元言語の定義

NAE における多次元言語の定義方法について述べる．

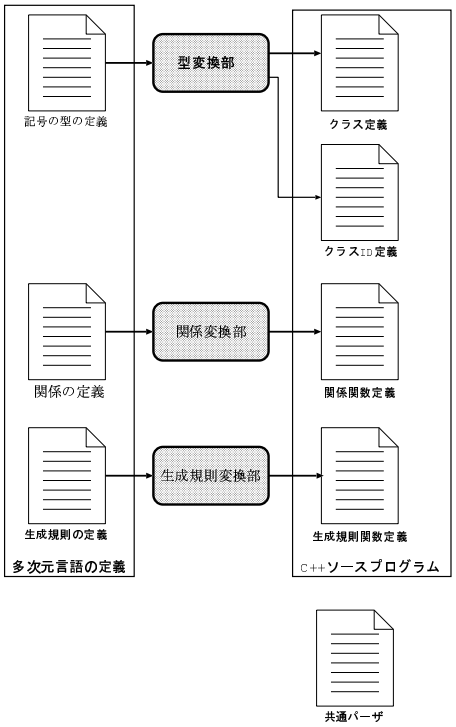

図 4　多次元言語パーザ自動生成システム NAE

3.3.1 記号の型の定義

型名とその型の属性を列挙することで記号の型を定義する．図 5 に示す Node 型,Edge 型を定義する例を示す．

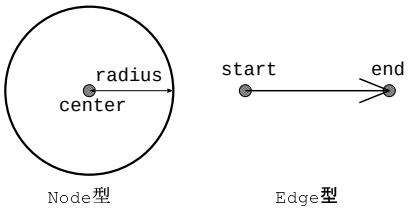

図 5　Circle 型と Arrow 型

```
terminal-type Node{
  int  center_x;
  int  center_y;
  int  radius;
  string label;
}
terminal-type Edge{
  int  start_x;
  int  start_y;
  int  end_x;
  int  end_y;
  string label;
}
```

Node 型は点を表す型である．その属性として，中心点の座標を表す int 型の center_x,center_y, 半径を表す int 型の radius,

ラベルを表す string 型の label を持っている.

Edge 型は枝を表す型である．その属性として，始点の座標を表す int 型の start_x,start_y，終点の座標を表す int 型の end_x,end_y, ラベルを表す string 型の label を持っている.

3.3.2 関係の定義

記号の属性を用いて記号間の関係を記述することで関係を定義する．すでに定義した Node 型,Edge 型を用いて図 6 のような関係 connect を定義する例を示す.

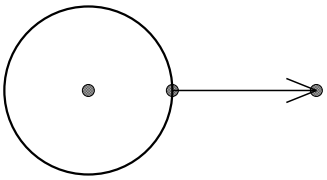

図 6 関係 connect

```
relation connect(Node SYM1, Edge SYM2)
{
  distance(SYM1.center_x, SYM1.center_y,
           SYM2.start_x, SYM2.start_y)
    >= (SYM1.radius - 1.0);
  distance(SYM1.center_x, SYM1.center_y,
           SYM2.start_x, SYM2.start_y)
    <= (SYM1.radius + 1.0);
}
```

関係 connect は Node 型の記号が Edge 型の記号の始点のつながっているという関係を表す．点 (x1,y1) と点 (x2,y2) の距離を求める distance(x1,y1,x2,y2) という関数が既に定義されているものとする.

3.3.3 生成規則の定義

すでに定義した Node 型,Edge 型関係 connect を用いて，Node 型の記号が Arc 型の記号に還元されることを意味する生成規則を定義する例を示す.

```
terminal-type Arc{
  string label;
  string start_label;
  string end_label;
}
```

Arc 型が上記のように定義されているとする．Arc 型は自分自身のラベルを表す string 型の label, 自分の始点につながる Node のラベルを表す string 型の start_label と自分の終点につながる Node のラベルを表す string 型の end_label を属性として持つ記号の型である.

```
SYM:Arc ::= SYM1:Arrow exist SYM2:Node SYM3:Node
{
  where
    connect(SYM2, SYM1) && connect(SYM3, SYM1);
  and
    SYM.label = SYM1.label;
    SYM.start_label = SYM2.label;
    SYM.end_label = SYM3.label;
}
```

SYM は Arc 型，SYM1 は Edge 型，SYM2 は Node 型，SYM3 は Node 型の記号である．exist 以降に記述された SYM2,SYM3 はこの記号が存在することが生成規則を適用するための制約となることを表している．where 以降には SYM2 が SYM1 の始点につながり，かつ，SYM3 が SYM1 の終点につながるという制約が記述されている．and 以降には属性に対する代入が列挙されている.

4. DUO インタプリタのシステム構成

DUO インタプリタのシステム構成について述べる.

DUO インタプリタのシステム構成を図 7 に示す．DUO インタプリタはインターフェースと本体の 2 つに大きく分けられる．インターフェースは Java で作成されている．本体は，記号解析部, 構文解析部, 問合せ変換部の 3 つの部分から構成される．構文解析部は NAE によって製作されている．NAE を用いることによって構文解析部の作成や修正が容易になる．また，実際のデータベース処理には演繹オブジェクト指向データベースシステム Quixote [6] を使用している.

- インターフェース

問合せの入力情報を文字列として本体に送る．問合せの入力情報には，点, 枝, 括弧等の要素の座標や問合せの条件などが含まれる．問合せの結果を受け取って表示する.

- 本体

記号解析部，構文解析部，問合せ変換部，ならびに，結果変換部で構成されている.

- 記号解析部

インターフェースから送られた文字列を記号の集合という形に変換し構文解析部へ送る.

- 構文解析部

記号解析部から送られた記号の集合に対して構文解析を行い，DUO のセマンティックスに沿った表現である DUO 表現へと変換を行う．構文解析部は多次元言語パーザ自動生成システム NAE を用いて作成されている.

- 問合せ変換部

問合せ変換部では，構文解析部から送られた DUO 表現を Quixote の問合せ表現へと変換し，パイプを用いて Quixote へ実際に問合せを行う.

- 結果変換部

Quixote への問合せの結果をインターフェースが受け取る結果として適切な結果に変換して，インターフェースに渡す．

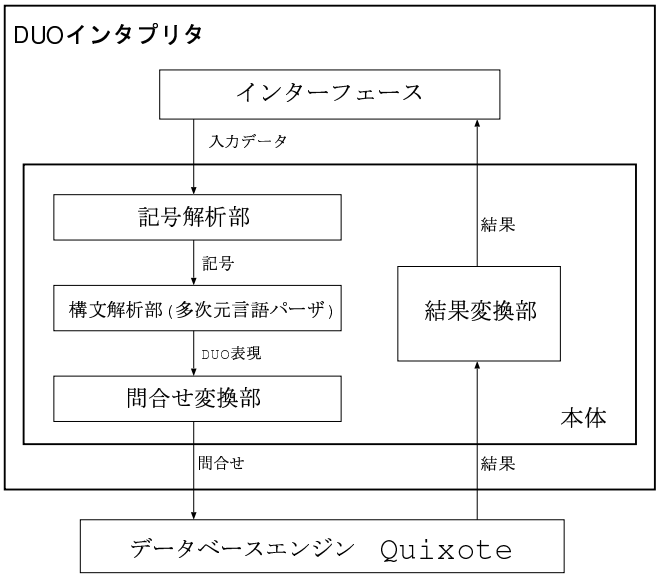

図 7　DUO インタプリタのシステム構成

5. 巡回問合せの処理

DUO インタプリタでは，多次元言語自動パーザ生成システム NAE に DUO 言語の定義を与えることによって構文解析部を作成している．巡回問合せを含む処理を可能にするために NAE に与える DUO 言語の定義を変更した．

5.1 従来の方法

従来の方法では，問合せの中から 1 つ点を見つけ，それを問合せグラフとし，問合せグラフの両端の要素のどちらかにつながる要素があればそれをつなげて問合せグラフとするという考え方に基づいて DUO 言語の定義を記述していた．DUO 言語の生成規則定義を簡略化したものを図 8 に示す．例えば，`se:SourceElms ::= fe:FullElms ae:ArcElms where connect(fe, ae)` という記述は，fe は FullElms 型の記号,ae は ArcElms 型の記号であり，fe と ae が connect という関係を満たすとき SourceElms 型の記号 se に還元されるということを表す．この生成規則定義は，点と枝のみを含み，括弧を含まない問合せを表す言語の定義である．

図 8 の定義に基づいて図 9 に示す問合せの構文解析を行った場合の過程を図 10 に示す．まず，点 a を表す Node, 点 c を表す Node, 枝 b を表す Edge を含む記号集合が構文解析部に渡される．次に，Node は FullElms に，Edge は ArcElms にそれぞれ還元される．次に，点 a と枝 b が connect 関係を満たすので点 a を含む FullElms と枝 b を含む ArcElms が SourceElms に還元される．そして，枝 b と点 c が connect 関係を満たすので点 a, 枝 b を含む SourceElms と点 c を含む FullElms が FullElms に還元される．最後に，FullElms が QueryGraph に還元されて，構文解析が終了する．

この方法は問合せを一つのパスで表現するという方法である．この方法では，複数のパスを含む問合せを処理することができない．

- Node 点を表す記号
- Edge 枝を表す記号
- SourceElms 終点が点ではないパスを表す記号
- FullElms 始点, 終点が共に点であるパスを表す記号
- QueryGraph 問合せグラフを表す記号

```
fe:FullElms ::= n:Node where true
ae:ArcElms ::= e:Edge where true
se:SourceElms ::= fe:FullElms ae:ArcElms
                  where connect(fe, ae)
fe:FullElms ::= se:SourceElms fe:FullElms
                  where connect(se, fe)
qg:QueryGraph ::= fe:FullElms where true
```

図 8　従来の生成規則定義

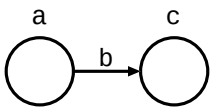

図 9　問合せの例

そのため，従来の方法では点を記号の情報に変換する際に，点に入ってくる枝の数と点から出ていく枝の数を調べ，多い方の数の分だけその点の情報を作成するようにしていた [2]．そうすると構文解析の結果として複数の問合せグラフが得られるのでこれを合成して Quixote 問合せへ変換することができる．図 11 の問合せの場合図 12 のように解析される．

しかし，この方法では巡回を含む問合せの処理を行うことができない．

5.2 枝と括弧に基づく方法

巡回問合せを処理するために，問合せ中の枝と括弧のそれぞれに対して，つながる要素を求め，それらを集めて問合せグラフとするという考え方に基づいて処理する．この考え方に基づく DUO 言語の生成規則定義を簡略化したものを図 13 に示す．`a:Arc ::= e:Edge exist n1:Node n2:Node` の exist は，exist 以降に記述された記号が制約の判定には使用されるが還元されないということを表している．この生成規則定義は，点と枝のみを含み，括弧を含まない問合せを表す言語の定義である．

図 13 の定義に基づいて図 9 に示す問合せの構文解析を行っ

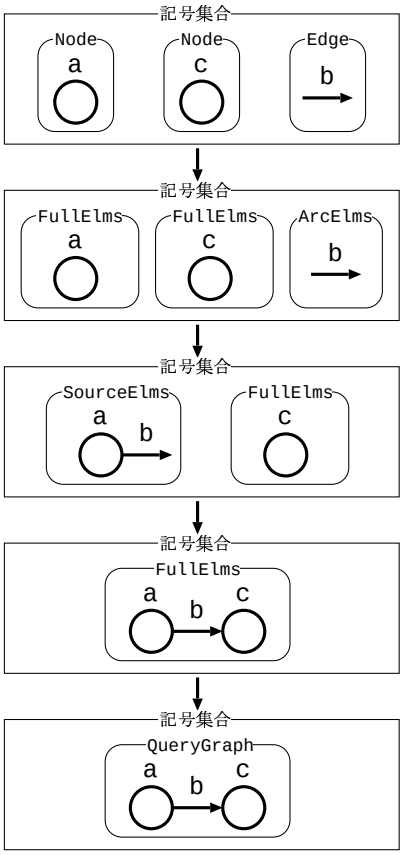

図 10　従来の生成規則定義による図 9 の問合せの構文解析過程

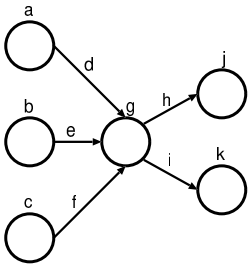

図 11　複数のパスを含む問合せ

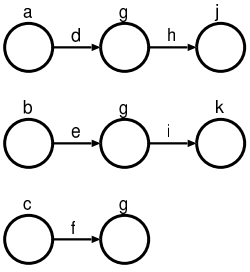

図 12　図 11 の解析結果

- Node 点を表す記号
- Edge 枝を表す記号
- Arc 始点と終点につながる点の情報を持つ枝を表す記号
- NodeSet Node の集合を表す記号
- ArcSet Arc の集合を表す記号
- QueryGraph 問合せグラフを表す記号

```
a:Arc ::= e:Edge exist n1:Node n2:Node
          where connect(n1, e) && connect(e, n2)
as1:ArcSet ::= as2:ArcSet a:Arc where true
as:ArcSet ::= a:Arc where true
ns1:NodeSet ::= ns2:NodeSet n:Node where true
ns:NodeSet ::= n:Node where true
qg:QueryGraph ::= as:ArcSet ns:NodeSet where true
```

図 13　枝と括弧に基づく生成規則定義

た場合の過程を図 14 に示す．まず，点 a を表す Node, 点 c を表す Node, 枝 b を表す Edge を含む記号集合が構文解析部に渡される．次に，点 a と枝 b が connect 関係を満たし，かつ，枝 b と点 c が connect 関係を満たすので，枝 b を表す Edge が Arc に還元される．この Arc は枝 b が点 a, 点 c とつながっているという情報を保持する．次に，全ての Node が集められて NodeSet に還元され，全ての Arc が集められて ArcSet に還元される．そして，NodeSet と ArcSet が QueryGraph に還元されて構文解析が終了する．

図 15 に示す問合せは点 c, 枝 d, 点 e, 枝 f, 点 c という巡回を含んでいる．図 13 の定義に基づいて図 15 に示す問合せの構文解析を行った場合の過程を図 16 に示す．先に挙げた例と同様に構文解析が進んで行き，最終的に QueryGraph に還元されて構文解析が終了する．枝につながる点を求めるだけなので，問合せの中に巡回を含む場合でも構文解析を行うことが可能である．

Quixote 問合せへの変換は構文解析の結果として得られる QueryGraph を変換することによって行う．QueryGraph から Quixote 問合せへ変換するには，QueryGraph の含む点, 枝, 括弧の一つ一つを Quixote 問合せに変換したものを組み合わせればよい．枝, 括弧を変換する場合，枝, 括弧につながる要素の情報があればよいので，問合せの中のパスを求める必要はない．

6. 問 合 せ 例

巡回問合せの例として，図 17 に示すグラフ構造に対する問合せ例を示す．このデータに対して図 18 に示す問合せを行う．この問合せは「途中で”a” という値を持つ city を通って自分自身に戻ってくるパスを持つ点を求める」という問合せである．検索結果を図 19 に示す．検索結果として，”a” という値を持つ city, “b” という値を持つ city, “c” という値を持つ city が求まっており，正しい検索結果が得られている．

7. お わ り に

本論文では，ラベル付き有向グラフで表現されたデータベー

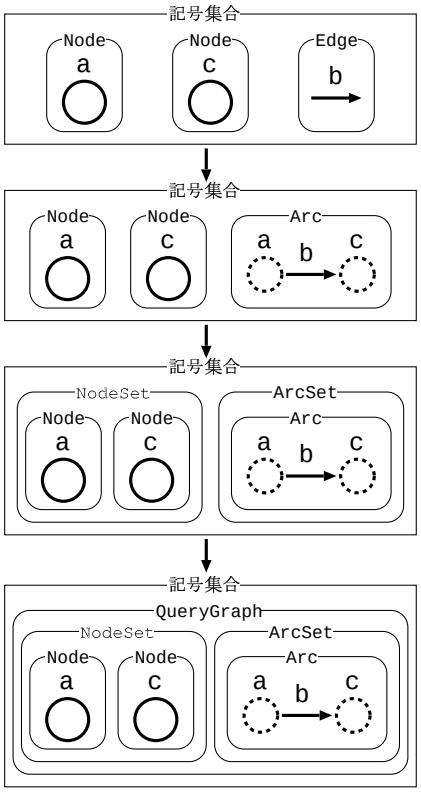

図 14　枝と括弧に基づく生成規則定義による図 9 の問合せの構文解析過程

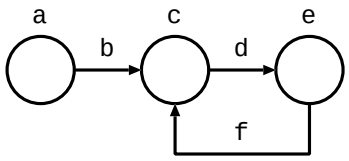

図 15　巡回を含む問合せの例

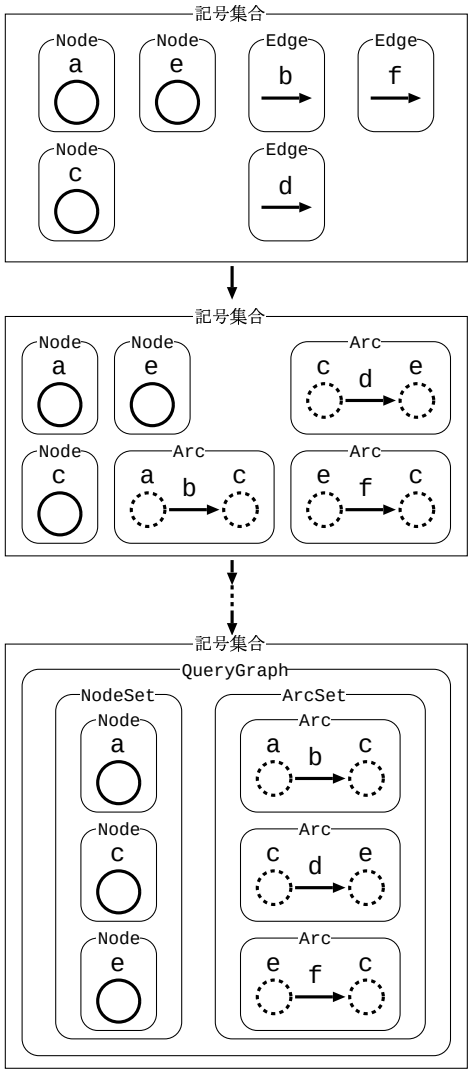

図 16　枝と括弧に基づく生成規則定義による図 15 の問合せの構文解析過程

スに対する問合せ言語 DUO のインタプリタにおける巡回グラフ問合せの実現について述べた．問合せの構文解析の方法を問合せ中の枝と括弧のそれぞれに対して，つながる要素を求め，それらを集めて問合せグラフとするという方法を採用することによって巡回グラフ問合せの処理が可能となった．

現在のインタプリタでは，DUO 文法を完全にはサポートしていない．具体的には，否定を表すスコープ, 問合せ結果の和集合や差集合を求めるための問合せ表現などをサポートしていない．これらの問題を解決し DUO 文法の完全なサポートを行なうことは今後の課題である．インターフェースの問合せ結果表示では，結果の構造によっては適切な表示がされないことがある．この問合せ結果表示の改良も課題である．

文　　献

[1] 宝珍輝尚: グラフィカル問い合わせ言語 DUO の問い合わせ能力, 情処論 *Vol36, No.4, pp.959-970 (1995).*

[2] 宮崎 則雄，宝珍輝尚，都司達夫，樋口 健: グラフに基づくデータベースに対するグラフィカル問合せ言語の実現, 第 *13* 回データ工学ワークショップ論文集, *C2-6 (2002).*

[3] 畠山竜輔，宝珍輝尚，都司達夫: 多次元言語パーザ自動生成システムを用いたグラフィカル問合せ言語インタプリタの実現, 情処研報 *DBS 115-14, pp.103-110 (1998).*

[4] 石井義知，宝珍輝尚，都司達夫: 多次元言語の構文解析に関する一手法, 信学技法 *SS96-64, pp.25-32 (1997).*

[5] Kim Marriot: Constraint Multiset Grammars, *1994 IEEE Symposium on Visual Languages, pp.118-125 (1994).*

[6] 財団法人 新世代コンピュータ技術開発機構: Quixote 言語マニュアル *(1995).*

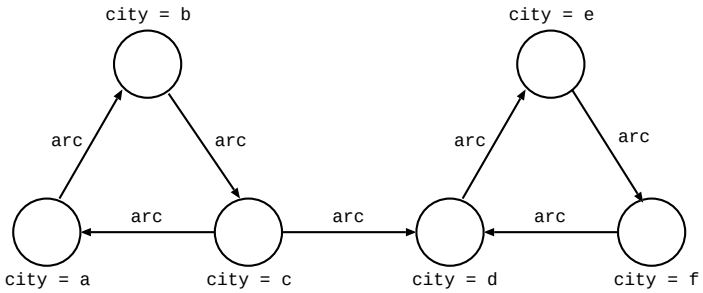

図 17　問合せの対象となるデータ

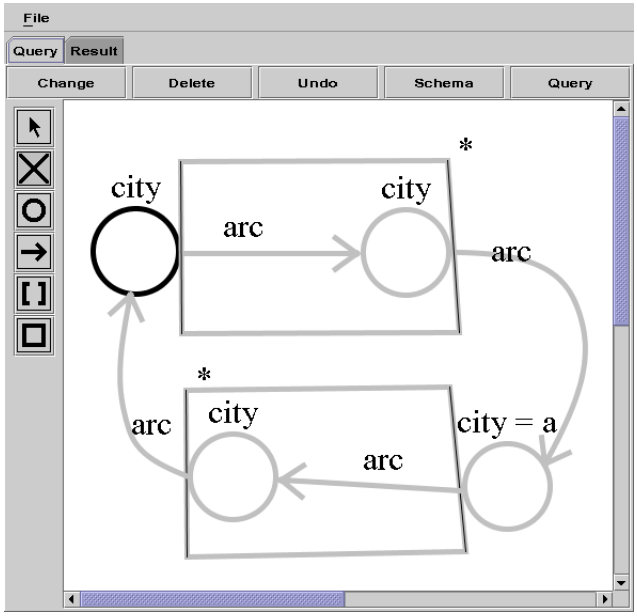

図 18　問 合 せ 例

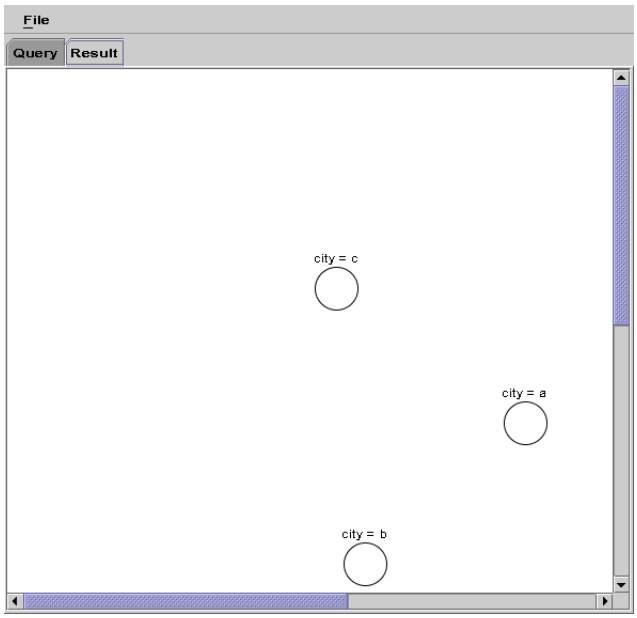

図 19　検 索 結 果