グラフィカル問合せ言語 DUO の問合せ能力

宝珍 輝尚†

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グラフィカルな問合せ言語 DUO の問合せ能力について述べる．DUO の問合せの基本は問合せグラフと呼ぶ一種のグラフである．問合せグラフでは，要素ラベル列集合上での正規表現を括弧を用いてグラフィカルに記述でき，簡単な問合せを宣言的に記述できる．また，否定，補集合，集合比較演算，集合演算をグラフィカルに記述できる．さらに，問合せ結果を導出データグラフとすることで複雑な問合せを副問合せに分解して記述できる．本論文では，DUO が，高い表現能力を持つグラフィカル問合せ言語 GraphLog と少なくとも同等の表現能力を持ち，問合せの複雑さに関して同様な問合せ記述特性を持つことを示す．

Querying Power of a Graphical Query Language : DUO

TERUHISA HOCHIIN †

This paper describes that the expressive power of DUO is more or equal to that of another graphical query language GraphLog, and that the querying characteristics of DUO is similar to that of GraphLog. A *query graph* is a fundamental unit in querying on a database, which is a collection of labeled directed graphs. The regular expressions on sets of element label strings can be graphically described in it by using braces. These expressions make the query declarative. Negation, complement, set comparisons, and set operations are also graphically specified. A query result can be a derived data graph.This enables users to make a complex query easily by decomposing it into subqueries.

1. はじめに

グラフィカルな問合せ言語の研究が盛んに行われている[1)~10)]．グラフで表現されるデータを扱うデータベース (DB) 応用分野に対しては，パスに関する検索条件が記述できる問合せ言語[1)~7)]が有効である．トロント大学で開発された GraphLog[1)] や G$^+$[3)] では，枝のラベルの正規表現を用いて問合せが可能である．また，アントワープ大で開発されている GOOD[5)] でも，枝のラベルの正規表現を記述できるマクロが開発されている[6)]．このような枝のラベルの正規表現は，枝のつながり方のパターンを指定した検索を可能にするので，パスに関する問合せにとって非常に有用である．しかし，正規表現を文字列で記述するため，この記述方法が真にグラフィカルであるとは言い難い．

筆者が提案中のグラフィカルな問合せ言語 DUO[7)] では，括弧を用いることでグラフィカルに正規表現が記述できる．また，GraphLog, G$^+$ や GOOD と同様に，複数の副問合せを用いて問合せが記述でき，複雑な問合せを容易に行える．さらに，パスに関する集合演算や集合比較演算がグラフィカルに記述できる．このような特徴を持つ DUO であるが，DUO の問合せ能力がどの程度か明確ではない．

そこで，本論文では，DUO の問合せ能力の明確化を目的とし，高い表現能力を持つ GraphLog を比較対象として，表現能力と問合せ記述特性を評価する．この結果，DUO が GraphLog と少なくとも同等の表現能力を持ち，問合せの複雑さに関して同様な問合せ記述特性を持つことを明らかにする．

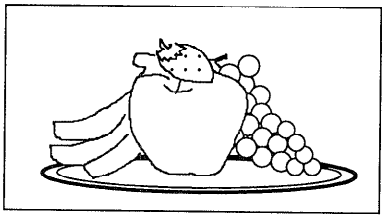
以下，2 章でグラフに基づく DB に対する問合せについて述べ，3 章で DUO を概説する．そして，4 章で DUO の表現能力と問合せ記述特性を GraphLog と比較して評価する．

2. グラフに基づく DB に対する問合せ

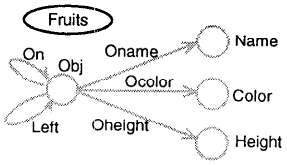
2.1 グラフに基づくデータモデル

ここでは，図 1(a) に示すような静物写真集の各写真の内容の表現を考える．静物には，例えば，果物や器の名前や色，および，物体間の相対的な位置関係といった一般的な特徴があり，これらはスキーマとして

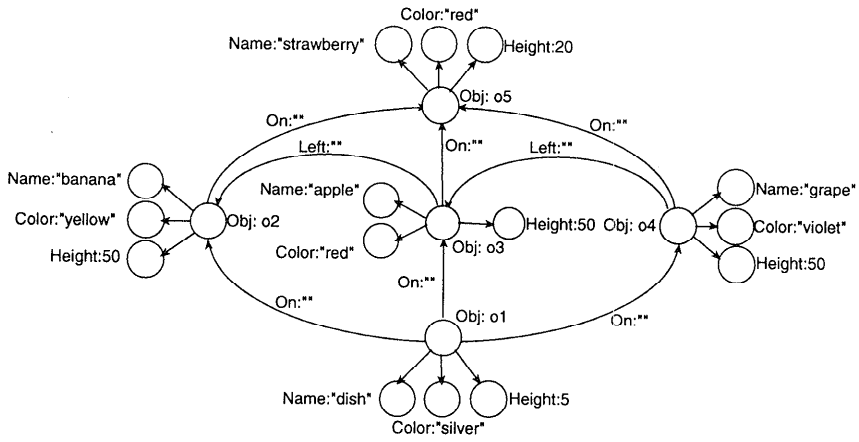
† 福井大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Fukui University



(a) 表現対象



(b) スキーマグラフ



(c) データ表現グラフ

図1 表現対象，スキーマグラフとデータ表現グラフの例
Fig. 1 Examples of (a) target, (b) schema graph and (c) data representation graph.

あらかじめ定義できる（図1(b)）．そして，物体の名前が「りんご」で，色は「赤」といった属性値や，りんごは皿の上にあるといった実体間の関連を設定できる（図1(c)）．

ここでは，図1(c)に示すような，データを表現するラベル付き有向グラフをデータ表現グラフと呼ぶ．また，データ表現グラフを成分とするグラフをデータグラフと呼ぶ．データグラフの集合がDBである．形式的には，データ表現グラフは，点集合Vと枝集合Eからなる順序組(V, E)で表現される，連結または非連結なグラフg(V, E)である．枝は向きを持ち，始点と終点を持つ．点および枝にはラベルが付く．ラベルは3つ組(d_{ID}, N, d)で表す．ここで，d_{ID}はグラフ要素をDB内で一意に識別する一意識別子，Nは同種のグラフ要素集合をデータグラフ内で一意に識別する要素名，dはデータである．データdは組（データ型，値）で表現する．以降（図を含む）では，簡単のため一意識別子d_{ID}を省略し「N：d」で要素を表現する．また，簡単のため，「ラベル中のデータの値がxである点や枝」を，「値がxの点や枝」と記述する．DB内でデータグラフを識別するためにデータグラフに付ける名前をデータグラフ名という．同一データグラフ名のデータグラフの要素名のみをラベルとした（同一要素名が複数ある場合は1つとした）グラフを，そのデータグラフのスキーマグラフという．

図1(b)はスキーマグラフである．楕円で囲った“Fruits”はデータグラフ名である．図1(c)はデータ表現グラフである．図1(c)では，枝Onや枝Leftには値がない．また，簡単のため，枝Oname，枝Ocolor，枝Oheightのラベルを省略している．

2.2 問合せの分類

一般にDBに対する問合せは問合せの難しさという観点から，単なる比較演算を使用するもの，集約演算を使用するもの，集合演算や集合比較演算を使用するものに分類される[11),12)]．さらに，グラフに基づくDBに対する問合せは，問合せの対象という観点から，値や実体に関するものとパスに関するものにも分類できる．これらの組み合わせで問合せを分類したのが**表1**である．表1において，分類番号1と2は単なる比較演算を使用するもの，分類番号3と4は集約演算を使用するもの，分類番号5と6は集合演算や集合比較演算を使用するものである．ここで，例えば，分類番号3の問合せとは，少なくとも「値集合の集約に関する検索条件」を含み，かつ，分類番号4以上の検索条件を

表 1 問合せの分類と問合せ例
Table 1 Classification and examples of queries.

分類番号	検索条件	問合せ例
1	属性値に関する検索条件	赤いものを求める.
2	パスに関する検索条件	Obj：o1 の上でも左でもない赤いものを求める.
3	値集合の集約に関する検索条件	Obj：o1 上に 3 個以上ある赤いものを求める.
4	パスに沿った値に関する検索条件	Obj：o1 上にある最遠の赤いものを求める.
5	値集合に対する集合演算や集合比較演算による検索条件	Obj：o1 上にある最遠の赤いもの, ならびに, Obj：o2 上のものを求める.
6	パス集合に対する集合演算や集合比較演算による検索	Obj：o1 上にある最遠の赤いものとパスを共有する Obj：o2 上のものの終点を求める.

含まない問合せを意味する.

3. グラフィカルな問合せ言語：DUO

2 章で述べた問合せを簡単に行うことを目的として, 筆者は DUO (graph-based *D*atabase graphical q*U*ery expressi*O*n) と名付けたグラフィカルな問合せ言語を提案している[7]. まず, DUO の概要を述べ, 次に, DUO の問合せに対する処理概要を示す.

3.1 DUO の概要

DUO ではスキーマを利用して問合せを記述する. DUO の問合せの基本は問合せグラフである. 問合せグラフは 1 以上の問合せパスの連接で表現される. ここで, 通常の, パスの始終要素がともに点のパスを完全パスと呼び, パスの始要素のみ (終要素のみ, 始終要素) が枝のパスを始 (終, 始終) 不完全パスと呼ぶ. 完全パスと 3 種の不完全パスを総称してパスと呼ぶ. 問合せに利用できるように, 要素に検索条件の指定が可能なラベルを付け, このような要素からなるパスを問合せパスという. このラベルは 4 つ組 (is, es, rc, V) で表される. ここで, is は一意識別子に関する条件で, {[¬]< 一意識別子 >|*all*} と指定できる. es は要素名に関する条件で, {[¬]< 要素名 >|*all*} と指定できる. V は変数の集合である. rc は検索条件であり, 通常のように, 関係演算子, 算術演算子, 論理演算子や後述の集約演算子が記述できる. 検索条件は文字列で指定する.

問合せパスでは, 検索対象の点や枝のラベル中のデータの値に関する検索条件に加えて, 検索するパスのパターンを一種の正規表現により記述できる. パスの和は, 図 2(a) のように, 括弧内に複数のパスを並記することで表現する. 図の「( )」内では, 枝 On または

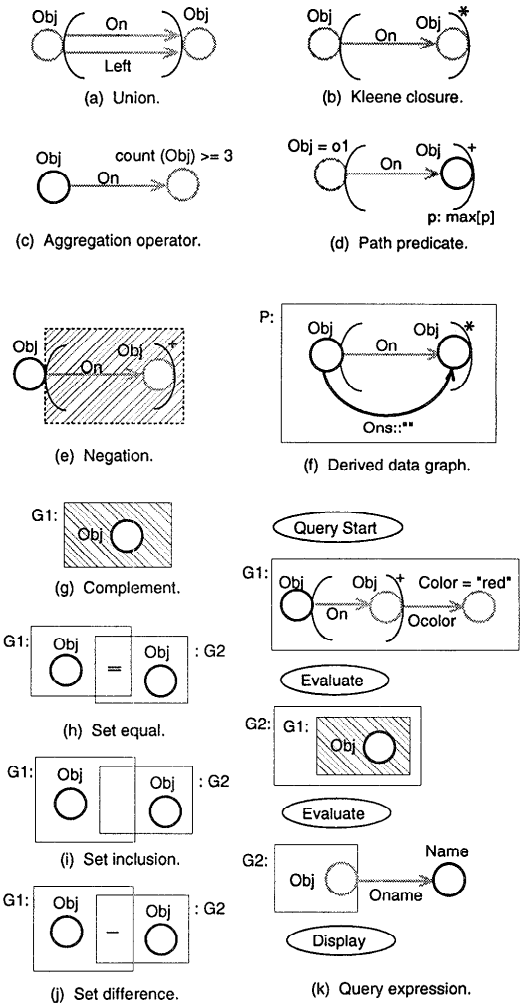


図 2 DUO の基本演算
Fig. 2 Primitives in DUO.

枝 Left を表している. また, Kleene (正) 閉包パスを括弧に *(+) を付加することで表現する (図 2(b)). ここで, Kleene (正) 閉包パスとは, ある始不完全パスまたは終不完全パス p の 0(1) 回以上の繰り返しを含むパスである. 図 2(b) の「( )*」は, 0 個以上の枝 On からなるパスを表す Kleene 閉包パスである. 括弧には, 組 (prc, V) で表されるラベルが付く. ここで, V は変数の集合であり, prc はパスに関する検索条件で, 前述の rc で記述可能な検索条件に加えて後述のパス述語が記述できる.

要素やパスに関する集約演算子 (*count*, *product*, *sum*, *max*, *min*, *ave*) ならびに否定も記述可能である. 集約演算子を用いた検索条件は, 関係演算子と同様に文字列で指定する. 例えば, 枝 On でつながる点 Obj の数が 3 以上であるという検索条件は, 図 2(c)

のように記述する．また，最大値や最小値を持つパスを求めるパス述語(*max*, *min*)が記述できる．例えば，値がｏ1の点Objから枝Onでつながり，その枝の数が最大となる点Objを求めるという問合せは図2(ｄ)のように記述する．否定は，図2(ｅ)のように否定範囲を点線で囲みハッチングを施すことで表す．図2(ｅ)では，枝Onで点Objにつながっていない点Objを表している．

問合せパスで指定した検索条件ならびに問合せパスが表現するパターンに合致するデータ表現グラフ集合が利用者の指定した形で返却される．返却要素は黒の太線で指定する(図2(ｃ)～(ｋ))．返却しないものはグレーのままである．

さらに，問合せの結果得られたデータ表現グラフを仮想的に定義されたデータグラフにまとめることができる．これを導出データグラフと呼ぶ．これにより問合せ結果を利用した問合せが可能である．導出データグラフは名前付きの四角形で表す．図2(ｆ)は導出データグラフの例で，検索結果の点Objを枝Onsでつないだデータ表現グラフからなるデータグラフを表す．導出データグラフにもスキーマグラフがある．導出データグラフの問合せ結果に対応するスキーマグラフが導出データグラフのスキーマグラフである．図2(ｆ)の例では，枝Onsとその始終点からなるグラフが導出データグラフPのスキーマグラフである．補集合は，導出データグラフにハッチングを施すことで表現する(図2(ｇ))．さらに，2つの導出データグラフの画面上での位置関係を利用して，集合比較演算や集合演算をグラフィカルに記述できる．図2(ｈ)では，導出データグラフG1の点Objの集まりが導出データグラフG2の点Objの集まりと集合的に等しいという条件を表している．図2(ｉ)では，G1の点Objの集まりがG2の点Objの集まりを包含するという条件を表している．集合演算も同様に表現できる．G1とG2の差は図2(ｊ)のように表現する．和や積は，図中の“－”を“|”や“&”とすることで表現する．

最後に，複数の導出データグラフを利用した問合せの例を図2(k)に示す．図2(ｋ)は，上に赤いObjがないObjの名前を求める問合せである．図2(ｋ)では，赤いものが上にある点Objの集合をG1とし，その補集合をG2としている．そして，G2の点Objをもとに名前を求めている．図2(ｋ)のQuery Start, Evaluate, Displayは，おのおの，「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．これらは，ディスプレイ上のボタンとなっている．

以上述べたDUOの構文を付録に示す．

3.2 DUOの問合せ処理

要素列，括弧，四角形といったDUOの問合せを表現する個々の要素（表現要素）が処理を表現しているととらえると，表現要素の組み合わせで問合せが表現できる．ここでは，個々の表現要素に対して，処理の対象，処理の内容を明らかにし，DUOの問合せのセマンティックスを明確化する．

（1） 要素（点と枝）

要素は6つ組(et, is, es, rc, V, r)で表現される．ここで，etは要素の種別(点か枝か)を表す．is, es, rc, Vは，前述のとおり，おのおの，一意識別子に関する条件，要素名に関する条件，検索条件，変数の集合である．変数は問合せグラフ内で可視である．rは，返却結果を表す，変数や集約演算を含む算術式である．rの指定がない要素は返却されない．

要素では，処理対象の個々の要素が以下に述べる要素全体での検索条件$cond_{elm}$を満足するか検査し，満足する要素の集合に対して集約関数のみの検索条件rc_{agg}を満足するか検査し，満足する要素の集合を処理結果とする．ここで，rcは，集約関数のみの検索条件rc_{agg}と集約関数を含まない検索条件rc_{ord}を用いて，$rc = rc_{ord} \wedge rc_{agg}$と表現できるとする．

処理対象は次のとおりである．

（ｉ） es＝pでpがDBに存在する点または枝のとき，pなる点または枝．(pがDBに存在しない場合は，後述の「問合せグラフ」で処理される．)

（ii） es＝￣pのとき，DBに存在する$p_i \neq p$なるすべての点p_iまたは枝p_i．

（iii） es＝allのとき，DBに存在するすべての点または枝．

要素全体での検索条件$cond_{elm}$は以下のとおりである．

（ｉ） V＝{ }のとき，適当な変数Xを割り当てて，$cond_{is}(X)\&rc_{ord}$．

（ii） V＝{X1, …, Xn}のとき，$cond_{is}(X1)\&rc_{ord}(X1, \cdots, Xn)$．

ここで，$cond_{is}$はisに対応する検索条件で，要素に変数Xが割り当てられているとき，以下のとおりである．

（ｉ） is＝olのとき，oid(X)＝ol．

（ii） is＝￣olのとき，oid(X)≠ol．

（iii） is＝allのとき，常に真．

ただし，oid( )は要素の一意識別子を得る関数．

（2） 要 素 列

要素列は要素s_iの列である．要素列では，以下に述

べる要素間の連接条件 $cond_{path}$ を検査し，満足した要素列の集合を処理結果とする．要素列が枝 $\{e_1, \cdots, e_n\}$ とその始終点から構成されている場合，要素列の連接条件 $cond_{path}$ は $cond_{e_1} \wedge \cdots \wedge cond_{e_n}$ で表される．ここで，$cond_{e_i}$ は枝の連結条件である．枝の連結条件は，2つの点 n_{i1}, n_{i2} と枝 e_i に対して，枝 e_i の始終点が，おのおの，n_{i1} と n_{i2} であるときに真となる．要素列全体の検索条件は，$cond_{path}$ と要素列を構成する個々の要素の検索条件の論理積で表される．

（3） 括　　弧

括弧は5つ組(E, cs, V, VA, prc)で表現される．ここで，Eは，画面において括弧に含まれる要素列を s_{i0} とすると，以下のような要素列の集合である．

（i） s_{i0} が完全パスのとき，s_{i0} そのもの．

（ii） s_{i0} が始不完全パスのとき，開き括弧に隣接する点を s_{i0} の最初に連接させた要素列．

（iii） s_{i0} が終不完全パスのとき，閉じ括弧に隣接する点を s_{i0} の最後に連接させた要素列．

（iv） s_{i0} が始終不完全パスのとき，開き括弧に隣接する点を s_{i0} の最初に連接させ，閉じ括弧に隣接する点を s_{i0} の最後に連接させた要素列．

また，csは，＊付き，＋付きか，＊や＋なしの指定，prcはパスに関する条件を表現する．Vは通常の変数の集合で，VAは値割り付けを持つ変数の集合である．変数は問合せグラフ内で可視である．

括弧では2段階で処理が行われる．第1段階では，必要ならば閉包をとり，第2段階では，パス述語を処理する．

(A) 第1段階

処理対象は，Eの要素列の集合である．これに対して，括弧の表す条件 $cond_{brace}$ を満足するか検査する．$cond_{brace}$ は，$E = \{s_1, \cdots, s_k\}$ の場合，おのおのの要素列 s_i 全体の表す検索条件を $cond_{s_i}$ とすると，$cond_{s_1} \vee \cdots \vee cond_{s_k}$ である．第1段階の結果は以下のとおりである．

（i） cs＝なし　のとき，P^1．

（ii） cs＝＊　のとき，$\bigcup_{i=0}^{\infty} P^i$．

（iii） cs＝＋　のとき，$\bigcup_{i=1}^{\infty} P^i$．

ここで，$P^0 = \{\ \}$，P^1 は $cond_{brace}$ を満足する要素列の集合，$P^i (i \geq 2)$ は，P^{i-1} 中の要素列の終点を始点として $cond_{brace}$ を満足する要素列を求め，それをもとの要素列に連接してできる要素列を P^{i-1} に加えてできる要素列集合．

(B) 第2段階

第1段階の結果である要素列がパスに関する条件prcを満足するか検査し，満足したならば，その要素列を結果とする．

（4） スコープ

点線の四角形で表現されるスコープは組 (p, ss) で表される．ここで，pは要素，要素列または括弧であり，ssはnotか否かの指定である．

ss＝notのときスコープは以下のように作用する．

（i） pが要素または要素列のとき，要素または要素列全体の表す検索条件 cond を¬cond とするように作用する．

（ii） pが括弧のとき，括弧の表す条件 $cond_{brace}$ を¬$cond_{brace}$，パスに関する条件prcを¬prcとするように作用する．

ここで，問合せグラフ全体を表す表現要素である「問合せグラフ」について述べる．

（5） 問合せグラフ

問合せグラフは組 (GP, GR) で表される．ここで，GPは，要素，要素列，括弧，スコープ，または，後述する導出データグラフ，集合比較演算，集合演算の集合であり，GRは問合せ結果となる要素または要素列の集合である．

問合せグラフでは，GPにより得られる要素や要素列をGRの要素や要素列で構成されるグラフに変換する．まず，DBに存在しないGRの要素または要素列中の要素に対して，その要素が格納されうる枠組みを一時的に定義する．その後，GRの要素または要素列中の要素のrに従って要素や要素の集合を加工し，要素列ならば要素列を構成してこれを結果とする．

（6） 導出データグラフ

実線の四角形で表現される導出データグラフは3つ組 (gn, qg, gs) で表される．ここで，gnは導出データグラフの名前，qgは問合せグラフ，gsは補集合をとるか否かの指定である．

qgを処理し，gs ≠ complement のとき，qgの結果を導出データグラフの結果とする．gs ＝ complement のときは，qgの結果以外の要素列を結果とする．

（7） 集合比較演算

集合比較演算は3つ組 (ct, g1, g2) で表される．ここで，ctは演算の種別でequal, includeであり，g1, g2は，導出データグラフまたはスコープである．

ct(g1, g2)が真ならば，g1とg2の結果を集合比較演算の結果とする．

（8） 集合演算

集合比較演算は3つ組 (st, g1, g2) により表される．ここで，stは演算の種別でunion, intersection,

＊ ラベルが n 項組で表されるラベル付き有向グラフである．

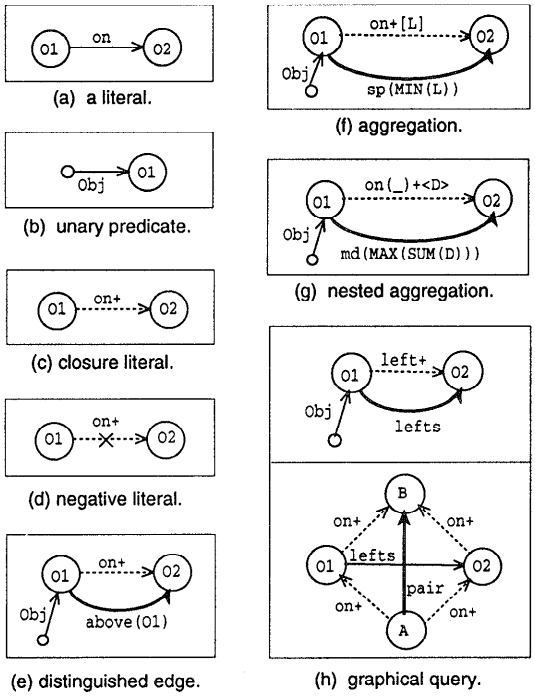


図3 GraphLog の基本演算.
Fig. 3 Primitives in GraphLog.

difference であり，g1，g2 は，導出データグラフである．st(g1，g2) を結果とする．

以上の表現要素を組み合わせることで DUO の問合せが表現できる．

4. 問合せ能力

ここでは，まず，評価対象とする GraphLog[1] について述べる．次に，DUO の表現能力を評価し，さらに，問合せ記述特性を明らかにする．

4.1 評価対象：GraphLog

GraphLog[1] の問合せ対象である DB は，ラベル付き有向マルチグラフ* である．

GraphLog の問合せの基本単位は問合せグラフである．問合せグラフでは，枝 on で結ばれている 2 つの点を図3(a)のように指定する．図では変数を大文字(O1，O2 等)で表している．また，点名が Obj という点は図3(b)のように指定する．このような始点が小さな丸の枝は unary 述語と呼ばれる．さらに，1 つ以上の枝 on で結ばれている 2 つの点を，図3(c)に示すように，on$^+$という閉包リテラルで表現できる．また，負の閉包リテラルは枝上に×印を付けて表現し（図3(d)），否定（¬）を表す．このほかに，パスのラベルを「｜」で連結してパスの和が表現でき，パスのラベルの前に「−」を付けてパスの反転が表現できる．

問合せグラフは，上記の指定の組み合わせに加えて，結果枝（a distinguished edge）を必ず 1 つだけ持つ．結果枝は，問合せ結果を表す 2 つの点間の関係であり，図3(e)の枝 above のように太線で表される．図3(e)では，1 つ以上の枝 on で結ばれている 2 つの点を枝 above で連結して問合せ結果としている．

また，集約演算子を点または結果枝に記述可能である．図3(f)のラベル on+[L] の [L] はパス長を変数 L に設定するという特殊な構文である．この変数 L を用いて，sp(MIN(L)) で最短のパスを求めている．さらに，2 つの集約演算子の入れ子も可能である．例えば，図3(g)の MAX(SUM(D)) のように，枝 on に格納されている値の和を求め，その和が最大であるパスを求めることができる．ここで，ラベル on(_)+⟨D⟩ は，枝 on の値を変数 D に設定するという特殊な構文で，入れ子になった 2 つの集約演算を使用する際に用いる．

グラフィカル問合せは，図3(h)に示すように，問合せグラフの有限集合である．1 以上の問合せグラフすべてを満足するグラフが問合せ結果として返却される．この例では，上部の問合せグラフで，1 つ以上の枝 left でつながる 2 つの点を，枝 lefts でつなげる．次に，その 2 つの点へ 1 つ以上の枝 on でつながる点 A と，1 つ以上の枝 on でその 2 つの点からつながる点 B を，枝 pair でつなげ，それを求めている．GraphLog の構文を参考のために付録に示す．

枝$\{e_0, e_1, \cdots, e_k\}$($e_0$ は結果枝）で構成される GraphLog の問合せグラフは，枝 e_i のラベルを s，始点を n_1，終点をn_2 とし，n_1，n_2 に対する変数を，おのおの，X_1，X_2 とすると，以下に示す手続きにより，s_0:− s_1&s_2&⋯&s_k．という Datalog プログラム[13] に変換できる[1),2)]．ただし，Datalog プログラムのヘッド部には非再帰の集約演算が存在できる．集約演算は 2 段の入れ子（AGG$_1$(AGG$_2$( ))）が可能なように拡張され，例えば，p(P, Q, MAX(SUM(D)))[P；Q]:− q(P, Q, D). のように表される[2)]．これは，始点を P，終点を Q とするパスに対して，パスを構成する枝の集合に対して SUM( ) を適用し，その結果の集合に対して MAX( ) を適用し，その値をラベルとし P，Q を始終点とする枝を結果とすることを表す．変換手続きを以下に示す．

(A) e_i($1 \le i \le k$) に対して以下を行う．

(1) s が p(X_3)(¬ p(X_3)) ならば，s_i は p(X_1, X_2, X_3)(¬ p(X_1, X_2, X_3)).

(2) s が $=(\neq, <, >, \le, \ge)$ ならば，s_i は $X_1 = X_2$($X_1 \neq X_2$, $X_1 < X_2$, $X_1 > X_2$, $X_1 \le X_2$, $X_1 \ge X_2$).

（3） sが $-p(X_3)$($\neg -p(X_3)$) ならば，s_i は $p'(X_1, X_2, X_3)$($\neg p'(X_1, X_2, X_3)$) で，

$p'(P, Q, R)$: $p(Q, P, R)$.

（4） sが $(p(X_3)|q(X_3))$($\neg(p(X_3)|q(X_3))$) ならば，s_i は $p'(X_1, X_2, X_3)$($\neg p'(X_1, X_2, X_3)$) で，

$p'(P, Q, R) :- p(P, Q, R)$.

$p'(P, Q, R) :- q(P, Q, R)$.

（5） sが $p(X_3)^+$ ($\neg p(X_3)^+$) ならば，s_i は $p'(X_1, X_2, X_3)$($\neg p'(X_1, X_2, X_3)$) で，

$p'(P, Q, R) :- p(P, Q, R)$.

$p'(P, Q, R) :- p(P, S, R)\&p'(S, Q, R)$.

（6） sが $p(\_)^+\langle D\rangle$($\neg p(\_)^+\langle D\rangle$) ならば，$s_i$ は $p(X_1, X_2, D)$. この場合，e_0 のラベルは必ず $q(AGG_1(AGG_2(D)))$ の形である。

（7） sが $p^+[L]$($\neg p^+[L]$) ならば，s_i は $p'(X_1, X_2, L)$($\neg p'(X_1, X_2, L)$) で，

$p_a(P, Q, 1) :- p(P, Q, R)$.

$p_a(Q, P, 1 :- p(P, Q, R)$.

$p'(P, Q, T) :- p_a(P, Q, T)$.

$p'(P, Q, L) :- p_a(P, S, L1)\&p'(S, Q, L2)\&L = L1+L2$.

（B） e_0 に対して以下を行う．ただし，AGGは，sum, prod, min, maxである。

（1） sが $p(X_3)$ ならば，s_0 は $p(X_1, X_2, X_3)$.

（2） sが $p(X_1, AGG(X_2), X_3)$ ならば，s_0 はsと同じ。

（3） sが $p(AGG(X_3))$ ならば，s_0 は $p(X_1, X_2, AGG(X_3))$.

（4） sが $p(AGG_1(AGG_2(D)))$ ならば，s_0 は $p(X_1, X_2, AGG_1(AGG_2(D)))[X_1 ; X_2]$. ただし，$X_1$, X_2 は，おのおの，枝 e_0 の始点と終点に対する変数。

グラフィカル問合せは，グラフィカル問合せを構成する個々の問合せグラフを変数の割り当てに注意して変換したDatalogプログラムの集まりで表される。

4.2 表現能力

ここでは，GraphLogで表現できる問合せがDUOで表現できることを示す．このために，GraphLogの基本的な表現がDUOで表現可能であることを示す。

GraphLogの問合せグラフが枝 $\{e_0, e_1, \cdots, e_k\}$($e_0$ は結果枝）で構成されているとする．前述のように，枝 e_i のラベルをs，始点を n_1，終点を n_2 とし，n_1, n_2 に対する変数を，おのおの，X_1, X_2 とする．GraphLogの問合せグラフからDUOの問合せグラフへの変換を以下に示す。

（A） $e_i(1 \leq i \leq k)$ に対して以下を行う。

（1） sが $p(X_3)$ の場合，

（a） 終点のみ（unary述語）の場合，（点，all, "p",, $\{X_2\}$,）なる要素とする。

（b） 上記以外の場合，要素列 $seq = elm_1elm_2elm_3$ とする．ここで，elm_1, elm_2, elm_3 は次のような要素である。

elm_1=（点，all, in(p),, $\{X_1\}$,），

elm_2=（枝，all, "p",, $\{X_3\}$,），

elm_3=（点, all, tn(p),, $\{X_2\}$,）.

ここで，in(p)はラベルpの枝の始点のラベル，tn(p)はラベルpの枝の終点のラベルを表す。

（2） sが $=(\neq, <, >, \leq, \geq)$ の場合，$X_1 = X_2(X_1 \neq X_2, X_1 < X_2, X_1 > X_2, X_1 \leq X_2, X_1 \geq X_2)$ という条件を作成し，始点または終点の検索条件rcに加える．例えば，$=$ の場合，$rc \wedge X_1 = X_2$ とする。

（3） sが $-p(X_3)$ の場合，導出データグラフ（gn_1, qg_1,）とする．ここで，gn_1 は適当な導出データグラフ名，qg_1 は問合せグラフで（{seq_1, seq_2}, {seq_2}），seq_1 は $elm_1elm_2elm_3$ なる要素列で，

elm_1=（点，all, in(p),, $\{X_1\}$,），

elm_2=（枝，all, "p",, $\{X_4\}$,），

elm_4=（点，all, tn(p),, $\{X_2\}$,）.

seq_2 は $elm_1elm_4elm_3$ なる要素列で，

elm_4=（枝，all, "$-$p",, $\{X_3\}$,）.

ただし，in($-$p)=tn(p)，tn($-$p)=in(p).

（4） sが $(p(X_3)|q(X_3))$ の場合，括弧（{seq_1, seq_2}，なし，{X_3}, { },）とする．seq_1 は $elm_1elm_2elm_3$ なる要素列で，

elm_1=（点，all, in(p),, $\{X_1\}$,），

elm_2=（枝，all, "p",, $\{X_3\}$,），

elm_3=（点，all, tn(p),, $\{X_2\}$,）.

seq_2 は $elm_1elm_4elm_3$ なる要素列で，

elm_4=（枝，all, "q",, $\{X_3\}$,）.

ただし，in(p)=in(q)，tn(p)=tn(q) である。

（5） sが $p(X_3)^+$ の場合，括弧（{seq}, +, { }, { },）とする．seqは $elm_1elm_2elm_3$ なる要素列で，

elm_1=（点，all, in(p),, $\{X_1\}$,），

elm_2=（枝，all, "p",, $\{X_3\}$,），

elm_3=（点，all, tn(p),, $\{X_2\}$,）.

（6） sが $p(\_)^+\langle X_3\rangle$ の場合，e_0 のラベルは必ず $q(AGG_1(AGG_2(X_3)))$ の形である．そこで，括弧（{seq}, +, { }, {$X_A = AGG_2(X_3)$},）とする。

seq は(5)と同じ．X_A は $AGG_2(X_3)$ を格納する一時的な変数で，後述の $q(AGG_1(AGG_2(X_3)))$ なるラベルの変換時に使用する．

(7) s が $p^+[L]$ の場合，({seq}, +, { }, {L = count(X_3)},) なる括弧とする．seq は(5)と同じ．

(8) s が $\neg p_{exp}$ (ただし，p_{exp} は上記の(1)～(7)の表現) の場合，スコープ (exp, not) とする．ただし，exp は上記の(1)～(7)を適用する．

(B) e_0 に対して以下を行う．ただし，ここでは，要素に対する変数 (X_1, X_2, X_3) が $e_i(1 \leq i \leq k)$ において定義されていない場合を示す．X_1, X_2, X_3 が定義されている場合は，以下において要素を表現する6つ組の中の変数集合は {} となる．例えば，(1)(a)の場合，(点, all, "p", , { }, X_2) となる．

(1) s が $p(X_3)$ の場合，

(a) 終点のみ(unary 述語)の場合，(点, all, "p", , {X_2}, X_2) なる要素とする．

(b) 上記以外の場合，seq = $elm_1elm_2elm_3$ なる要素列で，

elm_1=(点, all, in(p), , {X_1}, X_1),
elm_2=(枝, all, "p", , {X_3}, X_3),
elm_3=(点, all, tn(p), , {X_2}, X_2).

(2) s が $p(X_1, AGG(X_2), X_3)$ の場合，seq = $elm_1elm_2elm_3$ なる要素列で，

elm_1=(点, all, in(p), , {X_1}, X_1),
elm_2=(枝, all, "p", , {X_3}, X_3),
elm_3=(点, all, tn(p), , { }, $AGG(X_2)$).

X_2 は $e_i(1 \leq i \leq k)$ において定義されていなければならない．

(3) s が $p(AGG(X_3))$ の場合，seq = $elm_1elm_2elm_3$ なる要素列で，

elm_1=(点, all, in(p), , {X_1}, X_1),
elm_2=(枝, all, "p", , { }, $AGG(X_3)$),
elm_3=(点, all, tn(p), , {X_2}, X_2).

X_3 は $e_i(1 \leq i \leq k)$ において定義されていなければならない．

(4) s が $p(AGG_1(AGG_2(X_3)))$ の場合，$e_i(1 \leq i \leq k)$ に $p(\_)^+\langle X_3 \rangle$ なる e_i が必ず存在する．(A)(6)で示したように，この e_i のラベルは括弧({seq}, +, { }, {$X_A = AGG_2(X_3)$},) に変換される．そこで，次のような elm_1, elm_2, elm_3 で構成される要素列 $sl = elm_1elm_2elm_3$ とする．

elm_1=(点, all, in(p), , {X_1}, X_1),
elm_2=(枝, all, "p", , { }, $AGG_1(X_A)$),
elm_3=(点, all, tn(p), , {X_2}, X_2).

以上のように，GraphLog の問合せグラフの構成要素が3.2節で述べた DUO の問合せの表現要素に変換できるので，変換結果を用いて GraphLog の問合せグラフを DUO の問合せとすることができる．

GraphLog のグラフィカル問合せは問合せグラフの集まりである．したがって，グラフィカル問合せを構成する個々の問合せグラフを DUO の問合せグラフとし，作成した問合せグラフを実行順序に従って並べることで DUO の問合せに変換できる．

以上より，GraphLog のグラフィカル問合せは DUO の問合せに変換することができる．

4.3 問合せ記述特性

ここでは，問合せ記述が宣言的か手続き的かを表す問合せ記述特性を評価する．まず，評価の測度とする手続き度[11]について説明する．次に，評価方法を示し，評価結果を示す．

4.3.1 評価測度：手続き度

Welty と Stemple によって提案された手続き度[11]は言語のステップバイステップ度を示す測度である．手続き度 PM は下式で求められる．

$$PM = \frac{N_v}{N_{vo}} + \frac{N_o}{N_{oo}} \quad (1)$$

ここで，N_v は変数束縛の数，N_{vo} は許される変数束縛順序の数，N_o は演算の数，N_{oo} は許される演算順序の数である．許される順序とは評価される文で示された順序または言語の構文上および意味上評価される文と同じとみなされる順序である．以下の手続きを例に PM を求めてみる．

x=new node N；
y=select M；
copy value y to x；

この場合，束縛変数は x と y である．変数 x と y の順序を変えることはできないので N_{vo} は1である．3つの演算があるので演算数 (N_o) は3である．new node 文と select 文の順序は意味に影響しないので N_{oo} は2である．したがって，PM=2/1+3/2=3.5 である．

4.3.2 評価方法

表1の問合せ例を DUO と GraphLog で記述し，手続き度を計算することで評価する．以降では，表1の分類番号 x の問合せ例を“問合せ x”と呼ぶ．また，GraphLog では問合せ6を記述できないので，GROG[4]の表記法で拡張することとする．問合せは，図1(a)に示すスキーマに基づいて行う．手続き度の計算にあたり，DUO ならびに GraphLog の1問合せグラフを1演算とする．単なる条件記述のための変数は束

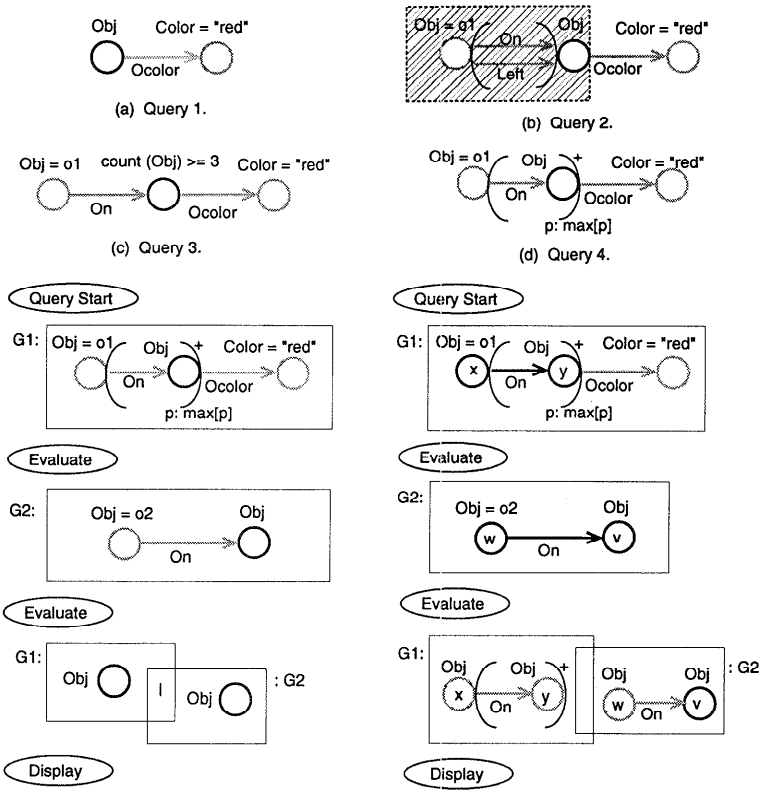


図4 DUOでの問合せ表現結果
Fig. 4 Query representations in DUO.

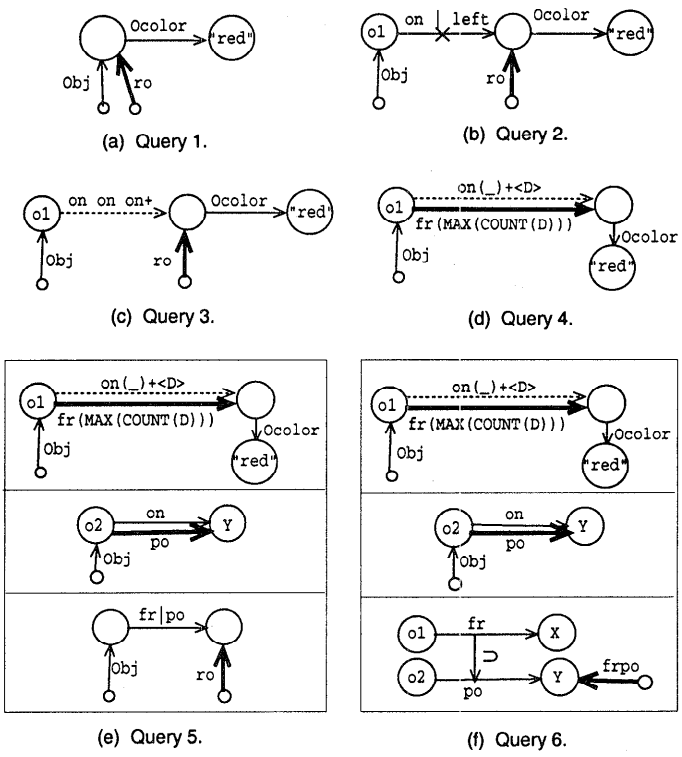


図5 GraphLogとその拡張での問合せ表現結果
Fig. 5 Query representations in GraphLog and its extension.

表 2 DUO と GraphLog での問合せ記述の手続き度
(*：GraphLog を拡張して使用)
Table 2 Values of Procedure Metrics for describing queries in DUO and GrophLog.
(*：Extended GraphLog is used.)

問合せ番号	DUO	GraphLog
1	1	1
2	1	1
3	1	1
4	1	1
5	3.5	2.5
6	7.5	6.5*

縛変数にカウントしないこととする.

4.3.3 評価結果

DUO と GraphLog (問合せ 6 は拡張 GraphLog) で記述した結果を，おのおの，**図 4** と**図 5** に示す．問合せ 5 では，第一演算と第二演算の順序は可換であり，変数 G 1 と G 2 の順序も可換である．問合せ 6 では，第一演算と第二演算の順序は可換であるが，演算 ⊃ のために変数 G 1 と G 2 の順序は可換ではない.

手続き度の計算結果を**表 2** にまとめて示す．問合せ 1 から 4 では DUO と GraphLog は手続き度が同一 (1) である．問合せ 5 と 6 では，両言語とも問合せ 1 から 4 と比較して手続き度が高くなる．これは，束縛変数を多数使用するためである．このように，問合せの複雑さに関して，DUO は GraphLog と同様な手続き度特性を持つ.

問合せ 5 と 6 では，DUO の方が GraphLog よりも手続き度が高い．これは，DUO では問合せを手続き的に記述するように制限しているのに対し，GraphLog では複数の問合せグラフを宣言的に記述させているからである.

5. おわりに

グラフに基づく DB に対するグラフィカルな問合せ言語 DUO の問合せ能力を，高い表現能力を持つ GraphLog と比較して評価した．この結果，DUO は GraphLog と少なくとも同等の表現能力を持ち，問合せの複雑さに関して同様な問合せ記述特性を持つことを明らかにした.

DUO は，複雑な問合せは手続き的に記述した方が良いという考え[11]に基づいて設計されてきた．しかし，GraphLog のような Logic Programming 形式で問合せを記述できる方がより問合せ記述が容易とも考えられる．また，問合せ記述特性のみでは問合せ記述の容易性を判断できるか疑問で，手続き記述の煩雑さも考慮する必要あるのではないかと考えられる.

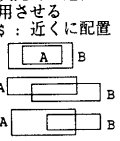
以上のような，DUO の Logic Programming 形式化や問合せ記述の容易性を評価する尺度の検討，ならびに，実装，類似度を考慮した検索，異表現同意義の考慮が今後の課題である.

謝辞 GraphLog について御教示いただいた Dr. Mariano P. Consens に感謝いたします．また，本論文執筆にあたり有益なコメントをいただいた論文査読委員の皆様に感謝いたします．さらに，本研究を行うにあたり日頃から熱心に議論していただいた，福井大学工学部情報工学科 都司達夫教授に感謝いたします．本研究は筆者が NTT 情報通信網研究所在職中に着手したものである．本研究を行うにあたり熱心に議論していただいた，井上潮主幹研究員をはじめとする NTT 情報通信網研究所の皆様に感謝いたします.

参考文献

1) Consens, M. P. and Mendelzon, A. O.: GraphLog: a Visual Formalism for Real Life Recursion, *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 404-416 (1990).
2) Consens, M. P. and Mendelzon, A. O.: Low-complexity aggregation in GraphLog and Datalog, *Theoretical Computer Science*, Vol. 116, pp. 95-116 (1993).
3) Curz, I. F., Mendelzon, A. O. and Wood, P. T.: G^+: Recursive Queries without Recursion, *Proc. 12th Int. Conference on Expert Database Systems*, pp. 355-368 (1988).
4) Mainguenaud, M.: GROG: Geographical Queries using Graphs, *Proceedings of Advanced Database System Symposium*, pp. 91-95 (1989).
5) Gyssens, M., Paredaens, J. and Gucht, D. V.: A Graph-Oriented Object Database Model, *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 417-424 (1990).
6) Andries, M. and Paredaens, J.: Macro's for the GOOD-Transformation Language, Technical Report 91-36, University of Antwerp (UIA) (1991).
7) Houchin, T.: DUO: Graph-based Database Graphical Query Expression, *Proc. 2nd Far-East Workshop on Future Database Systems*, pp. 286-295 (1992).
8) 宝珍輝尚：グラフィカルな問合せ言語 DUO の問合せ記述の容易性，1993 年信学春季全大，D-93 (1993).

```
| : 選択肢   @ : 2次元的に連結する
[] : 省略可   + : 作用させる
, : 並べて配置   $ : 近くに配置
A ◎ B : 覆う(右記)
A × B : 右のように クロスする
A ⊃ B : 右のように クロスする
C * S : Cの重なりにSを配置
```



付図 1 構文規則
Fig. A1 Syntax rules.

```
<query graph> ::= <qpath>
<qpath>      ::= { <node> | <reg path> } [ <path ext> ]
<node>       ::= <nfig> $ <elm label>
<nfig>       ::= "◯"

<elm label>  ::= STRING
<reg path>   ::= <cpath> | <upath> | <lpath>
<cpath>      ::= <qpath> @ <iic path>
                 | <tic path> @ <qpath>
                 | <qpath> @ <itic path> @ <qpath>
<upath>      ::= <qpath> @ <plist> @ <qpath>
<plist>      ::= <ibrace> <itic path list> <tbrace>
<lpath>      ::= <qpath> @ <ibrace> <iic path> <tbrace>
                 | <ibrace> <tic path> <tbrace> @ <qpath>
<iic path>   ::= <edge> @ <qpath> [ <path ext> ]
<tic path>   ::= <qpath> @ <edge> [ <path ext> ]
<itic path>  ::= <edge> [ @ <qpath> @ <edge> ][ <path ext> ]
<itic path list> :: = <itic path> [, <itic path list> ]
<edge>       ::= <efig> $ <elm label>
<efig>       ::= "⟶"
<ibrace>     ::= <ibfig>
<ibfig>      ::= "("
<tbrace>     ::= <tbfig> $ <csign> $ <brace label>
<tbfig>      ::= <ibfig> + <<rotation>>
<csign>      ::= "*" | "+"
<brace label> ::= STRING
<path ext>   ::= [ ◎ <scope> ] [ <get val> ]
<scope>      ::= <pos> | <neg>
<pos>        ::= <dot sq>
<dot sq>     ::= "┌┐"
                 "└┘"

<neg>        ::= <dot sq> + <n_hatching>
<n_hatching> ::= <<hatching>>
<get val>    ::= + <<bold>>

<query exp>  ::= "Query Start" <query seq> "Display"
<query seq>  ::= <eqg> [ "Evaluate" <query seq> ]
<eqg>        ::= <dqg> | <set op> | <set comp>
<dqg>        ::= <dqg bogy> $ <dg label>
<dg label>   ::= ":" $ STRING
<dqg body>   ::= <gqg> ◎ <sq> [ <complement> ]
<gqg>        ::= <query graph> | <dqg>
<sq>         ::= "┌──┐"
                 "└──┘"
<complement> ::= + <m_hatching>
<m_hatching> ::= <<hatching>>
<set op>     ::= <union> | <intersection> | <difference>
<union>      ::= <dqg> <uop> <dqg>
<uop>        ::= <inop> * "|"
<inop>       ::= ◎ | ⊃
<intersection> ::= <dqg> <iop> <dqg>
<iop>        ::= <inop> * "&"
<difference>   ::= <dqg> <dop> <dqg>
<dop>        ::= ⊃ [ * "-" ]
<set comp>   ::= <set equal> | <set inclusion>
<set equal>  ::= <gqg> <seop> <gqg>
<seop>       ::= <crop> * <scie>
<crop>       ::= × | ⊃
<scie>       ::= [ <sci> ] "="
<sci>        ::= "ABS" | "VAL" | "STR" | "MAP"
<set inclusion> ::= <gqg> <siop> <gqg>
<siop>       ::= ⊃ * [ <sci> ]
```

付図 2 DUO の構文
Fig. A2 Syntax of DUO.

9) Ioannidis, Y. E. (eds.): Special Issue: Advanced User Interface for Database Systems, SIGMOD RECORD, Vol. 21, No. 1 (1992).
10) Batini, C., Catarci, T., Costabile, M. F. and Levialdi, S.: Visual Query Systems: A Taxonomy, *Proc. IFIP WG 2.6 2nd Working Conference on Visual Database Systems*, pp. 159-173 (1991).
11) Welty, C. and Stemple, D. W.: Human Factors Comparison of a Procedural and a Non-procedural Query Language, *ACM Trans. Database Syst.*, Vol. 6, No. 4, pp. 626-649 (1981).
12) Reisner, P.: Use of Psychological Experimentation as an Aid to Development of a Query Language, *IEEE Trans. Softw. Eng.*, Vol. SE-3, No. 3, pp. 218-229 (1977).
13) Ullman, J. D.: Principles of Database and Knowledge Base Systems (Volume I), Computer Science Press (1988).

付録 DUO と GraphLog の構文

DUO と GraphLog の構文を示す．両言語ともグラフィカルな言語であるため，通常の BNF では構文を記述できない．そこで，ここでは，付図 1 に示す構文

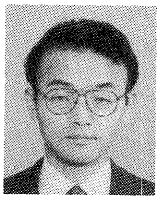
```
<query graph> ::= <qgraph>
<qgraph>     ::= <ext node> [ @ <qgelm> ]
<ext node>   ::= <node> [ <unary edge> ]
<node>       ::= <nlabel> ◎ <nfig>
<nlabel>     ::= STRING
<nfig>       ::= "◯"

<unary edge> ::= <upred edge> | <udist edge> | <uneg edge>
<upred edge> ::= <upfig> $ <pname>
<upfig>      ::= "⟶○"
<pname>      ::= STRING
<udist edge> ::= <upfig> + <<bold>> $ <dist exp>
<dist exp>   ::= <literal> | <pname> [ "(" <aggr exp> ")" ]
<literal>    ::= STRING
<aggr exp>   ::= <aggr func> "(" <dist exp> ")"
<aggr func>  ::= "count" | "product" | "sum"
                 | "max"   | "min"
<uneg edge>  ::= <upred edge> ◎ <negfig>
<negfig>     ::= "×"
<qgelm>      ::= <pred edge> @ <ext node> [ @ <qgelm> ]
<pred edge>  ::= <pos edge> | <neg edge> | <dist edge>
<pos edge>   ::= <prim edge> | <clos edge>
<prim edge>  ::= <oefig> $ <ext path exp>
<oefig>      ::= "⟶"
<ext path exp> ::= <path exp> | <arith exp>
<path exp>   ::= <pname> | "_"
                 | "~" "(" <path exp> ")"
                 | "(" <path exp> "|" <path exp> ")"
<arith exp>  ::= "=" | "<" | ">" | "<=" | ">="
<clos edge>  ::= <cefig> $ <path rexp>
<cefig>      ::= "⇢"
<path rexp>  ::= <path exp>
                 | "(" <path rexp>  <path rexp> ")"
                 | "(" <path rexp> ")" "+"
                 | "¬" "(" <path rexp> ")"
<neg edge>   ::= <pos edge> ◎ <negfig>
<dist edge>  ::= <oefig> + <<bold>> $ <dist exp>

<graphical query> ::= <query graph> [ <graphical query> ]
```

付図 3 GraphLog の構文
Fig. A3 Syntax of GraphLog.

規則を設ける．選択肢（“|”）と省略可（“[”, “]”）に加え，8つの規則を設ける．“@”は二次元的に隣接することを示す．これは，グラフの点と枝の接続関係の記述に用いる．“＋”は表示に関する処理を施すことを示す．例えば，＋≪hatching≫はハッチングを施すことを表す．ここで，≪hatching≫のように表示に関する処理は“≪”と“≫”で囲む．“◎”, “×”, “⊃”は，図形の配置規則である．“＊”は，“×”または“⊃”と共に用いられ，重なり部分に指定されたものを配置することを示す．“,”は平行に記述することを示し，“$”は近接して配置することを示す．

本規則を使用したDUOの構文を**付図2**に，GraphLogの構文を**付図3**に示す．

（平成6年3月22日受付）
（平成7年2月10日採録）



宝珍　輝尚（正会員）

昭和34年生．昭和57年名古屋工業大学電気工学科卒業．昭和59年同大大学院修士課程修了．同年，日本電信電話公社入社．NTT情報通信網研究所を経て，平成5年7月より福井大学工学部情報工学科助手，現在に至る．マルチメディアデータベース管理システム，拡張可能データベース管理システム，グラフィカルなデータベース問合せの研究に従事．電子情報通信学会，IEEE, ACM各会員．