ISSN 1346-5597

National Institute of Informatics

NII Technical Report

書き換えに基づく最適化のための XQuery の相対コストモデル

A Relative Cost Model of XQuery for Optimization Based on Rewriting

日高 宗一郎，加藤 弘之，吉川 正俊
Soichiro HIDAKA,Hiroyuki KATO, and Masatoshi YOSHIKAWA

NII-2008-003J
Feb. 2008

URL: http://www.nii.ac.jp/ Tokyo 101-8430, JAPAN +81-3-4212-2000 FAX: +81-3-3556-1916

書き換えに基づく最適化のためのXQueryの相対コストモデル(注*)

日高宗一郎[†,††] 加藤　弘之[†,††] 吉川　正俊[†††]

† 国立情報学研究所，東京都
†† 総合研究大学院大学 複合科学研究科 情報学専攻，東京都
††† 京都大学大学院情報学研究科，京都府

あらまし　XQueryはXMLに対する関数型の問合せ言語である．本論文では，ソースレベル変換の過程での性能利得を見積もることができるXQuery向けコストモデルを提案する．本研究の目的は，様々な書き換え手法を実エンジンを導入することなしに評価することを容易にすることである．本コストモデルは言語の持つ関数型の構造に従った単純な再帰関数で構成される．それ等は形式的意味論や良く知られた効率的なアルゴリズムを参考に決定されている．変換の前後の式のコストの解析的な比較がエンジンに依らず可能となっている．モデルの相対性により，解釈不要の成分を許容することができ，また，その部分はコスト比較の形式的な証明の妨げにもならない．さらに，必要であれば被演算数の評価順等のエンジン固有の評価戦略を反映させるように改変することも可能となっている．

キーワード　問合せ言語，プログラム変換，コストモデル

A Relative Cost Model of XQuery for Optimization Based on Rewriting

Soichiro HIDAKA[†,††], Hiroyuki KATO[†,††], and Masatoshi YOSHIKAWA[†††]

† National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan
†† Department of Informatics, The Graduate University for Advanced Studies, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan
††† Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 Japan

Abstract　XQuery is a functional query language for XML. We propose a relative XQuery cost model that is able to estimate the performance gain during source level transformation. This research facilitates the evaluation of various rewriting techniques without introducing real engines. The cost model consists of simple recursive functions based on functional language constructs. They are determined using formal semantics and other known efficient algorithms. Analytic comparison of costs between expressions before and after transformation is possible in an engine-independent manner. The relativity of the model allows uninterpreted components within, which do not affect the mathematical proof of the comparison. Moreover, it can be tailored to reflect engine specific evaluation strategies such as the order of evaluation of operands.

Keywords　Query languages, Program transformation, Cost model

1. ま え が き

データベースの問合せ最適化研究では，性能評価はしばしば実エンジンを用いた実験に基づいて行われる．しかし，その実験は自前のエンジンで行われるため[1]，結果が他のエンジンにも当てはまることは必ずしも保証されない．

本論文の対象としている言語であるXQuery[2]は比較的新しいため，研究段階にあるエンジンも多い．したがって，誰もが自由に利用できる安定した参照エンジンはまだ存在しない．更に，現存するエンジンも進化し続けるため，評価結果がすぐ古くなってしまう．エンジンの導入自体も手間が必要である．

本研究の目的は，書換えに基づく最適化の研究のための，評価基盤となる“仮想エンジン”のような役割を担うコストモデルを構築することにある．既存のコストモデルには，個々のエンジンの中で生成された物理プランごとのコストを具体的な値として見積もり[3], [4]，その値が最小となるようなものを選択させるものが多く見られる．本論文では，このような物理レベルではなくソースレベルでの書換えを対象としている．データベース操作言語におけるソースレベル変換は重要である．一般にデータベース操作言語は宣言的であり，より手続き的な論理

(注*)：本稿は電子情報通信学会論文誌 Vol.J91-D,No.04,Apr.2008 に掲載予定の論文「書換えに基づく最適化のための XQuery の相対コストモデル」のフルバージョンである．

代数などを経て，最終的に物理プランとして実行される．これら各レベルでの最適化が実施されるが，宣言的な言語における書換えは，手続き的な言語より大局的な最適化が可能である [5]．実際，SQL において「マジックセット書換え法」という再帰論理プログラムのために開発された最適化技術を非再帰 SQL に適用することで，大局的な問合せ最適化に成功している例もある [6]．

関係データベースに XML データを格納し，XQuery を SQL に変換することで問合せ処理を実行するシステムがいくつか提案されている ([7], [8] など)．このようなシステムにおいても，XQuery レベルでの最適化後に SQL を生成した方が結果としてより効率的な問合せが可能となる．実際，単純な経路式であっても，そこからナイーブに生成された SQL に対する，従来の関係データベースの最適化機構を用いた最適化問題は，多項式アルゴリズムが与えられない (intractable) ことが示されている [9]．

我々は，XQuery のソースレベル書換えに基づく最適化アルゴリズムを開発しており [10]，そこでは

$$\langle QName\rangle\{e\}\langle/QName\rangle/\mathbf{child}::*=e^{(注1)}$$

のように明らかに冗長性の削られるような書換え規則の適用に持ち込むために数多くの補助変換を用いている．このような補助変換が全体の最適化効果を損なわないことを保証することも本研究の動機の一部となっている．

本研究の寄与は，(1) 書換えにおけるコストの利得あるいは損失の，エンジン独立な形での簡易，静的，かつ証明可能な見積りを与えること，(2) 被演算数の評価順のようなエンジンごとの評価戦略の相違をある程度反映できることで，エンジンごとに書換え規則が選択できるようにすること，(3) 実エンジンの評価コストを実際の問合せについてモデル化できていることである．

データベース操作言語では，通常のプログラム言語と異なり，処理がデータ集約的であり，入力のデータのサイズが大きく作用する．提案するコストモデルは，このようなデータの大きさを考慮したコスト計算を行う．例えば XQuery では，繰返し構造に対応する **for** 式に対して，**for** $\$x$ **in** e_1 **return** e_2 における e_1 の評価結果の列としての長さが一般的に大きくなる．また，エレメント構築に対しては，仕様が求める，部分木の再帰的コピーを反映して $e//*$ の列としての長さに比例する要素を導入している．データ統合問題におけるスキーママッピングによる構造変換は，このようなエレメント構築が頻繁に起こり得る [11]．提案するコストモデルは，このような処理でもたらされる冗長性の削除のための，コストの定量化にも役立てることができる．

評価結果では，作業領域の有限性を考慮していないことによる実エンジンの振舞いとの乖離も確認された．しかしながら，このような有限性は，エンジンにパイプライン処理等を組み込むことにより捨象可能であるため，OS の分野で仮想記憶により仮想メモリ空間の制約が撤廃されたように，そのような有限性を考慮しないモデルも十分意義があると考えられる．

本論文の基本的な考え方を例により示す．同様の考え方が，図 1 に示されている全ての構文要素を任意に組み合わせた変換に対しても適用される．より自明でない適用結果については **3.** で詳述する．

以下の変換と，コスト計算式について考える．

$$e_1\ \mathbf{and}\ e_2$$
$$=\mathbf{if}\ (e_1)\ \mathbf{then}\ e_2\ \mathbf{else}\ \texttt{fn:false()} \qquad (\textsc{Taif})$$
$$\mathtt{c}(e_1\ \mathbf{and}\ e_2)=\mathtt{c}(e_1)+\mathtt{p}(e_1)\cdot\mathtt{c}(e_2) \qquad (\textsc{Cand})$$
$$\mathtt{c}(e_1\ \mathbf{and}\ e_2)=\mathtt{c}(e_1)+\mathtt{c}(e_2) \qquad (\textsc{Cnand})$$
$$\mathtt{c}(\mathbf{if}\ (e_1)\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3)=\mathtt{c}(e_1)$$
$$+\mathtt{p}(e_1)\cdot\mathtt{c}(e_2)+(1-\mathtt{p}(e_1))\cdot\mathtt{c}(e_3) \qquad (\textsc{Cif})$$

ここで，`fn:false()` は XQuery におけるブール値の偽を表現する正式な記法である．$\mathtt{c}(e)$ と $\mathtt{p}(e)$ は，それぞれ式 e のコスト (評価に要する時間) 及び式 e が真に評価される確率を表す．書換え規則 (Taif) により，**and** 式が意味的に等価な **if** 式で置き換えられる．(Cand) では e_1 が偽に評価された場合は e_2 を無視するという short-circuit 評価戦略を取り込んでいる．この戦略は形式的意味 [12] で明示的に許容されている．一方，(Cnand) は **and** 式が常に両方の被演算数を評価するという評価戦略をモデル化している．この非 short-circuit 評価戦略のもとでの (Taif) によるコスト変化は，単に次のように (Cif) を (Cnand) から引くことにより計算できる．

$$\begin{aligned}&\mathtt{c}(e_1)+\mathtt{c}(e_2)\\&\quad-\mathtt{c}(e_1)-\mathtt{p}(e_1)\cdot\mathtt{c}(e_2)-(1-\mathtt{p}(e_1))\cdot\mathtt{c}(\texttt{fn:false()})\\&=(1-\mathtt{p}(e_1))\cdot\mathtt{c}(e_2)-(1-\mathtt{p}(e_1))\cdot\mathtt{c}(\texttt{fn:false()})\\&=(1-\mathtt{p}(e_1))\cdot(\mathtt{c}(e_2)-\mathtt{c}(\texttt{fn:false()}))\end{aligned}$$

定数 `fn:false()` の評価コストが十分小さいとすれば，この計算結果は部分式 e_1 及び e_2 によらず常に正の値になることが分かる (e_2 も `fn:false()` である場合は正ではなく 0 となるが，そのような場合には (Taif) を用いず，左辺全体を `fn:false()` と変換する方が自然であろう)．ここで上記の簡単な計算において式 e_1 のコストはキャンセルされていることに注意されたい．これが提案するコストモデルの相対性の利点である (上記の寄与 (1))．また，short-circuit 評価戦略をとらないエンジンでは，(Taif) を逆方向 (右辺から左辺) に適用すると速度低下を招くため，そのような変換は適用すべきでないこと，一方 (Cand)(short-circuit 評価戦略をとるエンジン) では，そのような変換は安全であることをコストモデルから示すことができる (上記寄与 (2))．更に，実エンジンで本コストモデルを評価した結果，本モデルで記述できない，コストが文脈依存性や残存作業領域依存性を持つようなエンジンも存在することが確認できた．

本論文の以降の構成は次のとおりである．**2.** では提案するコストモデルの導出について述べる．**3.** では，最適化で用いられる様々な変換規則におけるコスト変化の計算について述べる

(注1)：この変換の正当性については **3.** で議論する．

```
XQ  := literal
  |  (XQ)                                          /* parenthesized expr. */
  |  XQ,XQ                                         /* sequence */
  |  for $QName in XQ (where XQ)?
       (order by XQ)? return XQ                    /* for expr. */
  |  let $QName := XQ (where XQ)?
       return XQ                                   /* let expr. */
  |  some $QName in XQ satisfies XQ                /* existential */
  |  every $QName in XQ satisfies XQ               /* universal */
  |  ⟨QName⟩{XQ}⟨/QName⟩                           /* direct elem. constructor */
  |  element QName {XQ}                            /* computed elem. const. */
  |  XQ/QName | XQ//QName                          /* relative path expr. */
  |  XQ[XQ]                                        /* predicate expr. */
  |  $QName                                        /* variable reference */
  |  if (XQ) then XQ else XQ                       /* conditional expr */
  |  XQ or XQ | XQ and XQ                          /* logical expr. */
  |  /QName | //QName                              /* absolute path expr. */
  |  QName(XQ,XQ,...,XQ)                           /* function call */
  |  op(XQ,XQ,... ,XQ)                             /* other uninterpreted functions
                                                     and n-ary operators */
```

図 1　本論文で扱う XQuery の抽象構文

Fig. 1　Abstract syntax of the XQuery subset discussed in this paper.

(上記寄与 (1))．また，変換規則の部分式の評価戦略の選択が変換によるコスト変化に与える影響についても取り上げる (寄与 (2))．**4.** では，実在の XML エンジンを用いたモデルの評価結果を示す (寄与 (3))．最後に，**5.** 及び **6.** で，関連研究及び今後の課題について述べる．

2. コストモデル

本章では，コストモデルの導出とその妥当性について述べる．対象とする XQuery の構文を図 1 に示す．最後の生成規則は，書換えに関与しない演算子や関数呼出しを包摂するものであり，そのコストは引数のコストと，演算子自体のコストの和で表される．演算子自体が書換えに関与しないため，コスト差分の計算時に値は解釈されず，証明にも支障を来さない．また，ユーザ定義の関数の再帰は考慮していない．

モデルは正しい結果をもたらす実装を対象とする必要があるため，(1) 基本的には形式的意味 [12] から導出する．ただし，(2) より効率的な評価戦略が知られている式については，その戦略を採用する．エレメント構築式については，そのコストが上記 (1)，(2) のどちらにも見つからないため，筆者等が独自に推定する．モデルの現実性を担保するために，Galax [13] 及び eXist [14] といった実エンジンを用いて，実エンジンをうまく模擬しているかどうかの検証も行う．

コストモデルは以下の三つの関数から構成される．**コスト関数** $\mathtt{c}(e)$ は式 e の部分式のコスト，サイズ，確率関数を用いて再帰的に定義される．**サイズ (補助) 関数** $\mathtt{s}(e)$ 及び**確率 (補助) 関数** $\mathtt{p}(e)$ は e の部分式のサイズ，確率関数を用いて定義される．$\mathtt{s}(e)$ 及び $\mathtt{p}(e)$ は e の部分式のコストによらないことに注意されたい．

構文要素ごとのコスト関数を表 1 に要約する. 実行順としては逐次実行が仮定されている．例えば，**for** 式のコスト (Cfor) は形式的意味 [12] を用いて決定する．形式的意味の *Dynamic Evaluation* 節において，前提部の判定式の数は入力列中のアイテムの数に等しい．更に，**return** 節 (e) の**一様**な評価コストを仮定して，(Cfor) を得る．

表中の定数のうち，C_e はエレメント構築の単価，C_{vr} は変数参照の単価，k はノードあたりの子ノード数 (次数) の平均値 (データに依存) を示す．k', k'', k''' は比例係数である．c, d については**経路式の評価コスト**の節で述べる．

仮定する実行モデルの抽象度

洗練された実装を持つエンジンでは，通常正規化，書換え，物理プランの生成等，処理が幾つかの段階に分かれている．実行モデルをより形式的に記述しようとするならば，ここで [15] のような，論理レベルの演算子 (図 1 に示したもの) からのコンパイルのフェーズ ($[\![\cdot]\!]$ で表す) を仮定し，そこからコスト関数を導出することも可能である(注2)．例えば，$[\![\textbf{for } \$x \textbf{ in } e \textbf{ return } e]\!] = seqmap\ (\lambda \$x.[\![e]\!])\ [\![e]\!]$，但し

$$
\begin{aligned}
&seqmap\ e\ ()   &&= ()\\
&seqmap\ e\ (i)   &&= (e\ i)\\
&seqmap\ e\ (i, s) &&= ((e\ i), (seqmap\ e\ s))
\end{aligned}
$$

3 番目の定義は，入力列がアイテム i と列 s を連結したものである場合の振舞いを記述している．*seqmap* は入力列 s の各アイテムについて (**return** 節に対応した)e を評価した結果を連結する． 同様に，(Csome) に対しては，$[\![\textbf{some } \$x \textbf{ in } e \textbf{ satisifies } p]\!] = esome\ (\lambda \$x.[\![p]\!])\ [\![e]\!]$，ただし

$$
\begin{aligned}
&esome\ p\ ()   &&= False\\
&esome\ p\ (i)   &&= (p\ i)\\
&esome\ p\ (i, s) &&= if\ (p\ i)\ True\ (esome\ \ p\ s)
\end{aligned}
$$

esome の 3 番目の定義は，(Cnsome) ではなく (Csome) に対応し，入力列の各アイテムに対し，述語 p が *True* を返すものを見つけ次第，残りのアイテムについて p を評価することなしに *True* を返せることをモデル化している．なお，限量式のモデル化に際し，**satisfies** 節が真と評価される確率は一様で，各束縛ごとに独立であることを仮定している．

XQuery Core との関係

for 節や **let** 節の複数ある FLWOR 式のコストは，**order by** 節がない場合は XQuery Core に正規化して算出する．**order by** 節がある場合は，正規化できない．**order by** 節は，複数の **in** 節に対し，**in** 節の数に対応する要素数を持ち，**in** 節の式のサイズの積に相当する長さのタプルストリームを生成したのち，それを並べ換えるというものであるが，Core を定義している形式的意味 [12] にはタプルストリームの概念がないため，Core に変換することができないからである．計算式は表中の (Cfwor) 及び (Sfwor) となる．**some**/**every** 式については，正規化して計算する．経路式は正規化することは可能であるが，正規化してしまうと，経路式の長さに比例した数の

(注2)：但し現行のコストモデルは [15] にあるような nested loop join や hash join のような，より具体的な評価アルゴリズムを捕捉することはできない．

表 1　図 1 の各構文要素に対するコスト関数の要約

Table 1　Summary of cost functions for language constructs in Fig. 1.

expression	cost	label	size	label
for $\$x$ **in** s **return** e	$\mathtt{c}(s)+\mathtt{s}(s)\cdot\mathtt{c}(e)$	CFOR	$\mathtt{s}(s)\cdot\mathtt{s}(e)$	SFOR
for $\$x$ **in** s **order by** g **return** e	$\mathtt{c}(s)+\mathtt{s}(s)\cdot(\mathtt{c}(g)+k'''\cdot\log(\mathtt{s}(s))+\mathtt{c}(e))$	COFOR		SOFOR
for $\$x$ **in** s **where** p **return** e	$\mathtt{c}(s)+\mathtt{s}(s)\cdot(\mathtt{c}(p)+\mathtt{p}(p)\cdot\mathtt{c}(e))$	CWFOR	$\mathtt{s}(s)\cdot\mathtt{p}(p)\cdot\mathtt{s}(e)$	SWFOR
for $\$x_1$ **in** s_1, $\$x_2$ **in** s_2 **return** e	$\mathtt{c}(s_1)+\mathtt{s}(s_1)\cdot\mathtt{c}(s_2)+\mathtt{s}(s_1)\cdot\mathtt{s}(s_2)\cdot\mathtt{c}(e)$		$\mathtt{s}(s_1)\cdot\mathtt{s}(s_2)\cdot\mathtt{p}(p)\cdot\mathtt{s}(e)$	
for $\$x_1$ **in** $=s_1$, $\ldots\$x_N$ **in** s_N **where** p **order by** g **return** e	$\sum_{i=1}^{N}\mathtt{c}(s_i)$ $+(\prod_{i=1}^{N}\mathtt{s}(s_i))$ $\cdot(\mathtt{c}(p)$ $+\mathtt{p}(p)\cdot(\mathtt{c}(g)+k'''\cdot\sum_{i=1}^{N}\log(\mathtt{s}(s_i))+\mathtt{c}(e)))$	CFWOR	$(\prod_{i=1}^{N}\mathtt{s}(s_i))\cdot\mathtt{p}(p)\cdot\mathtt{s}(e)$	SFWOR
literal	constant		(uninterpreted)	
$\langle QName\rangle\{e\}\langle/QName\rangle$	$\mathtt{c}(e)+(1+\mathtt{s}(e)+\mathtt{s}(e//*))\cdot C_e$	CDEC	1	SDEC
element $QName$ $\{e\}$		CCEC		SCEC
$e/QName$	$\mathtt{c}(e)+\mathtt{s}(e)^c\cdot k'$	CCSTEP	$k\cdot\mathtt{p}(\text{nodename}==QName)\cdot\mathtt{s}(e)$	SCSTEP
$e/q_1/q_2/\ldots/q_N$	$\mathtt{c}(e)+\mathtt{s}(e)^c\cdot N^d\cdot k'$	CGSTEP	$\mathtt{s}(e)\cdot k^N\cdot\prod_{i=1}^{N}\mathtt{p}(\text{nodename}==q_i)$	SGSTEP
$e//QName$	$\mathtt{c}(e)+k''\cdot\mathtt{s}(e//*)$	CDSTEP	$\mathtt{s}(e//*)\cdot\mathtt{p}(\text{nodename}==QName)$	SDSTEP
e_1, e_2	$\mathtt{c}(e_1)+\mathtt{c}(e_2)$	CSEQ	$\mathtt{s}(e_1)+\mathtt{s}(e_2)$	SSEQ
let $\$x:=s$ **return** e	$\mathtt{c}(s)+\mathtt{c}(e)$	CLET	$\mathtt{s}(e)$	SLET
$\$x$	C_{vr}	CVR	$\mathtt{s}(\text{dereference}(\$x))$	SVR
(e)	$\mathtt{c}(e)$	CPAREN	$\mathtt{s}(e)$	SPAREN
if (e_1) **then** e_2 **else** e_3	$\mathtt{c}(e_1)+\mathtt{p}(e_1)\cdot\mathtt{c}(e_2)+(1-\mathtt{p}(e_1))\cdot\mathtt{c}(e_3)$	CIF	$\mathtt{p}(e_1)\cdot\mathtt{s}(e_2)+(1-\mathtt{p}(e_1))\cdot\mathtt{s}(e_3)$	SIF
e_1 **or** e_2	$\mathtt{c}(e_1)+(1-\mathtt{p}(e_1))\cdot\mathtt{c}(e_2)$	COR	1	SOR
e_1 **and** e_2	$\mathtt{c}(e_1)+\mathtt{p}(e_1)\cdot\mathtt{c}(e_2)$	CAND	1	SAND
some $\$x$ **in** s **satisfies** p	$\mathtt{c}(s)+\mathtt{s}(s)\cdot\mathtt{c}(p)$	CNSOME	1	SSOME
	$\mathtt{c}(s)+\frac{1-(1-\mathtt{p}(p))^{\mathtt{s}(s)}}{\mathtt{p}(p)}\mathtt{c}(p)\quad(0<\mathtt{p}(p))$	CSOME		
every $\$x$ **in** s **satisfies** p	$\mathtt{c}(s)+\frac{1-\mathtt{p}(p)^{\mathtt{s}(s)}}{1-\mathtt{p}(p)}\mathtt{c}(p)\quad(\mathtt{p}(p)<1)$	CEVERY	1	SEVERY
$e_1[e_2]$	$\mathtt{c}(e_1)+\mathtt{s}(e_1)\cdot\mathtt{c}(e_2)$	CPRED	$\mathtt{p}(e_2)\cdot\mathtt{s}(e_1)$	SPRED
$\$QName(e_1,e_2,\ldots,e_N)$	$\mathtt{c}(e_1)+\mathtt{c}(e_2)+\cdots+\mathtt{c}(e_N)$ $+\mathtt{c}(\text{function\_body}(\$QName))$	CFCALL	$\mathtt{s}(\text{function\_body}(\$QName))$	SFCALL
expression	probability	label		
some $\$x$ **in** s **satisfies** p	$1-(1-\mathtt{p}(p))^{\mathtt{s}(s)}$	PSOME		
every $\$x$ **in** s **satisfies** p	$\mathtt{p}(p)^{\mathtt{s}(s)}$	PEVERY		
e_1 **and** e_2	$\mathtt{p}(e_1)\cdot\mathtt{p}(e_2)$	PAND		
e_1 **or** e_2	$1-(1-\mathtt{p}(e_1))\cdot(1-\mathtt{p}(e_2))$	POR		

表中の記号の意味: e,g: 式，s: 列としての式，p: 述語としての式，C_e,C_{vr},$c,d,k,k',k'',,k'''$: ある定数，x: エレメント及び変数，dereference($\$x$): $\$x$ に束縛される式，nodename: コンテクストノードの名前．C，S 及び P で始まるラベルはそれぞれコスト，サイズ，確率関数を表す．

for 式の入れ子になる．エンジンの中には後述の eXist のように，正規化しないでナビゲーションを最適化していると思われるものがあり，そのコストを正しく見積もるために，正規化せずに算出している．また，Core は概ね XQuery のサブセットとされているが，実際はコンテクストノードを暗黙のうちに参照する $fs:dot という特殊な変数は XQuery の範囲を逸脱してしまい，実エンジンは一般にはそれを直接入力することができない．ところが Core で単独で登場する child::*QName* というような軸式はその変数にコンテクストノードが束縛されていることを要求する．経路式を Core に変換しない理由はここにもある．direct element constructor については，computed element constructor に正規化可能であるが，タグ名が十分小さい場合，正規化により構文的に長くなってしまうため，本論文では正規化を特に行わない．**where** 節は **if** を用いた **return** 節に正規化できるが，コスト計算に影響はないことと，他の変換規則の証明の補助定理として用いることにより証明の簡略化が可能になるために残している．

経路式の評価コスト

Gottlob 等 [16] によると，経路式のデータ及び問合せ計算複雑性 (入力データの大きさ $|D|$ 及び入力問合せの大きさ $|Q|$ に関する計算複雑性) は何れも多項式時間である．そこで定義されている Core XPath(算術，文字列演算を含まず，ノードの集合のみを相手にする，XPath のサブセット) については計算複雑性は更に線形にまで小さくなる．

筆者等は形式的意味に忠実な Galax(注3) [13] と，別の二次記憶実装である eXist(注4) [14] について，経路式実行時の振舞いを測定した．$|Q|$ の調整には，[16] に倣い問合せ

$$\texttt{//a/b/}\underbrace{\texttt{parent::a/b/}\ldots\texttt{parent::a/b}}_{|Q|-1}$$

を用い，$|D|$ の調整には図 2 に示すデータ構造を用いた．

図 3 は eXist が問合せサイズ $|Q|$ 及びデータサイズ $|D|$ に対してそれぞれ線形に振舞うことを示している．

(注3)：http://www.galaxquery.org/

(注4)：http://exist.sourceforge.net/

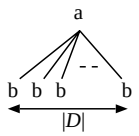

図 2　Gottlob テストに用いたデータ構造
Fig. 2　Data structure used for Gottlob test.

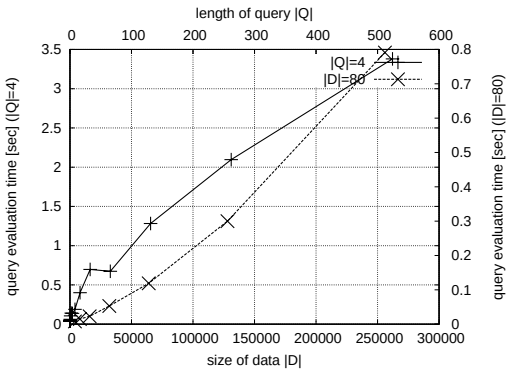

図 3　Gottlob テスト (eXist, $|D| = 80, |Q| = 4$)
Fig. 3　Gottlob test. (eXist, $|D| = 80, |Q| = 4$)

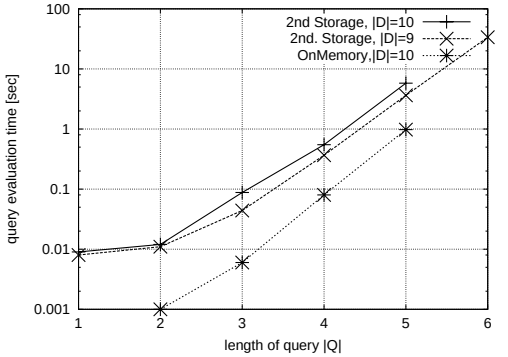

図 4　Gottlob テスト (Galax 0.4.0)
Fig. 4　Gottlob test. (Galax 0.4.0)

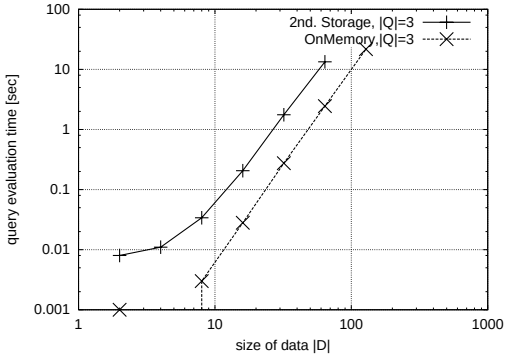

図 5　Gottlob テスト (Galax 0.4.0)
Fig. 5　Gottlob test. (Galax 0.4.0)

Galax の振舞いは問合せの大きさに対して指数的であった (図 4)．Galax の主記憶及び二次記憶実装の何れも，データサイズに対して多項式時間を要した (図 5)．これを受けて，本論文では eXist には (CGSTEP)，Galax には (CCSTEP) を導出した．ここでモデル中の指数 c 及び d は定数である．

エレメント構築コストの推定

仕様ではエレメント構築子は内容全体を複製(注5)することになっている．しかし，エレメント構築コスト自体にかかわる記述は形式的意味の中に含まれておらず，コストは直接推論することができなかった．

直感的にはコストはエレメント構築子の内容の大きさに比例すると考えられる．[17] では，エレメント構築式中のエレメント数に対する Galax の振舞いを測定している．コピーされたエレメント数と経過時間との間に比例関係が見られた．具体的な測定値については [17] を参照されたい．以上により，筆者等はエレメント構築コストとして (CDEC) 及び (CCEC) を導出した．ただし $\mathbf{s}(e//*)$ は e の評価結果の子孫ノードの総数(注6)を表すものとする．

3. コストモデルに基づく各変換対におけるコスト変化の見積り

3.1 コスト変化計算の意義

本章では，最適化とされている書換えアルゴリズムにおいて施される各種等価変換のコスト変化を前章迄で定義したコスト関数を用いて証明する．コストを維持若しくは減少させる等価変換のみからなるプログラム変換は，常にコストを維持もしくは減少させることが帰納的に示されるが，この証明により，そのために選択すべき等価変換が明らかになる．実際に筆者等が開発した最適化アルゴリズム [10] でもこれ等の等価変換を用いている．なお，意味の等価性の証明は付録 **1.** で示す．

3.2 変換規則の表記法

本節で取り上げる規則は意味的等価性が証明済であるため双方向に適用可能であるが，他の規則の適用を可能にする式変形や，コストを減少させるための変換が目的であるため，特に断らない限り等式 $e_1 = e_2$ において変換は左辺から右辺へ向かうものとする．(TFA) 等スコープの変更をもたらす変換については，自由変数の捕捉は名前の付け替えで回避するものとする．前提条件の必要なものは推論規則で表示するが，そうなっていないものは前提なしに成立する規則である．なお，各変換規則には T で始まるタグを付与してコスト関数と区別する．

3.3 変換規則とコスト変化

本節で挙げる変換規則は (TF1)，(TF2)，(TF3)，(TF5) 及び (TIF1) を除いて，全て [10] で用いられているものであり，**4.** で取り上げられている XMark を用いた問合せの最適化の例でも適用されているものである．

[定理 1] (モナド left unit 則 (for 式) [18])　変換

$$\frac{is\_singleton(s)}{\textbf{for } \$x \textbf{ in } s \textbf{ return } e \;=\; \textbf{let } \$x := s \textbf{ return } e} \qquad (\text{TFLU})$$

により，コストは変化しない．ただし，$is\_singleton(s)$ とは，

(注5)：ここでの “deep-copy” の意味は形式的意味における fs:item-sequence-to-node-sequence() に含まれているが，それ自体の意味は “非形式的” 意味 [2] を用いて記述されている．
(注6)：e が属性値等 child 軸を持たないデータの場合は 0 とする．

式 s が静的にシングルトン(注7) と判定できるということであり，**for** 式で束縛された変数参照やエレメント構築式，それ等を **let** 式で束縛した変数参照等が該当する．

(証明)

$$\begin{array}{lll} & \mathtt{c}(\text{左辺}) & \mathtt{c}(\text{右辺}) \\ = & (\textsc{Cfor}) & = \quad (\textsc{Clet}) \\ & \mathtt{c}(s) + \mathtt{s}(s) \cdot \mathtt{c}(e) & \mathtt{c}(s) + \mathtt{c}(e) \end{array}$$

ここで，$is\_singleton(s)$ により $\mathtt{s}(s) = 1$ であるため，両辺のコストは等しくなる． □

[定理 2] (モナド left unit 則 (限量式)) 変換

$$\frac{is\_singleton(s)}{\begin{array}{l}\textbf{some } \$x \textbf{ in } s \textbf{ satisfies } e \\ = \textbf{let } \$x := s \textbf{ return } e\end{array}} \quad (\textsc{Tslu})$$

によりコストは変化しない．

本変換規則は，(Tflu) と同様，論理式に関する繰返し構造を除去するものである．一般に **some** 式は入力列 s の各要素に e を適用した結果の論理和をとるものであるが，要素が一つしかない場合は，それへの e の適用結果がそのまま式全体の結果となる．

(証明)

$$\begin{array}{lll} & \mathtt{c}(\text{左辺}) & \mathtt{c}(\text{左辺}) \\ = & (\textsc{Cnsome}) \text{ による場合} & = \quad (\textsc{Csome}) \text{ による場合} \\ & \mathtt{c}(s) + \mathtt{s}(s) \cdot \mathtt{c}(e) & \mathtt{c}(s) + \frac{1-(1-\mathtt{p}(e))^{\mathtt{s}(s)}}{\mathtt{p}(e)} \cdot \mathtt{c}(e) \end{array}$$

$$\begin{array}{lll} & \mathtt{c}(\text{右辺}) \\ = & (\textsc{Clet}) \\ & \mathtt{c}(s) + \mathtt{c}(e) \end{array}$$

ここで，前提条件により $\mathtt{s}(s) = 1$．よって左辺に式 (Nsome),(Csome) の何れを用いても両辺のコストは一致し，**for** 式と同じ結果を得る．全称型 (**every**)

$$\frac{is\_singleton(s)}{\begin{array}{l}\textbf{every } \$x \textbf{ in } s \textbf{ satisfies } e \\ = \textbf{let } \$x := s \textbf{ return } e\end{array}} \quad (\textsc{Telu})$$

においても同様． □

[定理 3] (モナド right unit 則 (for 式) [18]) 変換

$$\textbf{for } \$x \textbf{ in } s \textbf{ return } \$x = s \quad (\textsc{Tfru})$$

でコストは減少する．

(証明)

$$\begin{array}{lll} & \mathtt{c}(\text{左辺}) & \mathtt{c}(\text{右辺}) \\ = & (\textsc{Cfor}) & = \\ & \mathtt{c}(s) + \mathtt{s}(s) \cdot \mathtt{c}(\$x) & \mathtt{c}(s) \end{array}$$

よって，$\mathtt{c}(\text{左辺}) > \mathtt{c}(\text{右辺})$． □

変数参照のコストを無視するとコスト不変となるが，増加にはならない．

[定理 4] (モナド結合則 (for 式) [18]) 変換

$$\begin{array}{lll} \textbf{for } \$x \textbf{ in for } \$y \textbf{ in } q & & \textbf{for } \$y \textbf{ in } q \\ \qquad \textbf{return } g & = & \textbf{return for } \$x \textbf{ in } g \quad (\textsc{Tfa}) \\ \textbf{return } e & & \qquad \textbf{return } e \end{array}$$

によりコストは変化しない．

本変換規則を含む結合則は，**in** 節にある **for** 構造を除去する (**return** 節の g が残る) ことで，left unit 則 (g がシングルトン式の場合) や分配則 (列式の場合)，結合則 (**for** 式の場合) の適用を可能にする．

(証明)

$$\begin{array}{ll} & \mathtt{c}(\text{左辺}) \\ = & (\textsc{Cfor}) \\ & \mathtt{c}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g) \\ & +\mathtt{s}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g) \cdot \mathtt{c}(e) \\ = & (\textsc{Cfor}),(\textsc{Sfor}) \\ & \mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(g) + \mathtt{s}(g) \cdot \mathtt{s}(q) \cdot \mathtt{c}(e) \end{array}$$

$$\begin{array}{ll} & \mathtt{c}(\text{右辺}) \\ = & (\textsc{Cfor}) \\ & \mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(\textbf{for } \$x \textbf{ in } g \textbf{ return } e) \\ = & (\textsc{Cfor}) \\ & \mathtt{c}(q) + \mathtt{s}(q) \cdot (\mathtt{c}(g) + \mathtt{s}(g) \cdot \mathtt{c}(e)) \\ = & \\ & \mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(g) + \mathtt{s}(g) \cdot \mathtt{s}(q) \cdot \mathtt{c}(e) \end{array}$$

よって両辺のコストは一致． □

[定理 5] (where 節のある for の結合則) 変換

$$\begin{array}{lll} \textbf{for } \$x \textbf{ in for } \$y \textbf{ in } q & & \textbf{for } \$y \textbf{ in } q \\ \qquad \textbf{where } h & & \qquad \textbf{where } h \\ \qquad \textbf{return } g & = & \textbf{return for } \$x \textbf{ in } g \\ \textbf{where } e & & \qquad \textbf{where } e \\ \textbf{return } f & & \qquad \textbf{return } f \end{array} \quad (\textsc{Twfa})$$

により，コストは不変である．

(証明)

$$\begin{array}{ll} & \mathtt{c}(\text{左辺}) \\ = & (\textsc{Cwfor}) \\ & \mathtt{c}(\textbf{for } \$y \textbf{ in } q \textbf{ where } h \textbf{ return } g) \\ & +\mathtt{s}(\textbf{for } \$y \textbf{ in } q \textbf{ where } h \textbf{ return } g) \\ & \cdot(\mathtt{c}(e) + \mathtt{p}(e) \cdot \mathtt{c}(f)) \\ = & (\textsc{Cwfor}),(\textsc{Swfor}) \\ & \mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(h) + \mathtt{s}(q) \cdot \mathtt{p}(h) \cdot \mathtt{c}(g) \\ & +\mathtt{s}(q) \cdot \mathtt{p}(h) \cdot \mathtt{s}(g) \cdot \mathtt{c}(e) \\ & +\mathtt{s}(q) \cdot \mathtt{p}(h) \cdot \mathtt{s}(g) \cdot \mathtt{p}(e) \cdot \mathtt{c}(f) \end{array}$$

$$\mathtt{c}(\text{右辺})$$

(注7)：XQuery では単一のアイテムで構成されている列を言う．

$=$ (Cwfor)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot (\mathsf{c}(h) + \mathsf{p}(h)$

$\cdot \mathsf{c}(\textbf{for } \$x \textbf{ in } g \textbf{ where } e \textbf{ return } f)$

$=$ (Cwfor)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot (\mathsf{c}(h) + \mathsf{p}(h)$

$\cdot(\mathsf{c}(g) + \mathsf{s}(g) \cdot (\mathsf{c}(e) + \mathsf{p}(e) \cdot \mathsf{c}(f))))$

$=$

$\mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(h)$

$+\mathsf{s}(q) \cdot \mathsf{p}(h) \cdot (\mathsf{c}(g) + \mathsf{s}(g) \cdot (\mathsf{c}(e) + \mathsf{p}(e) \cdot \mathsf{c}(f)))$

$=$

$\mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(h) + \mathsf{s}(q) \cdot \mathsf{p}(h) \cdot \mathsf{c}(g)$

$+\mathsf{s}(q) \cdot \mathsf{p}(h) \cdot \mathsf{s}(g) \cdot \mathsf{c}(e)$

$+\mathsf{s}(q) \cdot \mathsf{p}(h) \cdot \mathsf{s}(g) \cdot \mathsf{p}(e) \cdot \mathsf{c}(f)$

よって両辺のコストは一致する. □

[定理 6] (限量式の結合則) 変換

some $\$x$ **in** (**for** $\$y$ **in** q **return** g)
satisfies f

$=$ (Tsa)

some $\$y$ **in** q
satisfies some $\$x$ **in** g
satisfies f

により，コストは不変もしくは減少する．全称型

every $\$x$ **in** (**for** $\$y$ **in** q **return** g)
satisfies f

$=$ (Tea)

every $\$y$ **in** q
satisfies every $\$x$ **in** g
satisfies f

の場合も同様である．

a) コスト (Cnsome) による場合

(証明)

c(左辺)

$=$ (Cnsome)

$\mathsf{c}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g)$

$+\mathsf{s}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g) \cdot \mathsf{c}(f)$

$=$ (Cfor),(Sfor)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(g) + \mathsf{s}(g) \cdot \mathsf{s}(q) \cdot \mathsf{c}(f)$

c(右辺)

$=$ (Cnsome)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot$

$\mathsf{c}(\textbf{some } \$x \textbf{ in } g \textbf{ satisfies } f)$

$=$ (Cnsome)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot (\mathsf{c}(g) + \mathsf{s}(g) \cdot \mathsf{c}(f))$

$=$

$\mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(g) + \mathsf{s}(g) \cdot \mathsf{s}(q) \cdot \mathsf{c}(f)$

よって両辺のコストは一致. □

全称型でも同様の議論が成り立つ.

b) コスト (Csome) による場合

(証明)

c(左辺)

$=$ (Csome)

$\mathsf{c}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g)$

$+\frac{\mathsf{c}(f) \cdot (1-(1-\mathsf{p}(f))^{\mathsf{s}(\textbf{for } \$y \textbf{ in } q \textbf{ return } g)})}{\mathsf{p}(f)}$

$=$ (Cfor),(Sfor)

$\mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(g) + \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(q) \cdot \mathsf{s}(g)}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)$

c(右辺)

$=$ (Csome)

$\mathsf{c}(q) + \frac{\mathsf{c}(\textbf{some } \$x \textbf{ in } g \textbf{ satisfies } f)}{\mathsf{p}(\textbf{some } \$x \textbf{ in } g \textbf{ satisfies } f)}$

$\cdot(1 - (1 - \mathsf{p}(\textbf{some } \$x \textbf{ in } g \textbf{ satisfies } f))^{\mathsf{s}(q)})$

$=$ (Csome),(Psome)

$\mathsf{c}(q) + \frac{\mathsf{c}(g) + \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)}{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}}$

$\cdot(1 - (1 - (1 - (1 - \mathsf{p}(f))^{\mathsf{s}(g)}))^{\mathsf{s}(q)})$

$=$

$\mathsf{c}(q) + \frac{\mathsf{c}(g) + \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)}{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}}$

$\cdot(1 - (1 - \mathsf{p}(f))^{\mathsf{s}(g) \cdot \mathsf{s}(q)})$

$=$

$\mathsf{c}(q) + \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(g) \cdot \mathsf{s}(q)}}{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}} \cdot \mathsf{c}(g) + \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(g) \cdot \mathsf{s}(q)}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)$

但し，**for** 式のコストモデルにより，g のサイズ/コストは一様であり，**some** 式のコストモデルにより f の期待値は一様であると仮定している．ところで，$\mathsf{p}(f)$ を 0 に近づけると，**some** の述語の評価は入力列のより多くの要素に対して行わなければならなくなる．f が真に評価される場合に出会う機会は少なくなり，変換前後のコスト差も 0 に近づく．コストモデルにおいては，差は第 2 項

c(左辺) $-$ c(右辺)

$= \left(\mathsf{s}(q) - \frac{1-(1-\mathsf{p}(f))^{\mathsf{s}(g) \cdot \mathsf{s}(q)}}{1-(1-\mathsf{p}(f))^{\mathsf{s}(g)}}\right) \cdot \mathsf{c}(g)$

のみに現れている．上式において $\mathsf{p}(f)$ を 0 に近づけると，$\mathsf{c}(g)$ の係数のうちの第 2 項は $\mathsf{s}(q)$ に近づく．

$(1-\mathsf{p}(f))^{\mathsf{s}(g)}$ は $\mathsf{s}(g)$ 回 f が偽に評価される確率であり，第 2 項則ち

$$\left(\sum_{k=0}^{\mathsf{s}(q)-1} ((1-\mathsf{p}(f))^{\mathsf{s}(g)})^k\right) \cdot \mathsf{c}(g)$$

は，内側の **some** の評価は，$\$y$ の各束縛について前回の f の評価が全ての g の要素に関して偽に評価された場合に次の $\$y$ の束縛について再び評価されることを表現している．更に，$\mathsf{c}(g)$ の係数のこの第二項は常に $\mathsf{s}(q)$ より小さいため，変換によりコストは必ず改善されることが分かる. □

[定理 7] (上流の for に where 節のある限量式の結合則) 変換

some $\$y$ **in** (**for** $\$z$ **in** q
where h
return g)
satisfies f

$=$ (TWSA)

some $\$z$ **in** q
satisfies (h **and** (**some** $\$y$ **in** g
satisfies f))

により，コストは減少する(注8)．

(証明) $p \stackrel{\text{def}}{=} h$ **and** (**some** $\$y$ **in** g **satisfies** f)，$e \stackrel{\text{def}}{=}$**for** $\$z$ **in** q **where** h **return** g とおくと，

$\mathtt{p}(p)$
$=$ (PAND)
$\mathtt{p}(h) \cdot \mathtt{p}(\textbf{some } \$y \textbf{ in } g \textbf{ satisfies } f)$
$=$ (PSOME)
$\mathtt{p}(h) \cdot (1 - (1 - \mathtt{p}(f))^{\mathtt{s}(g)})$

$\mathtt{c}$(左辺)
$=$ (CSOME)
$\mathtt{c}(e)$
$+\mathtt{c}(f) \cdot \frac{1-(1-\mathtt{p}(f))^{\mathtt{s}(e)}}{\mathtt{p}(f)}$
$=$ (CWFOR),(SWFOR)
$\mathtt{c}(q) + \mathtt{s}(q) \cdot (\mathtt{c}(h) + \mathtt{p}(h) \cdot \mathtt{c}(g))$
$+\mathtt{c}(f) \cdot \frac{1-(1-\mathtt{p}(f))^{\mathtt{s}(q)\cdot\mathtt{p}(h)\cdot\mathtt{s}(g)}}{\mathtt{p}(f)}$
$=$
$\mathtt{c}(q) + \mathtt{s}(q) \cdot (\mathtt{c}(h) + \mathtt{p}(h) \cdot \mathtt{c}(g))$
$+\frac{\mathtt{c}(f)}{\mathtt{p}(f)} \cdot (1 - (1 - \mathtt{p}(f))^{\mathtt{s}(q)\cdot\mathtt{p}(h)\cdot\mathtt{s}(g)})$
$=$
$\mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(h) + \mathtt{p}(h) \cdot \mathtt{s}(q) \cdot \mathtt{c}(g)$
$+\frac{1-(1-\mathtt{p}(f))^{\mathtt{s}(q)\cdot\mathtt{p}(h)\cdot\mathtt{s}(g)}}{\mathtt{p}(f)} \cdot \mathtt{c}(f)$

$\mathtt{c}$(右辺)
$=$ (CSOME)
$\mathtt{c}(q) + \frac{\mathtt{c}(p)}{\mathtt{p}(p)} \cdot (1 - (1 - \mathtt{p}(p))^{\mathtt{s}(q)})$
$=$ (CAND)
$\mathtt{c}(q) + \frac{\mathtt{c}(h)+\mathtt{p}(h)\cdot\mathtt{c}(\,{}^{\textbf{some } \$y \textbf{ in } g}_{\textbf{satisfies } f}\,)}{\mathtt{p}(p)} \cdot$
$(1 - (1 - \mathtt{p}(p))^{\mathtt{s}(q)})$
$=$ (CSOME),(PAND)
$\mathtt{c}(q) + \frac{\mathtt{c}(h)+\mathtt{p}(h)\cdot(\mathtt{c}(g)+\frac{\mathtt{c}(f)}{\mathtt{p}(f)}\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})}$
$\cdot(1 - (1 - \mathtt{p}(h) \cdot (1 - (1 - \mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)})$
$=$
$\mathtt{c}(q)$
$+(\mathtt{c}(h) + \mathtt{p}(h) \cdot \mathtt{c}(g) + \frac{\mathtt{p}(h)\cdot\mathtt{c}(f)}{\mathtt{p}(f)} \cdot (1 - (1 - \mathtt{p}(f))^{\mathtt{s}(g)}))$
$\cdot\frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})}$
$=$
$\mathtt{c}(q) + \frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})} \cdot \mathtt{c}(h)$
$+\frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}} \cdot \mathtt{c}(g)$
$+\frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(f)} \cdot \mathtt{c}(f)$

$\therefore \mathtt{c}$(左辺) $> \mathtt{c}$(右辺) つまりこの変換は常にコストを減少させる． □

(注8)：本変換の意味的正当性は付録 **1.** に掲載している規則から容易に導くことができる．

ここでのコスト減少は直感的には以下のように説明できる．左辺では，**some** 式は **in** 節の全ての要素を走査せずに結果を確定できる場合があるのに対し，**for** 式の評価は入力列 q 全てに対して必要である．これはモデルが正格評価戦略に基づいていることによる．遅延評価戦略を仮定するならば両辺のコストは一致すると考えられる．つまり，右辺は遅延評価を (部分的に) 模倣しているとも言える．したがって，本コストモデルは本規則のような変換の適用により遅延評価を模倣することができる．

[定理 8] (分配則 (for 式) [18]) 変換

for $\$x$ **in** (q, q') **return** f $=$ (**for** $\$x$ **in** q **return** f), (**for** $\$x$ **in** q' **return** f) (TFD)

によりコストは不変である．

本変換規則を含む分配則は，**in** 節にある列式を除去する．

(証明)

$\mathtt{c}$(左辺)
$=$ (CFOR)
$\mathtt{c}(q,\ q') + \mathtt{s}(q,\ q') \cdot \mathtt{c}(f)$
$=$ (CSEQ),(SSEQ)
$\mathtt{c}(q) + \mathtt{c}(q') + (\mathtt{s}(q) + \mathtt{s}(q')) \cdot \mathtt{c}(f)$
$=$
$\mathtt{c}(q) + \mathtt{c}(q') + \mathtt{s}(q) \cdot \mathtt{c}(f) + \mathtt{s}(q') \cdot \mathtt{c}(f)$

$\mathtt{c}$(右辺)
$=$ (CSEQ)
$\mathtt{c}(\textbf{for } \$x \textbf{ in } q \textbf{ return } f)$
$+\mathtt{c}(\textbf{for } \$x \textbf{ in } q' \textbf{ return } f)$
$=$ (CFOR)
$\mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(f) + \mathtt{c}(q') + \mathtt{s}(q') \cdot \mathtt{c}(f)$

よって，$\mathtt{c}$(左辺) $= \mathtt{c}$(右辺) □

[定理 9] (分配則 (限量式)) 変換

some $\$x$ **in** (q, q') **satisfies** f $=$ (**some** $\$x$ **in** q **satisfies** f) **or** (**some** $\$x$ **in** q' **satisfies** f) (TSD)

は コ ス ト を 保 存 ((CNSOME) の場合) あ る い は 削 減 ((CSOME) の場合) する．

(証明) (CNSOME) の場合

$\mathtt{c}$(左辺)
$=$ (CNSOME)
$\mathtt{c}(q,\ q') + \mathtt{s}(q,\ q') \cdot \mathtt{c}(f)$
$=$ (CSEQ),(SSEQ)
$\mathtt{c}(q) + \mathtt{c}(q') + (\mathtt{s}(q) + \mathtt{s}(q')) \cdot \mathtt{c}(f)$
$=$
$\mathtt{c}(q) + \mathtt{c}(q') + \mathtt{s}(q) \cdot \mathtt{c}(f) + \mathtt{s}(q') \cdot \mathtt{c}(f)$

$$
\begin{aligned}
& \mathsf{c}(\text{右辺}) \\
= \quad & (\textsc{Cor}) \\
& \mathsf{c}(\textbf{some } \$x \textbf{ in } q \textbf{ satisfies } f) \\
& +\mathsf{c}(\textbf{some } \$x \textbf{ in } q' \textbf{ satisfies } f) \\
= \quad & (\textsc{Cnsome}) \\
& \mathsf{c}(q) + \mathsf{s}(q) \cdot \mathsf{c}(f) + \mathsf{c}(q') + \mathsf{s}(q') \cdot \mathsf{c}(f)
\end{aligned}
$$

よって両辺のコストは一致する． □

全称型

$$
\begin{array}{l}
\textbf{every } \$x \textbf{ in } (q,\ q') \\
\textbf{satisfies } f
\end{array}
=
\begin{array}{l}
(\textbf{every } \$x \textbf{ in } q \textbf{ satisfies } f) \\
\textbf{and } (\textbf{every } \$x \textbf{ in } q' \\
\qquad\qquad \textbf{satisfies } f)
\end{array}
\qquad (\textsc{Ted})
$$

の場合も同様．

(証明) (Csome) の場合

$$
\begin{aligned}
& \mathsf{c}(\text{左辺}) \\
= \quad & (\textsc{Csome}) \\
& \mathsf{c}((q,q')) + \tfrac{1-(1-\mathsf{p}(f))^{\mathsf{s}((q,q'))}}{\mathsf{p}(f)} \cdot \mathsf{c}(f) \\
= \quad & (\textsc{Cseq}),(\textsc{Sseq}) \\
& \mathsf{c}(q) + \mathsf{c}(q') + \tfrac{1-(1-\mathsf{p}(f))^{(\mathsf{s}(q)+\mathsf{s}(q'))}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{c}(\text{右辺}) \\
= \quad & (\textsc{Cor}),(\textsc{Csome}) \\
& \mathsf{c}(q) + \tfrac{1-(1-\mathsf{p}(f))^{\mathsf{s}(q)}}{\mathsf{p}(f)} \cdot \mathsf{c}(f) \\
& +(1-\mathsf{p}(\textbf{some } \$y \textbf{ in } q \textbf{ satisfies } f)) \cdot \\
& (\mathsf{c}(q') + \tfrac{1-(1-\mathsf{p}(f))^{\mathsf{s}(q')}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)) \\
= \quad & (\textsc{Psome}) \\
& \mathsf{c}(q) + (1-\mathsf{p}(f))^{\mathsf{s}(q)} \cdot \mathsf{c}(q') \\
& +\tfrac{1-(1-\mathsf{p}(f))^{(\mathsf{s}(q)+\mathsf{s}(q'))}}{\mathsf{p}(f)} \cdot \mathsf{c}(f)
\end{aligned}
$$

ここで，両辺で異なるのは $\mathsf{c}(q')$ の係数であるが，$0 \leqq (1-\mathsf{p}(f)) \leqq 1$ により，右辺では必ず 1 以下になる．よって変換により q' が評価される確率が減少し，コストが減少する．この傾向は $\mathsf{p}(f)$ や $\mathsf{s}(q)$ が大きい程顕著である． □

[定理 10] (child 軸の畳み込み) 変換

$$
\langle QName\rangle\{e\}\langle/QName\rangle/\textbf{child}::* = e \qquad (\textsc{Tc})
$$

により，コストは減少する．

意味的等価性は，左辺における child 軸が direct エレメント構築子の内容を出力することにより示される．step 式は重複するノードの除去と文書順による整順をも意味するものの，エレメント構築子がコピーを行うことにより，e の中で重複や捻れは起らない．よって右辺は単に e で表現される．

(証明)

$$
\begin{aligned}
& \mathsf{c}(\text{左辺}) \\
= \quad & (\textsc{Ccstep}) \\
& \mathsf{c}(\langle QName\rangle\{e\}\langle/QName\rangle) + \\
& k' \times \mathsf{s}(\langle QName\rangle\{e\}\langle/QName\rangle)^c \\
= \quad & (\textsc{Cdec}),(\textsc{Sdec}) \\
& \mathsf{c}(e) + (1 + \mathsf{s}(e) + \mathsf{s}(e//*)) \times C_e + k \times 1^c \\
= \quad & \\
& \mathsf{c}(e) + (1 + \mathsf{s}(e) + \mathsf{s}(e//*)) \times C_e + k
\end{aligned}
$$

$\mathsf{c}(\text{右辺}) = \mathsf{c}(e)$

よって，$\mathsf{c}(\text{左辺}) > \mathsf{c}(\text{右辺})$ □

ただし，ノード ID に基づく等価性はこの変換で損なわれる．変換におけるノード ID に基づく等価性の保持については，[19] で扱っている．ノード ID ではなく値に基づく等価性に依拠した書換えの研究も数多く存在するため (例えば [1], [20])，このような等価性の考慮も有意義と考える．

フィルタリング

例えば，変換

$$
\begin{aligned}
& (\langle qn_1\rangle\{e_1\}\langle/qn_1\rangle, \langle qn_2\rangle\{e_2\}\langle/qn_2\rangle)[\textbf{self}::qn_1] \\
=& \langle qn_1\rangle\{e_1\}\langle/qn_1\rangle
\end{aligned}
$$

はフィルタリングの静的評価であるが，このような変換では明らかに評価する式が単純化されており，コストの減少がもたらされる． □

β 簡約 (let 束縛の展開)

$$
\frac{\$x \text{ が } e_2 \text{ 中で一度しか出現しない}}{(\textbf{let } \$x := e_1 \textbf{ return } e_2) = e_2[e_1/\$x]} \qquad (\textsc{Tlet})
$$

右辺は，e_2 中に出現する $\$x$ を e_1 で置き換えたものを表す．

e_2 の中に $\$x$ が 2ヵ所以上存在すると，その数だけ評価が行われてしまうため，コストが増加してしまい，共通部分式削除の最適化に逆行する．また，e_1 の中にエレメント構築処理が含まれていると，その処理自体が複製されてしまうため，意味が変ってしまう．出現が 1 個所だけであればこの変換でコストは増加しない． □

[定理 11] (選択の融合 (filter/1) (注9)) 変換

$$
\begin{array}{l}
\textbf{for } \$x \textbf{ in for } \$y \textbf{ in } q \\
\qquad \textbf{where } g \qquad\quad \textbf{for } \$x \textbf{ in } q \\
\qquad \textbf{return } \$y \quad = \quad \textbf{where } g \textbf{ and } f \\
\textbf{where } f \qquad\qquad\quad \textbf{return } \$x \\
\textbf{return } \$x
\end{array}
\qquad (\textsc{Tf1})
$$

により，コストは増加しない．

(証明)

$$
\begin{aligned}
& \mathsf{c}(\text{左辺}) \\
= \quad & (\textsc{Cwfor}),(\textsc{Cvr}) \\
& \mathsf{c}(q) + \mathsf{s}(q) \cdot (C_{vr} + \mathsf{c}(g) + \mathsf{p}(g) \cdot \mathsf{c}(\$y) \\
& +\mathsf{s}(\textbf{for } \$y \textbf{ in } q \textbf{ where } g \textbf{ return } \$y)
\end{aligned}
$$

(注9)：ここでの付番は，本研究の契機のひとつとなった [21] に基づく．筆者等も XQuery で同様の変換規則を見い出したが，データモデルの相違のため，全ての規則が XQuery にも当てはまる訳ではなかった．例えば，XQuery の列は非可換な列連結演算子で構築されるため，集合和の演算子に基づく setmap/4 相当の変換規則は存在しない．

$$
\begin{aligned}
&\cdot(\mathtt{c}(f)+\mathtt{p}(f)\cdot\mathtt{c}(\$x))\\
=\quad&\text{(SWFOR),(CVR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(C_{vr}+\mathtt{c}(g))+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot C_{vr}\\
&+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{s}(\$y)\cdot(\mathtt{c}(f)+\mathtt{p}(f)\cdot C_{vr})\\
=\quad&\text{(SVR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot C_{vr}+\mathtt{s}(q)\cdot\mathtt{c}(g)+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot C_{vr}\\
&+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{c}(f)\\
&+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{p}(f)\cdot C_{vr}
\end{aligned}
$$

$$
\begin{aligned}
&\mathtt{c}(\text{右辺})\\
=\quad&\text{(CWFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(g\ \textbf{and}\ f)\\
&+\mathtt{p}(g\ \textbf{and}\ f)\cdot\mathtt{c}(\$x))\\
=\quad&\text{(PAND),(PAND),(CVR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot((C_{vr}+\mathtt{c}(g)+\mathtt{p}(g)\\
&\cdot(C_{vr}+\mathtt{c}(f)))+\mathtt{p}(g)\cdot\mathtt{p}(f)\cdot C_{vr})\\
=\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot C_{vr}+\mathtt{s}(q)\cdot\mathtt{c}(g)+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot C_{vr}\\
&+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{c}(f)+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{p}(f)\cdot C_{vr}
\end{aligned}
$$

よって，両辺のコストは一致. □

ここでは **return** 式が変数参照だけで構成されているため，変数参照のコストが考慮されている．しかし，このコストを無視してもコストの等価性には影響しない．したがって，以降では再び変数参照のコストを省略する．

[定理 12] (選択の移動 (filter/2)) 変換

$$
\begin{array}{l}
\textbf{for}\ \$x\ \textbf{in for}\ \$y\ \textbf{in}\ q\\
\qquad\textbf{where}\ g\\
\qquad\textbf{return}\ \$y\\
\textbf{return}\ f
\end{array}
=
\begin{array}{l}
\textbf{for}\ \$x\ \textbf{in}\ q\\
\textbf{where}\ g\\
\textbf{return}\ f
\end{array}
\qquad\text{(TF2)}
$$

により，コストは変化しない.

(証明)

$$
\begin{aligned}
&\mathtt{c}(\text{左辺})\\
=\quad&\text{(CWFOR),(CFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(g)+\mathtt{p}(g)\cdot\mathtt{c}(\$y))\\
&+\mathtt{s}(\textbf{for}\ \$y\ \textbf{in}\ q\ \textbf{where}\ g\ \textbf{return}\ \$y)\cdot\mathtt{c}(f)\\
=\quad&\text{(SWFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(g)+\mathtt{p}(g))+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{s}(\$y)\cdot\mathtt{c}(f)\\
=\quad&\text{(SVR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot\mathtt{c}(g)+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{c}(f)
\end{aligned}
$$

$$
\begin{aligned}
&\mathtt{c}(\text{右辺})\\
=\quad&\text{(CWFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(g)+\mathtt{p}(g)\cdot\mathtt{c}(f))\\
=\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot\mathtt{c}(g)+\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{c}(f)
\end{aligned}
$$

したがって，変換によりコストは変化しない. □

変数参照コストを含めるとコストは減少となる.

[定理 13] (filter/3) 変換

$$
\begin{array}{l}
\textbf{for}\ \$x\ \textbf{in for}\ \$y\ \textbf{in}\ q\\
\qquad\textbf{return}\ f\\
\textbf{where}\ g\\
\textbf{return}\ \$x
\end{array}
=
\begin{array}{l}
\textbf{for}\ \$y\ \textbf{in}\ q\\
\textbf{return}\\
\qquad\textbf{for}\ \$x\ \textbf{in}\ f\\
\qquad\textbf{where}\ g\\
\qquad\textbf{return}\ \$x
\end{array}
\qquad\text{(TF3)}
$$

により，コストは不変である.

(証明)

$$
\begin{aligned}
&\mathtt{c}(\text{左辺})\\
=\quad&\text{(CWFOR),(CFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot\mathtt{c}(f)\\
&+\mathtt{s}(\textbf{for}\ \$y\ \textbf{in}\ q\ \textbf{return}\ f))\cdot\mathtt{c}(g)\\
=\quad&\text{(SFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot\mathtt{c}(f)+\mathtt{s}(q)\cdot\mathtt{s}(f)\cdot\mathtt{c}(g)
\end{aligned}
$$

$$
\begin{aligned}
&\mathtt{c}(\text{右辺})\\
=\quad&\text{(CFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\\
&\cdot\mathtt{c}(\textbf{for}\ \$y\ \textbf{in}\ f\ \textbf{where}\ g\ \textbf{return}\ \$y)\\
=\quad&\text{(CWFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\\
&\cdot(\mathtt{c}(f)+\mathtt{s}(f)\cdot(\mathtt{c}(g)+\mathtt{p}(g)\cdot\mathtt{c}(\$y)))\\
=\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(\$x)+\mathtt{c}(f))\\
&+\mathtt{s}(f)\cdot(\mathtt{c}(\$y)+\mathtt{c}(g))\\
=\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot\mathtt{c}(f)+\mathtt{s}(q)\cdot\mathtt{s}(f)\cdot\mathtt{c}(g)
\end{aligned}
$$

よって，c(左辺) = c(右辺) □

変数参照コストを考慮しても，上記等価性は変らない.

[定理 14] (選択の入れ替え (filter/5)) 変換

$$
\begin{array}{l}
\textbf{for}\ \$x\ \textbf{in for}\ \$y\ \textbf{in}\ q\\
\qquad\textbf{where}\ g\\
\qquad\textbf{return}\ \$y\\
\textbf{where}\ f\\
\textbf{return}\ \$x
\end{array}
=
\begin{array}{l}
\textbf{for}\ \$x\ \textbf{in for}\ \$y\ \textbf{in}\ q\\
\qquad\textbf{where}\ f\\
\qquad\textbf{return}\ \$y\\
\textbf{where}\ g\\
\textbf{return}\ \$x
\end{array}
\qquad\text{(TF5)}
$$

のコスト変化は，述語のコストを使って

$\mathtt{c}(\text{左辺})-\mathtt{c}(\text{右辺}) = \mathtt{s}(q)\cdot((1-\mathtt{p}(f))\cdot\mathtt{c}(g)-(1-\mathtt{p}(g))\cdot\mathtt{c}(f))$

と表される.

(証明)

$$
\begin{aligned}
&\mathtt{c}(\text{左辺})\\
=\quad&\text{(CWFOR),(SWFOR)}\\
&\mathtt{c}(q)+\mathtt{s}(q)\cdot(\mathtt{c}(g)+\mathtt{p}(g)\cdot\mathtt{c}(\$y))\\
&+(\mathtt{s}(q)\cdot\mathtt{p}(g)\cdot\mathtt{s}(\$y))\cdot(\mathtt{c}(f)+\mathtt{p}(f)\cdot\mathtt{c}(\$x))
\end{aligned}
$$

$=$ (Svr)

$\mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{c}(g) + \mathtt{s}(q) \cdot \mathtt{p}(g) \cdot \mathtt{c}(f)$

$=$

$\mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{p}(g) \cdot \mathtt{c}(f) + \mathtt{s}(q) \cdot \mathtt{c}(g)$

$\mathtt{c}$(右辺)

$=$ 左辺にて f と g を入れ替える

$\mathtt{c}(q) + \mathtt{s}(q) \cdot \mathtt{p}(f) \cdot \mathtt{c}(g) + \mathtt{s}(q) \cdot \mathtt{c}(f)$

$\mathtt{c}$(左辺)$-\mathtt{c}$(右辺)$= \mathtt{s}(q) \cdot ((1-\mathtt{p}(f)) \cdot \mathtt{c}(g) - (1-\mathtt{p}(g)) \cdot \mathtt{c}(f))$

変数参照コストを算入してもコスト変化計算には影響しないことに注意.

よって，$\mathtt{p}(f)$ が小さく $\mathtt{c}(g)$ が大きい程また $\mathtt{p}(g)$ が大きく $\mathtt{c}(f)$ が小さい程，変換によりコストは小さくなる. □

[系 15] $\mathtt{c}(g) = \mathtt{c}(f)$ のとき，$\mathtt{p}(g) > \mathtt{p}(f)$ ならば，変換によりコストは減少する．つまり，絞り込みが強いフィルタを上流に配する方が得である.

(証明)

$$\begin{aligned} & \mathtt{c}(\text{左辺}) - \mathtt{c}(\text{右辺}) \\ = \;& \mathtt{s}(q) \cdot \{(1 - \mathtt{p}(f) - 1 + \mathtt{p}(g))\} \cdot \mathtt{c}(f) \\ = \;& \mathtt{s}(q) \cdot (\mathtt{p}(g) - \mathtt{p}(f)) \cdot \mathtt{c}(f) \\ > \;& 0 \end{aligned}$$

□

[系 16] $\mathtt{p}(g) = \mathtt{p}(f)$ のとき，$\mathtt{c}(g) > \mathtt{c}(f)$ ならば，変換によりコストは減少する．つまり，コストの低いフィルタを上流に配する方が得である.

(証明)

$\mathtt{c}(\text{左辺}) - \mathtt{c}(\text{右辺}) = \mathtt{s}(q) \cdot (1 - \mathtt{p}(f))(\mathtt{c}(g) - \mathtt{c}(f))$ □

[定理 17] (if/1) 変換

$$\begin{array}{lcl} \textbf{if } (e_1) & & \\ \textbf{then if } (e_2) & & \textbf{if } (e_1 \textbf{ and } e_2) \\ \quad \textbf{then } e_3 & = & \textbf{then } e_3 \\ \quad \textbf{else } e_4 & & \textbf{else } e_4 \\ \textbf{else } e_4 & & \end{array} \qquad \text{(Tif1)}$$

により，コストは変化しない.

(証明)

$\mathtt{c}$(左辺)

$=$ (Cif)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(\textbf{if } (e_2) \textbf{ then } e_3 \textbf{ else } e_4)$

$+(1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(e_4)$

$=$ (Cif)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot (\mathtt{c}(e_2) + \mathtt{p}(e_2) \cdot \mathtt{c}(e_3)$

$+(1 - \mathtt{p}(e_2)) \cdot \mathtt{c}(e_4)) + (1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(e_4)$

$=$

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(e_2) + \mathtt{p}(e_1) \cdot \mathtt{p}(e_2) \cdot \mathtt{c}(e_3)$

$+\mathtt{p}(e_1) \cdot (1 - \mathtt{p}(e_2)) \cdot \mathtt{c}(e_4) + (1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(e_4)$

$=$

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(e_2) + \mathtt{p}(e_1) \cdot \mathtt{p}(e_2) \cdot \mathtt{c}(e_3)$

$+(1 - \mathtt{p}(e_1) \cdot \mathtt{p}(e_2)) \cdot \mathtt{c}(e_4)$

$\mathtt{c}$(右辺)

$=$ (Cif)

$\mathtt{c}(e_1 \textbf{ and } e_2) + \mathtt{p}(e_1 \textbf{ and } e_2) \cdot \mathtt{c}(e_3)$

$+(1 - \mathtt{p}(e_1 \textbf{ and } e_2)) \cdot \mathtt{c}(e_4)$

$=$ (Cand),(Pand)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(e_2) + \mathtt{p}(e_1) \cdot \mathtt{p}(e_2) \cdot \mathtt{c}(e_3)$

$+(1 - \mathtt{p}(e_1) \cdot \mathtt{p}(e_2)) \cdot \mathtt{c}(e_4)$

したがって，$\mathtt{c}$(左辺)$= \mathtt{c}$(右辺) □

[定理 18] (if/4) 変換

$$\begin{array}{lcl} & & f(\textbf{if } (e_1) \textbf{ then } e_2 \textbf{ else } e_3) \\ = & & \textbf{if } (e_1) \textbf{ then } f(e_2) \textbf{ else } f(e_3) \end{array} \qquad \text{(Tif4)}$$

により，コストは変化しない.

(証明)

$\mathtt{c}$(左辺)

$=$ (Cfcall)

$\mathtt{c}(\textbf{if } (e_1) \textbf{ then } e_2 \textbf{ else } e_3) + \mathtt{c}(f)$

$=$ (Cif)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(e_2) + (1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(e_3) + \mathtt{c}(f)$

$\mathtt{c}$(右辺)

$=$ (Cif)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(f(e_2)) + (1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(f(e_3))$

$=$ (Cfcall)

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot (\mathtt{c}(e_2) + \mathtt{c}(f))$

$+(1 - \mathtt{p}(e_1)) \cdot (\mathtt{c}(e_3) + \mathtt{c}(f))$

$=$

$\mathtt{c}(e_1) + \mathtt{p}(e_1) \cdot \mathtt{c}(e_2) + (1 - \mathtt{p}(e_1)) \cdot \mathtt{c}(e_3) + \mathtt{c}(f)$

したがって，$\mathtt{c}$(左辺)$= \mathtt{c}$(右辺) □

3.4 コストモデルを用いた書換えの選択

筆者等が実際に変換規則 (Twsa) を Galax 0.3.5 及び eXist snapshot-20041119 (注10) に適用したところ，定理 7 にもかかわらず速度低下が観察された．具体的には，以下の問合せに関して，適用前:

(注10)：http://prdownloads.sourceforge.net/exist/eXist-snapshot-20041119.jar

```
for $a in distinct-values(doc('vl.xml')/bib/book/author)
where some $t in (for $b in doc('vl.xml')/bib/book
                  where some $ba in fn:data($b/author)
                        satisfies deep-equal($ba,$a)
                  return $b/title)
      satisfies contains(string($t),"abcdef")
return $a
```

と適用後:

```
for $a in distinct-values(doc('vl.xml')/bib/book/author)
where some $b in doc('vl.xml')/bib/book
      satisfies
             ((some $ba in fn:data($b/author)
               satisfies deep-equal($ba,$a))
              and
             (some $t in b/title
              satisfies contains(string($t),"abcdef")))
return $a
```

で，Galax で 9 割増，eXist で 2 割増 (入力文書サイズ約 23KB) となった．これはエンジンが意図しない振舞いをしているということであり，またこの変換は避けるべきであることを示している．更なる測定の結果，速度低下は，**and** 式の被演算数が常に両方評価されるという両実装における評価戦略に起因することが分かった (Galax 0.4.0 では，右被演算数は左被演算数が偽に評価された場合は評価されない)．これは，コストモデルが，高機能のエンジンであっても変換規則適用に際して何等かの陥穽があることを示唆した例である．なお，上記 **and** 評価戦略の違いは，次の問合せにより簡単に確認する事ができる．

$$\begin{aligned}&\mathrm{data}(\mathrm{doc}(\text{"andtest.xml"})/\mathrm{num}) = 1 \textbf{ and}\\ &\quad(\langle\mathrm{x}\rangle\{N,\ N\}\langle/\mathrm{x}\rangle = (\textbf{for } \$\mathrm{x} \textbf{ in } (1 \textbf{ to } N)\\ &\qquad\qquad\qquad\textbf{for } \$\mathrm{y} \textbf{ in } (1 \textbf{ to } N)\\ &\qquad\qquad\qquad\textbf{return } \langle\mathrm{x}\rangle\{\$\mathrm{x},\$\mathrm{y}\}\langle/\mathrm{x}\rangle))\end{aligned}$$

XML 文書 `andtest.xml` の内容を，`<num>0</num>` に設定することにより，**and** 式の左被演算数 data (doc("andtest.xml")/num) = 1 は実行時に偽となり，short-circuit 戦略をとっている場合，この問合せの実行は直ちに終了する．右被演算数に，計算に時間のかかる式を記述しているため，処理系に応じて N の値を大きくすることにより，short-circuit 戦略をとっていない場合との区別を判断するための，実行時間の差を十分多くとることができる．後述の Saxon-B，Qizx/open，eXist-1.1.1-newcore に関しても，同様に short-circuit 評価戦略をとっていることが分かった．

実際以下のように，**and** 式のコストを，両被演算数のコストの単純和 ($\mathtt{c}(e_1 \textbf{ and } e_2) = \mathtt{c}(e_1) + \mathtt{c}(e_2)$) とすることで両実装の評価戦略をコストモデルに取り込む事により，上記速度低下をモデル化することができる．

$p \stackrel{\mathrm{def}}{=} h \textbf{ and } (\textbf{some } \$y \textbf{ in } g \textbf{ satisfies } f)$，に対し，コストは

$$\begin{aligned}&\mathtt{c}(p)\\ =\ &\mathtt{c}(h) + \mathtt{c}(\textbf{some } \$y \textbf{ in } g \textbf{ satisfies } f)\\ =\ &\mathtt{c}(h) + \mathtt{c}(g) + \frac{1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}}{\mathtt{p}(f)}\mathtt{c}(f)\end{aligned}$$

となり，

$$\begin{aligned}&\mathtt{c}(\text{右辺})\\ =\ &\quad(\textsc{Csome})\\ &\mathtt{c}(q) + \frac{1-(1-\mathtt{p}(p))^{\mathtt{s}(q)}}{\mathtt{p}(p)}\cdot\mathtt{c}(p)\\ =\ &\quad\text{上記 } p \text{ の定義及び } (\textsc{Pand})\\ &\mathtt{c}(q) + \frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})}\\ &\cdot(\mathtt{c}(h) + \mathtt{c}(g) + \frac{1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}}{\mathtt{p}(f)}\mathtt{c}(f))\\ =\ &\\ &\mathtt{c}(q) + \frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})}\cdot\mathtt{c}(h)\\ &+\frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)})}\cdot\mathtt{c}(g)\\ &+\frac{1-(1-\mathtt{p}(h)\cdot(1-(1-\mathtt{p}(f))^{\mathtt{s}(g)}))^{\mathtt{s}(q)}}{\mathtt{p}(h)\cdot\mathtt{p}(f)}\cdot\mathtt{c}(f)\end{aligned}$$

$\mathtt{c}(g)$ に係るコストに **where** 節による選択が効かなくなる分，右辺が重くなることが分かる．さらに $\mathtt{p}(h)$ の値によっては，左辺よりコストが大きくなることも分かる [17].

and 式で両オペランドを評価してしまう処理系用には，**if** 式で以下の (Tsaif) のように規則を表現することにより，このような速度低下を食い止めることができ，意味的に **and** 版と同等であることから，**and** 式の評価戦略によらない結合則となる．

$$\begin{aligned}&\textbf{some } \$y \textbf{ in } (\textbf{for } \$z \textbf{ in } q\\ &\qquad\textbf{where } h\\ &\qquad\textbf{return } g)\\ &\textbf{satisfies } f\end{aligned} \quad = \quad \begin{aligned}&\textbf{some } \$z \textbf{ in } q\\ &\textbf{satisfies if } (h)\\ &\qquad\textbf{then some } \$y \textbf{ in } g\\ &\qquad\qquad\textbf{satisfies } f\\ &\qquad\textbf{else } \mathrm{fn{:}false}()\end{aligned} \qquad (\textsc{Tsaif})$$

3.5 ま と め

本章では，提案するコストモデルに基づき様々な変換規則のコスト変化を証明した．これ等の証明により，上記の変換のうちコストが増加しないもののみで構成される最適化は，全体としてコストは必ず変らないか減少すると結論づけることができる．

4. コストモデルの評価

本章では，ベンチマーク問合せを用いた実エンジンによる実験結果と，本コストモデルによる計算結果を比較することで提案するコストモデルを評価する．ベンチマークには大別してアプリケーションベンチマークとマイクロベンチマークの二種類がある．アプリケーションベンチマークは多くの構文要素から構成される現実的な問合せを用いたものであり，マイクロベンチマークはある構文要素の特徴を計測するための非現実的な問合せを用いたものである．本章ではまず **4. 1** で，XQuery の特徴的な構文要素である **FLWOR** 式とエレメント構築子に関するマイクロベンチマーク問合せ [22] について，4 種類の実エンジンを用いた実験結果と提案するコストモデルの計算結果の比較による評価を示す．次に **4. 2** では，アプリケーションベンチマークとして，エレメント構築子を含む合成問合せで，特に最適化の効果が高いと予想されるものを意図的に選んでその実験結果とコストモデルによる計算結果の比較をする．その結果当初の予想に反する結果が出たので，可能な考察を行う．

尚，アプリケーションベンチマークでは一つの問合せが多くの構文要素から構成されるので，4.2 節の計算例で見られるようにコスト計算が複雑になるばかりでなく，モデルとの乖離の要因の切り分けが困難となるため，敢えて本コストモデルから

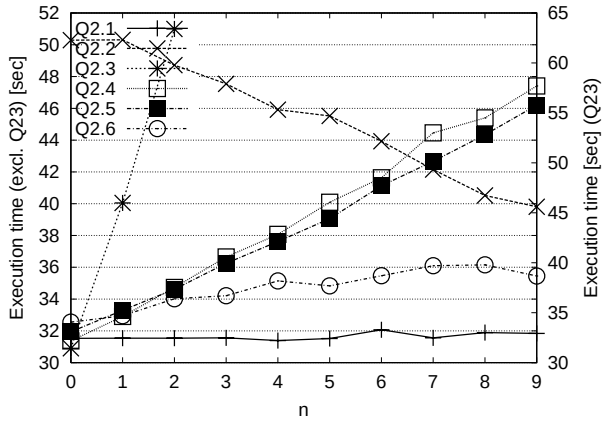

図 6　マイクロベンチマーク (Galax 0.4.0 主記憶実装)
Fig. 6　Microbenchmark. (Galax 0.4.0 main memory)

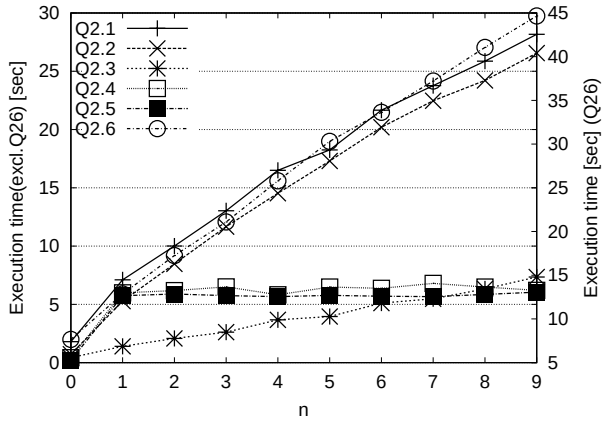

図 7　マイクロベンチマーク (eXist 1.1.1 newcore)
Fig. 7　Microbenchmark. (eXist 1.1.1 newcore)

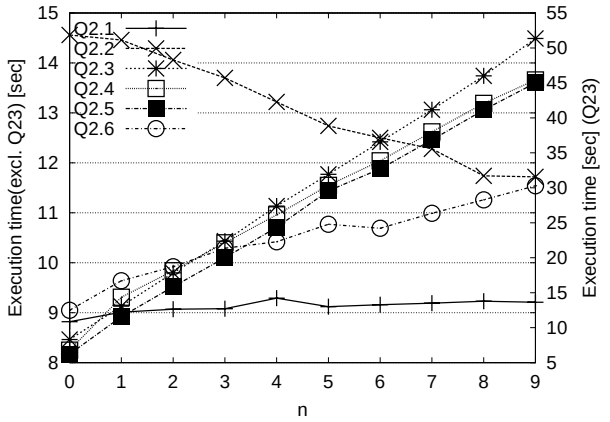

図 8　マイクロベンチマーク (Saxon-B 8.6.1)
Fig. 8　Microbenchmark. (Saxon-B 8.6.1)

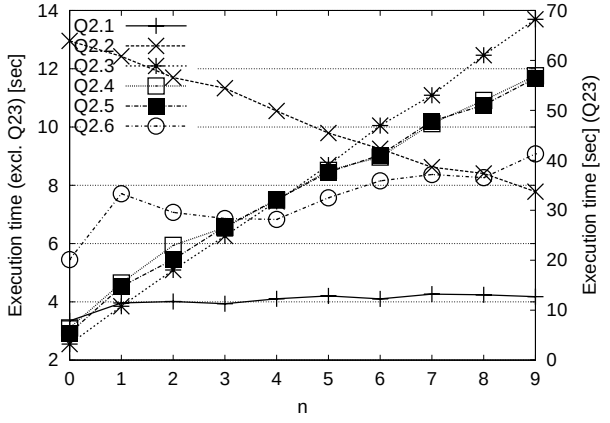

図 9　マイクロベンチマーク (Qizx/open 1.0)
Fig. 9　Microbenchmark. (Qizx/open 1.0)

十分に予想されるものを意図的に選んだことに注意されたい．また，本章で示す二種類のベンチマークの実験環境は，1.5GHz Xeon 4CPU SMP である (OS は Linux(カーネル 2.4.20))．なお，用いるエンジンの実行時間のうち eXist のみ直列化及び入力文書のロードにかかる時間を含んでいない．

4.1　マイクロベンチマーク問合せへのコストモデルの適用例

Manolescu 等 [22] は，XQuery の 6 種類の実装に対してマイクロベンチマークの適用結果を報告している．本節では，そのうち入手しやすい 4 種類の実装 (Galax,eXist,Saxon-B [23],Qizx/open [24]) を用いてこのマイクロベンチマーク問合せにコストモデルを適用し，性能評価結果との適合の程度を考察する．各実装の性能評価結果を図 6，7，8，9 に示す．これらの図の縦軸は左右二種類用意し，問合せの実行時間の性質が明確になるようにしている．例えば，図 9 では Q2.3 の問合せだけ右の縦軸の実行時間を示している．文献 [22] ではグラフを二つに分割しているが，本論文では紙面の都合で一つのグラフにまとめている．

データと問合せの性質

このベンチマークで用いられているデータは，高さ 18 の XML データでそれぞれの要素に一意な id 属性を持つ．ルート (レベル 1) 要素名は t1 で，3268 個の要素名 t2 の子を持つ．これ等子ノードはそれぞれ 10 個の子要素を持ち，その部分木には t3〜t12 の要素が一様に分布している．レベル 3 から 18 迄の要素は全て一つの子要素を持つ．レベル 19 は全て葉要素で，要素名 t13 を持つ．

次に，問合せ (Q2.1(n)〜Q2.6(n)) とその概要を再掲する．

Q2.1(n)

```
for $x in /t1/t2
return <res>{for $y in $x/*/*.../*
             return fn:data($y/@id)}
       </res>
```

Q2.2(n)

```
for $x in /t1/t2
return <res>{$x/*/*.../*}</res>
```

Q2.3(n)

```
for $x in /t1/t2
return <res>{$x//t3, $x//t4, ...,
             $x//t(n+2)}</res>
```

Q2.4(n)

```
for $x in /t1/t2
```

return <res>{$x/*[position()≦ n]}</res>

Q2.5(n)
for $x **in** /t1/t2
return $x/*[position()≦ n]

Q2.6(n)
for $x **in** /t1/t2 **return**
<res>{**for** x_1$ **in** $x/* **return**
　　<res>{**for** x_2$ **in** x_1$/* **return**
　　⋮
　　<res>{**for** x_{n+1}$ **in** x_n$/* **return**
　　　　<res>{x_{n+1}$//t13}</res>
　　}</res>
　　⋮
　　}</res>
}</res>

Q2.1(n) と Q2.2(n) は経路式 $x/*/*$\cdots$/* の長さが n である ($n=0$ のとき $x，$n=1$ のとき $x/*，$\cdots$) ため，$n$ の増加に伴いナビゲーションコストが増大する問合せである．Q2.1(n) は，出力結果の大きさは n によらず一定となるためナビゲーションコストを計測するための問合せである．Q2.2(n) は，n の増加に伴いエレメント構築される出力結果の大きさが減少する．Q2.3(n) と Q2.4(n) は n の増加に伴いエレメント構築される出力結果の大きさも増大する．Q2.3(n) が入力データ中に分散しているデータを返すのに対して，Q2.4(n) は入力データ中に密集しているデータを返す．Q2.5(n) は Q2.4(n) と似ているが出力結果はエレメント構築されないので，エレメント構築のコストが無い分，Q2.4(n) より効率的なはずである．Q2.6(n) は，n の増加に伴いエレメント構築の入れ子が深くなる出力結果となる．尚，Q2.6(n) は文献 [22] とその実験サイト(注11)とで異なるため，我々の判断で実験サイトのものを用いた．

マイクロベンチマークによる本コストモデルの検証

詳細は後記に譲るとし，検証結果を総括すると

- ナビゲーションコストとエレメント構築コストは，本モデルである程度表現できる (Q2.1(n),Q.2.2(n),Q2.3(n),Q2.4(n)). 但し，Q2.6 については増加することはわかるが，増加率まではわからなかかった．
- 入力データの特性をコストモデルでは表現できない (Q2.3(n),Q2.4(n)).
- 直列化コストを表現できない (Q2.5(n)).

Q2.1(n)
for $x **in** /t1/t2
return <res>{**for** $y **in** $x/*/*.../*
　　return fn:data($y/@id)}
　</res>

この問合せは n の増加に従いナビゲーションのコストが増大するが，res 要素の中に含まれるデータは属性値だけであり，常に一定である．そのコストは以下のように算出される．

e_0:=/t1/t2
e_1:= **for** $y **in** $x/*/*.../* **return** fn:data($y/@id)
e_2:= $x/*/*.../*
e_3:=fn:data($y/@id)

とすると

c(Q2.1(n))
= (Cfor)
c(e_0)+s(e_0)·c(<res>{e_1}</res>)
= (Cdec)
c(e_0)+s(e_0)·(c(e_1)
　　+C_e·(s(e_1)+s(e_1//*)+1))
= (Cfor),(Sfor)
c(e_0)+s(e_0)·(c(e_2)+s(e_2)·c(e_3)
　　+C_e·(s(e_2)·s(e_3) +s(e_1//*)+1))
= (Cgstep),(Sgstep)
c(e_0)+s(e_0)·(c($x)+s($x)^c · $n^d \cdot k'$ +s($x)·$k^n$ · c(e_3)
　　+C_e·(s($x)·$k^n$ · s(e_3) +s(e_1//*)+1))
= (s($x)==1), (s($e_3$)==1)
c(e_0)+s(e_0)·(c($x)+$n^d \cdot k'$ +k^n · c(e_3)
　　+C_e·(k^n+s(e_1//*)+1))

ここで，e が生成する木の高さを h とすると，h 回 child 軸を辿ることで，その木の葉まで到達するので，以下の定義でその木の大きさを静的に見積もることができる．

$$\mathtt{s}(e//*) \stackrel{\text{def}}{=} \mathtt{s}(e/*) + \mathtt{s}(e/*/*) + \cdots + \mathtt{s}(e/*/*\cdots/*)$$

この定義を用いると，Q2.1(n) について，e_1 の生成する木の高さは $h-n$ となるので，

s(e_1//*)
=
s(e_1/*) + s(e_1/*/*) + $\cdots$ + s(e_1/*/*$\cdots$/*)
= (Sgstep)
s(e_1) · k + s(e_1) · k^2 + $\cdots$ + s(e_1) · k^{h-n}
= s(e_1) == k^n

$$k^{n+1} + k^{n+2} + \cdots + k^h = \begin{cases} \frac{k^{h+1}-k^{n+1}}{k-1} & (k \neq 1 \text{ のとき}) \\ h-n & (k=1 \text{ のとき}) \end{cases}$$

より，$k \neq 1$ のとき

c(Q2.1(n))
= c(e_0)+s(e_0)·(c($x)+$n^d$ +k^n · c(e_3)
　　+C_e·(k^n+ $\frac{k^{h+1}-k^{n+1}}{k-1}$+1))
= c(e_0)+s(e_0)·(c($x)+$n^d$ +k^n · c(e_3)
　　+C_e·($\frac{k^{h+1}-k^n}{k-1}$+1))
= (c(e_3) − $\frac{C_e}{k-1}$) · (正数) · k^n + (正数) · d^n + (定数)

ここで，e_1 は属性値を返すものであるため，s(e_1//*) は 0 と解釈することができる．また，k の値は (Sgstep) に由来する，ノードの次数 (子ノードの数) を根拠とした定数であり，このベンチマークでは 1 と置くことができる．実験結果は n の値に対し，eXist の単調増加を除いて概ね一定の傾向を示している．本モデルでも $k=1$ と置くことによりエレメント構築子の内側

(注11)：http://www-rocq.inria.fr/~manolesc/microbenchmarks.tml

に来る式に関するコストの部分 (k^n に関する項) が n によらず一定となり，ナビゲーションにかかわる，n^d に関する項のみが変化するようになっているため，この項の寄与が相対的に小さいという仮定のもとでは，実験結果とモデルはよく適合していると考えられる．

Q2.2(n)

```
for $x in /t1/t2
return <res>{$x/*/*.../*}</res>
```

この問合せも n の増加に従いナビゲーションのコストが増大するが，res 要素の中には部分木が含まれる．そのコストは以下のように算出される．

$$
\begin{aligned}
& \mathtt{c}(\text{Q2.2}(n)) \\
= \quad & (\textsc{Cfor}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot\mathtt{c}(\texttt{<res>}\{e_2\}\texttt{</res>}) \\
= \quad & (\textsc{Cdec}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(e_2) \\
& \qquad +C_e\cdot(\mathtt{s}(e_2)+\mathtt{s}(e_2//*)+1)) \\
= \quad & (\textsc{Cgstep}),\ (\textsc{Sgstep}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x)+\mathtt{s}(\$x)^c\cdot n^d\cdot k' \\
& \qquad +C_e\cdot(\mathtt{s}(\$x)\cdot k^n+\mathtt{s}(e_2//*)+1)) \\
= \quad & (\mathtt{s}(\$x)==1) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x)+n^d\cdot k' \\
& \qquad +C_e\cdot(k^n+\mathtt{s}(e_2//*)+1))
\end{aligned}
$$

ここで，Q2.2(n) について，e_2 が生成する木の高さは $h-n$ となるので

$$
\begin{aligned}
& \mathtt{s}(e_2//*)=\mathtt{s}(e_2/*)+\mathtt{s}(e_2/*/*)+\cdots+\mathtt{s}(e_2/*/*.../*) \\
= \quad & (\textsc{Sgstep}) \\
& \mathtt{s}(e_2)\cdot k+\mathtt{s}(e_2)\cdot k^2+\cdots+\mathtt{s}(e_2)\cdot k^{h-n} \\
= \quad & (\mathtt{s}(e_2)==k^n) \\
& k^{n+1}+k^{n+2}+\cdots+k^h=\begin{cases}\frac{k^{h+1}-k^{n+1}}{k-1} & (k\neq 1\text{ のとき})\\ h-n & (k=1\text{ のとき})\end{cases}
\end{aligned}
$$

より，$k\neq 1$ のとき

$$
\begin{aligned}
& \mathtt{c}(\text{Q2.2}(n)) \\
= \ & \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x)+n^d\cdot k' \\
& \qquad +C_e\cdot(k^n+\tfrac{k^{h+1}-k^{n+1}}{k-1}+1)) \\
= \ & \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x)+n^d\cdot k' \\
& \qquad +C_e\cdot(\tfrac{k^{h+1}-k^n}{k-1}+1)) \\
= \ & -\tfrac{C_e\cdot\mathtt{s}(e_0)}{k-1}\cdot k^n+(\text{正数})\cdot n^d\cdot k'+(\text{定数})
\end{aligned}
$$

これにより k^n の係数はマイナスであるので，n に対してエレメント構築に関するコストは減少する．データに関する知識として $k=1$ を導入しても，

$$\mathtt{c}(\text{Q2.2}(n))=-(\text{正数})\cdot n+(\text{正数})\cdot n^d\cdot k'+(\text{定数})$$

という結果は同様に n の増加に対する当該コストの減少を記述している．一方ナビゲーションに関するコスト (第 2 項) は n に関して増加することを示している．実験結果では eXist を除きどの処理系も概ね減少傾向を示しているが，$0\leqq n\leqq 9$ の範囲ではエレメント構築に関するコストの方が凌駕しているという仮定のもとでは，正しくシミュレートしていることが分かる．

Q2.3(n)

```
for $x in /t1/t2
return <res>{$x//t3, $x//t4, ...,
            $x//t(n+2)}</res>
```

この問合せは n の増加に従い結果の大きさが増大する．入力文書中に分散しているデータを結果として返す問合せであり，そのコストは以下のように算出される．

e_4:=$x//t3, $x//t4, ..., $x//t(n+2)
e_5:=$x//t(n+2)

とすると

$$
\begin{aligned}
& \mathtt{c}(\text{Q2.3}(n)) \\
= \quad & (\textsc{Cfor}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot\mathtt{c}(\texttt{<res>}e_4\texttt{</res>}) \\
= \quad & (\textsc{Cdec}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(e_4) \\
& \qquad +C_e\cdot(1+\mathtt{s}(e_4)+\mathtt{s}(e_4//*))) \\
= \quad & (\textsc{Cseq}),\ (\textsc{Sseq}),\ (\mathtt{c}(e_5),\mathtt{s}(e_5)\text{ は }n\text{ によらず一定と仮定}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(e_5)\cdot n \\
& \qquad +C_e\cdot(1+\mathtt{s}(e_5)\cdot n+\mathtt{s}(e_5//*)\cdot n)) \\
= \quad & (n\text{ についてまとめる}) \\
& \mathtt{s}(e_0)\cdot(\mathtt{c}(e_5)+C_e\cdot(\mathtt{s}(e_5)+\mathtt{s}(e_5//*)))\cdot n+\mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot C_e
\end{aligned}
$$

実験結果は全て n に対して線形増加しており，実験結果とモデルはよく適合していると考えられる．

Q2.4(n)

```
for $x in /t1/t2
return <res>{$x/*[position()≦ n]}</res>
```

このコストは，

e_6:=$x/*[position()≦ n]

とすると

$$
\begin{aligned}
& \mathtt{c}(\text{Q2.4}(n)) \\
= \quad & (\textsc{Cfor}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot\mathtt{c}(\texttt{<res>}e_6\texttt{</res>}) \\
= \quad & (\textsc{Cdec}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(e_6) \\
& \qquad +C_e\cdot(1+\mathtt{s}(e_6) \\
& \qquad\qquad +\mathtt{s}(e_6//*))) \\
= \quad & (\textsc{Cpred}),(\textsc{Spred}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x/*)+\mathtt{s}(\$x/*)\cdot\mathtt{c}(\text{position}()\leqq n) \\
& \qquad +C_e\cdot(1+\mathtt{p}(\text{position}()\leqq n)\cdot\mathtt{s}(\$x/*) \\
& \qquad\qquad +\underline{\mathtt{s}(\$x/*[\text{position}()\leqq n]//*)})) \\
= \quad & (\mathtt{p}(\text{position}()\leqq n)==\tfrac{n}{\mathtt{s}(\$x/*)}) \\
& \mathtt{c}(e_0)+\mathtt{s}(e_0)\cdot(\mathtt{c}(\$x/*)+\mathtt{s}(\$x/*)\cdot\mathtt{c}(\text{position}()\leqq n) \\
& \qquad +C_e\cdot(1+n+\underline{n\cdot C})) \\
= \quad & (n\text{ についてまとめる}) \\
& \mathtt{s}(e_0)\cdot C_e\cdot(1+C)\cdot n+(\text{定数})
\end{aligned}
$$

尚，$\mathtt{s}(\$x/*[\text{position}()\leqq n]//*)==n\cdot C$ (但し，C はある定数) とした理由は，n に比例して $x/*[position() ≦ n] の子孫も増えるためである．

実験結果は n の値に対し eXist を除いて概ね線形増加してお

り，実験結果とモデルは eXist を除いて概ね適合していると考えられる．

Q2.5(n)

for $x **in** /t1/t2
return $x/*[position()≦ n]

c(Q2.5(n))
= (CFOR)
c(e_0) + s(e_0) · c($x/*[position() ≦ n])
= (CPRED)
c(e_0) + s(e_0) · (c($x/*) + s($x/*) · c(position() ≦ n))

n によらず定数のコストとなる．実験結果では，直列化にかかる時間を含まない eXsit とはよく適合しているが，本コストモデルはこのような時間を考慮していないため，他のエンジンの結果とは乖離している．

Q2.6(n)

この問合せは，n の増大に伴い深い入れ子 **return** 節となるものである．文献 [22] 同様，n を用いた一般形ではなく，具体的に $n = 0, 1, 2$ の場合についてそのコストを計算し，n に関するコストを考察する．

Q2.6(0)

for $x **in** /t1/t2 **return**
<res>{**for** $x1 **in** $x/* **return**
<res>{$x1//t13}</res>}
</res>

e_7:=**for** $x1 **in** $x/* **return** <res>{$x1//t13}</res>

とすると

c(Q2.6(0))
= (CFOR)
c(e_0) + s(e_0) · c(<res>e_7</res>)
= (CDEC),(C_0 := c(e_0)),(C_1 := s(e_0))
$C_0 + C_1 \cdot$ (c(e_7)
$+C_e \cdot$ (1+s(e_7)
+s(e_7//*)))
= (CFOR),(SFOR)
$C_0 + C_1 \cdot$ (c(= *mbox*$$x/*) + s($x/*) · c(<res>{$x1//t13}</res>)
$+C_e \cdot$ (1+s($x/*) · s(<res>{$x1//t13}</res>)
+s(e_7//*)))
= (CDEC),(SDEC),(C_2 := c($x/*)),($C_3$:= s($x/*))
$C_0 + C_1 \cdot (C_2 + C_3 \cdot$ (c($x1//t13)
$+C_e \cdot$ (1+s($x1//t13)
+s($x1//t13//*)))
$+C_e \cdot$ (1+$C_3 \cdot 1$
+s(e_7//*)))

Q2.6(1)

for $x **in** /t1/t2 **return**
<res>{**for** $x1 **in** $x/* **return**
<res>{**for** $x2 **in** $x1/* **return**
<res>{$x2//t13}</res>}
</res>}
</res>

e_8:=**for** $x2 **in** $x1/* **return** <res>{$x2//t13}</res>
e_9:=**for** $x1 **in** $x/* **return** <res>$e_8$</res>

とすると

c(Q2.6(1))
= (CFOR)
c(e_0) + s(e_0) · c(<res>e_9</res>)
= (CDEC),(C_0 := c(e_0)),(C_1 := s(e_0))
$C_0 + C_1 \cdot$ (c(e_9)
$+C_e \cdot$ (1+s(e_9)
+s(e_9//*)))
= (CFOR),(SFOR)
$C_0 + C_1 \cdot$ (c($x/*) + s($x/*) · c(<res>e_8</res>)
$+C_e \cdot$ (1+s($x/*) · s(<res>$e_8$</res>)
+s(e_9//*)))
= (CDEC),(SDEC),(C_2 := c($x/*)),($C_3$:= s($x/*))
$C_0 + C_1 \cdot (C_2 + C_3 \cdot$ (c(e_8)
$+C_e \cdot$ (1+s(e_8)
+s(e_8//*)))
$+C_e \cdot$ (1+$C_3 \cdot 1$
+s(e_9//*)))
= (CFOR),(SFOR)
$C_0 + C_1 \cdot (C_2 + C_3 \cdot$ (c($x1/*) + s($x1/*) · c(<res>{$x2//t13}</res>)
$+C_e \cdot$ (1+s($x1/*) · s(<res>{$x2//t13}</res>)
+s(e_8//*)))
$+C_e \cdot$ (1+C_3
+s(e_9//*)))
= (CDEC),(SDEC),(C_4 := c($x1/*)),($C_5$:= s($x1/*))
$C_0 + C_1 \cdot (C_2 + C_3 \cdot (C_4 + C_5 \cdot$ (c($x2//t13)
$+C_e \cdot$ (1+s($x2//t13)
+s($x2//t13//*)
$+C_e \cdot$ (1+C_5
+s(e_8//*)))
$+C_e \cdot$ (1+C_3
+s(e_9//*)))

Q2.6(2)

for $x **in** /t1/t2 **return**
<res>{**for** $x1 **in** $x/* **return**
<res>{**for** $x2 **in** $x1/* **return**
<res>{**for** $x3 **in** $x2/* **return**
<res>{$x3//t13}</res>}
</res>}
</res>}
</res>

e_{10}:=**for** $x3 **in** $x2/* **return** <res>{$x3//t13}</res>
e_{11}:=**for** $x2 **in** $x1/* **return** <res>$e_{10}$</res>
e_{12}:=**for** $x1 **in** $x/* **return** <res>$e_{11}$</res>

とすると

c(Q2.6(2))
= (CFOR)
c(e_0) + s(e_0) · c(<res>e_{12}</res>)
= (CDEC),(C_0 := c(e_0)),(C_1 := s(e_0))
$C_0 + C_1 \cdot$ (c(e_{12})
$+C_e \cdot$ (1+s(e_{12})
+s(e_{12}//*)))
= (CFOR),(SFOR)
$C_0 + C_1 \cdot$ (c($x/*) + s($x/*) · c(<res>e_{11}</res>)
$+C_e \cdot$ (1+s($x/*) · s(<res>$e_{11}$</res>)
+s(e_{12}//*)))
= (CDEC),(SDEC),(C_2 := c($x/*)),($C_3$:= s($x/*))
$C_0 + C_1 \cdot (C_2 + C_3 \cdot$ (c(e_{11})
$+C_e \cdot$ (1+s(e_{11})
+s(e_{11}//*)))
$+C_e \cdot$ (1+$C_3 \cdot 1$
+s(e_{12}//*)))
= (CFOR),(SFOR)
$C_0 + C_1 \cdot (C_2 + C_3 \cdot$ (c($x1/*) + s($x1/*) · c(<res>e_{10}</res>)
$+C_e \cdot$ (1+s($x1/*) · s(<res>$e_{10}$</res>)
+s(e_{11}//*)))
$+C_e \cdot$ (1+C_3
+s(e_{12}//*)))

$$
\begin{aligned}
= \quad & (\textsc{Cdec}),(\textsc{Sdec}),(C_4 := \mathtt{c}(\$\mathrm{x1}/*)),(C_5 := \mathtt{s}(\$\mathrm{x1}/*)) \\
& C_0 + C_1 \cdot (C_2 + C_3 \cdot (C_4 + C_5 \cdot (\mathtt{c}(e_{10}) \\
& \qquad + C_e \cdot (1 + \mathtt{s}(e_{10}) \\
& \qquad\quad + \mathtt{s}(e_{10}//*) \\
& \qquad + C_e \cdot (1 + C_5 \\
& \qquad\quad + \mathtt{s}(e_{11}//*))) \\
& \quad + C_e \cdot (1 + C_3 \\
& \qquad + \mathtt{s}(e_{12}//*))) \\
= \quad & (\textsc{Cfor}),(\textsc{Sfor}) \\
& C_0 + C_1 \cdot (C_2 + C_3 \cdot (C_4 + C_5 \cdot (\mathtt{c}(\$\mathrm{x2}/*) \\
& \qquad + \mathtt{s}(\$\mathrm{x2}/*) \cdot \mathtt{c}(\texttt{<res>}\{\$\mathrm{x3}//\mathrm{t13}\}\texttt{</res>}) \\
& \qquad + C_e \cdot (1 + \mathtt{s}(\$\mathrm{x2}/*) \cdot \mathtt{s}(\texttt{<res>}\{\$\mathrm{x3}//\mathrm{t13}\}\texttt{</res>}) \\
& \qquad\quad + \mathtt{s}(e_{10}//*) \\
& \qquad + C_e \cdot (1 + C_5 \\
& \qquad\quad + \mathtt{s}(e_{11}//*))) \\
& \quad + C_e \cdot (1 + C_3 \\
& \qquad + \mathtt{s}(e_{12}//*))) \\
= \quad & (\textsc{Cdec}),(\textsc{Sdec}),(C_6 := \mathtt{c}(\$\mathrm{x2}/*)),(C_7 := \mathtt{s}(\$\mathrm{x2}/*)) \\
& C_0 + C_1 \cdot (C_2 + C_3 \cdot (C_4 + C_5 \cdot (C_6 + C_7 \cdot (\mathtt{c}(\$\mathrm{x3}//\mathrm{t13}) \\
& \qquad + C_e \cdot (1 + \mathtt{s}(\$\mathrm{x3}//\mathrm{t13}) \\
& \qquad\quad + \mathtt{s}(\$\mathrm{x3}//\mathrm{t13}//*) \\
& \qquad + C_e \cdot (1 + C_7 \\
& \qquad\quad + \mathtt{s}(e_{10}//*) \\
& \qquad + C_e \cdot (1 + C_5 \\
& \qquad\quad + \mathtt{s}(e_{11}//*))) \\
& \quad + C_e \cdot (1 + C_3 \\
& \qquad + \mathtt{s}(e_{12}//*)))
\end{aligned}
$$

このコスト計算の結果は，n の増加に伴い，エレメント構築にかかわるコストが増加する一方で，$\$\mathrm{x}n//\mathrm{t13}$ にかかるコストが減少するものとなる．前者の増分は，後者の減少に比べて十分大きくなっており，増加の傾向という観点からは，実験結果と概ね符合する．しかし，増加の度合い (線形か否か等) に関しては本コストモデルでは見積もることができなかった．

4.2 エレメント構築式によるビュー定義に対する問合せへのコストモデルの適用例

問合せは XMark [25] の Q10 によるビュー定義と，それに対する収入が 10000 を超える市民が住んでいる都市を抜き出す問合せで構成される．

最適化前:

```
(: Q10: Grouping :)
(: Q10: List all persons according to their interest; use French markup in the
   result. :)
let $view := (
for $i in distinct-values($auction/site/people/person/profile/interest/@category)
let $p := for    $t in $auction/site/people/person
          where  $t/profile/interest/@category = $i
           return
             <personne>
               <statistiques>
                 <sexe> { $t/profile/gender/text() } </sexe>
                 <age> { $t/profile/age/text() } </age>
                 <education> { $t/profile/education/text() } </education>
                 <revenu> { fn:data($t/profile/@income) } </revenu>
               </statistiques>
               <coordonnees>
                 <nom> { $t/name/text() } </nom>
                 <rue> { $t/address/street/text() } </rue>
                 <ville> { $t/address/city/text() } </ville>
                 <pays> { $t/address/country/text() } </pays>
                 <reseau>
                   <courrier> { $t/emailaddress/text() } </courrier>
                   <pagePerso> { $t/homepage/text() } </pagePerso>
                 </reseau>
               </coordonnees>
               <cartePaiement> { $t/creditcard/text() } </cartePaiement>
             </personne>
return <categorie>
         <id> { $i } </id>
         { $p }
       </categorie>
)
for $p1 in $view/personne
where $p1/statistiques/revenu > 10000
return $p1/coordonnees/ville
```

最適化後:

```
for $i in distinct-values($auction/site/people/person/profile/interest/@category)
for $t in $auction/site/people/person
where ($t/profile/interest/@category = $i) and
      (<revenu> { fn:data($t/profile/@income) } </revenu> > 10000)
return <ville> { $t/address/city/text() } </ville>
```

この変換では，変換規則 (Twfa)，(Tflu)，**where** 節と **if** の相互変換 (Twif)，**if** から **and** への変換，β 簡約を用いている．この最適化によるコスト減少分は以下のようになる．コスト計算には，(Cfor)，(Cwfor)，(Cand)，(Pand)，(Cfcall)，(Clet) を用いている．

```
e1 = distinct-values(
     $auction/site/people/person/profile/interest/@category)
e2 = $auction/site/people/person
e3 = ($t/profile/interest/@category = $i)
e4 = <revenu> { fn:data($t/profile/@income) } </revenu>
e5 = <ville> { $t/address/city/text() } </ville>
```

とおくと

```
(: 最適化前 :)
  c(e1)
 +s(e1)*(c(e2)
         +s(e2)
           *(c(e3)+p(e3)*c(<personne>
                           ...
                           e4
                           ...
                           e5
                           ...
                          </personne>))
          +c(<categorie><id>{$i}</id>{$p}</categorie>))
 +c($view/personne)
 +s($view/personne)*(c($p1/statistiques/revenu)+c(10000)+c(>)
 +p(e4 > 10000)*c($p1/coordonnees/ville))
(: 最適化後 :)
  c(e1)
 +s(e1)*(c(e2)
         +s(e2)
           *(c(e3)+p(e3)*(c(e4)+c(10000)+c(>))
                        +p(e3)*p(e4 > 10000)*c(e5)))
```

最適化前 − 最適化後 =

```
s(distinct-values(
    $auction/site/people/person/profile/interest/@category))
·(s($auction/site/people/person)
  ·p($t/profile/interest/@category = $i)
  ·(c(<personne>
         ...
         <revenu>{fn:data($t/profile/@income) } </revenu> (: 1 :)
         ...
         <ville>{ $t/address/city/text() } </ville> (: 2 :)
         ...
         </personne>)

     −c(<revenu>{fn:data($t/profile/@income)}</revenu>) (: 1' :)

     −p(<revenu>{fn:data($t/profile/@income)}</revenu> > 10000)
     ·c(<ville>{$t/address/city/text()}</ville>) (: 2' :)

   +(c($p1/statistiques/revenu)
     +p($p1/statistiques/revenu > 10000)
     ·c($p1/coordonnees/ville))
  )
 +c(<categorie><id>{$i}</id>{$p}</categorie>)
)
+c($view/personne)
```

(: 1 :) 及び (: 2 :) の部分は，それぞれ (: 1' :) 及び (: 2' :) の部分と打ち消しあい (2' は 1 より小さい係数を持つ)，正の値が残る．したがって，上記の変換は問合せを高速化することが保証される．定量的には，上の結果から文書の大きさ (s) が大きくなると，結果の式のうち s に依存する上から 1 行目から 2 行目にかけての$auction から始まる二つの経路式に関するサイズの積の因子が支配的となり，コストの減少分は s の二乗に比例していることが分かる(注12)．実エンジンを用いた実行時間

(注12)：$auction/site/people/person が空でないことが前提であるが，実験に用いたデータはそのような性質を満たしていることを確認した．

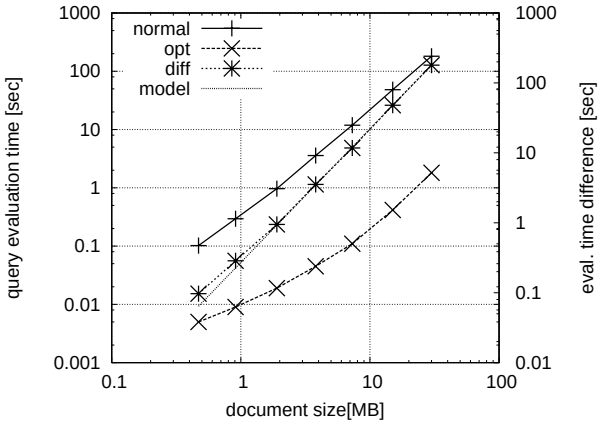

図 10　XMark Q10 に対する問合せ (Galax 0.4.0 主記憶実装)
Fig. 10　Query for Q10 of XMark. (Galax 0.4.0 main memory)

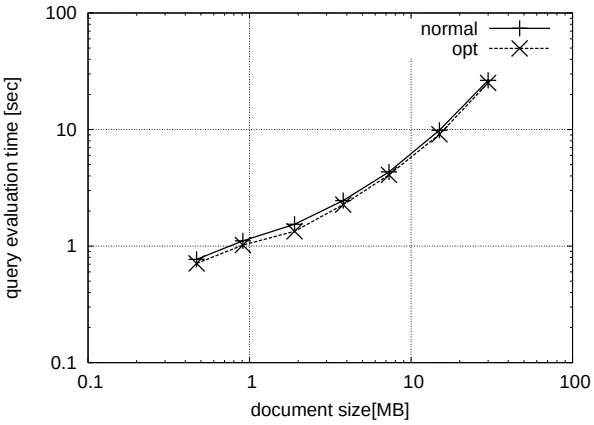

図 12　XMark Q10 に対する問合せ (Saxon-B 8.6.1)
Fig. 12　Query for Q10 of XMark. (Saxon-B 8.6.1)

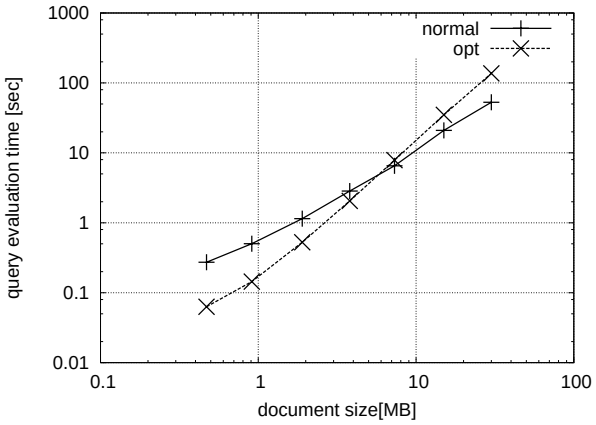

図 11　XMark Q10 に対する問合せ (eXist 1.1.1 newcore)
Fig. 11　Query for Q10 of XMark. (eXist 1.1.1 newcore)

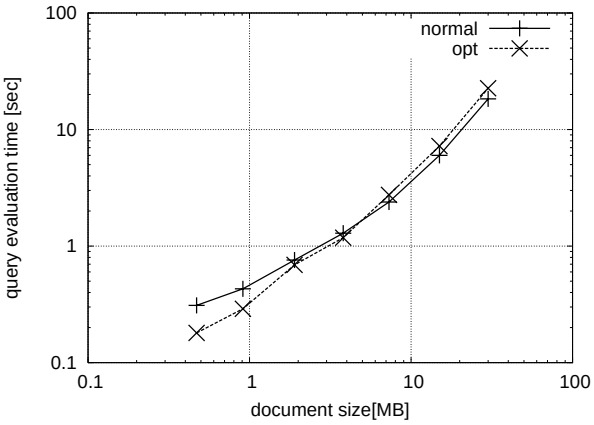

図 13　XMark Q10 に対する問合せ (Qizx/open 1.0)
Fig. 13　Query for Q10 of XMark. (Qizx/open 1.0)

を図 10，11，12，13 に示す．

時間測定は 1.5GHz Xeon 4CPU SMP 上で行った (OS は Linux(カーネル 2.4.20))．

Galax において，短縮された実行時間 (T) は図 10 の opt の値を normal の値から引いたもの (図中 diff) で得られ，回帰分析により $T = 0.27s^{1.9}$(図中 model) にフィットした．修正二乗総和 (切片つき) と二乗和の差の，修正二乗総和に対する割合 (線形回帰分析での R^2 値に相当) は 1.0 であった．

この結果により，コストモデルは実エンジンの最適化効果を模倣できることの一例が示された．

モデルから乖離する例とその理由に関する考察

Galax による上記実験ではモデルとうまく適合したが，他の三つのエンジンでは，うまく当てはまらないことが分かった．eXist(注13)では s が大きい時には書換え後の問合せの実行時間の方が書換え前を上回っている．理由として，eXist が中間結果として生成するエレメントの扱いが文脈によって変わることが考えられる．書換え前の問合せの場合，personne 要素が逐次二次記憶上に一時書き出される．書換え後の場合，ville 要素が最後の return 節で新たに構築されるが，それは一時書き出しの対象とはならず，メモリ上に確保される．更に，書換え前では category 属性に関する選択が処理の上流で行われるため発生する中間データは元の文書の大きさと同じオーダーに抑えられるが，書換え後では二つの **for** 節により category 属性の種類と person の数の積の規模のタプルストリームが一旦生成され，そのコストが，エレメント構築の削減の効果を打ち消しているものと考えられる．Saxon-B 及び Qizx/open は主記憶実装であり，中間結果の書出しは発生していない．コストモデルがこのような現象をモデル化できていない理由は以下のとおりである．

文脈依存性の欠如　eXist ではエレメント構築式の扱いが式の中の相対的な位置により (データの入出力の流れの最下流にあるかないかで) 異なるが，コストモデルではそのような差別を考慮していない．

資源の有限性の考慮の欠如　作業領域の確保は一般に，未使用の領域が少なくなるにつれて時間がかかる．現在のコストモデルではそのような有限性を考慮していない．

作業領域の有限性は，エンジンにパイプライン処理等を組み込むことにより捨象可能であるため，OS の分野で仮想記憶により仮想メモリ空間の制約が撤廃されたように，そのような有限

(注13)：先に用いたバージョンは新たに構築された要素内のナビゲーションが実装されていなかったため，eXist 1.1.1 newcore を用いた．

性を考慮しない本モデルも十分意義があると考えられる．

5. 関連研究

コスト変化の見積りに関しては，書換えのレベルでコストモデルを伴う等価変換規則は筆者らの知る限り存在しない．

[26] は XQuery の正規化について言及している．そこでの規則の目的はエレメント構築等バックエンドが効率的に扱うことができない構文要素を回避するためのものである．変換規則 NR5 等，本論文のコストモデルに基づいて計算するとコスト増加となる変換も見られる．

ストリーミング向けの XQuery 処理系について述べた [27] では，コストモデルの使用は意図的に排除している．その理由は (1) 統計情報は予測不能である，(2) XQuery に対しては，効果的なコストモデルの開発は困難である，(3) XQuery データモデルには順序が規定されているため，SQL に比べて変換の自由度が少なく効果も小さい，(4) 実際に扱った応用ではコストモデルは重要ではなかった，というものである．これ等に対する本論文の立場は，(1) コストモデルの相対性 (具体的な値の計算を不要にする) により問合せ対象データに独立した議論が可能になっている，(2) 本論文は，本質的に難しい実行時のプラン選択ではなく，静的に決められる利得や損失の計算を対象としている，(3) 実際には本論文で挙げたように多くの変換規則が存在する，(4) 筆者等は (3.4 節で述べたように) コストモデルでうまく説明できるエンジンの振舞いがあることを経験している．

XML データが RDB に格納され，XQuery が SQL に変換されるような実装では，コストは SQL エンジンのコストに基づいてモデル化されている．[8] は RDB に格納された XML の文脈でコストモデルを扱っている．モデルは述語の選択率や統計情報に基づいている．具体的には，RDB 用のコストモデル [28] が採用されている．[28] では，軸足を関係問合せの最適化に置き，その応用として，XQuery の **FLWOR** 式の基となった構文を持つ XML-QL から変換された問合せを対象としている．綿密なコスト関数が定義されている．

TIMBER [3] は XQuery を含む問合せを独自の tree algebra に変換するが，そこでコストの見積りが重要な役割を担っている．最適なパイプライン処理に繋がる問合せ実行プランを生成するアルゴリズムが提案されている．コスト関数には経路式における選択率の見積り方式 [29] とサイズ関数 [30] が用いられている．[30] は XQuery 全般ではなく twig 問合せを対象としたコスト見積りを扱っている．join のような具体的な操作を対象としている．

IBM DB2 XML に関する [4] は XQuery コストモデルの最近の研究成果のひとつである．XQuery は索引アクセス等を含む物理プランの候補集合に変換され，文書数やある特定の要素の配下の名前ごとの平均ノード数等の統計情報に基づきコスト見積りが行われる．

本論文は FLWOR や限量式等を含む XQuery 全般を対象としている．選択率の見積やサイズ関数を扱う先行研究は存在するが，本研究ではコスト関数に，解釈しない部分式を，それが変換に直接関与しない (変換の過程で書き変らない) 限り，安全に残すことができる．したがって本論文ではそのような関数の戻り値に独立した議論をすることが可能となっている．

本論文のコストモデルは候補となる物理プラン同士のコスト値を具体的に求め，最小値を持つプランを選択して実行するような使い方ではなく，ソースレベルでの利用を対象としている．[4] にも書換えのフェーズは存在し，そこでの変換規則は経験則に基づいている．本論文のモデルはエンジン独立の書換えを対象としている．

6. むすび

本論文では，ソースレベルの問合せ書換えに基づく XQuery の最適化の文脈で，式のコストを定量化するコストモデルを提案した．このモデルにより，本論文で述べられている全ての変換規則における相対的なコスト変化について証明することができた．更に，実エンジンで本コストモデルを評価した結果，本モデルで記述できない，コストが文脈依存性や残存作業領域依存性を持つようなエンジンも存在することが確認できた．

我々のコストモデルが “仮想エンジン” として XQuery の最適化研究等の簡易な評価ツールの役割を担うことができればと考えている．例えば，本コストモデルは基本的には形式的意味を用いてモデル化されているため，新規エンジン作成時の最適化機構の開発指針となる．具体的には，経路式について Gottlob が形式的意味に忠実に従った評価コストが非常に高価であることを示し，別の評価戦略の開発の契機となったことと同様に，本コストモデルがエレメントコンストラクタのコストを形式的に示したことにより，新たなエンジンを開発する際に，そのコストを避ける新たな評価戦略の開発を触発する契機となることも期待される．実際，筆者らは上記の考察に基づき新たな最適化機構を開発中である [10].

今後の課題としては，索引の利用や遅延評価，パイプライン評価等を考慮すること，マイクロベンチマークによるコストモデルの評価において，FLWOR, エレメントコンストラクタ以外の構文要素や入力サイズを変化させた場合も考慮すること，アプリケーションベンチマークの拡充が考えられる．

謝辞 本研究に対して有益なコメントやご助言を賜りました大阪大学の石原靖哲准教授に深く感謝致します．本研究の一部は，文部科学省科学研究費補助金若手研究 (B)「文書蓄積・検索等の非数値計算向け並列非同期入出力処理の高速化に関する研究」(14750339) 及び「副作用を考慮した関数型問い合わせ言語の並列性に関する研究」(17700041) の支援を受けている．

文献

[1] A. Deutsch, Y. Papakonstantinou and Y. Xu: “The NEXT logical framework for XQuery”, VLDB, pp. 168–179 (2004).

[2] M. F. Fernández, D. Florescu, J. Robie, D. Chamberlin, S. Boag and J. Siméon: “XQuery 1.0: An XML query language”, W3C recommendation, W3C (2007). http://www.w3.org/TR/2007/REC-xquery-20070123/.

[3] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, S. Paparizos, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu and C. Yu: “TIMBER: A native XML database”, VLDB Journal, **11**, 4, pp. 274–291

(2002).
[4] A. Balmin, T. Eliaz, J. Hornibrook, L. Lim, G. M. Lohman, D. Simmen, M. Wang and C. Zhang: “Cost-based optimization in DB2 XML”, IBM Syst. J., **45**, 2, pp. 299–319 (2006).
[5] G. Graefe: “Query evaluation techniques for large databases”, ACM Computing Surveys, **25**, 2, pp. 73–170 (1993).
[6] I. S. Mumick, S. J. Finkelstein, H. Pirahesh and R. Ramakrishnan: “Magic is relevant”, SIGMOD Conference, pp. 247–258 (1990).
[7] T. Grust, S. Sakr and J. Teubner: “XQuery on SQL hosts”, VLDB, pp. 252–263 (2004).
[8] P. Bohannon, J. Freire, P. Roy and J. Siméon: “From XML schema to relations: A cost-based approach to XML storage”, ICDE, pp. 64–75 (2002).
[9] R. Krishnamurthy, P. Kaushik and J. Naughton: “Efficient XML-to-SQL query translation: Where to add the intelligence?”, VLDB, pp. 144–155 (2004).
[10] H. Kato, S. Hidaka and M. Yoshikawa: “Rewriting XQuery by child-path folding”, Progress in Infromatics, 4, pp. 15–28 (2007).
[11] I. Tatarinov and A. Halevy: “Efficient query reformulation in peer data management systems”, SIGMOD Conference, pp. 539–550 (2004).
[12] J. Siméon, K. Rose, P. Wadler, M. Fernández, P. Fankhauser, M. Rys, D. Draper and A. Malhotra: “XQuery 1.0 and XPath 2.0 formal semantics”, W3C recommendation, W3C (2007). http://www.w3.org/TR/2007/REC-xquery-semantics-20070123/.
[13] Lucent: “Galax” (2003). http://db.bell-labs.com/galax/.
[14] W. Meier: “eXist: An open source native XML database”, Proceedings of the Web- and Database-Related Workshops, LNCS, pp. 169–183 (2002).
[15] C. Ré, J. Siméon and M. Fernández: “A complete and efficient algebraic compiler for XQuery”, ICDE, p. 14 (2006).
[16] G. Gottlob, C. Koch and R. Pichler: “Efficient algorithms for processing XPath queries”, VLDB, pp. 95–106 (2002).
[17] S. Hidaka, H. Kato and M. Yoshikawa: “An XQuery cost model in relative form”, Technical Report NII-2005-016E, National Institute of Informatics (2005). http://research.nii.ac.jp/TechReports/05-016E.html.
[18] M. F. Fernandez, J. Siméon and P. Wadler: “A Semi-monad for Semi-structured Data”, ICDT, pp. 263–300 (2001).
[19] 加藤 弘之, 日高 宗一郎：“拡張 Dewey Order を用いた distinct-doc-order のエミュレーションに基づく XQuery の書き換え規則”, 情報処理学会第 64 回プログラミング研究会発表資料 (2007).
[20] C. Koch: “On the role of composition in XQuery”, WebDB (Eds. by A. Doan, F. Neven, R. McCann and G. J. Bex), pp. 37–42 (2005).
[21] A. Poulovassilis and C. Small: “Algebraic query optimisation for database programming languages”, VLDB Journal, **5**, 2, pp. 119–132 (1996).
[22] I. Manolescu, C. Miachon and P. Michiels: “Towards micro-benchmarking XQuery”, ExpDB (Eds. by P. Bonnet and I. Manolescu), ACM, pp. 28–39 (2006).
[23] Michael Kay: “Saxon XSLT and XQuery Processor” (2001). http://sourceforge.net/projects/saxon.
[24] Axyana software: “Qizx/open: An open-source Java implementation of XQuery”. http://www.axyana.com/qizxopen.
[25] A. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu and R. Busse: “XMark: A benchmark for XML data management”, VLDB, pp. 974–985 (2002).
[26] I. Manolescu, D. Florescu and D. Kossmann: “Answering XML queries on heterogeneous data sources”, VLDB, pp. 241–250 (2001).
[27] D. Florescu, C. Hillery, D. Kossmann, P. Lucas, F. Riccardi, T. Westmann, M. J. Carey and A. Sundararajan: “The BEA streaming XQuery processor”, VLDB Journal, **13**, 3, pp. 294–315 (2004).
[28] P. Roy, S. Seshadri, S. Sudarshan and S. Bhobe: “Efficient and extensible algorithms for multi query optimization”, SIGMOD Conference, pp. 249–260 (2000).
[29] A. Aboulnaga, A. R. Alameldeen and J. F. Naughton: “Estimating the selectivity of XML path expressions for Internet scale applications”, VLDB, pp. 591–600 (2001).
[30] Y. Wu, J. M. Patel and H. V. Jagadish: “Estimating answer sizes for XML queries”, EDBT, pp. 590–608 (2002).

付　　録

1. 変換規則の意味的等価性の証明

本章では，本文で扱った変換規則の意味的等価性を証明する．

1.1 準　　備

a） 変換規則と自由変数

全ての式は，自由変数が含まれている場合その束縛を一意に定める環境が存在すると仮定している．

言明 $e_1 = e_2$ は，両辺が同じ環境のもとで評価される場合，その任意の環境において，等価性が保証されているというものである．つまり，変数束縛環境を Γ とすると，

$$\frac{\forall \Gamma \left(\begin{array}{c} \Gamma \vdash e_1 \text{ yields } v_1 \quad \Gamma \vdash e_2 \text{ yields } v_2 \\ v_1 \ == \ v_2 \end{array} \right)}{\forall \Gamma \vdash e_1 \ = \ e_2}$$

ここで e yields v とは式 e を評価した結果が値 v となることを表す．この言明の証明は，両辺が同一の式に簡約可能であることにより示される．両辺に出現する同じ名前の変数には，両者に同じ環境が与えられているという上記の仮定のもとでは同じ値が束縛される．よって同一の式は同一の値に評価されるため，自由変数の有無が言明の正しさを損なうことはない．

以降では各変換規則における $\Gamma \vdash$ は省略し，式 e_1 と式 e_2 の等価性を単に $e_1 = e_2$ と記す．

1.1.1 some,every,for 式を統一するための Φ_Q の導入

for と限量式を統一的に扱うために，以下のような二分リダクション演算子

$$\Phi_Q :: (a \to b) \to (b \to b \to b) \to b \to (\!(a)\!) \to b$$

を導入する．ただし，$(\!(T)\!)$ は型 T の列，$T_1 \to T_2$ は引数の型を T_1 とし戻り値の型を T_2 とする関数，$e :: T$ は式 e が型 T を持つことを示す．

a） 定　　義

$$\begin{array}{lll} \Phi_Q \ f \ op \ e \ () & = e & \text{(PQ1)} \\ \Phi_Q \ f \ op \ e \ (x) & = f \ x & \text{(PQ2)} \\ \Phi_Q \ f \ op \ e \ (q \ , \ q') & = (\Phi_Q \ f \ op \ e \ q) & \\ & \quad op \ (\Phi_Q \ f \ op \ e \ q') & \text{(PQ3)} \end{array}$$

q の型としては，任意の型 a の列を仮定している．また，演算子 op は結合的であるとする．上記の定義において，(PQ1) は入力が空，(PQ2) は入力がシングルトン，(PQ3) は入力が二つの列を結合したものになっている場合に対応する．

b） 型 の 制 約

$$\frac{f :: a \to b \quad op :: b \to b \to b \quad e :: b \quad q :: (\!|a|\!)}{(\Phi_Q\ f\ op\ e\ q) :: b}$$

[定理 19] ($\Phi_Q \leftrightarrow$ **for** 式対応)

$(\textbf{for}\ \$x\ \textbf{in}\ q\ \textbf{return}\ f)) = (\Phi_Q\ f\ ,\ ()\ q)$

(証明)　(1) 空入力の場合

　　for $\$x$ **in** () **return** f
$=$　　**for** の形式的意味
　　()
$=$　　(PQ1)
　　$\Phi_Q\ f\ ,\ ()\ ()$

(2) シングルトン入力の場合

　　for $\$x$ **in** y **return** f
$=$　　**for** の形式的意味において入力長が 1 の場合
　　$f\ y$
$=$　(PQ2)
　　$\Phi_Q\ f\ ,\ ()\ (y)$

(3) 任意の列 q の場合　帰納法による.

$(\textbf{for}\ \$x\ \textbf{in}\ q\ \textbf{return}\ f) = \Phi_Q\ f\ ,\ ()\ q$

を帰納法の仮定とする. 入力が (q, y) (y はシングルトンとする) のとき

　　for $\$x$ **in** (q, y) **return** f
$=$　　**for** の形式的意味, $q = (q_1, q_2, \ldots, q_N)$
　　$f\ q_1, f\ q_2, \ldots, f\ q_N, f\ y$
$=$　　形式的意味及び上記 (2)
　　$(\textbf{for}\ \$x\ \textbf{in}\ q\ \textbf{return}\ f), (\Phi_Q\ f\ ,\ ()\ y)$
$=$　　帰納法の仮定
　　$((\Phi_Q\ f\ ,\ ()\ q), (\Phi_Q\ f\ ,\ ()\ y))$
$=$　　(PQ3)
　　$\Phi_Q\ f\ ,\ ()\ (q, (y))$

つまり, q' が

$(\textbf{for}\ \$x\ \textbf{in}\ q'\ \textbf{return}\ f) = (\Phi_Q\ f\ ,\ ()\ q')$

を満たすならば, (q', y) は

$\textbf{for}\ \$x\ \textbf{in}\ (q',\ y)\ \textbf{return}\ f = ((\Phi_Q\ f\ ,\ ()\ (q',\ (y))))$

を満たす.

ところで, 空及びシングルトン列 q が入力の場合,

$(\textbf{for}\ \$x\ \textbf{in}\ q\ \textbf{return}\ f) = (\Phi_Q\ f\ ,\ ()\ q)$

は成り立つ. したがって, 帰納法により

$(\textbf{for}\ \$x\ \textbf{in}\ q\ \textbf{return}\ f) = (\Phi_Q\ f\ ,\ ()\ q)$

が証明された. □

[定理 20] ($\Phi_Q \leftrightarrow$ 限量式 対応)

$(\textbf{some}\ \$x\ \textbf{in}\ q\ \textbf{satisfies}\ f) = (\Phi_Q\ f \vee \mathsf{false}\ q)$

ただし, `fn:false()` をここでは false と略記する. 同様に `fn:true()` を true と記す.

(証明)　(1) 空入力

　　some $\$x$ **in** () **satisfies** f
$=$　　**some** の形式的意味
　　false
$=$　　(PQ1)
　　$\Phi_Q\ f \vee\ \mathsf{false}\ ()$

some の形式的意味は「少なくとも一つの入力について f () が true に評価される」となっており, その否定により偽となる.

(2) シングルトン入力

　　some $\$x$ **in** y **satisfies** f
$=$　　**some** の形式的意味
　　$f\ y$
$=$　　(PQ2)
　　$\Phi_Q\ f \vee\ \mathsf{false}\ (y)$

some の形式的意味において入力を y 一つだけにした場合, 結果はその f への適用結果 $f\ y$ のみで決定される.

(3) 任意の入力列 q の場合　帰納法による.

$(\textbf{some}\ \$x\ \textbf{in}\ q\ \textbf{satisfies}\ f) = (\Phi_Q\ f \vee\ \mathsf{false}\ q)$

を帰納法の仮定とする.

入力列 q にシングルトン y が連結されている場合

　　some $\$x$ **in** (q, y) **satisfies** f
$=$　　形式的意味, $q = (q_1, q_2, \ldots, q_N)$
　　$f\ q_1\ \vee\ f\ q_2\ \vee\ \cdots \vee\ f\ q_N\ \vee\ f\ y$
$=$　　形式的意味及び上記 (2)
　　some $\$x$ **in** q **satisfies** $f\ \vee\ (\Phi_Q\ f \vee\ \mathsf{false}\ (y))$
$=$　　帰納法の仮定
　　$(\Phi_Q\ f \vee \mathsf{false}\ q)\ \vee\ (\Phi_Q\ f \vee\ \mathsf{false}\ (y))$
$=$　　(PQ3)
　　$\Phi_Q\ f \vee\ \mathsf{false}\ (q, (y))$

□

every 式についても同様に適用され,

$(\textbf{every}\ \$x\ \textbf{in}\ q\ \textbf{satisfies}\ f) = (\Phi_Q\ f \wedge\ \mathsf{true}\ q)$

が示される. □

1.2　変換規則の証明

[TR 1] (**where**↔**if**)

　　for $\$x$ **in** q **where** f **return** g
$=$ **for** $\$x$ **in** q **return if** (f) **then** g **else** ()　　(TWIF)

(証明)　形式的意味における正規化規則による. □

[TR 2] (モナド left unit 則 (for 式)(TFLU))

(証明)　(PQ2) と 定理 19 による. □

[TR 3] (モナド left unit 則 (限量式)(TSLU),(TELU))

(証明)　(PQ2) と定理 20 による. □

[TR 4] (モナド right unit 則 (for 式)(TFRU))

(証明)　$q = (q_1, q_2, \ldots, q_N)$ のとき，形式的意味より

(**for** \$$x$ **in** q **return** \$$x$) $= (q_1, q_2, \ldots, q_N) = q$ □

[TR 5] (分配則 (限量式)(TSD),(TED))

(証明)　(PQ3) と定理 20 による． □

[定理 21] (Φ_Q 結合則)

$\Phi_Q\ f\ op\ e\ (\Phi_Q\ g\ ,\ ()\ q)$
$= \Phi_Q\ (\Phi_Q\ f\ op\ e\ g)\ op\ e\ q$　(TPA)

(証明)

左辺
= 　Φ_Q の定義
$\Phi_Q\ f\ op\ e\ (g\ q_1, g\ q_2, \ldots, g\ q_N)$
= 　(PQ3)
$(\Phi_Q\ f\ op\ e\ (g\ q_1))\ op\ (\Phi_Q\ f\ op\ e\ (g\ q_2))\ op$
$\ldots op\ (\Phi_Q\ f\ op\ e\ (g\ q_N))$
= 　(PQ3) (右辺から左辺)
$\Phi_Q\ (\Phi_Q\ f\ op\ e\ g)\ op\ e\ q$

g は右辺では外側の Φ_Q による変数束縛のスコープ内であるが，左辺のそれは外にある．よって g には外側の Φ_Q で束縛される変数を含んではならない (若しくは名前の付け変えを行う)． □

[TR 6] (モナド結合則 (for 式)(TFA))

(証明)　(TPA) における op と e をそれぞれ , と () で置き換える． □

[TR 7] (限量式の結合則 (TSA),(TEA))

(証明)　(TPA) の op と e を，**some** については $\vee$ と false で，**every** については $\wedge$ と true でそれぞれ置き換える． □

[TR 8] (if/1 (TIF1))

(証明)

左辺
=
if (e_1 **and** e_2)
then (**if** (true)
　　then (**if** (true) **then** e_3 **else** e_4)
　　else e_4)
else (**if** (e_1)
　　then (**if** (false) **then** e_3 **else** e_4)
else e_4)
= 　**if** の定義
if (e_1 **and** e_2)
then e_3
else (**if** (e_1) **then** e_4 **else** e_4)
= 　**if** の性質
if (e_1 **and** e_2) **then** e_3 **else** e_4
=
右辺

□

[TR 9] (if/4 (TIF4))

(証明)

左辺
=
if (e_1) **then** f(**if** (true) **then** e_2 **else** e_3)
　　else f(**if** (false) **then** e_2 **else** e_3)
= 　**if** の定義
if (e_1) **then** $f(e_2)$ **else** $f(e_3)$
=
右辺

□

[TR 10] (**where** 節付きの **for** 式の left unit 則)

for \$$x$ **in** y **where** f **return** g
=**if** ($f[y/\$x]$) **then** $g[y/\$x]$ **else** ()　(TWFLU)

この変換規則は他の規則の証明の簡略化のために導入したものである．

(証明)　シングルトン y について，

左辺
= 　(TWIF)
for \$$x$ **in** y **return if** (f) **then** g **else** ()
= 　(TFLU)
右辺

□

[TR 11] (where 節のある for の結合則 (TWFA))

(証明)

左辺
=
for \$$x$ **in** (**for** \$$y$ **in** q
　　where e
　　return f))
where g
return h
= 　(TWIF)
for \$$x$ **in for** \$$y$ **in** q
　　return if (e) **then** h **else** ()
return if (g) **then** h **else** ()
= 　(TFA)

for $y **in** q
return for $x **in** (**if** ($e$) **then** h **else** ())
return if (g) **then** h **else** ()

= (TIF4)

for $y **in** q
return if (e)
then (**for** $x **in** h
return if (g) **then** h **else** ())
else (**for** $x **in** ()
return if (g) **then** h **else** ())

= 空入力の **for**

for $y **in** q
return if (e)
then (**for** $x **in** h
return if (g) **then** h **else** ()
else ()

= (TWIF)

for $y **in** q
where e
return for $x **in** h
where g
return h

=

右辺

□

[TR 12] (filter/0)

for $x **in for** $y **in** q **where** g **return** $y
where h
return f

= **for** $x **in** q **where** (g[$x/$y] **and** h) **return** f

(TF0)

この変換規則は h によるフィルタリングを引き上げる．なお，この変換規則も他の規則の証明の簡略化のために導入したものである．

(証明)

左辺

= (TWFA)

for $y **in** q
where g
return for $x **in** $y **where** h **return** f

= (TWFLU)

for $y **in** q
where g($y)
return if ($h[$y/$x]) **then** $f[$y/$x] **else** ()

= (TWIF)

for $y **in** q
return if (g)
then (**if** (h) **then** f **else** ())
else ()

= (TIF1)

for $y **in** q
return if ($g **and** h))
then f
else ()

= (TWIF)(逆方向)

for $y **in** q
where ($g **and** h)
return f

=

右辺

□

[TR 13] (filter/1(TF1))

(証明)

左辺

=

for $x **in for** $y **in** q
where g
return $y
where f
return $x

= (TF0)

for $y **in** q **where** ($g **and** f[$y/$x]) **return** $y

=

右辺

□

[TR 14] (filter/2(TF2))

(証明)

左辺

=

for $x **in for** $y **in** q **where** g **return** $y
return f

= **where** 導入

for $x **in for** $y **in** q **where** g **return** $y
where true
return f

= (TF0)

for $y **in** q **where** ($g **and** true) **return** f[$y/$x]

= **and** の性質

for $\$y$ **in** q **where** g **return** $f[\$y/\$x]$

=

右辺

□

[TR 15] (filter/3(TF3))

(証明)

左辺

=

for $\$x$ **in for** $\$y$ **in** q **return** f
where g
return $\$x$

= (TWFA)

for $\$y$ **in** q
return for $\$x$ **in** f
where g
return $\$x$

=

右辺

□

[TR 16] (filter/5(TF5))

(証明)

左辺

=

for $\$x$ **in** (**for** $\$y$ **in** q **where** g **return** $\$y$)
where f
return $\$x$

= (TF1)

for $\$y$ **in** q
where (g **and** ($f[\$y/\$x]$))
return $\$y$

= **and** の性質

for $\$y$ **in** q
where ($f[\$y/\$x]$ **and** g)
return $\$y$

= (TF1) (右辺 → 左辺)

for $\$x$ **in for** $\$y$ **in** q **where** $f[\$y/\$x]$ **return** $\$y$
where $g[\$x/\$y]$
return $\$x$

=

右辺

□