5M-05

XMLStream処理のモデル化と検索言語の設計

松田 達希† 内田 友樹† 藤田 悟‡

法政大学大学院 情報科学研究科情報科学専攻† 法政大学 情報科学部‡

1. まえがき

近年, ストリームデータを高速に処理・分析するための複合イベント処理(CEP)の研究が盛んに行われている. CEP で扱うストリームデータの形式は様々であるが, 我々はその中でも XML 形式のデータに注目し, XML ストリーム向けの CEP エンジンの開発を行っている. XML は構造化されたデータであるため, データの検索や取得が容易である. しかしながら, 従来の XML 向けの問い合わせ言語では, 時々刻々と連続的に発生し変化していくストリームデータに対する時系列を踏まえた分析に適していない. そこで, 本稿では従来の単純な処理に加え, 時系列を考慮した分析や複数のストリームに渡った処理の流れのモデル化を行う.

2. 関連技術

XSeq は XPath を拡張した, XML ストリームデータ向けの問い合わせ言語である[1]. 時系列上に発生する XML に対する検索要求を記述することが可能であり, Visibly Pushdown Automaton(VPA)[2]を利用することで, 高速な処理も実現している. しかしながら, XSeq で扱えるのはシングルストリームに限られている.

Cayuga はマルチストリームに対応した検索言語及び CEP システムである[3]. 様々な形式のストリームデータに対応させることが可能であり, XML ストリームに特化したシステムではない.

3. マルチストリーム処理

マルチストリームに対する分析処理を, 代数表現を用いてモデル化する.

(1) Filtering

特定の条件を満たすストリームデータをフィルタリングし, 新たなストリームに出力する. 例えば, ストリーム:s に対してフィルタ:f をかけて s'として出力する処理は, 以下のように記述する.

s | f >> s' … (1)

(2) Combination

2 つのストリーム中のデータを別のストリームにまとめて出力することで, 2 ストリームを合流・結合させる. この処理は結合演算子「+」を用いて以下のように記述する.

s1 + s2 >> s3 … (2)

結合演算では交換法則と結合法則が成り立つ.

(3) Activation

あるストリームが別のストリーム処理を呼び起こし, 結果を合流させる. この処理は合成演算子:「*」を用いて以下のように記述できる.

s1 * s2 >> s3 … (3)

合成演算では, ストリームデータの到着順序は s1 が先, s2 が後という意味が含まれているため, 交換法則や結合法則は成り立たない.

(4) Decomposition

ストリーム中のデータを, 指定されたキー単位で分割し, 新たなストリームに出力する. キーは XML においてはタグに相当するものである. 例えば, ストリームデータを, 'key'をキーとして分割し出力する処理は分割演算子:「/」を用いて以下のように記述する.

s / key >> s' … (4)

(5) Partitioning

ストリーム中のデータを, 指定されたキーの値毎にまとめてストリームの配列として出力する. 'key'をキーとして処理する場合は以下のように記述する.

s[key] >> s'[] … (5)

特定の値を持つストリームを抽出する場合は, 以下のようにキーの値を指定してシングルストリームを生成する.

s'[key='value'] >> s'' … (6)

4. QLMXS

4.1. 問い合わせ言語

QLMXS は 3 節のマルチストリーム処理モデルに沿って設計された問い合わせ言語である[4]. 特に, XML ストリームデータに対して SQL ライクに検索要求が記述できるように設計した. 主に出力ストリームを指定する return 節, 出力フォーマットを指定する select 節, 入力ストリームを指定する from 節, フィルタ条件を記述する where 節等から成る.

例として, 株式売買システムより各時刻における各企業の株情報が XML ストリームデータとして流れてくるような状況を想定する.

```
Example1.  return  MyStock
           select  *
           from    Stock
           where   stock[price/text()>=100
                         and @date='2014-12-01']
```

Example1 は, 株情報のストリームから, 売値が 100 以上且つ日付が'2014-12-01'であるようなデータを新たなストリームへと出力するためのクエリである. return 節で出力ストリームを MyStock とし, select

---

Modeling and Designing Query Language for Multiple XML Streams

† Tatsuki Matsuda, Yuki Uchida, Graduate School of Computer and Information Sciences, Hosei University

‡ Satoru Fujita, Faculty of Computer and Information Sciences, Hosei University

節で出力形式を「*」, 即ち元のまま出力するように指定している. from 節では入力ストリームを Stock ストリームに指定し, where 節でそのストリーム中のデータに対するフィルタ条件を XPath に準ずる記法で記述している. Exmaple1 では, price タグの値が 100 以上且つ stock タグの属性 date が'2014-12-01'であるものを抽出するよう指定している.

マルチストリーム処理の中でも, Activation 処理を実現する場合, from 節では 1 つの入力ストリームしか指定できないようにしているため, 代わりに chaining 節を利用する. 使用例を Example2 に示す.

```
Example2.  return      MyStock2
           select      mystock[$1/stock, $2/stock]
           chaining    Stock1, Stock2
           where       [$1/stock/price/text()
                              = $2/stock/price/text()]
```

Exmaple2 は, 2 つの株情報ストリームから, 売値が等しいものを 1 つのデータにまとめて新たなストリームに出力するためのクエリである. chaining 節で 2 つの入力ストリームを指定しており, この場合, Stock1 が Stock2 のストリーム処理を呼び起こす形になっている. また, select 節では MyStock2 に出力する XML の形式を指定しており, Exmaple2 の場合は mystock がルート要素, その下に Stock1 と Stock2 の stock 以下の要素が格納される形式になっている. 1 入力目である Stock1 に対するパスを記述する際は $1 を, 2 入力目である Stock2 の場合は$2 をそれぞれパスの先頭に加える必要がある.

さらに, QLMXS では繰り返し処理の記述も可能であり, while 節や processing 節を用いて指定した時間の間繰り返すような処理を記述できる. 例を Example3 に示す.

```
Example3.  return       MyStock3
           select       mystock[$1/stock,
                          @ave = $sum div $cnt]
           chaining     Stock1, Stock2
           setting      $sum = 0, $cnt = 0
           processing   $sum = $sum +
                              $1/stock/price/text(),
                        $cnt = $cnt + 1
           where        [$1/stock/price/text() >=
                              $2/stock/price/text()]
           while        10min
```

Example3 は Stock1 の売値が Stock2 の売値よりも高くなっている間の Stock1 の平均売値と, 低くなった直後の株データを抽出するクエリである. setting 節では変数宣言を行うことができ, processing 節では where 節の条件が満たされたときに実行される処理を記述できる. while 節では最大処理継続時間を記述することができ, 処理が開始されてから指定された最大処理継続時間を超えた場合は処理を終了する.

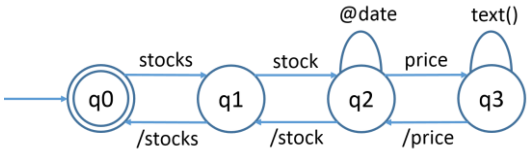

図 1　生成される VPA の簡易例

5. QLMXS を実装した CEP エンジン

著者らが開発している CEP エンジンでは, QLMXS クエリを解釈して生成した VPA を検索エンジンのコアとして利用している[5]. VPA はプッシュダウンオートマトンの制約を強めたものであり, スタックを持ち, スタックトップを参照しながら遷移先を決定し遷移していく. よって, VPA の生成には, 状態の集合や遷移するためのシンボル集合, 遷移時にスタックに格納されるシンボルの集合, そして各遷移関数が必要となる. 基本的には select 節や where 節中のパス形式による記述箇所からタグの親子関係を判別することで, 遷移シンボルや必要となるスタックシンボルの数等を決定し VPA を構築していく. 例として, Example1 を解釈した結果生成される VPA の簡略版を図 1 に示す.

6. 考察

QLMXS は, モデル化したマルチストリーム処理に沿った設計を行ったことで, XSeq ではできなかったマルチストリーム向けの網羅的な記述が可能となった. また, XPath に準じた記法を採用することで, 同じくマルチストリーム処理システムである Cayuga に比べ, より XML に特化した言語となっている.

7. まとめ

本稿では, マルチストリーム処理の代数的モデル化, マルチ XML　ストリーム向け検索言語:QLMXS の設計を行った. QLMXS により複数のストリームに跨った連続的問い合わせが比較的容易に記述可能となった. 今後の課題として, QLMXS から生成される VPA の最適化が挙げられる.

参考文献

[1] Mozafari B., Zeng K., Zaniolo C., "High-performance complex event processing over xml streams", Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, p.253-264 (2012).

[2] Alur R., Madhusudan P., "Visibly pushdown languages", Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, p.202-211 (2004).

[3] Demers A., Gehrke J., Panda B., Riedewald M., に Sharma V., White W., "Cayuga: A General Purpose Event Monitoring System", CIDR, Vol.7, p.412-422 (2007).

[4] 松田達希, 内田友樹, 藤田悟, “XMLStream 向け検索言語からの VPA の生成”, FIT2014, (2014)

[5] 内田友樹, 松田達希, 藤田悟, “VPA を用いた XMLStream 向け CEP エンジン”, FIT2014, (2014)