携帯機器向け XPath 処理系の設計と構築

宮坂 集策 † 石川 佳治 †† 北川 博之 †† 村山 敏清 §
† 筑波大学第三学群情報学類 †† 筑波大学電子・情報工学系 § ネットジーン

1 研究の背景と目的

XML はネットワーク時代の情報交換・流通のための汎用データフォーマットとして，携帯電話や PDA 機器などの携帯機器上での利用も期待されているが，携帯機器上での XML データ処理では，メモリ量や CPU パワーなど，利用できる計算機資源の制限が大きいという問題がある．携帯機器の処理能力も年々向上してはいるが，現時点ではメモリの利用をできるだけ抑えるようなプログラム開発技術が求められているのが現状である．

XML 処理のための API として，シーケンシャルに XML データを処理する SAX [1] がある．SAX は DOM のようにメモリ上に木を展開することはないため，要求するメモリ量が小さく，携帯機器上での XML 処理に適していると考えられる．しかし，SAX は軽量である反面，イベントベースの低レベルの記述が求められることから，プログラミングが煩雑になるだけでなく，プログラムの挙動が把握し難くバグが出易いなどの問題がある．特に，正規表現を用いたパスで指定される XML 文書要素を取得する場合などには，複雑な処理が求められることが多い．

以上の背景をもとに，本研究では資源の面で制約が大きい携帯機器上での組込みアプリケーションの開発を支援するためのツールとして，C 言語のプリプロセッサとしてはたらく XPath 処理系 **XPets**（XPath Embedding Tool over SAX）の開発を行っている．XPets は，C 言語のコメント行に埋め込まれた XPets 構文（XPath 式などの呼出しを含む）により指定された XML 文書要素の選択・抽出処理を，SAX のイベント処理を行う C 言語フラグメントに自動的に展開して元の C 言語プログラムに埋め込みする．これにより，組込み機器のプログラマは，XML の文書要素の取得のための煩雑なパターンマッチのプログラムを直接記述する必要がなくなり，組込みアプリケーション開発における労力を削減することができる．また，記述された XPath 式は効率的に処理可能な SAX のイベント処理へと変換されるため，処理コストの削減という点でも利点を有する．

Design and Development of an XPath Processing Tool for Mobile Devices
Shusaku Miyasaka†, Yoshiharu Ishikawa††, Hiroyuki Kitagawa††, and Toshikiyo Murayama§
† College of Information Sciences, Univ. of Tsukuba
†† Inst. of Inf. Sci. and Elec., Univ. of Tsukuba
§ Netgene Co. Ltd.

2 XPets 言語

XPath 言語 [2] は，XML 文書要素の選択のための言語であり，XML データ処理で広く用いられている．本稿で述べる XPets 言語は XPath 言語のサブセットを内包する言語であり，C 言語のコメント行に記述する形態をとる．

XPath 式の指定とハンドラ関数 XPets の処理系は，C（C++）言語のコメント行中で '@' で始まる行を XPets の構文として解釈する．XPets の基本となる構文は以下に示すパス構文である．

```
// @item = xpath("/catalog//item[@price < 10000]")
```

この文は，文字列として与えられた XPath 問合せを変数 item に束縛しており，XPets 処理系により，SAX のイベントを処理する C 言語のフラグメントに展開される．C 言語のフラグメントは，次々に発生する SAX イベントの内容を調べ，与えられた XPath 式にマッチする XML 文書要素に出会ったとき，

```
int *handler_item(void *user_data,
                  const char *name,
                  const char **attrs);
```

というハンドラ関数を呼び出す．ハンドラ関数名は “handler_変数名” という名前を持ち，name は要素名，attrs は要素名と要素値のシーケンス，user_data はユーザ定義データへのポインタである．ハンドラ関数を C プログラム中に記述することで，XPath 式を満たす要素に対してのみ適用される処理が記述できる．

以下のように XPath 式を複数に分割して書くことも可能である．

```
// @item = xpath("/catalog//item")
// @softitem = xpath("$item[@category = 'software']")
// @cheapitem = xpath("$item[@price < 10000]")
```

2，3 行目の $item は，1 行目のパスの値を表すため，/catalog//item[@category = "software"] という XPath 式が 2 行目では表現されている．なお，上の例では，実行時に handler_item()，handler_softitem()，handler_cheapitem() の 3 つのハンドラ関数が呼び出しされることになる．

外部関数機能 XML にはさまざまな応用があるため，XPath 式の条件記述の中でアプリケーションに固有のフィルタリング条件を用いることができれば，プログラム開発効率が向上すると考えられる．そのため，XPets 言語では外部関数の定義・利用機能を提供する．

例として以下のような地図データの XML 表現があるとする．poidata 要素は，地物オブジェクトを表す poi 要素のシーケンスを含んでいる．

```
<poidata>
 <poi longitude="140.0622" latitude="36.1320">
  <name>ABC Hotel</name>
  <address>...</address>
  ...
 </poi>
 ...
</poidata>
```

このような XML データの中から,「ある地点から 100m 以内にある地物を取り出せ」という選択をしたいとする．このときには，外部関数 dist() を用いて

```
/poidata/poi[dist(155000, 85000) < 100.0]
```

といった XPath 式を記述することとなる．dist() は，poi 要素の属性リストをもとに，与えられた地点からの距離を計算する外部関数である．

C プログラム中で上のような外部関数呼出しを含む XPath 式を用いるためには，

```
// @exfun float dist(int, int)
```

という外部関数の定義を記述しておく必要がある．XPets 処理系は，このような外部関数定義により外部関数の名前とシグネチャを把握し，C プログラム実行時に呼び出される外部関数の実体（C 言語で記述される）のシグネチャを決定する．上の例では，

```
float dist(int x, int y, void *user_data,
           const char *name, const char**attrs);
```

という関数が XPath 式の処理時に外部関数の実体として呼び出しされることになる．そのため，プログラマ側で上のような関数の実装を与えることになる．

変数の動的束縛　他の機能として，実行時に検索条件の値が決まるような状況下で用いる変数の動的束縛がある．以下は，動的に値が定まる price_str の内容を，変数 $pval に設定する例である．

```
char price_str[10];
/* price_str に動的に値を設定する処理が入る */
// @pval = price_str
// @item = xpath("/catalog//item[@price < $pval]")
```

3　実装方式

現在の XPets 処理系が扱う XPath 式は，'/', '//' によるロケーションステップと条件指定の組合せである．これは，メモリ効率と実行処理性能を考慮し，SAX を用いてシーケンシャルに処理できる範囲に問合せを限定した結果である．問合せ処理では，与えられた XPath 式から有限オートマトンを構築する方式をとる．

具体例として，以下の XPath 式が与えられたとする．

```
/catalog/products//item[@price < 10000]
```

XPets 処理系はこの問合せを構文解析し，まず，/catalog/products//item の部分を抜き出す．この式から，まず図 1 左の非決定性有限オートマトン（NFA）を作成する．図において，状態 0 は開始状態を表し，二重丸の状態は終了状態を表す．この図は，状態 0 から開始し，入力 XML データを SAX パーサで解析したとき，catalog という XML 要素に出会った場合，状態 1 に遷移することを表している．'*' は任意の XML 要素（catalog なども含む）を表す．SAX イベントの発生に応じて状態を遷移していき，終了状態に到達した場合に与えられたパス /catalog/products//item に達したことになる．

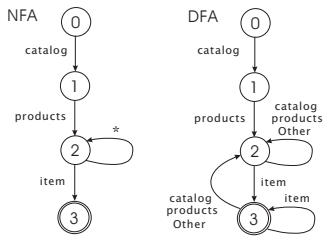

図 1: NFA と DFA

実際には，非決定性の遷移が含まれていると実装が容易でないため，図 1 右のように決定性オートマトン（NFA）に変換する [3]．この図において，Other は，XPath 式に現れる catalog, products, item 以外の任意の XML 要素を表す．XPets 処理系は，この NFA をもとに C 言語のフラグメントを生成する．C 言語フラグメントには，パス /catalog/products//item を満たす XML 要素に達した際，条件 "@price < 10000" を判定して，条件を満たす場合，対応するハンドラ関数を呼び出す処理を埋め込む．

4　まとめと今後の課題

XPets は，現在，Windows CE 環境上で稼動する SAX ライブラリ上で開発を進めている．XPath 式を SAX 上で評価する研究としては，XMLTK [4] などがあるが，組込みアプリケーションを想定したプリプロセッサ方式をとった点が本研究の特徴となっている．今後は，支援可能な XPath 式の拡張を行う予定である．

謝辞

本研究は, 筑波大学とネットジーンの共同研究「携帯情報機器に対応した XML 拡張問合せ処理系の開発」に基づく．また、文部科学省科学研究費（14019009 および 12480067）の助成による．

参考文献

[1] http://www.saxproject.org/

[2] http://www.w3c.org/TR/xpath

[3] J. ホップクロフト, J. ウルマン, オートマトン 言語理論 計算論 I, サイエンス社, 1984 年.

[4] I. Avlia-Campillo et al., XMLTK: An XML Toolkit for Scalable XML Stream Processing, *Proc. of PLAN-X*, 2002.