XML 문서의 자동 변환을 위한 XSLT 스크립트 생성

신동훈° 이경호
연세대학교 컴퓨터과학과
dhshin@icl.yonsei.ac.kr°, khlee@cs.yonsei.ac.kr

Generating XSLT Scripts for An Automated Transformation of XML Documents

Dong-Hoon Shin° Kyong-Ho Lee
Dept. of Computer Science, Yonsei University

요 약

본 논문에서는 XML 문서간의 자동 변환을 위한 효율적인 XSLT 스크립트를 생성하는 알고리즘을 제안한다. 제안된 방법은 XML DTD를 구성하는 단말 노드간의 일대일 대응관계가 주어져 있다는 가정 하에 XSLT 스크립트를 생성하는데 중점을 둔다. 제안된 알고리즘은 상향식과 하향식의 복합적인 접근 방식을 적용한다. 먼저 중간 노드간의 대응관계를 상향식으로 생성하며 하향식 깊이 우선 탐색을 적용하여 XSLT 스크립트를 생성한다. 실험결과, 제안된 방법은 기존 연구와 비교하여 XML 문서를 보다 빠르게 변환하는 XSLT 스크립트를 생성하였다.

1. 서 론

XML 문서가 인터넷을 비롯한 다양한 분야에서 정보 교환을 위한 표준으로 널리 사용되면서 XML 문서의 변환에 대한 필요성이 증가하고 있다. 본 논문에서는 XML 문서의 변환을 위해서 효율적인 XSLT 스크립트의 생성 방법을 제안한다. 본 논문에서 효율적인 XSLT 스크립트는 변환 수행 속도가 빠른 스크립트를 의미한다.

본 논문에서는 DTD(Document Type Definition)를 구성하는 엘리먼트(element)와 속성(attribute)을 효과적으로 표현하기 위한 문서 모델을 제안한다. 제안된 모델은 루트 노드를 포함하며 형제 노드 간에 순서가 존재하는 순서 트리에 기반한다. XML 문서의 엘리먼트와 문자열은 트리 구조의 노드를 구성한다. 문서 모델을 구성하는 노드는 엘리먼트와 속성을 표현하는 일반 노드와 내용 모델의 종류를 표현하는 내용 모델 노드로 나누어진다.

기존에 XML 문서 간의 변환을 위한 XSLT 스크립트를 생성하는 방법이 제안되었다. Kuikka 등 [1]과 Leinonen [2]은 사용자로부터 단말 노드에 대한 레이블 대응관계를 입력받아 중간 노드 대응관계를 생성하고 이를 바탕으로 XSLT 스크립트를 자동으로 생성한다. Tang과 Tompa [3]는 새로운 변환 기술 언어 Paired SynTrees를 제안하고 제안된 언어를 사용하여 기술한 변환 정보를 XSLT 스크립트로 변환한다. Pankowski [4]는 Tang과 Tompa가 제안한 방법에 착안하여 개념트리(Concept Tree)를 사용하는 새로운 변환 기술 언어 XDTrans를 제안하고 제안된 언어를 사용하여 기술한 변환 정보를 XSLT 스크립트로 변환한다. Su 등 [5]은 변환 연산과 비용모델을 사용하여 두 XML 스키마 간의 대응관계를 찾고 이를 바탕으로 XSLT 스크립트를 자동으로 생성한다.

기존의 연구들이 서로 완전히 같지는 않지만, XSLT 스크립트를 생성할 때 템플릿에 기반하고 있다는 것은 공통적이다. 사용자의 정의나 시스템의 계산으로 생성된 대응관계, 대응규칙, 혹은 부 구조들이 XSLT 스크립트를 생성할 때 각각 하나의 템플릿으로 변환된다.

· 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2003-003-D00429).

본 논문에서 제안하는 방법은 효율적인 XSLT를 생성하기 위해 기존의 방법들보다 적은 수의 템플릿으로 XSLT 스크립트를 생성하는데 중점을 둔다. 단, 입력되는 두 DTD에 대해 단말 노드간의 일대일 대응관계가 주어져 있다고 가정하며, 재귀적인 구조를 가지는 문서의 변환은 고려하지 않는다.

2. XSLT 스크립트 생성 알고리즘

제안된 알고리즘은 중간 노드의 대응관계 생성과 XSLT 스크립트 생성의 두 단계로 구성된다. 첫 번째 단계에서는 대응관계를 갖는 단말 노드의 경로를 비교하여 중간 노드 대응관계를 생성한다. 두 번째 단계에서는 중간 노드 대응관계를 기반으로 타겟 DTD에 하향식 깊이 우선(depth-first) 탐색 과정을 적용하여 XSLT 스크립트를 생성한다.

2.1 중간 노드 대응관계 생성

제안된 방법은 단말 노드 간의 일대일 대응 관계가 주어져 있다고 가정하고, 각각의 단말 노드 대응관계에 대해 상향식 방법으로 경로를 비교하여 중간 노드 대응관계를 생성한다. '*'가 붙어있는 중간 노드들만을 대상으로 대응관계를 생성하며 조상 순서를 위배하는 대응관계는 생성하지 않는다. 중간 노드 간의 대응관계 생성을 위해서 Kuikka 등의 방법에서 정의한 TAN(Terminating Associated Nonterminals) 집합 개념을 사용한다. 제안된 중간 노드 대응관계 생성 알고리즘은 <그림 1>과 같다.

제안된 방법은 소스와 타겟 DTD 트리에서 '*'가 붙어있는 중간 노드 간의 대응관계만을 고려한다. 단말 노드까지의 경로 상에 '*'가 붙어있는 중간 노드가 없다면, 문서에서 단말 노드는 한 번만 나타난다. 따라서 대응관계를 갖는 두 단말 노드의 경로 상에 '*'가 붙어있는 중간 노드가 없는 경우, 중간 노드 대응관계를 생성할 필요가 없다. 주어진 단말 노드 간의 대응관계를 사용하여 단말 노드간의 값을 복사하는 것만으로 변환이 가능하다. 또한 한쪽의 경로에만 '*'가 붙어있는 노드가 있는 경우에도 중간 노드 대응관계를 생성할 필요가 없다. 한쪽 단말 노드가 문서에서 한번밖에 나타나지 않기 때문에, 용량(capacity)차이로 인해 하나의 정보밖에 변환될 수 없다. 이 경우도 중간노드 대응관계 생성 없이 단말 노드간의 값 복사만으로 변환이 가능하다. '*'가 붙어있는 노드 간에 생성되는 대응관계는 XSLT 스크립트에서 for-each 문을 포함하여 반복되는 구조를 지원한다.

```
입력: 소스 DTD 트리, 타겟 DTD 트리, 단말 노드 대응관계 집합
출력: 중간 노드 대응관계 집합

· Ps, Pt : 루트 노드로부터 단말 노드 Ns, Nt 까지의 경로.
· Nss, Nts : Ps, Pt 상에 존재하는 '*'가 붙어있는 중간노드.
· Css, Cts : Nss와 Nts의 개수.
· Nss(i), Nts(j) : Ps, Pt를 단말 노드로부터 루트 노드까지 탐색할
                   경우 i번째와, j번째에 만나게 되는 Nss와 Ntt.
                   이때, 1≤i≤Css, 1≤j≤Cts.
· Cti(n1, n2) : TAN(n1)과 TAN(n2)에서 서로 대응관계에 있는
                원소들의 개수.
· level(n) : 전체 DTD 트리에서 노드 n의 level.

1:  while(단말 노드 대응관계 집합이 공집합이 아니다.)
2:  {
3:      단말 노드 대응관계 집합으로부터,
        대응관계 (Ns, Nt)를 꺼낸다;
4:      Ps, Pt, Css, Cts를 계산한다;
5       k = 1;  //선택 가능한 타겟 중간 노드를
                  첫 번째 노드로 초기화 한다.
6:      for(i=1; i≤Css; i++)
7:      {
8:        k≤j≤Cts 인 Nts(j)중에
          Cti(Nss(i), Nts(j))가 최대인 Nts(j)를 선택한다;
9:        조건을 만족하는 Nts(j)가 두 개 이상인 경우,
          level(Nts(j))가 가장 큰 Nts(j)를 선택한다;
10:       if( TAN(Nts(j))와 TAN(Nss(i))가 일대일 대응이다. )
11:       {
12:         (Nts(j), Nss(i))를 중간 노드 대응관계 집합에 추가한다;
13:         k = j+1; //조상 순서를 유지하도록, 다음번에
                       선택 가능한 타겟 중간 노드를
                       현재 노드의 상위 노드로 설정한다.
14:       }
15:       else
16:       {
17:         TAN(Nss(i))의 원소들 중에서 TAN(Nts(j))와
            대응관계를 갖지 않는 모든 노드 n에 대해,
18:         if( n과 대응하는 단말 노드의 경로 상에 '*'가
              붙어있는 중간 노드가 적어도 하나 존재한다. )
19:         {
20:           TAN(Nts(j))의 원소들 중에서 TAN(Nss(i))와
              대응관계를 갖지 않는 모든 노드 n'에 대해,
21:           if( n'과 대응하는 단말 노드의 경로 상에 '*'가
                붙어있는 중간 노드가 적어도 하나 존재한다. )
22:           {
23:             (Nts(j), Nss(i))를 중간 노드 대응관계 집합에
                추가한다;
24:             k = j+1;
25:           }
26:         }
27:       }
28:   }
29: }
```

그림 1. 중간 노드 대응관계 생성 알고리즘.

제안된 방법은 상향식 방법으로 소스 중간 노드들을 탐색하면서, 각각의 중간 노드에 대해 대응관계를 생성할 수 있는 타겟 중간 노드가 존재하는지 조사한다. 임의의 소스 중간 노드 n이 타겟 중간 노드 n'과 대응관계를 가질 때, n의 상위 중간 노드가 n'의 하위 중간 노드와 대응관계를 갖는 경우, 조상 순서를 위배하게 된다. 이러한 조상 순서를 위배하는 대응관계가 생성되지 않도록, 제안된 방법은 소스 중간 노드 n이 이미 대응관계가 생성된 타겟 중간 노드의 상위 노드에 대해서만 대응관계 생성 가능 여부를 조사하도록 한다.(<그림 1>의 5, 13 그리고 24줄 참조).

임의의 두 노드에 대해, TAN 집합이 서로 일대일 대응관계를 갖는 경우, 두 노드는 의미상 같은 노드로 간주될 수 있다. 따라서 제안된 방법은 TAN 집합이 서로 일대일 대응관계를 갖는 경우, 두 노드 간에 중간 노드 대응관계를 생성한다.(<그림 1>의 10~14줄 참조).

TAN 집합이 일대일 대응관계를 갖지 않는 경우에도 중간 노드 간에 대응관계가 생성될 수 있다. 이러한 경우, 문맥적 의미를 고려하여 대응관계를 생성한다. 임의의 소스 중간 노드 n과 타겟 중간 노드 n'에 대해 TAN 집합이 서로 일대일 대응관계를 갖지 않는 경우에도, TAN(n)의 노드 중에 대응관계를 갖지 않는 모든 노드가 타겟 문서에서 반복되어 나타날 수 있고, TAN(n')의 노드 중에 대응관계를 갖지 않는 모든 노드가 소스 문서에서 반복되어 나타날 수 있는 경우, 중간 노드 간에 대응관계를 생성한다.(<그림 1>의 15~27줄 참조).

2.2 XSLT 스크립트 생성

소스 DTD 트리와 타겟 DTD 트리 간에 중간 노드 대응관계가 생성되면 XSLT 스크립트 생성은 비교적 쉽게 이루어진다. 제안된 XSLT 스크립트 생성 알고리즘은 <그림 2>와 같다.

```
입력: 소스 DTD 트리, 타겟 DTD 트리, 단말 노드 대응관계 집합,
      중간 노드 대응관계 집합
출력: XSLT 스크립트

1. 소스 DTD tree의 루트 노드에 대한 템플릿을 연다.
2. 타겟 DTD tree를 깊이 우선 탐색 하면서 각각의 노드들에 대한
   태그를 열거나 닫는다.
   대응관계를 갖는 중간 노드들은 대응되는 소스 중간 노드들에
   대한 for-each 태그를 먼저 생성한다.
   대응관계를 두 개 이상 갖는 중간 노드는 대응되는 소스 중간
   노드들 중에서 가장 적게 반복되는 노드에 대한 for-each 태그
   를 먼저 생성한다.
   대응관계를 갖는 단말 노드인 경우 3을 수행한다.
   모든 노드에 대한 탐색이 끝났다면 4를 수행한다.
3. 소스 단말 노드와 타겟 단말 노드에 모두 '*'가 붙어있으면
   먼저 소스 단말 노드에 대한 for-each 태그를 생성한다.
   타겟 단말 노드에 대한 태그를 연다.
   소스 단말 노드에 대한 value-of 태그를 생성한다.
   타겟 단말 노드에 대한 태그를 닫는다. 2를 수행한다.
4. 소스 DTD tree의 루트 노드에 대한 템플릿을 닫는다.
```

그림 2. XSLT 스크립트 생성 알고리즘.

제안된 방법은 타겟 DTD의 루트 노드에 대한 템플릿을 생성하고, 타겟 DTD 트리를 깊이 우선 방식으로 탐색하면서 해당 노드에 대한 태그를 생성한다. 대응관계를 갖는 중간 노드들은 대응되는 소스 중간 노드 for-each 태그를 먼저 생성한다.

단말 노드 대응관계 집합에 속해있는 각각의 대응관계에 대해 경로를 비교하여 중간 노드 대응관계를 생성하기 때문에, 타겟 DTD 트리에서 2개 이상의 대응관계를 갖는 중간 노드가 존재할 수 있다. 하나의 노드와 2개 이상의 대응관계를 갖기도 하고, 서로 다른 노드들과 2개 이상의 대응관계를 갖기도 한다.

하나의 노드와 2개 이상의 대응관계를 갖는 경우는 단순하게 대응되는 소스 중간 노드에 대한 for-each 태그를 사용하여 반복되는 소스 중간 노드를 타겟 중간 노드로 변환한다. 타겟 중간 노드가 2개 이상의 소스 중간 노드와 대응관계를 갖는 경우, 대응되는 소스 중간 노드들 중에서 문서에서 가장 적게 반복되는 중간 노드를 선택하여 for-each 태그를 생성한다.

3. 성능분석

제안된 방법의 성능을 평가하기 위하여 Kuikka 등의 방법, Tang 등의 방법과 비교 실험을 하였다. 이때 실험에 사용된

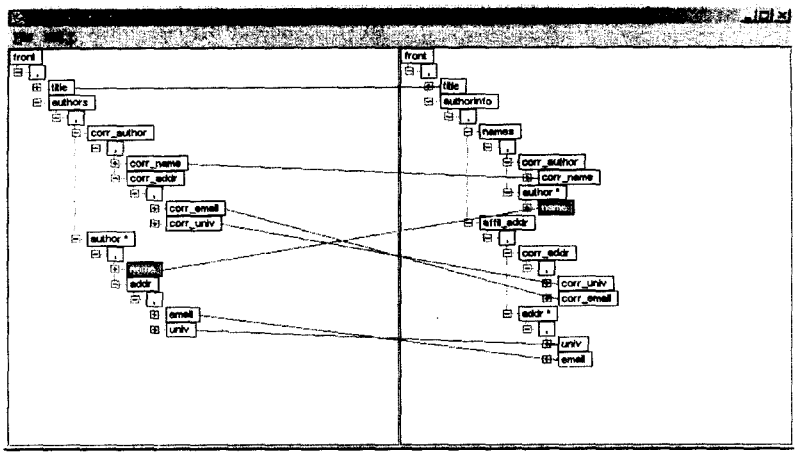

그림 3. 주어진 단말 노드 대응관계

DTD와 단말 노드 대응관계는 <그림 3>과 같다.

제안된 방법은 템플릿 수를 줄임으로써, 생성되는 XSLT 스크립트의 수행 속도를 향상시키는데 목적을 두고 있다. 따라서 전술한 방법들에 의해 생성된 XSLT 스크립트의 변환 수행시간이 문서의 크기에 따라 어떻게 변화하는지를 측정하였다.

주어진 소스 DTD를 따르도록 9개의 서로 다른 크기의 문서를 임의로 작성하였고, 한 문서에 대해 각각의 방법에 의해 생성된 XSLT 스크립트를 10번 적용하여 수행시간을 측정한 후 평균을 계산하였다. 실험 결과는 <그림 4>와 같다. 실험 결과, 제안된 방법이 생성한 XSLT 스크립트가 XML 문서를 가장 빠르게 변환하였다.

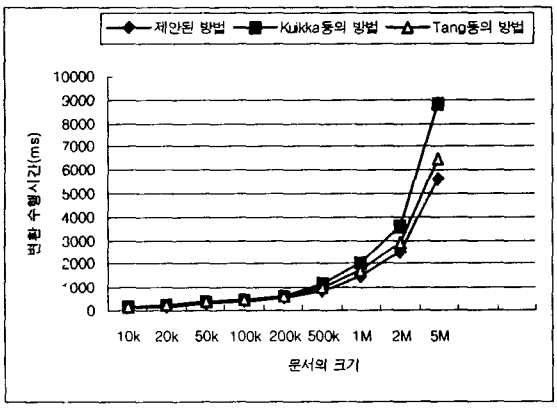

그림 4. 문서 크기에 따른 XSLT 스크립트의 변환 수행속도

4. 결론 및 향후연구

본 논문에서는 단말 노드 간의 일대일 대응관계가 주어져 있다고 가정하고 효율적인 XSLT 스크립트를 생성하는 알고리즘을 제안하였다. 주어진 단말 노드 대응관계를 사용하여 중간 노드 간의 대응관계를 생성한 후 이를 바탕으로 XSLT 스크립트를 생성한다. 이때, 변환 수행속도를 향상시키기 위하여 하나의 템플릿만을 사용하여 XSLT 스크립트를 생성한다.

기존의 방법들은 중간 노드 대응관계나 대응규칙의 수와 비례하는 수의 템플릿을 사용하여 XSLT 스크립트를 생성하였다. 그러나 제안된 방법은 대응관계의 수와 관계없이 하나의 템플릿을 사용하여 XSLT 스크립트를 생성한다. 같은 DTD를 따르는 서로 다른 크기의 문서들을 대상으로 한 실험에서 제안된 방법에 의해 생성된 XSLT 스크립트는 기존의 방법에 의한 XSLT 스크립트보다 빠른 변환 수행속도를 보였다. 따라서 제안된 방법은 XML 문서의 변환을 위한 효율적인 XSLT 스크립트를 생성한다고 할 수 있다.

한편, 제안된 방법은 단말 노드 간의 일대일 대응관계를 가정하고 있고, 재귀적인 구조를 가지는 문서를 고려하지 않는다. 일반적인 문서에서는 병합(merge)나 분할(split)과 같은 다대일이나 일대다의 대응관계가 존재할 수 있고 재귀적인 구조를 가지기도 한다. 현재는 이러한 문서들에 대해 올바른 변환을 수행하지 못한다. 본 논문은 제한적인 문서들에 한해서 기존의 방법들보다 더욱 빠른 수행속도를 갖는 XSLT 스크립트를 생성하는데 초점을 맞추었다.

향후에는 전술한 다대일이나 일대다의 대응관계가 주어진 경우와 재귀적인 구조를 가지는 문서에 대해서도 효율적인 XSLT 스크립트를 생성할 수 있는 방법을 연구할 것이다.

5. 참고문헌

[1] Eila Kuikka, Paula Leinonen, and Martti Penttonen, "Towards Automating of Document Structure Transformations," Proc. ACM Symposium on Document Engineering, pp. 103-110, McLean, Virginia, USA, Nov. 2002.

[2] Paula Leinonen, "Automating XML Document Structure Transformations," Proc. ACM Symposium on Document Engineering, pp. 26-28, Grenoble, France, Nov. 2003.

[3] Xuerong Tang and Frank Wm. Tompa, "Specifying Transformations for Structured Documents," Proc. 4th Int'l Workshop on the Web and Databases(WebDB), pp. 67-72, 2001.

[4] Tadeusz Pankowski, "Specifying Transformations for XML data," Proc. 29th Int'l Conf. Very Large DataBases, Berlin, Germany, Sep. 2003.

[5] Hong Su, Harumi Kuno and Elke A. Rundensteiner, "Automating The Translation of XML Documents," Proc. 3rd Int'l Workshop on Web Information and Data Management (WIDM), pp. 68-75, Atlanta, Georgia, Sep. 2001