검증 규칙을 포함한 XML 문서

남철기[0] 양재군 배재학
울산대학교 컴퓨터·정보통신공학부
cholki.nam@oracle.com

XML Documents Including Validation Rules

Chul-Ki Nam[0] Jae-Gun Yang Jae-Hak J. Bae
School of Computer Engineering and Information Technology
University of Ulsan

요 약

본 논문에서는 XML 문서에 문서검증을 위한 로직을 추가하였다. 로직을 기술하기 위해 규칙 표현에 효과적인 Prolog를 이용하였고 검증 규칙을 XML 문서로 변환하여 원래의 XML 문서에 포함시켰다. XML 문서검증에 관한 연구가 기존에는 주로 문서 구조의 검증에 치중한 반면 본 논문은 XML 엘리먼트의 데이터 값 검증에 역점을 두었다. 또한 폼 값의 검증을 XML과 Prolog를 이용하여 검증함으로써 스크립트언어를 사용해 검증하는 일반적인 방법을 개선하였다.

1. 서 론

XML(eXtensible Markup Language)은 과거 전자문서의 논리적 정보와 물리적 정보가 같이 표현되던 구조에서 문서의 논리적구조(XML), 물리적구조(XSL), 문서의 연결(Xlink, Xpointer)를 분리하려는 요구와 시도에 의해 생겨나게 되었다. XML은 이상적인 전자문서의 요건을 갖추고 있으며 현재 모든 형태의 데이터와 문서를 통합, 저장, 처리할 수 있는 프레임웍[1]을 제공한다.

이처럼 전자문서의 사실상의 표준인 XML로 문서를 작성하고 검증할 때 기존의 연구[2,3]들은 대부분 XML 문서의 구조검증에 국한되어 있다. 즉, 문서형 정의(DTD)에 맞게 유효한(valid)문서인지, 적격(well-formed)한지에 대한 문서 검증만 이루어 졌다. 이에 비해 XML 엘리먼트 값 검증에 대한 연구는 미비하다.

본 논문에서는 폼(form)의 정의와 표현을 분리하기 위해 XML과 XSL(eXtensible Stylesheet Language)을 이용하여 사용자가 입력할 수 있는 인터페이스를 HTML로 만들었다. 사용자가 HTML 폼에 입력한 데이터는 XML로 저장된다. 그리고 사용자가 데이터를 입력 할 때의 유의사항을 로직으로 표현하여 XML로 변환시켜서 사용자가 입력한 데이터가 저장되어 있는 XML문서에 함께 포함시켰다. 로직표현은 논리프로그래밍 언어이고 규칙표현에 효과적인 Prolog를 이용하였으며 궁극적으로 Prolog를 이용하여 XML 문서검증이 가능하다는 것을 확인하였다.

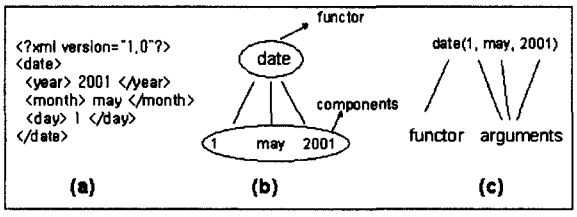
2. 관련 연구

XML 문서검증과 관련된 기존의 연구는 대부분 문서 구조검증에 관한 연구였다. XML 문서는 유효한(valid) 문서, 적격한(well-formed)문서로 대별된다. 유효한 문서는 DTD(Document Type Definition)를 포함하는 문서가 DTD의 정의대로 작성되어 있는지 검증용 파서로 검증된 문서이다. 적격한 문서는 DTD가 없이 작성되고 문서는 오직 하나의 루트 엘리먼트를 가지고 있고, 태그는 중첩규칙에 맞게 작성되어야 된다는 등의 여러 가지 규칙에 맞게 작성된 문서이다.

XML 문서 구조검증에 관한 대표적인 연구는 다음과 같다. 첫 번째는 Schematron[2]으로서, XML 인스턴스에 적용될 점검사항과 규칙등을 정의하여 XML 문서를 검증하는 방법이 있다. Schematron은 XML 스키마언어이고 XSLT기반의 XML문서 검증도구이다. 한편 유효한 XML 문서 검증을 위해 기존에 적용하던 DTD를 이용하지 않고 RELAX(REgular LAnguage description for XML)[3]를 이용한 사례도 있다.

3. Prolog와 XML과의 관계

Prolog와 XML과의 상호 관계를 살펴보면 XML 문서의 논리적 구조 자체는 구조적인 트리로 나타난다. 이러한 트리는 Prolog의 structure 객체로 매핑될 수 있다[4]. date를 표현하기 위해 [그림 1]에서 (a)는 XML, (b)는 트리 (c)는 트리를 Prolog로 표현했다.



[그림 1] Prolog와 XML과의 관계 표현

XML 문서는 엘리먼트로 구성되어 있고 각각의 엘리먼트는 <tag> ... </tag>의 형식이다. Prolog프로그램을

XML형식으로 변환하기 위해서는 Prolog에서 사용하는 Herbrand Term(constants, variables, structures)과 Horn Clause(fact, rule)를 XML 문서의 엘리먼트로 변환하기 위한 규칙이 필요한데 <표 1>과 같다.

<표 1> Prolog-XML 엘리먼트 변환 규칙

Herbrand Term	element	Horn Clause	element
constant(atom) constant(number) logic variables structures	<atom> <number> <var> <struc>	Predicate Relation Symbol Fact,Rule	<relator> <relator> <hn> <relationship>

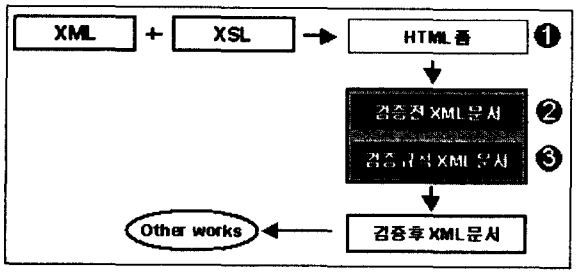
<표 1>의 변환 규칙으로 [그림 1]의 (c)에 해당하는 date(1, may, 2001)을 표현하면 다음과 같다.

```
<hn>
  <relationship>
    <relator> date </relator>
      <number> 1 </number>
      <atom> may </atom>
      <number> 2001 </number>
  </relationship>
</hn>
```

본 논문에서는 Prolog Term, Clause를 XML 기반으로 표현 할 수 있도록 Prolog를 XML로 변환하는 translator(이하 Prolog2XML(*))를 통해서 자동적으로 생성하고 있다.

4. 실험 방법

4.1 실험 과정



[그림 2] 처리 과정

본 논문은 XML 문서로 일관된 처리를 하고 HTML 기반의 폼을 개선하기 위해 XML에 XSL을 이용해서 ①과 같이 HTML 폼을 생성하였다. HTML 폼 인터페이스를 이용하여 사용자는 입력시 주의사항에 맞게 데이터를 입력한다. 입력된 데이터는 ②와 같이 XML 문서로 저장된다. 이 단계에서는 입력한 값이 입력 요구조건에 맞게 입력되었는지 검증하는 전처리 단계이다. ③에서는 입력시 유의사항에 해당하는 부분을 Prolog 로직으로 표현하고, Prolog2XML translator를 통해서 XML 문서로 작성하는 부분이다. 최종적으로 ②, ③두개의 XML 문서는 입력 데이터와 검증규칙이 함께 포함한 하나의 XML 문서로 표현된다. 이를 위해 다음과 같은 환경에서 실험하였다. (1) O/S : Microsoft Windows NT 4.0, (2) 구현 도구: SWI-Prolog (Version 4.0.9), Oracle9i Application Server(1.0.2.2), Oracle XML Parser for Java v2.

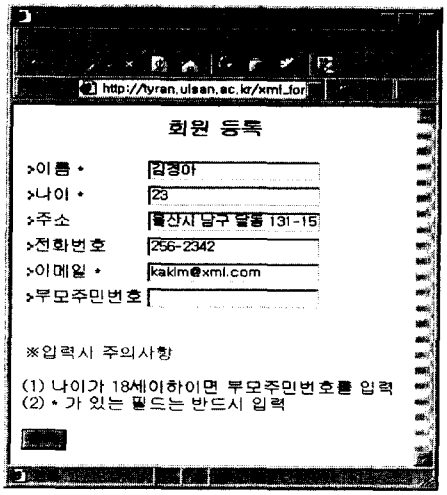
4.2 XML 기반 폼

서론에서도 언급하였지만 XML문서가 전자문서의 표준으로 되어가는 것은 문서의 내용, 표현, 구조가 분리되어 있어서 재사용성이 우수하고 데이터교환이 쉽기 때문이다. XML의 이러한 사상을 폼에도 적용시킬수 있다. 웹페이지에서 폼을 작성할 때 주로 사용하는 HTML은 다음과 같은 결점이 있다[5].

(1) 기능과 스타일이 혼합되어 있다.
(2) 기초적인 수준의 폼 정의만 가능하다.
(3) 폼 정의와 코드가 뒤섞여 있어서 재사용은 어렵다.
(4) 유연성이 부족하다.

HTML 기반 폼의 이와 같은 문제점으로 인하여 본 논문에서는 XML에 기반한 폼으로 작성하려고 한다. 하지만 HTML 폼은 XML에 XSL을 적용한 결과로 나타나게 되어 단지 사용자들이 데이터를 입력하기 위한 인터페이스로 존재하게 된다. XML이 플랫폼 독립적인 정보 교환 포맷으로 사용된다는 사실은 이미 널리 알려졌지만 XML이 웹 사이트 개발에 가져온 가장 큰 효과는 내용과 기능으로부터 포맷을 분리하였다는 점일 것이다.

4.3 실험 결과



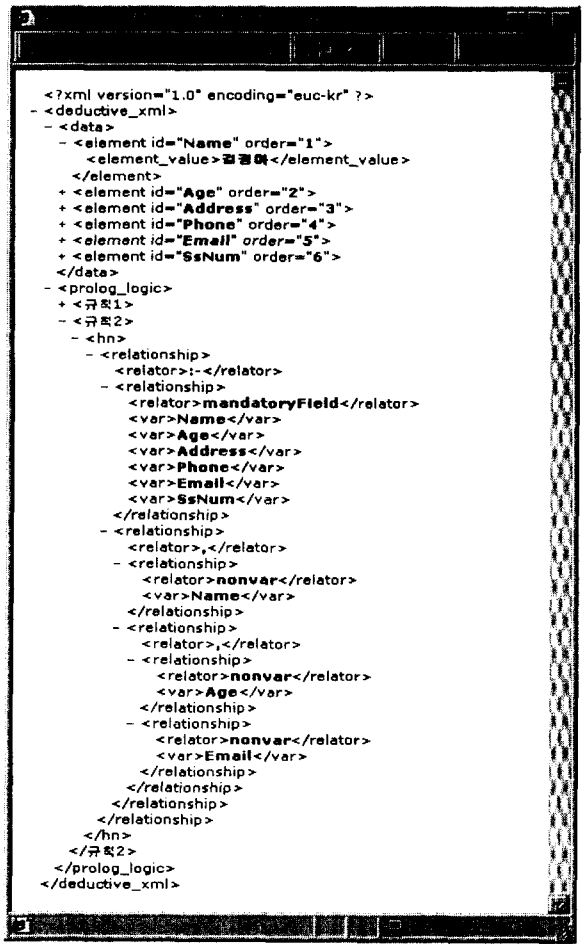
[그림 3] 사용자 입력폼

[그림 3]은 사용자 입력폼이다. XML에 폼정의를 하고 XSL을 이용하여 HTML을 생성하였다. HTML폼에서 입력값 인증은 주로 스크립트언어를 사용하지만 본 논문은 규칙표현에 효과적인 Prolog를 이용하여 개선하였다. [그

림 3]에서처럼 입력시 주의사항인 2가지의 규칙은 다음과 같이 각각 Prolog의 rule로 기술될 수 있다.

```
[규칙1] 나이가 18세 이하이면 부모주민번호 입력
  minorCheck(Name, Age, Address, Phone, Email, SsNum)
                :- Age =< 18, nonvar(SsNum).
[규칙2] *가 있는 필드는 반드시 입력
  mandatoryField(Name, Age, Address, Phone, Email, SsNum)
                :- nonvar(Name), nonvar(Age), nonvar(Email).
```

[그림 3]에서 사용자가 입력한 데이터는 XML 문서로 생성되고 [규칙1]과 [규칙2]를 Prolog2XML translator를 통해 생성된 XML 문서와 합쳐져서 [그림 4]와 같이 최종 결과를 생성한다. 즉, XML 문서에 데이터와 로직이 <규칙1>과 <규칙2>엘리먼트로 함께 표현되어 있다.



[그림 4] 검증 규칙이 포함된 XML 문서

[그림 5]는 [규칙2]를 SWI-Prolog에서 Prolog2XML translator를 통해 수행한 결과이다.

```
?- [prolog2xml].
% prolog2xml compiled 0.00 sec, 6,784 bytes

Yes
?- prolog2xml.
|: mandatoryField(Name, Age, Address, Phone, Email, SsNum) :-
|: nonvar(Name), nonvar(Age), nonvar(Email).
|: end_of_file.
<?xml version="1.0"?> <hn> <relationship> <relator> :- </relato
r> <relationship> <relator> mandatoryField </relator> <var> Nam
e </var> <var> Age </var> <var> Address </var> <var> Phone </va
r> <var> Email </var> <var> SsNum </var> </relationship> <relat
ionship> <relator> , </relator> <relationship> <relator> nonvar
 </relator> <var> Name </var> </relationship> <relationship> <r
elator> , </relator> <relationship> <relator> nonvar </relator>
 <var> Age </var> </relationship> <relationship> <relator> nonv
ar </relator> <var> Email </var> </relationship> </relationship
> </relationship> </relationship> </hn>

Yes
?-
```

[그림 5] Prolog2XML 수행 결과

5. 결론

본 논문은 XML의 논리적 구조가 Prolog의 Structure로 매핑 될 수 있고 의미론적으로 유사한 상호 연관성을 통하여 XML과 Prolog의 통합 표현을 시도하였다. 이러한 시도의 한 결과로 XML로 표현된 Prolog 로직을 생성할 수 있었다.

일반적으로 문서의 내용이 데이터만 포함하고 있는 경우는 "문서는 정적(static)이다." 라고 할 수 있다. 하지만 정적인 XML 문서에 Prolog 로직을 XML로 변환하여 포함 시킴으로써 XML 문서가 Prolog를 통해서 검증 규칙을 내포 할 수 있다는 것을 보였다. 향후에는 Prolog로 구현한 시스템에서 검증규칙을 포함한 XML 문서를 검증할 예정이다.

6. 참고 문헌

[1] 남철기, 김혜경, 배재학. 응용프로그램을 위한 일관된 XML뷰 제공에 관한 연구, 한국정보처리학회 춘계학술발표대회 논문집, 제8권 제1호, pp.1105-1108, 2001.

[2] Schematron, http://www.ascc.net/xml/resource/schematron/.

[3] Relax, http://www.xml.gr.jp/relax/.

[4] Harold Boley, Relationships between Logic Programming and XML, Proc. 14th Workshop Logische Programmierung, W"urzburg, Jan. 2000.

[5] XML폼, http://korea.internet.com/channel/content.asp?kid=2&nid=861&cid=193.

(*) Prolog2XML구현에 도움을 준 호주 Queensland 대학의 Benjamin Johnston에게 감사를 표한다.