Validated-XML 파서 생성기

신경희, 신현경, 유재우
숭실대학교 컴퓨터학과
{Khshin,hKshin,cwyoo}@ss.soongsil.ac.kr

A Validated-XML Parser Generator

Kyoung-Hee Shin, Hyun-Kyung Shin and Chea-Woo Yoo
School of Computing, Soongsil University.

XML이 차세대인터넷언어로 등장하므로서 XML을 이용한 문서들이 많이 생기고 있다. XML은 구조 중심으로 문서를 작성하는 언어이므로, 올바른 문서인지 확인 작업이 반드시 필요하다. 이러한 처리는 XML 파서에 의해 처리된다. 본 논문에서는 문서구조가 다른 여러가지 XML문서의 적합성을 점검하는 validated-XML 파서 생성기를 설계 구현한다. XML 파서 생성기는 구조에 따른 문서를 파싱하는 컴파일러-컴파일러로서, 프로그래밍 언어관점에서 설계하였다. XML파서 생성기는 내부적으로 문서형정의 파서와 문서파서로 구성되고 이 2개의 파서는 서로 연동되어 XML 문서를 동적인 환경에서 대화식으로 처리할 수 있으며, 편집기와 브라우저와 같은 어플리케이션과의 인터페이스가 가능하다.

1. 서론

인터넷 사용이 일상화되면서 인터넷상에 올라오는 문서들은 매일같이 수백만건이상이 된다. 웹문서 내용에 있어서 단순 텍스트에서 멀티미디어 문서에 이루기까지 문서 형태도 다양해지고 활용분야도 정보검색 뿐 아니라 e-비지니스, 전자상거래까지 확장되고 있다. 따라서 인터넷 웹 사용자의 확산으로 전 세계가 네트워크으로 연결되고 "web-life style"이라는 용어가 생겨날 정도로 사이버공간의의 삶이 우리생활의 일부가 되고 있다.
웹 문서를 작성하는데 가장 널리 사용되고 있는 HTML은 관련도구들이 많고 사용이 쉽다는 장점을 갖고 있는 반면에 다양해지는 웹 문서를 효과적으로 관리할 수 없다는 단점을 갖고 있다. HTML 문서는 내용중심이기보다는 표현중심으로 작성되기 때문에 문서검색이나 재사용 처리가 적절하지 않다. 이는 다양해지는 인터넷 활용에 큰 단점이 된다. HTML의 단점을 보완하기 위하여 1998년 W3C에서는 XML(eXtensible Markup Language)를 발표하였다.
XML은 태그를 이용하는 웹 문서용 언어로, XML의 문서구조화 특성과 인터넷사용이 용이한 HTML장점을 갖고 있는 차세대 인터넷 언어이다.
XML은 문서를 구조중심으로 표현하는 언어로, XML문서구성은 선언, 문서형정의(DTD), 문서인스턴스(DI)로 이루어진다. XML 선언은 문서버전과 엔코딩 문자에 대하여 언급한다. 문서타입에 따라 달라질 수 있는 문서의 형태를 정의하는 문서형정의는 문서열거순서를 결정하는것으로 문서문법이라고 할 수 있다. 실제 문서의 내용에 해당하는 문서 인스턴스는 문서형정의의

문서문법에 따라 작성된다. XML 문서인스턴스에 나타나는 태그는 시작태그와 끝태그가 한 쌍으로 이루어진다.
XML이 문서타입에 따라 유연하게 웹 문서를 작성할 수 있다는 큰 장점을 갖고 있지만, 이러한 문서를 작성하기 위하여 XML 문서 작성시스템이 필요하다. XML 문서를 작성하는 작성자는 문서형정의에 따라 작성해야 하고 시작태그와 끝태그를 반드시 존재해야 하는 규칙을 따라야 하는 하는데 이런 부담감은 XML 문서 시스템에서 자동으로 해결될 수 있다.
XML 문서작성시스템은 편집기능, 브라우징기능 그리고 파싱기능을 갖게 된다. 파싱기능은 XML 관리시스템의 기본적인 내부적 프로세스로서 이를 XML파서라고도 한다.
XML 파서는 XML 문서형태에 따라 well-formed XML파서와 validated-XML 파서로 구분한다. well-formed XML파서는 시작태그와 끝태그 순서대로 겹치지 않고 존재하는가를 점검하고 validated-XML 파서는 well-formed 파싱기능을 갖고 있으면서 해당 문서형정의에 따라 문서가 작성되어 있는지을 점검한다.
본 논문에서는 문서형정의 에 따라 XML 문서의 적합성을 점검하는 validate-XML 파서의 생성기를 설계하고 그에 따른 문서 편집환경을 제시한다.

2. validated-XML 파서 설계

XML이 인터넷 기반언어로 자리잡으면서 올바르게 작성된 XML문서가 요구되고 있다. 따라서 XML 문서작성 관련도구들이 많이 개발되고 있다. 이 도구들이 내부 프로세서로서 이용되는 XML 파서로는 MS 파서와 IBM 파서가 있다.
Microsoft XML parser는 C++로 구현된 non-validated한 것과 Java로 구현된 validated한 것으로 구분된다. 파서는 IE4.0에 내장되는 것으로, XML의 최소 신택스는 hand-coded 파싱하고 recursive-descent 파싱으로 사용이 쉽고 이해가 빠르도록 했다. Java로 구현된 MSXML은 XML이 길이에 구애 받지 않는 많은 어휘적 엘리먼트를 갖고 있기 때문에 별도의 파서 생성기를 사용하지 않았다. XML 파서에 의해 처리된 XML 문서는 DOM 인터페이스가 가능한 트리 정보를 생성한다. IBM 파서는 validating XML 파서로서 Java로 구현된것으로 DOM 과 SAX 특성을 지원한다. 파서 패키지는 XML 문서의 적합성 처리를 위한 파싱, 생성 그리고 조작하는 기능들을 클래스와 메소드로 구성한다. MS 파서와 IBM 파서의 프로세싱에 대한 문서를 제공하지않으므로 개발자는 각 파서가 제공하는 API에 의해 문서를 처리하게 된다.
본 논문에서 XML 문서를 일반 프로그래밍 관점에서 접근하여 XML 문서형정의에 따라 작성되는 문시를 컴파일하는 validated-XML 파시를 설계 및 구현 한다.
validated-XML파서는 컴파일러-컴파일러로서, XML 문서형정의내용이 XML 표준문법에 맞게 작성되어 있는지를 점검해야 한다. 올바르게 작성된 XML 문서형정의는 XML 문서구조를 명확하게 표현할 수 있기 때문이다. 이와 같이 XML 문서형정의를 점검하는 파서를 DTD 파서라고 한다. 이 DTD 파서는 문서형정의의 유효성을 점검한 후 문서 작성문법을 생성하게 된다. 이 문서문법은 실제 XML 문서를 작성할 때 사용태그의 순서를 결정하는 것으로 이는 문서파서의 문법으로 적용된다.
본 논문에서 설계하는 파서는 그림 1과 같이 내부적으로 DTD 파서와 문서파서 그리고 문서파서생성기로 구성하여 XML 문서의 적합성을 점검하는 validated-XML 파서이다.

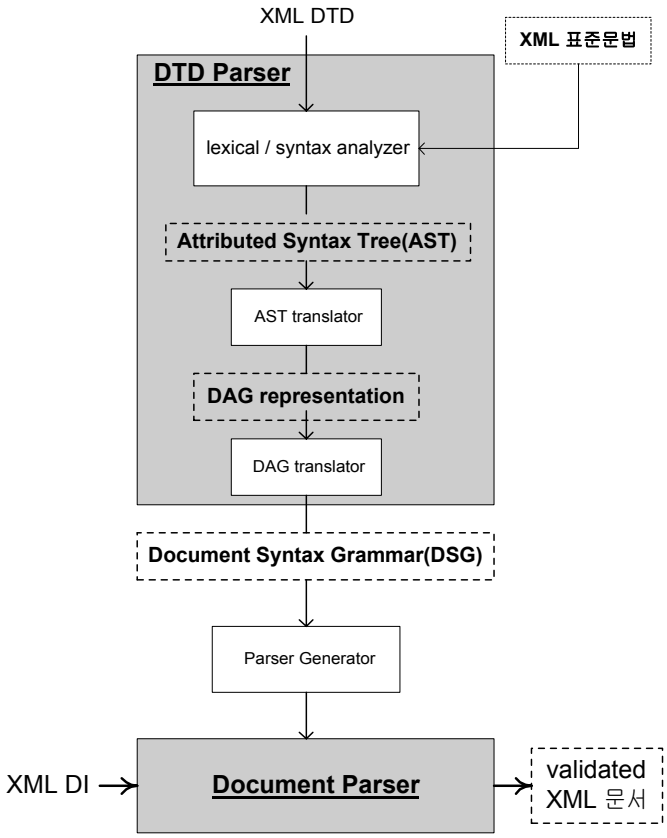

그림 1. XML 파서 설계

DTD 파서는 XML DTD를 입력받아 XML 표준 규칙에 맞게 작성된 것인지 어휘분석과 구문분석과정에서 확인한다. XML DTD의 엘리먼트와 엔티티 정보 그리고 여러 속성들을 포함하는 추상구문트리를 생성하는데 본 논문에서는 이 트리를 속성구문트리(Attributed Syntax Tree:AST)라 한다. 이 AST에서 엘리먼트 정보를 중심으로 문서구조정보를 추출하여 그래프 형태의 DAG(Directed Acyclic Graph) 정보로 변환한다. 변환된 DTD DAG는 XML 문서 파서를 생성하는데 기본이 되는 정보로 본 논문에서는 파서 구현을 위하여 그리고 편집기와의 인터페이스를 위하여 DAG 그래프 정보를 BNF 형태로 재 표현한다. BNF형태의 XML문서문법을Document Syntax grammar(DSG)라 한다. 이 DSG는 문서파서를 생성하는 문서 문법이 된다.

DTD 파서에 의해 생성된 DSG는 파서 생성기를 이용하여 해당 문서구조에 적합한 문서파서를 생성한다. XML 문서파서는 입력된 XML 문서가 해당 문서형정의 규칙에 따라 작성된 것인지를 적합성을 점검하게 된다. 문서 파싱 후 적합성이 확인되면 올바른 문서로 판단되면서 validated-XML 문서를 2가지 타입으로 생성한다. 하나는 내부 표현방법인 트리 형태의 문서정보이고 또 다른 하나는 문서전달을 위한 일반적인 텍스트형태의 문서이다.

3. DTD 파서

입력된 DTD가 올바른것인지를 점검하는 DTD 파서는 어휘/구문분석기, AST 변환기 그리고 DAG 변환기로 구성된다.

3.1 어휘/구문분석기

XML DTD의 토큰과 구분자들은 선언규칙에 따라 인식정도가 다르게 처리된다. DTD 어휘분석기는 표준에서 정의한 인식모드별로 토큰들을 구분하게 된다. DTD의 파라메타엔티티 참조는 DTD의 내용과 직접적인 연관성을 갖고 있으므로 어휘분석에서 처리되어야 한다. 파라메타 엔티티는 선언이 된 후 참조되어야 하므로 선언된 파라메타 엔티티내용은 별도의 테이블에 저장하고 토크 중 %name; 표현이 발견되면 해당 정보를 테이블에서 처리하도록 한다.
DTD 구문분석기는 작성된 DTD가 표준 XML 규칙을 충분히 지켰는가를 확인하는 문법적인 오류를 점검한다. 표준에서 정의한 DTD문법은 표-1과 같이 EBNF 형태로 재 표현이 가능하다.

```
DTD = MDO, "DOCTYPE", GI, document-type-declaration, MDC
document-type-declaration = ( element-declaration
                            ¦ entity-declaration
                            ¦ attribute-declaration )*
element-declaration = MDO, "ELEMENT", element-type, (declared-content ¦ model-group) , MDC
element-type = GI ¦ name-group
model-group = GRPO, content-token, ((OR, content-token)* ¦ (SEQ, content-token)* ) GRPC,
              (OPT¦PLUS¦REP)?
content-token = "#PCDATA" ¦ (GI, (OPT¦PLUS¦REP)?) ¦ model-group
```

표-1. DTD의 element 선언 규칙

EBNF 표기법으로 표현되는 DTD 표준문법은 LALR방식을 이용하여 입력된 DTD를 구문분석한다. 구문분석결과 생성되는 추상구문트리는 엘리먼트와 엔티티 그리고 애트리뷰트정보 등 선언된 모든 정보를 포함하게 되므로 본 논문에서는 속성구문트리(AST:Attribute Syntax Tree)라고 한다. 속성구문트리의 최대 차수는 3으로 하여 자식노드의 갯수를 제한한다. 자식노드가 3개 이상이 필요로 하는 경우는 더미 노드를 사용하여 논리적으로 볼 수 있는 자식 노드의 수를 확장한다. 속성구문트리에서 문서구문을 결정하는 것은 엘리먼트이므로 AST 변환기는 엘리먼트 정보를 이용하여 DTD 변환정보를 생성한다.

3.2 AST 변환기

AST 변환기는 DTD 속성구문트리를 이용하여 엘리먼트 중복선언, 선언되지 않은 엘리먼트의 어트리뷰트 선언 등과 같은 의미 검사를 한다. AST 변환기의 의미 검사가 완료되면 엘리먼트 정보를 이용하여 XML 문서구조를 나타내는 그래프를 생성한다. 속성구문트리를 순회하면서 엘리먼트 정보 중심으로 생성되는 그래프 차수가 2인 비순환방향그래프(DAG:Direct Acycle Graph)로 다음과 같은 노드정보를 갖는다.

```
struct GNode  { STRING elementNname;    /*엘리먼트 이름*/
                TYPE   operator;        /*connector 및 occurence indicator*/
                struct attributeDeclare; /*attribute 선언정보*/
                BOOLEAN  isTerminal;    /*PCDATA인지 검사*/
                LINK     nodePointer; }; /*노드연결정보*/
```

비순환방향그래프를 엘리먼트 선언정보로 엘리먼트 명과 엘리먼트의 연결자 및 발생자가 주요 정보가 된다. 따라서 비순환방향그래프의 노드는 엘리먼트정보를 포함하는 엘리먼트-노드와 연결자 및 발생자 정보를 포함하는 연산자-노드로 구분된다. 그래프 노드정보에서 엘리먼트를 나타내는 노드 정보에는 엘리먼트-선언 뿐 아니라 어트리뷰트-선언 내용을 포함한다. 특히 어트리뷰트선언 내용으로는 어트리뷰트의 이름, 값의 유형, 기본값의 타입들의 리스트들을 갖는다.

AST 변환기 처리 결과인 비순환방향그래프는 엘리먼트 정보에 의해 생성된다. 그래프를 구성하는 노드는 엘리먼트 노드와 연산자노드로 구분되고 이 노드들간의 관계는 방향성 있는 에지로 표현된다. 비순환방향그래프는 하향방식(top-down)으로 생성되고, 베이스-엘리먼트이름이 그래프의 루트 노드가 된다. 그래프의 깊이우선탐색방법(DFS:Depth First Search)으로 해당 엘리먼트 노드정보를 얻을 수 있다.

3.3 DAG 변환기

XML DTD의 비순환방향그래프는 실제 XML 문서를 파싱하는 문법정보이다. 본 논문에서는 파싱 구현과 편집기와의 인터페이스를 위하여 그래프 문법 정보를 BNF 방식으로 재 정의한다. 논문에서 정의한 BNF 표현은 엘리먼트 이름을 비단말(noterminal)로 정의하고, PCDATA나 EMPTY같은 선언을 단말(terminal)로 한다. 그리고 시작기호(start symbol)는 그래프의 루트노드의 엘리먼트 이름이 된다. 이 표현은 프로그래밍 언어의 context-free 문법을 기준으로 XML 문서형정의에의 반복이나 연결정보를 나타내는 연산자정보를 추가 표현한다. [그림 2]는 문서형정의 비순환방향그래프를 BNF 형태로 나타내 표현법으로 본 논문에서는 이를 DSG(Document Syntax Grammar)라 한다.

엘리먼트명 → 연산자(엘리먼트명이나 PCDATA의 리스트)

그림 2. DSG 문법 형식

4. 파서 생성기와 문서 파서

DTD 파서에 의해 생성된 DSG는 실제 문서를 검증하는 파서의 문법으로 이것은 XML 문서형정의에 따라 각각이 생성된다. 문서형정의에 따른 XML 문서를 검증하기 위하여 해당 문서파서를 별도로 생성한다는 것은 비 효율적이다. 따라서 XML 문서파서를 문서형정의구조에 따라 자동적으로 생성하기 위한 도구가 필요하다.

XML 문서파서생성기는 DTD파서의 출력인 DSG를 기본 문법으로 하여 문서파서를 동적으로 자동

생성한다. 파서 생성기에 의해 생성된 문서 파서는 XML 문서(XML DI)를 입력으로 하여 구조문법에 맞게 작성된 문서인지를 파싱한다.
문서파서는 어휘분석과 구문분석 모듈로 구성된다. 어휘분석과정에서는 토큰 인식상태를 태그상태와 태그가 아닌상태로 구분한다. 그리고 구문분석을 위하여 사용하기 되기 위하여 토큰 종류를 시작태그, 끝태그 그리고 문자열 3가지로 구분한다. 어휘분석모듈에서 구분한 토큰은 구문분석모듈로 전달되어 하향식처리기법에 의해 XML 문법에 따라 작성된 문서인지 검증한다. 문서파서의 구문분석은 해당 DSG 문법를 근거로 Recursive Decent 파싱에 의해 처리된다. 구문분석과정에서 생성되는 문서의 내부정보는 트리 구조로 표현한다. 트리를 구성하는 노드는 엘리먼트정보를 갖고 있고, 노드간의 연결은 연산자 정보에 의해 에지가 연결된다. 문서파서에 의해 생성되는 XML문서트리의 다음과 같은 노드정보로 구성된다.

```
struct DNode {
        STRING tagName;          /*태그명*/
        PRODUCT dsgInform;       /*DSG 문법정보*/
        ATTRIBUTE attributeList;/*속성정보*/
        STRING documentString;  /*문서텍스트*/
        struct DNode *brother, *children, *next, *prev; /*노드연결정보*/
};
```

문서파서는 문서편집기의 내장 프로세서로서 사용가능하므로 문서형정의에 따라 문서작성을 유도할 수 있다. 또한 문서파서의 결과인 문서트리정보를 이용하여 편집기에서의 대화식문서 조작이 가능하다.

5. 실험 및 결론

본 논문에서 설계한 XML 파서생성기는 문서형정의에 작성된 XML 문서의 적합성을 처리하는 validated 파서를 생성하는 프로세서로서 내부적으로 DTD파서와 문서파서로 구성된다. 이 파서생성기의 실험은 UNIX 환경에서 c 언어를 이용하였고, 어휘분석과 구문분석을 위하여 flex/yacc 툴을 사용하였다.
파서 생성기에 의해 만들어지는 XML 파서는 DTD파서를 내장하고 있어 다양한 유형의 XML 문서처리가 가능하다. 그리고 DTD파서에 의해 생성된 DSG 문서문법은 문서형정의별로 저장, 관리하게 하여 같은 유형의 XML 문서 파싱시 별도의 DTD파싱과정을 거치지 않고도 문서파싱 가능하게 하여 처리속도를 향상시켰다.
문서파서에 의해 문서구조와의 적합성이 확인된 문서는 문서공유를 위한 텍스트 타입과 내부처리를 위한 트리타입으로 각각 구분하여 생성하였다. 텍스트타입의 XML 문서는 문서형정의와 문서인스턴스를 포함하도록 하였다. 트리구조의 XML 문서정보는 DOM 과 같은 처리가 가능한 관련 API를 충분히 지원하므로, XML 편집기나 브라우저와의 인터페이스가 가능하다.
XML 파서 구현에 따른 문서내부구조에 대한 지식를 기초로 응용분야별 관련 어플리케이션 개발이 가능하다. 시스템 환경과 무관하게 작동하는 파싱 프로세스를 위해 java로의 구현이 이

루어져야 하고, XML 파서 결과 생성되는 문법정보나 문서정보에 관한 효율적인 관리 방법에 대한 연구가 요구된다. 또한 표준 API로 대두되고 있는 DOM 스펙을 지원할 수 있는 API 설계 및 구현이 필요하다.

Reference

1. Jean Paoli 외 4명, "Building XML Parsers For MicroSoft's IE4", *XML:Principles, Tools, and Techniques*, O'Relly, 1997.
2. XML Parser for Java, http://alphaworks.ibm.com/tech/XML4J.
3. Martin Bryan, *SGML:an Author's Guide to the standard generalized markup language*, Addison-Wesley Publishing Company, 1991.
4. Jos Warmer and Sylvia van Egmond, "The implementation of the Amsterdam SGML Parser", *Electronic publishing*, Vol.2(2), July 1989.
5. Elliotte Rusty Harold, *XML Bible*, IDG Books Worldwide Inc.,1999.
6. Extensible Markup Language(XML)1.0, http://www.w3.org/TR/REC-xml.
7. XML-Software. Xml parsers, *http://www.xmlsoftware.com/parses*7.
8. Charles F. goldfarb, Paul Prescod, *The XML HANDBOOK*, Prentice Hall PTR, 1998.8.
9. John R. Levine, Tony Mason & Doug Brown, *lex & yacc*, O'Reilly & Associates, Inc.1992
10. Alfred V. Aho, Jeffrey D. Ullman, *The Theory of Parsing, Translation, and Compiling Vol. 1:Parsing*, Prentic-hall Inc.