동적 구문지향 SGML 문서편집기

0
강춘길, 신경희, 유재우
숭실대학교 컴퓨터학과

{ckkang, khshin}@ss.ssu.ac.kr, cwyoo@comp.ssu.ac.kr

Dynamic Syntax-Directed SGML Document Editor

0
Chun-Kil Kang, Kyoung-Hee Shin, Chae-Woo Yoo
Dept. of Computing, Soongsil University

요 약

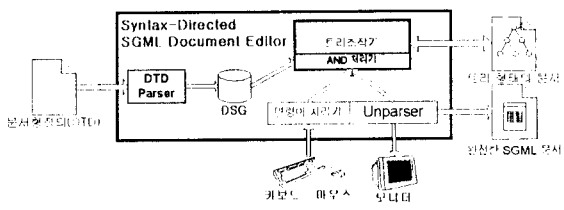
본 논문에서 다루는 SGML 문서편집기는 파서를 내포하는 구조로서, 문서편집을 위하여 DTD 가 입력되고, 입력된 DTD 는 DTD 파서에 의해 적합성을 점검하게 된다. DTD 파서는 표준규칙에 따라 DTD 를 파싱하고, 결과로 편집처리에 적합한 BNF-스타일 형태로 변환된 문법 테이블을 생성한다. SGML 문서 편집은 문법테이블 정보에 따라 처리된다. 문법 테이블의 크기는 엘리먼트 선언내용 중 컨텐트 모델의 표현에 따라 결정된다. 그 중 and(&) 접속자는 엘리먼트 개수에 따라 문법수가 기하급수적으로 증가하므로 DTD 파서에 의해 처리된다면 속도나 비용면에서 비효율적이다. 이에 본 논문에서는 고정된 문법테이블을 이용하는 SGML 편집기에 and 접속자로 표현된 엘리먼트 문법을 동적으로 확정 처리할 수 있는 기능이 추가· 향상된 구문지향적 SGML 문서편집기를 제시한다. 그러므로 향상된 구문지향 SGML 문서편집기는 고정된 문법을 이용한 문서편집 뿐 아니라 편집처리에 따른 변환에 능동적으로 처리할 수 있다는 특징을 갖게 된다.

1. 서론

SGML 문서 편집기는 사용자가 작성하고자 하는 특정 문서의 구조에 대한 지식이 없어도 편집기와의 인터페이스를 통해서 문서편집이 가능하도록 해야 한다. 즉, 문서 편집을 위해서 사용자가 취할 수 있는 문서의 태그를 선택하고, 그 태그에 따라서 편집기의 내부에서 사용자의 명령에 따른 동작을 해야 한다. 이런 기능의 편집기를 구현하기 위해서 문서의 구조에 따라 문서를 작성할 수 있는 구문 지향적 편집기가 필요하다. 기존의 SGML 문서 편집기는 한 두개의 특정한 DTD 만을 대상으로 하는 것이 대부분이며, 모든 종류의 SGML 문서를 편집할 수 있는 기능을 갖춘 편집기는 극히 소수이다.

본 논문에서 다루는 SGML 문서 편집기는 DTD 파서를 내장하는 구조로, 여러 가지 DTD 에 따른 SGML 문서편집이 가능하다는 장점이 있다. 편집기의 구조는 [그림 1]과 같다.

SGML 문서의 DTD 가 입력되면 DTD 파서는 SGML 표준규칙에 따라 구문 분석하여 적합성을 점검한다. 파서에 의해 검증된 DTD 는 문서편집에 적절한 형태로 변환 처리된 결과로 SGML 문서 구문 문법(Document Syntax Grammar: DSG)을 생성한다. 이 문법은 사용자가 입력한 명령어와 함께 처리되면서 트리 조작기에 전달되어 문서 트리를 생성하고 생성된 문서 트리 정보는 Unparser 에 의해 사용자에 전달된다. 트리 조작기는 Unparser 나 명령어 처리기의 요청에 의해 DSG 의 정보를 이용하여 트리를 확장/삭제/변경하기나 트리 형태의 문서로 저장한다. 특히) and p접속자와 연관되는 DSG 정보는 and 처리기에 의해 동적으로 구문문법을 생성 및 조작하여 문서편집에 이용된다. 명령어 처리기는 사용자가 입력한 해당 명령을 트리 조작기에 전달하고 Unparser 는 트리 조작기를 이용하여 트리 정보를 화면에 출력하기나 완전한 SGML 문서로 저장한다.



[그림 1] SGML 문서 편집기 구조

본 논문은 DTD 파서에 의해 생성되는 DSG 에 대하여 설명하고 and-관련 DSG 정보를 생성할 때 발생할 수 있는 문제점을 언급하고 이를 해결하기 위해 SGML 문서를 편집하면서 동적으로 and-관련-DSG 정보를 생성하여 효율적으로 처리할 수 있는 방법을 제시한다.

2. 문서 구문 문법

본 논문에서는 SGML 문서처리에 적절한 DTD 정보 형태를 문서 구문 문법(DSG)이라고 명명한다. SGML 문서 편집기에 입력된 DTD 정보는 최종적으로 DSG 형태로 저장 및 관리하게 된다. DSG 정보는 DTD 에 의해 생성된 비순환 방향그래프 정보를 문서 편집 환경에서 이용할 수 있도록 정의한 문서 생성 규칙(production rule)이다. DSG 정보표현은 프로그래밍언어의 BNF 표기법을 이용하였다. BNF 의 생성규칙 표현법에 따라 DTD 에 따라 작성되는 DSG 형태는 [그림 2]와 같다.

엘리먼트명→지시자 정보 (컨텐트모델의 부 엘리먼트 목록)

[그림 2] DSG 문서생성규칙 형식

엘리먼트 선언의 컨텐트 모델에 따른 생성된 DSG 형태는 접속자(connector), 반복자(occurrence)에 따라 구분된다.

1) 접속자에 따른 DSG

- seq(,)
  seq(,)는 선언 순서대로 발생됨을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B, C) >
  ⇓
  A → PROD_PAIR(B C)
  위와 같이 seq 로 연결된 엘리먼트 선언은 1 개의 DSG 규칙으로 변환된다.

- or(|)
  or(|)는 여러 개 엘리먼트 중 선택적으로 하나만 발생되는 것을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B | C)>
  ⇓
  A → PROD_ONE(B)
  | PROD_ONE(C)
  위와 같이 or 로 연결된 엘리먼트 선언은 컨텐트 모델의 엘리먼트 개수만큼의 DSG 규칙이 생성된다.

- and(&)
  and(&)는 엘리먼트들이 순서에 관계없이 모두 발생되는 것을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B & C)>
  ⇓
  A → PROD_AND(B C)
  위와 같이 and 를 사용한 엘리먼트 선언은 PROD_AND 라는 연결자 정보를 이용하여 1 개의 DSG 규칙으로 변환된다.

2) 반복자에 따른 DSG

- optional(?)
  optional(?)은 엘리먼트가 한번 나타나거나 또는 나타나지 않는 것을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B)?>
  ⇓
  A → PROD_NULL( )
  | PROD_ONE(B)
  위와 같이 ? 반복자를 포함하는 엘리먼트 선언은 2 개의 DSG 규칙으로 변환된다.

- plus(+)
  plus(+)는 엘리먼트가 최소 한번이상 발생하는 것을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B)+>
  ⇓
  A → PROD_ONE(B)
  | PROD_PAIR(B A)
  위와 같이 + 반복자를 포함하는 엘리먼트 선언은 2 개의 DSG 규칙으로 변환된다.

- star(*)
  star(*)는 엘리먼트가 0 개 이상 발생할 수 있음을 의미하는 것으로 해당 엘리먼트 선언의 DSG 형태는 다음과 같다.
  <!ELEMENT A - - (B)*>
  ⇓
  A → PROD_NULL()
  | PROD_PAIR(B A)
  위와 같이 * 반복자를 포함하는 엘리먼트 선언은 2 개의 DSG 규칙으로 변환된다.

엘리먼트의 컨텐트 모델에 표기된 접속자나 반복자에 따라 일정한 개수의 DSG 정보가 작성된다. DSG 규칙의 지시자 정보는 다음과 같이 4 개로 표현된다.

- PROD_NULL : 처리될 수 있는 엘리먼트가 없음을 의미한다.
- PROD_ONE : 한 개의 엘리먼트만이 표시되고, 하나의 엘리먼트만이 처리된다.
- PROD_PAIR : 여러 개의 엘리먼트들이 표시될 수 있고 표시된 엘리먼트들은 왼쪽에서 순서대로 처리된다.
- PROD_AND : 접속자 and(&)를 나타내는 지시자 정보이다.

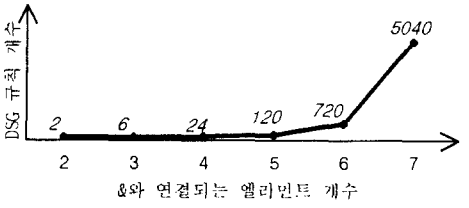
PROD_NULL, PROD_ONE, PROD_PAIR는 DTD 파서에 의해 해당 접속자나 반복자 의미를 충분히 고려하여 최적으로 나타낸 것이지만 PROD_AND는 DSG 형식만을 가지고 있다. 그러므로 DTD 파서에 의해 변환되는 접속· 반복자는 , | ? + * 이고 &는 문서편집과정에서 처리한다.

4. 컨텐트 모델의 동적인 처리

일반적으로 DTD 파서는 처리 속도를 향상 시키기 위하여 컨텐트 모델 내용을 문법 테이블로 재구성하여 이용된다[5]. 컨텐트 모델의 지시자(접속자, 반복자)에 의해 생성되는 문법

테이블은 and(&)를 제외한 지시자는 제한된 문법으로 표현된다. 그러나 and는 예외적이다. and(&)를 선언된 엘리먼트들이 순서에 관계없이 모두 발생해야 한다. 예를 들어 컨텐트 모델 ' (A & B)' 는 ' (A,B) | (B,A)' 로 확장 변형해야 한다.

and(&)로 연결된 컨텐트 모델은 [그림 3]에서 보는 바와 같이 n 개의 엘리먼트들을 처리해야 할 경우 n! 가지의 경우의 수가 나오기 때문에 5 개 이상의 엘리먼트 처리시 파싱의 처리속도가 느려지며, 만들어진 규칙 전체가 사용될 가능성이 적기 때문에 작업 메모리 낭비가 상당히 크게 된다.



[그림 3] 엘리먼트 개수에 따른 AND 연산 규칙의 개수

그러므로 and(&)로 이루어진 컨텐트 모델의 경우에는 DTD 파싱으로 DSG 문법을 고정적으로 생성하여 이용하기 보다는 SGML 문서 편집기의 문서작성 과정에서 동적으로 엘리먼트의 문법을 선택하도록 처리하는 방법을 제시한다.

본 논문에서 구현한 SGML 문서 편집기는 DTD 파서에 의해 생성된 DSG 문법 테이블 정보 중 PROD_AND 지시자를 직접 실행하지 않는다. 편집기는 테이블 형식으로 저장된 DSG 정보를 받아서 문서 편집을 시작하는데 PROD_AND 정보를 읽게 되면, DSG 의 PROD_AND(A1, A2, A3, ..., An)정보는 다음과 같이 처리된다.

처음 n 개의 엘리먼트 중에서 확장하는 경우 내부적으로,

PROD_ONE(A1) | PROD_ONE(A2) | ... | PROD_ONE(An)

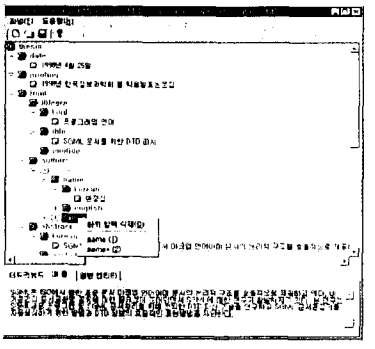
의 연산이 발생된다. 그 다음 또 한 개를 선택하는 경우에는,

PROD_ONE(A1) | PROD_ONE(A2) |...| PROD_ONE(An-1)

와 같이 n-1 개의 PROD_ONE 지시자 정보를 처리하게 된다. 결국은 and 접속자를 처리할 때에는 n 번의 선택을 하게 된다.

SGML 편집기는 사용자 입력을 기다리는 대기 시간이 길기 때문에 CPU가 idle 상태로 있는 시간이 많다. 따라서 편집기는 처리시간보다는 저장메모리에 우선순위가 높다고 할 수 있다. 그러므로 편집기는 앞서 언급했던 정적인 n!개의 테이블 생성보다는 n 번의 사용자 입력 중간중간에 동적으로 확장 가능한 엘리먼트를 처리함으로써 idle CPU 시간과 메모리의 효율적인 관리가 가능하다.

5. 평가 및 결론



[그림 4] SGML 에디터 실행화면

본 논문에서 구현한 문서 편집기의 구현환경은 Visual C++ 6.0, flex 2.3.8, bison 1.21 을 개발환경으로 하며 C, C++ 언어를 사용하였다. [그림 4]는 본 논문에서 구현한 편집기 실행 화면이다. 본 편집기를 통해서 사용자는 특정 문서의 내용(DTD)에 대한 지식이 없어도 편집기와의 인터페이스를 통해서 문서편집이 가능하며 트리 구조의 사용자 인터페이스만 제공하기 때문에 문법(DTD)에 어긋난 편집은 허용하지 않고, 컨텐트 모델 중 and 연산에 대한 엘리먼트 표현의 개수에 제한이 없어짐으로써 SGML 문서 작성시 보다 풍부한 표현을 할 수 있게 되었다. 또한 SGML 파싱에서 부하가 걸리는 컨텐트 모델 처리(and 연산)를 편집기에서 처리함으로써 CPU 처리시간과 메모리의 효율성도 증가되었다.

그러나 현재의 문서 편집 시스템에서는 컨텐트 모델 내에 and 연산을 할 때 오직 한개씩의 엘리먼트 연산만을 사용해야만 한다는 단점이 있다. 따라서 다수의 엘리먼트 그룹을 파싱하는 동적인 and 연산에 대해서도 처리가 가능하도록 기능을 향상시키는 것이 앞으로의 연구 과제이다.

참고문헌

[1] 박준서, 신경희, 문현주, 유재우, " 구조 지향적 SGML 문서 편집기", *정보과학회 봄 학술논문집 Vol. 25, No. 1*, 1998.

[2] 오덕환, " 구문 지향적 교육용 에디터의 구현에 관한 연구", *금오공과대학 논문집 제9집*, 1988.

[3] 원득창, 홍유선, 이수인, " SGML Parser를 이용한 SGML Document Editor의 설계에 관한 연구", *정보과학회 가을 학술논문집 Vol. 18, No. 2*, 1991.

[4] 한만진, 고영곤, 최윤철, " 문법 기반 범용 SGML DI 편집기의 설계 및 구현", *정보과학회논문지(C), 제3권, 제3호*, 1997년 6월.

[5] 이경호, 고승규, 백승욱, 변장원, 장은영, 최윤철, " DSSSL에 기반한 SGML 문서편집기의 설계 및 구현", *정보과학회논문지(C), 제 1권 제 6호*, 1998.