SGML 문서 저작 도구

안 보 희[†] · 유 재 우[††] · 송 후 봉[†††]

요 약

ISO 8879로 정의된 SGML은 문서의 논리적 구조를 정의하는 메타언어로서 전자문서의 기본 형식으로 많이 사용되고 있다. SGML 문서는 문서구조 정의와 이에 따라 작성된 실제문서로 구성되므로 저작 도구는 문서구조와 실제문서를 모두 작성하고 검증할 수 있어야 한다. 그러나 SGML 문서처리를 위한 정형화된 모델과 절차가 존재하지 않으므로 이러한 도구의 구성이 쉽지 않다. 본 연구에서는 SGML 구문분석기, 문서구조 정의를 위한 편집기, SGML 문서 편집기 및 형식 편집기 등으로 구성된 모형과 각 구성 요소의 정형화된 처리 방법을 제안하고 구현하였다. 사용자의 편의를 위하여 아이콘 기반의 시각 프로그래밍 기법을 사용하였으며, 한글 문제점들을 해결하는 통합적 문서 저작 환경을 윈도우즈 NT 시스템에서 Java와 C++ 언어를 사용하여 구현하였다.

An SGML Document Authoring Tool

Bo-Hee An[†] · Chae-Woo Yoo[††] · Hoo-Bong Song[†††]

ABSTRACT

SGML, defined as the ISO 8879, is a meta-language to define a document type, used as basic format for electronic documents. Since an SGML document is composed of a document type definition and a document instance conforms to the definition, it is necessary for SGML document authoring tools to compose and validate document type and document instance. In present, formal models and procedures for SGML documents are not defined, it's not easy to construct such tools. We propose a model of SGML authoring tool consists of SGML parser, document type definition editor, SGML document editor and style editor. We also introduce and implement formal procedure for each component. For user convenience, we adopted icon based visual programming method, and solved the HANGUL problem. The SGML authoring tool is implemented in Windows NT system using Java and C++ programming language.

1. 서 론

SGML(Standard Generalized Markup Language)은 컴퓨터간에 효율적인 문서교환을 위하여 국제표준화기구(ISO)에서 ISO 8879로 제정된 표준 규약이다[6,10]. SGML은 임의의 문서형태나 응용에 대하여 범용적인 마크업 언어(mark-up language)의 구문을 정의한 메타 언어[6]로서 특정 문서 형식에 구애받지 않고, 자체적으로 문서 형식을 정의하며, 문서의 개념적인 논리구조을 가지고 있어 복잡한 문서도 작성할 수 있고, 그 내용도 검증할 수 있다[1,2]. 따라서 복잡한 문서 정보뿐만 아니라 멀티미디어 정보, 하이퍼 정보까지도 처리 가능하다[3,8]. 이러한 이유로 최근 SGML은 전자문서의 기본 도구로 사용되어, 전자도서관이나 CALS(Commerce At Light Speed), IETM (Interactive Electronic Technical Manual) 등의 분야에 널리 사용되고 있다[4].

† 준 회 원 : (주)상지소프트 연구소장
†† 정 회 원 : 숭실대학교 컴퓨터학부 교수
††† 정 회 원 : 숭실대학교 전자계산학과 교수

논문접수 : 1998년 9월 23일, 심사완료 : 1998년 12월 3일

SGML을 이용하여 작성된 문서를 SGML 문서라고 하며, 이 문서는 SGML 선언부와 문서형 정의부(Document Type Definition: DTD), SGML 실제문서의 세 부분으로 구성된다[2]. 이렇게 세 부분으로 나뉘어 작성된 SGML 문서는 SGML 구문분석기(parser)에 의해 올바른 작성여부가 점검된다[5].

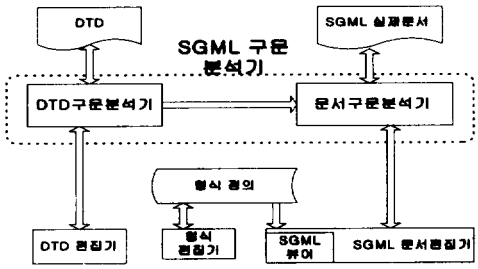
현재 SGML 문서를 저작하고 관리할 수 있는 상용 도구로는 Arbor Text의 ADEPT, Xsoft의 SGML Editor, Frame의 Frame Builder, Nissho Iwai의 SGML-plus, Opentext의 Opentext5, Xerox의 DocuBuilder1.2 등이 있다[13]. SGML 도구들은 크게 기존 워드프로세서나 탁상출판 시스템에 SGML 저작 기능을 부가하는 형태와 SGML을 기초로 한 저작 도구 형태로 분류할 수 있다. 본 시스템은 SGML을 기초로 한 저작 도구이다.

본 연구에서는 기존 SGML 지원 도구들이 갖고있는 다음과 같은 문제점들을 해결하기 위한 방안을 제안하고자 한다. 첫째, SGML 문서 처리에 필수적인 구성 도구들 중에서 몇 가지만이 제공되고 있어서 실제 처리에 어려움이 있다. 둘째, 내장하고 있는 SGML 구문분석기는 점진적 구문분석을 지원하고 있지 않기 때문에 구문분석 작업의 효율성이 떨어지며, 또한 오류들을 명확하게 알려주지 못함으로써 오류 수정이 비효율적이다. 셋째, DTD 작성을 문서형식표현 언어로 정의하기 때문에 작업의 난이도와 복잡도가 크다. 따라서 문서의 구조가 복잡한 경우에 DTD 작성 환경이 시각적으로 되어있지 않거나, 사용이 쉽게 되어있지 않은 도구는 DTD 작성이 매우 어렵게 된다. 넷째, SGML 실제문서를 작성하거나 편집할 경우, 작업중인 문서구조를 사용자가 쉽게 이해할 수 있는 형태로 보여줄 수 있어야 하며, 원하는 출력 형식으로 출력할 수 있어야 한다. 또한 마지막으로 고려하고자 하는 것은 한글 처리이다. 현재 국내에서 사용되고 있는 대부분의 SGML 지원 도구들은 국외에서 개발되어 국내에 수입된 제품들이기 때문에, 메뉴나 도움말만을 한글화하여 제품을 공급하고 있어서, 한글 문서들을 SGML 문서화하는데 어려움이 있으며, 문서의 태그 명을 한글로 정의할 수 없는 등의 큰 문제점을 가지고 있다. 이러한 문제점들을 해결하기 위하여 본 논문에서는 SGML 구문분석기, DTD 편집기, SGML 문서 편집기, 형식 편집기(style editor) 등으로 구성 통합된 저작 환경을 제안한다. 이 저작 도구는 사용자의 편의성을 고려하여, 시각적 프로그래밍 기법을 도입하였으며, 한글 문서 및 한글 DTD를 지원한다.

본 논문은, 2장에서 SGML 저작 도구의 구성 및 구현, 3장에서 실험 및 고찰, 마지막으로 4장에서 결론 및 향후 연구과제를 논한다.

2. 구성 및 구현

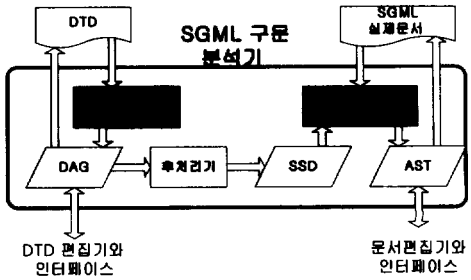
본 연구에서 SGML 문서 저작 시스템은 (그림 1)과 같이 구성하였다. SGML 구문분석기는 입력된 DTD와 SGML 실제문서가 표준 SGML 문법에 적합한가를 점검하고, DTD 편집기는 문서구조를 정의하고 편집하며, SGML 문서 편집기는 SGML 실제문서를 작성하고 편집하며, 형식 편집기는 출력 형식을 지정한다. SGML 뷰어는 편집중인 문서의 형식을 지원하며, SGML 문서 편집기에 통합되어 있다.



(그림 1) SGML 문서 저작 도구의 구성
(Fig. 1) Structure of SGML document authoring tool

2.1 SGML 구문분석기

구현된 SGML 구문분석기는 (그림 2)와 같은 구조로 DTD 구문분석기와 문서 구문분석기로 구분되며 DTD와 SGML 실제문서를 입력으로 그 문서의 적합성을

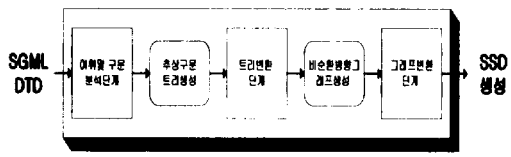


(그림 2) SGML 구문분석기 구조
(Fig. 2) Structure of SGML parser

점검한다. SGML 문서에 대한 문서구조 규칙을 알기 위하여 DTD가 DTD 구문분석기에 의해서 먼저 처리되고, 처리된 DTD 정보에 따라 SGML 실제문서들이 문서 구문분석기에 의해서 처리된다. 구문분석기 구현은 일반 프로그래밍 언어의 컴파일러 처리 단계를 적용하여 어휘분석 단계와 구문분석 단계를 거쳐 처리된다.

2.1.1 DTD 구문분석기

DTD 구문분석기는 입력된 DTD가 표준 ISO 8879 SGML 문법규칙에 적합한가를 판단한다. 먼저 어휘분석 단계에서는 참조구체구문 구분자 집합에 따라, 처리해야 하는 마크업을 인식하기 위하여 표준에서 정의한 10개 인식모드(CON, CXT, DS, DSM, GRP, LIT, MD, PI, REF, TAG)를 적용하였다 DTD의 구문분석 처리 1단계로 입력된 SGML DTD는 문맥자유문법에 따르는 일반 프로그래밍 언어의 구문분석 방법을 이용하여 어휘 및 구문분석을 한다. SGML DTD의 구문분석은 LALR 방식을 채택하였다. 구문분석 된 정보는 추상구문 트리를 생성하며, 트리 변환 단계를 거쳐, 입력된 DTD 정보를 DTD 편집기에서 편집 시 사용되는 그래프 자료구조인 비 순환 방향 그래프(Direct Acyclic Graph : DAG)로 생성한다. 이 DAG는 SGML 문서를 편집할 때 효율적인 문서구조 정보로 사용하기 위하여 후처리기에서 SSD(syntax & semantic definition)로 재 표현한다. (그림 3)은 DTD에서의 구문분석 처리 과정이다.



(그림 3) DTD 구문분석 처리 과정
(Fig. 3) DTD parsing

SGML 실제문서에서 나타나는 엘리먼트 및 문자열은 해당 SGML 실제문서를 미리 살펴보지 않고도 그것을 만족하는 기초내용 토큰은 하나만으로 정해져야 한다[5]. 따라서 DTD 내용 중 직접적인 문서구조 정보를 정의하는 엘리먼트 선언은 '구문변수 = 식' 과 같이 형식적인 생성규칙에 따라 표현할 수 있다. 그러므로 DTD 문법은 키워드 DOCTYPE과 엘리먼트 명인 GI 그리고 한 개 이상의 엘리먼트 선언으로 이루어진다. 이를 생성규칙에 따라 표현하면 (그림 4)와 같다.

```
DTD = MDO, "DOCTYPE", GI,
        엘리먼트-declaration+, MDC
엘리먼트-declaration =
        MDO, "엘리먼트", 엘리먼트-type,
        omitted-tag-minimization,
        (declared-content | model-group), MDC
엘리먼트-type = GI
              | name-group
model-group =
   GRPO, content-token, ((AND, content-token)*|
   (OR, content-token)* | (SEQ, content-token)*)
   GRPC, (OPT | PLUS, REP)?
Content-token = "#PCDATA"
              | (GI, (OPT | PLUS | REP)?)
              | model-group
declared-content = "CDATA"
                 | "RCDATA"
                 | "EMPTY"
omitted-tag-minimization = start-tag-minimization,
                           end-tag-minimization
start-tag-minimization = "O"
                       | MINUS
end-tag-minimization = "O"
                     | MINUS
```

(그림 4) DTD 문법
(Fig. 4) DTD grammar

DTD 구문분석 처리 과정을 거치면서 SSD가 생성될 때, <표 1>과 같은 SSD 생성 규칙들이 적용된다.

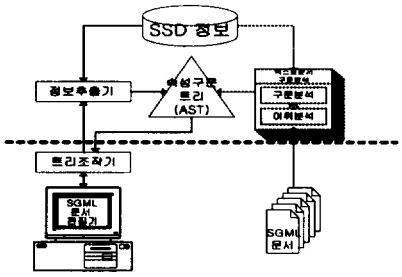
<표 1> SSD 생성
<Table1> SSD document production rule

생 성 규 칙	설 명
A⇒PROD_ONE (B)	A 태그 이후 B 태그만이 나온다.
A⇒PROD_NULL ( )	A 태그 이후 어떤 태그도 나오지 않는다.
A⇒PROD_PLUS (B)	A 태그 이후 B 태그는 한번 이상 반복적으로 나온다.
A⇒PROD_STAR (B)	A 태그 이후 B 태그는 안 나올 수 있거나 여러 번 반복적으로 나온다.
A⇒PROD_AND (B C)	부 엘리먼트 B 와 C는 AND 연결자로 연결되어 있어, A 태그 이후 B 태그가 나오고 C 태그가 나올 수 있고 또는 C 태그 후 B 태그가 나올 수 있다.
A⇒PROD_PAIR (B C)	부 엘리먼트 B 와 C는 SEQ 연결자로 연결되어 있어, SGML 문서에서 A 태그 이후 반드시 B 태그, C 태그 순서로 나온다.

2.1.2 문서 구문분석기

SGML 실제문서는 정해진 DTD에 따라 만들어져야 하므로, SGML 문서 편집기를 이용하여 실제문서를 작성하면 사용자의 부담을 덜어줄 수 있다. 따라서 본 연구에서는 SGML 문서 편집기를 이용하여 SGML 실제문서를 처리하는 SGML 문서 구문분석에 대하여 언

급한다. (그림 5)는 문서 구문분석기의 처리 과정이다.



(그림 5) 문서 구문분석기 처리 과정
(Fig. 5) Processing of document parser

구문분석 결과는 SSD 정보와의 인터페이스를 위한 형태와 SGML 시스템과의 호환을 위한 형태로 저장된다. SSD 정보와의 인터페이스를 위해서 AST(Attribute Syntax Tree : AST) 구조 형태로 저장되며, SGML 시스템과의 호환을 위해서는, 일반 다른 구문분석기를 이용하는 일반 텍스트구조 형태로 저장된다. 또한 이미 텍스트구조 형태로 작성되어 있는 SGML 실제문서는 SSD 정보의 문서생성 규칙을 구문분석 정보로 하여 구문분석 한 후 AST 형태로 재 표현하여 처리한다. (그림 6)은 SGML 문서의 AST 노드구조로, (그림 15(a))의 실제구조이다

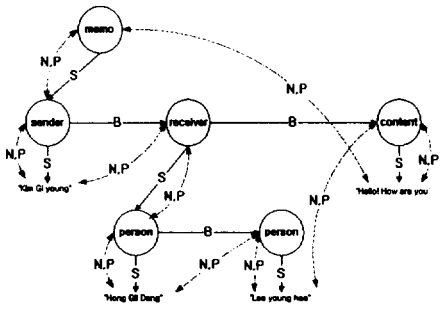
<table>
<tr><td colspan="4">태그 명</td></tr>
<tr><td colspan="4">문서 생성 규칙 정보</td></tr>
<tr><td colspan="4">문서 생성 규칙의 연결 · 지시자 정보</td></tr>
<tr><td colspan="4">속성 정보</td></tr>
<tr><td colspan="4">실제문서 텍스트 문자열</td></tr>
<tr><td colspan="4">반복 발생 가능 노드 정보</td></tr>
<tr><td colspan="2" rowspan="2">예외 정보</td><td colspan="2">포함 태그 정보</td></tr>
<tr><td colspan="2">배제 태그 정보</td></tr>
<tr><td>자식노드연결</td><td>형제노드연결</td><td>이전노드연결</td><td>다음노드연결</td></tr>
</table>

(그림 6) AST 노드구조
(Fig. 6) Structure of AST node

AST 노드의 연결 정보는 자식노드, 형제노드, 이전노드, 다음노드의 4개의 노드로 구성된다. 이 연결 정보는 SGML 문서 편집기에서 작성자에 의해 선택된 현재 태그(AST상의 현재 노드)를 중심으로 문서의 AST를 순회하게 된다. 즉 AST 상의 모든 노드는 서로 순환적으로 연결되어 있고, AST 노드 순회 과정에 의해 수정 작업이 이루어진다. (그림 7(a))는 SGML 문서 편집기에서 작성된 SGML 실제문서이고, (그림 7(b))는 AST의 노드 순회 예이다. SGML 문서 편집기에서 메모 DTD를 이용하여 작성된 실제문서 (그림 7(a))는 (그림 7(b))와 같이 태그를 중심으로 작성된 AST 트리구조를 동시에 갖게된다. (그림 7(b))의 N은 다음노드 연결, P는 이전노드 연결, S는 자식노드 연결, B는 형제노드 연결을 의미한다.

```
<memo>
  <sender>Kim Gi young</sender>
  <receiver>
    <person>Hong Gil Dong</person>
    <person>Lee young hae</person>
  <content>Hello! How are you</content>
</memo>
```

(그림 7(a)) SGML 실제문서
(fig. 7(a)) SGML document



(그림 7(b)) AST 노드 순회 경로
(Fig. 7(b)) AST node traverse path

2.2 DTD 편집기

DTD는 문서의 논리구조를 정의하므로 DTD를 쉽게 설계하고 생성하기 위해서는 문서구조를 시각적으로 보여주고, 쉽게 편집할 수 있도록 해야한다. 이를 위해 엘리먼트, 속성, 엔티티등 DTD의 기본이 되는 요소들을 화면상에 독립된 객체들로 표현하고 이들 간의 관계는 트리와 유사한 구조로 나타내며, 메뉴, 툴바, 클립보드, 마우스를 이용한 끌어다 놓기 방법으로 변경할 수 있도록 하였다. 또한 속성, 엔티티 등과 같은 보조적인 객체는 시각적인 간결함을 유지하기 위하여 화면상에 나타내지 않고 독립적인 다이얼로그를 이용

하여 편집할 수 있게 하였다. 이렇게 얻어진 DTD는 고유의 양식뿐만 아니라 텍스트 파일로도 저장할 수 있도록 해서, DTD 편집기 외부의 SGML 시스템과도 쉽게 데이터를 교환할 수 있게 하였다. 사용자의 이해와 작업을 쉽게 하기 위해 트리 표현 방식을 택하였으며, 그래픽 상태에서 직접 조작하여 편집하는 방식을 취하였다.

2.2.1 DTD 시각기호

(1) 엘리먼트 정보

DAG의 노드 정보는 DTD의 엘리먼트 정보를 기본으로 하여 설계하였다. DAG는 하향방식으로 생성되는 그래프로서, 그래프의 가장 상위 노드인 루트 노드로는 엘리먼트 선언의 기본문서 엘리먼트가 된다. 표현되는 노드 정보는 엘리먼트, 연결자, 지시자 노드의 3가지 형태로 구별되며, 지시자 노드는 엘리먼트 선언 내용 모델에서 표현되는 '?', '*', '+' 에 대한 정보를 갖게되고, 연결자 노드는 ',', '&', '|' 에 대한 정보를 갖는다. 따라서 이런 3가지 노드 정보는 구분 정보 필드에 의해 구분되는데, 이 필드 값이 엘리먼트이면 엘리먼트 노드로 처리되고 필드 값이 OPERATOR이면 지시자 노드나 연결자 노드로 처리된다. 이 노드는 엘리먼트 명, 시작과 끝태그의 생략 정보, 예외 정보 등을 갖게 되며, 이 예외 정보는 포함엘리먼트 정보와 배제엘리먼트 정보로 구분된다.

(2) 속성 정보

속성 선언은 의미적인 구조 정보를 표현한다. 속성 정보의 자료구조는 속성 명, 속성 선언 형태, 속성 선언 형태 값이 그룹이면 속성 값의 리스트, 속성 디폴트 형태, 속성 선언 형태의 값이 그룹일 때의 디폴트 속성 값, 다음 속성과의 연결 등의 정보를 가진다. 하나의 엘리먼트에는 여러 개의 속성이 선언될 수 있으므로 이것은 다음 선언과의 연결 필드에 의해 리스트 형태로 저장된다.

(3) 엔티티 정보

엔티티 정보는 어휘-구문분석 단계에서 만들어진다. 엔티티 정보는 테이블 형태로 저장되고 테이블 탐색키는 엔티티 명을 사용하였다. 만일 같은 엔티티 명을 가진 선언이 여러 개 존재한다면 먼저 선언된 것만이 유효하고 나중에 선언된 엔티티는 경고를 출력한다. DTD에서 엔티티는 일반 엔티티, 매개변수 엔티티의 2가지 타입으로 구분하므로 일반 엔티티 테이블과 매개변수 엔티티 테이블로 분리하여 관리한다. 엔티티 정보는 (그림 8)과 같은 구조로 설계, 구현하였다.

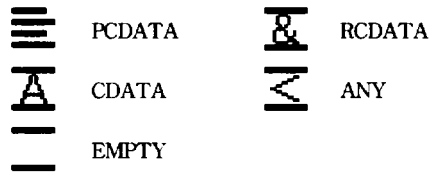
(4) DTD 시각표현

시각표현 방식은 트리구조로, 아이콘을 이용하여 시각화하였다. 이들 이외의 다른 엔티티들은 시각적으로 접근할 수 없고 다이얼로그 등을 통해 편집해야 한다. DAG구조인 DTD를 트리 형태로 표현하기 위해, 하나의 노드에 대한 참조가 중복되는 경우, 그 노드를 중복 표시하였으며, 중복 노드들은 동시에 확장될 수 없도록 제한하여 일관성을 깨뜨리는 노드의 수정을 방지하였다.

① 엘리먼트와 엔티티

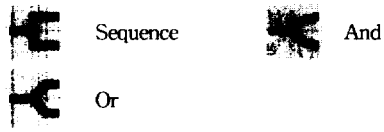
(그림 9) 엘리먼트 시각기호
(Fig. 9) Element icons

② 텍스트 데이터(터미널)



(그림 10) 텍스트 데이터 시각기호
(Fig 10) Text data icons

③ 엘리먼트 접속자

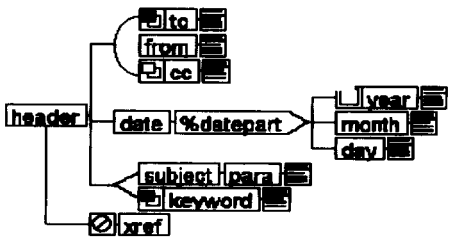


(그림 11) 엘리먼트 접속자 시각기호
(Fig. 11) Element concatenator icons

④ 엘리먼트 발생지시자

(그림 12) 엘리먼트 발생 지시자 시각기호
(Fig. 12) Element occurrence indicator icons

(그림 9)에서부터 (그림 12)의 시각기호를 이용하여 (그림 13(a))와 같이 작성된 시각 DTD는 (그림 13(b)) 의 텍스트 DTD와 동등한 문서구조이다.



(그림 13(a)) DTD의 시각적 표현
(Fig. 13(a)) Graphical representation of DTD

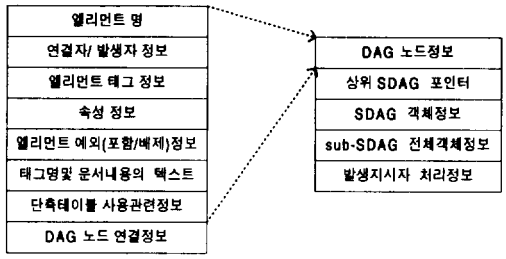
```
<!DOCTYPE header [
<!ENTITY % dateprt "(year?, month, day)" >
<!ELEMENT header - -
 ((to+, from, cc*), date, (subject | keyword+))
                                    -(xref)>
<!ELEMENT to - - (#PCDATA)>
<!ELEMENT cc - - (#PCDATA)>
<!ELEMENT date - - (%dateprt;)>
<!ELEMENT year - - (#PCDATA)>
<!ELEMENT month - - (#PCDATA)>
<!ELEMENT day - - (#PCDATA)>
<!ELEMENT subject - - (para)>
<!ELEMENT para - - (#PCDATA)>
<!ELEMENT keyword - - (#PCDATA)>
<!ELEMENT xref - - (#PCDATA)>
]>
```

(그림 13(b)) 텍스트 DTD
(Fig. 13(b)) Text DTD

2.2.2 내부 자료구조

화면에서 DTD 선언문을 그래픽 적으로 표현하기 위하여 부가적인 화면정보와 편집 시 필요한 정보를 저장할 수 있는 자료구조가 요구된다. 이에 본 논문의 DTD 편집기는 DTD 구문분석기에 의해 생성된 DAG 노드 정보와 화면에 그래픽으로 표현하기 위한 그림자 DAG 즉, SDAG(shadow DAG) 노드 정보의 2가지 자료구조를 이용한다. (그림 14)는 DAG 노드 정보(a)와 SDAG 노드 정보(b)를 도식화한 것이다.

DAG 노드는 엘리먼트 선언과 관련된 모든 정보의 집합체로서 DAG 노드 연결정보에 의해 문서의 구조가 정립된다. SDAG 노드는 화면편집을 위해 객체의 위치 및 크기를 비롯한 부가적인 표현 정보를 저장한다. 상위 SDAG 포인터는 노드들을 순회할 때 사용되며, SDAG 객체 정보는 터미널 노드의 위치 및 크기를 기억하는 사각형 객체이다. sub-SDAG 전체객체 정보는 하위 노드 전체를 포함한 사각형의 위치 및 크기를 기억하고 있으며, 발생 지시자 처리 정보는 DAG와 sub-DAG가 가질 수 있는 발생 지시자를 화면상에 그려주기 위한 정보이다.



(그림 14) DAG와 SDAG 노드정보
(Fig. 14) Node information of DAG and SDAG

2.2.3 편집 기능

DTD 편집기에서는 구문분석기에서 추출한 정보를 이용하여 아이콘과 시각기호로 문서의 구조를 표시하며 이를 직접 조작할 수 있는 기능을 제공한다. 편집기의 주요 기능은 DTD의 표기법 수정, DTD의 외부 매개변수 엔티티 수정, 속성 관련 작업 등 이다.

2.3 SGML 문서 편집기

SGML 문서 편집기는 SGML 문서를 작성하는 도구로서 크게 문서구조 뷰어, 문서 편집기, SGML 뷰어의 3부분으로 구성되어 있다.

문서구조 뷰어는 작성 중인 문서의 구조를 트리 형식으로 표현하여 사용자에게 문서구조에 대한 정보를 제공하고, 문서태그의 삭제나 추가가 가능하게 함으로써 사용의 편의성을 높였다. 또한 문서를 실제로 편집할 때, 사용자 편의성 측면에서 색으로 구분함으로써 사용자의 혼란을 감소시켰다. SGML 뷰어는 현재 작업 중인 SGML 문서에 출력 양식을 지정하거나, 미리 저장되어 있는 형식 파일을 적용함으로써 출력 형태를 필요할 때마다, 즉시 확인 가능하다. 따라서 원하는 출력 양식을 실시간으로 수정, 최상의 출력을 얻을 수 있으며, DTD 편집기에서 정의한 형식 정의에 의해 출

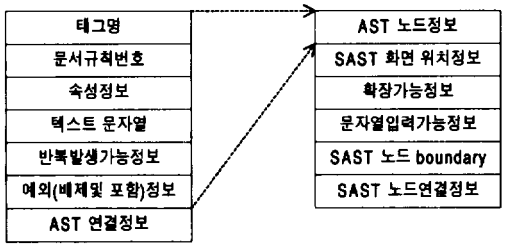
력 형식이 적용된, 미리 보기 기능으로 최종 출력물에 대한 정보를 사용자에게 제공한다.

2.3.1 내부 자료구조

SGML 실제문서를 편집하는 문서 편집기는 SSD의 문서구조 정보를 이용한다. 편집기에 입력되는 문서내용은 문서 구문분석기에 의해 SSD를 중심으로 문서에 대한 AST가 생성, 조작 및 관리된다. 문서 편집 시 태그의 조작이나 문서 내용의 변경 등은 SSD에 부합하여야 하고, AST의 정보를 일관성 있게 유지시켜야 한다. AST 노드는 태그 명, 속성 정보, 노드간의 링크로 구성되어 있으며, AST의 자료구조는 (그림 15(a))로 (그림 6)과 같은 구조이다.

본 논문에서 구현된 문서 편집기는 구문지향 적이면서 시각적인 기능을 가지며, 문서 구문분석기에 의해 생성된 AST를 화면에 표현하는데 별도의 자료구조인 그림자 AST(Shadow AST: SAST)를 사용하였다. 이 SAST 노드는 AST 노드를 중심으로 시각적인 요소를 첨가하여 생성한 노드로 (그림 15(b))와 같은 구조이다.

SAST 화면위치 정보는 시작태그의 위치를 나타내고, SAST 노드 경계는 시작태그의 경계로 이것은 끝태그의 위치 정보가 된다. 확장가능 정보는 태그의 색을 구별하는 정보이고, 문자열 입력 가능 정보는 해당 태그가 문자열을 입력받을 수 있는 태그인지를 구별하는 정보이다. SAST 노드 연결 정보는 화면에 출력되는 SAST 노드들의 연결 정보이다.



(그림 15) AST와 SAST 자료구조
(Fig. 15) Data structure of AST and SAST

2.3.2 편집 기능

SGML 실제문서 작성은 DTD에서 정의된 규칙에 의해 논리구조를 나타내는 태그의 처리와 문서의 실제 내용인 텍스트 처리로 구분될 수 있다. 문서 편집기의 주요 기능은 태그 수정, 입력 데이터 수정, 사용자 입력 제한, 한글 처리, 미리 보기 기능 등이다.

2.4 형식 편집기

SGML에서는 구조에 기반 한 문서처리를 원칙으로 하기 때문에 화면이나 인쇄 매체로의 출력 형태를 지정할 수 있는 방법이 없다. 그러므로 이러한 점을 보완하기 위해서는 SGML 문서 내에 존재하는 엘리먼트들의 출력 형식을 DTD와는 별도로 지정한다.

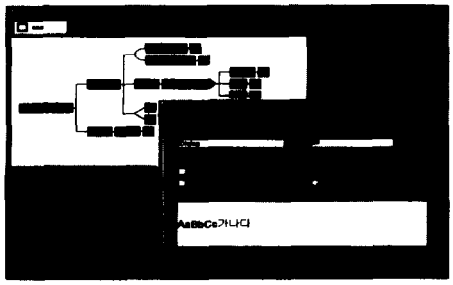
문서의 출력 형식을 지정하기 위한 표준에는 여러 가지가 있으나, 그 중 대표적인 것이 CSS1(Cascading Style Sheet 1)과 DSSSL(Document Style Semantics and Specification Language)이다. CSS1은 W3C (World Wide Web Consortium)에서 HTML(Hypertext Markup Language) 문서에 쉽게 형식을 정해주기 위하여 제정한 것으로, 사용자가 사용하기 쉽게 되어 있으며, 보통 탁상 출판에서 쓰이는 용어로 형식을 표현한다. DSSSL은 SGML 문서의 다양한 처리 정보를 기술하기 위한 국제 표준이다.

본 논문의 형식 편집기에는 CSS1을 사용하였으며, DSSSL의 기능을 참조하였다. 이 형식 편집기는 특정한 DTD에 대한 엘리먼트들의 형식을 정해주고, 이를 SGML 브라우저에서 인식할 수 있는 형태로 파일에 저장하는 프로그램이다. 형식 설정 항목으로는 글꼴, 색상과 배경, 텍스트, 상자, 분류 설정 등 출력 형태에 관한 속성들을 각 엘리먼트 별로 편집하는 기능을 제공한다. 이는 CSS1은 사용자가 이해하기 쉽다는 장점이 있으나, 웹 브라우저와 같은 온라인 표시에 적합하도록 구성되어 있어, SGML에서 그대로 사용하기에는 적절하지 못한 면이 있고, 반면 DSSSL은 페이지 설정과 다단 설정 기능이 포함되어 있어, 실제 출력을 염두에 두어야 하는 SGML 저작 도구에 CSS1보다 강력한 기능을 제공한다. 그러나 DSSSL의 전체적인 내용은 사용자가 사용하거나 이해하기에 너무나 방대하기 때문에, 본 논문에서는 DSSSL에서 꼭 필요한 내용만을 선택하여 적용함으로써 사용이 용이하도록 하였다.

3. 실험 및 고찰

본 논문은 SGML 문서 작성을 위한 통합된 저작 도구를 제안하였다. 제안된 저작 도구의 구현과 실험

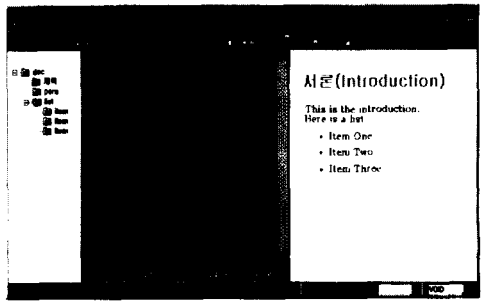
은 Windows NT 시스템에서 수행되었다. DTD 편집기와 형식 편집기는 Java 언어를 사용하였고, SGML 문서 편집기와 SGML 뷰어는 Visual C++를 사용하여 구현하였다. (그림 16)은 DTD 편집기와 특정 엘리먼트의 출력 형식을 정의하는 형식 편집 기능의 실행화면을 보이고 있다. 트리를 이용한 DTD 편집은 사용자에게 전체 DTD구조에 대한 이해와 편집 작업을 용이하게 한다.



(그림 16) DTD 편집기 실행화면
(Fig. 16) DTD editor snapshot

(그림 17)은 SGML 문서 편집기의 실행화면으로, 세 개의 부 화면으로 구성된다. 첫 번째, 왼쪽에 나타나는 화면은 편집중인 문서의 논리구조를 트리 형태로 보여주는 문서구조 뷰어이다. 두 번째로 가운데 화면은 실제로 문서를 편집하는 문서 편집기이고, 오른쪽에 있는 세 번째 화면은 편집된 문서에 형식 내용을 적용시킨, 최종 출력 형식을 미리 보기 형식으로 보여주는 SGML 뷰어이다. 이와 같은 화면구성은 편집자에게 최소의 조작으로 문서의 편집을 가능하게 한다. 이 SGML 문서 편집기는 다음과 같은 특징을 가지고 있다.

첫째, 통합된 저작 도구를 제시하고 있으며, SGML 구문분석기는 점진적 구문분석 방법을 적용함으로써 구문분석 과정의 효율성을 높였고, 따라서 오류가 발생하면, 이를 출력하는 기능을 제공함으로써 신속하게 수정할 수 있다. 둘째, 사용자는 SGML 실제문서를 편집하는 중에 항상 그 문서의 논리적 구조를 파악하고 있어야만 하므로, SGML 문서 편집기의 문서구조 창을 통해 문서의 구조를 트리 형태로 보여줌으로써 사용자가 편집 중인 문서의 문맥구조를 실시간으로 한눈에 파악할 수 있도록 하였다. 셋째, SGML 문서의 형태를 정의하는 표준인 CSS1, DSSSL 등의 활용 기술을 형식 편집기에 적용함으로써, 사용자가 원하는 형식 속성을 자유롭게 편집할 수 있으며, 고품질의 출력 결과를 얻을 수 있다. 마지막으로 기존 SGML 도구들은 한글 문서들을 SGML 문서화하는데 어려움이 있으나, 본 연구에서는 한글 처리와 관련하여 SGML 편집기에 한글 입력 기능과 그밖에 한글 DTD를 구문분석 할 수 있는 DTD 구문분석기를 내장함으로써, 문서의 태그 명을 한글로 사용하여 한글 문서의 구조적 특징을 적용할 수 있는 장점이 있다.



(그림 17) SGML 문서 편집기 실행화면
(Fig. 17) SGML document editor snapshot

본 연구에서 제시한 SGML 저작 도구와 국내외 SGML 저작 도구와의 성능을 비교하면 <표 2>와 같다.

〈표 2〉 국내외 저작 도구와의 성능 비교
〈Table 2〉 Performance comparison among other tools

성능 구분	본 논문 제시 도구	국내 유사 도구(HISOS)	해외(Author/Editor)
관련도구 통합성	통합된 패키지로 제공	유사 기능 제공	DTD 편집 기능 없음
점진적구문분석 및 오류검사	SGML 구문 분석기에 내장	해당기능 없음	해당 기능 없음
문서의 시각적 표현	트리 형태로 표현. 아이콘과 태그 색으로 구분	유사기능을 제공하나 미비한 부분존재	유사 기능을 제공하나 미비한 부분존재
CSS1, DSSSL을 지원	형식 에디터에서 지원	형식에디터에서 DSSSL과 같은 표준을 적용 못함	편집기에서 형식 지정으로 표준을 적용 못함
자유로운 한글 사용	엘리먼트의 입력 데이터에 한글 지원으로 구조적 의미의 이해가 가능	유사 기능 제공	한글 처리를 고려하지 않으므로 정확한 처리 불가능

4. 결 론

본 논문에서 제안한 SGML 저작 도구는 SGML 문서 저작을 용이하게 하며, SGML의 본래 취지에 부합되도록 최종 문서 작성자는 문서의 논리구조에만 집중하고 출력 형식에 관여하지 않도록 하는 편집 모델을 제시하였다. DTD 편집과 SGML 문서 편집 기능이 하나로 통합되어 있어서 문서 작성 시간을 단축할 수 있으며, 일반 워드프로세서 수준의 편집 기능은 물론 SGML 문서구조에 특수하게 요구되는 각종 대화형 편집 기능이 지원되어 편집 작업이 보다 쉬워지도록 하였다. 점진적 구문분석 및 오류검색 기능으로 인해 DTD에 대한 지식이 없어도 문서를 쉽게 작성할 수 있도록 하였으며, 또한 한글이 문서 내용과 태그에도 사용 가능해 짐으로써, 문서구조 파악이 더욱 용이해 졌으며, 이로 인하여 한글문서 수용능력이 탁월하다. 이러한 기능들은 다른 SGML 저작 도구에서는 지원되지 않는 것으로 이를 통해 거리감을 가졌던 사용자들도 거부감을 덜 수 있으리라 생각된다. 따라서 이 도구의 구현으로 SGML이 수요자 층을 넓혀서 범용 적으로 사용될 수 있으리라 사려되며, 따라서 문서 작성 메카니즘에 새로운 시대를 열 수 있으리라 기대된다.

향후의 연구 과제로는 편집중인 문서와 미리 보기 기능을 실시간에 동기화 하는 작업이 바람직하므로 이를 위해 점진적 SGML 구문분석기에 대한 추가적인 연구가 필요하다. 그리고 SGML 표준과 함께 출력 형식에 대한 표준인 DSSSL과 SPDL(Standard Page Description Language)의 구현도 남아 있는 과제이다. 또한 하이퍼 정보와 멀티미디어 정보를 처리하기 위해 HyTime(Hypermedia/Time-based Structuring Language) 등에 대한 연구도 병행되어야 할 것으로 본다.

참 고 문 헌

[1] B. E. Travis and D. C. Waldt, *The SGML Implementation Guide*, Springer, 1995.

[2] C. F. Goldfarb and Y. Rubinsky, *The SGML Handbook*, Clarendon Press. Oxford, 1990.

[3] E. van Herwijnen, *Practical SGML*, Kluwer Academic Pub., 1994.

[4] Eve Maler and Jeanne El Andaloussl, *Developing SGML DTDs From Text To Model To Markup*, Prentice-Hall, 1996.

[5] Jos Warmer, Sylvia vanEgmond, The implementation of the Amsterdam SGML, *Electronic Publishing*, Vol.2(2), pp.65-90, July 1989.

[6] M. Bryan, *SGML : An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, 1988.

[7] Martin Colby and David S. Jackson, *Using SGML*, QUE, 1996.

[8] Sean McGrath, *PARSEME.1ST SGML For Software Developer*, Prentice-Hall, 1998.

[9] Thomas W. Reps and Tim Teitelbaum, *The Synthesizer Generator*, Spring-Verlag, 1988.

[10] ISO, International Standard ISO/IEC8879 : *Information Processing-Text and Office System-Standard Generalzed Markup Language(ISO 8879)*, Geneva/Newyork, 1986.

[11] 한국산업표준심의회, *문서기술언어 SGML KS C 5913*, 한국표준협회발행, 1993.

[12] 현득창, SGML Parser를 이용한 SGML Document Editor의 구현에 관한 연구, *정보과학회 논문지*, 제20권 제4호, pp.484-494, 1993.

[13] 현득창, 이수연, DTD/SGML 문서 저작 도구의 설계 및 구현, *정보처리학회 논문지*, 제4권 제4호, pp.944-954, 1997.

안 보 회

e-mail : bhan@ss.soongsil.ac.kr

1975년 숭실대학교 전자계산학과 졸업(이학사)

1986년 숭실대학교 대학원 전자계산학과(공학석사)

1999년 숭실대학교 대학원 전자계산학과(공학박사)

1975년~1977년 숭실대학교 전자계산소 연구원

1978년~1980년 한국전자통신연구소 전임연구원

1981년~1985년 숭실대학교 전산원 전임강사

1986년~현재 (주)상지소프트 연구소장으로 재직중.

관심분야 : 시스템소프트웨어, 소프트웨어공학, 프로그래밍환경 등

유 재 우

e-mail : cwyoo@computing.soongsil.ac.kr

1976년 숭실대학교 전자계산학과 (학사)

1978년~1985년 한국과학기술원 전산학과(석사, 박사)

1986년~1987년 Cornell Univ. Visiting Scientist

1983년~현재 숭실대학교 컴퓨터학부 교수

관심분야 : 컴파일러, 프로그래밍 환경, HCI

송 후 봉

e-mail : hbsong@ss.soongsil.ac.kr

1956년 육군사관학교를 졸업

1986년 중앙대학교 전산학과 석사 학위 취득

1989년 조선대학교 전기공학과(전산기공학) 박사학위 취득

1971년~1997년 숭실대학교 전자계산학과 교수로 재직

1998년~현재 숭실대학교 전자계산학과 명예교수로 재직중

관심분야 : 운영체제, 프로그래밍언어, 컴퓨터보조학습 등