유사 구조를 가지는 XML 문서들의 DTD 통합 알고리즘

(A Unification Algorithm for DTDs of XML Documents having a Similar Structure)

유 춘 식 † 우 선 미 †† 김 용 성 †††
(Chun Sik Yoo) (Seon Mi Woo) (Yong Sung Kim)

요 약 논리적으로 동일한 종류에 속하여 서로 유사한 구조를 가지는 많은 XML 문서들이 서로 다른 종류로 분류되어 서로 다른 문서형 정의(DTD)를 가지게 되는 경우가 많다. 이로 인하여 XML 문서를 저장하기 위한 데이타베이스의 스키마가 서로 다르게 되고, 동일한 데이타베이스에 저장되어야 하는 XML 문서들이 서로 다른 데이타베이스에 저장되는 문제점이 발생하게 된다. 이러한 문제점을 해결하기 위하여 본 논문에서는 유한 오토마타와 트리구조를 이용하여 유사한 구조를 가지는 XML 문서들의 DTD를 통합하는 알고리즘을 제안한다. 유한 오토마타는 DTD의 반복연산자나 연결자를 표현하기에 적합하고 표현 방법이 단순하므로 DTD 통합 알고리즘의 복잡도를 감소시킬 수 있다. 또한 제안한 알고리즘의 효과성을 검증하기 위하여 국내 학회 논문지의 논문 DTD를 통합하는데 본 논문에서 제안한 알고리즘을 적용한다.

키워드 : XML DTD 통합, 트리, 유한 오토마타, 스키마 통합

Abstract *There are many cases that many XML documents have different DTDs in spite of* having a similar structure and being logically same kind of document. For this reason, it occurs a problem that these XML documents have different database schema and are stored in different databases. So, in this paper, we propose an algorithm that unifies DTDs of these XML documents using the finite automata and the tree structure. The finite automata is suitable for representing repetition operators and connectors of DTD, and is a simple representation method for DTD. By using the finite automata, we are able to reduce the complexity of algorithm. And we apply a proposed algorithm to unify DTDs of science journals.

Key words : XML DTD Unification, Tree, Finite Automata, Schema Integration

1. 서 론

최근 인터넷과 WWW의 대중화로 인하여 정보를 획득할 수 있는 정보원(information source)이 매우 다양해졌다[1]. 그러나 WWW이나 각종 데이타베이스 시스템은 정보를 표현하는 방법이나 정보 처리를 위한 데이타 모델이 서로 다르기 때문에, 이러한 정보원들에서 사용자가 필요로 하는 정보를 찾아내는 것은 많은 시간과 노력이 요구되는 어려운 작업이다[2-4]. 이러한 문제점을 해결하기 위한 방법으로 멀티미디어 데이타를 포함하는 디지털 문서를 정보의 내용에 대한 손실없이 이기종 시스템 사이에 효율적으로 저장·처리·전송하고, 사용자에게 다양한 검색 환경을 제공하기 위하여 XML (eXtensible Markup Language)[5]을 일반적으로 사용하고 있다[6,7]. XML은 구조적 문서를 표현하고 상호 교환하기 위한 전자문서 표현형식[3,6,7]으로서, XML은 문서의 개념적인 논리 구조와 내용 구조를 기술할 수 있으며, 멀티미디어 문서와 하이퍼링크를 포함하는 하이퍼미디어 문서까지도 작성할 수 있다[5]. 이러한 XML 문서들은 문서의 구조를 정의하기 위해 문서형 정의(DTD;Document Type Definition)를 선언하고 이를 이용하여 엄격한 문서 구조 검사를 수행한다.

이러한 DTD를 작성하는 문서 제작자, 제작 시기, 제작 환경 등에 따라 논리적으로는 같은 종류에 속하는 문서일지라도 서로 다른 DTD를 정의하여 사용하는 경

† 학생회원 : 전북대학교 전산통계학과
csyoo@chonbuk.ac.kr
†† 정 회 원 : 전북대학교 BK21 전자정보사업단 교수
smwoo@chonbuk.ac.kr
††† 종신회원 : 전북대학교 전자정보공학부 교수
yskim@chonbuk.ac.kr
논문접수 : 2004년 2월 27일
심사완료 : 2004년 8월 11일

우가 많다. DTD가 다르면 XML 문서가 저장되는 데이타베이스의 스키마가 다르게 되고, 데이타베이스의 스키마가 다르면 동일한 질의를 여러 데이타베이스에서 반복하여 수행해야 한다. 이로 인하여 정보 검색을 수행하기 위한 시간이나 복잡도가 증가하게 되고, 검색 결과의 질 또한 저하되게 된다. 이를테면, 국내외 학회의 논문지에 게재된 논문들의 구조를 분석해 보면 매우 유사한 구조를 가지고 있다는 것을 알 수 있다[8]. 그러나 이들을 XML 문서로 변환하는 과정에서 서로 다른 종류의 문서로 간주하게 되어 각 학회별로 고유한 DTD를 별도로 가지게 된다. 결국 개념적으로는 같은 종류의 문서들임에도 불구하고 서로 다른 데이타베이스에 저장하게 되며, 사용자의 질의를 처리하기 위하여 관련된 모든 데이타베이스에 접근해야만 한다. 결과적으로 XML 문서 데이타베이스를 구축하기 위한 비용과 질의 처리를 위한 데이타베이스 접근 횟수가 증가하게 되고, 검색의 효율성 또한 상당히 감소하게 된다. 따라서 유사한 구조를 가지면서 개념적으로 같은 종류에 속하는 XML 문서들의 DTD를 통합하여 하나의 통합 DTD를 생성하는 기법과 통합 DTD를 이용하여 XML 문서 데이타베이스를 구축하기 위한 방법에 대한 연구가 필요하다[2,3,6-9].

따라서 본 논문에서는 XML 문서에 대한 데이타베이스 구축에 소요되는 비용과 데이타베이스를 검색하는 과정에서 소요되는 비용을 줄이기 위하여, 트리 구조와 유한 오토마타(Finite Automata)를 이용하여 DTD를 표현하고 표현된 유사 DTD들(공통된 주제 영역에서 산출된 DTD들)을 자동으로 통합하는 알고리즘을 제안한다. 그리고 이를 통해 XML 문서 데이타베이스에 대한 검색 효율을 높이고 일관성 있게 XML 문서 데이타베이스를 관리·운영하기 위한 환경을 제공하고자 한다.

2. 관련 연구

유한 오토마타와 트리 구조를 이용하여 DTD를 모델링하는 연구들을 살펴보면, SGML 문서를 대상으로 하는 연구들에는 [10, 11, 12]이 있으며 [2, 6, 9]은 XML 문서를 대상으로 하고 있다. [10]은 동일한 문서 클래스에 대한 DTD를 생성하는 것을 목적으로 한다. 이 논문에서는 수작업으로 문서에 태그를 삽입한 후, 각 문서를 대상으로 삽입된 태그를 입력으로 하는 유한 오토마타를 작성하였다. 그런 후, 작성된 유한 오토마타를 병합하여 해당 문서 클래스에 대한 정규식을 도출하고, 도출된 정규식에 의해 DTD를 생성하는 방법을 기술하고 있다. [11]은 SGML 문서의 내용모델을 작성하기 위해 필요한 오토마타 이론과 집합 이론을 소개하고 있다. [12]은 SGML DTD를 자동으로 생성하기 위한 프로토타입을 만들어 문서 구조를 분석하였는데, DTD를 구성하는 요소들은 트리 구조로 표현하고 있다.

[6]은 XML 문서에 대한 질의 결과의 구조를 표현하기 위해, 정규 트리(regular tree)를 이용하여 XML DTD를 모델링하고, 트리 오토마타(tree automata) 모델에 기반한 XML 질의 대수를 제안하였다. [9]은 트리 문법 추론 기법(tree grammar inference technique)을 이용하여 이형(heterogeneous)의 XML DTD를 통합하기 위한 방법을 제안하고 있다. 이 논문은 유사 도메인의 DTD 사이의 이름짓기(naming)와 구조적 유사성을 이용하여 DTD 클러스트링을 수행하고, 각 클러스터를 대상으로 스키마 학습 모듈을 이용하여 스키마를 생성한다. [2]은 정보원들에 대한 동질적인 뷰를 제공하는 중개 스키마(mediated schema)를 기술하기 위하여, 정보원의 엘리먼트들과 중개 스키마의 엘리먼트들 사이의 사상을 자동적으로 생성하는 방법을 제안한다. 이 논문에서는 XML DTD를 대상으로 트리를 구축한 후, 시소러스를 이용한 용어 정합(term matching)을 이용하였다.

한편 [1]은 반구조화된(semi-structured) 정보원과 구조화된 정보원들을 통합하기 위하여 자동화된 구조 해결(automated structure resolution) 접근법을 사용하고 있다. 이 논문에서는 HDG(Hierarchical Data Graph)를 이용하여 정보원들의 구조적 차이를 해결하고 이를 통해 정보원들에 대한 단일 뷰(unified view)를 제공한다. [13]은 다중 전략(multi-strategy) 학습 접근법을 이용하여 정보원의 스키마와 질의 인터페이스 사이의 의미적 사상을 자동적으로 찾는 기법을 제안하고 있다. 이 논문에서 제안한 기법은 다수의 학습자 모듈을 이용하는데 각 학습자 모듈의 결과를 메타 학습자를 이용하여 조합하는 방법을 사용하였다. [7]은 객체지향 기반의 표준 스키마(object-oriented based canonical schemata)를 이용하는 이형의 XML 스키마에 대한 통합 기법을 제안하였다. 통합된 XML 스키마는 질의 처리과정에서 온톨로지(ontology)로 사용된다. [3]에서는 스키마 사이의 관계성을 기술하고, Telos라는 메타 모델링 언어를 이용하여 XML DTD를 공통 개념 스키마로 통합하는 기법을 제안하고 있다. [4]에서는 분산된 이형의 정보원에 대한 단일 질의 인터페이스를 제공하기 위하여, 지식 표현 기법을 이용해서 중개자(mediator)를 구축한다.

그러나 이들 연구들은 수작업으로 태깅된 문서들을 대상으로 하거나 단순한 DTD 모델링 방법만을 사용하여 구조적 유사성만을 통합하고 있다. 그리고 일부 연구들은 질의 인터페이스를 위한 온톨로지로 사용하기 위하여 DTD를 통합하는 것을 목적으로 하고 있다. 이로 인해 진정한 의미의 DTD 통합이라는 목적을 이루기는 부족하다. 따라서 개념적으로 동일한 종류에 속하는 구조적 문서의 DTD를 통합하는 통합 DTD 생성 방법이

필요하다[8].

3. XML DTD와 유한 오토마타

3장에서는 XML 문서의 논리적 구조를 정의하기 위한 XML DTD와 XML DTD를 유한 오토마타로 변환하는 방법에 대해 기술한다.

3.1 XML DTD

XML 문서의 논리적 구조에 대한 제한, 즉 XML 문서에서 사용할 태그와 이 태그들을 이용한 XML 문서의 논리적/계층적 구조는 문서형 선언(Document Type Declaration)을 통해 정의된다[5]. 즉 문서형 선언에서 해당 문서 클래스에 대한 문법을 제공하는 마크업 선언이 이루어지며, 문서 클래스에 대한 문법을 문서형 정의(DTD;Document Type Definition)라고 한다. 마크업 선언은 엘리먼트 형 선언, 속성 리스트 선언, 엔티티 선언, 표기법 선언 등을 포함한다[5].

(1) 엘리먼트 형 선언(Element Type Declaration)

엘리먼트가 가질 수 있는 내용에 대한 제한을 정의하는 부분으로서, 엘리먼트 형 선언에 의해 XML 문서의 논리적 구조(트리 구조)를 파악할 수 있다. 엘리먼트 형 선언은 해당 엘리먼트의 이름(엘리먼트 형)과 해당 엘리먼트가 가질 수 있는 엘리먼트 내용 부분으로 구성되며, 선언 형식은 다음과 같다.

<!ELEMENT 엘리먼트_이름 엘리먼트_내용>

엘리먼트 내용은 내용을 가지지 않는 빈 엘리먼트를 정의하기 위한 "EMPTY", 내용으로 임의의 엘리먼트나 문자 데이타를 가지는 엘리먼트를 정의하기 위한 "ANY", 자식 엘리먼트만을 내용으로 포함할 수 있는 엘리먼트를 정의하기 위한 내용모델(Content Model) 그리고 문자 데이타나 지정된 자식 엘리먼트를 내용으로 가지는 엘리먼트를 정의하기 위한 혼합 내용(Mixed Content) 부분으로 구성된다. 내용모델에는 자식 엘리먼트의 순서나 선택을 지정하기 위한 연결자(connector)와 자식 엘리먼트의 발생횟수를 지정하기 위한 반복연산자(repetition operator)를 사용할 수 있는데, 반복연산자에는 '?', '+', '*'가 있고 연결자에는 ',', '|'가 있다.

(2) 속성 리스트 선언(Attribute-List Declaration)

엘리먼트의 속성들과 이 속성들이 가질 수 있는 값들을 정의하는 부분으로서, 속성 리스트의 선언 형식은 다음과 같다.

<!ATTLIST 엘리먼트_이름 [속성_이름 속성_형 기본값_선언]* >

속성 형은 해당 속성의 값으로 사용할 수 있는 값에 대한 특성을 정의하는 것으로서, 문자열 형, 토큰형, 열거형이 있다. 기본값 선언은 해당 속성의 값을 필수적으로 지정해야 하는지의 여부와 해당 속성의 값이 주어지지 않았을 경우에 사용할 기본값(default value)을 정의한다.

(3) 엔티티 선언(Entity Declaration)

XML 문서 내에서 참조될 수 있는 각종 개체를 정의하는 부분으로서, 엔티티에는 XML 문서의 내부 어디에서나 참조할 수 있는 일반 엔티티(General Entity)와 DTD 내부에서 사용하기 위한 매개변수 엔티티(Parameter Entity)가 있다. 일반 엔티티를 선언하는 형식은 다음과 같다.

<!ENTITY 엔티티_이름 "엔티티_정의">

매개변수 엔티티는 DTD 내부에서만 정의되고 참조되는 엔티티로서, 선언 형식은 다음과 같다.

<!ENTITY % 엔티티_이름 "엔티티_정의">

(4) 표기법 선언(Notation Declaration)

XML 문서에 포함되어 있는 데이타 중에서 XML 처리기에 의해 처리되지 않고 특별한 처리가 필요한 엔티티들에 대한 표기법 이름, 해당 표기법 이름을 가지는 엔티티의 내용 데이타를 처리할 응용을 기술한다.

3.2 XML DTD의 유한 오토마타로의 변환

XML 문서의 구조는 주로 엘리먼트 형 선언에 의해 정의된다. 그리고 엘리먼트 형 선언 중에 엔티티를 포함할 수 있다. 그러므로 본 논문에서도 XML 문서의 구조를 통합하기 위한 대상을 엘리먼트와 엔티티로 한정한다.

XML DTD에 정의된 문서 구조를 통합하기 위해서는 먼저 DTD를 트리 구조와 유한 오토마타로 변환해야 한다. XML DTD를 트리 구조로 변환하는 방법은 간단하다. 즉 엘리먼트 형 선언 중의 내용모델과 혼합내용에서 연결자와 반복연산자를 무시하고, 단순히 내용모델과 혼합내용에 정의된 순서대로 부모 엘리먼트와 자식 엘리먼트의 계층적 구조만을 표현한다. 그 이유는 연결자와 반복연산자는 유한 오토마타를 이용하여 표현하기 때문이다. 예 1은 엘리먼트 형 선언을 트리 구조로 변환하는 예이다.

XML DTD의 내용 중에서 유한 오토마타로 변환되는 대상은 연결자와 반복연산자인데, 연결자에는 ',', '|'

예 1) DTD의 엘리먼트 형 선언을 트리 구조로 변환하는 예

DTD : <!ELEMENT 저자 (한글이름, 영문이름, 역할?, 회원구분, 전자우편주소?)+ >

변환된 트리 :

그림 1 트리 구조로 변환된 DTD

가 있고 반복연산자에는 '?', '+', '*'가 있다. 연결자와 반복연산자에 대한 기본 변환 방법은 그림 2와 같다. 반복연산자 중에서 '?' 연산자는 단독으로 사용되는 경우가 없으므로 제외하였다.

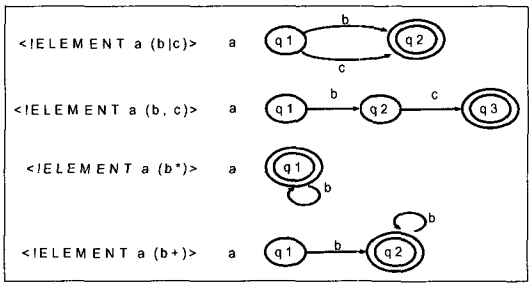

그림 2 기본적인 유한 오토마타로의 변환 방법

연결자와 반복연산자가 같이 사용되는 경우를 보면 표 1과 같다. 표 1은 자식 엘리먼트가 두 개일 때 반복연산자와 연결자의 가능한 조합의 경우를 나타내고 있다.

그림 3은 표 1의 각 예를 유한 오토마타로 변환하는 방법을 나타내고 있다. 자식 엘리먼트가 세 개 이상인 경우에는 자식 엘리먼트가 두 개인 경우를 반복적으로 적용하며, 이때 상태 변이는 자식 엘리먼트가 두 개인 경우의 최종 상태에서 다음 상태로의 변이가 이루어진다.

표 1 연결자와 반복연산자의 가능한 조합

가능한 조합		가능한 경우의 예	비고
연결자	반복연산자		
'|'	'?'	① <!ELEMENT a (b|c)>	
		② <!ELEMENT a (b?|c)>	의미없음
		③ <!ELEMENT a (b|c?)>	의미없음
		④ <!ELEMENT a (b|c)?>	의미없음
','	'?'	⑤ <!ELEMENT a (b,c)>	
		⑥ <!ELEMENT a (b?,c)>	
		⑦ <!ELEMENT a (b,c?)>	
		⑧ <!ELEMENT a (b,c)?>	의미없음
'|'	'*'	⑨ <!ELEMENT a (b*|c)>	
		⑩ <!ELEMENT a (b|c*)>	
		⑪ <!ELEMENT a (b|c)*>	
','	'*'	⑫ <!ELEMENT a (b*,c)>	
		⑬ <!ELEMENT a (b,c*)>	
		⑭ <!ELEMENT a (b,c)*>	
'|'	'+'	⑮ <!ELEMENT a (b+|c)>	
		⑯ <!ELEMENT a (b|c+)>	
		⑰ <!ELEMENT a (b|c)+>	
','	'+'	⑱ <!ELEMENT a (b+,c)>	
		⑲ <!ELEMENT a (b,c+)>	
		⑳ <!ELEMENT a (b,c)+>	

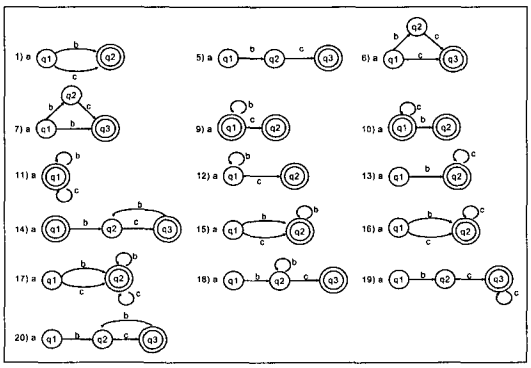

그림 3 표 1에 대한 유한 오토마타로의 변환

예 2는 엘리먼트 형 선언을 유한 오토마타로 변환하는 예이다.

예 2) DTD의 엘리먼트 형 선언을 유한 오토마타로 변환하는 예

DTD :

<!ELEMENT 저자 (한글이름, 영문이름, 역할?, 회원구분, 전자우편주소?)+ >

변환된 유한 오토마타 :

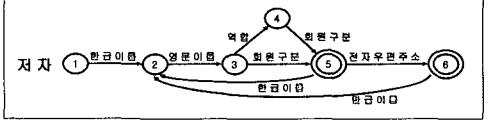

그림 4 유한 오토마타로 변환된 DTD

4. 통합 DTD 생성 알고리즘

4장에서는 논리적으로 유사한 구조를 가지지만 서로 다른 종류의 문서로 간주되어 각각 별도의 DTD를 가지는 문서들에 대하여 트리 구조와 유한 오토마타를 이용한 DTD 통합 알고리즘을 제안한다. 제안된 알고리즘은 MS Windows XP Professional 환경에서 "Java API for XML Processing(JAXP)"을 사용하여 구현하였으며, 국내 2개 학회 논문지의 논문 DTD 통합에 제안한 알고리즘을 적용하여 알고리즘의 성능을 확인하였다.

유사 구조를 가지는 XML 문서들의 DTD를 통합하는 가장 간단한 방법은 예 3과 같이 엘리먼트의 내용을 "ANY"로 선언하는 방법이다.

예 3) "ANY" 선언을 이용한 DTD 통합

DTD 1 : <!ELEMENT 학회논문 (논문지정보,논문정보,본논문정보,후방?)>

DTD 2 : <!ELEMENT 학회논문 (전방,본문,참고문헌,부록?,후방?)>

통합된 DTD :

<!ELEMENT 학회논문 ANY>

그러나 이 방법은 지나친 일반화(*generalization*)로 인해 각 XML 문서의 구조가 무시되어 의미 손실이 발생하므로 적합하지 않은 방법이다. 이러한 문제점을 해결하기 위하여 예 4와 같이 각 XML 문서의 구조를 유지하면서 DTD를 통합하기 위해 각 XML 문서의 DTD를 OR('|')로 묶는 방법도 있다.

예 4) OR('|')를 이용한 DTD 통합
DTD 1 : <!ELEMENT 학회논문 (논문지정보,논문정보,본논문정보,후방?)> DTD 2 : <!ELEMENT 학회논문 (전방,본문,참고문헌,부록?,후방?)>
통합된 DTD : <!ELEMENT 학회논문 ((논문지정보,논문정보,본논문정보,후방?)|(전방,본문,참고문헌,부록?,후방?))>

그러나 이 방법은 각 XML 문서의 구조는 유지되지만 통합하고자 하는 XML 문서의 개수가 두 개 이상인 경우나 통합하고자 하는 각 DTD가 복잡한 구조를 가지는 경우에는 단순히 OR로 각 DTD를 묶기 어려운 문제점이 있다. 그리고 OR로 묶을 엘리먼트를 결정하는 것도 어려우며 통합된 엘리먼트의 내용 모델이 매우 복잡하고 길어지게 된다. 또한 통합된 DTD를 파싱하는 과정에서도 애매모호(*ambiguity*) 문제가 발생할 수 있다. 이러한 문제점들을 해결하기 위해 통합 대상인 각 XML 문서의 구조를 최대한 반영하여 의미 손실을 최소화할 수 있는 DTD 통합 방법이 필요하다.

4.1 유사 DTD 통합 알고리즘

본 논문에서 제안하는 DTD 통합 방법은 XML DTD와 XML 문서 내용과의 관계 그리고 DTD의 선언 중에서 속성 리스트에 대한 선언은 고려하지 않는다. 다음 <정의 1>은 본 논문에서의 DTD 통합 대상인 "유사 구조를 가지는 XML 문서"에 대한 정의이다.

<정의 1> 다음 조건들을 모두 만족하는 XML 문서들을 "유사 구조, 즉 유사 DTD를 가지는 XML 문서"라 한다.
첫째, 같은 분야(domain)의 문서를 표현하는 XML 문서들이며, 둘째, 이러한 XML 문서가 일정한 형태를 가지는 원본 문서들을 표현하기 위한 문서들이고, 셋째, 문서의 구조를 정의하기 위하여 동일한 용어(term)나 같은 의미를 가지는 용어, 비슷한 의미를 가지는 용어들을 사용하는 XML 문서들(즉, 같은 엘리먼트 이름이나 유사한 의미의 엘리먼트 이름을 사용하여 문서의 형식이 정의된 XML 문서들)이고, 마지막으로 엘리먼트들 사이의 계층적 구조(형제 관계, 부모-자식 관계)가 유사한 XML 문서들을 말한다.

이러한 "유사 구조를 가지는 XML 문서"들의 DTD에 대한 통합은 다음 <정의 2>와 같이 정의한다.

여기에서 "공통 식별자(common identifier)"라 함은 두 개 이상의 유사 DTD들에 공통으로 존재하는 엘리먼트를 의미한다.

<정의 2> "DTD 통합(DTD Unification)" 각 유사 DTD들의 텍스트 엘리먼트들을 보존하면서, 공통 식별자, 트리 구조 그리고 유한 오토마타를 이용하여 각 유사 DTD들의 유사 구조를 의미 손실을 최소화하면서 통합한 단일 DTD를 생성한다.

결과적으로, 생성된 단일 DTD(통합 DTD;Unified DTD)는 유사 DTD를 가지는 XML 문서들에 의해 표현되는 공통 주제영역에 대한 범용 개념스키마(global conceptual schema)의 역할을 하게 된다.

트리 구조와 유한 오토마타를 사용하여 표현된 유사 DTD들을 통합하는 개략적인 알고리즘은 그림 5와 같다. DTD 통합 과정은 먼저 여러 DTD에서 사용된 엘리먼트들 중에서 같은 의미를 갖는 서로 다른 이름의 엘리먼트를 동일한 엘리먼트 이름으로 통일한 후, 트리와 유한 오토마타를 이용하여 DTD를 표현한다. 다음으로 트리와 유한 오토마타에 대한 병합 과정을 거쳐 유사 DTD들을 하나의 DTD로 통합한 후 병합된 DTD(트리와 유한 오토마타로 표현)를 DTD(텍스트로 표현)로 역변환한다. 마지막으로 통합 DTD가 XML에서 정의하고 있는 DTD 문법 규칙에 적합한가를 검증하기 위하여 DTD 파서가 역변환된 통합 DTD를 검사한다. DTD에 대한 검증이 끝나면 완전한 하나의 통합 DTD가 생성된다.

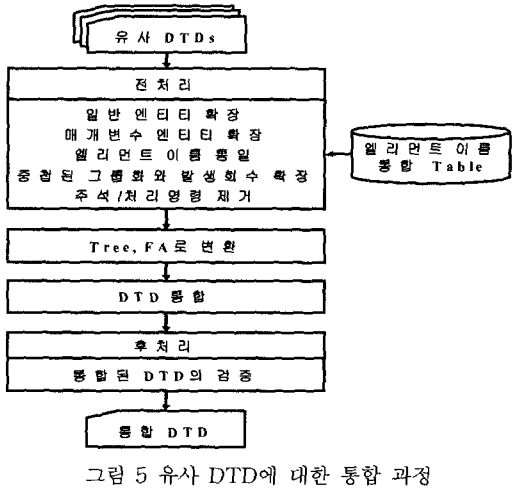

그림 5 유사 DTD에 대한 통합 과정

4.2 전처리

DTD의 논리적 구조를 통일하기 전에 일반 엔티티의 확장, 매개변수 엔티티의 확장, 엘리먼트 이름의 통일, 중첩된 그룹과 발생 횟수 확장 그리고 주석과 처리 명령 제거 등을 수행한다.

4.2.1 일반 엔티티 확장

각 DTD의 일반 엔티티를 확장하기 위하여 본 논문에서는 다음과 같은 규칙을 정의하여 일반 엔티티를 확장한다. 일반적으로 일반 엔티티는 문서의 구조를 표현하기 위해 사용되는 것이 아니라 약어나 외부 데이타를 이용하기 위해 사용되는 것이므로 DTD 통합 과정의 복잡도를 줄이기 위해 일반 엔티티를 확장한다.

<규칙 1> 일반 엔티티 확장 규칙 : 용어의 약어를 원래의 용어로 변환한다. 시스템 엔티티(System Entity)에 대해서는 외부 파일에 대한 경로를 새로 지정해 주고, 공중 엔티티(Public Entity)에 대해서는 중복되는 것을 한번씩 사용되도록 하여 그대로 확장한다.

예 5는 <규칙 1>을 적용한 예이다.

예 5) 일반 엔티티를 확장하는 예

DTD : <!ENTITY XML "eXtensible Markup Language" >
XML 문서 : <서론>본 논문의 &XML;이란 ...이다</서론>

규칙 적용 후 :
<서론>본 논문의 eXtensible Markup Language이란 ...이다</서론>

4.2.2 매개변수 엔티티 확장

매개변수 엔티티는 엘리먼트명, 엘리먼트 내용, 속성명, 속성 내용에 사용되는 것으로서, <규칙 2>를 정의하여 매개변수 엔티티를 확장한다. 일반 엔티티와 유사하게 매개변수 엔티티도 자주 사용되는 내용모델에 대한 약어를 이용하기 위해 사용하는 것이므로 DTD 통합 과정의 복잡도를 줄이기 위해 이를 원래의 내용모델로 확장한다.

<규칙 2> 매개변수 엔티티 확장 규칙 : 일반 엔티티 확장과 유사한 방법, 즉 매개변수 엔티티 선언에서 정의되어 있는 원래의 내용으로 확장한다.

예 6은 <규칙 2>를 적용한 예이다.

예 6) 매개변수 엔티티를 확장하는 예

DTD : <!ENTITY % m.sec "(제목?, 단락*, 절*)" >
<!ELEMENT 요약 (%m.sec;) >

규칙 적용 후 :
<!ELEMENT 요약 (제목?, 단락*, 절*) >

4.2.3 엘리먼트 이름 통일

각 DTD에서 사용되는 엘리먼트 이름들 중에서 서로 유사하거나 동일한 의미를 지닌 엘리먼트들의 이름을 엘리먼트 이름 통합 테이블(Element Name Resolution Table)을 이용하여 통일한다. 각 유사 DTD들이 같은 주제 영역에 대한 DTD들이므로 텍스트 엘리먼트에 대한 동음이의어(Homonym)는 발생하지 않는다고 가정할 수 있으며, 텍스트 엘리먼트가 아닌 엘리먼트에 대해서도 같은 이름을 가진 엘리먼트는 엘리먼트의 내용이 다르더라도 비슷한 목적을 위해 사용된 것으로 간주할 수 있다. 그러므로 엘리먼트 이름 통합 테이블은 해당 주제 영역에서 사용되는 이형동의어(Synonym)와 같은 이름과 같은 의미를 가지지만 부모 엘리먼트에 의해 구분될 필요가 있는 엘리먼트들의 이름을 처리하기 위한 것으로, 용어의 의미와 용어의 계층적 구조(부모-자식 관계)를 고려하여 전문가에 의해 작성된다. 표 2는 엘리먼트 이름 통합 테이블 중 일부를 나타내고 있다.

표 2 엘리먼트 이름 통합 테이블

DTD-1	DTD-2	통합 엘리먼트 이름
본문	논문내용	본문
출판전방	논문정보	논문정보
제목	한글제목	한글제목
대체제목	영문제목	영문제목

4.2.4 중첩된 그룹과 발생 횟수 문제

엘리먼트의 내용모델 중에서 중첩된 그룹 문제와 그룹의 발생 횟수 문제는 다음과 같은 규칙들을 정의하여 해결한다.

<규칙 3> ','그룹의 반복연산자 문제해결 규칙 : 엘리먼트의 내용모델이 ','로 그룹화된 상태에서 반복연산자가 지정된 경우에 대해서는 그룹을 하나로 묶을 수 있는 임시 태그를 만든다.

<규칙 3>은 DTD를 트리와 유한 오토마타로 표현하는 과정에서 발생되는 복잡도를 줄이기 위해 엘리먼트들의 순서 관계를 표현하는 ',' 연결자와 반복연산자를 분리하기 위한 규칙이다. <규칙 3>에 의해 엘리먼트들의 순서 관계를 유지하면서 반복연산자가 나타내는 반복을 표현할 수 있다. 예 7은 <규칙 3>을 적용한 예이다.

예 7) ',' 그룹에 지정된 반복연산자를 분리하는 예

<!ELEMENT 저자그룹 (저자, 소속)* >

규칙 적용 후 :
<!ELEMENT 저자그룹 tmp* >
<!ELEMENT tmp (저자, 소속) >

한편 '|' 연결자는 엘리먼트의 순서와는 무관하게 엘리먼트의 존재 여부만을 나타내므로 <규칙 4>에 의해 그룹을 구성하는 각각의 엘리먼트에 반복연산자를 적용하여 그룹을 제거한다. 이를 통해 트리와 유한 오토마타를 이용한 DTD 표현 과정에서의 복잡도를 줄일 수 있다.

<규칙 4> '|' 그룹의 반복연산자 문제 해결 규칙 : 엘리먼트 형 선언의 내용모델 중에서 '|'로 연결된 그룹 중에서 반복연산자가 지정된 경우, 지정된 반복연산자로 내용모델을 확장하여 그룹을 제거한다.

예 8은 <규칙 4>를 적용한 예이다.

예 8) '|' 그룹에 지정된 반복연산자를 확장하는 예

<!ELEMENT 저작권 (날짜|(소속기관, 소속부서?, 직위?)|이름)+ >

규칙 적용 후 :
<!ELEMENT 저작권 (날짜+|(소속기관, 소속부서?, 직위?)+|이름+) >

‘,’ 연결자가 사용된 그룹 전체에 대한 반복연산자가 지정이 되지 않은 경우에 내부의 그룹은 단순히 가독성을 높이거나 이해를 돕기 위해 사용된 것으로 볼 수 있으므로 내부의 그룹은 특별한 의미를 가지지 않는다고 볼 수 있다. <규칙 5>는 이러한 내용모델에 사용된 내부 그룹을 제거하기 위한 규칙이다.

<규칙 5> 중첩된 그룹 문제 해결 규칙 : 엘리먼트의 내용이 중첩구조로 되어 있고, 순서지정이 ‘,’로 되어 있으며 내포된 그룹 전체에 대한 반복연산자가 지정이 되지 않은 경우에는 안의 내포된 괄호를 제거한다.

예 9는 <규칙 5>를 적용한 예이다.

예 9) 중첩된 그룹을 제거하는 예

<!ELEMENT 논문지정보 ((논문지명,대체논문지명?),ISSN?,권,호,출판일,(학회명,대체학회명?))>

규칙 적용 후 :

<!ELEMENT 논문지정보 (논문지명,대체논문지명?,ISSN?,권,호,출판일,학회명,대체학회명?)>

4.3 트리와 유한 오토마타를 이용한 DTD 표현

기존의 연구들은 DTD를 모델링하기 위하여 트리, 구조도(Structure Diagram), 객체도(Object Diagram) 등을 주로 사용하였다. 그렇지만 트리 구조는 DTD의 반복연산자나 연결자를 표현하지 못하고, 구조도나 객체도는 관계성의 종류가 많으며 객체와 관계성을 표현하는 방법이 너무 다양하여 DTD 통합에 사용하기에는 적합하지 않다. 따라서 본 논문에서는 DTD를 표현하기 위하여 트리 구조와 유한 오토마타를 함께 사용한다. 유한 오토마타는 DTD의 반복연산자나 연결자를 표현하기에 적합하고, 표현 방법이 단순하기 때문에 DTD를 통합하는 과정에서의 복잡도를 감소시킬 수 있다.

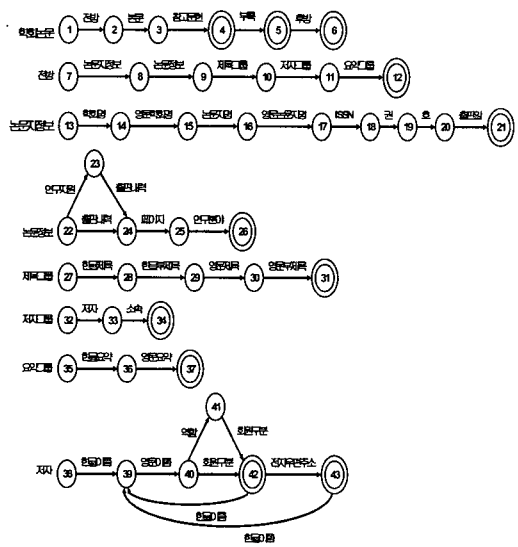

그림 6 논문지 A에 대한 유한 오토마타

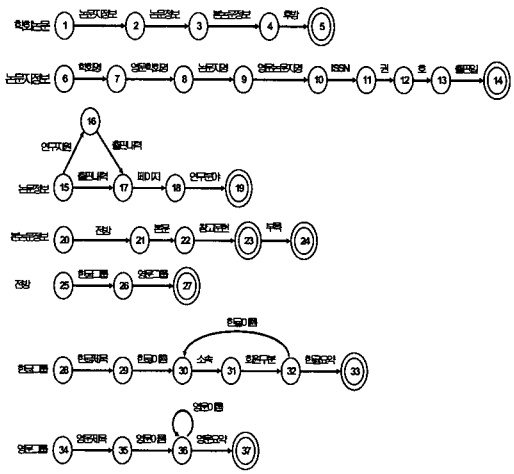

그림 7 논문지 B에 대한 유한 오토마타

그림 6과 그림 7은 각각 논문지 A와 논문지 B의 논문에 대한 DTD를 유한 오토마타를 사용하여 표현한 것(일부)이고, 그림 8와 그림 9는 트리 구조를 사용하여 표현한 것(일부)이다.

4.4 DTD 통합

DTD 통합 과정은 유한 오토마타와 트리 구조를 이용하여 표현된 DTD들의 엘리먼트의 논리적 구조를 통합하는 단계이다. 엘리먼트 형 선언에서 자식 엘리먼트들에 대한 순서를 지정하는 내용모델은 각각의 DTD마다 서로 다르다. 이러한 내용모델, 즉 엘리먼트들 사이의 계층 구조를 일관성있게 통일하기 위하여 본 논문에서는 트리 구조를 사용한다.

[알고리즘 1]은 각 통합 대상 DTD(유사 DTD)의 트리를 통합하여 통합 DTD(UDTD; Uniform DTD) 트리를 생성하는 단계와 UDTD 트리를 순회하면서 각 유사 DTD의 유한 오토마타를 통합하여 UDTD FA를 생성하는 단계로 이루어진 유사 DTD 통합 알고리즘이다.

첫 번째 단계는 네 개의 세부 단계로 나누어지는데

[알고리즘 1] 유사 DTD의 통합

```
make_UDTD(Tree[],FA[],UDTD_Tree,UDTD_FA)
입력 : Tree(유사 DTD들의 트리 표현),
       FA(유사 DTD들의 오토마타 표현)
출력 : UDTD_Tree(통합 DTD의 트리 표현),
       UDTD_FA(통합 DTD의 오토마타 표현)
begin
1. 유사 DTD의 트리를 통합.
   1-1. 형제 노드를 결정(decision_sibling).
   1-2. 부모 노드를 결정(decision_parent).
   1-3. cid가 아닌 모든 엘리먼트 이름을 UDTD_Tree에 통
        합(merge_id).
   1-4. UDTD_Tree를 축약(compact_UDTD_Tree).
2. cid의 부모 노드의 FA를 통합(merge_cid_FA).
end.
```

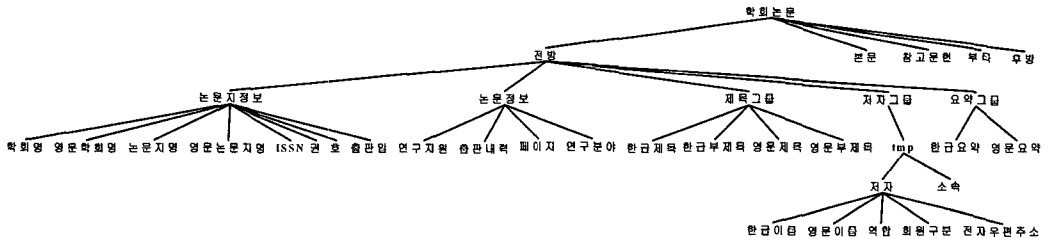

그림 8 논문지 A에 대한 트리

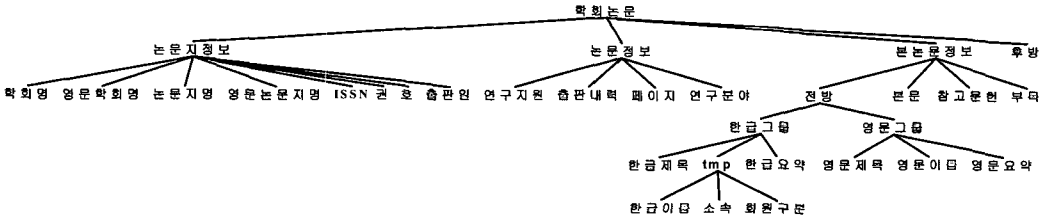

그림 9 논문지 B에 대한 트리

단계 1-1과 단계 1-2는 유사 DTD들에 같이 존재하는 공통 식별자(cid)들을 대상으로 각각 형제 노드와 부모 노드를 결정(decision_sibling, decision_parent)하여 UDTD 트리를 구성한다. 그리고 단계 1-3에서는 터미널 노드이면서 특정 DTD에만 존재하는 공통 식별자들을 UDTD에 통합(merge_id)하고, 단계 1-4에서는 생성된 UDTD 트리를 축약한다(compact_UDTD_Tree). 이 과정에서 UDTD 트리의 깊이를 줄이게 된다.

유사 DTD 통합의 두 번째 단계(단계 2)에서는 생성된 UDTD 트리를 너비 우선 탐색(BFS;Breadth-First Search)으로 순회하면서, 단계 1-2와 단계 1-4를 통해 결정된 cid의 부모 노드들을 대상으로 유사 DTD들의 유한 오토마타를 통합한다(merge_cid_FA).

[알고리즘 2]는 cid 노드의 형제 노드들을 결정하기 위한 알고리즘이다. 엘리먼트들 중에서 두 개 이상의 통합 대상 DTD들에 동시에 존재하는 공통 식별자들인 cid의 형제 노드들의 합집합을 각 유사 DTD에서 구한다.

[알고리즘 2] 형제 노드 결정

```
decision_sibling(Tree[],Sibs_cid)
입력 : Tree
출력 : Sibs_cid(각 cid의 형제 노드들의 집합)
begin
 for( 모든 cid ) {
  Sibs_cid[i] = 각 유사 DTD들을 대상으로 구한 cid_i
                의 형제들의 합집합
 } /* endfor */
 for( Sibs_cid의 각 집합 ) {
  if( Sibs_cid[i]와 Sibs_cid[j]의 교집합이 공집합
      이 아니면 ) {
  두 집합의 크기를 비교하여 크기가 큰 집합에 크기
  가 작은 집합을 소속시킨다.
  } /* endif */
 } /* endfor */
end.
```

다음은 [알고리즘 2]에 의해 결정된 cid들의 형제 노드들의 집합 중 일부를 보여주고 있다.

{학회명, 영문학회명, 논문지명, 영문 논문지명, ISSN, 권, 호, 출판일}
{한글제목, 영문제목, 한글요약, 영문요약}
{전방, 본문, 참고문헌, 부록, 후방, 논문지정보, 논문정보}

[알고리즘 3]은 [알고리즘 2]에서 결정된 cid들의 형제 노드 집합에 대한 부모 노드를 결정하기 위한 알고리즘이다. 먼저 유사 DTD들 중에서 해당 형제 노드의 부모 노드들 중에서 tmp 노드가 존재하면 tmp 노드를 부모 노드로 하고, 최종 부모 노드를 결정하기 위한 비교에서는 tmp 노드의 부모 노드를 사용하도록 한다. 그런 후 최종 부모 노드를 결정한다. 즉 각 유사 DTD에서의 cid의 부모 노드가 동일하면 해당 노드를 부모 노드로 하고, 부모 노드가 다르면 해당 cid의 형제 노드들을 보다 많이 포함하고 있는 유사 DTD의 부모 노드를 최종 부모 노드로 결정한다. 마지막으로 넌터미널(non-terminal) 노드이면서 부모 노드가 결정되지 않은 cid를 대상으로 해당 cid에서 루트 노드까지의 경로를 UDTD 트리에 복사함으로써 해당 cid의 부모 노드를 결정한다. 그림 10은 [알고리즘 3]에 의해 생성된 UDTD 트리이다.

[알고리즘 4]는 터미널 노드이면서 cid가 아닌 엘리먼트들을 UDTD 트리에 통합하기 위한 알고리즘이다. 즉 특정 유사 DTD 트리에만 존재하는 엘리먼트들 중에서 터미널 노드인 엘리먼트들을 대상으로 해당 엘리먼트로부터 루트 노드까지의 경로를 UDTD 트리에 복사한다.

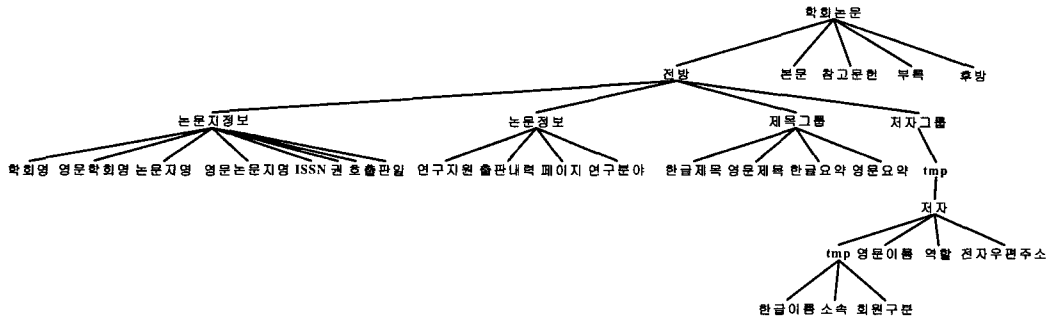

그림 10 [알고리즘 3]을 적용한 결과(축약되지 않은 UDTD Tree)

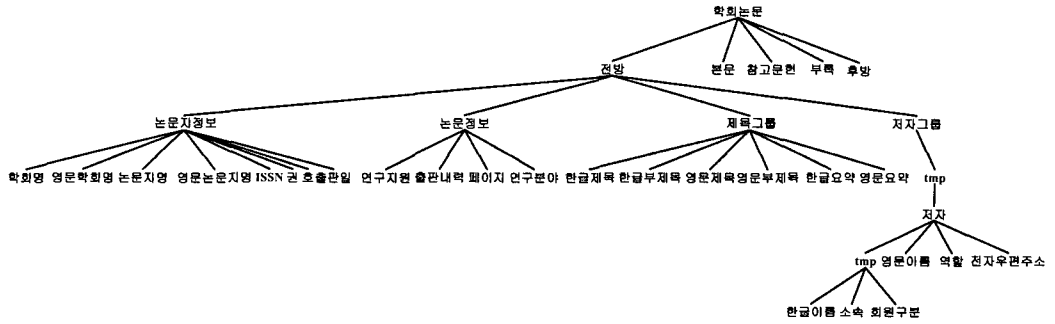

그림 11 [알고리즘 4]를 적용한 결과(축약되지 않은 UDTD Tree)

[알고리즘 3] 부모 노드 결정

```
decision_parent(Tree[],Sibs_cid,Uncompact_UDTD_Tree)
입력 : Tree, Sibs_cid
출력 : Uncompact_UDTD_Tree(축약되지 않은 UDTD_Tree)
begin
 for( 모든 cid ) {
  if( cid의 부모 노드가 tmp 노드 ) {
   tmp 노드를 cid와 cid의 형제의 부모 노드로 한다.
  } /* endif */
  if(DTD 트리에 존재하는 cid의 부모 노드가 동일) {
     해당 부모 노드를 cid와 cid의 형제의 부모 노드
     로 한다.
  }
  else {
      크기가 가장 큰 cid의 형제를 포함하고 있는
      DTD의 부모 노드를 cid와 cid의 형제의 부모 노
      드로 한다.
  } /* endif */
 } /* endfor */
 부모 노드가 결정되지 않은 각 cid의 부모 노드를 결정.
end.
```

그림 11은 [알고리즘 4]에 의해 생성된 UDTD 트리이다. 그림 10과 비교해 보면 "제목그룹"의 자식 노드에 "한글부제목"과 "영문부제목"이 추가되어 있음을 알 수 있다. "한글부제목"과 "영문부제목"은 논문지 A의 논문 DTD에 포함되는 터미널 노드이지만 논문지 B의 논문 DTD에는 포함되지 않는 노드들이다.

[알고리즘 4] 터미널 노드이면서 cid가 아닌 모든 엘리먼트 통합

```
merge_id(Tree[],Uncompact_UDTD_Tree)
입력 : Tree, Uncompact_UDTD_Tree
출력 : Uncompact_UDTD_Tree
begin
 for(터미널 노드이면서 cid가 아닌 모든 엘리먼트들){
   해당 엘리먼트로부터 루트(root) 노드까지의 경로
   를 UDTD_Tree에 복사한다.
 } /* endfor */
end.
```

[알고리즘 5]는 현재까지 만들어진 UDTD 트리를 축약하기 위한 알고리즘이다. 즉 "tmp" 노드와 단 하나의 자식 노드만을 가지는 넌터미널 노드들을 UDTD 트리에서 삭제하고, 해당 넌터미널 노드의 자식 노드들을 해당 넌터미널 노드의 부모 노드의 자식 노드로 소속시킨다. 이를 통해 UDTD 트리의 깊이를 감소시키고 UDTD FA를 통합하는 과정에서의 복잡도를 감소시킨다. 그림 12는 [알고리즘 5]에 의해 생성된 최종 UDTD 트리이다. 그림 11과 비교해 보면 "저자" 노드의 조상 노드인 "tmp"와 "저자그룹"이 삭제되어 "저자" 노드의 부모 노드가 "전방" 노드가 된다.

[알고리즘 6]은 cid 노드의 부모 노드의 유한 오토마타를 통합하기 위한 알고리즘이다. 즉 [알고리즘 5]에 의해 생성된 최종 UDTD 트리를 BFS로 순회하면서 만

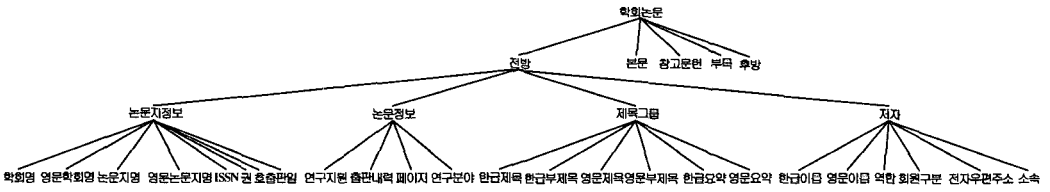

그림 12 [알고리즘 5]를 적용하여 생성된 최종 UDTD Tree

[알고리즘 5] UDTD_Tree 축약

```
compact_UDTD_Tree(Uncompact_UDTD_Tree, UDTD_Tree)
입력 : Uncompact_UDTD_Tree
출력 : UDTD_Tree
cur_NT : 현재 처리하고 있는 넌터미널 노드
parent_cur_NT : cur_NT의 부모 노드
children_cur_NT : cur_NT의 자식 노드들
begin
 for( UDTD_Tree의 모든 넌터미널 노드 ) {
  if(cur_NT가 tmp 노드이거나 children_cur_NT의 크
    기가 1이면서 넌터미널 노드) {
  children_cur_NT를 parent_cur_NT의 자식 노드로 한다.
  UDTD_Tree에서 cur_NT를 삭제한다.
  } /* endif */
 } /* endfor */
end.
```

나는 각 넌터미널 노드를 대상으로 하여 유한 오토마타를 통합한다. 유한 오토마타에 대한 통합 방법을 살펴보면, 모든 유사 DTD의 유한 오토마타에서 연속된 두 개의 동일한 입력 기호(엘리먼트 이름)에 대한 상태 전이가 발생하면 이러한 상태 전이를 UDTD FA에 추가한다. 만약 이러한 조건을 만족하지 못하면 표 3과 같이 3가지 경우로 나누어 유한 오토마타를 통합한다.

조건3의 경우는 텍스트 형태의 UDTD를 생성할 때 해당 입력 기호들을 OR 연결자("|")로 연결하고 해당 그룹에 대해 "+" 반복 연산자를 지정하게 된다. 그림 13은 논문지 A의 논문 DTD와 논문지 B의 논문 DTD를 통합하여 생성된 UDTD FA이다. UDTD 트리 구조는 [알고리즘 5]가 적용된 결과(그림 12)와 동일하다.

[알고리즘 6] cid의 부모 노드의 유한 오토마타 통합

```
merge_cid_FA(FA[],UDTD_Tree,UDTD_FA)
입력 : FA, UDTD_Tree
출력 : UDTD_Tree, UDTD_FA
begin
 while(통합할 노드가 존재하지 않을 때까지) {
  if(통합 조건이 성립) {
   해당 입력 기호에 대한 상태 전이를 UDTD_FA에 추가
  }
  else {
  if( 조건1을 만족 ) {
   해당 FA의 해당 입력 기호에 대한 상태를 전이
    /* 해당 입력 기호 무시 */
  }
  else if( 조건2를 만족 ) {
   해당 입력 기호를 선택 사항으로 처리하여
    UDTD_FA에 추가
  }
  else { /*조건3:조건1과 조건2에 해당하지 않을 때*/
   입력 기호들에 대한 모든 순열들을 UDTD_FA에 추가
  } /* endif */
 } /* endif */
} /* endwhile */
end.
```

4.5 후처리

후처리 단계는 통합 알고리즘에 따라 통합된 DTD가 XML 구문에 맞게 기술되었는지 검증하는 단계로서 본 논문에서는 Microsoft사의 MSXML 파서를 이용하였다. 그림 14는 검증 단계를 통해 최종적으로 생성된 통합 DTD를 나타내고 있다. 그림 14의 통합 DTD를 살펴 보면 원래의 DTD들(논문지 A의 논문 DTD, 논문지 B의 논문

표 3 FA를 통합하기 위한 처리 조건과 처리 방법

조건		처리
통합 조건	모든 FA에서 연속된 두 개의 동일한 입력 기호에 대한 상태 전이가 발생하는 경우	해당 입력 기호에 대한 상태 전이를 UDTD_FA에 추가하고 계속 유한 오토마타 통합 과정 수행
조건1	① cid이면서 UDTD 트리 상의 다른 forest에 속하는 입력 기호("한글요약", "영문요약")를 가지는 유한 오토마타가 존재하거나, ② cid가 아니면서 넌터미널 노드인 입력 기호("제목그룹")를 가지는 유한 오토마타가 존재하는 경우	해당 유한 오토마타의 해당 입력 기호에 대한 상태를 전이하여 해당 입력 기호를 무시하고 계속 유한 오토마타 통합 과정 수행
조건2	cid가 아니면서 터미널 노드인 입력 기호("한글부제목", "영문부제목")를 가지는 유한 오토마타가 존재하는 경우	해당 입력 기호를 선택 사항으로 처리하여 UDTD FA에 추가
조건3	통합 조건, 조건1, 조건2에 해당하지 않는 경우	해당 입력 기호들에 대한 모든 순열들을 UDTD FA에 추가한 후, 유한 오토마타 통합 과정 계속

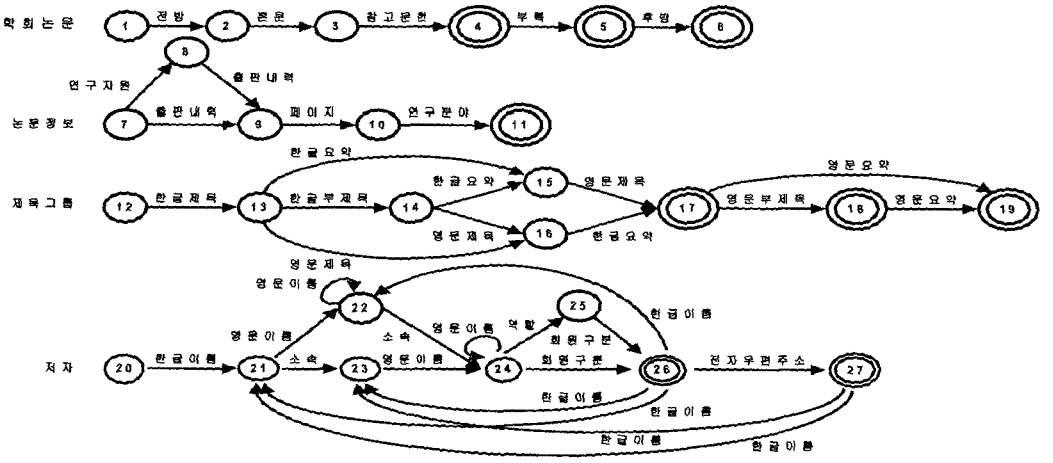

그림 13 통합 DTD의 유한 오토마타

DTD)을 가지는 XML 문서들의 모든 내용(텍스트 엘리먼트로 표현되는 문서의 내용)을 포함하면서 각 DTD들의 구조가 큰 의미 손실이 없이 유지됨을 알 수 있다.

```
<!DOCTYPE 학회논문 [
<!ELEMENT 학회논문 (전방, 본문, 참고문헌, 부록?, 후방?)>
<!ELEMENT 전방 (논문지정보, 논문정보, 제목그룹, 저자)>
<!ELEMENT 논문지정보 (학회명, 영문학회명, 논문지명, ISSN, 권, 호, 출판일)>
<!ELEMENT 학회명 (#PCDATA)>
...
<!ELEMENT 논문정보 (연구지원?, 출판내력, 페이지, 연구분야)>
<!ELEMENT 출판내력 (접수일, 승인일)>
...
<!ELEMENT 제목그룹 (한글제목, 한글부제목?, (한글요약|영문제목)+, (영문부제목?, 영문요약)?>
<!ELEMENT 한글제목 (#PCDATA)>
...
<!ELEMENT 저자 (한글이름, (영문이름|소속)+, 역할?, 회원구분, 전자우편주소?)+>
<!ELEMENT 한글이름 (#PCDATA)>
 ... ] >
```

그림 14 텍스트로 변환된 통합 DTD

5. 결론 및 향후 연구과제

최근 국내외적으로 많은 기관에서 XML을 이용하여 문서를 작성, 저장, 검색하기 위한 사업을 수행하고 있으며 구축된 XML 데이타베이스를 사용자에게 서비스하기 위하여 다양한 시스템들을 개발하고 있다. 그런데 이러한 XML 문서들 중에는 논리적 구조가 매우 유사함에도 불구하고 서로 다른 종류의 문서로 간주되어 각각 별도의 문서형 정의(DTD)를 가지는 경우가 많다. 이러한 XML 문서들은 서로 다른 데이타베이스 구조를 가지기 때문에 데이타베이스를 구축하기 위한 비용이 증가하게 되고, 하나의 질의를 처리하기 위해 여러 데이타베이스에 접근해야 하는 문제점이 발생한다.

따라서 본 논문에서는 트리 구조와 유한 오토마타를 이용하여 유사한 구조를 가지는 XML 문서들의 DTD를 통합하는 알고리즘을 제안하고 이를 구현하였다. 본 논문에서 제안한 통합 DTD 생성 알고리즘을 사용하게 되면 개념적으로 같은 종류에 속하는 여러 종류의 XML 문서들에 대한 검색 과정에서 데이타베이스에 접근하는 횟수를 줄일 수 있다. 그리고 동일한 질의를 수행하기 위하여 여러 개의 데이타베이스를 검색해야 하는 문제점을 해결할 수 있으며, 결과적으로 XML 문서 데이타베이스에 대한 검색 효율을 향상시킬 수 있다. 또한 XML 문서 데이타베이스에 대한 보다 효과적인 관리·운영 환경을 제공할 수 있으며, 데이타베이스를 구축하기 위한 비용을 절감할 수 있다.

향후 본 논문에서 제안한 통합 DTD 생성 알고리즘을 보다 다양한 종류의 XML 문서에 적용하여 본 논문에서 제안한 알고리즘을 보다 상세화하기 위한 연구와 각 유사 DTD를 가지는 XML 문서를 통합 DTD에 의한 XML 문서로 변환하는 방법에 대한 연구가 필요하다. 그리고 사용자가 효과적이고 일관성 있게 XML 문서를 검색하기 위하여 메타데이타를 이용하는 검색 기법에 대한 연구가 필요하다. 또한 XML 문서 데이타베이스의 스키마와 메타데이타의 스키마를 자동으로 생성하고, 이를 이용하여 XML 문서 데이타베이스와 메타데이타베이스를 자동으로 구축하는 기법에 대한 연구가 수행되어야 한다.

참 고 문 헌

[1] Seung-Jin Lim Yiu-Kai Ng, "An Automated

Integration Approach for Semi-Structured and Structured Data," 3rd Int'l Symposium on Cooperative Database Systems and Applications(CODAS 2001), pp. 15-24, Beijing, China, Apr. 2001.
[2] Chantal Reynaud, Jean-Pierre Sirot, Dan Vodislav, "Semantic Integration of XML Heterogeneous Data Sources," Int'l Database Engineering & Application Symposium (IDEAS2001), pp. 199-208, Grenoble, France, July 2001.
[3] Patricia Rodriguez-Gianolli, John Mylopoulos, "A Semantic Approach to XML-based Data Integration," 20th Int'l Conf. on Conceptual Modeling (ER'2001), pp. 117-132, Yokohama, Japan, Nov. 2001.
[4] Marie-Christine Rousset, Chantal Reynaud, "Knowledge representation for information integration," Information Systems, Vol. 29, pp. 3-22, 2004.
[5] XML 1.0(Third Edition), W3C Recommendation, http://www.w3.org/TR/2004/REC-xml-20040204, Feb. 2004.
[6] Boris Chidlovskii, "Using Regular Automata as XML schemas," 4'th IEEE Advances in Digital Libraries Conference(ADL 2000), pp. 1-10, Washington, USA, May 2000.
[7] Ronaldo dos Santos Mello, Silvana Castano, Carlos Alberto Heuser, "A method for unification of XML schemata," Information and Software Technology, Vol. 44, No. 4, pp. 241-249, 2002.
[8] Chun-Sik Yoo, Seon-Mi Woo, Yong-Sung Kim, "Automatic Generation Algorithm of Uniform DTD for Structured Documents," Proc. of IEEE Region 10 Conf. TENCON'99, Vol. II, pp. 1095-1098, 1999.
[9] Euna Jeong, Chun-Nan Hsu, "Veiw Inference for Heterogeneous XML Information Integration," Journal of Intelligent Information Systems, Vol. 20, No. 1, pp 81-99, 2003.
[10] Helena Ahonen, "Generating Grammars for Structured Documents Using Grammatical Inference Methods," University of Helsinki, Ph. D Thesis, 1996.
[11] OmniMark, "OmniMark : Content Model Algebra," *http://www.exoterica.com/white/ cma/cma.htm*
[12] Keith E. Shafer, Roger Thompson, "Translating Mathematical Markup for Electronic Documents," http://www.oclc.org/fred/docs/www4.htm
[13] Anhai Doan, Pedro Domingos, "Learning to Match the Schemas of Data Sources:A Multistrategy Approach," Machine Learning, Vol. 50, pp. 279-301, 2003.
[14] Elisa Bertino, Giovanna Geurrini, Marco Mesiti, "A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications," Information Systems, Vol. 29, pp. 23-46, 2004.
[15] Murali Mani, Dongwon Lee, "XML to Relational Conversion using Theory of Regular Tree Grammars," 1st VLDB Workshop on Efficiency and Effectiveness of XML Tools, and Techniques (EEXTT 2002), pp. 81-103, Hong Kong, China, Aug. 2002.
[16] Wolfgang May, Georg Lausen, "A uniform framework for integration of information from the web," Information Systems, Vol. 29, pp. 59-91, 2004.

유 춘 식
전북대학교 전산통계학과. 1991년 전북대학교 전산통계학과(이학사), 1994년 전북대학교 전산통계학과(이학석사), 1996년~현재 전북대학교 전산통계학과 박사과정. 관심분야는 XML, 스키마 통합, 유비쿼터스, 적응형 사용자 인터페이스, W3 웹 서비스 등

우 선 미
전북대학교 BK21 전자정보사업단 기금교수. 1995년 전북대학교 전산통계학과(이학사). 1997년 전북대학교 전산통계학과(이학석사). 2001년 전북대학교 전산통계학과(이학박사). 2004년~현재 전북대학교 BK21 전자정보사업단 기금교수. 관심분야는 XML, 사용자 중심의 정보검색, 순위결정, 정보필터링 등

김 용 성
전북대학교 전자정보공학부 교수. 1978년 고려대학교 수학과(이학사). 1984년 광운대학교 전산학과(이학석사). 1992년 광운대학교 전산학과(이학박사). 1985년~현재 전북대학교 전자정보공학부 교수. 1996년~1998년 1월 한국학술진흥재단 전문위원. 관심분야는 XML, 사용자 중심의 정보검색, 다중 사용자 인터페이스, W3C 웹 서비스 등