DOI: 10.3745/KIPSTD.2008.15-D.4.441

XForms 페이지의 접근제어를 위한 공유 조건식의 효율적 계산 방법

이 은 정†

요 약

최근 폼 기반의 웹 시스템에 대한 접근 제어 방법이 서비스 기반 아키텍처를 가지는 클라이언트 시스템 구현에 유용한 방법으로 주목받고 있다. 특히 XForms 언어는 서버와 상호작용하는 XML 기반의 사용자 인터페이스를 기술하는 언어로 많이 채용되고 있다. 이 논문에서는 XForms 페이지에 대한 XPath 기반의 접근 제어 규칙을 효율적으로 계산하는 알고리즘을 제안한다. XForms 페이지는 바인딩된 XML 노드에 대한 연속된 질의로 모델링할 수 있으며 클라이언트 시스템은 사용자 인터페이스를 생성하면서 XPath 규칙을 계산한다. XPath 규칙은 인스턴스 데이터를 이용하는 조건부를 가지는데 이 조건부의 계산이 규칙들 사이에 또 연속된 질의 사이에 중복되는 경우가 많다. 중복되는 조건부의 효율적인 계산을 위해서 조건부 그래프 모델을 제안하여 동일한 컨텍스트 노드에 대해 이전에 계산된 결과를 재사용하는 방법을 제안하였다. 이 방법은 각 조건부 식이 해당되는 XML 노드에 대해 한번만 계산되는 것을 보장한다.

키워드 : 접근제어, 공통조건부, XPath, XForms

Efficient Evaluation of Shared Predicates for XForms Page Access Control

Eun-Jung Lee†

ABSTRACT

Recently, access control on form-based web information systems has become one of the useful methods for implementing client systems in a service-oriented architecture. In particular, XForms language is being adopted in many systems as a description language for XML-based user interfaces and server interactions. In this paper, we propose an efficient algorithm for the evaluation of XPath-based access rules for XForms pages. In this model, an XForms page is a sequence of queries and the client system performs user interface realization along with XPath rule evaluations. XPath rules have instance-dependent predicates, which for the most part are shared between rules. For the efficient evaluation of shared predicate expressions in access control rules, we proposed a predicate graph model that reuses the previously evaluated results for the same context node. This approach guarantees that each predicate expression is evaluated for the relevant xml node only once.

Key Words : Access Control, Shared Predicates, XPath, XForms

1. 서 론

XML 접근 제어는 사용자에게 제한된 데이터 접근 만을 허용하는 방법으로 주로 서버에서 처리되었다[1-2]. 그러나 최근에는 사용자 인터페이스 요소를 접근 제어의 대상으로 하여 선별적인 액션을 허용하는 것이 웹기반 시스템을 중심으로 도입되었다[3-4]. 특히 웹사이트에서 사용자 권한에 맞는 페이지의 생성을 위해 접근 제어 기법을 많이 사용한다. 이때 XML 트리에 대한 질의 언어인 XPath[5]를 이용한 접근 제어 규칙을 통해 XML 데이터에 대한 접근 제어를 기술하고 이를 통해 웹페이지를 생성하는 서버 기반 기법이 소개되었다[3].

한편 서비스 기반 아키텍춰가 보급되면서 차세대 웹폼 언어 표준인 XForms[5] 기반의 클라이언트 설계 방법이 웹서비스 서버와의 통신을 기술하는데 유용한 방법으로 등장하였다. XForms 페이지는 XML 모델 인스턴스에 대한 질의의 집합으로 볼 수 있으며 브라우저나 클라이언트 어플리케이션이 인터페이스 요소의 순서에 따라 질의를 처리하는 것으로 볼 수 있다.

이 논문은 XForms 페이지의 UI 생성 단계에서 적용할 수 있는 클라이언트 기반의 접근 제어 방법을 찾고자 한다.

※ 본 연구는 경기도의 경기도지역협력연구센터사업[컨텐츠융합소프트웨어연구센터]과 경기대학교특성화사업[생물자원보존,복원,개발을위한BEIT융합기술특성화사업]의 일환으로 수행하였음.
† 종신회원 : 경기대학교 컴퓨터과학전공 부교수
논문접수 : 2008년 4월 1일
수 정 일 : 1차 2008년 5월 19일, 2차 2008년 5월 22일
심사완료 : 2008년 5월 23일

여기서 효율적인 계산을 위해서는 연속된 UI 요소를 생성할 때 접근 제어 계산의 중복되는 부분을 재사용하는 방법을 제안한다. 공통 조건부는 접근제어에서 XPath 계산을 최적화하는 방법으로 많이 연구되었으나 XForms 페이지 접근 제어에서는 XForms 요소(액션 템플릿) 형태의 질의가 바인딩된 XML 컨텍스트 요소에 대해 주어지므로 XML 트리에 대한 질의에서의 최적화 방법과는 달라지게 된다.

이 논문의 기여는 두가지 측면이다. 우선 XForms 페이지와 접근 규칙을 위한 액션 템플릿 모델을 제안하였다. 접근 규칙은 XML 인스턴스 값과 클라이언트 컨텍스트 값을 이용하는 조건부를 기술한다. 두 번째로, 연속된 액션 템플릿의 공통 조건부 계산 결과를 재사용하는 방법으로 접근 제어를 위한 효과적인 계산 방법을 제안하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴보고 3장은 문제 정의와 제안된 모델을 소개한다. 4장에서는 조건부 그래프를 정의하고 이를 이용한 계산 알고리즘을 제시한다. 5장은 결론을 맺는다.

2. 관련 연구

XML 데이터의 접근 제어 모델을 제시한 논문들이 많이 있었다[1-2, 7-10]. 이 모델들은 사용자가 허용된 XML 데이터 요소만 접근하도록 허용한다. XML 데이터의 접근 제어 성능을 향상시키기 위한 연구도 있었다. 이전 연구는 주로 접근 규칙을 효율적으로 검색하는 방법과[7-9] 정적 분석 방법을 통한 실행 시의 권한 검사를 줄이는 시도로 나누어진다[10-11]. 웹 폼이나 웹기반 시스템에 대한 접근 제어 연구도 있었다[3, 4, 12].

공통 조건부를 이용한 최적화 방법을 도입한 방법들이 있었다. Gupta 등의 XPush 시스템에서는 공통 조건부를 이용한 상향식 오토매타를 제안하였다[13]. Hou 등은 공통 조건부를 이용하기 위하여 XPath 부분식을 인덱싱하는 방법을 제안하였다[14]. 한편 접근제어 분야에서는 Jeon 등이 제안한 XACT 시스템에서 Murata 등의 정적인 접근규칙 분석 방법[10]을 확장하여 조건부를 동적으로 처리하는 방법을 제안하였다[11]. Lee 등은 접근 제어를 결합한 질의 처리 과정에서 공통 조건부 개념을 도입하였다[9]. 이들은 동적인 조건부 개념을 제안하여 조건식의 중간 결과를 재사용하는 방법을 소개하였다. XACT는 본 논문에서 제안하는 조건부 그래프와 유사한 트리 모델인데, 이 연구는 Murata의 방법을 확장하여 개별적인 질의 XPath에 대해 조건절의 포함관계를 정적으로 분석하는데 목적이 있었다. 그러나 본 논문에서는 연속된 질의(액션 템플릿)의 접근 권한 계산에서 기존에 계산된 조건절의 동적인 결과를 재사용하는데 목적이 있다는 점에서 다르다.

그런가 하면 웹 서비스 접속을 위한 클라이언트를 XForms 언어를 이용하여 개발하는 방법이 제안되었다[15, 16]. XML 인스턴스 데이터의 접근 제어를 통해 XForms 페이지에 결합된 접근 제어를 기술하는 방법도 제시되었다[4, 12, 17]. Carminati 등은 XML 데이터에 대한 웹 기반의 접근 제어 방법을 연구하였다[17]. Calders 등은 웹 폼 페이지와 접근 제어 규칙에서 얻어지는 작업 흐름을 분석하는 방법을 제안하였다[12]. 이들은 웹 폼 컨트롤의 접근성이라는 개념을 소개하였다. Thompson 등은 이를 확장하여 웹 폼 설계에서 프로세스를 개선하는 방법을 제시하였다[4]. 이들의 방법은 폼 컨트롤에 대한 접근제어로 표현되는 웹페이지 내부 또는 웹페이지 간의 흐름 관계를 분석하여 이를 통해 접근성 또는 종료 가능성 등의 성질을 모델링하였다. 본 논문에서 제안된 조건부 그래프를 이용한 접근 제어 계산 방법은 이러한 작업 흐름 분석에서 데이터 의존적인 컨트롤의 접근제어를 위한 효과적인 계산 방법으로 적용될 수 있다.

XForms 페이지에 대한 접근 제어 계산은 이전의 접근 제어와 다른 최적화 방법이 필요하나 기존 문헌에서는 이에 대한 연구는 없었다. 본 논문에서는 접근 제어를 계산하기 위하여 XForms 페이지의 폼 컨트롤이 XML 인스턴스 노드와 바인딩되었을 때의 접근 권한이 계산되어야 한다는 점을 주목하여 효율적인 계산 알고리즘을 제안하고자 한다.

3. XML 트리와 접근 권한 모델

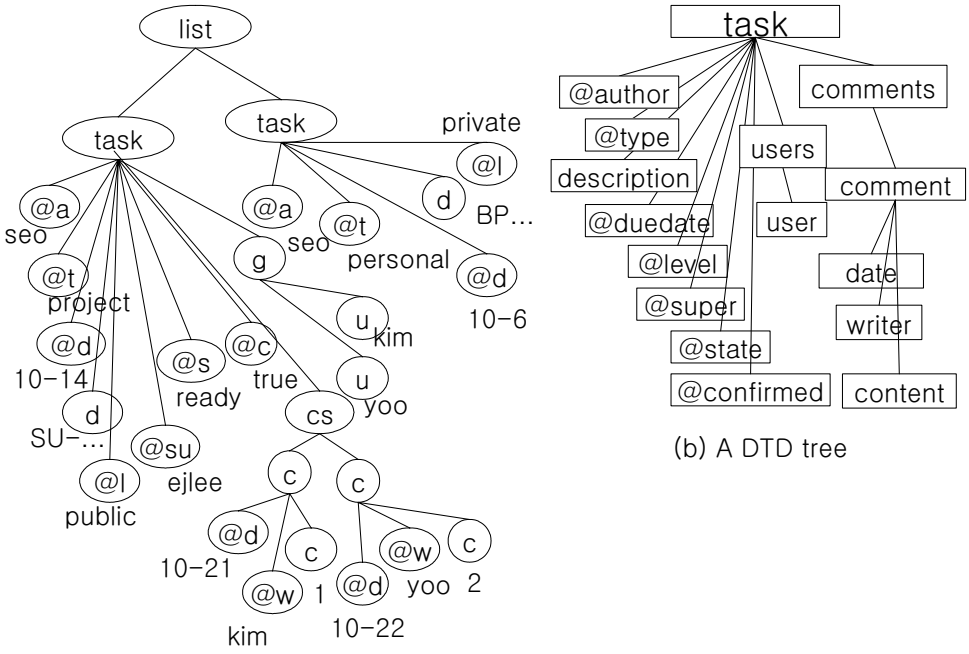
3.1. XML 트리와 스키마 정의

XML 데이터 인스턴스는 트리이며 $T_{XML} = (N_{XML}, E_{XML})$로 표시된다. 노드 $x \in N_{XML}$는 $text()$를 가지는 터미널 노드이거나 내부 노드이다. 속성은 터미널 노드로 취급된다. 한편 스키마는 DTD 트리 $D = (N_{DTD}, E_{DTD})$로 정의되는데 DTD의 재귀적 요소 정의는 DTD 트리의 경로를 XML 인스턴스 트리와 동일한 깊이만큼 확장하므로서 제거한다. 이것은 Suciu 등이 제안한 Summary의 개념과 동일하다[13]. 이 방법은 DTD 트리가 재귀를 갖지 않으면서 깊이가 XML 인스턴스의 깊이를 넘지 않는 것을 보장한다. 이러한 XML 트리와 DTD 트리의 예가 (그림 1)에 나와 있다.

집합 L이 D에서 정의된 노드 레이블의 집합이라고 하자. XML 트리의 노드 x와 DTD 트리의 노드 v에 대해 레이블로의 매핑 l이 정의되어서 $l(x) \in L$이고 $l(v) \in L$이다. 노드 레이블의 연속 $p \in L^*$를 경로라 하고 XML 트리와 DTD 트리의 노드 x와 v에 대해 루트로부터의 경로를 각각 $path(x)$와 $path(v)$라 한다. 주어진 경로 $p \in L^*$에 대해 트리에서의 후손 노드 집합을 $descend$ 함수로 정의하며 $descend(v, p) \subset N_{DTD}$, $descend(x, p) \subset N_{XML}$로 표시한다. T가 D에 대해 유효한 트리이면 N_{XML}에 대해 매핑 $M_{X->D}(x) = \{v \in D \mid path(v) = path(x), x \in N_{XML}\}$ 가 정의된다.

3.2. 사례 시나리오

접근 규칙이 주어졌을 때 브라우저 시스템은 허용된 데이터 아이템만 보여주거나, 입력 컨트롤을 쓰기 권한이 없고 읽기 권한만 있는 경우 그냥 텍스트로 보여주는 등 권한에 따라 화면을 구성할 수 있다. 한편 추가나 삭제 등의 버튼



(a) An example XML instance tree

(b) A DTD tree

(그림 1) 타스크 리스트 XML 트리와 DTD 트리 예

user : kim

	author	type	description	date		
1	seo	project	SU - Senario page design	10/14	Comment	Delete
2	yoo	project	SI - server REST implementation	10/14	Comment	Delete
3	kim	lab	BO - paper presentation - Bouganon	10/11	Comment	Delete

user : seo

	author	type	description	date		
1	seo	project	SU - Senario page design	10/14	Comment	Delete
2	seo	personal	BP - birthday party of hyung	10/6	Comment	Delete
3	yoo	project	SI - server REST implementation	10/14	Comment	Delete
4	kim	lab	BO - paper presentation - Bouganon	10/11	Comment	Delete

(그림 2) 타스크 리스트에 대한 접근제어 인터페이스 화면

을 권한이 허용되는 경우에만 활성화를 시킬 수 있다. 예를 들어 (그림 2)에서 SU 타스크는 사용자 seo와 kim에게 모두 보여지지만 seo는 모든 필드를 수정할 수 있고 kim은 읽기만 가능하다. 또한 seo는 BOU 타스크를 읽을 수만 있으면서 의견 추가는 가능하다.

XForms 요소가 XML 인스턴스 요소에 바인딩 될 때 접근 권한에 의해 그 액션이 허용될지 여부가 결정될 수 있다. 그 요소들은 화면에 등장하는 순서로 권한 계산이 이루어진다. 하나의 XForms 요소가 사용자나 컨텍스트 변수 값에 따라 또는 XML 인스턴스 요소의 값에 따라 허용되기도 하고 금지되기도 한다.

3.3. XPath 경로식과 접근 권한 규칙

이 논문에서 사용되는 XPath 수식은 다음과 같이 정의된다. 이것은 기존의 논문의 정의를 참조하였다[10, 12, 13].

[정의 1] XPath p는 다음과 같이 정의된다.

$p = p \mid p$ '*' $\mid p$ '//',

〈표 1〉 (그림 1)의 XML 데이터에 대한 접근 제어 정책 예

```
r1 = (Create, /l/t)
r2 = (Delete, /l/t[@a=$u])
r3 = (Read, /l/t[@l=1 | @a=$u or (@l!=3 and (g/u=$u | @s=$u))]/*)
r4 = (Update, /l/t[@a=$u]/{@t, @d, d, @l}
r5 = (Create, /l/t[@a=$u | (@l!=3 & (g/u=$u | @s=$u))]/c)
r9 = (Read, /l/t[@a=$u | (@l!=3 & (g/u=$u | @s=$u))]/c/*)
…
```

$p := p$ '/' $p \mid l[expr] \mid l,\ \ l \in L,$

이고 조건부 *expr*은 다음과 같이 정의된다.

$expr := expr\ op\ expr \mid term,\ term := p \mid value \mid variable,$

여기서 *op*은 논리 또는 비교 연산자이고 *value*와 *variable*은 각각 리터럴 값과 $로 시작되는 변수 이름을 나타낸다.

변수는 조건부에서 시스템 정의 값을 참조하기 위해 사용된다. 예를 들어 사용자 아이디, 날짜, 시간 등을 변수로 사용할 수 있다. 변수는 보통 XPath 계산 전에 값이 결정되므로 계산 과정에서는 리터럴 값과 동일하게 취급될 수 있다. 이하에서는 XPath의 마지막에 나타나는 * 또는 //를 tail로 표시한다.

경로 $p = p_0\ e0\ p_1 \cdots p_k\ e_k\ tail$에 대해 결과 노드의 집합을 $target(p) = \{v \in N_{DTD} \mid path(v)\ matches\ p_0\ p_1 \cdots p_k\ tail\}$로 표시하고 조건부 수식은 $expr(p) = e_0\ and\ e_1 \cdots and\ e_k$ 로 표시한다. XPath p에 의해 표현되는 노드 집합을 *answerset(p)* 라 한다. 주어진 XML 노드 *x*에 대해 부분조건식이 모두 만족되어야 *x*가 p를 만족하므로 다음과 같이 표현할 수 있다.

$answerset(p) = \{x \mid M_{X \to D}(x) \in target(p)$이고 $expr(p)$가 x에 대해 참이다.$\}$

조건식 e_i에 대응하는 DTD 노드의 집합은 $base(e_i) = \{w \in N_{DTD} \mid p_0\ p_1 \cdots pi = path(w)\}$로 표시된다.

접근 제어를 기술하기 위하여 규칙은 액션 유형과 대상 노드 집합으로 표시할 수 있다. 규칙은 일반적으로 XML 인스턴스의 값과 컨텍스트 변수의 값을 이용하는 조건부를 통해 허용 조건을 기술한다. 접근 제어를 위한 액션 유형은 T = {Read, Update, Create, Delete}로 정의된다.

[정의 2] 접근 제어 규칙은 $r = (t, p)$로 표시되는데, 여기서 $t \in T$, p는 XPath이다. 접근 제어 규칙의 집합은 접근 제어 정책이라 불리고 P로 표시된다.

접근 제어 정책은 허용되는 모든 연산을 나열하는 것으로 가정한다. 다음은 (그림 1)의 XML 트리에 대한 접근 정책의 예이다.

4. 효율적인 접근 권한 계산 방법

4.1. XForms 페이지와 액션 템플릿

XForms는 사용자 인터페이스를 기술하는 언어로 액션 템플릿을 작성할 수 있다. 액션 템플릿은 사용자 액션에 대한 템플릿 명세이며 XForms 페이지는 액션 템플릿의 집합이라고 볼 수 있다. 액션 템플릿은 접근 규칙과 같은 방식으로 $at = (t, p)$와 같이 표현된다.

XForms 페이지에서 input, select, output 같은 사용자 인터페이스 요소들이 액션 템플릿이다. 또한 insert, delete 등의 액션 요소도 액션 템플릿을 표현한다. XForms 요소와 액션 유형의 관계는 다음과 같다.

[정의 3] XForms 요소 *e*에 대해 대응하는 액션 유형 *e.type*은 다음과 같다.

- $l(e) = output$이면 $e.type = Read$,
- $l(e) = input, select, textarea$이면 $e.type = Update$,
- $l(e) = delete$이면 $e.type = Delete$,
- $l(e) = insert$이면 $e.type = Create$.

XForms 요소와 액션 유형 간에 관계를 단순화하기 위하여 여기서는 trigger 대신 insert와 delete 요소를 액션 템플릿으로 정하였다. 일반적으로 insert, delete 요소는 trigger나 submit의 자식 요소로 등장한다.

그 이외에 group이나 repeat 같은 인터페이스 요소들은 인스턴스 트리를 순회하는 구조를 표현한다. 액션 템플릿 요소들은 이 구조 요소의 자식으로 나타나는데, 구조 요소에 대응하는 DTD 노드들을 *템플릿 베이스* 노드라 부른다. XForms의 액션 템플릿 요소들은 ref, 또는 nodeset 속성을 가져서 컨텍스트 베이스 노드로부터의 상대 경로를 표시한다. XForms 언어에서 정의되는 액션 템플릿은 액션 타입 *t*, 템플릿 베이스 노드 v_b, 그리고 상대 경로 *ref*에 대해 $at = (t, v_b, ref)$로 표현할 수 있다.

<표 2>의 XForms 페이지 코드에서는 @type, description, @state의 Update 액션과 @author의 Read 액션, 그리고 task 요소의 삭제 액션을 정의한다. 여기서 템플릿 베이스 노드는 'task' 노드이다.

한편 액션 인스턴스는 액션 템플릿과 인스턴스 노드가 바

인딩되면서 생성된다. 같은 액션 템플릿이라도 인스턴스 노드에 따라 접근 허용 여부가 달라진다. 예를 들면 (그림 2)의 화면에서 task 요소의 삭제 버튼이 task에 따라 활성화 또는 비활성화된다.

4.2. 조건부 그래프 모델

XPath의 집합 P에 대해 경로 오토매타 $A(P)$가 만들어질 수 있다. 여기서 각 에지는 조건부가 아닌 노드 테스트 부분의 레이블을 나타내게 된다. 본 논문에서 다루는 XPath는 마지막에만 *나 //를 허용하므로 이 오토매타는 DTD 트리의 일부 노드를 가지는 트리가 된다. A(P)에서 각 XPath p의 $target(p)$에 속하는 노드들을 최종 노드라고 한다. 그림 1의 XML 예제에 대한 경로 오토매타가 그림 3의 위 부분에 보여진다.

XPath의 조건부 수식은 부분조건식의 and로 나타내진다. 예를 들어 $p1$ = /a/b[c='1']/d[e='3']이라면 $expr(p1)$ = (c='1') and (e='3')이 된다. XML 인스턴스 노드 x가 $answerset(p)$에 포함되려면 $x \in target(p)$이고 x가 $expr(p)$를 만족하여야 한다. x가 $expr(p)$를 만족하는가 여부는 $eval(expr(p), x)$로 계산되는데 이것은 p를 구성하는 부분식 e_i, $1 \le i \le m$에 대해 $\wedge_{1 \le i \le m}\ eval(e_i, x)$일 때 참이다.

조건부 수식의 상당 부분이 XPath 규칙들 간에 공통될 뿐 아니라 XML 인스턴스 노드들 간에도 조건부 수식의 계산 결과가 재사용될 수 있는 경우가 많다. 예를 들어 <표 1>에서 접근 규칙의 XPath들이 @level 속성의 값에 대한 조건식 부분을 공유한다. 기존 연구에서는 공통된 조건식 부분을 활용한 계산 방법은 있었으나 동일한 XPath가 연속된 XML 인스턴스 노드에 대해 계산되는 경우의 계산 결과 재사용은 검토된 바가 없다.

여기서는 부분조건식의 계산결과를 최대한 재사용할 수 있도록 하기 위하여 공통되는 조건식 부분을 모두 표현하는 조건부 그래프를 제안한다.

[정의 4] XPath $p \in P$에 대해 조건식 $expr(p)$이 부분조건식 $\{e_1, e_2, \cdots, e_m\}$를 가진다고 할 때 조건부 그래프 $G(p) = (Q_p, E_p, \{q_p\})$가 다음과 같이 정의된다.

i) $Q_p = Q_{op} \cup Q_{lpath} \cup Q_{value}$ 로 표시되며 수식의 각 연산자와 항에 대해 다음과 같이 노드가 대응된다.
   A. Q_{op} = {q | and, or, op의 연산자에 대응하는 노드 q},
   B. Q_{lpath} = {q | $lpath$ 경로식에 대응하는 노드 q},
   C. Q_{value}= {q | 리터럴 값이나 변수 v에 대응하는 노드 q}
ii) 각 부분수식 e_i에 대해서 $ast(e_i) = (Q^i_p, E^i_p, \{q^i_{sp}\})$이고 $q \in Q^i_{op}$이면 이 연산자의 좌우 피연산자 부분트리의 루트를 각각 q_l, q_r이라고 할 때 조건부 그래프의 에지로 (q,q_l), $(q,q_r) \in E^i_p$ 가 추가된다.
iii) $(q_p\ ,\ q^i_s,) \in E^i_p$, $1 \le i \le m$.

표기상의 편의를 위해 $ref(q)$와 $base(q)$는 $q \in Q_{lpath}$일 때 상대경로와 베이스 컨텍스트 노드를 표시한다. 여기서 q가 속하는 부분조건식이 e_i라면 $base(q) = base(e_i)$이다. 예를 들어 (그림 3)에서 $base(l_1)=v_{task}$이고 $base(l_5)=v_{comment}$이다. 한편 $q \in Qop$일 때 $op(q)$는 비교 또는 논리 연산자이다.

[정의 5] $G(p_k) = (Q_{pk}, E_{pk}, \{q_{sk}\})$, $p_k \in P$일 때 $N = \cup_{1 \le k \le m} Q_{pk}$라 하자. 두 노드 q_1, $q_2 \in N$가 다음과 같은 조건을 만족하면 동치라 하고 $q_1 \equiv q_2$로 표시한다.

i) $base(q_1) = base(q_2)$이고
ii) q_1, $q_2 \in Q_{value}$이면 $value(q_1) = value(q_2)$,
iii) q_1, $q_2 \in Q_{lpath}$이면 $base(q_1) = base(q_2)$이고 $ref(q_1) = ref(q_2)$,
iv) q_1, $q_2 \in Q_{op}$이면 $p(q_1) = op(q_2)$이고 $q_1.left \equiv q_2.left$ 이고 $q_1.right \equiv q_2.right$.

[정의 6] 접근 정책이 $P = \{(t_1, p_1),..., (t_n, p_n)\}$라 할 때 조건부 그래프 $G = (Q, E, Q_{start})$는 개별 XPath의 조건부 그래프 $G(p_k) = (Q_{pk}, E_{pk}, \{q_{sk}\})$, $1 \le k \le n$를에 대해 $Q = \cup Q_{pk}$, $E = \cup E_{pk}$, $Q_{start} = \{Q_{s1}, ..., Q_{sn}\}$에서 동등한 노드를 하나로 합친 것이다.

접근정책이 주어지면 경로 오토매타와 조건부 그래프를 생성할 수 있다. 이를 이용하여 효율적으로 접근 권한을 검사하는 방법을 찾는 것이 이 논문의 목적이다. 다음 정의는 먼저 접근 권한을 경로 오토매타와 조건부 그래프를 이용하여 정의한다.

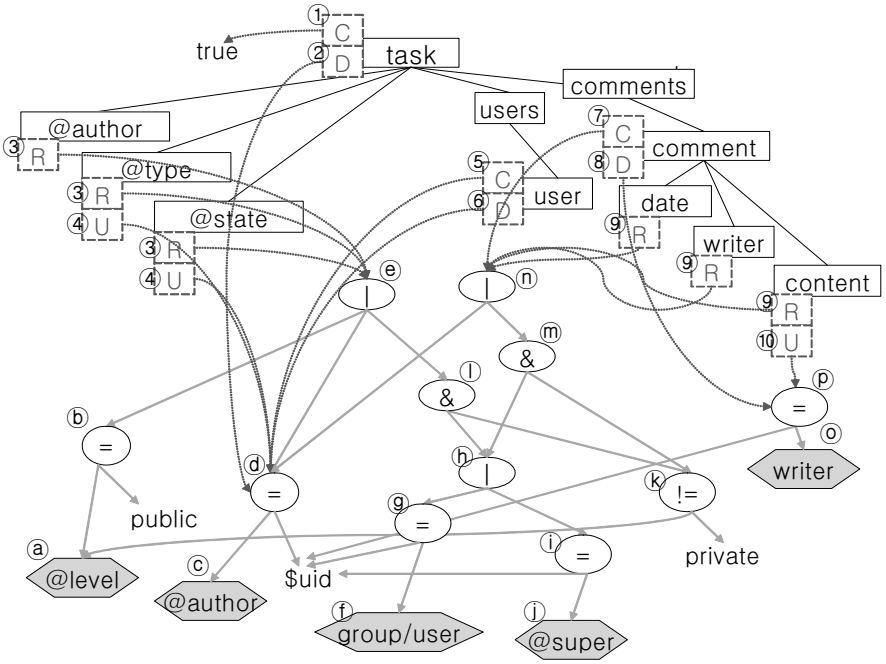
[정의 7] 접근 정책 $P = \{(t_k, p_k) \mid 1 \le k \le n\}$와 p_k의 조건부 그래프 $G(p_k)$를 통합한 그래프 G가 주어졌다고 하자. 규칙 (t_k, p_k)에 대한 조건부 그래프 $G(p_k)$에서 시작노드가 q_k이고 대상노드 집합이 $V_i = target(p_i)$라 하면 해당하는 규칙은 다음과 같이 표시된다.

$$R = \{(t_k, V_k, q_k) \mid V_k = target(p_k),\ 1 \le k \le n\}.$$

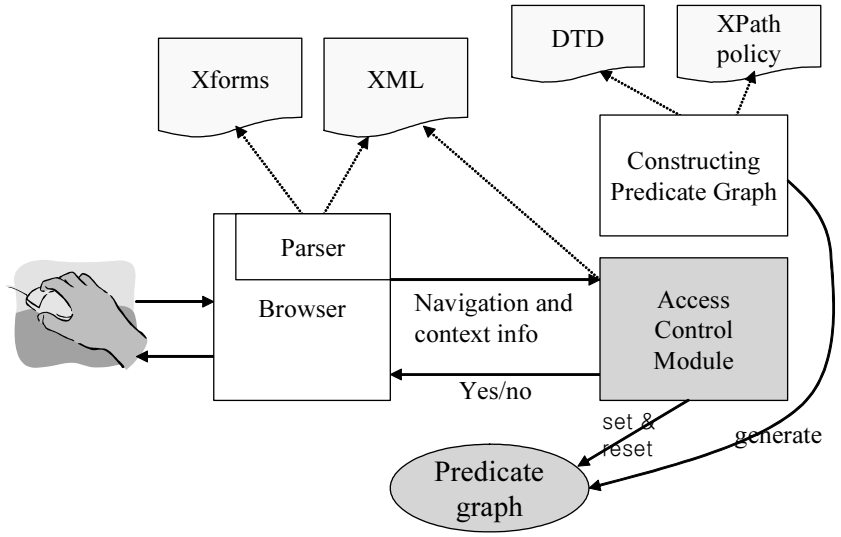
(그림 3)은 경로 오토매타와 조건부 그래프, 그리고 접근 권한을 함께 표현한 것이다. 여기서 C, D, R, U 로 표시된 사각형은 각각 추가, 삭제, 읽기, 수정 권한 규칙을 나타낸다. 이 액션이 허용되는 DTD 노드에 표시되었다. 한편 이들 규칙의 조건부를 조건부 그래프의 노드로 표시된다. 이때 조건부 그래프의 노드는 공통되는 부분 조건식에 대해 공유되고 있음을 볼 수 있다. 한편 조건부 그래프는 연산자 노드((그림 3)의 원 노드)와 경로 노드((그림 3)의 육각형 노드), 그리고 값을 가지는 노드로 나누어지고 각 규칙에 대해 하나의 시작노드를 가진다. 시작노드는 연산자 노드이다.

4.3. 계산 알고리즘

액션의 접근제어는 바인딩된 XML 데이터에 대해 만족되



(그림 3) <표 1>의 접근 정책에 대한 조건부 그래프 (일부)



(그림 4) 접근 제어 모듈을 포함하는 클라이언트 시스템 아키텍춰

는 접근 규칙이 있는가를 찾는 계산이다. 한편 정적인 접근 제어의 계산은 XML 데이터와 조건부를 고려하지 않고 접근 규칙의 타겟노드와 액션 템플릿의 타겟 노드를 검사하는 것으로 액션 템플릿에 대해 해당하는 규칙이 없다면 접근 불가라 부른다.

[보조정리 8] 주어진 액션 템플릿을 $at=(t, p)$라 하자. 만약 $t = t_r$, $target(p) \subset V_r$를 만족하는 규칙 $r=(t_r, V_r, q_r) \in R$이 없다면 at는 접근 불가이다. 접근 불가 템플릿은 어떤 XML 인스턴스 노드에 대해서도 허용되지 않는다.

[정리 9] 액션 템플릿 $at = (t, p)$와 XML 인스턴스 x가 주어졌고 $M_{X \to D}(x) = v$라 하자. 액션 인스턴스 $at(x)$는 다음 조건을 만족하면 허용된다.

(1) $t = t_r$, $target(p) \in V_r$을 만족하는 접근규칙 $r = (t_r, V_r, q_r) \in R$이 존재하고

(2) $evalExpr(q_r, x) = true$를 만족한다.

본 논문에서 고려하는 클라이언트 시스템은 UI 생성 모듈(일반적으로 브라우저임)과 독립적으로 존재하는 접근 제어 모듈이 있어서 주어진 액션 인스턴스가 허용되는가를 계산하는 역할을 담당할 수 있다. 전체 시스템 구조는 (그림 4)에 나와 있다. XForms 페이지는 사용자 인터페이스 요소와 서버 통신 정보를 가지고 있다. 브라우저는 XForms 요소를 차례로 방문하면서 바인딩되는 모든 XML 데이터에 대해

액션 인스턴스를 생성하게 된다. 이때 각 액션 인스턴스에 대해 네비게이션 정보와 컨텍스트 정보를 제공하여 접근 제어 모듈에게 허용 여부를 물어본다.

[알고리즘 1] 접근권한검사(e, x, vb, v)
입력 - e : XForms 요소
- x : 현재의 컨텍스트 노드
- vb : 템플릿 베이스 노드
- v = $M_{X\to D}$(x)

출력 - 액션 인스턴스의 접근이 허용되면 참을 반환하고 아니면 거짓을 반환함

static : xprev ← null
(1) reset(v, $M_{X\to D}$(xprev))
(2) xprev ← x ;
(3) if e.name ∈ {input, output, delete, insert,...} :
(3-1) t ← e.type
(3-2) for all r = (t, Vr, qr) s.t. v ∈ Vr
(3-2-1) if evalExpr(qr, v, x) return true;
(3-3) return false;

[알고리즘 1]은 XForms 요소와 컨텍스트 정보를 이용하여 접근 제어를 계산한다. 함수 CheckInaccessible은 DTD 컨텍스트 노드를 xprev 변수에 유지하면서 다음 호출에서 바뀌었는지 검사한다. 바뀐 경우 reset 함수를 호출하는데 여기서는 새로운 컨텍스트 노드에 대해 기계산된 임시 결과 중에서 무효가 된 조건부 계산 값들을 리셋한다[알고리즘 2]. 또한 evalExpr 함수는 주어진 인스턴스 노드 x에 대해 조건부 그래프의 노드 qr에 해당하는 조건식 부분을 계산한다. 이 때 재사용 가능한 계산 결과는 다시 계산하지 않는다[알고리즘 3].

XForms 페이지는 repeat나 group으로 구조화된 액션 템플릿 트리이다. 접근제어의 계산이 XML 인스턴스 트리의 순차적 방문 순서를 따른다면 조건부 수식의 계산 결과가 많은 부분 재사용될 수 있다. 다음 보조정리는 연속된 액션 인스턴스에 대해 조건부 계산 결과를 재사용할 수 있는 조건을 제시한다.

[보조정리 10] 액션 템플릿 $at = (t, v)$과 연속된 바인딩 인스턴스 노드 x_1, x_2가 주어졌을 때 q가 $at(x_1)$에 대해 계산되었다면 다음 조건을 만족하면 $at(x_2)$가 q의 계산 결과를 재사용할 수 있다.
i) ∃ $(t, V, q_s) \in R$ *s.t.* $v \in V$,
ii) q_s가 q로의 경로를 가지고 있고
iii) $base(q)$가 v의 조상이고 $base(q) \neq$ v인 경우

이를 일반화하여 임의의 두 액션 인스턴스에 대해 조건부 노드의 계산 결과를 재사용하기 위한 조건은 다음과 같다.

[보조정리 11] 액션 템플릿 $at_1 = (t_1, v_1)$와 $at_2 = (t_2, v_2)$가 주어지고 $M_{X\to D}(x_1) = v_1$, $M_{X\to D}(x_2) = v_2$인 바인딩 인스턴스 노드 x_1, x_2가 주어졌다고 가정한다. 조건부 노드 q가 $at_1(x_1)$에 대해 계산되었고 유효할 때 다음 조건을 만족하면 $at_2(x_2)$가 q의 계산 결과를 재사용할 수 있다.
i) ∃ $(t_1, V_1, q_{s1}), (t_2, V_2, q_{s2}) \in R$ *s.t.* $v_1 \in V_1$, $v_2 \in V_2$,
ii) q_{s1}과 q_{s2}가 q로의 경로를 가지고 있고
iii) $base(q) \neq v_1$ 또는 $v_1 \neq v_2$이면서 $base(q)$가 v_1과 v_2의 공통 조상인 경우

그러므로 바인딩 인스턴스 노드가 바뀌었을 때 재사용 가능하지 않은 모든 계산값은 리셋되어야 한다. 즉 조건부 그래프 노드 중에서 베이스 노드가 새로운 노드의 조상이 아닌 경우는 리셋되어야 한다. 중간 계산 결과는 두 종류로 나눌 수 있다. 연산자 노드는 참 또는 거짓을 가지며, 아직 계산되지 않았거나 재사용가능하지 않은 경우 ⊥을 가진다. 또한 lpath 노드인 경우는 계산된 노드 집합을 가지거나 ⊥을 가진다. 표현 상의 편의를 위하여 조건부 그래프의 노드에 대해 다음을 정의한다.

- $nodeset(q) \subset N_{XML}$ or $nodeset(q) = \perp$ if $q \in Q_{lpath}$
- $evalValue(q) \in \{\perp, true, false\}$ if $q \in Q_{op}$

이상의 결과를 이용하면 컨텍스트 노드가 바뀌었을 때의 리셋 알고리즘이 다음과 같다. 리셋 알고리즘은 원래 컨텍스트 노드 x1과 새로운 컨텍스트 노드 x2에 대응하는 DTD 노드를 입력으로 받는다. 입력된 두 DTD 노드가 같은 경우는 동일한 액션 템플릿에 대해 다른 인스턴스 노드를 바인딩한 경우이다. 그리고 v1과 v2가 다른 경우는 액션 템플릿이 바뀌거나 repeat 또는 group 등으로 다른 인터페이스 요소로 이동한 경우이다.

[알고리즘 2] reset(v1, v2)
입력 - 조건부 그래프 G = (Q, E, Qstart),
- 이전 및 현재의 템플릿 베이스 노드 v1과 v2

출력 - 무효화된 조건부 노드들을 리셋함

(1) if v1 = null or v1 is an ancestor of v2, then return;
(2) if v1 ≠ v2, w ← common ancestor node of v1, v2;
(2-1) else w ← v1.parent;
(3) v ← v2;
(4) repeat
(4-1) for all q ∈ Q_{lpath} s.t. base(q) = v,
(4-1-1) nodeset(q) ← ⊥;
(4-1-2) for all q′ s.t. (q′, q) ∈ E
(4-1-2-1) evalValue(q′) ← ⊥;
(4-2) v ← v.parent;
(5) until v ≠ w;

이제 조건부에 대한 계산 알고리즘을 살펴볼 준비가 되었다. 주어진 조건부 그래프 노드 q와 컨텍스트 베이스 DTD 노드 v, 그리고 XML 인스턴스 노드 x에 대해서 다음 알고리즘은 q에 해당하는 조건부 부분식이 x에 대해 참인가를

계산한다.

[알고리즘 3] evalExpr(q, v, x)

입력 - q ∈ V, G = (Q, E, Q_{start})는 조건부 그래프임,
- v ∈ N_{DTD} (템플릿 베이스 노드)
- x : $M_{X \to D}(x)$ = v를 만족하는 XML 인스턴스 노드

출력 - q에 해당하는 부분조건식이 x에 대해 만족되면 참, 아니면 거짓을 반환함

(1) result ← false;
(2) if evalValue(q) ≠ unknown, return evalValue(q).
(3) if q ∈ Q_{start},
(3-1) result ← true;
(3-2) for all q' ∈ next(q),
(3-2-1) if eval_expr(q', x) = false
(3-2-1-1) result ← false; break;
(4) if q ∈ Q_{lpath},
(4-1) if nodeset(q) = unknown
(4-1-1) nodeset(q) ← evalNodeSet(q, x)
(4-2) result ← (nodeset(q) ≠ ∅)
(5) if q ∈ Q_{op} and op(q) ∈ {and, or},
(5-1) Let q_{left}, q_{right} ∈ next(q),
(5-2) result ← op(q)(evalExp(q_{left}, x), evalExpr (q_{right}, x)).
(6) if q ∈ Q_{op} and op(q) is an comparative operator,
(6-1) pathdiff ← path(v, base(q_{left}));
(6-2) for all x' ∈ decsendent(x, pathdiff);
(6-2-1) for all x" ∈ evalNodeSet(q_{left}, x')
(6-2-1-1) if op(q)(x", q_{right})
(6-2-1-1-1) result ← true;
(6-2-1-1-2) break;
(7) evalValue(q) ← result;
(8) return result;

evalNodeSet(q, x) = descendents(x, lpath(q))

위의 알고리즘 1, 2, 3은 다음과 같은 단계로 XForms 요소들을 처리하게 된다.

〈표 2〉 예제 XForms 페이지 코드 (일부)

```
<table>
  <xf:repeat ref = 'task'>
    <tr>
      <td><xf:input ref='@type'></td>
      <td><xf:output ref='@author'></td>
      <td><xf:input ref='description'></td>
      <td><xf:input ref='@state'></td>
      <td><xf:trigger><xf:delete ref='.'/></xf:trigger></td>
    </tr>
  </xf:repeat>
```

i) repeat나 group 노드는 컨텍스트 노드를 새로 설정한다.
ii) 컨텍스트 노드가 바뀐 경우 리셋 함수를 통해 무효가 된 계산 결과를 리셋한다.
iii) 액션 템플릿 요소에 대해 XML 노드와 바인딩된 액션 인스턴스의 접근 허용 여부를 계산한다. 이 때 계산된 값을 가진 노드는 그대로 이용한다.
iv) 각 계산 단계에서는 계산 결과를 조건부 그래프 노드에 설정해 둔다.

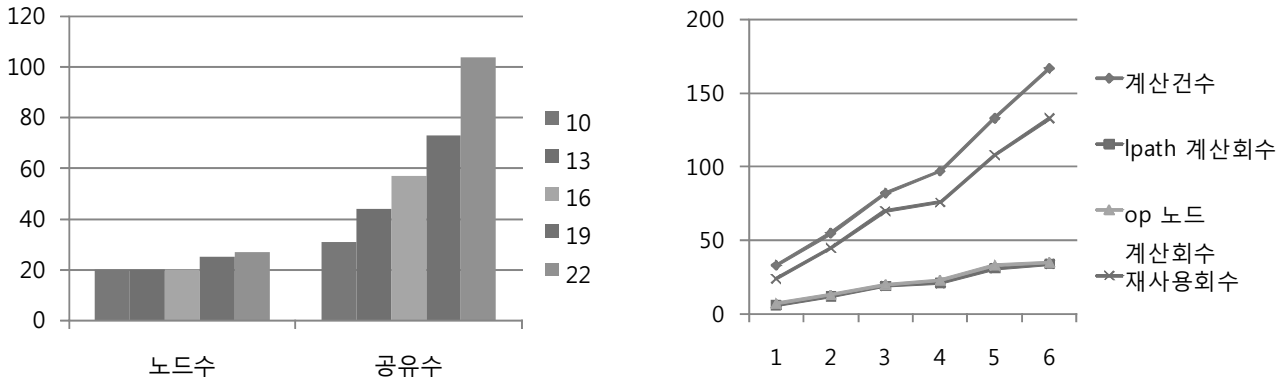
<표 3>은 <표 2>의 XForms 페이지 부분과 (그림 1)의 xml 트리 인스턴스에 대해 XForms 요소 별로 바인딩된 xml 노드에 대해 조건부 노드의 계산 또는 재사용 과정을 보여준다.

(그림 5)는 시뮬레이션에 의해 접근 제어 규칙의 개수에 따른 조건부 그래프 노드 수의 증가 추세와 XForms 페이지의 액션 인스턴스 개수의 증가에 따른 재사용 노드 수 및 비율을 보여준다. 이 표에서 보여지듯이 제안된 조건부 그래프의 계산 결과는 바인딩 컨텍스트 노드가 바뀔 때 한번 계산된 후 계속 재사용됨을 알 수 있다.

제안된 접근 제어 방법은 적응형 사용자 인터페이스 기법으로 응용이 가능하다. 예를 들면 Update에 대한 액션 템플릿 요소가 접근이 거부된 경우(*) Read 연산으로 다시 계산

〈표 3〉 조건부 재사용을 통한 접근 제어 규칙 계산 과정

바인딩컨텍스트 노드		대상 노드 상대 경로	바인딩컨텍스트 노드	규칙	계산노드	재사용	리셋	허용
repeat task		task	task SU					
	input (*)	@type		R4	c, d			X
	output(**)	@type		R3	a,b,e			O
	input	@state		R4		d		X
	output	@state		R3		e		O
	output	@author		R3		e		O
	...							
	insert	comments/comment		R7	f,g,h,k,m,n	d		X
	delete	.	task BP	R2		d		X
			task BP				a,c,f -> b,d,g, -> e,h,m,n	
	input	@type		R4	f,c			X
	output	@type		R3	e,b,a			X



(그림 5) 실험결과 (a) 규칙의수에 따른 조건부그래프노드개수 및 공유회수 (b) 접근제어계산시 조건부 재사용회수

하여 수정 기능은 제공하지 않더라도 데이터를 보여주기만 하는 것은 가능한지 확인할 수 있다(**). 한편 브라우저에서 group이나 repeat 등에 대해 포함되는 모든 사용자 인터페이스 요소들이 허용되지 않는다면 아예 전체를 숨기는 것이 가능하다. 예를 들어 <표 3>에서 task BP에 대한 모든 xforms 요소의 액션 인스턴스가 거절되므로 repeat 전체를 생략하는 것이 가능하다. 이것은 브라우저에서 접근제어 모듈을 호출하는 부분을 확장하여 구현될 수 있다.

5. 결 론

본 논문에서는 XForms 페이지에 대한 XPath 기반의 접근 규칙을 효율적으로 계산하는 알고리즘을 제안하였다. 기존의 XPath 접근 제어 계산 방법과 달리 여기서는 주어진 액션 템플릿과 바인딩되는 XML 인스턴스 노드에 대해 접근 제어가 계산되어야 한다. 연속되는 액션 템플릿의 계산에서 재사용 가능한 조건부 계산 결과를 모두 찾기 위하여 조건부 그래프를 제안하였고 계산 결과가 무효가 될 때 reset 하는 알고리즘을 제안하였다. 이 방법은 각 조건부 부분식이 xml 노드에 대해 여러 번 계산되는 것을 방지해 준다. 제안된 방법을 XForms 예제 페이지에 대해 접근제어하는 시나리오를 통하여 사례를 살펴보았다.

제안된 방법은 현재 연구팀에서 개발된 XForms 브라우저인 Xeni 플랫폼[18]에 확장 개발 중이며 조건부 노드의 재사용을 통해 XForms 페이지의 접근 제어에서 상당한 성능 향상을 보일 것으로 기대된다.

제안된 방법은 XSL이나 XUL, 또는 XML 기반의 다른 사용자 인터페이스 기술 언어에도 적용될 수 있는데, 사용자 인터페이스 요소를 액션 템플릿으로 보고 XML 데이터에 대한 접근 권한을 계산하는 시스템 모델에서는 모두 적용가능하다. 또한 제안된 XPath 기반의 사용자 인터페이스 접근 제어 방법은 적응적인 사용자 인터페이스 생성을 위한 분야에서도 활용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] Damiani, E. et al., "A Fine-Grained Access Control System for XML Documents," *ACM Trans. on Information and System Security*, Vol.5, No.2, pp.169-202, May 2002.

[2] B. Luo, D. Lee, W.-C. Lee, and P. Liu, "QFlter:Fine-grained run-time xml access control via nfa-based query rewriting," Proceedings of the thirteenth ACM international conference on Information and knowledge management, pp.543.552, NewYork, USA, 2004.

[3] InfoPath 2007, http://office.microsoft.com/infopath.

[4] S. Thompson, and T. Torabi, "A Process Improvement Approach to Improve Web Form Design and Usability," 18th International Conference on Database and Expert Systems Applications, pp.570-574, 2007.

[5] 월드와이드웹, XML, DTD, XPath, 기타 표준, http://www.w3c.org.

[6] E. Bruchez, "XForms and the eXist XML database: a perfect couple," Wellesley, XML Conference and Exibition, Marriott Copley Place Boston, Massachusetts, USA, December 2007.

[7] Fundulaki, I.andMarx, M.,"Specifying Access Control Policies for XML Documents with XPath," In *Proc. 9th ACM Symp. on Access Control Models and Technologies*, pp.61-69, Yorktown Heights, New York, June, 2004.

[8] C.-H. Lim, S. Park, and S. H. Son. "Access control of xml documents considering update operations," Proceedings of the 2003 ACM workshop on XML security, pp.49-59, NewYork, USA, 2003.

[9] J. Lee, K. Whang, W. Han, and I. Song, "The dynamic predicate: integrating access control with query processing in XML databases," VLDB Journal, Vol.16, No.3, pp.371-387, July, 2007.

[10] M. Murata, A. Tozawa, and M. Kudo, "XML Access Control Using Static Analysis," In *Proc. 10th ACM Conf. on Computer and Communications Security*, pp.73-84, Washington DC, USA, Oct., 2003.

[11] J. Jeon, Y. Chung, M. Kim, Y. Lee, "Filtering XPath expressions for XML access control," Computers and Security Vol.23, pp. 591-605, 2004.

[12] T. Calders, S. Dekeyser, J. Hidders, and J. Paredaens, "Analyzing workflows implied by instane-dependent access

rules," PODS'06, pp.100-109, Chicago, USA, June, 2006.
[13] A. Gupta and D. Suciu, "Stream processing of xpath queries with predicates," Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp.419-430, San Diego, California, 2003.
[14] S. Hou, and H. Jacobson, "Predicate-based filtering of XPath expressions," ICDE'06, pp.53-53, 2006.
[15] K. Song, and K. Lee, "An automated generation of xforms interfaces for web services," IEEE International Conference on Web Services 2007, pp.856-863, Seoul, Korea, July 2007.
[16] J. He, amd I. Yen, "Adaptive User Interface Generation for Web Services," Proceedings of the IEEE International Conference on e-Business Engineering, pp.536-539, 2007.
[17] Carminati, B. and Ferrari, E., "AC-XML Documents: Improving the Performance of a Web Access Control Module," In *Proc. 10th ACM Symp. on Access Control Models and Technologies*, pp. 67-76, Stockholm, Sweden, June, 2005.
[18] 유가연, "오픈 API 플랫폼을 위한 XForms 브라우저 개발," 석사학위논문, 경기대학교 일반대학원 컴퓨터과학과, 2007.

이 은 정

e-mail : ejlee@kyoggi.ac.kr
1988년 서울대학교 계산통계학과(학사)
1990년 한국과학기술원 전자계산학과 (공학석사)
1994년 한국과학기술원 전자계산학과 (공학박사)
1994년~2000년 전자통신연구원 선임연구원
2001년~현 재 경기대학교 전자계산학과 부교수
관심분야 : XML 처리 기술, 웹서비스