# 인라인 정규트리문법

유가연, 이은정
경기대학교 일반대학원 전자계산학과
e-mail: {despoin, ejlee}@kyonggi.ac.kr

# Inlined Regular Tree Grammar

Ga-Yeon Yoo, Eun-Jung Lee
Dept of Computer Science, Kyonggi University

요 약

형식 언어에서 XML 문서를 정의하는 방법으로 정규트리문법을 이용할 수 있다. 한편 XML 데이터에서 재귀와 반복과 같이 필요한 경우만을 제외하고 터미널 노드를 상위 노드의 직접 자식으로 여기는 것을 인라인이라고 한다. 본 논문에서는 XML 데이터 처리에서 사용되는 인라인 기법을 정규트리문법에 적용하여 터미널 노드만으로 이루어진 터미널 언어를 정의하고, 정규트리문법에 대응하는 인라인 정규트리문법을 소개한다. 또한 일반적인 정규트리문법으로부터 인라인 정규트리문법으로의 변환 알고리즘을 소개한다.

## 1. 서 론

프로그램간의 데이터 교환을 위한 수단으로써의 XML은 컨텐츠 모델을 정의하는 DTD나 XML 스키마를 통해 XML 인스턴스를 정의하게 된다. 형식 언어에서 XML 문서를 표현하는 방법으로 정규트리문법을 이용할 수 있다. 정규트리문법은 트리들의 집합을 언어로 가지는 문법으로 유효한 트리의 집합을 정의할 수 있다.

XML데이터를 어플리케이션에서 사용하기 위하여 트리 내부의 노드의 구조 정보를 생략하고 값을 가지는 터미널 노드를 중심으로 트리를 표현하는 것을 인라인이라 한다. 인라인된 트리는 터미널 노드만이 XML인스턴스를 표현하는 값을 가진다는 점을 이용하여 반복과 재귀 등의 필요한 경우에만 구조정보를 유지하고 그 이외의 경우에는 내부 노드를 제거하고 상위노드의 직접 자식으로 표현한다.

인라인을 적용한 몇 가지 XML 연구를 찾아 볼 수 있다. XML 데이터 정의로부터 관계형 데이터베이스 테이블을 생성하는 방법[2] 으로서, 반복과 재귀의 경우를 제외하고 모든 데이터를 하나의 테이블로 표현하는 방법이다. 또한 XML 데이터의 인라인 방법[1]에서는 인라인 기법을 적용하여 최소 개수의 클래스를 생성할 수 있는 새로운 XML 바인딩 방법을 제안하였다.

본 논문에서는 XML 데이터 처리에서 사용되는 인라인 기법을 정규트리문법에 적용하여, 터미널 노드만으로 이루어진 터미널 언어를 정의하고 정규트리 문법에 대응하는 인라인 정규트리 문법을 소개하고 알고리즘을 살펴본다.

## 2. 관련연구

### 2.1 정규트리문법

트리를 정의하기 위한 방법으로 트리들의 집합을 언어로 가지는 정규트리 문법이 있다. 기존의 정규

트리 문법[2]에 대한 연구에서 트리에 대한 정규트리문법 G=(N, T, S, P)를 다음과 같이 정의한다.

**[정의 1]**

*N : 넌 터미널 심볼들의 유한 집합*

*T : 터미널 심볼들의 유한집합*

*S : 시작 심볼들의 집합(S는 N의 유한집합)*

*P : X→ aγ의 형식을 가지는 생성규칙들의 집합, X ∈N, a∈T, γ은 N에 대한 정규표현이다.*

*항상 X는 왼쪽, aγ은 오른쪽에 표현한다. 그리고 γ은 이 생성규칙의 컨텐츠 모델이라고 부른다.*

이 논문에서는 위의 정규트리 문법에서 추가적으로 터미널 노드를 다음과 같이 정의하고자 한다.

**[정의 2]** ***터미널 노드****는 PCDATA와 같은 데이터 노드가 아니라 자식으로 데이터 노드를 가지며, 다른 요소를 자식으로 가지지 않는 노드이다. 속성 노드는 모두 터미널 노드에 포함된다.*

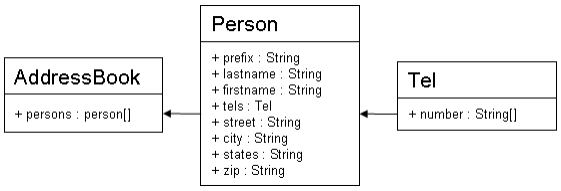
### 2.2 인라인 DTD

XML 정의문법에서 바인딩 클래스를 생성하여야 할 요소들을 추출하는 방법으로 DTD에 인라인을 적용한 연구[1]가 있다.

```
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name, tel, address)>
<!ELEMENT name (prefix, lastname, firstname)>
<!ELEMENT tel (number)*>
<!ELEMENT address (street, city, states, zip)>
```

(그림 1) 바인딩 클래스를 작성할 DTD

(그림 1)의 DTD에서 name이나 address와 같이 #PCDATA타입의 자식 요소들만을 가지는 요소들에 대해서 모두 클래스를 생성하는 것이 아니라 반복이나 생략 또는 재귀적으로 호출되는 요소들에 대해서만 클래스를 생성하게 된다.



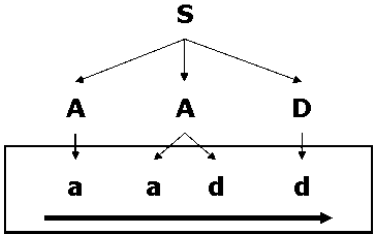

(그림 2) 인라인 적용 바인딩 클래스

(그림 2)의 인라인이 적용된 바인딩 클래스를 살펴보면 구조표현에 필요한 반복요소 person과 tel을 제외하고 모두 상위 요소 클래스에 직접 필드로 생성된 것을 알 수 있다.

### 3. 인라인 정규트리의 터미널 언어

2장에서는 정규트리 문법에 대한 정의를 살펴보았다. 이번 장에서는 인라인 트리 문법을 선언하기 위하여 터미널 문장 및 터미널 언어를 정의한다.

**[정의 3]** *XML 트리 인스턴스 t에 대해, t의 터미널 노드에 대응하는 심볼을 왼쪽에서 오른쪽으로 순서대로 연결한 스트링을* ***터미널 문장****이라고 한다.*




(그림 3) 트리 인스턴스 t

**[예 1]** 위의 정의에 의하여 (그림 3)에서의 터미널 문장은 x = aadd이다.

**[정의 4]** *주어진 정규트리 문법 G에 대해 x∈T*을 터미널 문장으로 가지는 트리 t가 G의 언어에 속할 때 x를 G의* ***터미널 문장****이라고 한다.*

*또한 G의 터미널 언어 W(G) = {x|x는 t의 터미널 문장이고 t ∈ L(G)}으로 표현한다.*

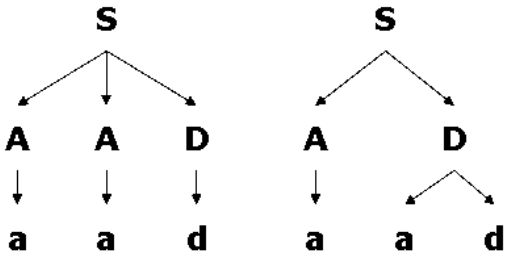
**[예 2]** 아래의 문법 G1의 터미널언어는 W(G1) = (a|ad|d)*(ad|d)로 표현할 수 있다.

(문법 G1)
S → A* D
A → a | D
D → ad | d

(그림 4) 문법 G1

**[정의 5]** *정규트리 언어에서 한 터미널 문장을 표현하는 인스턴스가 두 개 이상인 경우 이 터미널 문장을* ***모호하다****고 한다. 정규트리 문법 G가 모호한 터미널 문장을 가지지 않을 때* ***터미널 문장 결정적****이라고 한다.*

**[예 3]** (그림 4)의 정규트리문법 G1은 모호한 터미널 문장 x1 = add를 가지므로 터미널 문장 결정적이지 않다.




(그림 5) 문법 G1의 인스턴스 트리

위의 정의를 통하여 터미널 문장 결정적인 터미널 문장은 한 터미널 문장을 표현하는 인스턴스를 오직 하나만 가지기 때문에 아래의 정리가 명백함을 증명할 수 있다.

**[정리 1]** *터미널 문장 결정적인 정규트리 문법은 터미널 문장만으로 트리구조를 결정할 수 있다.*

### 4. 인라인 정규트리 문법

이 절에서는 정규트리 문법에 대응하는 인라인 정규트리 문법을 정의하고, 정규트리 문법으로부터 바로 인라인 정규트리 문법을 얻을 수 있는 알고리즘을 소개한다.

인라인 정규트리문법을 정의하기 위해서는 각 심볼의 생성규칙을 터미널 심볼만으로 이루어진 정규표현으로 나타내고자 한다.

그러나 자기 자신으로 돌아오는 경로가 있는 ***재귀 심볼***의 경우는 터미널 심볼만으로 이루어진 정규표현을 나타낼 수 없으므로 이들을 포함한 터미널 정규표현을 ***인라인된 정규표현***이라고 한다. 여기서 재귀 심볼 집합이란 사이클을 이루는 넌 터미널 심볼들 중에서 시작 심볼에서 가장 가까운 심볼의 집합을 나타낸다.

**[정의 6]** *정규트리 문법 G에 대해* ***인라인 정규트리문법 GI****의 생성규칙 집합* ***PI****는 다음과 같이 정의된다. 문법 G의 재귀 심볼 집합을 R이라 하자.*

*PI = {A -> aγ′ | A -> aγ ∈ P, γ′은 γ의 인라인 정규표현, A ∈ {S} U R }*

*A -> aγ ∈ P, γ = $X_1X_2...X_n$이고 Xi ∈ T U N U 정규표현기호 라면, 이에 대응하는 인라인 생성규칙은 A->aγ′이고 γ′ = $w_1w_2...w_n$이다.*

*여기서,*

*i) $w_i = X_i$, if $X_i < R$ U T U 기호이고,*

*ii) $w_i = W(X_i)$, if $X_i < N - R$이다.*

다음은 정규트리 문법을 인라인 정규트리 문법으로 변경하기 위한 알고리즘을 살펴본다.

**[알고리즘] inline_rgt**
Input : 문법 G, 넌 터미널 심볼 A
output : 인라인 생성규칙 집합 PI
넌 터미널집합의 경로 path=ε, 재귀심볼집합 rv ={S}

I . rv = ε일 때까지 다음을 반복한다.
1. A∈rv, rv = vr - {A}
2. γ = $X_1X_2...X_n$
a. 만일 심볼 $X_i$가 넌 터미널 A라면
i . if A ∈ path or A ∈ rv
① rv ← rv U {A}
② γ′ ← γ′ · A
ii A ∈ T U 정규표현기호
① γ′ ← γ′ · A
iii . else
① γ′ ← γ′ · inline_rgt(G, A)

**[예 4]** 문법 G2의 인라인 정규트리문법은

| (문법 G2) |
|---|
| S → A (b*d \| D) E* |
| A → a \| De |
| D → C \| aA |
| E → c \| f* |

G2I = {S -> A(b*d | c | aA) (c | f*)*, A -> a | ce | aAe}이다.

**[정의 7]** *모든 트리에 대하여 하나의 인라인 정규트리문법으로 표현할 수 있는 정규 트리문법에 속하는 트리가 하나인 경우 그 트리는* ***결정적****이다.*

**[예 5]** G1I = {S → (a|ad|d)*(ad|d)}의 인라인 정규트리문법은 아래와 같이 하나 이상의 문법으로 표현이 가능하므로 인라인 정규트리 문법 결정적이지 않다.

| (문법 G1-1) | (문법 G1-2) |
|---|---|
| S → A* D | S → A* D |
| A → a \| D | A → a \| ad \| d |
| D → ad \| d | D → ad \| d |

위의 정의를 통하여 정규트리 문법 결정적인 터미널 문장은 한 터미널 문장을 표현하는 인스턴스를 오직 하나만 가지기 때문에 아래의 정리가 명백함을 증명할 수 있다.

**[정리 2]** *터미널 문장 결정적인 정규트리 문법은 터미널 문장만으로 트리구조를 결정할 수 있다*

## 5. 결론

본 논문에서는 XML데이터를 정의하기 위한 정규트리 문법에 대응하는 인라인 정규트리 문법을 정의하고 정규트리 문법을 인라인 정규트리 문법으로 바꾸는 알고리즘을 소개하였다.

제안된 방법은 어플리케이션에서 XML 데이터의 컨텐츠 모델을 정의하고 인라인을 위한 알고리즘으로서 쓰일 수 있을 것이다.

## 참고문헌

[1] 이은정, 유가연, “XML 데이터의 인라인 바인딩 방법”, '정보처리학회논문지A' 제13-A권 제 1 호, 2006년 2월

[2] Makoto Murata, Dongwon Lee, Murali Mani, Kohsuke Kawaguchi, “Taxonomy of XML Schema Languages Using Formal Language Theory”, In ACM Transactions of Internet Technology (TOIT), November 2005.

[3] Boris Chidlovskii, "Using Regular Tree Automata as XML schemas", 2000, IEEE.

[4] Anne Bruggemann-Klein, Derick Wood, "One-Unambiguous Regular Languages", May 28, 1994.

[5] M. Mani, “Constraint Specification for XML: A Closer Look”, Extreme Markup Languages, Montreal, Canada, August 2004.

[6] Robert A. van Engelen, “Constructing Finite State Automata for High-Performance XML Web Services”, International Conference on Internet Computing 2004. 975-981.

[7] Martin Kempa, Volker Linnemann, “Towards Valid XML Applications”, SSGRR, 2002.

[8] Elsevier B.V. Shiyong Lu, Yezhou Sun, Mustafa Atay, Farshad Fotouhi, "On the consistency of XML DTDs", 2004.