파스 트리를 이용한 불완전한 XML 문서의 완성

조용윤*, 박용관, 유재우
서울대학교 컴퓨터학과
(ycho*, psiver)@amin.ssu.ac.kr, cwyoo@comp.ssu.ac.kr

Completion of Incomplete XML Document Using Parse Tree

Yong-Yoon Cho*, Ho-Byung Park, Chae-Woo Yoo
School of Computer, SoongSil University

요 약

프로그래머는 프로그램 작성 중 문법적으로 올바르지 않은 문장을 입력할 수 있다. 문법적으로 불완전한 문장 입력은 사용자의 계속적인 편집 상태를 중요하고, 정상적인 파싱을 보장하지 않는 원인이 된다. 따라서, 사용자는 편집기에 제공하는 적절한 오류 처리를 통해 문법적으로 잘못된 문장에 대해 수정 작업을 실시해야 한다. 에디터 환경에서의 사용자 입력은 입력 부분의 완료, 오류에서 전환하지 못한 형태로 입력될 수 있다. 따라서, 에디터가 문법적으로 빠져있는 부분의 실질이 무엇인지 정확히 인식할 수 없고, 사용자가 입력한 문장의 완성과 개선된 원문을 보급할 수 있는 방법이 필요하다. 본 논문의 핵심은 사용자의 문법적으로 올바르지 않은 문장 입력에 대해 불완전한 문장 입력을 정의하고, 이를 사용자 입력에 대해 개선한 원문을 보급할 수 있는 방법을 제시한다.

1. 서론

프로그래머 편집기의 파서에 의해 프로그래밍 문법적으로 바르게, 결과로 파스 트리로 표현된다. 편집 기반의 프로그래밍 편집 환경에서 사용될 수 있는 상황적 파산 기법이 있다. 파산 트리가 상형으로 생성되는 점에서는 특이하게 논의의 논리적 분류 수단의 흐름을 잃어버리지 못하도록 하여, 이에 대부분의 흐름을 가진 문장화에 보다 적합하게 계속적인 합의를 보장할 수 있다. 본 논문은 응용이 문법적으로 올바르지 않은 문장 입력에 대해 해석을 수행해 보고, 이를 사용자 입력에 대해 개선한 원문을 보급할 수 있는 방법을 제시한다.

2. 본론

편집 환경에서 입력되는 문장은 사용자에게 문법적으로 올바르지 않은 문장으로 보고, 이를 사용자 입력에 대해 개선한 원문을 보급할 수 있는 방법을 제시한다.

2.1 불완전한 입력에 대한 인식

하향식 형태의 서브스트링을 인식하는 방법은 기본적으로 모두 가능하다. 편집 기반의 인식은 사용자 입력에 대한 고유한 특성을 보편적으로 적용할 수 있다. 본 논문의 핵심은 사용자의 사용자 입력에 대해 개선한 원문을 보급할 수 있는 방법을 제시한다. 본 논문에서 제안한 알고리즘은 다음과 같다.

[그림 1] 불완전한 XML 입력

위의 완전하지 않은 문장에 대해 편집기의 해체 부분 파산 트리를 완성하지 않고 파산을 종료한다. 따라서, 파산을 완전화할 때, 적절한 문법 정리를 이용해 파산 트리의 부여면과 그 부분의 부분 트리를 구성하여 인식적 입력과 상호작용한 문장을 보장할 수 있는 방법이 필요하다. 본 논문에서는 파산 트리의 부속 입력에 대해 개선한 원문을 보급할 수 있는 방법을 제시한다. 제안한 편집 기반의 내용은 문법적으로 부속 입력에 대해 고유한 특성을 보유하고 개선한 입력에 대해 문법적으로 빠져있는 입력 상태를 개선하는 결과를 보장할 수 있는 방향성 지지도에 발생을 한다. 본 논문에서 제안한 알고리즘은 다음과 같다.

reduce A := $β$ 일 때

case $|β| + 1 > |stack|$ : 정상적인 LR 방향으로 파산;

case $|β| = |stack|$ : placeholder로 a에 대한 노드를 생성;

goto(A)에 대한 다음과 같은 조사;

A로 출력;

case $|β| > |stack|$ : goto(A)에 대한 다음과 같은 조사;

A로 출력;
제안된 알고리즘을 이용하여 [그림 1] 과 같은 불완전 입력에 대해 표
식이 정확적으로 끝나면 [그림 2]와 같이 불완전 노드를 포함하는 파스
트리가 구성된다.

[그림 2] 불완전 입력 인식

루트노드에서 단일노드까지의 모든 경로는 입력된 부분 문법의 가능한
해석으로 해석된 파스 스테이의 top 부분에 해당하며, 이 경로의 root에 해당하는 노드의 상태는 스테이의 top에 해당하게 된다.

2.2 불완전 입력에 대한 파스 트리 생성

파서가 누락된 것으로 인식된 삐칠을 적절히 위치에 생성시키고 파
식을 계산하기 위해서는 두가지 방법이 파스 경로를 따라 다양한
형태의 줄을 가지도록 스테이의 여러 가지 줄의 형태를 모두 표현할 수
있는 방법이 필요하다. 따라서, 파스 레이블은 [알고리즘 2]에 따라 루
트 트리에 대해 추가적인 정보를 제공할 수 있도록 확장되어야 한다.
확장된 LR 파스 레이블을 기준으로 가능한 알고리즘은 다음과 같다.

[알고리즘 2] 불완전 출력을 위한 파스 레이블 확장

입력: 문법 G와 канonical collection C=(I_1, I_2, ..., I_n)

출력: n개의 스테이를 갖는 ACTION 함수로 구성된 확장형 파스표

방법:
(1) [알고리즘 1]에 의해 파스 레이블을 구성한다.

(2) ACTION(S_s, $) = “error” 항목을 다음과 같이 수정한다.

(2.1) 간단(A→α·β)의 형태가 I에 있으며,

ACTION(I_s, $) = “reduce “A→α β”

(2.2) ACTION(S_s, $) = “error”

파스의 완성하려면 여러 가지 형태가 나올 경우 각각의 트리는 별개로 구성
하는 것이 가능하다. 이는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성하게 된다. 또한, 이
는 여러 가지 형태의 경우를 별도로 구성게
이것은 불완전한 입력에 대해 문법적으로 모호한 파싱 동작을 해결할 수 있는 방법을 제공한다.

3.2 불완전 XML 입력 문장에 대한 인식과 완성

문법 G'에 대해 [그림 1]에 나타난 불완전 XML 입력
<lastname><firstname>yong-yoon</firstname><lastname></lastname>의 파싱 과정은 [표 3]과 같다.

<table>
<thead>
<tr>
<th>STACK</th>
<th>INPUT</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>&lt;n&gt;</td>
<td>$ s 1</td>
</tr>
<tr>
<td>2 0 &lt;&gt; 1</td>
<td>&lt;d&gt;</td>
<td>$ s 7</td>
</tr>
<tr>
<td>3 0 &lt;&gt; 1 &lt;&gt; 7</td>
<td>cho</td>
<td>$ s 11</td>
</tr>
<tr>
<td>4 0 &lt;&gt; 1 &lt;&gt; 7 7 y 11</td>
<td>y y</td>
<td>$ r 7</td>
</tr>
<tr>
<td>5 0 &lt;&gt; 1 &lt;&gt; 7 7 PC</td>
<td>y y</td>
<td>$ goto 12</td>
</tr>
<tr>
<td>6 0 &lt;&gt; 1 &lt;&gt; 7 7 PC 12</td>
<td>y y</td>
<td>$ s 14</td>
</tr>
<tr>
<td>7 0 &lt;&gt; 1 &lt;&gt; 7 7 PC &lt;&gt; 14</td>
<td>y y y</td>
<td>$ r 5</td>
</tr>
<tr>
<td>8 0 &lt;&gt; 1 F</td>
<td>y y</td>
<td>$ goto 5</td>
</tr>
<tr>
<td>9 0 &lt;&gt; 1 F 5</td>
<td>y y</td>
<td>$ s 6</td>
</tr>
<tr>
<td>10 0 &lt;&gt; 1 F 5 &lt;&gt; 6</td>
<td>y y y</td>
<td>$ s 11</td>
</tr>
<tr>
<td>11 0 &lt;&gt; 1 F 5 &lt;&gt; 6 &lt;&gt; cho 11</td>
<td>y y</td>
<td>$ r 7</td>
</tr>
<tr>
<td>12 0 &lt;&gt; 1 F 5 &lt;&gt; 6 PC</td>
<td>y y</td>
<td>$ goto 10</td>
</tr>
<tr>
<td>13 0 &lt;&gt; 1 F 5 &lt;&gt; 6 PC 10</td>
<td>y y</td>
<td>$ s 13</td>
</tr>
<tr>
<td>14 0 &lt;&gt; 1 F 5 &lt;&gt; 6 PC 10 &lt;&gt;</td>
<td>y &lt;&gt;</td>
<td>$ r 6</td>
</tr>
<tr>
<td>15 0 &lt;&gt; 1 F 5 L &lt;&gt;</td>
<td>&lt;&gt; $</td>
<td>goto 9</td>
</tr>
<tr>
<td>16 0 &lt;&gt; 1 F 5 L 9</td>
<td>&lt;&gt; $</td>
<td>$ r 4</td>
</tr>
<tr>
<td>17 0 &lt;&gt; 1 NE 3</td>
<td>&lt;&gt; $</td>
<td>goto 3</td>
</tr>
<tr>
<td>18 0 &lt;&gt; 1 NE 3 &lt;&gt;</td>
<td>&lt;&gt; $</td>
<td>$ s 8</td>
</tr>
<tr>
<td>19 0 &lt;&gt; 1 NE 3 &lt;&gt; &lt;&gt;</td>
<td>&lt;&gt; $</td>
<td>$ r 2</td>
</tr>
<tr>
<td>20 0 N S</td>
<td>$</td>
<td>goto 2</td>
</tr>
<tr>
<td>21 0 S N</td>
<td>$</td>
<td>accept</td>
</tr>
</tbody>
</table>

[표 3] 불완전 XML 입력에 대한 파싱 과정

파서는 초기 상태를 결정하기 위해서 [표 2]의 파싱 테이블을 참조한다. 파서는 파싱표에 첫 번째 입력인 '<firstname>'에 해당하는 값에서 모든 시프트 동작을 찾는다. 촉각 동작은 이전에 입력에 적응되는 것들이므로 여기서는 시프트 동작만을 선택하여 그 상태공간을 초기 상태로 한다.


따라서 부족한 기호인 '<lastname>'에 해당하는 기호노드를 생성하고 촉각을 수행한다. 촉각 후 상태는 goto 표를 참조하여 '<lastname>'에 대한 전이 상태를 결정한다.

[그림 4] 부수 트리의 노드 생성

실험에서 보고 있는 예에서는 4와의 두 가지 shift 상태가 결정된다. 4와 9중 유효한 파스 트리를 선택하기 위해 두 개의 파스 트리를 모두 구해야 한다. 따라서, 이 상태들에 대해 각각 파싱을 수행하기 위해 스택을 두 가지 상태로 분리한 후 계속된 파싱 과정에서 유효하지 않은 파스 트리를 9로, 유효한 파스 트리를 4번의 shift 동작을 갖는 파스 트리를 선택한다. 파싱 결과는 [그림 5]과 같다.

[그림 5] 파스 트리 인식과 완성

4. 결론

본 논문은 사용자가 프로그램을 입력하는 과정에서 불완전한 입력을 입력할 경우 발생할 수 있는 파싱 오류를 개선하기 위함 불완전한 입력을 인식하고 완성하는 파싱 방법을 제안하고 실험하였다. 제안된 파싱 방법은 부족한 실점상을 생성할 수 있으며, 그 실점을 푸리므로 하는 부수 트리를 파싱하기 위해 부수 정보를 갖는 확장 파싱 테이블을 이용한다. 따라서, 사용자는 프로그램 편집기에 있어서 불완전한 입력에 대해 문법적으로 올바른 파싱이 된 참고 문헌


[8] Shenning G., "How to build LR parsers which accept incomplete