효율적 Embedding을 위한 SMIL Processor의 설계 및 개발

장 동옥*, 강 미연, 정 원호, 이 은철+, 김 도완+, 김 종대+, 김 윤수+
덕성여자대학교 전산학과, 삼성전자 멀티미디어 연구소+

Design and Development of SMIL Processor for efficient Embedding

D.-O. Jang, M.-Y. Kang, W.-H. Chung, E.-C. Kim, D. Kim, J. D. Kim, & Y. S. Kim,
Dept. of Computer Science, Duksung Women's University, SEC MM Research Center

요 약

XML 언어로 설계된 SMIL(Synchronized Multimedia Integration Language)은 멀티미디어 객체들의 순차적 혹은 병렬적 동기화를 효율적으로 할 수 있는 마크업 언어로써, web을 이용한 원격 강의나 홍보 등을 더욱 생생하고 dynamic하게 보여줄 수 있어, 그 사용이 확대될 전망이다. 본 논문에서는 각종 웹 단말기에 손쉽게 embedding될 수 있는 SMIL 프로세서에 대한 설계가 제안된다. 웹 응용을 위해, 속도의 개선과 시스템 독립적인 function들로 구성되는 parser와 응용에 적합한 API의 설계에 주안점을 두었으며, 추후 XML parser function들과 API 설계를 위해 가능한 적은 수정을 통하여 재사용이 가능하도록 하는데 또한 주안점을 두고 있다.

1. 서 론

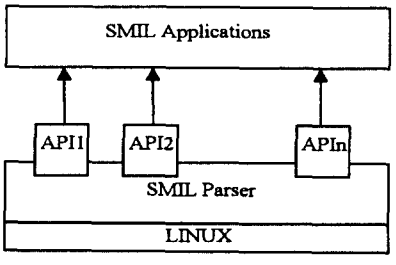
차세대 web 언어로 각광을 받고있는 XML언어[1][2][3]로 설계된 SMIL(Synchronized Multimedia Integratin Language)은 web 상에서 오디오, 비디오, 텍스트, 사운드 등의 멀티미디어 객체들을 동기화시켜 표현하는 것을 목적으로 W3C에서 표준화시킨 마크업 언어이다[4]. 독립적인 각 미디어 객체들을 순차적 혹은 병렬적으로 스케쥴링할 수 있으며, layout까지 지정할 수 있어, web을 통한 원격강의, 홍보 등과 같은 응용에서 SMIL을 이용하면 더욱 생생하고 dynamic한 효과를 기대할 수 있다. 이미 RealPlayer에서는 SMIL을 지원하기 시작하고 있으며, MIE5.0 부터는 XML을 지원하므로 SMIL 또한 지원된다. 본 논문에서는 SMIL 전용 웹 단말 하드웨어에 효율적으로 embedding할 수 있는 SMIL parser function들과 각종 SMIL 응용에 효율적인 API를 가지는 SMIL processor에 관한 설계가 제안된다. SMIL processor 설계의 기본 목적은 복잡하고 난해하지 않도록 하여야 한다는 것이다. 그것은 소프트웨어 product의 기술적 이전 측면에서 매우 중요한 부분이라 할 수 있으며, 지속적인 시스템의 upgrade를 위해서도 중요한 측면이다. 이를 바탕으로 본 연구에서 중요하게 고려한 사항들은 다음과 같다.

1) 사용이 용이하면서도 각종 application이 필요로 하는 효율적인 정보의 제공,
2) 각종 function의 효율화로 API 실행의 고속화, 그리고
3) 가능한 device-dependent한 부분을 줄이며, 피할 수 없는 경우 application에서 볼 수 없도록 wrapping을 하여 추상화시킨다.

본 연구에서는, 이상과 같은 사항들을 고려하여 가장 cost-effective한 data structure를 설계하려고 노력하였으며, 그 개략도가 [그림-1]이다.

2. SMIL Parse Table의 구성

API의 고속화 및 function의 단순화, 자료의 원시상태 저장을 위하여 SMIL 문서 전체를 메모리 상에 상주시킨 후 처리하는 것으로 하였다. 이는 SMIL 문서의 구성을 이루고 있는 대부분의 정보가 attribute value들로 구성되고 있으며, 이 값들은 결국 메모리 어딘가에 저장되어 있어야 하기 때문이며, 향후 SMIL 문서의 access를 용이하게 하여 DOM[5] 설계시 보다 나은 access 환경을 제공할 수 있기 때문이다. SMIL 문서는 단위 block들의 doubly-linked list로써 구현되며, 각 page의 크기는 1 Kbytes이다. 각종 device 의존적인 함수는 wrapping을 하였으며, 하나의 모듈로 구성하여, 수정 및 확장을 용이하게 하였다.



[그림-1] SMIL Processor의 개략도

SMIL parse table은 SMIL 문서에서 사용된 각종 element에 관한 정보를 저장하고 있으며, 메모리에 상주하고 있는 SMIL 문서를 scan하면서 각종 wellformedness 및 validity 여부를 조사하면서 구축된다. 트리구조

인 경우 발생하는 탐색의 오버헤드를 줄일 수 있다는 장점이 있으나, 고정된 크기를 가진다는 단점이 있다. Parse table에 저장되는 정보는 SMIL 문서를 구성하는 element 및 attribute에 관한 정보를 지시하는 주소들이며, 본 연구의 핵심이 되는 자료구조인데, [그림-2]와 같은 entry들의 list로 구성된다. 모든 SMIL 정보는 이 table로부터 얻어질 수 있다.

```
typedef   enum {START, EMPTY}        elementType ;
struct smilParseTable {
  unsigned number ;              /* 저장되는 element의 순서 */
  char *eStart                   /* element 시작 주소 */
  unsigned eLen                  /* element size */
  unsigned eId ;                 /* 현재 element의 고유번호 */
  unsigned eLevel ;              /* 현재 element의 level */
  elementType eType;              /* 현재 element의 type */
  smilParseTable  *eParent ;    /* 현 element의 parent */
  smilParseTable  *eChildren ;  /* 현 element의 children */
  smilParseTable  *eSiebling    /* 현 element의 siebling */
  char *attrStart ;          /* 현 element의 attribute start 주소*/
  char *attrEnd ;            /* 현 element의 attribute end 주소 */
  char *attrBlock ;            /* 현 element의 attr block 주소 */
};
```

[그림-2] Parser table 각 entry의 구조

Element의 순서, hierarchy 검증을 위한 level, 각 element의 고유번호인 eId, element type, element의 parent, children, 그리고 siebling 등의 모든 element 정보가 수록되는 곳이다. 또한 중요한 정보로 element의 attribute 정보가 수록된 block의 주소 정보가 수록되어 attribute 값이 저장된다. attrStart는 element의 attribute가 시작하는 주소로 보통 element name 다음의 최초 문자의 주소가 될 것이며, attrEnd는 attribute가 끝나는 주소로 지정된다. Element name의 경우 본 논문에서는 SMIL 언어에서 사용 가능한 모든 19개의 element name들 각각에 0부터 31 사이의 고유번호인 eId를 부여하여 짧은 시간에 element를 탐색할 수 있도록 하였으며, 이를 위해 string 문자의 값을 합하여 번호를 생성하는 hash function을 사용하였다. 이 eId만 있으면, element name string들을 eId 순으로 저장하고 있는 테이블인 **elementNameSpace**로부터 해당 element string을 손쉽게 얻을 수 있다. 이 hash function은 parser 함수들의 효율적인 설계를 위해 근간이 되는 중요한 함수로 3장에서 자세히 기술된다.

ParseTable의 구축을 위한, SMIL 문서의 scanning은 메모리 상의 DLL을 scanning하는 과정이며, 다음과 같은 문자들을 event로 하는 finite automata에 의해 이루어지며, wellformedness가 검증된다.

AutomataCharSet =
{< / > " ' ! - isblank isalphanum # & ; = notavail},

3. Element Name Space

SMIL 문서에 사용된 element name들이 적정하게 사용되었는가를 확인하기 위해서는 string compare가 필수적이다. 그러나, 매 element 마다 string compare를 한다는 것은 시간적인 측면에서 오버헤드가 크므로, 본 연구에서는 특별한 hash function을 사용하여 element name 및 hierarchy 검증을 하고 있다.

3.1 Element Name Space의 구성

SMIL 문서에서 사용되고 있는 element는 총 19개이다. 그러나, name 검증을 위하여, name space의 크기를 32개로 정하여 hash function을 사용하여 name space의 각 element마다 0 - 31 사이의 고유번호(eId)를 부여하고 있다. 그리고, 존재하지 않는 eId인 경우 NULL 문자를 사용하고 있다. 고유번호인 eId는 **StringSumModulo(char *, unsigned)**로 명명된 함수에 의해 구해지며, input string의 각 문자들의 코드 값을 더하여 modulo한 값을 구하는 함수이며, conflict가 생기는 경우, 해당 element들을 비교하여 범위 내의 사용하지 않는 값을 할당한다. 이러한 방법은 사전(dictionary)탐색 등에 유효하게 사용될 수 있으리라 판단된다. Element의 고유번호 발생을 위한 알고리즘은 다음과 같다.

Step-1) Argument string의 코드 값 sum에 대한 modulo-32를 구한다.
Step-2 Conflict가 발생하면, conflict한 element들을 비교하여 사용하고 있지 않는 번호 중에서 하나를 선택하여 eId에 할당한다.
Step-3 Return eId

3.2 Element Name 검증

이를 사용하면 [그림-3]과 같은 과정을 통해, 간단히 element name을 검증할 수 있다. 여기서 element는 input name string이며 elementNameSpace는 SMIL element name들의 list이다.

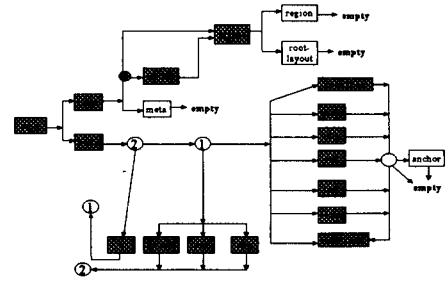
```
unsigned ElementNameValidation(element) {
   eId = SumStringModulo(element, ModuloEvalue);
   if(!StringCompare(element, elementNameSpace[eId]);
      then return TRUE;
      else return FALSE;
```

[그림-3] SMIL element name 검증 알고리즘

4. Element Name Hierarchy 검증

SMIL parser는 문서에 사용된 element들이 상호 hierarchy 관계를 만족하고 있는가를 검증하여야 한다. SMIL 언어에서 사용하는 19개의 element들간의 hierarchy는 [그림-4]에서 보여준 바와 같다. 몇 개의 element들은 상이한 특성을 가지고 있다 - 예를 들면 switch element인 경우 head에서 사용될 경우와 body에서 사용될 경우 다른 hierarchy를 갖는다 -. 이러한 element들간의 hierarchy 검증은 SMIL DTD에서 정의된 element hierarchy로부터 파악될 수 있으며, DTD로부터 구성될 수 있다.

그러므로, SMIL parser는 [그림-4]에 기반을 두고 element들간의 hierarchy를 검사하는 것이 일반적이라 할 수 있으나, 본 연구에서는 검증의 효율을 위해 Bit-AND 방식을 통해 검사하는 방법을 채택하였다. 이는 특정 마크업 언어인 SMIL의 경우 용이하게 통용될 수 있으나, 일반적인 XML 언어 parser인 경우에는 매우 복잡할 수 있다.



[그림-4] SMIL element hierarchy diagram

먼저 **32-bit integer**를 사용하여 각 bit 번호를 각 element 고유번호에

대응시켜 해당 element의 flag로 사용한다. 상기 flag data를 사용하면, 각 element가 가질 수 있는 children element들을 나타내는 정보는 간단히 children flag들의 Bit-ORing을 통해 얻어질 수 있다. 그리하여 SMIL언어의 각 element가 가질 수 있는 children에 대한 정보 표현의 예를 들면 다음과 같다.

```
#define smilChildren                (headXbit | bodyXbit)
#define headChildren      (layoutXbit | switchXbit | metaXbit)
```

이는 smil element는 head와 body element를 children으로 가질 수 있다는 것을 의미하며, head element는 layout, switch, 그리고 meta를 element로 가질 수 있다는 것을 의미하고 있다. 여기서 headXbit, bodyXbit 등은 32-bit integer에서 해당 element 고유번호에 해당하는 bit가 set된 정수 값을 의미하고 있다.

Children element의 확인 과정은 현재 element string으로부터 **SumStringModulo** 함수를 통해 구한 값을 index로 **elementXbitList**에서 해당 flag를 얻어 바로 전의 element string으로부터 구한 값을 index로 하여 ChildrenElement에서 그 element가 children으로 가질 수 있는 element들을 얻어 단순히 Bit-AND를 하여 0이 아니면 child로 인정된다.

이때, switch element가 head 부분에 있느냐 혹은 body 부분에 있느냐에 따라 headChildrenElement 혹은 bodyChildrenElement가 선택되며, 이는 switch element일 경우에만 해당된다. 그러므로 바로 전의 element가 switch element가 아닌 경우에는 어느 것을 사용해도 무방하다.

5. Element Attributes

SMIL의 각 element가 가지는 attribute의 총합은 38가지이다. Parser함수는 attribute들의 정확성을 검증해야 하는데, 이는 크게 attribute name 검증과 attribute 포함 검증으로 나눌 수 있다. 이를 위해, 먼저 element에서와 같이 attribute name space를 구성하고, 그것을 바탕으로 attribute 검증 방법을 제시한다. Attribute block은 각 element마다 하나씩 지정된다. 비록 각 element마다 가지는 attribute의 수가 다르나, 그 element가 SMIL 문서상에서 어떤 attribute를 가지느냐와 attribute value의 신속한 access를 위하여 각 element 마다 attribute 고유번호의 범위에 해당하는 64개의 pointer를 갖도록 설계한다.

5.1 Attribute Name Space의 구성

Element name에서와 마찬가지로, SMIL parser는 SMIL 문서의 각 element에서 사용된 각 attribute의 name이 올바른가와, 해당 element가 가질 수 있는 attribute인가를 검사하여야 한다. 이를 위해서 modulo-64 연산을 사용하여 각 attribute에 고유한 number를 할당하게된다. 이를 바탕으로 SMIL 언어에서 사용되는 attribute들의 name space를 설정하면 다음과 같다.

```
#define maxNumOfAttr                    64
char      *attrNameSpace[maxNumOfAttr] =
{         (01) "system-caption",
          (02) "alt",
            .....
          (60) "z-index"
          (61-63) "\0", "\0", "\0"          };
```

5.2 Attribute Name 검증

주어진 attribute string에 대한 modulo-64 연산을 하여 얻어진 고유번호를 가지면 attrNameSpace로부터 attribute name이 올바른가의 검증은 쉽게 이루어질 수 있다. 이는 앞에서 element name 검증과 유사한 Bit-AND과정을 통해 간단하게 이루어질 수 있다.

5.3 Attribute Name 포함관계 검증

특정 element에서 사용된 attribute들이 그 element가 가질 수 있는 attribute인가를 검증하기 위해 본 연구에서는 검증 시간을 줄이기 위하여, 2차원 bit array인 arrtCheckTable을 사용하여 검사하는 방법을 고안하였고 Array를 구성하고 있는 각 수는 해당 attribute를 가지고 있는 모든 element들을 bit-by-bit로 표현하고 있으며, 각 bit에 해당하는 element는 elementXbitList에 나타나 있다. 예를 들면,

#define attrType (layoutXbit | mediaObjectXbit)

이는 layout과 mediaObject element는 Type attribute를 가지고 있다는 것을 의미하고 있다. 상기 자료구조와 element hierarchy를 검사하기 위해 정의된 XbitFlagList를 이용하면 간단한 Bit-AND 동작을 통하여 주어진 attribute가 현재 element가 가질 수 있는 것인지 쉽게 확인할 수 있다.

5.4 Attribute Block의 구성

SMIL 언어에서 사용되고 있는 attribute들의 총 개수는 38개이다. 이들은 StringSumModulo 함수에 의해 0 - 63 사이의 고유번호를 지정 받게 된다. 이 attribute block은 64개의 pointer들의 집합으로 SMIL 문서에서 사용된 각 element마다 하나의 attribute block을 지정 받는다. 각 element에 대한 attribute block의 생성과 시작 주소는 ParseTable 구축 시 할당되며, 그 시작 주소가 테이블의 attribute block field에 잡혀진다. SMIL 문서에서 attribute를 가지는 모든 element는 64 개의 (char *) array로 구성되는 attribute block을 가지며, 그 block 내의 각 field는 attribute 처리 과정에서 ParseTable에 저장된 attribute start와 attribute end address를 기반으로 attribute 내용을 scanning하면서 구하여 채워진다.

6. SMIL Start/End Tags Matching

SMIL문서의 tag들은 nested 형태의 계층구조를 갖기 때문에, 보다 효율적인 검사를 위하여 스택을 사용하고 있다. start tag가 나올 때마다 스택에 계속 push를 한다. 그리고 end tag(혹은 empty element tag)가 발견되면, 먼저 스택의 top값을 pop하여 발견된 end tag와 일치하는지를 검사하여 tag의 매칭을 검증하고 있다.

7. 결론 및 향후과제

향후 응용이 증가할 SMIL 언어를 위한 프로세서의 설계가 효율적 embedding과 속도개선 측면을 강조하여 제안되었으며, 10여개의 유용한 API가 설계되었다. API 및 parser에 대한 상세 설계는 지면 관계 상 간략히 할 수 밖에 없었다. 현재 좀더 효율적인 해쉬 함수와 Bit-AND방식을 통한 계층구조 검사에 대해 계속 연구 중에 있으며, 향후계획으로는 XML 프로세서 및 API 설계로 확장될 것이다. SMIL로 인하여 앞으로 web상에서의 화려한 presentation을 기대해도 좋으리라 생각한다.

참고문헌

[1] http://www.w3.org/TR/1998/REC-xml-19980210

[2] http://www.w3.org/TR/1999/REC-xml-names-19990114

[3] Ian S. Graham, Liam Quin "XML Specification Guide" Wiley Computer Publishing, 1999

[4] http://www.w3.org/TR/REC-smil

[5] http://www.w3.org/TR/1999/REC-DOM-Level-1-19981001