SGML기반의 텍스트 문서 브라우저의 구현

장명길, 이혜란, 주종철, 박동인
시스템공학연구소

Implementation of A Browser for SGML-based Text Documents

Myung-Gil Jang, Heran Lee, Zongcheol Zhoo, Dong-In Park
Systems Engineering Research Institute

요 약

기존의 문서 시스템은 문서의 논리적인 정보와 문서의 외양에 관련된 처리정보들이 확연히 구분되지 않음으로써 서로 다른 시스템간의 문서 교환시 정보의 손실을 가져올 뿐 아니라, 문서의 저장방식에서도 순차 화일 구조를 갖기 때문에 문서의 논리적 요소에 대한 대화식 검색이 불가능하다.

이러한 단점을 극복하고자 문서의 논리적 구조 및 내용을 중심으로 작성 가능한 표준 메타 언어인 SGML이 제정되었으며, 본 연구에서는 SGML 문서를 인식하고 해석하기 위한 SGML파서와 문서의 논리적 구조를 반영하는 저장구조 및 이를 이용한 브라우저를 구현하였다.

1. 서론

최근 문서편집, 탁상출판(Desktop publishing), 하이퍼미디아 등의 발전에도 불구하고 문서 처리(document processing)가 다른 데이타 처리분야에 비교할 때 낙후된 이유는 다음과 같은 문제들을 안고 있기 때문인 것으로 보인다. [6]

첫째, 문서의 '내용'과 '외양'에 관련된 정보의 혼재, 즉 문서의 논리적인 구조와 레이 아웃 정보 등이 충분히 구분되지 않는 점이다. 둘째, 문서의 저장이 순차화일로 구성되어 있어 문서 내용에 대한 직접적인 접근이 이루어지지 않는다.

또한 이같은 문서 형태는 문서의 저장구조 및 관련 정보들이 처리장치나 응용시스템에 의존적이기 때문에 다른 시스템에 맞는 형태로 변환할 경우 정보의 손실을 가져오며, 저장 문서 개개의 요소에 대한 대화식 편집이나 검색이 불가능하다.

이러한 문제들을 극복하고자 시스템에 의존적인 문서처리와 관련된 정보와 문서의 논리적 정보를 구분하고, 이기종간의 문서교환이 가능하도록 한 표준화된 언어가 ISO 8879에 의해 제정된 SGML(Standard Generalized Markup Language)이다.

SGML은 문서의 논리적인 구조를 기술하는 메타언어로서 문서 작성자가 문서의 외양과 상관없이 문서 자체의 내용 및 논리적 구조에 집중하여 작성할 수 있게 하며, 문서외양에 관련된 처리는 별도의 응용시스템(포메터 등)을 필요로 한다.

SGML 문서는 문서의 논리적 구조를 포함하며, 문서요소간의 링크정보 등도 포함할 수 있으므로 기존의 순차화일 구조가 아닌 새로운 저장구조를 가져야 문서구조를 반영하는 각종 검색을 충분히 지원할 수 있다.

본 연구에서는 SGML문서를 인식하고 검증할 뿐 아니라 문서를 논리적 구조를 갖는 중간 형태로 변환시켜 주기 위한 SGML파서와 파싱 결과를 검색 가능한 형태로 저장하고, 이를 문서의 논리적 구조에 따라 브라우징할 수 있는 브라우저를 구현하였다.

2. SGML 관련 연구

2.1 SGML 문서

SGML문서는 다음과 같이 3개의 부분으로 구성된다. [8]

(1) SGML 선언부(SGML Declaration)

DTD와 문서 텍스트에서 사용할 문자들(Characters)과 문서내에 사용될 특별한 SGML기능들(features)을 정의한다.

(2) 문서 형태 정의부(DTD: Document Type Definition)

문서 데이타를 작성할 때의 규칙과 해당 문서의 구조를 정의한다.

(3) 문서 데이타(Document Instance)

참조하는 DTD에 대한 선언(<!DOCTYPE...>) 및 문서의 실제 텍스트에 해당한다. DTD에 정의된 규칙에 따라 문서 내용에 마크업(markup)된다.

DTD와 문서데이타는 SGML선언부에서 정의된 문자들만을 사용하며, 위의 세 부분은 각각 별개의 화일로 저장될 수 있다. 보통 DTD는 문서의 종류에 따라 달라지고, 같은 유형의 문서인 경우 동일한 DTD를 갖는 문서 데이타가 여러개 있을 수 있다.

DTD는 특정 형태의 문서 생성규칙으로서 크게 엘리먼트(element), 속성(attribute), 엔티티(entity) 선언부를 두고 있다.

엘리먼트 선언부는 문서의 논리적인 구성요소를 나타내는 구별자(GI: Generic Identifier)와 그것의 Content Model을 정의한다. Content Model은 문서의 계층구조를 나타내기 위하여 각 엘리먼트의 구성요소를 기술한 것이며, 엘리먼트 이름인 GI는 문서데이타에서 마크업으로 사용된다. 예를 들면 엘리먼트로 To, From, Body, Close로 구성된 memo라는 문서의 최상위(root) 엘리먼트에 대한 선언은 다음과 같다.

```
<!ELEMENT memo --((to & from), body, close?)>
```

속성 선언부는 엘리먼트의 특성을 기술하기 위한 것으로 속성이름과 값에 대해 정의한다. 다음은 속성 선언부의 한 예이다.

```
<!ATTLIST memo   status  (confiden|public) public >
```

엔티티 선언부는 매우 긴 문자열이나 외부화일 또는 특정 시스템에 의존하는 오브젝트(object) 등을 참조하는데 사용할 이름을 정의한다. 다음은 엔티티 선언의 한 예이다.

```
<!ENTITY  SERI  'Systems  Engineering  Research  Institute>
```

2.2 링크정보의 정의

SGML문서에서 어떤 엘리먼트는 Content Model이 없이 오로지 다른 엘리먼트를 참조하기 위한 목적으로 선언된다. 보통 일반 문서에서 참고문헌(bibliography)이나 각주(Footnote)에 대한 참조 표시를 실제 내용은 없이 식별자만을 이용하는 것과 같은 이치이다. 다음은 참조용 엘리먼트와 링크정보를 기술하기 위한 속성 선언부의 한 예이다.

```
<!ELEMENT bibref   - 0    EMPTY >
<!ATTLIST bibref   IDREF  #REQUIRED >
```

위의 링크정보를 갖는 엘리먼트가 참조하는 실제 내용(referece text)이 있는 엘리먼트는 식별자(ID)의 속성정보를 갖도록 다음과 같이 선언된다.

```
<!ELEMENT bib    - 0      (#PCDATA) >
<!ATTLIST bib    id       ID        #IMPLIED >
```

다음은 위의 엘리먼트를 사용하여 작성한 문서 데이타의 한 예이다.

```
<p>
.....
조사는 크게 3 형태로 분류한다. <bibref refid = "SERI-1">
</p>
.....
<bib id = "SERI-1"> 제 1 회 기계번역 WORKSHOP 발표 논문집,
시스템공학연구소, 1988</bib>
```

2.3 SGML관련 연구

SGML을 이용한 시스템은 응용 시스템에 따라 제공하는 기능이 달라질 수 있으나 대략 다음과 같은 기능들을 만족시키는 시스템을 개발하고 있다.

(1) SGML문서 작성기(SGML Authoring Tool)

사용자가 DTD의 작성규칙에 따라 문서를 작성해야 하는 부담을 덜고 문서 내용에만 관심을 갖고 작성할 수 있도록 편리한 저작환경을 제공하며, SGML문서 데이타의 검증 및 문서교환을 위한 Import/Export기능 등을 지원한다. [7]

(2) 문서 변환기 (Document Translator)

기존에 다른 언어로 작성된 문서들을 표준화된 문서형태인 SGML문서로 변환하기 위한 시도로서, 예를 들면 Latex나 troff 등으로 작성된 문서를 SGML문서로 상호 변환시킬 수 있는 기능을 지원한다. [9]

(3) SGML문서 포맷터(WYSIWYG text formater)

문서의 외양과 관련된 처리로서 동일한 SGML문서는 포맷 정보에 따라서 다양한 외양을 지닐 수 있다.

(4) 마크업 및 링크정보에 근거한 문서 검색 및 브라우징

SGML문서의 논리적 구조 및 문서 요소간 링크정보를 이용하여 다양한 검색이 가능하며, 문서구조를 따라 브라

우징할 수 있는 기능을 지원한다.

(5) DBMS를 이용한 문서저장

일반적으로 정형화된 데이타와 같이 문서와 관련된 정보들을 데이타베이스에 저장하려는 시도로서 DTD를 개념 스키마로, 문서의 각종 포맷정보는 외부 스키마로 저장함으로써 문서정보의 공유 및 관리를 향상시키고자 한다. [1, 2]

(5) HyTime, HyQ를 이용한 검색엔진

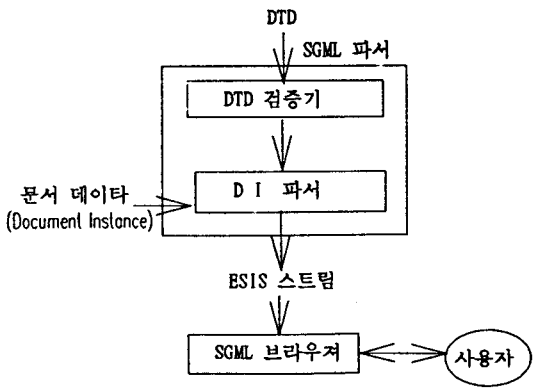
텍스트이외에 이미지나 소리 등을 포함하는 문서들을 대상으로 상호 참조 및 관련성을 기술하기 위한 SGML의 확장기능인 HyTime과 이에 기반한 문서를 검색하기 위한 표준화된 검색언어인 HyQ를 지원한다. 이는 검색과 관련된 정보까지도 응용 시스템에 독립적인 형태를 취함으로써 교환가능한 문서정보의 극대화를 꾀하고자 한다. [5, 10]

3.SGML문서 브라우저의 구현

3.1 개요

SGML 문서 브라우저 시스템은 크게 SGML 파서와 SGML 브라우저, 두 부분으로 구성된다. SGML 파서는 DTD와 문서 데이타(DI:Document Instance)를 입력으로 받아들여 그 문서들을 검증하고, 그 결과로 ESIS(Element Structure Information Set)스트림을 출력한다. SGML 브라우저는 ESIS 스트림을 분석하여 일반트리 문서구조를 생성하고 이를 토대로 브라우징를 위한 검색 명령어들을 수행한다.

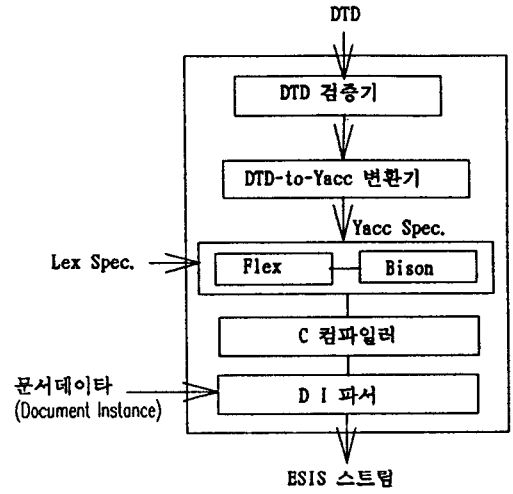
본 시스템은 UNIX 4.3상에서 C 언어로 구현하였으며 시스템의 전체 구성도는 (그림 1)과 같다.



(그림 1) SGML 문서 브라우저 구성도

3.2 SGML 파서

SGML 파서는 문서 데이타의 마크업들을 인식하여 그 문서의 논리적 구조를 밝히는 프로그램이다. 파서는 DTD와 DI를 각각 검증하고 파싱하기 위해 (그림2)와 같이 구성되어 있다.



(그림 2) SGML 파서 구성도

DTD 검증기(Validator)는 입력된 DTD가 SGML문법에 맞게 작성되었는가를 검사하며, DI 파서는 입력된 문서 데이타가 앞서 검증한 DTD에 맞게 작성되었는가를 검증하고 문서구조를 나타낼 수 있는 완전한 SGML 문서형태로 출력한다. DTD와 문서 데이타에 사용된 모든 토큰을 인식하고 해석하기 위한 문법의 구현은 Flex와 Bison 툴을 이용하였다.

DI 파서를 위한 문법은 입력된 DTD마다 달라지므로 DTD-to-Yacc 변환기(Converter)가 검증된 DTD로부터 Bison의 Yacc Spec.을 생성한다. 즉, DTD의 엘리먼트 선언부에서 문서의 논리적 구조를 표현하는 Content Model 부분에 상응하는 Yacc Rule로 바꾼다.

DTD-to-Yacc 변환기는 크게 2 단계(2 PASS)로 이루어진다. 첫 단계에서는 엔티티 매니져(Entity Manager)가 DTD의 엔티티 선언부를 분석하면서 엔티티 테이블을 구축한 후, 각 엔티티 참조부(Entity Reference)에 대한 대치를 수행한다. 두번째 단계에서는 입력 DTD의 엘리먼트 선언부의 Model Group을 분석하면서 이를 Yacc Rule로 변환하는 일을 수행한다.

본 파서는 다음과 같은 특징을 지닌다.
첫째, Flex와 Bison을 이용한다. Flex는 한글 문자와

같은 8 bit 문자를 입력 패턴으로 받아들일 수 있도록 -8 옵션을 두고 있어서, 한글 SGML 문서의 처리를 가능하게 한다. 또한, Bison은 Rule의 수에 제한이 없으므로 Yacc 보다도 복잡한 입력을 처리할 수 있게 한다.

둘째, 파서의 출력 결과로서 ESIS(Element Structure Information Set) 스트림의 형태를 지정할 수 있다. 일반적으로 ESIS 구조는 이를 응용시스템에서 이용하기 위해 문서의 논리적인 구조를 잘 반영하는 형태를 지니고 있다. 본 파서는 사용자의 선택에 따라 다음과 같이 3 가지 형태의 ESIS 스트림을 생성한다.

(1)문서 데이타에서 생략된 모든 태그를 복원한 형태(fully tagged ESIS)로서 ASP(Amsterdam SGML Parser)의 출력형태와 같으며, 일반적으로 문서 교환에 많이 쓰이는 구조이다.

(2) 본 시스템에서 정의한 형태(Default ESIS)로서 문서의 구조 정보를 잘 나타내면서 가독성(readability)이 좋은 구조이다.

(3) 본 시스템의 SGML 브라우저에서 사용하기 위한 형태로서 엘리먼트가 자신의 하위 엘리먼트를 가지는 지의 여부에 따라 다음과 같은 구조를 갖는다.

● 하위 엘리먼트를 가지는 엘리먼트:

```
(엘리먼트의 GI &Number
 하위 엘리먼트의 GI
 )
```

● 하위 엘리먼트를 가지지 않는 최하위 엘리먼트:

```
(최하위 엘리먼트의 GI &Number #Number)
```

'&Number'는 해당 엘리먼트의 속성(Attribute)를 참조하기 위한 인덱스로서 선택적(optional)이며, '#Number'는 최하위 엘리먼트의 내용인 텍스트 데이타(PCDATA)를 참조하기 위한 인덱스이다. (그림 3)은 메모 타입의 문서 데이타와 이의 파싱 결과인 브라우저용 ESIS 스트림을 나타낸다.

메모 문서 데이타	브라우저용 ESIS스트림
<!doctype Memo> <Memo status ='confiden'> <To>시스템공학연구소 <From>홍길동 <Body> <P id='first'> 저는 <q> 홍길동</q>이라고 합니다 . <p id='last'>찾아뵙겠습 니다. <close>안녕히 계세요. </Memo>	DOCTYPE : Memo (Memo &0 (To #0) (From #1) (Body (P &1 #2 (Q #3) #4) (P &2 #5)) (Close #6))

(그림 3) 브라우저를 위한 ESIS 스트림의 예

본 파서는 SGML선언부에 대한 분석은 수행하지 않으며, (그림 4)은 SGML 8879에서 명시하는 기능들 중에서 본 파서가 지원하는 범위를 나타낸다.

SGML 기능(Features)		파서의 지원여부
SGML Declaration		X
기본기능	Element Declaration	O
	Attribute Declaration	O
	Entity Declaration	O
	Notation	X
	Content Model Exceptions	X
부가기능	Short Reference	X
	Shorttag	O
	Omittag	O

(그림 4) 파서의 SGML 지원 범위

3.3 SGML 브라우저

SGML 브라우저는 ESIS스트림을 검색에 알맞는 트리구조로 저장하는 전처리기와 문서 정보를 검색하는 브라우저, 두 부분으로 구성된다.

3.4.1 전처리기

ESIS 스트림으로부터 일반트리 저장구조를 생성하는 과정은 다음과 같다. 먼저 ESIS 스트림을 분석하여 GTREE INFO라는 중간표현 Rule로 바꾸고, 이를 다시 일반트리의 구조로 생성한다. GTREE INFO Rule의 형식은 다음과 같이 정의된다.

엘리먼트 : (하위 엘리먼트 | #Number)*

이 Rule은 콜론(:)을 중심으로 왼쪽에 엘리먼트의 이름이 오고 오른쪽은 그 엘리먼트가 포함하는 하위 엘리먼트 들을 둔다. 엘리먼트가 PCDATA를 가지는 경우는 자신의 하위엘리먼트를 #Number로 표현하며 PCDATA 정보에 대한 참조 인덱스를 가진다.

예를 들어, 메모 문서 DTD의 일부 규칙이 <!ELEMENT Body - O (P*) > 이고, 그 규칙에 해당하는 ESIS 스트림의 부분이 '... (Body (P ...) (P ...)) ... '이라고 할 때, 이에 대하여 생성되는 GTREE INFO Rule은 다음과 같다.

body : P P_2

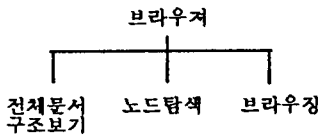
GTREE INFO Rule로부터 생성되는 일반트리의 데이타 구조는 다음과 같다.

노드 이름
노드 위치
*부모 노드
*이전 자매 노드
*다음 자매 노드
자식노드 수
*첫번째 자식노드

여기서 노드이름은 문서의 구성요소에 해당하는 엘리먼트 고유의 식별자(ID)로 사용되며 이를 통해 해당 엘리먼트와 관련된 속성 테이블과 PCDATA 테이블이 참조된다.

3.4.2 브라우저

SGML 브라우저가 제공하는 검색기능은 다음과 같이 3가지 메뉴를 통해 이루어 진다.



● 전체 문서구조 보기 :일반트리를 Pre_order로 탐색하면서 노드이름과 자식 노드의 갯수에 대한 정보를 트리 형태로 출력한다.

● 노드 탐색 : 특정 GI에 해당하는 노드를 탐색하거나 특정 문자열을 포함하는 최하위 노드를 탐색하여 그 위치정보(트리의 경로)를 출력한다.

● 브라우징 : 트리형태의 문서구조를 따라가면서(일종의 Navigation기능) 검색하는 기능으로 다음과 같은 검색 명령어들을 지원한다.

(1) H : Go to the hub node
(2) C : Show child
(3) A : Show attribute
(4) D : Show content
(5) N : Go to first child
(6) B : Go to previous sibling
(7) F : Go to next sibling
(8) P : Go to parent
(9) L : Show a link(for an anchor)
(10) S : Show the position of the current node

(1), (5)-(8), (10)은 트리 상에서 노드의 위치를 이동할 때 기본적으로 필요한 명령어들이다. 이러한 명령어들은 수행시에 현재 위치한 노드의 엘리먼트 이름과 속성정보를 제공한다. (9)는 사용자의 요구에 따라 링크정보를 갖는 엘리먼트가 참조하는 목적(Anchor) 엘리먼트로 이동할 수 있는 서브 메뉴를 제공한다.

(2)는 현재 노드의 자식 노드들에 관한 정보를 알려주며 (3)과 (4)는 각각 속성(attribute) 정보와 내용 텍스트(PCDATA)를 보기 위한 명령어이다.

4. 실험

4.1 SGML 파서

메모와 보고서 문서를 대상으로 실험한 결과, 문서 요소별 크기는 (그림 5)와 같다.

	GI	Att-list	Attri-bute/GI	Entity	PCDATA	Tree Node
memo	8	3	1	0	7	15
report	55	15	2	15	119	293

(그림 5) 실험 문서 통계

4.2 SGML 브라우저

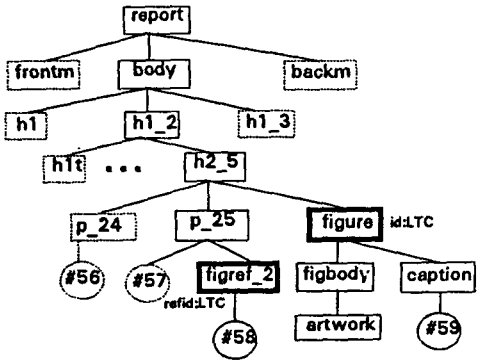
4.2.1 전체 문서구조 보기

(그림 6)은 메모문서의 전체 구조보기의 예로서 각 노드의 이름과 자식노드의 수를 보여 주며, #Number는 단말 노드로서 DTD에서 PCDATA로 정의된 엘리먼트에 해당한다.

```
Memo (4)
+---To (1)
+---+---#0 (0)
+---From (1)
+---+---#1 (0)
+---Body (2)
+---+---P (3)
+---+---+---#2 (0)
+---+---+---Q (1)
+---+---+---+---#3 (0)
+---+---+---#4 (0)
+---+---P_2 (1)
+---+---+---#5 (0)
+---Close (1)
+---+---#6 (0)
```

(그림 6) 메모문서의 전체구조 보기의 예

4.2.2 브라우징



(그림 7) 보고서 문서의 계층구조 일부

다음은 (그림 7)의 보고서 문서 일부분 중에서 링크 정보를 갖는 엘리먼트를 중심으로 브라우징을 실험한 예이다. 참조용 엘리먼트인 'figref'에 대한 내용검색은 실제로 'figure'라는 엘리먼트를 참조함으로써 이루어지므로 브라우저는 현재 위치를 참조될 목적 노드(Anchor node)인 'figure'로 이동하게 된다.

```
Browser> c
 Child Node[0]: #57
 Child Node[1]: figref_2
Browser> n
 Pseudo-Element[#57]:
```

어휘변환단계는 어휘 선택규칙과 여러가지 대역사전들로 구성되며 이들의 대략적인 관계는 구성도

```
Browser> f
 (LINK) Element : figref_2
 IDREF : LTC
Browser> s
 Current Position : figref_2 > p_25 > h2_5 > h1_2 >
body > report
Browser> 1
 Choose an Anchor:
 [0] Cancel
 [1] (ANCHOR) Element: figure
 Anchor? 1
 Element: figure
 ID : LTC
Browser> s
 Current Position : figure > h2_5 > h1_2 > body >
report
```

5. 결론

문서의 논리적인 구조를 반영할 뿐 아니라 이기종간의 문서교환을 위해서 제정된 SGML을 실제 문서 시스템에서 활용하기 위해 SGML문서를 인식할 수 있는 파서와 SGML문서만이 지니는 특성을 반영한 저장구조 및 이에 근거한 브라우저를 구현하였다.

앞으로 멀티미디어용 문서를 겨냥한 시스템이 되기 위해서는 텍스트 뿐만 아니라 이미지나 특정 시스템에 의존하는 외부 화일들을 처리하는 엔티티 지원 기능과 'Notation' 등, 부가적인 SGML 기능들도 추가되어야 한다. 또한 SGML에 기반한 문서 검색을 위해서 정형화된 기능명세 및 검색언어 등을 연구해야 하며, 엘리먼트를 오브젝트 단위로 처리할 수 있는 저장방식으로 전환될 필요가 있다.

참고 문헌

[1] Karl Aberer, Klemens Bohm, Christoph Huser, "The Prospects of Publishing Using Advanced Database Concepts", GMD-IPSI, Aug. 1993

[2] Klemens Bohm, Karl Aberer, Christoph Huser, "Extending the Scope of Document Handling: The Design of an OODBMS Application Framework for SGML Document Storage", GMD-IPSI, Aug. 1993

[3] Martin Bryan, *SGML An author's guide to the Standard Generarized Markup Language*, Addison-Wesley, 1988

[4] VODAK Design Specification Document: VML - VODAK Model Language DRAFT, Version:3.1, Aug. 93

[5] SO/IEC 10744: Information technology - Hypermedia/HyTime-based Structuring Language(HyTime), 1992

[6] Han Schouten, "SGML *CASE The Storage of Documents in Databases", SGML Users' Group Bulletin Vol4, Number1, 1989

[7] SoftQuad Inc, SoftQuad Author/Editor Version 3.0 for Motif, March, 1994.

[8] Goldfarb, C.F., *The SGML Handbook*, Oxford Univ. Press, 1990.

[9] S. A. Mamrak, et. al., "Technical Documentation for The Integrated Chameleon Architecture", Dept. of Information Science, Ohio State Univ., 1992

[10] John F. Koegel, et. al., "HyOctane: A HyTime Engine for the MMIS", ACM Multimedia '93, 1993