Incorporating Dimensions in XML and DTD*

Manolis Gergatsoulis[1], **Yannis Stavrakas**[1,2], and **Dimitris Karteris**[1]

[1] Institute of Informatics & Telecommunications,
National Centre for Scientific Research (N.C.S.R.) 'Demokritos',
153 10 Aghia Paraskevi Attikis, Greece.
[2] Knowledge & Database Systems Laboratory
National Technical University of Athens (NTUA), 157 73, Athens, Greece.
{manolis,ystavr}@iit.demokritos.gr
dkart@tee.gr

Abstract. In this paper we investigate various aspects of representing multidimensional information in the frame of the WWW. *Multidimensional XML* (MXML) is an extension of XML suitable for representing data that assume different facets, having different value or different structure, under different *contexts*. In Multidimensional XML, elements and attributes may depend on a number of *dimensions*, that define worlds under which variants of those elements or attributes hold. Moreover, we propose an extension of DTD that takes dimensions into account and is suitable for describing the logical structure of MXML documents. We also present a graph data model for MXML, and show how MXML can be reduced to conventional XML for a given world.
Keywords: Multidimensional Languages, Semistructured Data, XML, Web Databases.

1 Introduction

XML is a markup language suitable for data representation and exchange over the Web [3]. XML resembles HTML, but unlike HTML, it focuses on the structure of data rather than on presentation. XML can be seen either from a document-centric perspective or from a data-centric one. The document-centric view originates from SGML, the markup language that inspired the design of XML, and sees XML as a way to embed in a Web document information about its structure. The data-centric perspective has been adopted by those that perceive XML as a data exchange language over the Web. From this perspective the emphasis is on querying and on describing the relationships between pieces of data, in a way similar to a database schema.

Although the main characteristic of XML is its extensibility in terms of defining new element types at will, it falls short when it comes to representing *multidimensional information*, that is, information that presents different facets under

* This work has been partially supported by the Greek General Secretariat of Research and Technology under the project "Executable Intensional Languages and Intelligent Applications in Multimedia, Hypermedia and Virtual Reality" of ΠENEΔ'99.

different contexts. As a simple example imagine a report that needs to be represented at various degrees of detail and in various languages. A solution would be to create a different XML document for every possible combination. Such an approach is certainly not practical, since it involves excessive duplication of information. What is more, the different variants are not associated as being parts of the same entity. The problem of varying entities is especially present in the frame of the WWW, where information providers cannot assume too much about the background *context* of the information consumers. Therefore, there is need for data models and languages suitable for representing and exchanging multidimensional data over the Web.

Ideas on how this problem can be handled are given in [11, 10], where a formalism, called Multidimensional XML (MXML), is presented. MXML extends XML by allowing *context specifiers* to qualify elements and attribute values, and specify the contexts under which the document and its components have meaning. MXML was influenced by *Intensional HTML* (IHTML) [12], a Web authoring language, based on and extending ideas proposed for a software versioning system [9]. IHTML allows a single Web page to have different variants and to dynamically adapt itself to a given context.

In this paper, a) we motivate the use of and specify MXML syntax and semantics by reviewing and further extending the formalism presented in [11, 10], b) we clarify how the structure and the content of MXML elements and attributes depend on dimensions, c) we propose an extension of Document Type Definition (DTD) that takes into account dimensions and can be used to describe the logical structure of MXML documents, d) we present a data model for MXML, called MXMLGraph (MXMLG in short) and discuss some properties of MXML, and e) we show that, given a specific world, it is possible to obtain a conventional XML document, which constitutes the facet of the MXML document under that specific world.

2 Incorporating Dimensions in XML Documents

In a *multidimensional XML document* (*MXML document* in short), dimensions may be applied to elements and attributes. An element whose content depends on one or more dimensions is called *multidimensional element*. An attribute whose value depends on one or more dimensions is called *multidimensional attribute*.

2.1 Dimensions and Worlds

The notion of *world* is fundamental in MXML. A world represents an environment under which data in a multidimensional document obtain a meaning. A world is determined by assigning values to a set of *dimensions*.

Definition 1. *Let $\mathcal{S}$ be a set of dimension names and for each $d \in \mathcal{S}$, let $\mathcal{D}_d$, with $\mathcal{D}_d \neq \emptyset$, be the domain of d. A* **world** *W is a set of pairs (d, u), where $d \in \mathcal{S}$ and $u \in \mathcal{D}_d$ such that for every dimension name in $\mathcal{S}$ there is exactly one element in W.*

MXML uses *context specifiers* that specify sets of worlds. Context specifiers qualify the variants (or *facets*) of multidimensional elements and attributes, relating each variant to the set of worlds under which the variant becomes the holding one for the corresponding multidimensional entity. Two context specifiers are called *mutually exclusive* if they specify disjoint sets of worlds.

2.2 The Syntax of Multidimensional XML

The syntax of XML is extended as follows in order to incorporate dimensions. In particular, a multidimensional element has the form:

```
<@element_name attribute_specification>
  [context_specifier_1]
    <element_name attribute_specification_1>
      element_content_1
    </element_name>
  [/]
  . . .
  [context_specifier_N]
    <element_name attribute_specification_N>
      element_content_N
    </element_name>
  [/]
</@element_name>
```

A multidimensional element is denoted by preceding the element name with the special symbol "`@`", and encloses one or more *context elements* that constitute facets of that multidimensional element, holding under specific worlds specified by the corresponding *context specifier*. Context elements have the same form as conventional XML elements. All context elements of a multidimensional element have the same name which is the name of the multidimensional element.

To declare a multidimensional attribute we use the following syntax:

```
attribute_name = [context_specifier_1] attribute_value_1 [/] . . .
                 [context_specifier_n] attribute_value_n [/]
```

Therefore, a multidimensional attribute is assigned a set of context-value pairs. Each context-associated value becomes the holding value of the attribute under the worlds specified by the corresponding context specifier.

A *context specifier* is of the form:

```
dimension_1_specifier, ..., dimension_m_specifier
```

where `dimension_i_specifier`, $1 \leq \mathtt{i} \leq \mathtt{m}$, is a *dimension specifier* of the form:

```
dimension_name  specifier_operator  dimension_value_expression
```

A *specifier_operator* is one of $=$, $!=$, `in`, `not in`. If the *specifier_operator* is either $=$ or $!=$, the *dimension_value_expression* consists of a single dimension value. Otherwise, if the *specifier_operator* is either `in` or `not in`, the *dimension value expression* is a set of values of the form $\{\mathtt{value}_1, \ldots, \mathtt{value}_k\}$.

A context specifier may also be the reserved word "`default`", where `[default]` represents all worlds not covered by the context specifiers of the same entity. Finally, the context specifier [ ] represents the set of all possible worlds.

Example 1. A part of an imaginary menu of a restaurant described in MXML.

```
<restaurant>
  <menu>
    <salad name = "Chef's salad" vegetarian = [season = summer]"yes"[/]
                                          [season != summer]"no"[/] >
      <@comment>
        [language = English, detail = low]
          <comment> A traditional salad. </comment>
        [/]
        [language = English, detail = high]
          <comment>
            A salad, with a long history which
            is connected with the tradition of the town.
          </comment>
        [/]
        [language = French, detail in {low, high}]
          <comment> Une salade regionale traditionelle. </comment>
        [/]
      </@comment>
      <@price>
        [season = summer] <price> 3 USD </price> [/]
        [default] <price> 4 USD </price> [/]
      </@price>
      <ingredient> tomato </ingredient>
      <@ingredient>
        [season != summer] <ingredient> bacon </ingredient> [/]
      </@ingredient>
      <ingredient> olive oil </ingredient>
      <@ingredient>
        [occasion = special]
          <ingredient special_supplier=[season in {spring, summer}]"sp1"[/]
                                        [default]"sp2"[/] >
            <name> special sauce </name>
            <remarks> Must order three days in advance </remarks>
          </ingredient>
        [/]
        [default] <ingredient> normal sauce </ingredient> [/]
      </@ingredient>
    </salad>
  </menu>
  <supplier scode="sp1">
    <name> John Smith </name> <address> 234 XYZ Street </address>
  </supplier>
  <supplier scode="sp2"> ...  </supplier>
<\restaurant>
```

2.3 Dimensions Applied to Elements

While multidimensional elements can only contain context elements, context elements may contain other multidimensional elements, conventional elements, or any combination of the two in an arbitrary depth. Context elements of the same multidimensional element are not required to have the same content, or even to conform to the same structural constraints. Therefore, dimensions can affect the content of an element in every aspect, be it its structure or its value.

The effect of context in element value: Consider the element `comment` in Example 1, which is a multidimensional element whose value depends on the dimensions `language` and `detail`. The context specifier of the third context element of `comment` is `[language = French, detail in {low, high}]`, and represents possible worlds where `language = French` and `detail = low` or `language = French` and `detail = high`. In all these worlds, the value of `comment` is "`Une salade regionale traditionelle`".

The effect of context in element structure: Context specifiers also affect the element structure. For example, the fourth `ingredient` element in Example 1 contains the subelements `name` and `remarks` for the context `[occasion=special]`, but for all the other contexts, i.e. for all other possible values of the dimension `occasion` (implied by `[default]`), it contains no subelements.

Notice that it is not necessary for a multidimensional element to have context elements for every possible world. For example, the multidimensional element

```
<@ingredient>
        [season != summer] <ingredient> bacon </ingredient> [/]
</@ingredient>
```

in Example 1, has no facet for the context `[season = summer]`.

Finally, a multidimensional element or attribute whose only facet holds under every possible world, can be substituted by a conventional element or attribute.

2.4 Dimensions Applied to Attributes

Each context element can have its own (conventional or multidimensional) attributes, exactly as it can have its own (conventional or multidimensional) child elements. Within the same multidimensional element, context elements may have different attributes, exactly as they may have different child elements. Notice that in Example 1, a salad `ingredient` has the attribute `special_supplier` for the context `[occasion=special]`, but for any other context (denoted by `[default]`) `ingredient` has no attributes.

Attributes of type "ID", "IDREF" and "IDREFS" can be attached to context and conventional elements. By using attributes of types "IDREF" and "IDREFS", context and conventional elements are able to point to multidimensional, conventional, or context elements. Multidimensional elements can be attached only attributes of type "ID"; hence, although multidimensional elements can be pointed to by IDREF attributes, they cannot themselves point to other elements.

The IDREF attribute `special_supplier` in the fourth `ingredient` of Example 1, has the value `"sp1"` for the context `[season in {spring, summer}]` and the value `"sp2"` for all the other contexts (represented by `[default]`), thus pointing to different elements depending on the value of the dimension `season`.

2.5 Well-Formed MXML

The notion of *well-formed* MXML is an extension of the notion of well-formed XML. In addition, to the XML well-formedness criteria an MXML document must also exhibit the property of *context well-formedness*, which is defined below.

Definition 2. *An MXML document D is said to be* context-deterministic *iff for every multidimensional element (attribute) M in D the following condition holds: If $c_1, c_2, \ldots, c_n$ are the context specifiers qualifying the context elements (attribute values) of M then c_i is mutually exclusive with c_j for all $i \neq j$.*

In a context-deterministic MXML document each multidimensional element or attribute has at most one holding facet under any specific world.

Definition 3. *An MXML documents D is said to be* context well-formed *iff it is context deterministic and the following conditions hold: 1) For every multidimensional element there exists at least one context element, and 2) For every multidimensional attribute there exists at least one facet of that attribute.*

Context well-formedness ensures that for every multidimensional entity there is at least one world under which this entity has meaning (a holding facet).

3 Multidimensional DTD

In XML, a *Document Type Definition* (DTD) [3] is used for defining constraints on the logical structure of a document. In this section, we propose an extension of DTD, called *Multidimensional DTD*, that takes dimensions into account and is suitable for describing the logical structure of MXML documents.

Dimension Declarations: Dimensions are declared in MDTD through *dimension declarations* of the form:

```
<!DIMENSION dimension_name dimension_domain>
```

Using dimension declarations we can declare a dimension and associate with it a set of possible values. For example, the declaration

```
<!DIMENSION language {English, French}>
```

denotes that '`language`' is a dimension name and constraints its possible values to elements of the set {`English, French`}.

In the frame of this paper we assume finite dimension domains, described by enumerating their elements. Other ways of representation, as well as infinite domains, may also be useful, however they fall out of the scope of this paper.

Multidimensional Element Declarations: *Element declarations* of conventional DTD are also used in MDTD for conventional elements. Another construct, called *multidimensional element declaration*, is introduced, to deal with context dependent elements. The syntax of the new construct is:

```
<!MULTIELEMENT element_name associated_dimensions type_decl>.
```

The dimensions on which a multidimensional element depends on are declared in '`associated_dimensions`'. For example, in the following declaration:

```
<!MULTIELEMENT comment {language, detail} (#PCDATA)>
```

`comment` is declared to be a multidimensional element which depends on the dimensions `language` and `detail`.

Multidimensional element declarations allow separate constraints for the context elements of a multidimensional element. For example in:

```
<!MULTIELEMENT ingredient {season, occasion}
```

```
        [occasion = special] ((name, remarks?) | #PCDATA) [/]
        [default] (#PCDATA) [/]>
```

the type of the element `ingredient` is declared to be either `(name, remarks?)` or `(#PCDATA)` whenever the value of the dimension `occasion` is `special`; in any other case, the type of the element `ingredient` is declared to be `(#PCDATA)`.

Attribute Declarations: *Attribute declarations* have been extended to take into account multidimensional attributes. In the declaration

```
<!ATTLIST salad name CDATA #REQUIRED
       vegetarian {season} CDATA #IMPLIED>
```

the element `salad` is declared to have two attributes, namely `name` and `vegetarian`. The value of the attribute `name` does not depend on dimensions, while the value of the attribute `vegetarian` depends on the dimension `season`.

Attribute declarations allow to declare that an attribute is present under some contexts, while it is absent under other contexts. For example, in

```
<!ATTLIST ingredient
    [occasion=special] special_supplier {season} IDREF #REQUIRED [/]>
```

the element `ingredient` is declared to have the attribute `special_supplier` only for the contexts where the value of the dimension `occasion` is `special`; in this case, the attribute must exists for every possible value of the dimension `season`. In all other contexts the element `ingredient` has no attributes.

Example 2. A MDTD for the MXML document of Example 1.

```
<!DOCTYPE restaurantDTD [
<!DIMENSION language  {English, French}>
<!DIMENSION detail    {low, high, exhaustive}>
<!DIMENSION season    {spring, summer, fall, winter}>
<!DIMENSION occasion  {special, normal}>
<!ELEMENT restaurant  (menu | supplier)*>
<!ELEMENT menu  (salad+, first+, maindish+, dessert+)>
<!ELEMENT salad (comment?, price, ingredient*)>
<!ATTLIST salad name CDATA #REQUIRED
            vegetarian {season} CDATA #IMPLIED>
<!MULTIELEMENT comment {language, detail} (#PCDATA)>
<!MULTIELEMENT price {season} (#PCDATA)>
<!MULTIELEMENT ingredient {season, occasion}
       [occasion = special] ((name, remarks?) | #PCDATA) [/]
       [default] (#PCDATA) [/]>
<!ATTLIST ingredient
       [occasion = special] special_supplier {season} IDREF #REQUIRED [/]>
<!ELEMENT name    (#PCDATA)>
<!ELEMENT remarks (#PCDATA)>
<!ELEMENT supplier (name, address)>
<!ATTLIST supplier scode ID #REQUIRED>  ]>
```

4 A Data Model for MXML

Graph based data models are often used to represent XML data [1, 5, 4, 6]. In this section, we propose a data model, called *Multidimensional XML Graph* (or

MXMLG), suitable for modelling MXML documents. MXMLG provides nodes and edges of appropriate type for representing multidimensional information.

Definition 4. *Let $\mathcal{CS}$ be a set of context specifiers and $\mathcal{D}_e$, $\mathcal{D}_a$, $\mathcal{T}$ be three sets called* element names, attribute names, *and* text values *respectively. A* multidimensional XML graph $G = (N, E, r, \mathcal{CS}, \mathcal{D}_e, \mathcal{D}_a, \mathcal{T})$ *is a finite directed edge-labelled graph such that:*
1) $N = N_{me} \cup N_{ce} \cup N_{ma} \cup N_a \cup N_t$ *where* N_{me}, N_{ce}, N_{ma}, N_a *and* N_t *are disjoint sets of nodes, called* multidimensional element nodes, context element nodes, multidimensional attribute nodes, (context) attribute nodes, *and* text nodes *respectively.*
2) $E = E_e \cup E_a \cup E_{ec} \cup E_{ac} \cup E_r \cup E_t$ *where* $E_e \subseteq N_{ce} \times \mathcal{D}_e \times (N_{ce} \cup N_{me})$ *is a set of edges called* element edges, $E_a \subseteq (N_{ce} \cup N_{me}) \times \mathcal{D}_a \times (N_a \cup N_{ma})$ *is a set of edges called* attribute edges, $E_{ec} \subseteq N_{me} \times \mathcal{CS} \times N_{ce}$, *is a set of edges called* element context edges, $E_{ac} \subseteq N_{ma} \times \mathcal{CS} \times N_a$ *is a set of edges called* attribute context edges, $E_r \subseteq N_a \times (N_{ce} \cup N_{me})$ *is a set of edges called* attribute reference edges, *and* $E_t \subseteq (N_a \cup N_{ce}) \times N_t$ *is a set of edges called* text edges.
3) $L_t : N_t \rightarrow \mathcal{T}$ *is a labeling function for text nodes.*
4) r *is a specific node in* N_{ce}, *called the* root node *such that: a) Each node in the graph is reachable from* r, *and b)* $G' = (N, E_e \cup E_a \cup E_{ec} \cup E_{ac} \cup E_t)$ *is a tree rooted at* r.

MXMLG can also represent conventional XML documents since XML can be considered as a special case of MXML.

The MXMLGraph for the document of Example 1 is shown in Figure 1.

5 Properties of MXML

Context Propagation: A context specifier gives the *explicit context* of the entity that qualifies. When element or attributes are combined to form an MXML document, the explicit context of an entity does not alone determine the worlds under which that entity holds, since when an entity e_2 is part of another entity e_1, then e_2 can have substance only under the worlds that e_1 has substance. This can be conceived as if the context under which e_1 holds is inherited to e_2. The context propagated in that way is combined with (constraint by) the explicit context of each element to give the *inherited context* for that element. For determining the inherited context of an attribute, the explicit context of the attribute is used to constrain the inherited context of the element that contains the attribute. The inherited contexts can be considered as the "real" contexts for elements and attributes in the frame of the document where they occur.

Reducing MXML to XML: Each MXML document represents in fact a set of conventional XML documents. Given a world w, an MXML document can be reduced to a conventional XML document which is the facet of the multidimensional document under w. The reduction process is defined in the procedure that follows, where for convenience we consider the MXMLG graphs G and G' that correspond to the MXML and XML documents respectively.

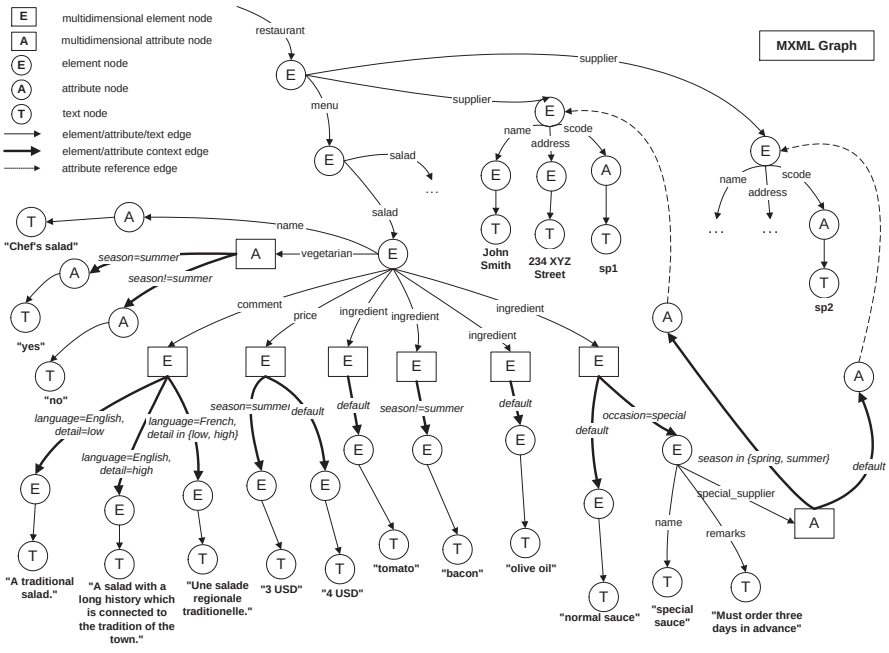

Fig. 1. The Multidimensional XML Graph for Example 1.

Procedure reduceMXMLG(G, w, G')

Step 1: Remove all context edges (e, C, e') from G for which $v \notin U_C$ where $(d, v) \in w$ and $(d, U_C) \in C$. Then remove all subgraphs not accessible from r.

Step 2: For every element/attribute context edge of the form (p, C, q), in the graph G' obtained in step 1, do the following: Let $(e_1, l_1, p), \ldots, (e_k, l_k, p)$ be all edges in G leading to p. Then replace each (e_i, l_i, p), for $i = 1, \ldots, k$ by an edge (e_i, l_i, q) of the same type. Remove the edge (p, C, q) and the node p.

Step 3: Prune all subtrees which have no text leaf node.

A system that implements the above process and demonstrates a number of examples is presented in [7].

Validity of MXML Documents: The validity of an MXML document is defined with respect to an MDTD, and is an extension of the notion of validity for conventional XML documents. Given a world w, it is possible to apply a process similar to the one presented above, and reduce an MDTD to a conventional DTD that holds under w. An MXML document M is *valid with respect to an MDTD under a world* w, if the conventional XML document D obtained by reducing M for that particular world w is valid with respect to the DTD that is the MDTD facet under w. An MXML document is *valid with respect to an MDTD* if it is valid with respect to that MDTD under every possible world.

6 Discussion and Motivation for Future Work

Investigating potential applications of MXML is an interesting direction. The representation of time-dependent information using MXML is promising [8], since various notions of time can be seen as MXML dimensions. The use of MXML to encode geographical information, where objects depend on dimensions like `scale` and `theme` is another area that we examine. Other schema languages for MXML, could also be investigated. Research on query languages for XML is especially active [1, 2, 5], however, in this paper we do not consider query languages for MXML. A query Q on an MXML document D can be seen as a pair (Q_w, w) where Q_w is a query on the conventional XML document which is the facet of D under the world w. The development of a "multidimensional query language" especially designed for the MXML data model is in our immediate plans.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.
2. A. Bonifati and S. Ceri. Comparative analysis of five XML query languages. *SIGMOD Record*, 29(1), March 2000.
3. T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0 (second edition). http://www.w3.org/TR/REC-xml, October 2000.
4. J. Clark and S. DeRose. XML Path Language (XPath), Version 1.0 (W3C Recommendation). http://www.w3.org/TR/xpath, 1999.
5. A. Deutch, M. Fernández, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for XML. http://www.w3.org/TR/NOTE-xml-ql, 1999.
6. M. Fernandez and J. Robie. XML Query Data Model (W3C Working Draft 11 May 2000). http://www.w3.org/TR/query-datamodel, 1999.
7. M. Gergatsoulis, Y. Stavrakas, D. Karteris, A. Mouzaki, and D. Sterpis. A Web-based System for Handling Multidimensional Information through MXML. *It will be presented at 5th East-European Conference on Advances in Databases and Information Systems (ADBIS), Vilnius, Lithuania, September 2001.*
8. T. Mitakos, M. Gergatsoulis, Y. Stavrakas, and E. V. Ioannidis. Representing time-dependent information in multidimensional XML. *Proc. of the 23rd Int. Conf. "Information Technology Interfaces" (ITI'01), Pula, Croatia, June 2001.*
9. J. Plaice and W. W. Wadge. A New Approach to Version Control. *IEEE Transactions on Software Engineering*, 19(3):268–276, 1993.
10. Y. Stavrakas, M. Gergatsoulis, and T. Mitakos. Representing context-dependent information using Multidimensional XML. In J. Borbinha and T. Baker, editors, *Research and Advanced Technology for Digital Libraries (ECDL'00), Proceedings*, Lecture Notes in Computer Science (LNCS) 1923, pages 368–371. Springer, 2000.
11. Y. Stavrakas, M. Gergatsoulis, and P. Rondogiannis. Multidimensional XML. In P. Kropf, G. Babin, J. PLaice, and H. Unger, editors, *Distributed Communities on the Web, Third International Workshop (DCW'2000)*, Lecture Notes in Computer Science (LNCS) 1830, pages 100–109. Springer-Verlag, 2000.
12. W. W. Wadge, G. D. Brown, M. C. Schraefel, and T. Yildirim. Intensional HTML. In *Proc. of the 4th Int. Workshop on Principles of Digital Document Processing (PODDP '98)*, LNCS 1481, pages 128–139. Springer, March 1998.