Developing XML Documents with Guaranteed "Good" Properties

D.W. Embley[1] and W.Y. Mok[2]

[1] Brigham Young University, Provo, Utah 84602, USA, embley@cs.byu.edu
[2] University of Alabama at Huntsville, Huntsville, Alabama 35899, USA, mokw@email.uah.edu

Abstract. Many XML documents are being produced, but there are no agreed-upon standards formally defining what it means for complying XML documents to have "good" properties. In this paper we present a formal definition for a proposed canonical normal form for XML documents called XNF. XNF guarantees that complying XML documents have maximally compact connectivity while simultaneously guaranteeing that the data in complying XML documents cannot be redundant. Further, we present a conceptual-model-based methodology that automatically generates XNF-compliant DTDs and prove that the algorithms, which are part of the methodology, produce DTDs to ensure that all complying XML documents satisfy the properties of XNF.

1 Introduction

Many DTDs (Document Type Definitions) for XML documents are being produced (e.g. see [XML]), and soon many XML-Schema specifications [XML00] for XML documents will be produced. With the emergence of these documents, we should be asking the question, "What constitutes a good DTD?"[1] Intuitively, this should ensure that complying XML documents are compactly connected in as few hierarchies as possible while simultaneously ensuring that no data value in any complying document can be removed without loss of information.

In this paper we formalize these ideas by defining XNF, a normal form for XML documents,[2] and we present a way to generate DTDs that are guaranteed to be in XNF. We assume that the DTDs produced are for XML documents representing some aspect of the real world—those for which conceptual modeling makes sense.[3] Under this assumption, we argue that to produce a "good" DTD for an application A, we should first produce a conceptual-model instance for

[1] Since we do not address issues beyond hierarchical structure in this document, we discuss the issues in terms of DTDs rather than XML-Schemas.

[2] The idea of XNF is based on nested normal form as defined in [MEN96] and is also related to other similar nested normal forms such as those surveyed in [Mok].

[3] The class of documents we are considering is certainly large, but also certainly not all-inclusive. We do not, for example, consider text documents where the order of textual elements is important.

A and then apply a transformation guaranteed to produce an XNF-compliant DTD.

Although we can guarantee XNF compliance, we cannot guarantee uniqueness. In general, several "good" DTDs, correspond to any given conceptual-model instance. Selecting the best depends on usage requirements and viewpoints that are "in the eye of the beholder." Sometimes these usage requirements or viewpoints should even cause the principles of XNF to be violated, but most of the time XNF-compliance should be compatible with usage requirements and viewpoints. Heuristic "rules of thumb" can go a long way toward resolving this problem of nonuniqueness and can often produce results that are highly satisfactory. We believe, however, that the ultimate resolution should be synergistic. Given heuristic rules and the principles of XNF, a system should work with a user to derive a suitable application DTD. The system can automatically derive reasonable XNF-compliant DTDs. The user may adjust, reject, or redo any of the generated suggestions. The system can check the revisions and report any violations of XNF so that the user is aware of the consequences of the revisions. Iterating until closure is reached, the user can further revise the DTD, and the system can evaluate and provide feedback.

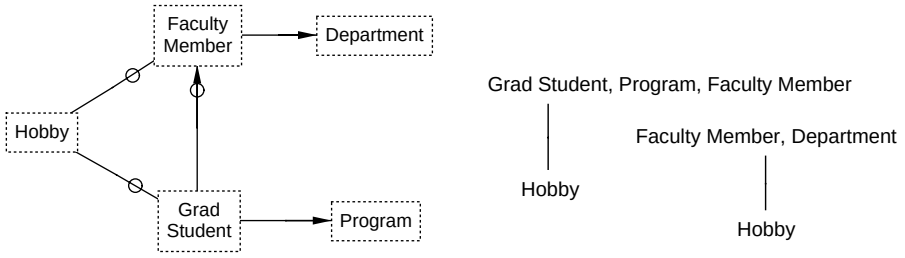
We are aware of only one other research effort that closely parallels our work—namely [BGH00].[4] [BGH00] makes the same argument we make, namely (1) that graphical conceptual-modeling languages offer one of the best—if not the best—human-oriented way of describing an application, (2) that a model instance should be transformed automatically into an XML DTD (or XML schema), and (3) that the transformation should maximize connectivity among the XML components and should minimize the redundancy in complying XML documents. The authors of [BGH00] use Object Role Modeling [Hal99] as their conceptual model and give a set of twelve heuristics for generating the "best" XML schema. What is missing is the guarantee that their transformation achieves maximum connectivity and minimum redundancy. In this paper we use a more generic conceptual model and a simpler set of heuristics to achieve similar results, but the main contribution is the guarantee that complying XML documents satisfy the formal properties XNF.

We present our contribution as follows. Section 2 provides motivating examples and foundation definitions. Besides arguing that we can produce DTDs that yield only XNF-compliant XML documents, we also provide examples to show that even for simple applications, it is easy to produce (and thus nontrivial to avoid) DTDs that have redundancy and have more hierarchical clusters than necessary. In Section 3 we present straightforward algorithms that guarantee the properties of XNF for a large number of practical cases. We then show in Section 4, however, that these straightforward algorithms depend on the given conceptual-model instance being in a particular canonical form. Although many

[4] Others, such as [BR00,CSF00], discuss a UML-to-XML transformations, but they do not investigate properties of conceptual-model generated XML documents. Similarly, OMG's XMI effort [XMI] also provides a way to represent UML in XML, but the effort is devoid of XNF-like guarantees for XML documents.

practical model instances are naturally specified in the required canonical form, some are not. We therefore explain how to achieve this canonical form Section 4. We then prove that for a given conceptual-model instance in the canonical form, the algorithms in Section 3 yield XNF-compliant DTDs. In Sections 5 and 6 we generalize our approach and give algorithms for producing XNF-compliant DTDs for a more-inclusive set of conceptual-model instances. We conclude in Section 7 and present the status of our implementation.

2 Motivating Example



(a) Sample Diagram (b) Sample Scheme Trees

Fig. 1. Simple Faculty-Student-Hobbies Diagram and Scheme Trees

As a motivating example, consider the conceptual-model diagram in Figure 1(a). Although based on [EKW92], the conceptual modeling approach we present here is generic. Users model the real world by constraint-augmented hypergraphs, which we call *CM hypergraphs* (conceptual-model hypergraphs). Vertices of CM hypergraphs are object sets denoted graphically as named rectangles. The object set *Hobby* in Figure 1(a), for example, may denote the set {*Chess*, *Hiking*, *Sailing*}. In the general conceptual model, edges have two or more connections to object sets, but we restrict ourselves until Section 5 to edges with just two connections. Edges representing functional relationships have arrowheads on the range side. In Figure 1(a), a graduate student enrolls in only one program (e.g. *PhD*, *MS*) and has only one faculty-member advisor. A connection between an object set and a relationship set may be optional or mandatory, denoted respectively by an "O" on the edge near the connection or the absence of an "O." A faculty member need not have hobbies and need not have advisees, but must be in a department. The inclusion constraints, which for Figure 1(a) are only simple referential integrity constraints, must all hold. For optional participation, the inclusion constraint allows a proper subset, but for mandatory participation, the subset can never be proper. Later in the paper, a triangle with its apex connected to a generalization object set and its base connected to one or more specialization object sets will denote an explicit inclusion constraint—the objects in any specialization object set must be a subset of the objects in the generalization object set.

An XML document has a hierarchical structure with a single root. The set of structures immediately below the single root constitutes a forest of hierarchical trees. It is this forest of trees we wish to derive from a conceptual-model instance.

We abstractly represent each tree in this forest as a *scheme tree* [MEN96].

Example 1. The two scheme trees in Figure 1(b) cover the model instance in Figure 1(a). Textually written, the scheme trees in Figure 1(b) are *(Grad Student, Program, Faculty Member, (Hobby)*)** and *(Faculty Member, Department, (Hobby)*)**. □

From a forest of scheme trees, we can derive a DTD for any model instance. There are several ways we can represent a scheme tree as a DTD (especially if we use XMI [XMI] or features such as ATTLIST, ID, and IDREF). We do not concern ourselves further with this issue, but rather concentrate on generating a forest of XNF-compliant scheme trees.

Example 2. As motivational examples, consider the following scheme-tree forests.

1. *(Department, (Faculty Member, (Hobby)*, (Grad Student, Program, (Hobby)*)*)*)**
2. *(Faculty Member, Department, (Hobby)*, (Grad Student, Program, (Hobby)*)*)**
3. *(Hobby, (Faculty Member)*, (Grad Student)*)*, (Grad Student, Program, Faculty Member)*, (Department, (Faculty Member)*)**
4. *(Hobby, (Faculty Member, Department)*, (Grad Student, Program, Faculty Member)*)**
5. *(Faculty Member, Department, (Hobby, (Grad Student)*)*)*, (Grad Student, Faculty Member, Program)** □

We claim (and will shortly give the formal basis for showing) that the first two sample scheme-tree forests in Example 2, which each consist of a single tree, can never have a redundant data value in any complying XML document. The third scheme-tree forest, as well as the scheme-tree forest in Figure 1(b), also never allow redundancy, but both have more than one scheme tree and thus neither is as compact as the first two sample scheme trees. The fourth sample scheme-tree forest allows redundancy—since faculty members and grad students are listed repeatedly for each additional *Hobby* in which they participate, department values for faculty members and program values as well as faculty advisors for grad students can be redundant. The first tree of the fifth scheme-tree forest also allows redundancy—whenever faculty members have the same hobbies, all the graduate students that also share these hobbies are listed.

Definition 1. *Let T be a scheme tree for a CM hypergraph M, and let t be a scheme-tree instance over T. A data value v in t is redundant with respect to a constraint C that holds for M if when v is replaced in t by some symbol, say $\perp$, where $\perp$ is not in t, C implies $\perp = v$.* □

Although many constraints are possible, we consider only functional constraints, multivalued constraints, and inclusion constraints. We further restrict this set to those that are conceptual-model compliant in the sense that they occur naturally within a conceptual-model instance as defined below. This set of constraints is a common standard set that is sufficient for many, if not most, practical cases.

Let T be a scheme tree. A *path* of T is a sequence of nodes from the root node of T to a leaf node of T. Thus, if T has n, $n \geq 1$, leaf nodes, T has n paths. A scheme tree T is *properly constructed* for a CM hypergraph M if every path of T embeds a sequence of some connected edges in M.

Definition 2. *Let T be a properly constructed scheme tree for a CM hypergraph M, and let t be a scheme-tree instance over T. Let $X \rightarrow Y$ be a functional edge in M that is contained in a path of T, and let s be a subtuple over XY in t. Let A be an attribute of Y,[5] and let a be the A value in s. Then t has redundancy with respect to the functional constraint $X \rightarrow Y$ if the a-value in s appears more than once in t. If such a scheme-tree instance t can exist, we say that T has potential redundancy with respect to the functional constraint $X \rightarrow Y$.* □

Example 3. Scheme-tree four in Example 2 has potential redundancy with respect to the functional edge *Faculty Member* → *Department* because faculty members appear once for each hobby and may participate in several hobbies. Similarly, since grad students also appear once for each hobby and may participate in several hobbies, there is potential redundancy with respect to both the functional constraints *Grad Student* → *Program* and *Grad Student* → *Faculty Member*.

Definition 3. *Let T be a properly constructed scheme tree for a CM hypergraph M, and let t be a scheme-tree instance over T. Let X—Y be an edge in M that is contained in a path of T, and let s be a subtuple over XY in t. Let y be the Y subtuple in s.[6] Then t has redundancy with respect to the multivalued constraint X—Y if the y-subtuple in s appears more than once in t. If such a scheme-tree instance t can exist, we say that T has potential redundancy with respect to the multivalued constraint X—Y.* □

Example 4. The first scheme tree in the fifth scheme-tree forest in Example 2 has potential redundancy with respect to the edge *Hobby* — *Grad Student*. Whenever faculty members have the same hobby, hobby values appear more than once.

Definition 4. *A scheme-tree forest F corresponds to a conceptual-model instance M if each tree of F is a properly constructed scheme tree and the union of the edges in all the scheme trees of F covers the edges in M.* □

Definition 5. *Let F be a scheme-tree forest corresponding to a conceptual-model instance M. Let T be a scheme tree in F with only a root node and only one*

[5] A is Y for the binary case, but in general, Y is a set of attributes.
[6] For the binary case y is a value, but in general, y is a subtuple.

object set G in the root node. If there exist object sets S_1, ..., S_n within the nodes of F such that the S_i's $(1 \leq i \leq n)$ are specializations of G in M and $G = \cup_{i=1}^{n} S_i$ for all scheme-tree-forest instances for F, the values in G are redundant and T has potential redundancy with respect to the inclusion constraint specifying that the S_i's are union specializations of G. □

Example 5. To illustrate inclusion constraints and redundancy with respect to inclusion constraints, we present the CM hypergraphs in Figure 2. Figure 2(a) is the same as Figure 1(a) except that *Hobby* is optional for both *Faculty* and *Grad Student*. The diagram in Figure 2(a) allows all hobbies for both faculty and students to be listed, not just those shared jointly by at least one faculty member and grad student as is the case for the hobbies in the diagram in Figure 1(a). Using the application model instance in Figure 2(a), however, we cannot include all of the data values in scheme-tree instances for either the first or the second scheme tree in Example 2. This is because the model instance in Figure 2(a) allows hobbies to be listed that are neither faculty hobbies nor student hobbies; thus an "extra" scheme tree is necessary to accommodate these hobbies. If this is not what we want (it probably isn't), we should model our microworld as in Figure 2(b) where the union constraint[7] on the generalization/specialization ensures that *Hobby* contains only hobbies that are faculty hobbies (contained in *Faculty Hobby*) or student hobbies (contained in *Grad Student Hobby*). Since *Hobby* = *Faculty Hobby* ∪ *Grad Student Hobby*, *Hobby* is redundant and we should eliminate it as Figure 2(c) shows. For Figure 2(c), the first and second scheme tree in Example 2 apply, except that the *Hobby* element associated with *Faculty* should be *Faculty Hobby* and the *Hobby* element associated with *Grad Student* should be *Grad Student Hobby*. With this solution, we eliminate redundancy with respect to inclusion constraints, and we also have better names for XML tags. □

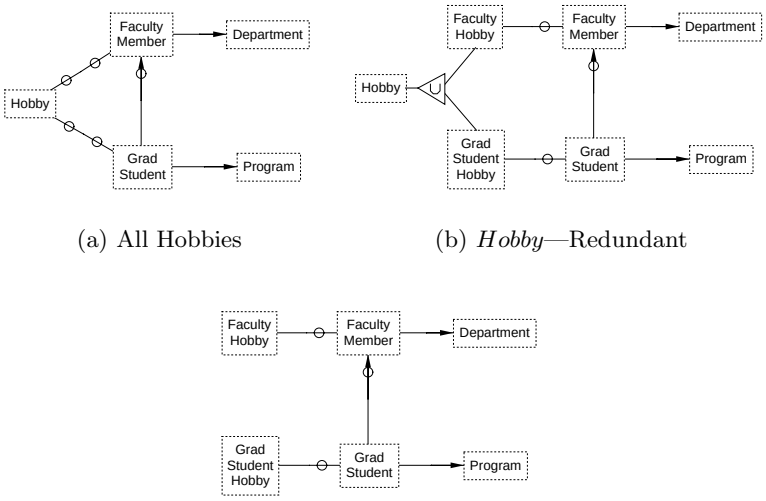
Definition 6. *Let M be a CM hypergraph. Let F be a scheme-tree forest corresponding to M. F is in XNF_C if each scheme tree in F has no potential redundancy with respect to a specified set of constraints C and F has as few, or fewer, scheme trees as any other scheme-tree forest corresponding to M in which each scheme tree has no potential redundancy with respect to C. When all constraints apply or when the specified set of constraints is clear from the context, we write XNF_C simply as XNF.* □

Example 6. We claim (and will later show) that only the first two scheme-tree forests in Example 2 are in XNF. □

3 Binary Algorithm

In Figure 3 We present our first algorithm to generate scheme-tree forests. This algorithm requires the input to be a conceptual-model instance with only binary edges connecting vertices and no explicit generalization/specialization.

[7] A union symbol inside a generalization/specialization triangle denotes that the generalization object set is a union of the specialization object sets.



(a) All Hobbies (b) *Hobby*—Redundant

(c) No ID Redundancy

Fig. 2. Illustration for Inclusion Dependencies and ID Redundancy

Example 7. Consider the model instance in Figure 1(a) as the input to Algorithm 1. If we select *Department* as the root node, we make *Faculty Member* a child of *Department* and designate it as a continuation attribute and then make *Hobby* a child of *Faculty Member*; further since *Faculty Member* is a continuation attribute, we make *Grad Student* another child of *Faculty Member* and designate it as a continuation attribute and then add *Program* in the node with *Grad Student* and finally make *Hobby* a child node of the node containing *Grad Student*. The result is the first scheme tree in Example 2. If we select *Faculty Member* as the starting node, we can generate the second scheme tree in Example 2. If we select *Grad Student* as the starting node, proceed as far as we can, and then select *Faculty Member* from the remaining unmarked nodes and proceed, we can generate the scheme-tree forest in Figure 1(b). Similarly, we can generate the third scheme-tree forest in Example 2, by starting with *Hobby*, then *Grad Student*, and then *Department*. We cannot, however, generate either the fourth or the fifth scheme-tree forest in Example 2. □

Observe from our discussion of Examples 2, 3, 4, and 7 that Algorithm 1 disallows the sample scheme-tree forests that have potential redundancy. Indeed, we claim, and will prove in Section 4 that Algorithm 1 can be used to generate only scheme-tree forests that have no potential redundancy with respect to functional and multivalued constraints.

Although Algorithm 1 can guarantee no potential redundancy, it does not guarantee that the scheme trees are as compact as possible. To get XNF (no potential redundancy *and* maximum compactness), we can add the following to Algorithm 1:

– Before the *Until* statement, add the following statements:

```
Input: a binary CM hypergraph H
    (with no explicit generalization/specialization).
Output: a scheme-tree forest.
Until all vertices and edges have been marked
    If one or more unmarked vertices remain
        Let V be a selected unmarked vertex in H;
        Mark V in H;
    Else
        Select an unmarked edge E;
        Let V be one of the two vertices of E;
    Make V the root node of a new scheme tree T;
    Designate V in T as a continuation attribute;
    While there is an unmarked edge E = (A, B) in H
            such that A is a continuation attribute in T
        Mark E in H;
        If the B-E connection is mandatory, Mark B in H;
        If A ↔ B
            Add B to T in the node containing A;
            If the B-E connection is mandatory
                Designate B in T as a continuation attribute;
        Elseif A → B
            Add B to T in the node containing A;
        Elseif B → A
            Add B to T in a new child node of the node containing A;
            If the B-E connection is mandatory
                Designate B in T as a continuation attribute;
        Else Add B to T in the node containing A;
```

Fig. 3. Algorithm 1—Binary Algorithm

```
Compute the functional closure of each vertex using only fully functional edges
    (i.e. using only functional edges whose domain side is not optional);
Order the vertices first on the number of closures in which the vertex appears (descending)
    then on the closure size (descending), and finally alphabetical (ascending);
Discard from the tail-end of the ordered list, those vertices included in only a single closure;
Order the discarded vertices on the number of incident edges (descending) and then
    alphabetical (ascending);
Append this list of ordered "discarded" vertices to the tail-end of the first ordered list;
```

– Change the *If-Else* that selects the root of a new scheme tree to:

```
From the ordered list of vertices, select the first unmarked vertex V in H to be
    the root node of a new scheme tree T.
If there is no unmarked vertex left in the list, then select the marked vertex V such that V is
    in an unmarked edge and V comes before any other vertices in unmarked edges in the list.
```

We call Algorithm 1 with this modification *Algorithm 1.1.*

Example 8. For the CM hypergraph in Figure 1(a), fully functional closures are: $Department^+ = \{Department\}$, $Faculty\ Member^+ = \{Faculty\ Member, Department\}$, $Grad\ Student^+ = \{Grad\ Student, Faculty\ Member, Department, Program\}$, $Program^+ = \{Program\}$, $Hobby^+ = \{Hobby\}$. Thus, *Department* is included in three closures, *Faculty Member* is included in two, *Grad Student*, *Program*, and *Hobby* are included in one with *Grad Student* having the largest closure size. Since the last three vertices on the list are included in only a single closure, they are ordered according to their respective number of incident edges: *Grad Student* has 3, *Hobby* has 2, and *Program* has 1. Hence, the order is *Department*, *Faculty Member*, *Grad Student*, *Hobby*, *Program*. Observe that for Algorithm 1.1, we produce the first scheme tree in Example 2. □

An alternate way to select the starting node for a new scheme tree is:

Let V be the selected vertex in Algorithm 1.1.
If V has exactly one incident unmarked edge E and E is fully functional from W to V, select W;
Else Select V;

We call Algorithm 1 with this modification *Algorithm 1.2*.

Example 9. Observe that Algorithm 1.2 produces the second scheme tree in Example 2. □

As we shall prove in the next section, we can use Algorithm 1 to guarantee no potential redundancy for any starting vertex in Figure 1(a). Further, the algorithm can guarantee the fewest scheme trees if the starting vertices for scheme trees are chosen according to one of the two criteria presented. Thus, we can use Algorithm 1.1 or Algorithm 1.2 to produce scheme trees in XNF.

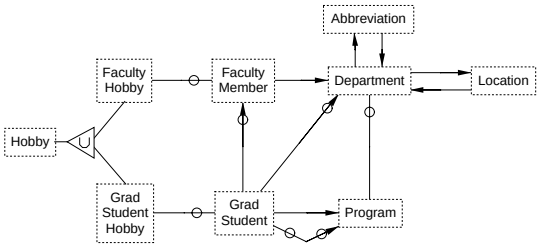
4 Assumptions, Requirements, and Guarantees

Unfortunately, Algorithms 1, 1.1, and 1.2 do not work for any CM hypergraph. Two conditions are required: (1) canonical and (2) binary. We discuss the canonical requirement in this section and show how to remove the binary requirement in the next section where we give a more general algorithm for producing scheme trees for XNF-compliant XML documents.

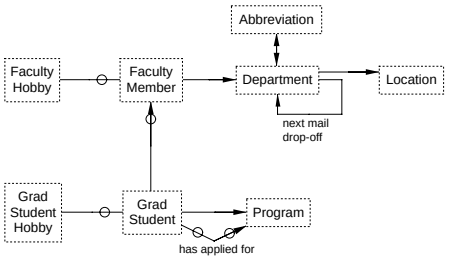
Definition 7. *A binary CM hypergraph H is* canonical *if (1) no edge of H is redundant; (2) no vertex of H is redundant; and (3) bidirectional edges represent bijections.* □

Example 10. Figure 4(a) shows a noncanonical CM hypergraph; Figure 4(b) shows its canonical counterpart. To illustrate the edge-redundancy requirement, consider the edge between *Grad Student* and *Department*. If it means the department of the student's faculty advisor, it is redundant.[8] Its removal preserves both information and constraints—we can recompute it as a join-project over the advisor and faculty/department relationship sets, and the constraints of these same two relationship sets imply both its functional and its mandatory and optional participation constraints. Similarly, if the edge between *Program* and *Department* represents the department that administers the student's program, it is redundant and can be removed. Assuming that the edge with optionals between *Grad Student* and *Program* represents a program a student has applied for, whereas the edge with no optionals represents the program in which the student is currently enrolled, neither edge is redundant. To illustrate the vertex-redundancy requirement, consider *Hobby*, which is a redundant object set as explained earlier in Example 5. To illustrate the bidirectional-edge requirement, consider the functional edges between *Department* and *Abbreviation*

[8] We make neither the universal-relation assumption nor the universal-relation-scheme assumption [FMU82,Ken81].



(a) Noncanonical



(b) Canonical

Fig. 4. Noncanonical and Canonical Model Instances

and between *Department* and *Location*. For the abbreviations, we have a bijection between departments and their abbreviations (e.g. *Computer Science* and *CS*), but for the locations we have a permutation—*Department* → *Location* gives the address of the department, whereas *Location* → *Department* gives the department for next mail drop for a mail carrier. Figure 4(b) shows a canonical CM hypergraph. No relationship set or object set is redundant, and the bidirectional edge[9] represents its only bijection. Since we have a permutation, we choose to model it as a permutation in a recursive relationship set Figure 4(b) shows.[10] □

Using the model instances in Figure 4(a), we can illustrate the redundancy problems that can arise from Algorithm 1 when a CM hypergraph is not canonical.

[9] Note that $A \leftrightarrow B$ is not the same as $A \rightarrow B$ and $A \leftarrow B$. The former indicates a bijection, whereas the latter simply means two functional relationships between A and B.

[10] This choice does not effect our XNF result, but it is a heuristic transformation that almost always produces a more pleasing result.

Example 11. Starting with *Department* in Figure 4(a), Algorithm 1 generates the scheme-tree forest with trees *(Hobby)** and *(Department, Abbreviation, (Abbreviation)*, Location, (Location)*, (Faculty Member, (Faculty Hobby)*, (Grad Student, (Grad Student Hobby)*, Program, Program)*)*, (Grad Student)*, (Program)*)**. First observe that *(Hobby)** is redundant as explained in Example 5. Next observe that *(Program)** is a list of programs for a department, but according to the model, these can all be computed by finding the programs of the grad students being advised by faculty members in the department. Similarly, *(Grad Student)** is a list of grad students in a department, but these can all be computed by finding the grad students being advised by faculty members in the department. Finally, observe that the constructions *Abbreviation, (Abbreviation)** and *Location, (Location)** are strange. For the departmental abbreviations which are in a one-to-one correspondence with the departments, we should drop the second mention in *(Abbreviation)** because it is redundant. For the locations, *(Location)** represents the address of the department whereas *Location* represents the address of the department next on the list in a mail route; thus, they should both be left as they are. As we shall see in the next example, however, we can fix this awkward construction by a heuristic modification. □

Example 12. By way of contrast to Example 11, starting with *Department* in Figure 4(b),[11] Algorithm 1 generates the scheme tree forest *(Department, Abbreviation, Location, Department of next mail drop-off, (Faculty Member, (Faculty Hobby)*, (Grad Student, (Grad Student Hobby)*, Program, Program applied for)*)*)**. Observe that the redundant lists of programs, grad students, and hobbies are not present and that the redundant abbreviations and awkward locations have disappeared. □

We now state our theorem proving that Algorithm's 1.1 and 1.2 generate XNF scheme trees.[12]

Theorem 1. *Let H be a canonical CM hypergraph with no explicit generalization/specialization whose edges are all binary. If F is a scheme-tree forest generated from H by Algorithm 1.1 or 1.2, then each tree in F is in* $\text{XNF}_{\{\text{FC,MC}\}}$.[13] □

5 *N*-ary Algorithm

In this section we generalize Algorithm 1 for CM hypergraphs with n-ary edges. Three new problems arise in the generalization: (1) n-ary edges may be compositions of edges with lesser arity, (2) connecting sets of attributes between

[11] For clarity we make use of the names designating the meaning of some of the relationship sets. We suggest the use of clarifying information whenever there may be ambiguity, or even just whenever additional clarifying explanations are appropriate or desired.

[12] Proofs for this and all other theorems can be found in a technical report at http://osm.cs.byu.edu/Papers.html.

[13] In this notation, FC denotes a functional constraint, MC denotes a multivalued constraint, and IC denotes an inclusion constraint.

relationship sets may have different meanings, and (3) there are more degrees of freedom for scheme-tree configurations. We discuss each of these problems in turn.

Edge Decomposition. As one example of edge decomposition, consider the edge connecting *Name*, *Address*, *Phone*, and *Major* in Figure 5(a). If the phone for the person identified by the *Name-Address* pair is the home phone at the address of the person (i.e. is not the cell phone, for example), the 4-ary edge can be reduced by making the phone dependent only on the address as Figure 5(b) shows. As another example of edge decomposition, consider the 5-ary edge in Figure 5(a). Assuming that the schedule depends only on the course itself (the normal assumption unless the course is an individual-instruction course), we can decompose the edge as Figure 5(b) shows. Whether we can split the day from the time depends on the scheduling policy; our choice in Figure 5(b) assumes that courses can be scheduled at different times on different days.

(a) Noncanonical (b) Canonical

Fig. 5. *N*-ary Noncanonical and Canonical Model Instances

There are multiple ways an edge can be decomposed, but these are all found in the relational database literature. Thus, we only mention them here without redeveloping them. Included are reductions to satisfy the requirements of 3NF (head and tail reductions in [Emb98]), reductions to satisfy the stronger requirements of BCNF (embedded FD reductions [Emb98]), and reductions to satisfy the even stronger requirements of 4NF and 5NF (non-FD edge reductions [Emb98]). The sample reduction for home phones above is a head reduction, and the sample reduction for course schedules is a non-FD edge reduction.

To accommodate these requirements, we now augment our definition of what it means for a CM hypergraph to be canonical.

Definition 8. *A binary CM hypergraph H is* canonical *if (1) no edge of H is redundant; (2) no vertex of H is redundant; (3) bidirectional edges represent bijections; and (4) every n-ary edge is fully reduced.* □

Example 13. The CM hypergraph in Figure 5(a) is noncanonical—neither the 4-ary nor the 5-ary edge is fully reduced. Assuming, as stated earlier, that the *Course-Day-Time* relationship set cannot be further reduced, the CM hypergraph in Figure 5(b) is canonical. □

Different Meanings. Consider *Name-Address* pairs in Figure 5(b). Suppose there are two names n_1 and n_2 and two addresses a_1 and a_2. Further suppose that in the relationship set with $Major$, n_1 relates to a_1 and n_2 relates to a_2, but in the relationship set with *Course*, n_1 relates to a_2 and n_2 relates to a_1. Under

these assumptions, we cannot have the scheme tree *(Major, (Name, Address, (Course)*)*)**, which we would expect should be permissible. We cannot have this scheme tree because under *Major* our scheme-tree instance would have the subtuples $\{(Name, n_1), (Address, a_1)\}$ and $\{(Name, n_2), (Address, a_2)\}$, but in order to nest courses under these tuples, we need $\{(Name, n_1), (Address, a_2)\}$ and $\{(Name, n_2), (Address, a_1)\}$.

In general, to provide for nesting, the projections on the intersecting attribute sets between two edges must be identical for every valid interpretation. This condition holds when the projections on the intersection object sets between two edges have the "*same meaning*." Indeed, for the CM hypergraph in Figure 5(b), *Name-Address* pairs have the same meaning in both the *Major* relationship set and the *Course* relationship set—in both, the pair identifies an individual student.

Degrees of Freedom. Consider the ternary is-taking-course relationship set in the CM hypergraph in Figure 5(b). The scheme trees we may create for this relationship set include, for example, (a) *(Course, Name, Address)**, (b) *(Address, (Name, Course)*)**, and (c) *(Name, (Address, (Course)*)*)**. Whereas there are only three possible scheme trees for a nonfunctional binary relationship set, there are thirteen possible scheme trees for a nonfunctional ternary relationship set such as the is-taking-course relationship set in Figure 5(b). For an n-ary edge in general, we may have any of the $2^n - 1$ sets of vertices in the root node of its scheme tree, any of the $2^{n_1} - 1$ sets of vertices of the remaining n_1 vertices not chosen for the root in its first sublevel node, any of the $2^{n_2} - 1$ sets of vertices of the remaining n_2 vertices not chosen for the root or first sublevel node in its second sublevel node, and so forth.

The additional degrees of freedom make it more difficult to specify a scheme-tree generation algorithm, but the idea for the generalization of Algorithm 1 is straightforward—we are still searching for the largest hierarchical structures embedded within the CM hypergraph. Figure 6 shows Algorithm 2, which generalizes Algorithm 1 by allowing n-ary edges.

Example 14. Consider the model instance in Figure 5(b) as the input to Algorithm 2. We choose the initial single-path scheme tree to be *Major* as the root, with *Name* and *Address* together in a child node of the root. We mark all three attributes in the given hypergraph and designate all of them as continuation attributes in the scheme tree. Since the edge {*Name*, *Address*, *Course*} satisfies the five conditions of the while-loop, we add *Course* as a child of the node containing *Name* and *Address*. *Course* is then marked and designated as a continuation attribute. However, we can attach neither the edge {*Course*, *Day*, *Time*} nor the edge $Address \rightarrow Phone$ to the scheme tree since we cannot find the node N in Algorithm 2 that satisfies the five conditions for these two edges. In particular, Condition 4 cannot be satisfied. We thus create a scheme tree for each of these two edges. The resulting scheme-tree forest is: *(Major, (Name, Address, (Course)*)*)**, *(Course, (Day, Time)*)**, *(Address, Phone)**. □

Although still an open question for future research, there appears to be no effective algorithm for guaranteeing the fewest possible scheme trees. The prob-

Input: a canonical CM hypergraph H (with no explicit generalization/specialization).
Output: a scheme-tree forest.
Until all edges and vertices have been marked
    If one or more unmarked edges remain
        Select an unmarked edge E in H;
        Create a single-path scheme tree T from E such that
            the set of nodes in T is a partition of E, and
            either each vertex in the root node of T is mandatory for E,
            or there is at most one vertex in the root node of T;
        Mark E in H
        For each vertex V in E
            If the V-E connection is mandatory or V is the root node of T,
                Mark V in H and designate V in T as a continuation attribute;
        While there is an unmarked edge E in H such that
                (1) P is a path in T,
                (2) D is the maximal nonempty set of attributes in $E \cap P$ that
                    has the "same meaning" in both E and P,
                (3) each of the attributes in D is designated as a continuation attribute in P,
                (4) there exists a node N in P such that $D \subseteq Ancestor(N)$ and
                    $E \rightarrow Ancestor(N)$;
                (5) If there are more than one nodes that satisfy Condition 4,
                    let N be the lowest one.
            Mark E in H;
            Create a single-path scheme tree T' from $E - D$ such that
                the set of nodes in T' is a partition of $E - D$;
            Make the root node of T' a child of N;
            For each vertex V in E
                If the V-E connection is mandatory,
                    Mark V in H and designate V in T as a continuation attribute;
    Else
        Select an unmarked vertex U;
        Make U the root node of a new scheme tree;
        Mark U in H;

Fig. 6. Algorithm 2—N-ary Algorithm

lem is that there are too many degrees of freedom—too many ways to add an n-ary edge to a scheme tree. Unfortunately, the choice makes a difference and the proper choice cannot always be determined until additional edges are added to a scheme tree. Thus, backtracking is required.

We can nevertheless apply variations to Algorithm 2 that are similar to the variations of Algorithm 1, as specified in Algorithm 1.1 and 1.2. For CM hypergraphs whose edges only intersect on single object sets or whose only multiple-object-set edge intersections follow a single path in the hypergraph, we can use variations similar to Algorithm 1.1 and 1.2 to guarantee minimality. Since real-world CM hypergraphs tend to have mostly binary edges or tend to satisfy these stricted constraints, there is usually an effective algorithm for generating XNF. Furthermore, when these conditions do not hold, the subgraphs over which non-deterministic backtracking must be applied is usually small enough to allow for an exhaustive search. We leave for future research precise characterizations of these claims.

In the meantime, for this paper, we let Algorithms 2.1 and 2.2 be similar in spirit to Algorithms 1.1 and 1.2,[14] and we let Algorithm 2.3 be Algorithm 2 altered (1) to generate nondeterministic threads for all possible node configurations when an edge is added to a scheme tree and (2) to select a final minimal

[14] We do not specify these exactly since we cannot prove that they lead to XNF scheme-tree forests.

scheme-tree forest from the ones nondeterministically generated as a final step in the algorithm.

Theorem 2. *Let H be a canonical CM hypergraph with no explicit generalization/specialization. Let T be a scheme tree generated by Algorithm 2. T has neither redundancy with respect to a multivalued constraint nor redundancy with respect to a functional constraint. Further, if F is a scheme-tree forest generated from H by Algorithm 2.3, then each tree in F is in* $\mathrm{XNF}_{\{\mathrm{FC,MC}\}}$. □

6 General Algorithm

In this section we further generalize our algorithms to allow CM hypergraphs with explicit generalization/specialization denoted by *ISA* triangles in CM hypergraphs. This generalization causes two new problems: (1) object sets may be redundant, and (2) ISA constructs must be considered in scheme-tree generation algorithms. We discussed and illustrated object-set redundancy in Section 2: whenever we can compute the contents of an isolated root generalization object set as explained in Example 5, we eliminate it. To handle all other ISA constructs, we collapse them into roles and thus eliminate them too. Once all ISA constructs have been eliminated, we are left with CM hypergraphs that can be processed by Algorithm 2, or by Algorithm 1 if all relationship sets happen to be binary. We call this Algorithm 3.[15]

Theorem 3. *Let H be a canonical CM hypergraph. If F is a scheme-tree forest generated from H by Algorithm 3, then each tree in F is in* $\mathrm{XNF}_{\{\mathrm{FC,MC,IC}\}}$. □

7 Concluding Remarks

We have proposed and formally defined XNF (XML Normal Form). XNF guarantees that complying XML documents have no redundant data values and have maximally compact connectivity. We have also developed conceptual-model-based algorithms (Algorithms 1.1, 1.2, 2.3, and 3) to generate DTDs to ensure that complying documents are in XNF, and we have proved that these algorithms are correct (Theorems 1–3).

We have implemented a tool to work synergistically with a user to develop XNF-compliant DTDs. Currently our tool allows users to specify CM diagrams, to convert diagrams to CM hypergraphs, to apply either Algorithms 1.1 or 1.2, or to select root nodes for scheme trees and then apply Algorithm 1. In another tool, we have implemented, users can synergistically convert CM hypergraphs into canonical hypergraphs.

As for current and future work, we need to integrate these two tools so that we can have the synergistic system we desire. We also need to implement Algorithms 2 and 3 and their variations, and we need to formally characterize

[15] See http://osm.cs.byu.edu/Papers.html for examples and further details.

the restrictions to n-ary CM hypergraphs for which there exist effective scheme-tree generation algorithms. We also intend to investigate additional heuristic variations of scheme-tree generation algorithms and synergistic development of XNF-compliant DTDs.

Acknowledgements: This material is based upon work supported by the National Science Foundation under grant No. IIS-0083127. Kimball Hewett programmed the interface we used for our implementation, Eric Carter implemented the canonicalization procedures, and Troy Walker coded the implemented algorithms described in this paper.

References

[BGH00] L. Bird, A. Goodchild, and T. Halpin. Object role modelling and xml-schema. In *Proceedings of the Ninteenth International Conference on Conceptual Modeling (ER2000)*, pages 309–322, Salt Lake City, Utah, October 2000.

[BR00] V. Bisová and K. Richta. Transformation of uml models into xml. In *Proceedings the 2000 ADBIS-DASFAA Symposium on Advances in Databases and Information Systems*, pages 33–45, Prague, Czech Republic, September 2000.

[CSF00] R. Conrad, Deiter Scheffner, and J.C. Freytag. XML conceptual modeling using UML. In *Proceedings of the Ninteenth International Conference on Conceptual Modeling (ER2000)*, Salt Lake City, Utah, October 2000. 558–571.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[Emb98] D.W. Embley. *Object Database Development: Concepts and Principles*. Addison-Wesley, Reading, Massachusetts, 1998.

[FMU82] R. Fagin, A.O. Mendelzon, and J.D. Ullman. A simplified universal relation assumption and its properties. *ACM Transactions on Database Systems*, 7(3):343–360, September 1982.

[Hal99] T. Halpin. *Conceptual Schema & Relational Database Design*. WytLytPub, revised 2nd edition, 1999.

[Ken81] W. Kent. Consequences of assuming a universal relation. *ACM Transactions on Database Systems*, 6(4):539–556, December 1981.

[MEN96] W.Y. Mok, D.W. Embley, and Y-K. Ng. A normal form for precisely characterizing redundancy in nested relations. *ACM Transactions on Database Systems*, 21(1):77–106, March 1996.

[Mok] W.Y. Mok. A comparative study of various nested normal forms. *IEEE Transactions on Knowledge and Data Engineering*. (to appear).

[XMI] Home Page for OASIS's XML Metadata Interchange (XMI). URL: http://www.oasis-open.org/cover/xmi.html.

[XML] Home Page for a listing of organizations producing industry-specific XML DTDs. URL: http://xml.org/xmlorg_registry/index.html.

[XML00] XML schema part 0: Primer: W3c working draft, 2000. URL: http://www.w3.org/TR/2000/WD-xmlschema-0-20000407/.