XSP TECHNOLOGY FOR XML SYSTEMS DESIGN & DEVELOPMENT

A Mathematical Foundation For Resolving Critical XML Design & Development Issues

ABSTRACT. The advantages of Extended Set Processing, XSP, which favors generalized data access operations over the specialized data access structures of traditional data processing technologies, is presented. It's application to the design and development of highly distributed XML based systems is studied from a formal perspective.

For an operation-centric approach, like XSP, to work, all operations must to be formally defined. For operations to be formally defined requires that their operands be formally defined. This is achieved in the paper through the use of Extended Set Theory, XST, that provides a formal means to capture the *mathematical identity* of any and all data structures. This includes both conceptual data structures and physical data structures, along with the formal expression of any transformation of them. This, in turn, provides a criteria for *strong data independence* allowing the implementation of systems with adaptive performance control. Specific attention will be given to capturing the mathematical identities of XML structures and Relational Data structures; transformations between them; and the operations required for processing heterogeneous data in highly distributed environments.

1. INTRODUCTION

The pervasiveness of the World Wide Web; the advances in processing power and storage capacity; and the universal craving for instant information are creating data access and information sharing problems that are not being adequately addressed by traditional data processing technologies.

For this reason a new approach to data access and information sharing is being initiated in the form of an Extensible Markup Language, XML, document processing standard proposed by the World Wide Web Consortium, W3C, [Wc98]. Both academics and businessmen have embraced the potential of this XML processing standard with great enthusiasm. However, the realization of this potential is not without certain implementation challenges.

Three implementation challenges are addressed by this paper. Specifically, they are to provide low-level support to high-level applications for:

1) Processing more than one XML document at a time.
2) Processing distributed XML documents.
3) Processing a mix of XML documents and legacy data together.

Without support to provide applications with all three of the above capabilities, no real distributed information sharing or data processing can be accomplished using XML documents. Not only do these three capabilities have to be implemented for general application support, but also implemented in a way that does not impede overall system performance. A resolution to these implementation issues is presented in this paper.

This presentation begins with an introduction to the XML movement, then introduces the concept of a *VPS* schema that provides an architectural abstraction of a computer platform. The *VPS* schema is then used to show the relevance of *mathematical identity* and *strong data independence* to the resolution of the above issues. Next, the concept of XSP technology is introduced to provide formal 'handles' on *mathematical identity* and *strong data independence* as they apply to systems implementation. XSP technology is then used to articulate the design, development, and operation requirements of highly distributed XML based systems supporting information sharing through parallel processing of heterogeneous data.

Copyright © 2000 INTEGRATED INFORMATION SYSTEMS ~ ANN ARBOR MI ~ iis@umich.edu

2. THE XML MOVEMENT

Though XML technologies are relatively new and the standards are still in a state of flux, many companies have already been formed or refocused to capitalize on the potential promise of XML. According to Dr. David Maier, [Ma98]:

> XML threatens to expand beyond its document markup origins to become the basis for data interchange on the Internet. Once it becomes pervasive, it's not hard to imagine that many information sources will structure their external view as a repository of XML data, no matter what their internal storage mechanisms. It is not a great leap from there to viewing the combined information on the Internet as one big distributed XML database.

If Dr. Maier is correct (and his record indicates that he probably is), the whole face of computing, as we now know it, will change dramatically. Dr. Maier's use of the word *threatens* is not ill chosen. Change is not always for the better. Will XML be a plague or a panacea? What assurance is there that the new XML processing standards will co-exist compatibly with existing data processing implementations? Will overall data processing performance improve, or will performance be dramatically worse? If internal storage mechanisms can be structurally different than the external structure, what technology will ensure data integrity when transforming from structure to structure? The promise of XML can not be willed into existence by a committee, it has to be derived into existence through a sound technology.

The title of this paper purports that XSP technology can play a prominent role in the design and development of XML based systems. For Dr. Maier's prediction of 'viewing the combined information on the Internet as one big distributed XML database' to become a reality, some seemingly intractable distributed data processing issues need to be resolved. Delineating and resolving these XML based distributed data processing issues is the focus of this paper.

2.1. XML: Plague or Panacea. The justification for introducing any new technology needs to be very compelling in order to attract interest. It is necessary to show that only with the new technology can certain advantages be achieved, and that without the new technology certain disadvantages are assured. Such is the case with XML.

There may be many reasons why XML is attractive, but there is one singularly significant reason why the approach can offer great potential advantages or can promote very serious disadvantages.

All XML DOCUMENTS ARE WELL-FORMED!

This means there are precise rules dictating a well-defined specification of both the XML document content and its structure. Unlike those of the Relational Data Model, RDM, these rules are not mathematical rules of concurrence, but are empirical rules of form. This is a mixed blessing.

This standard structural uniformity of representation certainly provides ease of cross platform data interchange, but it is not the ideal data structure representation for all potential processing needs. In point of fact, it is a less than ideal data structure representation for most processing needs. This amounts to a panacea for data sharing, but a plague for data processing.

Unfortunately, the 'plague' may already have started, [Ma98]: 'it's not hard to imagine that many information sources will structure their external view as a repository of XML data, no matter what their internal storage mechanisms.' If as Dr. Maier predicts, we may eventually be faced with 'one big distributed XML database' and if all existing data and all future data has to be structured as an XML document prior to processing, timely information access will be a 'thing of the past'.

Since XML specifications distinguish between XML-content and XML-structure, it would seem that if a new technology could provide a mathematical foundation that preserved this distinction and at the same time provided operations that transformed 'XML-structures' while preserving 'XML-content', such a technology might attract some interest. For then, all data could have any internal

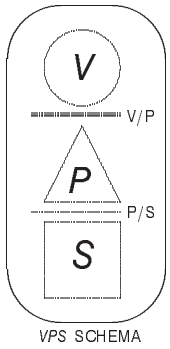
and external form desired and still be processed in an XML environment without having been coerced into an XML structure. Extended Set Processing, XSP, technology provides this form of data structure independence within and between the external and internal levels.

The key to all XSP solutions, including adaptive performance control, lies in the concept of *mathematical identity*. This is the essence of XSP technology. Since the very nature of the XSP approach is highly mathematical in nature, some mathematics is request for presenting pertinent concepts. This paper is intended to be read and understood by both practicing mathematicians and by non mathematicians. The paper will include the mathematics for completeness, but will place primary emphasis on textual explanations. It is hoped that the necessity of the mathematics will not impede an intuitive understanding of XSP technology concepts.

3. VPS SYSTEM ARCHITECTURE SCHEMA

Before launching into the particulars of XSP technology, the reasons for requiring such a technology should be well understood. With XML based systems the difficulty concerns low level implementation issues and not high level presentation issues. These issues can best be understood by relating them to specific levels within a system's architecture.

Every system platform can be thought of in terms of five components: three disjoint architectural levels, connected by two intervening interfaces, as the visual aid to the right, a *VPS* schema, depicts. Each one of these five components has a very specific role in the overall behavior of the system. The user view level, *V*, is that part of the system perceived by people. The system processing level, *P*, consists of the system's CPUs and memory. The secondary storage level, *S*, includes all of the system's immediate secondary storage devices. Mathematically, these three architectural levels are disjoint *spaces* of *operations* and *operands* that are structurally distinct from each other. These spaces are connected only through the two interfaces. The View and the Processing spaces are connected through the *V/P* interface. The Processing and Storage spaces are connected through the *P/S* interface (a.k.a. the I/O interface). It is very important to notice that everything above the *V/P* interface is *external to the machine* environment and that everything below the *V/P* interface is *internal to the machine* environment. Multiple systems would be connected through an additional *P/P* interface.



VPS SCHEMA

3.1. VPS & ANSI/SPARC. This *VPS* system architectural schema should not be confused with the ANSI/SPARC three level architecture,[Da95]. The ANSI/SPARC architecture was introduced in the late 1970s to provide a framework for describing general database concepts and for explaining the structure of specific database systems. The ANSI/SPARC schema consists of three levels: the external level, which was concerned with the way data is viewed by individual users; the conceptual level, which was concerned with the way data was viewed by the community of users; and internal level, which was concerned with how data was physically represented in the computer.

It is important to note that the RDM deals specifically with the external and conceptual levels only. To quote Date,[Da95],

> The internal level will *not* be "relational," because the objects at that level will not be just (stored) relational tables - instead, they will be the same kinds of object found at the internal level of any other kind of system (stored records, pointers, indexes, hashes, etc.). In fact, relational theory as such has *nothing whatsoever* to say about the internal level; it is concerned with how the database looks to the *user*.

The *VPS* system architectural schema is slightly different yet completely compatible with the ANSI/SPARC three level architecture. It ANSI/SPARC schema emphasizes an interest in user views of data, while the *VPS* schema is biased toward machine representations of data.

The *VPS* schema View level includes both the external and conceptual ANSI/SPARC levels, while the ANSI/SPARC internal level is separated into the Processing level and the Storage level. The schemas are functionally compatible, yet each has its own specific bias, as is evident from the different resolutions. The ANSI/SPARC schema is more concerned with the details of the View level, while the *VPS* schema is more concerned with the details of the internal level. For our purposes we will generally use the term *external level* to mean *not the internal level*. This is compatible with Dr. Maier's usage in [Ma98].

By using the *VPS* schema architectural levels of a system to provide context, many knotty issues can be untangled. For example the RDM term 'table' lives in only one of the levels, in *V*. The term 'file' also lives in only one level, in *S*. Though 'table' and 'file' are related, they do not co-exist at the same level. Other terms like 'relation', 'object', 'record', 'cylinder', 'index', 'B-tree', and 'access method' have residence at some levels with relationships to other terms at other levels. Which terms belong where? Which terms, if any, co-exist in more than one environment, but with totally different structures? How do these terms relate across interfaces? Is the relationship between these terms across interfaces data dependent or data independent?

3.2. Strong Data Independence. The term 'data dependence' was introduced in the '70s to describe the fact that an operation at one level needed to know how data was structured at a different level in order to execute correctly. The term 'data independence' describes the fact that an operation at one level *DID NOT* need to know how data was structured at a different level in order to execute correctly. The capability of data independence did not exist at either the *V/P* interface nor at the *P/S* interface until introduced at the *V/P* interface by the RDM. It should be noted that no commercial system has ever provided data independence at the *P/S* interface, yet data independence is the key to functional generality, performance optimization, and distributed data access.

When the term *data independence* is used in the context of databases, it means that *knowledge of how data is represented at the internal level* is not available to users or applications a the conceptual and external levels.

This concept, when actually applied to RDM implementations, is more of a data representation 'isolation' than one of data representation 'independence'. For if a RDM implementation were truly independent, then the form of an SQL query at the external level would have no bearing on the performance of the execution of that query at the internal level, but it does.

A different term needs to be introduced when it is important to distinguish 'data representation independence' from 'data representation isolation'. The term used will be *strong data independence*.

4. XSP TECHNOLOGY

To achieve the development of strong data independence between an two distinct environments requires any candidate technology to provide a common mathematical representation of both the data to be operated on and the operations to be performed on such data. RDM technology does this at the *V/P* interface by representing data as 'relations' with 'relational' operations.

Three tasks are now required of any new technology: define the operations and operands that need to be represented by each environment of interest; define a notation that is suitable within each environment; and choose a common body of mathematics that is rich enough to provide well-defined mathematical representations of the operations and operands for all environments of interest. This last requirement is necessary for ensuring well-defined, function preserving transformations between environments. (RDM technology fails on this last requirement, since it does not provide any formal representations for the internal level).

XSP Technology is defined to be a formal system for specifying mathematically sound operations and operands that can be executed by a digital computer. XSP Technology consists of three separate formal specifications: Theory, Notation, and Processing.

XST: Extended Set Theory - *Formal axiomatic specification of extensions to the foundations of Classical set theory that are necessary to support the modeling of computer based operations and operands.*

XSN: Extended Set Notation - *Formal set theoretic notation expressing operations and operands of XST that preserve the mathematical identity of all conceptual and computer-based structures being modeled for a system.*

XSP: Extended Set Processing - *Formal specification of a system of XSN defined operations and operands that can be executed by a computer.*

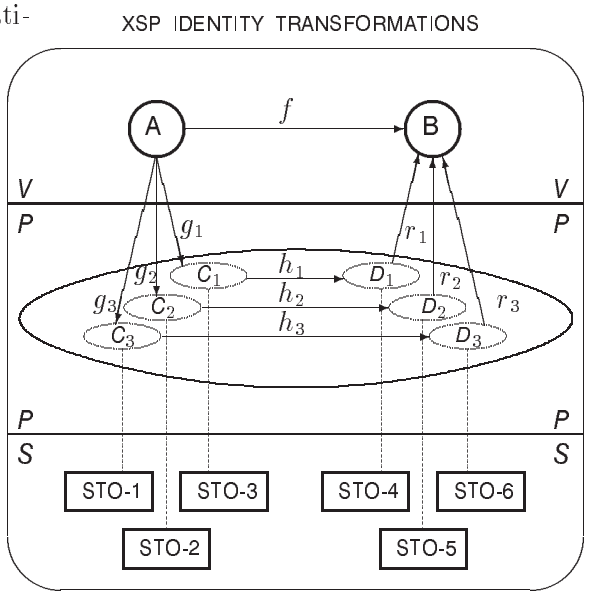
Working backwards, the final objective is an XSP implementation that supports XML document processing and provides strong data independence at the *V/P*, *P/S*, and *P/P* interfaces.

The role of XSN is to ensure that all choices of representation (linguistic, symbolic, and algorithmic) preserve the same mathematical content when represented in terms of extended sets, Xsets.

XST, of course, provides the mathematical content. If any operation, operand, or combination of operations can not be justified within the formal confines of the mathematical framework, then it can not be included in an XSP implementation. This ensures that all behavior of any XSP implementation will be mathematically sound (at least in principle).

4.1. XSP Identity Transformations:. The final objective of applying XSP technology to XML based system implementations, or any other system implementation, is a formal specification, or mathematical blueprint, of the operations and structures that produce the desired system behavior.

In terms of a *VPS* schema, this means capturing the mathematical identities of all operations and structures at each level along with the mathematical description of all transformations between levels. In the *VPS* diagram to the right: A and B represent any two collections of XML documents, or any two collections of RDM tables, or any combination of XML documents and RDM tables. What ever they are, their mathematical identities must be captured by *Xsets*. Since A and B only exist in the *V* space, the mathematical identities of A and B must be captured *as they exist* in the *V* space, not as they exist after a transformation to a *P* space. In fact, the transformation of A to an implementation friendly representation in a *P* space is modeled by g_i to C_i. While the transformation from an implementation friendly representation to B is modeled by r_i from D_i. The process f must also be defined on A and B as they exist in the *V* space. The process h_i must then be defined in the *P* space from C_i to D_i such that the combined application of processes preserve the behavior of f. More formally, the following must hold for all 'i', (with 'i' controlling performance options): *For all* $\mathbf{x}$ *in* A, $r_i(h_i(g_i(\mathbf{x}))) = f(\mathbf{x})$.



This *VPS* diagram is in the spirit of what mathematicians call a commutative diagram, except that it is more complicated. The basic mathematical concept is still the same. Whatever f does to elements of A that maps them to elements of B, then r_i on h_i on g_i on elements of A had better map to the same results in B.

Implementations based on the mapping of the mathematical identities between spaces are 'strongly data independent'. That is, no space has knowledge of how another space organizes data. This is a much stronger condition then is the RDM 'data-independence', that just isolates user applications from knowledge of how data is represented in secondary storage. As will be shown later in this paper *strong data independence* is essential for information sharing in highly distributed systems.

5. XML BASED SYSTEMS

Like the RDM, XML structures are defined as they exist in a user application and are, therefore, completely contained in the external level, the *V*-Level. Like the RDM, XML specifications makes no restrictions (though possibly suggestions) about supporting operations and structures at the internal level, the *P* and *S* levels. Unlike the RDM, which communicates between the external level and the internal level with mathematically well-defined operations on mathematically well-defined structures, the XML specification provides no means for communicating across the *V/P* interface except by passing the structured representation of the XML documents as they exist at the *V*-Level.

Thus, the XML standard *does not support any form data independence*!

This is not progress. In fact it is a through back to the pre-RDM database days where 'navigating through secondary storage' was state-of-the-art technology. Those were the days where the way data was structured in the *V*-Level was also the way data was structured in the *P*-Level, and was also the way data was structured in the *S*-Level. This data dependence between levels hindered expressive functionality and crippled high performance processing. Data dependence was alleviated by the advent of the RDM. It now appears that data dependence may be resurrected by XML.

The only way to preserve data independence in XML is by communicating through the *V/P* interface in the same mathematical manner as does the RDM, with mathematically well-defined operations on mathematically well-defined operands. There is no widely accepted technology currently available that is able to provide mathematically well-defined operations on mathematically well-defined XML documents. It has been shown,[Ch00], that even the RDM technology fails in several ways. Thus, no widely accepted technology is currently available to provide XML with strong data independence at either interface and certainly not between system platforms.

To allow development of XML to provide general functionality at the View level, optimal performance at the Processing level, adaptive data restructuring at the Storage level, and unrestricted information sharing between platforms requires a new technology that can support strong data independence, at both interfaces and between system platforms.

5.1. XML Basics:. There are generally three files that are processed by an XML-compliant application to display XML content:

The XML Document -
: This file contains the document data, typically tagged with meaningful XML elements, some of which may contain attributes.

An XSL Stylesheet -
: The stylesheet dictates how document elements should be formatted when they are displayed, whether it be in a word processor or a browser. Note that one can apply different stylesheets to the same document, depending on the environment, thus changing its appearance without affecting any of the underlying data. The separation between content and formatting is an important distinction in XML.

A DTD, Document Type Definition -
: This file specifies rules for how XML document elements, attributes, and other data are defined and logically related in an XML-compliant document.

These are the three basic file types in an XML based system. Two of them are structurally the same and the other is a degenerate case of the other two. So there is really only one XML data type. It should be noted here that the term 'data type' is not universally defined in only one way. Some definitions would allow for any variation between files to necessitate their being of different data types. That distinction may be of some value in those contexts, but here in a formal set-theoretic environment, the structural distinctions can be 'discovered' by operations and therefore files do not need to be segregated by data types, but before any mathematically well-defined operations can be developed for an XML data type, an XML data type must be identified as a mathematically well-defined operand.

All this really means is that a mathematical notation, any mathematical notation, needs to be chosen that rigorously, and completely captures all the characteristics and properties intrinsic to the definition of a well-formed XML document. If the same notation is used in exactly the same way for all three file the there will be one and only one XML data type.

An XML data type will be shown to conform to an XST definition of 'nested Xsets of n-tuples'.

5.2. XML Document Representation. The syntax of an XML document is deceptively simple. There are only three basic components: *elements*, *tags*, and *attributes*.

Every XML document is of the form:

$$
\begin{array}{l}
<\text{root}> \\
\quad <\text{tag}_1><\text{Elem}_1></\text{tag}_1> \\
\quad <\text{tag}_2><\text{Elem}_2></\text{tag}_2> \\
\qquad \vdots \\
\quad <\text{tag}_n><\text{Elem}_n></\text{tag}_n> \\
</\text{root}>
\end{array}
$$

Where each Elem_i can be of the form:

$$
\begin{array}{l}
<\text{Elem}_i> \\
\quad <\text{tag}_1><\text{elem}_1></\text{tag}_1> \\
\quad <\text{tag}_2><\text{elem}_2></\text{tag}_2> \\
\qquad \vdots \\
\quad <\text{tag}_n><\text{elem}_n></\text{tag}_n> \\
</\text{Elem}_i>
\end{array}
$$

Giving an XML document the ability to have an arbitrary depth of nested collections of tagged elements.

$$
\begin{array}{l}
<\text{root}> \\
\quad <\text{tag}_1> \\
\qquad <\text{Elem}_i> \\
\qquad\quad <\text{tag}_1><\text{elem}_1></\text{tag}_1> \\
\qquad\quad <\text{tag}_2><\text{elem}_2></\text{tag}_2> \\
\qquad\quad <\text{tag}_n><\text{elem}_n></\text{tag}_n> \\
\qquad </\text{Elem}_i> \\
\quad </\text{tag}_1> \\
\quad <\text{tag}_2><\text{Elem}_2></\text{tag}_2> \\
\qquad \vdots \\
\quad <\text{tag}_n><\text{Elem}_n></\text{tag}_n> \\
</\text{root}>
\end{array}
$$

The key to the richness of XML structures is in this nesting of collections of tagged elements. These collections can be 'ordered' or 'unordered' and each with an arbitrary, but finite, depth.

The 'unordered' or 'ordered' issue is not universally agreed upon just yet. For example:

```
<A>                          <A>
    <B>aaa</B>                   <C>ccc</C>
    <C>bbb</C>                   <C>bbb</C>
    <C>ccc</C>                   <B>aaa</B>
</A>                         </A>
```

the two XML documents above may or may not be considered to be the 'same'. They are certainly *informationally equivalent* as far as the content within the tags is concerned, but the relationship of the tagged elements, within the A tag, is not preserved. Ideally, it would be nice to have a formal notation that would be able to express these distinctions in terms of their mathematical identities..

5.3. Mathematical Identity. The mathematical identity of any object is just a formal expression that captures everything that is 'meaningful' about the content and structure of the object. For modeling data processing systems, the mathematical notation used must be rich enough to capture the mathematical identity of data. The current candidate for this task is XST.

This means that for any data representation, like an XML document or a RDM table, there exists an XST expression of the data representation that preserves everything 'meaningful' about the content and structure of the data representation. Our current objective is to capture the mathematical identity of XML documents, then mathematical identity of RDM tables and relations.

As was mentioned earlier, every XML document must be *well-formed*. That means that every XML document must conform to the notational and structural rules for constructing XML documents. The examples above reflect these rules. The question now becomes: "Can these notational and structural rules be transformed to a formal mathematical discipline?" If so, then the mathematical identity of an XML document will have been captured.

The concept of *well-formed* comes from mathematics. Even though well-defined mathematical symbols are used in an expressing, that expression may not be well-formed. None of the following expressions are well-formed:

1) $7 + 9 * 3$	4) $\sqrt{4} + \sqrt{4}$
2) $*8(/+ = 5 < 7$	5) $< a, b > \cap < a, x >$
3) $\{2, -2\} + \{2, -2\}$	6) $< a, b, c > \cap < a, x >$

Some of the above are not well-formed because the notation is ambiguous and some are not well-formed because the structure does not reflect sound mathematical behavior.

In case (1) the mathematical identity of the expression is either 48 or 34 depending on the order in which the operations are executed. The ad hoc committee way to solve the problem is to establish rules of precedence that establish one operation as having to be executed before another. The proper mathematical resolution is the use of parentheses giving either $(7 + 9) * 3$ or $7 + (9 * 3)$.

In case (2) any possibly intended mathematical identity is a complete mystery.

Case (3) and (4) are equivalent transformations of each other and therefore represent the same mathematical identity. Though case (4) is the more familiar expression and easier to resolve by ad hoc intervention, case (3) shows that case (4) does not have an obvious mathematical resolution. Standard practice is to ignore the original expression of case (4) and change it to one that can be resolved like: $|\sqrt{4}| + |\sqrt{4}|$. This, of course, now gives the modified case (4) a totally different mathematical identity than that of case (3).

Case (5) and (6) both have a well-defined mathematical identities, $< a, a >$ and $\emptyset$ respectively. Both of which are mathematically useless, therefore one never sees set-operations defined between n-tuples, even though n-tuples are well-defined sets. This 'non-well-formedness' of n-tuples is germane to resolving the mathematical identities of XML documents and will be discussed further.

5.4. XML Documents & n-tuples. Since the last section just denigrated the well-formedness of n-tuples, it would not seem to follow that n-tuples be used to capture the mathematical identity of XML documents. Yet that is what we are about to do.

N-tuples are known to be 'notationally nice and mathematically mean'. We will take advantage of the 'nice' part and tame the 'mean' part later.

The 'nice' part allows us to take a tagged element expression like: <C>ccc</C>
and transform it to an n-tuple notation by: $< C, ccc >$.

A list of tagged element expression like:
<B>aaa</B>
<C>bbb</C>
<C>ccc</C>

Could be represented either as a set of n-tuples like: $\{ < B, aaa >, < C, bbb >, < C, ccc > \}$

or as an n-tuple of n-tuples like: $\ll < B, aaa >, < C, bbb >, < C, ccc > \gg$.

Though this last use of the n-tuple notation is mathematically 'mean', it does allow a notational distinction between the previous list and the following list of tagged element expressions:

<C>ccc</C>
<C>bbb</C>
<B>aaa</B>

as expressed by an n-tuple of n-tuples like: $\ll < C, ccc >, < C, bbb >, < B, aaa > \gg$.

It is not a big leap of faith that allows:

<A>
<B>aaa</B>
<C>bbb</C>
<C>ccc</C>
</A>

to be transformed into either: <A>
$\{ < B, aaa >, < C, bbb >, < C, ccc > \}$
</A>

or into: <A>
$\ll < B, aaa >, < C, bbb >, < C, ccc > \gg$.
</A>

The original XML document can now be expressed in two different ways:

as: $\ll A, \{ < B, aaa >, < C, bbb >, < C, ccc > \} \gg$

or as: $\ll A, \ll < B, aaa >, < C, bbb >, < C, ccc > \gg \gg$

It might seem from the above expression that the mathematical identity of an XML document had been captured. Especially since there exists a well-defined transformation from the above notation back to the original XML document expression, which was well-formed to start with.

Notational isomorphism does not insure the capturing of mathematical identity. Each notation has to be well-formed in its host context. N-tuples are not unambiguously defined in classical set theory, therefore the expression in the context of classical set theory, is not well-formed.

If the n-tuple notation is used in the context of a notational 'short hand', then the expression is well-formed, but no mathematical identity has been captured since there is no mathematical context. It gets worse.

The last formulation is an expression of 'nested n-tuples', or 'n-tuples inside of n-tuples'. In set-theoretic parlance this means 'an n-tuple is a member of another n-tuple'. 'Membership' is what set theory is all about. The problem is that the membership condition for n-tuples is not all that well defined in Classical set theory, CST.

5.5. A Taste of Set Theory. This paper is not intended as treatise on set theory, but if previous attacks on the virility of Classical set theory are not substantiated and subsequently resolved, there will be no support for considering an extended set theory for capturing the mathematical identity of XML documents. The supporting arguments for n-tuple membership ambiguity and the extensions to CST provided by Extended Set Theory, XST, can be found in [CH00]. (The necessary formulations required for this paper can be found in the appendix).

Briefly, when attempting to model data and operations on data in a formal way, the CST concept of *n-tuple* is generally used to formally support the concept of a record and operations on records. Unfortunately, the CST definition of an *n-tuple* does not allow it to behave in a meaningful way when operated on by legitimate set operations. This makes formal models of record processing systems, based on the CST *n-tuple*, either very limited or 'mathematically unsound', or both.

5.6. XSN: Extended Set Notation. Extended Set Notation (XSN) denotes the extended membership concept of XST by using the construct of 'exponent' to signify the 'scope' component of membership. An extended set is referred to as an *Xset* to distinguish its extended membership condition from the CST set membership. An Xset can be thought of as a regular classical set of elements, except that the elements also have 'exponents'. For example: $\mathbf{A} = \{x, y, z\}$ is a classical set, while $\mathbf{B} = \{x^A, x^B, y^{\{a,b\}}, z^6\}$ is an Xset. These 'exponents' are called *scopes*.

It is important to notice that XST subsumes CST. Any CST set can be immediately 'converted' to a Xset just by giving every element a scope of $\emptyset$. Thus the classical set $\mathbf{A} = \{x, \{y\}, \{\{z\}\}\}$ becomes the Xset $\mathbf{A} = \{x^{\emptyset}, \{y^{\emptyset}\}^{\emptyset}, \{\{z^{\emptyset}\}^{\emptyset}\}^{\emptyset}\}$.

Not only does XSN preserve all non-ambiguous set-theoretic definitions from CST, but it also provides a resolution to the n-tuple definition, giving: $< x_1,\ x_2, ...,\ x_n > \equiv\ \{\ x_1^1,\ x_2^2, ...,\ x_n^n\ \}$. The following, though possibly bizarre, are legitimate set expressions using XSN:

$$\left\{\{a^{\{b^{\{c^d\}}\}}\}\right\}, \quad \left\{\{A^z, \{B^y, \{C^x, D^w\}^v\}^u\}^t\right\}, \quad \left\{< a, < b >, \{c, d\} >^{<x,y>},\ << e >>^{\{\{f^s\}^Q\}}\right\}.$$

In viewing the above, it may help to remember that a formal modeling notation, of and by itself, is intrinsically meaningless. The notation is purely a syntactical framework for defining consistency of operations on well-defined objects. Any 'meaningful' situation modeled by XSN will enjoy this behavioral consistency, but not because of any inherent 'meaningfulness' in the definitions.

5.7. XML Documents As Xsets. In the context of XST, the following expressions are now well-formed:

The set of n-tuples: $\{ <\ B,\ aaa\ >, <\ C,\ bbb\ >, <\ C,\ ccc\ > \}$

An n-tuple of n-tuples: $\mathbf{<}<\ B,\ aaa\ >, <\ C,\ bbb\ >, <\ C,\ ccc\ >\mathbf{>}$.

The XSN expressions of the above as Xsets are: $\left\{\ \{\ B^1,\ aaa^2\ \},\ \{\ C^1,\ bbb^2\ \},\ \{\ C^1,\ ccc^2\ \}\ \right\}$

and: $\left\{\ \{\ B^1,\ aaa^2\ \}^1,\ \{\ C^1,\ bbb^2\ \}^2,\ \{\ C^1,\ ccc^2\ \}^3\ \right\}$

The original XML document can now be expressed as two different Xsets:

as: $\left\{\ A^1,\ \{\ \{\ B^1,\ aaa^2\ \},\ \{\ C^1,\ bbb^2\ \},\ \{\ C^1,\ ccc^2\ \}\ \}^2\ \right\}$

or as: $\left\{\ A^1,\ \{\ \{\ B^1,\ aaa^2\ \}^1,\ \{\ C^1,\ bbb^2\ \}^2,\ \{\ C^1,\ ccc^2\ \}^3\ \}^2\ \right\}$

and even as: $\left\{\ A^1,\ \{\ \{\ C^1,\ ccc^2\ \}^1,\ \{\ C^1,\ bbb^2\ \}^2,\ \{\ B^1,\ aaa^2\ \}^3\ \}^2\ \right\}$

Which one, if any, captures the mathematical identity of the original XML document? The last two reflect an ordering distinction on the elements as presented earlier.

These are not the only choices. In the context of XST, a possibly more representative reflection of the mathematical identity of an XML document can be achieved by equating *tags* with *scopes*, as in: $\left\{ \{ aaa^{B},\ bbb^{C},\ ccc^{C} \}^{A} \right\}$ or by: $\left\{ \{ aaa^{<1,B>},\ bbb^{<2,C>},\ ccc^{<3,C>} \}^{<1,A>} \right\}$.

5.8. RDM Tables & Relations As Xsets. All the details of capturing the mathematical identity of XML structures have not established yet. But, since part of our goal is to be able to mix RDM data and XML data together in the same process, it may be a good time to introduce the mathematical identity of RDM tables and relations.

Again, since the mathematical identity of any object is just a formal expression that captures everything that is 'meaningful' about the content and structure of the object, all that needs to be done is to show that XSN is rich enough to capture everything we need to know about RDM tables and relations. Below are twelve RDM tables:

$$\mathbf{T}_1 = \begin{array}{|c|c|c|}\hline A & B & C \\ \hline a & b & c \\ \hline x & y & z \\ \hline \end{array} \quad \mathbf{T}_2 = \begin{array}{|c|c|c|}\hline A & B & C \\ \hline x & y & z \\ \hline a & b & c \\ \hline \end{array} \quad \mathbf{T}_3 = \begin{array}{|c|c|c|}\hline A & C & B \\ \hline a & c & b \\ \hline x & z & y \\ \hline \end{array} \quad \mathbf{T}_4 = \begin{array}{|c|c|c|}\hline A & C & B \\ \hline x & z & y \\ \hline a & c & b \\ \hline \end{array} \quad \mathbf{T}_5 = \begin{array}{|c|c|c|}\hline B & A & C \\ \hline b & a & c \\ \hline y & x & z \\ \hline \end{array} \quad \mathbf{T}_6 = \begin{array}{|c|c|c|}\hline B & A & C \\ \hline y & x & z \\ \hline b & a & c \\ \hline \end{array}$$

$$\mathbf{T}_7 = \begin{array}{|c|c|c|}\hline B & C & A \\ \hline b & c & a \\ \hline y & z & x \\ \hline \end{array} \quad \mathbf{T}_8 = \begin{array}{|c|c|c|}\hline B & C & A \\ \hline y & z & x \\ \hline b & c & a \\ \hline \end{array} \quad \mathbf{T}_9 = \begin{array}{|c|c|c|}\hline C & A & B \\ \hline c & a & b \\ \hline z & x & y \\ \hline \end{array} \quad \mathbf{T}_{10} = \begin{array}{|c|c|c|}\hline C & A & B \\ \hline z & x & y \\ \hline c & a & b \\ \hline \end{array} \quad \mathbf{T}_{11} = \begin{array}{|c|c|c|}\hline C & B & A \\ \hline c & b & a \\ \hline z & y & x \\ \hline \end{array} \quad \mathbf{T}_{12} = \begin{array}{|c|c|c|}\hline C & B & A \\ \hline z & y & x \\ \hline c & b & a \\ \hline \end{array}$$

All twelve of the above are distinct RDM-tables that represent the same unique RDM-relation. Therefore, we need to define twelve tables, one relation, and one transformation that takes any one of the twelve tables into the same relation. Doing this will also provide some experience before tackling the more complex transformations of XML structures.

For modeling any item both the content and the structure of the item has to be captured by the mathematical identity. The modeling strategy using Xsets in the following is to let the *content* be represented by *elements* and the *structure* be captured by the *scopes*. Let $\mathbf{T}_1$ and $\mathbf{T}_{12}$ be defined as follows:

$$\mathbf{T}_1 = \begin{array}{|c|c|c|}\hline A & B & C \\ \hline a & b & c \\ \hline x & y & z \\ \hline \end{array} \qquad \mathbf{T}_1 \ = \ \left\{ \{a^{<1,A>}, b^{<2,B>}, c^{<3,C>}\}^{1}, \{x^{<1,A>}, y^{<2,B>}, z^{<3,C>}\}^{2} \right\}$$

$$\mathbf{T}_{12} = \begin{array}{|c|c|c|}\hline C & B & A \\ \hline z & y & x \\ \hline c & b & a \\ \hline \end{array} \qquad \mathbf{T}_{12} \ = \ \left\{ \{z^{<1,C>}, y^{<2,B>}, x^{<3,A>}\}^{1}, \{c^{<1,C>}, b^{<2,B>}, a^{<3,A>}\}^{2} \right\}$$

$\mathbf{T}_1$ and $\mathbf{T}_{12}$ both represent RDM tables as 2 by 3 matrices with *rows* being represented as Xsets whose scopes represent the order of the row in the matrix. The elements with in the row-Xset represent the content of the matrix, with their scopes representing their *column* position along with the RDM *domain name* associated with the column.

$\mathbf{T}_1$ and $\mathbf{T}_{12}$ are just two RDM tables out of twelve. The mathematical identity of all twelve need to show common content and separate structure. Any one of the twelve can be represented by the following definition:

Definition 5.1. RDM Table:

$$\mathcal{R}dmT(i,j,k,I,J) \ = \ \left\{ \{a^{<i,A>}, b^{<j,B>}, c^{<k,C>}\}^{I}, \{x^{<i,A>}, y^{<j,B>}, z^{<k,C>}\}^{J} \right\}$$

Given this definition, $\mathbf{T}_1 = \mathcal{R}dmT(1,2,3,1,2)$ and $\mathbf{T}_{12} = \mathcal{R}dmT(3,2,1,2,1)$. All the others have equally obvious substitutions. This establishes a unique mathematical identity for the twelve distinct RDM tables. The mathematical identity of the common underlying RDM relation can be expressed by: $\mathbf{R} = \Big\{ \{a^{<0,A>}, b^{<0,B>}, c^{<0,C>}\}^0, \{x^{<0,A>}, y^{<0,B>}, z^{<0,C>}\}^0 \Big\}$, which can easily be transformed into $\mathbf{R}' = \Big\{\{z^C, y^B, x^A\},\ \{a^A, c^C, b^B\}\Big\}$.

All twelve RDM tables and the underlying RDM relation now have Xset representations of their mathematical identities. All that is left is to define a transformation from tables to the unique underlying relation. These are not general definitions, but are specific to this example.

Definition 5.2. Table to Relation: $\mathcal{T}R\Big(\mathcal{R}dmT(i,j,k,I,J)\Big) = \mathcal{R}dmT(0,0,0,0,0)$

This formally asserts that distinct, but *informationally equivalent* RDM-tables can be mathematically transformed into their underlying unique RDM-relation. This is an example of using extended set notation to capture both the representation and content of data. The question now is whether or not the same can be done for XML documents.

5.9. Tags As Scope Sets. As was mentioned earlier, there is a third component: *attributes*. The syntax of an XML document allows properties of the elements to be expressed in terms of attributes, but the attributes themselves actually appear as a component of the tags.

Attributes in the XML sense are not the same as attributes in the RDM sense. Tags are closer to playing the role of RDM-attributes while XML-attributes are more akin to 'properties' of elements.

The XML structure on the right is stylized from [Ma00]:

```
<P>
   <N L="f">EL1</N>
   <R C="u">EL2</R>
   <A F="x" L="f">
      <S>EL3</S><Z>EL4</Z>
      <I>EL5</I><Y>EL6</Y>
   </A>
</P>
```

As an Xset, it might be expressed as:

$$
\begin{array}{l}
\{ \\
\quad \{ \\
\qquad \{EL1\}^{<1,N,\{<L,\text{“}f\text{”}>\}>}, \\
\qquad \{EL2\}^{<2,R,\{<C,\text{“}u\text{”}>\}>}, \\
\qquad \{ \\
\qquad\quad \{EL3\}^{<1,S>}, \\
\qquad\quad \{EL4\}^{<1,Z>}, \\
\qquad\quad \{EL5\}^{<1,I>}, \\
\qquad\quad \{EL6\}^{<1,Y>} \\
\qquad \}^{<3,A,\{<F,\text{“}x\text{”}>,<L,\text{“}f\text{”}>\}>} \\
\quad \}^{<1,P>} \\
\}
\end{array}
$$

Notice that the XML-attributes within a tag form an unordered set. Therefore the following two sets are identical: $\Big\{ \{ \mathbf{X}\}^{<3,A,\{<F,\text{“}x\text{”}>,\ <L,\text{“}f\text{”}>\}>} \Big\} = \Big\{ \{ \mathbf{X}\}^{<3,A,\{<L,\text{“}f\text{”}>,\ <F,\text{“}x\text{”}>\}>} \Big\}$.

Though this previous demonstration is not conclusive evidence that every XML structure has at least one Xset representation that captures its mathematical identity, it can be shown that such is indeed the case. The important issue there is that if no XML structure could be faithfully represented as a mathematical operand, then no mathematical operations could be defined for processing any XML structure. Thus, there would be no mathematical basis for supporting the processing of data, which was the whole justification for introducing XSP technology in the first place.

Now that the mathematical identity of data has been demonstrated (at least in part), how does that help with improving information sharing, distributed data access, and overall system performance control in a combined XML and RDM environment?

6. FORMAL MODELING

So far, it has been demonstrated that XML documents can be formally modeled as XML-Xsets, but it has not yet been shown that any operations on XML-Xsets can be formally modeled. The most immediate issue is to determine if any operations can be found that work on XML-Xsets as they exist in the *V* environment. This is equivalent to finding an example of an A, B and f from the *VPS* schema. If that fails, the paper ends. If successful, then two issues arise. Do f-operations exist that allow the mixing of RDM-Xsets and XML-Xsets? Do g-operations exist with images in the *P* environment that preserve the mathematical identities of Xsets in the *V* environment? Natural candidates for f-operations in the *V* environment would be those of the RDM Relational Algebra. No obvious candidates exist for g-operations that map from *V* to *P*.

The Relational Algebra of the RDM, along with some secular operations, certainly provides all the necessary functionality required for processing RDM relations, but do Relational Algebra operations work on XML-Xsets? Answer: not really. In fact, Relational Algebra operations do not even work well on RDM defined relations and tables, for the RDM model (as much as it is mathematically revered) does not provide a mathematically sound foundation for relations, tables, and the Relational Algebra. (This is counter to popular belief and will be shown in detail later.)

Not only do operations have to be defined for XML-Xsets, but now RDM-Xsets have to be given a mathematically sound definition, then the Relational Algebra has to be extended and made mathematically sound also. This RDM revamping will come later. The issue at hand is still to establish an example that f-operations exist in the *V* environment.

6.1. Modeling XML & RDM Operands. Unlike RDM tables which, to users, are structureless (rows and columns have no fixed horizontal nor vertical orientation), XML documents are highly structured combinations of 'order' and 'nesting'. In one case the Xset surrogate must preserve the lack of structure, and in the other case must preserve a high degree of structure. Both cases are challenging for a formal modeling, since the most likely candidate, classical set theory, fails miserably.

The formal underpinnings of the RDM rely on the set theoretic concept of n-tuple. These n-tuples are very fragile constructs. They can be manipulated as a whole, but they mathematically disintegrate when operated on. The RDM relies on operations on n-tuples.

So far, researchers have had sense enough not to try to formally model XML documents and operations on these documents. Graph theory is considered adequate, since all XML documents can be spread out as tree structures and elements can be accessed by chasing up and down branches. Though these structures have a different mathematical identity that the original XML documents, they have the advantage of capturing the XML data so that it can be implemented and processed in a machine environment. Unfortunately, this approach does not produce either the f-operations nor the g-operations that are required for supporting strong data independence.

The converse is not true. A formal modeling that supports strong data independence could have a g-operation that takes arbitrary XML documents and produces the friendly tree structure as a C_i in the *P* environment. This would allow branch scrambling when desirable, but would also allow morphing the trees into other forms that had better mechanisms for data processing.

Though 'mathematical identity' has been presented as the intrinsic essence of XSP technology, very little mathematics has been employed up until now to support any functional modeling or data independence claims. That is about to change.

6.2. XML-Xset Query Operations. It has been previously argued that, before proceeding much further, at least one example of an A, f, and B (from the *VPS* schema) is required, where A and B are XML-Xsets. There is little value in an example where A and B are RDM-Xsets, since

those are basic to the RDM. Now, when it comes to an example of A, g, and C_i with A as an RDM-Xset, the RDM is of no value and the problem becomes quite interesting.

Two XML-Xsets are required for our pivotal example. Since the W3C committee is the recognized authority for guiding XML standardization, the following example is taken from the W3C XML QUERY REQUIREMENTS Working Draft 15 August 2000: Use Cases for XML Queries, B.1.2 (see Appendix B).

$$
\begin{aligned}
\mathsf{A} = \{ & \\
& \{ \\
& \quad \{ \\
& \quad\quad \{\ TCP/IP\ \}^{<1,Title>}, \\
& \quad\quad \{\ \{Stevens\}^{<1,Last>},\ \{W.\}^{<2,First>}\ \}^{<2,Author>}, \\
& \quad\quad \{\ Addison\text{-}Wesley\ \}^{<3,Publisher>}, \\
& \quad\quad \{\ 69.95\}^{<4,Price>} \\
& \quad \}^{<1,Book,\{<year,\text{“}1994\text{”}>\}>}, \\
& \quad \{ \\
& \quad\quad \{\ UNIX\ \}^{<1,Title>}, \\
& \quad\quad \{\ \{Stevens\}^{<1,Last>},\ \{W.\}^{<2,First>}\ \}^{<2,Author>}, \\
& \quad\quad \{\ Addison\text{-}Wesley\ \}^{<3,Publisher>}, \\
& \quad\quad \{\ 69.95\}^{<4,Price>} \\
& \quad \}^{<2,Book,\{<year,\text{“}1992\text{”}>\}>}, \\
& \quad \{ \\
& \quad\quad \{\ Data\ on\ the\ Web\ \}^{<1,Title>}, \\
& \quad\quad \{\ \{Abiteboul\}^{<1,Last>},\ \{Serge\}^{<2,First>}\ \}^{<2,Author>}, \\
& \quad\quad \{\ \{Buneman\}^{<1,Last>},\ \{Peter\}^{<2,First>}\ \}^{<3,Author>}, \\
& \quad\quad \{\ \{Suciu\}^{<1,Last>},\ \{Dan\}^{<2,First>}\ \}^{<4,Author>}, \\
& \quad\quad \{\ Morgan\ Kaufmann\ Publishers\ \}^{<5,Publisher>}, \\
& \quad\quad \{\ 39.95\}^{<6,Price>} \\
& \quad \}^{<3,Book,\{<year,\text{“}2000\text{”}>\}>}, \\
& \quad \{ \\
& \quad\quad \{\ Digital\ TV\ \}^{<1,Title>}, \\
& \quad\quad \{ \\
& \quad\quad\quad \{Gerbarg\}^{<1,Last>}, \\
& \quad\quad\quad \{Darcy\}^{<2,First>}, \\
& \quad\quad\quad \{CITI\}^{<3,Affiliation>}, \\
& \quad\quad \}^{<2,Editor>}, \\
& \quad\quad \{\ Kluwer\ Academic\ Publishers\ \}^{<3,Publisher>}, \\
& \quad\quad \{\ 129.95\}^{<4,Price>} \\
& \quad \}^{<4,Book,\{<year,\text{“}1999\text{”}>\}>} \\
& \}^{<1,Bib>} \\
\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{B} = \{ & \\
& \{ \\
& \quad \{ \\
& \quad\quad \{\ TCP/IP\ \}^{<1,Title>}, \\
& \quad \}^{<1,Book,\{<year,\text{“}1994\text{”}>\}>}, \\
& \quad \{ \\
& \quad\quad \{\ UNIX\ \}^{<1,Title>}, \\
& \quad \}^{<2,Book,\{<year,\text{“}1992\text{”}>\}>}, \\
& \}^{<1,Bib>} \\
\}
\end{aligned}
$$

In this example the above set A is the body on an XML document and B is the result of exercising the following query on A: *List all books published by Addison Wesley after 1991, including their year and title.*

This query is the f that has to be expressed strictly in terms of a set-theoretic expression that 'generates' the elements of B from the elements of A.

The real expressive power of XML comes from complex nested structurings of *elements*, *tags*, and *attributes*. These are the essential 'handles' for supporting the querying of XML documents. Though A is a very simple structure, it is not very amenable to being processed by classical set operations for several reasons. The most obvious being that CST has no way to deal directly with *tags* and *attributes*, since the only construct in CST is *elements*. 'Directly' is a key word here since a convoluted use of elements could reflect a relationship between XML *elements*, *tags*, and *attributes*.

This, however, would not be a faithful representation of the mathematical identity of the XML document.

Showing why CST does not work, does not enhance any argument for supporting that XSP does work, but it does reenforce the need for XSP operations to accommodate *tags*, and *attributes*. This means, for most readers, that an already arcane notation is about to become 'arcaner'. Following is an XST expression of membership for: $f(\mathsf{A}) = \mathsf{B}$.

Definition 6.1. W3C Query B.1.2:

$$f(\mathsf{A}) = \Big\{ y^{<1,Bib>}\colon\ (\exists z)\big(\ z \in_{<1,Bib>} \mathcal{NER}(\ \mathsf{A}, \{Addison\text{-}Wesley\}^{Publisher}\ )\ \&$$
$$y = \mathcal{NSR}\Big(\ \mathcal{NSR}(\ z, \{Book, Title\}\ ),\ \{< year, \text{“}x\text{”} >\colon\ 1991 < x \leq 2010\}\ \Big)\ \Big\}.$$

The validity of the above definition for f is probably not immediately obvious to all readers, nor is that really relevant. The important issue here is not the 'why' of its validity, but the 'fact' of its validity. (This paper is not intended as a tutorial on XST, but for the insatiably curious, with lots of time on their hands, the 'why' is explained in the appendix.)

Assuming that the above definition is indeed valid, it has been shown that a purely mathematical modeling exists that transforms one XML document into another. This all happens in the V environment. Just as all RDM modeling also resides in the V environment. The next step is to show that a g-operation exists that takes A into the P environment.

6.3. XML-Xset Xenomorphic Operations. Operations that take conceptual Xsets into physical Xsets behave like most mathematical morphisims in that they preserve the 'operational' structure of the V space, but they generally do not preserve the representational structure. The very nature of the P and S environments preclude most direct images. Just the simple mapping of the integer 10 maps to $< a, A >$ where a is the address of a location containing the hexadecimal representation of 10. So a 'true' preservation of $10 + 11$ would map to $< a, A > + < b, B >$. The operation '+', in this context, does not make a whole lot of mathematical sense.

Neither A nor B fit very nicely into a physical storage environment. The usual approach of changing them both to tree structures, and then mapping the tree structure results into physical storage, is a typical programming approach to the problem, but it is not a valid mathematical approach. A mathematical solution has to be of the form: $g(x) = y$.

Three xenomorphic g-operations will be presented for mapping A from the V space to the P space. But first, some target representation type needs to be established for the P space. The simplest and most ubiquitous candidate is the character string. In order to use the character string representation for Xsets, two notational conversions need to be defined that take character string expressions of membership expressions to membership expressions of character string expressions, and visa versa. These are they:

Definition 6.2. Character String Expression of Membership Expression:

$\mathcal{C}seme(\ \{\ x^{y}\ \}\ )\ =\ \{\texttt{x[y]}\}$.

Definition 6.3. Membership Expression of Character String Expression:

$\mathcal{M}ecse(\ \{\texttt{x[y]}\}\ )\ =\ \{\ x^{y}\ \}$.

For example:
For $\mathbf{Q} = \{\ a^{<1,A>},\ \{\ b^{<1,B>}\ \}^{<2,C>}\ \}$, $\mathcal{C}seme(\ \mathbf{Q}\ ) = \{$ `a[<1,A>], { b[<1,B>] }[<2,C>]` $\}$.
As with any character string, a convention needs to be established regarding the treatment of 'blanks', choice of ASCII or EBCDIC, choice LF and CR or just LF, choice of delimiters, and other representation particulars that do not impact the content of the character string.

Clearly, given these notational transformations, the most straight forward g-operation (isomorphic in this case) is $\mathcal{C}seme$. Let $g_1 \equiv \mathcal{C}seme$, then C_1 = Cseme(A) plops the XML-Xset A from the V space to C_1 in the P space, giving:

```
Cseme(A) = { {{{TCP/IP}[<1,Title>],
                  {{Stevens}[<1,Last>], {W.}[<2,First>]}[<2,Author>],
                  {Addison−Wesley}[<3,Publisher>],
                  {69.95}[<4,Price>]
               }[<1,Book,{<year,``1994">}>],
               {{ UNIX}[<1,Title>],
                  {{Stevens}[<1,Last>], {W.}[<2,First>]}[<2,Author>],
                  {Addison−Wesley}[<3,Publisher>],
                  {69.95}[<4,Price>]
               }[<2,Book,{<year,``1992">}>],
               {{Data on the Web}[<1,Title>],
                  {{Abiteboul}[<1,Last>], {Serge}[<2,First>]}[<2,Author>],
                  {{Buneman}[<1,Last>], {Peter}[<2,First>]}[<3,Author>],
                  {{Suciu}[<1,Last>], {Dan}[<2,First>]}[<4,Author>],
                  {Morgan Kaufmann Publishers}[<5,Publisher>],
                  {39.95}[<6,Price>]
               }[<3,Book,{<year,``2000">}>],
               {{Digital TV}[<1,Title>],
                  {{Gerbarg}[<1,Last>],
                     {Darcy}[<2,First>],
                     {CITI}[<3,Affiliation>],
                  }[<2,Editor>],
                  {Kluwer Academic Publishers}[<3,Publisher>],
                  {129.95}[<4,Price>]
               }[<4,Book,{<year,``1999">}>]
            }[<1,Bib>]
         }
```

Though the above is really just one long character string, it has been presented indented for readability.

The XML-Xset B also has a character string equivalent, D_1. If $r_1 \equiv \mathcal{M}ecse$, then $\mathsf{B} = \mathcal{M}ecse(\mathsf{D}_1)$. Using the Xop definition for $f(\mathsf{A})$, presented earlier, as a guide for defining an equivalent algorithm, h_1, then: *For all* $\mathbf{x}$ *in* A, $r_1(h_1(g_1(\mathbf{a}))) = f(\mathbf{x})$. Which demonstrates an implementation instance of $f(\mathsf{A}) = \mathsf{B}$. This, however, may not be an overall ideal implementation. Thought the physical Xset representations in P and S are conceptually clean, the algorithm for h_1 may not be very efficient and even if it were, it is certainly not very general.

A more desirable implementation would be one that had a 'flater' physical representation for physical Xsets, so that a more general family of h_i operations could be implemented. This then requires more complicated g_i and r_i operations that can transform arbitrarily complex XML-Xsets in the V space to and from non-nested Xsets in the P space. One such g operation is represented by the following:

Let $\mathbf{Q} = \{a^A,\ \{b^B, w^D\}^C,\ \{\{d^D, w^D\}^E\}^F\ \}$.
Define $g(\mathbf{Q})$ such that $g(\mathbf{Q}) = \{\ a^{<A>},\ b^{<C,B>},\ w^{<C,D>},\ d^{<F,E,D>},\ w^{<F,E,D>}\ \}$.

This xenomorphic operation took the nested Xset $\mathbf{Q}$ and converted it to a non-nested Xset with every element being atomic and with new scope components preserving the 'nested' information. No nesting information was lost since the operation is reversible. This was a pretty simple case, that could be done by hand. What is required is a set-theoretic operation that will perform this 'de-nesting' for any possible XML-Xset. Here is that operation:

Definition 6.4. Element Expansion

$$\mathcal{E}x(\mathbf{Q}) = \Big\{ y^z : tup(z) \;\&\; (\exists w_1)(w_1 \in_{\rho_1(z)} \mathbf{Q}) \;\&\; (\exists w_j)\Big((1 < j \leq \#(z)) \;\&\; (w_j \in_{\rho_j(z)} w_{j-1}) \;\&\; \Big(j = \#(z) \longrightarrow (\forall x, s)(x \notin_s w_j) \;\&\; (y = w_j) \Big) \Big) \Big\}.$$

Again, this definition is presented as a testament to its existence and not to its self-explanatory nature. The underlying concept is simpler than the notation belies. Let $g_2(\mathbf{Q}) \equiv \mathcal{C}seme(\mathcal{E}x(\mathbf{Q}))$, then

$\mathcal{E}x(\mathsf{A}) = \{$
$\{Abiteboul\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<2,Author>,<1,Last>>}$,
$\{Addison\text{-}Wesley\}^{<<1,Bib>,<1,Book,\{<year,\text{“}1994\text{”}>\}>,<3,Publisher>>}$,
$\{Addison\text{-}Wesley\}^{<<1,Bib>,<2,Book,\{<year,\text{“}1992\text{”}>\}>,<3,Publisher>>}$,
$\{Buneman\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<3,Author>,<1,Last>>}$,
$\{CITI\}^{<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<2,Editor>,<3,Affiliation>>}$,
$\{Dan\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<4,Author>,<2,First>>}$,
$\{Darcy\}^{[<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<2,Editor>,<2,First>>}$,
$\{DataontheWeb\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<1,Title>>}$,
$\{DigitalTV\}^{<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<1,Title>>}$,
$\{Gerbarg\}^{<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<2,Editor>,<1,Last>>}$,
$\{KluwerAcademicPublishers\}^{<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<3,Publisher>>}$,
$\{MorganKaufmannPublishers\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<5,Publisher>>}$,
$\{Peter\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<3,Author>,<2,First>>}$,
$\{Serge\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<2,Author>,<2,First>>}$,
$\{Stevens\}^{<<1,Bib>,<1,Book,\{<year,\text{“}1994\text{”}>\}>,<2,Author>,<1,Last>>}$,
$\{Stevens\}^{<<1,Bib>,<2,Book,\{<year,\text{“}1992\text{”}>\}>,<2,Author>,<1,Last>>}$,
$\{Suciu\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<4,Author>,<1,Last>>}$,
$\{TCP/IP\}^{<<1,Bib>,<1,Book,\{<year,\text{“}1994\text{”}>\}>,<1,Title>>}$,
$\{UNIX\}^{<<1,Bib>,<2,Book,\{<year,\text{“}1992\text{”}>\}>,<1,Title>>}$,
$\{W.\}^{<<1,Bib>,<1,Book,\{<year,\text{“}1994\text{”}>\}>,<2,Author>,<2,First>>}$,
$\{W.\}^{<<1,Bib>,<2,Book,\{<year,\text{“}1992\text{”}>\}>,<2,Author>,<2,First>>}$,
$\{39.95\}^{<<1,Bib>,<3,Book,\{<year,\text{“}2000\text{”}>\}>,<6,Price>>}$,
$\{69.95\}^{<<1,Bib>,<1,Book,\{<year,\text{“}1994\text{”}>\}>,<4,Price>>}$,
$\{69.95\}^{<<1,Bib>,<2,Book,\{<year,\text{“}1992\text{”}>\}>,<4,Price>>}$,
$\{129.95\}^{<<1,Bib>,<4,Book,\{<year,\text{“}1999\text{”}>\}>,<4,Price>>}$
$\}$

$g_2(\mathsf{A}) = \{$
{Abiteboul}[< <1,Bib>, <3,Book,{<year,``2000">}>, <2,Author>, <1,Last> >],
{Addison−Wesley}[< <1,Bib>, <1,Book,{<year,``1994">}>, <3,Publisher> >],
{Addison−Wesley}[< <1,Bib>, <2,Book,{<year,``1992">}>, <3,Publisher> >],
{Buneman}[< <1,Bib>, <3,Book,{<year,``2000">}>, <3,Author>, <1,Last> >],
{CITI}[< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>, <3,Affiliation> >],
{Dan}[< <1,Bib>, <3,Book,{<year,``2000">}>, <4,Author>, <2,First> >],
{Darcy}[< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>, <2,First> >],
{Data on the Web}[< <1,Bib>, <3,Book,{<year,``2000">}>, <1,Title> >],
{Digital TV}[< <1,Bib>, <4,Book,{<year,``1999">}>, <1,Title> >],
{Gerbarg}[< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>, <1,Last> >],
{Kluwer Academic Publishers}[< <1,Bib>, <4,Book,{<year,``1999">}>, <3,Publisher> >],
{Morgan Kaufmann Publishers}[< <1,Bib>, <3,Book,{<year,``2000">}>, <5,Publisher> >],
{Peter}[< <1,Bib>, <3,Book,{<year,``2000">}>, <3,Author>, <2,First> >],
{Serge}[< <1,Bib>, <3,Book,{<year,``2000">}>, <2,Author>, <2,First> >],
{Stevens}[< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>, <1,Last> >],
{Stevens}[< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>, <1,Last> >],

```
{Suciu}[< <1,Bib>, <3,Book,{<year,``2000">}>, <4,Author>, <1,Last> >],
{TCP/IP}[< <1,Bib>, <1,Book,{<year,``1994">}>, <1,Title> >],
{UNIX}[< <1,Bib>, <2,Book,{<year,``1992">}>, <1,Title> >],
{W.}[< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>, <2,First> >],
{W.}[< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>, <2,First> >],
{39.95}[< <1,Bib>, <3,Book,{<year,``2000">}>, <6,Price> >],
{69.95}[< <1,Bib>, <1,Book,{<year,``1994">}>, <4,Price> >],
{69.95}[< <1,Bib>, <2,Book,{<year,``1992">}>, <4,Price> >],
{129.95}[< <1,Bib>, <4,Book,{<year,``1999">}>, <4,Price> >]
}
```

For g_2, representation choices were made to support variable length records, as depicted above. Thus the operation $g_2(\mathsf{A})$ gives C_2 in the P environment. Thus, C_2 is just a file of variable length, ASCII records that contains all the 'structure' and 'content' information of Xset A from the V environment.

Notice that C_2 could be derived from C_1, and conversely, C_1 could be derived from C_2. The process from C_2 to A is well-defined, since g_1 is reversible. Therefore, g_2 is reversible, thus r_2 exists if D_2 is the same type as C_2. This just leaves the construction of h_2 to complete the second implementation of $f(\mathsf{A}) = \mathsf{B}$.

$h_{2a}(\mathsf{C}_2)$ =
```
{
{Addison−Wesley}[< <1,Bib>, <1,Book,{<year,``1994">}>, <3,Publisher> >],
{Addison−Wesley}[< <1,Bib>, <2,Book,{<year,``1992">}>, <3,Publisher> >],
{Stevens}[< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>, <1,Last> >],
{Stevens}[< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>, <1,Last> >],
{TCP/IP}[< <1,Bib>, <1,Book,{<year,``1994">}>, <1,Title> >],
{UNIX}[< <1,Bib>, <2,Book,{<year,``1992">}>, <1,Title> >],
{W.}[< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>, <2,First> >],
{W.}[< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>, <2,First> >],
{69.95}[< <1,Bib>, <1,Book,{<year,``1994">}>, <4,Price> >],
{69.95}[< <1,Bib>, <2,Book,{<year,``1992">}>, <4,Price> >]
}
```

$h_2(\mathsf{C}_2)$ =
```
{
{TCP/IP}[< <1,Bib>, <1,Book,{<year,``1994">}>, <1,Title> >],
{UNIX}[< <1,Bib>, <2,Book,{<year,``1992">}>, <1,Title> >],
}
```

Looking at the two results above, it can be seen that h_2 has been broken up into two parts. The first result is derived by checking that the 'element field' is equal to 'Addison−Wesley', and that the 'scope field' contained 'Publisher', and that the 'scope field' containing 'year' had a value greater than 1991. These conditions are true only for elements with scopes containing '<1,Book,{<year,"1994">} >' and '<2,Book,{<year,"1992">} >' Therefore the first result is the subset of C_2 that has elements with scopes matching these conditions.

The second result is a subset of the first result containing only elements with scopes containing 'Title'.

Clearly, h_2 is more general in nature than was h_1. In fact h_2 is almost a RDM operation except that neither C_2 nor the result D_2 satisfy conditions for RDM operands, embedded in a P space. Such conditions would include some mechanism for associating 'Domain Names' with 'columns'. Neither of which are even defined for C_2 or D_2. This suggests possibilities for a third implementation of $f(\mathsf{A}) = \mathsf{B}$.

Two approaches exist for making h_3 more RDM oriented: extend the operations of the Relational Algebra to be well behaved on operands of type C_2, or define g_3 and r_3 to produce and operate on

P based Xsets that conform to RDM specifications. (Both these conditions are difficult to achieve since the RDM only specifies conditions for V environment.) However, if a g-operation produced a P based Xset that could easily map back to a V based RDM-Xset, then all the RDM conditions of the V space could be 'pulled back' into the P space.

The following Xset, $g_{3a}(\mathsf{A})$, is just a rewriting of $g_2(\mathsf{A})$ with a judicious use of blanks to create a representation of a fixed-length record file.

```
{Abiteboul}                 [< <1,Bib>, <3,Book,{<year,``2000">}>, <2,Author>,   <1,Last>        >]
{Addison$-$Wesley}          [< <1,Bib>, <1,Book,{<year,``1994">}>, <3,Publisher>                 >]
{Addison$-$Wesley}          [< <1,Bib>, <2,Book,{<year,``1992">}>, <3,Publisher>                 >]
{Buneman}                   [< <1,Bib>, <3,Book,{<year,``2000">}>, <3,Author>,   <1,Last>        >]
{CITI}                      [< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>,   <3,Affiliation> >]
{Dan}                       [< <1,Bib>, <3,Book,{<year,``2000">}>, <4,Author>,   <2,First>       >]
{Darcy}                     [< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>,   <2,First>       >]
{Data on the Web}           [< <1,Bib>, <3,Book,{<year,``2000">}>, <1,Title>                     >]
{Digital TV}                [< <1,Bib>, <4,Book,{<year,``1999">}>, <1,Title>                     >]
{Gerbarg}                   [< <1,Bib>, <4,Book,{<year,``1999">}>, <2,Editor>,   <1,Last>        >]
{Kluwer Academic Publishers} [< <1,Bib>, <4,Book,{<year,``1999">}>, <3,Publisher>                >]
{Morgan Kaufmann Publishers} [< <1,Bib>, <3,Book,{<year,``2000">}>, <5,Publisher>                >]
{Peter}                     [< <1,Bib>, <3,Book,{<year,``2000">}>, <3,Author>,   <2,First>       >]
{Serge}                     [< <1,Bib>, <3,Book,{<year,``2000">}>, <2,Author>,   <2,First>       >]
{Stevens}                   [< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>,   <1,Last>        >]
{Stevens}                   [< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>,   <1,Last>        >]
{Suciu}                     [< <1,Bib>, <3,Book,{<year,``2000">}>, <4,Author>,   <1,Last>        >]
{TCP/IP}                    [< <1,Bib>, <1,Book,{<year,``1994">}>, <1,Title>                     >]
{UNIX}                      [< <1,Bib>, <2,Book,{<year,``1992">}>, <1,Title>                     >]
{W.}                        [< <1,Bib>, <1,Book,{<year,``1994">}>, <2,Author>,   <2,First>       >]
{W.}                        [< <1,Bib>, <2,Book,{<year,``1992">}>, <2,Author>,   <2,First>       >]
{39.95}                     [< <1,Bib>, <3,Book,{<year,``2000">}>, <6,Price>                     >]
{69.95}                     [< <1,Bib>, <1,Book,{<year,``1994">}>, <4,Price>                     >]
{69.95}                     [< <1,Bib>, <2,Book,{<year,``1992">}>, <4,Price>                     >]
{129.95}                    [< <1,Bib>, <4,Book,{<year,``1999">}>, <4,Price>                     >]
```

By shortening some element names along with other editorial modifications, Xset $g_{3a}(\mathsf{A})$ can be mapped to a RDM representation in the V environment, giving:

{ELEM}[< <D1>, <D2,D3,D4>, <D5>, <D6> >]

ELEM	D1	D2	D3	D4	D5	D6
Abiteboul	1,Bib	3,Book	year	``2000"	2,Author	1,Last
Add$-$Wes	1,Bib	1,Book	year	``1994"	3,Publis	-
Add$-$Wes	1,Bib	2,Book	year	``1992"	3,Publis	-
Buneman	1,Bib	3,Book	year	``2000"	3,Author	1,Last
CITI	1,Bib	4,Book	year	``1999"	2,Editor	3,Affn
Dan	1,Bib	3,Book	year	``2000"	4,Author	2,First
Darcy	1,Bib	4,Book	year	``1999"	2,Editor	2,First
Data Web	1,Bib	3,Book	year	``2000"	1,Title	-
Digital	1,Bib	4,Book	year	``1999"	1,Title	-
Gerbarg	1,Bib	4,Book	year	``1999"	2,Editor	1,Last
Kluwer	1,Bib	4,Book	year	``1999"	3,Publis	-
Morgan	1,Bib	3,Book	year	``2000"	5,Publis	-
Peter	1,Bib	3,Book	year	``2000"	3,Author	2,First
Serge	1,Bib	3,Book	year	``2000"	2,Author	2,First
Stevens	1,Bib	1,Book	year	``1994"	2,Author	1,Last
Stevens	1,Bib	2,Book	year	``1992"	2,Author	1,Last
Suciu	1,Bib	3,Book	year	``2000"	4,Author	1,Last
TCP/IP	1,Bib	1,Book	year	``1994"	1,Title	-
UNIX	1,Bib	2,Book	year	``1992"	1,Title	-
W.	1,Bib	1,Book	year	``1994"	2,Author	2,First
W.	1,Bib	2,Book	year	``1992"	2,Author	2,First
39.95	1,Bib	3,Book	year	``2000"	6,Price	-
69.95	1,Bib	1,Book	year	``1994"	4,Price	-
69.95	1,Bib	2,Book	year	``1992"	4,Price	-
129.95	1,Bib	4,Book	year	``1999"	4,Price	-

This RDM view in the V space provides all the information about 'domain names' and 'column' descriptions necessary to construct a $g_3(\mathsf{A})$ that maps A to a small family of Xsets in P. This family of Xsets in P provides all the information necessary to support Relational operations in the P space. Thus, h_3 represents a full compliment of RDM operations. This in turn, allows f in the V space to be any combination of RDM operations.

We have just shown how to support RDM operations on those XML documents that could be

modeled by an RDM-Xset. This may prompt some to wonder whether or not all XML documents can be operated upon by RDM operations. The question is confining, but the answer is clearly 'yes', since any XML structure can be 'flattened' into a family of Xsets that are amenable to RDM operations. This, in turn, allows existing SQL 'frontends' to process these transformed XML documents.

RDM operations, as they are currently known, are probably not the ideal collection of operations to process XML documents. RDM operations are not rich enough to deal with nested Xsets and there is no way to distinguish 'elements' from 'scopes', or in XML parlance 'content' from 'structure'.

It should not go unnoticed that modeling exercises in the last section, with the g_i, h_i, and r_i operations, were also a demonstration of *strong data independence* at the *V/P* interface since h_i knew nothing about the existence of A or B and f knew nothing about the existence of C_i or D_i.

7. CONCLUSION

There were three implementation challenges to be addressed by this paper. Specifically, they were to show how to provide low-level support to high-level applications for:

1) Processing more than one XML document at a time.
2) Processing distributed XML documents.
3) Processing a mix of XML documents and legacy data together.

This paper introduced XSP technology for using set processing operations to capture and manipulate the mathematical identity of any and all XML documents.

This was achievable because all XML documents *have a mathematical identity by definition.* It is the well-formedness of XML documents that provides the potential for high-performance distributed information sharing. It is also the well-formedness of XML documents that provides the greatest implementation hurdle for traditional data processing technologies.

Switching from 'structure-centric' implementation strategies to an 'operation-centric' implementation strategy allows the well-formedness of XML documents to become an asset instead of a liability.

This paper showed that an 'operation-centric' technology could provide implementations with 'strong data independence' which is necessary for achieving robust information sharing in a highly distributed environment.

It was also shown that RDM operations are applicable to XML documents, but are incomplete in their functional capabilities. Thus a richer calculus is needed in the *V* space in order to take full advantage of the information capacity of XML structures.

Work yet needed to be done is to discover what new Xops need to be defined to provide a 'complete' operational capability for manipulating both XML-content and XML-structure on highly distributed platforms. A good start at this would be to find a covering set of Xops that addresses all the use cases of the W3C XML QUERY REQUIREMENTS referenced in Appendix B.

Appendix A. XSETS, RSHIPS, & TUPLES

The following material is provided in the spirit of completeness, but it is not essential for understanding the intuition underlying the basic ideas of the paper.

A.1. SET MEMBERSHIP.

In Classical set theory, CST, a set $\mathbf{A}$ is defined solely in terms of the *TRUTH* or *FALSITY* of some given unary function on the variable $\mathbf{x}$, $\Gamma(\mathbf{x})$. The predicate Γ is an interpretation of the undefined condition *membership of*. Thus, if we were to define a set $\mathbf{A}$ in terms of some Γ, $\Gamma(\mathbf{x})$ would have to be either true or false for every choice of $\mathbf{x}$. That is:

If $\Gamma(\mathbf{x}) =$ **TRUE**, then $\mathbf{x}$ is an *element* of $\mathbf{A}$, and

If $\Gamma(\mathbf{x}) =$ **FALSE**, then $\mathbf{x}$ is not an *element* of $\mathbf{A}$.

These two assertions can be combined as: $\mathbf{A} = \{x : \Gamma(x)\} \iff (\forall x)(x \in \mathbf{A} \leftrightarrow \Gamma(x))$.

Membership is an 'undefined condition', and just because it may have a reasonable interpretation should not detract from its 'undefinedness'. Now let us assume a second undefined condition represented by (but not defined by) the *scope* of y. Given a binary truth-functional $\Gamma(a, b)$, which is either true or false for all instances of a and b, we can extend the definition of sets, from a dependency on just a single condition for membership, to a dependence on two conditions for membership: **element** and **scope**.

$$\mathbf{A} = \{x^y : \Gamma(x, y)\} \iff (\forall x, y)(x \in_y \mathbf{A} \leftrightarrow \Gamma(x, y)).$$

This constitutes the truth-functional condition for a set in extended set theory, XST, and indicates that $\mathbf{A}$ is a set defined by Γ such that: for all x and y, x is an *element* of $\mathbf{A}$ under some *scope* represented by y if and only if x and y satisfy the condition $\Gamma(x, y)$.

The notation '$x \in_y \mathbf{A}$' is equivalent to the expression *x is a y-element of set* $\mathbf{A}$.

Thus, somewhat simplistically stated, extended set membership extends Classical set membership from a dependency on just one logical condition to a dependency on two logical conditions. It should also be noted that Classical set membership is subsumed under extended set membership since every unary truth-functional $\Gamma(x)$, for a CST set definition, can be re-expressed as a binary truth-functional $\Gamma(x, \emptyset)$, for an XST set definition.

A.2. BASIC XST DEFINITIONS. Instead of assuming the undefined term, $\in$, to be a binary predicate of the form $x \in z$, as in classical set theories, $\in$ is assumed to be a ternary predicate of the form $x \in_y z$. To accommodate *individuals* or *atoms*, the null set, $\emptyset$, will also be assumed as an undefined term. In extended set theory the familiar notation of classical set theory may always be preserved since it can be subsumed under the extended membership predicate by: $x \in A \equiv x \in_\emptyset A$.

In the following definitions, capital letters shall be used only for sets, while lower case letters may take as their values either sets or individuals.

Definition A.1. Set: Given the null set, $\emptyset$, and the extended membership predicate, $\in$, then:

$$Y \text{ is a set} \longleftrightarrow (\exists x, s)(\ x \in_s Y\ ) \text{ or } Y = \emptyset.$$

Definition A.2. Definition Schema for a Set: $Y = \{x^s\colon\ \mathbf{\Gamma}_Y(x, s)\ \} \longleftrightarrow (\forall x, s)(\ x \in_s Y \longleftrightarrow \mathbf{\Gamma}_Y(x, s)\ )$.

Definition A.3. Membership Convention: $x \in Y \leftrightarrow (\exists s)(x \in_s Y)$,

Definition A.4. Scope Set: $\mathcal{S}(\mathbf{A}) = \{y^y\colon\ (\exists x)(\ x \in_y \mathbf{A}\ )\}$.

Definition A.5. Constituent Set: $\breve{\mathbf{A}} = \{\ x^x\colon\ (\exists y)(\ x \in_y \mathbf{A})\}$.

Definition A.6. Constituent Scope Set: $\mathcal{S}c(\mathbf{A}) = \{\ y^y\colon\ (\exists x, s)(\ x \in_s \mathbf{A}\ \&\ y \in_y \mathcal{S}(x))\}$.

Definition A.7. Set Inversion: $\widehat{\mathbf{A}} = \{\ y^x\colon\ x \in_y \mathbf{A}\ \}$.

Definition A.8. Scope Restriction:

$$\mathbf{A}^{[\sigma]} = \{\ x^s\colon\ x \in_s \mathbf{A}\ \ \&\ \ s \in \breve{\sigma}\ \},\quad \mathbf{A}^{\dagger\sigma\dagger} = \{\ x^s\colon\ x \in_s \mathbf{A}\ \ \&\ \ s \notin \breve{\sigma}\ \}.$$

Definition A.9. Scope Transform: $\mathbf{A}^{\prec\mu\succ} = \{x^t\colon\ (\exists s)(\ x \in_s \mathbf{A}\ \ \&\ \ t \in_s \mu\ \}$.

Definition A.10. Constituent Scope Transform: $\mathbf{Q}^{\lhd\mu\rhd} = \Big\{z^s\colon\ (\exists x)\Big(\ x \in_s \mathbf{Q}\ \&\ \ z = x^{\prec\mu\succ} \neq \emptyset\Big)\Big\}$.

Definition A.11. Scope Functional:

$$\mathcal{S}_f(\mathbf{Q}) \iff \mathbf{Q} \neq \emptyset \ \ \& \ \ (\forall x, y, s)\Big(\ x \in_s \mathbf{Q} \ \& \ y \in_s \mathbf{Q} \ \ \rightarrow \ \ x = y \ \Big),$$

$$\mathcal{S}_F(\mathbf{Q}) \iff \mathcal{S}_f(\mathbf{Q}) \ \& \ \mathcal{S}_f(\widehat{\mathbf{Q}}).$$

Definition A.12. Subsets:

$$\mathbf{A} \ \subseteq \ \mathbf{B} \ \longleftrightarrow \ (\forall x, s)\Big(\ x \in_s \mathbf{A} \ \rightarrow \ x \in_s \mathbf{B} \Big),$$

$$\mathbf{A} \ \subset \ \mathbf{B} \ \longleftrightarrow \ \mathbf{A} \ \subseteq \ \mathbf{B} \ \ \& \ \ \mathbf{A} \ \neq \ \mathbf{B}.$$

Definition A.13. Non-Empty Subsets:

$$\mathbf{A} \subsetdot \mathbf{B} \iff \emptyset \neq \mathbf{A} \subset \mathbf{B},$$

$$\mathbf{A} \subseteqdot \mathbf{B} \iff \mathbf{A} \subseteq \mathbf{B} \ \ \& \ \ \mathbf{B} \neq \emptyset \rightarrow \mathbf{A} \neq \emptyset.$$

Definition A.14. Unions & Intersections:

$$\mathbf{A} \ \cup \ \mathbf{B} \ = \ \Big\{ \ x^y\colon \ x \in_y \mathbf{A} \text{ or } x \in_y \mathbf{B} \ \Big\}, \ \ \mathbf{A} \ \cap \ \mathbf{B} \ = \ \Big\{ \ x^y\colon \ x \in_y \mathbf{A} \text{ and } x \in_y \mathbf{B} \ \Big\}.$$

Definition A.15. Relative Complement: $\mathbf{A} \ \sim \ \mathbf{B} = \Big\{ \ x^y\colon \ x \in_y \mathbf{A} \ \& \ x \notin_y \mathbf{B} \Big\}.$

Definition A.16. Symmetric Difference:

$$\mathbf{A} \ \triangle \ \mathbf{B} = \Big\{ \ x^y\colon \ (\ x \in_y \mathbf{A} \ \& \ x \notin_y \mathbf{B} \ ) \text{ or } \ (\ x \in_y \ \mathbf{B} \ \& \ x \notin_y \mathbf{A} \ ) \Big\}.$$

Definition A.17. Neutralize:

$$A^{*r} \ = \ \{ \ x^r\colon \ (\exists y)(\ x \in_y \mathbf{z})\}.$$

A.3. EXTENDED SET OPERATIONS.

Definition A.18. Domain: $\mathfrak{D}_\sigma(\mathbf{Q}) = \Big\{ \ x^s\colon \ (\exists z)\big(\ z \in_s \mathbf{Q}\big) \ \ \& \ \ \big(x = z^{[\sigma]} \neq \emptyset \ \big) \Big\}.$

Definition A.19. Range: $\mathfrak{R}_\tau(\mathbf{Q}) = \Big\{ \ y^s\colon \ (\exists z)\big(\ z \in_s \mathbf{Q}\big) \ \ \& \ \ \big(\ y = z^{\prec\tau\succ} \ \neq \emptyset \ \big) \Big\}.$

Definition A.20. Restriction: $\mathbf{Q}\big\|_\sigma A \ = \Big\{ \ z^s\colon \ (\exists a)\big(a \in_s A \ \ \& \ \ z \in_s \mathbf{Q}\big) \ \ \& \ \ \big(\ a^{[\sigma]} \subseteqdot z \ \big) \Big\}.$

Definition A.21. Image: $\mathbf{Q}[\![\mathbf{A}]\!]_{<\sigma,\tau>} = \Big\{ y^s\colon (\exists a, z)\big(a \in_s \mathbf{A} \ \& \ z \in_s \mathbf{Q}\big) \ \& \ \big(a^{[\sigma]} \subseteqdot z \rightarrow \ y = z^{\prec\tau\succ} \neq \emptyset\big) \Big\}.$

Example(s) A.1. Let $\mathbf{f} = \Big\{ \ \{x^a, y^b\} \ \{y^a, x^b\} \ \Big\}$, then

a) $\mathbf{f}[\![\mathbf{f}]\!]_{<\{a^a\},\{a^b\}>} = \big\{\{x^a\}, \{y^a\}\big\},$

b) $\mathbf{f}[\![\mathbf{f}[\![\mathbf{f}]\!]_{<\{a^a\},\{a^b\}>}]\!]_{<\{a^a\},\{a^b\}>} = \big\{\{x^a\}, \{y^a\}\big\},$

Theorem A.1. $\mathbf{Q}[\![\mathbf{A}]\!]_{<\sigma,\tau>} \ = \mathfrak{R}_\tau\big(\mathbf{Q}\big\|_\sigma \mathbf{A}\big).$

A.4. RSHIPS.

Definition A.22. *Rship*: An *Rship* is any set '**Q**' all of whose elements are sets, but with the single caveat that the null set does not appear as a scope element in any element of the set '**Q**'. In the following definition: $\mathcal{X}(A)$ means A is a set, not an atomic element.

$$\mathit{Rship}(\mathbf{Q}) \iff \Big[\ \mathcal{X}(\mathbf{Q}) \ \ \& \ \ \emptyset \notin \mathcal{S}c(\mathbf{Q}) \ \ \& \ \ (\forall x)\big(x \in \mathbf{Q} \ \rightarrow \ \mathcal{X}(x)\big) \ \Big] \ \ \text{or} \ \ \mathbf{Q} = \emptyset.$$

Theorem A.2. *For* $\mathit{Rship}(\mathbf{F})$ *and* $\mathit{Rship}(\mathbf{G})$*:*

a) $(\mathbf{F}/_\sigma \mathbf{G})[\![\mathbf{A}]\!]_\tau = \ \mathfrak{R}_\tau\Big(\ (\mathbf{F}/_\sigma \mathbf{G})\big\|\mathbf{A} \ \Big),$

b) $(\mathbf{F}/_\sigma \mathbf{G})[\![\mathbf{A}]\!]_\tau = \ \mathfrak{R}_\tau\Big(\ (\mathbf{F}\big\|\mathbf{A})/_\sigma \mathbf{G}\Big),$

c) $(\mathbf{F}/_\sigma \mathbf{G})[\![\mathbf{A}]\!]_\tau = \ \mathfrak{R}_\tau(\ \mathbf{G}\big\|\mathfrak{R}_\sigma(\mathbf{F}\big\|\mathbf{A}) \ ),$

d) $(\mathbf{F}/_\sigma \mathbf{G})[\![\mathbf{A}]\!]_\tau = \ \mathbf{G}[\![\mathfrak{R}_\sigma(\mathbf{F}\big\|\mathbf{A})]\!]_\tau,$

e) $(\mathbf{F}/_\sigma \mathbf{G})[\![\mathbf{A}]\!]_\tau = \ \mathbf{G}[\![\mathbf{F}[\![\mathbf{A}]\!]_\sigma]\!]_\tau.$

A.5. TUPLES & N-TUPLES.

Definition A.23. Natural Numbers: $\mathbf{N} = \{\ 1, 2, 3, 4, ...\}$ Positive Natural numbers.

Definition A.24. Natural Numbers (1 to n): $\mathbf{N}(n) \ = \ \{\ x^x : \ x \in \mathbf{N} \ \ \& \ \ x \leq n\}$.

Definition A.25. Indexed Set: A set Q is said to be indexed by a set K when:

$$Q = \{A_i^{\tau_i}\}_{i \in K} \iff (\exists A, \tau)\Big((\forall x, s)\big(x \in_s Q \rightarrow (\exists i \in K)(x = A_i \ \ \& \ \ s = \tau_i)\big) \ \ \& \ \ (\forall i \in K)\big(A_i \in_{\tau_i} Q\big)\Big).$$

This definition holds for any choice of K, but is particularly useful when $K = \mathbf{N}(n)$.

Definition A.26. n-ary Union & Intersection:

$$\begin{aligned} \textstyle\bigcup_0(\mathbf{Q}) \ &= \ \mathbf{Q}, & \textstyle\bigcap_0(\mathbf{Q}) \ &= \ \mathbf{Q}, \\ \textstyle\bigcup_1(\mathbf{Q}) \ &= \ \bigcup\bigcup_0(\mathbf{Q}), & \textstyle\bigcap_1(\mathbf{Q}) \ &= \ \bigcap\bigcap_0(\mathbf{Q}), \\ \textstyle\bigcup_{n+1}(\mathbf{Q}) \ &= \ \bigcup\bigcup_n(\mathbf{Q}), & \textstyle\bigcap_{n+1}(\mathbf{Q}) \ &= \ \bigcap\bigcap_n(\mathbf{Q}), \ \text{for } n \text{ in } \mathbf{N}. \end{aligned}$$

Theorem A.3. $Q = \bigcup\limits_{i \tilde{\in} I} \{x_i^{s_i}\} \iff Q = \{x_i^{s_i}\}_{i \in I}$.

Definition A.27. Self Indexing Sets: A set Q is self indexing $\iff$ Q is indexed by $\mathcal{S}(Q)$.

Theorem A.4. *A set Q is self indexing* $\iff$ $Q = \{Q_j^j\}_{j \in \mathcal{S}(Q)}$.

It should be noted that the concept of indexing developed above does not depend on any prior definition of 'function' nor on any prior definition of '$< x, y >$'. The role usually played by these is filled using the concept of 'Scope Functional' [A.11] which provides the required utility without constraining the yet to be defined concepts of 'function' and 'ordered pair'.

Definition A.28. Positive Scope Set:

$$pos(\mathbf{x}) = n \iff \Big(\ n > 0 \ \ \& \ \ \mathcal{S}(\mathbf{x}) \subseteq \mathbf{N}(n) \ \ \& \ \ (\exists y \in_n \mathbf{x})\Big) \text{ or } \Big(n = 0\Big).$$

Definition A.29. Unique Positive Scope Set:

$$upos(\mathbf{x}) = n \iff pos(\mathbf{x}) = n \ \ \& \ \ \Big((\forall y, z, i)\big(y \in_i \mathbf{x} \ \& \ z \in_i \mathbf{x} \rightarrow \ y = z\Big).$$

Definition A.30. Tight Positive Scope Set:

$$tpos(\mathbf{x}) = n \iff pos(\mathbf{x}) = n \ \ \& \ \ \mathcal{S}(\mathbf{x}) = \mathbf{N}(n) \ .$$

Definition A.31. n-Tuple: $tup(\mathbf{x}) = n \iff tpos(\mathbf{x}) = upos(\mathbf{x}) = n$.

Definition A.32. Tuple Set:

$$\mathcal{T}up(\mathbf{Q}) = n \iff \mathcal{S}c(\mathbf{Q}) = \mathbf{N}(n) \ \ \& \ \ (\forall x)\big(x \in \mathbf{Q} \rightarrow \ tup(x) \leq n).$$

Definition A.33. Homogeneous Tuple Set:

$$\mathcal{H}Tup(\mathbf{Q}) = n \iff \mathcal{S}c(\mathbf{Q}) = \mathbf{N}(n) \ \ \& \ \ (\forall x)\big(x \in \mathbf{Q} \rightarrow \ tup(x) = n).$$

Definition A.34. Tuple Notation:

$$< x_1, x_2, x_3, .., x_n >_\alpha \ = \ \{x_i^{s_i} \colon \ \mathcal{S}_F(\alpha) \ \ \& \ \ s_i \in_i \alpha \ \ \& \ \ \mathcal{S}(\alpha) = \mathbf{N}(n) \ \}.$$

Notice that this definition confines tuples to be finite. Though finite is not requisite, it is in keeping with traditional use and intent of tuples.

Example(s) A.2.

Given $\alpha_1 \ = \ \{A^1, \ B^2, \ C^3\}, \ \ \alpha_2 \ = \ \{B^3, \ D^2, \ X^1\}$ and $\alpha_3 = \ \{A^4, \ B^3, \ C^5, \ D^1, X^2\}$,

then $< a, \ b, \ c >_{\alpha_1} \ = \ \{a^1, \ b^2, \ c^3\}^{\prec \alpha_1 \succ} \ = \ \{a^A, \ b^B, \ c^C\}$

and $< a, \ b, \ c >_{\alpha_1} \cup < x, \ d, \ b >_{\alpha_2} = < d, \ x, \ b, \ a, \ c >_{\alpha_3}$.

Theorem A.5. *For all α such that $\mathcal{S}_F(\alpha)$,*

$$< x_1, x_2, x_3, .., x_n >_\alpha = < y_1, y_2, y_3, .., y_n >_\alpha \iff (\forall i)(i \in \mathcal{S}(\alpha) \ \ \& \ \ x_i = y_i).$$

Theorem A.6. $< x_1, x_2, x_3, .., x_n >_\alpha \rightarrow \ \alpha$ *is a n-tuple.*

Definition A.35. Convention: $< x_1, \ x_2, ..., \ x_n > \equiv < x_1, \ x_2, ..., \ x_n >_{\mathbf{N}(n)} \ = \ \{\ x_1^1, \ x_2^2, ..., \ x_n^n \ \}$.

Theorem A.7. $< x_1, x_2, x_3, .., x_n >_\alpha = < x_1, x_2, x_3, .., x_n >^{\prec \alpha \succ}$.

Theorem A.8. $< x_1, x_2, x_3, .., x_n >_\alpha \rightarrow \Big(\big(< x_1, x_2, x_3, .., x_n > ^{\prec \alpha \succ} \big) ^{\widehat{\prec \alpha \succ}} = < x_1, x_2, x_3, .., x_n > \Big).$

Definition A.36. Element Projection: $\boldsymbol{\rho}_s(x) = y \iff y \in_s x \ \& \ (\forall z)(z \in_s x) \rightarrow y = z).$

Theorem A.9. *If* $\mathbf{Q}$ *is scope functional, then* $\boldsymbol{\rho}_i(\mathbf{Q})$ *is defined for all* i *in* $\mathcal{S}(\mathbf{Q})$.

Theorem A.10. $\boldsymbol{\rho}_i(< x_1, x_2, .., x_n >) = x_i.$

Definition A.37. Concatenation:

$$\mathbf{x} \cdot \mathbf{y} = \{ w^i \colon (\exists n)(pos(\mathbf{x}) + pos(\mathbf{y}) = n > 0) \ \& \ ((w \in_i \mathbf{x}) \text{ or } (\exists j)(w \in_j \mathbf{y} \ \& \ i = j + pos(\mathbf{x}))\}.$$

Definition A.38. Set Concatenation:

$$\mathbf{Q} \times \mathbf{R} = \{ z^s \colon (\exists x, y)\big(x \in_s \mathbf{Q} \ \& \ y \in_s \mathbf{R}\big) \ \& \ \big(z = x \cdot y \neq \emptyset\big)\}.$$

Though the above operation is defined for any two arbitrary sets, $\mathbf{Q}$ and $\mathbf{R}$, the result is only non-empty when either set contains an element with positive scope sets. The operation is most interesting when both sets are Tuple sets.

Definition A.39. Scope Hierarchy: $\mathcal{S}h(\mathbf{Q}) = \Big\{ X^i \colon \ i \in \mathbf{N} \ \& \ \bigcup_{i-1}(\mathbf{Q}) \neq \emptyset \rightarrow X = \mathcal{S}(\bigcup_{i-1}(\mathbf{Q})) \ \Big\}.$

Example(s) A.3.

Given $A = \{a^1, \{b^2, w^4\}^3, \{v^2, \{d^4\}^5\}^6 \},$

then $\mathcal{S}h(A) = \{ \{1, 3, 6\}^1, \{2, 4, 5\}^2, \{4\}^3\}.$

Definition A.40. Convention:

$\mathcal{S}h_i(\mathbf{Q}) = X \rightarrow X \in_i \mathcal{S}h(\mathbf{Q}),$

$\mathcal{S}h_i(\mathbf{Q}) = \emptyset \rightarrow \neg(\exists X)(X \in_i \mathcal{S}h(\mathbf{Q})).$

Definition A.41. Rank: $\mathcal{R}(\mathbf{Q}) = \#(\mathcal{S}h(\mathbf{Q})).$

Theorem A.11. $\mathcal{R}(\mathbf{Q}) = i \longrightarrow \bigcup_i(\mathbf{Q}) = \emptyset.$

Definition A.42. Scope Signature:

$$\mathcal{S}S(\mathbf{Q}) = \Big\{ y^s \colon (\exists x)(x \in_s \mathbf{Q}) \ \& \ \big[(\neg \mathcal{X}(x) \rightarrow y = \emptyset) \text{ or } (\mathcal{X}(x) \rightarrow y = \mathcal{S}S(x))\big] \ \Big\}.$$

Example(s) A.4.

Given $A = \{a^1, \{b^2, w^2, v^2\}^3, \{\{d^4, e^2, f^3\}^5\}^6 \},$

then $\mathcal{S}S(A) = \{\emptyset^1, \{\emptyset^2\}^3, \{\{\emptyset^2\, \emptyset^3\, \emptyset^4\}^5\}^6 \}.$

A.6. NESTED SET OPERATIONS.

Definition A.43. Partial Scope:

$$\mathcal{P}S(s, \mathbf{Q}) = \Big\{ x^{<i,r>} \colon \ (\ s \in r \ ) \ \& \ i \in \mathbf{N} \ \& \ (\ x \in_r \bigcup_{i-1}(\mathbf{Q}) \ ) \ \Big\}.$$

Definition A.44. Nested Scope:

$$\mathcal{N}S(s, \mathbf{Q}) = \Big\{ x^i \colon \ i \in \mathbf{N} \ \& \ (\ x \in_s \bigcup_{i-1}(\mathbf{Q}) \ ) \ \Big\}.$$

Definition A.45. Nested Element:

$$\mathcal{N}E(x, \mathbf{Q}) = \Big\{ s^i \colon \ i \in \mathbf{N} \ \& \ (\ x \in_s \bigcup_{i-1}(\mathbf{Q}) \ ) \ \Big\}.$$

Definition A.46. Tag Map:

$$\mathcal{T}m(\mathbf{A}, \mathbf{B}) = \Big\{ < s, r > \colon \ (\exists x)(\ x \in_s \mathbf{A} \ \& \ \ x \in_r \mathbf{B} \ ) \Big\}.$$

Definition A.47. Similar Subset:

$$\mathbf{A} \sqsubseteq \mathbf{B} \iff (\forall x, s)(\ x \in_s \mathbf{A} \ \rightarrow \ x \in \mathbf{B} \ ) \Big\}.$$

Theorem A.12. *For all* $\mathbf{A}$, $\mathbf{A} \sqsubseteq \mathbf{A}$.

Definition A.48. Fuzzy Subset:

$$\mathcal{F}sub(\mathbf{A}, \mathbf{B}) \iff (\forall x, s)(\ x \in_s \mathbf{A} \ ) \longrightarrow (\exists r)(\ x \in_r \mathbf{B} \ \ \& \ \ s \dot{\sqsubseteq} r \ ).$$

Definition A.49. Fuzzy Element:

$$\mathcal{F}elm(x, s, \mathbf{A}) \longrightarrow (\exists r)(\ x \in_r \mathbf{A} \ \ \& \ \ s \dot{\sqsubseteq} r \ ).$$

Definition A.50. Scope Expansion

$$\mathcal{S}x(\mathbf{Q}) = \Big\{ z^z : tup(z) \ \&\ (\exists w_1)(w_1 \in_{\boldsymbol{\rho}_1(z)} \mathbf{Q}) \ \&\ (\exists w_j)\Big((1 < j \le \#(z)) \ \& \ (w_j \in_{\boldsymbol{\rho}_j(z)} w_{j-1}) \ \&\ \Big(j = \#(z) \longrightarrow (\forall x, s)(x \notin_s w_j) \Big) \Big) \Big\}.$$

Definition A.51. Element Expansion

$$\mathcal{E}x(\mathbf{Q}) = \Big\{ y^z : tup(z) \ \&\ (\exists w_1)(w_1 \in_{\boldsymbol{\rho}_1(z)} \mathbf{Q}) \ \&\ (\exists w_j)\Big((1 < j \le \#(z)) \ \& \ (w_j \in_{\boldsymbol{\rho}_j(z)} w_{j-1}) \ \&\ \Big(j = \#(z) \longrightarrow (\forall x, s)(x \notin_s w_j) \ \&\ (y = w_j) \Big) \Big) \Big\}.$$

Example(s) A.5.

Given $\mathbf{Q} = \{a^A, \{b^B, w^D\}^C, \{\{d^D, w^D\}^E\}^F\}$, then

$\mathcal{S}x(\mathbf{Q}) = \{ <A>, <C,B>, <C,D>, <F,E,D> \}$, and

$\mathcal{E}x(\mathbf{Q}) = \{ a^{<A>}, b^{<C,B>}, w^{<C,D>}, d^{<F,E,D>}, w^{<F,E,D>} \}$.

Definition A.52. Nested Scope Restrict:

$$\mathcal{N}SR(\mathbf{Q}, \beta) = \Big\{ y^s : (s \in \beta) \ \&\ (\exists z)(z \in_s \mathbf{Q}) \ \& \ \Big((\forall w, v)(w \notin_v z \ \rightarrow \ y = z) \text{ or } (\exists w, v)(w \in_v z \ \rightarrow \ y = \mathcal{N}SR(z, \beta)) \Big) \Big\}.$$

Definition A.53. Nested Element Restrict:

$$\mathcal{N}ER(\mathbf{Q}, A) = \Big\{ z^s : (z \in_s \mathbf{Q}) \ \&\ (\exists a, v, i)\Big((a \in_v A) \ \&\ \mathcal{F}elm(a, v, \textstyle\bigcup_i(\mathbf{Q})) \Big) \Big\}.$$

Example(s) A.6.

For $\mathbf{Q} = \{ \{ a^A, b^B, c^C, d^D \}^W, \{ e^A, f^B, g^B, h^D \}^W, \{ i^A, j^B, \{k^E\}^F \}^V \}$,

$\mathcal{N}ER(\mathbf{Q}, \{k^E\}) = \{ \{ i^A, j^B, \{k^E\}^F \}^V \}$,

$\mathcal{N}SR(\mathbf{Q}, \{W, B\}) \cup \mathcal{N}SR(\mathbf{Q}, \{A, F, V, E\}) = \{ \{ b^B \}^W, \{ f^B, g^B \}^W, \{ i^A, \{k^E\}^F \}^V \}$.

Definition A.54. Named Sets:

Given $\mathcal{U}$ where $x \in \mathcal{U} \ \rightarrow \ tup(x) = 2 \ \&\ \neg\mathcal{X}(\boldsymbol{\rho}_1(x)) \ \&\ (\forall a, b, c)(< a, b > \in \mathcal{U} \ \&\ < a, c > \in \mathcal{U} \ \rightarrow \ b = c)$,

then $\mathcal{N}_{\mathcal{U}}(A)$ is defined by:

1) $(\exists B)(< A, B > \in \mathcal{U} \longrightarrow \mathcal{N}_{\mathcal{U}}(A) = \mathcal{N}_{\mathcal{U}}(B))$, or

2) $(\forall B)(< A, B > \notin \mathcal{U} \ \&\ \neg\mathcal{X}(A) \longrightarrow \mathcal{N}_{\mathcal{U}}(A) = A$, or

3) $(\forall B)(< A, B > \notin \mathcal{U} \ \&\ \mathcal{X}(A) \longrightarrow \mathcal{N}_{\mathcal{U}}(A) = \{ z^s : (\exists x)(x \in_s A \ \&\ z = \mathcal{N}_{\mathcal{U}}(x)) \}$.

Example(s) A.7.

For $\mathbf{A} = \{ a, \text{“}B\text{”} \}$, $\mathbf{B} = \{ b, c \}$, $\mathbf{Q} = \{ a, \{ b, c \} \}$, $\mathcal{U} = \{ < \text{“}B\text{”}, \mathbf{B} > \}$,

then $\mathcal{N}_{\mathcal{U}}(\mathbf{A}) = \mathbf{Q}$, $\mathcal{N}_{\mathcal{U}}(\text{“}B\text{”}) = \mathbf{B}$, $\mathcal{N}_{\mathcal{U}}(\mathbf{B}) = \mathbf{B}$.

A.7. NOTATIONAL EQUIVALENCIES.

Definition A.55. Character String Expression of Membership Expression:

$\mathcal{C}seme(\{ x^y \}) = \{$`x[y]`$\}$.

Definition A.56. Membership Expression of Character String Expression:

$\mathcal{M}ecse(\{$`x[y]`$\}) = \{ x^y \}$.

Example(s) A.8.

For $\mathbf{Q} = \{ a^{<1,A>}, \{ b^{<1,B>} \}^{<2,C>} \}$, $\mathcal{C}seme(\mathbf{Q}) =$ `{ a[<1,A>], { b[<1,B>] }[<2,C>] }`.

Appendix B. W3C: XML QUERY REQUIREMENTS

W3C Working Draft 15 August 2000

This version:
http://www.w3.org/TR/2000/WD-xmlquery-req-20000815
Latest version:
http://www.w3.org/TR/xmlquery-req
Previous version:
http://www.w3.org/TR/2000/WD-xmlquery-req-20000131
Editors:
Don Chamberlin (IBM Almaden Research Center) <chamberlin@almaden.ibm.com>
Peter Fankhauser (GMD-IPSI) <fankhaus@darmstadt.gmd.de>
Massimo Marchiori (W3C/MIT/UNIVE) <massimo@w3.org>
Jonathan Robie (Software AG) <jonathan.robie@SoftwareAG-USA.com>

Copyright 2000 W3C (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

B Use Cases for XML Queries

The use cases listed below were created by the XML Query Working Group to illustrate important applications for an XML query language. Each use case is focused on a specific application area, and contains a Document Type Definition (DTD) and example input data. Each use case specifies a set of queries that might be applied to the input data, and the expected results for each query. Since the English description of each query is concise, the expected results form an important part of the definition of each query, specifying the expected output format.

Some of the use cases assume that input is provided in the form of one or more documents with specific names such as "http://www.bn.com/bib.xml". Other use cases are based on implicit (unnamed) input documents. The input environment for each use case is stated in its Document Type Definition (DTD) section.

These use cases represent a snapshot of an ongoing work. Some important application areas are not yet adequately covered by a use case. The XML Query Working Group reserves the right to add, delete, or modify individual queries or whole use cases as the work progresses. The presence of a query in this set of use cases does not necessarily indicate that the query will be expressible in the XML Query Language(s) to be created by the XML Query Working Group.

B.1 Use Case "XMP": Experiences and Exemplars

This use case contains several example queries that illustrate requirements gathered from the database and document communities.

B.1.1 Document Type Definitions (DTD)

Most of the example queries in this use case are based on a bibliography document named "http://www.bn.com/bib.xml" with the following DTD:

```
<!ELEMENT bib    (book* )>
<!ELEMENT book   (title, (author+ | editor+ ), publisher, price )>
<!ATTLIST book   year CDATA  #REQUIRED >
<!ELEMENT author (last, first )>
<!ELEMENT editor (last, first, affiliation )>
<!ELEMENT title  (#PCDATA )>
<!ELEMENT last   (#PCDATA )>
<!ELEMENT first  (#PCDATA )>
<!ELEMENT price  (#PCDATA )>
<!ELEMENT affiliation (#PCDATA )>
<!ELEMENT publisher   (#PCDATA )>
```

B.1.2 Sample Data

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price> 65.95</price>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>
    <book year="2000">
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price> 39.95</price>
    </book>
    <book year="1999">
        <title>The Economics of Technology and Content for Digital TV</title>
        <editor>
               <last>Gerbarg</last><first>Darcy</first>
                <affiliation>CITI</affiliation>
        </editor>
            <publisher>Kluwer Academic Publishers</publisher>
        <price>129.95</price>
    </book>
</bib>
```

B.1.9 Queries and Results

B.1.9.1 Q1

List books published by Addison Wesley after 1991, with their year and title.

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
```

References

1. [Ab00] Abiteboul, Serge; Buneman, Peter; Suciu, Dan: *Data on the Web, From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, 2000
2. [Ch68] Childs, D. L.: *Feasibility of a Set-Theoretic Data Structure: A General Structure Based on a Reconstituted Definition of Relation*, Proc. IFIP Congress 1968
3. [Ch77] Childs, D. L.:*Extended Set Theory: A General Model for Very Large, Distributed, Backend Information Systems*, Third International Conference On Very Large Databases, Tokyo, Japan, 1977
4. [Ch86] Childs, D. L.:*A Mathematical Foundation For Systems Development*, NATO ASI Series, Vol F24, Database Machines, Edited by A. K. Sood and A. H. Qureshi, Springer-Verlag, 1986
5. [Ch00] Childs, D L: *Axiomatic Extended Set Theory*, Unpublished, 1990-2000
6. [Da95] Date, C, J.: *An Introduction to Database Systems*, 6th. edition, Addison-Wesley, 1995
7. [Da00] Date, C, J.: *An Introduction to Database Systems*, 7th. edition, Addison-Wesley, 2000
8. [Ma83] Maier, D.,: *The Theory of Relational Databases*, Computer Science Press, Inc., 1983
9. [Ma00] Martin, Didier; *et al*: *Professional XML*, Wrox Press Ltd., 2000