XSP TECHNOLOGY: Theory & Practice

Formal Modeling & Practical Implementation of XML & RDM Systems

INTRODUCTION

XSP (extended set processing) Technology introduces: [1] a formal mathematical foundation, extended set theory, XST, for defining and mapping user application expectations into internal machine representations; and [2] practical software implementations based on XST that assist in the design, development, and use of XML and RDM systems.

It is common knowledge that RDM advocates tout the mathematically superiority of the RDM because of its grounding in set theory. In point of fact, the RDM works in spite of set theory, not because of it. It is also common knowledge that XML devotees make no claim of any kind that XML-Structures and operations on these structures have any formal counterpart in set theory, yet they do. In this paper the formal foundations of RDM and XML systems will be examined in the light of XST to provide practical relevance of XSP Technology to the field of software systems development.

It is very important to understand that a deep knowledge and understanding of XST was required to develop the existing software, but it is also equally important to understand that no knowledge nor understanding of the mathematics of XST is required to use XSP software.

Need For a Theory

Every technology must have a sound underlying theory to support the consistency and predictability of the methods promoted by the technology. In this context, the term theory is respected as an articulation of a body of rules governing the relationships and behavior of objects in a specific system of interest.

Unfortunately, within the discipline of software development, the term *theory* is often considered synonymous with speculative, unproven, impracticable, and is generally viewed as antithetical to the pragmatic reality of systems implementation.

One might wonder why a formal underlying theory is revered by one discipline for guidance, control, and understanding, but that the very mention of a theory is shunned by another discipline as being irrelevant, impracticable, and unproductive.

For implementing software systems the answer is quite simple - no theory is known to have been discovered that allows a formal modeling of the actual computer environment, where all software ultimately resides to be executed. Without such a formal model there can be no language for expressing any applicable formal theory.

This is not to say that formal models do not exist in the field of software development. They abound at modeling the world external to the computer, and abstract characterizations of machine behavior, but none venture to faithfully model the actual internal environment of the computer, until now. XSP Technology is based on just such a formal model.

So why has a formal model been so difficult to find? What exactly is the formal model introduced by XSP Technology, and now that such a formal model exists (hence a theory, thus a technology) why is it of any practical value for implementing better systems, particularly XML and RDM systems?

Deficient Mathematical Foundations

The unlikely culprit that has been thwarting the discovery of a formal foundation to modeling computer based systems is the deficiency of adequate mathematical support. The existing body of accepted formal mathematics is inadequate to meet the modeling needs required to formally describe the user application environment, the internal machine environment, and embeddings of application expectations into executable realizations. This incredulous assertion, obviously, needs to be amplified.

It is well known that the basic building block of all computer implementations is the *record*. It is also well known exactly what is meant by the term *record*. Given records $[a, b, c]$ and $[x, b, y, z]$, anyone with a background in computers knows that b is the 'second place element' of both records.

To provide a formal model of records, the notation of set theory is generally employed giving: $< a, b, c >$ and $< x, b, y, z >$, which seems to preserve the notion that b is the 'second place element' of both n-tuples. Unfortunately, according to accepted set-theoretic definitions of *membership*, b is not even an element of either n-tuple.

The use of the 'set-theoretic notation' for an n-tuple has to be carefully distinguished from the use of the 'set-theoretic membership condition' of an n-tuple. When used to represent a record, the former is just a notational homage to set theory while the later is an actual use of set theory. The former works, the later fails. This failure is due to a deficiency in the foundations of set theory. The 'set-theoretic membership condition' of n-tuple is not well defined by any classical set theory.

In the RDM records are equated to n-tuples. The fact, that the RDM works as well as it does, is because the underlying Relational Algebra is defined using an intelligent interpretation of how n-tuples behave, instead of a rigorous set-theoretic adherence to n-tuple behavior.

Records & n-Tuples

There exits a fundamental problem when the set-theoretic notation for n-tuples is used to model records. There is no problem with records *per se*. The problem is in trying to formally model records with set theory, ANY set theory. If set theory were not used to model records, there would be no problem.

The problem arises because set theory requires that n-tuples be defined as sets. Since n-tuples are sets, they must have well-defined membership conditions and must exhibit meaningful behavior when operated on by set operations, such as: $< a, b > \cap < a, c >$ and $< a, b, c > \cup < x, y, z >$.

All set theories depend on the ability to determine if a 'membership condition' is TRUE or FALSE.

For records, though they are not sets, this is quite simple:

1) is a a member of the record $[a, b, c]$? Answer: YES.
2) is b a member of the record $[a, b, c]$? Answer: YES.
3) is c a member of the record $[a, b, c]$? Answer: YES.

If records are modeled by n-tuples (thus as sets) where $[a, b, c]$ is equated with $< a, b, c >$, then:

4) is a a member of the n-tuple $< a, b, c >$? Answer: ?
5) is b a member of the n-tuple $< a, b, c >$? Answer: ?
6) is c a member of the n-tuple $< a, b, c >$? Answer: ?

In short, there is no problem with the concept of records, but there is a problem with the concept of n-tuples. The problem exists in set theory, since in set theory n-tuples are not universally and unambiguously well-defined. Within certain restricted uses, n-tuples behave very well. In the general context of set theory n-tuples behave very badly, simple because it is not always clear how to determine if an element *is* or *is not* a member of any given n-tuple. If the issue of membership

does not arise, n-tuples are well-behaved - by default, since no behavior is required of them.

Though formal set-theoretic fundamentals are not required, nor even relevant, for using software derived from XSP Technology, a healthy insight can be gained from a brief exposure to subtle problems caused by trying to model the programming concept of a *record* with the set-theoretic concept of *n-tuple*. (A more brain wrenching set-theoretic exposure can be found in [ref. 1-4].)

It is rumored that Edison knew 10,000 ways not to make a light bulb work. They may all be of some academic interest, but the only one of practical interest is one that does work. Similarly with the set-theoretic definition of an n-tuple. The only set-theoretic definition of n-tuple that is of any practical interest is one that does work.

'Work' in this context means that operations on n-tuples behave as any intelligent person would expect them to behave, and not with the bazaar behavior they generally exhibit under classical set-theoretic definitions. More specifically, we would like a set-theoretic definition of n-tuple that formally reflected our intuitive notion of how records behave. From the previous example, it would mean that a, b, and c are members of the n-tuple $< a, b, c >$, and that $< a, b, c > \cap < x, b, z >$ would somehow result in a set that has b as the second and only element, something like $< -, b, - >$.

Classical Sets & Extended Sets

The formal development of Axiomatic Extended Set Theory [ref. 4] provides all the necessary mathematical machinery to achieve the above results, and more. Unfortunately, the arcane mathematical development obscures the simple underlying notation that makes it 'work'. That simple notion will now be exposed.

In Classical Set Theory, CST, set membership is defined in terms of a single condition, *element*. In Extended Set Theory, XST, set membership is defined in terms of two conditions. *element* and *scope*. Both terms, *element* and *scope*, are undefined mathematical primitives that can assume any meaning that one wants to bestow on them. They have NO intrinsic meaning in and of themselves. Any terms could have been chosen as long as they are used consistently. This mathematical disclaimer having been said, of course the terms were chosen to reflect that 'elements' are going to be considered to be items of interest and that 'scopes' are going to be considered as an associated property of an element. The following examples will illustrate these two membership conditions.

In CST the notation $\mathbf{A} = \{a, \{b\}, C\}$ means that a is an element of $\mathbf{A}$, $\{b\}$ is an element of $\mathbf{A}$, and C is an element of $\mathbf{A}$. While in XST the notation $\mathbf{B} = \{a^r, \{b^s\}^t, C^u\}$ means that a is an r-element of $\mathbf{B}$, $\{b^s\}$ is a t-element of $\mathbf{B}$, and C is a u-element of $\mathbf{B}$.

The elements of $\mathbf{A}$ and $\mathbf{B}$ are exactly the same though the membership conditions are different, since $\mathbf{B}$ also has scope values r, t, and u. $\mathbf{A}$ is a CST set and therefore has no scope values. However, every CST set can be transformed into a XST set by assigning every element in a CST set a null scope. Thus $\mathbf{A}$ becomes the XST set $\{a^{\emptyset}, \{b^{\emptyset}\}^{\emptyset}, C^{\emptyset}\}$.

n-tuples in XST

Since scopes can be anything that an element can be, including sets themselves, scopes can be chosen to be integers. Thus the set $\{a^1, b^2, c^3\}$ is well defined in XST. As are the sets $\{x^1, b^2, z^3\}$ and $\{b^2\}$. It should be noted that the following also holds:

$$\{a^1, b^2, c^3\} \cap \{x^1, b^2, z^3\} \;=\; \{b^2\}.$$

Thus, an obvious notational choice of the concept of an n-tuple is the following:

$$< x_1, x_2, x_3, ..., x_n > \equiv \; \{{x_1}^1, {x_2}^2, {x_3}^3, ..., {x_n}^n\}.$$

It then follows that:

$$< a, b, c > \cap < x, b, z > = \{b^2\}.$$

Armed with this XST definition for n-tuple, all the intuitive notations about the behavior of records can be preserved when modeled by n-tuple notation. This also allows anomalies in the RDM to be corrected and would legitimize the bragging rights of the RDM as being mathematically sound. As is shown below and more extensively in [ref.5], a more exotic use of scopes also provides a formal foundation for processing XML-Structures.

Before continuing, however, it should be stressed that the concept of scopes is not just a notational slight-of-hand that can be supported by any existing set theory. It isn't and it can't. For a better understand of this, please appeal to [ref. 4].

The Power of Scopes

Classical sets, by definition, are unstructured collections of elements. Where the elements themselves can also be sets. By contrast, extended sets can be viewed as classical sets with *structure*. This concept of a 'structured set' is no more an oxymoron than is the term 'multi-valued-function'. So, in a naive way, scopes can be considered to add *structure* to sets. What that structure is, depends on the choice of scope values. The choice of integers, though obvious, is not necessarily the only choice with practical ramifications for software development.

Any subset of a computer memory can be faithfully modeled by an extended set, where the elements of the set are byte values and the scopes of the elements are memory addresses. As simple as this example might be, it is an example of a component of the internal machine environment that now has a legitimate mathematical model. True that the notation is not new, but the underlying membership is now unambiguous. Thus two subsets of RAM can now be treated with mathematical rigor: their intersections, unions, symmetric differences, and relative complements are now well-defined.

By extending set operations beyond their usual reliance on elements only, and considering scope relationships (like every $i + 4$ value), then projections and selections can be legitimately defined on subsets of memory. Certainly, there are computer programs that already do this, but now there can be computer programs that have a rigorous mathematical model to ensure consistency and reliability. If eliminating the possibility of erratic and inconsistent behavior while ensuring scaleability is not practical - what is?

XSP software as developed over the last 30 years has encapsulated the mathematical soundness of record modeling by n-tuples so that even the mathematically challenged can benefit from mathematically sound software implementations.

Until recently, XSP software as focused on supporting mathematically sound relational algebra operations at the I/O interface, but with the discovery that XML-structures could also be made mathematically sound under XST, new XSP software operations have been developed to support the querying and processing of XML-structures. How XML zealots can claim the same bragging rights for mathematical soundness as do RDM zealots is presented in the following section.

XML Tags as XST Scopes

The mathematical foundation for RDM systems is based on the concept of an n-tuple, which is falsely believed to be mathematically well-defined. The mathematical foundation for XML systems, for all intents and purposes, is not known to exist and therefore, by default, is falsely believed to be not mathematically well-defined. As it turns out the very nature of an XML-structure ensures that it is mathematically well-defined, under XST. Since n-tuples also depend on XST in order to be

mathematically well-defined, this puts XML-structures on a mathematical par with RDM-relations. In fact, it can be shown that XML-structures are mathematically richer than RDM-relations and contrary to RDM lore, RDM-relations can be absorbed by XML-structures. (see ref. [5])

The reason XML-structures turn out to be mathematically sound can be shown by comparison to HTML-structures which are not intrinsically mathematically sound. XML-structures are 'properly nested' HTML structures are not necessarily 'properly nested'. As an example $\{[a]\}$ is a proper nesting. $\{[a\}]$ is not a proper nesting.

RDM-relations draw on one set-theoretic property: sequencing. XML-structures draw on two set-theoretic properties: sequencing and nesting. In CST these to properties usurp one another in that one is used to define the other. In XST these two properties are independent of each other.

For example: In CST the cardinality of $\{ < a, b >, \{\{a\}, \{a, b\}\} \}$ is 1.
In XST the cardinality of $\{ < a, b >, \{\{a\}, \{a, b\}\} \}$ is 2.

The following is a very simple XML-structure:

$< P >$
$< p >$
$< n > Alan < /n >$
$< a > 42 < /a >$
$< e > abc.com < /e >$
$< /p >$
$< p >$
$< n > Mary < /n >$
$< a > 29 < /a >$
$< e > mky.com < /e >$
$< /p >$
$< /P >$

Which could be represented by:

$$P = \left\{ \{ Alan^{<1,n>}, 42^{<2,a>}, abc.com^{<3,e>} \}^{<1,p>}, \{ Mary^{<1,n>}, 29^{<2,a>}, mky.com^{<3,e>} \}^{<2,p>} \right\}.$$

When XML tags are embedded in XST scopes, XML-structures become well-defined extended sets, (ref. [5]). Since existing XSP software already relies on scopes to support structures internal to the machine environment, it is a relatively easy task to extend the existing software to support structures external to the machine environment, like XML-structures captured as extended sets.

XSP Software

XSP software provides a low level functionally complete query support and data access capability that can out perform any existing commercial RDMS in processing XML data. The XSP software has already been shown to out perform existing RDBMSs in processing flat file relational data, as demonstrated using a well known standard benchmark.

XSP software takes the form of a set of function calls for processing XML-structures. These calls can be executed interactively, or called from C programs. XSP software can be used to augment existing systems or as 'what if' software for experimenting with development approaches and performance issues that would otherwise be too expensive and time consuming to explore.

XSP software description:

* Low level primitive functions for manipulating XML-structures
* Functions are extensions of the Relational Algebra
* No knowledge of set theory is required to use the functions
* Strong Data Independent interface between application and storage

The XSP functions can be viewed as extensions of the Relational Algebra that treat XML Tags as XST Scopes and XML elements as XST elements. These XSP functions are XML-complete, in the sense that they support any Relational Algebra type functionality definable in terms of XPath, XQuery, SQL, QUILT, X???, on any structure, or mixed collection of structures, that are well-formed by XML standards or by RDM standards. This includes all XML-structures, DTDs, RDFs, RDM-Relations, etc.. These XSP functions provide 'tangible proof' that a mathematically sound, operation-centric foundation exists for XML processing.

XSP software has been ported to all major mainframe and PC operating systems and now provides the capability to augment existing systems in the exploration and access of heterogeneous and distributed data for real time applications or for assisting in design and system development.

References

1. [Ch68] Childs, D L: Feasibility of a Set-Theoretic Data Structure: A General Structure Based on a Reconstituted Definition of Relation, Proc. IFIP Congress, Edinburgh Scotland, 1968

2. [Ch77] Childs, D L: Extended Set Theory: A General Model for Very Large, Distributed, Back-end Information Systems, Third International Conference On Very Large Databases, Tokyo, Japan, 1977

3. [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

4. [Ch00] Childs, D L: *Axiomatic Extended Set Theory*, 2000

5. [Ch01] Childs, D L: *XSP Technology for XML Systems Design & Development*, 2001