一种基于操作集计算生成有效 XML 文档的方法*)

卑小贤　陈　平　刘西洋
(西安电子科技大学软件工程研究所　西安 710071)

摘　要　目前 XML 工具的编辑能力不足以保证生成有效的 XML 文档，缺乏对文档模式的合理利用。而由 XML Schema 定义的文档模式，不仅提供了有效性验证标准，同时蕴含了生成有效 XML 文档的准则。本文提出了一种生成有效 XML 文档的方法。该方法基于 XML 文档模式图定义不同节点的操作集合，以及该操作集合上的计算和相应操作语义，并分析论证了该方法本身的有效性。

关键词　XML 文档，操作集计算，XML 文档模式图，XML Schema，有效性

An Approach for Creating Valid XML Document Based on Operation Set Computing

BEI Xiao-Xian　CHEN Ping　LIU Xi-Yang
(Software Engineering Institute, Xidian University, Xi' an 710071)

Abstract　Currently XML tools cannot guarantee the validity of XML document. They aren't built upon document schema well. Document schema defined by XML schema provides not only the validating criterion, but also implies the rules for generating valid XML documents. In this paper, an approach is provided for creating valid XML documents. It is implemented by computing operation sets defined on the schema graph of XML document. And semantics of operations is specified. Also its validity is proved.

Keywords　XML document, Operation set computing, Schema graph of XML document, XML schema, Validity

XML 已成为一种事实上的信息交换标准格式，文档有效性是其基本要求。XML Schema 作为 XML 文档模式的一种定义语言，对文档进行规范，是文档有效性的依据，且已被 W3C 组织确定为定义 XML 文档模式的推荐标准[1,2]。XML Schema 定义包括两个方面：①定义复杂数据的嵌套关系，形成严格层次化的树型结构；②定义简单数据的数据类型，即规范数据项内容[3]。因此，XML 文档有效性需要从结构有效性和数据内容有效性两个方面来验证。其中，结构有效性是数据有效性的前提，因为违背结构约束条件的数据是不能被正确识别的。反之，生成 XML 文档也须从这两个方面考虑。

目前常见的 XML 工具，一般都具有 XML 文档以及 XML Schema 的编辑生成能力，即生成符合 XML 语法的文档，如 XML Explorer、XML Spy 等。部分工具可以依据文档模式对 XML 文档进行一致性验证，如 XML Spy 等。但这些验证本身还存在不完整性，如文[4]中所述，不能发现递归引用导致的不可实例化环，而且没有将 XML 文档模式作为严格的生成依据。因为，由 XML Schema 定义的文档模式不仅提供了有效性验证标准，同时蕴含了生成有效 XML 文档的准则。所以，在没有严格依据规范和验证能力不完整的情况下，编辑生成的 XML 文档有效性就难于保证。本文提出了一种基于 XML 文档模式图的方法，通过定义不同节点的操作集合，及该操作集合上的计算和相应操作语义，来控制对 XML 文档的编辑生成。

1　XML 文档模式图

XML Schema 定义的文档模式都可以用 XML 文档模式图来表示[4]。文[4]就是基于文档模式图进行模式本身的有效性验证。图 1 是购物订单 XML Schema 的模式图片断(模式文档参见文[3]中的 po. xsd)。在图 1 中，purchaseOrder 是根元素，它的类型是 purchaseOrderType。而 purchaseOrderType 是复杂类型，包含了 shipTo 等子元素以及 orderDate 属性。同时，shipTo 和 billTo 又是定义在复杂类型 USAddress 基础上的。而元素 items 由复杂类型 Items 来限制。comment 是定义在内建数据类型 string 上的，属性 orderDate 类型为内建数据类型 Date。图 1 中还给出了复杂类型包含的元素定义顺序和实例数量范围，以及属性的实例数量范围。例如，comment 元素的约束三元组(3,0,1)，表示顺序为 3、最小实例数量为 0、最大为 1。

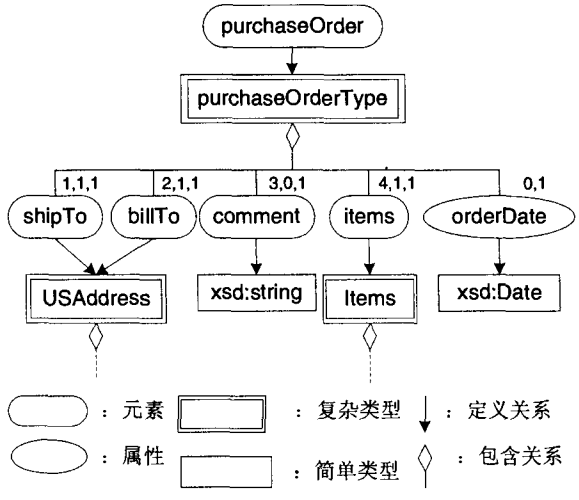


图 1　购物订单(po. xsd)文档模式图

图 2 是图 1 的一个实例片断(参考文[3]中的 po. xml)。如果当前操作节点为 purchaseOrder，对于插入操作而言，由

*)基金项目："十五"国防预研项目(413150501)。卑小贤　博士研究生；刘西洋　副教授；陈　平　教授，博士生导师。

于所有子节点都存在，且到达实例数量最大值，所以不能再插入任何子节点，而且作为文档根元素也是不允许删除的。如果当前操作节点是 comment，由于其实例数量超过最小实例数量，所以是可以删除的。而且，因为是简单类型元素，所以没有子节点可以插入，但是可以编辑内容。由此可见，对于这样一棵实例文档数据树，任何节点总是可以计算出该节点在当前状态下的有效操作集合，即不破坏与文档模式一致性要求的操作集合。

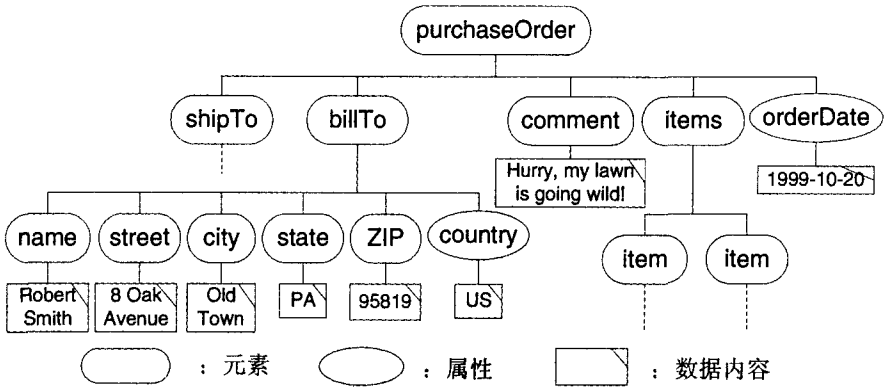


图 2　购物订单(po. xml)实例图

2　操作集的定义和维护

由上述实例分析可以得出：1)文档模式蕴含了操作集合；2)具体的有效操作集得由实例文档状态和文档模式共同作用计算得到。

XML 文档是一棵层次化的树型数据文档，对 XML 文档的编辑都可以落实到这棵数据树的某一个节点上。而任何一个节点都有对应的类型定义，所以依据该节点类型就可以计算出当前状态下允许的有效操作集合。为了便于定义，根据 XML Schema 规范[1]，将节点分成 3 类：简单元素节点(Simple Element，Es)、复杂元素节点(Complex Element，Ec)和属性节点(Attribute，A)。其中，简单元素和属性构成了 XML 文档树的叶子节点，一定是简单数据类型。而复杂元素是 XML 文档树的内部节点，定义在复杂类型上。

通常情况下，对于一棵树的操作包括节点的删除、插入和修改。本文规定，在节点上的操作只能作用于自身及子孙节点，也即只能进行插入子节点、删除本节点和修改本节点内容等操作。由此，可以定义如下操作集合：

OperationSet={Delete，Insert，Modify}

OperationSet(A)={⟨Delete，Self⟩，⟨Modify，Self⟩}

OperationSet(Es)={⟨Delete，Self⟩，⟨Modify，Self⟩}

OperationSet(Ec)={⟨Delete，Self⟩，⟨Insert，Self. Def. childrens⟩，⟨Insert，Self. Def. attributes⟩}

其中，A、Es 和 Ec 是操作对象，Self 表示当前操作节点，Def 表示节点的类型定义。childrens 指的是复杂元素节点定义中的子元素集合，attributes 指的是定义中的属性集合。

2.1　最大操作集定义

最大操作集(Max Operation Set，MaxOS)是指 XML 文档树的当前节点上的所有可能操作集合。节点的定义类型决定了操作可能集合。最大操作集合定义如下：

MaxOS(A)={⟨Delete，Self⟩，⟨Modify，Self⟩}

MaxOS(Es)={⟨Delete，Self⟩，⟨Modify，Self⟩}

MaxOS(Ec)={⟨Delete，Self⟩}∪{⟨Insert，E⟩|E∈Self. Def. childrens}∪{⟨Insert，A⟩|A∈Self. Def. attributes⟩}

2.2　有效操作集计算

有效操作集合(Valid Operation Set，VOS)是指 XML 文档树的当前节点上的所有有效操作集合，它是最大操作集合的一个真子集，取决于当前操作节点的状态。

VOS(A)={⟨Delete，Self⟩|Self. count＞Self. Def. min}∪{⟨Modify，Self⟩|Self. Def. fixed=false}

VOS(Es)={⟨Delete，Self⟩|Self. count＞Self. Def. min}∪{⟨Modify，Self⟩|Self. Def. fixed=false}

VOS(Ec)={⟨Delete，Self⟩|Self. count＞Self. Def. min}∪{⟨Insert，E⟩|E∈Self. Def. childrens and E. count＜E. Def. max}∪{⟨Insert，A⟩|A∈Self. Def. attributes and A. count＜A. Def. max}

其中，min，max 表示该类型节点允许实例的最小数量、最大数量。count 是该类型节点实际实例数量。fixed 表示该类型节点内容是否固定。

2.3　缺省操作集计算

缺省操作集(Default Operation Set，DOS)是指在默认状态下插入节点时所激发的新操作集合。而新的操作也一定是插入操作，主要用以保证文档结构有效性。所以，缺省操作集计算发生在初始空实例文档和插入复杂元素节点时。定义如下：

DOS(ϕ)={⟨Insert，E⟩|E is root element}

DOS(Ec)={⟨Insert，E⟩×E. Def. min|E∈Self. Def. childrens and E. Def. min＞0}∪{⟨Insert，A⟩|A∈Self. Def. attributes and A. Def. min＞0}

初始空实例文档时，首先插入必定是根元素，而且是唯一的。文[4]通过构造文档模式图可以分析得出根元素。由于子元素实例最小数量可能大于 1，所以插入子元素必须保证必要的数量，才能保证结构有效性。这里“×E. Def. min”就是表示实例化 min 个 E 元素。由于是缺省操作，同类型元素实例结构都一样，所以一般只需实例一次，其它节点可以通过复制方式来实例。

2.4　操作语义以及操作序列计算

操作集合中的元素是二元组⟨操作名，对象⟩。操作对象类型不同，操作实现也是不同的，所以必须明确操作的语义。而且，为了保证操作结果有效性，对于复杂元素节点往往是操作序列。操作语义如下：

⟨Delete，N⟩：当 N 为 A 或者 Es 节点时，只需删除该节点；当 N 为 Ec 节点时，包括所有子节点在内都删除；并且 N. count=N. count−1，即实例数量减 1。

〈Modify, N〉:编辑该节点内容。编辑结束时,一般进行数据有效性验证,如果有效,则 N. valid=True;否则,N. valid=false。

〈Insert, N(A|Es)〉:插入 A 或者 Es 节点时,需要赋予初值 N. value=N. Def. default;并且 N. count=N. count+1,即实例数量加 1。

〈Insert, N(Ec)〉:插入 Ec 节点时,N. count=N. count+1;同时执行该节点的缺省操作集 DOS(Ec)中的所有插入操作。对于最小实例数量大于 1 的节点,还需要多次插入。

通过缺省操作集构成的操作序列来保证实例文档的结构有效性。结合数据内容编辑后数据有效性检查,可以保证数据有效性,从而保证文档整体有效性。

2.5 操作集状态维护

操作定义是建立在基本操作插入、删除和修改的基础上,是由文档模式和实例文档共同维护的。根据上述定义,数据需要维护更新才能保证操作集计算的正确性。其中,文档模式需要维护所有定义数据,也即与 Def 相关的数据。图 1 中给出了最常用的 min、max 的约束元组。由于文档模式是预先定义的,所以数据不必随时更新。而与实例文档相关的数据,如实例数量 count,则需要及时更新,以保证有效操作集计算的正确。一般由复杂类型元素实例节点维护状态数据。

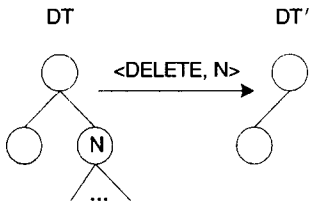
3 基于操作集计算方法的有效性分析

本节给出各种操作集的定义以及操作语义描述,并对所提出的方法进行有效性分析论证。

3.1 操作有效性分析

3.1.1 删除操作有效性分析

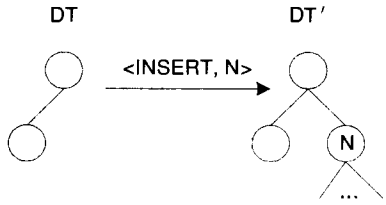
假设 DT 是有效状态,则经过操作〈Delete, N〉后的状态 DT′也是有效的。



证明(反证法):假设 DT′无效。由于 DT′中所有节点只有 Parent(N)发生状态改变,其它节点都没有改变。因此,除了 Parent(N)以外的所有节点还是有效的。而 Parent(N)中也只是少了节点 N,所以仅有 N. count(=N. count−1)发生改变。因此,只可能是 N. count 导致了 DT′无效,即 N. count>N. def. max 或者<N. def. min。但是,操作〈Delete, N〉是在 N. def. max≥N. count>N. def. min 条件下发生的,不可能 N. count<N. def. min。所以,DT′也是有效的。

3.1.2 插入操作有效性分析

假设 DT 是有效状态,则经过操作〈Insert, N〉后的状态也是有效的。



证明:DT 中所有节点只有 Parent(N)发生状态改变、其它节点都没有改变。因此,除了 Parent(N)以外的所有节点还是有效的。而 Parent(N)有效性如前类似证明。对于 N 节点的有效性,分为两种情况考虑:如果是属性节点和简单元素节点,由于其不含有子节点,所以本身结构一定有效;如果是复杂元素节点,那一定是通过缺省操作递归生成的。所以 N 节点有效性可由缺省操作有效性来保证。

3.1.3 修改操作有效性分析

修改操作只会改变数据项内容,不会影响文档结构,所以不会影响结构有效性。对于修改操作,一般采用 ECA 机制进行数据有效性分析[5]。也即每次数据修改完毕,自动捕获事件,激发数据有效性分析。并且,如上修改操作语义定义,会自动对数据有效性进行标记。

3.2 缺省操作集有效性分析

一个节点的结构是否有效,取决于子节点类型和数量。也就是必须包含的节点一定存在,而且存在的子节点实例数量也是有效的。前一句话要求 N. def. min>0 的子节点必须出现,后一句话要求 N. Def. min≤N. count≤N. def. max。如前定义,DOS(Ec)中只插入必要的子节点,也就是 N. Def. min>0 且 N. count=N. Def. min。而且,由于缺省操作集是间接递归定义,可以保证插入的每一个节点都是必需的且有效的,所以结构也是有效的。

3.3 结论

结论 1 任何有效性的编辑状态都是可到达的。

即操作集计算策略是完备的。令 DT=DOS(ϕ),DT′为任何一个有效状态。不容置疑,DT 一定是 DT′的子树。证明可达,也即只需构造一个有效操作序列,能到达 DT′。假设 DT′的树高为 n,构造方法如下:

(1)从第一层开始,将 DT 与 DT′比较,分别插入不存在的节点,直到第一层节点类型和数量都相同。由于每一次插入操作都是有效的,而且都是必要的节点,所以结果还是 DT′的子树。

(2)依次从第二层直到第 n 层重复上述操作。其结果就是 DT′。

如此即说明可以从 ϕ 到达任何有效状态。反之,任何有效状态可以达到 ϕ。最直接的方法就是删除根节点。实际过程可以类似以上的构造方法,从第一层到第 n 层,重复使用删除和插入操作,就可以到达任何有效状态。

结论 2 任何非有效状态可以达到有效状态。

也只需构造一个操作系列。最直接的方法是删除根节点就可以了。实际过程可以是:找出树中所有无效节点,从第 n 层开始,对于 N. count>N. def. max 的节点进行删除操作,对于 N. count<N. def. min 的节点进行插入操作,直到有效为止。如此重复直到第 1 层。

结论 1 和结论 2 说明操作集计算策略并没有以损失编辑能力为代价,但又大大地缩小状态空间,保证了编辑状态的有效性。

3.4 操作集计算性能分析

操作集计算所需的数据来自文档模式和实例文档。由于文档模式中的数据在编辑时不需要维护,且是直接引用,所以对此不需额外空间。实例文档中主要维护同类型子节点实例数量。假定实例文档节点数为 N,则空间复杂度是 $O(N)$。时间复杂度主要是计算当前节点有效操作集。有效操作集是对最大操作集计算而得的真子集,而最大操作集大小取决于文档模式中当前节点的类型定义。假定最大操作集大小为 M,则有效性计算时间复杂度就是 $O(M)$。实际应用中,M—

般都是一个很小的数。所以，虽然操作集计算也会带来一些时空上负荷，但都是线性复杂度，且很小。

通过操作集计算获得的好处是保证每次操作都是有效的，致使文档结构总是有效性的，从而提高编辑生成质量。同时，有缺省操作有效性保证，可以使得工作效率提高，也使用户集中关注数据项有效性。

以操作有效性来保证文档结构有效性，是一种前效策略，可以消除盲目性，提高了效率。而一致性验证发现错误是后效手段，不能避免重复工作。而且，如果对结构认识不清楚，这种后效手段还是无法从根本上解决问题。对于数据内容编辑，除了 ECA 机制进行及时验证外，还可以通过采用相应数据类型编辑控件来保证数据有效性。从某种意义上可以认为编辑控件的功能也是一种有效操作。

结束语 基于上述三个基本操作上的操作集合定义已经满足了 XML 文档的编辑需要。而作为编辑理念中的复制、剪切和粘贴这 3 个操作也容易在基本操作集的基础上扩充实现。复制的目的是为了粘贴。所以，对复制操作可以不做约束，完全由粘贴操作有效性计算来控制。复制的对象应该是以当前节点为根的子树。粘贴的结果是在当前节点前后或者子节点中插入一颗子树。在操作对象类型一致的前提下，粘贴操作的有效性等同于插入操作有效性。剪切操作是删除和复制的操作组合。所以，剪切操作的有效性可以等同于删除操作的有效性。对于鼠标的拖放操作，也可以采用同样的方式计算其有效性。

综上所述，本文提出了一种基于操作集计算方法来保证 XML 数据文档的结构有效。如果结合 ECA 机制或者借助编辑控件对简单类型数据进行验证，就可以完全保证编辑生成的 XML 文档的有效性。

参 考 文 献

1 Radiya A, Dixit V. The basics of using XML Schema to define elements. http://www.ibm.com
2 van der Vlist E. Using W3C XML Schema. http://www.xml.com/pub/a/2000/11/29/schemas/part1.html
3 XML Schema: Part 0: Primer. http://www.w3.org/TR/xmlschema-0
4 董亚绢，卑小贤，等. 一种 XML 文档模式有效性验证算法. 计算机工程与应用(已录用)
5 Bailey J, Poulovassilis A, Wood P T. An Event-Condition-Action Language for XML. WWW2002, 2002

(上接第 64 页)

NAF(29)=(1,0,0,−1,0,1)

计算 NAF 和通过 NAF 求解点乘的运算分别由算法 1 和算法 2 给出。

算法 1 求 NAF 值

输入：正整数 k
输出：NAF(k)
(1)$i\leftarrow 0$
(2)While $k\geq 1$do
If k is odd then:
$k_i\leftarrow 2-(k \bmod 4)$, $k\leftarrow k-k_i$
Else: $k_i\leftarrow 0$
$k_i\leftarrow k/2$, $i\leftarrow i+1$
(3)Return $(k_{i-1}k_{i-2},\cdots,k_1,k_0)$.

算法 2 NAF 方法求点乘

输入：NAF(k)$=\sum_{i=0}^{l-1}k_i2^i$
输出：kP
(1)$Q\leftarrow 0$.
(2)For i from l-1 downto 0 do
$Q\leftarrow 2Q$
If $k_i=1$ then $Q\leftarrow Q+P$.
If $k_i=-1$ then $Q\leftarrow Q-P$.
(3) return(Q).

为了进一步减少算法 1 的运行时间，我们在算法 1 基础上使用了窗口技术[9]。假设宽度为 w，则整数 k 可表示为 $k=\sum_{i=0}^{l-1}k_i2^i$，其中每个非零系数 k_i 是奇数，且$|k_i|<2^{w-1}$。此时，先计算整数 k 的窗口为 w 的 NAF 表示，记为 NAF$_w$(k)，然后再计算点乘。只需将算法 1 稍作修改，便可求得 NAF$_w$(k)。修改之处为：算法 1 步骤(2)中原来的“$k_i\leftarrow 2-(k \bmod 4)$”改为“$k_i\leftarrow k \bmod 2^w$”，其中“$k_i\leftarrow k \bmod 2^w$”表示满足下式的整数 u：$u\equiv k(\bmod 2^w)$，且$-2^{w-1}\leqslant u<2^{w-1}$。例如，给定窗口 $w=4$，$k=22310$，则有

$22310=2^{15}-5\times 2^{11}-7\times 2^5+3\times 2$

因此，在计算 $22310P$ 时，可预先计算 $3P$，$5P$，$7P$，然后计算

$22310P=10^{15}P-2^{11}(5P)-2^5(7P)+2(3P)$

算法 3 窗口 NAF 方法求点乘

输入：整数 k，w，$P(x,y)\in E(F_{2^m})$
输出：kP
//预计算，仅需一次
//计算 uP，for u 为奇数，且 $2<u<2^{w-1}$
(1)$P_0\leftarrow P$, $T\leftarrow 2P$.
(2)For i from 1 to $2^{w-2}-1$ do
$P_i\leftarrow P_{i-1}+T$
//主计算
(3)计算 NAF$_w(k)=(u_{l-1},u_{l-2},\cdots,u_1,u_0)$.
(4)$Q\leftarrow O$.
(5)For j from $l-1$ downto 0 do
$Q\leftarrow 2Q$
If $u_j\neq 0$ then:
$i\leftarrow(|u_j|-1)/2$
If $u_j>0$ then $Q\leftarrow Q+P_i$;
Else $Q\leftarrow Q-P_i$
(6)return(Q)

总结 椭圆曲线密码体制的诱人之处在于，它是建立在有限域椭圆曲线离散对数问题这个数学难题之上的，而此难题比大整数分解及素域乘法群离散对数问题更为难解。因此，与其他公钥密码体制相比，椭圆曲线密码体制在同等安全强度下可以使用长度短得多的密钥，在存储效率、通信带宽及计算效率等方面表现出明显的优势。本文在 Linux VPN 网关的设计中，将椭圆曲线 Diffie-Hellman 密钥交换应用于 IKE 实现过程，达到了动态协商和管理 IPSec SA 的基本要求，大大提高了密钥分配的效率。对于椭圆曲线密码体制而言，点乘运算是最耗时的操作，为此在实现过程中采用了点乘运算的快速实现算法。考虑到由于进行加密和隧道通信，VPN 安全网关负载增大，因此下一步的主要工作将是采取措施对整个系统进行优化，以减轻网关 CPU 的负担。

参 考 文 献

1 Stallings W. Cryptography and Network Security: Principles and a Practice(Third Edition). Prentice Hall, 2002
2 Davis C R. IPSec: VPN 的安全实施. 周永彬，等译. 北京：清华大学出版社，2002
3 RFC2367. PF_KEY Key Management API, Version 2, 1998
4 RFC2409. The Internet Key Exchange (IKE), 1998
5 IEEE P1363. Standard Specification for Public-key Cryptography, 2000
6 ANSI X9.63. Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, 1999
7 朱艳琴. 基于 ECC 的密码系统研究与设计. 微电子学与计算机，2003,20(12):51～53
8 周玉洁，冯登国. 公开密钥密码算法及其快速实现. 北京：国防工业出版社，2002
9 Hankerson D, et al. Software Implementation of Elliptic Creve Cryptography Over Binary Fields:[Technical report CORR 2000-42]. University of Wtaerloo, 2000. http://www.cacr.math.uwaterloo.ca