ISSN 0868–5886 НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 2000, том 10, № 4, с. 72–78

ОРИГИНАЛЬНЫЕ СТАТЬИ

УДК 621.333

© Ю. М. Шерстюк

МОДЕЛЬ И АЛГОРИТМ ПОСТРОЕНИЯ СИНТАКСИЧЕСКИ УПРАВЛЯЕМОГО РЕДАКТОРА XML-ОПИСАНИЙ

Для построения редакторов XML-файлов может быть использован синтаксически управляемый перевод. Главной целью создания XML-редакторов является предоставление развитого интерфейса пользователя. Такой интерфейс должен позволять получать в каждой точке диалога полную информацию о возможных действиях пользователя по редактированию файла, включая допустимые к использованию теги XML. В основу создания такого редактора может быть положен логический процессор, являющийся реализацией специального абстрактного автомата. Управляющая таблица автомата строится по содержимому DTD-файла на основе вычисления модифицированных отношений Вирта—Вебера. Данный подход позволяет пользователю создавать синтаксически корректные XML-файлы без знания DTD.

ВВЕДЕНИЕ

Язык XML (eXtensible Markup Language, "Расширяемый язык разметки") представляет собой компактное упрощенное подмножество языка SGML (Standard Generalized Markup Language), разработанное Консорциумом W3C [1, 2].

Занимая промежуточное место между SGML и HTML (HyperText Markup Language), язык XML предназначен для описания других языков: в отличие от HTML, в XML нет ни одного заранее определенного тега с фиксированным значением, перечень тегов и синтаксис задаются в виде специального описания применительно к каждому конкретному использованию языка. Поддержка XML в различных средах (и в первую очередь — в распространенных браузерах IE 5.0 и NetscapeNavigator, средах программирования Delphi, C++, Python, Jawa и др.) через интерфейсы DOM и SAX позволяет реализовать процедурную семантику произвольного языка, порожденного на базе XML. Это объясняет повышенный интерес к XML как к лингвистическому средству создания специализированных прикладных (предметных) языков представления информации. Такие языки, создание которых заключается в формировании Document Type Definition (DTD) в рамках стандарта XML, получили обобщающее название "XML-приложения" и в настоящее время находят все более широкое применение в самых различных областях.

ПОСТАНОВКА ЗАДАЧИ СОЗДАНИЯ XML-РЕДАКТОРА

Динамичность синтаксиса языка (а точнее — потенциальная бесконечность множества XML-приложений) порождает ряд проблем, одной из которых является создание адаптируемого редактора XML-документов.

В качестве основного требования к такому редактору выступает возможность формирования и модификации состоятельных XML-файлов, т.е. файлов, содержание которых отвечает требованиям того или иного DTD без каких-либо изменений программного кода редактора. Состоятельный XML-файл (XML-документ) всегда создается и обрабатывается в рамках конкретного XML-приложения, а XML-редактор по возможности должен носить достаточно общий характер, ибо разработка уникальных программ редактирования в интересах конкретных приложений в общем случае нерентабельна.

На практике возможны низкоуровневый и высокоуровневый подходы к организации работы XML-редакторов. Однако низкоуровневый подход (непосредственная работа с текстовым файлом, содержащим теги XML) малоэффективен. В свою очередь, применение высокоуровневого подхода, прежде всего предполагающего высокоразвитый интерфейс пользователя, весьма затруднено ввиду того, что XML — метаязык, порождающий языки с различными синтаксисом и семантикой. Если разработчики HTML-редакторов ориентировались на неизменный набор тегов языка HTML, определений их семантики и правил композиции, то "универсальный" XML-редактор должен создаваться на совершенно иных принципах, обеспечивающих возможность его применения в потенциально неограничиваемой области XML-приложений.

Опыт использования XML и результаты анализа возможностей создания инструментов работы с XML-документами позволили сформулировать

основные требования к подобному "универсальному" XML-редактору.

ОСНОВНЫЕ ТРЕБОВАНИЯ

Разрабатываемый XML-редактор должен обеспечивать:

— формирование синтаксически правильных XML-документов, состоятельных для указанного (выбранного) DTD;

— освобождение пользователя от необходимости знания содержания DTD: редактор должен осуществлять постоянное информирование пользователя о допустимых действиях по редактированию файла, которые не нарушат синтаксической правильности формируемого файла;

— возможность использования предметных (прагматических) синонимов для тегов и их атрибутов, позволяющая пользователю работать в рамках понятийного аппарата той или иной предметной области;

— наглядное представление структуры редактируемого XML-документа (как правило — древовидное);

— управление отображением элементов, их атрибутов и содержимого;

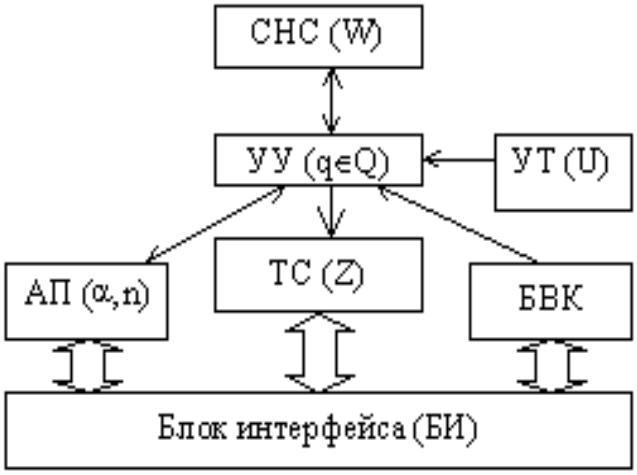
— возможность подключения и использования процедур, определяемых пользователем, для выполнения различных специфических операций над редактируемым XML-документом и его компонентами (фрагментами, элементами, атрибутами).

Процедуры, определяемые пользователем (UDP — User Defined Procedure) призваны обеспечить разнообразную обработку как всего редактируемого файла, так и его отдельных компонентов, которая может потребоваться в интересах того или иного XML-приложения. Так, например, для элемента *picture* можно предусмотреть просмотр содержащегося в нем графического файла непосредственно в процессе редактирования подмножества XML с DTD, разрешающим использование элемента *picture*. Для этого необходимо разработать соответствующую процедуру просмотра (внешнюю по отношению к редактору) и указать редактору на ее связь с элементом picture.

СТРУКТУРА РЕДАКТОРА И ОСНОВНЫЕ ПРИНЦИПЫ ЕГО ФУНКЦИОНИРОВАНИЯ

Сформулированные требования к XML-редактору определили следующие основные принципы его функционирования:

— в любой, произвольный момент времени пользователь может выполнять только те операции редактирования, которые не нарушат состоятельности редактируемого XML-документа (т.е. редактируемый документ всегда состоятелен);



Структура процессора

— адаптация редактора на конкретные DTD, процедуры, определяемые пользователем, и средства отображения должны осуществляться путем изменения информации в управляющей таблице, т.е. без изменения в программе редактора.

Для создания формальной модели редактора, отвечающего указанным выше требованиям, может быть использован подход, основанный на идеях синтаксически управляемого перевода и метауправления [3].

Синтаксически управляемый XML-редактор может быть представлен в виде абстрактного процессора, структура которого изображена на рисунке.

В состав процессора входят следующие блоки.

Ассоциативная память (АП). Потенциально бесконечная АП содержит дерево тегов и нетерминальных символов обрабатываемого XML-описания. Один из тегов является фокусным. Понятие нетерминального символа описано ниже.

Устройство управления (УУ) имеет память состояний и обеспечивает выполнение тактов функционирования процессора.

Управляющая таблица (УТ) содержит информацию, необходимую для вычисления значений индикаторов доступных операций на табло состояния и выполнения этих операций. Ее состав приведен ниже.

Табло статуса (ТС) отображает перечень команд пользователя, доступных в данный момент для выполнения.

Табл. 1. Команды пользователя по управлению процессором

Тип команд	Группа команд	Команда
Навигация	Вертикальная	Вверх (к старшему тегу) Вниз (к младшему тегу)
	Горизонтальная	Предыдущий Следующий
	Переход	На неопределенный символ
Формирование и модификация	Потомки	Порождение (o1) Добавление потомков в начало (o2) Добавление потомков в конец (o3) Удаление всех потомков (o4)
	Нетерминал	Вывод из нетерминала (o5)
	Вставка	Перед фокусом (o6) После фокуса (o7)
	Обработка группы фокуса	Удаление (o8) Свертка к нетерминалу (o9) Замена альтернативой (o10) Префиксный повтор группы (o11) Постфиксный повтор группы (o12)

Блок ввода команд (БВК) обеспечивает считывание команды пользователя и передачу ее в устройство управления.

Блок интерфейса (БИ) обеспечивает отображение АП, ТС для пользователя и ввод команд пользователя в БВК.

Список неопределенных символов (СНС) содержит список тех тегов, которые должны содержать подчиненные (вложенные) теги, на данный момент отсутствующие, а также нетерминальные символы, имеющиеся в АП.

Процессор представляет собой формальную систему As=(Q, U, q0, S, Z, R, W), компонентами которой являются:

Q — множество состояний устройства управления;

U — управляющая таблица;

q0 — начальное состояние устройства управления;

S — главный символ XML-описания (значение параметра *doctype* тега **<XML>**);

Z — алфавит статусов;

R — алфавит команд;

W — алфавит списка неопределенных символов.

Конфигурация ТС определяется совокупностью (q, a, n, z, r, w), где q — состояние УУ, a — фокусный элемент АП, n — индекс фокусного элемента в АП, z — индикация на табло статуса, r — команда пользователя, w — содержимое СНС.

Для процессора определены следующие глобальные команды пользователя:

— создание нового XML-описания: АП содержит единственный элемент, определяемый значением *doctype*;

— загрузка XML-описания из файла;

— сохранение XML-описания и (или) выход (завершение работы) допустимы, если УУ в состоянии q0; сопровождаются записью тегов и их параметров из АП в указанный файл.

Типовой цикл работы процессора содержит ряд тактов по выполнению следующих действий:

— определение строки статуса и вывод ее на ТС;

— ожидание команды пользователя, ее анализ и выполнение;

— возврат в состояние q0 из состояний, соответствующих завершению выполнения команды.

Потенциально возможными командами пользователя являются команды навигации (перемещения фокуса), а также команды обработки, перечисленные в табл. 1.

РЕАЛИЗУЮЩИЙ АВТОМАТ

Главная сложность в реализации процессора состоит в определении возможности выполнения тех или иных команд из приведенного выше перечня для любого положения фокуса. Выполнение команды возможно, если в результате совершае-

мых действий редактируемый файл остается синтаксически правильным, т.е. его синтаксис будет продолжать соответствовать DTD-описанию.

С этих позиций редактирование XML-файла необходимо рассматривать в виде совокупности процессов вывода (порождения тегов) и редукции (удаления тегов) в рамках грамматики более старшего в иерархическом отношении тега.

Поскольку в XML структура тега не зависит от контекста этого тега, для каждого описанного в DTD-файле тега можно автоматически сформировать соответствующую порождающую грамматику. Процесс порождения может быть реализован соответствующим автоматом с магазинной памятью, реализующим алгоритм "перенос—свертка". Именно в порождающих грамматиках появляются нетерминальные символы, обозначающие допустимые последовательности и альтернативы в структуре тегов.

Правила порождающей грамматики служат исходными данными для вычисления отношений предшествования (отношений Вирта—Вебера) [3]. Однако задача анализа контекста фокусного элемента в редактируемом файле отлична от обычной задачи редукции основ — поиск контекста должен производиться в обоих направлениях. Эта специфика привела к необходимости переопределения отношения "конец основы" и введения нового отношения — "другая основа". По матрице отношений предшествования вычисляется матрица отношения "допускает вставку". Особенностью формирования указанных матриц является необходимость обработки DTD-описаний вида A+ и A*, при наличии которых порождающая грамматика становится рекурсивной и укорачивающей (+ — знак транзитивного замыкания, * — рефлексивного).

Указанные выше особенности процессов вывода и редукции определили следующий алгоритм построения управляющей таблицы редактора.

Шаг 1. Построение грамматик для тегов, описанных в DTD-файле.

Обработке подлежат все теги **<!ELEMENT>** в DTD-файле, за исключением имеющих описатели EMPTY и #PCDATA. Теги **<!ELEMENT>** с описателем ANY в DTD-файле недопустимы.

Обработка каждого тега **<!ELEMENT>** заключается в его синтаксическом анализе и порождении грамматики, описывающей его синтаксис. С этой целью используется аппарат семантических процедур и специальный магазин (подробное описание процесса порождения с использованием семантических процедур изложено в [3]). Правила продукции грамматики, применяемой для разбора тегов **<!ELEMENT>**, а также соответствующие им семантические процедуры представлены в табл. 2. В описании семантических процедур в табл. 2 используются:

— процедура *putstack* (ps): помещает в магазин свой аргумент;

— функция *getstack* (gs): осуществляет выброс верхнего элемента из магазина и возвращает его в качестве результата вызова;

— функция *Ind*, возвращающая уникальный номер, единый в рамках обработки одного правила продукции;

— функция [ ], помещающая аргумент в выходной поток синтаксического анализатора;

— символ пустой цепочки e.

Порожденные правила грамматики помещаются в первую часть управляющей таблицы в виде U1[тег], где "тег" — имя тега, определяемого обработанным элементом **<!ELEMENT>**.

Шаг 2. Минимизация порожденных грамматик.

Для каждого тега, имеющего грамматику в первой части управляющей таблицы, выполнить следующие действия над правилами этой грамматики:

— удалить все правила вида A $\rightarrow$ тег, для которых альтернатив вывода из символа A нет, и все вхождения A в другие правила заменить на "тег";

— если есть два правила с одинаковыми правыми частями ($A1 \rightarrow \alpha$ и $A2 \rightarrow \alpha$, либо $A1(+) \rightarrow \alpha$ и $A2(+) \rightarrow \alpha$), второе правило удалить, а все вхождения A2 заменить на A1 (при этом $\alpha \neq e$, т.е. не является пустой цепочкой);

— если есть правила вида $A1 \rightarrow A2$, $A2 \rightarrow \alpha$, где A1, A2 — нетерминальные символы, и нет других правил с левой частью A1 и A2, то заменить их правилом вида $A1 \rightarrow \alpha$ с заменой в оставшихся правилах A2 на A1;

— если есть правила вида $A1(+) \rightarrow A2$, $A2 \rightarrow \alpha$, где A1, A2 — нетерминальные символы порожденной грамматики, и нет других правил, в которых A2 используется как в левой, так и в правой частях, то заменить их правилом вида $A1(+) \rightarrow \alpha$, за исключением случая, когда A1 — начальный символ грамматики;

— если есть правила вида $A1 \rightarrow A2$, $A2(+) \rightarrow \alpha$, где A1, A2 — нетерминальные символы порожденной грамматики, и нет других правил, где используется A2 как в левой, так и в правой частях, то заменить их правилом вида $A1(+) \rightarrow \alpha$, за исключением случая, когда A1 — начальный символ грамматики.

Табл. 2. Правила продукции грамматики для разбора описателей **<!ELEMENT>**

Правило	Семантические процедуры
S → (P)	НЕТ
S → (PT	a:=gs; b:=gs; ps(b a)
T → ,P)	НЕТ
T → ,PT	a:=gs; b:=gs; ps(b a)
C → (PQ	a:=gs; b:=gs; ps(b | a)
Q → | P)	НЕТ
Q → | PQ	a:=gs; b:=gs; ps(b | a)
P → тег?	["группа"Ind → тег | e], ps("группа"Ind)
P → тег+	["группа"Ind(+ → тег], ps("группа"Ind)
P → тег*	["группа"Ind(+) → тег, "группа"Ind →e], ps("группа"Ind)
P → E?	["группа"Ind → gs | e], ps("группа"Ind)
P → E+	["группа"Ind(+ → gs], ps("группа"Ind)
P → E*	["группа"Ind(+) → gs, "группа"Ind(+) → e], ps("группа"Ind)
N → E?	["имя" → gs | e], ps("имя")
N → E+	["имя"(+) → gs], ps("имя")
N → E*	["имя"(+) → gs, "имя" → e], ps("имя")
P → тег	ps(тег)
P → E	НЕТ
N → E	[N → gs]
E → S	["группа"Ind → gs], ps("группа"Ind)
E → C	["группа"Ind → gs], ps ("группа"Ind)

Шаг 3. Построение матрицы отношений следования и матрицы вставки.

Для каждого тега, имеющего грамматику в первой части управляющей таблицы, выполнить действия по формированию матрицы отношений следования по содержимому правил этой грамматики.

Пусть "тег" — имя обрабатываемого тега, U — совокупность правил продукции, порожденных для этого тега на шаге 1 и минимизированных на шаге 2.

Шаг 3.1. Подготовка рабочей копии правил продукции.

Правила вида "тег → α" либо "тег(+) → α" заменить на правила вида "N → α" либо "N(+) → α" соответственно, т.е. N становится новым начальным символом грамматики вместо "тег".

Шаг 3.2. Обработка e-правил.

Выполнить удаление e-правил, используя соответствующий алгоритм эквивалентного преобразования грамматик. Пример такого алгоритма приведен в [2] на стр. 29.

Шаг 3.3. Обработка правил вида A(+) → α.

Каждое правило вида "A(+) → α", где A — не начальный символ грамматики, заменить на

"$A \to \alpha$", а все вхождения A в другие правила заменить на цепочку из двух символов "AA".

Шаг 3.4. Построение матриц [$FIRST^+$] и [$LAST^+$].

Размерность всех матриц, используемых далее в алгоритме M×M (где M — мощность алфавита грамматики), если не указана другая размерность.

Определение отношений FIRST и LAST, правила построения булевых матриц этих отношений ([FIRST] и [LAST]) и их транзитивных замыканий ([$FIRST^+$] и [$LAST^+$]) приведены в [3].

Шаг 3.5. Построение матриц отношений следования [=], [<], [>], [#].

Отношения "=", "<", ">" и "#" определяются на множестве символов грамматики следующим образом:

— $X = Y \longleftrightarrow A \to \alpha XY\beta$: имеется правило, в правой части которого за X непосредственно следует Y (α, β — произвольные, возможно, пустые, цепочки);

— $X < Y \longleftrightarrow A \to \alpha XB\beta$, $B \to {}^+Y\gamma$: Y начинает цепочку, выводимую из B, который следует за X в каком-либо правиле;

— $X > Y \longleftrightarrow A \to \alpha BY\beta$, $B \to {}^+\gamma X$: X заканчивает цепочку, выводимую из B, который следует перед X в каком-либо правиле;

— $X \# Y \longleftrightarrow A \to \alpha BR\beta$, $B \to {}^+\gamma X$, $R \to {}^+Y\delta$: X заканчивает цепочку, выводимую из B, который в каком-либо правиле следует перед R, из которого выводится цепочка, начинающаяся с Y.

Исходя из введенных определений:

— матрица [=] строится путем просмотра правых частей всех правил: если в правой части есть символ a, непосредственно за которым следует символ b, то [=](a, b):=1;

— матрица [<]:= [=] * [$FIRST^+$];

— матрица [>]:= [$LAST^+$]$^{-1}$ * [=];

— матрица [#]:= [$LAST^+$]$^{-1}$ * [=] * [$FIRST^+$].

Шаг 3.6. Построение обобщенной матрицы отношений следования.

Обобщенная матрица отношений следования для каждого тега является второй частью управляющей таблицы — U2(тег). Каждый ее элемент формируется на основе матриц [=], [<], [>], [#]: если в какой-либо ячейке одной из этих матриц имеется значение "1", то в соответствующую ячейку обобщенной матрицы заносится символ отношения [=], [<], [>] или [#].

Если имеет место наложение отношения "#" с одним из "<", "=", ">", то берется символ из "<", "=", ">". Если же имеет место наложение одного из отношений "<", "=", ">" с другим из их числа, то обобщенная матрица отношений построена быть не может — необходимо изменить описание тега в DTD-файле. Имеется формальное доказательство следующей теоремы, позволяющей формировать изначально допустимые описания тегов в DTD-файле: для описания тега вида **<!ELEMENT** A B> матрица отношений следования всегда может быть построена, если ни в одной из альтернатив описания, составляющих B, никакой тег не повторяется.

Шаг 3.7. Построение матрицы вставки.

Определим отношение "перед": X перед Y $\longleftrightarrow A \to aBCd$, B [LAST*] X, B = C, C [FIRST*] Y — есть сентенциальная форма, в которой X непосредственно предшествует Y. Тогда [перед]:= [$LAST^*$]$^{-1}$ * [=] * [$FIRST^*$]. Отношение "между" определим на основе использования отношения "перед": если X перед Y, Y перед Z, и $Y \to {}^*e$, то между (X, Z)=Y.

Тогда вычисление матрицы вставки, являющейся булевой матрицей отношения "между", производится следующим образом:

— в множество правил продукции добавить правило $N \to eNe$, где N — начальный символ грамматики, e — символ пустой цепочки;

— вычислить [$LAST^*$]$^{-1}$:= [$LAST^+$]$^{-1}$ + I, [$FIRST^*$]:= [$FIRST^+$] + I, где I — единичная матрица;

— вычислить [перед]:= [$LAST^*$]$^{-1}$ * [=] * [$FIRST^*$];

— вычислить матрицу вставки как матрицу U3(Na×Na), где Na — количество терминальных символов, включая e, используя выражение

— $(\forall i)(\forall j)(\forall k)(1 \le i,j,k \le Na)$ ([перед](i,k) $\wedge$ [перед](k,j) $\wedge$ (k-й символ есть нетерминал) $\to$ U3(i,j)=k-й символ).

Совокупность порождающей грамматики (U1), матрицы отношений предшествования (U2) и матрицы отношения "допускает вставку" (U3) составляют управляющую таблицу процессора, индексированную по именам тегов. Отдельным разделом (U4) в управляющую таблицу могут входить синонимы тегов и атрибутов, а также имена UDP.

Наличие управляющей таблицы приводит к тому, что при любом положении фокуса (для любого фокусного элемента) в рамках порождающей грамматики старшего тега определяются:

— группа фокусного элемента (ГФЭ) — основа, которой принадлежит фокусный элемент;

— левый контекст (ЛК) и правый контекст (ПК) группы фокусного элемента — соответственно левая и правая основы.

Границы основ находятся по отношению "другая основа". Для ГФЭ, ЛК и ПК выполняется их свертка к нетерминальному символу порождающей грамматики. Далее по правилам грамматики определяются возможность удаления символа,

альтернативы для замены символа и т.д., а по матрице "допускает вставку" определяются множества символов, которые могут быть помещены между ГФЕ и ее контекстами. Таким образом, любое перемещение фокуса в общем случае вызывает выделение и редукцию трех основ (групп).

ЗАКЛЮЧЕНИЕ

Алгоритм выполнения операций пользователя по редактированию XML-файла при известных основах и результатах их редукции достаточно прост, а любая операция редактирования есть вывод или редукция в сентенциальной форме порождающей грамматики тега, старшего для фокусного элемента.

Все описанные выше элементы построения и функционирования редактора имеют алгоритмическое решение. Опытная реализация редактора XML-описаний, осуществлявшаяся для апробации предложенного подхода, выполнена с использованием языка Python и имеющихся для него синтаксических анализаторов XML и DTD-файлов. Для сопряжения с UDP разработан специальный Application Programm Interface (API).

СПИСОК ЛИТЕРАТУРЫ

1. *Сандра Э. Эдди.* XML: Справочник. СПб.: Питер, 1999. 312 с.
2. *Питтс Н.* XML за рекордное время: Пер. с англ. М.: Мир, 2000. 444 с.
3. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. М.: Мир, 1977. Т 1, 2. 1104 с.

Военный университет связи, Санкт-Петербург

Материал поступил в редакцию 20.10.2000.

MODEL AND ALGORITHM FOR CONSTRUCTING A SYNTAX-DIRECTED EDITOR OF XML DESCRIPTIONS

Yu. M. Sherstyuk

Military University of Communication, Saint-Petersburg

Syntax-directed translation may be used to build XML file editors. The main purpose of intelligent XML editors is to provide a high-level user interface. Such interface must ensure full information on possible user actions at each point of dialogue and on appropriate XML tags. The editor can be built around a logical processor — realization of a special abstract automaton. The control table of this automaton is built using the contents of the DTD file and based on calculation of the modified Wirth—Weber ratios. This approach allows the user to create syntactically correct XML files without knowledge of DTD.