Свойства схем данных XML*

Л.Г. Новак, С.Д. Кузнецов

1. Введение

Расширяемый Язык Разметки (XML) [1] становится в последнее время доминирующим стандартом представления и обмена данными в Интернете. Подобно языку HTML, XML является поднабором языка SGML. Однако существует набор фундаментальных отличий XML от других языков разметки, одно из которых заключается в том, что разметка документа является семантической. Перечислим основные свойства языка XML [2]:

- *Независимый формат данных*. При использовании XML как формата выходных и входных данных приложения, данные становятся независимы от самого приложения, что повышает способность взаимодействия.
- *Одни данные, несколько представлений*. В силу того, что формат не зависит от приложения, очевидно, что одни и те же данные можно отображать разными способами и разными приложениями.
- *Улучшенные возможности поиска данных*. Поскольку XML определяет семантическую структуру документа, это способствует созданию дополнительных возможностей для поиска информации. Например, индексирующие и поисковые средства могут работать не только с самими данными, но и с разметкой (*метаданными*)
- *Облегчение доступа к данным*. В мире в настоящее время существует большое количество информации, доступ к которой затруднен из-за того, что она хранится в разнородных и несовместимых форматах. Перевод в формат XML откроет доступ к таким данным.
- *Более простая разработка приложений*. XML делает необязательной реализацию поддержки большого количества бинарных форматов, вследствие чего разработка приложений становится значительно проще.
- *Использование готовых решений*. При управлении данными, как правило возникает необходимость решения «шаблонных» задач, таких, как верификация данных, лексический и синтаксический разбор, и т.д. Переход к формату XML способен убрать необходимость создания своих собственных реализаций для решения подобных задач.
- *XML файл может быть прочитан человеком*.
- *XML поддерживается большим количеством стандартов*. Эти стандарты предназначены для того, чтобы гарантировать совместимость приложений пользователя и готовых решений. В их число входят стандарты API для лексического и синтаксического анализа (SAX), стандарт для управления объектной моделью документа (DOM) и другие стандарты, которые будут рассмотрены ниже.

Как видно из перечисленных свойств, в XML заложен потенциал для революционных изменений в области хранения и обработки данных.

Из множества задач программирования, решаемых с помощью XML, можно выделить подкласс, включающий такие задачи, как обмен сообщениями между приложениями, управление данными некоторой предметной области, хранение полуструктурированных данных и т.д. [2,3]. Для каждой отдельно взятой задачи необходимо сужение допустимого множества XML-документов. Поэтому зачастую необходима спецификация множества возможных XML-документов, которые будут обрабатываться. Для этого используются словари XML-документов, по сути являющиеся схемами данных, представленных в формате XML. Здесь необходимо заметить, что логическая модель данных XML довольно близка к модели полуструктурированных данных [4].

Схемы XML (как и схемы данных, представленных в терминах любой другой модели данных) предназначены для описаний структурных и семантических ограничений, которые должны выполняться в любом экземпляре данных, соответствующем этой модели. Про множество данных, на которых выполняются ограничение схемы, говорят, что оно *удовлетворяет* данной схеме. Характерным примером структурного ограничения для XML-документов является спецификация содержания элементов (например, *элемент с именем A может содержать только элементы с именем B*), а примером семантического ограничения – спецификации ключей (*атрибут a, содержащийся в элементе A, должен обладать уникальным значением среди множества значений атрибутов a*).

В текущее время существует около дюжины языков спецификаций схем данных, поддерживаемых различными организациями (W3C, OASIS и т.д.). Наиболее популярными языками являются DTD [1], XML Schema [5], RelaxNG [6], XDR [7]. Как отмечалось выше, все они предназначены для спецификации структурного и семантического содержания XML документов. Несмотря на то, что языки спецификации схем достаточно сильно различаются по выразительной силе, многие принципы построения ограничений схожи. В большей степени это касается структурных ограничений. Поэтому многие свойства структурных ограничений в терминах одного языка имеют свои аналоги для другого языка спецификации. Язык DTD является одним из

* Работа частично поддержана грантом РФФИ 02-01-01088-а

наиболее простых языков спецификации схем. Выразительная мощность этого языка существенно ниже, чем в других [17], однако этот язык имеет определенные преимущества. Во-первых, DTD является де-факто стандартом спецификации схем (по крайней мере, пока стандарт XML Schema не будет окончательно утвержден). В большом количестве алгоритмов и программных средств, предназначенных для управления XML-данными, для определения используется именно DTD. Во-вторых, схемы DTD можно однозначно отобразить на другие XML-схемы. Стоит заметить, что решения многих задач управления данными XML инвариантны по отношению к выбору языка спецификаций, что облегчает переход от одной схемы к другой.

Основной целью данной работы является изучение свойств XML-схем и методов преобразования схем данных над моделью XML, которые могут быть использованы в качестве вспомогательного инструмента для создания и реализации некоторых задач, связанных с управлением данными и мета-данными XML. Основная идея нашего подхода заключается в разработке методики, позволяющей существенно упростить реализацию этих алгоритмов, за счет выделения подклассов из всего многообразия XML-схем. Выделенный подкласс должен обладать следующим свойством: реализация алгоритма для схем, принадлежащих данному подклассу, существенно упрощается по сравнению с алгоритмом, работающим на всем многообразии схем XML. Схемы, принадлежащие определенному подклассу, называются *схемами, представленными в нормальной форме*.

Статья организована следующим образом. В начале мы приводим формальное определение структурных частей (сигнатур) схем XML, основанное на регулярных грамматиках деревьев. Преимущество такого представления структурных ограничений заключается в том, что любую схему, выраженную на языке регулярных грамматиках деревьев, можно отобразить на существующие языки спецификаций схем XML и наоборот, структурные ограничения, выраженные на наиболее распространенных языках спецификаций схем выразимы с помощью регулярных грамматик деревьев. В следующем разделе, мы приводим классификацию типов регулярных грамматик и их соответствие языкам спецификаций схем XML. Далее мы вводим преобразования схем XML, приводящие их к эквивалентному виду (с точностью до отношения эквивалентности). В четвертом разделе мы определяем *нормальные формы схем* и приводим теоремы существования нормальных форм для любой схемы. Затем, мы обсуждаем методы использования алгоритмов нормализации для решения практических задач, связанных с управлением XML-данными, в частности для построения отображения моделей данных. Наконец, в последнем разделе, мы описываем логические языки, предназначенные для формулирования ограничений целостности XML.

2. Структурные ограничения XML

В этом разделе мы приводим формальное определение схем, состоящих из структурных ограничений, и формулируем термин «*валидируемость*». Также мы приводим определения эквивалентности схем и отношения порядка на схемах, которые будут использоваться в дальнейшем. Раздел начинается с определения регулярных выражений хорошо известных из литературы по грамматикам и языкам программирования.

Определение 1 (*Регулярные выражения над множеством символов E*). Множество регулярные выражения над множеством E (**reg(E)**)определяются следующим образом:

1) ε - регулярное выражение, где ε обозначает «пустой список».
2) $\forall e \in E$: e- регулярное выражение.
3) Если e_1 – регулярное выражение, то (e_1), e_1*, e_1?, e_1+ - тоже регулярные выражения.
4) Если e_1 и e_2– регулярные выражения, то e_1,e_2 и $e_1|e_2$ - тоже регулярные выражения

Определение 2*(Порождаемые последовательности)*. Пусть r- регулярное выражение над множеством E. Тогда конечная (м.б. пустая) последовательность $s=[e_{0,..,}e_n]$ символов, где $e_i \in E$, порождается выражением r (s|=r), тогда и только тогда, когда выполняется одно из следующих соотношений:

1) Если $r \equiv \varepsilon$ и $s \equiv []$ – пустая последовательность;
2) $\exists e \in E$: $r \equiv e$ и $s \equiv [e]$;
3) $r \equiv (r_1)$ и $s|=r$;
4) $r \equiv r_1$* и $s \in S$, где $S \equiv \{[],[s_{1,..,}s_n]\}$, где $\forall i : s_i |=r_1$;
5) $r \equiv r_1$+ и $s \in S$, где $S \equiv \{[s_{1,..,}s_n]\}$, где $\forall i : s_i |=r_1$;
6) $r \equiv r_1$? и $s \in S$, где $S \equiv \{[],[s_1]\}$, где $s_1 |=r_1$;
7) $r \equiv r_1,r_2$ и $s \equiv [e_{0,...,}e_k, f_{0,..,}f_n]$, где $[e_{0,...,}e_k] |=r_1$ и $[f_{0,..,}f_n] |=r_2$;
8) $r \equiv r_1|r_2$ и $s |=r_1$ или $s |=r$.

Множество всех порождаемых последовательностей регулярного выражения r над множеством E называется *регулярным множеством* и обозначается так: $\Re_E(r)$.

Пример 1.

Пусть E={0,1}.

Множество последовательностей, порождаемых регулярным выражением (0|1)(0|1) состоит из множества последовательностей длины 2, содержащих элементы 0 и 1:

[0,0];[0,1];[1,0];[1,1].

Регулярное выражение (0|1)* порождает множество последовательностей произвольной длины, состоящих из 0 и 1, то есть полное множество всех последовательностей над множеством E. □

Определение 3 *(Эквивалентность регулярных выражений*). Пусть $r_1,r_2 \in reg(E)$. Тогда

$$r_1 \cong r_2, \text{ если } \forall\ s = [e_0,..,en]\ :\ s|= r_1 \Leftrightarrow s|= r_2$$

Определение 3' *(Эквивалентность регулярных выражений*). Пусть $r_1,r_2 \in reg(E)$. Тогда

$$r_1 \cong r_2, \text{ если } \Re_E(r_1) \Leftrightarrow \Re_E(r_2)$$

Пример 2.

Регулярные выражения r*,r и r+ эквивалентны, где r – произвольное регулярное выражение над множеством r.

Покажем это. Пусть s|= r+ . Тогда по определению 2 $s \equiv [s_{1,..,}s_n]$, где $\forall i : s_i |= r$. Тогда $s_1|=r$ и $[s_2,..,s_n]|=r^*$ и, значит, s|= r*,r . В обратную сторону утверждение доказывается аналогично. □

Теорема 1 (Замена выражений). *Пусть f_1 и f_2 – есть идентичные регулярные выражения над множеством $\{E,r_1\}$ и $\{E,r_2\}$, соответственно ,где r_1 и r_2 – обозначения регулярных выражений над множеством E (f_1 получается из f_2 путем замены символа r_2 на r_1 и наоборот). Пусть f'_1 и f''_2 - это два регулярных выражения над множеством E, получаемые, соответственно, из f_1 и f_2, с помощью замены символов r_1 и r_2 на регулярные выражения над множеством E. Тогда $r_2 \cong r_1 \Rightarrow f''_1 \cong f.''$* □

Например, из этой теоремы следует, что выражение a|a+ ≅ a|(a*,a), так как a+ ≅ (a*,a).

Определение 4 (*Структурные схемы XML документов*). [12] Структурная схема XML документов есть совокупность (T,E,A,p,a,r), где:

- T – множество, состоящее из всевозможных доменов.
- E - множество типов элементов; тип элемента состоит из имени и условного обозначения, являющегося уникальным идентификатором типа.
- A – множество типов атрибутов. Каждый тип включает в себя:
  - имя атрибута,
  - домен принимаемых значений
  - идентификатор обязательности (должен ли атрибут быть заполнен)
  - уникальный идентификатор типа атрибута.
- p есть функция из множества E в reg({E,T}). *p:E→ reg({id(E),T})*, где id(E)- множество уникальных идентификаторов типа элемента.
- a есть функция из множества E в множество всех подмножеств множества A – pows(A). a: *E→pows(A),* причем для любого типа элементов e типы атрибутов из множества a(e) должны обладать уникальным именем.
- r ∈ E и называется *типом корневого элемента*. Для Множества E должно быть соблюдено следующее условие: $\forall e_0 \in E$, e ≠r ∃ $(e_0,e_{1,..,}e_n)$: $\forall i<n\ e_i \in E$ и ∃ последовательность $s=[a_0,..,a_{j-1},e_i,a_j,..a_n]$ $s \in \Re_{\{id(E),T\}}(p(e_{i+1}))$, e_n=r. Это условие означает «достижимость до любого элемента от корня».

Определение, данное выше, является достаточно универсальным способом спецификации структурных ограничений схем XML. Достаточно легко показать, что структурные ограничения, заданные выражениями на таких языках спецификации схем, как XML Schema, DTD, Relax NG отображаются в структурные схемы. В качестве примера, мы приведем пример отображения схемы, выраженной на языке DTD в структурную схему:

Пример 3.

```
<!ELEMENT Product (Name, Developer?, Summary?,
Description?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Developer (#PCDATA)>
<!ELEMENT Summary (#PCDATA)>
<!ELEMENT Description (Para | List)+>
<!ELEMENT Para (#PCDATA | Link)*>
<!ELEMENT List (Item+)>
<!ELEMENT Item (#PCDATA | Link)*>
<!ELEMENT Link (#EMPTY)>
<!ATTLIST Link    URL CDATA #REQUIRED>
```

Данной схеме DTD соответствует структурная схема (T,E,A,p,a,r), где:

```
T≅{#PCDATA}
E≅{{Product, product}, {Name, name}, {Developer,
developer}, {Summary, summary}
,{Description,  description },{Para, para },{List,
list}, {Item, item}, {Link, link}}
```

(здесь и далее тип элемента представляется как пара – {имя, Идентификатор})

```
A≅{(URL, CDATA, Required,url)}

p:
p(product)= (name, developer?, summary?, description?)
p(name)= p(Developer)=p(Summary)= #PCDATA
p(description)= (para | list)+
p(para)=p(Item) = (#PCDATA | link)*
p(list)= Item+
p(link)= ε

a:
```

a(link)={URL}
a(product)=a(name)=...=a(list) ≅{}

r=product□

Таким образом, можно установить, что при отображении в структурную схему каждому имени элемента в DTD соответствует уникальный тип элемента. Множество T состоит из типа #PCDATA (Термин PCDATA обозначает произвольный набор символов, интерпретируемый синтаксическим анализатором как текстовый узел). Каждому атрибуту соответствует свой тип атрибута, значения которого устанавливаются согласно свойствам типа атрибута в DTD. Наконец, отображение p задается исходя из регулярных выражений, определяющих структуру элемента DTD. Однако, стоит отметить, что ограничения целостности, которые могут присутствовать в DTD (атрибуты типа ID или IDREF) никоим образом не отображаются на структурную схему. Ограничения целостности мы обсудим в последнем разделе работы.

Заметим, что в зависимости от регулярного выражения, соответствующего элементам их типы можно классифицировать следующим образом:

- элементы *пустого* содержания : $p(e) \in \{\varepsilon\}$,
- элементы, *содержащие данные* : $p(e) \in reg(T)/\{\varepsilon\}$,
- элементы *элементного* содержания: $p(e) \in reg(id(E))/\{\varepsilon\}$,
- элементы *смешанного* содержания : $p(e) \in reg(\{id(E),T\})/\{ reg(E) \cup reg(T)\}$

В нашем примере, link – это элемент пустого содержания, name, developer, summary – элементы содержащие данные, product, description и list – элементного содержания, и наконец para и item – смешанного.

Стоит заметить, что структурным схемам вида *(T,E,A,p,a,r)* однозначно соответствуют регулярные грамматики деревьев [13], если положить следующее:

- множество F и id(E) являются *нетерминальными* символами грамматики, где F – множество типов текстовых узлов, id(E) – множество уникальных идентификаторов типов элементов;
- T и name(E) – терминалы грамматики, где name(E) – множество имен элементов;
- Отображение p(e) заменяется правилом продукции одного из следующих двух видов: $x \to a\ r$, где $x \in id(E)$, $a \in name(E)$, $r \in reg(\{E,F\})$ или $x \to a\ \varepsilon$,где $x \in F$, $a \in T$.

В следующем разделе мы опишем классы регулярных грамматик и их соответствие языкам спецификаций схем.

Следующие определения описывают понятие *валидируемости* XML документа. Здесь и далее XML документ рассматривается в рамках модели XML, представленной в первой главе.

Определение 5 (*Интерпретация*). Интерпретация I XML документа D в терминах структурной схемы S=(T,E,A,p,a,r) – это набор отображений $I=(\phi,\varphi,\sigma)$, где:

- ϕ – это отображение E_D, -множества элементов документа, на множество E;
- φ - это отображение A_D, -множества атрибутов документа на множество A;
- σ - это отображение T_D, - множества текстовых узлов документа на множество T.

Также должны выполняться следующие условия:

- (*согласование имен элементов*) Пусть name – функция, ставящая в соответствие узлу документа его имя. Тогда $\forall e \in E_D$: $name(e)= name(\phi(e))$;
- (*согласование имен и значений атрибутов*) Пусть value–функция, ставящая в соответствие узлу документа его значение. Тогда $\forall a \in A_D$: $name(a)= name(\varphi(a))$, $value(a) \in dom(\varphi(a))$, где dom(x) – это домен принимаемых значений типа атрибута;
- (*согласование текстовых узлов*) $\forall t \in T_D$: $value(t) \in \sigma(t)$
- (*согласование атрибутов с элементами*) Пусть $A_e=\{a_i\}$ $i=[0,..,n_e]$ – множество атрибутов элемента e. Тогда $\forall e \in E_D$: $\forall i \in [0,..,n_e]$ $\varphi(a_i) \in a(\phi(e))$;
- (*согласование обязательных атрибутов*) $\phi^{-1}(e_s)$ – множество элементов документа D, которые отображаются в тип элемента e_s. Также пусть $R(e_s)$ – это подмножество $a(e_s)$, в которое входят те и только те типы атрибутов, у которых проставлен идентификатор обязательности. Тогда $\forall e_s \in E$ $\forall a_s \in R(e_s)$ $\forall e \in \phi^{-1}(e_s)$ $\exists a \in A_e : \varphi(a)= a_s$;
- (*согласование корневого элемента*) Для r_D – корневого элемента документа D : $\phi(r_D)=r$;
- (*согласование содержания элемента*) Пусть $C_e = [e_0,..,e_n]$ – есть упорядоченная последовательность элементов и текстовых узлов, вложенных в e. Тогда $\forall e \in E_D$: $\mathbf{I(e_0),.., I(e_n) \models p(\phi(e))}$, где $I(e_i)$ – это одно из двух отображений $\{\phi,\sigma\}$ (в зависимости от типа узла).

Определение 6 (*Валидность*). Документ D является *валидным* документом для структурной схемы S (*удовлетворяет схеме S),* если существует интерпретация I в терминах S (Обозначается D|=S).

Данное определение является ключевым для всего дальнейшего рассмотрения. Введем следующее обозначение: DB(S) – множество всех документов XML, удовлетворяющих данной схеме.

Утверждение 1 (Корректность валидности). *Пусть D – схема, выраженная на языке спецификации DTD и S – соответствующая ей структурная схема.*

Тогда DB(D)⊆DB(S). Если схема D не содержит ограничений целостности, тогда DB(D)=DB(S).

Для доказательства утверждения достаточно использовать свойства отображения схем DTD в структурные схемы (они очевидно следуют из примера 3). □

Аналогичные утверждения можно сформулировать и доказать для других языков спецификации схем.

Заметим, что далеко не всегда существует единственная интерпретация одного и того же документа. Нижеследующий пример демонстрирует случай множественной интерпретации одного и того же документа.

Пример 4.

На Рис. 1 представлена структурная схема и документ XML.

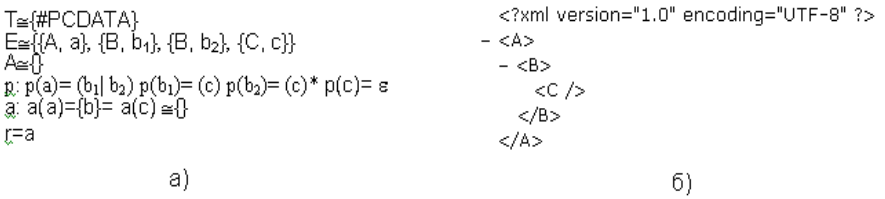

Рис. 1. *а) структурная схема; б) документ*

Документ XML содержит три элемента: A, B и C. Исходя из определения интерпретации, отображение I должно ставить в соответствие каждому элементу тип элемента из множества E с таким же именем, как и у элемента. Поэтому в любой интерпретации элементу A соответствует тип a, элементу C тип c. А вот для элемента B существует два разных типа в которые он мог бы отображаться b_1 и b Достаточно легко убедиться, что в обоих случаях будут выполняться условия интерпретации.□

Определение 7 (*Тривиальные схемы*). Структурная схема называется *тривиальной*, если существует и притом единственный XML документ, валидный для данной схемы.

Утверждение 2 (Существование тривиальной схемы). *Для любого XML документа существует тривиальная структурная схема, для которой данный документ валиден.*

Для доказательства утверждения достаточно воспользоваться индукцией по глубине документа XML – максимальному расстоянию от корня дерева XML до листа. База индукции при n= В этом случае документ XML должен иметь следующий вид (представление в терминах модели XML [1]) – Рис. 2.

Рис 2. *XML документ глубины 1*

Как видно из рисунка, все узлы дерева помимо корня являются листами. Для формирования структурной схемы, необходимо выполнить следующие действия:

1) Множество E формируется следующим образом: для каждого узла типа «элемент», мы создаем отдельный тип элемента.
2) Множество A формируется следующим образом: для каждого узла типа «атрибут», мы создаем отдельный тип атрибута. Доменный тип состоит из одного значения – значения данного узла в документе.
3) Множество T формируется следующим образом: для каждого текстового узла в документе мы создаем отдельный домен, состоящий только из одного значения.
4) Отображение a задается по следующим правилам: для любого типа элемента e – множество a(e) состоит из типов атрибутов, соответствующих атрибутам того элемента XML, который задавал e.
5) Отображение e задается по следующим правилам: для любого типа элемента e – p(e) – это выражение вида $(e_0,..,e_n)$ где e_i это либо тип элемента, либо домен, задаваемый i-м дочерним узлом того элемента XML, который задавал e.
6) Тип элемента r (корневой тип) задается корневым элементом дерева XML.

Легко убедиться, что исходный документ удовлетворяет данной схеме. Также любой XML документ, удовлетворяющий данной схеме, совпадает с исходным документом. То есть схема является тривиальной.

Индуктивный переход осуществляется следующим образом. Пусть утверждение доказано для документа, максимальная глубина которого равна n. Пусть у нас есть документ XML глубины n+1. В терминах XML модели, его можно представить в виде дерева глубины n+1. Рассмотрим множество поддеревьев, с корнями в дочерних узлах корневого документа исходного дерева. Их максимальная глубина не превышает n. По предположению индукции им ставятся в соответствие тривиальные схемы. Общая схема формируется путем объединения множеств E,T,A каждой из этих тривиальных

схем и продлением отображений a и p . Затем мы формируем еще один тип элемента r, соответствующий корню исходного XML документа, и продляем отображения a и p на него. Отображение a(r) возвращает множество атрибутов корневого элемента, а $p(r)=(r_{0,..,}r_n)$, где r_i – корневой тип элемента тривиальной схемы, порожденный i-м узлом. □

Способ создания тривиальной схемы, использованный в утверждении 2, задает инъективное отображение множества документов XML на множество схем. Этот результат используется в работе [15] для реализации алгоритмов трансляции выражений алгебры управления структурными схемами в выражения языка запроса к данным XML.

Легко показать, что все домены из множества T – доменных типов тривиальной схемы содержат в точности одно значение.

Лемма 1 (Достаточное условие тривиальности). *Любая схема S=(T,E,A,p,a,r) такая, что для любого типа элементов e, регулярное выражение p(e) имеет вид $r_1,..,r_n$, где r_i есть символы базового алфавита, является тривиальной или пустой схемы.*□

3. Классы регулярных грамматик

В этом разделе мы приводим классификацию структурных схем. Данный метод заимствован из работы [13], где он используется для классификации грамматик деревьев.

Определение 8 (*Локальные структурные схемы*). Структурная схема называется локальной, если не существует двух типов элементов с одинаковым именем.

Структурная схема из примера 3 является локальной, в то время как схема из примера 4 не является таковой. Следующее утверждение выполняется для локальных схем.

Утверждение 3 (Единственность интерпретации). *Пусть S=(T,E,A,p,a,r) локальная структурная схема и XML документ D валиден для S. Пусть также любые два домена из множества T не пересекаются и для любого типа элемента e, мультимножество имен типов атрибутов из множества a(e) содержит только уникальные значения. Тогда существует, и, притом единственная интерпретация документа D в терминах S.*

Существование интерпретации следует из самой формулировки утверждения. Для доказательства единственности воспользуемся формулировкой интерпретации. Из правила *согласования имён элементов*, и локальности схемы следует, что в любой интерпретации каждый элемент документа XML должен отображаться на один и тот же тип элемента, так как имена всех типов уникальны. Из того, что любые два домена не пересекаются и из свойства *согласования текстовых узлов* следует, что в любой интерпретации каждый узел документа XML должен отображаться на один и тот же домен. Таким

образом, достаточно проверить, что отображение атрибутов сохраняется в любой интерпретации. Это следует из свойств *согласования атрибутов с элементами*, *согласования имен и значений атрибутов* и из того, что для любого типа элемента e, мультимножество имен типов атрибутов из множества a(e) содержит только уникальные значения. □

Прежде чем описать следующий класс структурных схем, приведем следующее определение, относящееся к регулярным выражениям:

Определение 9 (*Допустимые символы*). Пусть r- регулярное выражение над множеством M. Тогда $\Delta_M(r)$ – это множество, содержащее все элементы из M, которые присутствуют в записи регулярного выражения.

Например, если E={0,1,2}, то $\Delta_M((0^*,1^*))$= {0,1}

Теорема 2 (Критерий допустимости). *Пусть r- регулярное выражение над E. Тогда*

$$\forall e \in E:\ e \in \Delta_M(r) \Leftrightarrow \exists s=[e_{0,..,}e_{i-1},e,e_{i+1},..,e_n]: s|=r$$

Определение 10 (*Однотипные структурные схемы*). Структурная схема S=(T,E,A,p,a,r) называется однотипной, если для любого типа элемента e, все типы элемента из множества $\Delta_E(p(e))$ обладают разными именами.

Определение 11 (*Ограничено-однотипные структурные схемы*). Структурная схема S=(T,E,A,p,a,r) называется ограниченно-однотипной, если для любого типа элемента e, выполняется следующее условие:

$$\forall s_1=(e_0,..,e_n),\ \forall s_2=(e'_0,..,e'_m), \text{ где } s_1|=p(e) \text{ и } s_2|=p(e),\ \text{ и } \forall i:\ \forall j<i\ \ e'_j=e_j \Rightarrow name(e_i)\neq name(e'_i)$$

Следующие два утверждения очевидны и будут приведены без доказательств.

Утверждение 4 (Вложение типов). *Любая локальная структурная схема является однотипной структурной схемой. Любая однотипная структурная схема является ограниченно-однотипной структурной схемой.* □

Утверждение 5 (Достаточное условие однотипности). Пусть структурная схема S=(T,E,A,p,a,r) обладает следующим свойством: $\forall e \in E:\ |\Delta_M(p(e))|<=1$ *(Количество допустимых символов не превышает 1). Тогда S является однотипной структурной схемой.* □

Утверждение 6 (Единственность интерпретации) *Пусть S=(T,E,A,p,a,r) ограниченно-однотипная структурная схема и XML документ D валиден для S. Пусть также любые два домена из множества T не пересекаются и для любого типа элемента e, мультимножество имен типов атрибутов из множества a(e) содержит только уникальные значения. Тогда существует и притом единственная интерпретация документа D в терминах S.*

Для доказательства этого утверждения необходимо воспользоваться свойством *согласования содержания элемента.*

В заключении этого раздела, заметим, что исследования, проведенные в работе [13] показали, что множество структурных схем, соответствующих схемам,

выраженным на языке DTD принадлежит классу локальных структурных схем. Множество структурных схем, соответствующих схемам, выраженным на языке DTD принадлежит классу однотипных структурных схем. И наконец, множество структурных схем, соответствующих схемам, выраженным на языке Relax NG, является полным множеством структурных схем.

4. Преобразования структурных схем

В этом разделе мы обсудим различные методы преобразования структурных схем. Эти преобразования можно разделить на три вида:

- *Эквивалентные* преобразования – приводящие к схеме эквивалентной исходной.
- *Слабо-эквивалентные* – приводящие к схеме, множество валидируемых документов, которой совпадает с множеством валидируемых документов исходной схемы с точностью до перестановки порядка элементов в документе.
- *Упрощающие* – приводящие к схеме не эквивалентной исходной. Однако для любого документа, валидируемого исходной схемой должен существовать документ валидируемый получаемой схемой, отличающийся только порядком следования элементов.

Определение 12 (*Эквивалентность структурных схем*). Схемы D и D' эквивалентны, если множества валидируемых XML документов каждой из этих схем совпадают.

Теорема 3 (Достаточное условие эквивалентности). *Две схемы S=(T,E,A,p,a,r) и S'=(T',E',A',p',a',r') эквивалентны если существует взаимно однозначное отображение M*=(ϕ,φ,σ), где *ϕ:E→ E' ; φ: E→ E'; σ: T→ T' обладающее следующими свойствами (отображение M*$^{-1}$ *обладает аналогичными свойствами):*

1) $\sigma(t) \equiv t$
2) *name(e)= name(ϕ(e))*
3) $\phi(r) \equiv r$
4) *a≡φ (a) (сохраняются все свойства типов атрибута)*
5) $\forall e \ \forall a \in a(e)$: $\varphi(a) \in a'(\phi(e))$ *и* $R(\phi(e)) \subseteq \varphi(a(e))$
6) $\forall s=[s_0,..,s_n]\ s|=p(e) \Rightarrow s'=[M(e_0),..,M(e_n)]|=p'(\phi(e))$

Для доказательства достаточно проверить, что каждый XML документ, удовлетворяющий схеме S должен удовлетворять схеме S' . Проверка того, что каждый XML документ, удовлетворяющий схеме S' удовлетворяет схеме S, производится аналогично. Итак, пусть D|=S , где D – XML документ. В силу определения 6 должна существовать интерпретация I=(ϕ',φ',σ') документа D в терминах S. Рассмотрим отображение I'=M•I =(ϕ•ϕ', φ•φ', σ•σ'). Докажем, что это интерпретация документа D в терминах S'.

1) $\forall e \in E_D$: name(e)= name (ϕ'(e))=name(ϕ•ϕ'(e)) (согласование имени элементов).
2) $\forall a \in A_D$: name(e)= name (φ'(a))=name(φ•φ' (a)) value(a) ∈dom(φ'(a)) = dom(φ•φ'(a)) (*согласование имен и значений атрибутов*).
3) $\forall t \in T_D$: value(t) ∈σ' (t) ≡σ•σ' (t) (*согласование текстовых узлов*)
4) $\forall\ a_e \in A_e$: φ '(a_e) ∈a(ϕ'(e)) ⇒φ•φ' (a_e) ∈ a(ϕ•ϕ' (e)) (*согласование атрибутов с элементами*).
5) $\forall e \in E_D$: $I(e_0),..,\ I(e_n)$ |= p(ϕ'(e)) ⇒ $I(e_0),..,\ I(e_n)$ |= p(ϕ•ϕ' (e)) (*согласование содержания элемента*).

Остальные свойства интерпретации проверяются аналогично. □

В случае, когда E≡E' шестое условие принимает следующий вид p≅p'.

Следствие 1 (Критерий эквивалентности схем, отличающихся только структурами). *Пусть S=(T,E,A,**p**,a,r) и S'=(T,E,A,**p'**,a,r) две структурные схемы, у которых множество валидируемых XML документов непустое, и отличающиеся только регулярными выражениями, задающими структурное вложение. Тогда схемы S и S' эквивалентны тогда и только тогда, когда ∀e∈ E p(e) ≅ p'(e).*

Достаточное условие является следствием теоремы 3, а необходимое условие проверяется на множестве экземпляров XML документов, удовлетворяющих схемам. □

4.1. Эквивалентные преобразования

Утверждение 7 (Эквивалентные регулярные выражения). *Следующие регулярные выражения являются эквивалентными:*

(r) ≅**r** (3.1.1) **r?**≅**r|ε** (3.1.2) **r+**≅**r*,r** (3.1.3) **r****≅**r*** (3.1.4) **r1| r_2** ≅ **r_2|r1** (3.1.5)

(r_1| r_2),r_3≅ **(r_1, r_3)|(r_1, r_2)** (3.1.6) **ε***≅**ε** (3.1.7) **(r_1| r_2)*** ≅ **(r_1*, r_2*)*** (3.1.8)

r*≅ **r+|ε** (3.1.9) **r, ε**≅**r** (3.1.10) **r_1*, r_1***≅ **r_1*** (3.1.11) **r_1*, r_1+**≅ **r_1+** (3.1.12)

r_1*, r_1?≅ **r_1*** (3.1.13) **r?*≅r*** (3.1.14) **r+*≅r*** (3.1.15) **r++≅r+** (3.1.16)

(r_1|r_2)+ ≅ **(r_1+,r_2+)+|(r_2+, r_1+)+| r_1+| r_2+|(r_1+, r_2+)+,r_1+|(r_2+, r_1+),r_2+** (3.1.17)

Доказательство этого утверждения хорошо известно из теории регулярных грамматик. Заметим, что перечисленные пары регулярных выражений далеко не исчерпывают полный список эквивалентных регулярных выражений. □

Утверждение 8 (Эквивалентные преобразования структурных схем). *Пусть S=(T,E,A,p,a,r) – структурная схема. Тогда любая схема, получаемая из*

данной путем замены регулярного выражения на эквивалентное приводит к схеме S'=(T,E,A,p',a,r) эквивалентной исходной схеме. □

4.2. Слабо-эквивалентные преобразования

Определение 13 (*Слабо-эквивалентные регулярные выражения*). Два выражения r_1 и r_2 являются слабо-эквивалентными ($r_1 \approx r_2$) , если для любой последовательности $s=[s_0..s_n]$, такой что $s| = r_1$ существует последовательность $s'=[s_{k(0)}..s_{k(n)}]$, где k есть подстановка на множестве {0,..,n} , такая что $s'| = r_2$ и наоборот, для любой последовательности $s=[s_0..s_n]$, такой что $s| = r_2$ существует последовательность $s'=[s_{k(0)}..s_{k(n)}]$, где k есть подстановка на множестве {0,..,n}, такая что $s'| = r_1$.

Утверждение 9 (Слабо-эквивалентные регулярные выражения). *Следующие регулярные выражения являются слабо-эквивалентными:*

$\mathbf{r_1, r_2 \approx r_2, r_1}$ (3.2.1) $\mathbf{(r_1^*, r_2)^* \approx \varepsilon | r_1^*, r_2^+}$ (3.2.2)

Докажем сначала первое утверждение. Пусть s=[s0..sn] |= r_1, r_2. Тогда из определения 2 следует, что $s \equiv [e_{0,..,}e_k, f_{0,..,}f_n]$, где $[e_{0,..,}e_k] | = r_1$ и $[f_{0,..,}f_n] | = r_2$. Значит $s' \equiv [f_{0,..,}f_n, e_{0,..,}e_k] | = r_2, r_1$.

Докажем второе утверждение. Пусть $s|=(r_1^*, r_2)^*$ Рассмотрим два случая: s-пустая или непустая последовательность. Если s=[], то $s|=\varepsilon$. Пусть s непустая последовательность символов. Тогда s можно представить в следующем виде $[\mathbf{s}^1_0,..,\mathbf{s}^1_{k(1)},\mathbf{s}^1,..,\mathbf{s}^n_0,..,\mathbf{s}^n_{k(n)},\mathbf{s}^n]$, где $\mathbf{s}^i_j| = r_1$, $\mathbf{s}^i| = r$. Тогда последовательность $s'=[\mathbf{s}^1_0,..,\mathbf{s}^1_{k(1)},..,\mathbf{s}^n_0,..,\mathbf{s}^n_{k(n)},\mathbf{s}^1,..,\mathbf{s}^n]| = r_1^*, r_2^+$. Что и требовалось доказать. В обратную сторону утверждение доказывается аналогично. □

Определение 14 (*Ослабленная интерпретация*). Ослабленной интерпретацией I XML документа D в терминах структурной схемы S=(T,E,A,p,a,r) называется набор отображений $I=(\phi,\varphi,\sigma)$, удовлетворяющий всем свойствам интерпретации, кроме *согласования содержания элемента.* Условие согласования содержания элемента заменяется следующим:

- (*согласование содержания элемента*) Пусть $C_e = [e_0,..,e_n]$ – есть упорядоченная последовательность элементов и текстовых узлов, вложенных в e. Тогда $\forall e \in E_D$: $I(e_{k(0)}),.., I(e_{k(n)}) | = p(\phi(e))$, где k(i)- есть подстановка на множестве {0,..,n}.

Определение 15 (*Ослабленная Валидность*). Документ D является *ослабленно-валидным* документом для структурной схемы S (*слабо удовлетворяет схеме S),* если существует ослабленная интерпретация I в терминах S (Обозначается $D|\approx S$).

Определение 16 (*Слабая эквивалентность структурных схем*). Схемы D и D' слабо эквивалентны, если множества слабо валидируемых XML документов каждой из этих схем совпадают.

Следующие утверждения являются очевидными и приводятся без доказательства.

Утверждение 10 (Слабая эквивалентность эквивалентных регулярных выражений). *Если регулярные выражения являются эквивалентными, то они являются слабо-эквивалентными.* □

Следствие 2 (Достаточное условие слабой эквивалентности). *Если схема S и S' являются эквивалентными, то они являются слабо-эквивалентными.* □

Теорема 4 (Критерий слабой эквивалентности схем, отличающихся только структурами). *Пусть S=(T,E,A,**p**,a,r) и S'=(T,E,A,**p'**,a,r) две структурные схемы, у которых множество валидируемых XML документов непустое, и отличающиеся только регулярными выражениями, задающими структурное вложение. Тогда схемы S и S' эквивалентны тогда и только тогда, когда $\forall e \in E$ $p(e) \approx p'(e)$.* □

Как будет показано в дальнейшем, в силу того, что при *слабо-эквивалентных* преобразованиях (замене регулярного выражения на слабо-эквивалентное) теряется только семантика порядка следования элементов, то их удобно использовать для трансляции из XML модели в модели, не использующие порядок в структурном описании.

Также, преобразование 3.2.2 ведет к «выравниванию» схемы (в английской литературе используется термин "flattening"), тем самым, приводя её к более простому виду - без вложенных операторов Клини (*).

Отдельно стоит заметить, что преобразование $\mathbf{(r_1, r_2)^* \rightarrow r_1^*, r_2^*}$, часто встречающееся в алгоритмах трансляции XML модели в реляционную, не является слабо-эквивалентным.

4.3. Упрощающие преобразования

Все преобразования, которые будут представлены ниже, ведут к потере определенной, достаточно большой части информации о структуре документа. Однако они достаточно часто используются на практике. Обусловлено это тем, что упрощение структурных ограничений приводит к существенному уменьшению сложности решения многих задач, встречающихся на практике.

Определение 17 (*Упрощение регулярного выражения*). *Регулярное выражение r_2 над E является упрощением р.в. r_1 над E ($r_1 \langle r_2$) , если множество символов E, формирующих r_2 , совпадает с множеством символов, формирующих r_1, и выполняется следующее условие. Для любой последовательности $s=[s_0..s_n]$ такой, что $s| = r_1$ существует последовательность $s'=[s_{k(0)}..s_{k(n)}]$, где k есть подстановка на множестве [0,n] и $s'| = r$.*

Утверждение 3. *Для регулярных выражений r_1 и r_2*

$$r1 \lhd r2 , r2 \lhd r1 \Leftrightarrow r1 \approx r2,$$

$$r1 \cong r2 \Rightarrow r1 \lhd r2 , r2 \lhd r1$$

Для доказательства первого предложения в прямую сторону ($r_1 \lhd r_2 , r_2 \lhd r_1 \Rightarrow r_1 \approx r_2$) достаточно воспользоваться определением упрощения. Чтобы доказать утверждение в обратную сторону, необходимо воспользоваться критерием допустимости символов из множества E (теорема 2). Вторая часть утверждения вытекает из первой и из утверждения 10. □

Из этого утверждения непосредственно вытекает, что слабо-эквивалентные и эквивалентные преобразования являются упрощающими.

Утверждение 4. (Упрощающие преобразования) *Следующие преобразования регулярных выражений являются упрощающими:*

$(r_1, r_2)^* \lhd r_1^*, r_2^*$	(3.3.1)
$(r_1, r_2)+ \lhd r_1+, r_2+$	(3.3.2)
$(r_1, r_2)? \lhd r_1?, r_2?$	(3.3.3)
$(r_1 \mid r_2) \lhd r_1?, r_2?$	(3.3.4)
$r_1?, r_1? \lhd r_1^*$	(3.3.5)
$r_1+, r_1+ \lhd r_1+$	(3.3.6)

Доказательство этого утверждения напрямую следует из определения порождаемых последовательностей. Заметим, что мы перечислили не все упрощающие преобразования. □

Пример 5.

Дано следующее описание элемента : <!ELEMENT a ((b|c|e|)?,(e?|(f?,(b,b)*))*)>

((b|c|e|)?,(e?|(f?,(b,b)*))*) ⟨ (b|c|e|)?,(e?|(f?,b*,b*))* ⟨ (b|c|e|)?,(e?|(f?,b*))* ⟨
(3.3.1) (3.1.11) (3.3.4)

(b|c|e|)?,(e??,(f?,b*)?)* ⟨ (b|c|e|)?,(e??,f??,b*?)* ⟨ b?,c?,e?,(e??,f??,b*?)* ⟨
(3.3.3) (3.3.4) (3.3.1)

b?,c?,e?,e??*,f??*,b*?* ⟨ b?,c?,e?,e*,f*,b* ⟨ **b*,c?,e*,f***
(3.1.4),(3.1.14) (3.1),(3.1.13)

Как видно из примера исходная схема приобретает весьма простой вид. Следует учесть, что информация об относительном порядке элементов утеряна,

но при этом семантика множественности сохранена (например, элемент с может быть максимум один у элемента a).

Определение 18 (*Упрощение схемы*). Схема D' является упрощением схемы D, если множество валидируемых документов первой схемы принадлежит множеству слабо-валидируемых элементов второй.

Критерий и достаточное условие того, что схема S является упрощением схемы S', формулируются и доказываются таким же образом, как и для слабо-эквивалентных схем.

В следующем разделе, мы определим нормальные формы схем XML документов и докажем теоремы существования нормальных форм произвольных структурных схем.

5. Нормальные формы структурных схем

В этом разделе мы опишем нормальные формы структурных схем: представлений структурных ограничений, записанных определенным образом. Также мы сформулируем и докажем теоремы существования нормальных форм для любой структурной схемы.

5.1. Первая нормальная форма

Определение 19 (*Конъюнктивно-множественные регулярные выражения*). Конъюнктивно-множественные (к.-м.) регулярные выражения над множеством E ($reg_{KM(E)}$)определяются следующим образом:

1) ε - к.-м. регулярное выражение, где ε обозначает «пустой список».
2) ∀e ∈ E: e- к.-м. регулярное выражение.
3) Если r_1 – к.-м. регулярное выражение, то (r_1), r_1*- к.-м. регулярные выражения.
4) Если r_1 и r_2– к.-м. регулярные выражения, то r_1, r_2 - к.-м. регулярные выражения.

Определение 20 (*Первая нормальная форма*). Схема S=(T,E,A,p,a,r) представлена в первой нормальной форме (*эквивалентная форма*), если :

$$\forall e \in E\ p(e)=r_0|..|r_n\text{, где } \forall i\ r_i \in reg_{KM}(\{id(E),T\})$$

Теорема 5 (Существование первой нормальной формы). *Для любой схемы* S=(T,E,A,p,a,r) *существует схема эквивалентная ей, которая представлена в первой нормальной форме*.

Для доказательства этой теоремы следует воспользоваться *следствием 1 из теоремы 3*. Пусть e – некий элемент схемы S=(T,E,A,p,a,r). Соответственно p(e) – регулярное выражение. Используя эквивалентные преобразования регулярных выражений **r?≅r|ε** (3.1.2) и **r+≅r*,r** (3.1.3) мы приходим к регулярному выражению, соответствующему исходному, но не содержащему операций **?** и **+.** После чего следует воспользоваться преобразованиями **$(r_1| r_2),r_3 \cong (r_1, r_3)|(r_1, r_2)$** (3.1.6) и **$(r_1| r_2)^* \cong (r_1^*, r_2^*)^*$** (3.1.8), после которых

операция конкатенации (“|”) «поднимается». Таким образом, для любого типа элемента e, p(e) преобразуется в выражение p’(e) вида $r_0|..|r_n$, где $\forall i\ r_i \in reg_{KM}(\{id(E),T\})$. В силу следствия 1 новая схема S’=(T,E,A,p’,a,r), представленная в первой нормальной форме, эквивалентна схеме S. □

Следует заметить, что для регулярных выражений с использованием операции *позитивного замыкания* (“+”) вместо операции Клини (“*”)теорема о существовании нормальной формы также верна.

Определение 19’ (*Конъюнктивно-множественные регулярные выражения*). Конъюнктивно-множественные (к.-м.) регулярные выражения над множеством E ($reg_{KM}(E)$)определяются следующим образом:

1) ε - к.-м. регулярное выражение, где ε обозначает «пустой список»;
2) $\forall e \in E$: e- к.-м. регулярное выражение;
3) Если r_1 – к.-м. регулярное выражение, то (r_1), r_1+- к.-м. регулярные выражения.

Если r_1 и r_2 – к.-м. регулярные выражения, то r_1, r_2 – к.-м. регулярные выражения.

Принимая альтернативное определение конъюнктивно-множественных регулярных выражений, доказательство теоремы 5 частично меняется. Так, вместо преобразования **r+≅r*,r** используется **r*≅r+|ε** (3.1.9). А вместо преобразования $\mathbf{(r_1|\ r_2)^* \cong (\ r_1^*,\ r_2^*)^*}$ для поднятия конкатенации применяется $\mathbf{(r_1|r_2)+ \cong (\ r_1+,r_2+)+|(\ r_2+,\ r_1+)+|\ r_1+|\ r_2+|(\ r_1+,\ r_2+)+,r_1+|(\ r_2+,\ r_1+),r_2+}$ (3.1.17)

Пример 6.

Приведем к первой форме $r_0|..|r_n$, где $\forall i\ r_i \in reg_{KM}(\{id(E),T\})$ следующее регулярное выражение: ((b|c)?,(f?,b*)*), используя эквивалентные преобразования:

(b|c)?,(f?,b*)*≅(b|c|ε),((f|ε),b*)*≅(b|c|ε),((f,b*)|(b*))*≅(b|c|ε),((f,b*)*,(b**))*≅ (b|c|ε),((f,b*)*,b*)*≅(b|c|ε),((f,b*)*)*≅(b|c|ε),(f,b*)*≅b,(f,b*)*| c,(f,b*)*| (f,b*)*. □

5.2. Вторая нормальная форма

Определение 21 (*Конъюнктивные регулярные выражения*). Конъюнктивные (к.) регулярные выражения над множеством E ($reg_K(E)$)определяются следующим образом:

1) ε - регулярное выражение, где ε обозначает «пустой список»;
2) $\forall e \in E$: e- к. регулярное выражение;
3) Если r_1 – к. регулярное выражение, то (r_1)-к. регулярные выражения;
4) Если r_1 и r_2– к. регулярные выражения, то r_1, r_2 - тоже к. регулярное выражение;
5) Если $r =(e_{0,..,}e_n)$, где $\forall i : e_i \in E$, то r* и r+ - к. регулярные выражения.

Определение 22 (*Вторая нормальная форма*). Схема S=(T,E,A,p,a,r) представлена во второй нормальной форме (*слабо-эквивалентная нормальная форма*), если:

$\forall e \in E\ p(e)=r_0|..|r_n$, где $\forall i\ r_i \in reg_K(E)$

Теорема 6 (Существование второй нормальной формы). *Для любой схемы S=(T,E,A,p,a,r) существует схема слабо-эквивалентная ей, представленная во второй нормальной форме.*

Для доказательства этой теоремы, необходимо воспользоваться результатами *Теоремы* 5. Для исходной схемы S существует эквивалентная схема S’, структурные ограничения которой имеют вид $r_0|..|r_n$, где $\forall i\ r_i \in reg_{KM}(E)$. Далее, для каждого r_i мы воспользуемся преобразованием $\mathbf{(r_1^*,\ r_2)^* \approx \varepsilon|\ r_1^*,\ r_2+}$ (3.2.2) для уменьшения вложенных операторов * и +. После чего, если выражение r_2 содержит операцию *, то воспользуемся преобразованием (3.1.3) для замены оператора r_2+ на r_2*,r Таким образом, используя индукцию по длине регулярного выражения и по глубине «вложенности» операций * и +, приходим к доказательству теоремы. Используя преобразования (3.2.1) и (3.1.11)-(3.1.16) можно добиться существенного упрощения выходной формы. □

Пример 7. Приведем регулярное выражение из предыдущего примера ко второй нормальной форме.

(b|c)?,(f?,b*)*≅b,(f,b*)*| c,(f,b*)*| (f,b*)*≈(b|c|ε)(b*f+|ε)≈b+f+|b|cb*f+|c|b*f+|ε≈ b|cb*f+|c|b*f+|ε (в последнем переходе использовалось эквивалентное преобразование r*|r+≅r*). □

5.3. Третья нормальная форма

Определение 23 (*Простые регулярные выражения*). Простые (п.) регулярные выражения над множеством E ($reg_S(E)$)определяются следующим образом*:*

1) ε -п. регулярное выражение, где ε обозначает «пустой список»;
2) $\forall e \in E$: e- п. регулярное выражение;
3) Если r_1 и r_2– п. регулярные выражения, то r_1, r_2 - тоже п. регулярное выражение;
4) Если r =e , где e∈ E, то r* ,r? r+ - п. регулярные выражения.

Определение 24 (*Третья нормальная форма*). Схема S=(T,E,A,p,a,r) представлена во третьей нормальной форме (*простая нормальная форма*), если:

$\forall e \in E\ p(e)=r$, где $r \in reg_S(E)$

Теорема 7 (Существование третьей нормальной формы). *Для любой схемы S=(T,E,A,p,a,r) существует схема, являющаяся ее упрощением, и представленная в третьей нормальной форме*.

Для доказательства этой теоремы следует воспользоваться упрощающими преобразованиями для построения новой структурной схемы, являющейся упрощением исходной схемы и представленной в третьей нормальной форме. Для доказательства того, что такая схема существует необходимо воспользоваться индукцией по длине регулярного выражения. □

Пример 8. Рассмотрим регулярное выражение из примера 7.

(b|c)?,(f?,b*)*→(b?,c?)?,f?*,b**→b??,c??,f?*,b**→b?,c?,f*,b*→b*,c?,f*

В отличие от первой и второй нормальных форм, для третьей нормальной формы можно сформулировать и доказать теорему единственности. Пусть на множестве E введено отношение порядка. Тогда, определим простые упорядоченные регулярные выражения следующим образом.

Определение 25 (*Простые упорядоченные регулярные выражения*). Простые упорядоченные (п. у.) регулярные выражения над множеством E ($reg_{SO(E)}$)определяются следующим образом:

1) ε -п. у. регулярное выражение, где ε обозначает «пустой список»;
2) ∀e ∈ E: e- п. у. регулярное выражение;
3) Если r =e , где e∈ E, то r* ,r? r+ - п. у. регулярные выражения;
4) Если r_1 и r_2– п. регулярные выражения, и ∀a_1,a_2 ∈E, таких, что ∃s_1= [e_0,..,e_{i-1},a_1,e_{i+1},..,e_n], s_2= [e'_0,..,e'_{i-1},a_2,e'_{i+1},..,e'_n] $s_1|=r_1$, и $s_2|=r_2$ верно, что $e_1<e_2$, то r_1, r_2 - тоже п. у. регулярное выражение.

Определение 24' (*Третья нормальная форма*). Схема S=(T,E,A,p,a,r) с заданным отношением порядка на множестве E представлена в третьей нормальной форме (*простая нормальная форма*), если:

∀e ∈ E p(e)=r , где r ∈ regSO(E)

Если не существует двух типов элементов с одинаковым именем, то отношение порядка на множестве E может соответствовать лексикографическому порядку на множестве имен элементов.

Теорема 8 (Существование и единственность третьей нормальной формы). *Для любой схемы S=(T,E,A,p,a,r), такой, что на множестве E задано отношение порядка, существует и единственная схема S'=(T,E,A,p',a,r) , представленная в третьей нормальной форме, являющаяся ее упрощением*. □

6. Применение Нормальных форм

В этом разделе мы приводим краткий обзор способов применения нормальных форм при решении типичных задач управления данными.

Валидация XML документов. Валидация XML документов является одним из наиболее распространенных средств управления XML документами. Валидация документов используется при создании XML-СУБД, обмене сообщениями, трансформации XML документов. Одна из основных проблем при валидации документа заключается в том, что до сих пор не существует единого стандарта для XML схем: схемы могут быть выражены на языках DTD, XML Schema, Relax NG, и т.д. Однако на сегодняшний день существует ряд исследований, ориентированных на создание универсального валидатора. В этих работах схемы представляются в виде регулярных грамматик деревьев, аналогичных структурным схемам. Основная часть алгоритмов валидации – разбор списка потомков – заключается в следующем: определить

удовлетворяет ли упорядоченный список потомков данного элемента его модели содержания (структурному ограничению).

Преобразование схем к первой нормальной форме, как впрочем, и применение эквивалентных преобразований имеют следующее значение для алгоритмов валидации:

1) После преобразования к 1НФ операции ? и + устраняются, что позволяет воспользоваться алгоритмами валидации «классических» регулярных выражений, определяемых как замыкание операций "*", "|", "," над базовым алфавитом.
2) Все операции конкатенации ("|") становятся внешними, что сильно структурирует модель содержания. Это приводит к упрощению алгоритма разбора списка потомков.

Сопоставление схем. Задача сопоставления схем заключается в поиске эквивалентных частей в разных схемах. Решение этой проблемы применяется для интеграции данных. В текущее время ведутся исследования по автоматическому поиску зависимостей [16]. Методы автоматического сопоставления схем можно классифицировать следующим образом [16]:

- *Поиск в схемах/поиск в данных*. Алгоритмы сопоставления схем могут исследовать зависимости только в схемах или зависимости по удовлетворяющим XML-документам.
- *Элементный/структурный поиск*. Поиск зависимостей может применяться либо к отдельным элементам, либо к структурам (фактически, учитываются или нет структурные ограничения).
- *Лингвистический/логический поиск*. Поиск зависимостей осуществляется по лингвистическому принципу (например, по именам элементов) или по семантическим ограничениям (например, по типам данным).

В первую очередь, приведение схем к нормальным формам оказывает влияние на элементный лингвистический поиск. Приведение схем к 3НФ с заданным отношением порядка существенно упрощает поиск зависимостей и фактически сводит задачу к поиску изоморфных поддеревьев в деревьях с именованными узлами и с ребрами, размеченными «*», «+» и «?». Подробнее способы сопоставления схем будут описаны в следующей главе.

Трансляция моделей. Одна из основных задач, встречающихся при создании систем хранения, управления и интеграции данных, заключается в трансляции моделей, в терминах которых экземпляры данных предоставляются, в термины «единой» модели данных. Так, несмотря на то, что XML и языки запросов к XML-данным завоёвывают в последнее время всё большую популярность, потребность хранения XML данных в «традиционных» СУБД и, соответственно, необходимость трансляции до сих пор остаётся. Обосновано это тем, что многолетний опыт, накопленный при изучении и реализациях реляционных и объектно-ориентированных СУБД невозможно игнорировать.

Рассмотрим основные виды трансляции данных, определенных в терминах модели XML.

6.1. XML->Relational

Одним из основных направлений исследования методов трансляции моделей является трансляция XML-модели данных в реляционную [9,10]. Отличительными особенностями реляционной модели данных являются:

- отсутствие упорядоченности кортежей и атрибутов,
- трехуровневая модель (отношение-кортеж- атрибут), в отличие от произвольной глубины XML схем,
- отсутствие атрибутов, имеющих своим значением множество,
- отсутствие рекурсии.

Существует различные методы автоматического представления XML документов в реляционных СУБД:

- *Хранение XML- данных в BLOB*.
- *Модельно-ориентированная трансляция*. Данный вид трансляции не зависит от структурных ограничений, определенных в схеме схемы, а целиком опирается на свойства модели данных XML [20].
- *Трансляция, ориентированная на данные*. Данные алгоритмы трансляции оперируют с XML-данными, не представленными никакой схемой [19]. На начальном этапе трансляции производится вывод схемы, представляющей XML-данные.
- *Трансляция с оценкой эффективности.* Алгоритм трансляции, представленный в работе [18], анализирует способ трансляции XML-данных таким образом, чтобы запросы, предопределенные приложением, выполнялись наиболее эффективно.
- *Структурно-ориентированная трансляция*. Эта трансляция опирается на информацию, полученную из XML-схемы. В алгоритмах этого типа [9,10] используются структурные ограничения, явно присутствующие в схеме, а также выводятся неявные ограничения, исследуемые в процессе анализа схемы.

Проведенные исследования [9,10], касающиеся проблемы структурно-ориентированной трансляции из XML в реляционную модель выявили ряд возникающих трудностей и показали методы решения:

1) Наличие оператора конкатенации в модели содержания. Например, пусть у нас есть определение модели содержания элемента <!ELEMENT r (a|b)>. Переводя в реляционную модель, наиболее близким отображением будет таблица r с двумя полями: a и b. Однако эта схема не будет отражать тот факт, что в элементе r может встретиться либо a либо b. Поэтому необходимо добавить семантическое ограничение: «если значение в поле a непустое, то значение в поле b должно быть пустым и наоборот». Если есть вложенные конкатенации (то есть конкатенации не на самом верхнем

уровне регулярного выражения), мы должны использовать 1НФ для вынесения конкатенаций на самый верхний уровень.

2) Наличие оператора Клини (*). Если в модели содержания какого-то узла встречается другой элемент с оператором * (<!ELEMENT r a*>), то в этом случае для элемента придется создавать отдельное отношение. Если же структура содержит вложенные операторы * (<!ELEMENT r (a*,b)*), то количество отношений существенно увеличится. Обычно для решения этой проблемы используются преобразования схем аналогичных 3НФ (упрощение схемы).

Однако, на наш взгляд, правильнее использовать 2 НФ для представления схемы XML:

- Во первых, 2НФ оставляет практически всю семантику схемы.
- Во вторых, семантика порядка, которая теряется во 2НФ, не оказывает никакого влияния на трансляцию в реляционную модель, вследствие отсутствия семантики порядка в реляционной модели.

6.2. XML->Semistructured

Другим направлением трансляции XML является трансляция в полуструктурированные и объектно-ориентированные модели. Существенное отличие этих моделей от реляционной заключается в том что, данные представляются в виде ориентированного графа с именованными узлами. Последнее свойство, как правило, снимает необходимость использования 2НФ и 3НФ. Поэтому, в общем случае транслируется схема, приведенная к 1НФ или непосредственно исходная схема. Например, для полуструктурированной модели данных YAT [14] мы использовали представление структурных схем в 1НФ, в силу особенностей данной модели (отсутствие операторов + и ? и наличие оператора | с ограниченными свойствами)[15].

6.3. *Relational -> XML*

В работе [21] перечислены основные методы автоматизации представления реляционных данных в терминах модели данных XML:

- *Плоская трансляция*. Данный подход является наиболее тривиальным способом отображения схемы реляционной базы данных в XML-схему. Трансляция задается следующим образом:
    1. Имя отношения переходит в элемент с таким же именем. Содержимое корневого элемента состоит из произвольного набора элементов, имена которых соответствуют именам отношения в базе данных.
    2. Каждому отношению базы данных ставится в соответствие тип элемента, имя которого совпадает с именем отношения. Множество типов атрибутов, относящихся к данному типу элемента, соответствуют

паре {тип домена, имя атрибута} из заголовка отношения.

- *Вложенная трансляция (Nesting-Based Translation)*. Основной недостаток плоской трансляции заключается в том, что при создании XML-схемы не используются такая структурная возможность для моделирования XML, как наличие повторяющихся подэлементов. *Вложенная трансляция* устраняет этот недостаток. В работе [22] показано, каким образом достигается вложенная трансляция для отношений, представленных в 3НФ.
- *Трансляция с использованием «зависимостей по включению»*. Термин «зависимость по включению» используется в теории баз данных [21] как обобщение внешних ключей. Использование трансляции такого типа [21] позволяет вкладывать элементы, построенные из разных отношений друг в друга, исходя из информации о внешних ключах отношений и прочих зависимостей по включению. Если у отношения существует внешний ключ «на себя», получаемая схема XML будет рекурсивной.

Дополнительную информацию об этих и прочих видах трансляции из модели XML в другие модели и наоборот, можно получить в работе [21].

7. Ограничения целостности XML

В последнем разделе главы мы опишем различные виды ограничений целостности для XML-данных. Эти исследования, направленные на систематизацию логических способов задания ограничений целостности, были проведены в работах [8,12].

Итак, пусть S=(T,E,A,p,a,r) – Структурная схема, D- произвольный XML-документ, валидируемый S. Рассмотрим следующие логические выражения, являющиеся определениями *ограничений целостности над схемой S.*

Определение 26 (*Ограничение ключа*). Ограничением ключа λ над схемой S=(T,E,A,p,a,r) называется логическое выражение вида $K(e)\rightarrow e$, где $e\in E$, $K(e)\subseteq a(e)$.

Определение 27 (*Документ удовлетворяет ограничению ключа*). XML-документ D, валидируемый схемой S=(T,E,A,p,a,r) удовлетворяет ограничению ключа $\lambda=K(e)\rightarrow e$, если для любой интерпретации $I=(\phi,\varphi,\sigma)$ выполняется следующее условие:

$$\forall x,y\in\phi^{-1}(e): \bigcap_{l\in K(e)}(x.l = y.l) \rightarrow x\equiv y$$

где x.l – это прообраз типа атрибута l, являющийся атрибутом элемента x. Под равенством атрибутов в документе, здесь и далее, подразумевается равенство значений.

Пример 9.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <a>
    <b c="1" d="1" />
    <b c="1" d="2" />
    <b c="2" d="3" />
  </a>
```

Рис. 3. *Пример документа XML.*

На Рис. 3 представлена документ XML, удовлетворяющий следующей структурной схеме:

T≅{a}

E≅{{a, A}, {b, B}} A≅{(C,c, CDATA, Required), (D,d, CDATA, Required)}

p:p(a)= (b*) p(b)= ε a:a(b)={C,D} a(a)={}

r=a.

Из утверждения 3 следует, что существует единственная интерпретация документа в терминах этой схемы.

Рассмотрим следующее ограничение ключа $\lambda= \{C\}\rightarrow B$. Существует два элемента с именем b, у которых значения атрибута с совпадают. Следовательно, документ не удовлетворяет данному ограничению ключа. Однако тот же самый документ удовлетворяет другому ограничению ключа: $\{C,D\}\rightarrow B$. □

Определение 28 (*Ограничение включения*). Ограничением включения λ над схемой S=(T,E,A,p,a,r) называется логическое выражение вида $L_1(e_1)\rightarrow L_2(e_2)$, где $e_1,e_2\in E$, L_1и L_2 упорядоченные множества, такие что $L_1\subseteq a(e_1)$, $L_2\subseteq a(e_2)$, $|L_1|=|L_2|$.

Определение 29 (*документ удовлетворяет ограничению включения*). XML-документ D, валидируемый схемой S=(T,E,A,p,a,r) удовлетворяет ограничению включения $\lambda= L_1(e_1)\rightarrow L_2(e_2)$, если для любой интерпретации $I=(\phi,\varphi,\sigma)$ выполняется следующее условие:

$$\forall x\in\phi^{-1}(e_1)\ \exists\ y\in\phi^{-1}(e_2): \bigcap_{i=[1,|L_1|]}(x.i = y.i)$$

где x.i и y.i – это прообразы i-х по порядку типов атрибута из упорядоченных множеств $L_1(e_1)$ и $L_2(e_2)$, являющихся атрибутами элементов x и y соответственно.

Пример 10. Рассмотрим структурную схему и XML-документ из предыдущего примера. Приведенный XML-документ удовлетворяет следующему ограничению включения: $\lambda=\{C\}_B\rightarrow\{D\}_B$. □

Определение 30 (*Ограничение внешнего ключа*). Ограничением включения λ над схемой S=(T,E,A,p,a,r) называется комбинация ограничения включения $L_1(e_1) \rightarrow L_2(e_2)$ и ограничения ключа $L_2(e_2) \rightarrow e$.

Определение 31 (*Документ удовлетворяет ограничению внешнего ключа*). XML-документ D, валидируемый схемой S=(T,E,A,p,a,r) удовлетворяет ограничению внешнего ключа $\lambda = L_1(e_1) \rightarrow L_2(e_2)$; $L_2(e_2) \rightarrow e_2$, если он удовлетворяет обоим ограничениям целостности, составляющим ограничение внешнего ключа для любой интерпретации.

После того, как мы определили логические выражения, предназначенные для формулирования ограничений целостности, мы можем сформулировать определения, соответствующие формальным определениям из раздела 1.6 (определение 1.4-1.7).

Определение 32 (*Схема данных XML*). Схема данных *XML* – это пара (S,E), где S – это структурная схема, а E- множество ограничений целостности над S, сформулированных в виде логических выражений следующего вида – ограничение ключа, ограничение включения, ограничение внешнего ключа.

Определение 33 (*Валидируемость документа XML*). XML-документ D валидируется схемой (S,E), если D|=S и D удовлетворяет всем ограничениям целостности из E.

8. Заключение

Основной целью данной работы является изучение свойств схем данных XML-документов. В работе представлено формальное определение структурных схем и изучены методы преобразования схем-экземпляров, обладающих свойством сохранения той или иной семантики. Также в работе представлены нормальные формы структурных ограничений и доказаны теоремы существования нормальных форм для произвольной структурной схемы. Как уже было показано выше, в большинстве исследований, касающихся проблем управления данными и моделями XML тем или иным образом можно установить класс семантических правил, которыми можно пренебречь для эффективности реализации. Нормальные формы схем как раз и являются способом приведения схемы к более простому виду с потерей части семантики. В заключительной части работы мы описываем способы поддержки ограничений целостности.

В данный момент нами разработаны алгоритмы преобразования схем DTD и Relax NG в структурные схемы с последующим приведением к нормальным формам. На следующем этапе мы планируем расширить список поддерживаемых языков спецификаций схем (XDR, XML Schema). Затем мы планируем перейти к более детальному изучению способов применения нормальных форм схем на практике.

Литература

1. *Extensible Markup Language (XML) 0 (Second Edition) W3C Recommendation 6 October 2000.* http://www.w3.org/TR/2000/REC-xml-20001006
2. Бумфрей Ф. , Диренцо О. И др. *XML:Новые перспективы WWW*. М.:ДМК 2000
3. Bourret R. XML and Databases, http://www.rpbourret.com/xml/XMLAndDatabases.htm
4. Suciu, D. Semistructured Data and XML. In Proc. of the Int. Conf. on Foundations of Data Organization. 1998.
5. H. S. Thompson, D. Beech, M. Maloney, and N.Mendelsohn (Eds). XML Schema Part 1: Structures, W3C Recommendation, May 200http://www.w3.org/TR/xmlschema-1/.
6. J. Clark and M. Murata (Eds). “RELAX NG Tutorial”.OASIS Working Draft, Jun. 200 http://www.oasisopen.org/committees/relax-ng/tutorial.html.
7. Microsoft. XML Schema Developer’s guide Internet document, May 2000 http://msdn.microsoft.com/xml/XMLGuide/schema-overwiew.asp
8. W. Fan and L. Libkin. On XML integrity constraints in the presence of dtds. In Proc. ACM PODS, 200
9. J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton. “Relational Databases for Querying XML Documents: Limitations and Opportunities”. In VLDB, Edinburgh, Scotland, Sep. 1999.
10. Murali Mani, Dongwon Lee, *XML to Relational Conversion using Theory of Regular Tree Grammars* , Proceedings of the 28th VLDB Conference,Hong Kong, China, 2002
11. D.Florescu, D. Kossmann, *Storing and Querying XML Data Using RDBMS* , IEEE Data Eng. Bulletin, 22(3):27-34, Sep 1999
12. W. Fan and J. Simeon. "Integrity Constraints for XML". In ACM PODS, Dallas, TX, May 2000.
13. M. Murata, D. Lee, and M. Mani. Taxonomy of XML Schema Languages using Formal Language Theory. In Extreme Markup Languages, Montreal, Canada, 2001
14. S. Cluet et al. "Your mediator needs data conversion!." In Proc. of the ACM SIGMOD Conf. on Management of Data, Washington, USA, pp. 177--188, 1997.
15. L. Novak Mediation system implementation based on specification of XML schema integration: generic approach. To be appeared
16. E. Rahm, and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4):334-350. Dec. 200
17. Dongwon Lee, Wesley W. Chu.Comparative Analysisof Six XML Schema Languages. SIGMOD Record 29(3): 76-87 (2000).
18. P. Bohannon, J. Freire, P. Roy, and J. Simeon. “From XML Schema to Relations: A Cost-Based Approach to XML Storage”. In IEEE ICDE, San Jose, CA, Feb. 200
19. A. Deutsch, M. F. Fernandez, and D. Suciu. “Storing Semistructured Data with STORED”. In ACM SIGMOD, Philadephia, PA, Jun.1998.
20. T. Shimura, M. Yoshikawa, and S. Uemura. “Storage and Retrieval of XML Documents using Object-Relational Databases”. In Int’l Conf. on Database and Expert Systems Applications (DEXA), pp. 206–217,Florence, Italy, Aug. 1999.
21. Dongwon Lee, Murali Mani, Frank Chiu, Wesley W. Chu, “*NeT and CoT: Translating Relational Schemas to XML Schemas using Semantic Constraints* “,CIKM, 2002
22. D. Lee, M. Mani, F. Chiu, and W. W. Chu., “Nesting-based Relational-to-XML Schema Translation”. In Int’l Workshop on the Web and Databases (WebDB), Santa Barbara, CA, May 2001.