УДК 519.7

С.П. Тимофеев

ПРОБЛЕМА ЗАЦИКЛИВАНИЙ В СХЕМАХ XML W3C

1. Введение

Схемы XML, определенные консорциумом W3C в спецификации [1], являются важным компонентом новейшей информационной технологии XML, которая находится в стадии своего становления и поэтому не все ее аспекты детально проработаны. Нами обнаружено, что спецификация схем XML W3C (далее – схем) позволяет задавать схемы, *некорректные* в том смысле, что им не соответствует ни один документ или элемент XML. Могут быть также заданы *частично корректные* схемы, которые хотя и определяют некоторое непустое множество документов (или элементов) XML, но содержат такие определения, которым не соответствует ни один объект. Этими определениями могут быть декларации элементов, дефиниции типов и дефиниции групп. Причина состоит в *зацикленности определений*. Подобное явление в традиционной логике называется *порочным кругом*.

Данная проблема не затронута в [1] и не отображена в доступной нам литературе. Отметим, что самый мощный инструмент технологии XML, XMLSpy [2] обнаруживает лишь простейшие зацикливания и даже, в некоторых случаях, генерирует несостоятельные документы по частично корректным схемам.

Целью работы является частичное решение проблемы зацикливаний в схемах. Для достижения указанной цели решены следующие *задачи*: формально определен некоторый простой класс S_1 схем и доказана теорема о необходимом и достаточном условии их корректности, а также корректности отдельных, входящих в них деклараций элементов XML; разработан абстрактный алгоритм анализа корректности схем указанного класса, который запрограммирован в Delphi.

Алгоритм дополнительно вычисляет минимальную высоту, допустимую для дерева состоятельного элемента XML для каждой корректной декларации в схеме, что может служить средством ее семантического анализа. Алгоритм запрограммирован в Delphi.

Как известно, схемы XML являются представлениями грамматик деревьев [3, 4, 5]. Несложный анализ показал, что зацикленным описаниям в схеме соответствуют *непроизводящие символы* в грамматике. В [4] приведен алгоритм выявления непроизводящих символов для грамматик с простыми правилами. Предложенный нами алгоритм является, по сути, усложненным представлением этого алгоритма в терминах структуры схем XML. Усложнение связано с тем, что абстракцией схемы XML является грамматика деревьев с правилами общего вида, а также с тем, что наш алгоритм дополнительно вычисляет минимальную высоту дерева состоятельного элемента XML (далее – элемента) для каждой корректной декларации, что может служить средством семантического анализа схемы.

Полезно сформулировать общую идею обоих алгоритмов: постепенное расширение множества корректных определений такими, которые имеют хотя бы один вариант своего задания в терминах других определений, состоящий только из корректных определений. В начале алгоритма это множество состоит из всех терминальных определений, присутствующих в схеме. На абстрактном уровне указанная идея сформулирована в [6], где приведена и исследована математическая модель системы взаимосвязанных понятий весьма общего вида. Именно эта работа навела нас на мысль исследовать схемы XML на предмет зацикливаний. Заметим в заключение, что структура определения одного объекта в терминах других объектов в схеме XML называется *моделью содержания* определяемого объекта.

Для понимания материала, изложенного в данной статье, необходимо знакомство со спецификацией [1].

2. Определение класса схем S_1 и демонстрационный пример

Класс S_1 задан таким образом, чтобы максимально облегчить исследование корректности внутри множества деклараций элементов XML (далее – деклараций). Описание класса S_1 в терминах абстрактной структуры, как это делается в спецификации [1], оказывается излишне громоздким (особенно это касается компонента *Particle*), поэтому мы предпочли описание в терминах XML-представления схем. На рис. 1 описано это представление в виде редуцированного метаопределения XML-представления схемы из [1].

Схема из S_1 может содержать только 4 типа компонентов, представляемых элементами XML (так как схема сама является документом XML):

– декларация схемы в целом – элемент *Schema*;

– декларация элемента – элемент *Element*;

– дефиниция сложного типа – элемент *ComplexType*;

– модельная группа – элемент *Sequence* или *Choice* (не допускается *All*).

```
<schema
 attributeFormDefault =
  (qualified | unqualified) : unqualified
 elementFormDefault =
  (qualified | unqualified) : unqualified
 targetNamespace = anyURI
 {any attributes with non-schema namespace}>
 Content: element*
</schema>

<element
 maxOccurs =
  (positiveInteger | unbounded)  : 1
 minOccurs = nonNegativeInteger : 1
 name = NCName
 ref = QName
 type = QName
 {any attributes with non-schema namespace}>
 Content: complexType?
</element>

<complexType
 Content: (choice | sequence)
</complexType>

<choice
 Content: element+
</choice>

<sequence
 Content: element+
</sequence>
```

Рис. 1. XML-представление класса схем S_1

Тип дерева документа схемы из S_1 наглядно (и вполне формально) описывается диаграммой классов UML (рис. 2). Корневой узел *Schema* содержит непустую последовательность дочерних узлов *глобальных* деклараций. Декларация может быть *терминальной* или *нетерминальной;* в последнем случае она имеет единственный дочерний узел *ComplexType.* Последний, в свою очередь, имеет ровно один дочерний узел модельной группы. Наконец, этот узел имеет непустую последовательность дочерних узлов *локальных* деклараций.

Атрибуты в схеме из S_1 могут быть только у декларации схемы в целом (на рисунке не показаны) и у деклараций элементов. Допустимые типы атрибутов в декларациях элементов:

– *name*;
– *ref*;
– *type* (только с именем встроенного типа);
– *minOccurs*;
– *maxOccurs* (не может быть 0);
– произвольные атрибуты любого пользовательского пространства имен.

Таким образом, схемы S_1 описывают документы, состоящие только из элементов.

Применительно к S_1 удобно использовать следующие термины. Будем говорить, что нетерминальная декларация *владеет группой Sequence* или *Choice* (или имеет *модель содержания Sequence* или *Choice*, причем под группой будем понимать последовательность деклараций элементов, являющихся ближайшими потомками декларации *элемента-*

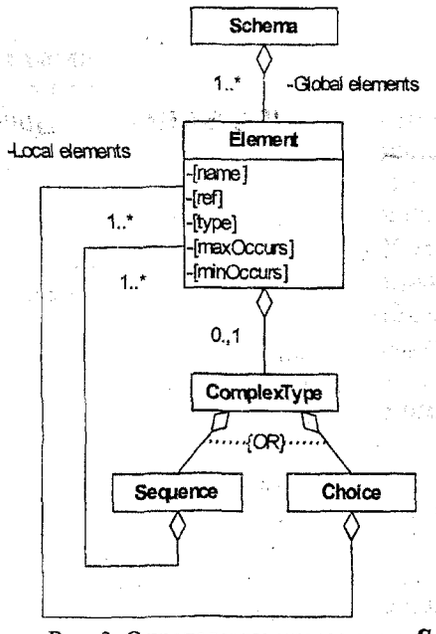

Рис. 2. Определение класса схем S_1

владельца. Будем также говорить в контексте группы о *родительской* и *дочерних* декларациях элементов; для обоснования этих терминов следует рассмотреть *редукт* дерева документа схемы, в котором оставлены только декларации элементов.

Обсуждение.

Приведенное определение S_1 является определением множества выделением:

$$S_1 = \{S \in \boldsymbol{S} | Schema_1(S)\}, \qquad (1)$$

где $Schema_1(S)$ – ограничительный предикат S_1 в универсуме схем $\boldsymbol{S}$, представленный диаграммой UML, которую можно считать заданием этого предиката на высокоуровневом формальном графовом языке UML. Представление на традиционном логико-математическом языке не приводится, поскольку требует введения дополнительных обозначений и непосредственно нам не требуется.

На рис. 3 приведен текст демонстрационной схемы SchemaWithLoops, принадлежащей S_1, а на рис. 6 ее диаграмма, сформированная XMLSpy. Диаграмма любой $S \in S_1$ является двудольным графом, хотя в общем случае это неверно. В блоки, изображающие декларации, мы вставили две характеристики. Первая равна минимальной высоте дерева состоятельного элемента, если декларация корректна, или символу ∞, если она некорректна. Вторая характеристика равна условному номеру варианта модели содержания декларации, что объяснено ниже в комментариях к формуле (14).

XMLSpy версии 2004 г. по этой схеме генерирует несостоятельный документ, приведенный на рис. 4 (несостоятельны элементы B и Er2). XMLSpy версии 2005/2 генерирует также несостоятельный документ длиной в 6429 строки. Пример состоятельного документа минимально возможной высоты 4 приведен на рис. 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:CHK="http://SergTim/CheckSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:annotation>
  <xs:documentation>Примеры для анализа
зацикливания</xs:documentation>
 </xs:annotation>
 <xs:element name="A">
  <xs:complexType>
   <xs:choice>
    <xs:element ref="B"/>
    <xs:element name="Er1">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Data" type="xs:string"/>
       <xs:element ref="Loop"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="Loc1">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="data"/>
       <xs:element name="Loc2">
        <xs:complexType>
         <xs:choice>
          <xs:element ref="Loop"/>
          <xs:element ref="A"/>
          <xs:element name="Any"/>
         </xs:choice>
        </xs:complexType>
       </xs:element>
       <xs:element name="MayEmp1">
        <xs:complexType>
         <xs:sequence>
          <xs:element ref="Loop" minOccurs="0"/>
           <xs:element name="Loc3"
                       minOccurs="0">
            <xs:complexType>
             <xs:sequence>
              <xs:element ref="A"/>
             </xs:sequence>
            </xs:complexType>
           </xs:element>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="B">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="data"/>
   <xs:element name="MayEmp2" minOccurs="0">
    <xs:complexType>
     <xs:choice>
      <xs:element ref="B"/>
      <xs:element ref="Loop" minOccurs="0"/>
     </xs:choice>
    </xs:complexType>
   </xs:element>
   <xs:element name="Er2">
    <xs:complexType>
     <xs:choice>
      <xs:element ref="B"/>
      <xs:element ref="Loop"/>
     </xs:choice>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="Loop">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Loop"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="data" type="xs:string"/>
</xs:schema>
```

Рис. 3. Текст демонстрационной схемы

```
<?xml version="1.0" encoding="ISO-8859-5"?>
<!--Sample XML file generated by XMLSPY v2004
rel. 3 U (http://www.xmlspy.com)-->
<A
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\Users\Serg
tim\
XML\SchCHK\SchemaWithLoops.xsd">
  <B>
    <data>String</data>
    <MayEmp2/>
    <Er2/>
  </B>
</A>
```

Рис. 4. Несостоятельный документ XML, сгенерированный XMLSpy по демонстрационной схеме

```
<?xml version="1.0" encoding="ISO-8859-5"?>
<A
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:noNamespaceSchemaLocation="D:\Users\Ser
gtim\
XML\SchCHK\SchemaWithLoops.xsd">
  <Loc1>
    <data xsi:type="xs:string">String</data>
    <Loc2>
      <Any >Text</Any>
    </Loc2>
    <MayEmp1/>
  </Loc1>
</A>
```

Рис. 5. Пример состоятельного документа для демонстрационной схемы

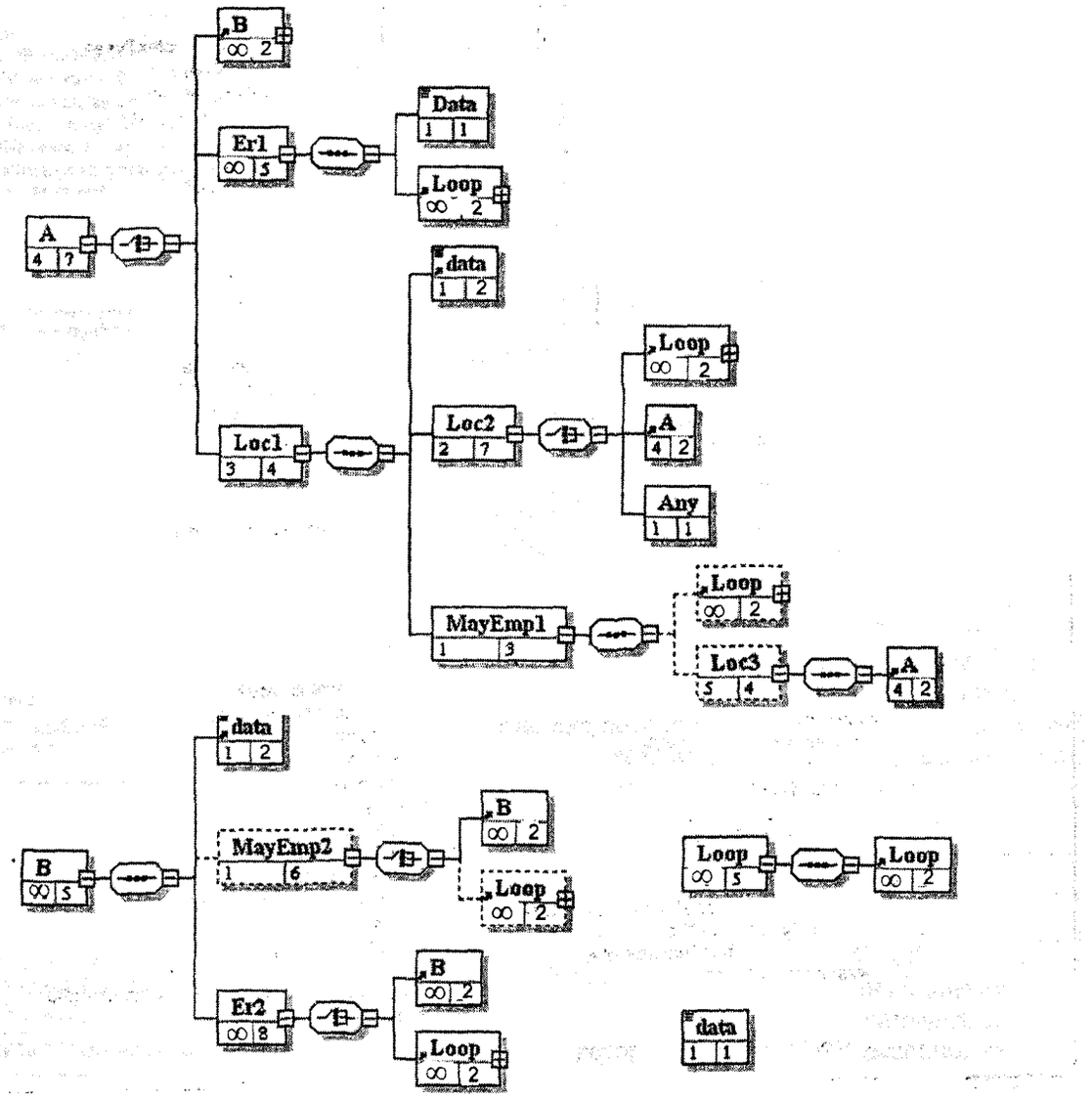

Рис. 6. Демонстрационная схема XML с номерами уровней деклараций элементов и номерами вариантов модели содержания

Опишем кратко демонстрационную схему (рис. 6). Она содержит четыре глобальные декларации: A (частично корректна, поэтому вся схема частично корректна); Loop (некорректна, поскольку зациклена сама на себя); B (некорректна: она имеет модель содержания sequence с некорректным членом); data (корректна: она терминальная). Локальные декларации: Any (корректна: она терминальная); Er1 (некорректна: она имеет модель содержания sequence с некорректным членом); Loc2 (корректна: она имеет модель содержания choice с корректным членом), Loc1 и Loc3 (корректны: они имеют модель содержания sequence, все члены которой корректны); MayEmp1, MayEmp2 корректны, так как имеют пустой вариант модели содержания.

3. Понятия и обозначения, используемые при анализе схем класса S_1

Общие обозначения: $A = \Gamma(B,C,\ldots,D)$ – разбиение множества A; $\mathfrak{P}(A)$ – булеан A.

Для сокращения выкладок, введем умалчиваемую область пробегания переменной $d^{\oplus}$: $d^{\oplus} \in M^{\oplus}(d)$; кроме того, введем следующие сокращенные обозначения (A – произвольное выражение):

$$\begin{aligned} \forall d^{\oplus}\left(d^{\oplus} \in A\right) &\rightleftharpoons \forall d^{\oplus} \in A; \\ \exists d^{\oplus}\left(d^{\oplus} \in A\right) &\rightleftharpoons \exists d^{\oplus} \in A. \end{aligned} \qquad (2)$$

Специальные обозначения.

Поскольку наш анализ касается одной (произвольной) схемы S, мы не указываем в явном виде зависимость от нее вводимых ниже понятий. На рис. 7 наглядно представлена совокупность этих понятий и их взаимосвязь.

D – множество деклараций элементов в S (далее – просто «деклараций»);

T – множество терминальных деклараций;

N – множество нетерминальных деклараций; $D = \Gamma(T,N)$;

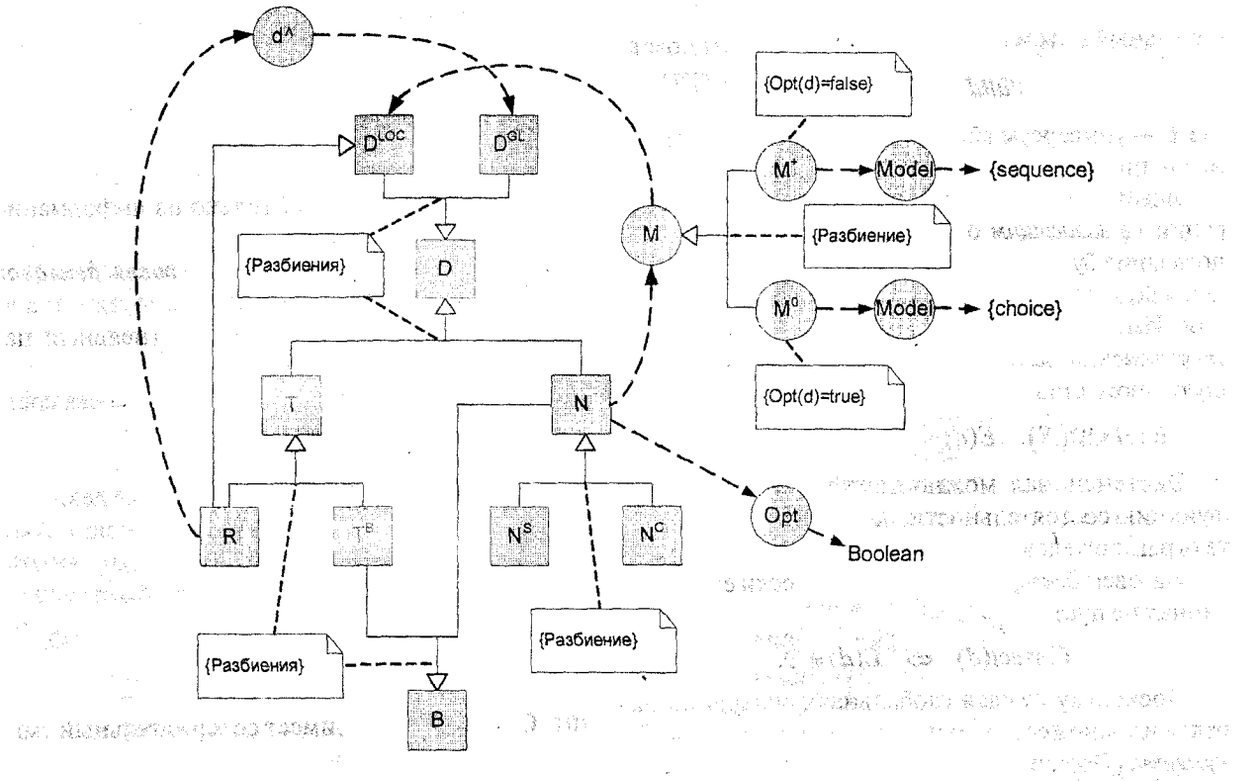

Рис. 7. Система понятий, связанных с множеством деклараций

N^S – множество нетерминальных деклараций, владеющих группой *sequence*;

N^C – множество нетерминальных деклараций, владеющих группой *choice*; $N = \Gamma\left(N^S, N^C\right)$.

Model(d) – функция модели содержания нетерминальной декларации: *sequence* или *choice*;

T^B – множество базовых терминальных деклараций (не имеющих атрибута *ref*);

R – множество ссылочных деклараций (имеющих атрибут *ref*); $T = \Gamma(T^B, R)$;

B – множество базовых деклараций: $B = \Gamma\left(T^B, N\right)$.

D^{GL} – множество глобальных деклараций; глобальная декларация не может быть ссылочной: $D^{GL} \cap R = \varnothing$.

D^{LOC} – множество локальных деклараций; $D = \Gamma\left(D^{GL}, D^{COR}\right)$;

$d^\wedge$ – функция разыменования, равна глобальной декларации, на которую ссылается $d \in R$;

Opt(d) – предикат, истинный для опциональной декларации (имеющей атрибут *minOccurs*=«0»); глобальная декларация не может быть опциональной;

$M(d)$ – множество деклараций, являющихся членами группы, которой владеет нетерминальная декларация d;

$M^\circ(d)$ – множество опциональных деклараций, являющихся членами группы, которой владеет нетерминальная декларация:

$$M^\circ(d) = \left\{d \in M(d) \middle| Opt(d)\right\};$$

$M^\oplus(d)$ – множество неопциональных деклараций, являющихся членами группы, которой владеет нетерминальная декларация:

$$M^\oplus(d) = \left\{d \in M(d) \middle| \neg Opt(d)\right\};$$

$M^\oplus(d)$ и $M^\oplus(d)$ разбивают $M(d)$.

4. Определение предиката корректности схемы

Спецификация схемы по сути определяет ее как многосортную модель, собственные элементы носителя которой называются «компонентами схемы», а отношения – «свойствами компонентов». Определяется также XML-представление схемы в виде *документа схемы*, причем компоненты отображаются в его элементы, а большинство свойств – в атрибуты.

Важнейшими для нашей задачи являются *декларации элементов*, универсум которых обозначим через ***D***. Множество деклараций $D(S)$ в любой схеме $S \in \boldsymbol{S}_1$ образует непустой лес, корни которого – глобальные декларации, а вторичные узлы – локальные декларации. Каждая декларация d совместно с зависящими от нее компонентами схемы, а зависимость может быть рекурсивной, порождает множество *Valid*(d) соответствующих ей *последовательностей состоятельных элементов*. Последовательности возникают потому, что локальные декларации могут иметь *ограничители повторения (Occurrence Constraints) minOccurs* и *maxOccurs*. Обозначим пустую последовательность элементов через ε, она входит в *Valid*(d), если *minOccurs* $= 0$.

Определение функции $Valid(d)$ составляет значительный объем спецификации; ее тип:

$$Valid: \boldsymbol{D}\times(\mathfrak{P}(\boldsymbol{E}))^{*}, \qquad (3)$$

где $\boldsymbol{E}$ – универсум абстрактных элементов согласно спецификации XML Infoset [7].

Логично считать свойство корректности декларации *не зависящим* от ограничителей повторения; последние будут учтены для декларации, от которой непосредственно зависит рассматриваемая декларация. Имея это в виду, определим важное понятие *экстенсионала декларации* как функцию, формирующую множество ее состоятельных элементов:

$$E: \boldsymbol{D}\times\mathfrak{P}(\boldsymbol{E});\quad E(d)=\{e|(e)\in Valid(d)\setminus\{\varepsilon\}\}.$$

Экстенсионал можно также определить как функцию состоятельности для декларации без учета ограничителей повторения.

Назовем *декларацию корректной*, если ее экстенсионал не пуст:

$$Correct(d) \;\Leftrightarrow\; E(d)\neq\varnothing. \qquad (4)$$

Поскольку каждая глобальная декларация определяет множество состоятельных *элементов документа* (Document Element), а в любом документе XML [8] имеется ровно один такой элемент, с учетом того, что в любой $S\in\boldsymbol{S}_1$ присутствуют глобальные декларации, естественно дать следующее определение состоятельности схемы $S\in\boldsymbol{S}_1$ (используем полиморфное обозначение):

Назовем *схему* $S\in\boldsymbol{S}_1$ *корректной*, если в ней есть хоть одна корректная глобальная декларация:

$$Correct(S) \;\Leftrightarrow\; \exists\left(d\in D^{GL}\right)Correct(d). \qquad (5)$$

Введем также понятия *частичной корректности* и *полной корректности*: корректная схема частично корректна, если в ней есть хоть одна некорректная декларация, в противном случае она полностью корректна:

$$\begin{aligned} &PartiallyCorrect(S)\Leftrightarrow\\ &\quad Correct(S),\ \exists(d\in D)\neg Correct(d);\\ &CompletelyCorrect(S)\Leftrightarrow\\ &\quad \forall(d\in D)\,Correct(d). \end{aligned} \qquad (6)$$

5. Выражение предиката корректности декларации через функцию номера уровня декларации

Описание функции состоятельности деклараций составляет значительную часть спецификации схемы. В то же время, для нашей задачи не требуется иметь точного ее определения, важно лишь знать, порождается ли хоть один элемент. Поэтому разумно искать непосредственное рекурсивное определение предиката корректности.

Анализ показал, что задача практически не усложнится, если мы выразим предикат корректности через функцию $\ell(d)$, возвращающую минимальную высоту *основы дерева состоятельного элемента* декларации либо особое значение, если декларация некорректна. Знание этой функции весьма полезно при семантическом анализе схемы. Под основой дерева элемента мы понимаем редукт дерева элемента, состоящий только из информационных единиц элементов.

Назовем *функцией номера уровня декларации* $\ell(d)$ функцию, определенную на множестве деклараций произвольной схемы S (неявный параметр функции) и равную:

– минимальной высоте основы дерева состоятельного элемента этой декларации, если декларация корректна;

– нечисловому значению ∞, если декларация некорректна:

$$\ell: D\to\mathbb{N}\cup\{\infty\};$$

$$\ell(d)=\begin{cases}\min(h(e))\,|\,e\in E(d), & \text{если } E(d)\neq\varnothing;\\ \infty, & \text{в противном случае.}\end{cases} \qquad (7)$$

Обозначение ∞ имеет содержательный смысл: алгоритм порождения элементов по некорректной декларации никогда не завершится, в качестве промежуточного результата будет формироваться растущий элемент сколь угодно большой высоты.

Предикат корректности очевидным образом выражается через функцию $\ell(d)$:

$$Correct_1(d) \;\Leftrightarrow\; \ell(d)\neq\infty \;\Leftrightarrow\; \ell(d)\in\mathbb{N}. \qquad (8)$$

Теорема 1. Множество числовых значений функции $\ell(d)$ либо пусто, либо представляет собой некоторый начальный интервал натуральных чисел:

$$im(\ell)|_{\mathbb{N}}=i=\overline{1,n},\ \text{где } n\geq 0. \qquad (9)$$

Через n мы обозначили число (точнее – функцию от S), равное максимальному числовому значению функции $\ell(d)$ либо 0, если такого нет (то есть все декларации рассматриваемой схемы некорректны).

Доказательство.

Вначале докажем, что если существует состоятельный элемент $e\in E(d)$ высоты h, то в схеме найдется декларация d', для которой $\ell(d')=\ell(d)-1$. Это следует из:

– свойства высоты произвольного дерева [9]: высота дерева на 1 больше максимальной высоты его поддеревьев; высота одноэлементного дерева равна 1;

– очевидного свойства экстенсионала декларации для $S\in\boldsymbol{S}_1$: если некоторый состоятельный элемент e декларации d входит в состав некоторого состоятельного элемента e' для декларации d', то состоятельным для d' будет и элемент e'', полученный из e' путем замены элемента e на любой иной состоятельный для d элемент.

Теперь требуемое утверждение доказывается по индукции. □

С целью описания алгоритма вычисления $\ell(d)$ введем ряд дополнительных понятий.

Назовем *расширенным образом функции* $\ell(d)$ множество $\mathbb{I}$, определяемое формулой:

$$\mathbb{I} = i = \overline{1,n} \cup \{\infty\}.$$

Назовем *уровнем деклараций* L_i множество деклараций, имеющих одинаковый номер уровня:

$$L_i = \{d \in D \,|\, \ell(d) = i\}, \quad i \in \mathbb{I}.$$

Уровень назовем *корректным*, если его индекс числовой, и *некорректным*, если индекс равен ∞. Уровни деклараций обладают очевидными свойствами: они образуют разбиение множества деклараций, причем корректные уровни не могут быть пустыми (их может вообще не быть, если n=0), а некорректный уровень может быть пуст; множество корректных деклараций обозначим D^{COR}:

$$\begin{aligned} &D = \Gamma\left(\{L_i \,|\, i \in \mathbb{I}\}\right); \\ &D^{COR} = D \setminus L_\infty = \cup\{L_i \,|\, i \in \mathbb{N}\}. \end{aligned} \tag{10}$$

Назовем *объединенным уровнем деклараций* множество:

$$\begin{aligned} U_i = \left\{ \begin{array}{ll} \cup\{L_j \,|\, i \le j\}, & \text{если } i = \overline{1,n}; \\ \varnothing, & \text{в противном случае;} \end{array} \right\}, \\ i \in \mathbb{N} \cup \{0\}. \end{aligned} \tag{11}$$

Заметим, что объединенные уровни определены для всех неотрицательных чисел, не могут содержать некорректный уровень, и представляют собой возрастающую по включению последовательность множеств деклараций, максимальное из которых равно множеству корректных деклараций:

$$\begin{aligned} &d \in U_i \;\Leftrightarrow\; \ell(d) \le i; \\ &U_i \subset U_j \;\; для \;\; i < j; \\ &U_i \subseteq D^{COR}; \;\; U_n = D^{COR}; \\ &d \in U_i \;\Leftrightarrow\; \ell(d) \le i; \\ &\left(i, j = \overline{1,n}\right). \end{aligned} \tag{12}$$

Лемма 1. Вычислимость функции $\ell(d)$ является следствием вычислимости последовательности объединенных уровней $(U_1, U_2, \ldots, U_n)$, для доказательства которой достаточно указать такой вычислимый функционал $\mathcal{F}$, чтобы выполнялась формула:

$$\begin{aligned} &U_i = \mathcal{F}(S, U_{i-1}); \;\; i = \overline{1,n}; \\ &\varnothing = \mathcal{F}(S, U_n). \end{aligned} \tag{13}$$

Комментарий: содержимое любого уровня должно полностью определяться совокупным содержимым предыдущих уровней (и структурой схемы). Истинность утверждения леммы очевидна.

Следствие. Приведенный ниже абстрактный алгоритм вычисляет множество корректных деклараций для любой схемы $S \in \boldsymbol{S}_1$.

Алгоритм 1.

```
function CorrectElemDecl(S : S1) : 𝔓(D);
var  i : integer;
     L, U : 𝔓(D);
begin
     i := 1;  U := ∅;
     repeat
          L := 𝓕(S, U);
          if L ≠ ∅ then
               i := i + 1;
               U := U ∪ L;
          end if;
     until  L = ∅;
     result := U;
end.
```

6. Функция номера уровня для класса схем $\boldsymbol{S}_1$

Теорема 2. Функция номера уровня для $S \in \boldsymbol{S}_1$ определяется формулой (14), которая имеет вид условного выражения:

$$\begin{array}{ll}
\ell(d) \;= & \\
\textit{if} \;\; d \in T^B \;\; \textit{then} & //1 \\
\textit{elif} \;\; d \in R \;\; \textit{then} \;\; \ell(d^\wedge) & //2 \\
\textit{elif} \;\; d \in N^S \;\; \textit{then} & \\
\quad \textit{if} \;\; \forall (d' \in M)\, Opt(d') \;\; \textit{then} \;\; 1 & //3 \\
\quad \textit{elif} \;\; \forall (d' \in M \,|\, \neg Opt(d'))\, \ell(d') \ne \infty \textit{then} & //4 \\
\qquad 1 + \max(\ell(d') \,|\, d' \in M(d), \neg Opt(d')) & \\
\quad \textit{else} \;\; \infty & //5 \\
\quad \textit{endif} & \\
\textit{else} \;\; \{d \in N^C\} & \\
\quad \textit{if} \;\; \exists (d' \in M)\, Opt(d') \;\; \textit{then} \;\; 1 & //6 \\
\quad \textit{elif} \;\; \exists (d' \in M)\, \ell(d') \ne \infty \;\; \textit{then} & //7 \\
\qquad 1 + \min(\ell(d') \,|\, d' \in M(d), \ell(d') \ne \infty) & \\
\quad \textit{else} \;\; \infty & //8 \\
\quad \textit{endif} & \\
\textit{endif}. &
\end{array} \tag{14}$$

Доказательство.

Формально доказательство следует из:

– свойств функции *Valid* [1, 5];

– определения $\boldsymbol{S}_1$, формализованного нами в виде диаграммы UML;

– определения функции $\ell(d)$.

Содержательное доказательство (см. также диаграмму демонстрационного примера схемы, на которой для каждой декларации указан ее номер уровня и номер применяемого случая вышеприведенной формулы).

Случай 1. Любая базовая терминальная декларация может порождать терминальные элементы, высота которых, по определению, равна 1. При этом декларации, имеющие атрибут *type* со значением имени примитивного типа, порождают только терминальные элементы, а имеющие этот атрибут со значением *anyType*, могут порождать произвольные элементы. Если атрибут отсутствует, по умолчанию принимается *anyType* (пример: декларация Any).

Случай 2. Любая ссылочная декларация порождает в точности такое же множество элементов, как и ссылаемая декларация.

Случай 3. Любая нетерминальная декларация с моделью содержания *sequence*, все дочерние декларации которой опциональны, может порождать пустой элемент, то есть терминальный элемент без содержимого (не путать с отсутствующим элементом *e*, входящим в пустую последовательность ε, которую могут порождать опциональные декларации).

Случай 4. Любая нетерминальная декларация с моделью содержания *sequence*, все неопциональные дочерние декларации которой корректны, корректна и имеет номер уровня на 1 больше максимального номера уровня таких дочерних деклараций.

Случай 5. Любая нетерминальная декларация с моделью содержания *sequence*, имеющая хоть одну некорректную неопциональную дочернюю декларацию, некорректна.

Случай 6. Любая нетерминальная декларация с моделью содержания *choice*, имеющая хоть одну опциональную дочернюю декларацию, может порождать пустой элемент.

Случай 7. Любая нетерминальная декларация с моделью содержания *choice*, все дочерние декларации которой неопциональны и хоть одна из них корректна, является корректной и ее номер уровня на 1 больше минимального номера уровня корректных дочерних деклараций.

Случай 8. Любая нетерминальная декларация с моделью содержания *choice*, все дочерние декларации которой неопциональны и некорректны, некорректна.

7. Разбиение на подуровни

Из теоремы 2 (случай 2) следует, что любой корректный уровень L_i можно разбить на *основной подуровень* L_i^B и *ссылочный подуровень* L_i^R, причем последний вычисляется на основе первого (значением $d^\wedge$ всегда является некоторая глобальная декларация, которая сама не может быть ссылочной):

$$L_i = \Gamma(L_i^B, L_i^R);\quad i=\overline{1,n}. \tag{15}$$

Для удобства анализа разобьем дополнительно базовые подуровни для $i \ge 2$ на подмножества:

$$L_i^B = \Gamma\left(L_i^S, L_i^C\right),$$
$$\text{где: } L_i^S = L_i \cap N^S;\ L_i^C = L_i \cap N^C; \tag{16}$$
$$i=\overline{1,n}.$$

Следствие теоремы 2. Корректные уровни деклараций для $S \in \boldsymbol{S}_1$ определяются формулами:

$$L_i = \Gamma\left(L_i^B = \begin{pmatrix} \text{if } i=1 \text{ then } L_1^B \\ \text{else } \Gamma\left(L_i^S, L_i^C\right)\end{pmatrix},\ L_i^R\right),\quad i=\overline{1,n};$$

$$L_1^B = \left\{d \in N^S \middle| M^\oplus(d)=\varnothing\right\} \cup \left\{d \in N^C \middle| M^\circ(d) \ne \varnothing\right\};$$

$$L_i^S = \left\{d \in N^S \middle| \begin{matrix} M^\oplus(d) \ne \varnothing,\ \forall d^\oplus \ell(d^\oplus) \ne \infty, \\ i = 1 + \max(\ell(d^\oplus) | d^\oplus \in M^\oplus)\end{matrix}\right\};$$

$$L_i^C = \left\{d \in N^C \middle| \begin{matrix} M^\circ(d) = \varnothing,\ \exists d' \ell(d') \ne \infty, \\ i = 1 + \min(\ell(d') | d' \in M)\end{matrix}\right\};$$
$$i=\overline{2,n};$$

$$L_i^R = \left\{d \in R \middle| d^\wedge \in L_i^B\right\};\quad i=\overline{1,n}.$$

Формула для L_i^B не пригодна для непосредственного вычисления, так как представляет собой рекурсивное определение. Преобразуем ее, заменив функции *max* и *min* на их кванторные представления:

$$k = \max(\ell(d^\oplus) | d^\oplus \in M^\oplus) \Leftrightarrow$$
$$\forall d^\oplus\ \ell(d^\oplus) \ne \infty,\ \exists d^\oplus\left(\ell(d^\oplus) = k\right),$$
$$\forall d^\oplus\left(\ell(d^\oplus) \le k\right);$$
$$k = \min(\ell(d^\oplus) | d^\oplus \in M^\oplus) \Leftrightarrow$$
$$\exists d^\oplus\left(\ell(d^\oplus) \ne \infty, \ell(d^\oplus) = k\right),$$
$$\forall (d^\oplus | \ell(d^\oplus) \ne \infty)\left(\ell(d^\oplus) \ge k\right).$$

В результате получим:

$$L_i^S = \left\{d \in N^S \middle| \begin{matrix} M^\oplus(d) \ne \varnothing, \\ \forall d^\oplus \ell(d^\oplus) \ne \infty, \\ \exists d^\oplus\left(\ell(d^\oplus) = i-1\right), \\ \forall d^\oplus\left(\ell(d^\oplus) \le i-1\right)\end{matrix}\right\},$$

$$L_i^C = \left\{d \in N^C \middle| \begin{matrix} M^\circ(d) = \varnothing, \\ \exists d'\left(\ell(d') \ne \infty, \ell(d') = i-1\right), \\ \forall (d' | \ell(d') \ne \infty)\left(\ell(d') \ge i-1\right)\end{matrix}\right\},$$
$$i=\overline{2,n}.$$

Используя формулы для L_i, U_i, упрощаем:

$$L_i^S = \left\{ d \in N^S \middle| \begin{matrix} M^{\oplus}(d) \neq \varnothing, & \exists d^{\oplus} \in L_{i-1}, \\ \forall d^{\oplus} \in U_{i-1} \end{matrix} \right\}, \quad i = \overline{2,n}; \tag{17}$$

$$L_i^C = \left\{ d \in N^C \middle| \begin{matrix} M^{\ominus}(d) = \varnothing, & \exists d' \in L_{i-1}, \\ \forall d' \notin U_{i-2} \end{matrix} \right\}, \quad i = \overline{2,n}. \tag{18}$$

Лемма 2. Множества L_i^S, L_i^C можно задать формулами:

$$L_i^S = \left\{ d \in N^S \setminus U_{i-1} \middle| \forall d^{\oplus} \in U_{i-1} \right\}, \quad i = \overline{2,n}; \tag{19}$$

$$L_i^C = \left\{ d \in N^C \setminus U_{i-1} \middle| \exists d' \in U_{i-1} \right\}, \quad i = \overline{2,n}. \tag{20}$$

Доказательство (демонстрируется рис. 8).

Обозначим через $\widetilde{L_i^S}, \widetilde{L_i^C}$ множества, определяемые формулами (19), (20).

• Докажем равенство $L_i^S = \widetilde{L_i^S}$;

Пусть $d \in L_i^S$, тогда $d \notin U_{i-1}$. Все остальные конъюнкции в (19) истинны, поскольку присутствуют в (17), следовательно, $d \in \widetilde{L_i^S}$.

Обратно, пусть $d \in \widetilde{L_i^S}$. Тогда:

1. d корректна, поскольку все ее дочерние декларации корректны (ф-ла (14), случаи 3,4).

2. Уровень d не ниже i, так как $d \notin U_{i-1}$. Следовательно, существует $j \geq i$ и L_j^S такие, что $d \in L_j^S$.

3. Осталось доказать, что $j = i$. Действительно, подставляя $d \in L_j^S$ в (17), получим: $\exists d^{\oplus} \in L_{j-1}$ (если отцовская декларация имеет уровень j, то есть неопциональная дочерняя декларация уровня $j-1$ (14), случай 4); с учетом $j \geq i$ уровень последней не ниже $i-1$: $\exists d^{\oplus} \notin U_{i-2}$. С другой стороны, согласно посылке дочерние декларации имеют уровень $i-1$ или ниже: $\forall d^{\oplus} \in U_{i-1}$. Следовательно, существует неопциональная дочерняя декларация уровня $i-1$: $\exists d^{\oplus} \in L_{i-1}$, и, по (17), $d \in L_i^S$ (мы доказали истинность всех конъюнкций в этой формуле).

• Докажем равенство $L_i^C = \widetilde{L_i^C}$.

Пусть $d \in L_i^C$, тогда $d \notin U_{i-1}$ и $\exists d' \in L_{i-1}$. Из второго следует, что $\exists d' \in U_{i-1}$. Все остальные конъюнкции в (20) истинны, поскольку присутствуют в (18), следовательно, $d \in \widetilde{L_i^S}$.

Обратно, пусть $d \in \widetilde{L_i^C}$. Тогда:

1. d корректна, поскольку у нее есть корректная дочерняя декларация (14), отрицание случая 7).

2. Уровень d не ниже i, так как $d \notin U_{i-1}$. Следовательно, существует $j \geq i$ и L_j^C такие, что $d \in L_j^C$.

3. Осталось доказать, что $j = i$. Действительно, подставляя $d \in L_j^C$ в (18), получим: $\forall d' \notin U_{j-2}$ (если отцовская декларация имеет уровень j, то все дочерние должны иметь уровень не ниже $j-1$); с учетом $j \geq i$ имеем $\forall d' \notin U_{i-2}$. С другой стороны, согласно посылке существует дочерняя декларация уровня $i-1$ или ниже: $\exists d' \in U_{i-1}$. Следовательно, существует дочерняя декларация уровня $i-1$, $\exists d' \in L_{i-1}$, и, по (18), $d \in L_i^C$ (мы доказали истинность всех конъюнкций в этой формуле). □

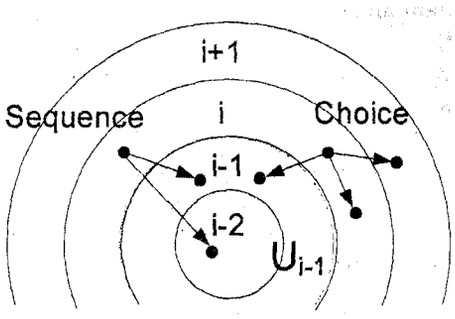

Рис. 8. Иллюстрация к лемме 2

Практический смысл доказанной леммы: определяющий предикат для L_i значительно упрощается, если не рассматривать множество U_{i-1} деклараций, заведомо не принадлежащих L_i. Применительно к алгоритму вычисления уровней это означает, что на очередном шаге мы не анализируем декларации из ранее вычисленных уровней.

Замечание. При подстановке в правые части формул (19) и (20) значения $i = n+1$ получим $\varnothing$, поскольку для всех элементов этих множеств $\ell(d) > n$ (так как $d \notin U_n$), чего, по теореме 1, не может быть ни для одного d. Этот факт далее используется как условие окончания алгоритма 2.

Лемма 3.

Функция $\ell(d)$ для S_1 вычислима, причем функционал $\mathcal{F}(S,U)$ из леммы 1 вычисляется следующей последовательностью операторов присваивания:

$$\begin{aligned} L^B := \ & \textbf{if} \quad U = \varnothing \quad \textbf{then} \\ & T^B \cup \left\{ d \in N^S \middle| M^{\oplus}(d) = \varnothing \right\} \\ & \cup \left\{ d \in N^C \middle| M^{\ominus}(d) \neq \varnothing \right\} \\ & \textbf{else} \\ & \left\{ d \in N^S \setminus U \middle| \forall d^{\oplus} \in U \right\} \\ & \cup \left\{ d \in N^C \setminus U \middle| \exists d^{\oplus} \in U \right\}; \\ & \textbf{end if}; \\ L^R := & \left\{ d \in R \middle| d^{\wedge} \in L^B \right\}; \\ L := & \ L^B \cup L^R. \end{aligned} \tag{21}$$

Доказательство: следует из следствия к теореме 2 и леммы 2. □

Теорема 3.

a) Необходимым и достаточным условием корректности произвольной декларации $d \in S \in \boldsymbol{S}_1$ является ее принадлежность множеству D^{COR}, вычисляемому алгоритмом 1 после замены в нем оператора присваивания $L := \mathcal{F}(S,U)$ на последовательность, описанную в лемме 3.

b) Необходимым и достаточным условием корректности схемы $S \in \boldsymbol{S}_1$ является корректность хотя бы одной ее глобальной декларации.

Доказательство: следует из леммы 1 и леммы 3.

Следствие. Приведенный ниже Алгоритм 2 осуществляет анализ корректности схемы $S \in \boldsymbol{S}_1$.

Замечание. При записи данного алгоритма мы перешли к нотации, более близкой к традиционным языкам программирования. Для этого:

– выразили множества $M^{\oplus}, M^{\circ}$ с помощью множества M и предиката $Opt(d)$;

– оператор присваивания для L^R заменили на эквивалентный *в контексте предлагаемого алгоритма*, то есть с предусловием $\{U = U_{i-1} \cup L_i^B\}$:

$$L^R := \left\{ d \in R \setminus U \middle| d^{\wedge} \in U \right\};$$

(переменная U на шаге i цикла последовательно принимает следующие значения:

$$U_{i-1},\ U_{i-1} \cup L_i^B,\ U_{i-1} \cup L_i^B \cup L_i^R = U_i\ );$$

– выполнили ряд дополнительных несложных преобразований.

Алгоритм 2.

function $CheckSchema(S : \boldsymbol{S}_1)$:

$$\left\{ \begin{array}{l} D^{COR} : set\ of(\mathrm{D}); \\ n : integer; \quad // \text{ Число корректных уровней} \\ RC : \{Incorrect, PartiallyCorrect, CompletelyCorrect\} \end{array} \right\};$$

var $i : integer$; // *N текущего уровня деклараций*

$L, U, D^{COR} : set\ of(\mathrm{D})$;

// *Здесь должны быть определены в терминах*

// *DOM – интерфейса* [10] *функции от* S:

// $T^B, N^S, N^C, M, R,$

// $d^{\wedge}, D, D^{GL}$ *и предикат* $Opt(d)$.

begin

$i := 1;\ U := \varnothing;\ D^{COR} := \varnothing;$

repeat

// *Здесь* $U = U_{i-1}$

if $i = 1$ ***then***

$$L := T^B \cup \left\{ d \in N^S \middle| \forall (d' \in M(d)) \neg Opt(d') \right\}$$

$$\cup \left\{ d \in N^C \middle| \exists (d' \in M(d))\ \neg Opt(d') \right\}$$

else

$$L := \left\{ d \in N^S \setminus U \middle| \forall \left(d' \in M(d) \middle| \neg Opt(d') \right) d' \in U \right\}$$

$$\cup \left\{ d \in N^C \setminus U \middle| \exists (d' \in M(d))\ d' \in U \right\};$$

end if; // *Здесь* $L = L_i^B$

$U := U \cup L;$ // *Здесь* $U = U_{i-1} \cup L_i^B$

$U := U \cup \left\{ d \in R \setminus U \middle| d^{\wedge} \in U \right\};$ // *Здесь* $U = U_i$

if $L \neq \varnothing$ ***then*** $i := i + 1;$ ***end if***;

until $L = \varnothing;$

$n := i - 1;$

$D^{COR} := U;$

if $D^{COR} \cap D^{GL} = \varnothing$ ***then*** $RC := Incorrect;$

elif $D^{COR} = D$ ***then*** $RC := CompletelyCorrect;$

else $RC := PartiallyCorrect;$

end if;

end.

Алгоритм несложно усовершенствовать так, чтобы он определял номер уровня и, следовательно, корректность отдельных деклараций. Мы предлагаем следующий метод: в каждую декларацию алгоритм вставляет специальный атрибут *пользовательского пространства имен* [1], значением атрибута является номер уровня декларации. Такое решение максимально упрощает реализацию и обладает тем достоинством, что информация о корректности деклараций сохраняется в схеме. Объявление указанного атрибута в нашей реализации имеет вид:

```
xmlns:СНК=http://SergTim/CheckSchema.
```

На рис. 9 показана трасса программы, разработанной в соответствии с усовершенствованным алгоритмом, при анализе демонстрационной схемы.

8. Заключение

Предварительное исследование показало, что приведенные в статье принципы решения проблемы зацикливаний в схемах класса $\boldsymbol{S}_1$ применимы также к анализу схем общего вида. Введенные в работе понятия и обозначения должны упростить эту задачу.

Считаем полезным заметить, что в ходе работы над статьей мы обнаружили, что отношения между математическими объектами удобно задавать в виде диаграммы классов UML, которую следует рассматривать как *высокоуровневое формальное определение в графовых терминах*.. В результате существенно уменьшается объем и увеличивается наглядность математического описания. Конечно, указанный факт требует отдельного рассмотрения.

```
 1   1   Случай 1  A/Er1/Data
 2   1   Случай 1  A/Loc1/Loc2/Any
 3   1   Случай 3  A/Loc1/MayEmp1
 4   1   Случай 6  B/MayEmp2
 5   1   Случай 1  data
 6   1   Случай 2  A/Loc1/data^
 7   1   Случай 2  B/data^
 8   2   Случай 7  A/Loc1/Loc2
 9   3   Случай 4  A/Loc1
10   4   Случай 7  A
11   4   Случай 2  A/Loc1/Loc2/A^
12   4   Случай 2  A/Loc1/MayEmp1/Loc3/A^
13   5   Случай 4  A/Loc1/MayEmp1/Loc3
14   ∞   Случай 2  A/B^
15   ∞   Случай 5  A/Er1
16   ∞   Случай 2  A/Er1/Loop^
17   ∞   Случай 2  A/Loc1/Loc2/Loop^
18   ∞   Случай 2  A/Loc1/MayEmp1/Loop^
19   ∞   Случай 5  B
20   ∞   Случай 2  B/MayEmp2/B^
21   ∞   Случай 2  B/MayEmp2/Loop^
22   ∞   Случай 8  B/Er2
23   ∞   Случай 2  B/Er2/B^
24   ∞   Случай 2  B/Er2/Loop^
25   ∞   Случай 5  Loop
26   ∞   Случай 2  Loop/Loop^
Конец.
```

Рис. 9. Трасса программы при анализе демонстрационной схемы

Перспективы дальнейших исследований в данном направлении состоят в решении проблемы зацикливаний для произвольных схем XML, где зацикленными могут быть также дефиниции типов и групп.

Надеемся, что предложенная работа будет полезна специалистам по технологии XML.

Список литературы: 1. *XML Schema Part* 1: Structures Second Edition W3C Recommendation 28 October 2004 http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. 2. *XMLSpy 2005* Home Edition. http://www.altova.com. 3. *M. Murata, D. Lee, and M. Mani.* Taxonomy of XML Schema Languages using Formal Language Theory. In Extreme Markup Languages, Montreal, Canada, 2001. 4. *Tree Automata* Techniques and Applications. http://www.grappa.univ-lille3.fr/tata/. 1997. 5. *XML Schema:* Formal Description. W3C Working Draft, 25 September 2001. http://www.w3.org/TR/2001/WD-xmlschema-formal-20010925/. 6. *Жолткевич Г.Н., Семенова Т.В.* К проблеме формализации концептуального моделирования информационных систем // Вісник Харк. нац. ун-ту ім. В.Н. Каразіна. Сер. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. 2003. № 605. Вип. 2. С. 33-42. 7. *XML* Information Set W3C Recommendation 24 October 2001. http://www.w3.org/TR/2001/REC-xml-infoset-20011024. 8. *Extensible* Markup Language (XML) 1.0 (Third Edition) W3C Recommendation 04 February 2004 http://www.w3.org/TR/2004/REC-xml-20040204. 9. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. М.: Мир, 1978. Т.1. 612 с. 10. *Document* Object Model (DOM) Level 1 Specification (Second Edition) Version 1.0 W3C Working Draft 29 September, 2000. http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929.

Поступила в редколлегию 25.05.2005