УДК 519.7

С.П. Тимофеев

ПРЕДСТАВЛЕНИЕ СХЕМ ТРАНСЛЯЦИИ СХЕМАМИ XML (с приложением к компьютерной лингвистике)

1. Постановка задачи

Схема трансляции – это формализм, используемый для задания *трансляции*, определяемой как отображение между одним (входным) и другим (выходным) формальным языком. Трансляция может быть также определена как отображение между деревьями разбора входной и выходной формальной грамматики. Схемы трансляции делятся на *синтаксически управляемые (СУ-схемы)* и *обобщенные (общего вида)*. Первые являются полностью декларативными описаниями и представляют собой две взаимосвязанные контекстно-свободные грамматики (КСГ); вторые являются частично процедурными и представляют собой входную КСГ, дополненную *элементами трансляции (семантическими процедурами)*. СУ-схемы описывают достаточно узкий, хотя и часто встречающийся класс трансляций; с помощью обобщенных схем можно описать трансляции произвольной сложности. Схемы трансляции используются в средствах автоматизации разработки трансляторов — генераторах трансляторов, называемых также компиляторами компиляторов, генераторами синтаксических анализаторов [1, 2, 3, 4].

Схема XML – это описание некоторого класса документов XML, называемых состоятельными по отношению к данной схеме. Описание имеет вид вариантной структуры общего элемента этого класса. Известно, что схему XML можно рассматривать как записанную в особой нотации формальную грамматику, задающую формальный язык, цепочками которого являются состоятельные документы XML [5, 6].

Технология XML обладает и другими полезными для нашей задачи средствами:

– абстрактная модель данных XML Infoset [7] определяет универсум древовидных структур различной конфигурации (деревьев XML);

– язык XML определяет де-факто стандартное линейное представление древовидных структур в виде документов XML [8];

– программные средства технологии XML обеспечивают разнообразные операции манипулирования и визуализации для деревьев и схем XML [9, 10].

Известно, что построение трансляторов весьма трудоемко и принадлежит наивысшей сфере искусства программирования [4]. В связи с развитием средств общения человека с компьютером на естественном языке (точнее, на формализованных подмножествах ЕЯ), является *актуальной* проблема упрощения и ускорения разработки трансляторов [11, 12]. Примером служит актуальная задача компьютерной лексикографии – трансляция традиционных словарей в лексикографические БД [13, 14, 15]. Настоящая работа возникла в результате анализа именно этой задачи.

Другой актуальной задачей, требующей создание трансляторов, является *задача преобразования документов произвольного формата в документы XML* [15]. Перевод всех документов, находящихся в Сети, на XML – необходимое условие повышения точности поиска. В связи с этим заметим, что технология XML находится еще в стадии становления и далеко не все ее задачи, например, рассматриваемая, решены. Также важно заметить, что указанная выше лексикографическая задача может рассматриваться как частный случай этой задачи технологии XML.

Применительно к названным задачам, которые будем называть *целевыми*, мы обнаружили *пять недостатков* традиционных схем трансляции.

1. Выходная грамматика в обобщенных схемах трансляции определяется *неявно*, что не соответствует обычной логике разработки словаря *от содержания к форме* [12]. Аналогично, в технологии XML документ XML, представляющий некоторое информационное содержание, первичен, а его различные *презентации* вторичны.

2. *Линейный формат*. Известно, что как только задачу удается переложить на графовый язык и к тому же перейти к визуальному проектированию, трудоемкость решения резко снижается.

3. Нет встроенных средств манипулирования *деревьями разбора*.

4. Нет встроенных средств *диалоговой трансляции*. Данная возможность и не требуется в трансляторах языков программирования, но бывает весьма полезно в задачах компьютерной лингвистики, когда не удается (или весьма трудоемко) полностью формализовать некоторое подмножество ЕЯ и приходится довольствоваться неоднозначной грамматикой.

5. Нет встроенных средств, обеспечивающих работу с *обогащенными текстами* (RTF, DOC и т. д.) на уровне шрифтовых атрибутов, и приходится работать с байтовым кодом.

Некоторые из затронутых проблем нашли свое частичное решение в следующих работах.

В работах [13, 14, 15] введены двухуровневые диалоговые грамматики и двухуровневые схемы XML в качестве средства описания структуры сло-

варной статьи, а также предложены принципы создания лексикографических БД на основе словарей. Настоящая работа представляет собой развитие идей, сформулированных в этих работах.

Задачами исследования являются:

– Выявление отличительных особенностей целевых задач с точки зрения трансляции, а также содержательное определение соответствующего класса трансляций.

– Разработка такого варианта организации схемы трансляции, который был бы лишен указанных недостатков применительно к задачам этого класса.

– Разработка концепции программной реализации соответствующих трансляторов.

– Демонстрация выдвинутых предложений на простом примере из области компьютерной лексикографии.

2. Предлагаемые принципиальные решения

2.1. Для поддержки логики проектирования словарей и других документов «от содержания к форме» предлагается аппарат *обратных схем трансляции*, представляющих собой *выходную* грамматику, дополненную элементами трансляции. Традиционные схемы трансляции в данном контексте естественно назвать *прямыми схемами*. Существенным здесь является интерпретация выходной грамматики как *исчисления деревьев*, а не цепочек (что более адекватно в области лингвистики [16]), причем деревья порождения выходной грамматики отождествляются с деревьями содержательной структуры входного текста. Такое решение позволяет в ряде случаев не вводить операции конструирования выходного дерева – оно будет формироваться *по умолчанию* транслятором; в остальных случаях потребуются операции только для некоторых поддеревьев.

2.2. Для перехода к графовому представлению схем трансляции (как прямых, так и обратных) предлагается использовать определенным образом организованные схемы XML, названные нами *XML-схемами трансляции*. Идея заключается в том, чтобы рассматривать схемы XML как представления *произвольных* формальных грамматик, безотносительно к документам XML. Такая интерпретация связана с упомянутой выше трактовкой формальных грамматик как исчислений деревьев. Элементы трансляции записываются в качестве значений особых атрибутов и, возможно, элементов схемы XML. Визуальное проектирование XML-схем трансляции выполняется посредством редакторов XML (мы предпочли XMLSpy [10]).

2.3. Для представления деревьев разбора предлагается использовать *деревья XML*, причем:

– в случае трансляции общего вида использовать *DOM-интерфейс*, обеспечивающий операции произвольного манипулирования деревьями [9];

– для важного класса L-атрибутных трансляций [1], не требующих явного построения входного дерева разбора и выполняемых за один проход в процессе восходящего синтаксического анализа, можно ограничиться *линейным* представлением выходного дерева в виде документа XML, что максимально упрощает реализацию.

2.4. Принципы организации диалоговой трансляции описаны нами в [14]. Введенные в этой работе *фразы описания диалога* предлагаем представлять в составе XML-схем трансляции особыми атрибутами и элементами.

Предлагаем также новый метод трансляции для случая неоднозначной входной грамматики, не требующий специальных средств для организации диалога – *метод уточняющих пометок*, суть которого в следующем. Неоднозначная грамматика предварительно превращается в *однозначную* путем вставки дополнительных символов либо замены вхождений каждого неоднозначно интерпретируемого символа на некоторый новый символ, соответствующий контексту. В приводимом далее примере запятая в смысле «разделитель элементов словарной статьи» заменяется на «,||», а в смысле «знак препинания внутри элемента словарной статьи» – на «,|». В процессе трансляции по уточненной грамматике, при обнаружении любой запятой во входном тексте будет возникать синтаксическая ошибка и эксперт должен будет приписать к запятой одну или две вертикальные черточки, в зависимости от контекста, смысл которого программа самостоятельно определить не может. Таким образом, «проблемный» диалог приобретает форму диалога корректировки синтаксических ошибок и может быть реализован стандартными средствами генераторов трансляторов.

2.5. Для трансляции обогащенных текстов предлагаем два метода:

– Предварительное конвертирование входного текста в простое *подмножество HTML* (без тегов *meta* и с фиксированным порядком вложенности тегов, определяющих атрибуты шрифта), после чего осуществляется трансляция простого (необогащенного) текста.

– Разрабатывается программа особого *лексического анализатора, чувствительного к шрифтовым атрибутам*. Для каждой используемой их комбинации программа выдает отдельный вариант токена (например, *RusWord_bold_italic* – русская словоформа, набранная полужирным курсивом). Данный вариант более трудоемок в реализации, но более удобен в использовании.

Более подробно пункты 4 и 5 предполагается раскрыть в отдельной статье.

3. Используемые термины и обозначения

Термином *грамматика* будем обозначать некоторую КСГ, т.е. четверку:

$$G=(\Sigma,N,Z,P),$$

где Σ – множество терминальных символов (алфавит языка); Z – начальный символ, $Z \in N$; N – множество нетерминальных символов; P – множество правил вывода, в которых допускаем регулярные выражения.

Примечание. Регулярные выражения необходимы, чтобы деревья разбора грамматики могли иметь произвольную конфигурацию и напрямую соответствовали деревьям содержательной структуры. Грамматики с простыми правилами (продукциями), используемые в генераторах трансляторов семейства Jacc [1], этому условию не удовлетворяют.

$L(G)$ – (формальный) язык, задаваемый грамматикой G.

$T(G)$ – множество деревьев разбора (порождения), задаваемое грамматикой G.

$G|A$ – проекция грамматики G на нетерминал A, т.е. грамматика, получаемая из G путем назначения нетерминала A стартовым символом и удаления всех символов и правил, не зависящих, прямо или косвенно, от A.

$T^{XML}(G|A)$ – представление деревьев $T(G|A)$ в линейном виде – в виде множества элементов XML.

G_1, G_2 – *входная* и *выходная грамматики*. Соответствующими индексами помечаются компоненты обеих грамматик. Деревья разбора входной грамматики $T(G_1)$ будем называть *деревьями поверхностной структуры*, а деревья порождения выходной грамматики $T(G_2)$ – *деревьями содержательной структуры*.

Функция трансляции:

$$\tau: L(G_1) \to T(G_2).$$

Примечание. Выбранный нами вариант определения типа функции трансляции соответствует исследуемому классу трансляций. Множество $T(G_2)$ естественно назвать выходным *древесным формальным языком*, учитывая тенденцию к максимально широкой трактовке термина «язык» [12]. Заметим, что в работах по семантике схем XML [17] под древесным языком понимается иная математическая структура.

СУ-схема трансляции – это пятерка:

$$S^0=(\Sigma_1,\Sigma_2,N,Z,P^0),$$

где Σ_1,Σ_2,N,Z – определены выше; P^0 – множество *правил трансляции* вида:

$$A \to \alpha,\beta,$$

где $\alpha \in (N \cup \Sigma_1)^*$, $\beta \in (N \cup \Sigma_2)^*$,
причем вхождения нетерминалов в цепочку β образуют перестановку вхождений нетерминалов в цепочку α.

СУ-схема называется *простой*, если сохраняется порядок вхождений нетерминалов.

Схему трансляции общего вида можно определить как пару:

$$S=(G_1,E_1),$$

где E_1 – множество *элементов трансляции* совместно с соответствием между ними и вхождениями грамматических символов в правилах (вместо задания соответствия можно доопределить структуру правил грамматики). Заметим, что контекстные ограничения неявно контролируются элементами трансляции с использованием побочного эффекта: в случае их нарушения генерируется сообщение об ошибке. В рамках атрибутных трансляций, элементы трансляции рассматриваются как вызовы функций вычисления синтезируемых и наследуемых атрибутов [1]. Синтезируемый атрибут, ассоциированный с начальным символом грамматики, представляет выход транслятора; в наших целевых задачах это дерево содержательной структуры.

Токены (называемые также не совсем корректно лексемами) – это терминальные символы уровня синтаксического анализа. Лексические значения всех токенов в наших примерах имеют строковый тип [1].

4. Пример из компьютерной лексикографии

Рассмотрим трансляцию в документ XML фрагмента словаря [18] в составе трех специально подобранных словарных статей:

(1) **asg, asgn** *см.* **assign**
(2) **first-in, first-out** в порядке поступления, 'первым пришёл – первым вышел'
(3) **code-dependent system** система, зависящая от данных, кодозависимая система. *См.* **code-sensitive system**

Рис. 1. Фрагмент словаря

Соответствующая этому фрагменту упрощенная структура словаря может быть описана следующим образом.

Корпус словаря (Dic) представляет собой последовательность словарных статей (Entry), оформленных в виде абзацев и упорядоченных по алфавиту. Заглавный термин (Term) выделен полужирным шрифтом. В статье может быть несколько заглавных терминов, если они расположены рядом в порядке алфавита и имеют идентичное описание. Правая часть статьи имеет два варианта структуры. Первый вариант: отсылка на развернутый термин (RefFull). Второй вариант: набор переводных эквивалентов (Tran), разделяемых запятыми, и необязательная отсылка на синоним (RefSyn). Переводные эквиваленты набираются обычным шрифтом. Отсылочные пометы выделяются курсивом: *«см.»* (на разверну-

тый термин) и «*См.*» (на синоним). Второстепенные детали типографского оформления мы опустим.

На рис. 3 и 4 изображены грамматики G_1 и G_2, соответствующие вышеприведенному описанию.

На рис. 5 изображено дерево содержательной структуры для фрагмента словаря и представляющий это дерево документ XML. Дерево поверхностной структуры приведено в [13].

Принципиальную роль играет предлагаемая нами *модифицированная нотация*, в которой записаны грамматики: содержательные токены выделены полужирным шрифтом, а форматирующие – подчеркнуты. Благодаря такой нотации, выходная грамматика может быть получена *автоматически* из входной путем удаления форматирующих токенов и последующего упрощения регулярных выражений. Несложно увидеть, что входная грамматика в модифицированной нотации оказывается *представлением простой СУ-схемы трансляции*, описывающей преобразование фрагмента словаря в документ XML. На рис. 6 представлен фрагмент эквивалентной схемы трансляции в нотации Yacc, который генерирует программу транслятора на языке C. Результат трансляции заносится в глобальную строковую переменную Out. Из-за недопустимости регулярных выражений приходится вводить дополнительные нетерминалы, а ввиду отсутствия подразумеваемого определения выходной грамматики, в элементы трансляции вставляются операции конструирования линейного представления выходного дерева в форме документа XML.

Отметим, что входная грамматика *неоднозначная*, т.к. запятая, в зависимости от контекста, обозначает либо разделитель элементов словарной статьи, либо знак препинания внутри одного элемента. В схеме трансляции для Yacc мы использовали метод уточняющих пометок, однако более комфортным для эксперта является *диалоговая* трансляция, экран диалога будет иметь вид:

Укажите смысл запятой:

- Разделитель элементов словарной статьи
- Знак препинания внутри одного элемента

Рис. 2. Уточняющий диалог

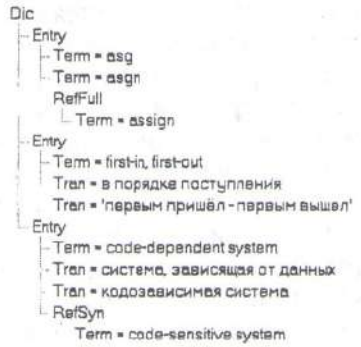
В соответствии с выбором эксперта, транслятор будет использовать нужное правило грамматики для обработки запятой.

```
   Правила синтаксического уровня
Dic        →  [Entry (EOL Entry)*] [EOL]
Entry      →  Term (',' Term)* ( RefFull | Tran (',' Tran)* ['.' RefSyn] )
RefFull    →  NoteFull Term
RefSyn     →  NoteSyn  Term

   Правила лексического уровня
Term       →  EngWord-bold ( [','] EngWord-bold)*
Tran       →  ["'"] RusWord-bold ( [',' | ' - '] RusWord-bold)* ["'"]
EOL        →  <Конец абзаца: #$D#$A>
NoteFull   →  <Строка  'см.', набранная курсивом>
NoteSyn    →  <Строка  'См.', набранная курсивом>
EngWord-bold →  <Английское слово, набранное полужирным шрифтом:
                  EngChar+ [ '-' EngChar+ ] >
RusWord-bold →  <Русское слово, набранное полужирным шрифтом:
                RusChar+['-'RusChar+] >
```

Рис. 3. Входная грамматика для фрагмента словаря в модифицированной нотации

```
   Правила синтаксического уровня
Dic        →  Entry*
Entry      →  Term+ ( RefFull | Tran+ [RefSyn] )
RefFull    →  Term
RefSyn     →  Term

   Правила лексического уровня
Term       →  EngWord-bold ( [','] EngWord-bold)*
Tran       →  ["'"] RusWord-bold ( [',' | ' - '] RusWord-bold)* ["'"]
EngWord-bold →  <Английское слово, набранное полужирным шрифтом:
                  EngChar+ [ '-' EngChar+ ] >
RusWord-bold →  <Русское слово, набранное полужирным шрифтом:
                RusChar+['-'RusChar+] >
```

Рис. 4. Выходная грамматика для фрагмента словаря в модифицированной нотации




```
<?xml version="1.0" encoding="Windows-1251"?>
<Dic>
<Entry><Term>asg</Term><Term>asgn</Term>
<RefFull><Term>assign</Term></RefFull></Entry>
<Entry><Term>first-in, first-out</Term><Tran>в порядке
поступления</Tran><Tran>'первым пришёл - первым
вышел'</Tran></Entry>
<Entry><Term>code-dependent
system</Term><Tran>система, зависящая от
данных</Tran><Tran>кодозависимая
система</Tran><RefSyn><Term>code-sensitive
system</Term></RefSyn> </Entry>
</Dic>
```

Рис. 5. Дерево содержательной структуры для фрагмента словаря и представляющий это дерево документ XML

```
Dic      :  DicBody   {Out=Prologue + "<Dic>" + $1 + "</Dic>"};
DicBody  :  Entry                  {$$="<Entry>"+$1+"</Entry>"}
         |  DicBody Entry          {$$=$1+$2}      ;
Entry    :  Left Right             {$$=$1+$2}      ;
Left     :  Term                   {$$="<Term>"+$1+"</Term>"}
```

Рис. 6. Фрагмент схемы трансляции общего вида в нотации Yacc (с уточненной входной грамматикой)

5. Содержательное определение исследуемого класса трансляций

Анализ указанных выше задач, решаемых аппаратом трансляции, позволил выявить следующие их *особенности*.

5.1 Трансляция заключается в переходе от *формы* к *содержанию* для некоторого класса информационных объектов (словарей, документов). В теории трансляции в этой связи используются термины *конкретный синтаксис* и *абстрактный синтаксис*. В технологии XML: *структура представления (презентации)* документа и его *содержательная структура* (структура *информационного наполнения*). В лингвистике: *поверхностная* и *глубинная* структура текста. В лексикографии: *словарь* и *словарная информация* (абстрактная *лексикографическая система* [12]).

5.2 Содержательная структура является деревом порождения выходной грамматики. Отсюда следует, что *необходимо использовать регулярные выражения* в правилах грамматики (см. п. 3).

5.3 Токены обоих грамматик можно разбить на два класса: *содержательные* и *разметочные*. Содержательные токены являются *типами содержательных элементов данных*, которые будут сохранены в БД. Выходная грамматика содержит только такие токены; соответственно, все листья выходного дерева помечены ими. Входная грамматика содержит как содержательные, так и разметочные токены; последние необходимы для представления *древесной* содержательной структуры посредством *линейной* структуры входного текста. Разметочные токены необходимы для *распознавания структуры* входного текста, их можно также назвать *структурирующими* или *форматирующими*.

5.4 Имеется значительная «близость» между деревьями разбора G_1 и соответствующими им деревьями порождения G_2. Близость заключается в том, что входное дерево или значительный его корневой сегмент получается из выходного дерева (или его корневого сегмента) путем прикрепления к некоторым внутренним узлам добавочных листьев, помеченных разметочными токенами; пример см. в [13]. Соответственно, выходное дерево может быть получено из входного путем удаления разметочных листьев и, возможно, замены лишь небольшого числа поддеревьев небольшой высоты на другие, «непохожие» поддеревья. Соответствие между каждой парой таких поддеревьев задается *неявно*, путем указания *функции элементной трансляции*. Приведем примеры.

Пример 1. Номер омонима в словаре часто обозначается римским числом, представленным латинскими буквами, а в базе данных – арабской цифрой. Функция элементной трансляции преобразует строку, представляющую римское число, в одну цифру.

Пример 2. Пусть некоторая словоформа входного текста транслируется в запись, содержащую морфологическую и грамматическую информацию относительно этой словоформы. Функцией элементной трансляции служит процедура морфологического анализа.

Пример 3. Пусть некоторое предложение входного текста транслируется в синтаксическое дерево ЕЯ. Функцией элементной трансляции служит процедура синтаксического анализа текста ЕЯ.

Будем говорить, что между входными и выходными деревьями (и соответственно, грамматиками) имеется *простое соответствие*, если нет «непохожих» пар поддеревьев, соответствующих друг другу при трансляции; в противном случае имеется *усложненное соответствие*.

Анализ показал, что случай простого соответствия описывается *простой* схемой трансляции с дополнительным ограничением, указанным выше в пункте (3) списка особенностей (поэтому мы и ввели термин «простое соответствие». Наш демонстрационный пример описывается простой схемой трансляции. Для таких схем в разделе 4 была предложена *специальная нотация*, значительно более наглядная, чем используемая в схемах трансляции общего вида. Основная причина – введенная нами *подразумеваемая интерпретация* выходной грамматики как исчисления деревьев, которые отождествляются с деревьями содержательной структуры; последние представляются документами XML, служащими выходом транслятора. Поэтому *нет семантических правил*. Теоретическая нотация для СУ-схем также менее удобна, чем предлагаемая, из-за своей избыточности.

Дадим краткое содержательное определение описанного класса: трансляция текстового документа в его древовидную содержательную структуру при условии «близости» последней с поверхностной структурой документа; текст может быть обогащенным (RTF, DOC и т.п.), а входная грамматика неоднозначной, что приводит к диалоговому режиму трансляции. Назовем данный класс *TDC-трансляциями* (аббревиатура по первым буквам слов Text, Document, Content). Это не множество в математическом смысле, а *открытая совокупность*, поскольку понятие близости между грамматиками не задано формально.

6. Определение обратной схемы трансляции

С целью удобства задания трансляций класса FCS, введем аппарат *обратных схем трансляции*, который можно рассматривать как *удобную нотацию* для простых и «почти простых» схем трансляции, а также СУ-схем общего вида (с перестановками нетерминалов). Как известно, удобная нотация часто приводит к качественно новым результатам, в чем мы убедимся применительно к данному случаю. Отметим, что объем данной работы не позволил дать полностью формальное определение, поэтому демонстрационный пример на рис. 6 следует считать неотъемлемой частью определения.

Обратной схемой трансляции назовем пару:

$$S_{REV} = (G_2, E_2),$$

где G_2 – выходная ФГ; E_2 – множество элементов трансляции совместно с соответствием между ними и элементами регулярных выражений в правых частях правил грамматики. Синтаксически соответствие оформляем с помощью двоеточия в виде пар $A : e$.

Каждый элемент трансляции $e \in E$ является строкой *атрибутов представления*, записываемых в произвольном порядке, причем все они необязательны:

$$e ::= [\underline{beg} = b]\,[\underline{end} = e]\,[\underline{sep} = s]$$
$$[\underline{font} = f]\,[\underline{num} = n]\,[\underline{tran} = t]$$

Уточненный синтаксис: элемент трансляции, содержащий несколько атрибутов, берется в скобки; атрибуты разделяются запятыми; пустой элемент трансляции опускается. Пара «атрибут=значение» может быть взята в квадратные скобки, если «значение» является ε-порождающим регулярным выражением.

Значениями атрибутов *beg*, *end*, *sep* являются выражения вида:

$$r\,[:f]\,[:d]\,['|'\,r\,[:f]\,[:d]]^*,$$

где r – регулярное выражение, задающее вариант лексического значения разметочного токена; f – вектор атрибутов шрифтового оформления; d – описание диалога в форме *Dialog(j,k)*, где j – номер диалога, k – номер пункта диалога.

Значение атрибута *num* – натуральное число; атрибута *tran* – имя функции, атрибута *font* – вектор атрибутов шрифтового оформления.

Содержательное описание.

6.1 Каждый из атрибутов *beg*, *end*, *sep* задает один или несколько альтернативных разметочных токена, для каждого из которых может быть указано своё шрифтовое оформление и уточняющий диалог. При этом атрибут *beg* задает разметочные токены, стоящие *перед* нетерминалом B в соответствующем правиле входной грамматики, атрибут *end* – *после*, а атрибут *sep*, допускаемый только для итераций – *между* повторяющимися нетерминалами. То есть схемному правилу вида:

$$A \rightarrow \ldots B^+ : (\underline{beg} = b,\ \underline{end} = e,\ \underline{sep} = s) \ldots$$

соответствует во входной грамматике правило следующего вида (или ему эквивалентного):

$$A \rightarrow \ldots b\,B\,(s\,B)^*\,e \ldots$$

6.2 Атрибут *font* задает вектор атрибутов шрифтового оформления. Если B нетерминал, то шрифтовые атрибуты *наследуются* для всех подчиненных B вхождений грамматических символов.

6.3 Атрибут *num* задает порядковый номер нетерминала в соответствующей продукции входной грамматики.

Несложно доказать, что если в обратной схеме трансляции присутствуют только описанные выше атрибуты, причем их значениями являются цепоч-

ки токенов (а не произвольные регулярные выражения), она является представлением некоторой СУ-схемы; если отсутствует атрибут *num* – то представлением простой СУ-схемы; соответствующую обратную схему естественно также назвать *простой*.

6.4 Атрибут *tran* задает имя *элементной функции трансляции*, сигнатура которой:

$$t: L(G_1 \mid A) \to T^{XML}(G_2 \mid A).$$

Функция t частичная, обладает побочным эффектом в виде сообщения о синтаксической ошибке. Содержательно, эта функция задает трансляцию множества фраз, определяемых нетерминалом A в G_1, в множество поддеревьев, определяемых этим же нетерминалом в G_2. Элементная трансляция для нетерминалов с атрибутом *tran* задается *процедурным* образом, а для нетерминалов без этого атрибута – *декларативным* образом, что является отличительной чертой СУ-схем по сравнению со схемами общего вида. Проекции грамматик $G_1 \mid A$ и $G_2 \mid A$ имеют правила, несопоставимые в рамках СУ-трансляций и, в нашей терминологии, «далеки» друг от друга (см. раздел 5 настоящей статьи).

Формально: при наличии атрибута *tran* обратная схема трансляции является представлением такой прямой схемы трансляции общего вида, в которой правилу:

$$A \to \ldots B : (num = n,\ tran = t) \ldots$$

соответствует, в нотации *Jacc*, правило:

$$A \to \ldots B \{\$\$ = \ldots + t(\$n) + \ldots\} \ldots$$

Пояснение. Синтезируемый строковый атрибут для нетерминала A является конкатенацией, содержащей одним из аргументов вызов функции t для вхождения нетерминала номер n этого правила, т.е. B. Функция t возвращает поддерево, порождаемое данным нетерминалом, в виде элемента XML.

На рис. 6 изображена обратная схема трансляции для демонстрационного примера.

7. XML-схемы трансляции

Схемами XML можно представлять как прямые, так и обратные текстовые схемы трансляции, но наиболее удобны они для представления обратных схем. Ниже приводятся в качестве примера XML-схемы трансляции для демонстрационного фрагмента словаря, а также некоторые специфические особенности организации таких схем. Заметим, что ввиду чрезвычайной мощности языка схем XML, возможны различные варианты.

Специфические особенности выбранного нами способа представления *обратной* схемы трансляции схемой XML следующие (рис. 9):

– каждому содержательному токену соответствует сложный тип с именем *t<имя токена>*;

– все атрибуты представления являются атрибутами (а не элементами) схемы;

– каждое употребление (возможно, итеративное) содержательного токена имеет тип, полученный путем расширения указанного выше типа. В расширении задаются атрибуты разметки. Во избежание коллизии имен, таких атрибутов в определении типа и в его расширении, используется различный регистр первой буквы (в примере *sep* и *Sep*). Значения атрибутов разметки представляются фасетом *enumeration* для строкового типа, каждому элементу которого соответствует отдельный компонент значения атрибута;

– описание диалога представляет собой отдельный элемент с именем *Dialog<номер>*.

В *прямой* XML-схеме трансляции, рис. 10, разделение синтаксического и лексического уровней осуществляется с помощью *расширяемых типов*: FToken – «форматирующий токен» (простой тип) и CToken – содержательный токен (сложный тип). Все элементы, расширяющие эти типы, принадлежат лексическому уровню. Диалог в этой схеме не реализован.

Анализ показал, что представление агрибутов на диаграммах схем XML весьма громоздко. Предлагаем альтернативный, хотя и неформальный метод их представления с помощью выносок, вручную пристыкованных к блокам диаграммы схемы XML. На рис. 8 изображена получаемая таким образом *частично формализованная обратная XML-схема трансляции* для демонстрационного примера. Предлагается использовать схемы такого типа в качестве удобного средства проектирования, «графового псевдокода» для обратных схем трансляции. Кроме того, предлагается использовать их в качестве средства *двухаспектного* описания (уровни презентации и содержания) *структуры словарной статьи* [13, 19].

8. Принципы реализации трансляторов, управляемых XML-схемами трансляции

Интерпретационный вариант. Заключается в разработке программы-интерпретатора XML-схемы трансляции, осуществляющей трансляцию в режиме динамического связывания со схемой. Перспективный, но трудоемкий в реализации вариант.

Компиляционный вариант. Разрабатывается программа метатранслятора, преобразующего XML-схему трансляции в обычную схему трансляции для некоторого существующего генератора трансляторов. Данный вариант проще в реализации, но более громоздок.

9. Основные результаты и выводы

Проанализированы две актуальные задачи из компьютерной лингвистики и из технологии XML, сводящиеся к разработке трансляторов. Сформу-

```
Правила синтаксического уровня

Dic       →  Entry*:(sep=EOL, [end=EOL])
Entry     →  Term*:sep="," ( RefFull | Tran*:sep="," [RefSyn] )
RefFull   →  Term:beg=NoteFull
RefSyn    →  Term:beg=("." NoteSyn)

Правила лексического уровня

Term      →  EngWord*:sep="-"
Tran      →  RusWord*:( sep=("," | " - "), font="bold", [beg="'"], [end="'"] )
EOL       →  #$D#$A
NoteFull  →  "см.":font="italic"
NoteSyn   →  "См.":font="italic"
EngWord   →  EngChar+["-"EngChar+]
RusWord   →  RusChar+["-"RusChar+]

Описание диалогов

Dialog1   →  Title: "Укажите смысл запятой:"
             Item1: ("Разделитель элементов словарной статьи", ",||" )
             Item2: ("Знак препинания внутри одного элемента", ",|" )
```

Рис. 7. Обратная схема трансляции

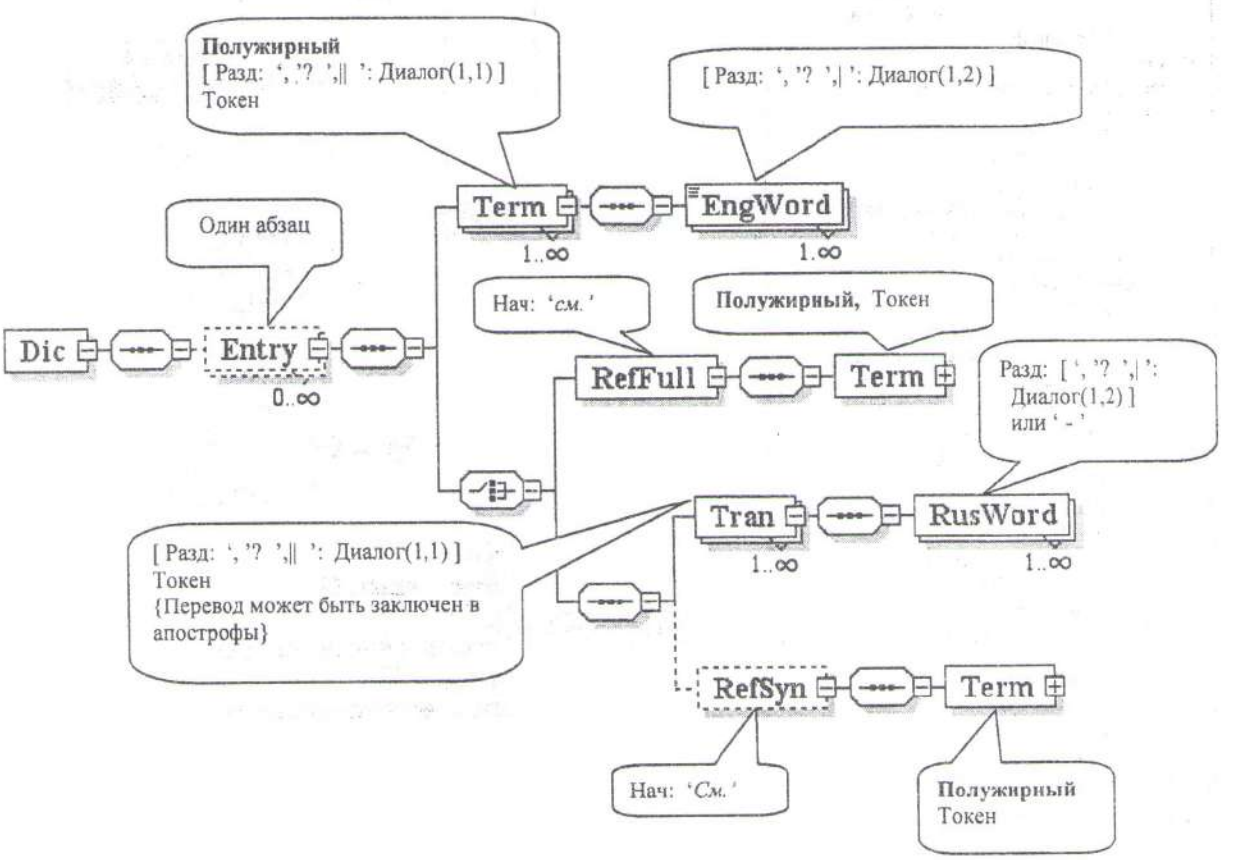
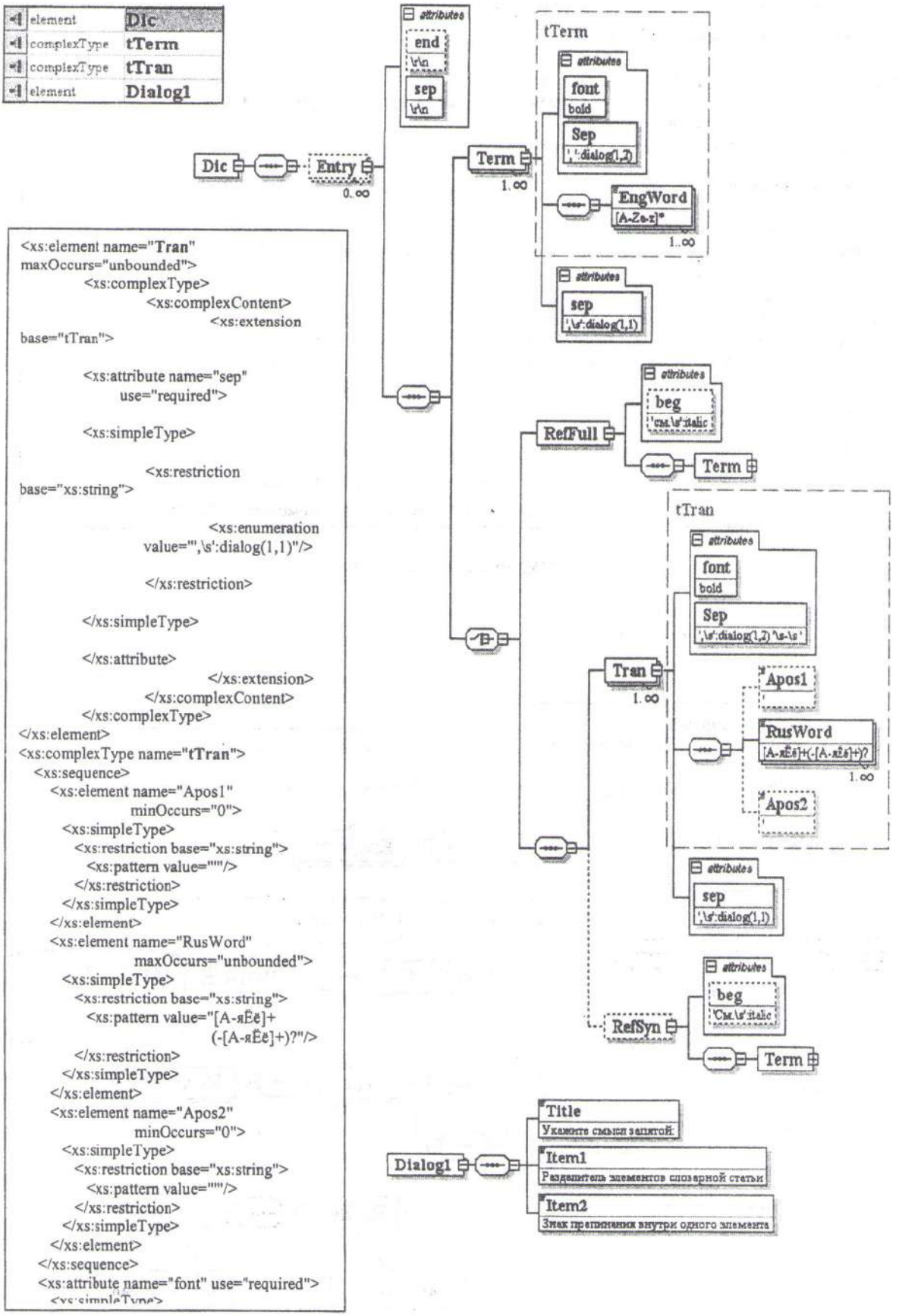
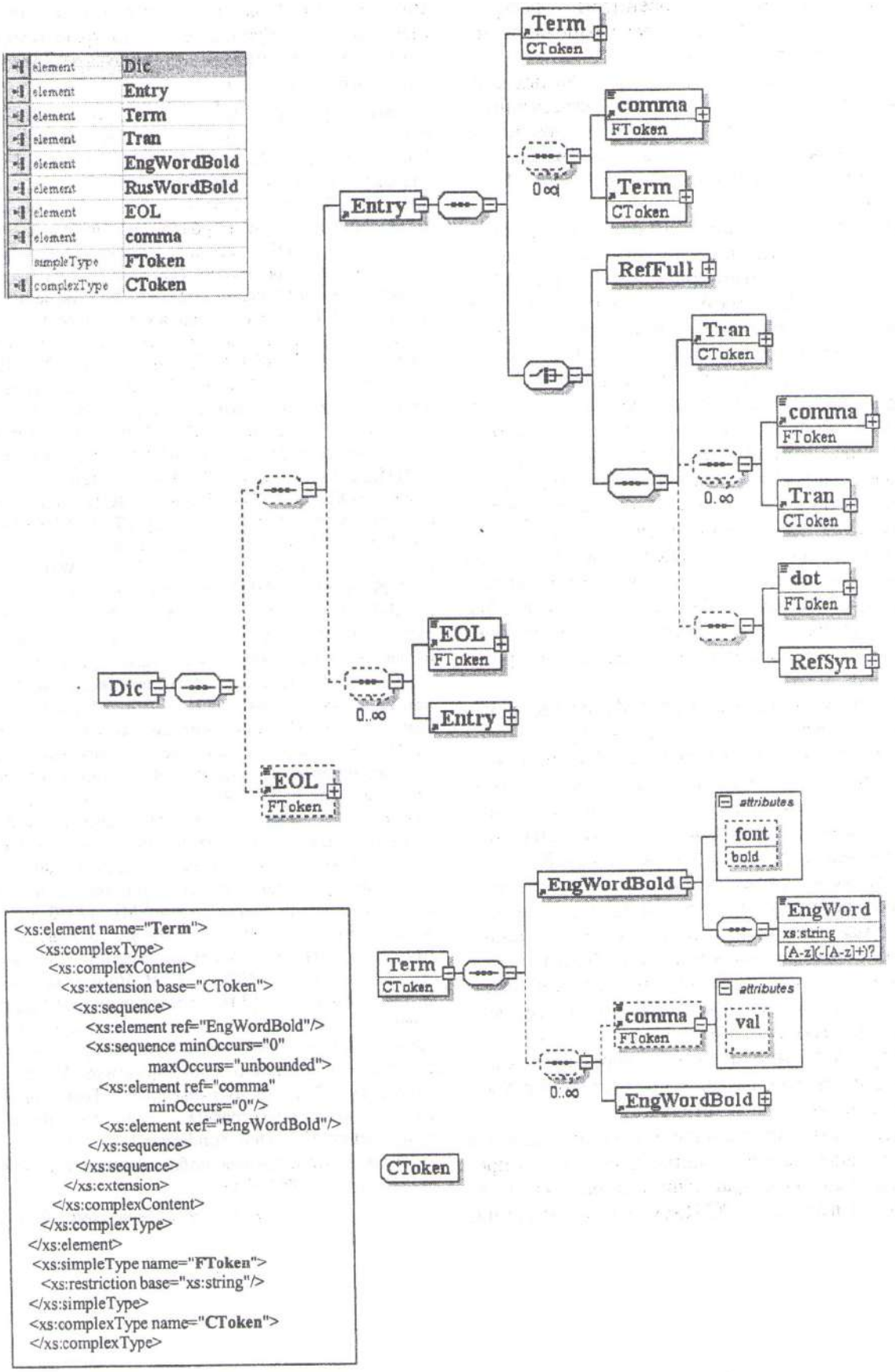


Рис. 8. Частично формализованная обратная XML-схема трансляции для демонстрационного словаря, она же – описание структуры словарной статьи




```
<xs:element name="Tran"
maxOccurs="unbounded">
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension
base="tTran">

        <xs:attribute name="sep"
            use="required">

        <xs:simpleType>

                <xs:restriction
base="xs:string">

                        <xs:enumeration
                value="',\s':dialog(1,1)"/>

                </xs:restriction>

        </xs:simpleType>

        </xs:attribute>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
<xs:complexType name="tTran">
  <xs:sequence>
    <xs:element name="Apos1"
                minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="'"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="RusWord"
                maxOccurs="unbounded">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="[А-яЁё]+
                        (-[А-яЁё]+)?"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Apos2"
                minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="'"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="font" use="required">
    <xs:simpleType>
```

Рис. 9. Обратная XML-схема трансляции
(список глобальных элементов, фрагмент текста, основная диаграмма и диаграмма диалога)




```
<xs:element name="Term">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="CToken">
        <xs:sequence>
          <xs:element ref="EngWordBold"/>
          <xs:sequence minOccurs="0"
                maxOccurs="unbounded">
            <xs:element ref="comma"
                minOccurs="0"/>
            <xs:element ref="EngWordBold"/>
          </xs:sequence>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:simpleType name="FToken">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:complexType name="CToken">
</xs:complexType>
```

Рис. 10. Прямая XML-схема трансляции
(список глобальных элементов, фрагмент текста, диаграмма синтаксического уровня, диаграмма токена «термин» и диаграмма расширяемого типа «содержательный токен»)

лированы отличительные особенности этих задач и определен соответствующий класс трансляций, названный TDC-трансляциями.

Выявлены недостатки традиционных схем трансляции применительно к этому классу и предложены принципы их преодоления, в частности, предложен формализм обратных схем трансляции, ориентированный именно на описание TDC-трансляций.

Основным результатом работы является предложение представлять схемы трансляции с помощью особым образом интерпретируемых схем XML, названных XML-схемами трансляции. Эти схемы оказались особенно удобными для представления именно обратных схем трансляции. Использование XML-схем трансляции позволит значительно ускорить процесс разработки трансляторов за счет перехода к их визуальному непроцедурному проектированию с помощью редакторов XML. В области компьютерной лексикографии это означает значительное ускорение создания лексикографических БД на основе традиционных словарей; в технологии XML – ускорение процесса преобразования документов специфических форматов в формат XML. Предложено использовать технологию XML также для облегчения программной реализации трансляторов, в частности, для реализации деревьев разбора.

Предложено использовать XML-схемы трансляции в качестве средства описания структуры словарной статьи – одного из основных объектов лексикографии. Описание получается *двухаспектным* – описываются поверхностная и содержательная структуры в их взаимосвязи; известные нам способы описания являются *одноаспектными*. Консультации с лексикографами убедили, что получен весьма удобный на практике метод наглядного описания структуры словарной статьи, особенно вариант частично-формализованных схем. Данный метод можно рассматривать и вне связи с процессом трансляции, а в общем контексте проектирования лексикографических систем [12].

Результаты работы представляют вклад одновременно в теорию трансляции, технологию XML и компьютерную лингвистику.

Перспективами дальнейших исследований в данном направлении являются: формальное определение класса TDC-трансляций; разработка трансляторов, управляемых XML-схемами трансляции; апробация предложенного метода на разнообразных задачах, требующих создания трансляторов, в том числе на традиционных задачах разработки языков программирования.

Список литературы: 1. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 2. М.: Мир, 1978. — 487 с. 2. *Ахо А., Сети Р., Ульман Дж.* Компиляторы: принципы, технологии, инструменты: Пер. с англ. М.: «Вильямс», 2001. — 768 с. 3. *Костельцев А.В.* Построение интерпретаторов и компиляторов, СПб: Наука и техника, 2001. — 224 с. 4. *Мозговой М. В.* Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. СПб.: Наука и техника, 2005. – 320 с. 5. XML Schema Part 1: Structures Second Edition W3C Recommendation 28 October 2004 http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. 6. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. 2-е изд. М., «Вильямс». — 528 с. 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. 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. 10. XMLSpy 2005 Home Edition. http://www.altova/com. 11. *Широков В.А.* Інформаційна теорія лексикографічних систем. К.: Довіра, 1998. — 331 с. 12. *Широков В.А.* Феноменологія лексикографічних систем. К.: Наукова думка, 2004. — 319 с. 13. *Буслик Н.Н., Тимофеев* С.П. Об одной модели структуры словарной статьи и ее применении к задаче преобразования традиционного словаря в документ XML // Проблемы бионики. – Харьков. № 59. 2003. – С. 8–18. 14. *Буслик Н.Н., Тимофеев С.П.* Диалоговые двухуровневые формальные грамматики и их использование при создании лексических баз данных // Проблемы бионики, 2002, №57. 15. *Буслик Н.Н., Тимофеев С.П.* Преобразование традиционных словарей в документы XML. // Сб. научных трудов ХНУРЭ. – 2004. – С. 86–87. 16. *Гладкий Н.В., Мельчук И.А.* Элементы математической лингвистики. — М.: Наука, 1969. — 192 с. 17. *M. Murata, D. Lee, and M. Mani.* Taxonomy of XML Schema Languages using Formal Language Theory. In Extreme Markup Languages, Montreal, Canada, 2001. 18. *Борковский А.Б.* Англо-русский словарь по программированию и информатике. М.: Русский язык, 1987. — 333 с. 19. *Дубичинский В.В.* Теоретическая и практическая лексикография. – Вена, Харьков: Wienar Slawistischer Almanach Sonderband 45, 1998. – 156 с. 20. *Грейвс М.* Проектирование баз данных на основе XML. М.: «Вильямс», 2002. — 640 с.

Поступила в редколлегию 20.10.2005