# Язык запросов к совокупности XML-документов, соединенных при помощи ссылок языка XLink

*Д.А. Лизоркин*
*(МГУ, ИСП РАН, lizorkin@ispras.ru)*

**Аннотация.** Язык ссылок XML (XLink) — это язык описания межресурсных связей при помощи атрибутов XML и специального пространства имен. Спецификация языка XLink Консорциума Всемирной Сети предоставляет лишь структуры данных для описания ссылок и минимальную модель их поведения.
В данной статье предлагается язык, который позволяет приложению прозрачным образом формулировать запросы к ссылкам XLink и осуществлять переходы по определяемым этими ссылками дугам. Предлагаемый язык был назван XPathLink, поскольку разрабатывался как органичное расширение языка адресации структурных частей XML-документа XPath. Язык XPathLink инкапсулирует сложности синтаксиса XLink от прикладного приложения и предоставляет более высокий уровень абстракции при обработке совокупности XML-документов, соединенных ссылками языка XLink, по сравнению с существующими подходами.
Рассматривается реализация предлагаемого языка XPathLink функциональными методами. Благодаря интеграции полученной реализации с языком программирования общего назначения Scheme на уровне узлов обрабатываемых XML-документов и функций Scheme достигается функциональность языка запросов к XML-документам, связанным ссылками XLink.

## *1. Введение*

Современные практические приложения часто характеризуются большим многообразием хранимых и обрабатываемых ресурсов, для которых возникает желание специфицировать семантику их родства в виде системы межресурсных связей. Мощным и гибким решением данной задачи является язык ссылок XML (XML Linking Language — XLink) [1], позволяющий описывать взаимосвязи между ресурсами в виде ссылок, с использованием XML и отдельного пространства имен. Язык ссылок XML начал разрабатываться Консорциумом Всемирной Сети [2] вскоре после появления самого языка XML, и первоначально предполагался как вторая часть Спецификации XML [3].

Один из основных принципов дизайна языка XLink — обеспечить возможность использования ссылок XLink приложениями в различных предметных областях [4]. Ввиду того, что использование ссылок XLink ориентировано в



первую очередь на **приложения** и лишь во вторую очередь предназначено для **представления человеку**, при разработке языка XLink изначально создавался такой синтаксис ссылок, чтобы их семантика могла быть распознана широким классом прикладного программного обеспечения.

Поскольку язык XLink ориентировался на возможность его применения в самых разных предметных областях, Спецификация XLink предоставляет лишь структуры данных для описания ссылок и минимальную модель их поведения [1]. Все действия по распознаванию элементов языка XLink в XML-документе и обработке описываемых этими элементами ссылок перекладываются на конкретное приложение, т.к. могут зависеть от используемой в данной предметной области модели обработки ссылок.

Ввиду того, что синтаксис языка XLink достаточно сложен и описания ссылок на практике зачастую получаются многословными, сложным является и разбор разметки, отвечающей элементам языка XLink, с целью извлечения семантики описанных ссылок. Возникает желание иметь в качестве надстройки над структурами данных языка XLink язык более высокого уровня, который бы предоставил приложению возможность прозрачным образом формулировать запросы к ссылкам XLink и осуществлять переходы по определяемым этими ссылками дугам, и инкапсулировал сложности синтаксиса XLink.

Язык, который бы обеспечивал приложение желаемой функциональностью при обработке XML-документов со ссылками XLink, до недавнего времени разработан не был. В частности, как будет показано в разделе 3, язык XQuery, позиционируемый Консорциумом Всемирной Сети как язык запросов, применимый к разнообразным типам источников XML-данных [5], не предоставляет возможностей по обработке имеющихся в XML-документе ссылок языка XLink в соответствии со Спецификацией XLink.

В настоящей статье предлагается язык, предоставляющий возможность формулировать запросы к ссылкам XLink и осуществлять переходы между XML-документами и их частями по дугам, определяемым внутри ссылок XLink. Разработанный язык получил название **XPathLink**, поскольку базируется на языке адресации структурных частей XML-документа XPath [6] и расширяет его поддержкой семантики языка XLink. Язык XPathLink обладает следующими свойствами:

- Он полностью соответствует дизайну XPath и добавляет 3 дополнительных оси к грамматике XPath;
- Он естественным образом соответствует Модели данных языка XPath [6] и органично расширяет каждый из имеющийся в модели тип узлов возможностью иметь одно дополнительное свойство.

Предлагаемый в статье язык XPathLink был полностью реализован с помощью функциональных методов программирования, и предоставляет мощный и выразительный инструмент для обработки XML-документов, связанных ссылками XLink.

Дальнейшее содержание статьи организовано следующим образом. В разделе 2 дается краткий обзор языка XLink и предоставляемых им возможностей по описанию связей между ресурсами. Раздел 3 посвящен рассмотрению родственных работ в области обработки документов, содержащих ссылки языка XLink. В разделе 4 обсуждается язык адресации структурных частей XML-документа XPath и показывается, как функциональность XPath может быть естественным образом расширена переходами по дугам языка XLink. В разделе 5 предлагаются средства формулирования запросов к дуге языка XLink как к самостоятельной сущности, что достигается как дальнейшее расширение XPath. Детали реализации предлагаемого в данной статье языка запросов, комбинирующего возможности XPath и XLink, рассматриваются в разделе 6. Ограничения, присущие предлагаемому языку запросов, обсуждаются в разделе 7. Раздел 8 завершает статью.

## 2. Обзор языка XLink

Язык ссылок XML (XML Linking Language, XLink) — это язык описания межресурсных связей с помощью XML и отдельного пространства имен.

На дизайн языка XLink в значительной степени повлияли следующие стандарты [1]:

- Язык разметки гипертекстовых документов HTML, определяющий несколько типов элементов, которые представляют ссылки. Наиболее известным инструментом для определения межресурсных связей при создании гипертекстовых документов являются гиперссылки, задаваемые при помощи элемента `A` языка HTML, где под гиперссылкой понимается такой вид ссылки, основным назначением которой является представление человеку [1].
- Язык описания межресурсных связей HyTime, обладающий более богатыми выразительными возможностями, нежели HTML, и позволяющий определять входящие и сторонние ссылки, а также описывать некоторые их семантические свойства.

Язык XLink обеспечивает полную функциональность гиперссылок HTML, и гораздо большее [7]: он позволяет устанавливать отношение связи между более чем двумя ресурсами, ассоциировать различные метаданные со ссылками, соединять ресурсы без их модификации [4].

Хотя ссылки XLink описываются на XML, с их помощью можно соединять не только XML-документы, но и другие виды ресурсов. Понятие ресурса определяется в IETF RFC 2396 как любая адресуемая единица информации или сервиса [8]. Если ресурс представляет собой правильно сформированный (well-formed) XML-документ, то спецификация XLink считает ресурсом также любую часть этого документа, определяемую идентификатором фрагмента на языке указателей XML (XML Pointer Language — XPointer) [9]. Идентификатор фрагмента языка XPointer может дополнять унифицированный идентификатор (URI) XML-документа.



Относительно конкретной ссылки Спецификация XLink подразделяет все ресурсы на локальные и удаленные. В терминах XLink, **локальный ресурс** — это элемент XML, который участвует в ссылке за счет того, что ссылочный элемент является для него родительским [10]. Ресурс, который участвует в ссылке благодаря тому, что к нему адресуются с помощью унифицированного идентификатора URI, считается **удаленным** (remote), даже если он располагается в том же XML-документе, что и ссылка, или даже внутри ссылочного элемента. Заметим, что один и тот же ресурс может быть локальным для одной ссылки XLink и удаленным — для другой.

Язык XLink вводит два типа ссылок.

1. **Простая ссылка (simple link)** — это ссылка, которая ассоциирует в точности два ресурса — один локальный и один удаленный — и определяет семантику перехода от первого ко второму. Предоставляемая простой ссылкой функциональность по связыванию ресурсов является наиболее распространенной (например, в эту же категорию попадают ссылки `A` и `IMG` языка HTML). Синтаксис простых ссылок ориентирован на краткость записи, и поэтому у простых ссылок нет какой-либо специальной внутренней структуры.
   Использование простой ссылки иллюстрируется примером, показанном на рис. 1, который будет использоваться в ходе дальнейшего обсуждения в данной статье. Рисунок выражает систему заказа товаров некоторого электронного магазина. Будем считать, что электронный магазин оперирует такими ресурсами, как каталог товаров, информация о клиентах и сделанные клиентами заказы товаров из каталога. Поскольку обозначенные 3 ресурса имеют разнородную структуру, разумно хранить их в виде 3 отдельных XML-документов, каждый из которых показан на рис. 1. Простые ссылки XLink позволяют установить связь каждого конкретного заказа с элементами из каталога, которые были заказаны, и с клиентом, который сделал заказ.

```
<?xml version='1.0'?>
<catalogue>
 <printer>
  <lot>001</lot>
  <descr>Ink jet</descr>
  <price>450</price>
 </printer>
 <keyboard>
  <lot>002</lot>
  <price>20</price>
 </keyboard>
 <display>
  <lot>003</lot>
  <descr>Color,
Digital</descr>
```

```
<?xml version='1.0'?>
<!DOCTYPE clients [
 <!— Атрибут person-id имеет тип ID —>
 <!ATTLIST person
    person-id   ID  #REQUIRED>
 <!— Остальные описания схемы опущены —>
]>
<clients>
 <person person-id="per1">
  <name>John Smith</name>
  <email>johnsmith@company.com</email>
  <VIP/>
 </person>
 <person person-id="per2">
  <name>Paul Brown</name>
```

```
   <warranty>2                    <email>paul@brown.net</email>
years</warranty>                 </person>
   <price>500</price>           </clients>
  </display>
</catalogue>

<?xml version='1.0'?>
<purchase-orders xmlns:xlink="http://www.w3.org/1999/xlink">
  <order>
    <entry>
      <item xlink:type="simple"
          xlink:href="catalogue.xml#xpointer(//printer[lot=001])"/>
      <quantity>2</quantity>
    </entry>
    <entry>
      <item xlink:type="simple"
          xlink:href="catalogue.xml#xpointer(//display[lot=003])"/>
      <quantity>2</quantity>
    </entry>
    <customer xlink:type="simple"
           xlink:href="clients.xml#per1"/>
  </order>
  <order>
    <entry>
      <item xlink:type="simple"
          xlink:href="catalogue.xml#xpointer(//keyboard[lot=002])"/>
      <quantity>1</quantity>
    </entry>
    <customer xlink:type="simple"
           xlink:href="clients.xml#per2"/>
  </order>
</purchase-orders>
```

*Рис. 1. Набор связанных с помощью XLink XML-документов, выражающих систему заказа товаров. Документ "catalogue.xml" (слева вверху) содержит каталог товаров; документ "clients.xml" (справа вверху) — данные о клиентах; документ "purchase-orders.xml" (внизу) — сделанные клиентами заказы из каталога.*

Элемент языка XML, являющийся ссылкой XLink, может носить произвольное имя, а принадлежность элемента к языку XLink и ссылочный тип элемента устанавливаются глобальным атрибутом `xlink:type`, где префикс `xlink` связывается с пространством имен, зарезервированным языком XLink. По аналогии с гиперссылками HTML, удаленный ресурс, на который указывает простая ссылка XLink, определяется атрибутом `xlink:href`. Значением атрибута `xlink:href` служит Унифицированный Идентификатор URI, возможно, включающий в себя идентификатор фрагмента на языке XPointer — для целевых ресурсов, представляющих собой фрагмент XML-документа.



В документе `"purchase-orders.xml"` на рис. 1 простыми ссылками XLink являются элементы языка XML с именами `item` и `customer`; первые указывают на элементы каталога, вторые — на клиентов. Идентификация конкретного клиента осуществляется с использованием имеющихся в XML-документе `"clients.xml"` атрибутов типа `ID`, и при данном способе идентификации может использоваться сокращенный синтаксис XPointer, напоминающий обращение к именованному якорю в HTML. Для идентификации элемента в каталоге (XML-документ `"catalogue.xml"`) используется полный синтаксис XPointer. Из рассматриваемого примера легко видеть, что по сравнению с гиперссылками HTML простые ссылки XLink обладают более гибкими и мощными возможностями по связыванию ресурсов и описанию семантики этих связей.

2. **Расширенная ссылка (extended link)** — это ссылка, которая выражает полную функциональность языка XLink. Расширенная ссылка может объединять произвольное количество участвующих в ней ресурсов, и участвующие ресурсы могут быть любой комбинацией локальных и удаленных.
   Использование или следование по ссылке с какой-либо целью называется **переходом** (traversal). Несмотря на то, что расширенные ссылки могут соединять произвольное количество ресурсов, переход всегда включает в себя ровно 2 ресурса [1]. Информация о том, как осуществлять переход между парой ресурсов, включающая в себя направление перехода и его семантику, задается при помощи **дуги** (arc).
   Ввиду того, что расширенная ссылка предоставляет мощные возможности по связыванию ресурсов, структура расширенной ссылки может быть достаточно сложной. В общем случае она включает в себя элементы типа **локатор** [2], которые указывают на удаленные ресурсы; элементы типа **ресурс**, содержащие локальные ресурсы; элементы типа **дуга**, которые определяют дуги и условия перехода по ним.
   Дуга в языке XLink является ориентированной, и 2 ресурса, соединяемые дугой, называются соответственно **исходным** (starting resource) и **целевым** (ending resource) [2]. Дугу называют **входящей** (inbound), если ее исходный ресурс является удаленным ресурсом, а целевой ресурс — локальным. Дуга называется **сторонней** (third-party), если ни один из соединяемых ею ресурсов не является локальным. Благодаря наличию в языке XLink входящих и сторонних дуг обеспечивается возможность соединять ресурсы без их модификации.
   Обычно элементы типа “расширенная ссылка” располагаются отдельно от тех ресурсов, которые они соединяют (например, в совершенно разных документах). Расширенные ссылки важны для ситуаций, когда соединяемые ресурсы доступны только для чтения; или когда модификация этих ресурсов является дорогостоящей и сложной операцией, тогда как модификация отдельно располагающейся ссылки

достаточно проста; или когда ресурсы имеют форматы, не поддерживающие встроенные ссылки (как для многих мультимедийных форматов).

Пример расширенной ссылки будет рассмотрен в разделе 5.

Хотя простые ссылки концептуально являются подмножеством расширенных ссылок, они синтаксически различны. В частности, простая ссылка определяет дугу неявным образом, и поэтому для преобразования простой ссылки в расширенную ссылку требуется осуществить несколько структурных преобразований.

Можно говорить о том, что предназначением простой ссылки является удобная короткая форма записи для эквивалентного случая расширенной ссылки [1]. Один элемент вида “простая ссылка” объединяет в себе базовую функциональность элементов типа “расширенная ссылка”, “локатор”, “дуга” и “ресурс”. В том случае, когда реально требуется лишь подмножество свойств этих элементов, простая ссылка удобна как альтернатива расширенной ссылке.

## 3. Родственные работы по предметной области

В данном разделе рассматриваются работы, в которых делается попытка обеспечить язык запросов к совокупности XML-документов, связанных ссылками языка XLink, или предоставляются высокоуровневые возможности переходов по дугам XLink.

### 3.1. XQuery

В качестве языка запросов, применимого к источникам XML-данных разнообразных типов, Консорциум Всемирной Сети [2] позиционирует язык Xquery [5]. Хотя на момент написания настоящей статьи язык XQuery еще не был переведен из чернового статуса в статус официальной Рекомендации Консорциума, уже можно с уверенностью говорить о том, что в своей текущей версии XQuery не предоставляет средств по обработке ссылок языка XLink в соответ-ствии со Спецификацией XLink, что подтверждается следующими соображениями:

- Описания ссылок языка XLink строятся на сложном и многословном синтаксисе, базирующемся на пространстве имен и иерархической взаимосвязи элементов языка XLink. При написании на XQuery запросов к XML-документам, содержащим элементы языка XLink, вся работа по распознаванию ссылок XLink и интерпретации их семантики целиком ложится на разработчика запросов. Необходимо также заметить, что получение информации о дугах, описанных в единственной расширенной ссылке языка XLink, требует написания на XQuery нескольких операций естественного соединения (join), выполнение которых дает значительную нагрузку на вычислитель языка XQuery.
- При соединении с помощью ссылки XLink удаленного ресурса, представляющего собой фрагмент некоторого XML-документа, идентификация данного фрагмента внутри документа осуществляется



с помощью языка XPointer, и идентификатор фрагмента включается в значение одного из глобальных атрибутов языка XLink. При обработке ссылок XLink с помощью XQuery для определения ресурсов, соединяемых с помощью ссылки, необходимо динамически вычислить идентификатор фрагмента на языке XPointer, полученный в качестве значения атрибута. В Библиотеке базовых функций XQuery [11] не существует функции, позволяющей интерпретировать идентификатор фрагмента, записанный на языке XPointer. Автору также не известна ни одна реализация XQuery, которая бы предоставляла интерпретатор языка XPointer внутри вычислителя запросов.

Предлагаемый в настоящей статье язык запросов к совокупности XML-документов, связанных ссылками XLink, базируется на подмножестве XQuery — языке XPath, назначением которого является адресация структурных частей XML-документа. В настоящей статье предлагается расширение XPath высокоуровневой поддержкой XLink, функциональность языка запросов достигается благодаря возможности тесной интеграции XPath с функциональным языком программирования общего назначения Scheme.

### 3.2. Браузеры с поддержкой XLink

Под поддержкой языка XLink браузером будем понимать способность браузера распознавать элементы языка XLink в XML-документе и обеспечивать пользовательский интерфейс для переходов между ресурсами по дугам XLink. Среди браузеров, обеспечивающих поддержку языка XLink, следует упомянуть такие, как Amaya, Mozilla и DocZilla. Первые два реализуют поддержку языка XLink лишь в масштабе простых ссылок XLink, браузер DocZilla обеспечивает также навигацию по расширенным ссылкам.

Наличие лишь одного известного автору браузера, поддерживающего ссылки языка XLink в полном объеме (DocZilla), мотивируется сформулированными Консорциумом Всемирной Сети требованиями к языку XLink [4]. В то время как простые ссылки XLink явным образом ориентированы на внешнее представление, презентационный аспект расширенных ссылок XLink многими аналитиками ставится под сомнение [12].

Необходимо заметить, что браузеры с поддержкой языка XLink по своей природе нацелены на наглядное представление документов со ссылками XLink человеку. Поскольку браузеры являются прикладным программным продуктом, при их разработке не ставятся задачи обеспечения других приложений средствами высокоуровневой обработки XML-документов со ссылками XLink. В настоящей статье решается системная задача, и язык запросов к совокупности XML-документов, связанных ссылками XLink, разрабатывается из расчета на то, чтобы быть использованным другими приложениями.

### 3.3. Интерфейсы прикладного программирования

Интерфейсы прикладного программирования для обработки XML-документов, соединенных ссылками языка XLink, предоставляют системы XLip, X2X и ExtremeKnit. Все 3 обозначенные системы предоставляют интерфейсы прикладного программирования на языке объектно-ориентированного программирования Java.

Ввиду использования объектно-ориентированного языка программирования для обработки XML-документов, все 3 системы — XLip, X2X и ExtremeKnit — страдают от проблемы несоответствия импеданса [13]. Так, в интерфейсах прикладного программирования, предоставляемых каждой из рассматриваемых 3 систем, для представления и обработки XML-документов со ссылками XLink разработана своя собственная система классов, и каждая из этих систем классов достаточно далека от древовидной структуры Информационного Пространства XML [14].

Помимо показанной проблемы несоответствия импеданса, все 3 интерфейса прикладного программирования являются достаточно низкоуровневыми. С различными вариациями отдельные методы языка Java используются для таких базовых операций, как загрузка отдельного XML-документа, получения набора ссылок, нахождения исходного или целевого ресурса для данной ссылки и т.п. Упомянутые методы связаны друг с другом только по их аргументам и возвращаемым результатам, что вынуждает разработчиков приложений вводить много локальных переменных и прослеживать сложные взаимосвязи между классами языка Java. Приложению приходится проводить много низкоуровневой рутинной работы, прежде чем будут получены все необходимые данные для переходов по ссылкам XLink и обработки связанных этими ссылками XML-документов.

Язык запросов, предлагаемый в настоящей статье, обеспечивает более высокий уровень абстракции над синтаксисом языка XLink по сравнению с рассмотренными интерфейсами прикладного программирования. Использование функциональных методов программирования для сделанной реализации обеспечивает интеграцию предлагаемого языка запросов с языком программирования общего назначения Scheme на уровне узлов обрабатываемых документов (как списковых структур данных языка Scheme) и функций Scheme, что позволяет избежать проблемы несоответствия импеданса [15].

## *4. Язык XPath и переходы по дугам языка XLink*

Предлагаемый в статье язык запросов к XML-документам, связанным между собой с помощью ссылок XLink, будет базироваться на языке адресации частей XML-документа XPath. В данном разделе сначала дается обзор языка XPath, затем рассматривается предлагаемое расширение функциональности XPath, обеспечивающее переходы по дугам языка XLink.



### 4.1. Обзор XPath

Назначение языка XPath — адресация структурных частей XML-документа. Ввиду того, что XML-документ является, в сущности, древовидной структурой, модель данных языка XPath [6] представляет документ как дерево узлов.

Вычисление любого выражения XPath осуществляется относительно **контекста**. Основными составляющими контекста являются:

- Узел XML-документа (называемый также **контекстным узлом**);
- **Контекстная позиция** и **контекстный размер**, которые используются при определении взаимоотношения контекстного узла с остальными узлами.

Основной конструкцией языка XPath является **путь доступа** (location path) [6]. Путь доступа применяется к контекстному узлу, и результатом вычисления является **набор узлов** (node-set) [2], состоящий из (возможно, нескольких) узлов, выбранных с помощью данного пути доступа относительно контекстного узла. Выбранные узлы соответствуют элементам, атрибутам, текстовым данным и другим частям XML-документа [16].

Путь доступа состоит из последовательности одного или более **шагов доступа** (location step), синтаксически отделяемых друг от друга символом косой черты (“/”). Шаг доступа включает в себя 3 составляющие:

- **Ось** (axis), определяющую соотношение в дереве между узлами, в контексте которых вычисляется шаг доступа, и узлами, которые выбирает шаг доступа. Ось можно считать направлением движения по дереву, представляющему XML-документ [16]. Спецификация XPath определяет 13 различных осей. Они включают в себя оси для спуска к листьям дерева, для подъема в сторону корня, для выбора соседних узлов и т.п. Синтаксически имя оси отделяется от остальной части шага адресации с помощью двойного двоеточия (“::”);
- **Тест узла** (node test), который определяет тип и, возможно, имя узлов, выбираемых шагом доступа. В то время как ось определяет направление движения, тест узла определяет желаемые узлы, которые должны быть выбраны;
- Ноль или более **предикатов** (predicates). Каждый предикат синтаксически записывается в квадратных скобках и используется для дальнейшего просеивания набора узлов, выбираемых шагом доступа.

Шаги в пути доступа вычисляются по очереди слева направо. Самый левый шаг вычисляется первым, обычно по отношению к контекстному узлу — корню дерева XML-документа. Каждый последующий шаг доступа выбирает набор узлов, который вычисляется по отношению к набору узлов, выбранному предыдущим шагом доступа. Набор узлов, выбранный самым правым шагом доступа — это результат всего пути доступа для данного XML-документа.

Пример пути доступа языка XPath приведен на рис. 2. Данный путь доступа состоит из 4 шагов доступа, во 2-м шаге доступа имеется один предикат, остальные шаги доступа предикатов не содержат. Если вернуться к рис. 1 и рассмотреть показанный на рис. 2 XML-документ `"clients.xml"`, то нетрудно

видеть, что путь доступа на рис. 2 для данного документа выбирает имя (name) человека (person), у которого имеется атрибут person-id со значением "per2". Специальный тест узла языка XPath text() используется в данном шаге доступа для адресации к текстовому узлу с целью получения имени.

```
/child::clients/child::person[attribute::person-id = "per2"]/child::name/child::text()
```

*Рис. 2. Пример пути доступа, который выбирает имя клиента, имеющего атрибут person-id со значением "per2"*

Помимо рассмотренного синтаксиса для записи путей доступа (называемого также полным синтаксисом), Спецификацией XPath определяется также **укороченный синтаксис** (abbreviated syntax) для наиболее употребительных конструкций языка. При дальнейшем изложении мы будем пользоваться двумя такими правилами укороченного синтаксиса:

- Ось child используется в шаге доступа по умолчанию, т.е. спецификатор child:: может опускаться;
- Разделитель в две идущие подряд косые черты ("//") символизирует шаг доступа /descendant-or-self::node()/, выбирающий контекстный узел и всех его узлов-потомков.

В виде сокращенного синтаксиса путь доступа, рассмотренный на рис. 2, может быть переписан так:

```
//person[attribute::person-id = 'per2']/
  name/text()
```

Мы будем пользоваться сокращенным синтаксисом для компактной записи рассматриваемых далее примеров.

## 4.2. Расширение XPath переходами по дугам языка XLink

Заметим, что термин **ось** (axis) в XPath очень близок термину **переход** (traverse) в XLink, поскольку и тот, и другой подразумевают перемещение с одного места в документе на другое. Данное наблюдение убеждает нас в том, что при построении на основе XPath языка запросов к совокупности связанных XML-документов операции перехода по дуге XLink в языке XPath должна соответствовать ось. По аналогии с англоязычным названием для перехода в XLink назовем эту ось "traverse".

На содержательном уровне, если имеются узлы A и B, такие, что для некоторой дуги XLink узел A является исходным ресурсом, а узел B – целевым, то при применении оси traverse к узлу A как к контекстному узлу результатом будет узел B.

Если следовать общему стилю, принятому в Спецификации XPath при определении осей, то более строгое определение оси traverse будет выглядеть так:



**Определение 1.** Ось traverse содержит все узлы, являющиеся целевыми ресурсами для всех дуг XLink, для которых контекстный узел служит исходным ресурсом.

Из определения следует, что ось traverse возвращает непустой набор узлов только тогда, когда контекстный узел является исходным ресурсом хотя бы для одной дуги языка XLink.

Необходимо отметить, что осью traverse могут быть выбраны узлы, находящиеся в другом XML-документе, нежели контекстный узел; т.к. дуги языка XLink могут соединять ресурсы, находящиеся в разных XML-документах. Также в результате применения к контекстному узлу оси traverse может быть получен набор узлов из нескольких разных XML-документов, поскольку контекстный узел может быть исходным ресурсом для нескольких дуг XLink, и целевые ресурсы этих дуг могут располагаться в нескольких разных XML-документах.

Если целевой ресурс является удаленным ресурсом в терминах XLink и представляет собой целиком XML-документ (т.е. при адресации к данному ресурсу не используется идентификатор фрагмента на языке XPointer), то осью traverse будет выбран элемент документа. Если при адресации к удаленному целевому ресурсу Унифицированный Идентификатор Ресурса (URI) используется совместно с идентификатором фрагмента на языке XPointer и данный идентификатор фрагмента вычисляется в некоторый набор узлов, то осью traverse будет выбран весь этот набор узлов.

Выражение на языке XPath, расширенном предлагаемой осью traverse, позволяет прикладному приложению оперировать не с одним деревом XML-документа, а уже с деревьями нескольких документов, соединенных между собой с помощью ссылок языка XLink. При этом оси, определяемые Спецификацией XPath, замкнуты внутри отдельного дерева документа, а ось traverse позволяет осуществить переход между деревьями.

Необходимо отметить, что хотя совокупность XML-документов, соединенных ссылками XLink, представляет собой граф, вычисление любого выражения XPath, расширенного предлагаемой осью traverse, всегда будет конечным по времени. Данное свойство предлагаемого расширения языка XPath объясняется тем, что все оси, являющиеся транзитивным замыканием других осей, не могут зациклиться, поскольку замкнуты внутри конкретного дерева XML-документа, где циклы отсутствуют по определению дерева. Вычисление оси traverse также всегда конечно, поскольку для любого набора узлов существует лишь конечное число дуг XLink, исходными ресурсами которых являются узлы из данного набора.

Предложенная ось органичным образом вписывается в язык XPath, и при использовании данной оси в шаге доступа XPath полностью сохраняется семантика остальных составляющих шага доступа — теста узла и предикатов. Ввиду того, что по оси traverse можно перейти на узлы произвольного типа,

тест узла помогает конкретизировать тип и, возможно, имя узлов, выбираемых шагом доступа.

Предикаты могут быть использованы для дальнейшего просеивания получаемого набора узлов в соответствии с некоторыми более сложными условиями. Как и для других осей, определенных спецификацией XPath, при применении предикатов к результату оси `traverse` контекстный размер равен количеству узлов в наборе, подлежащему фильтрации. Что касается контекстной позиции каждого узла, то для неупорядоченных наборов узлов — таких как атрибуты и объявления пространств имен — в Спецификации XPath сопоставление каждого узла с контекстной позицией объявляется зависящим от реализации [6]. Аналогичный подход может быть применен и для набора узлов, получаемый в результате оси `traverse`, т.к. по этой оси в общем случае осуществляется переход сразу по нескольким дугам XLink, которые не упорядочены между собой. Альтернативным подходом к сопоставлению узла с контекстной позицией может быть подход, принятый в языке XQuery: узлы в пределах одного XML-документа упорядочиваются в порядке обхода дерева документа, узлы из разных документов упорядочиваются произвольным, но единообразным образом для конкретной реализации [5].

Предлагаемая дополнительная ось `traverse` позволяет прикладному приложению осуществлять переходы по дугам языка XLink полностью прозрачным образом, без необходимости распознавать элементы XLink в XML-документе и разбирать их в соответствии с синтаксисом языка XLink с целью извлечения семантики дуг. Вне зависимости от того, где была определена дуга — в простой ссылке, или в расширенной ссылке, располагающейся в том же документе, или даже в отдельном документе, — переход по дуге осуществляется унифицированным образом.

Для иллюстрации предлагаемого расширения языка XPath осью `traverse` вернемся к рисунку 1, выражающему систему заказа товаров в виде 3 связанных XML-документов, и рассмотрим несколько примеров написания практических запросов к данной системе связанных документов.

**Пример 1.** Найдем имена всех клиентов, заказавших принтеры.

Для получения ответа на этот запрос требуется соединить данные изо всех 3 XML-документов на рис. 1, и соединение должно проводиться на основе ссылок XLink, связывающих части этих документов. Поскольку в рассматриваемой системе связанных XML-документов все ссылки языка XLink исходят из документа "`purchase-orders.xml`", описывающего сделанные заказы, то запрос будет адресоваться именно к этому документу, и наличие ссылок XLink позволит выбрать необходимую информацию из остальных документов с помощью оси `traverse`. Путь доступа, реализующий требуемый запрос, может быть записан следующим образом:

```
//order[entry/item/traverse::printer]/
  customer/traverse::person/name/text()
```



Шаг доступа, содержащий предикат и записанный в первой строке пути доступа, выбирает те выполненные заказы, в которых имеется хотя бы одно вхождение (`entry`) принтера среди заказанных товаров. Ось `traverse` внутри предиката осуществляет переход из каждого вхождения в заказе на каталог товаров; и тест узла `printer` позволяет указать, что из всех заказанных элементов каталога нас интересуют принтеры.

Во второй строке пути доступа, после того, как заказы были выбраны, осуществляется адресация к именам клиентов, сделавших соответствующие заказы. При адресации к именам клиентов снова используется ось `traverse`, и в этом случае она осуществляет переход уже на конкретного человека в документе "`clients.xml`", т.к. переход осуществляется по ссылке, ведущей от заказа к покупателю.

Результат вычисления запроса выбирает имя искомого клиента:

**John Smith**

**Пример 2.** Найдем список товаров, заказанных важным клиентом (в описании которого имеется вложенный элемент `<VIP/>`).

С помощью языка XPath, расширенного осью `traverse`, данный запрос может быть реализован в виде следующего пути доступа:

```
//order[customer/traverse::person/VIP]/
  entry/item/traverse::*
```

В отличие от примера 1, здесь заказы фильтруются в соответствии с условием, накладываемым уже на клиента.

Результат запроса состоит из двух элементов каталога, и в соответствии с нотацией, принятой в XQuery для записи последовательности из нескольких узлов [5], мы записываем результат в круглых скобках, отделяя узлы друг от друга при помощью запятых:

```
( <printer>
    <lot>001</lot>
    <descr>Ink jet</descr>
    <price>450</price>
  </printer>,
  <display>
    <lot>003</lot>
    <descr>Color, Digital</descr>
    <warranty>2 years</warranty>
    <price>500</price>
  </display> )
```

## 5. Адресация к дугам языка XLink

В предыдущем разделе было предложено простое и органичное расширение языка XPath осью `traverse`, которая предоставляет возможность осуществлять

переходы по дугам XLink, и на основе практических примеров была проиллюстрирована гибкость предложенного расширения.

Необходимо заметить, что расширение языка XPath лишь осью `traverse` обеспечивает ограниченные функциональные возможности для адресации структурных частей XML-документов, связанных между собой ссылками XLink. Данное наблюдение мотивируется тем, что ось `traverse` осуществляет переход сразу по всем дугам XLink, для которых контекстный узел является исходным ресурсом, а таких дуг может быть несколько. Для шага доступа со спецификатором оси `traverse` присутствующие в шаге доступа тест узла и предикаты позволяют приложению наложить условия на интересующие целевые ресурсы, но не на дуги, по которым осуществляется переход в данном шаге доступа. В практических задачах обработки XML-документов, связанных ссылками XLink, может быть желательно выбирать только некоторые из дуг, по которым необходимо осуществить переход из контекстного узла, а также формулировать запросы непосредственно к дугам и их семантическим атрибутам.

В соответствии с обозначенными в разделе 1 пожеланиями к языку, который бы позволил приложению формулировать запросы к XML-документам, связанным при помощи ссылок XLink, необходимо инкапсулировать сложности синтаксиса языка XLink и предоставить приложению прозрачное представление для имеющихся в XML-документах ссылок. Для достижения поставленной цели в настоящей статье предлагается представление каждой дуги XLink в виде информационной единицы (information item) Информационного Пространства XML [14], которое может рассматриваться как представление (view) в терминах реляционных баз данных и позволяет формулировать запросы к дуге XLink единообразно с другими структурными частями XML-документа.

В данном разделе рассматривается предлагаемый способ представления дуги XLink как информационной единицы; затем в качестве дальнейшего расширения к языку XPath вводятся 2 дополнительные оси, позволяющие приложению формулировать запросы к дугам XLink как к самостоятельным сущностям.

## 5.1. Дуга XLink как слабоструктурированные данные

Для наглядности предлагаемого представления дуги XLink в виде информационной единицы в данном разделе для записи примера будет использоваться синтаксис XML. При этом необходимо отметить, что XML является не единственным синтаксисом для записи дуги XLink в виде информационной единицы: в частности, как будет показано в разделе 6, при реализации предложенного языка запросов XPathLink функциональными методами для представления дуги XLink использовался формат SXML [17]. В данном разделе мы будем оперировать терминами Информационного Пространства XML с целью подчеркнуть независимость предлагаемого представления дуги XLink от конкретного внешнего синтаксиса.



Каждой дуге XLink будет соответствовать информационная единица типа “элемент” (element information item).

С каждой дугой XLink сопоставим имя, которое будет содержаться в локальном имени информационной единицы, представляющей дугу. Имена дуг не представлены в явном виде в синтаксисе языка XLink и в настоящей статье предлагаются для удобной классификации дуг приложением. Наличие имени у дуги предоставляет приложению удобный способ выбрать группу дуг, обладающих общими свойствами, без необходимости проводить детальный анализ внутренней структуры информационных единиц. Предлагается сопоставлять с каждой дугой XLink одно из следующих 6 имен:

- Дуге, определяемой простой ссылкой XLink, сопоставим имя **simple** (простая). Спецификация XLink определяет, что описание простой ссылки XLink содержит дугу неявным образом, и эта дуга обеспечивает переход от локального ресурса простой ссылки к ее удаленному ресурсу.
- Дуге, определяемой внутри расширенной ссылки XLink, сопоставим одно из имен **outbound** (исходящая), **inbound** (входящая), **third-party** (сторонняя) или **local-to-local** (от локального к локальному), в зависимости от типа дуги. Определения первых трех типов дуг введены Спецификацией XLink и уже обсуждались в разделе 2. Спецификация XLink не предусматривает какого-либо специального названия для дуги, и исходный, и целевой ресурсы которой являются локальными, ввиду редкой практической потребности для дуги данного типа. Для полноты рассмотрения возможных вариантов определяемых внутри расширенной ссылки типов дуг, для дуги, соединяющей 2 локальных ресурса, выбрано имя **local-to-local**.
- Дуге, семантическая роль которой состоит в указании на ссылочную базу (linkbase), сопоставим имя **linkbase**. Ссылочная база — это отдельный XML-документ, содержащий описания входящих и сторонних ссылок. Дуга, описанная как указывающая на ссылочную базу, позволяет приложению идентифицировать обрабатываемые ресурсы как исходные ресурсы для входящих и сторонних ссылок, которые описаны в ссылочной базе. По сравнению с остальными дугами, дуга, указывающая на ссылочную базу, должна обрабатываться особым образом: вместо перехода по дуге приложение должно загрузить ссылочную базу и извлечь описанные в ней ссылки для последующего использования. Для описания дуги как указывающей на ссылочную базу в языке XLink используется предопределенное значение глобального атрибута с семантикой роли дуги. При представлении дуги, указывающей на ссылочную базу, в виде информационной единицы, предлагается иметь для такой дуги особое имя с целью облегчения ее обработки приложением.

Представляющая дугу XLink информационная единица типа элемент имеет дочерние информационные единицы, которые содержат необходимые данные о соединяемых дугой ресурсах и семантических параметрах дуги.

### *5.1.1. Исходный и целевой ресурсы дуги*

Для хранения данных об исходном и целевом ресурсах дуги Xlink предлагается использовать при ее представлении в виде информационной единицы дочерние информационные единицы типа “элемент”. По аналогии с синтаксисом языка XLink для записи исходного и целевого ресурса дуги информационной единице, представляющей исходный ресурс, дадим локальное имя **from**, а информационной единице, представляющей целевой ресурс — локальное имя **to**.

Заметим, что при представлении дуги XLink в виде информационной единицы не требуется использование пространств имен XML [18]. Как отмечается в [19], пространства имен в XML используются для того, чтобы позволить прило-жению однозначным образом распознать имя элемента или атрибута в XML-документе [20]. В частности, Спецификация XLink использует собственное глобальное пространство имен, чтобы элементы языка XLink могли быть распознаны среди прочих элементов XML-документа. В предлагаемом в настоящей статье представлении дуги XLink в виде информационных единиц ситуация иная: про **все** информационные единицы данного представления известно, что они служат для описания дуги, и поэтому использования в именах информационных единиц какого-либо пространства имен не требуется.

В синтаксисе языка XLink дуги и соединяемые ими ресурсы описываются отдельно друг от друга, а исходный и целевой ресурс дуги специфицируются опосредованным образом — при помощи меток. Для того, чтобы избавить приложение от необходимости самостоятельно сопоставлять метки языка XLink при определении соединяемых дугой ресурсов, предлагается при представлении дуги в виде информационной единицы использовать полные описания ее исходного и целевого ресурса, явным образом включенные внутрь описания дуги.

Дополнительно, для обеспечения прозрачности межресурсных связей, предлагается единообразное представление как для локальных, так и для удаленных ресурсов языка XLink. Данное единообразие достигается за счет наличия у информационной единицы, представляющей исходный (аналогично — целевой) ресурс дуги, любых из следующих 3-х дочерних информационных единиц типа “элемент”:

- Информационная единица с именем **uri**, которая содержит последовательность символов, представляющих унифицированный идентификатор URI данного ресурса. Для удаленного ресурса XLink его унифицированный идентификатор явным образом задается в описании ресурса на языке XLink. Для локального ресурса XLink его идентификатором является унифицированный идентификатор XML-документа, в котором описан данный локальный ресурс.
- Информационная единица с именем **nodes**, которая содержит сам ресурс — узел XML-документа или набор узлов — в качестве своих дочерних информационных единиц. Для локального ресурса XLink таким



узлом является сам этот ресурс, т.к. только узел XML-документа может выступать в качестве локального ресурса.
- Информационная единица с именем **xpointer**, которая содержит последовательность символов, представляющих идентификатор фрагмента на языке XPointer для данного ресурса. Идентификатор фрагмента может использоваться для определения удаленного ресурса XLink и задается явным образом в описании ресурса.

Необходимо отметить, что в зависимости от способа обработки XML-документов со ссылками языка XLink конкретным приложением, при описании дуги XLink в виде информационной единицы для данного приложения могут быть предпочтительными различные способы представления соединяемых дугой ресурсов. Так, если по семантике приложения предполагается осуществлять переходы по дуге многократно, то для обеспечения быстроты переходов разумно заранее вычислить удаленный целевой ресурс дуги и хранить полученный в результате вычисления набор узлов в описании дуги — как дочерние для информационной единицы **nodes**. Напротив, если предполагается долговременное хранение предлагаемого представления дуги XLink на диске, то с целью поддержания взаимосвязи между дугой и соединяемыми ею ресурсами, предпочтительно описывать эти ресурсы не “по значению” — с использованием информационной единицы **nodes**, — а “по ссылке” — с использованием совокупной семантики информационных единиц **uri** и **xpointer.**

### *5.1.2. Семантические параметры дуги*

В соответствии со Спецификацией XLink, описание дуги может сопровождаться необязательными семантическими и поведенческими атрибутами. Семантические атрибуты включают в себя машинно-понимаемую **роль дуги** (arcrole) в системе межресурсных связей и понимаемый человеком **заголовок** (title). Поведенческие атрибуты определяют момент **активизации** дуги (actuate) и способ **демонстрации** целевого ресурса (show).

Каждый из упомянутых атрибутов языка XLink в предлагаемом представлении дуги XLink выражается информационной единицей типа “элемент”, которая носит локальное имя, совпадающее с локальным именем соответствующего атрибута. Каждая из информационных единиц содержит последовательность дочерних символов, которые выражают значение атрибута.

По уже обсуждавшимся в предыдущем пункте причинам, для информационных единиц в предлагаемом представлении дуги XLink не требуется использование какого-либо пространства имен. Использование информационных единиц типа “элемент” для выражения атрибутов языка XLink используется из соображений единообразного дизайна предлагаемого представления дуги XLink.

### *5.1.3. Пример*

На рис. 3 приведен пример расширенной ссылки языка XLink, которая соединяет несколько ресурсов о джазовом музыканте Луисе Армстронге. Из

рисунка легко видеть, что Спецификация XLink не накладывает ограничений на имя элемента, являющегося расширенной ссылкой, а принадлежность этого элемента языку XLink и его ссылочный тип задаются глобальным атрибутом `xlink:type`.

Расширенная ссылка на рис. 3 соединяет 3 ресурса, а именно:

- Описание биографии Луиса Армстронга. Это локальный ресурс языка XLink, т.е. всё содержимое ресурса целиком располагается внутри ссылочного элемента. Элемент языка XML специфицируется как локальный ресурс XLink с помощью наличия у данного элемента глобального атрибута `xlink:type` со значением `"resource"`.
- Песни Луиса Армстронга, которые располагаются в некотором XML-документе `"louis-songs.xml"`. Относительно рассматриваемой на рис. 3 расширенной ссылки этот ресурс является удаленным, поскольку к нему адресуются по его Унифицированному Идентификатору Ресурса. Удаленный ресурс описывается элементом языка XLink типа локатор [2], который специфицируется на языке XLink с помощью глобального атрибута `xlink:type` со значением `"locator"`.
- Упоминания о Луисе Армстронге в прессе. Данный ресурс также является удаленным, и представляет собой набор узлов — набор статей о музыканте, — выбираемых из документа `"www.press.com/archive.xml"` идентификатором фрагмента языка XPointer.

```
<louis-armstrong xmlns:xlink="http://www.w3.org/1999/xlink"
                 xlink:type="extended">
  <biography xlink:type="resource"  xlink:label="bio">
    <name>Louis Daniel Armstrong</name>
    <born>August 4, 1901</born>
    ...  <!— Описание биографии Луиса Армстронга —>
  </biography>
  <songs xlink:type="locator"  xlink:label="songs"
         xlink:href="louis-songs.xml"/>
  <press xlink:type="locator"
         xlink:href="www.press.com/archive.xml#xpointer(paper[keyword='Armstrong'])"
         xlink:label="papers"/>
  <to-bio xlink:type="arc"
          xlink:from="songs"  xlink:to="bio"
          xlink:actuate="onRequest"/>
  <arc xlink:type="arc"
       xlink:from="songs"  xlink:to="papers"/>
</louis-armstrong>
```

*Рис. 3. Расширенная ссылка XLink, соединяющая различные ресурсы о музыканте Луисе Армстронге.*



```
( <inbound>
    <from>
      <uri>louis-songs.xml</uri>
    </from>
    <to>
      <nodes>
        <biography xlink:type="resource"  xlink:label="bio">
          <name>Louis Daniel Armstrong</name>                    (а)
          <born>August 4, 1901</born>
          ...  <!— Описание биографии Луиса Армстронга —>
        </biography>
      </nodes>
    </to>
    <actuate>onRequest</actuate>
  </inbound>,
  <third-party>
    <from>
      <uri>louis-songs.xml</uri>
    </from>
    <to>                                                         (б)
      <uri>www.press.com/archive.xml</uri>
      <xpointer>xpointer(paper[keyword='Armstrong'])</xpointer>
    </to>
  </third-party> )
```

*Рис. 4: Представленные в виде информационных единиц и записанные в синтаксисе XML дуги языка XLink, которые были определены в расширенной ссылке на рис. 3*

Каждый из описанных в расширенной ссылке ресурсов помечен собственной меткой, задаваемой значением атрибута `xlink:label`. Эти метки используются для описания дуг. В расширенной ссылке на рис. 3 определяются 2 дуги: от песен Луиса Армстронга к его биографии и от песен к упоминаниям о музыканте в прессе. Заметим, что последняя дуга соединяет два удаленных ресурса.

На рис. 4 для обеих дуг приведено их представление в виде информационных единиц, предложенное в рамках настоящей статьи и описанное выше в данном разделе. Буквой `(а)` на рис. 4 помечена дуга, ведущая от песен к биографии; буквой `(б)` — дуга, ведущая от песен к упоминаниям о музыканте в прессе. Для наглядности визуального восприятия предлагаемого представления дуг XLink в виде информационных единиц на рис. 4 используется синтаксис XML; альтернативно может использоваться любой другой синтаксис, который соответствует аналогичному набору информационных единиц Информационного Пространства XML. Заметим, что поведенческий атрибут способа активизации дуги, ведущей к биографии Луиса Армстронга, на рис. 4 `(а)` находит свое выражение в виде элемента с одноименным локальным именем.

Рассмотренный пример будет использоваться в ходе последующего изложения — при определении языковых средств для формулирования запросов к дугам XLink.

## 5.2. Оси для адресации к дугам XLink

Теперь, когда определен способ представления каждой дуги XLink в виде информационной единицы Информационного Пространства XML, рассмотрим предлагаемое в данной статье дальнейшее расширение языка XPath, предоставляющее приложению возможность прозрачным образом формулировать запросы к дугам XLink.

Расширим функциональность языка XPath возможностью получить информацию обо всех дугах XLink, по которым можно осуществить переход из контекстного узла, т.е. таких, для которых контекстный узел является их исходным ресурсом. Следуя единообразию в дизайне предлагаемого расширения языка XPath поддержкой языка XLink, естественно оформить данную операцию получения всех дуг, исходным ресурсом которых является контекстный узел, в виде новой оси XPath. По аналогии с англоязычным термином для дуги, введенным Спецификацией XLink, данную ось назовем "`arc`".

**Определение 2.** Ось `arc` содержит все узлы, являющиеся в терминах Информационного Пространства XML представлением в виде информационных единиц всех дуг XLink, для каждой из которых контекстный узел служит исходным ресурсом.

Из определения следует, что ось `arc` возвращает непустой набор узлов только для такого контекстного узла, который является исходным ресурсом хотя бы для одной дуги XLink. Каждый из узлов, возвращаемый осью `arc`, имеет тип “элемент”, и его имя и внутренняя структура служат представлением для дуги XLink, которую этот элемент выражает.

При использовании в шаге доступа XPath спецификатора оси `arc` полностью сохраняется семантика остальных составляющих шага доступа — теста узла и предикатов. Например, тест узла позволяет выбрать дуги XLink по их именам — в соответствии с одним из 6 имен, предложенных для дуг в пункте 5.1.

Необходимо подчеркнуть, что каждый из узлов, возвращаемый предложенной осью `arc`, полностью соответствует Модели данных XPath [6], и поэтому к этим узлам далее могут применяться произвольные выражения XPath, с полным сохранением семантики этих выражений. В частности, предикаты могут использоваться для фильтрации дуг в соответствии со структурой представляющих их информационных единиц: например, выбрать дуги с конкретным значением некоторого семантического параметра XLink или наложить условие на тип целевого ресурса дуги.

Заметим, что в Модели Данных XPath [6], являющейся частью стандарта языка XPath Консорциума Всемирной Сети, узел типа “элемент” имеет ассоциированный с ним набор узлов типа “атрибут” и набор узлов типа “объявление пространства имен”, и относительный порядок узлов в каждом из наборов считается зависимым от конкретной реализации. В полной аналогии с



наборами атрибутов и объявлениями пространств имен можно говорить, что предлагаемое в данной статье расширение языка XPath поддержкой XLink дополняет Модель Данных XPath наличием ассоциированного с узлом **любого типа** набора выходящих из него дуг XLink. Каждая дуга реализуется в Модели Данных XPath узлом типа “элемент” и является представлением дуги в виде информационной единицы, как было описано в данном разделе.

Вернемся к расширенной ссылке на рис. 3 и представлению определяемой этой ссылкой дуг на рис. 4 и рассмотрим несколько кратких примеров использования предложенной оси `arc`. Необходимо отметить, что в соответствии с описаниями ресурсов и дуг, имеющимися в рассматриваемой расширенной ссылке, элемент документа (document element) [2] XML-документа "`louis-songs.xml`" является исходным ресурсом сразу для 2 дуг языка XLink. (На рис. 3 документ "`louis-songs.xml`" опущен ввиду того, что его структура не важна для последующих рассматриваемых примеров).

**Пример 3.** Пусть контекстным узлом является элемент документа XML-документа "`louis-songs.xml`". Выберем все сторонние дуги (third-party arcs), которые выходят из данного узла.
С использованием предлагаемой оси `arc` интересующие сторонние дуги могут быть получены с помощью выражения XPath — шага доступа:

**arc::third-party**

Результатом данного шага доступа является узел — приведенное на рис. 4 (б) представление дуги XLink в виде информационной единицы, записанное на данном рисунке в синтаксисе языка XML.
Заметим, что для получения данного результата приложению не требуется осуществлять разбор разметки языка XLink, поскольку эта работа полностью инкапсулирована предлагаемым в настоящей статье языком XPathLink.

**Пример 4.** Рассмотрим тот же контекстный узел, что и в примере 3. Выберем теперь все выходящие из данного контекстного узла дуги, имеющие поведенческий атрибут активизации со значением "`onRequest`".
Интересующие дуги могут быть получены с помощью следующего шага доступа:

**arc::*[actuate="onRequest"]**

Результатом данного шага доступа является узел — представление дуги, приведенное на рис. 4 (а). Необходимо отметить, что вычисление используемого в шаге доступа предиката осуществляется с полным сохранением семантики языка XPath, т.к. предложенное представление дуг XLink, подлежащих фильтрации по условию предиката, унифицировано с другими узлами XML-документа.

Наличие оси `arc` позволяет приложению формулировать запросы к дугам XLink и накладывать условия на интересующие дуги с помощью предикатов XPath. Когда интересующие приложение дуги XLink выбраны, приложению может потребоваться осуществить по ним переход. Для обеспечения данной функциональности можно было бы использовать введенную в разделе 4 ось

traverse, расширив ее и для узлов, представляющих дуги XLink в виде информационных единиц. Однако в данной статье предлагается не перегружать ось traverse излишне сложной функциональностью, поскольку негативным примером подобной перегруженности служит ось parent (родитель) в Стандарте языка XPath. Так, для контекстного узла типа атрибут ось parent выбирает содержащий его элемент (owner element) [14], а для контекстного узла любого другого типа — его родительский узел. Данная функциональная перегруженность оси parent приводит к тому, что транзитивное замыкание оси parent — ось ancestor-or-self — выбирает узлы нескольких разных типов, будучи примененная к контекстному узлу типа “атрибут”.

Для сохранения простоты семантики предлагаемого языка XPathLink, являющегося расширением языка XPath поддержкой языка XLink, в настоящей статье используется отдельная дополнительная ось для перехода к целевому ресурсу дуги XLink из информационной единицы, представляющей дугу. Данная ось получила название traverse-arc, т.е. “переход из дуги”.

**Определение 3.** Ось traverse-arc содержит все узлы, являющиеся целевыми ресурсами для контекстного узла, представляющего в виде информационной единицы дугу XLink. Для любого другого контекстного узла ось traverse-arc содержит пустой набор узлов.

Как отмечалось выше, семантика оси traverse-arc во многом напоминает семантику оси traverse, и области определения этих осей дополняют друг друга. Общность осей traverse-arc и traverse позволяет нам сразу перейти к рассмотрению примера.

**Пример 5.** Вернемся к примеру 3 и дополним его переходом по дуге. Пусть контекстным узлом снова является элемент документа XML-документа "louis-songs.xml". Осуществим переход по всем сторонним дугам, для которых контекстный узел является исходным ресурсом.
При наличии предлагаемых в статье дополнительных осей XPath искомая последовательность действий может быть оформлена в виде следующего пути доступа:

**arc::third-party/traverse-arc::***

Результатом вычисления данного пути доступа является набор узлов — ресурс, описанный расширенной ссылкой на рис. 3 и содержащий упоминания о Луисе Армстронге в прессе.

Предложенные в настоящей статье 3 дополнительные оси для языка XPath связаны следующим отношением: для произвольного теста узла NodeTest справедливо

$$\text{traverse::NodeTest} \equiv \text{arc::*/traverse-arc::NodeTest} \ ,$$

где символ тождественного равенства обозначает совпадение наборов узлов, выбираемых путями доступа в левой и правой частях тождества, для любого контекстного узла. Справедливость данного утверждения следует из того



очевидного наблюдения, что последовательное применение осей arc и traverse-arc реализует те же действия, что и ось traverse, если не требуется накладывать условие на интересующие дуги XLink, по которым следует осуществить переход.

## 6. Реализация

В данном разделе рассматривается реализация предлагаемого языка XPathLink при помощи функциональных методов программирования. В основе подхода к обработке XML-данных функциональными методами лежит SXML — представление Информационного Пространства XML в виде S-выражений [21]. Функциональный язык программирования Scheme, использованный для реализации, естественным образом обрабатывает S-выражения и, таким образом, SXML. В данном разделе дается краткий обзор SXML, затем рассматриваются ключевые моменты сделанной в рамках настоящей статьи реализации предлагаемого языка XPathLink.

### 6.1. Обзор SXML

SXML — это абстрактное синтаксическое дерево XML-документа в форме S-выражения. Языки SXML и XML могут рассматриваться как два синтаксически различных представления Информационного Пространства XML [14].

Язык XML использует язык разметки SGML для представления информационных единиц Информационного Пространства XML и их свойств. Древовидная структура документа (свойства **родитель** и **ребенок** информационных единиц Информационного Пространства XML) выражается при помощи вложенных тегов разметки [13].

Язык SXML использует для представления информационных единиц Информационного Пространства XML и их свойств S-выражения языка Scheme. Древовидная структура документа выражается при помощи вложенных списков. Каждая из информационных единиц Информационного Пространства XML представляется в виде S-выражения, первым членом которого является либо имя информационной единицы (для типов “элемент” и “атрибут”), либо служебное имя, предусмотренное для информационной единицы данного типа в грамматике SXML [17].

Пример простого XML-документа и его представления на SXML приведены на рис. 5, наглядно демонстрирующем соответствие между вложенными тегами XML и вложенными списками SXML.

```
<?xml version='1.0'?>
<doc>
 <tag attr1="value1" attr2="value2">
  <nested>Text node</nested>
 </tag>
```

```
(*TOP* (*PI* xml "version='1.0'")
 (doc
  (tag (@ (attr1 "value1") (attr2 "value2"))
   (nested "Text node")
  )
```

```
  <empty/>                              (empty)
</doc>                                  ))
```

*Рис. 5. XML-документ (левый столбец) и его представление в SXML.*

## 6.2. Разбор разметки языка XLink

При реализации предлагаемого языка запросов к совокупности XML-документов, связанных при помощи ссылок XLink, важным моментом технического характера является разбор элементов языка XLink в соответствии со спецификацией этого языка и инкапсуляция сложностей синтаксиса XLink от пользовательского приложения.

С целью создания единообразной среды для работы приложения как с SXML-документом, так и с дугой XLink последняя также записывается в формате SXML. Для разбора XML-документа, содержащего ссылки языка XLink, и представления документа и дуг XLink на SXML была реализована специализированная разновидность парсера SSAX [22]. SSAX — это парсер для разбора XML-документов, написанный на языке Scheme в чисто функциональном стиле и предоставляющий Простой Интерфейс Прикладного Программирования для XML (Simple API for XML — SAX). Интерфейс парсера SSAX основан на событиях, и в рамках данной статьи специализированные обработчики событий были реализованы для конструирования представления на SXML описанных в XML-документе дуг XLink, одновременно с конструированием представления на SXML самого разбираемого XML-документа. Реализованная архитектура специализированного парсера обеспечивает построение представления на SXML для документа и всех описанных в нем дуг XLink за один проход по документу.

Узел SXML-документа, являющийся исходным ресурсом для некоторой дуги XLink, соединяется со сконструированным представлением этой дуги при помощи концепции так называемых вспомогательных списков грамматики SXML. Вспомогательный список — это S-выражение, первым членом которого является служебный символ `'@@`. Вспомогательный список не может быть перепутан ни с одной другой информационной единицей SXML-документа, поскольку служебный символ `'@@` не может быть корректным именем языка XML. Хранение предлагаемого представления дуг XLink внутри вспомогательного списка подчеркивает близость относительно Модели данных XPath между набором выходящих из данного узла дуг и набором атрибутов данного узла, поскольку атрибуты на SXML синтаксически записываются внутри списка со служебным именем `'@`.

Необходимо отметить, что соединяемые ссылками XLink удаленные ресурсы, являющиеся фрагментами XML-документа, на языке XPointer могут специфицироваться по уникальному идентификатору — значению атрибута типа ID. Например, на рассмотренном ранее рис. 1, выражающем систему заказа товаров, по атрибутам типа ID определяются ресурсы внутри документа



`"clients.xml"`. Тип атрибута описывается не в самом XML-документе, но в его схеме; и поэтому разбор схемы XML-документа необходим для идентификации ресурсов, представляющих собой фрагменты внутри данного документа. Сделанная в рамках данной статьи реализация обеспечивает извлечение описаний атрибутов типа ID из схемы на языке Определения Типа Документа (Document Type Definition — DTD) [2].

Заметим, что документы на HTML также представляют собой слабоструктурированные данные, а семантика гиперссылок HTML, задаваемых при помощи элемента `A`, может рассматриваться как частный случай семантики дуг XLink. Из сделанного наблюдения следует, что применение предлагаемого в настоящей статье языка XPathLink может быть без изменений расширено на случай формулирования запросов к совокупности документов на HTML, связанных гиперссылками. Парсер HtmlPrag, позволяющий сконструировать представление в виде S-выражений для практических HTML-документов [23], в рамках данной статьи был дополнен обработкой гиперссылок HTML по аналогии с дугами XLink.

На высоком уровне, для получения в виде SXML набора связанных документов, прикладному приложению предоставляется функция `xlink:documents` (имя `documents` выбрано по аналогии с языками XSLT и XQuery, а префикс `xlink` подчеркивает связь данной функции с языком XLink). Функция принимает в качестве аргументов один или более Унифицированный Идентификатор Ресурса (URI) для интересующих приложение документов. По предоставленным Унифицированным Идентификаторам Ресурсов функция получает соответствующие документы, конструирует их представление на SXML и связывает все узлы SXML-документов, являющиеся исходными ресурсами для дуг языка XLink, с представлением этих дуг на SXML. Семантика функции `xlink:documents` объединяет в себе следующие действия:

- Получение документов по их Унифицированным Идентификаторам Ресурса, определение типа каждого документа: XML или HTML;
- Конструирование с помощью соответствующего парсера (специализированная реализация SSAX или HtmlPrag) представления документа на SXML и построение представления на SXML описанных в данном документе дуг XLink или гиперссылок HTML;
- Загрузка ссылочных баз языка XLink с целью получения тех дуг XLink, исходные и/или целевые ресурсы которых располагаются в интересующих приложение документах;
- Ассоциирование исходных ресурсов с соответствующими дугами XLink (учитывая ту специфику языка XLink, что дуги в общем случае описываются в другом месте относительно местоположения их исходных ресурсов).

Наличие единой функции `xlink:documents`, обеспечивающей проведение всех описанных выше действий за один высокоуровневый вызов, способствует простоте использования сделанной реализации прикладным приложением.

## 6.3. Реализация предложенных осей как расширения к SXPath

SXPath — это реализация XPath на языке функционального программирования Scheme, предоставляющая язык запросов к документам на SXML. Реализация SXPath трактует путь доступа как составной запрос к дереву документа или его ветви. Отдельный шаг доступа представляет собой комбинацию проекции, выборки или транзитивного замыкания [24]. Несколько шагов доступа комбинируются с помощью операций последовательного применения или объединения.

Библиотека SXPath состоит из набора низкоуровневых предикатов, фильтров, операций выборки и комбинаторов; и функций высокого уровня, реализованных в терминах низкоуровневых функций.

В рамках данной статьи предложенные 3 дополнительные оси, обеспечивающие поддержку в XPath языка XLink, были реализованы в качестве расширения к SXPath. Примечательно, что оси `traverse` и `traverse-arc`, осуществляющие переход по дугам, прозрачным для приложения образом вызывают вычислитель языка XPointer, когда требуется разыменовать (resolve) идентификатор фрагмента на XPointer для целевого ресурса дуги. Упомянутые оси также могут прозрачным для приложения образом загружать по Унифицированным Идентификаторам Ресурса те документы, в которых располагаются целевые ресурсы перехода и которые ранее не были загружены с помощью функции `xlink:documents`. Данное свойство сделанной реализации делает ее мощным инструментом для работы с ресурсами в масштабах Всемирной Сети.

Для иллюстрации сделанной реализации вернемся к некоторым рассмотренным ранее примерам и посмотрим их вычисление с помощью предложенного расширения SXPath поддержкой языка XLink.

**Пример 6.** Вычислим пример 1 с помощью SXPath, расширенного поддержкой языка XLink:

```
((sxpath/c
  "//order[entry/item/traverse::printer]/
   customer/traverse::person/name/text()")
 (xlink:documents "purchase-orders.xml"))
```

Высокоуровневая функция `sxpath/c` получает на вход выражение XPath и конструирует реализацию этого выражения в виде комбинации низкоуровневых примитивов библиотеки SXPath. Сконструированная реализация выражения XPath имеет сигнатуру функции, которая затем применяется к набору узлов. В данном примере этот набор состоит из единственного узла — представленного на SXML документа `"purchase-orders.xml"`.



Результат вычисления данного кода на языке Scheme представляет собой список — набор узлов, состоящий из единственного текстового узла:

```
("John Smith")
```

**Пример 7.** Вычислим пример 2 с помощью SXPath, расширенного поддержкой языка XLink.

```
((sxpath/c
  "//order[customer/traverse::person/VIP]/
   entry/item/traverse::*")
 (xlink:documents "purchase-orders.xml"))
```

Результатом вычисления данного кода является следующий список, представляющий собой набор узлов на SXML:

```
((printer
   (lot "001")
   (descr "Ink jet")
   (price "450"))
 (display
   (lot "003")
   (descr "Color, Digital")
   (warranty "2 years")
   (price "500")))
```

## 6.4. Язык запросов

Тесная интегрированность SXPath с языком программирования общего назначения Scheme предоставляет функциональность языка запросов к SXML-документам.

Проводя аналогии с языком запросов к XML-документам XQuery, можно обозначить следующие соответствия между конструкциями XQuery и возможностями Scheme:

- Итерация по членам последовательности `for-return` языка XQuery реализуется на Scheme функцией `map`. Функция `map` получает на вход функцию от одного аргумента и список, и формирует новый список, последовательно применяя полученную в качестве аргумента функцию к каждому из членов аргумента-списка.
- Функции и операторы XQuery [11], а также функции XQuery, определяемые пользователем, реализуются на Scheme также функциями — стандартными, библиотечными или определяемыми пользователем. Дополнительно, в языке Scheme функции могут использоваться как объекты первого класса, что не поддерживается в XQuery.
- Конструкторы различных типов узлов в языке XQuery реализуются на Scheme конструкторами списков. Более того, наличие в языке Scheme выражений квази-цитирования (quasiquote) и снятия цитирования

(unquote) позволяет компактным и наглядным образом комбинировать константные выражения и фрагменты вычисляемых выражений. Аналогичные идеи используются в синтаксисе XSLT для комбинирования литеральных элементов результата [2] и исполняемых инструкций.

Рассмотрим совместное использование SXPath и Scheme как языка запросов к SXML-документам на конкретном примере.

**Пример 8.** Вновь обратимся к рисунку 1 и подведем счет для каждого сделанного заказа. Счет будет включать в себя имя клиента и общую цену с учетом количества единиц каждого заказанного товара.

Получение желаемого результата требует использования языка запросов, поскольку требуется не просто выбрать некоторые узлы из XML-документов, но также сконструировать новые узлы, которых в самих документах нет. На рис. 6 показано решение поставленного запроса в двух вариантах:

- На языке XQuery, расширенном предлагаемыми в настоящей статье дополнительными осями XPath;
- На Scheme, благодаря тесной интеграции SXPath с языком программирования.

```
for $order in document("purchase-orders.xml")//order
return <bill>
       <total-price>{ fn:sum( for $entry in $order/entry
                      return item/traverse::*/price * quantity )
       }</total-price>
       {$order/customer/traverse::person/name}
      </bill>
```

```
(map
 (lambda (order)
  `(bill
    (total-price
     ,(apply + (map (sxpath/c "item/traverse::*/price * quantity")
                    ((sxpath/c "entry") order))))
    ,@((sxpath/c "customer/traverse::person/name") order)))
 ((sxpath/c "//order")
  (xlink:documents "purchase-orders.xml")))
```

*Рис. 6. Вычисление счета для каждого заказа: с помощью XQuery, расширенного поддержкой XLink, (вверху) и с помощью Scheme (внизу). См. также пример 8.*

Из рис. 6 легко видеть, что соответствие между выражениями языка XQuery и вызовами функций языка Scheme является достаточно прямолинейным, и многие конструкции языка XQuery имеют свое наглядное отражение в виде примитивов языка Scheme над деревьями SXML-документов.



Результатом вычисления представленного на рис. 6 кода на Scheme является набор узлов, выражающих на SXML искомые счета для сделанных заказов:

```
((bill
  (total-price 1900)
  (name "John Smith"))
 (bill
  (total-price 20)
  (name "Paul Brown")))
```

## 7. Ограничения предлагаемого языка запросов

Предложенный в данной статье язык XPathLink не предоставляет средств по работе с ресурсами XLink, которые не являются узлами или наборами узлов. Такими ресурсами могут быть только удаленные ресурсы (т.е. те, которые участвуют в ссылке XLink благодаря тому, что к ним адресуются с помощью унифицированного идентификатора URI), и к их числу относятся:

- Ресурсы, которые имеют формат, отличный от XML и XHTML, например, мультимедийные форматы;
- Ресурсы, при адресации к которым используется идентификатор фрагмента на языке XPointer, вычисляющийся в **интервал** (range) или последовательность символов внутри текстового узла.

Введенные ограничения мотивированы требованием обеспечить замкнутость всех операций языка запросов относительно сущности **набор узлов**. Так, интервалы языка XPointer нарушают иерархическую структуру документа, поскольку разрывают границы разметки, и поэтому не могут быть представлены в виде набора узлов модели данных XPath.

Заметим, что идентичные ограничения приняты и в Спецификации Языка Преобразований XSL при обеспечении доступа к документам, отличным от обрабатываемого XML-документа [25]. Данная аналогия подтверждает, что введенные ограничения предложенного в настоящей статье языка XPathLink являются оправданными.

## 8. Заключение

Данная статья посвящена вопросу обработки совокупности XML-документов как системы связанных ресурсов, описанной с помощью языка XLink. На основе анализа существующих работ в области обработки XML-документов, связанных ссылками XLink, была отмечена потребность в наличии языка высокого уровня, который бы инкапсулировал сложности синтаксиса XLink и обеспечивал приложение возможностями формулировать запросы к ссылкам XLink и осуществлять переходы по определяемым этими ссылками дугам.

Был проведен обзор языка адресации структурных частей XML-документа Xpath, и было замечено непосредственное семантическое соответствие между

переходом по дуге в терминах языка XLink и осью в XPath. Было предложено ввести в XPath дополнительную ось `traverse`, которая осуществляет переход из контекстного узла как из исходного ресурса дуги XLink на ее целевой ресурс. Были рассмотрены свойства предложенной дополнительной оси, и сценарии ее использования были проиллюстрированы практическими примерами.

Для обеспечения прикладного приложения возможностью прозрачным образом формулировать запросы к дугам XLink как к самостоятельным сущностям было разработано представление для дуг в виде информационной единицы Информационного Пространства XML. В язык XPath была добавлена ось `arc`, позволяющая приложению для данного контекстного узла получить все дуги XLink, для которых контекстный узел служит исходным ресурсом. Было отмечено единообразие предложенного слабоструктурированного представления дуг XLink и других узлов XML-документа, и рассмотрено место дуг XLink в Модели Данных языка XPath. Обсуждались преимущества наличия двух различных осей для перехода по дуге XLink из узла XML-документа и из слабоструктурированного представления дуги.

Рассматривались детали сделанной в рамках данной статьи реализации функциональными методами предложенного расширения XPath. Тесная интеграция сделанной реализации с языком программирования общего назначения Scheme обеспечивает прикладное приложение функциональностью языка запросов. Предложенный в статье язык запросов и его реализация функциональными методами являются мощным и гибким инструментом для обработки совокупности XML-документов, связанных ссылками XLink.

**Литература**


1. S. DeRose, E. Maler and D. Orchard (editors). XML Linking Language (XLink) Version 1.0. W3C Recommendation 27 June 2001. http://www.w3.org/TR/xlink/
2. Когаловский М.Р. Глоссарий по технологиям платформы XML. Версия 4 (25-11-2003). http://www.elbib.ru/index.phtml?page= elbib/rus/methodology/xmlbase/glossary_XML
3. T. Bray and S. DeRose (editors). Extensible Markup Language (XML): Part 2. Linking. W3C Working Draft April-06-97. http://www.w3.org/TR/WD-xml-link-970406.html
4. S. J. DeRose (editor). XML XLink Requirements Version 1.0. W3C Note 24-Feb-1999. http://www.w3.org/TR/NOTE-xlink-req
5. S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie and J. Simeon (editors). XQuery 1.0: An XML Query Language. W3C Working Draft, 12 November 2003. http://www.w3.org/TR/2003/WD-xquery-20031112/
6. J. Clark и S. DeRose (редакторы). Язык XML Path (XPath) Версия 1.0. Рекомендация Консорциума Всемирной Сети от 16 Ноября 1999. http://citforum.ru/internet/xpath/index.shtml
7. Лизоркин Д.А. и Лисовский К.Ю. Языки XSLT и XLink и их реализация функциональными методами. Электронные Библиотеки, 2003, Том 6, Выпуск 5. http://www.elbib.ru/index.phtml?page= elbib/rus/journal/2003/part5/LL
8. T. Berners-Lee, R. Fielding, U.C. Irvine and L. Masinter. Request for Comments: 2396. Uniform Resource Identifiers (URI): Generic Syntax. Network Working Group, August 1998. http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc2396.html





9. P. Grosso, E. Maler, J. Marsh and N. Walsh (editors). XPointer Framework. W3C Recommendation 25 March 2003. http://www.w3.org/TR/2003/REC-xptr-framework-20030325/
10. Лизоркин Д.А. и Лисовский К.Ю. Реализация XLink — языка ссылок XML — с помощью функциональных методов. Принята к публикации в журнал ``Программирование", 2005, номер 1. http://www.maik.rssi.ru/cgi-bin/journal.pl?name=procom&page=main
11. A. Malhotra, J. Melton and N. Walsh (editors). XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Working Draft 23 July 2004. http://www.w3.org/TR/2004/WD-xpath-functions-20040723/
12. B. DuCharme. XLink: Who Cares? XML.com, O'Reilly Media. http://www.xml.com/pub/a/2002/03/13/ xlink.html
13. Лисовский К.Ю. Разработка XML-приложений на языке Scheme. Программирование, выпуск 28, номер 4, 2002. http://www.maik.rssi.ru/journals/procom.htm
14. J. Cowan and R. Tobin (editors). XML Information Set (Second Edition). W3C Recommendation 4 February 2004. http://www.w3.org/TR/xml-infoset/
15. Лизоркин Д.А. и Лисовский К.Ю. SXML: XML-документ как S-выражение. Электронные библиотеки, 2003, Том 6, Выпуск 2. http://www.elbib.ru/index.phtml?page= elbib/rus/journal/2003/part2/LK
16. Лизоркин Д.А. и Лисовский К.Ю. Язык XML Path (XPath) и его функциональная реализация SXPath. Электронные Библиотеки, 2003, Том 6, Выпуск 4. http://www.elbib.ru/index.phtml?page= elbib/rus/journal/2003/part4/LL
17. O. Kiselyov. SXML, revision 3.0, March 12, 2004. http://okmij.org/ftp/Scheme/SXML.html
18. T. Bray, D. Hollander and A. Layman (editors). Namespaces in XML. World Wide Web Consortium 14-January-1999. http://www.w3.org/TR/REC-xml-names/
19. J. Clark. XML Namespaces. February 4, 1999. http://www.jclark.com/xml/xmlns.htm
20. Лизоркин Д.А. и Лисовский К.Ю. Пространства имен в XML и SXML. Электронные библиотеки, 2003, Том 6, Выпуск 3. http://www.elbib.ru/index.phtml?page= elbib/rus/journal/2003/part3/LL
21. O. Kiselyov and K. Lisovsky. XML, XPath, XSLT Implementation as SXML, SXPath and SXSLT. International Lisp Conference ILC 2002, San Francisco. October, 2002. http://www.okmij.org/ftp/papers/SXs.pdf
22. O. Kiselyov. A better XML parser through functional programming. Practical Aspects of Declarative Languages: 4th International Symposium, PADL 2002. Springer-Verlag Heidelberg, ISSN: 0302-9743. http://www.okmij.org/ftp/papers/XML-parsing.ps.gz
23. Neil W. Van Dyke. HtmlPrag: Pragmatic Parsing of HTML to SHTML and SXML. July 2004. http://www.neilvandyke.org/htmlprag/
24. K. Lisovsky. STX: Scheme-enabled XSLT processor. http://www.pair.com/lisovsky/transform/stx/
25. J. Clark (редактор). Язык преобразований XSL (XSLT) Версия 1.0. Рекомендация Консорциума Всемирной Сети от 16 ноября 1999. http://www.rol.ru/news/it/helpdesk/ xslt01.htm