ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ГОСТ Р ИСО 24610-2— 2013**

# Менеджмент языковых ресурсов

# СТРУКТУРЫ ЭЛЕМЕНТОВ

## Часть 2

## Декларация системы элементов

**ISO 24610-2:2011**
**Language resource management — Feature structures — Part 2: Feature system declaration (IDT)**

**Издание официальное**

**Москва**
**Стандартинформ**
**2015**

gost.e-ecolog.ru

## Предисловие

1 ПОДГОТОВЛЕН ЗАО «Проспект» на основе собственного аутентичного перевода на русский язык международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 55 «Терминология, элементы данных и документация в бизнес-процессах и электронной торговле»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 8 ноября 2013 г. № 1389-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 24610-2:2011 «Менеджмент языковых ресурсов. Структуры элементов. Часть 2. Декларация системы элементов» (ISO 24610-2:2011 «Language resource management — Feature structures — Part 2: Feature system declaration»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2015

Настоящий стандарт не может быть воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

## Содержание

1 Область применения . . . 1
2 Нормативные ссылки . . . 1
3 Термины и определения . . . 1
4 Общая структура стандарта . . . 4
5 Базовые понятия . . . 4
5.1 Рассматриваемые типизированные структуры элементов . . . 4
5.2 Типы . . . 5
5.3 Иерархии наследования типов . . . 8
5.4 Ограничения для типов . . . 9
5.5 Опциональные (стандартные) значения и недоопределение . . . 10
5.6 Категоризация . . . 11
6 Определение формальной правильности и адекватности . . . 12
6.1 Общее описание . . . 12
6.2 О стандарте ИСО 24610 . . . 13
7 Система элементов для грамматики . . . 17
7.1 Общие сведения . . . 17
7.2 Выборочные FSD . . . 18
8 Декларация системы элементов . . . 22
8.1 Общие сведения . . . 22
8.2 Привязка текста к декларациям систем элементов . . . 22
8.3 Общая структура декларации системы элементов . . . 24
8.4 Декларации элементов . . . 25
8.5 Ограничения структуры элементов . . . 30
Приложение А (обязательное) Схема XML для структур элементов . . . 33
Приложение В (обязательное) Детализированный пример . . . 41
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации . . . 44
Библиография . . . 45

## Введение

ИСО 24610 состоит из двух отдельных частей.

- Часть 1 *Представление структуры элементов* — посвящена описанию структур, обеспечивающих неформальное, но достаточно явное выражение их характеристик, а также описанию представления структур элементов с использованием языка XML вообще и различных типов таких структур в частности. В этой части закладываются основы правильного форматирования конструируемых XML-ссылок, обеспечивающих обмен структурами элементов (возможно с выделением типов) между приложениями.

- Часть 2 *Декларация системы элементов* — предоставляет стандартный метод реализации различных типов структур элементов в языковой среде XML: сначала путем определения множества типов и их иерархии, затем посредством формулирования ограничений, касающихся различных типов, на множестве элементов и их допустимых значений, и, наконец путем введения множества условий, касающихся надежности структур элементов в аспекте их использования в конкретных приложениях, особенно в целях управления языковыми ресурсами.

Структура элементов — это структура данных общего назначения, которая идентифицирует и группирует отдельные элементы посредством присваивания каждому из них конкретного значения. Благодаря универсальности структур элементов они могут использоваться для представления самых разных типов информации. Существующие связи между различными «порциями» информации и их реализация в языке разметки образуют некоторый метаязык для представления контента лингвистического характера. Более того, подобная реализация позволяет сформировать описание множества элементов и значений, соответствующих конкретным типам и их ограничениям, посредством декларирования системы элементов или с помощью других механизмов языка XML, обсуждаемых в данной части ИСО 24610.

Некоторые положения данной части заимствованы из ИСО 24610-1:2006 в целях обеспечения полной независимости части 2 от части 1.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Менеджмент языковых ресурсов**

**СТРУКТУРЫ ЭЛЕМЕНТОВ**

**Часть 2**

**Декларация системы элементов**

Language resource management. Feature structures. Part 2. Feature system declaration

**Дата введения — 2015—01—01**

## 1 Область применения

В настоящем стандарте предлагается формат представления, хранения и обмена для структур элементов в прикладных системах, основанных на использовании естественного языка, как для аннотирования, так и для формирования лингвистических данных. Основная цель состоит в том, чтобы предложить такой формат машинной обработки, который позволяет определить иерархию типов и декларировать ограничения, накладываемые на множество спецификаций элементов и на операции со структурами элементов, обеспечивая таким образом средства контроля соответствия каждой структуры элементов их базовой спецификации. Структуры элементов — это важнейшая часть многих формализаций в лингвистике и основополагающий механизм представления информации, используемой или порождаемой в приложениях, связанных с построением языковых систем.

## 2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при использовании настоящего стандарта. В случае ссылок на документы, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В случае, когда дата утверждения не приведена, следует пользоваться последней редакцией ссылочных документов, включая любые поправки и изменения к ним.

ИСО 24610-1:2006 Управление языковыми ресурсами. Структуры элементов. Часть 1. Представление структуры элементов (ISO 24610-1:2006, Language resource management — Feature structures — Part 1: Feature structure representation)

ИСО/МЭК 19757-2:2008 Информационные технологии. Язык определения схемы документа (DSDL). Часть 2. Валидация на основе регулярной грамматики. RELAX NG (ISO/IEC 19757-2:2008, Information technology — Document Schema Definition Language (DSDL) — Part 2: Regulargrammar-based validation — RELAX NG)

## 3 Термины и определения

Для целей настоящего стандарта используются термины и определения по ИСО 19757-2, а также терминология, приведенная ниже:

3.1 **ограничение по допустимости** (admissibility constraint): Спецификация множества **разрешенных элементов** (3.2) и **допустимых значений элементов** (3.3), ассоциируемая с конкретным **типом** (3.24).

**Издание официальное**

3.2 **разрешенный элемент** (admissible feature): Элемент, для которого соответствующая **структура элементов** (3.14) определенного **типа** (3.24) может нести в себе конкретное **значение** (3.17).

Примечание — В некоторых интерпретациях этот термин часто приобретает оттенок обязательности, т. е. считается, что структуры элементов конкретного типа должны содержать в себе значение для каждого разрешенного элемента. Однако в данном случае этот термин не предполагает обязательного присутствия элемента.

3.3 **разрешенное значение элемента** (admissible feature value): **Значение** (3.17), которое должно быть отнесено к категории **допустимых элементов** (3.2) в **структурах элементов** (3.14) данного **типа** (3.24).

3.4 **атомарный тип** (atomic type): Пользовательский **тип** (3.24), который не имеет декларируемых или наследуемых **допустимых элементов** (3.2).

3.5 **множество с повторяющимися элементами** (bag): Триплет, образованный целым числом $n$, множеством $S$ и функцией отображения целых чисел в диапазоне от 1 до $n$ в элементы $S$.

Примечание — Множество с повторяющимися элементами — это промежуточный объект между обычным множеством (как совокупностью неупорядоченных элементов) и списком (где отдельные элементы могут встречаться многократно).

3.6 **встроенный элемент** (built-in): Элемент, не определяемый пользователем, но могущий появиться вместо **структуры элементов** (3.14), например в качестве **значения элемента** (3.17).

Примечание — Встроенные элементы могут быть атомарными или составными. К первым относятся численные, строковые, символьные и двоичные элементы; ко вторым — **коллекции** (3.7) и применяемые логические операторы: например дизъюнкция, отрицание и слияние (см. п. 5.2.4).

3.7 **коллекция** (collection): **Значение элемента** (3.17), содержащее совокупность возможных значений, которые представлены в виде списка, обычного множества или **множества с повторяющимися элементами** (3.5).

3.8 **ограничение** (constraint): Компонент спецификации, которая идентифицирует некоторую коллекцию **структур элементов** (3.14) как неадекватную.

Примечание 1 — Все ограничения по своей синтаксической форме импликативны, хотя некоторые из них выделяются как ограничения по допустимости. См. **адекватность** (3.31) и 5.4. Все структуры элементов, которые не исключены явным образом как неадекватные, считаются адекватными.

Примечание 2 — Структура элементов, не идентифицированная таким образом как не соответствующая никакому из ограничений в системе элементов, считается адекватной.

3.9 **значение по умолчанию, стандартное значение** (default value): **Значение** (3.17), присваиваемое **элементу** (3.12) в том случае, когда оно не определено

***Пример*** — В датском языке при отсутствии явного указания грамматического рода ему присваивается значение «мужской».

Примечание — Структура элементов не может содержать элементов, для которых не указано соответствующее значение.

3.10 **пустая структура элементов** (empty feature structure): **Структура элементов** (3.14), не содержащая никакой информации.

Примечание — Пустая структура элементов категоризирует все другие структуры элементов.

3.11 **расширение** (extension): Преобразование типа **категоризации** (3.21).

Примечание — Структура элементов $F$ расширяет $G$ тогда и только тогда, когда $G$ категоризирует $F$.

3.12 **элемент** (feature): Свойство или аспект объекта, формально представляемые как функция, отображающая объект в его соответствующее **значение** (3.17).

3.13 **спецификация элементов** (feature specification): Связывание **элемента** (3.12) с его **значением** (3.17) в описании структуры элементов.

3.14 **структура элементов** (feature structure): Структура записей, которая ставит в соответствие каждой коллекции элементов одно **значение** (3.17).

Примечание 1 — Каждое значение представляет собой структуру элементов или более простой **встроенный элемент** (3.6), такой как строка.

Примечание 2 — Структуры элементов частично упорядочены. Минимальными в этом упорядочении являются пустые структуры элементов.

3.15 **система элементов** (feature system): **Иерархия типов** (3.26), в которой каждый **тип** (3.24) ассоциируется с коллекцией **ограничений по допустимости** (3.1) и **импликативными ограничениями** (3.18).

Примечание — Сравните **декларация типа** (3.25).

3.16 **декларация системы элементов** (feature system declaration); FSD: Описание конкретной **системы элементов** (3.15).

3.17 **значение для элемента** (feature value): Объект или совокупность объектов, характеризующие некоторое свойство другого объекта.

3.18 **импликативное ограничение** (implicational constraint): Ограничение типа «если $G$, то $H$», где $G$ и $H$ — это **структуры элементов** (3.14).

Примечание — Такое ограничение идентифицирует любую структуру элементов $F$ как неадекватную, когда $G$ категоризирует $F$, а $F$ и $H$ обычно не имеют адекватного расширения. См. **категоризация** (3.21) и п. 8.5. Часто ограничение такого вида используется при обращении к импликативным ограничениям, которые одновременно не являются ограничениями по допустимости.

3.19 **интерпретация** (interpretation): Минимально информативное (т. е. наиболее общее) **расширение** (3.11) **структуры элементов** (3.14), которое совместимо с множеством ограничений, объявленным в **декларации системы элементов** (3.16).

3.20 **частичный порядок** (partial order): Множество $S$, для которого определено отношение $\leq$ на $S \times S$, которое 1) рефлексивно (для всех $s \in S$, $s \leq s$), 2) антисимметрично (для всех $p$, $q \in S$, если $p \leq q$ и $q \leq p$, то $p = q$), и 3) транзитивно (для всех $p$, $q$, $r \in S$, если $p \leq q$ и $q \leq r$, то $p \leq r$).

Примечание — Множество целых чисел $Z$ частично упорядочено, но дополнительно оно обладает свойством, согласно которому для каждого $p$, $q \in Z$ выполняется условие $p \leq q$ или $q \leq p$. Этим свойством обладает не любой частичный порядок. Например, такой частичный порядок, как таксономическая классификация организмов по типам, родам и видам, указанным свойством не обладает; не обязательно обладают этим свойством также иерархии типов. Типизированные структуры элементов системы не имеют этого свойства, если (a) данное свойство присуще иерархии их типов, и (b) иерархия типов состоит из единственного типа либо каждый тип $y$ ограничен присутствием одного-единственного подходящего элемента.

3.21 **категоризация** (subsumption): Свойство, связывающее две структуры элементов $G$ и $F$ таким образом, что $G$ считается принадлежащей $F$ тогда и только тогда, когда $F$ несет в себе всю информацию, которую содержит $G$.

Примечание — Формальное определение представлено ниже, в 5.6.

3.22 **подтип** (subtype): **Тип** (3.24), на который распространяются ограничения и соответствующие характеристики, содержащиеся в другом типе.

3.23 **супертип, надтип** (supertype): **Тип** (3.24), от которого другой тип наследует ограничения и соответствующие элементы.

Примечание — $s$ является подтипом $t$ тогда и только тогда, когда $t$ — супертип $s$. Каждый тип является подтипом и супертипом самого себя.

3.24 **семантический тип** (semantic type): **Тип**, характеризующий выражение, с помощью которого коллекция **структур элементов** (3.14) различается как идентифицируемый и концептуально значимый класс.

Примечание — Как это следует из имени *семантический тип*, типы, о которых идет речь в данной части ИСО 24610, не предназначены для различения структур элементов или их спецификаций по синтаксису.

3.25 **декларация типа** (type declaration): Информационная структура, декларирующая **супертипы** (3.23), **допустимые элементы** (3.2), **значения допустимых элементов** (3.3), **ограничения по допустимости** (3.1) и **импликативные ограничения** (3.18) для данного **типа** (3.24).

Примечание — Ограничения, накладываемые на тип в результирующей системе элементов, — это ограничения, объявленные в декларации дополнительно к унаследованным от супертипов.

3.26 **иерархия типов** (type hierarchy): **Частичный порядок** (3.20) на множестве **типов** (3.24).

Примечание — См. ИСО 24610-1:2006, Приложение С, *Наследуемые иерархии типов*.

3.27 **типизированная структура элементов** (typed feature structure); TFS: **Структура элементов** (3.14), несущая в себе **тип** (3.24).

3.28 **типизация** (typing): Присваивание **семантического типа** (3.24) **встроенному элементу** (3.6) либо **структуре элементов** (3.14), атомарной или составной.

Примечание — Семантические типы в системах элементов частично упорядочены и имеют множественные отношения наследования.

3.29 **недоопределение** (underspecification): Предоставление неполной информации о **значении** (3.17).

Примечание — Недоопределение обычно категоризирует одно значение из диапазона возможных значений, которые могут быть сведены к единственному значению путем последовательного наложения ограничений. См. **категоризация** (3.21).

3.30 **формальная правильность** (well-formedness): Синтаксическое соответствие представления **структуры элементов** (3.14) ИСО 24610-1.

3.31 **адекватность** (validity): Соответствие **типизированной структуры элементов** (3.27) действующим **ограничениям** (3.8) **конкретной системы элементов** (3.15).

Примечание — См. раздел 6.

## 4 Общая структура стандарта

Основное содержание настоящего документа отражено в четырех разделах — 5, 6, 7 и 8.

- В разделе 5 *Базовые понятия* — рассматривается определение типизированных структур элементов и вводятся понятия атомарных и составных типов структур элементов, коллекций и прочих операторов, могущих фигурировать в значениях элементов; затем описываются понятия наследуемых типов, иерархий типов, ограничений типов, значений по умолчанию и недоопределения, которые имеют важнейшее значение для конструирования систем элементов.
- В разделе 6 *Определение формальной правильности и адекватности* — обсуждаются условия отмеченности и достоверности структур элементов.
- Раздел 7 *Система элементов для грамматики* — иллюстрирует способ определения типов с использованием иерархии и ограничений типов, в рамках которых декларируются допустимые элементы и значения для конкретных типов.
- В разделе 8 *Декларация системы элементов* — показывается, каким образом система элементов может быть декларирована и преобразована в валидатор.

Эта главная часть документа включает в себя два приложения:

- приложение А содержит the XML-схему для данной части ИСО 24610;
- приложение В содержит развернутый пример.

## 5 Базовые понятия

### 5.1 Рассматриваемые типизированные структуры элементов

Типизированные структуры элементов (TFS) вводятся как базовые записи для управления языковыми ресурсами.

Для получения более подробной информации следует обратиться к ИСО 24610-1:2006, пункт 4.7 *Типизированные структуры элементов* и приложение С *Типизированные иерархии наследования*.

В данном документе TFS определяется формально как кортеж на конечном множестве элементов **Feat**, который состоит из коллекции X элементов, не входящих в структуру, и иерархии типов **Type** с отношением $\leq$, где **Type** — это конечное множество типов, а отношение $\leq$определяет выделение подтипов на множестве **Type**.

Структура элементов представляет собой кортеж <Q, $\gamma$, $\theta$, $\delta$>, в котором:

a) Q — множество узлов,

b) $\gamma \in Q$ — корневой узел структуры элементов,

c) $\theta$: $Q \rightarrow$ **Type** является функцией частичного упорядочения, и

d) $\delta$: **Feat** $\times Q \rightarrow Q \cup X$ — функция частичного означивания элементов, такая, что для всех $q \in Q$ существует последовательность элементов $F_1, ..., F_n$, в которой $\delta[F_n, ... \delta(F_1, \gamma) ... ] = q$.

Обозначение <fs> показывает узлы. Приведенное выше определение отличается от стандартного, используемого в лингвистике и теории вычислительных систем тем, что во-первых типизация осуществляется частично, а не полностью (т. е. типы определяются не для всех структур элементов) и во-вторых значения элементов не обязательно должны представлять собой структуры элементов; однако эти

значения могут извлекаться из коллекции, отмеченной другими элементами XML, такими, как строковые, численные, символьные и двоичные (выше им соответствует обозначение X).

Узлы типизируются, тогда как сами элементы — нет.

Приведенное ниже XML-представление структуры элементов считается формально правильным; в нем атрибут «тип» указывается для каждого из двух элементов <fs>.

***Пример*** — Типизированная структура элементов:

```
<fs type="word">
  <f name="orth">
    <string>had</string>
  </f>
  <f name="morphoSyntax">
    <fs type="verb">
      <f name="tense">
        <symbol value="past"/>
      </f>
      <f name="auxiliary">
        <binary value="false"/>
      </f>
    </fs>
  </f>
</fs>
```

Имя элемента ORTH обозначает орфографию, т. е. общепринятое написание слова или фразы.

Данное XML-представление показывает, каким образом определяются морфосинтаксические характеристики английского слова «had» как невспомогательного глагола в прошедшем времени.

В альтернативной «матричной» или «AVM» нотации имена типов обычно пишутся строчными буквами, иногда курсивом или текстовым типографским шрифтом; имена элементов пишутся заглавными буквами, а строковые элементы заключаются в кавычки. Двоичные значения отмечаются знаками «плюс» (+) или «минус» (–). В данном документе эти соглашения тоже соблюдаются. Представленная выше структура элементов должна при использовании матричной нотации выглядеть так, как показано на рисунке 1.

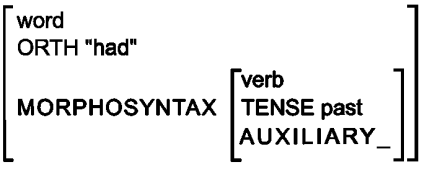



Рисунок 1 — Матричная нотация

### 5.2 Типы

#### 5.2.1 Атомарные типы

Наряду со структурами с встроенными элементами (<symbol>, <string>, <numeric> и <binary>) могут существовать структуры элементов, имеющие тип, но не имеющие элементов. Такие структуры называются простыми или атомарными структурами элементов, а типы, которые допускают отсутствие элементов в декларации системы элементов (FSD), именуются атомарными типами.

В результате всегда имеется возможность декларирования новых атомарных типов и использования их вместо вышеупомянутых встроенных элементов для задания простых значений. Например, приведенная выше структура элементов при условии декларирования в FSD дополнительных типов had, past и false могла бы быть представлена так, как показано ниже.

***Пример*** — Альтернативная формулировка типизированной структуры элементов:

```
<fs type="word">
  <f name="orth">
    <fs type="had"/>
  </f>
  <f name="morphoSyntax">
    <fs type="verb">
```

```
      <f name="tense">
        <fs type="past"/>
      </f>
      <f name="auxiliary">
        <binary value="false"/>
      </f>
    </fs>
  </f>
</fs>
```

Существует различие между двумя классами встроенных элементов: <string> (строковый) и <symbol> (символьный), <binary> (двоичный), <numeric> (численный). В качестве содержимого элемента <string> допустима любая строка, тогда как в элементах <symbol>, <binary> и <numeric> набор допустимых значений строго ограничен. Для отражения такого различия значения членов последнего класса определяются с использованием атрибута *value*. Тип <binary>, например, ассоциируется с четырьмя значениями: true (истина), false (ложь), plus (эквивалент true) и minus (эквивалент false).

Примечание — В ИСО 24610-1:2006 был введен тип *binary (двоичный)*, но в схеме W3C XML (2001) он называется *Boolean (булев)*.

Задача кодировщика состоит в том, чтобы осуществить правильный выбор между кодированием атомарных типов и встроенных элементов. В данной части ИСО 24610 различие между двумя вышеуказанными классами не проводится.

### 5.2.2 Составные типы

Типы, не являющиеся атомарными, называются составными. К ним относятся все типы, декларируемые кодировщиком в FSD, где объявляются или наследуются допустимые элементы. Элемент допустим для некоторого типа только в том случае, если структурам элементов данного типа декларацией FSD разрешается принимать те или иные значения. Из этого не следует, что структуры элементов не могут произвольно ассоциироваться с теми или иными типами независимо от их элементного наполнения. Такое ассоциирование возможно, но проверяться на адекватность FSD смогут лишь те структуры элементов, которые содержат только элементы, разрешенные какой-либо FSD. Различие между адекватностью и формальной правильностью рассматривается более подробно в разделе 6.

Все типы, декларируемые пользователем (независимо от того, атомарные они или составные) являются семантическими представлениями, т. е. синтаксически выглядят похожими друг на друга, если не принимать во внимание значения атрибутов типов. Интерпретация реального смысла этих типов посредством наложения ограничений по допустимости, ограничений на возможные значения разрешенных элементов (<vRange>) и прочих ограничений в виде логических импликаций — это задача валидатора.

Встроенные элементы, определенные для представлений структур элементов (FSR) в рамках ИСО 24610-1:2006, являются чисто синтаксическими, могут использоваться без декларирования в FSD, а потому их декларирование в FSD невозможно. Они могут появляться в ограничениях по диапазону значений или в импликативных ограничениях, однако сами не могут иметь таких ограничений (поскольку не имеют допустимых элементов) и сами не могут накладывать никаких ограничений.

### 5.2.3 Коллекции

Однако не все встроенные элементы столь просты, как элементы, отмеченные выше. Некоторые грамматические элементы — такие как спецификаторы (SPR), дополнения (COMPS) и аргументы (ARGS) — считаются обладающими списком грамматических значений, особенно в контекстных грамматиках [10, 12]. В других языках в отличие от английского некоторые из указанных элементов могут иметь в качестве своих значений другие коллекции: это могут быть простые множества или мультимножества. В языке с относительно произвольным порядком слов (например, в немецком, корейском или японском) элемент COMPS может анализироваться как принимающий значения множества или мультимножества, а не списка дополнений. Таким образом для приложений более общего характера в ИСО 24610-1:2006 вводятся в качестве встроенных методов компоновки значений составных элементов простые множества, мультимножества и списки.

**Коллекции** (<vColl>; ISO 24610-1:2006, п. 5.8, *Коллекции как значения составных элементов*) снабжаются атрибутом способа организации (org), который принимает значения «list», «set» и «bag». В списках важную роль играют порядок и многократность вхождения элементов.

В множествах с повторяющимися элементами важна только многократность вхождения элементов (такие множества часто называются мультимножествами). Применительно к обычным множествам ни порядок, ни многократность вхождения элементов не играют роли.

Например, элемент глаголов ARGS может представляться посредством определения способа организации коллекции <vColl> как списка значений, каждое из которых относится к типу *phrase*.

***Пример*** — Списковое значение

```
<fs type="word">
 <f name="orth">
  <string>put</string>
 </f>
 <f name="args">
  <vColl org="list">
   <fs type="phrase">
    <vLabel name="L1"/>
    <f name="nominal">
     <binary value="plus"/>
    </f>
   </fs>
   <fs type="phrase">
    <vLabel name="L2"/>
    <f name="nominal">
     <binary value="plus"/>
    </f>
   </fs>
   <fs type="phrase">
    <vLabel name="L3"/>
    <f name="prepositional">
     <binary value="plus"/>
    </f>
   </fs>
  </vColl>
 </f>
</fs>
```

Этот тип коллекций можно было бы отнести к *списковым* [*list (phrase)*], однако полиморфные списки пока еще не поддерживаются данной частью ИСО 24610. Рассмотренный тип эквивалентен приведенной ниже нотации AVM, NP обозначает структуру элементов типа *phrase* с положительным элементом NOMINAL, а конкретней — именную группу, а PP соответствует структуре элементов тип *phrase* с положительным ПРЕДЛОЖНЫМ элементом, а именно — предложной группе. Числа в прямоугольниках являются пометами для разметки совместного использования структуры, как показано на рисунке 2.

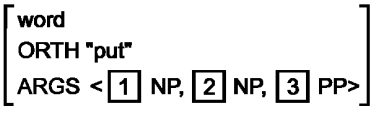



Рисунок 2 — Разметка совместного использования структуры

### 5.2.4 Операторы

Еще один класс встроенных элементов — это операторы, которые принимают один или несколько встроенных элементов или структур элементов в качестве своих аргументов, но вместо конструирования из них коллекции указывают некоторое значение, получаемое на их основе тем или иным методом.

**Дизъюнкции** (<vAlt>; ИСО 24610-1:2006, пункт 5.9.2) указывают одно из значений их аргументов. Однако структура элементов, содержащая дизъюнкцию, не может представлять структуры множественного типа. Дизъюнкция — это единственное значение, которое не определяет точно конкретный вариант из числа возможных. Дизъюнкции могут рассматриваться как объединения их аргументов в рамках частичного порядка, установленного категоризацией (см. 5.6).

**Отрицания** (<vNeg>; ИСО 24610-1:2006, пункт 5.9.3) имеют единственный аргумент и указывают значение, которое не является их аргументом. Отрицание эквивалентно дизъюнкции всех значений, которые не соответствуют его аргументу. Фактически отрицание не является логической функцией отрицания конкретного значения, а скорее представляет собой дополнение того значения в полной булевой решетке, которое содержит частичный порядок, установленный категоризацией.

**Слияние** (<vMerge>; ИСО 24610-1:2006, пункт 5.9.4, *Коллекция значений*) указывает конкатенацию или объединение нескольких значений и/или коллекций значений в соответствии с настройкой их атрибута *org*. Этот атрибут принимает те же значения и тот же смысл, которые содержатся в <vColl>.

### 5.3 Иерархии наследования типов

Иерархия типов <Type, ⊴> достаточно подробно рассматривается в приложении С ИСО 24610-1:2006. Эта структура обычно отображается как ориентированный ациклический граф с единственной вершиной. Данная вершина часто имеет метку **top** и представляет самый общий тип, который совместим со всеми типизированными структурами элементов. Подтипы соединяются со своими супертипами и располагаются уровнем ниже. Максимально конкретизированные типы появляются в самом низу графа. Они взаимно несовместимы друг с другом, что обычно бывает либо абсолютно ясно, либо иногда отображается другим конкретизированным типом (**bottom**), который является единственным самым нижним элементом. В рамках данной части ИСО 24610 тип **bottom** не используется.

На рисунке 3 показан пример, иллюстрирующий частичную иерархию типов для живой природы.

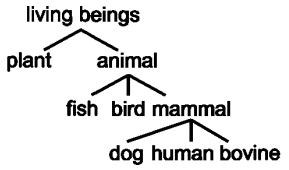



Рисунок 3 — Иерархия типов для живой природы

В соответствии с этим рисунком живая природа (**living beings**) разделяется на растительность (**plant**) и животный мир (**animal**). Далее животные разбиваются на классы рыб (**fish**), птиц (**bird**) и млекопитающих (**mammal**). Собаки (**dog**), люди (**human**) и крупный рогатый скот (**bovine**) — вол, корова, бык — принадлежат к классу млекопитающих.

Иерархии типов не всегда имеют древовидную структуру; в схеме может быть два или больше ответвлений, сходящихся в одном узле. Когда такое случается, это означает, некоторый тип имеет несколько супертипов и свойства, наследуемые от всех них. Пример подобной иерархии приведен на рисунке 4.

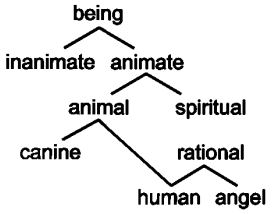



Рисунок 4 — Средневековая иерархия живых существ

Здесь тип *human* (*человек*) имеет два родительских типа: animal (животное) и rational (разумное существо). Следовательно, человек рассматривается одновременно и как животное (подобно собаке) и как мыслящее существо.

Эти типы частично упорядочены с помощью отношения выделения подтипов $\leq$ на множестве всех типов. Тип $\tau$ является подтипом по отношению к типу $\sigma$ тогда и только тогда, когда $\sigma$ имеет более общий характер по сравнению с $\tau$, т. е., когда множество структур элементов типа $\sigma$ содержит в себе множество структур элементов типа $\tau$. Так как тип *animate* (*одухотворенный*) является в приведенном выше примере более общим по отношению к типу *animal*, все животные определяются как одушевленные. Тип $\sigma$ считается супертипом типа $\tau$ тогда и только тогда, когда $\tau$ является подтипом $\sigma$.

Непосредственные супертипы какого-либо типа часто называются его родителями.

Подтип наследует все свойства от своего супертипа. Например, тип *human* наследует все свойства от своих супертипов (каковыми являются *being*, *animate*, *animal*, *spiritual* и *rational*).

На рисунке 5 приведен несколько измененный лингвистический пример из грамматики Коупстейка [2].

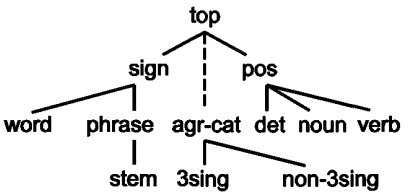



Рисунок 5 — Иерархия типов для вершины простой грамматики

Данная иерархия типов имеет единственную вершину. Это самый общий тип, не имеющий ни родителей, ни непосредственных супертипов. Тип *top* — это также единственный подтип самого себя.

Каждый тип имеет имя и у каждого типа за исключением наивысшего имеется один родитель. У типа с именем *top* есть четыре непосредственных подтипа. Подтипы *phrase* (*речевой оборот*) и *det* (*определяющее слово*) — не сопоставимы в том смысле, что ни один из них не является подтипом другого.

В зависимости от степени сложности грамматики иерархия типов может оказаться очень сложной. Некоторые ее участки могут быть универсальными для всех языков, тогда как другие могут быть очень специфичными для конкретного языка. Так тип соглашения *agr-cat* (*соглашение по категоризации*) в английском языке имеет только два непосредственных подтипа: *3sing* и *non-3sing* (например, «sings» и «sing»).

Тип *det* обозначает определяющее слово (determiner), такое как артикли «the» или «a»; *3sing* указывает на 3-е лицо единственного числа, а *non-3sing* указывает категории соглашений, отличные от *3sing*.

Это различие характерно для правил согласования глаголов в английском языке.

### 5.4 Ограничения для типов

Иерархия типов представляет собой основу, на которой строятся все остальные разделы грамматики, которые принимают форму ограничений для структур элементов на множестве пользовательских типов. Такие ограничения бывают, как минимум, трех видов: 1) импликативные, 2) по разрешенным элементам и 3) по допустимым значениям элементов. Все они могут быть выражены в импликативной форме:

- если структура элементов относится к типу *verb*, то она может иметь элемент AUXiliary,
- если структура элементов относится к типу *verb*, она может иметь элемент INVerted,
- если структура элементов относится к типу *verb*, то ее значением AUX должно быть “binary”,
- если структура элементов относится к типу *verb*, то ее значением INV должно быть “binary”,
- если структура элементов относится к типу *verb* и ее значение AUX отрицательно, то ее значением INV должно быть “negative”.

Первые два из этих ограничений являются ограничениями по допустимости. Они говорят о том, что конкретный элемент может использоваться в структурах элементов определенного типа. Следующая пара ограничений касается значений допустимых элементов и называется иногда «ограничениями по значению» или «ограничениями по диапазону». Они говорят о том, какие значения должен принимать конкретный элемент, когда он входит в структуру элементов данного типа. Последнее из ограничений имеет наиболее общую форму, однако этот вид ограничений говорит о том, что когда структура элементов приобретает некоторую конкретную форму (определяемую типами, значениями элементов и т. п.), она должна удовлетворять каким-то другим критериям (опять же выраженным в терминах типов, значений элементов и т. п.). Эта последняя форма ограничения обычно представляет собой то, что подразумевается под импликативным ограничением синтаксической конструкции. Каждая из этих трех форм имеет свою синтаксическую структуру в FSD. Далее показан пример кодирования вышеуказанных ограничений применительно к глаголу.

***Пример*** — Ограничение для типа *verb*

```
<fsDecl type="verb">
 <fDecl name="aux">
  <vRange>
   <binary/>
  </vRange>
 </fDecl>
 <fDecl name="inv">
  <vRange>
   <binary/>
```

```
    </vRange>
   </fDecl>
   <fsConstraints>
    <cond>
     <fs>
      <f name="aux">
       <binary value="false"/>
      </f>
     </fs>
     <then/>
     <fs>
      <f name="inv">
       <binary value="false"/>
      </f>
     </fs>
    </cond>
   </fsConstraints>
  </fsDecl>
```

Два первых вида определяются вместе внутри элемента <fDecl>, причем второй из них описывается частью <vRange> указанной декларации, тогда как третий определяется в форме условной конструкции «если..., то...» (<cond>).

### 5.5 Опциональные (стандартные) значения и недоопределение

Некоторые элементы, образующие структуру, подлежат обязательному определению, а некоторые — нет. Так во французском языке спецификация элементов NUMBER (ЧИСЛО) и GENDER (РОД) обязательна для имен существительных и прилагательных, а в английском языке элемент NUMBER должен определяться для каждого существительного, а определение элемента GENDER — не обязательно и требуется только для местоимений третьего лица единственного числа «he», «she» и «it».

Тем не менее встречаются случаи, когда некоторые обязательные элементы не определяются. Для таких случаев имеются два вероятных исхода: 1) если определено стандартное значение по умолчанию, то считается, что именно оно и должно быть присвоено, и 2) если значение по умолчанию не определено, то присваиваемое значение элемента выводится логически из действующего ограничения элемента по диапазону значений.

Английские неисчисляемые существительные, такие как «вода» и «воздух», по умолчанию определяются как несчетные и не имеющие множественного числа. Отсюда следует, что для них не требуется определения элемента NUMBER, хотя сам элемент NUMBER обязателен. В английском языке некоторые исчисляемые существительные (например, «sheep») могут иметь одну и ту же форму в единственном и множественном числе. Когда элемент NUMBER не определен, считается, что его значение относится к некоторому более общему типу, такому как *number*, который является супертипом всех значений разрешенных элементов.

Грамматические описания часто бывают недоопределенными в целях обеспечения возможности обобщения. Так в английском языке глаголы разделяются при необходимости на ряд дополнительных категорий; непереходные глаголы — например, «smile» (улыбаться) и «bark» (лаять) — присоединяются только к подлежащему; переходные глаголы — такие как «love» (любить) и «attack» (атаковать, нападать) — присоединяются только к подлежащему и требуют за собой прямого дополнения. Есть еще и дитранзитивные («дважды транзитивные») глаголы — например, «give» (давать), «put» (класть), которые имеют при себе подлежащее и одновременно — прямое и косвенное дополнения. Однако многие грамматические явления не относятся ни к одному из перечисленных выше специфических подклассов. В качестве примера подобных явлений в английском языке можно привести согласование подлежащего с глаголом (правильную форму «The dog barks» и неправильную «the dog bark») или инверсию глагольной формы посредством ее вынесения в позицию перед подлежащим («Does the dog bark?» в противоположность неверной форме «Do the dog attacks Jane?»). Поскольку спецификация данного элемента не дает описания вышеуказанных грамматических явлений, он остается недоопределенным.

Ниже приводится еще один пример недоопределения. Анализ предложения типа «The sheep attacked Jane» может оказаться недоопределенным в части значения NUMBER для элемента «sheep». Неоднозначность этого элемента отмечается явным образом лишь в случае особой необходимости.

Значения по умолчанию определяются в FSD с помощью элемента <vDefault>, как объясняется в 8.4, и могут быть получены из FSR с использованием элемента <default> (ИСО 24610-1:2006, пункт 5.10).

### 5.6 Категоризация

Структура элементов $F$ категоризирует другую такую структуру $G$ ($F \subseteq G$) тогда и только тогда, когда $G$ содержит в себе всю информацию, имеющуюся в структуре $F$. «Информация» предоставляется структурой элементов двумя путями: посредством типизации элементов и посредством уравнивания маршрутов. Если рассматривать структуры элементов как пары маршрутов, связанные отношением эквивалентности ($\equiv$), и как функцию частичной типизации на множестве маршрутов ($\Theta$), то формально $< \equiv_F, \Theta_F > \subseteq < \equiv_G, \Theta_G >$ тогда и только тогда, когда $\equiv_F \subseteq \equiv_G$; при этом, если для всех $\pi \in \text{Paths}_F \cap \text{Paths}_G$ определено $\Theta_F(\pi)$, то $\Theta_G(\pi)$ определено и является подтипом $\Theta_F(\pi)$. Когда $F \subseteq G$, говорят, что $G$ расширяет $F$.

Представление типизированной структуры элементов в настоящем документе имеет более общий характер по сравнению с представлениями, часто фигурирующими в лингвистической литературе и в теоретических публикациях по вопросам логики типизации элементов. Это имеет место в силу того, что в нашем случае присутствуют символьные, строковые, численные значения элементов и значения, отличные от элементов <fs>. Что же касается расширений и категоризации, то строковые, символьные, численные и булевые (двоичные) элементы ведут себя так, будто они являются типами, не содержащими разрешенных элементов, которые одновременно частично упорядочены, но никак не связаны с остальной частью иерархии наследования. Иначе говоря, они не связаны отношениями выделения подтипов ни с какими другими типами, кроме самих себя. Структуры элементов таких «типов» категоризируются только ими самими и наиболее общей не типизированной структурой <fs/>, и они не имеют никаких других расширений, отличных от них самих. Следует соблюдать осторожность в отношении определения категоризации в рамках расширенного представления типизированных структур элементов, так как между сходными по виду символами, строками, числами и т. п. все же могут существовать или отсутствовать какие-то связи. Практикуемое более широкое рассмотрение аспектов идентичности таких объектов оказывается несовместимым с тем представлением идентичности, которое обеспечивает логика типизации структур элементов на множестве их собственных типов; а именно такой подход используется в данной части ИСО 24610 как для структур элементов, так и для других объектов, когда они встречаются в рамках структур элементов.

В рамках логики типизации элементов зачастую исключаются из детализированной формализации еще и дизъюнкции, однако они могут трактоваться как объединения аргументов соответствующих типизированных структур в рамках отношения частичного порядка на множестве типизированных структур элементов, порожденных категоризацией. Аналогично отрицание значения может рассматриваться как объединение всех структур, которые не соответствуют отрицаемому значению при выполнении операции унификации. Коллекции обычно зависят от способа организации структур элементов. Списки выглядят при частичном порядке категоризации так, как если бы они кодировались как типизированные структуры элементов с использованием FSD.

***Пример*** — Фрагмент FSD

```
<fsDecl type="list" baseTypes="top"/>
<fsDecl type="e-list" baseTypes="list">
 <fsDescr>Empty lists</fsDescr>
</fsDecl>
<fsDecl type="ne-list" baseTypes="list">
 <fsDescr>Non-empty lists</fsDescr>
 <fDecl name="first"/>
 <fDecl name="rest">
  <vRange>
   <fs type="list"/>
  </vRange>
 </fDecl>
</fsDecl>
```

Одно множество с повторяющимися элементами (мультимножество) $B_1$ категоризирует другое мультимножество $B_2$ тогда и только тогда, когда существует общая сюръекция $\sigma$ между элементами двух мультимножеств, такая, что для всех $b_1$ в области $B_1$ с кратностью $\mu_1(b_1)$ и для всех $b_2$ в области $B_2$ с кратностью $\mu_2(b_2)$ выполняются следующие условия:

1) $b_1 \subseteq \sigma(b_1)$,

2) $\mu_2(b_2) = \Sigma\mu_1(b_1)$,

$b_1 : \sigma(b_1) = b_2$,

а $\sigma$ можно расширить до полной функции $\sigma^*$, связывающей подструктуры элементов двух мультимножеств таким образом, что для всех подструктур $c$ элементов из $B_1$:

3) $\sigma^*(c) = \sigma(c)$, если $c$ является элементом $B_1$ и

4) $\sigma^*[\delta(F, c)] = \delta[F, \sigma^*(c)]$ для каждого $F \in$ **Feat**, такого, что значение $\delta(F, c)$ определено.

Аналогично одно множество $S_1$ категоризирует другое множество $S_2$ тогда и только тогда, когда применимы условия 1), 3) и 4). Это означает, например, что двухэлементное множество $\{F_1, F_2\}$ категоризирует одноэлементное множество $\{G_1\}$, если одновременно $F_1 \subseteq G_1$ и $F_2 \subseteq G_1$. Такая частично упорядоченная интерпретация множеств называется теорией множеств Полларда — Мошайра (Pollard-Moshier), которая наиболее популярна в логике типизации элементов.

Кроме того, мультимножество категоризирует любой список, который является перестановкой его элементов. Множество категоризирует мультимножество, если область мультимножества является обычным множеством, т. е. все без исключения элементы множества появляются в мультимножестве один раз или многократно.

Комбинация коллекций (<vMerge>) занимает в частичном порядке категоризации то же положение, что и результат конкатенации или объединения, который эта комбинация определяет вместе с методом организации, если таковой необходим.

Рефлексивное или транзитивное замыкание всех этих условий порождает отношение категоризации, фигурирующее в данной части ИСО 24610.

## 6 Определение формальной правильности и адекватности

### 6.1 Общее описание

#### 6.1.1 Общие замечания

В данном разделе проводится различие между использованием понятий «формальная правильность» и «адекватность», поскольку эти понятия имеют отношение к представлениям структур элементов и к системам элементов. В теоретической лингвистике, даже в ее частях, якобы основанных на использовании логики типизированных элементов, они часто используются как синонимы или в значениях, отличных от их традиционного понимания в формальной логике и в XML. Использование вышеуказанных понятий в формальной логике и в языке XML тоже различно. Поэтому прежде чем приступить к определению этих понятий, целесообразно дать краткий обзор трактовок рассматриваемых понятий в двух указанных областях.

#### 6.1.2 Формальная логика

В формальной логике понятия формальной правильности и адекватности четко разграничиваются. Формальная правильность — это синтаксическая концепция, тогда как адекватность — понятие семантическое. Цепочка символов в логике считается формально правильной, если она определяется с помощью набора правил ее формирования. В логике первого порядка, например, последовательность символов $\forall x\,[H(x) \rightarrow [G(x) \rightarrow H(x)]]$ считается формально правильной формулой, в которой $\forall$ — это квантор всеобщности, $x$ — отдельная переменная, стрелка $\rightarrow$ соответствует двоичному пропозициональному оператору, $G$ и $H$ — символы одноместного предиката, а все скобки обеспечивают должное согласование. В то же время символ $\forall x$ сам по себе формально неправилен, поскольку правило построения синтаксических конструкций с кванторами требует, чтобы за каждым квантором с переменной следовало формально правильное выражение. Таким образом, в данном случае правила построения синтаксических конструкций вычленяют множество формально правильных формул из всего множества произвольных строковых записей.

Далее семантические правила логики первого порядка обеспечивают интерпретацию этих формально правильных формул посредством оценки их значений истинности. Поскольку логика первого порядка бивалентна, каждая формула, содержащая в себе атомарные формулы, истинна или ложна относительно некоторой интерпретации (или модели) и, возможно, относительно присваивания значений переменным в случае так называемых открытых формул наподобие $G(x)$ и $H(x)$. Формула $G(x)$ справедлива относительно некоторой модели и некоторого присваивания значений тогда и только тогда, когда значение, присваиваемое переменной $x$, принадлежит множеству возможных значений $G$ рассматриваемой модели. Допустим, что $x$ — это Джейн, а $G$ — это множество девушек. Тогда выражение $G(x)$ истинно в предположении, что Джейн — девушка. Однако формула $\forall x[H(x) \rightarrow [G(x) \rightarrow H(x)]]$ справедлива всегда относительно любой модели или любого присваивания значений, потому что данная формула есть одна из форм описания тавтологии $[p \rightarrow [q \rightarrow p]]$ в логике высказываний. Такая формула называется адекватной. В общем случае формально правильная формула считается адекватной, если она справедлива для всякой интерпретации/модели. Одна из семантических задач в логике состоит в том, чтобы выделить все без исключения адекватные формулы из тотального множества формально правильных формул.

### 6.1.3 Язык XML

В языке XML тоже проводится четкое различие между двумя рассмотренными выше понятиями. Документы на языке XML могут быть формально правильными или неправильными (недействительными), и тогда формально правильные документы могут оказаться адекватными либо неадекватными (недействительными). Как и в большинстве других языков разметки документов, формально правильный XML-документ должен отвечать требованиям ряда правил, касающихся единственности корневого элемента, правильности вложенных структур и согласованности объектов документа. Критерии адекватности XML-документов установить труднее, однако прежде всего адекватные документы должны быть хорошо согласованными и выверенными на соответствие всем ограничениям (правилам), устанавливаемым далее грамматикой документов, например внутренними или внешними правилами определения типов документов [Document Type Definition (DTD)], схемой XML и форматом RELAX NG либо каким-то иным форматом. Неадекватные XML-документы называются также недействительными.

## 6.2 О стандарте ИСО 24610

### 6.2.1 Определения

В ИСО 24610 «формальная правильность» и «адекватность» рассматриваются как принципиально разные понятия в форме, которая в общем аналогична структурам языка XML, но роль DTD в ней играют декларации систем элементов, определяемые данным стандартом:

- представление структуры элементов формально правильно тогда и только тогда, когда оно соответствует определениям деклараций структур элементов и правилам типизации, определенным в ИСО 24610-1:2006 или его более поздних версиях;

- следствие: каждое представление структуры элементов на языке XML должно отвечать условиям формальной правильности XML-документов в части наличия единственного корневого элемента, соответствующей структуры вложений и согласованности элементов;

- представление структуры элементов на языке XML адекватно тогда и только тогда, когда оно формально правильно и соответствует системе элементов, декларированной (в DTD, схеме XML или в каком-то ином формате) для конкретного приложения, в котором используются типизированные структуры элементов.

Формально правильная структура элементов считается типизированной в отношении согласованной (в смысле XML) декларации FSD тогда и только тогда, когда каждый элемент <fs>, содержащийся в ней (включая и саму декларацию, если она является элементом <fs>), несет в себе значение атрибута для типа, явно декларированного в каком-либо элементе <fsDecl> декларации системы элементов FSD. Каждое типизированное представление структуры элементов однозначно указывает типизированную структуру элементов. В настоящем стандарте понятия «типизированная структура элементов» и «представление типизированной структуры элементов» используются как синонимы, но не все структуры элементов в ИСО 24610-1:2006 являются типизированными; между тем понятия «адекватность» и «проверка адекватности» в отношении FSD имеют смысл только применительно к типизированным представлениям структур элементов.

Типизированная структура элементов может быть формально неправильной по нескольким причинам. Для обеспечения ее адекватности необходимо прежде всего, чтобы она была согласованной и удовлетворяющей как условиям формальной правильности для языка XML, так и определениям структуры и типизации элементов. Кроме того, значения элементов структуры должны находиться в пределах декларированных допустимых диапазонов. Наконец, типизированная структура должна удовлетворять действующим ограничениям ее типов и ограничениям, наследуемым от базовых типов.

В отличие от языковой среды XML типизированные структуры элементов частично упорядочены с использованием отношения категоризации (см. 5.6). Говорить о категоризации несогласованных (формально неправильных) представлений структур элементов просто не имеет смысла, но она приобретает смысл применительно к неадекватным представлениям структур элементов, если они типизированы. Адекватные типизированные структуры элементов способны категоризировать неадекватные и наоборот, и поэтому адекватность и категоризация не могут использоваться для взаимного построения каких-либо логических выводов относительно друг друга, но и сама потребность в простом установлении факта адекватности типизированных структур элементов возникает довольно редко. Вместо процедуры определения адекватности почти всегда производится поиск адекватного расширения (3.11) типизированной структуры элементов. Каждая адекватная типизированная структура элементов абсолютно точно имеет хотя бы одно адекватное расширение, а именно — самое себя. Некоторые неадекватные типизированные структуры элементов не имеют адекватных расширений, но те из них, у которых такое адекватное расширение есть, являются уникальными и наиболее общими (или, соответственно, наименее информативными). Это наиболее общее адекватное расширение используется в качестве

программного агента (proxy) для всего множества адекватных расширений, которые категоризируются адекватной или неадекватной типизированной структурой элементов.

В лингвистической литературе часто можно встретить и другие категории типизированных структур элементов. Особенно примечательны структуры элементов с тотальной соподчиненностью, в которых каждый обязательный элемент принимает значение из строго определенного диапазона (<vRange>), что равносильно обеспечению адекватности деклараций FSD независимо от ограничений вида <cond> или <bicond>. Кроме того, объекты, называемые здесь *представлениями структур элементов*, во многих отношениях больше совпадают с тем, что специалисты по прикладной лингвистике называют *дескрипторами структур,* чем со структурами элементов. Дескрипторный язык, используемый в большинстве лингвистических приложений типизированных структур элементов, достаточно консервативен для того, чтобы его можно было легко встраивать в представления структур элементов, охватываемые ИСО 24610-1:2006; подавляющее большинство таких структур составляет основу грамматики фразовых категорий, управляемых вершинами (HPSG — Head-driven Phrase Structure Grammar) [10]. Однако в указанном стандарте есть представления структур элементов (FSR), для которых соответствие единственному эквивалентному описанию в лучшем случае весьма расплывчато из-за наличия их зависимости от FSD, применительно к которым такая эквивалентность может быть доказана.

### 6.2.2 Анализ синтаксиса типизированной структуры элементов в XML

6.2.2.1 Общие сведения

Обзор представлений синтаксиса типизированной структуры элементов дается ниже в привязке к соответствующим именам элементов и их шаблонам.

6.2.2.2 Введение имен

a) имена структурных элементов и их свойств: fs, f
b) имена значений элементов: (fs), string, symbol, binary, numeric, vLabel
c) имена атрибутов элементов: name, type, org, value
d) имена конструкторов коллекций: vColl
e) имена элементов-операторов: vAlt, vNot, vMerge, default

6.2.2.3 Базовая модель

```
<fs type="Type">
 <f name="featureName">
  <fs type="featureValueType">VALUE</fs>
 </f>
</fs>
```

6.2.2.4 Модели значений элементов

a) для типов значений атомарных элементов:

```
<fs type="atomicType"/>
```

b) для значений структур элементов:

```
<fs type="featureValueType">VALUE</fs>
```

c) коллекции:

```
<vColl org="collectionType">
 <fs type="Member1Type">VALUE1</fs>
 <fs type="Member2Type">VALUE2</fs> ...
</vColl>
```

d) дизъюнкция:

```
<vAlt>
 <fs type="Disjunct1Type">VALUE1</fs>
 <fs type="Disjunct2Type">VALUE2</fs> ...
</vAlt>
```

e) отрицание (взятие дополнения)

```
<vNot>
 <fs type="NegatedValue">VALUE</fs>
</vNot>
```

или

```
<vNot>
 <vAlt>
  <fs type="NegatedValue1">VALUE1</fs>
  <fs type="NegatedValue2">VALUE2</fs>
 </vAlt>
</vNot>
```

***Пример***:

```
<fs type="pos">
 <f name="agr">
  <fs type="agr-cat">
   <f name="per">
    <vNot>
     <fs type="3rd"/>
    </vNot>
   </f>
   <f name="num">
    <vNot>
    <fs type="singular"/>
   </vNot>
  </f>
 </fs>
</f>
</fs>
```

### 6.2.3 Иллюстративные примеры формальной правильности

Структура элементов — это частично рекурсивная функция из элементов в их значения. Отсюда следует, что представление fs/ допустимо (как представление пустой структуры элементов), но представление, которое определяет элементы без значений, недопустимо.

***Пример 1*** — Формальная правильность

a) **<fs type**="top"/>

b) **<fs type**="TYPE">
**<f name**="FEATURE"/>
<!-- WRONG -->
**</fs>**

Здесь: а) — формальная правильность, б) — несогласованность.

Тип присваивается каждой структуре элементов или каждому значению элемента, но не самому элементу. Следовательно, элемент с именем “f” не может иметь атрибут с именем “type”.

***Пример 2*** — Формальная правильность:

a) **<fs**/>

b) **<fs type**="top"/>

c) **<fs type**="TYPE1">
**<f name**="FEATURE" **type**="TYPE2">
**<fs type**="top"/>
**</f>**
<!--WRONG -->
**</fs>**

d) **<fs type**="TYPE1">
**<f name**="FEATURE">
**<fs type**="TYPE2"/>
**</f>**
**</fs>**

Здесь:

а) формально правильное представление — самое общее представление нетипизированной структуры элементов;

b) тоже формально правильное представление; *top* обычно считается самым общим типом, что делает данное представление наиболее общим, хотя в рамках данного стандарта оно не требуется, и тип *top* не является встроенным — он должен декларироваться только в случае необходимости его использования;

c) несогласованное представление, потому что данный тип присваивается элементу;

d) согласованное представление, так как значение элемента может типизироваться.

### 6.2.4 Иллюстрация адекватности

6.2.4.1 Условия

Условия адекватности зависят от конкретной системы элементов, содержащей ограничения типов. Спецификация этих ограничений будет рассмотрена в 6.2.4.3, а ниже представлена типизированная структура элементов, которая является адекватной относительно этой спецификации FSD.

6.2.4.2 Иллюстративный пример адекватности

```
<fs type="word">
 <f name="orth">
  <string>Mia</string>
 </f>
 <f name="head">
  <fs type="noun">
   <f name="agr">
    <fs type="agr-cat">
     <f name="person">
      <fs type="3rd"/>
     </f>
     <f name="number">
      <fs type="singular"/>
     </f>
    </fs>
   </f>
  </fs>
 </f>
 <f name="spr">
  <vColl org="list"/>
 </f>
 <f name="comps">
  <vColl org="list"/>
 </f>
</fs>
```

Соответствующая данному представлению нотация AVM показана на рисунке 6.

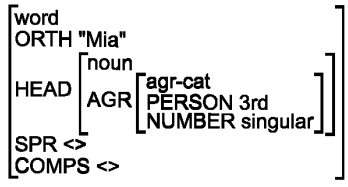



Рисунок 6 — Соответствующий формат AVM

Примечание — Здесь, как и раньше, именам элементов *orth*, *head*, *agr*, *spr* и *comps* из представления XML соответствуют имена *ORTH*, *HEAD*, *AGR*, *SPR* и *COMPS* в представлении AVM.

Пример адекватной типизированной структуры элементов приведен на рисунке 6. Данная структура TFS адекватна, так как удовлетворяет всем ограничениям для типа *word*. Ниже представлена самая общая адекватная типизированная структура элементов данного типа, которая категоризирует TFS, отображенную на рисунке 6.

6.2.4.3 Общее ограничение для типа *word*

```
<fs type="word">
 <f name="orth">
  <fsval kind="string"/>
 </f>
```

```
<f name="head">
 <fs type="pos"/>
</f>
<f name="specifier">
 <vColl org="list"/>
</f>
<f name="complements">
 <vColl org="list"/>
</f>
</fs>
```

Соответствующая структура AVM показана на рисунке 7.

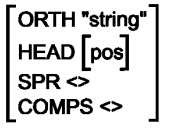



Рисунок 7 — Соответствующая структура AVM

Элемент за элементом можно проследить, каким образом показанная структура элементов категоризирует TFS, приведенную в 6.2.4.2. В этой TFS строковое значение “Mia” присваивается элементу ORTH; тип *noun* является подтипом *pos* для элемента HEAD, а пустой список представляет собой разновидность списковой коллекции одновременно для элементов SPEC и COMPS.

## 7 Система элементов для грамматики

### 7.1 Общие сведения

Типизированные структуры элементов очень широко используются для разработки грамматик, лексики и других лингвистических ресурсов и приложений. В одной из реализаций грамматики лингвистических ресурсов английского языка (ERG) [3] все компоненты, образующие грамматику, включая определение типов, правила построения речевых оборотов и лексические единицы, представляются структурами элементов. При таком подходе каждая из структур элементов должна быть уникально расширяемой до наиболее общей структуры, которая удовлетворяет сопутствующей системе типов, образованной иерархией типов и множеством ограничений типов. Как будет видно из 8.4, некоторые типы могут также ассоциироваться с коллекцией разрешенных элементов и их допустимыми значениями, равно как с импликативными ограничениями более общего характера. Поскольку значения элементов могут быть обязательными (требуемыми), опциональными (факультативными) или стандартными (присваиваемыми по умолчанию), эти различия тоже подлежат отражению в системе элементов.

Декларации допустимости вместе с конкретными отношениями выделения подтипов, образующих иерархию типов, могут быть выражены и часто выражаются в виде типовых структур элементов с помощью контекста, который указывает, что эти декларации должны рассматриваться как утверждения относительно адекватности, а не просто как данные. При определенных синтаксических ограничениях, суженных до набора допустимых ограничений настоящего стандарта (которые можно наблюдать при работе с ERG и ее производными фрагментами), импликативные ограничения типов могут кодироваться и как структуры элементов объектного уровня, т. е. как структуры элементов, выведенные из той же системы типов, что и базовая грамматика. В условиях соглашений ERG все ограничения типов представляют собой односторонние импликации (<cond>), в которых единственным предшествующим членом отношения (антецедентом) является только тип. Кроме того, и лексикон, и правила построения речевых оборотов в грамматике могут рассматриваться как ограничения типов на уровне объектов путем использования дизъюнкций (ИСО 24610-1:2006, пункт 5.9.2); в противном случае они оказываются для ERG нераспознаваемыми.

Однако даже без привлечения рассмотренных выше предположений относительно соглашений и ограничений применительно к каждому компоненту грамматики всегда доступно представление метауровня как структуры элементов. В случае использования правил построения фраз для кодирования такого представления используются дополнительные типы и элементы: например, конституента в левой или правой части соответствующего правила. Правило составления заголовка, которое поглощает один аргумент заголовка фразы, можно закодировать, например так, как показано на рисунке 8.

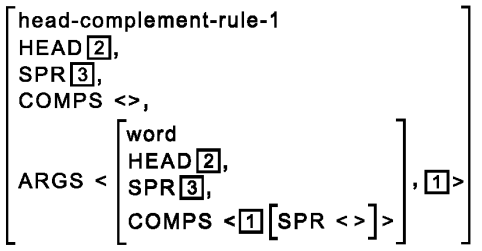



Рисунок 8 — Правило 1 для составления заголовка

В данном случае для представления тождества и конституент в правой части этого правила предназначены соответственно дополнительный тип (head-complement rule-1) и элемент (ARGS). Дополнительный тип, рассматриваемый как импликативное ограничение, может также трактоваться как логическое условие (антецедент), а все остальное — как следствие конъюнкции. Здесь используются также списки (обозначенные угловыми скобками), являющиеся одной из составных коллекций ИСО 24610-1:2006, описанных в пункте 5.8 *Коллекции как значения составных элементов*, хотя в объектно-ориентированной системе типов грамматики они могут не играть никакой роли. В публикациях по лингвистике такие методы кодирования могут оказаться достаточными, однако не стоит заблуждаться по поводу реальных возможностей отступления от формальных правил. В соответствии с настоящим стандартом для декларирования системы типов, разрешенных элементов, ограничений допустимых значений элементов и импликативных ограничений типов в грамматике должны использоваться определенные здесь декларации систем элементов (FSD), а не просто элементы <fs> из ИСО 24610-1:2006. Всегда возможны и методы метауровневого кодирования, но такие возможности в данном стандарте не рассматриваются. Его основная цель — показать прямое назначение FSD, т. е. продемонстрировать, каким образом информация деклараций должна использоваться приложением, в рамках которого человек может не участвовать в процедуре формулирования логических выводов. Следовательно, для соответствия настоящему стандарту грамматика должна определяться, как минимум, следующими компонентами: 1) декларацией системы элементов (иерархии типов, декларации допустимости и ограничения типов), 2) лексиконом и 3) коллекцией правил построения фразовых структур. Выполнение первого требования обеспечивается обязательным использованием FSD. Для обеспечения соответствия требованиям 2) и 3) рекомендуется кодировать лексические единицы и коллекции правил с помощью синтаксических обозначений, принятых в других официальных документах, где их статус как лексем или продукционных правил указан явным образом и однозначно; однако допустимо также использование FSD.

### 7.2 Выборочные FSD

#### 7.2.1 Общие замечания

В данном подразделе показывается для примера модифицированная декларация системы элементов грамматики Grammar 2 из работы Коупстейка [2].

#### 7.2.2 Определение типов и их иерархии

Типы определяются посредством указания их родительских супертипов. Иерархии типов не всегда имеют древовидную структуру (так как подтип может иметь и больше одного родительского супертипа), однако существуют ограничения, накладываемые на их форму: например требование единственности самого общего типа.

Пример декларации типов элементов для иерархии типов приведен на рисунке 9.

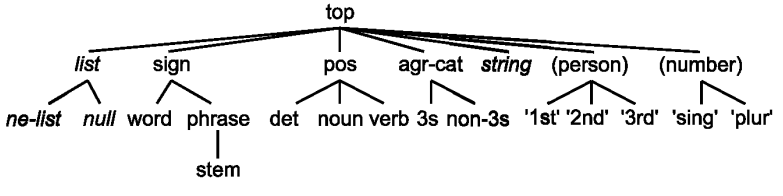



Рисунок 9 — Простая иерархия типов для английского языка

Каждая ветвь этого дерева просто указывает на экземпляр отношения «подтип — супертип»; обозначение *top* или *T* часто используется в качестве имени типа, присваиваемого каждой структуре элементов. Выделенные курсивом типы *list* и *string* являются встроенными элементами, которые в согласованной FSD (показанной ниже) не должны присутствовать явным образом. Типы, заключенные в скобки, могут декларироваться, однако для представления относящихся к ним ограничений в данном случае используются дизъюнкции над их подтипами с символическим кодированием (обозначенные одиночными кавычками). Следует также иметь в виду, что показываемые здесь списки не являются полиморфными.

Различие между системой типов, фактически декларируемой в FSD, и иерархией типов состоит в том, что системы типов определяют не только отношения присваивания подтипов, но еще и разрешенные элементы, ограничения по допустимым значениям разрешенных элементов и прочие ограничения по типам и значениям относящихся к ним элементов.

### 7.2.3 Декларирование ограничений по типам

В следующем примере нет ограничений типов, не говоря уже об ограничениях по допустимым значениям элементов; поэтому нет никакой необходимости в использовании элементов <cond> и <bicond> (см. раздел 8.5). Начнем его рассмотрение с неформального представления разрешенных элементов и их допустимых значений (см. рисунок 10). Здесь в каждой структуре элементов показываются разрешенные элементы и минимально допустимые значения для соответствующего ей типа.

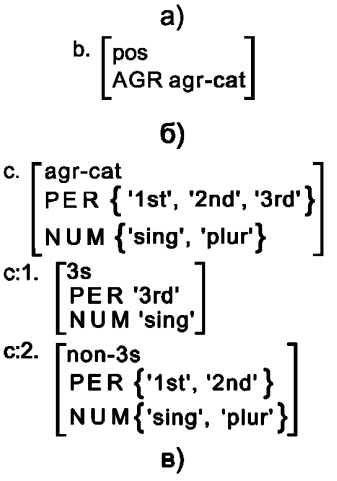



Рисунок 10 — Разрешенные элементы и их допустимые значения

Обозначения 3s и non-3s уточняют ограничения по допустимым значениям элементов PER и NUM, суживая дизъюнкцию, которая представляет ограничение.

Эта система типов, взятая в целом, кодируется так, как показано ниже.

***Пример*** — Система типов

```
<fsDecl type="sign">
 <fsDescr>Базовый тип для лингвистических символов</fsDescr>
 <fDecl name="head">
 <fDescr>Указывает синтаксическое оформление символа</fDescr>
 <vRange>
  <fs type="pos"/>
 </vRange>
</fDecl>
<fDecl name="spr">
 <fDescr>Показывает спецификаторы символа</fDescr>
 <vRange>
  <vColl org="list"/>
 </vRange>
</fDecl>
<fDecl name="comps">
```

```
      <fDescr>Показывает дополнения символа</fDescr>
      <vRange>
       <vColl org="list"/>
      </vRange>
     </fDecl>
    </fsDecl>
    <fsDecl type="word" baseTypes="sign">
     <fsDescr>Базовый тип для отдельных слов</fsDescr>
     <fDecl name="orth">
      <fDescr>Орфографическое представление для данного слова</fDescr>
      <vRange>
       <string/>
      </vRange>
     </fDecl>
    </fsDecl>
    <fsDecl type="phrase" baseTypes="sign">
     <fsDescr>Базовый тип для символов фразовых структур</fsDescr>
     <fDecl name="wordlist">
      <fDescr>Слова внутри данной фразы</fDescr>
      <vRange>
       <vColl org="list"/>
      </vRange>
     </fDecl>
    </fsDecl>
    <fsDecl type="stem" baseTypes="phrase">
     <fsDescr>Глагольная основа сформированной фразы</fsDescr>
     <fDecl name="head">
      <fDescr>Многословный заголовок, подлежащий использованию для всех фраз с данной основой</fDescr>
      <vRange>
       <fs type="verb"/>
      </vRange>
     </fDecl> copyright ИСО 2011 — Все права сохраняются ИСО 24610-2 <fDecl name="spr">
     <fDescr>Показывает спецификаторы символа</fDescr>
     <vRange>
      <vColl org="list"/>
     </vRange>
    </fDecl>
    <fDecl name="comps">
     <fDescr>Показывает дополнения символа</fDescr>
     <vRange>
      <vColl org="list"/>
     </vRange>
    </fDecl>
   </fsDecl>
   <fsDecl type="pos">
    <fsDescr>Parts of speech</fsDescr>
    <fDecl name="agr">
     <fDescr>Информация о соглашениях по данной части речи</fDescr>
     <vRange>
      <fs type="agr-cat"/>
     </vRange>
    </fDecl>
   </fsDecl>
   <fsDecl type="agr-cat">
    <fsDescr>Блок соглашения по стилю GPSG</fsDescr>
    <fDecl name="per">
     <fDescr>Значение для грамматического лица</fDescr>
     <vRange>
      <vAlt>
       <symbol value="1st"/>
```

```
      <symbol value="2nd"/>
      <symbol value="3rd"/>
     </vAlt>
    </vRange>
   </fDecl>
   <fDecl name="num">
    <fDescr>Значение для грамматического числа</fDescr>
    <vRange>
     <vAlt>
      <symbol value="sing"/>
      <symbol value="plur"/>
     </vAlt>
    </vRange>
   </fDecl>
  </fsDecl>
  <fsDecl type="3s" baseTypes="agr-cat">
   <fsDescr>Блок соглашения для третьего лица единственного числа</fsDescr>
   <fDecl name="per">
    <fDescr>Значение для грамматического лица</fDescr>
    <vRange>
     <symbol value="3rd"/>
    </vRange>
   </fDecl>
   <fDecl name="num">
    <fDescr>Значение для грамматического числа</fDescr>
    <vRange>
     <symbol value="sing"/>
    </vRange>
   </fDecl>
  </fsDecl>
  <fsDecl type="non-3s" baseTypes="agr-cat">
   <fsDescr>Блок соглашения для третьего лица единственного числа</fsDescr>
   <fDecl name="per">
    <fDescr>Значение для грамматического лица</fDescr>
    <vRange>
     <vAlt>
      <symbol value="1st"/>
      <symbol value="2nd"/>
     </vAlt>
    </vRange>
   </fDecl>
   <fDecl name="num">
    <fDescr> Значение для грамматического числа</fDescr>
    <vRange>
     <vAlt>
      <symbol value="sing"/>
      <symbol value="plur"/>
     </vAlt>
    </vRange>
   </fDecl>
  </fsDecl>
```

Некоторые элементы переобъявляются как допустимые в качестве подтипов для типов, в привязке к которым они уже были объявлены ранее. Такое повторное декларирование допустимо, хотя в случае невозможности унификации ограничений многократно наследуемых значений элементов адекватной структуры элементов такого типа не будет; определение типа *top* необходимо лишь в том случае, если это имя действительно будет фигурировать как тип в структуре элементов.

# 8 Декларация системы элементов

Примечание — Данный раздел представляет собой модифицированный вариант раздела 18.11 рекомендаций *TEI Guidelines P5*, 2005, посвященного рассмотрению декларации системы элементов. Более детальное обсуждение процедур логического вывода на основе FSD и еще один представительный пример можно найти в журнальной публикации Д. Теренса Лэнгендона и Гэри Ф. Саймонса [7].

## 8.1 Общие сведения

Стандартная декларация системы элементов (FSD) предназначена для использования в сочетании со структурой элементов (fs), соответствующей требованиям ИСО 24610-1:2006, хотя может применяться и для документирования любых систем структурных элементов. Цели такого использования изложены в разделе 1 («Область применения») вышеуказанного стандарта.

FSD реализует важную функцию точного документирования того информационного содержимого (контента), которое кодировщик, определенный системой разметки структур элементов, использовал в тексте на языке XML. Кроме того, FSD — это еще и важный ресурс, который стандартизует правила логического вывода, применяемые в программных средствах для контроля правильности разметки структур элементов в тексте и для обеспечения полной интерпретации недоопределенных структур элементов.

Следует, однако, иметь в виду, что возможен целый ряд терминологических расхождений между настоящим стандартом и сложившейся практикой как формальной логики, так и прикладной лингвистики типизированных структур элементов. В частности то, что в структурах элементов понимается как «интерпретация» структуры, на самом деле не является таковой в модельно-теоретическом смысле, а представляет собой минимально информативное (или, что то же, наиболее общее) расширение (см. 5.6) данной структуры элементов, которое соответствует множеству ограничений, декларированных в FSD. В рамках лингвистического приложения такая система ограничений является основным выразительным средством грамматики используемого естественного языка. Однако существует заметное различие в понимании того, какую же модельно-теоретическую интерпретацию (если таковая вообще существует) имеют структуры элементов в подобных приложениях. Этот аспект формальной интерпретации не имеет отношения к данному стандарту. Термин *valid* (*адекватная*), как правило, относится еще и к области формальной семантики, но в данном стандарте он обычно используется для описания чисто синтаксического аспекта формальной правильности в смысле, определяемом самой логикой типизированной структуры элементов, в отличие от понятия «формальная правильность» применительно к уровню кодирования, охватываемому настоящим стандартом (см. раздел 6).

В следующем подразделе показывается, каким образом в XML-кодограмме должна использоваться информация заголовка для формирования ссылок на любые ассоциируемые с ней FSD. В разделах 3, 4 и 5 описывается общая структура FSD и подробно рассматривается способ кодирования ее частей. Соответствующий развернутый пример дается в Приложении Б.

## 8.2 Привязка текста к декларациям систем элементов

Чтобы прикладная система программного обеспечения (ПО) могла использовать декларации систем элементов для автоматической интерпретации кодограмм или даже для отыскания человеком соответствующих деклараций, которые документируют систему элементов, используемую при разметке, должна существовать формальная ссылка из закодированных текстов на соответствующие декларации. Однако схема, в которой декларируется синтаксис системы элементов, должна быть отделена от схемы представления структуры элементов, которая является лишь реализацией этой системы.

Связь между FSD и документом, в котором используются структуры элементов, объявленные в этой декларации, материализуется в настоящем стандарте таким образом, чтобы эта связь не противоречила правилам ее включения в блок <encodingDesc> заголовка документа <teiHeader> [14]. Элемент fsdDecl может использоваться применительно к каждому отдельному типу структуры элементов, как показано ниже (в рамках данного стандарта для определения таких элементов используется «компактный» вариант языка схем RELAX NG):

```
element fsdDecl
{
  att.global.attributes,
  attribute type { data enumerated }?
  attribute url { data.pointer },
  empty
}
```

Элемент декларации системы элементов **<fsdDecl>** [FSD (feature-system declaration) declaration] указывает декларацию, которая содержит определения для конкретного типа структуры элементов. Помимо глобальных атрибутов тип идентифицирует структуру элементов, задокументированную в FSD; при этом предполагается, что данный элемент будет значением атрибута «тип» хотя бы одной структуры элементов. Таким значением может быть любая строка символов, но если это значение содержит пробел, оно подлежит нормализации: не должно быть ни одной предваряющей или завершающей цепочки символов пробелов и больше одного символа пробела внутри. Атрибут «тип» является опциональным. Когда необходим более строгий контроль, для определения типа вместо этого атрибута может использоваться глобальный атрибут xml:id, и в этом случае адекватным идентификатором должно быть имя. Если не используется ни тот, ни другой атрибут, то предполагается, что элемент <fsdDecl> идентифицирует декларации FSD для всех типов структур элементов, используемых при кодировании.

Элемент **url** обеспечивает связь с объектом, который содержит декларацию системы элементов. Его значением должен быть унифицированный идентификатор ресурса RFC 2396 [Uniform Resource Identifier (URI)].

Для данной FSD может существовать множество элементов *fsdDecl* — по одному для каждого типа структуры, которую этот элемент определяет. В приведенном ниже примере файл *Lexicon.fsd* содержит FSD, в которой в свою очередь содержатся определения структур элементов для лексических статей (<fs type="entry">) и лексических подстатей (<fs type="subentry">). Файл *Gazdar.fsd* содержит другую FSD, содержащую определение типа структуры элементов, которая носит название **GPSG**:

```
<TEI>
 <teiHeader>
  <fileDesc>
<!-- ... -->
 </fileDesc>
 <encodingDesc>
<!-- ... -->
 <fsdDecl type="GPSG" url="Gazdar.fsd"/>
 <fsdDecl type="entry" url="Lexicon.fsd"/>
 <fsdDecl type="subentry" url="Lexicon.fsd"/>
<!-- ... -->
 </encodingDesc>
</teiHeader>
<!--Сюда вставляется текст -->
</TEI>
```

В этом примере показан элемент <fsdDecl> внутри элемента <encodingDesc> для каждого отдельного значения, используемого в качестве типа элементов <fs> самого документа. В данном случае, например, декларация системы элементов, используемая структурами элементов типа «статья» и «подстатья» должны присутствовать в объекте, в элементе URL **Lexicon.fsd**.

Настоящий стандарт не устанавливает никакого способа обеспечения уникальности значений типов для элементов fsdDecl, равно как не требует и того, чтобы каждое значение типа, определенное в элементе <fs>, декларировалось также в элементе <fsdDecl> и гарантировалась невозможность появления множества элементов <fsdDecl> в одном и том же дескрипторе <encodingDesc> - с опциональным атрибутом типа или без него.

Кодировщикам, которым требуются такие ограничения (которые могут быть иногда весьма полезными для обеспечения согласованной и точной разметки), рекомендуется разрабатывать инструментальные средства для их принудительного введения с использованием таких механизмов, как язык утверждений Schematron.

В FSDs допускается присутствие следующих элементов: *fsd*, *fsDecl*, *fsDescr*, *fDecl*, *fDescr*, *vRange*, *vDefault*, *if*, *then*, *fsConstraints*, *cond*, *bicond* и *iff*.

Поскольку синтаксис FSD не зависит ни от какого-либо модуля TEI, ни от FSR ИСО 24610-1, он должен использоваться в сочетании со стандартными модулями tei, заголовка и ядра в соответствии с требованиями ИСО 24610-1.

В общих чертах FSD состоит из одной либо нескольких деклараций структур элементов (<fsDecl>), одного или нескольких определений элементов (<fDecl>) и нулевого или большего числа ограничений структур элементов (<cond> и/или <bicond>). Определения элементов и ограничения структуры элементов действуют только в рамках деклараций структур элементов.

### 8.3 Общая структура декларации системы элементов

Декларация системы элементов кодируется как документ типа <fsd>. Помимо своих глобальных атрибутов этот документ содержит две части: опциональный заголовок (который дает библиографическую информацию для файла) и совокупности деклараций структур элементов, каждая из которых определяет один тип структуры элементов. Каждая декларация структуры элементов, в свою очередь, состоит из трех частей: опционального описания (которое дает словесный комментарий, поясняющий, что именно кодирует данный тип структуры элементов); обязательной совокупности деклараций элементов (которая устанавливает ограничения по диапазону значений и задает значения по умолчанию для элементов структуры данного типа) и опциональных ограничений структуры элементов (которые определяют *в числе других* ограничения на совместную встречаемость значений элементов). Рекомендуется кодировать заголовок как элемент <teiHeader> (см. [14], глава 2). Прочие вышеперечисленные компоненты уникальны для деклараций системы элементов. Поэтому появляется целый ряд новых элементов, представленных ниже:

- **<fsd>** (feature system declaration) содержит декларацию системы элементов;
- **<fsDecl>** (feature structure declaration) декларирует один тип структуры элементов;
- **<fsDescr>** [feature structure description (в FSD)] описывает в текстовой форме, что именно характеризуется типом структуры элементов, объявленным во вложении <fsDecl>;
- **<fDecl>** (feature declaration) декларирует единственный элемент, определяя его имя, способ организации, диапазон допустимых значений и, возможно, но не обязательно — его значение, присваиваемое по умолчанию;
- **<fsConstraints>** (feature-structure constraints) определяет другие конкретные ограничения, накладываемые на адекватные структуры элементов внутри данной FSD.

Декларации элементов и ограничения структур элементов рассматриваются в двух последующих подразделах (8.4 и 8.5). Спецификация аналогичных элементов <fsDecl> может быть упрощена путем введения иерархии наследования для различных типов структур элементов. Каждый элемент <fsDecl> может содержать одно или несколько имен базовых типов *baseTypes*, от которых он наследует декларации элементов и ограничения (эти типы часто называются «супертипами»).

Предположим, для примера, что <fsDecl type="Basic"> содержит <fDecl name="one"> и <fDecl name="two">, а <fsDecl type="Derived" baseTypes="Basic"> содержит только <fDecl name="three">. Тогда любой экземпляр <fs type="Derived"> должен включать в себя все три элемента. Это происходит потому, что <fsDecl type="Derived"> наследует две декларации элементов от <fsDecl type="Basic">, когда он определяет базовый тип Basic.

***Пример*** — Приведенный ниже пример показывает общую структуру полной декларации FSD:

```
<fsd>
 <teiHeader>
<!—Заголовок как для документа TEI -->
 </teiHeader>
 <fsDecl type="SomeName">
  <fsDescr>Описывает сущность, представляемую данным типом fs </fsDescr>
 <fDecl name="featureOne">
 <!—Декларация для featureOne -->
 </fDecl>
 <fDecl name="featureTwo">
 <!-- Декларация для featureTwo -->
 </fDecl>
 <fsConstraints>
 <!-- Здесь указываются ограничения структуры элементов -->
 </fsConstraints>
</fsDecl>
<fsDecl type="AnotherType">
<!—Декларируется другой тип структуры элементов -->
 </fsDecl>
</fsd>
```

Формальное определение <fsd> и ее компонентов выглядит следующим образом:

```
element fsd { att.global.attributes, fsd.content }
fsd.content = teiHeader?, fsDecl+
fsDecl = element fsDecl
```

```
{
   att.global.attributes
   fsDecl.attributes,
   fsDecl.cont
}
fsDecl.content = fsDescr?, fDecl+, fsConstraints?
fsDecl.attributes =
   attribute type { data.enumerated },
   attribute baseTypes { list { data.name+ } }?
fsDescr = element fsDescr
{
   att.global.attributes
   fsDescr.content
}
fsDescr.content = macro.limitedContent
```

Атрибут *baseTypes* дает имена одного или нескольких типов, от которых данный тип наследует спецификации элементов и ограничения; если данный тип содержит спецификацию элементов с тем же именем, что и у спецификации, наследуемой от любого из типов, определяемых этим атрибутом, или имеет место наследование нескольких спецификаций с одним и тем же именем, то возможные значения этого элемента определяются посредством унификации. Аналогично совокупность применимых ограничений выводится путем сочетания ограничений, заданных явно внутри данного элемента, с ограничениями, которые подразумеваются атрибутом *baseTypes*.

Когда базовый тип не определен, ни спецификация элементов, ни ограничение не наследуются.

Несмотря на то, что настоящей частью ИСО 24610 предусматривается возможность использования стандартных значений элементов, наследование свойств определяется как монотонное.

Процесс комбинирования ограничений может приводить к противоречию: например, в том случае, когда две спецификации для одного и того же элемента определяют несогласованные диапазоны значений и хотя бы одна из таких спецификаций является обязательной. В подобных случаях адекватной структуры элементов для определяемого типа не существует.

Каждый из типов, определенных в *baseTypes*, должен представлять собой одно слово, разрешенное списком имен XML; например, в имени не должно быть пробелов, и оно не может начинаться с цифр. Множественные базовые типы отделяются друг от друга пробелами: например, <fsDecl type="Sub" baseTypes="Super1 Super2">.

Атрибут <fsDescr> может содержать любой текст, за исключением определенных служебных элементов (например, *del*), используемых для транскрибирования существующих текстов.

### 8.4 Декларации элементов

#### 8.4.1 Общие замечания

Каждый элемент объявляется в элементе <fDecl>, в котором атрибут имени указывает объявляемое свойство; этот атрибут соответствует атрибуту имени декларируемых элементов <f>.

Элемент <fDecl> состоит из трех частей: необязательного текстового описания, в котором должно объясняться, какой именно элемент и какие его значения представляются; обязательной спецификации диапазона значений, которая декларирует разрешенные значения элемента; и необязательной спецификации стандартных значений, где декларируются значение по умолчанию, подлежащее выбору в том случае, когда названный элемент не появляется в <fs>. При этом может определяться либо единственное безусловное стандартное значение, либо множество обусловленных значений.

#### 8.4.2 Логический вывод типа для обязательных элементов

Если в некоторой структуре элементов какой-то элемент

- не является опциональным (т. е. обязателен),
- не имеет присваиваемого значения или получает значение <default> (см. ИСО 24610-1:2006, пункт 5.10, *Значения по умолчанию*) и
- либо не имеет заданного значения по умолчанию, либо имеет обусловленные стандартные значения, для которых не удовлетворяется ни одно из условий,

то значением такого элемента в самом общем адекватном расширении структуры элементов будет наиболее общее значение, представленное в его элементе <vRange> в случае блочной организации, одноэлементного множества и множества с повторяющимися элементами или в списке, содержащем данный элемент, при сложной организации.

**8.4.3 Логический вывод типа для опциональных элементов со значениями по умолчанию**

Если в некоторой структуре элементов, какой-то элемент:

- является опциональным,
- не имеет присваиваемого значения или получает значение <default> и
- имеет заданное значение по умолчанию либо обусловленные стандартные значения, для которых одно из условий выполнено,

то значением такого элемента в самом общем адекватном расширении структуры элементов, если таковое существует, будет подходящее стандартное значение. Очевидно, что данный элемент принимает это значение и в том случае, когда он обязателен и для него определено значение по умолчанию.

**8.4.4 Логический вывод типа для опциональных элементов без стандартных значений**

Если в некоторой структуре элементов какой-то элемент:

- является опциональным,
- не имеет присваиваемого значения или получает значение <default> и
- либо не имеет заданного значения по умолчанию, либо имеет обусловленные стандартные значения, для которых не удовлетворяется ни одно из условий,

то для такого элемента в самом общем адекватном расширении структуры элементов, когда она существует, не будет присваиваемого значения. Подобная ситуация допустима, поскольку данный элемент необязателен.

**8.4.5 Возможность неудачного исхода логического вывода**

Структура элементов может не иметь адекватного расширения, когда подходящее значение элемента по умолчанию (стандартное значение) несовместимо с его диапазоном допустимых значений, объявленным в декларации. В этом случае для принудительного обеспечения соответствия действующим критериям необходимо использовать дополнительные инструментальные средства.

**8.4.6 Элементы и атрибуты деклараций элементов**

Элемент **<fDecl>** (декларация элемента) декларирует единственный элемент, определяя его имя, способ организации, диапазон допустимых значений, не обязательное значение по умолчанию и показывает, обязателен ли сам данный элемент или не обязателен. В декларациях элементов используются элементы и атрибуты, перечисленные ниже:

- **name** указывает имя декларируемого элемента; оно соответствует атрибуту «имя» элементов <f> в тексте;
- **org** определяет способ упорядочения значений элемента;
- **optional** показывает, является или не является данный элемент опциональным в структуре элементов декларируемого типа;
- **<fDescr>** [дескриптор элемента (в FSD)] содержит текстовое описание сущности, представляемой декларируемым элементом, и ее значения.
- **<vRange>** задает диапазон допустимых значений для элемента как <fs>, <vAlt> или built-in; чтобы значение <f> было правильным, оно должно принадлежать заданному диапазону; если <f>содержит множество значений (санкционированных атрибутом *org*), то каждое из них должно лежать в заданном диапазоне vRange;
- **<vDefault>** декларирует значение по умолчанию, которое должно выбираться при отсутствии в структуре элементов экземпляра <f> для данного имени; если ограничений нет, то при этом определяется один или несколько элементов <fs> либо простых значений (в зависимости от конкретного значения атрибута *org* вложенного элемента fDecl); если элемент <vDefault> обусловлен, то он определяется как один или несколько элементов <if>; когда значения по умолчанию не определены или ограничения не удовлетворены, то не выбирается никакое значение;
- **<if>** определяет обусловленное значение по умолчанию для данного элемента; условие задается как структура элементов; оно удовлетворяется в том случае, когда категоризирует структуру элементов в тексте, для которой ищется значение по умолчанию;
- **<then>** отделяет условие от стандартного значения в элементе <if> или логическое условие от вывода в элементе <cond>.

**8.4.7 Декларации элементов и категоризации**

Логика контроля правильности значений элементов и проверки совпадения условий для выбора и применения значений по умолчанию основывается на использовании операции категоризации. Это стандартная операция в системе формализации, строящейся на выделении структурных элементов. Структура элементов FS категоризирует все структуры элементов, которые совместимы с ней и не менее информативны, чем она сама: т. е. все структуры элементов, которые определяют все те значения элементов, что и FS, вместе со значениями, которые категоризируются значениями FS, и которые имеют те же множественные входы, что и FS [1]. Формальное определение см. в 5.1.

В стиле данного выше неформального определения можно расширить также область действия операции категоризации путем распространения ее на отношения дизъюнкции и отрицания, на конкретные примитивы и на использование атрибутов в языке разметки: например, элемент <vAlt>, содержащий значение *v*, категоризирует *v*. Отрицание значения *v* (представляемое элементом <vNot> и фигурирующее в ИСО 24610-1:2006, пункт 5.9.3) категоризирует любое значение, которое не объединяется с *v* или (как в случаях дизъюнкций и отрицаний) не включает в себя *v*; например, структура

```
<vNot>
 <numeric value="0"/>
</vNot>
```

категоризирует любое численное значение отличное от нуля.

Значение <**fs type**="X"/>, даже если оно неверно, категоризирует любую структуру элементов типа X.

### 8.4.8 Пример деклараций элементов

8.4.8.1 Приведенный ниже пример деклараций элементов в обобщенной грамматике лексических структур (GPSG) заимствован из книги [4]. В приложении к этой книге (с. 245—247) авторы предложили в частности систему элементов для английского языка:

Диапазоны значений элементов:

- INV {+, -}
- SUBJ {+, -}
- CONJ {and, both, but, either, neither, nor, or, NIL}
- COMP {for, that, whether, if, NIL}
- AGR CAT
- PFORM {to, by, for, . . . }

Стандартные значения элементов:

- FSD 1: [-INV]
- FSD 2: ~ [CONJ]
- FSD 9: [INF, +SUBJ] → [COMP for]

8.4.8.2 Следует иметь в виду, что в рассматриваемом примере аббревиатура «FSD» обозначает не декларации систем элементов («feature system declarations»), фигурирующие в настоящем стандарте, а присущие грамматике GPSG стандартные значения спецификаций элементов («feature specification defaults»). Элемент INV, указывающий, нарушен ли в предложении прямой порядок слов, может принимать только два значения: *plus* (+) и *minus* (–). Если этот элемент не определен, то стандартное правило (здесь это FSD 1) гласит, что по умолчанию всегда предполагается значение *minus*. Декларация для этого элемента должна кодироваться следующим образом:

```
<fDecl name="inv">
 <fDescr>инвертированное предложение</fDescr>
 <vRange>
  <vAlt>
   <binary value="true"/>
   <binary value="false"/>
  </vAlt>
 </vRange>
 <vDefault>
  <binary value="false"/>
 </vDefault>
</fDecl>
```

Диапазон значений определяется как дизъюнкция (а точнее — как результат операции «исключающее ИЛИ») над значениями, которые могут быть представлены значением <binary>, т. е. значением должны быть либо «истина», либо «ложь», но не то и другое и не отсутствие значения.

8.4.8.3 Элемент CONJ указывает на то, что в данной конструкции используется поверхностная форма конъюнкции. Знак тильды (~) в стандартном правиле (см. выше FSD 2) представляет операцию отрицания. Это значит, что по умолчанию данный элемент отсутствует, т. е. конъюнкции не существует.

Ситуация отсутствия CONJ отлична от ситуации, в которой CONJ присутствует, но в диапазон ее допустимых значений входит ноль (NIL). В контексте проводимого авторами анализа значение NIL показывает, что операция конъюнкции имеет место, но в поверхностной форме предложения нет явно выраженной конъюнкции. Декларация элементов, ассоциируемая с данным элементом должна кодироваться следующим образом:

```
<fDecl name="conj">
 <fDescr>поверхностная форма конъюнкции</fDescr>
 <vRange>
  <vAlt>
   <symbol value="and"/>
   <symbol value="both"/>
   <symbol value="but"/>
   <symbol value="either"/>
   <symbol value="neither"/>
   <symbol value="nor"/>
   <symbol value="or"/>
   <symbol value="NIL"/>
   <binary value="false"/>
  </vAlt>
 </vRange>
 <vDefault>
  <binary value="false"/>
 </vDefault>
</fDecl>
```

В данном случае можно обойтись и без элемента <vDefault>, поскольку единственным носителем информации об отсутствии других допустимых значений служит двоичное значение «ложь».

8.4.8.4 Элемент COMP указывает на то, что в конструкции используется поверхностная форма комплементайзера. По своему диапазону значений он аналогичен элементу CONJ. Однако стандартное правило этого элемента (см. выше FSD 9) является обусловленным. Оно гласит, что если глагол стоит в инфинитивной форме (элемент VFORM в правиле не упоминается, так как это единственный элемент, могущий принимать значение INF), а в конструкции предложения имеется подлежащее, то дальше должно использоваться дополнение с предлогом *for*. Например, для того чтобы сделать имя Джон подлежащим инфинитивного оборота «It is necessary to go», необходимо использовать дополнение с предлогом *for*; т. е. следует написать «It is necessary *for* John to go». Декларация элементов, ассоциируемая с данным элементом должна кодироваться следующим образом:

```
<fDecl name="comp">
 <fDescr>поверхностная форма комплементайзера</fDescr>
 <vRange>
  <vAlt>
   <symbol value="for"/>
   <symbol value="that"/>
   <symbol value="whether"/>
   <symbol value="if"/>
   <symbol value="NIL"/>
  </vAlt>
 </vRange>
 <vDefault>
  <if>
   <fs>
    <f name="vform">
     <symbol value="INF"/>
    </f>
    <f name="subj">
     <binary value="true"/>
    </f>
   </fs>
   <then/>
   <symbol value="for"/>
  </if>
 </vDefault>
</fDecl>
```

8.4.8.5 Элемент AGR хранит все элементы, относящиеся к соглашению о прямом порядке слов в предложении. Газдар с соавторами [4] определяют диапазон значений этого элемента как CAT. Это говорит о том, что значением данного элемента является категория, и этим термином в книге обозначается структура элементов. В действительности это слишком слабое утверждение, потому что здесь не только допустима любая структура, но она еще должна быть и структурой элементов для соглашения (которая в развернутом примере в конце соответствующего раздела определена авторами как содержащая элементы грамматического лица и грамматического числа). Данное ограничение по диапазону значений кодируется с помощью следующей декларации элементов:

```
<fDecl name="agr">
 <fDescr>соглашение о лице и числе</fDescr>
 <vRange>
  <fs type="Agreement"/>
 </vRange>
</fDecl>
```

Отсюда следует, что рассматриваемое значение должно представлять собой структуру элементов типа *Agreement*. В детализированном примере, приведенном в Приложении А настоящего стандарта, представлен тип <fsDecl type="Agreement">, который включает в себя элементы <fDecl name="pers"> и <fDecl name="num">.

8.4.8.6 Элемент PFORM показывает поверхностную форму предлога, используемую в языковой конструкции. Поскольку элемент PFORM был определен ранее как открытое множество, в представленной ниже спецификации диапазона используется тип <string>, а не <symbol>.

```
<fDecl name="pform">
 <fDescr>словоформа предлога</fDescr>
 <vRange>
  <vNot>
   <string/>
  </vNot>
 </vRange>
</fDecl>
```

***Пример*** — Приведенная ниже конструкция, в которой используется значение с отрицанием:

```
<vNot>
 <string/>
</vNot>
```

категоризирует любую непустую строку.

8.4.8.7 Далее рассматриваются декларации элементов. Класс model.featureVal включает в себя все возможные значения элементов, в том числе структуры элементов, дизъюнкции (<vAlt>) и сложные коллекции (<vColl>).

```
fDecl = element fDecl
{
   att.global.attributes,
   fDecl.attributes,
   fDecl.content
}
fDecl.attributes =
   attribute name { data.name },
   attribute optional { xsd:boolean }?,
   attribute org { "unit" | "set" | "bag" | "list" }?
fDecl.content = fDescr?, vRange, vDefault?
fDescr = element fDescr
{
   att.global.attributes,
   macro.limitedContent
}
vRange = element vRange
{
```

```
    att.global.attributes,
    model.featureVal
}
vDefault = element vDefault
{
    att.global.attributes,
    ( model.featureVal+ | if+ )
}
if = element if
{
    att.global.attributes,
    ( ( fs | f ), then, ( model.featureVal ) )
}
then = element then
{
    att.global.attributes,
    empty
}
```

### 8.5 Ограничения структуры элементов

Для гарантии адекватности структуры элементов может потребоваться нечто большее, чем простая спецификация диапазона допустимых значений для каждого элемента. Могут оказаться необходимыми ограничения совместной встречаемости каких-то значений в рамках одной и той же структуры элементов или во вложенной структуре.

Такие ограничения структуры элементов выражаются как ряд последовательно применяемых и условных и биусловных критериев в части <fsConstraints> декларации <fsDecl>. Конкретная структура элементов адекватна лишь в том случае, если она удовлетворяет всем связанным с ней ограничениям. Элемент <cond> кодирует обычное условное высказывание типа «если..., то...» булевой логики, которое успешно выполняется, когда либо следствие принимает значение «истина», либо условие принимает значение «ложь». Элемент <bicond> кодирует биусловную операцию («если и только если») булевой логики. Эта операция успешно выполняется только в том случае, когда соответствующие условные высказывания истинны в обоих направлениях. В ограничениях элементов структуры первый член отношения и вывод выражаются структурами элементов; они считаются истинными, если их структура элементов категоризирует (см. 8.4, *Декларации элементов*) искомую структуру. С методической точки зрения, если первый член отношения принимает значение «истина», то вывод тоже должен быть истинным, так как истинность вывода утверждается, а не просто проверяется. Таким образом условие выполняется принудительно посредством определения правила, согласно которому первый член отношения не категоризирует (и никогда не должен категоризировать) данную структуру элементов, или путем введения правила, по которому первый член отношения обязательно категоризирует данную структуру элементов, а затем выполняется операция унификации над выводом и этой структурой (при успешном выполнении этой операции ее результат будет категоризирован следствием). На практике необходимость в принудительном выполнении подобных ограничений может возникать в те периоды, когда факт справедливости ограничений применительно к данной структуре элементов просто не установлен; в этом случае ограничение должно непрерывно проверяться по мере усиления его информативности до тех пор, пока не будет определено значение «истина» или не произойдет остановка вычислительного процесса по какой-то иной причине.

Часть <fsConstraints> декларации FSD образуется следующими элементами:

- **fsConstraints** (feature-structure constraints) определяет ограничения, накладываемые на информационное содержание адекватной структуры элементов;
- **cond** (conditional feature-structure constraint) задает условное ограничение структуры элементов; вывод (следствие) и антецедент определяются как структуры элементов или коллекции структур элементов; ограничение удовлетворяется, если как антецедент, так и следствие категоризируют данную структуру элементов, или если антецедент ее не категоризирует;
- **bicond** (biconditional feature-structure constraint) определяет биусловное ограничение структуры элементов; и следствие, и антецедент определяются как структуры элементов или как коллокации структур элементов; ограничение удовлетворяется, если как антецедент, так и следствие категоризируют данную структуру элементов, или если оба ее не категоризируют;

- **then** отделяет условие от стандартного значения в элементе <if>, или логическое условие от вывода в элементе <cond>;
- **iff** отделяет условие от следствия в элементе <bicond>.

Ниже приводится пример ограничений структуры элементов, который касается отслеживания "совместной встречаемости элементов" и заимствован из системы элементов для английского языка [4, с. 246]:

- FCR 1: [+INV] $\rightarrow$ [+AUX, FIN]
- FCR 7: [BAR 0] $\leftrightarrow$ [N] & [V] & [SUBCAT]
- FCR 8: [BAR 1] $\rightarrow$ ~ [SUBCAT]

Первое ограничение говорит, что если языковая конструкция инвертирована, то она должна также содержать вспомогательный глагол и глагол в инфинитиве; т. е.:

```
<cond>
 <fs>
  <f name="inv">
   <binary value="true"/>
  </f>
 </fs>
 <then/>
 <fs>
  <f name="aux">
   <binary value="true"/>
  </f>
  <f name="vform">
   <symbol value="fin"/>
  </f>
 </fs>
</cond>
```

Второе ограничение говорит о том, что если языковая конструкция содержит нулевое значение переменной BAR (т. е. является лексемой), эта конструкция должна содержать также значение для элементов N, V и SUBCAT. Кроме того, поскольку это биусловное ограничение, то при наличии значений для N, V и SUBCAT, должно выполняться условие BAR='0', т. е.:

```
<bicond>
 <fs>
  <f name="bar">
   <symbol value="0"/>
  </f>
 </fs>
 <iff/>
 <fs>
  <f name="n"/>
  <f name="v"/>
  <f name="subcat"/>
 </fs>
</bicond>
```

Примечание — Здесь в соответствии с ИСО 24610-1:2006, пункт 5.10 *стандартные значения*, (107), элемент <f name="n"> трактуется, например как имеющий значение из допустимого диапазона, что эквивалентно следующей записи:

```
<f name="n">
 <vAlt>
  <binary value="true"/>
  <binary value="false"/>
 </vAlt>
</f>
```

Последнее ограничение говорит о том, что если в языковой конструкции элемент BAR принимает значение 1 (т. е. является фразовой структурой), то элемент SUBCAT должен отсутствовать (~). Это не

биусловное ограничение, так как имеются другие экземпляры, для которых элемент SUBCAT не подходит, т. е.:

```
<cond>
 <fs>
  <f name="bar">
   <symbol value="1"/>
  </f>
 </fs>
 <then/>
 <fs>
  <f name="subcat">
   <binary value="false"/>
  </f>
 </fs>
</cond>
```

Формальная декларация для ограничений структуры элементов кодируется так, как показано ниже. При этом следует иметь в виду, что элементы <cond> и <bicond> используют пустые теги <then> и <iff>, соответственно для разделения логического условия и следствия. Это делается в основном ради обеспечения удобочитаемости.

```
fsConstraints = element fsConstraints
{
   att.global.attributes,
   ( cond | bicond )*
}
cond = element cond
{
   att.global.attributes,
   ( ( fs | f ), then, ( fs | f ) )
}
bicond = element bicond
{
   att.global.attributes,
   ( ( fs | f ), iff, ( fs | f ) )
}
iff = element iff
{
   att.global.attributes,
   empty
}
```

**Приложение А**
**(обязательное)**

## Схема XML для структур элементов

```
macro.limitedContent = (text | model.limitedPhrase | model.inter)*
macro.xtext = (text | model.gLike)*
att.global.attributes =
  att.global.attribute.xmlid,
  att.global.attribute.n,
  att.global.attribute.xmllang,
  att.global.attribute.xmlbase,
  empty

att.global.attribute.xmlid =
  ## (идентификатор) обеспечивает однозначную идентификацию элемента, несущего в себе
  ## атрибут.
  attribute xml:id { xsd:ID }?
att.global.attribute.n =
  ## (номер) указывает номер (или иную метку) для элемента, который
  ## не обязательно уникален в рамках данного документа.
  attribute n {
    list {
      xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }+
    }
  }?
att.global.attribute.xmllang =
  ## (язык) указывает язык информационного наполнения (контента) элемента с помощью
  ## тега, сгенерированного в соответствии с BCP 47
  attribute xml:lang { xsd:language }?
att.global.attribute.xmlbase =
  ## предоставляет ссылку на URI, с помощью которого приложения могут
  ## преобразовать ссылки на относительные URI в абсолютные
  ## адреса URI.
  attribute xml:base { xsd:anyURI }?
model.gLike = notAllowed
model.featureVal.complex = fs | vColl | vNot | vMerge
model.featureVal.single =
  binary | symbol | numeric | \string | vLabel | \default | vAlt
model.placeStateLike = notAllowed
model.qLike = notAllowed
model.nameLike = model.placeStateLike
model.featureVal = model.featureVal.complex | model.featureVal.single
model.pPart.data = model.nameLike
model.inter = model.qLike
model.limitedPhrase = model.pPart.data
fsdDecl =
  ## предоставляет декларацию системы элементов, состоящую из одной или нескольких
  ## деклараций структур элементов или ссылок на декларации структур элементов.
  element fsdDecl {
    (fsDecl | fsdLink)+,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
fsDecl =
  ## (декларация структуры элементов) объявляет один тип структуры элементов.
```

```
    element fsDecl {
      (fsDescr?, fDecl+, fsConstraints?),
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## присваивает имя объявленному типу структуры элементов.
      attribute type { xsd:Name },
      ## присваивает имя одной или нескольким типизированным структурам элементов,
      ## от которых данный тип наследует спецификации элементов и
      ## ограничения; если этот тип включает в себя спецификацию элементов
      ## с таким же именем, как у любой из спецификаций, определяемых данным
      ## атрибутом, или если наследуется больше одной спецификации с тем же именем,
      ## то множество возможных значений устанавливается с помощью операции
      ## унификации. Аналогично выводится множество применимых ограничений
      ## путем комбинирования ограничений, заданных явно внутри данного элемента,
      ## с ограничениями, которые влечет за собой атрибут baseTypes.
      ## Если атрибут baseTypes определен, то наследование спецификации элементов
      ## или ограничения не происходит.
      attribute baseTypes {
        list { xsd:Name+ }
      }?,
      empty
    }
  fsDescr =
    ## (описание системы элементов (в FSD)) содержит текстовое представление
    ## сущности, характеризуемой данным типом структуры элементов
    ## объявленной во вложении fsDecl.
    element fsDescr {
      macro.limitedContent,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }

  fsdLink =
    ## (ссылка на декларацию системы элементов) ассоциирует имя
    ## типизированной структуры элементов с ее декларацией
    ## структуры элементов.
    element fsdLink {
      empty,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## определяет тип структуры элементов, подлежащей документированию;
      ## это будет значение атрибута типа хотя бы в одной структуре элементов.
      attribute type { xsd:Name },
      ## формирует указатель на элемент декларации структуры элементов
      ## (fsDecl) в рамках данного документа или за его пределами.
      attribute target { xsd:anyURI },
      empty
    }
  +fDecl =
    ## (декларация элемента) объявляет одиночный элемент
    ## посредством указания его имени, способа организации,
    ## диапазона допустимых значений и значения по умолчанию (не обязательно)
    element fDecl {
      (fDescr?, vRange, vDefault?),
      att.global.attribute.xmlid,
```

```
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    ## указывает имя декларируемого элемента; проверяет на совпадение
    ## атрибут имени f элементов в тексте.
    attribute name { xsd:Name },
    ## сигнализирует о том, может ли или не может присутствовать
    ## значение данного элемента.
    [ a1:defaultValue = "true" ] attribute optional { xsd:boolean }?,
    empty
  }
fDescr =
  ## (описание элемента (в FSD)) содержит текстовое представление
  ## сущности, характеризуемой объявленным элементом,
  ## и его значения.
  element fDescr {
    macro.limitedContent,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
vRange =
  ## (диапазон значений) определяет диапазон допустимых значений элемента
  ## в форме fs, vAlt, или примитива;
  ## чтобы значение f было правильным, оно должно
  ## категоризироваться заданным диапазоном; если f
  ## содержит повторяющиеся значения (как разрешено атрибутом org),
  ## то каждое значение должно категоризироваться диапазоном vRange.
  element vRange {
    model.featureVal,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
vDefault =
  ## (значение по умолчанию) декларирует стандартное значение, которое должно
  ## предоставляться в том случае, когда структура элементов
  ## не содержит экземпляра f для данного имени; если стандартное значение
  ## не связано условиями, то оно определяется как один или (в зависимости от
  ## значения атрибута org вложения fDecl) как несколько
  ## элементов fs либо примитивов;
  ## если стандартное значение обусловлено, то оно определяется как
  ## один или несколько элементов if; если стандартное значение не определено
  ## или условия не выполнены, то значение по умолчанию не присваивается.
  element vDefault {
    (model.featureVal+ | if+),
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
if =
  ## определяет обусловленное стандартное значение для элемента; условие
  ## определяется как структура элементов и считается выполненным, если оно
  ## категоризирует структуру элементов в тексте, для которого
  ## ищется стандартное значение.
  element if {
```

```
      ((fs | f), then, model.featureVal),
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  then =
    ## разделяет условие и стандартное значение в элементе if или
    ## логическое условие и следствие в элементе cond.
    element then {
      empty,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  fsConstraints =
    ## (ограничения структуры элементов) задает ограничения по
    ## контенту адекватных структур элементов.
    element fsConstraints {
      (cond | bicond)*,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  cond =
    ## определяет обусловленное ограничение структуры элементов;
    ## следствие и логическое условие определяются как структуры;
    ## элементов или как коллекции структур элементов; ограничение
    ## удовлетворяется, если и логическое условие, и следствие
    ## одновременно категоризируют данную структуру элементов
    ## или если логическое условие ее не категоризирует.
    element cond {
      ((fs | f), then, (fs | f)),
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  bicond =
    ## определяет биусловное ограничение структуры элементов;
    ## и следствие, и логическое условие определяются как
    ## структуры элементов или как группы структур элементов;
    ## ограничение удовлетворяется, если и логическое условие
    ##, и следствие одновременно категоризируют или одновременно
    ## не категоризируют даннную структуру элементов.
    element bicond {
      ((fs | f), iff, (fs | f)),
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  iff =
    ## (тогда и только тогда) разделитель логического условия и следствия
    ## в элементе bicond.
```

```
  element iff {
   empty,
   att.global.attribute.xmlid,
   att.global.attribute.n,
   att.global.attribute.xmllang,
   att.global.attribute.xmlbase,
   empty
  }
fs =
  ## представляет структуру элементов, т. е. коллекцию
  ## пар "элемент - значение", организованную как структурная единица.
  element fs {
   f*,
   att.global.attribute.xmlid,
   att.global.attribute.n,
   att.global.attribute.xmllang,
   att.global.attribute.xmlbase,
   ## определяет тип структуры элементов.
   attribute type { xsd:Name }?,
   ## содержит ссылки на спецификации элементов,
   ## образующие данную структуру элементов.
   attribute feats {
    list { xsd:anyURI+ }
   }?,
   empty
  }
f =
  ## элемент, представляющий спецификацию значений элемента,
  ## т. е. ассоциирующий имя со значением любого из нескольких
  ## типов types.
  element f {
   model.featureVal*,
   att.global.attribute.xmlid,
   att.global.attribute.n,
   att.global.attribute.xmllang,
   att.global.attribute.xmlbase,

   ## предоставляет имя для элемента.
   attribute name { text },
   ## (значение элемента) содержит ссылку на любой элемент, который может быть
   ## использован для представления значения элемента.
   attribute fVal { text }?,
   empty
  }
binary =
  ## (двоичное значение) представляет ту часть спецификации значений,
  ## элемента, которая может содержать любое из двух возможных
  ## значений.
  element binary {
   empty,
   att.global.attribute.xmlid,
   att.global.attribute.n,
   att.global.attribute.xmllang,
   att.global.attribute.xmlbase,
   ## предоставляет двоичное значение.
   attribute value { xsd:boolean },
   empty
  }
symbol =
  ## (символьное значение) представляет ту часть спецификации
  ## значений элемента, которая содержит один из конечных
  ## списков символов.
```

```
    element symbol {
      empty,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## предоставляет символьное значение для элемента – только одно из,
      ## конечного списка, который может быть определен в декларации элементов.
      attribute value {
        xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }
      },
      empty
    }

  numeric =
    ## (численное значение) представляет ту часть спецификации значений
    ## элементов, которая содержит числовое значение или диапазон значений.
    element numeric {
      empty,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## дает нижнюю границу для представляемого численного значения,
      ## а также (если максимум еще не определен) его верхнюю границу,
      attribute value { xsd:double | xsd:decimal },
      ## дает верхнюю границу для представляемого численного значения,
      attribute max { xsd:double | xsd:decimal }?,
      ## показывает, должно ли производиться усечение представляемого
      ##  численного значения для получения целого числа.
      attribute trunc { xsd:boolean }?,
      empty
    }
  string =
    ## (строковое значение) представляет ту часть спецификации значений элементов
    ## которая содержит строку
    element string {
      macro.xtext,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  vLabel =
    ## (метка значения) представляет ту часть спецификации значений элементов,
    ## которая появляется в структуре элементов больше чем в одной точке,
    element vLabel {
      model.featureVal?,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## предоставляет имя для общей точки.
      attribute name {
        xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }
      },
      empty
    }
  vColl =
    ## (коллекция значений) представляет ту часть спецификации значений
    ## элементов, которая содержит множественные значения, организованные
```

```
  ## как простое множество, множество с повторяющимися элементами или список.
  element vColl {
    (fs | model.featureVal.single)*,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    ## (способ организации) показывает способ организации данного значения или совокупности
    ## значений как простого множества, множества с повторяющимися элементами или списка,
    attribute org {
    ## показывает, что данные значения организованы в виде простого множества,
    "set"
    |
    ## показывает, что данные значения организованы в виде множества
    ## с повторяющимися элементами (мультимножества).
    "bag"
    |
    ## показывает, что данные значения организованы в виде списка,
    "list"
  }?,
  empty
}
default =
  ## (стандартное значение элемента) представляет ту часть
  ## спецификации значений элементов, которая содержит
  ## значение, присваиваемое по умолчанию,
  element default {
    empty,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
vAlt =
  ## (дизъюнкция значений) представляет часть спецификации
  ## значений элементов, содержащую множество значений,
  ## только одно из которых может быть правильным.
  element vAlt {
    (model.featureVal, model.featureVal+),
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
vNot =
  ## (значение с отрицанием) представляет значение элемента,
  ## которое является инверсией его содержимого.
  element vNot {
    model.featureVal,
    att.global.attribute.xmlid,
    att.global.attribute.n,
    att.global.attribute.xmllang,
    att.global.attribute.xmlbase,
    empty
  }
vMerge =
  ## (объединенная коллекция элементов) представляет значение элемента,
  ## которое является результатом слияния значений элементов,
  ## содержащихся в его дочерних узлах, с применением способа организации,
  ## определенного атрибутом org.
```

```
    element vMerge {
      model.featureVal+,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      ## указывает способ организации результирующих слитых значений как обычного
      ## множества, множества с повторяющимися элементами или списка.
      attribute org {
      ## показывает, что результирующие значения организованы в виде множества,
      "set"
      |
      ## показывает, что результирующие значения организованы в виде множества
      ## с повторяющимися элементами (мультимножества).
      "bag"
      |
      ## показывает, что результирующие значения организованы в виде списка,
      "list"
    }?,
    empty
  }
  fLib =
    ## () компонует библиотеку свойств.
    element fLib {
      f+,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  fvLib =
    ## (feature-value library) компонует библиотеку
    ## повторно используемых значений элементов
    ## (включая сложные структуры элементов),
    element fvLib {
      model.featureVal*,
      att.global.attribute.xmlid,
      att.global.attribute.n,
      att.global.attribute.xmllang,
      att.global.attribute.xmlbase,
      empty
    }
  start = fsdDecl
```

**Приложение В**
**(справочное)**

## Детализированный пример

В заключение ниже полностью воспроизводится FSD из фрагментарного примера, рассмотренного в разделе 8:

```
<TEI>
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>Пример FSD, заимствованный из книги Gazdar et al GPSG feature system for English</title>
    <respStmt>
     <resp>запрограммировал</resp>
     <name>Гэри Ф. Саймонс</name>
    </respStmt>
   </titleStmt>
   <publicationStmt>
    <p>Данный пример был впервые закодирован Гэри F. Саймонсом (Summer Institute of Linguistics, Dallas, TX) 28 января 1991 г.
Пересмотрен 8 апреля 1993 г. для приведения в соответствие со спецификацией FSD из версии P2 руководящих материалов TEI. Повторно пересмотрен в декабре 2004 г. для обеспечения соответствия стандарту представления структуры элементов, разработанному совместно с ISO TC37/SC4.
    </p>
   </publicationStmt>
   <sourceDesc>
    <p>Настоящий пример FSD не содержит полного описания системы элементов.
Он основан на фрагментах, заимствованных из системы элементов для английского языка, представленной в приложении (с. 245—247) к книге Generalized Phrase Structure Grammar, by Gazdar, Klein, Pullum, and Sag (Harvard University Press, 1985).</p>
   </sourceDesc>
  </fileDesc>
 </teiHeader>
 <fsdDecl>
  <fsDecl type="GPSG">
   <fsDescr>Кодирует структуру элементов английского языка для анализа с использованием грамматики GPSG (Gazdar, Klein, Pullum, and Sag)</fsDescr>
   <fDecl name="INV">
    <fDescr>инвертированное предложение</fDescr>
    <vRange>
     <vAlt>
      <binary value="true"/>
      <binary value="false"/>
     </vAlt>
    </vRange>
    <vDefault>
     <binary value="false"/>
    </vDefault>
   </fDecl>
   <fDecl name="CONJ">
    <fDescr>поверхностная форма конъюнкции</fDescr>
    <vRange>
     <vAlt>
      <symbol value="and"/>
      <symbol value="both"/>
      <symbol value="but"/>
      <symbol value="either"/>
      <symbol value="neither"/>
```

```
 <symbol value="nor"/>
 <symbol value="or"/>
 <symbol value="NIL"/>
 </vAlt>
</vRange>
<vDefault>
 <binary value="false"/>
</vDefault>
</fDecl>
<fDecl name="COMP">
<fDescr>поверхностная форма комплементайзера</fDescr>
<vRange>
 <vAlt>
  <symbol value="for"/>
  <symbol value="that"/>
  <symbol value="whether"/>
 <symbol value="if"/>
 <symbol value="NIL"/>
</vAlt>
</vRange>
<vDefault>
 <if>
  <fs>
   <f name="VFORM">
    <symbol value="INF"/>
   </f>
   <f name="SUBJ">
    <binary value="true"/>
   </f>
  </fs>
  <then/>
  <symbol value="for"/>
 </if>
</vDefault>
</fDecl>
<fDecl name="AGR">
<fDescr>соглашение о лице и числе</fDescr>
<vRange>
 <fs type="Agreement"/>
</vRange>
</fDecl>
<fDecl name="PFORM">
<fDescr>словоформа предлога</fDescr>
<vRange>
 <vNot>
  <string/>
  </vNot>
 </vRange>
</fDecl>
<fsConstraints>
 <cond>
  <fs>
   <f name="INV">
    <binary value="true"/>
   </f>
  </fs>
  <then/>
  <fs>
   <f name="AUX">
    <binary value="true"/>
```

```
  </f>
  <f name="VFORM">
   <symbol value="FIN"/>
  </f>
 </fs>
</cond>
<bicond>
 <fs>
  <f name="BAR">
   <symbol value="0"/>
  </f>
 </fs>
 <iff/>
 <fs>
  <f name="N">
   <binary value="true"/>
  </f>
  <f name="V">
   <binary value="true"/>
  </f>
  <f name="SUBCAT">
   <binary value="true"/>
  </f>
 </fs>
</bicond>
<cond>
 <fs>
  <f name="BAR">
   <symbol value="1"/>
  </f>
 </fs>
 <then/>
 <fs>
  <f name="SUBCAT">
   <binary value="false"/>
  </f>
 </fs>
</cond>
</fsConstraints>
</fsDecl>
<fsDecl type="Agreement">
<fsDescr>Этот тип структуры элементов кодирует соглашение о порядке слов в английском предложении</fsDescr>
<fDecl name="PERS">
<fDescr>лицо (первое, второе или третье)</fDescr>
<vRange>
 <vAlt>
  <symbol value="1"/>
  <symbol value="2"/>
  <symbol value="3"/>
 </vAlt>
</vRange>
</fDecl>
<fDecl name="NUM">
<fDescr>число (единственное или множественное)</fDescr>
<vRange>
 <vAlt>
 <symbol value="sg"/>
 <symbol value="pl"/>
</vAlt>
```

```
    </vRange>
   </fDecl>
  </fsDecl>
  </fsdDecl>
 </TEI>
```

**Приложение ДА**
**(справочное)**

## Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

| Обозначение ссылочного международного стандарта | Степень соответствия | Обозначение и наименование соответствующего национального стандарта |
|---|---|---|
| ИСО 24610-1:2006 | — | * |
| ИСО/МЭК 19757-2:2008 | — | * |
| * Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов. | | |

## Библиография


[1] Carpenter B. *The Logic of Typed Feature Structures.* — Cambridge University Press — Cambridge — 1992

[2] Copestake A. *Implementing Typed Feature Structure Grammars.* — CSLI Publications. — Stanford. — 2002

[3] Flickinger, D. On building a more efficient grammar by exploiting types. In: *Collaborative Language Engineering* (ed. Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii and Hans Uszkoreit), CSLI Publications, Stanford. — 2002. — pp. 1—17

[4] Gazdar G., Klein E., Pullum G. and Sag I. Generalized Phrase Structure Grammar. — Harvard University Press. — Cambridge, MA. — 1985

[5] Johnson M., Attribute-Value Logic and the Theory of Grammar — CSLI Lecture Notes 16. — Stanford. — 1988

[6] Kay M., Unification. In: *Computational Linguistics and Formal Semantics* (ed. Michael Rosner and Roderick Johnson). — Cambridge University Press. — Cambridge. — 1992. — pp. 1—30

[7] Langendoen D.T. and Simons G.F. A rationale for the TEI recommendations for feature-structure markup. — *Computers and the Humanities*, 29. — 1995. — pp. 191—209

[8] Pereira F.C.N. *Grammars and Logics of Partial Information, SRI International Technical Note 420.* — SRI International, Menlo Park, CA. — 1987

[9] Pollard C.J. and Sag I.A. Information-based Syntax and Semantics. — Vol. 1 Fundamentals. — *CSLI Lecture Notes*, **13**, Stanford, 1987

[10] Pollard C.J. and Sag I.A., *Head-driven Phrase Structure Grammar.* — The University of Chicago Press. — Chicago. — 1994

[11] Pollard, C.J. and Moshier, M.A. Unifying partial descriptions of sets. In: *Information, Language and Cognition* (ed. Philip P. Hanson) — The University of British Columbia Press. — Vancouver. — 1990, pp. 285—322

[12] Sag I.A., Wasow T. and Bender E.M. *Syntactic Theory: A Formal Introduction.* — 2nd edition. — CSLI Publications. — Stanford. — 2003

[13] Shieber S.M. An Introduction to Unification-Based Approaches to Grammar. — *CSLI Lecture Notes*, **4**, Stanford. — 1986

[14] Text Encoding Initiative Consortium. — *The TEI Guidelines. — P5.* — 2005

[15] Vijay-Shanker K. and Joshi A.K. Feature-structure based tree adjoining grammar. *Proceedings of COLING'88.* — 1988

УДК 001.4:006.354 ОКС 01.140.20

Ключевые слова: управление языковыми ресурсами, декларация системы элементов, терминология

Редактор *Н.А. Аргунова*
Технический редактор *В.Н. Прусакова*
Корректор *Ю.М. Прокофьева*
Компьютерная верстка *И.А. Налейкиной*

Сдано в набор 06.04.2015. Подписано в печать 11.06.2015. Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 5,58. Уч.-изд. л. 4,80. Тираж 36 экз. Зак. 2133.

Издано и отпечатано во ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru