МСЭ-Т

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

Z.105

(07/2003)

СЕРИЯ Z: ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

Методы формального описания (FDT) – Язык спецификации и описания (ЯСО)

ЯСО, объединенный с модулями ASN.1 (ЯСО/ASN.1)

Рекомендация МСЭ-Т Z.105

РЕКОМЕНДАЦИИ МСЭ СЕРИИ Z

ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

МЕТОДЫ ФОРМАЛЬНОГО ОПИСАНИЯ (FDT)	
Язык спецификации и описания (ЯСО)	**Z.100–Z.109**
Применение методов формального описания	Z.110–Z.119
Диаграмма последовательности сообщений (MSC)	Z.120–Z.129
Расширенный язык определения объектов (eODL)	Z.130–Z.139
Тестирование и нотация управления тестами (TTCN)	Z.140–Z.149
Нотация требований пользователя (URN)	Z.150–Z.159
ЯЗЫКИ ПРОГРАММИРОВАНИЯ	
CHILL: язык высокого уровня МСЭ	Z.200–Z.209
ЯЗЫК "ЧЕЛОВЕК-МАШИНА"	
Общие принципы	Z.300–Z.309
Базовый синтаксис и диалоговые процедуры	Z.310–Z.319
Расширенный язык MML для видеотерминалов	Z.320–Z.329
Спецификация интерфейса "человек-машина"	Z.330–Z.349
Информационно-ориентированные интерфейсы "человек-машина"	Z.350–Z.359
Интерфейсы "человек-компьютер" для управления сетями электросвязи	Z.360–Z.369
КАЧЕСТВО	
Качество программного обеспечения электросвязи	Z.400–Z.409
Аспекты качества, ориентированные на протокол Рекомендаций	Z.450–Z.459
МЕТОДЫ	
Методы проверки достоверности и тестирования	Z.500–Z.519
ПРОМЕЖУТОЧНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ	
Среда распределенной обработки	Z.600–Z.609

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ.

Рекомендация МСЭ-Т Z.105

ЯСО, объединенный с модулями ASN.1 (ЯСО/ASN.1)

Резюме

Настоящая Рекомендация определяет, как модули Абстрактной синтаксической нотации (АСН) версии 1 (ASN.1) могут быть использованы в сочетании с Языком спецификации и описания (ЯСО). Этот текст заменяет семантические отображения от ASN.1 в ЯСО, определенные в Рекомендации МСЭ-Т Z.105 (1999). Использование нотации ASN.1, встроенной в ЯСО, ранее определенное в Рекомендации МСЭ-Т Z.105 (1995), не определяется этой Рекомендацией.

Основной сферой применения этой Рекомендации является спецификация телекоммуникационных систем. Совместное использование ЯСО и ASN.1 предоставляет согласованный путь для указания структуры и поведения телекоммуникационных систем вместе с данными, сообщениями и кодированием сообщений, которые эти системы используют.

Настоящая версия Рекомендации Z.105 делает необходимыми выравнивания с Рекомендациями ASN.1 2002. Важные изменения включают отображение значений языка XML, улучшенное отображение значений битовых строк и добавленное отображение соответствующих конструкций ASN.1 для расширений.

Источник

Рекомендация МСЭ-Т Z.105 была утверждена 17-й исследовательской комиссией 17 МСЭ-Т (2001–2004 гг.) по Рекомендации МСЭ-Т A.8, процедура от 7 июля 2003 года.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи. Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

Всемирная ассамблея по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяет темы для изучения Исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение Рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соответствие положениям данной Рекомендации является добровольным делом. Однако в Рекомендации могут содержаться определенные обязательные положения (для обеспечения, например, возможности взаимодействия или применимости), и тогда соответствие данной Рекомендации достигается в том случае, если выполняются все эти обязательные положения. Для выражения требований используются слова "shall" ("должен", "обязан") или некоторые другие обязывающие термины, такие как "must" ("должен"), а также их отрицательные эквиваленты. Использование таких слов не предполагает, что соответствие данной Рекомендации требуется от каждой стороны.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на то, что практическое применение или реализация этой Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, обоснованности или применимости заявленных прав интеллектуальной собственности, независимо от того, отстаиваются ли они членами МСЭ или другими сторонами вне процесса подготовки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ не получил извещение об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для реализации этой Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что это может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ.

СОДЕРЖАНИЕ

Введение

- **Цель**

Настоящая Рекомендация определяет, как модули Абстрактной синтаксической нотации версии 1 (ASN.1) могут быть использованы в сочетании с Языком спецификации и описания (ЯСО). Замысел состоит в том, что структура и поведение систем описываются с помощью языка ЯСО, в то время как параметры обмениваемых сообщений описываются через ASN.1. Настоящая Рекомендация определяет отображение конструкций ASN.1 в уже существующих конструкциях ЯСО, и содержит только небольшое расширение для Рекомендации МСЭ-Т Z.100, позволяющее использовать модули ASN.1.

- **Охват**

Настоящая Рекомендация представляет семантическое определение для комбинации ЯСО и модулей ASN.1. Представлено отображение данных ASN.1, определяемых в модуле в соответствующих конструкциях ЯСО, которые определены в Рекомендации МСЭ-Т Z.100 [1], включая операторов, которые могут применяться для данных ASN.1. Пункты данных ASN.1 могут затем использоваться внутри ЯСО (используя нотацию ЯСО).

Использование нотации ASN.1, встроенной в ЯСО, определено в Рекомендации МСЭ Z.107 [2].

- **Применение**

Основной сферой применения этой Рекомендации является спецификация систем электросвязи. Совместное использование ЯСО и ASN.1 предоставляет согласованный путь для указания структуры и поведения систем электросвязи, вместе с данными, сообщениями и кодированием сообщений, которые эти системы используют.

ПРИМЕЧАНИЕ. – "Спецификация" в этой Рекомендации включает определение требований в стандартном, типа Рекомендации, документе, или в документе на поставку продукции, и описание реализации.

Спецификация согласуется с этой Рекомендацией, если и только если она согласуется с синтаксическими и семантическими грамматическими правилами для формального технического языка, определяемого Рекомендацией (который включает опорные языки ASN.1 и ЯСО). Согласие подразумевает, что каждая возможная динамическая интерпретация спецификации согласуется с правилами языка. Спецификация, которая использует расширения языка, не согласуется.

Инструмент не полностью поддерживает язык, если он отклоняет некоторые конструкции языка или тот имеет статическую или динамическую интерпретацию спецификации на языке, который не соответствует языковой семантике.

- **Статус/стабильность**

Этот текст заменяет семантические отображения из ASN.1 в ЯСО, которые определены в Рекомендации МСЭ-Т Z.105 (1999). Использование нотации ASN.1, вложенной в ЯСО, ранее определенное в Рекомендации МСЭ-Т Z.105 (1995), не определяется в этой Рекомендации.

Изменения для Рекомендаций МСЭ-Т X.680 [3], X.681 [4], X.682 [5] и X.683 [6], или Z.100 [1] могут потребовать модификаций для этой Рекомендации.

Настоящая Рекомендация является полным справочным руководством, описывающим соединение ЯСО и модулей ASN.1.

Рекомендация МСЭ-Т Z.105

ЯСО, объединенный с модулями ASN.1 (ЯСО/ASN.1)

1 Обзор

Настоящая Рекомендация определяет, как модули ASN.1 могут быть использованы в сочетании с ЯСО. Модули ASN.1 важны в описаниях ЯСО, так что определения данных ASN.1 отображаются во внутреннее представление ЯСО, используя эквивалентные конструкции ЯСО и формируя совместно с остальным описанием ЯСО полную спецификацию.

ЯСО является языком для спецификации и описания телекоммуникационных систем. ЯСО имеет концепцию для:

- структурирования систем;
- определения поведения систем;
- определения данных, используемых системами.

ASN.1 является языком для определения данных. С языком ASN.1 связаны правила кодирования, которые определяют, как значения ASN.1 передаются в виде битовых потоков во время коммуникации.

1.1 Цель

Объединение ЯСО и ASN.1 предоставляет согласованный путь для указания структуры и поведения телекоммуникационных систем вместе с данными, сообщениями и кодированием сообщений, которые эти системы используют. Структура и поведение могут быть описаны с использованием ЯСО, а данные и сообщения, используя ASN.1. Кодирование этих сообщений может быть описано через ссылку на соответствующие правила кодирования, которые определяются для ASN.1.

Полное использование ЯСО (включая типы данных) поддерживается этой Рекомендацией.

1.2 Характеристики комбинации ЯСО и модулей ASN.1

Системы, описываемые ЯСО, объединенным с модулями ASN.1, имеют следующие характеристики:

- структура и поведение определяются с использованием концепций ЯСО;
- сигнальная структура ЯСО, то есть, типы параметров сигнала и их субтипы определяются в модулях ASN.1;
- внутренние данные могут определяться либо типами ASN.1, либо сортами ЯСО;
- значения данных кодирования, определяемые в ASN.1, могут быть определены через ссылку на соответствующие правила кодирования. Кодирование не входит в сферу действия этой Рекомендации.

1.3 Нотация ASN.1, которая может использоваться в сочетании с ЯСО

Использование ASN.1, как это определено в Рекомендациях МСЭ-Т X.680, X.681, X.682 и X.683, поддерживается в сочетании с ЯСО, при этом надо осознать, что некоторые конструкции ASN.1 не могут быть успешно отображены в ЯСО (или, по крайней мере, отображение не идентифицируется и не указывается в этой Рекомендации). Конструкции, которые не могут быть отображены в ЯСО, будут существовать в пакетах ASN.1, используемых как источник преобразования. Во время преобразования в ЯСО они эффективно обрабатываются так, как будто бы они отсутствуют и не вызывают никаких проблем для успешного преобразования других конструкций. Такими конструкциями являются маркер расширения и маркер исключения, определяемые в Рекомендации МСЭ-Т X.680, которые могут быть представлены в ASN.1, но игнорируются при преобразовании в ЯСО. Некоторые конструкции ASN.1 никогда не преобразуются в ЯСО как таковые, но содержат информацию, которая может быть направлена или использована в преобразовании. Известными примерами таких конструкций являются относительные ограничения, как определено в Рекомендации МСЭ-Т X.682, классов информационных объектов и наборов информационных объектов (см. статью 5).

Использование ЯСО поддерживается, как определено в Рекомендации МСЭ-Т Z.100 [1].

Модули ASN.1, которые используются в преобразовании в ЯСО, могут также использоваться для генерации кодеров и декодеров, обеспечивающих заданные правила кодирования. Спецификация данных ЯСО, неявно выводимая из модулей ASN.1, не должна использоваться для генерации кодеров и декодеров, так как часть информации, которая имеет отношение к кодированию, может быть потеряна при преобразовании в ЯСО.

1.4 Структура этой Рекомендации

Настоящая Рекомендация не является самодостаточной: отображение, определяемое в этой Рекомендации, базируется на Рекомендации МСЭ-Т Z.100 и Рекомендациях МСЭ-Т X.680, X.681, X.682 и X.683. Язык применяется так, как это определено в Рекомендации МСЭ-Т Z.100, за исключением того факта, что правило формирования пакета (<package>) расширено, чтобы разрешить прямое использование модулей ASN.1. Настоящая Рекомендация структурирована следующим образом:

Статья 3 определяет изменения в Рекомендации МСЭ-Т Z.100 для включения модулей ASN.1.

Статья 4 определяет отображение типов ASN.1 в Рекомендации МСЭ-Т X.680 и значений в Рекомендации МСЭ-Т Z.100 для того, чтобы включить типы и значения данных ASN.1.

Статья 5 определяет отображение типов ASN.1, определяемых с использованием информационных объектов, классов и наборов информационных объектов. Использование конструкций Рекомендации МСЭ-Т X.682 также рассматривается в этой статье.

Статья 6 определяет отображение параметризованных типов ASN.1 для данных Рекомендации МСЭ-Т Z.100 для включения параметризованных типов данных ASN.1.

Статья 7 определяет добавления в пакет Predefined (предопределенные), необходимые для поддержки использования ASN.1.

1.5 Соглашения, использованные в этой Рекомендации

Как правило, применяются соглашения Рекомендации МСЭ-Т Z.100: например, ключевые слова появляются как строчные буквы полужирного шрифта, и заранее заданные имена начинаются с заглавной буквы. Однако, в примерах ASN.1, используются соглашения ASN.1 для того, чтобы не нарушать правила ASN.1 и улучшить читабельность для пользователей ASN.1: например, ключевые слова появляются с заглавной буквы.

Для грамматических конструкций ASN.1 даются ссылки на исходные документы. Однако грамматические конструкции ASN.1 также скопированы в эту Рекомендацию в тех местах, где чувствуется необходимость повысить ее читабельность. В случае конфликта предпочтение отдается оригинальной конструкции ASN.1.

2 Ссылки

Следующие Рекомендации МСЭ-Т и другие ссылки содержат обеспечение, которое, через ссылки в тексте, предоставляет поддержку для настоящей Рекомендации. На момент публикации указанные редакции являются действующими. Все Рекомендации и другие ссылки подлежат пересмотру; поэтому пользователям настоящей Рекомендации рекомендуется изучать возможность применения самой последней редакции Рекомендации и других ссылок, перечисленных ниже. Список текущих действующих Рекомендаций МСЭ-Т регулярно публикуется. Ссылка на документ внутри этой Рекомендации не указывает на статус Рекомендации, как на отдельный документ.

[1] ITU-T Recommendation Z.100 (2002), *Specification and Description Language (SDL)*.

[2] ITU-T Recommendation Z.107 (1999), *SDL with embedded ASN.1*.

[3] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[4] ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.

[5] ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.

[6] ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

3 Пакет

Грамматика ASN.1

ModuleDefinition (определение модуля), **ModuleIdentifier** (идентификатор модуля), **DefinitiveIdentifier** (окончательный идентификатор), **Imports** (импорт) и **Exports** (экспорт) определяются в 12.1/X.680.

Модель

Продукция <package> (пакет) расширяется следующим образом:

```
<package> ::=     (пакет)
          <package definition> | <package diagram> | <module definition>
          (<определение пакета>      <схема пакета>       <определение модуля>)
<module definition>     ::=
          ModuleDefinition    (определение модуля)
```

где **ModuleDefinition** не окончательно определен в Рекомендации МСЭ-Т X.680.

<module definition> имеет такое же значение, как <package definition> где:

- **ModuleIdentifier** (идентификатор модуля) (без любого **DefinitiveIdentifier** – (окончательный идентификатор)) соответствует <package name> (имя пакета);
- **Imports (импорт)** соответствует <package use clause> (оператор использования пакета);
- **Exports (экспорт)** соответствует <interface> (интерфейсу).

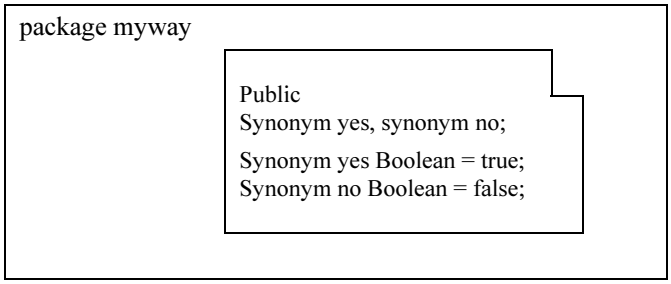
Пакет ASN.1 преобразуется в эквивалентный ЯСО, перед тем как он будет рассматриваться как пакет, и перед любыми преобразованиями Z.100. В случае этого преобразования имена преобразуются в полные квалифицированные идентификаторы, когда ЯСО запрашивает или разрешает идентификатор вместо имени. Однако, для краткости, это часто опускается в примерах в этой Рекомендации.

Пример

Определение модуля ASN.1:

```
myway DEFINITIONS ::=
     BEGIN
     EXPORTS yes, no;
     yes  BOOLEAN ::= TRUE
     no   BOOLEAN ::= FALSE
END
```

это то же самое, что и:



Аналогично, когда пакет используется в импорте **(imports)** другого пакета:

```
IMPORTS yes FROM myway;
```

Это то же самое, что и <package reference clause> (оператор ссылки на пакет):

use myway/yes;

ПРИМЕЧАНИЕ. – Так как ЯСО не поддерживает значений идентификатора объекта для идентификации пакета, модули ASN.1 с тем же **modulereference** (ссылка на модуль)**,** но отличными **DefinitiveIdentifiers** (окончательные идентификаторы) будут потенциально вызывать проблемы разрешения имени.

4 Определение и использование данных

Различные определения использования данных описываются следующим образом:

Грамматика ASN.1	Определение правил грамматической продукции, представляющих ту конструкцию, которая должна быть представлена в ЯСО.
Модель	Описание преобразований различных частей грамматики ASN.1 в продукции ЯСО. Эта часть обращается как к грамматике ЯСО, представленной в виде <SDL grammar rule> (грамматические правила ЯСО), так и к грамматике ASN.1, представленной в виде **ASN.1GrammarRule** (грамматические правила ASN.1).

4.1 Отображение имени

Грамматика ASN.1

Имена ASN.1 соответствуют лексическим пунктам ASN.1, определяемым в Рекомендации МСЭ X.680, 11, так что только буквы, цифры и дефис разрешены в именах ASN.1. Если символ дефиса используется в ЯСО, он будет интерпретироваться как оператор минус.

Модель

Имена ASN.1, содержащие символ дефиса, отображаются в лексически сходных именах ЯСО, кроме того, что символы дефиса преобразуются в символы подчеркивания.

Пример

Имя ASN.1 **`my-example-name`** отображается в my_example_name в SDL.

4.2 Определения переменных и данных

4.2.1 Назначение типа

Грамматика ASN.1

TypeAssignment (назначение типа), **Type** (тип), **constrainedType** (ограничение типа) и **Constraint** (ограничение) определяются в 15.1/X.680.

Модель

Если **Type** это **typereference** (тип ссылки), тогда **TypeAssignment** то же самое, что и <syntype definition> (определение синтакс. типа), содержащее только эквивалент ЯСО для **Type**.

Если **Type** это **constrainedType**, тогда **TypeAssignment** то же самое, что и <syntype definition> содержащее только эквивалент ЯСО для **Constraint**.

Если **Type** это ни **typereference,** ни **constrainedType**, тогда **TypeAssignment** представляется через <partial type definition> (частичное определение типа), где поле < properties expression> (выражение свойств) остается пустым, и где поле <formal context parameters > (формальные контекстные параметры) пропущено.

Пример

Назначение типа ASN.1:

```
Mytype ::= AnotherType -- typereference
```

то же самое, что и:

syntype Mytype = AnotherType **endsyntype** Mytype; /* здесь опущена полная квалификация. */

Назначение типа ASN.1:

```
S ::= INTEGER (0..5 | 10)
```

то же самое, что и:

syntype S = <<**package** Predefined>> Integer **constants** (0..5,10) **endsyntype** S;

Назначение типа ASN.1:

```
Integerlist ::= SEQUENCE OF INTEGER
```

то же самое, что и:

value type Integerlist {**inherits** <<**package** Predefined>>String
< <<**package** Predefined>> Integer > (" = <<**package**
Predefined>>Emptystring)
}

4.2.2 Назначение значения

Грамматика ASN.1

ValueAssignment (назначение значения) и **XMLValueAssignment** (назначение значения XML) определяются в 15.2/X.680.

Модель

ValueAssignment и **XMLValueAssignment** представляются через <synonym definition item> (пункт определения синонима).

Пример

Определение ASN.1:

```
yes BOOLEAN ::= TRUE
```

то же самое, что и:

synonym yes <<**package** Predefined>>Boolean = <<**package** Predefined>> true;

4.3 Типы выражений

4.3.1 Последовательность

Грамматика ASN.1

SequenceType (тип последовательн.), **ComponentType** (тип компонента), **ExtensionAndException** (расширение и исключение), **OptionalExtensionMarker** (дополнительный маркер расширения), **ExtensionAdditionGroup** (группа добавления расширения) и **VersionNumber** (номер версии) определяются в 24.1/X.680. **SetType** (тип установки) определяется в 26.1/X.680.

Модель

SequenceType представляется как <structure definition> (определение структуры), содержащее <field> (поле) для каждого **NamedType** (именной тип) **SequenceType**. Поле (<field>) содержит одно имя поля (<field name>), которое то же самое, что и **identifier** (идентификатор) ASN.1

для **NamedType**, а <field sort> (сортировка поля), это тип (**Type),** преобразованный в идентификатор сортировки (<sort identifier>) ЯСО.

Если поле **ComponentType,** содержащее **NamedType,** это **OPTIONAL** (опционное), тогда поле ЯСО будет содержать ключевое слово **optional** (опционное).

Если поле **ComponentType,** содержащее **NamedType** включает в себя **DEFAULT Value** (значение по умолчанию), поле ЯСО будет содержать ключевое слово по умолчанию **(default)** и значение, преобразованное в <constant expression> (константное выражение) после **default** (значение по умолчанию).

Поле **ComponentType** это **COMPONENTS OF Type** (компоненты типа), представленные в виде списка упорядоченных полей (<field>), по одному для каждого поля, связанного с типом (**Type)**. Эти поля вставляются в позиции списка **COMPONENTS OF Type** в том порядке, в каком поля существуют в типе **(Type)**.

Присутствие полей **ExtensionAndException** и **OptionalExtensionMarker** в **SequenceType** игнорируется в преобразовании.

Присутствие поля **ExtensionAdditionGroup** (группа добавления расширения) в **ExtensionAddition** (добавление расширения) преобразуется так, что скобки версии ("[[","]]") и **VersionNumber** (номер версии) игнорируются.

Пример

Тип ASN.1:

```
S    ::=  SEQUENCE {
       a  INTEGER,
       b  IA5String OPTIONAL,
       c  PrintableString DEFAULT "d"}
```

то же самое, что и:

value type S
{
struct
a <<**package** Predefined>> Integer;
b <<**package** Predefined>> IA5String optional;
c <<**package** Predefined>> PrintableString default 'd';
}

ПРИМЕЧАНИЕ 1. – Нет разницы между использованием ключевого слова **SEQUENCE (последовательность)** и **SET (установка)**. Это упрощение по сравнению с Рекомендацией МСЭ X.680.

ПРИМЕЧАНИЕ 2. – В этой Рекомендации, теги не являются необходимыми для выявления различия между компонентами одного и того же типа: Предполагается автоматическое тегирование ASN.1.

4.3.2 Последовательность из

Грамматика ASN.1

SequenceOfType (последовательность типов) определяется в 25.1/X.680.

Модель

Указание **SequenceOfType** то же самое, что и указание предопределенной сортировки строк, имеющей поле ЯСО преобразования типа **(Type)** в качестве первого *фактического контекстного параметра* (<actual context parameter>) и имя Emptystring (пустая строка), определяемое как литеральное имя для пустой строки.

Если ограничение размера ASN.1 указывается для типа (**Type)**, тогда **SequenceOfType** является синтаксическим типом (syntype), имеющим ограничение на преобразованный размер как *диапазонное условие* (<range condition>) (см. 4.4). Родственной сортировкой для syntype является **SequenceOfType** без ограничения размера ASN.1. Эта родственная сортировка имеет неявное и уникальное имя и определяется в самой ближайшей области блока, заключающей в себе присутствие **SequenceOfType**.

Пример

Определение ASN.1:

```
phonenumber ::= SEQUENCE SIZE (8) OF INTEGER (0..9)
```

то же самое, что и три определения ЯСО:

value type S1
{
inherits <<**package** Predefined>> String <S2> (" = Emptystring)
}
syntype S2 = <<**package** Predefined>> Integer **constants** (0..9) endsyntype;
syntype phonenumber = S1 **constants** size (8) endsyntype phonenumber;

4.3.3 Выбор

Грамматика ASN.1

ChoiceType (тип выбора), **ExtensionAdditionAlternative** (альтернативное добавление расширения) и **ExtensionAdditionAlternativesGroup** (группа альтернативного добавления расширения) определяются в 28.1/X.680.

Модель

ChoiceType представляется как *определение выбора* (<choice definition>), содержащее *поле* (<field>) для каждого **NamedType** (именной тип) в **ChoiceType**.

Присутствие **ExtensionAndException** и **OptionalExtensionMarker** в **ChoiceType** игнорируются в преобразовании.

Присутствие **ExtensionAdditionAlternativesGroup** в **ExtensionAdditionAlternatives** преобразуются так, что скобки версии ("[[", "]]") и **VersionNumber** игнорируются.

Пример

Выбор типа ASN.1:

```
C ::= CHOICE {
a    INTEGER,
b    REAL }
```

то же самое, что и:

value type C **Choice**
{
a <<**package** Predefined>> Integer;
b <<**package** Predefined>> Real;
}

4.3.4 С нумерацией

Грамматика ASN.1

EnumeratedType (нумерованный тип), **EnumerationItem** (пункт перечисления) и **ExceptionSpec** (исключение спецификации) определяются в 19.1/X.680.

Модель

Поле **EnumeratedType** представляется через *частичное определение типа* (<partial type definition>), где <properties expression> (выражение свойств) пусто, и где <formal context parameters> (формальные контекстные параметры) пропущены. Для каждого **EnumerationItem**, **identifier** (идентификатор) преобразуется в <literal signature> (литеральную сигнатуру), которая имеет то же самое имя, как и **EnumerationItem**. Если **EnumerationItem** содержит **SignedNumber** (знаковое число) (или **DefinedValue** (определенное значение)), тогда литеральное имя (<literal name>) литеральной сигнатуры (<literal signature>) следует перед преобразованием ЯСО для **SignedNumber** (или **DefinedValue,** соответственно).

Маркеры расширения ("...") и **ExceptionSpec** в **EnumeratedType** игнорируются в преобразовании в ЯСО.

Определение:

```
colours ::= ENUMERATED {blue(3),red, yellow(0)}; (цвета синий-3, красный, желтый-0)
```

то же самое, что и:

value type colours {
literals blue = 3, red, yellow = 0
}

4.3.5 Целое число

Грамматика ASN.1

IntegerType (тип целого числа), **NamedNumberList** (список именованных чисел) и **NamedNumber** (именованное число) определяются в 18.1/X.680.

Модель

Тип целого числа **IntegerType** ASN.1 отображается в <<**package** Predefined>>Integer (предопределенный пакет).

Указание **IntegerType** вместе с **NamedNumberList** то же самое, что и указание <synonym definition> (определение синонима) в ближайшем окружении блока с одним *пунктом синонимического определения* (<synonym definition item>) для каждого **NamedNumber**. *Идентификатор* (**identifier)** числа **NamedNumber** преобразуется в *синонимическое имя* (<synonym name>). *Сортировка* (<sort>) *пунктов синонимического определения* (<synonym definition item>) соответствует целому числу <<**package** Predefined>> (предопределенный пакет). **SignedNumber** (знаковое число) или **DefinedValue** (определенное значение) для **NamedNumber** используется как *константное выражение* (<constant expression>) для *пункта синонимического определения* <synonym definition item>.

Пример

Определение ASN.1:

```
Standards ::= SEQUENCE OF INTEGER{z100(0),x680(1),z10x(2)}
```

то же самое, что и:

value type standards {
inherits
<<**package** Predefined>>String<<<**package** Predefined>> Integer > ("= EmptyString)
}
synonym z100 Integer = 0;
synonym x680 Integer = 1,
synonym z10x Integer = 2;

4.3.6 Диапазон значений

Грамматика ASN.1

ValueRange (диапазон значений) определяется в 47.4/X.680.

Модель

Указание ограничения ASN.1 для **ValueRange** представляется как указание, содержащее *сортировку* (<sort>) и добавление представления ограничения ASN.1 для **ValueRange** после ключевого слова **constants** (константы) в <syntype> (синтаксический тип).

Пример

Определение ASN.1:

```
S    ::=  INTEGER(0..5 | 10)
```

эквивалентно:
syntype S = <<**package** Predefined>> Integer **constants** (0..5, 10) endsyntype S;

Как выводится *диапазонное условие* (<range condition>), описывается ниже.

4.3.7 Битовая строка

Грамматика ASN.1

`BitStringType` (тип битовой строки), **`NamedBitList`** (именованный битовый список) и **`NamedBit`** (именованный бит) определяются в 21.1/X.680.

Модель

Поле **BitStringType** ASN.1 отображается в <<**package** Predefined>>Bitstring.

Указание **BitStringType** с **NamedBitList** то же самое, что и указание <synonym definition> (синонимическое определение) в ближайшем окружении блока с *одним пунктом синонимического определения* (<synonym definition item>) для каждого **NamedBit**. *Идентификатор* (**identifier) NamedBit** преобразуется в *синонимическое имя* (<synonym name>). *Сортировка* (<sort>) *пунктов синонимического определения* (<synonym definition item>) является целым числом <<**package** Predefined>>. *Число* (**number)** или **DefinedValue** (предопределенное значение) используются как *константное выражение* (<constant expression>) для <synonym definition item>.

4.3.8 OctetString (Строка октетов)

Грамматика ASN.1

OctetStringType (тип строки октетов) определяется в 22.1/X.680.

Модель

Тип ASN.1 **OctetStringType** отображается в <<**package** Predefined>>Octetstring.

4.3.9 Установка

Грамматика ASN.1

SetOfType (тип установки) определяется в 27.1/X.680.

Модель

Указание **SetOfType** то же самое, что и указание <<**package** Predefined>>BagSort, имеющего преобразование типа (**Type)** ЯСО в качестве первого *фактического контекстного параметра* (<actual context parameter>) и имя Emptybag (пусто множество), определяемое как литеральное имя для пустого множества (bag).

Если ограничение размера ASN.1 указывается для *типа* (**Type)**, **SetOfType** является синтаксическим типом (syntype), имеющим ограничение на преобразуемый размер как <range condition> (диапазонное условие) (см. 4.4). Родственной сортировкой для syntype является **SetOfType** без ограничения размера ASN.1. Эта родственная сортировка имеет неявное и уникальное имя и определяется в самой ближайшей области блока, заключающей в себе присутствие **SetOfType**.

4.4 Диапазонное условие

Модель

Диапазонное условие определяет набор значений. Оно используется в ЯСО для определения syntype (синтаксический тип). Оно имеет связанную родственную сортировку, которая является той сортировкой, которая указывает в определении syntype. Значение находится внутри набора значений, если оператор, обозначенный через поля идентификации оператора, будет "истинным" (**true**), когда он применяется к значению.

Идентификатор оператора для данного диапазонного условия, таким образом, определен как:

value type A

operators o: S -> Boolean;

/* где o выводится конкретного синтаксиса ASN.1, как пояснено ниже */

endvalue type A;

Каждый диапазон (Range) в диапазонном условии ASN.1 вкладывается в свойства оператора, определяющего набор значений:

o(V) == range1 или range2 или ... или rangeN (диапазон 1 … N)

Если syntype указывается без диапазонного условия, тогда значение оператора будет **true**.

В дальнейшем пояснении того, как каждый диапазон (Range) вносит вклад в результат оператора, V обозначает значение аргумента. Каждый вклад должен быть правильно сформированным, а это означает, что используемые операторы должны существовать с сигнатурой, подходящей для контекста.

- Если никакое из ключевых слов **MIN** и **MAX** не указывается в **ClosedRange** (закрытый диапазон), тогда **ClosedRange** вкладывается с:

  E1 rel1 V и V rel2 E2

  где E1 - значение **LowerEndValue** (значение нижнего конца) и E2 - значение **UpperEndValue** (значение верхнего конца).

  Если "<" указывается для **LowerEndValue,** тогда rel1 это оператор "<", в противном случае, это оператор "<=".

  Если "<" указывается для **UpperEndValue** тогда rel2 это оператор "<" в противном случае, это оператор "<=".

  Если ключевое слово **MIN** указывается, и ключевое слово **MAX** не указывается, **Range** (диапазон) вкладывается с:

  V rel2 E2

  Если ключевое слово **MAX** указывается, а ключевое слово **MIN** не указывается, **Range** (диапазон) вкладывается с:

  E1 rel1 V

  Если оба ключевых слова **MIN** и **MAX** указываются, оператор всегда выдает **true**.

- **ContainedSubType** (содержащий субтип) вкладывается с:

  o1(V),

  где o1 – неявный оператор, определяющий набор значений для типа (**Type),** указанного в **ContainedSubType**.

- A **SizeConstraint** (ограничение размера) вкладывается с:

  o1(length(V)) (длина V),

  где o1 – неявный оператор, определяющий набор значений для *условия диапазона* (<range condition>), указанного в **SizeConstraint**.

- **InnerTypeConstraints** (ограничения внутреннего типа) вкладывается с либо:

  if length(V) = 0 **then** true **else** o1(first(V)) **and** o(Substring(V,2,length(V)-1)) **fi**; либо

  if length(V) = 0 **then** true **else** o1(take(V)) **and** o(del(take(V), V)) **fi**

  (if=если**,** length=длина, **then=**тогда, true=истина, **else**=иначе, first=первый **and**=и, Substring=субстрока, take=взять, del=стереть)

  то, что подходит для сортировки V. o – это неявный оператор **InnerTypeConstraints** вкладов в, а o1 – это неявный оператор для *диапазона* (**Range),** указанного в **InnerTypeConstraints**.

  InnerTypeConstraints имеет вклад для каждого содержащегося **NamedConstraint** (именованное ограничение), который указывает ограничения для поля (см. 4.2.1), обозначенные через *идентификатор* **(Identifier)** родственной сортировки.

  Для цели получения вкладов, выведенный тип ASN.1 создается следующим образом:

  a) ключевое слово **PRESENT** (присутствует) добавляется в **NamedConstraints,** которое не имеет конечного ключевого слова (**PRESENT** (присутствует), **ABSENT** (отсутствует) или **OPTIONAL** (дополнительно));

b) Ограничения **NamedConstraints** из формы **Identifier ABSENT** (идентификатор отсутствия) добавляются для всех полей (то есть идентификаторов), не указанных явно в **NamedConstraint**. **NamedConstraints** добавляются в **InnerTypeConstraints** перед тем, как вклады каждого **NamedConstraint** выводятся.

ПРИМЕЧАНИЕ. – В случае управляющего типа в CHOICE (выбор), выведенный тип может оказаться недопустимым по отношению к Рекомендации МСЭ X.680, но он используется только для цели отображения в ЯСО и не воздействует на первоначальный тип ASN.1 или его кодирование.

Если *диапазон* (**Range)** указывается для **NamedConstraint**, вкладом является:

E и если FPresent(V), тогда o1(V), иначе **true** fi,

где E – присутствующее ограничение для поля, F (от имени оператора FPresent) – это имя опционного поля, и o1 – это неявный оператор для *диапазона* **(Range)**. Если **Range** пропущен, вкладом является только присутствующее ограничение E.

Присутствующее ограничение для поля F:

FPresent(V)

в случае, если **NamedConstraint** для поля содержит ключевое слово **PRESENT**; и

not FPresent(V)

в случае если **NamedConstraint** для поля содержит ключевое слово **ABSENT**. Во всех случаях присутствующее ограничение соответствует значению истина (true).

4.5 Значения выражений

4.5.1 Значение выбора

Грамматика ASN.1

ChoiceValue (значение выбора) определяется в 28.10/X.680.

Модель

Поле **ChoiceValue** представляется как *операторное приложение* (<operator application>), имеющее **Value** (значение) как аргумент. *Идентификатор оператора* (<operator identifier>) в *операторном приложении* (<operator application>) содержит *квалификатор* (<qualifier>), представляющий *тип* (**Type)** и имя оператора, которым будет *идентификатор* (**identifier)**.

Пример

Поле `ChoiceValue`:

```
myvalue : Mychoice
```

представляется как:

myvalue(Mychoice)

В том случае, когда **ChoiceValue** может обозначить одно из нескольких операторных приложений (то есть, поле из более чем одного выбора сортировки), используется квалификатор:

```
MyType ::= CHOICE ……
myvalue : Mychoice
```

который затем представляется как:

<<type Mytype>> myvalue(Mychoice)

4.5.2 Composite primary (Составная первичность)

Составная первичность (composite primary) надстраивает значения для представлений ЯСО соответствующих составных типов.

4.5.2.1 Последовательное значение

Грамматика ASN.1

SequenceValue (последовательное значение), **XMLSequenceValue** (последовательное значение XML), **ComponentValueList** (список компонентных значений) и **XMLComponentValueList** (список компонентных значений XML) определяются в 24.17/X.680.

ПРИМЕЧАНИЕ. – Нет разницы между **`SetValue (установить значение)`** и **`SequenceValue`**. Это упрощение по сравнению с Рекомендацией МСЭ X.680.

Модель

SequenceValue и **XMLSequenceValue** отображены в *пункте синонимического определения* (<synonym definition item>). В отображении, **ComponentValueList** или **XMLComponentValueList** обеспечены для конструктора типа структурных данных в ЯСО. Конструктор типа данных ЯСО требует, чтобы все поля задавались как входные, так что поля, которые пропускаются в списке **ComponentValueList,** должны быть представлены пустыми в ЯСО. Применение конструктора типа структурных данных будет так же воздействовать на ЯСО, как и на ASN.1.

Пример

```
MYTYPE ::= SEQUENCE{
     a INTEGER,
     b INTEGER OPTIONAL,
     c INTEGER DEFAULT 0,
     d INTEGER,
     e INTEGER OPTIONAL,
     f INTEGER DEFAULT 0
}
myValue MYTYPE ::= {a 1, b 1, c 1, d 1}
```

В этом примере поля **`a`**, **`b`**, **`c`** и **`d`** из **`myValue`** имеют назначенные значения, а поля **`e`** и **`f`** не имеют назначения.

synonym myValue MYTYPE = (. 1, 1, 1, 1, , .);

Следствием будет то, что поля a, b, c и d из myValue будут установлены на 1, поле e будет отсутствовать, а поле f будет принимать значение по умолчанию 0.

4.5.2.2 Последовательность значений

Грамматика ASN.1

SequenceOfValue (последовательность значений) и **XMLSequenceOfValue** (последовательность значений XML) определяются в 25.3/X.680.

Модель

SequenceOfValue и **XMLSequenceOfValue** оба представлены как:

MkString(E1) // MkString(E2)// ... // MkString(En),

где E1, E2, ..., En – это *значения* (**Values) SequenceOfValue** или **XMLValueOrEmpty** (значения XML или пустые) из **XMLSequenceOfValue** в порядке их появления. Если ни одного значения **Values** или **XMLValues** не указывается, **SequenceOfValue** или **XMLSequenceOfValue** представляется как имя Emptystring (пустая строка).

Квалификатор *типа* (**Type)** из составной первичности (Composite Primary), который содержит **SequenceOfValue,** предшествует каждому оператору MkString или литералу Emptystring, соответственно.

4.5.2.3 Object identifier value (Значение идентификатора объекта)

Грамматика ASN.1

ObjectIdentifierValue (значение идентификатора объекта) определяется в 31.3/X.680.

Модель

ObjectIdentifierValue игнорируется при преобразовании в ЯСО.

ObjectIdentifierValue является компонентом ASN.1, используемым для различия между модулями, которые имеют одинаковые имена, но различные объектные идентификаторы. Так как имена модулей и объектные идентификаторы не могут быть однозначно отображены в пакетном идентификаторе, который используется в операторах пакетного использования, компонента объектного идентификатора игнорируется при преобразовании в ЯСО. Идентификация соответствующего модуля, таким образом, открыта для ручных или инструментальных специальных решений.

4.5.2.4 Действительное значение

Грамматика ASN.1

RealValue (действительное значение) и **XMLRealValue** (действительное значение XML) определяются в 20.6/X.680.

Форма **0** используется для нулевых значений; альтернативная форма для **NumericRealValue** (численное действительное значение) не должна использоваться для нулевых значений.

Связанным типом для определения значения и целей выделения подтипов является:

```
SEQUENCE {
     mantissa INTEGER,          (мантисса)
     base INTEGER (2|10),       (основание)
     exponent INTEGER           (показатель степени, или экспонента)
       -- Связанным математическим действительным числом является "mantissa"
       -- умноженная на "base", возведенное в степень "exponent"
     }
```

Модель

Значения ASN.1 **NumericalRealValue** (численное действительное значение) и **XMLNumericRealValue** (численное действительное значение XML) отображаются в реальной сортировке значений ЯСО с действительным значением, вычисленным при преобразовании. Значения **SpecialRealValue** (специальное действительное значение) и **XMLSpecialRealValue** (специальное действительное значение XML) будут преобразовываться в самое большое положительное или отрицательное значение, соответственно.

ПРИМЕЧАНИЕ. – Преобразование **SpecialRealValue** не соответствует требованиям семантики для ASN.1, так как оно является директивой кодеру/декодеру на использование специального кода, указывающего на значения – ∞ (минус бесконечность). Так как кодирование не связано с данными в ЯСО, преобразуемыми из данных ASN.1, такое упрощение можно считать допустимым.

Пример

Определение ASN.1:

```
r50  REAL ::= { mantissa 5, base 10, exponent 1}
```

то же самое, что и:

synonym r50 Real = 50.0;

4.5.2.5 Значение целого числа

Грамматика ASN.1

IntegerValue (значение целого числа) и **XMLIntegerValue** (значение целого числа XML) определяются в 18.9/X.680.

Модель

IntegerValue и **XMLIntegerValue** отображаются в сортированном значении ЯСО <<**package** Predefined>>Integer с тем же самым действительным значением.

4.5.2.6 Булево значение

Грамматика ASN.1

BooleanValue (булево значение) и **XMLBooleanValue** (булево значение XML) определяются в 17.3/X.680.

Модель

BooleanValue и **XMLBooleanValue** отображаются в сортированном значении ЯСО <<**package** Predefined>>Boolean, где значения **TRUE** и **<true>** (истина) отражают булев литерал true (истина) и **FALSE** и **<false>** отражают булев литерал false (ложь).

4.5.3 Первичные строки

Грамматика ASN.1

CharacterStringValue (значение символьной строки) и **XMLCharacterStringValue** (значение символьной строки XML) определяются в 36.3/X.680.

BitStringValue (значение битовой строки) и **XMLBitStringValue** (значение битовой строки XML) определяются в 21.9/X.680.

Модель

Значение ASN.1 **StringValue** (строчное значение), содержащее **cstring** (имя ASN.1 для символьной строки, ограниченной знаками " с двух сторон, в начале и конце), представляет *идентификатор литерала символьной строки* (<character string literal identifier>), состоящий из *типа* (**Type)** и *литерала символьной строки* (<character string literal>) с тем же самым *текстом* (<text>), что и в строке *текста* **(Text)** ASN.1. *Типом* (**Type)** для **cstring** является **IA5Type,** как определяется этой Рекомендацией.

Значения **BitStringValue,** содержащие а **bstring** или **hstring,** отображаются в операторах ЯСО <<**package** Predefined>>Bitstring с тем же синтаксисом, при условии, что длина для типа управляющей битовой строки не ограничивается.

При условии, что длина для типа управляющей битовой строки ограничивается, применяются указанные ниже правила.

Значение **BitStringValue,** содержащее **bstring** или **hstring,** отображаются в операторах ЯСО <<**package** Predefined>>Bitstring с тем же синтаксисом. Однако, если длина **bstring** или **hstring** меньше, чем максимальная длина для типа управляющей битовой строки, значение **bstring** или **hstring** расширяется до максимальной длины со всеми добавленными битами, установленными в 0.

Значение **BitStringValue,** определяемое с использованием **IdentifierList** (список идентификаторов), оценивается как имеющее битовое значение 1 во всех битовых позициях, определяемых перечисленным *идентификатором* **(identifier)** в список **IdentifierList**. Значения оставшихся битов, вплоть до максимальной длины для типа управляющей битовой строки, устанавливаются в 0. Результирующая строка отображается в операторах ЯСО <<**package** Predefined>>Bitstring, которым предшествуют символы ' и за которыми следует пара символов 'B.

Значение **XMLBitStringValue,** содержащее **xmlbstring,** отображается в операторах ЯСО <<**package** Predefined>>Bitstring, которым предшествуют символы ' и за которыми следует пара символов 'B. Однако, если длина **xmlbstring** меньше, чем максимальная длина для типа управляющей битовой строки, значение **xmlbstring** расширяется до максимальной длины со всеми добавленными битами, установленными в 0.

Пример

Использование типа битовой строки для моделирования значений **bit map** (битовой карты), упорядоченного множества с фиксированным размером для логических переменных, указывающих, будет ли конкретное условие сохраняться для каждого соответствующего упорядоченного множества объектов.

```
DaysOfTheWeek ::= BIT STRING {
       sunday(0), monday (1), tuesday(2),     (воскресенье, понедельник, вторник)
       wednesday(3), thursday(4), friday(5), (среда, четверг, пятница)
       saturday(6) } (SIZE (7))               (суббота, размер)

     sunnyDaysLastWeek1 DaysOfTheWeek ::= {sunday, monday, wednesday}
     sunnyDaysLastWeek2 DaysOfTheWeek ::= '1101000'B
```

Отображение в ЯСО будет давать следующее:

synonym sunday Integer = 0; (воскресенье)
synonym monday Integer = 1; (понедельник)
synonym tuesday Integer = 2; (вторник)
synonym wednesday Integer = 3; (среда)
synonym thursday Integer = 4; (четверг)
synonym friday Integer = 5; (пятница)
synonym saturday Integer = 6; (суббота)
synonym sunnyDaysLastWeek1 DaysOfTheWeek = '1101000'B;
synonym sunnyDaysLastWeek2 DaysOfTheWeek = '1101000'B;

Использование типа битовой строки для моделирования значений **bit map** (битовой карты), упорядоченного множества переменного размера для логических переменных, указывающих, будет ли конкретное условие сохраняться для каждого соответствующего упорядоченного множества объектов.

```
DaysOfTheWeekVar ::= BIT STRING {
       sunday(0), monday (1), tuesday(2),     (воскресенье, понедельник, вторник)
       wednesday(3), thursday(4), friday(5),  (среда, четверг, пятница)
       saturday(6) } (SIZE (0..7))            (суббота, размер)

     sunnyDaysLastWeek3 DaysOfTheWeekVar ::= {sunday, monday, wednesday}
     sunnyDaysLastWeek4 DaysOfTheWeekVar ::= '1101'B
```

Отображение в ЯСО будет давать следующее (нет повторения синонимов для битовых имен):

synonym sunnyDaysLastWeek3 DaysOfTheWeek = '1101000'B;
synonym sunnyDaysLastWeek4 DaysOfTheWeek = '1101000'B;

4.5.4 Спецификация установки элемента

Грамматика ASN.1

ElementSetSpecs (спецификация установки элемента) определяется в 46.1/X.680.

Модель

Два или более наборов значений могут комбинироваться с использованием этой нотации. Результирующий набор оценивается в преобразовании, и результат отображается в ЯСО.

Маркеры расширения ("...") в **ElementSetSpecs** игнорируются в преобразовании для ЯСО.

5 Отображение типов ASN.1, определяемое в модулях ASN.1 с использованием информационных объектов, классов информационных объектов и наборов информационных объектов

5.1 Введение

Рекомендация МСЭ-Т X.681 обеспечивает нотацию ASN.1, которая позволяет классам информационных объектов, а также и отдельным информационным объектам и их наборам быть определенными и обеспеченными собственными именами для ссылок в системе. Класс информационного объекта представляет собой шаблон для сбора информации, которую составляют атрибуты любого представителя этого класса. Информационные объекты обеспечивают механизм групповых таблиц внутри языка ASN.1. Такая групповая таблица определяет ассоциации специфических наборов значений полей или типов. Это особенность заменяет ранее использовавшуюся конструкцию MACRO (доступную в ASN.1:1990) и первоначально применявшуюся для заполнения пропусков в определении типа в зависимости от одного или более ключевых полей.

Эта статья предполагает, что все конструкции ASN.1, определенные в Рекомендациях МСЭ-Т X.681, X.682 и X.683, могут быть использованы в модулях ASN.1. Затем она идентифицирует, какая информация, содержащаяся в классах информационных

объектов ASN.1, в информационных объектах и наборах информационных объектов может быть полезной при отображении соответствующих целевых объектов ЯСО. Отображения, которые возможны и полезны, определяются. Следует заметить, что часть информации не будет представлена в ЯСО из-за различий в природе двух языков.

5.2 Определение и назначение классов информационных объектов

Грамматика ASN.1

ObjectClassAssignment (назначение класса объекта) определяется в 9.1/X.681.

Модель

Определения **ObjectClass** (класс объекта) в ASN.1 не имеют прямого соответствия в ЯСО.

5.3 Тип поля класса информационных объектов

Грамматика ASN.1

ObjectClassFieldType (тип поля класса объекта), **FixedTypeValueFieldSpec** (спецификация поля фиксированного значения типа) и **FixedTypeValueSetFieldSpec** (спецификация поля набора фиксированных значений типа) определяются в 14.1/X.681.

Модель

Типы ASN.1 могут быть определены с использованием нотации **ObjectClassFieldType** для извлечения информации из полей спецификаций классов информационных объектов при отсутствии табличных ограничений. Такие типы ASN.1 могут быть отображены в ЯСО, при условии, что в их определении используются только поля **FixedTypeValueFieldSpec** или **FixedTypeValueSetFieldSpec**. Отображение для типа значения ЯСО выполняется так, как уже было определено в пункте 4.3 для значения **FixedTypeValueFieldSpec** или **FixedTypeValueSetFieldSpec,** установленного из справочных спецификаций классов информационных объектов.

Нотация **ObjectClassFieldType** используется также по отношению к табличным ограничениям, как определено в 5.6.2.

Пример

Если ASN.1 содержит следующую спецификацию:

```
EXAMPLE-CLASS ::= CLASS {
     &TypeField                                  OPTIONAL,    -- класс поля 1
     &fixedTypeValueField         INTEGER        OPTIONAL,    -- класс поля 2
     &variableTypeValueField      &TypeField     OPTIONAL,    -- класс поля 3
     &FixedTypeValueSetField      INTEGER        OPTIONAL,    -- класс поля 4
     &VariableTypeValueSetField   &TypeField     OPTIONAL     -- класс поля 5
}
WITH SYNTAX {
     [TYPE-FIELD                        &TypeField]
     [FIXED-TYPE-VALUE-FIELD            &fixedTypeValueField]
     [VARIABLE-TYPE-VALUE-FIELD         &variableTypeValueField]
     [FIXED-TYPE-VALUE-SET-FIELD        &FixedTypeValueSetField]
     [VARIABLE-TYPE-VALUE-SET-FIELD     &VariableTypeValueSetField]
}

ExampleType ::= SEQUENCE {
     integerComponent1  EXAMPLE-CLASS.&fixedTypeValueField,       -- поле 1
     integerComponent2  EXAMPLE-CLASS.&FixedTypeValueSetField     -- поле 2
}

exampleValue ExampleType ::= {
     integerComponent1       123,                -- поле 1
     integerComponent2       456                 -- поле 2
}
```

Предметы, которые могут быть отображены в ЯСО, это **`ExampleType`** и **`exampleValue`**:

```
value type ExampleType {                                    (тип значения пример тип)
        struct
        integerComponent1   <<package Predefined>> Integer,   /* поле 1 */
        integerComponent2   <<package Predefined>> Integer    /* поле 2 */
}
synonym exampleValue ExampleType = (. 123, 456 .);
```

5.4 Определение и назначение информационных объектов

Грамматика ASN.1

ObjectAssignment (назначение объекта) определяется в 11.1/X.681.

Модель

Определения информационных объектов в модуле ASN.1 не имеют эквивалентного отображения в ЯСО.

5.5 Информация от информационных объектов

Грамматика ASN.1

InformationFromObjects (информация от объектов) определяется в 15.1/X.681.

Модель

К информации из колонки связанной таблицы для информационного объекта или набора информационных объектов могут обращаться различные случаи из нотации **InformationFromObjects**.

В модуле ASN.1, тип ASN.1 может быть указан с полями, определенными через использование нотации **InformationFromObjects**. Такой тип ASN.1 может быть отображен в ЯСО, при условии, что все появления нотации **InformationFromObjects** могут быть расширены до значения или типа. Тип ASN.1 как таковой отображается так, как указывается в 4.3, в то время как семантика расширения нотации **InformationFromObjects** следует семантике ASN.1.

5.6 Спецификация ограничений

Грамматика ASN.1

GeneralConstraint (общее ограничение), **TableConstraint** (табличное ограничение) и **UserDefinedConstraint** (ограничения, определяемое пользователем) определяются в 8.1/X.682.

Модель

Типы, указанные с использованием **TableConstraint,** отображаются в ЯСО в соответствии с правилами, представленными в 5.6.2. Типы, указанные с использованием **UserDefinedConstraint,** не могут быть отображены в ЯСО.

5.6.1 Ограничения, определяемые пользователем

Грамматика ASN.1

UserDefinedConstraint (ограничения, определяемые пользователем) определяется в 9.1/X.682.

Модель

Эта форма спецификации ограничений может рассматриваться как специальная форма комментирования ASN.1, так как она не может быть полностью обработана на компьютере. Поэтому, спецификации типа ASN.1, использующие нотацию **UserDefinedConstraint,** не могут быть отображены в ЯСО.

5.6.2 Табличные ограничения

Грамматика ASN.1

TableConstraint (табличное ограничение), **SimpleTableConstraint** (простое табличное ограничение) **и ComponentRelationConstraint** (ограничение на взаимоотношения компонентов) определяются в 10.3/X.682.

Модель

Ограничивающая нотация может появиться (в круглых скобках) после любого использования синтаксической конструкции "Type" (Тип). Создатели прикладного программного обеспечения могут использовать эту нотацию для определения структурированных типов данных с последующими ограничениями на значения их полей. Примерами таких ограничений являются ограниченный диапазон отдельных компонентов или указание на взаимоотношения между компонентами. Ранней формой этой нотации является **SimpleTableConstraint**, а последней – **ComponentRelationConstraint**.

Для типов с **SimpleTableConstraint**, применяются следующие правила преобразования.

Перед тем, как ограниченный тип может быть отображен в ЯСО, некоторые типы значений ЯСО должны быть сконструированы из спецификации классов информационных объектов и спецификации набора ограничивающих информационных объектов следующим образом:

a) Для каждого набора информационных объектов создается некоторое количество типов значений ЯСО. Типы генерируются так, чтобы для каждого поля CLASS (класс) информационного объекта, связанного с набором информационных объектов, генерировался один тип значения ЯСО. Имя типа составляется как конкатенация имени набора информационных объектов, символа подчеркивания ('_') и имени согласованного поля класса.

b) Если полем класса является **FixedTypeValueFieldSpec** (спецификация поля фиксированн. значения типа), конструируется синтаксический тип (syntype) ЯСО. Тип syntype имеет диапазонное ограничение, то есть объединение значений, указанных согласованным полем каждого информационного объекта в наборе информационных объектов.

c) Если полем класса информационных объектов будет **VariableTypeValueFieldSpec** (спецификация поля переменного значения типа), конструируется выборный тип ЯСО. Выборный тип конструируется так, чтобы все типы, обнаруженные в согласованном поле всех информационных объектов, принадлежащих набору ограничивающих информационных объектов, включались бы в выбор. Имена полей выбора выводятся как эквиваленты нижнего регистра (строчные буквы) согласованных типов.

Ограниченный ASN.1-тип может быть теперь отображен в ЯСО. Тип как таковой отображается так, как определено в 4.3. Имена полей ЯСО такие же, как и имена полей ASN.1. Спецификация ASN.1 для различных опций сохраняется при преобразовании. Для каждого поля ASN.1, ограниченного набором информационных объектов, тип ЯСО указывается как тип, созданный из спецификации классов информационных объектов и из спецификации набора ограничивающих информационных объектов (пункты от a) до c)).

Для спецификаций ASN.1-типа, использующих нотацию **ComponentRelationConstraint**, применяются те же самые правила преобразования типа. Помимо этого, для каждого типа ASN.1 с нотацией **ComponentRelationConstraint** генерируется также проверочный метод, который отслеживает информационный объект и проверяет ограничения. Проверочный метод возвращает значение "true" (истина), если все относительные ограничения такие, как и ожидается, и значение "false" (ложь), если любое из относительных ограничений нарушено.

Следующие шаги необходимы для конструирования проверочного метода:

Для каждого элемента, вовлеченного в относительное ограничение (имеющего присоединенную к нему нотацию **ComponentRelationConstraint** или упоминаемого в любой нотации **ComponentRelationConstraint**), генерируется локальная декларация переменной тестирования. Генерирование выполняется по следующей схеме:

'dcl <test var name> <field type>; <test var name> := < field ref>;'
(имя переменн. тестиров.)(тип поля)(имя переменн. тестиров.)(поле ссылки),

где <test var name> - уникальное имя для каждой переменной тестирования, < field type> - тип элемента, < field ref> - ссылка на этот элемент. Если элемент присутствует, переменная используется для значения соответствующего поля объекта.

Для каждого относительного ограничения, один тест генерируется для каждой комбинации ограничивающих значений или типов в определении набора объектов. Каждый тест генерируется с использованием следующей схемы:

'если (<test expr> и не (<value test>), тогда { return False; }
если ((тест. эксперимент) и не (тест. значение)) тогда (возврат ложь),

где <test expr> – результат комбинации одного теста для каждого ограничивающего значения или типа с использованием логического оператора "И". Для ограничивающих значений тест определяется как:

'<test var name> = <test value>'
(имя переменн. тестиров.) = (значение теста),

где <test var name> – имя переменной тестирования, как описано выше, и <test value> – соответствующее значение из определения набора объектов.

Для ограничивающих типов тест определяется как:

'<test var name>.<ispresent method>'
(имя переменн. тестиров.) (метод опр. присутствия),

где <test var name> – имя переменной тестирования, как описано выше, и <ispresent method> – метод, который проверяет, что соответствующий тип присутствует.

<value test> – результат комбинации одного теста для каждого значения или типа ограниченного элемента в определении набора объектов, который соответствует значениям или типам в показанном выше параметре <test expr>, с использованием логического оператора "ИЛИ". Для значений каждый тест выдается как:

'<test var name> = <value>'
(имя переменн. тестиров.) (значение),

где <test var name> – имя переменной, соответствующее ограниченному полю, <value> – значение из определения набора объектов.

Для типов, тест определяется, как:

'<test var name>.<ispresent method>'
(имя переменн. тестиров.) (метод опр. присутствия),

где <test var name> – имя переменной, соответствующее ограниченному полю, и <ispresent method> – метод, который проверяет, что соответствующий тип присутствует.

Для каждого поля строки в типе генерируется цикл, соответствующий следующей схеме:

'loop(dcl <loop var> Integer := 1; <loop var> <= length(<string field>);
<loop var> := <loop var> + 1) { <loop body> }',

где <loop var> – уникальное имя переменной, <string field> - ссылка на обрабатываемое поле строки, и <loop body> – результат применения шагов преобразований из этой статьи к элементам в строке.

Пример 1

Пример типа с нотацией **SimpleTableConstraint**:

```
ErrorReturn ::= SEQUENCE
{
  errorCategory ERROR-CLASS.&category({ErrorSet}) OPTIONAL,
  errors SEQUENCE OF SEQUENCE
  {
    errorCode ERROR-CLASS.&code
                          ({ErrorSet}),
    errorInfo ERROR-CLASS.&Type
                          ({ErrorSet})
  } OPTIONAL
}
```

При условии, что спецификации класса и набора объектов были:

```
ERROR-CLASS ::= CLASS
{
  &category PrintableString (SIZE(1)),
  &code     INTEGER,
  &Type
}
WITH SYNTAX {&category &code &Type }

ErrorSet ERROR-CLASS ::=
{
  { "A" 1 INTEGER } |
  { "A" 2 REAL } |
  { "B" 1 CHARACTER STRING } |
  { "B" 2 GeneralString }
}
```

Типами ЯСО, извлекаемыми из спецификации ограничений, будут:

```
syntype ErrorSet_category = PrintableString (SIZE(1))
  constants 'A', 'B'
endsyntype;

syntype ErrorSet_code = <<package Predefined>> Integer
  constants 1, 2
endsyntype;

value type ErrorSet_Type { choice
  integer              <<package Predefined>> Integer;
  real                 <<package Predefined>> Real;
  characterString      <<package Predefined>> CharacterString;
  generalString        <<package Predefined>> GeneralString;
}
```

Сконструированный тип ЯСО будет следующим:

```
value type ErrorReturn { struct
  errorCategory ErrorSet_category optional;
  errors String <
    { struct
      errorCode ErrorSet_code,
      errorInfo ErrorSet_Type } > optional;
}
```

Никаких проверочных методов сгенерировано не будет.

Пример 2

Пример типа с нотацией **`ComponentRelationConstraint`**.

```
ErrorReturn ::= SEQUENCE
{
  errorCategory ERROR-CLASS.&category({ErrorSet}) OPTIONAL,
  errors SEQUENCE OF SEQUENCE
  {
    errorCode ERROR-CLASS.&code
                          ({ErrorSet}{@errorCategory}),
    errorInfo ERROR-CLASS.&Type
                          ({ErrorSet}{@errorCategory,@.errorCode})
  } OPTIONAL
}
```

Соответствующий тип ЯСО будет следующим:

```
value type ErrorReturn {
struct
  errorCategory ErrorSet_category optional;
  errors String <
    { struct
     errorCode ErrorSet_code,
     errorInfo ErrorSet_Type } > optional;
  method Check() -> Boolean
  {
        dcl t1 ErrorSet_category;
        dcl p1 Boolean;
        p1 := this.errorCategoryPresent();
        если (p1 = True)
{
                t1 := this.errorCategory;
        }
        если ((p1 = False) и (this.errorsPresent() =  True))
        {
                return False;
        }
        loop (dcl i1 Integer := 1; I <=length(errors); i1 := i1+1)
        {
                dcl t2 ErrorSet_code, t3 ErrorSet_Type;
                t2 := this.errors[i1].errorCode;
                t3 := this.errors[i1].errorInfo ;
                если (t1="A"и не ( t2=1 или t2=2))
                {
                return False;
                }
                если (t1="B"  и не ( t2=1 или t2=2))
                {
                return False;
                }
                если (t1="A"  t2=1 и не (t3.integerPresent()))
                {
                return False;
                }
```

```
                если (t1="A" и t2=2 и не (t3.realPresent()))
                {
                return False;
                }
                если (t1="B" и t2=1 и не (t3.characterStringPresent()))
{
                return False;
                }
                если (t1="B" и t2=2 и не (t3.generalStringPresent))
{
                return False;
                }
        }
   }
}
```

6 Отображение параметризуемых спецификаций ASN.1

Рекомендация МСЭ-Т X.683 [6] определяет способ параметризации спецификации ASN.1. Все концепции ASN.1 могут быть параметризованы. Эта особенность разрешает частичную спецификацию типов или значений внутри модуля ASN.1 с завершением полной спецификации путем добавления фактических параметров во время установки.

Рекомендация МСЭ-Т Z.100 определяет эквивалентную концепцию формальных контекстных параметров.

6.1 Параметризуемое назначение

Грамматика ASN.1

Следующие операторы параметризуемого назначения, соответствующие каждому из операторов назначения, указываются в Рекомендациях МСЭ-Т X.680 и X.681.

ParameterizedAssignment (параметризуемое назначение),
ParameterizedTypeAssignment (параметризуемое назначение типа),
ParameterizedValueSetTypeAssignment (параметризуемое назначение типа набора значений),
ParameterizedObjectClassAssignment (параметризуемое назначение класса объекта),
ParameterizedObjectAssignment (параметризуемое назначение объекта),
ParameterizedObjectSetAssignment (параметризуемое назначение набора объектов)
Эти конструкции определяются в 8.1/X.683.

Модель

Использование всех форм **ParameterizedAssignment** поддержано внутри модулей ASN.1.

ParameterizedTypeAssignment может быть отображено в ЯСО, как определяется в 6.2, полагаясь на механизм формальных контекстных параметров ЯСО.

ParameterizedValueSetTypeAssignment, **ParameterizedObjectClassAssignment**, **ParameterizedObjectAssignment**, **ParameterizedObjectSetAssignment** могут использоваться в модулях ASN.1 для того, чтобы они могли быть использованы другими спецификациями ASN.1, но при этом сами не будучи отображенными для ЯСО.

6.2 Параметризуемое назначение типа

Грамматика ASN.1

ParameterizedTypeAssignment (параметризуемое назначение типа) и **ParameterList** (список параметров) определяется в 8.1/X.683.

Модель

Различие между обычными и параметризуемыми типами ASN.1 состоит в том, что список **ParameterList** следует значению **typereference** (ссылка на тип), а формальные параметры, содержащиеся в **ParameterList,** используются в определении типа (**Type)**.

Type (тип), определяемый в ASN.1 с использованием параметров из **ParameterList,** отображается в соответствующий тип ЯСО (как определено в 4.2.1), при условии, что параметры ASN.1 являются либо значением, либо параметрами. Такие параметры отображаются в *формальные контекстные параметры* (<formal context parameters>) типа ЯСО. Параметр типа ASN.1 отображается в *сортированном контекстном параметре* (<sort context parameter>) ЯСО, а значение параметра ASN.1 отображается в *синонимическом контекстном параметре* (<synonym context parameter>) ЯСО.

Параметризуемые типы ASN.1, имеющие параметры, которые не являются типами или значениями, не могут быть отображены в ЯСО непосредственно. Однако, если параметры могут быть вначале расширены в типы или значениями, результирующий тип или значение ASN.1 могут быть отображены в ЯСО, как определено в 6.3.

Пример

Определение ASN.1-типа:

```
TemplateMessage {INTEGER : minSize, INTEGER : maxSize, IndicatorType } ::=
SEQUENCE
{
    asp         INTEGER,
    pdu         OCTET STRING(SIZE(minSize..maxSize)),
    indicator   IndicatorType
}
```

отображается в тип ЯСО:

value type TemplateMessage
<**synonym** minSize <<**package** Predefined>> Integer; **synonym** maxSize <<**package** Predefined>> Integer; **value type** IndicatorType>
struct
{
    asp Integer;
    pdu <<**package** redefined>>Octetstring (**SIZE**(minSize:maxSize));
    indicator IndicatorType;
}

6.3 Обращение к параметризуемым определениям типа ASN.1

Грамматика ASN.1

ParameterizedType (параметризуемый тип), **ParameterizedValue** (параметризуемое значение) и **ActualParameterList** (список фактических параметров) определяются в 9.2/X.683.

Модель

Ссылки **ParameterizedType** и **ParameterizedValue** используются в ASN.1 для определения новых типов и значений ASN.1 через обеспечение списка **ActualParameterList**.

Если определение **ParameterizedType** было таково, что его было возможно отобразить в ЯСО, и список **ActualParameterList** содержит только параметры **Type** (тип) и **Value** (значение), тогда ссылки ASN.1 на такие определения могут быть отображены в конкретизациях типа ЯСО с контекстными параметрами так, что элементы списка **ActualParameterList** отображаются в *фактические контекстные параметры* (<actual context parameters>). Пример 1 показывает одно такое отображение.

Если в соответствии с 6.2 определение **ParameterizedType** не может быть отображено в определение ЯСО-типа с контекстными параметрами, ссылки на такие определения **ParameterizedType** могут быть отображены в ЯСО так, что значение такого типа полностью расширяется до уровня типов, определяемых в статье 4, до того, как отображение в ЯСО будет выполнено.

Если список **ActualParameterList** содержит **ValueSet** (набор значений), **DefinedObjectClass** (класс определенного объекта), **Object** (объект) или **ObjectSet** (набор объектов), отображение такого типа в ЯСО выполняется таким образом, что значение такого типа полностью расширяется

до уровня типов, определяемых в статье 4, до того, как отображение в ЯСО будет выполнено. Пример 2 показывает одно такое отображение.
Типы ASN.1 и значения, извлеченные из ссылочных параметров параметризованных определений ASN.1, могут быть отображены в ЯСО, как определено в статье 4.

Пример 1

Параметризованный тип, использованный в примере в пункте 6.2, может быть использован для определения простого ASN.1, как следует далее:

```
ActualMessage ::= TemplateMessage{10, 20, BOOLEAN}
```

Этот может быть отображено в тип ЯСО:

value type ActualMessage : TemplateMessage < 10, 20, <<**package** Predefined>> Boolean >

Пример 2

Далее следует пример определения ASN.1-типа, извлеченного с использованием параметра, которым является информационный объект. Модули ASN.1 должны содержать соответствующее определение класса информационного объекта с параметризуемым назначением, имеющим объект этого класса в качестве фиктивного параметра, назначение значения информационного объекта и ссылку на определение параметризуемого типа.

```
MESSAGE-PARAMETERS ::= CLASS {
     &maximum-priority-level          INTEGER,
     &maximum-message-buffer-size     INTEGER
}
WITH SYNTAX {
     THE MAXIMUM PRIORITY LEVEL IS         &maximum-priority-level
     THE MAXIMUM MESSAGE BUFFER SIZE IS    &maximum-message-buffer-size
}
Message-PDU { MESSAGE-PARAMETERS : param } ::= SEQUENCE {
     priority-level     INTEGER   (0..param.&maximum-priority-level),
     message   BMPString (SIZE    (0..param.&maximum-message-buffer-size))
}
my-message-parameters MESSAGE-PARAMETERS ::= {
     THE MAXIMUM PRIORITY LEVEL IS 10
     THE MAXIMUM MESSAGE BUFFER SIZE IS 2000
}
MY-Message-PDU ::= Message-PDU { my-message-parameters }
```

Семантика ASN.1 такова, что с таким определением класса, определением параметризованного типа и определением значения объекта, тип, получающийся в результате преобразования значения MY-Message-PDU, является эквивалентным следующему:

```
MY-Message-PDU ::=      SEQUENCE {
     priority-level     INTEGER   (0..10),
     message            BMPString (SIZE    (0..2000))
}
```

Результирующий тип ASN.1 может быть отображен в тип ЯСО как:

value type MY_ Message _PDU {
struct
priority_level <<**package** Predefined>> INTEGER (0..10);
message <<**package** Predefined>> BMPString (**SIZE** (0..2000));
}

6.4 Обращение к параметризуемым определениям значения ASN.1

Грамматика ASN.1

ParameterizedType (параметризуемый тип), **ParameterizedValue** (параметризуемое значение) и **ActualParameterList** (список фактических параметров) определяются в 9.2/X.683.

Модель

Ссылки **ParameterizedValue** используются в ASN.1 для определения новых значений ASN.1 через обеспечение списка **ActualParameterList**.

Ссылки **ParameterizedValue** отображаются в ЯСО таким образом, что содержание такой спецификации значений полностью расширяется до уровня назначения значений, определенного в статье 4, до того, как отображение в ЯСО выполняется.

6.5 Обращение к другим параметризуемым определениям ASN.1

ParameterizedValueSetType (тип набора параметризуемых значений), **ParameterizedObjectClass** (класс параметризуемого объекта), **ParameterizedObjectSet** (набор параметризуемых объектов) и **ParameterizedObject** (параметризуемый объект) определяются в 9.2/X.683.

Модули ASN.1 могут содержать спецификацию наборов значений, классов объектов, наборов объектов и объектов, определяемых через отсылку к **SimpleDefinedType** (простой определяемый тип) со списком **ActualParameterList** (список фактических параметров). Такие спецификации не отображаются в ЯСО.

7 Добавления в пакет Predefined

Следующие определения должны быть добавлены в пакет Predefined (предопределенные) для того, чтобы поддержать объединение модулей ASN.1 с ЯСО.

```
syntype NumericChar = Character constants
' ',      '0',  '1',  '2',  '3','4',  '5',  '6',
'7',      '8',  '9' endsyntype;
/* */
/*      Сортировка NumericString – численной строки   */
/*              Определение                           */
value type NumericString
        inherits String < NumericChar > ( '' = emptystring )
        adding
                operators ocs in nameclass
                        "" ( ('0':'9') or """" or (' ') )+ "" -> this NumericString;
/*      символьные строки любой длины любых символов от пробела  ' ' до '9'        */
axioms
        for all c in NumericChar nameclass (
        for all cs, cs1, cs2 in ocs nameclass (
        spelling(cs) == spelling(c)                                     ==> cs == mkstring(c);
/*      строка 'A' формируется из символа  'A' и т. д.      */
        spelling(cs) == spelling(cs1) // spelling(cs2),
        length(spelling(cs2)) == 1                                      ==> cs == cs1 // cs2;
        ));
endvalue type NumericString;
/* */
syntype PrintableChar = Character constants
' ',  '0',  '1',  '2',  '3',  '4',  '5',  '6',
```

```
'7',  '8',   '9',   'A',   'B',   'C',   'D',   'E',
'F',  'G',   'H',   'I',   'J',   'K',   'L',   'M',
'N',  'O',   'P',   'Q',   'R',   'S',   'T',   'U',
'V',  'W',   'X',   'Y',   'Z',   'a',   'b',   'c',
'd',  'e',   'f',   'g',   'h',   'i',   'j',   'k',
'l',  'm',   'n',   'o',   'p',   'q',   'r',   's',
't',  'u',   'v',   'w',   'x',   'y',   'z',   "'",
'(',  ')',   '+',   ',',   '-',   '.',   '/',   ':',
'=',  '?'
constants;
/* */

/*   Сортировка PrintableString - печатной строки  */
/*   Определение                                   */
value type PrintableString
     inherits String < PrintableChar > ( '' = emptystring )
     adding
         operators ocs in nameclass
                          "" ( (' ':'&') or """" or ('(': '?') )+ "" -> this PrintableString;
/*   символьные строки любой длины любых символов от пробела ' ' до  '?'     */
axioms
     for all c in PrintableChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                                  ==> cs  == mkstring(c);
/*   строка 'A' формируется из символа 'A' и т. д.   */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                                   ==> cs  == cs1 // cs2;
     ));
endvalue type PrintableString;
/* */
syntype TeletexChar = Character constants
/*   символы указываются в таблице 3/X.680   */ endsyntype;
/* */
/*   Сортировка TeletexString - строки телетекста  */
/*   Определение                   */
value type TeletexString
     inherits String < TeletexChar > ( '' = emptystring )
     adding
                          operators ocs in nameclass
/*   символы указываются в таблице 3/X.680   */ -> this TeletexString;
axioms
     for all c in TeletexChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                                  ==> cs  == mkstring(c);
/*   строка ‘A’ формируется из символа 'A' и т. д.  */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                                   ==> cs  == cs1 // cs2;
     ));
endvalue type TeletexString;
syntype VideotexChar = Character constants
/*   символы указываются в таблице 3/X.680   */ endsyntype;
/* */
```

```
/*   Сортировка VideotexString - строки видеотекста          */
/*   Определение                                             */
value type VideotexString
     inherits String < VideotexChar > ( '' = emptystring )
     adding
          operators ocs in nameclass
/*   символы указываются в таблице 3/X.680   */ -> this VideotexString;
axioms
     for all c in VideotexChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                             ==> cs  == mkstring(c);
/*   строка 'A' формируется из символа 'A' и т. д.  */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                              ==> cs  == cs1 // cs2;
     ));
endvalue type VideotexString;

syntype IA5Char = Character endsyntype;

syntype IA5String = Charstring endsyntype;

value type GeneralChar
     literals /* Все G и все C наборы+SPACE(пробел)+DELETE(стирание) в таблице 3/X.680*/
operators
     gchr                    ( Integer ) -> this GeneralChar;
endvalue type;

value type UniversalChar
     literals /*   см. 37.6/X.680   */
operators
     uchr                    ( Integer ) -> this UniversalChar;
endvalue type;
/* */

/*   Сортировка UniversalCharString -   строки универсальных символов  */
/*   Определение                                                        */

value type UniversalCharString
     inherits String < UniversalChar > ( '' = emptystring )
     adding
          operators ocs in nameclass
/*   смотрите 37.6/X.680     */ -> this UniversalCharString;
axioms
     for all c in UniversalChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                             ==> cs  == mkstring(c);
/*   строка 'A' формируется из символа 'A' и т. д.  */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                              ==> cs  == cs1 // cs2;
     ));
```

```
endvalue type UniversalCharString;
/* */

/*   Сортировка UTF8String - строки UTF8      */
syntype UTF8String = UniversalCharString endsyntype;
/* */

/*   Сортировка GeneralCharString - строки общих символов   */
/*   Определение                                             */
value type GeneralCharString
     inherits String < GeneralChar > ( '' = emptystring )
     adding
        operators ocs in nameclass
/*  Все G и все C наборы + SPACE (пробел) + DELETE (стирание) в таблице 3/X.680        */
-> this GeneralCharString;
/*   символьные строки любой длины любых символов от пробела ' ' до  '?'     */
axioms
     for all c in GeneralChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                              ==> cs  == mkstring(c);
/*   строка 'A' формируется из символа 'A' и т. д.  */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                               ==> cs  == cs1 // cs2;
     ));
endvalue type GeneralCharString;
/* */

syntype GraphicChar = GeneralChar constants
/*   Все G+SPACE (пробел)+DELETE (стирание) как указывается в таблице 3/X.680        */
endsyntype;
/* */

/*   Сортировка GraphicCharString - строки графических символов      */
/*   Определение                                                      */

value type GraphicCharString

     inherits String < GraphicChar > ( '' = emptystring )
     adding
        operators ocs in nameclass
/*   Все G+SPACE (пробел)+DELETE (стирание) как указывается в таблице 3/X.680        */
-> this GraphicCharString;
axioms
     for all c in GraphicChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                              ==> cs  == mkstring(c);
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                               ==> cs  == cs1 // cs2;
     ));
endvalue type GraphicCharString;
```

```
syntype VisibleChar = Character constants
/* символы указываются в таблице 3/X.680 */
endsyntype;
/* */

/*   Сортировка VisibleString – видимой строки     */
/*   Определение                                   */
value type VisibleString
     inherits String < VisibleChar > ( '' = emptystring )
     adding
        operators ocs in nameclass
/*   символы указываются в таблице 3/X.680   */
-> this VisibleString;
axioms
     for all c in VisibleChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                          ==> cs  == mkstring(c);
/*   строка 'A' формируется из символа 'A' и т. д.  */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                           ==> cs  == cs1 // cs2;
     ));
endvalue type VisibleString;

syntype BMPChar = UniversalChar CONSTANTS /* см. 37.15/X.680 */ endsyntype;
/* */
/*   Сортировка BMPCharString – строка символов BMP   */
/*    Определение                                     */
value type BMPCharString
     inherits String < BMPChar > ( '' = emptystring )
     adding
        operators ocs in nameclass
/*   см. 37.15/X.680      */ -> this BMPCharString;
axioms
     for all c in BMPChar nameclass (
     for all cs, cs1, cs2 in ocs nameclass (
     spelling(cs) == spelling(c)                          ==> cs  == mkstring(c);
/* строка 'A' формируется из символа 'A' и т. д. */
     spelling(cs) == spelling(cs1) // spelling(cs2),
     length(spelling(cs2)) == 1                           ==> cs  == cs1 // cs2;
     ));
endvalue type BMPCharString;
/* */

value type NULL
literals NULL

endvalue type;
```

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ

Серия A	Организация работы МСЭ
Серия B	Значения выражений: определения, символы, классификация
Серия C	Общие телекоммуникационные статистические данные
Серия D	Общие тарифные принципы
Серия E	Эксплуатация глобальных сетей, телефонная связь, операции по обслуживанию и факторы, связанные с человеком
Серия F	Нетелефонные службы связи
Серия G	Передающие системы и носители, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией функций
Серия J	Кабельные сети и передача телевизионных, звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Конструкция, установка и защита кабелей и других элементов вне станции
Серия M	TMN и сетевое обслуживание: международные передающие системы, телефонные цепи, телеграфия, факсимильная связь и арендуемые каналы
Серия N	Обслуживание: схемы международных звуковых программ и телевизионных передач
Серия O	Технические требования к измерительному оборудованию
Серия P	Качество телефонной передачи, телефонное оборудование, сети местных линий
Серия Q	Коммутация и сигнализация
Серия R	Телеграфная передача
Серия S	Терминальное оборудование телеграфной службы
Серия T	Терминалы для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	Коммуникации между сетями передачи данных и открытой системой
Серия Y	Глобальная информационная инфраструктура и аспекты протокола Internet
Серия Z	**Языки и общие соображения по программному обеспечению для телекоммуникационных систем**

Отпечатано в Швейцарии
Женева, 2004 г.