

INTERNATIONAL STANDARD

ISO/IEC
8613-10

Second edition
1995-10-15

Information technology — Open Document Architecture (ODA) and Interchange Format —

Part 10: Formal specifications

*Technologies de l'information — Architecture de document ouverte (ODA)
et format de transfert —*

Partie 10: Spécifications formelles



Reference number
ISO/IEC 8613-10:1995
AVA Documents
Document Sharing Hub

Contents

	Page
Foreword	iii
1 Scope	1
2 Normative references	2
3 Definitions	2
4 Syntax and semantics of the specification language	3
4.1 Basic concepts	3
4.2 Syntax of the specification language	4
4.3 Predicate symbols with built-in semantics	7
4.4 Operator symbols with built-in semantics	7
4.5 Other terms	8
4.6 Notational simplifications	9
5 Structure of the formal specifications	11
6 Commonly used definitions	13
7 Formal specification of the document structures	20
7.1 Sets of constituents	24
7.2 Constituents	35
7.3 Attributes	61
7.4 Subsidiary definitions	87
7.4.1 Predicates	87
7.4.2 Functions	95
7.5 Additional terminological definitions	97
7.6 Index of predicate symbols, operator symbols and attribute names	100
Annex A Tutorial on the specification language	107
A.1 Introduction	107
A.2 Atomic constructs	107
A.3 Composite constructs	108
A.4 Spots	109
A.5 Predicates	109
A.6 Operators	110
A.7 Predicates for the formal specifications of ISO 8613	113
A.8 Further examples	113

11	Formal specification of the geometric graphics content architectures	198
11.1	Interface to the Document Profile	200
11.2	Interface to the Document Architecture	201
11.3	Attributes of the Geometric Graphics Content Architecture	203
12	Formal specification of the defaulting mechanism for defaultable attributes	228
12.1	General functions	228
12.2	Functions for the values of defaultable attributes	245
12.2.1	Defaultable attributes of ISO/IEC 8613-2	245
12.2.2	Defaultable attributes of ISO/IEC 8613-6	274
12.2.3	Defaultable attributes of ISO/IEC 8613-7	298
12.2.4	Defaultable attributes of ISO/IEC 8613-8	310
13	Index of predicate symbols, operator bols and attribute names	327
Annex A:	Tutorial on the specification language	354
A.1	Introduction	354
A.2	Atomic constructs	354
A.3	Composite constructs	355
A.4	Spots	356
A.5	Predicates	356
A.6	Operators	357
A.7	Predicates for the formal specifications of ISO/IEC 8613	360
A.8	Further examples	360

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 8613-10 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 18, *Document processing and related communication*.

This second edition is a revision of the first edition (ISO/IEC 8613-10:1991). It also incorporates amendment 1:1991, amendment 2:1991, amendment 3:1992, amendment 4:1992 and amendment 5:1993.

ISO/IEC 8613 consists of the following parts, under the general title *Information technology — Open Document Architecture (ODA) and Interchange Format*:

- *Part 1: Introduction and general principles*
- *Part 2: Document structures*
- *Part 3: Abstract interface for the manipulation of ODA documents*
- *Part 4: Document profile*
- *Part 5: Open Document Interchange Format*
- *Part 6: Character content architectures*
- *Part 7: Raster graphics content architectures*
- *Part 8: Geometric graphics content architectures*
- *Part 9: Audio content architectures*
- *Part 10: Formal specifications*
- *Part 11: Tabular structures and tabular layout*
- *Part 12: Identification of document fragments*
- *Part 13: Spreadsheet*
- *Part 14: Temporal relationships and non-linear structures*

Annex A of this part of ISO/IEC 8613 is for information only.

Information technology — Open Document Architecture (ODA) and Interchange Format —

Part 10: Formal specifications

1 Scope

The purpose of ISO/IEC 8613 is to facilitate the interchange of documents.

In the context of ISO/IEC 8613, documents are considered to be items such as memoranda, letters, invoices, forms and reports, which may include pictures and tabular material. The content elements used within the documents may include graphic characters, geometric graphics elements and raster graphics elements, all potentially within one document.

NOTE — ISO/IEC 8613 is designed to allow for extensions, including hypermedia features, spreadsheets and additional types of content such as audio and video.

In addition to the content types defined in this International Standard, ODA also provides for arbitrary content types to be included in documents.

ISO/IEC 8613 applies to the interchange of documents by means of data communications or the exchange of storage media.

It provides for the interchange of documents for either or both of the following purposes:

- to allow presentation as intended by the originator;
- to allow processing such as editing and reformatting.

The composition of a document in interchange can take several forms:

- formatted form, allowing presentation of the document;
- processable form, allowing processing of the document;
- formatted processable form, allowing both presentation and processing.

ISO/IEC 8613 also provides for the interchange of ODA information structures used for the processing of interchanged documents.

This part of ISO/IEC 8613

- specifies a formal description technique appropriate for describing the technical specifications of the document structures (ISO/IEC 8613-2), the document profile (ISO/IEC 8613-4) and the content architectures (currently ISO/IEC 8613-6, -7 and -8);
- gives formal specifications of the document structures, the document profile and the content architectures using this formal description technique.

The aim of the formal specifications of ODA (FODA) is to provide a precise and unambiguous interpretation of the technical specifications in other parts of ISO/IEC 8613 (currently parts 2, 4, 6, 7, and 8), using formal syntax and formal semantics.

FODA can be used

- as a basis for implementations of ISO/IEC 8613;
- as a validation tool for the verification of conforming systems;
- as a reference point for examining future extensions and revisions to ISO/IEC 8613.

If a discrepancy between the natural language text and the formal specifications should be discovered, the natural language text should be regarded as the valid interpretation of this International Standard until the discrepancy is resolved.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 8613. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 8613 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 2022:1994, *Information technology — Character code structure and extension techniques*.

ISO/IEC 6429:1992, *Information technology — Control functions for coded character sets*.

ISO/IEC 6937:1994, *Information technology — Coded graphic character set for text communication — Latin alphabet*.

ISO 8601:1988, *Data elements and interchange formats — Information interchange — Representation of dates and times*.

ISO/IEC 8613-1:1994, *Information technology — Open Document Architecture (ODA) and Interchange format: Introduction and general principles*.

ISO/IEC 8613-2:1995, *Information technology — Open Document Architecture (ODA) and Interchange format: Document structures*.

ISO/IEC 8613-4:1994, *Information technology — Open Document Architecture (ODA) and Interchange format: Document profile*.

ISO/IEC 8613-6:1994, *Information technology — Open Document Architecture (ODA) and Interchange format: Character content architectures*.

ISO/IEC 8613-7:1994, *Information technology — Open Document Architecture (ODA) and Interchange format: Raster graphics content architectures*.

ISO/IEC 8613-8:1994, *Information technology — Open Document Architecture (ODA) and Interchange format: Geometric graphics content architectures*.

ISO/IEC 8632-1:1992, *Information technology — Computer graphics — Metafile for the storage and transfer of picture description information — Part 1: Functional specification*.

ISO/IEC 8632-3:1992, *Information technology — Computer graphics — Metafile for the storage and transfer of picture description information — Part 3: Binary encoding*.

ISO/IEC 9541-2:1991, *Information technology — Font information interchange — Part 2: Interchange format*.

3 Definitions

For the purposes of this part of ISO/IEC 8613, the definitions given in ISO/IEC 8613-1 apply.

4 Syntax and semantics of the specification language

This clause describes the formal description technique used for the formal specifications.

NOTE — A tutorial on this formal description technique is given in annex A.

4.1 Basic concepts

ISO/IEC 8613 describes document structures, the document profile and the content architectures in terms of abstract information constructs which are drawn from the following structural categories:

- An ODA construct may be an **atomic** construct, e.g. an attribute name or a natural number within an object identifier.
- An ODA construct may be a **composite** construct, i.e. may consist of other constructs. With respect to their interrelationship, three kinds of composition are distinguished. An ODA construct may be
 - a) a **set**;
 - b) a **function** (mapping);
 - c) a **sequence** (list, string);

of other ODA constructs.

For example, a specific layout description is a set (of constituents), a constituent is a nomination (see below) which is a function or mapping (from attribute names onto attribute values), and an attribute value of 'subordinates' or of 'object identifier' is a sequence (of atomic natural numbers).

It is these very structures which are captured by the language used for the formal specifications of ISO/IEC 8613. The language used is called IMCL, *Information Modelling by Composition Language*. The semantics of the specification language consist of the following abstract elements:

- the universe which is a non-empty set of **entities** of the following kinds:
 - a) constructs;
 - b) spots;
 - c) spotsets (i.e. sets of spots);
 - d) the entity UNDEF ("undefined");
- functions from the universe to the universe, that is, operators on entities of the universe;
- predicates in the universe, that is, predicates on entities of the universe.

A **construct** is an information object which is one of the following:

- an atomic construct or **atom**, for short;
- a composite construct or **compound**, for short, which may be
 - a) a **collection**, which is an unordered set of component constructs;
 - b) a **nomination**, which is a function that can be regarded as an unordered set of ordered pairs where each pair consists of a *name* and a *value*;
 - c) a **catenation**, which is a sequence of component constructs.

The special terminology for composite constructs is to distinguish them from other sets, functions or sequences.

In order to be able to address components in constructs of arbitrary compositional structure, the concept of a **spot** is introduced. This concept is an abstract counterpart for the intuitive idea associated with pointing into an information structure at some position and saying "here". However, in general the "here" is not identified uniquely by the component construct as such (e.g. in a word, the same letter may occur several times), but rather by the context in which it appears. To deal conceptually with the idea of "here" requires a way to identify contexts.

The concept of a spot allows the distinction to be made between a considered construct and its position within a comprising composite construct of which it is a component. For example, the character string "data" (a catenation) has the component constructs 'd', 'a' and 't'. Whereas 'd' and 't' appear at one spot each, the 'a' appears at two spots, namely at the second and at the fourth position counted from the front end. So, "data" has four

component spots, but only three component constructs. If a construct is considered outside any context, it is said to be at its **ownspot**.

Spots are usually selected by selection criteria. However, a selection criterion need not be unique. Thus, the objects most naturally dealt with are not even spots, but rather sets of spots or **spotsets**. Consequently, there are no expressions for single spots, but for singleton spotsets, instead, i.e. for spotsets with only one spot (see $\in$ in 4.3 and $\wedge$ in 4.4).

It should be noted that the specification language is built on first-order predicate logic and mathematical set theory.

4.2 Syntax of the specification language

This subclause defines the syntax of the specification language, i.e., each expression in the formal specifications is built using the syntax rules given in this clause. The semantics of the terminal symbols appearing in the syntax rules are specified in 4.3 to 4.5.

Remarks on the meta-language:

The symbol pairs $\{ \}$, $[]$ and $- . -$ as well as the symbols $::=$, $|$, $\dots$, and $\equiv$ belong to the meta-language. They have the following meanings:

$::=$	separates the meta-variable to be defined (left-hand side) from the meta-language expression which defines it (right-hand side)
$\{ \}$	delimit a syntactical unit
$[]$	delimit a syntactical unit and indicate that this syntactical unit is optional, i.e. may also be absent NOTE 1: The meta-language symbols $[$ and $]$ are different from the special-characters $[$ and $]$ used in the production rules for empty-constant, explicit-composition-term and extensional-collection-term.
$- . -$	delimit a comment in the meta-language text
$ $	separates alternative syntactical units, i.e. indicates a choice of exactly one of the syntactical units e.g. $\{a m p\}x$ means ax or mx or px NOTE 2: The meta-language symbol $ $ is different from the special-character $ $ used in terms denoting sets.
$\dots$	is a convenient notation for recursive definitions: the symbol follows a syntactical unit which may appear one or more times, i.e. which may be repeated several times e.g. $\{yf\}\dots$ means yf or $yfyf$ or $yfyfyf$ etc. e.g. $i[so]\dots$ means i or iso or $isoso$ etc.
$\equiv$	A space in the language definition requires one or more blanks in the defined language expression. Just the reverse is indicated by the symbol $\equiv$, which requires an immediate juxtaposition of the neighboring strings of the specification language. Where syntactical uniqueness is not affected, blanks may be omitted in expressions of the specification language (e.g. before and after parentheses).

For the sake of readability, the symbols “(” and “)” are used as meta-variables (instead of word-symbols such as “left-del” or “right-del”). All other meta-variables are strings of lower-case letters (with the hyphen for linking the components of a meta-variable).

Production rules:

expression $::=$

formula | term

formula $::=$

prime-formula |

not formula | formula { and | or | impl | iff | xor } formula |

$\exists$ var (formula) | $\forall$ var (formula) |

(formula)

NOTE 3: The terminal symbols used in this production rule have the usual semantics of first-order predicate logic: not is the logical negation, and, or, xor (exclusive or), impl (implies) and iff (if and only if) are the usual logical connectors, $\forall$ (for all) and $\exists$ (exists) are the logical quantifiers.

prime-formula ::=

[parameter-part] predicate-symbol-part ...
[parameter-part predicate-symbol-part ...] ... [parameter-part]

predicate-symbol-part ::=

upper-case-letter [$\equiv$ letter | $\equiv$ digit]... $\equiv$ lower-case-letter [$\equiv$ letter | $\equiv$ digit]... |
= | $\neq$ | $<$ | $\leq$ | $>$ | $\geq$ | $\in$ | $\notin$ | $\hat{\in}$ | $\subset$ | $\supset$ | $\supseteq$

NOTE 4: The semantics of the terminal symbols ($=$, $\neq$, ... $\supseteq$) in this production rule are specified in 4.3.

term ::=

var |
constant |
operator-term |
explicit-composition-term |
conditional-term |
extensional-collection-term |
extensional-spotset-term |
spot-selection-term |
(term)

var ::=

lower-case-letter [$\equiv$ letter | $\equiv$ digit]... [$\equiv$ subscript-digit]...

constant ::=

standard-constant |
nonstandard-constant

standard-constant ::=

UNDEF |
empty-constant |
number-atom-constant

empty-constant ::=

[] —. empty collection .— |
[:] —. empty nomination .— |
[$\rightarrow$] —. empty catenation .— |
 $< >$ —. empty spotset .—

number-atom-constant ::=

[+ $\equiv$ | - $\equiv$] digit [$\equiv$ digit]... [$\equiv$. $\equiv$ digit [$\equiv$ digit]...]

nonstandard-constant ::=

$\equiv$ character [$\equiv$ character]... $\equiv$ ' —. restriction on apostrophe occurrence .—

operator-term ::=

[parameter-part] operator-symbol-part ...
[parameter-part operator-symbol-part ...] ... [parameter-part]

operator-symbol-part ::=

upper-case-letter [$\equiv$ upper-case-letter | $\equiv$ digit | $\equiv$ _]... |
 $\wedge$ | + | - | * | / | $\cup$ | $\cap$ | \ | // | . | • | $\downarrow$ | $\uparrow$

NOTE 5: The semantics of the terminal symbols ($\wedge$, +, ... $\uparrow$) in this production rule are specified in 4.4.

explicit-composition-term ::=

[term [; term]...] —. collection .— |
[term : term [; term : term]...] —. nomination .— |
[$\rightarrow$ term [$\rightarrow$ term]... $\rightarrow$] —. catenation .— |
" $\equiv$ character [$\equiv$ character]... $\equiv$ " —. catenation of characters, restriction on quote occurrence .—

conditional-term ::=

IF formula THEN term ELSE term

NOTE 6: The semantics of the terminal symbols (IF , THEN and ELSE) in this production rule are specified in 4.5.

extensional-collection-term ::=

[var | formula] —. collection of constructs for which the formula holds.—

extensional-spotset-term ::=

< var | formula > —. union of singleton spotsets for which the formula holds.—

spot-selection-term ::=

term spot-selection-clause |
elliptic-spot-selection-term

spot-selection-clause ::=

<var || formula> |
<formula> —. x s is assumed for var .— |
<name-specification [, name-specification] ... >

elliptic-spot-selection-term ::=

term { . | • | ↓ | ↑ } name-specification

NOTE 7: The semantics of the terminal symbols (. , • , ↓ and ↑) in this production rule are specified in 4.5.

name-specification ::=

nonstandard-constant | var

parameter-part ::=

term | (term [, term] ...)

character ::=

letter | digit | subscript-digit | special-character

letter ::=

upper-case-letter | lower-case-letter

upper-case-letter ::=

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

lower-case-letter ::=

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

digit ::=

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

subscript-digit ::=

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

special-character ::=

. | , | ; | + | - | —. etc. —

4.3 Predicate symbols with built-in semantics

A sequence of predicate-symbol-parts is referred to as a **predicate symbol**. For each n-ary predicate symbol there is an n-ary predicate on the universe of the specification language, i.e. an n-ary relation on entities of the universe. Some predicate symbols have built-in semantics which are introduced by the following.

NOTE — The predicate-symbol-parts are syntactically distinguished from operator-symbol-parts and variables.

True	means	the valid fact (something stated as being true)
False	means	the invalid fact (something stated as being false)
IsAtom(t)	means	t is an atomic construct or atom, for short
IsNat(t)	means	t is a natural number (1, 2, ...; zero excluded)
IsInt(t)	means	t is an integer number (... -2, -1, 0, 1, 2, ...)
IsReal(t)	means	t is a real number
IsCol(t)	means	t is a collection
IsNom(t)	means	t is a nomination
IsCat(t)	means	t is a catenation
IsSpotset(t)	means	t is a spotset
IsSingle(t)	means	t is a singleton spotset
$t_1 = t_2$	means	t_1 is equal to t_2 (all entities)
$t_1 \neq t_2$	means	<u>not</u> $t_1 = t_2$
$t_1 < t_2$	means	t_1 is less than t_2 (numbers)
$t_1 \leq t_2$	means	t_1 is less than or equal to t_2
$t_1 > t_2$	means	t_1 is greater than t_2
$t_1 \geq t_2$	means	t_1 is greater than or equal to t_2
$t_1 \in t_2$	means	t_1 is element of t_2 (collections)
$t_1 \notin t_2$	means	<u>not</u> $t_1 \in t_2$
$t_1 \dot{\in} t_2$	means	t_1 is singleton spotset and subset of t_2 (spotsets)
$t_1 \subset t_2$	means	t_1 is subset of t_2 (collections or spotsets)
$t_1 \subseteq t_2$	means	t_1 is subset of or equal to t_2 (collections or spotsets)
$t_1 \supset t_2$	means	t_2 is subset of t_1 (collections or spotsets)
$t_1 \supseteq t_2$	means	t_2 is subset of or equal to t_1 (collections or spotsets)

The unary predicate symbols mean predicates for expressing that an entity belongs to a certain class or “type” of entities, i.e. has a particular property. The binary predicate symbols refer to predicates which indicate whether or not a particular relationship holds for two entities.

4.4 Operator symbols with built-in semantics

A sequence of operator-symbol-parts is referred to as an **operator symbol**. For each n-ary operator symbol there is an n-ary operator or function from the universe to the universe of the specification language, i.e., a mapping from n-ary tuples of entities onto entities of the universe. Some operator symbols have built-in semantics which are introduced by the following.

NOTE — The operator-symbol-parts are syntactically distinguished from predicate-symbol-parts and variables. For all operators it holds that the result is UNDEF, if a parameter term does not meet the requirement stated below.

C t	If t denotes a singleton spotset, C t denotes the component construct at the spot given by t .
N t	If t denotes a singleton spotset of a spot that is a component of a nomination (“immediately inward” of a nomination is the formal term), then N t denotes the name construct of the component as it is within the nomination.
F t	If t denotes a set of exactly one spot immediately inward of a catenation spot, F t denotes the front part of this catenation up to but excluding the component given by t (catenation of components with lower position than t).

$\text{CARD } t$	If t denotes a collection or a spotset, $\text{CARD } t$ denotes its cardinality, i.e., the number of component constructs or the number of spots.
$\text{R } t$	If t denotes a set of exactly one spot immediately inward of a catenation spot, $\text{R } t$ denotes the rear part of this catenation up to but excluding the component given by t (catenation of components with higher position than t).
$t_1 + t_2 \quad t_1 - t_2 \quad t_1 * t_2 \quad t_1 / t_2$	If t_1 and t_2 denote numbers, the terms denote numbers as known from arithmetic. The usual precedence rules for arithmetical operators apply.
$t_1 \cup t_2 \quad t_1 \cap t_2 \quad t_1 \setminus t_2$	If t_1 and t_2 denote either collections or spotsets, the terms denote their set-theoretic union, intersection or difference, i.e. either collections or spotsets.
$t_1 // t_2$	If t_1 and t_2 denote catenations, $t_1 // t_2$ denotes the catenation obtained by concatenating the two catenations in the given order.
$\sim t$	If t denotes a construct, then $\sim t$ ("ownspot of t ") denotes the singleton spotset containing the ownspot of t .
$t \bullet$	If t denotes a spotset containing no atom spots (spots with atoms), then $t \bullet$ ("next inwards") denotes the set of all spots which are immediately inward of the spots of the spotset t .
$t \downarrow$	If t denotes a spotset, $t \downarrow$ denotes the set of all terminal spots inward of or - for atoms and empty constructs - equal to the spots given by t (read "most inward").
$t \circ$	If t denotes a spotset without ownspots, $t \circ$ denotes the set of all spots which are immediately outward of the spots given by t (read "next outward").
$t \uparrow$	If t denotes a spotset, $t \uparrow$ denotes the set of all ownspots outward of or - for ownspots - equal to the spots given by t (read "most outward").
$t \bullet \quad t \downarrow \quad t \circ \quad t \uparrow$	If t denotes the empty spotset, the operator-terms denote the empty spotset, too.

The normal evaluation order for expressions is from left to right, with the following exceptions:

- If a term is enclosed in opening and closing parentheses this term is evaluated first;
- Operators have precedence over predicates;
- Between operators the precedence order is:
 - 1.: $\sim$ (ownspot of)
 - 2.: $\bullet \quad \circ \quad \downarrow \quad \uparrow$ (next inward, next outward, most inward, most outward)
 - 3.: spot-selection-clause
 - 4.: spot-selection-term
 - 5.: other operators

4.5 Other terms

Apart from operator-terms there are other compound terms which result in constructs or spotsets (or UNDEF). Their built-in semantics are introduced by the following.

$[t_1; t_2; t_3]$	If t_i denote constructs, the whole term denotes the collection which contains the constructs t_i as components. (This is an example for explicit-composition-term)
$[\]$	Denotes the empty collection. (This is an example for empty-constant)
$[n_1 : c_1; n_2 : c_2]$	If n_i and c_i denote constructs, where all n_i are distinct, the whole term denotes the nomination which contains the constructs c_i as components under the (unique) names n_i . (This is an example for explicit-composition-term)
$[:]$	Denotes the empty nomination. (This is an example for empty-constant)

$[\rightarrow m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow]$	If m_i denote constructs, the whole term denotes the catenation which contains the constructs m_i as components — also referred to as members — in the indicated sequence. (This is an example for explicit-composition-term)
"ODA Part 2"	Denotes the catenation $[\rightarrow 'O' \rightarrow 'D' \rightarrow 'A' \rightarrow ' ' \rightarrow 'P' \rightarrow 'a' \rightarrow 'r' \rightarrow 't' \rightarrow ' ' \rightarrow '2' \rightarrow]$. A string of characters enclosed in quotes denotes the catenation of those characters. A pair of quotes in the string stands for a single one in the catenation. (This is an example for explicit-composition-term)
$[\text{---}]$	Denotes the empty catenation. (This is an example for empty-constant)
$\langle \rangle$	Denotes the empty spotset. (This is an example for empty-constant)
IF formula THEN t_1 ELSE t_2	If t_1 and t_2 are terms, the whole term denotes the same as t_1 or t_2 , depending on whether the formula is True or False, respectively. (This is an example for conditional-term)
$[\textit{var} \mid \text{formula}]$	Denotes the collection of all constructs $\textit{var}$ which satisfy the formula. (This is an example for extensional-collection-term)
$\langle \textit{var} \mid \text{formula} \rangle$	Denotes the spotset which is the union of all singleton spotsets $\textit{var}$ which satisfy the formula. (This is an example for extensional-spotset-term)
$t \langle \textit{var} \parallel \text{formula} \rangle$	If t denotes a (possibly empty) spotset, the whole term denotes the union of all singleton spotsets $\textit{var}$ which contain a spot taken from t and for which the formula is True. (This is an example for spot-selection-term)
	Three elliptic notations are provided for frequently occurring spot-selection-clauses:
$t \langle \text{formula} \rangle$	If a variable $\textit{var}$ is not introduced explicitly, the abbreviated term $t \langle \text{formula} \rangle$ is evaluated for the standard variable xs (singleton set of the spot under examination or examination spot, for short). (This is an example for spot-selection-term)
$t \langle n_1, n_2, \dots \rangle$	If the formula has the structure $N \textit{var} = n_1 \text{ or } N \textit{var} = n_2 \text{ or } \dots$ where n_i are name-specifications, the formula may be abbreviated as a list of name-specifications. (This is an example for spot-selection-term)
$t.n \quad t \downarrow n \quad t * n \quad t \uparrow n$	If there is only one name-specification n used for spot selection, an elliptic-spot-selection-term is provided as an abbreviation of special spot-selection-terms (ending with $.$, $*$ etc.). The n stands for $\langle N \textit{xs} = n \rangle$ (see name qualification in programming languages). (This is an example for elliptic-spot-selection-term)

4.6 Notational simplifications

The common notational simplifications for successive logical quantifications can be used. The following examples explain these "short-hand" notations which are usually applied in first-order predicate logic:

The expression	$\forall x(\forall y(\exists z(\text{formula})))$
may be written as	$\forall x \forall y \exists z (\text{formula})$
or even as	$\forall x, y \exists z (\text{formula})$

A further abbreviation is used to help emphasize the "essential part" of a quantified formula:

The expression	$\forall x(x \in m \text{ impl formula})$
may be written as	$\forall x \in m (\text{formula})$
and	$\exists x(x \in m \text{ and formula})$
may be written as	$\exists x \in m (\text{formula})$

This notation can be combined with the previous one:

The expression	$\forall x(x \in m \text{ impl } \forall y(\exists z(z \in p \text{ and formula))))$
----------------	--

may be written as $\forall x \in m, y \exists z \in p$ (formula)

However, note that the expression

$$\forall y, x \in m \exists z \in m \text{ (formula)}$$

includes the restriction $y \in m$ and has therefore to be expanded to

$$\forall y(y \in m \text{ impl } \forall x(x \in m \text{ impl } \exists z \in m \text{ (formula)}))$$

The same abbreviations are used for $\hat{c}$.

5 Structure of the formal specifications

This clause outlines the general concepts for the formal specifications. Those terms which are used at several places throughout the formal specifications are contained in clause 6.

Formally speaking, the formal specifications consist of a single formula in first-order predicate logic. This formula is called the overall formula. The formula consists of other formulae which are connected by and:

$$\text{formula}_1 \text{ and formula}_2 \text{ and formula}_3 \text{ and ... formula}_n$$

This overall formula is distributed over clauses 7 to 11. Its outline is as follows:

... IsInterchangeSet^{2.1}(*doc*) ...
and ... IsGenericDocumentDescription^{2.2}(*gdoc*) ...
 (see clause 7)
and ... (*cont*)DescribesContPortOf^{2.153}(*b*)
and ... IsDocumentProfile^{4.1}(*cst*) ...
 (see clause 8)
and ... IsCIEColour^{4.96}(*v*) ...
and ... IsDefaultableCharacterContentArchitectureAttribute^{6.1}(*att*) ...
 (see clause 9)
and ... IsGraphicCharacter^{6.113}(*v*) ...
and ... IsDefaultableRasterGraphicsContentArchitectureAttribute^{7.1}(*att*) ...
 (see clause 10)
and ... IsInterleavingFormatValue^{7.32}(*v*) ...
 ... IsDefaultableGeometricGraphicsContentArchitectureAttribute^{8.1}(*att*) ...
 (see clause 11)
and ... IsRegisteredEdgeType^{8.61}(*v*) ...

The complete part of each section of the overall formula is given in the indicated clauses. The overall formula has been split according to the individual parts of ISO/IEC 8613.

In the present context, each of these formulae is called a “definition” and identified through a unique reference number. A definition defines either a concept used in the narrative part of ISO/IEC 8613 or a concept which has a subsidiary function in the network of definitions in that it has been separated and “encapsulated” to render other definitions more readable.

The definitions are grouped into several clauses and subclauses. For example, within the formal specifications of the document structures the definitions relating to the sets of constituents are contained in 7.1, those relating to constituents in 7.2 and those relating to attributes in 7.3. In addition, concepts which are not used in the formal specification but are used in the text of ISO/IEC 8613-2 are defined in 7.5. The definitions relating to the document profile are given in clause 8, those relating to the character content architectures in clause 9, etc.

The “factorization” of definitions is only for the convenience of authors and readers; it does not in any way impair the formal rigidity of the approach.

Variables occurring in the definitions are always bound by universal ($\forall$) or existential ($\exists$) quantifiers. Therefore, once a value has been chosen for a variable it has to be retained throughout the scope of the quantifier wherever the variable appears.

All predicates apart from those pertaining to the specification language are defined using the same format. For unary predicates the format is:

$\forall \text{ variable } (\text{predicate-symbol}(\text{variable}) \text{ iff formula})$

A similar format is applied for n -ary predicates. There, the variables have been placed around the predicate symbol in a “natural” way, e.g. $(id)IsIdContIn(b)$ is expected to be read as “ id is an identifier of a content portion in the basic object b ”.

A “predicate” in first-order predicate logic is a propositional pattern where some places are left free for the insertion of individual entities. An example is the pattern “... is greater than ...”. Once the free places have been occupied by entities, the pattern becomes a proposition for which the “truth value” can be evaluated. The proposition “5 is greater than 2” evaluates to True, whereas the propositions “3 is greater than 5” and “a Mercedes is greater than SC 18” evaluate to False. Note that a proposition may never evaluate to an undefined truth value.

A unary predicate is a pattern with only one free parameter place. Its definition is evaluated by substituting an individual entity for the parameter. This yields a proposition on the left-hand side of the iff and transforms the whole expression in parentheses into a proposition.

A uniform format has also been adopted for the definition of most operators. It relies on the conditional-term of the specification language:

$\forall variable (operator-symbol(variable) = \text{IF } formula \text{ THEN } term \text{ ELSE UNDEF})$

An “operator” is a term pattern where some places are left free for the insertion of individual entities. An example is the pattern “... plus ...”. Once the free places have been occupied by entities, the pattern becomes a term, i.e. an expression which evaluates to or denotes an individual entity. The term “5 plus 2” denotes the entity 7, whereas the term “WG 3 plus SC 18” denotes the entity which is represented by UNDEF in the formal specification language. Therefore, a term evaluates to “undefined”, if the operator is not defined for the inserted entities.

It should be noted that the formal specifications cannot be applied *directly* to the Open Document Interchange Format (ISO/IEC 8613-5) and the reader should not necessarily expect to see any direct correspondences. The formal specifications are based on the textual description of the concepts of ISO/IEC 8613 and do not necessarily reflect the ODIF encoding of ODA documents. Those clauses in ISO/IEC 8613 which specify the ODIF encoding usually impose additional rules for the make-up of the interchanged data stream. These additional rules are considered outside of the scope of the formal specifications: The ODIF encoding specifies these additional rules formally and there is therefore no need for repeating these additional rules in FODA.

Furthermore, although defaultable attributes or parameters are not required to be present in the interchanged data stream they are considered to be present for the formal specifications since their value, unlike that of non-mandatory attributes, must be derivable by the defaulting mechanism defined in ISO/IEC 8613-2, 9.1.2.4. The concept of defaultable attributes or parameters is formally covered by the predicate $IsPlaceholder$ (see clause 6).

Throughout this formal specification the predicate “ $IsAtom(v)$ ” is used whenever v is an entity defined in an International Standard other than ISO/IEC 8613. For example,

```

1   $\forall v$ 
2   $({}_o IsISO6937MSCharacter(v) \text{ iff}$ 
3     $IsAtom(v))_o$ 
```

is used to indicate that the entity v , a character from the minimum subrepertoire of ISO 6937-2, is defined in ISO 6937. The definition of this entity is outside the scope of ISO/IEC 8613.

Where advantageous, *imported definitions* may be used in lieu of $IsAtom(v)$. An imported definition is a function or a predicate whose semantics are defined by an International Standard other than ISO/IEC 8613. Within the scope of this part of ISO/IEC 8613 the semantics for imported definitions are expressed only by semiformal descriptions with references to the appropriate International Standards. The formal specification of these definitions is outside the scope of this part of ISO/IEC 8613.

The following hints are intended to help the reader understand the formal specifications. Note that there is no reading or evaluation sequence in a formula apart from that which results from applying the precedence rules defined for logical symbols and operator symbols (along with parentheses). The overall formulae are *not* intended to imply a sequence of definitions and need *not* be read in the order shown. Rather the order is intended to support a systematic approach to the specifications. The same applies for any definition which in turn consists of formulae.

Superscripts are used to indicate the number of the definition of a predicate symbol or operator symbol (e.g., $IsISO6937MSCharacter^{1,21}(v)$). A superscript does not appear if the symbol is part of the specification language

(i.e., is a predefined symbol) or is the one currently under definition. These superscripts (reference numbers) are not part of the specification language but are only for the convenience of the readers so they can quickly look up the definition of a referenced symbol.

Furthermore, line numbers and subscripted parentheses are used to structure the formulae and to establish cross references between the semi-formal descriptions and the formal definitions (see above). Again, these line numbers and subscripts are not part of the formal specifications but are only for the convenience of the readers.

Each definition of the formal specification is introduced by a so-called semi-formal description. These semi-formal descriptions give an explanation of the definitions to which they belong in English, making an extensive use of the terms of the specification method. The semi-formal descriptions do not belong to the formal specification itself, however, they are intended to help understand the formal definitions.

In the semi-formal descriptions, references to line numbers are put in parentheses, e.g., (8) or (11,12) or (4-10), whereas references to expressions between subscripted parentheses are prefixed by “expr.”, e.g. (expr. 5).

6 Commonly used definitions

This clause contains those definitions which are not specific to the formal specification of the document structures, the document profile or one of the content architectures only.

Semiformal Description 1.1

Predicate “is a non-empty collection”

An entity *col* is a non-empty collection (set of constructs) if it is a collection which is not empty.

Definition 1.1

- 1 $\forall col$
- 2 $({}_o \text{IsNeCol}(col) \text{ iff}$
- 3 $\text{IsCol}(col) \text{ and } col \neq []_o)$

Semiformal Description 1.2

Predicate “is a non-empty nomination”

An entity *nom* is a non-empty nomination (mapping of names onto constructs) if it is a nomination which is not empty.

Definition 1.2

- 1 $\forall nom$
- 2 $({}_o \text{IsNeNom}(nom) \text{ iff}$
- 3 $\text{IsNom}(nom) \text{ and } nom \neq [:]_o)$

Semiformal Description 1.3

Predicate “is a non-empty catenation”

An entity *cat* is a non-empty catenation (sequence of constructs) if it is a catenation which is not empty.

Definition 1.3

- 1 $\forall cat$
- 2 $({}_o \text{IsNeCat}(cat) \text{ iff}$
- 3 $\text{IsCat}(cat) \text{ and } cat \neq [\rightarrow]_o)$

Semiformal Description 1.4

Predicate “is an empty collection”

An entity *col* is an empty collection if it contains no components.

Definition 1.4

- 1 $\forall col$
- 2 $({}_o \text{IsEmptyCol}(col) \text{ iff}$
- 3 $col = []_o)$

Semiinformal Description 1.5

Predicate “is an empty nomination”

An entity *nom* is an empty nomination if it contains no components.

Definition 1.5

- 1 $\forall nom$
- 2 $({}_o \text{IsEmptyNom}(nom) \text{ iff}$
- 3 $nom = [:]_o)$

Semiinformal Description 1.6

Predicate “is an empty catenation”

An entity *cat* is an empty catenation if it contains no components.

Definition 1.6

- 1 $\forall cat$
- 2 $({}_o \text{IsEmptyCat}(cat) \text{ iff}$
- 3 $cat = [\rightarrow]_o)$

Semiinformal Description 1.7

Predicate “is a non-negative integer”

An entity *int* is a non-negative integer if it is a natural number or zero.

Definition 1.7

- 1 $\forall int$
- 2 $({}_o \text{IsNnInt}(int) \text{ iff}$
- 3 $\text{IsNat}(int) \text{ or } int = 0_o)$

Semiinformal Description 1.8

Predicate “is a pair of non-negative integers”

A pair of non-negative integers is a catenation of two non-negative integers.

Definition 1.8

- 1 $\forall q$
- 2 $({}_o \text{IsPairOfNnInt}(q) \text{ iff}$
- 3 $\exists l, r$
- 4 $({}_1 q = [\rightarrow l \rightarrow r \rightarrow] \text{ and } \text{IsNnInt}^{1.7}(l) \text{ and } \text{IsNnInt}^{1.7}(r)_1)_o)$

Semiformal Description 1.9

Predicate “is a pair of positive integers”

A pair of positive integers is a catenation of two positive integers.

Definition 1.9

```

1   $\forall q$ 
2   $({}_o \text{IsPairOfPosInt}(q) \text{ iff}$ 
3     $\exists l, r$ 
4     $({}_i q = [ \rightarrow l \rightarrow r \rightarrow ] \text{ and } \text{IsNat}(l) \text{ and } \text{IsNat}(r)_i)_o)$ 
```

Semiformal Description 1.10

Predicate “is an octet string”

An octet string is an atomic construct in the formal specification.

Definition 1.10

```

1   $\forall v$ 
2   $({}_o \text{IsOctetString}(v) \text{ iff}$ 
3     $\text{IsAtom}(v)_o)$ 
```

Semiformal Description 1.11

Function “position from the front end”

PF returns the position number of the indicated component, the count beginning with the first component and ending at and including the considered component. The operand p is required to denote a set of exactly one spot immediately inward of a catenation.

Definition 1.11

```

1   $\forall p$ 
2   $({}_o \text{PF}(p) =$ 
3     $\text{IF } \text{IsSingle}(p) \text{ and } \text{IsCat}(C \ p^*)$ 
4     $\text{THEN } \text{LENGTH}^{1.16} (F \ p) + 1$ 
5     $\text{ELSE } \text{UNDEF}_o)$ 
```

Semiformal Description 1.12

Function “position from the rear end”

PR returns the position number of the indicated component, the count beginning with the last component and ending at and including the considered component. The operand p is required to denote a set of exactly one spot immediately inward of a catenation.

Definition 1.12

```

1   $\forall p$ 
2   $({}_o \text{PR}(p) =$ 
3     $\text{IF } \text{IsSingle}(p) \text{ and } \text{IsCat}(C \ p^*)$ 
4     $\text{THEN } \text{LENGTH}^{1.16} (R \ p) + 1$ 
5     $\text{ELSE } \text{UNDEF}_o)$ 
```

Semiformal Description 1.13

Function “first component”

HEAD returns the first component of a non-empty catenation *cat*.

Definition 1.13

```

1   $\forall cat$ 
2  ( $_o$  HEAD(cat) =
3    IF IsNeCat1.3(cat)
4    THEN C ( $\sim cat \bullet \langle xs \parallel PF^{1.11} xs = 1 \rangle$ )
5    ELSE UNDEFo)

```

Semiformal Description 1.14

Function “last component”

LASTC returns the last component of a catenation *cat*.

Definition 1.14

```

1   $\forall cat$ 
2  ( $_o$  LASTC (cat) =
3    IF IsNeCat1.3(cat)
4    THEN C ( $\sim cat \bullet \langle xs \parallel PR^{1.12} xs = 1 \rangle$ )
5    ELSE UNDEFo)

```

Semiformal Description 1.15

Function “tail component”

TAIL returns the rear part of a non-empty catenation *cat* where the first component has been removed.

Definition 1.15

```

1   $\forall cat$ 
2  ( $_o$  TAIL(cat) =
3    IF IsNeCat1.3(cat)
4    THEN R ( $\sim cat \bullet \langle xs \parallel PF^{1.11} xs = 1 \rangle$ )
5    ELSE UNDEFo)

```

Semiformal Description 1.16

Function “length of a catenation”

LENGTH returns the number of component spots of the catenation *cat*.

Definition 1.16

```

1   $\forall cat$ 
2  ( $_o$  LENGTH (cat) =
3    IF IsCat(cat)
4    THEN CARD ( $\sim cat \bullet$ )
5    ELSE UNDEFo)

```

Semiformal Description 1.17

Function “collection of component constructs”

COLC returns the collection of the component constructs at the spots of the (possibly empty) spotset ss .

Definition 1.17

```

1   $\forall ss$ 
2   $({}_o \text{COLC}(ss) =$ 
3    IF  $\text{IsSpotset}(ss)$ 
4    THEN  $[g | \exists p \in ss (g = C p)]$ 
5    ELSE UNDEF $_o$ )
```

Semiformal Description 1.18

Function “name set of a nomination”

NAMS returns the name set of the given nomination n , i.e. the collection of names of the components of the nomination.

Definition 1.18

```

1   $\forall n$ 
2   $({}_o \text{NAMS}(n) =$ 
3    IF  $\text{IsNom}(n)$ 
4    THEN  $[m | \exists p \in \sim n \bullet (m = N p)]$ 
5    ELSE UNDEF $_o$ )
```

Semiformal Description 1.19

Predicate “is the placeholder for any attribute value”

An entity v is the placeholder for the value of a defaultable attribute if it is the atom with this particular interpretation.

NOTE — The term 'placeholder for any attribute value' is not an ISO/IEC 8613 attribute value but a construct which is introduced to achieve consistency throughout the formal specifications and to distinguish between mandatory and defaultable attributes or parameters. The actual value is dependent on the respective attribute and the defaulting rules and will be determined during the processing of a document (editing process, layout process, imaging process) whenever a value for a defaultable attribute or parameter is needed.

Definition 1.19

```

1   $\forall v$ 
2   $({}_o \text{IsPlaceholder}(v) \text{ iff}$ 
3     $v = \text{'placeholder for any attribute value'}_o)$ 
```

Semiformal Description 1.20

Predicate “is an ISO 6937 minimal subrepertoire string”

An ISO 6937 minimal subrepertoire string is a catenation of ISO 6937 minimal subrepertoire characters.

Definition 1.20

```

1   $\forall v$ 
2   $({}_o \text{IsISO6937MSString}(v) \text{ iff}$ 
3     $\text{IsNeCat}^{1.3}(v) \text{ and } \forall m \in \sim v \bullet (\text{IsISO6937MSCharacter}^{1.21}(C m))_o)$ 
```

Semiformal Description 1.21

Predicate “is an ISO 6937 minimal subrepertoire character”

An ISO 6937 minimal subrepertoire character is an element of the minimum subrepertoire of ISO 6937-2 and is considered an atomic construct in the formal specification.

Definition 1.21

- 1 $\forall z$
- 2 $({}_0 \text{IsISO6937MSCharacter}(z) \text{ iff}$
- 3 $\text{IsAtom}(z)_0)$

Semiformal Description 1.22

Predicate “is a sequence of different non-negative integers”

A sequence of different non-negative integers is a catenation of non-negative integers, where no integer occurs more than once.

Definition 1.22

- 1 $\forall q$
- 2 $({}_0 \text{IsSeqOfDiffNnInt}(q) \text{ iff}$
- 3 $\text{IsNeCat}^{1.3}(q) \text{ and}$
- 4 $\forall m \in \sim q \bullet (\text{IsNnInt}^{1.7}(C\ m)) \text{ and}$
- 5 $\text{CARD}(\text{COLC}^{1.17}(\sim q \bullet)) = \text{CARD}(\sim q \bullet)_0)$

Semiformal Description 1.23

Predicate “is a sequence of non-negative integers”

A sequence of non-negative integers is a catenation of non-negative integers, where an integer may occur more than once.

Definition 1.23

- 1 $\forall q$
- 2 $({}_0 \text{IsSeqOfNnInt}(q) \text{ iff}$
- 3 $\text{IsNeCat}^{1.3}(q) \text{ and}$
- 4 $\forall m \in \sim q \bullet (\text{IsNnInt}^{1.7}(C\ m))_0)$

Semiformal Description 1.24

Predicate “is a non-negative real number”

A non-negative real number is a real number whose value is greater than or equal to zero.

Definition 1.24

- 1 $\forall v$
- 2 $({}_0 \text{IsNnReal}(v) \text{ iff}$
- 3 $\text{IsReal}(v) \text{ and } v \geq 0.0_0)$

Semiinformal Description 1.25

Predicate “is a positive real number”

A positive real number is a real number whose value is greater than zero.

Definition 1.25

- 1 $\forall v$
- 2 $({}_0 \text{IsPosReal}(v) \text{ iff }$
- 3 $\text{IsReal}(v) \text{ and } v > 0.0_0)$

7 Formal specification of the document structures

Within the semiformal descriptions there is usually a reference to one or more clause numbers. Within this clause 7 these clause numbers refer to the respective clauses in ISO/IEC 8613-2, unless explicitly otherwise specified.

What follows is the outline of the formula which specifies the ODA Document Structures, according to the specifications in ISO/IEC 8613-2. The dots indicate formal text fragments which have been left out for the sake of readability.

The full formula can be obtained by replacing each line (apart from the and) with the definition which is referenced by the superscript of the predicate symbol or operator symbol, respectively. Thereby the variables used in the definition of the predicate have to be replaced by those appearing in the outline (if they are different). In general, mnemonic names for variables have been chosen. Formally this is insignificant, since a definition with, for example, the variable *slay* must work for any value of *slay*, not only for a correct specific layout description. That is, the definition yields True for a particular value of *slay*, if and only if this value is a correct specific layout description, otherwise the definition yields False. Note that a definition is a formula, hence it may never yield an undefined result, whatever value has been inserted for the variable.

-.sets of constituents: .-

... IsInterchangeSet^{2.1}(*doc*) ...
and ... IsGenericDocumentDescription^{2.2}(*gdoc*) ...
and ... IsProtectedDocumentDescription^{2.3}(*pdoc*) ...
and ... IsDocumentDescription^{2.4}(*doc*) ...
and ... IsProcessable^{2.5}(*doc*) ...
and ... IsFormattedProcessable^{2.6}(*doc*) ...
and ... IsFormatted^{2.7}(*doc*) ...
and ... IsDocumentBody^{2.8}(*doby*) ...
and ... IsGenericDescription^{2.9}(*g*) ...
and ... IsCompleteGenericDescription^{2.10}(*g*) ...
and ... IsGenericLogicalDescription^{2.11}(*glog*) ...
and ... IsGenericLayoutDescription^{2.12}(*glay*) ...
and ... IsSpecificDescription^{2.13}(*s*) ...
and ... IsSpecificLogicalDescription^{2.14}(*slog*) ...
and ... IsSpecificLayoutDescription^{2.15}(*slay*) ...
and ... IsLayoutStyleSet^{2.16}(*laystys*) ...
and ... IsPresentationStyleSet^{2.17}(*prestys*) ...
and ... IsProtectedPart^{2.18}(*ppart*) ...

..constituents: .-

and ... IsRootClassDescription^{2.19}(*cst*) ...
and ... IsLogicalRootClassDescription^{2.20}(*cst*) ...
and ... IsCompositeLogicalObjectClassDescription^{2.21}(*cst*) ...
and ... IsBasicLogicalObjectClassDescription^{2.22}(*cst*) ...
and ... IsLogicalObjectClassDescription^{2.23}(*cst*) ...
and ... IsLayoutRootClassDescription^{2.24}(*cst*) ...
and ... IsPageSetClassDescription^{2.25}(*cst*) ...
and ... IsCompositePageClassDescription^{2.26}(*cst*) ...
and ... IsBasicPageClassDescription^{2.27}(*cst*) ...
and ... IsFrameClassDescription^{2.28}(*cst*) ...
and ... IsBlockClassDescription^{2.29}(*cst*) ...
and ... IsLayoutObjectClassDescription^{2.30}(*cst*) ...
and ... IsCompositeObjectClassDescription^{2.31}(*cst*) ...
and ... IsCompositeLayoutObjectClassDescription^{2.32}(*cst*) ...
and ... IsBasicLayoutObjectClassDescription^{2.33}(*cst*) ...
and ... IsBasicObjectClassDescription^{2.34}(*cst*) ...

and ... IsObjectClassDescription^{2.35}(*cst*) ...
and ... IsRootDescription^{2.36}(*cst*) ...
and ... IsLogicalRootDescription^{2.37}(*cst*) ...
and ... IsCompositeLogicalObjectDescription^{2.38}(*cst*) ...
and ... IsBasicLogicalObjectDescription^{2.39}(*cst*) ...
and ... IsLogicalObjectDescription^{2.40}(*cst*) ...
and ... IsLayoutRootDescription^{2.41}(*cst*) ...
and ... IsPageSetDescription^{2.42}(*cst*) ...
and ... IsCompositePageDescription^{2.43}(*cst*) ...
and ... IsBasicPageDescription^{2.44}(*cst*) ...
and ... IsPageDescription^{2.45}(*cst*) ...
and ... IsFrameDescription^{2.46}(*cst*) ...
and ... IsBlockDescription^{2.47}(*cst*) ...
and ... IsLayoutObjectDescription^{2.48}(*cst*) ...
and ... IsCompositeObjectDescription^{2.49}(*cst*) ...
and ... IsBasicObjectDescription^{2.50}(*cst*) ...
and ... IsCompositeLayoutObjectDescription^{2.51}(*cst*) ...
and ... IsBasicLayoutObjectDescription^{2.52}(*cst*) ...
and ... IsObjectDescription^{2.53}(*cst*) ...
and ... IsContentPortionDescription^{2.54}(*cst*) ...
and ... IsLayoutStyle^{2.55}(*cst*) ...
and ... IsPresentationStyle^{2.56}(*cst*) ...
and ... IsSealedDocumentProfileDescription^{2.57}(*cst*) ...
and ... IsEncipheredDocumentProfileDescription^{2.58}(*cst*) ...
and ... IsPreEncipheredDocumentBodyPartDescription^{2.59}(*cst*) ...
and ... IsPostEncipheredDocumentBodyPartDescription^{2.60}(*cst*) ...

and ... IsAttributeSet^{2.61}(*as*) ...
and ... IsProfileAttributeSetPart2^{2.62}(*as*) ...

and ... IsBindingsValueExpression^{2.63}(*v*) ...
and ... IsStringExpression^{2.64}(*v*) ...
and ... IsAtomicStringExpression^{2.65}(*v*) ...
and ... IsStringFunction^{2.66}(*v*) ...
and ... IsNumericExpression^{2.67}(*v*) ...
and ... IsNumericFunction^{2.68}(*v*) ...
and ... IsObjectOrObjectClassIdExpression^{2.69}(*v*) ...
and ... IsObjectOrObjectClassSelectionFunction^{2.70}(*v*) ...
and ... IsBindingReferenceExpression^{2.71}(*v*) ...
and ... IsBindingReference^{2.72}(*v*) ...
and ... IsBindingName^{2.73}(*v*) ...
and ... IsBindingSelectionFunction^{2.74}(*v*) ...
and ... IsCurrentInstanceFunction^{2.75}(*v*) ...
and ... IsObjectTypeValue^{2.76}(*v*) ...
and ... IsLogicalObjectId^{2.77}(*v*) ...
and ... IsLayoutObjectId^{2.78}(*v*) ...
and ... IsObjectId^{2.79}(*v*) ...
and ... IsLogicalObjectClassId^{2.80}(*v*) ...
and ... IsLayoutObjectClassId^{2.81}(*v*) ...
and ... IsObjectClassId^{2.82}(*v*) ...
and ... IsSeqOfObjectClassId^{2.83}(*v*) ...
and ... IsConstructionExpression^{2.84}(*v*) ...
and ... IsConstructionType^{2.85}(*v*) ...
and ... IsConstructionTerm^{2.86}(*v*) ...

-attributes: -

and ... IsContentArchitectureClassValue^{2.87}(v) ...
and ... IsPresentationAttributesValue^{2.88}(v) ...
and ... IsBindingsValue^{2.89}(v) ...
and ... IsDefaultValueListsValue^{2.90}(v) ...
and ... IsPositionValue^{2.91}(v) ...
and ... IsDimensionsValue^{2.92}(v) ...
and ... IsBorderValue^{2.93}(v) ...
and ... IsBalanceValue^{2.94}(v) ...
and ... IsLayoutPathValue^{2.95}(v) ...
and ... IsTransparencyValue^{2.96}(v) ...
and ... IsColourValue^{2.97}(v) ...
and ... IsMediumTypeValue^{2.98}(v) ...
and ... IsProtectionValue^{2.99}(v) ...
and ... IsLayoutStyleId^{2.100}(v) ...
and ... IsBlockAlignmentValue^{2.101}(v) ...
and ... IsConcatenationValue^{2.102}(v) ...
and ... IsFillOrderValue^{2.103}(v) ...
and ... IsLayoutObjectReference^{2.104}(v) ...
and ... IsLayoutCatName^{2.105}(v) ...
and ... IsSetOfLayoutCatNames^{2.106}(v) ...
and ... IsOffsetValue^{2.107}(v) ...
and ... IsSameLayoutObjectValue^{2.108}(v) ...
and ... IsSeparationValue^{2.109}(v) ...
and ... IsPresentationStyleId^{2.110}(v) ...
and ... IsLayoutContentPortionId^{2.111}(v) ...
and ... IsLogicalContentPortionId^{2.112}(v) ...
and ... IsTypeOfCodingValue^{2.113}(v) ...
and ... IsContentInformationValue^{2.114}(v) ...
and ... IsAlternativeCharacterSetsString^{2.115}(v) ...
and ... IsCodingAttributesValue^{2.116}(v) ...
and ... IsCharSeq^{2.117}(v) ...
and ... IsCharacter^{2.118}(z) ...
and ... IsEncipheredValue^{2.119}(v) ...
and ... IsSealedValue^{2.120}(v) ...
and ... IsSealedDocumentProfileInformationValue^{2.121}(v) ...
and ... IsProtectedPartId^{2.122}(v) ...
and ... IsEncipheredBodyPartId^{2.123}(v) ...
and ... IsSetOfSealIds^{2.124}(v) ...
and ... IsSealId^{2.125}(v) ...
and ... IsColourExpression^{2.126}(v) ...
and ... IsIndexedColourExpression^{2.127}(v) ...
and ... IsDirectColourExpression^{2.128}(v) ...
and ... IsColourSpecificationValue^{2.129}(v) ...
and ... IsColourToleranceValue^{2.130}(v) ...
and ... IsColourTable^{2.131}(v) ...
and ... IsSetOfColourTableEntries^{2.132}(v) ...
and ... IsColourTableEntry^{2.133}(v) ...
and ... IsFloatabilityRangeValue^{2.134}(v) ...

and ... (s)IsDescribedBy^{2.135}(g) ...
and ... (s)IsCompletelyDescribedBy^{2.136}(g) ...

and ... IsContentConnected^{2.137}(csts) ...
and ... IsContentComplete^{2.138}(csts) ...
and ... IsClassImSubComplete^{2.139}(g) ...
and ... IsImSubComplete^{2.140}(s) ...

-subsidary definitions: -

$\underline{and} \dots (slog)IsContainedIn^{2.141}(slay) \dots$
 $\underline{and} \dots (id)IsClassIdImSubIn^{2.142}(gco) \dots$
 $\underline{and} \dots (id)IsIdImSubIn^{2.143}(sco) \dots$
 $\underline{and} \dots (id)IsIdContIn^{2.144}(b) \dots$
 $\underline{and} \dots (idq)IsSeqClassIdImSubIn^{2.145}(gco) \dots$
 $\underline{and} \dots (idq)CoveredBy^{2.146}(r) \dots$
 $\underline{and} \dots (id)OccursIn^{2.147}(m) \dots$
 $\underline{and} \dots (go)IsInitialGenericIn^{2.148}(gos) \dots$
 $\underline{and} \dots (so)IsInitialSpecificIn^{2.149}(sos) \dots$
 $\underline{and} \dots (lgo)DescribesClassImSubOf^{2.150}(hgo) \dots$
 $\underline{and} \dots (lgo)DescribesClassSubOf^{2.151}(hgo)In(g) \dots$
 $\underline{and} \dots (lso)DescribesImSubOf^{2.152}(hso) \dots$
 $\underline{and} \dots (cont)DescribesContPortOf^{2.153}(b)$

NOTE — Other predicates or operators which are used here, but are defined in clause 6, are not listed here.

In principle a given interchange set may be verified to conform to ISO/IEC 8613 by establishing

$\mathcal{AS} \text{ } \underline{impl} \text{ } IsInterchangeSet^{2.1}(-\text{this-term}-)$

where $\mathcal{AS}$ stands for the conjugation of all axioms and definitions and the argument of $IsInterchangeSet$ stands for a formal expression (a term) which denotes the particular interchange set in question. If the extended overall formula yields True, the document description conforms to ISO/IEC 8613, if False, it does not. An undefined result is impossible.

7.1 Sets of constituents

Semiinformal Description 2.1

Predicate “is an interchange set” (clause 6.3)

An entity *doc* is an interchange set if it is a generic document description, a document description or a protected document description.

Definition 2.1

- 1 $\forall doc$
- 2 $({}_o \text{IsInterchangeSet}(doc) \text{ iff}$
- 3 $\text{IsGenericDocumentDescription}^{2.2}(doc) \text{ or } \text{IsDocumentDescription}^{2.4}(doc)$
- 4 $\text{ or } \text{IsProtectedDocumentDescription}^{2.3}(doc)_o)$

Semiinformal Description 2.2

Predicate “is a generic document description” (clause 6.3.10)

A generic document description consists of a document profile and a document body (4-6). The document body consists of the union of four entities *glog*, *glay*, *laystys* and *prestys* (7). If the layout style set is non-empty, the generic logical structure is non-empty (8). Either a generic logical or a generic layout description (or both) must always be present (9). The document profile and the document body satisfy the constraints imposed by the content architectures (10-12).

Definition 2.2

- 1 $\forall gdoc$
- 2 $({}_o \text{IsGenericDocumentDescription}(gdoc) \text{ iff}$
- 3 $\exists prof, glog, glay, laystys, prestys, doby$
- 4 $({}_1 gdoc = [prof] \cup doby \text{ and}$
- 5 $\text{IsDocumentProfile}^{4.1}(prof) \text{ and}$
- 6 $\text{IsDocumentBody}^{2.8}(doby) \text{ and}$
- 7 $doby = glog \cup glay \cup laystys \cup prestys \text{ and}$
- 8 $({}_2 \text{IsNeCol}^{1.1}(laystys) \text{ impl } \text{IsNeCol}^{1.1}(glog) {}_2) \text{ and}$
- 9 $({}_3 \text{IsNeCol}^{1.1}(glog) \text{ or } \text{IsNeCol}^{1.1}(glay) {}_3) \text{ and}$
- 10 $\text{SatisfiesCharacterContentArchitectureConstraints}^{6.4}(prof, doby) \text{ and}$
- 11 $\text{SatisfiesRasterGraphicsContentArchitectureConstraints}^{7.4}(prof, doby) \text{ and}$
- 12 $\text{SatisfiesGeometricGraphicsContentArchitectureConstraints}^{8.4}(prof, doby)_o)$

Semiformal Description 2.3

Predicate “is a protected document description” (clause 6.6)

A protected document description *pdoc* contains a non-empty subset *ppart* which is a protected part.

NOTE — Only the existence of a (non-empty) protected part in a protected document description is formally specified. After deciphering the protected part and creating the objects contained in the protected part, the interchange set is either a generic document description or a document description.

Definition 2.3

- 1 $\forall pdoc$
- 2 $(\text{IsProtectedDocumentDescription}(pdoc) \text{ iff}$
- 3 $\exists upart, ppart$
- 4 $(\text{pdoc} = upart \cup ppart \text{ and}$
- 5 $\text{IsNeCol}^{1.1}(ppart) \text{ and } \text{IsProtectedPart}^{2.18}(ppart)_1)_0$

Semiformal Description 2.4

Predicate “is a document description” (clause 6.3)

An entity *doc* is a document description if it is a document profile *prof* (5) or if it is an entity which is processable, formatted processable or formatted (5–6), according to the value of the document profile attribute 'document architecture class' (7–13), with a 'resource document' specified in the profile if any 'resource' is specified in the document (14–15).

Definition 2.4

- 1 $\forall doc$
- 2 $(\text{IsDocumentDescription}(doc) \text{ iff}$
- 3 $\exists prof$
- 4 $(\text{IsDocumentProfile}^{4.1}(prof) \text{ and}$
- 5 $(\text{doc} = [prof] \text{ or } \text{IsProcessable}^{2.5}(doc) \text{ or}$
- 6 $\text{IsFormattedProcessable}^{2.6}(doc) \text{ or } \text{IsFormatted}^{2.7}(doc)_2) \text{ and}$
- 7 $doc \neq [prof] \text{ impl}$
- 8 $(\text{C } \neg prof \bullet \text{'document architecture class'} = \text{'processable'} \text{ iff}$
- 9 $\text{IsProcessable}^{2.5}(doc)_4) \text{ and}$
- 10 $(\text{C } \neg prof \bullet \text{'document architecture class'} = \text{'formatted processable'} \text{ iff}$
- 11 $\text{IsFormattedProcessable}^{2.6}(doc)_5) \text{ and}$
- 12 $(\text{C } \neg prof \bullet \text{'document architecture class'} = \text{'formatted'} \text{ iff}$
- 13 $\text{IsFormatted}^{2.7}(doc)_6) \text{ and}$
- 14 $\forall cst \in doc$
- 15 $(\text{'resource'} \in \text{NAMS}^{1.18}(cst) \text{ impl } \text{'resource document'} \in \text{NAMS}^{1.18}(prof)_7)_1)_0$

Semiinformal Description 2.5

Predicate "is processable" (clauses 6.3.9, 6.3.13)

An entity *doc* is processable if it is a collection of constituents consisting of a document profile *prof* along with a document body *doby* (4,5) which includes (possibly empty) constituent sets *glog*, *slog*, *glay*, *laystys* and *prestys* (6). Attributes in the document profile indicate the presence of a specific logical description (which must always be present) and of (optionally) a generic logical description, generic layout description, layout style set and presentation style set (7-15). If the value of the profile attribute 'generic logical structure' is 'complete generator set', the specific logical description *slog* is completely described by the generic logical description *glog* (16,17). If the profile attribute 'generic layout structure' is specified, its value is 'complete generator set' and the generic layout description *glay* is a complete generic description (18-20). The profile attribute 'specific layout structure' is not specified (21). The profile attribute 'document architecture class' has the value 'processable' (22). The document profile and the document body satisfy the constraints imposed by the content architectures (23-25).

Definition 2.5

```

1  ∀ doc
2  (₀ IsProcessable(doc) iff
3    ∃ prof, doby, glog, slog, glay, laystys, prestys
4    (₁ doc = [prof] ∪ doby and
5      IsDocumentProfile4.1(prof) and IsDocumentBody2.8(doby) and
6      doby = glog ∪ slog ∪ glay ∪ laystys ∪ prestys and
7      'specific logical structure' ∈ NAMS1.18(prof) and IsSpecificLogicalDescription2.14(slog) and
8      (₂ 'generic logical structure' ∈ NAMS1.18(prof) iff
9        (₃ IsNeCol1.1(glog) and IsGenericLogicalDescription2.11(glog)₃)₂) and
10     (₄ 'generic layout structure' ∈ NAMS1.18(prof) iff
11       (₅ IsNeCol1.1(glay) and IsGenericLayoutDescription2.12(glay)₅)₄) and
12     (₆ 'layout styles' ∈ NAMS1.18(prof) iff
13       (₇ IsNeCol1.1(laystys) and IsLayoutStyleSet2.16(laystys)₇)₆) and
14     (₈ 'presentation styles' ∈ NAMS1.18(prof) iff
15       (₉ IsNeCol1.1(prestys) and IsPresentationStyleSet2.17(prestys)₉)₈) and
16     (₁₀ C ^prof • 'generic logical structure' = 'complete generator set' impl
17       (slog)IsCompletelyDescribedBy2.136(glog)₁₀) and
18     (₁₁ 'generic layout structure' ∈ NAMS1.18(prof) impl
19       (₁₂ C ^prof • 'generic layout structure' = 'complete generator set' and
20         IsCompleteGenericDescription2.10(glay)₁₂)₁₁) and
21     'specific layout structure' ∉ NAMS1.18(prof) and
22     C ^prof • 'document architecture class' = 'processable' and
23     SatisfiesCharacterContentArchitectureConstraints6.4(prof, doby) and
24     SatisfiesRasterGraphicsContentArchitectureConstraints7.4(prof, doby) and
25     SatisfiesGeometricGraphicsContentArchitectureConstraints8.4(prof, doby)₁)₀)

```

Semiformal Description 2.6

Predicate "is formatted processable" (clauses 6.3.9, 6.3.13)

An entity *doc* is formatted processable if it is a collection of constituents consisting of a document profile *prof* along with a document body *doby* (4,5) which includes (possibly empty) constituent sets *glog*, *slog*, *glay*, *slay*, *laystys* and *prestys* (6). Attributes in the document profile indicate the presence of a specific logical description, the specific layout description, the generic layout description (these three sets are always present) and of (optionally) a generic logical description, layout style set and presentation style set (7-15). If the value of the profile attribute 'generic logical structure' is 'complete generator set', the specific logical description *slog* is completely described by the generic logical description *glog* (16,17). If the value of the profile attribute 'generic layout structure' is 'complete generator set', the specific layout description *slay* is completely described by the generic layout description *glay* (18,19). The specific logical description is contained in the specific layout description (20). The profile attribute 'document architecture class' has the value 'formatted processable' (21). The document profile and the document body satisfy the constraints imposed by the content architectures (22-24).

Definition 2.6

```

1  ∀ doc
2  (₀ IsFormattedProcessable(doc) iff
3    ∃ prof, doby, glog, slog, glay, slay, laystys, prestys
4    (₁ doc = [prof] ∪ doby and
5      IsDocumentProfile4.1(prof) and IsDocumentBody2.8(doby) and
6      doby = glog ∪ slog ∪ glay ∪ slay ∪ laystys ∪ prestys and
7      'specific logical structure' ∈ NAMS1.18(prof) and IsSpecificLogicalDescription2.14(slog) and
8      'specific layout structure' ∈ NAMS1.18(prof) and IsSpecificLayoutDescription2.15(slay) and
9      'generic layout structure' ∈ NAMS1.18(prof) and IsGenericLayoutDescription2.12(glay) and
10     (₂ 'generic logical structure' ∈ NAMS1.18(prof) iff
11       (₃ IsNeCol1.1(glog) and IsGenericLogicalDescription2.11(glog)₃)₂) and
12     (₄ 'layout styles' ∈ NAMS1.18(prof) iff
13       (₅ IsNeCol1.1(laystys) and IsLayoutStyleSet2.16(laystys)₅)₄) and
14     (₆ 'presentation styles' ∈ NAMS1.18(prof) iff
15       (₇ IsNeCol1.1(prestys) and IsPresentationStyleSet2.17(prestys)₇)₆) and
16     (₈ C ~prof • 'generic logical structure' = 'complete generator set' impl
17       (slog)IsCompletelyDescribedBy2.136(glog)₈) and
18     (₉ C ~prof • 'generic layout structure' = 'complete generator set' impl
19       (slay)IsCompletelyDescribedBy2.136(glay)₉) and
20     (slog)IsContainedIn2.141(slay) and
21     C ~prof • 'document architecture class' = 'formatted processable' and
22     SatisfiesCharacterContentArchitectureConstraints6.4(prof, doby) and
23     SatisfiesRasterGraphicsContentArchitectureConstraints7.4(prof, doby) and
24     SatisfiesGeometricGraphicsContentArchitectureConstraints8.4(prof, doby)₁)₀)

```

Semiformal Description 2.7

Predicate “is formatted” (clauses 6.3.9, 6.3.13)

An entity *doc* is formatted if it is a collection of constituents consisting of a document profile *prof* along with a document body *doby* (4,5) which includes (possibly empty) constituent sets *slay*, *glay* and *prestys* (6). Attributes in the document profile indicate the presence of a specific layout description (which must always be present) and of (optionally) a generic layout description and presentation style set (7-11). The value of the profile attribute ‘generic layout structure’, if specified, is ‘partial generator set’ or ‘factor set’ (12,13). The profile attributes ‘specific logical structure’, ‘generic logical structure’ and ‘layout styles’ are not specified (14-16). The profile attribute ‘document architecture class’ has the value ‘formatted’ (17). The document profile and the document body satisfy the constraints imposed by the content architectures (18-20).

Definition 2.7

```

1  ∀ doc
2  (₀ IsFormatted(doc) iff
3    ∃ prof, doby, glay, slay, prestys
4    (₁ doc = [prof] ∪ doby and
5      IsDocumentProfile4.1(prof) and IsDocumentBody2.8(doby) and
6      doby = glay ∪ slay ∪ prestys and
7      ‘specific layout structure’ ∈ NAMS1.18(prof) and IsSpecificLayoutDescription2.15(slay) and
8      (₂ ‘generic layout structure’ ∈ NAMS1.18(prof) iff
9        (₃ IsNeCol1.1(glay) and IsGenericLayoutDescription2.12(glay)₃)₂) and
10     (₄ ‘presentation styles’ ∈ NAMS1.18(prof) iff
11       (₅ IsNeCol1.1(prestys) and IsPresentationStyleSet2.17(prestys)₅)₄) and
12     (₆ ‘generic layout structure’ ∈ NAMS1.18(prof) impl
13       C ~ prof . ‘generic layout structure’ ∈ [‘partial generator set’; ‘factor set’]₆) and
14     ‘specific logical structure’ ∉ NAMS1.18(prof) and
15     ‘generic logical structure’ ∉ NAMS1.18(prof) and
16     ‘layout styles’ ∉ NAMS1.18(prof) and
17     C ~ prof . ‘document architecture class’ = ‘formatted’ and
18     SatisfiesCharacterContentArchitectureConstraints6.4(prof, doby) and
19     SatisfiesRasterGraphicsContentArchitectureConstraints7.4(prof, doby) and
20     SatisfiesGeometricGraphicsContentArchitectureConstraints8.4(prof, doby)₁)₀)

```

Semiformal Description 2.8

Predicate “is a document body” (clause 6.3.14)

An entity *doby* is a document body if it is a collection of (possibly empty) constituents *glog*, *slog*, *glay*, *slay*, *laystys* and *prestys* (3,4). If *glog* is not empty it is a generic logical description (5). If *slog* is not empty it is a specific logical description (6). If *glay* is not empty it is a generic layout description (7). If *slay* is not empty it is a specific layout description (8). If *laystys* is not empty it is a layout style set (9). If *prestys* is not empty it is a presentation style set (10). If a generic description and a corresponding specific description are present in a document body, the specific one is described by the generic one (11, 12), i.e. the specific description is an instance of the class described by the generic description. All references to layout object classes from attributes of layout styles are satisfied (17-19, 24-27), i.e., each referenced layout object class exists in the generic layout structure. All references to layout categories from attributes of layout styles are satisfied (17, 20-22, 28, 29), i.e., each referenced layout category is specified by some layout style. All references to layout styles from attributes of layout styles are satisfied (22, 23), i.e., each referenced layout style exists in the layout style set. All references from *glay* to *glog* by means of ‘logical source’ are satisfied (31, 32). All objects or object classes which are referenced by the attribute ‘balance’ from the specific or generic layout structure, respectively, are present (33-38). All references to layout categories by means of the attribute ‘permitted categories’ are satisfied (39-41). If the attributes ‘imaging order’ and ‘subordinates’ are both specified their values must specify the same set of constituents (42-43).

Definition 2.8

$\forall doby$
 $(_0 \text{ IsDocumentBody}(doby) \text{ iff}$
 $\exists glog, slog, glay, slay, laystys, prestys$
 $(_1 doby = glog \cup slog \cup glay \cup slay \cup laystys \cup prestys \text{ and}$
 $(_2 \text{ IsNeCol}^{1.1}(glog) \text{ impl IsGenericLogicalDescription}^{2.11}(glog)_2) \text{ and}$
 $(_3 \text{ IsNeCol}^{1.1}(slog) \text{ impl IsSpecificLogicalDescription}^{2.14}(slog)_3) \text{ and}$
 $(_4 \text{ IsNeCol}^{1.1}(glay) \text{ impl IsGenericLayoutDescription}^{2.12}(glay)_4) \text{ and}$
 $(_5 \text{ IsNeCol}^{1.1}(slay) \text{ impl IsSpecificLayoutDescription}^{2.15}(slay)_5) \text{ and}$
 $(_6 \text{ IsNeCol}^{1.1}(laystys) \text{ impl IsLayoutStyleSet}^{2.16}(laystys)_6) \text{ and}$
 $(_7 \text{ IsNeCol}^{1.1}(prestys) \text{ impl IsPresentationStyleSet}^{2.17}(prestys)_7) \text{ and}$
 $(_8 (_9 \text{ IsNeCol}^{1.1}(slog) \text{ and IsNeCol}^{1.1}(glog)_9) \text{ impl } (slog) \text{ IsDescribedBy}^{2.135}(glog)_8) \text{ and}$
 $(_{10} (_{11} \text{ IsNeCol}^{1.1}(slay) \text{ and IsNeCol}^{1.1}(glay)_{11}) \text{ impl } (slay) \text{ IsDescribedBy}^{2.135}(glay)_{10}) \text{ and}$
 $\forall cst \in (glog \cup slog) (_{12} \forall x \in \hat{cst} \cdot (N x = \text{'layout style'}) \text{ impl}$
 $(_{13} \exists ls \in laystys (C \hat{ls} \cdot \text{'layout style identifier'} = C x)_{13})_{12}) \text{ and}$
 $\forall cst \in (glog \cup slog \cup glay \cup slay) (_{14} \forall x \in \hat{cst} \cdot (N x = \text{'presentation style'}) \text{ impl}$
 $(_{15} \exists ps \in prestys (C \hat{ps} \cdot \text{'presentation style identifier'} = C x)_{15})_{14}) \text{ and}$
 $\forall cst \in laystys (_{16} \forall y \in \hat{cst} \cdot$
 $(_{17} (_{18} \text{ IsLayoutObjectClassId}^{2.81}(C y) \text{ impl}$
 $(_{19} \exists go \in glay (C \hat{go} \cdot \text{'object class identifier'} = C y)_{19})_{18}) \text{ and}$
 $(_{20} \text{ IsLayoutCatName}^{2.105}(C y) \text{ impl}$
 $(_{21} \exists ls \in laystys (C \hat{ls} \cdot \text{'layout category'} = C y)_{21})_{20}) \text{ and}$
 $(_{22} \text{ IsLayoutStyleId}^{2.100}(C y) \text{ impl}$
 $(_{23} \exists ls \in laystys (C \hat{ls} \cdot \text{'layout style identifier'} = C y)_{23})_{22}) \text{ and}$
 $\forall l, r$
 $(_{24} (_{25} N y = \text{'same layout object'} \text{ and}$
 $C y = [\text{'logical object'} : l; \text{'layout object'} : r] \text{ and IsLayoutObjectClassId}^{2.81}(r)_{26}) \text{ impl}$
 $(_{27} \exists go \in glay (C \hat{go} \cdot \text{'object class identifier'} = r)_{27})_{25}) \text{ and}$
 $(_{28} (_{29} N y = \text{'same layout object'} \text{ and}$
 $C y = [\text{'logical object'} : l; \text{'layout object'} : r] \text{ and IsLayoutCatName}^{2.105}(r)_{29}) \text{ impl}$
 $(_{30} \exists ls \in laystys (C \hat{ls} \cdot \text{'layout category'} = r)_{30})_{28})_{24})_{17})_{16}) \text{ and}$
 $\forall a \in \hat{glay} \cdot \cdot \text{'logical source'}$
 $(_{31} \exists go \in glog (C \hat{go} \cdot \text{'object class identifier'} = C a)_{31}) \text{ and}$
 $\forall a \in \hat{slay} \cdot \cdot \text{'balance'}$
 $(_{32} C a \neq \text{'null'} \text{ impl}$
 $\forall b \in \hat{(C a)} \cdot (_{33} \exists so \in slay (C \hat{so} \cdot \text{'object identifier'} = b)_{33})_{32}) \text{ and}$
 $\forall a \in \hat{glay} \cdot \cdot \text{'balance'}$
 $(_{34} C a \neq \text{'null'} \text{ impl}$
 $\forall b \in \hat{(C a)} \cdot (_{35} \exists go \in glay (C \hat{go} \cdot \text{'object class identifier'} = b)_{35})_{34}) \text{ and}$
 $\forall a \in C \hat{slay} \cdot \cdot \text{'permitted categories'}$
 $(_{36} a \neq \text{'null'} \text{ impl}$
 $\exists ls \in laystys (_{37} C \hat{ls} \cdot \text{'layout category'} = a_{37})_{36}) \text{ and}$
 $\forall cst \in slay$
 $(_{38} ([\text{'imaging order'}; \text{'subordinates'}] \subset \text{NAMS}^{1.18}(cst)) \text{ impl}$
 $[x \mid x \in \hat{cst} \cdot \text{'imaging order'}] = [y \mid y \in \hat{cst} \cdot \text{'subordinates'}]_{38})_{10})$

Semiformal Description 2.9

Predicate “is a generic description” (clause 6.10)

An entity g is a generic description, if it is a set of object class descriptions (5,6) and optional content portion descriptions (10,11). It is content connected and is complete with respect to the content portion descriptions and the object class descriptions (13-15). A generic description has not more than one root class description (12). The values of the attributes 'object class identifier' are unique within the set g (7-9).

Definition 2.9

```

1   $\forall g$ 
2  ( $_0$  IsGenericDescription( $g$ ) iff
3     $\exists ocs, conts$ 
4    ( $_1 g = ocs \cup conts$  and
5      IsNeCol1.1( $ocs$ ) and
6       $\forall cst \in ocs$  (IsObjectClassDescription2.35( $cst$ )) and
7       $\forall x, y \in ocs$ 
8      ( $_2 x \neq y$  impl
9         $C \sim x \cdot \text{'object class identifier'} \neq C \sim y \cdot \text{'object class identifier'}$ )  $_2$ ) and
10     IsCol( $conts$ ) and
11      $\forall cst \in conts$  (IsContentPortionDescription2.54( $cst$ ))  $_1$ ) and
12     CARD [ $cst | cst \in g$  and IsRootClassDescription2.19( $cst$ )]  $\leq 1$  and
13     IsContentConnected2.137( $g$ ) and
14     IsContentComplete2.138( $g$ ) and
15     IsClassImSubComplete2.139( $g$ ) $_0$ )

```

Semiformal Description 2.10

Predicate “is a complete generic description” (clauses 6.3.4, 7.5)

An entity g is a complete generic description if it is a generic description with one or more initial generic constituents (object class descriptions) igs (5). Exactly one of them (r) is a logical or layout root class description (6,8) and refers to at least one subordinate object class description (7). In addition, every non-initial object class description is directly or indirectly referred to from exactly one initial generic constituent (9-13). Thus, a complete generic description is a set of connected, directed, single-rooted graphs (via the attribute 'generator for subordinates').

Definition 2.10

```

1   $\forall g$ 
2  ( $_0$  IsCompleteGenericDescription( $g$ ) iff
3    IsGenericDescription2.9( $g$ ) and
4     $\exists igs, r$ 
5    ( $_1 igs = [cst | (cst) \text{IsInitialGenericIn}^{2.148}(g)]$  and  $r \in igs$  and
6      IsRootClassDescription2.19( $r$ ) and
7      'generator for subordinates'  $\in$  NAMS1.18( $r$ ) and
8      not  $\exists ig \in igs$  ( $ig \neq r$  and IsRootClassDescription2.19( $ig$ )) and
9       $\forall cst \in g$ 
10     ( $_2$  not ( $cst \in igs$  or IsContentPortionDescription2.54( $cst$ )) impl
11        $\exists ig \in igs$ 
12       ( $_3 (cst) \text{DescribesClassSubOf}^{2.151}(ig) \text{In}(g)$ ) and
13        $\forall h \in igs$  ( $_4 h = ig$  or not ( $cst$ ) DescribesClassSubOf}^{2.151}(h) \text{In}(g)  $_4$ )  $_3$ )  $_2$ ) $_0$ )

```

Semiformal Description 2.11

Predicate “is a generic logical description” (clauses 6.3.4, 7.5)

An entity *glog* is a generic logical description if it is a generic description containing only logical object class descriptions and content portion descriptions (4-6). For a content portion description the attribute 'content identifier logical' is present and its value is a generic logical identifier.

Definition 2.11

```

1   $\forall glog$ 
2   $(_0 \text{ IsGenericLogicalDescription}(glog) \text{ iff}$ 
3     $\text{ IsGenericDescription}^{2.9}(glog) \text{ and}$ 
4     $\forall cst \in glog$ 
5     $(_1 \text{ IsLogicalObjectClassDescription}^{2.23}(cst) \text{ or}$ 
6     $(_2 \text{ IsContentPortionDescription}^{2.54}(cst) \text{ and}$ 
7       $\text{'content identifier logical'} \in \text{NAMS}^{1.18}(cst) \text{ and}$ 
8       $\text{IsLogicalObjectClassId}^{2.80}(C \hat{=} cst \bullet \text{'content identifier logical'})_2)_1)_0$ 

```

Semiformal Description 2.12

Predicate “is a generic layout description” (clause 7.5.7)

An entity *glay* is a generic layout description if it is a generic description containing only layout object class descriptions and content portion descriptions (4-6). For a content portion description the attribute 'content identifier layout' is present and its value is a generic layout identifier.

Definition 2.12

```

1   $\forall glay$ 
2   $(_0 \text{ IsGenericLayoutDescription}(glay) \text{ iff}$ 
3     $\text{ IsGenericDescription}^{2.9}(glay) \text{ and}$ 
4     $\forall cst \in glay$ 
5     $(_1 \text{ IsLayoutObjectClassDescription}^{2.30}(cst) \text{ or}$ 
6     $(_2 \text{ IsContentPortionDescription}^{2.54}(cst) \text{ and}$ 
7       $\text{'content identifier layout'} \in \text{NAMS}^{1.18}(cst) \text{ and}$ 
8       $\text{IsLayoutObjectClassId}^{2.81}(C \hat{=} cst \bullet \text{'content identifier layout'})_2)_1)_0$ 

```

Semiformal Description 2.13

Predicate “is a specific description” (clause 7.1)

An entity s is a specific description if it is a set of object descriptions (5,6) and optional content portion descriptions (7,8). It is content connected and is complete with respect to the content portion descriptions and the object descriptions (10-12). A specific description has exactly one root description (9) which is the only initial specific constituent (object description) in s (13-15). Thus the object description r is the root of a tree which is established by the attributes 'subordinates' and 'content portions' and contains all constituents of s as nodes.

Definition 2.13

- 1 $\forall s$
- 2 $({}_0 \text{IsSpecificDescription}(s) \text{ iff}$
- 3 $\exists os, conts$
- 4 $({}_1 s = os \cup conts \text{ and}$
- 5 $\text{IsNeCol}^{1.1}(os) \text{ and}$
- 6 $\forall cst \in os (\text{IsObjectDescription}^{2.53}(cst)) \text{ and}$
- 7 $\text{IsCol}(conts) \text{ and}$
- 8 $\forall cst \in conts (\text{IsContentPortionDescription}^{2.54}(cst))_1) \text{ and}$
- 9 $\text{CARD} [cst | cst \in s \text{ and } \text{IsRootDescription}^{2.36}(cst)] = 1 \text{ and}$
- 10 $\text{IsContentConnected}^{2.137}(s) \text{ and}$
- 11 $\text{IsContentComplete}^{2.138}(s) \text{ and}$
- 12 $\text{IsImSubComplete}^{2.140}(s) \text{ and}$
- 13 $\exists r$
- 14 $({}_2 [r] = [cst | (cst) \text{IsInitialSpecificIn}^{2.149}(s)] \text{ and}$
- 15 $\text{IsRootDescription}^{2.36}(r)_2)_0)$

Semiformal Description 2.14

Predicate “is a specific logical description” (clauses 7.2)

An entity $slog$ is a specific logical description if it is a specific description containing only logical object descriptions and content portion descriptions (4-6). For a content portion description the attribute 'content identifier logical' is present and its value is a specific logical identifier.

Definition 2.14

- 1 $\forall slog$
- 2 $({}_0 \text{IsSpecificLogicalDescription}(slog) \text{ iff}$
- 3 $\text{IsSpecificDescription}^{2.13}(slog) \text{ and}$
- 4 $\forall cst \in slog$
- 5 $({}_1 \text{IsLogicalObjectDescription}^{2.40}(cst) \text{ or}$
- 6 $({}_2 \text{IsContentPortionDescription}^{2.54}(cst) \text{ and}$
- 7 $\text{'content identifier logical'} \in \text{NAMS}^{1.18}(cst) \text{ and}$
- 8 $\text{IsLogicalObjectId}^{2.77}(C \wedge cst \bullet \text{'content identifier logical'})_2)_1)_0)$

Semiformal Description 2.15

Predicate “is a specific layout description” (clauses 7.3)

An entity *slay* is a specific layout description if it is a specific description containing only layout object descriptions and content portion descriptions (4-6). For a content portion description the attribute 'content identifier layout' is present and its value is a specific layout identifier (7-8). The immediate subordinates of the layout root or a page set are page sets and/or pages (9-11). The immediate subordinates of a composite page are frames and/or blocks (12-15). The immediate subordinates of a frame are frames and/or blocks (16-19). If blocks are immediate subordinates of a constituent, the attribute 'balance' must not be specified for this constituent (15, 19), otherwise it must be present or it is defaulted.

Definition 2.15

$$\begin{aligned}
 &1 \quad \forall \textit{slay} \\
 &2 \quad ({}_0 \text{IsSpecificLayoutDescription}(\textit{slay}) \textit{ iff} \\
 &3 \quad \text{IsSpecificDescription}^{2.13}(\textit{slay}) \textit{ and} \\
 &4 \quad \forall \textit{cst} \in \textit{slay} \\
 &5 \quad ({}_1 ({}_2 \text{IsLayoutObjectDescription}^{2.48}(\textit{cst}) \textit{ or} \\
 &6 \quad ({}_3 \text{IsContentPortionDescription}^{2.54}(\textit{cst}) \textit{ and} \\
 &7 \quad \text{'content identifier layout'} \in \text{NAMS}^{1.18}(\textit{cst}) \textit{ and} \\
 &8 \quad \text{IsLayoutObjectId}^{2.78}(\textit{C} \hat{\sim} \textit{cst} \bullet \text{'content identifier layout'}) \textit{ }_3)_2) \textit{ and} \\
 &9 \quad ({}_4 ({}_5 \text{IsLayoutRootDescription}^{2.41}(\textit{cst}) \textit{ or } \text{IsPageSetDescription}^{2.42}(\textit{cst}) \textit{ }_5) \textit{ }_4) \textit{ impl} \\
 &10 \quad \forall \textit{so} \in \text{IMSUB}^{2.154}(\textit{cst}, \textit{slay}) \\
 &11 \quad (\text{IsPageSetDescription}^{2.42}(\textit{so}) \textit{ or } \text{IsPageDescription}^{2.45}(\textit{so})) \textit{ }_4) \textit{ and} \\
 &12 \quad ({}_6 \text{IsCompositePageDescription}^{2.43}(\textit{cst}) \textit{ impl} \\
 &13 \quad \forall \textit{so} \in \text{IMSUB}^{2.154}(\textit{cst}, \textit{slay}) \\
 &14 \quad ({}_7 (\text{IsFrameDescription}^{2.46}(\textit{so}) \textit{ and } \text{'balance'} \in \text{NAMS}^{1.18}(\textit{cst})) \textit{ or} \\
 &15 \quad (\text{IsBlockDescription}^{2.47}(\textit{so}) \textit{ and } \text{'balance'} \notin \text{NAMS}^{1.18}(\textit{cst})) \textit{ }_7)_6) \textit{ and} \\
 &16 \quad ({}_8 \text{IsFrameDescription}^{2.46}(\textit{cst}) \textit{ impl} \\
 &17 \quad \forall \textit{so} \in \text{IMSUB}^{2.154}(\textit{cst}, \textit{slay}) \\
 &18 \quad ({}_9 (\text{IsFrameDescription}^{2.46}(\textit{so}) \textit{ and } \text{'balance'} \in \text{NAMS}^{1.18}(\textit{cst})) \textit{ or} \\
 &19 \quad (\text{IsBlockDescription}^{2.47}(\textit{so}) \textit{ and } \text{'balance'} \notin \text{NAMS}^{1.18}(\textit{cst})) \textit{ }_9)_8)_1)_0)
 \end{aligned}$$

Semiformal Description 2.16

Predicate “is a layout style set” (clause 6.3.5)

An entity *laystys* is a layout style set if it is a non-empty collection of layout styles (3,4) whose identifiers are unique within that collection (5-7).

Definition 2.16

$$\begin{aligned}
 &1 \quad \forall \textit{laystys} \\
 &2 \quad ({}_0 \text{IsLayoutStyleSet}(\textit{laystys}) \textit{ iff} \\
 &3 \quad \text{IsNeCol}^{1.1}(\textit{laystys}) \textit{ and} \\
 &4 \quad \forall \textit{cst} \in \textit{laystys} (\text{IsLayoutStyle}^{2.55}(\textit{cst})) \textit{ and} \\
 &5 \quad \forall \textit{x}, \textit{y} \in \textit{laystys} \\
 &6 \quad ({}_1 \textit{x} \neq \textit{y} \textit{ impl} \\
 &7 \quad \textit{C} \hat{\sim} \textit{x} \bullet \text{'layout style identifier'} \neq \textit{C} \hat{\sim} \textit{y} \bullet \text{'layout style identifier'} \textit{ }_1)_0)
 \end{aligned}$$

Semiformal Description 2.17

Predicate “is a presentation style set” (clause 6.3.5)

An entity *prestys* is a presentation style set if it is a non-empty collection of presentation styles (3,4) whose identifiers are unique within that collection (5-7).

Definition 2.17

- 1 $\forall \text{prestys}$
- 2 $(\text{IsPresentationStyleSet}(\text{prestys}) \text{ iff}$
- 3 $\text{IsNeCol}^{1.1}(\text{prestys}) \text{ and}$
- 4 $\forall \text{cst} \in \text{prestys} (\text{IsPresentationStyle}^{2.56}(\text{cst})) \text{ and}$
- 5 $\forall x, y \in \text{prestys}$
- 6 $(x \neq y \text{ impl}$
- 7 $C \sim x \cdot \text{'presentation style identifier'} \neq C \sim y \cdot \text{'presentation style identifier'}_1)_0)$

7.2 Constituents

In constituents, the values of defaultable attributes need not be specified explicitly — as opposed to mandatory or non-mandatory attributes. In these cases the formal specification provides a placeholder for the value which is either specified elsewhere in the document description or specified in the International Standard as the default value (5.1.2.4). Such a “dummy value” is formally defined as satisfying the predicate *IsPlaceholder*^{1.19}.

Semiformal Description 2.18

Predicate “is a protected part” (clause 6.6)

A protected part consists of sealed document profile descriptions, enciphered document profile descriptions, pre-enciphered document body part descriptions and post-enciphered document body part descriptions.

Definition 2.18

- 1 $\forall \text{ppart}$
- 2 $(\text{IsProtectedPart}(\text{ppart}) \text{ iff}$
- 3 $\forall \text{cst} \in \text{ppart}$
- 4 $(\text{IsSealedDocumentProfileDescription}^{2.57}(\text{cst}) \text{ or}$
- 5 $\text{IsEncipheredDocumentProfileDescription}^{2.58}(\text{cst}) \text{ or}$
- 6 $\text{IsPreEncipheredDocumentBodyPartDescription}^{2.59}(\text{cst}) \text{ or}$
- 7 $\text{IsPostEncipheredDocumentBodyPartDescription}^{2.60}(\text{cst})_1)_0)$

Semiformal Description 2.19

Predicate “is a root class description” (clauses 7.5.6, 7.5.7)

A root class description is either a logical root class description or a layout root class description.

Definition 2.19

- 1 $\forall \text{cst}$
- 2 $(\text{IsRootClassDescription}(\text{cst}) \text{ iff}$
- 3 $\text{IsLogicalRootClassDescription}^{2.20}(\text{cst}) \text{ or } \text{IsLayoutRootClassDescription}^{2.24}(\text{cst})_0)$

Semiformal Description 2.20

Predicate “is a logical root class description” (clause 7.5.6)

A logical root class description is an attribute set (nomination) which may contain the attributes 'application comments', 'bindings', 'default value lists', 'generator for subordinates', 'layout style', 'object class identifier', 'object type', 'protection', 'resource', 'sealed', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have 2 and 'document logical root' as their respective values (13-14). None of the attribute values is defaultable (12).

Definition 2.20

```

1   $\forall cst$ 
2  ( $\text{IsLogicalRootClassDescription}(cst)$  iff
3     $\text{IsAttributeSet}^{2.61}(cst)$  and
4     $\text{NAMS}^{1.18}(cst) \supseteq$  ['object class identifier'; 'object type'] and
5     $\text{NAMS}^{1.18}(cst) \subseteq$  ['application comments'; 'bindings';
6      'default value lists'; 'generator for subordinates';
7      'layout style'; 'object class identifier';
8      'object type'; 'protection';
9      'resource'; 'sealed';
10     'user-readable comments'; 'user-visible name'] and
11    $\forall a \in \sim cst.$ 
12   ( $\text{not IsPlaceholder}^{1.19}(C a)$  and
13     ( $\text{N } a = \text{'object class identifier'}$  impl  $C a = [ \rightarrow 2 \rightarrow ]$  ) and
14     ( $\text{N } a = \text{'object type'}$  impl  $C a = \text{'document logical root'}$  ) ) and
15   not ( $cst$ )DescribesClassImSubOf2.150( $cst$ )o)
```

Semiformal Description 2.21

Predicate “is a composite logical object class description” (clause 7.5.6)

A composite logical object class description is a logical root class description or an attribute set (nomination) which may contain the attributes 'application comments', 'bindings', 'default value lists', 'generator for subordinates', 'layout style', 'object class identifier', 'object type', 'protection', 'resource', 'sealed', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a logical object class identifier and a 'composite logical object' as their respective values (14-15). None of the attribute values is defaultable (13).

Definition 2.21

```

1   $\forall cst$ 
2  ( $\text{IsCompositeLogicalObjectClassDescription}(cst)$  iff
3     $\text{IsLogicalRootClassDescription}^{2.20}(cst)$  or
4    ( $\text{IsAttributeSet}^{2.61}(cst)$  and
5       $\text{NAMS}^{1.18}(cst) \supseteq$  ['object class identifier'; 'object type'] and
6       $\text{NAMS}^{1.18}(cst) \subseteq$  ['application comments'; 'bindings';
7        'default value lists'; 'generator for subordinates';
8        'layout style'; 'object class identifier';
9        'object type'; 'protection';
10       'resource'; 'sealed';
11       'user-readable comments'; 'user-visible name'] and
12      $\forall a \in \sim cst.$ 
13     ( $\text{not IsPlaceholder}^{1.19}(C a)$  and
14       ( $\text{N } a = \text{'object class identifier'}$  impl  $\text{IsLogicalObjectClassId}^{2.80}(C a)$  ) and
15       ( $\text{N } a = \text{'object type'}$  impl  $C a = \text{'composite logical object'}$  ) ) )
```

Semiformal Description 2.22

Predicate “is a basic logical object class description” (clause 7.5.6)

A basic logical object class description is an attribute set (nomination) which may contain the attributes 'application comments', 'bindings', 'content architecture class', 'content generator', 'content portions', 'enciphered', 'layout style', 'object class identifier', 'object type', 'presentation style', 'protection', 'resource', 'sealed', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a logical object class identifier and a 'basic logical object' as their respective values (15-16). None of the attribute values is defaultable (14).

Definition 2.22

```

1   $\forall cst$ 
2  ( ${}_o$  IsBasicLogicalObjectClassDescription( $cst$ ) iff
3    IsAttributeSet2.61( $cst$ ) and
4    NAMS1.18( $cst$ )  $\supseteq$  ['object class identifier'; 'object type'] and
5    NAMS1.18( $cst$ )  $\subseteq$  ['application comments'; 'bindings';
6      'content architecture class'; 'content generator';
7      'content portions'; 'enciphered';
8      'layout style'; 'object class identifier';
9      'object type'; 'presentation style';
10     'protection'; 'resource';
11     'sealed'; 'user-readable comments';
12     'user-visible name'] and
13     $\forall a \in \hat{cst}.$ 
14    ( ${}_1$  not IsPlaceholder1.19(C  $a$ ) and
15      ( ${}_2$  N  $a$  = 'object class identifier' impl IsLogicalObjectClassId2.80(C  $a$ )  ${}_2$ ) and
16      ( ${}_3$  N  $a$  = 'object type' impl C  $a$  = 'basic logical object'  ${}_3$ )  ${}_1$ )  ${}_o$ )

```

Semiformal Description 2.23

Predicate “is a logical object class description” (clause 7.5.6)

A logical object class description is either a basic logical object class description or a composite logical object class description.

Definition 2.23

```

1   $\forall cst$ 
2  ( ${}_o$  IsLogicalObjectClassDescription( $cst$ ) iff
3    IsBasicLogicalObjectClassDescription2.22( $cst$ ) or
4    IsCompositeLogicalObjectClassDescription2.21( $cst$ )  ${}_o$ )

```

Semiinformal Description 2.24

Predicate “is a layout root class description” (clause 7.5.7)

A layout root class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'default value lists', 'generator for subordinates', 'object class identifier', 'object type', 'resource', 'sealed', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have 0 and 'document layout root' as their respective values (17–18). None of the attribute values is defaultable (12). The value of 'balance' is either 'null' or a sequence of immediately subordinate object class identifiers. In the latter case, the attribute 'generator for subordinates' must be specified (13–16).

NOTE — The required correspondence between these object class identifiers and the value of 'generator for subordinates' (5.4.2.1) can only be checked during the layout process.

Definition 2.24

```

1   $\forall cst$ 
2  ( $\text{IsLayoutRootClassDescription}(cst)$  iff
3     $\text{IsAttributeSet}^{2.61}(cst)$  and
4     $\text{NAMS}^{1.18}(cst) \supseteq [\text{'object class identifier'}; \text{'object type'}]$  and
5     $\text{NAMS}^{1.18}(cst) \subseteq [\text{'application comments'}; \text{'balance'};$ 
6       $\text{'bindings'};$ 
7       $\text{'generator for subordinates'}; \text{'object class identifier'};$ 
8       $\text{'object type'};$ 
9       $\text{'sealed'};$ 
10      $\text{'user-readable comments'};$ 
11      $\text{'user-visible name'}]$  and
12      $\forall a \in \sim cst.$ 
13     ( $\text{not IsPlaceholder}^{1.19}(C a)$  and
14       ( $N a = \text{'balance'}$  impl
15         ( $C a = \text{'null'}$  or
16           ( $\text{'generator for subordinates'} \in \text{NAMS}^{1.18}(cst)$  and
17              $\forall so \in C \sim cst. \text{'balance'}. ((C so) \text{IsClassIdImSubIn}^{2.142}(cst))$   $\text{'_s'}_2)$  and
18             ( $N a = \text{'object class identifier'}$  impl  $C a = [\rightarrow 0 \rightarrow]$   $\text{'_s'}_3)$  and
19             ( $N a = \text{'object type'}$  impl  $C a = \text{'document layout root'}$   $\text{'_s'}_1)$  and
19             not ( $cst$ )DescribesClassImSubOf2.150( $cst$ ) $\text{'_o'}$ )
```

Semiformal Description 2.25

Predicate “is a page set class description” (clause 7.5.7)

A page set class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'default value lists', 'generator for subordinates', 'layout stream categories', 'layout stream sub-categories', 'object class identifier', 'object type', 'resource', 'sealed', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a layout object class identifier and 'page set' as their respective values (18, 19). None of the attribute values is defaultable (13). The value of 'balance' is either 'null' or a sequence of immediately subordinate object class identifiers. In the latter case, the attribute 'generator for subordinates' must be specified (14-17).

NOTE — The required correspondence between these object class identifiers and the value of 'generator for subordinates' (5.4.2.1) can only be checked during the layout process.

Definition 2.25

```

1   $\forall cst$ 
2  ( $_0$  IsPageSetClassDescription( $cst$ ) iff
3    IsAttributeSet2.61( $cst$ ) and
4    NAMS1.18( $cst$ )  $\supseteq$  ['object class identifier'; 'object type'] and
5    NAMS1.18( $cst$ )  $\subseteq$  ['application comments'; 'balance';
6      'bindings'; 'default value lists';
7      'generator for subordinates'; 'layout stream categories';
8      'layout stream sub-categories'; 'object class identifier';
9      'object type'; 'resource';
10     'sealed'; 'user-readable comments';
11     'user-visible name'] and
12   $\forall a \in \hat{cst} \bullet$ 
13    ( $_1$  not IsPlaceholder1.19( $C\ a$ ) and
14      ( $_2$   $N\ a = \text{'balance'}$  impl
15        ( $_3$   $C\ a = \text{'null'}$  or
16          ( $_4$  'generator for subordinates'  $\in$  NAMS1.18( $cst$ ) and
17             $\forall so \in C \cap \hat{cst} \bullet \text{'balance'} \bullet ((C\ so) \text{IsClassIdImSubIn}^{2.142}(cst))\ _4$ ) and
18            ( $_5$   $N\ a = \text{'object class identifier'}$  impl  $\text{IsLayoutObjectClassId}^{2.81}(C\ a)\ _5$ ) and
19            ( $_6$   $N\ a = \text{'object type'}$  impl  $C\ a = \text{'page set'}$ ) and

```

Semiformal Description 2.26

Predicate “is a composite page class description” (clause 7.5.7)

A composite page class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'colour', 'colour of layout object', 'default value lists', 'dimensions', 'generator for subordinates', 'layout stream categories', 'layout stream sub-categories', 'medium type', 'object class identifier', 'object type', 'page position', 'resource', 'sealed', 'transparency', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a layout object class identifier and a 'composite or basic page' as their respective values (23, 24). None of the attribute values is defaultable (16). The value of 'balance' is either 'null' or a sequence of immediately subordinate object class identifiers. In the latter case, the attribute 'generator for subordinates' must be specified (17–20). The sub-parameters 'rule A', 'rule B' and 'variable page height' may not be specified for the attribute 'dimensions' (21, 22).

NOTE — The required correspondence between these object class identifiers and the value of 'generator for subordinates' (5.4.2.1) can only be checked during the layout process.

Definition 2.26

```

1   $\forall cst$ 
2  ( $\text{IsCompositePageClassDescription}(cst)$  iff
3     $\text{IsAttributeSet}^{2.61}(cst)$  and
4     $\text{NAMS}^{1.18}(cst) \supseteq$  ['object class identifier'; 'object type'] and
5     $\text{NAMS}^{1.18}(cst) \subseteq$  ['application comments'; 'balance';
6      'bindings'; 'colour';
7      'colour of layout object'; 'default value lists';
8      'dimensions'; 'generator for subordinates';
9      'layout stream categories'; 'layout stream sub-categories';
10     'medium type'; 'object class identifier';
11     'object type'; 'page position';
12     'resource'; 'sealed';
13     'transparency'; 'user-readable comments';
14     'user-visible name'] and
15      $\forall a \in \sim cst \bullet$ 
16     ( $\text{not IsPlaceholder}^{1.19}(C a)$  and
17       ( $N a = \text{'balance'}$  impl
18         ( $C a = \text{'null'}$  or
19           ( $\text{'generator for subordinates'} \in \text{NAMS}^{1.18}(cst)$  and
20              $\forall so \in C \sim cst \bullet \text{'balance'} \bullet ((C so) \text{IsClassIdImSubIn}^{2.142}(cst))$   $)_3)_2$ ) and
21       ( $N a = \text{'dimensions'}$  impl
22         ( $\forall b \in a \bullet (\neg N b \notin [\text{'rule A'; 'rule B'; 'variable page height'}] \neg)_6)_5$ ) and
23       ( $N a = \text{'object class identifier'}$  impl  $\text{IsLayoutObjectClassId}^{2.81}(C a)$   $)_8$ ) and
24       ( $N a = \text{'object type'}$  impl  $C a = \text{'composite or basic page'}$   $)_9)_0$ )

```

Semiformal Description 2.27

Predicate "is a basic page class description" (clause 7.5.7)

A basic page class description is an attribute set (nomination) which may contain the attributes 'application comments', 'bindings', 'colour', 'colour of layout object', 'content architecture class', 'content background colour', 'content colour table', 'content foreground colour', 'content generator', 'content portions', 'dimensions', 'enciphered', 'layout stream categories', 'layout stream sub-categories', 'medium type', 'object class identifier', 'object colour table', 'object type', 'page position', 'presentation attributes', 'presentation style', 'resource', 'sealed', 'transparency', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a layout object class identifier and a 'composite or basic page' as their respective values (22, 23). None of the attribute values is defaultable (19). The sub-parameters 'rule A' and 'rule B' may not be specified for the attribute 'dimensions' (20, 21).

Definition 2.27

```

1   $\forall cst$ 
2  ( ${}_0 \text{IsBasicPageClassDescription}(cst)$  iff
3     $\text{IsAttributeSet}^{2.61}(cst)$  and
4     $\text{NAMS}^{1.18}(cst) \supseteq [ \text{'object class identifier'}; \text{'object type'} ]$  and
5     $\text{NAMS}^{1.18}(cst) \subseteq [ \text{'application comments'}; \text{'bindings'};$ 
6       $\text{'colour'};$   $\text{'colour of layout object'};$ 
7       $\text{'content architecture class'};$   $\text{'content background colour'};$ 
8       $\text{'content colour table'};$   $\text{'content foreground colour'};$ 
9       $\text{'content generator'};$   $\text{'content portions'};$ 
10      $\text{'dimensions'};$   $\text{'enciphered'};$ 
11      $\text{'layout stream categories'};$   $\text{'layout stream sub-categories'};$ 
12      $\text{'medium type'};$   $\text{'object class identifier'};$ 
13      $\text{'object colour table'};$   $\text{'object type'};$ 
14      $\text{'page position'};$   $\text{'presentation attributes'};$ 
15      $\text{'presentation style'};$   $\text{'resource'};$ 
16      $\text{'sealed'};$   $\text{'transparency'};$ 
17      $\text{'user-readable comments'}; \text{'user-visible name'} ]$  and
18     $\forall a \in {}^{\sim}cst.$ 
19    ( ${}_1 \text{not IsPlaceholder}^{1.19}(C a)$  and
20      ( ${}_2 N a = \text{'dimensions'}$  impl
21        ( ${}_3 \forall b \in a. . ({}_4 N b \notin [ \text{'rule A'}; \text{'rule B'} ] )$ ) ${}_3$ ) ${}_2$ ) and
22      ( ${}_5 N a = \text{'object class identifier'}$  impl  $\text{IsLayoutObjectClassId}^{2.81}(C a)$ ) ${}_5$ ) and
23      ( ${}_6 N a = \text{'object type'}$  impl  $C a = \text{'composite or basic page'}$ ) ${}_6$ ) ${}_0$ )

```

Semiformal Description 2.28

Predicate “is a frame class description” (clause 7.5.7)

A frame class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'border', 'colour', 'colour of layout object', 'default value lists', 'dimensions', 'generator for subordinates', 'layout path', 'layout stream categories', 'layout stream sub-categories', 'logical source', 'object class identifier', 'object colour table', 'object type', 'permitted categories', 'position', 'resource', 'sealed', 'transparency', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a layout object class identifier and 'frame' as their respective values (25, 26). None of the attribute values is defaultable (18). The value of 'balance' is either 'null' or a sequence of immediately subordinate object class identifiers (19–22). In the latter case, the attribute 'generator for subordinates' must be specified (21, 22). The sub-parameter 'variable page height' may not be specified for the attribute 'dimensions' (23, 24).

NOTE: The required correspondence between these object class identifiers and the value of 'generator for subordinates' (5.4.2.1) can only be checked during the layout process.

Definition 2.28

```

1   $\forall cst$ 
2  ( $\text{IsFrameClassDescription}(cst)$  iff
3     $\text{IsAttributeSet}^{2.61}(cst)$  and
4     $\text{NAMS}^{1.18}(cst) \supseteq [\text{'object class identifier'; 'object type'}]$  and
5     $\text{NAMS}^{1.18}(cst) \subseteq [\text{'application comments'; 'balance';
6      'bindings'; 'border';
7      'colour'; 'colour of layout object';
8      'default value lists'; 'dimensions';
9      'generator for subordinates'; 'layout path';
10     'layout stream categories'; 'layout stream sub-categories';
11     'logical source'; 'object class identifier';
12     'object colour table'; 'object type';
13     'permitted categories'; 'position';
14     'resource'; 'sealed';
15     'transparency'; 'user-readable comments';
16     'user-visible name'] and
17    $\forall a \in \sim cst \bullet$ 
18   ( $\text{not } \text{IsPlaceholder}^{1.19}(C\ a)$  and
19     ( $N\ a = \text{'balance'}$  impl
20       ( $C\ a = \text{'null'}$  or
21         ( $\text{'generator for subordinates'} \in \text{NAMS}^{1.18}(cst)$  and
22            $\forall so \in C \sim cst \bullet \text{'balance'} \bullet ((C\ so) \text{IsClassIdImSubIn}^{2.142}(cst))$   $\text{, ,})_2$ ) and
23         ( $N\ a = \text{'dimensions'}$  impl
24           ( $\forall b \in a \bullet (\text{'variable page height'} \neq b)$   $\text{, ,})_6$ ) and
25         ( $N\ a = \text{'object class identifier'}$  impl  $\text{IsLayoutObjectClassId}^{2.81}(C\ a)$   $\text{, ,})_8$ ) and
26         ( $N\ a = \text{'object type'}$  impl  $C\ a = \text{'frame'}$   $\text{, ,})_9$ ))$ 
```

Semiformal Description 2.29

Predicate “is a block class description” (clause 7.5.7)

A block class description is an attribute set (nomination) which may contain the attributes 'application comments', 'bindings', 'border', 'colour', 'colour of layout object', 'content architecture class', 'content background colour', 'content colour table', 'content foreground colour', 'content generator', 'content portions', 'dimensions', 'enciphered', 'layout stream categories', 'layout stream sub-categories', 'object class identifier', 'object colour table', 'object type', 'position', 'presentation attributes', 'presentation style', 'resource', 'sealed', 'transparency', 'user-readable comments' and 'user-visible name'. The attributes 'object class identifier' and 'object type' are both required and have a layout object class identifier and 'block' as their respective values (24, 25). None of the attribute values is defaultable (19). The sub-parameters 'rule A', 'rule B' and 'variable page height' may not be specified for the attribute 'dimensions' (20, 21). The parameter 'variable position' may not be specified for the attribute 'position' (22, 23).

Definition 2.29

```

1   $\forall cst$ 
2  ( $_0$  IsBlockClassDescription( $cst$ ) iff
3    IsAttributeSet2.61( $cst$ ) and
4    NAMS1.18( $cst$ )  $\supseteq$  ['object class identifier'; 'object type'] and
5    NAMS1.18( $cst$ )  $\subseteq$  ['application comments'; 'bindings';
6      'border'; 'colour';
7      'colour of layout object'; 'content architecture class';
8      'content background colour'; 'content colour table';
9      'content foreground colour'; 'content generator';
10     'content portions'; 'dimensions';
11     'enciphered'; 'layout stream categories';
12     'layout stream sub-categories'; 'object class identifier';
13     'object colour table'; 'object type';
14     'position'; 'presentation attributes';
15     'presentation style'; 'resource';
16     'sealed'; 'transparency';
17     'user-readable comments'; 'user-visible name'] and
18     $\forall a \in \sim cst.$ 
19    ( $_1$  not IsPlaceholder1.19(C  $a$ ) and
20      ( $_2$  N  $a$  = 'dimensions' impl
21        ( $_3$   $\forall b \in a..$  ( $_4$  N  $b \notin$  ['rule A'; 'rule B'; 'variable page height']  $_4$ ) $_3$ ) $_2$ ) and
22        ( $_5$  N  $a$  = 'position' impl
23          ( $_6$   $\forall b \in a..$  ( $_7$  N  $b \neq$  'variable position'  $_7$ ) $_6$ ) $_5$ ) and
24          ( $_8$  N  $a$  = 'object class identifier' impl IsLayoutObjectClassId2.81(C  $a$ )  $_8$ ) and
25          ( $_9$  N  $a$  = 'object type' impl C  $a$  = 'block'  $_9$ ) $_1$ )

```

Semiformal Description 2.30

Predicate “is a layout object class description” (clause 7.5.7)

A layout object class description is either a basic layout object class description or composite layout object class description.

Definition 2.30

```

1   $\forall cst$ 
2  ( $_0$  IsLayoutObjectClassDescription( $cst$ ) iff
3    IsBasicLayoutObjectClassDescription2.33( $cst$ ) or
4    IsCompositeLayoutObjectClassDescription2.32( $cst$ ) $_0$ )

```

Semiformal Description 2.31

Predicate “is a composite object class description” (clauses 7.5.6, 7.5.7)

A composite object class description is either a composite logical object class description or a composite layout object class description.

Definition 2.31

- 1 $\forall cst$
- 2 $({}_o \text{IsCompositeObjectClassDescription}(cst) \text{ iff}$
- 3 $\text{IsCompositeLogicalObjectClassDescription}^{2.21}(cst) \text{ or}$
- 4 $\text{IsCompositeLayoutObjectClassDescription}^{2.32}(cst)_o)$

Semiformal Description 2.32

Predicate “is a composite layout object class description” (clause 7.5.7)

A composite layout object class description is a layout root class description or a page set class description or a composite page class description or a frame class description.

Definition 2.32

- 1 $\forall cst$
- 2 $({}_o \text{IsCompositeLayoutObjectClassDescription}(cst) \text{ iff}$
- 3 $\text{IsLayoutRootClassDescription}^{2.24}(cst) \text{ or}$
- 4 $\text{IsPageSetClassDescription}^{2.25}(cst) \text{ or}$
- 5 $\text{IsCompositePageClassDescription}^{2.26}(cst) \text{ or}$
- 6 $\text{IsFrameClassDescription}^{2.28}(cst)_o)$

Semiformal Description 2.33

Predicate “is a basic layout object class description” (clause 7.5.7)

A basic layout object class description is either a basic page class description or a block class description.

Definition 2.33

- 1 $\forall cst$
- 2 $({}_o \text{IsBasicLayoutObjectClassDescription}(cst) \text{ iff}$
- 3 $\text{IsBasicPageClassDescription}^{2.27}(cst) \text{ or}$
- 4 $\text{IsBlockClassDescription}^{2.29}(cst)_o)$

Semiformal Description 2.34

Predicate “is a basic object class description” (clauses 7.5.6, 7.5.7)

A basic object class description is either a basic logical object class description or a basic layout object class description.

Definition 2.34

- 1 $\forall cst$
- 2 $({}_o \text{IsBasicObjectClassDescription}(cst) \text{ iff}$
- 3 $\text{IsBasicLogicalObjectClassDescription}^{2.22}(cst) \text{ or}$
- 4 $\text{IsBasicLayoutObjectClassDescription}^{2.33}(cst)_o)$

Semiinformal Description 2.35

Predicate “is an object class description” (clauses 7.5.6, 7.5.7)

An object class description is either a logical object class description or a layout object class description. The “binding value” is a collection of catenations with two components where the second one is a bindings value expression.

Definition 2.35

- 1 $\forall cst$
- 2 $({}_0 \text{IsObjectClassDescription}(cst) \text{ iff}$
- 3 $({}_1 \text{IsLogicalObjectClassDescription}^{2.23}(cst) \text{ or}$
- 4 $\text{IsLayoutObjectClassDescription}^{2.30}(cst)_1) \text{ and}$
- 5 $\forall a \in \sim cst.$
- 6 $({}_2 N a = \text{'bindings'} \text{ impl}$
- 7 $({}_3 \text{IsNeCol}^{1.1}(C a) \text{ and}$
- 8 $\forall b \in (C a) \exists l, r$
- 9 $({}_4 b = [\rightarrow l \rightarrow r \rightarrow] \text{ and } \text{IsBindingsValueExpression}^{2.63}(r) \text{ }_4)_3)_2)_0)$

Semiinformal Description 2.36

Predicate “is a root description” (clauses 7.5.6, 7.5.7)

A root description is either a logical root description or a layout root description.

Definition 2.36

- 1 $\forall cst$
- 2 $({}_0 \text{IsRootDescription}(cst) \text{ iff}$
- 3 $\text{IsLogicalRootDescription}^{2.37}(cst) \text{ or } \text{IsLayoutRootDescription}^{2.41}(cst)_0)$

Semiinformal Description 2.37

Predicate “is a logical root description” (clause 7.2.1)

A logical root description is an attribute set (nomination) which contains the attributes 'application comments', 'bindings', 'object identifier', 'object type', 'protection', 'subordinates', 'user-readable comments', 'user-visible name' and, optionally, the attributes 'enciphered', 'layout style', 'default value lists', 'object class' and 'sealed'. The attributes 'object class' and 'object identifier' have a logical object class identifier and 3 as their respective values (16, 17). The attribute 'object type' has 'document logical root' as its value (18, 19). If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (20).

Definition 2.37

```

1   $\forall cst$ 
2   $({}_o \text{IsLogicalRootDescription}(cst) \text{ iff}$ 
3     $\text{IsAttributeSet}^{2.61}(cst) \text{ and}$ 
4     $\text{NAMS}^{1.18}(cst) \supseteq [ \text{'application comments'}; \text{'bindings'};$ 
5       $\text{'object identifier'}; \text{'object type'};$ 
6       $\text{'protection'}; \text{'subordinates'};$ 
7       $\text{'user-readable comments'}; \text{'user-visible name'} ] \text{ and}$ 
8     $\text{NAMS}^{1.18}(cst) \subseteq [ \text{'application comments'}; \text{'bindings'};$ 
9       $\text{'default value lists'}; \text{'enciphered'};$ 
10      $\text{'layout style'}; \text{'object class'};$ 
11      $\text{'object identifier'}; \text{'object type'};$ 
12      $\text{'protection'}; \text{'sealed'};$ 
13      $\text{'subordinates'}; \text{'user-readable comments'};$ 
14      $\text{'user-visible name'} ] \text{ and}$ 
15     $\forall a \in \sim cst.$ 
16     $({}_1 ({}_2 N a = \text{'object class'} \text{ impl } \text{IsLogicalObjectClassId}^{2.80}(C a) {}_2) \text{ and}$ 
17       $({}_3 N a = \text{'object identifier'} \text{ impl } C a = [ \rightarrow 3 \rightarrow ] {}_3) \text{ and}$ 
18       $({}_4 N a = \text{'object type'} \text{ impl}$ 
19         $({}_5 C a = \text{'document logical root'} \text{ or } \text{IsPlaceholder}^{1.19}(C a) {}_5) {}_4) \text{ and}$ 
20       $({}_6 \text{IsPlaceholder}^{1.19}(C \sim cst. \text{'object type'}) \text{ impl } \text{'object class'} \in \text{NAMS}^{1.18}(cst) {}_6) {}_0)$ 

```