INTERNATIONAL STANDARD

ISO/IEC 8613-10

First edition
1991-06-01

Information processing — Text and office systems — Office Document Architecture (ODA) and interchange format —

Part 10:
Formal specifications

Traitement de l'information — Bureautique — Architecture des documents de bureau (ODA) et format d'échange —

Partie 10: Spécifications formelles

Reference number
ISO/IEC 8613-10 : 1991 (E)

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*.

Development of this International Standard has been in parallel with

— ECMA 101 : 1985, *Office document architecture*;

— CCITT Recommendation T.37 (1984): *Document interchange protocol for the telematic services*;

— CCITT Recommendations in the T.410 series (1988): *Open Document Architecture (ODA) and Interchange format*.

ISO 8613 consists of the following parts, under the general title *Information processing — Text and office systems — Office Document Architecture (ODA) and interchange format*:

— *Part 1: Introduction and general principles*

— *Part 2: Document structures*

— *Part 4: Document profile*

— *Part 5: Office document interchange format (ODIF)*

— *Part 6: Character content architectures*

— *Part 7: Raster graphics content architectures*

— *Part 8: Geometric graphics content architectures;*

— *Part 10: Formal specifications.*

NOTE — At present, there are no parts 3 and 9.

Further parts may be added to this International Standard.

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

Information processing — Text and office systems — Office Document Architecture (ODA) and interchange format —

Part 10: Formal specifications

1 Scope

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

In the context of ISO 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 8613 is designed to allow for extensions, including typographical features, colour, spreadsheets and additional types of content such as sound.

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

Furthermore, ISO 8613 allows for the interchange of documents containing one or more different types of content such as character text, images, graphics and sound.

This part of ISO/IEC 8613

— specifies a formal description technique appropriate for describing the technical specifications of the document structures (ISO 8613-2), the document profile (ISO 8613-4) and the content architectures (ISO 8613-6, -7 and -8);

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 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 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

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, ∀ (for all) and ∃ (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 [≡ letter |≡ digit]... ≡ lower-case-letter [≡ letter |≡ digit]... |
= |≠|<|≤|>|≥|∈|∉|∈̂|⊂|⊆|⊃|⊇

NOTE 4: The semantics of the terminal symbols (=, ≠, ... ⊇) 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 [≡ letter |≡ digit]... [≡ subscript-digit]...

constant : : =
standard-constant |
nonstandard-constant

standard-constant : : =
UNDEF |
empty-constant |
number-atom-constant

empty-constant : : =
[ ] –. empty collection .– |
[:] –. empty nomination .– |
[→] –. empty catenation .– |
< > –. empty spotset .–

number-atom-constant : : =
[+ ≡| − ≡] digit [≡ digit]... [≡ . ≡ digit [≡ digit]...]

nonstandard-constant : : =
' ≡ character [≡ character]... ≡ ' –. restriction on apostrophe occurrence .–

operator-term : : =
[parameter-part] operator-symbol-part ...
[parameter-part operator-symbol-part ...]...[parameter-part]

operator-symbol-part : : =
upper-case-letter [≡ upper-case-letter |≡ digit |≡ _]... |
^ | + | − | * | / | ∪ | ∩ | \ | // | • | * |↓|↑

NOTE 5: The semantics of the terminal symbols (^, +, ... ↑) in this production rule are specified in 4.4.

explicit-composition-term : : =
[term [; term]...] –. collection .– |
[term : term [; term : term]...] –. nomination .– |
[→ term [→ term]... →] –. catenation .– |
" ≡ character [≡ character]... ≡ " –. catenation of characters, restriction on quote occurrence .–

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	*not* $t_1 = t_2$
$t_1 < t_2$	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	*not* $t_1 \in t_2$
$t_1 \hat{\in} t_2$	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).

[→ m_1 → m_2 → m_3 →]	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 [→ 'O' → 'D' → 'A' → ' ' → 'P' → 'a' → 'r' → 't' → ' ' → '2' →]. 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)
[→]	Denotes the empty catenation. (This is an example for empty-constant)
< >	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)
[*var* | formula]	Denotes the collection of all constructs *var* which satisfy the formula. (This is an example for extensional-collection-term)
< *var* | formula >	Denotes the spotset which is the union of all singleton spotsets *var* which satisfy the formula. (This is an example for extensional-spotset-term)
t <*var* || formula >	If t denotes a (possibly empty) spotset, the whole term denotes the union of all singleton spotsets *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 < formula >	If a variable *var* is not introduced explicitly, the abbreviated term t < formula > 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 <$n_1, n_2, ...$>	If the formula has the structure N *var* = n_1 *or* N *var* = n_2 *or* ... 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)
$t.n$ $t \downarrow n$ $t^{\bullet}n$ $t \uparrow n$	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 $\bullet$, $^{\bullet}$ etc.). The n stands for <N *xs* = n> (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	$\forall x \forall y \exists z$ (formula)
or even as	$\forall x, y \exists z$ (formula)

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

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

This notation can be combined with the previous one:

The expression $\forall x(x \in m$ *impl* $\forall y(\exists z(z \in p$ *and* formula)))

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, each of these formal specifications is a single formula in first-order predicate logic. This formula, being given near the beginning of each formal specification, is called the overall formula. The formula consists of other formulae which are connected by *and*:

$$\text{formula}_1 \ \textit{and} \ \text{formula}_2 \ \textit{and} \ \text{formula}_3 \ \textit{and} \ \ldots \ \text{formula}_n$$

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 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 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 the ISO 8613-2 are defined in 7.5.

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:

$\forall$ *variable* (*predicate-symbol*(*variable*) *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*) = IF *formula* THEN *term* 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 Office Document Interchange Format (ISO 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 8613 and do not necessarily reflect the ODIF encoding of ODA documents. Those clauses in ISO 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.

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 $(_0\,\mathrm{IsNeCol}(col)$ *iff*
3 $\quad \mathrm{IsCol}(col)$ *and* $col \neq [\,]_0)$

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 $(_0\,\mathrm{IsNeNom}(nom)$ *iff*
3 $\quad \mathrm{IsNom}(nom)$ *and* $nom \neq [:]_0)$

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 $(_0\,\mathrm{IsNeCat}(cat)$ *iff*
3 $\quad \mathrm{IsCat}(cat)$ *and* $cat \neq [\hookrightarrow]_0)$

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 $(_0\,\mathrm{IsEmptyCol}(col)$ *iff*
3 $\quad col = [\,]_0)$

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   ∀q
2   (₀IsPairOfPosInt(q) iff
3     ∃l,r
4     (₁q = [ → l → r → ] and IsNat(l) and IsNat(r)₁)₀)
```

Semiformal Description 1.10

Predicate "is an octet string"

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

Definition 1.10

```
1   ∀v
2   (₀IsOctetString(v) iff
3     IsAtom(v)₀)
```

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   ∀p
2   (₀PF(p) =
3     IF IsSingle(p) and IsCat(C p•)
4     THEN LENGTH^1.16 (F p) + 1
5     ELSE UNDEF₀)
```

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   ∀p
2   (₀PR(p) =
3     IF IsSingle(p) and IsCat(C p•)
4     THEN LENGTH^1.16 (R p) + 1
5     ELSE UNDEF₀)
```

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   ∀ ss
2   (o COLC (ss) =
3     IF IsSpotset(ss)
4     THEN [g | ∃ p ∈̂ 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   ∀ n
2   (o NAMS(n) =
3     IF IsNom(n)
4     THEN [m | ∃ p ∈̂ ˆn • (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 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   ∀ v
2   (o IsPlaceholder(v) iff
3     v = '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   ∀ v
2   (o IsISO6937MSString(v) iff
3     IsNeCat^1.3(v) and ∀ m ∈̂ ˆv • (IsISO6937MSCharacter^1.21(C m)) o)
```

Semiformal 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   ∀ v
2   (₀ IsPosReal(v) iff
3     IsReal(v) and v > 0.0₀)
```

and ... IsBasicLayoutObjectClassDescription$^{2.35}(cst)$ ...
and ... IsBasicObjectClassDescription$^{2.36}(cst)$ ...
and ... IsObjectClassDescription$^{2.37}(cst)$ ...

and ... IsRootDescription$^{2.38}(cst)$ ...

and ... IsLogicalRootDescription$^{2.39}(cst)$ ...
and ... IsCompositeLogicalObjectDescription$^{2.40}(cst)$ ...
and ... IsBasicLogicalObjectDescription$^{2.41}(cst)$ ...
and ... IsLogicalObjectDescription$^{2.42}(cst)$ ...

and ... IsLayoutRootDescription$^{2.43}(cst)$ ...
and ... IsPageSetDescription$^{2.44}(cst)$ ...
and ... IsCompositePageDescription$^{2.45}(cst)$ ...
and ... IsBasicPageDescription$^{2.46}(cst)$ ...
and ... IsPageDescription$^{2.47}(cst)$ ...
and ... IsFrameDescription$^{2.48}(cst)$ ...
and ... IsBlockDescription$^{2.49}(cst)$ ...
and ... IsLayoutObjectDescription$^{2.50}(cst)$ ...

and ... IsCompositeObjectDescription$^{2.51}(cst)$ ...
and ... IsBasicObjectDescription$^{2.52}(cst)$ ...
and ... IsCompositeLayoutObjectDescription$^{2.53}(cst)$ ...
and ... IsBasicLayoutObjectDescription$^{2.54}(cst)$ ...
and ... IsObjectDescription$^{2.55}(cst)$ ...

and ... IsContentPortionDescription$^{2.56}(cst)$ ...
and ... IsCharacterContentPortionDescription$^{2.57}(cst)$ ...
and ... IsRasterGraphicsContentPortionDescription$^{2.58}(cst)$ ...
and ... IsGeometricGraphicsContentPortionDescription$^{2.59}(cst)$ ...

and ... IsLayoutStyle$^{2.60}(cst)$ ...
and ... IsPresentationStyle$^{2.61}(cst)$ ...

-.attributes: .-

and ... IsAttributeSet$^{2.62}(as)$ ...
and ... IsProfileAttributeSetPart2$^{2.63}(as)$ ...

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

and ... (*slog*)IsContainedIn$^{2.141}$(*slay*) ...
and ... (*id*)IsClassIdImSubIn$^{2.142}$(*gco*) ...
and ... (*id*)IsIdImSubIn$^{2.143}$(*sco*) ...
and ... (*id*)IsIdContIn$^{2.144}$(*b*) ...
and ... (*idq*)IsSeqClassIdImSubIn$^{2.145}$(*gco*) ...
and ... (*idq*)CoveredBy$^{2.146}$(*r*) ...
and ... (*id*)OccursIn$^{2.147}$(*m*) ...
and ... (*go*)IsInitialGenericIn$^{2.148}$(*gos*) ...
and ... (*so*)IsInitialSpecificIn$^{2.149}$(*sos*) ...
and ... (*lgo*)DescribesClassImSubOf$^{2.150}$(*hgo*) ...
and ... (*lgo*)DescribesClassSubOf$^{2.151}$(*hgo*)In(*g*) ...
and ... (*lso*)DescribesImSubOf$^{2.152}$(*hso*) ...
and ... (*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 8613 by establishing

$\mathcal{AS}$ *impl* IsInterchangeSet$^{2.1}$(-.**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 8613, if False, it does not. An undefined result is impossible.

Semiformal Description 2.3

Predicate "is a document description" (2.3)

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 2.3

1 $\forall\, doc$
2 $(_0$ IsDocumentDescription(doc) *iff*
3 $\exists\, prof$
4 $(_1$ IsDocumentProfilePart2$^{2.20}(prof)$ *and*
5 $(_2\, doc = [prof]$ *or* IsProcessable$^{2.4}(doc)$ *or*
6 IsFormattedProcessable$^{2.5}(doc)$ *or* IsFormatted$^{2.6}(doc)\ _2)$ *and*
7 $doc \neq [prof]$ *impl*
8 $(_3(_4$ C $\hat{}prof$ • 'document architecture class' = 'processable' *iff*
9 IsProcessable$^{2.4}(doc)_4)$ *and*.
10 $(_5$ C $\hat{}prof$ • 'document architecture class' = 'formatted processable' *iff*
11 IsFormattedProcessable$^{2.5}(doc)_5)$ *and*
12 $(_6$ C $\hat{}prof$ • 'document architecture class' = 'formatted' *iff*
13 IsFormatted$^{2.6}(doc)_6)$ *and*
14 $\forall\, cst \in doc$
15 $(_7$ 'resource' $\in$ NAMS$^{1.18}(cst)$ *impl* 'resource document' $\in$ NAMS$^{1.18}(prof)\ _7)_3)_1)_0)$

Semiformal Description 2.5

Predicate "is formatted processable" (2.3.6, 2.3.11, 8.2)

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 2.5

1 $\forall\, doc$
2 $(_0\, \mathrm{IsFormattedProcessable}(doc)$ *iff*
3 $\exists\, prof, doby, glog, slog, glay, slay, laystys, prestys$
4 $(_1\, doc = [prof] \cup doby$ *and*
5 $\mathrm{IsDocumentProfilePart2}^{2.20}(prof)$ *and* $\mathrm{IsDocumentBody}^{2.7}(doby)$ *and*
6 $doby = glog \cup slog \cup glay \cup slay \cup laystys \cup prestys$ *and*
7 'specific logical structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and* $\mathrm{IsSpecificLogicalDescription}^{2.13}(slog)$ *and*
8 'specific layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and* $\mathrm{IsSpecificLayoutDescription}^{2.14}(slay)$ *and*
9 'generic layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and* $\mathrm{IsGenericLayoutDescription}^{2.11}(glay)$ *and*
10 $(_2$ 'generic logical structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *iff*
11 $(_3\, \mathrm{IsNeCol}^{1.1}(glog)$ *and* $\mathrm{IsGenericLogicalDescription}^{2.10}(glog)_3)_2)$ *and*
12 $(_4$ 'layout styles' $\in \mathrm{NAMS}^{1.18}(prof)$ *iff*
13 $(_5\, \mathrm{IsNeCol}^{1.1}(laystys)$ *and* $\mathrm{IsLayoutStyleSet}^{2.15}(laystys)_5)_4)$ *and*
14 $(_6$ 'presentation styles' $\in \mathrm{NAMS}^{1.18}(prof)$ *iff*
15 $(_7\, \mathrm{IsNeCol}^{1.1}(prestys)$ *and* $\mathrm{IsPresentationStyleSet}^{2.16}(prestys)_7)_6)$ *and*
16 $(_8\, \mathrm{C}\ \hat{}prof \bullet$ 'generic logical structure' = 'complete generator set' *impl*
17 $(slog)\mathrm{IsCompletelyDescribedBy}^{2.136}(glog)_8)$ *and*
18 $(_9\, \mathrm{C}\ \hat{}prof \bullet$ 'generic layout structure' = 'complete generator set' *impl*
19 $(slay)\mathrm{IsCompletelyDescribedBy}^{2.136}(glay)_9)$ *and*
20 $(slog)\mathrm{IsContainedIn}^{2.141}(slay)$ *and*
21 $\mathrm{C}\ \hat{}prof \bullet$ 'document architecture class' = 'formatted processable' *and*
22 $\mathrm{SatisfiesCharacterContentArchitectureConstraints}^{2.17}(prof, doby)$ *and*
23 $\mathrm{SatisfiesRasterGraphicsContentArchitectureConstraints}^{2.18}(prof, doby)$ *and*
24 $\mathrm{SatisfiesGeometricGraphicsContentArchitectureConstraints}^{2.19}(prof, doby)_1)_0)$

Definition 2.7

$\forall\, doby$
$(_0\,\mathrm{IsDocumentBody}(doby)\ \underline{iff}$
$\quad \exists\, glog, slog, glay, slay, laystys, prestys$
$\quad (_1\, doby = glog \cup slog \cup glay \cup slay \cup laystys \cup prestys\ \underline{and}$
$\quad\quad (_2\,\mathrm{IsNeCol}^{1.1}(glog)\ \underline{impl}\ \mathrm{IsGenericLogicalDescription}^{2.10}(glog)_2)\ \underline{and}$
$\quad\quad (_3\,\mathrm{IsNeCol}^{1.1}(slog)\ \underline{impl}\ \mathrm{IsSpecificLogicalDescription}^{2.13}(slog)_3)\ \underline{and}$
$\quad\quad (_4\,\mathrm{IsNeCol}^{1.1}(slay)\ \underline{impl}\ \mathrm{IsSpecificLayoutDescription}^{2.14}(slay)_4)\ \underline{and}$
$\quad\quad (_5\,\mathrm{IsNeCol}^{1.1}(glay)\ \underline{impl}\ \mathrm{IsGenericLayoutDescription}^{2.11}(glay)_5)\ \underline{and}$
$\quad\quad (_6\,\mathrm{IsNeCol}^{1.1}(laystys)\ \underline{impl}\ \mathrm{IsLayoutStyleSet}^{2.15}(laystys)_6)\ \underline{and}$
$\quad\quad (_7\,\mathrm{IsNeCol}^{1.1}(prestys)\ \underline{impl}\ \mathrm{IsPresentationStyleSet}^{2.16}(prestys)_7)\ \underline{and}$
$\quad\quad (_8\,(_9\,\mathrm{IsNeCol}^{1.1}(slog)\ \underline{and}\ \mathrm{IsNeCol}^{1.1}(glog)_9)\ \underline{impl}\ (slog)\mathrm{IsDescribedBy}^{2.135}(glog)_8)\ \underline{and}$
$\quad\quad (_{10}\,(_{11}\,\mathrm{IsNeCol}^{1.1}(slay)\ \underline{and}\ \mathrm{IsNeCol}^{1.1}(glay)_{11})\ \underline{impl}\ (slay)\mathrm{IsDescribedBy}^{2.135}(glay)_{10})\ \underline{and}$
$\quad\quad \forall\, cst \in (glog \cup slog)\ (_{12}\forall\, x \in \hat{}cst \bullet\ (\mathrm{N}\ x = \text{'layout style'})\ \underline{impl}$
$\quad\quad (_{13}\exists\, ls \in laystys\ (\mathrm{C}\ \hat{}ls \bullet \text{'layout style identifier'} = \mathrm{C}\ x)_{13})_{12})\ \underline{and}$
$\quad\quad \forall\, cst \in (glog \cup slog \cup glay \cup slay)\ (_{14}\forall\, x \in \hat{}cst \bullet\ (\mathrm{N}\ x = \text{'presentation style'})\ \underline{impl}$
$\quad\quad (_{15}\exists\, ps \in prestys\ (\mathrm{C}\ \hat{}ps \bullet \text{'presentation style identifier'} = \mathrm{C}\ x)_{15})_{14})\ \underline{and}$
$\quad\quad \forall\, cst \in laystys\ (_{16}\forall\, y \hat{\in} \hat{}cst \bullet$
$\quad\quad (_{17}\,(_{18}\,\mathrm{IsObjectClassId}^{2.83}(\mathrm{C}\ y)\ \underline{impl}$
$\quad\quad\quad (_{19}\exists\, go \in glay\ (\mathrm{C}\ \hat{}go \bullet \text{'object class identifier'} = \mathrm{C}\ y)_{19})_{18})\ \underline{and}$
$\quad\quad\quad (_{20}\,\mathrm{IsLayoutCatName}^{2.112}(\mathrm{C}\ y)\ \underline{impl}$
$\quad\quad\quad\quad (_{21}\exists\, ls \in laystys\ (\mathrm{C}\ \hat{}ls \bullet \text{'layout category'} = \mathrm{C}\ y)_{21})_{20})\ \underline{and}$
$\quad\quad\quad \forall\, l, r$
$\quad\quad\quad (_{22}\,(_{23}\,(_{24}\,\mathrm{N}\ y = \text{'same layout object'}\ \underline{and}\ \mathrm{C}\ y = [\rightarrow l \rightarrow r \rightarrow]\ \underline{and}\ \mathrm{IsObjectClassId}^{2.83}(r)_{24})\ \underline{impl}$
$\quad\quad\quad\quad (_{25}\exists\, go \in glay\ (\mathrm{C}\ \hat{}go \bullet \text{'object class identifier'} = r)_{25})_{23})\ \underline{and}$
$\quad\quad\quad\quad (_{26}\,(_{27}\,\mathrm{N}\ y = \text{'same layout object'}\ \underline{and}\ \mathrm{C}\ y = [\rightarrow l \rightarrow r \rightarrow]\ \underline{and}\ \mathrm{IsLayoutCatName}^{2.112}(r)_{27})\ \underline{impl}$
$\quad\quad\quad\quad\quad (_{28}\exists\, ls \in laystys\ (\mathrm{C}\ \hat{}ls \bullet \text{'layout category'} = r)_{28})_{26})_{22})_{17})_{16})\ \underline{and}$
$\quad\quad \forall\, a \hat{\in} \hat{}glay \bullet \bullet \text{'logical source'}$
$\quad\quad (_{29}\exists\, go \in glog\ (\mathrm{C}\ \hat{}go \bullet \text{'object class identifier'} = \mathrm{C}\ a)_{29})\ \underline{and}$
$\quad\quad \forall\, a \hat{\in} \hat{}slay \bullet \bullet \text{'balance'}$
$\quad\quad (_{30}\,\mathrm{C}\ a \neq \text{'null'}\ \underline{impl}$
$\quad\quad\quad \forall\, b \hat{\in} \hat{}(\mathrm{C}\ a) \bullet\ (_{31}\exists\, so \in slay\ (\mathrm{C}\ \hat{}so \bullet \text{'object identifier'} = b)_{31})_{30})\ \underline{and}$
$\quad\quad \forall\, a \hat{\in} \hat{}glay \bullet \bullet \text{'balance'}$
$\quad\quad (_{32}\,\mathrm{C}\ a \neq \text{'null'}\ \underline{impl}$
$\quad\quad\quad \forall\, b \hat{\in} \hat{}(\mathrm{C}\ a) \bullet\ (_{33}\exists\, go \in glay\ (\mathrm{C}\ \hat{}go \bullet \text{'object class identifier'} = b)_{33})_{32})\ \underline{and}$
$\quad\quad \forall\, cst \in slay$
$\quad\quad (_{34}\,([\text{'imaging order'}; \text{'subordinates'}] \subset \mathrm{NAMS}^{1.18}(cst))\ \underline{impl}$
$\quad\quad\quad [x \mid x \hat{\in} \hat{}cst \bullet \text{'imaging order'}] = [y \mid y \hat{\in} \hat{}cst \bullet \text{'subordinates'}]_{34})_1)_0)$

Semiformal Description 2.10

Predicate "is a generic logical description" (2.3.12, 3.5.6)

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 2.10

1 $\forall\, glog$
2 $({}_0$ IsGenericLogicalDescription$(glog)$ *iff*
3 IsGenericDescription$^{2.8}(glog)$ *and*
4 $\forall\, cst \in glog$
5 $({}_1$ IsLogicalObjectClassDescription$^{2.25}(cst)$ *or*
6 $({}_2$ IsContentPortionDescription$^{2.56}(cst)$ *and*
7 'content identifier logical' $\in$ NAMS$^{1.18}(cst)$ *and*
8 IsLogicalObjectClassId$^{2.81}$(C ^cst • 'content identifier logical')$_2)_1)_0)$

Semiformal Description 2.11

Predicate "is a generic layout description" (2.3.12)

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 2.11

1 $\forall\, glay$
2 $({}_0$ IsGenericLayoutDescription$(glay)$ *iff*
3 IsGenericDescription$^{2.8}(glay)$ *and*
4 $\forall\, cst \in glay$
5 $({}_1$ IsLayoutObjectClassDescription$^{2.32}(cst)$ *or*
6 $({}_2$ IsContentPortionDescription$^{2.56}(cst)$ *and*
7 'content identifier layout' $\in$ NAMS$^{1.18}(cst)$ *and*
8 IsLayoutObjectClassId$^{2.82}$(C ^cst • 'content identifier layout')$_2)_1)_0)$

Semiformal Description 2.14

Predicate "is a specific layout description" (2.3.12, 3.3.1, 5.4.2.1)

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 2.14

1 $\forall\, slay$
2 $(_0$ IsSpecificLayoutDescription$(slay)$ *iff*
3 IsSpecificDescription$^{2.12}(slay)$ *and*
4 $\forall\, cst \in slay$
5 $(_1 (_2$ IsLayoutObjectDescription$^{2.50}(cst)$ *or*
6 $(_3$ IsContentPortionDescription$^{2.56}(cst)$ *and*
7 'content identifier layout' $\in$ NAMS$^{1.18}(cst)$ *and*
8 IsLayoutObjectId$^{2.79}$(C ^cst• 'content identifier layout' $_3)_2)$ *and*
9 $(_4 (_5$ IsLayoutRootDescription$^{2.43}(cst)$ *or* IsPageSetDescription$^{2.44}(cst)$ $_5)$ *impl*
10 $\forall\, so \in$ IMSUB$^{2.154}(cst, slay)$
11 (IsPageSetDescription$^{2.44}(so)$ *or* IsPageDescription$^{2.47}(so)$)$_4$) *and*
12 $(_6$ IsCompositePageDescription$^{2.45}(cst)$ *impl*
13 $\forall\, so \in$ IMSUB$^{2.154}(cst, slay)$
14 $(_7$ (IsFrameDescription$^{2.48}(so)$ *and* 'balance' $\in$ NAMS$^{1.18}(cst)$) *or*
15 (IsBlockDescription$^{2.49}(so)$ *and* 'balance' $\notin$ NAMS$^{1.18}(cst)$) $_7)_6$) *and*
16 $(_8$ IsFrameDescription$^{2.48}(cst)$ *impl*
17 $\forall\, so \in$ IMSUB$^{2.154}(cst, slay)$
18 $(_9$ (IsFrameDescription$^{2.48}(so)$ *and* 'balance' $\in$ NAMS$^{1.18}(cst)$) *or*
19 (IsBlockDescription$^{2.49}(so)$ *and* 'balance' $\notin$ NAMS$^{1.18}(cst)$) $_9)_8)_1)_0)$

Semiformal Description 2.15

Predicate "is a layout style set" (2.3.5)

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 2.15

1 $\forall\, laystys$
2 $(_0$ IsLayoutStyleSet$(laystys)$ *iff*
3 IsNeCol$^{1.1}(laystys)$ *and*
4 $\forall\, cst \in laystys$ (IsLayoutStyle$^{2.60}(cst)$) *and*
5 $\forall\, x, y \in laystys$
6 $(_1\, x \neq y$ *impl*
7 C ^x• 'layout style identifier' $\neq$ C ^y• 'layout 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 2.20

Predicate "is a document profile of Part 2" (ISO 8613-4, 5.2)

(The formal specification of the document profile has only been carried out here, as far as it is necessary for the formal specification of the document structures described in ISO 8613-2.)

A document profile is a set of profile attributes (as far as required for ISO 8613-2). The attribute 'document architecture class' is a mandatory attribute (4) and its value is either 'formatted', 'processable' or 'formatted processable'. If the value of the attribute 'document architecture class' is 'processable', and the attributes 'generic layout structure' and 'specific logical structure', but not 'specific layout structure' are specified, the value of the attribute 'generic layout structure' is 'complete generator set' (6-10). If the value of the attribute 'document architecture class' is 'formatted processable', and the attributes 'generic layout structure', 'specific logical structure' and 'specific layout structure' are specified, the value of the attribute 'generic layout structure' is 'complete generator set' (11-15). If the value of the attribute 'document architecture class' is 'formatted', and the attributes 'generic layout structure' and 'specific layout structure', but not 'specific logical structure' and 'generic logical structure' are specified, the value of the attribute 'generic layout structure' is either 'partial generator set' or 'factor set' (16-21).

Definition 2.20

1 $\forall prof$
2 $({}_{0}\,\mathrm{IsDocumentProfilePart2}(prof)$ *iff*
3 $\mathrm{IsProfileAttributeSetPart2}^{2.63}(prof)$ *and*
4 'document architecture class' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
5 C ^$prof$ • 'document architecture class' $\in$ ['formatted';'processable';'formatted processable'] *and*
6 $({}_{1}({}_{2}$C ^$prof$ • 'document architecture class' = 'processable' *and*
7 'specific logical structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
8 'specific layout structure' $\notin \mathrm{NAMS}^{1.18}(prof)$ *and*
9 $({}_{3}$'generic layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *impl*
10 C ^$prof$ • 'generic layout structure' = 'complete generator set'$_{3}$)$_{2}$) *or*
11 $({}_{4}$C ^$prof$ • 'document architecture class' = 'formatted processable' *and*
12 'specific logical structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
13 'specific layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
14 'generic layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
15 C ^$prof$ • 'generic layout structure' = 'complete generator set'$_{4}$) *or*
16 $({}_{5}$C ^$prof$ • 'document architecture class' = 'formatted' *and*
17 'specific logical structure' $\notin \mathrm{NAMS}^{1.18}(prof)$ *and*
18 'generic logical structure' $\notin \mathrm{NAMS}^{1.18}(prof)$ *and*
19 'specific layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *and*
20 $({}_{6}$'generic layout structure' $\in \mathrm{NAMS}^{1.18}(prof)$ *impl*
21 C ^$prof$ • 'generic layout structure' $\in$ ['partial generator set';'factor set']$_{6}$)$_{5}$)$_{1}$)$_{0}$)

Semiformal Description 2.23

Predicate "is a composite logical object class description" (3.5.6)

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', '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 respective values (14-15). None of the attribute values is defaultable (13).

Definition 2.23

```
1   ∀ cst
2   (0 IsCompositeLogicalObjectClassDescription(cst) iff
3     IsLogicalRootClassDescription^2.22(cst) or
4     (1 IsAttributeSet^2.62(cst) and
5       NAMS^1.18(cst) ⊇ ['object class identifier';   'object type'] and
6       NAMS^1.18(cst) ⊆ ['application comments';  'bindings';
7                         'default value lists';      'generator for subordinates';
8                         'layout style';             'object class identifier';
9                         'object type';              'protection';
10                        'resource';                 'user-readable comments';
11                        'user-visible name'] and
12      ∀ a ∈̂ ^cst •
13      (2 not IsPlaceholder^1.19(C a) and
14        (3 N a = 'object class identifier' impl IsLogicalObjectClassId^2.81(C a) 3) and
15        (4 N a = 'object type' impl C a = 'composite logical object'4)2)1)0)
```

Semiformal Description 2.26

Predicate "is a layout root class description" (3.5.7)

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', '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 (16-17). None of the attribute values is defaultable (11). 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.

NOTE — The required correspondance 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 $(_0$IsLayoutRootClassDescription(cst) *iff*
3 IsAttributeSet$^{2.62}(cst)$ *and*
4 NAMS$^{1.18}(cst) \supseteq$ ['object class identifier'; 'object type'] *and*
5 NAMS$^{1.18}(cst) \subseteq$ ['application comments'; 'balance';
6 'bindings'; 'default value lists';
7 'generator for subordinates'; 'object class identifier';
8 'object type'; 'resource';
9 'user-readable comments'; 'user-visible name'] *and*
10 $\forall\, a \in \hat{}cst \bullet$
11 $(_1$ *not* IsPlaceholder$^{1.19}(C\ a)$ *and*
12 $(_2 N\ a =$ 'balance' *impl*
13 $(_3 C\ a =$ 'null' *or*
14 $(_4$ 'generator for subordinates' $\in$ NAMS$^{1.18}(cst)$ *and*
15 $\forall\, so \in C\ \hat{}cst \bullet$ 'balance' $\bullet$ $((C\ so)$IsClassIdImSubIn$^{2.142}(cst))\ {}_4)_3)_2)$ *and*
16 $(_5 N\ a =$ 'object class identifier' *impl* $C\ a = [\ \rightarrow 0 \rightarrow\ ]\ {}_5)$ *and*
17 $(_6 N\ a =$ 'object type' *impl* $C\ a =$ 'document layout root' $_6)_1)$ *and*
18 *not* (cst)DescribesClassImSubOf$^{2.150}(cst)_0)$

Semiformal Description 2.28

Predicate "is a composite page class description" (3.5.7)

A composite page class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'colour', 'default value lists', 'dimensions', 'generator for subordinates', 'medium type', 'object class identifier', 'object type', 'page position', 'resource', '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 respective values (21, 22). None of the attribute values is defaultable (14). 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.

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

The sub-parameters 'rule A', 'rule B' or 'variable page height' may not be specified for the attribute 'dimensions' (19, 20).

Definition 2.28

```
1   ∀ cst
2   (0 IsCompositePageClassDescription(cst) iff
3     IsAttributeSet^2.62(cst) and
4     NAMS^1.18(cst) ⊇ ['object class identifier';      'object type'] and
5     NAMS^1.18(cst) ⊆ ['application comments';        'balance';
6                       'bindings';                    'colour';
7                       'default value lists';         'dimensions';
8                       'generator for subordinates';  'medium type';
9                       'object class identifier';     'object type';
10                      'page position';               'resource';
11                      'transparency';                'user-readable comments';
12                      'user-visible name'] and
13  ∀ a ∈̂ ^cst •
14  (1 not IsPlaceholder^1.19(C a) and
15    (2 N a = 'balance' impl
16      (3 C a = 'null' or
17        (4 'generator for subordinates' ∈ NAMS^1.18(cst) and
18          ∀ so ∈̂ C ^cst • 'balance' • ((C so)IsClassIdImSubIn^2.142(cst)) 4)3)2) and
19    (5 N a = 'dimensions' impl
20      (6 ∀ b ∈̂ a •• (7 N b ∉ ['rule A'; 'rule B'; 'variable page height'] 7)6)5) and
21    (8 N a = 'object class identifier' impl IsLayoutObjectClassId^2.82(C a) 8) and
22    (9 N a = 'object type' impl C a = 'composite or basic page' 9)1)0)
```

Semiformal Description 2.30

Predicate "is a frame class description" (3.5.7)

A frame class description is an attribute set (nomination) which may contain the attributes 'application comments', 'balance', 'bindings', 'border', 'colour', 'default value lists', 'dimensions', 'generator for subordinates', 'layout path', 'logical source', 'object class identifier', 'object type', 'permitted categories', 'position', 'resource', '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 respective values (22-23). None of the attribute values is defaultable (15). 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.

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

The sub-parameter 'variable page height' may not be specified for the attribute 'dimensions'.

Definition 2.30

```
1   ∀ cst
2   (0 IsFrameClassDescription(cst) iff
3     IsAttributeSet^2.62(cst) and
4     NAMS^1.18(cst) ⊇ ['object class identifier';     'object type'] and
5     NAMS^1.18(cst) ⊆ ['application comments';     'balance';
6                       'bindings';                 'border';
7                       'colour';                   'default value lists';
8                       'dimensions';               'generator for subordinates';
9                       'layout path';              'logical source';
10                      'object class identifier';  'object type';
11                      'permitted categories';     'position';
12                      'resource';                 'transparency';
13                      'user-readable comments';   'user-visible name'] and
14    ∀ a ∈̂ ^cst •
15    (1 not IsPlaceholder^1.19(C a) and
16      (2 N a = 'balance' impl
17        (3 C a = 'null' or
18          (4 'generator for subordinates' ∈ NAMS^1.18(cst) and
19            ∀ so ∈̂ C ^cst • 'balance' • ((C so)IsClassIdImSubIn^2.142(cst)) 4)3)2) and
20      (5 N a = 'dimensions' impl
21        (6 ∀ b ∈̂ a •• (7 N b ≠ 'variable page height' 7)6)5) and
22      (8 N a = 'object class identifier' impl IsLayoutObjectClassId^2.82(C a) 8) and
23      (9 N a = 'object type' impl C a = 'frame' 9)1)0)
```

Semiformal Description 2.33

Predicate "is a composite object class description" (3.5.6, 3.5.7)

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

Definition 2.33

1 $\forall cst$
2 $(_0$ IsCompositeObjectClassDescription(cst) *iff*
3 IsCompositeLogicalObjectClassDescription$^{2.23}(cst)$ *or*
4 IsCompositeLayoutObjectClassDescription$^{2.34}(cst)_0)$

Semiformal Description 2.34

Predicate "is a composite layout object class description" (3.5.7)

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

Definition 2.34

1 $\forall cst$
2 $(_0$ IsCompositeLayoutObjectClassDescription(cst) *iff*
3 IsLayoutRootClassDescription$^{2.26}(cst)$ *or*
4 IsPageSetClassDescription$^{2.27}(cst)$ *or*
5 IsCompositePageClassDescription$^{2.28}(cst)$ *or*
6 IsFrameClassDescription$^{2.30}(cst)_0)$

Semiformal Description 2.35

Predicate "is a basic layout object class description" (3.5.7)

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

Definition 2.35

1 $\forall cst$
2 $(_0$ IsBasicLayoutObjectClassDescription(cst) *iff*
3 IsBasicPageClassDescription$^{2.29}(cst)$ *or*
4 IsBlockClassDescription$^{2.31}(cst)_0)$

Semiformal Description 2.36

Predicate "is a basic object class description" (3.5.6, 3.5.7)

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

Definition 2.36

1 $\forall cst$
2 $(_0$ IsBasicObjectClassDescription(cst) *iff*
3 IsBasicLogicalObjectClassDescription$^{2.24}(cst)$ *or*
4 IsBasicLayoutObjectClassDescription$^{2.35}(cst)_0)$

Semiformal Description 2.39

Predicate "is a logical root description" (3.2.1)

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 'layout style', 'default value lists', and 'object class'. The attributes 'object class' and 'object identifier' have a logical object class identifier and 3 as their respective values (15-16). The attribute 'object type' has 'document logical root' as its value (17-18). If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (19).

Definition 2.39

1 $\forall\, cst$
2 $(_0$ IsLogicalRootDescription(cst) *iff*
3 IsAttributeSet$^{2.62}(cst)$ *and*
4 NAMS$^{1.18}(cst) \supseteq$ ['application comments'; 'bindings';
5 'object identifier'; 'object type';
6 'protection'; 'subordinates';
7 'user-readable comments'; 'user-visible name'] *and*
8 NAMS$^{1.18}(cst) \subseteq$ ['application comments'; 'bindings';
9 'default value lists'; 'layout style';
10 'object class'; 'object identifier';
11 'object type'; 'protection';
12 'subordinates'; 'user-readable comments';
13 'user-visible name'] *and*
14 $\forall\, a \in \hat{}cst \bullet$
15 $(_1 (_2$ N $a =$ 'object class' *impl* IsLogicalObjectClassId$^{2.81}$(C a) $_2)$ *and*
16 $(_3$ N $a =$ 'object identifier' *impl* C $a = [\rightarrow 3 \rightarrow]_3)$ *and*
17 $(_4$ N $a =$ 'object type' *impl*
18 $(_5$ C $a =$ 'document logical root' *or* IsPlaceholder$^{1.19}$(C a) $_5)_4)_1)$ *and*
19 $(_6$ IsPlaceholder$^{1.19}$(C $\hat{}cst \bullet$ 'object type') *impl* 'object class' $\in$ NAMS$^{1.18}(cst)_6)_0)$

Semiformal Description 2.41

Predicate "is a basic logical object description" (3.2.3)

A basic logical object description is an attribute set (nomination) which contains the attributes 'application comments', 'bindings', 'content architecture class', 'object identifier', 'object type', 'protection', 'user-readable comments', 'user-visible name' and, optionally, the attributes 'layout style', 'content portions', 'content generator', 'presentation style' and 'object class'. The attributes 'object class' and 'object identifier' have a logical object class identifier and a logical object identifier as respective values (16, 17). The attribute 'object type' has 'basic logical object' as its value (18, 19). At least one of the attributes 'object class', 'content portions' and 'content generator' must be specified for this constituent (20). If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (21).

Definition 2.41

```
1   ∀ cst
2   (0 IsBasicLogicalObjectDescription(cst) iff
3     IsAttributeSet^2.62(cst) and
4     NAMS^1.18(cst) ⊇ ['application comments';      'bindings';
5                       'content architecture class'; 'object identifier';
6                       'object type';                'protection';
7                       'user-readable comments';     'user-visible name'] and
8     NAMS^1.18(cst) ⊆ ['application comments';      'bindings';
9                       'content architecture class'; 'content generator';
10                      'content portions';           'layout style';
11                      'object class';               'object type';
12                      'object identifier';          'presentation style';
13                      'protection';                 'user-readable comments';
14                      'user-visible name'] and
15    ∀ a ∈̂ ~cst •
16    (1 (2 N a = 'object class' impl IsLogicalObjectClassId^2.81(C a) 2) and
17      (3 N a = 'object identifier' impl IsLogicalObjectId^2.78(C a) 3) and
18      (4 N a = 'object type' impl
19        (5 C a = 'basic logical object' or IsPlaceholder^1.19(C a) 5)4)1) and
20    (6 NAMS^1.18(cst) ∩ ['content portions';'object class';'content generator'] ≠ [ ]6) and
21    (7 IsPlaceholder^1.19(C ~cst • 'object type') impl 'object class' ∈ NAMS^1.18(cst)7)0)
```

Semiformal Description 2.42

Predicate "is a logical object description" (3.2.2, 3.2.3, 5.3.5.4)

A logical object description is either a basic logical object description or a composite logical object description. The "binding value", if specified, is a collection of catenations with two components where the second one is a bindings value expression.

Definition 2.42

```
1   ∀ cst
2   (0 IsLogicalObjectDescription(cst) iff
3     (1 IsBasicLogicalObjectDescription^2.41(cst) or
4       IsCompositeLogicalObjectDescription^2.40(cst)1) and
5     ∀ a ∈̂ ~cst •
6     (2 (3 N a = 'bindings' and not IsPlaceholder^1.19(C a) 3) impl
7       (4 IsNeCol^1.1(C a) and
8         ∀ b ∈ (C a) ∃ l, r
9           (5 b = [ → l → r → ] and IsBindingsValueExpression^2.64(r) 5)4)2)0)
```

Semiformal Description 2.44

Predicate "is a page set description" (3.3.1.2)

A page set description is an attribute set (nomination) which contains the attributes 'application comments', 'balance', 'bindings', 'object identifier', 'object type', 'subordinates', 'user-readable comments', 'user-visible name' and, optionally, the attributes 'default value lists' and 'object class'. The attributes 'object class' and 'object identifier' have a layout object class identifier and a layout object identifier as respective values (17-18). The attribute 'object type' has 'page set' as its value. If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (21). The value of the attribute 'balance' is either 'null' or a sequence of immediately subordinate object identifiers (14-16).

Definition 2.44

1 $\forall cst$
2 $(_0$ IsPageSetDescription(cst) *iff*
3 IsAttributeSet$^{2.62}(cst)$ *and*
4 NAMS$^{1.18}(cst) \supseteq$ ['application comments'; 'balance';
5 'bindings'; 'object identifier';
6 'object type'; 'subordinates';
7 'user-readable comments'; 'user-visible name'] *and*
8 NAMS$^{1.18}(cst) \subseteq$ ['application comments'; 'balance';
9 'bindings'; 'default value lists';
10 'object class'; 'object identifier';
11 'object type'; 'subordinates';
12 'user-readable comments'; 'user-visible name'] *and*
13 $\forall a \in \hat{}cst \bullet$
14 $(_1 (_2$ N $a =$ 'balance' *impl*
15 $(_3$ IsPlaceholder$^{1.19}$(C a) *or* C $a =$ 'null' *or*
16 $\forall so \in$ C $\hat{}cst \bullet$ 'balance' $\bullet$ ((C so)IsIdImSubIn$^{2.143}(cst)$) $_3)_2)$ *and*
17 $(_4$ N $a =$ 'object class' *impl* IsLayoutObjectClassId$^{2.82}$(C a) $_4)$ *and*
18 $(_5$ N $a =$ 'object identifier' *impl* IsLayoutObjectId$^{2.79}$(C a) $_5)$ *and*
19 $(_6$ N $a =$ 'object type' *impl*
20 $(_7$ C $a =$ 'page set' *or* IsPlaceholder$^{1.19}$(C a) $_7)_6)_1)$ *and*
21 $(_8$ IsPlaceholder$^{1.19}$(C $\hat{}cst \bullet$ 'object type') *impl* 'object class' $\in$ NAMS$^{1.18}(cst)_8)_0)$

Semiformal Description 2.46

Predicate "is a basic page description" (3.3.1.3)

A basic page description is an attribute set (nomination) which contains the attributes 'application comments', 'bindings', 'colour', 'content architecture class', 'dimensions', 'medium type', 'object identifier', 'object type', 'page position', 'presentation attributes', 'transparency', 'user-readable comments', 'user-visible name' and, optionally, the attributes 'content portions', 'presentation style', and 'object class'. The attributes 'object class' and 'object identifier' have a layout object class identifier and a layout object identifier as respective values (22, 23). The attribute 'object type' has 'composite or basic page' as its value. At least one of the attributes 'object class' and 'content portions' must be specified for this constituent (26). If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (27). The sub-parameters 'rule A' or 'rule B' may not be specified for the attribute 'dimensions' (20, 21).

Definition 2.46

```
1   ∀ cst
2   (0 IsBasicPageDescription(cst) iff
3     IsAttributeSet2.62(cst) and
4     NAMS1.18(cst) ⊇ ['application comments';     'bindings';
5                      'colour';                   'content architecture class';
6                      'dimensions';               'medium type';
7                      'object identifier';        'object type';
8                      'page position';            'presentation attributes';
9                      'transparency';             'user-readable comments';
10                     'user-visible name'] and
11    NAMS1.18(cst) ⊆ ['application comments';     'bindings';
12                     'colour';                   'content architecture class';
13                     'content portions';         'dimensions';
14                     'medium type';              'object class';
15                     'object identifier';        'object type';
16                     'page position';            'presentation attributes';
17                     'presentation style';       'transparency';
18                     'user-readable comments';   'user-visible name'] and
19    ∀ a ∈̂ ^cst •
20    (1 (2 N a = 'dimensions' impl (3 IsPlaceholder1.19(C a) or
21       (4 ∀ b ∈̂ a •• (5 N b ∉ ['rule A'; 'rule B'] 5)4)3)2) and
22     (6 N a = 'object class' impl IsLayoutObjectClassId2.82(C a) 6) and
23     (7 N a = 'object identifier' impl IsLayoutObjectId2.79(C a) 7) and
24     (8 N a = 'object type' impl
25       (9 C a = 'composite or basic page' or IsPlaceholder1.19(C a) 9)8)1) and
26    (10 NAMS1.18(cst) ∩ ['content portions';'object class'] ≠ [ ]10) and
27    (11 IsPlaceholder1.19(C ^cst • 'object type') impl 'object class' ∈ NAMS1.18(cst)11)0)
```

Semiformal Description 2.47

Predicate "is a page description" (3.3.1.3)

A page description is either a composite page description or a basic page description.

Definition 2.47

```
1   ∀ cst
2   (0 IsPageDescription(cst) iff
3     IsCompositePageDescription2.45(cst) or IsBasicPageDescription2.46(cst)0)
```

Semiformal Description 2.49

Predicate "is a block description" (3.3.1.5)

A block description is an attribute set (nomination) which contains the attributes 'application comments', 'bindings', 'border', 'colour', 'content architecture class', 'dimensions', 'object identifier', 'object type', 'position', 'transparency', 'user-readable comments', 'user-visible name' and, optionally, the attributes 'content portions', 'presentation attributes', 'presentation style' and 'object class'. The attributes 'object class' and 'object identifier' have a layout object class identifier and a layout object identifier as respective values (22, 23). The attribute 'object type' has 'block' as its value. At least one of the attributes 'object class' and 'content portions' must be specified for this constituent (26). If the value of the attribute 'object type' is defaulted the attribute 'object class' must be present (27). The sub-parameters 'rule A', 'rule B' or 'variable page height' may not be specified for the attribute 'dimensions' (19-21).

Definition 2.49

```
1   ∀ cst
2   (0 IsBlockDescription(cst) iff
3     IsAttributeSet^2.62(cst) and
4     NAMS^1.18(cst) ⊇ ['application comments';      'bindings';
5                       'border';                    'colour';
6                       'content architecture class'; 'dimensions';
7                       'object identifier';         'object type';
8                       'position';                  'transparency';
9                       'user-readable comments';    'user-visible name'] and
10    NAMS^1.18(cst) ⊆ ['application comments';      'bindings';
11                      'border';                    'colour';
12                      'content architecture class'; 'content portions';
13                      'dimensions';                'object class';
14                      'object identifier';         'object type';
15                      'position';                  'presentation attributes';
16                      'presentation style';        'transparency';
17                      'user-readable comments';    'user-visible name'] and
18    ∀ a ∈̂ ^cst •
19    (1 (2 N a = 'dimensions' impl (3 IsPlaceholder^1.19(C a) or
20        (4 ∀ b ∈̂ a • •
21          (5 N b ∉ ['rule A'; 'rule B'; 'variable page height'] 5)4)3)2) and
22      (6 N a = 'object class' impl IsLayoutObjectClassId^2.82(C a) 6) and
23      (7 N a = 'object identifier' impl IsLayoutObjectId^2.79(C a) 7) and
24      (8 N a = 'object type' impl
25        (9 C a = 'block' or IsPlaceholder^1.19(C a) 9)8)1) and
26    (10 NAMS^1.18(cst) ∩ ['content portions'; 'object class'] ≠ [ ]10) and
27    (11 IsPlaceholder^1.19(C ^cst • 'object type') impl 'object class' ∈ NAMS^1.18(cst)11)0)
```

Semiformal Description 2.53

Predicate "is a composite layout object description" (3.3)

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

Definition 2.53

1 $\forall cst$
2 $(_0$ IsCompositeLayoutObjectDescription(cst) *iff*
3 IsLayoutRootDescription$^{2.43}(cst)$ *or*
4 IsPageSetDescription$^{2.44}(cst)$ *or*
5 IsCompositePageDescription$^{2.45}(cst)$ *or*
6 IsFrameDescription$^{2.48}(cst)_0$)

Semiformal Description 2.54

Predicate "is a basic layout object description" (3.3)

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

Definition 2.54

1 $\forall cst$
2 $(_0$ IsBasicLayoutObjectDescription(cst) *iff*
3 IsBasicPageDescription$^{2.46}(cst)$ *or*
4 IsBlockDescription$^{2.49}(cst)_0$)

Semiformal Description 2.55

Predicate "is an object description" (3.2, 3.3)

An object description is either a logical object description or a layout object description.

Definition 2.55

1 $\forall cst$
2 $(_0$ IsObjectDescription(cst) *iff*
3 IsLogicalObjectDescription$^{2.42}(cst)$ *or*
4 IsLayoutObjectDescription$^{2.50}(cst)_0$)

Semiformal Description 2.59

Predicate "is a geometric graphics content portion description" (3.1.3)

A geometric graphics content portion description is considered an atomic construct in the formal specification of the document structure. It is formally defined in the formal specification of the geometric graphics content architecture.

Definition 2.59

```
1   ∀cont
2   (0 IsGeometricGraphicsContentPortionDescription(cont) iff
3     IsAtom(cont)0)
```

Semiformal Description 2.60

Predicate "is a layout style" (2.3.5)

A layout style is an attribute set (nomination) which contains the attribute 'layout style identifier' and, optionally, the attributes 'application comments', 'block alignment', 'concatenation', 'fill order', 'indivisibility', 'layout category', 'layout object class', 'new layout object', 'offset', 'same layout object', 'separation', 'synchronization', 'user-readable comments' and 'user-visible name'. The attributes 'layout object class' and 'layout style identifier' have a layout object class identifier and a layout style identifier as their respective values (15-16). None of the attribute values is defaultable (14).

Definition 2.60

```
1   ∀cst
2   (0 IsLayoutStyle(cst) iff
3     IsAttributeSet^2.62(cst) and
4     NAMS^1.18(cst) ⊇ ['layout style identifier'] and
5     NAMS^1.18(cst) ⊆ ['application comments';   'block alignment';
6                       'concatenation';          'fill order';
7                       'indivisibility';         'layout category';
8                       'layout object class';    'layout style identifier';
9                       'new layout object';      'offset';
10                      'same layout object';     'separation';
11                      'synchronization';        'user-readable comments';
12                      'user-visible name'] and
13    ∀a ∈ ^cst •
14    (1 not IsPlaceholder^1.19(C a) and
15      (2N a = 'layout object class' impl IsLayoutObjectClassId^2.82(C a) 2) and
16      (3N a = 'layout style identifier' impl IsLayoutStyleId^2.107(C a) 3)1)0)
```

Semiformal Description 2.61

Predicate "is a presentation style" (2.3.5)

A presentation style is an attribute set (nomination) which contains the attribute 'presentation style identifier' and, optionally, the attributes 'border', 'colour' 'presentation attributes', 'transparency' 'user-readable comments' and 'user-visible name'. The attribute 'presentation style identifier' has a presentation style identifier as its values (11). None of the attribute values is defaultable (10).

Definition 2.61

1 $\forall cst$
2 $(_0$IsPresentationStyle(cst) *iff*
3 IsAttributeSet$^{2.62}(cst)$ *and*
4 NAMS$^{1.18}(cst) \supseteq$ ['presentation style identifier'] *and*
5 NAMS$^{1.18}(cst) \subseteq$ ['border'; 'colour';
6 'presentation attributes'; 'presentation style identifier';
7 'transparency'; 'user-readable comments';
8 'user-visible name'] *and*
9 $\forall a \hat{\in} \hat{}cst \bullet$
10 $(_1$ *not* IsPlaceholder$^{1.19}$(C a) *and*
11 N a = 'presentation style identifier' *impl* IsPresentationStyleId$^{2.117}$(C$a_{)_1})_0$)

7.3 Attributes

Semiformal Description 2.62

Predicate "is an attribute set" (clause 5)

In this definition the attribute names are linked with the ranges of the attribute values.

Definition 2.62

```
1   ∀ as
2   (0 IsAttributeSet(as) iff
3     (1 IsNeNom1.2(as) and
4       ∀ a ∈ ~as •
5       (2(3 N a = 'alternative representation' impl
6           IsAlternativeCharacterSetsString2.128(C a) 3) and
7         (4 N a = 'application comments' impl
8           IsOctetString1.10(C a) 4) and
9         (5 N a = 'balance' impl
10          (6 IsPlaceholder1.19(C a) or IsBalanceValue2.101(C a) 6)5) and
11        (7 N a = 'bindings' impl
12          (8 IsPlaceholder1.19(C a) or IsBindingsValue2.96(C a) 8)7) and
13        (9 N a = 'block alignment' impl
14          (10 IsPlaceholder1.19(C a) or IsBlockAlignmentValue2.108(C a) 10)9) and
15        (11 N a = 'border' impl
16          (12 IsPlaceholder1.19(C a) or IsBorderValue2.100(C a) 12)11) and
17        (13 N a = 'coding attributes' impl
18          (14 IsPlaceholder1.19(C a) or IsCodingAttributesValue2.129(C a) 14)13) and
19        (15 N a = 'colour' impl
20          (16 IsPlaceholder1.19(C a) or IsColourValue2.104(C a) 16)15) and
21        (17 N a = 'concatenation' impl
22          (18 IsPlaceholder1.19(C a) or IsConcatenationValue2.109(C a) 18)17) and
23        (19 N a = 'content architecture class' impl
24          (20 IsPlaceholder1.19(C a) or IsContentArchitectureClassValue2.88(C a) 20)19) and
25        (21 N a = 'content generator' impl
26          IsStringExpression2.65(C a) 21) and
27        (22 N a = 'content identifier layout' impl
28          IsLayoutContentPortionId2.118(C a) 22) and
29        (23 N a = 'content identifier logical' impl
30          IsLogicalContentPortionId2.119(C a) 23) and
31        (24 N a = 'content information' impl
32          IsContentInformationValue2.124(C a) 24) and
33        (25 N a = 'content portions' impl
34          IsSeqOfDiffNnInt1.22(C a) 25) and
35        (26 N a = 'default value lists' impl
36          IsDefaultValueListsValue2.97(C a) 26) and
37        (27 N a = 'dimensions' impl
38          (28 IsPlaceholder1.19(C a) or IsDimensionsValue2.99(C a) 28)27) and
39        (29 N a = 'fill order' impl
40          (30 IsPlaceholder1.19(C a) or IsFillOrderValue2.110(C a) 30)29) and
41        (31 N a = 'generator for subordinates' impl
42          IsConstructionExpression2.85(C a) 31) and
43        (32 N a = 'imaging order' impl
44          IsSeqOfDiffNnInt1.22(C a) 32) and
45        (33 N a = 'indivisibility' impl
```

```
46          (34 IsPlaceholder1.19(C a) or IsLayoutObjectReference2.111(C a) 34)33) and
47        (35 N a = 'layout category' impl
48          (36 IsPlaceholder1.19(C a) or C a = 'null' or
49            IsLayoutCatName2.112(C a) 36)35) and
50        (37 N a = 'layout object class' impl
51          (38 IsPlaceholder1.19(C a) or C a = 'null' or
52            IsLayoutObjectClassId2.82(C a) 38)37) and
53        (39 N a = 'layout path' impl
54          (40 IsPlaceholder1.19(C a) or IsLayoutPathValue2.102(C a) 40)39) and
55        (41 N a = 'layout style' impl
56          IsLayoutStyleId2.107(C a) 41) and
57        (42 N a = 'layout style identifier' impl
58          IsLayoutStyleId2.107(C a) 42) and
59        (43 N a = 'logical source' impl
60          IsLogicalObjectClassId2.81(C a) 43) and
61        (44 N a = 'medium type' impl
62          (45 IsPlaceholder1.19(C a) or IsMediumTypeValue2.105(C a) 45)44) and
63        (46 N a = 'new layout object' impl
64          (47 IsPlaceholder1.19(C a) or IsLayoutObjectReference2.111(C a) 47)46) and
65        (48 N a = 'object class identifier' impl
66          IsObjectClassId2.83(C a) 48) and
67        (49 N a = 'object class' impl
68          IsObjectClassId2.83(C a) 49) and
69        (50 N a = 'object identifier' impl
70          IsObjectId2.80(C a) 50) and
71        (51 N a = 'object type' impl
72          (52 IsPlaceholder1.19(C a) or IsObjectTypeValue2.77(C a) 52)51) and
73        (53 N a = 'offset' impl
74          (54 IsPlaceholder1.19(C a) or IsOffsetValue2.114(C a) 54)53) and
75        (55 N a = 'page position' impl
76          (56 IsPlaceholder1.19(C a) or IsPairOfNnInt1.8(C a) 56)55) and
77        (57 N a = 'permitted categories' impl
78          (58 IsPlaceholder1.19(C a) or C a = 'null' or
79            IsSetOfLayoutCatNames2.113(C a) 58)57) and
80        (59 N a = 'position' impl
81          (60 IsPlaceholder1.19(C a) or IsPositionValue2.98(C a) 60)59) and
82        (61 N a = 'presentation attributes' impl
83          (62 IsPlaceholder1.19(C a) or IsPresentationAttributesValue2.92(C a) 62)61) and
84        (63 N a = 'presentation style' impl
85          IsPresentationStyleId2.117(C a) 63) and
86        (64 N a = 'presentation style identifier' impl
87          IsPresentationStyleId2.117(C a) 64) and
88        (65 N a = 'protection' impl
89          (66 IsPlaceholder1.19(C a) or IsProtectionValue2.106(C a) 66)65) and
90        (67 N a = 'resource' impl
91          IsISO6937MSString1.20(C a)67) and
92        (68 N a = 'same layout object' impl
93          (69 IsPlaceholder1.19(C a) or IsSameLayoutObjectValue2.115(C a) 69)68) and
94        (70 N a = 'separation' impl
95          (71 IsPlaceholder1.19(C a) or IsSeparationValue2.116(C a) 71)70) and
96        (72 N a = 'subordinates' impl
97          IsSeqOfDiffNnInt1.22(C a) 72) and
98        (73 N a = 'synchronization' impl
99          (74 IsPlaceholder1.19(C a) or C a = 'null' or IsLogicalObjectId2.78(C a) or
```

```
100          IsObjectOrObjectClassIdExpression^2.70(C a) 74)73) and
101      (75 N a = 'transparency' impl
102          (76 IsPlaceholder^1.19(C a) or IsTransparencyValue^2.103(C a) 76)75) and
103      (77 N a = 'type of coding' impl
104          IsTypeOfCodingValue^2.120(C a) 77) and
105      (78 N a = 'user-readable comments' impl
106          (79 IsPlaceholder^1.19(C a) or IsCharSeq^2.133(C a) 79)78) and
107      (80 N a = 'user-visible name' impl
108          (81 IsPlaceholder^1.19(C a) or IsCharSeq^2.133(C a) 81)80)2)1)0)
```

Semiformal Description 2.63

Predicate "is a profile attribute set of Part 2"

(The formal specification of the attribute set for the document profile has only been carried out here, as far as it is necessary for the formal specification of the document structures described in ISO 8613-2.)

A profile attribute set is a non-empty nomination with the indicated value ranges for the specified attributes.

Definition 2.63

```
1   ∀as
2   (0 IsProfileAttributeSetPart2(as) iff
3     IsNeNom^1.2(as) and
4     ∀a ∈ ^as •
5     (1 (N a ∈ ['generic layout structure'; 'generic logical structure'] impl
6        C a ∈ ['partial generator set'; 'complete generator set'; 'factor set']) and
7       (N a = 'specific layout structure' impl C a = 'present') and
8       (N a = 'specific logical structure' impl C a = 'present') and
9       (N a = 'layout styles' impl C a = 'present') and
10      (N a = 'presentation styles' impl C a = 'present') and
11      (N a = 'document architecture class' impl
12       C a ∈ ['formatted'; 'formatted processable'; 'processable']) 1)0)
```

Semiformal Description 2.64

Predicate "is a bindings value expression" (5.1.3)

A bindings value expression is either a string expression or a numeric expression or an object or object class identifier expression.

Definition 2.64

```
1   ∀v
2   (0 IsBindingsValueExpression(v) iff
3     IsStringExpression^2.65(v) or
4     IsNumericExpression^2.68(v) or
5     IsObjectOrObjectClassIdExpression^2.70(v) 0)
```

Semiformal Description 2.65

Predicate "is a string expression" (5.1.3.1)

A string expression is a sequence of one or more substring expressions.

Definition 2.65

```
1   ∀ v
2   (₀ IsStringExpression(v) iff
3     (₁ IsNeCat^1.3(v) and
4     ∀ m ∈̂ ~v • (IsAtomicStringExpression^2.66(C m))₁)₀)
```

Semiformal Description 2.66

Predicate "is an atomic string expression" (5.1.3.1)

An atomic string expression is either an octet string (string literal) or a binding reference or a nomination, where the name is a string function and the component is a numeric expression.

Definition 2.66

```
1   ∀ v
2   (₀ IsAtomicStringExpression(v) iff
3     IsOctetString^1.10(v) or IsBindingReference^2.73(v) or
4     ∃ n, c
5     (₁ v = [n : c] and
6       IsStringFunction^2.67(n) and IsNumericExpression^2.68(c)₁)₀)
```

Semiformal Description 2.67

Predicate "is a string function" (5.1.3.1)

A string function is one of the following: 'MAKE-STRING', 'UPPER-ALPHA', 'LOWER-ALPHA', 'UPPER-ROMAN', 'LOWER-ROMAN'.

Definition 2.67

```
1   ∀ v
2   (₀ IsStringFunction(v) iff
3     v ∈ ['MAKE-STRING'; 'UPPER-ALPHA'; 'LOWER-ALPHA'; 'UPPER-ROMAN';
4          'LOWER-ROMAN']₀)
```

Semiformal Description 2.68

Predicate "is a numeric expression" (5.1.3.2)

A numeric expression is either an integer (numeric literal) or a binding reference or a nomination, where the name is a numeric function and the component is either a numeric expression (if the numeric function is 'INCREMENT' or 'DECREMENT') or an object identifier or an object or object class identifier expression (if the numeric function is 'ORDINAL').

Definition 2.68

```
1   ∀v
2   (0 IsNumericExpression(v) iff
3     IsInt(v) or IsBindingReference^2.73(v) or
4     ∃n,c
5     (1 v = [n : c] and IsNumericFunction^2.69(n) and
6       (2 n ∈ ['INCREMENT';'DECREMENT'] impl
7         IsNumericExpression^2.68(c)2) and
8       (3 n = 'ORDINAL' impl
9         (4 IsObjectId^2.80(c) or IsObjectOrObjectClassIdExpression^2.70(c) 4)3)1)0)
```

Semiformal Description 2.69

Predicate "is a numeric function" (5.1.3.2)

A numeric function is one of the following: 'INCREMENT', 'DECREMENT', 'ORDINAL'.

Definition 2.69

```
1   ∀v
2   (0 IsNumericFunction(v) iff
3     v ∈ ['INCREMENT';'DECREMENT';'ORDINAL']0)
```

Semiformal Description 2.70

Predicate "is an object or object class identifier expression"(5.1.3.3)

An object or object class identifier expression is either the selection function 'CURRENT-OBJECT' or a nomination where the name is an object or object class selection function (except 'CURRENT-OBJECT') and the component is an object or object class identifier expression (if the selection function is 'SUPERIOR-OBJECT' or 'PRECEDING-OBJECT'), or a current instance function (if the selection function is 'CURRENT-INSTANCE'). (This is a recursive definition.)

Definition 2.70

```
1   ∀v
2   (0 IsObjectOrObjectClassIdExpression(v) iff
3     v = 'CURRENT-OBJECT' or
4     ∃n,c
5     (1 v = [n : c] and
6       IsObjectOrObjectClassSelectionFunction^2.71(n) and
7       n ≠ 'CURRENT-OBJECT' and
8       (2 n ∈ ['SUPERIOR-OBJECT';'PRECEDING-OBJECT'] impl
9         IsObjectOrObjectClassIdExpression^2.70(c)2) and
10      (3 n = 'CURRENT-INSTANCE' impl IsCurrentInstanceFunction^2.76(c)3)1)0)
```

Semiformal Description 2.71

Predicate "is an object or object class selection function" (5.1.3.3)

An object or object class selection function is one of the following: 'CURRENT-OBJECT', 'CURRENT-INSTANCE', 'SUPERIOR-OBJECT', 'PRECEDING-OBJECT'.

Definition 2.71

1 $\forall v$
2 $(_0$ IsObjectOrObjectClassSelectionFunction(v) *iff*
3 $v \in$ ['CURRENT-OBJECT'; 'CURRENT-INSTANCE'; 'SUPERIOR-OBJECT';
4 'PRECEDING-OBJECT'$]_0)$

Semiformal Description 2.72

Predicate "is a binding reference expression" (5.1.3.4)

A binding reference expression is either an object identifier or an object class identifier or a binding selection function.

Definition 2.72

1 $\forall v$
2 $(_0$ IsBindingReferenceExpression(v) *iff*
3 IsObjectId$^{2.80}(v)$ *or* IsObjectClassId$^{2.83}(v)$ *or*
4 IsBindingSelectionFunction$^{2.75}(v)_0)$

Semiformal Description 2.73

Predicate "is a binding reference" (5.1.3.4)

A binding reference is a pair where the first component is a binding reference expression and the second one is a binding identifier.

Definition 2.73

1 $\forall v$
2 $(_0$ IsBindingReference(v) *iff*
3 $\exists l, r$
4 $(_1 v = [\rightarrow l \rightarrow r \rightarrow]$ *and*
5 IsBindingReferenceExpression$^{2.72}(l)$ *and* IsBindingName$^{2.74}(r)_1)_0)$

Semiformal Description 2.74

Predicate "is a binding name" (5.1.3.4)

A binding name is a string of characters from the minimum subrepertoire of ISO 6937-2.

Definition 2.74

1 $\forall v$
2 $(_0$ IsBindingName(v) *iff*
3 IsISO6937MSString$^{1.20}(v)_0)$

Semiformal Description 2.75

Predicate "is a binding selection function" (5.1.3.4)

A binding selection function is either 'CURRENT-OBJECT' or a nomination where the name is 'SUPERIOR' or 'PRECEDING' (in this case the component is an object or object class identifier expression) or the name is 'CURRENT-INSTANCE' (in this case the component is a current instance function).

Definition 2.75

1 $\forall v$
2 $(_0$IsBindingSelectionFunction(v) *iff*
3 $v =$ 'CURRENT-OBJECT' *or*
4 $\exists f$
5 $(_1 (_2 v =$ ['SUPERIOR' : f] *and* IsObjectOrObjectClassIdExpression$^{2.70}(f)_{\ 2}$) *or*
6 $(_3 v =$ ['PRECEDING' : f] *and* IsObjectOrObjectClassIdExpression$^{2.70}(f)_{\ 3}$) *or*
7 $(_4 v =$ ['CURRENT-INSTANCE' : f] *and* IsCurrentInstanceFunction$^{2.76}(f)_{\ 4})_1)_0$)

Semiformal Description 2.76

Predicate "is a current instance function" (3.5, 5.1)

A current instance function is a pair where the first component is either an object class identifier or an object type value, and the second one is either an object identifier or an object class identifier or an object or object class identifier expression.

Definition 2.76

1 $\forall v$
2 $(_0$IsCurrentInstanceFunction(v) *iff*
3 $\exists l, r$
4 $(_1 v = [\rightarrow l \rightarrow r \rightarrow]$ *and*
5 $(_2$IsObjectClassId$^{2.83}(l)$ *or* IsObjectTypeValue$^{2.77}(l)_2$) *and*
6 $(_3$IsObjectId$^{2.80}(r)$ *or* IsObjectClassId$^{2.83}(r)$ *or*
7 IsObjectOrObjectClassIdExpression$^{2.70}(r)_3)_1)_0$)

Semiformal Description 2.77

Predicate "is an object type value" (5.3.1.1)

An object type value is one of the following: 'basic logical object', 'block', 'composite logical object', 'document layout root', 'document logical root', 'frame', 'composite or basic page', 'page set'.

Definition 2.77

1 $\forall v$
2 $(_0$IsObjectTypeValue(v) *iff*
3 $v \in$ ['basic logical object'; 'block'; 'composite logical object';
4 'document layout root'; 'document logical root'; 'frame';
5 'composite or basic page'; 'page set'$]_0$)

Semiformal Description 2.78

Predicate "is a logical object identifier" (5.3.1.2)

A logical object identifier is a sequence of non-negative integers where the first integer is 3.

Definition 2.78

1 $\forall v$
2 $(_{o}\,\mathrm{IsLogicalObjectId}(v)$ *iff*
3 $\quad \mathrm{IsSeqOfNnInt}^{1.23}(v)$ *and* $\mathrm{HEAD}^{1.13}(v) = 3_{o})$

Semiformal Description 2.79

Predicate "is a layout object identifier" (5.3.1.2)

A layout object identifier is a sequence of non-negative integers where the first integer is 1.

Definition 2.79

1 $\forall v$
2 $(_{o}\,\mathrm{IsLayoutObjectId}(v)$ *iff*
3 $\quad \mathrm{IsSeqOfNnInt}^{1.23}(v)$ *and* $\mathrm{HEAD}^{1.13}(v) = 1_{o})$

Semiformal Description 2.80

Predicate "is an object identifier" (5.3.1.2)

An object identifier is either a logical object identifier or a layout object identifier.

Definition 2.80

1 $\forall v$
2 $(_{o}\,\mathrm{IsObjectId}(v)$ *iff*
3 $\quad \mathrm{IsLogicalObjectId}^{2.78}(v)$ *or* $\mathrm{IsLayoutObjectId}^{2.79}(v)_{o})$

Semiformal Description 2.81

Predicate "is a logical object class identifier" (5.3.1.3)

A logical object class identifier is a sequence of non-negative integers where the first integer is 2.

Definition 2.81

1 $\forall v$
2 $(_{o}\,\mathrm{IsLogicalObjectClassId}(v)$ *iff*
3 $\quad \mathrm{IsSeqOfNnInt}^{1.23}(v)$ *and* $\mathrm{HEAD}^{1.13}(v) = 2_{o})$

Semiformal Description 2.82

Predicate "is a layout object class identifier" (5.3.1.3)

A layout object class identifier is a sequence of non-negative integers where the first integer is 0.

Definition 2.82

1 $\forall v$
2 $(_{o}\,\text{IsLayoutObjectClassId}(v)$ *iff*
3 $\quad\text{IsSeqOfNnInt}^{1.23}(v)$ *and* $\text{HEAD}^{1.13}(v) = 0_{o})$

Semiformal Description 2.83

Predicate "is an object class identifier" (5.3.1.3)

An object class identifier is either a logical object class identifier or a layout object class identifier.

Definition 2.83

1 $\forall v$
2 $(_{o}\,\text{IsObjectClassId}(v)$ *iff*
3 $\quad\text{IsLogicalObjectClassId}^{2.81}(v)$ *or* $\text{IsLayoutObjectClassId}^{2.82}(v)_{o})$

Semiformal Description 2.84

Predicate "is a sequence of object class identifiers"

Each element of a sequence of object class identifiers is an object class identifier.

Definition 2.84

1 $\forall v$
2 $(_{o}\,\text{IsSeqOfObjectClassId}(v)$ *iff*
3 $\quad\text{IsCat}(v)$ *and* $\forall m \in \hat{}v \bullet (\text{IsObjectClassId}^{2.83}(\text{C } m))_{o})$

Semiformal Description 2.85

Predicate "is a construction expression" (5.3.2.1)

A construction expression is either a construction type or a construction term.

Definition 2.85

1 $\forall m$
2 $(_{o}\,\text{IsConstructionExpression}(m)$ *iff*
3 $\quad\text{IsConstructionType}^{2.86}(m)$ *or* $\text{IsConstructionTerm}^{2.87}(m)_{o})$

Semiformal Description 2.86

Predicate "is a construction type" (5.3.2.1)

A construction type is a sequence of two elements where the first one is either 'sequence', 'aggregate' or 'choice' and the second one is a non-empty sequence, whose elements are construction terms.

Definition 2.86

```
1   ∀m
2   (0 IsConstructionType(m) iff
3     ∃l, q
4     (1 m = [ → l → q → ] and
5       l ∈ ['sequence'; 'aggregate'; 'choice'] and
6       IsNeCat^1.3(q) and ∀t ∈ ^q • (IsConstructionTerm^2.87(C t)) 1)0)
```

Semiformal Description 2.87

Predicate "is a construction term" (5.3.2.1)

A construction term is either a sequence with one element where the element is a construction type, or a construction term is a sequence of two elements where the first one is either 'opt', 'reqrep' or 'optrep' and the second one is either an object class identifier or a construction type.

Definition 2.87

```
1   ∀m
2   (0 IsConstructionTerm(m) iff
3     (1 ∃q (2 m = [ → q → ] and IsConstructionType^2.86(q) 2) or
4       ∃l, q
5       (3 m = [ → l → q → ] and
6         l ∈ ['opt'; 'reqrep'; 'optrep'] and
7         (IsObjectClassId^2.83(q) or IsConstructionType^2.86(q)) 3)1)0)
```

Semiformal Description 2.88

Predicate "is a content architecture class value" (5.3.4.1)

The value of the attribute 'content architecture class' is either a character content architecture class value, a raster graphics content architecture class value or a geometric graphics content architecture class value.

Definition 2.88

```
1   ∀v
2   (0 IsContentArchitectureClassValue(v) iff
3     (1 IsCharacterContentArchitectureClassValue^2.89(v) or
4       IsRasterGraphicsContentArchitectureClassValue^2.90(v) or
5       IsGeometricGraphicsContentArchitectureClassValue^2.91(v) 1)0)
```

Semiformal Description 2.89

Predicate "is a character content architecture class value" (5.3.4.1)

A character content architecture class value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex C.

Definition 2.89

```
1  ∀v
2  (0 IsCharacterContentArchitectureClassValue(v) iff
3    IsAtom(v)0)
```

Semiformal Description 2.90

Predicate "is a raster graphics content architecture class value" (5.3.4.1)

A raster graphics content architecture class value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex D.

Definition 2.90

```
1  ∀v
2  (0 IsRasterGraphicsContentArchitectureClassValue(v) iff
3    IsAtom(v)0)
```

Semiformal Description 2.91

Predicate "is a geometric graphics content architecture class value" (5.3.4.1)

A geometric graphics content architecture class value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex E.

Definition 2.91

```
1  ∀v
2  (0 IsGeometricGraphicsContentArchitectureClassValue(v) iff
3    IsAtom(v)0)
```

Semiformal Description 2.92

Predicate "is a presentation attributes value" (5.4.4)

The value of the attribute 'presentation attributes' is a collection where each element is either a character content presentation attribute, a raster graphics content presentation attribute or geometric graphics content presentation attribute.

Definition 2.92

```
1  ∀v
2  (0 IsPresentationAttributesValue(v) iff
3    IsCol(v) and
4    ∀a ∈ v
5    (1 IsCharacterContentPresentationAttribute^2.93(a) or
6      IsRasterGraphicsContentPresentationAttribute^2.94(a) or
7      IsGeometricGraphicsContentPresentationAttribute^2.95(a) 1)0)
```

Semiformal Description 2.93

Predicate "is a character content presentation attribute" (5.4.4)

A character content presentation attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex C.

Definition 2.93

1 $\forall v$
2 $(_{o}\,\text{IsCharacterContentPresentationAttribute}^{2.93}(v)$ *iff*
3 $\text{IsAtom}(v)_{o})$

Semiformal Description 2.94

Predicate "is a raster graphics content presentation attribute" (5.4.4)

A raster graphics content presentation attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex D.

Definition 2.94

1 $\forall v$
2 $(_{o}\,\text{IsRasterGraphicsContentPresentationAttribute}^{2.94}(v)$ *iff*
3 $\text{IsAtom}(v)_{o})$

Semiformal Description 2.95

Predicate "is a geometric graphics content presentation attribute" (5.4.4)

A geometric graphics content presentation attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex E.

Definition 2.95

1 $\forall v$
2 $(_{o}\,\text{IsGeometricGraphicsContentPresentationAttribute}^{2.95}(v)$ *iff*
3 $\text{IsAtom}(v)_{o})$

Semiformal Description 2.96

Predicate "is a bindings value" (5.3.5.4)

The value of the attribute 'bindings' is a set (3) where each element of the set is a catenation of two components (5,6). The first component is an ISO 6937 character string (6) which is unique within the set (9, 10). The second component is either an attribute value expression, an integer, a octet string or an object identifier (7, 8).

Definition 2.96

1 $\forall v$
2 $({}_{0}\,\text{IsBindingsValue}(v)$ *iff*
3 $\text{IsNeCol}^{1.1}(v)$ *and*
4 $\forall a \in v\ \exists f, l$
5 $({}_{1}\,a = [\rightarrow f \rightarrow l \rightarrow]$ *and*
6 $\text{IsISO6937MSString}^{1.20}(f)$ *and*
7 $({}_{2}\,\text{IsBindingsValueExpression}^{2.64}(l)$ *or* $\text{IsOctetString}^{1.10}(l)$ *or*
8 $\text{IsInt}(l)$ *or* $\text{IsObjectId}^{2.80}(l)\ {}_{2})_{1})$ *and*
9 $\forall p, q \in v$
10 $({}_{3}\,p \neq q$ *impl* $\text{HEAD}^{1.13}(p) \neq \text{HEAD}^{1.13}(q)_{3})_{0})$

Semiformal Description 2.97

Predicate "is a default value lists value" (5.3.5.5)

A value of the attribute 'default value lists' is a non-empty nomination where the name is either 'composite logical object', 'basic logical object' 'composite or basic page', 'frame' or 'block' and the component is an attribute set, whose name set depends on the name. For 'frame' or 'block' the component can also be a set of attribute sets.

Definition 2.97

```
1   ∀v
2   (0 IsDefaultValueListsValue(v) iff
3     IsNeNom1.2(v) and
4     NAMS1.18(v) ⊆ ['composite logical object'; 'basic logical object';
5                    'composite or basic page'; 'frame'; 'block'] and
6     ∀a ∈̂ v •
7     (1 (2 N a = 'composite logical object' impl
8         (3 IsAttributeSet2.62(C a) and
9           NAMS1.18(C a) ⊆ ['layout style'; 'protection']3)2) and
10      (4 N a = 'basic logical object' impl
11        (5 IsAttributeSet2.62(C a) and
12          NAMS1.18(C a) ⊆ ['content architecture class'; 'layout style';
13                           'presentation style'; 'protection']5)4) and
14      (6 N a = 'composite or basic page' impl
15        (7 IsAttributeSet2.62(C a) and
16          NAMS1.18(C a) ⊆ ['colour'; 'content architecture class'; 'dimensions'; 'medium type';
17                           'page position'; 'presentation attributes'; 'presentation style';
18                           'transparency']7)6) and
19      (8 N a = 'frame' impl
20        (9 (10 IsAttributeSet2.62(C a) and
21            NAMS1.18(C a) ⊆ ['colour'; 'border'; 'dimensions'; 'layout path';
22                             'permitted categories'; 'position'; 'transparency']10) or
23          (11 IsCol(C a) and ∀as ∈ (C a)
24            (12 IsAttributeSet2.62(as) and
25              NAMS1.18(as) ⊆ ['colour'; 'border'; 'dimensions'; 'layout path'; 'permitted categories';
26                              'position'; 'transparency']12)11)9)8) and
27      (13 N a = 'block' impl
28        (14 (15 IsAttributeSet2.62(C a) and
29            NAMS1.18(C a) ⊆ ['border'; 'colour'; 'content architecture class'; 'dimensions';
30                             'position'; 'presentation attributes'; 'presentation style';
31                             'transparency']15) or
32          (16 IsCol(C a) and ∀as ∈ (C a)
33            (17 IsAttributeSet2.62(as) and
34              NAMS1.18(as) ⊆ ['border'; 'colour'; 'content architecture class'; 'dimensions';
35                              'position'; 'presentation attributes'; 'presentation style';
36                              'transparency']17)16)14)13)1)0)
```

Semiformal Description 2.98

Predicate "is a position value" (5.4.1.1)

The value of the attribute 'position' is a nomination where the name is either 'fixed position' or 'variable position' and the component is a nomination again (4–7, 10, 11). For the name 'fixed position' the names are 'horizontal position' or 'vertical position' and the components are non-negative integers (6–9). For 'variable position' the names are 'offset', 'separation', 'alignment' and 'fill order' (10–12). For 'offset' the component is a nomination, where the names are 'leading offset', 'trailing offset', 'left-hand offset' and 'right-hand offset' and the components are non-negative integers (14–19). For 'separation' the component is also a nomination, where the names are 'leading edge', 'trailing edge' and 'centre separation' and the components are non-negative integers (20–25). For 'alignment' the component is either 'right-hand aligned', 'centred' or 'left-hand aligned' (26-28). For 'fill order' the component is either 'normal order' or 'reverse order' (29-31). The parameters and sub-parameters are independently defaultable (9, 15, 19, 21, 27, 30).

Definition 2.98

$\forall v$
$(_0$ IsPositionValue(v) *iff*
$\exists n, c$
$(_1 v = [n : c]$ *and*
$(_2 n \in$ ['fixed position'; 'variable position'] *and*
$(_3 n =$ 'fixed position' *impl*
$(_4$ IsNom(c) *and*
NAMS$^{1.18}(c) =$ ['horizontal position'; 'vertical position'] *and*
$\forall a \hat{\in} \hat{} c \bullet$ (IsPlaceholder$^{1.19}$(C a) *or* IsNnInt$^{1.7}$(C a)) $_4)_3)$ *and*
$(_5 n =$ 'variable position' *impl*
$(_6$ IsNom(c) *and*
NAMS$^{1.18}(c) =$ ['offset'; 'separation'; 'alignment'; 'fill order'] *and*
$\forall a \hat{\in} \hat{} c \bullet$
$(_7 (_8$ N $a =$ 'offset' *impl*
$(_9$ IsPlaceholder$^{1.19}$(C a) *or*
$(_{10}$ IsNeNom$^{1.2}$(C a) *and*
NAMS$^{1.18}$(C a) = ['leading offset'; 'trailing offset'; 'left-hand offset'
'right-hand offset'] *and*
$\forall b \hat{\in} a \bullet$ (IsPlaceholder$^{1.19}$(C b) *or* IsNnInt$^{1.7}$(C b)) $_{10})_9)_8)$ *and*
$(_{11}$ N $a =$ 'separation' *impl*
$(_{12}$ IsPlaceholder$^{1.19}$(C a) *or*
$(_{13}$ IsNeNom$^{1.2}$(C a) *and*
NAMS$^{1.18}$(C a) = ['leading edge'; 'trailing edge';
'centre separation'] *and*
$\forall b \hat{\in} a \bullet$ (IsPlaceholder$^{1.19}$(C b) *or* IsNnInt$^{1.7}$(C b)) $_{13})_{12})_{11})$ *and*
$(_{14}$ N $a =$ 'alignment' *impl*
$(_{15}$ IsPlaceholder$^{1.19}$(C a) *or*
C $a \in$ ['right-hand aligned'; 'centred'; 'left-hand aligned'] $_{15})_{14})$ *and*
$(_{16}$ N $a =$ 'fill order' *impl*
$(_{17}$ IsPlaceholder$^{1.19}$(C a) *or*
C $a \in$ ['normal order'; 'reverse order']$_{17})_{16})_7)_6)_5)_2)_1)_0)$

Semiformal Description 2.99

Predicate "is a dimensions value" (5.4.1.2)

The value of the attribute 'dimensions' is a nomination of two components where the name is either 'horizontal dimension' or 'vertical dimension'. The component is a nomination again, where the name is either 'maximum size' (the corresponding component ist 'applies'), 'fixed dimension' (the corresponding component is then a positive integer) or 'variable page height' (the corresponding component is any integer) or 'rule A' or 'rule B'. In the last two cases the component is once again a nomination where the name is 'minimum dimension' or 'maximum dimension' and the component is a positive integer. The parameters are independently defaultable. The sub-parameter 'variable page height' is not permitted for the parameter 'horizontal dimension'.

Definition 2.99

1 $\forall v$
2 $(_0$ IsDimensionsValue(v) *iff*
3 IsNeNom$^{1.2}(v)$ *and*
4 NAMS$^{1.18}(v)$ = ['horizontal dimension'; 'vertical dimension'] *and*
5 $\forall a \hat{\in} \hat{}v \bullet$
6 $(_1$ IsPlaceholder$^{1.19}$(C a) *or*
7 $(_2 \exists n, c$
8 $(_3$ C a = $[n : c]$ *and*
9 $n \in$ ['fixed dimension'; 'maximum size'; 'variable page height'; 'rule A'; 'rule B'] *and*
10 $(_4 (_5 n$ = 'fixed dimension' *impl* IsNat$(c)_5)$ *and*
11 $(_6 n$ = 'maximum size' *impl* c = 'applies'$_6)$ *and*
12 $(_7 n$ = 'variable page height' *impl* IsInt$(c)_7)$ *and*
13 $(_8 n \in$ ['rule A'; 'rule B'] *impl*
14 $(_9$ IsNom(c) *and*
15 NAMS$^{1.18}(c) \subseteq$ ['minimum dimension'; 'maximum dimension'] *and*
16 $\forall b \hat{\in} \hat{}c \bullet$ (IsNat(C b)) $_9)_8)_4)$ *and*
17 N a = 'horizontal dimension' *impl* $n \neq$ 'variable page height'$_3)_2)_1)_0)$

Semiformal Description 2.100

Predicate "is a border value" (5.4.1.3)

The value of the attribute 'border' is a nomination of up to four components where the names are either 'left-hand-edge', 'right-hand-edge', 'leading edge' or 'trailing edge'. The component is either 'null' or a nomination of up to three components, where the names are either 'border line width', 'border line type' or 'border freespace width'. The components with the name 'border line width' or 'border freespace width' are non–negative integers. The component with the name 'border line type' is one of the following: 'solid', 'dashed', 'dot', 'dash-dot', 'dash-dot-dot', 'invisible'.

Definition 2.100

```
1   ∀ v
2   (0 IsBorderValue(v) iff
3     IsNom(v) and
4     NAMS1.18(v) ⊆ ['left-hand-edge'; 'right-hand-edge'; 'trailing edge';
5                    'leading edge'] and
6     ∀ a ∈̂ ~v •
7     (1 C a = 'null' or
8       (2 IsNom(C a) and
9         NAMS1.18(C a) ⊆ ['border line width'; 'border line type';
10                         'border freespace width']) and
11        ∀ b ∈̂ a •
12        (3 (4 N b ∈ ['border line width'; 'border freespace width'] impl
13            IsNnInt1.7(C b)4) and
14          (5 N b = 'border line type' impl
15            C b ∈ ['solid'; 'dashed'; 'dot'; 'dash-dot'; 'dash-dot-dot'; 'invisible']5)3)2)1)0)
```

Semiformal Description 2.101

Predicate "is a balance value" (5.4.2.1)

The value of the attribute 'balance' is either 'null' or a sequence of more than one identifiers of immediately subordinate object descriptions (in the case of object descriptions) or object class descriptions (in the case of object class descriptions).

Definition 2.101

```
1   ∀ v
2   (0 IsBalanceValue(v) iff
3     v = 'null' or
4     (1 IsNeCat1.3(v) and LENGTH1.16 (v) > 1 and
5       (2 ∀ a ∈̂ ~v • (IsLayoutObjectId2.79(C a) ) or
6         ∀ a ∈̂ ~v • (IsLayoutObjectClassId2.82(C a) )2)1)0)
```

Semiformal Description 2.102

Predicate "is a layout path value" (5.4.2.2)

A layout path value is one of the following: '0°', '90°', '180°', '270°'.

Definition 2.102

```
1   ∀ v
2   (0 IsLayoutPathValue(v) iff
3     v ∈ ['0°'; '90°'; '180°'; '270°']0)
```

Semiformal Description 2.103

Predicate "is a transparency value" (5.4.3.2)

The value of the attribute 'transparency' is either 'transparent' or 'opaque'.

Definition 2.103

```
1   ∀ v
2   (₀ IsTransparencyValue(v) iff
3     v ∈ ['transparent';'opaque']₀)
```

Semiformal Description 2.104

Predicate "is a colour value" (5.4.3.3)

The value of the attribute 'colour' is either 'colourless' or 'white'.

Definition 2.104

```
1   ∀ v
2   (₀ IsColourValue(v) iff
3     v ∈ ['colourless';'white']₀)
```

Semiformal Description 2.105

Predicate "is a medium type value" (5.4.3.5)

The value of the attribute 'medium type' is a nomination of two components. For the name 'nominal page size' the component is a pair of positive integers, the first one specifying the horizontal, the second one the vertical dimensions of the page. For the name 'side of sheet' the component is either 'recto', 'verso' or 'unspecified'. The parameters are independently defaultable.

Definition 2.105

```
1   ∀ v
2   (₀ IsMediumTypeValue(v) iff
3     IsNeNom^1.2(v) and
4     NAMS^1.18(v) = ['nominal page size';'side of sheet'] and
5     ∀ a ∈̂ v •
6     (₁ (₂ N a = 'nominal page size' impl
7       (₃ IsPlaceholder^1.19(C a) or IsPairOfPosInt^1.9(C a)₃)₂) and
8       (₄ N a = 'side of sheet' impl
9       (₅ IsPlaceholder^1.19(C a) or C a ∈ ['recto';'verso';'unspecified']₅)₄)₁)₀)
```

Semiformal Description 2.106

Predicate "is a protection value" (5.5.1)

A protection value is either 'protected' or 'unprotected'.

Definition 2.106

```
1   ∀ v
2   (₀ IsProtectionValue(v) iff
3     v ∈ ['protected';'unprotected']₀)
```

Semiformal Description 2.107

Predicate "is a layout style identifier" (5.6.1)

A layout style identifier is a pair of two non-negative integers where the first integer is 4.

Definition 2.107

1 $\forall v$
2 $(_o$IsLayoutStyleId(v) *iff*
3 IsPairOfNnInt$^{1.8}(v)$ *and* HEAD$^{1.13}(v) = 4_o)$

Semiformal Description 2.108

Predicate "is a block alignment value" (5.7.1)

A block alignment value is one of the following: 'right-hand aligned', 'left-hand aligned', 'centred', 'null'.

Definition 2.108

1 $\forall v$
2 $(_o$IsBlockAlignmentValue(v) *iff*
3 $v \in$ ['right-hand aligned'; 'left-hand aligned'; 'centred'; 'null']$_o)$

Semiformal Description 2.109

Predicate "is a concatenation value" (5.7.2)

The value of the attribute 'concatenation' is either 'concatenated' or 'non-concatenated'.

Definition 2.109

1 $\forall v$
2 $(_o$IsConcatenationValue(v) *iff*
3 $v \in$ ['concatenated'; 'non-concatenated']$_o)$

Semiformal Description 2.110

Predicate "is a fill order value" (5.7.3)

The value of the attribute 'fill order' is either 'normal order' or 'reverse order'.

Definition 2.110

1 $\forall v$
2 $(_o$IsFillOrderValue(v) *iff*
3 $v \in$ ['normal order'; 'reverse order']$_o)$

Semiformal Description 2.111

Predicate "is a layout object reference" (5.7.4, 5.7.7)

A layout object reference is either **'object type page'** or **'null'** or a layout category name or a layout object class identifier.

Definition 2.111

1 $\forall v$
2 $(_0$ IsLayoutObjectReference(v) *iff*
3 $v \in$ ['object type page'; 'null'] *or*
4 IsLayoutCatName$^{2.112}(v)$ *or* IsLayoutObjectClassId$^{2.82}(v)$ $_0)$

Semiformal Description 2.112

Predicate "is a layout category name" (5.7.5)

A layout category name is a string of characters from the minimum subrepertoire of ISO 6937-2.

Definition 2.112

1 $\forall v$
2 $(_0$ IsLayoutCatName(v) *iff*
3 IsISO6937MSString$^{1.20}(v)_0)$

Semiformal Description 2.113

Predicate "is a set of layout category names"

Each element of a set of layout category names is a layout category name.

Definition 2.113

1 $\forall v$
2 $(_0$ IsSetOfLayoutCatNames(v) *iff*
3 IsCol(v) *and*
4 $\forall cst \in v$ (IsLayoutCatName$^{2.112}(cst))_0)$

Semiformal Description 2.114

Predicate "is an offset value" (5.7.8)

The value of the attribute **'offset'** is a nomination of four components with the names **'leading offset'**, **'trailing offset'**, **'left-hand offset'** and **'right-hand offset'**. The components are non-negative numbers. The parameters are independently defaultable.

Definition 2.114

1 $\forall v$
2 $(_0$ IsOffsetValue(v) *iff*
3 IsNeNom$^{1.2}(v)$ *and*
4 NAMS$^{1.18}(v)$ = ['leading offset'; 'trailing offset'; 'left-hand offset';
5 'right-hand offset'] *and*
6 $\forall a \hat{\in} \hat{} v \bullet$ (IsPlaceholder$^{1.19}$(C a) *or* IsNnInt$^{1.7}$(C a))$_0)$

Semiformal Description 2.115

Predicate "is a same layout object value" (5.7.9)

The value of the attribute 'same layout object' is a nomination of up to two components. For the name 'logical object' the component is either 'null', a logical object identifier or an object or object class identifier expression. If the component is not 'null' the name 'layout object' must be specified. For the name 'layout object' the component is either 'object type page', a layout object class identifier or a layout category name.

Definition 2.115

```
1   ∀ v
2   (0 IsSameLayoutObjectValue(v) iff
3     IsNeNom1.2(v) and
4     NAMS1.18(v) ⊆ ['layout object'; 'logical object'] and
5     ∀ a ∈̂ ^v •
6     (1 N a = 'logical object' impl
7       (2 (3 C a = 'null' or IsLogicalObjectId2.78(C a) or IsObjectOrObjectClassIdExpression2.70(C a)3) and
8         C a ≠ 'null' impl 'layout object' ∈ NAMS1.18(v) 2) and
9       N a = 'layout object' impl
10      (4 C a = 'object type page' or IsLayoutObjectClassId2.82(C a) or IsLayoutCatName2.112(C a)4)1)0)
```

Semiformal Description 2.116

Predicate "is a separation value" (5.7.10)

The value of the attribute 'separation' is a nomination, i.e., a set of pairs (*name*, *value*). The *name* is either 'leading edge', 'trailing edge' or 'centre separation'; the *value* is a non-negative integer. The parameters are independently defaultable.

Definition 2.116

```
1   ∀ v
2   (0 IsSeparationValue(v) iff
3     IsNeNom1.2(v) and
4     NAMS1.18(v) = ['leading edge'; 'trailing edge'; 'centre separation'] and
5     ∀ a ∈̂ ^v •
6     (1 IsPlaceholder1.19(C a) or IsNnInt1.7(C a)1)0)
```

Semiformal Description 2.117

Predicate "is a presentation style identifier" (5.8.1)

A presentation style identifier is a pair of non-negative integers where the first integer is 5.

Definition 2.117

```
1   ∀ v
2   (0 IsPresentationStyleId(v) iff
3     IsPairOfNnInt1.8(v) and HEAD1.13(v) = 50)
```

Semiformal Description 2.118

Predicate "is a layout content portion identifier" (5.9.1)

A layout content portion identifier is either a layout object identifier or a layout object class identifier whose length is greater than 1.

Definition 2.118

1 $\forall v$
2 ($_0$IsLayoutContentPortionId(v) *iff*
3 IsLayoutObjectId$^{2.79}$(v) *or* IsLayoutObjectClassId$^{2.82}$(v) *and*
4 LENGTH$^{1.16}$ (v) > 1_0)

Semiformal Description 2.119

Predicate "is a logical content portion identifier" (5.9.1)

A logical content portion identifier is either a logical object identifier or a logical object class identifier whose length is greater than 1.

Definition 2.119

1 $\forall v$
2 ($_0$IsLogicalContentPortionId(v) *iff*
3 IsLogicalObjectId$^{2.78}$(v) *or* IsLogicalObjectClassId$^{2.81}$(v) *and*
4 LENGTH$^{1.16}$ (v) > 1_0)

Semiformal Description 2.120

Predicate "is a type of coding value" (5.9.2)

The value of the attribute 'type of coding' is either a character content type of coding value, a raster graphics content type of coding value or a geometric graphics content type of coding value.

Definition 2.120

1 $\forall v$
2 ($_0$IsTypeOfCodingValue(v) *iff*
3 ($_1$IsCharacterContentTypeOfCodingValue$^{2.121}$(v) *or*
4 IsRasterGraphicsContentTypeOfCodingValue$^{2.122}$(v) *or*
5 IsGeometricGraphicsContentTypeOfCodingValue$^{2.123}$(v) $_1$)$_0$)

Semiformal Description 2.121

Predicate "is a character content type of coding value" (5.9.2)

A character content type of coding value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex C.

Definition 2.121

1 $\forall v$
2 ($_0$IsCharacterContentTypeOfCodingValue(v) *iff*
3 IsAtom(v)$_0$)

Semiformal Description 2.122

Predicate "is a raster graphics content type of coding value" (5.9.2)

A raster graphics content type of coding value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex D.

Definition 2.122

```
1   ∀v
2   (0 IsRasterGraphicsContentTypeOfCodingValue(v) iff
3     IsAtom(v)0)
```

Semiformal Description 2.123

Predicate "is a geometric graphics content type of coding value" (5.9.2)

A geometric graphics content type of coding value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex E.

Definition 2.123

```
1   ∀v
2   (0 IsGeometricGraphicsContentTypeOfCodingValue(v) iff
3     IsAtom(v)0)
```

Semiformal Description 2.124

Predicate "is a content information value" (5.9.3.1)

The value of the attribute 'content information' is either a character content information value, a raster graphics content information value or a geometric graphics content information value.

Definition 2.124

```
1   ∀v
2   (0 IsContentInformationValue(v) iff
3     (1 IsCharacterContentInformationValue2.125(v) or
4       IsRasterGraphicsContentInformationValue2.126(v) or
5       IsGeometricGraphicsContentInformationValue2.127(v) 1)0)
```

Semiformal Description 2.125

Predicate "is a character content information value" (5.9.3.1)

A character content information value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex C.

Definition 2.125

```
1   ∀v
2   (0 IsCharacterContentInformationValue(v) iff
3     IsAtom(v)0)
```

Semiformal Description 2.126

Predicate "is a raster graphics content information value" (5.9.3.1)

A raster graphics content information value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex D.

Definition 2.126

```
1  ∀ v
2  (₀ IsRasterGraphicsContentInformationValue(v) iff
3    IsAtom(v)₀)
```

Semiformal Description 2.127

Predicate "is a geometric graphics content information value" (5.9.3.1)

A geometric graphics content information value is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex E.

Definition 2.127

```
1  ∀ v
2  (₀ IsGeometricGraphicsContentInformationValue(v) iff
3    IsAtom(v)₀)
```

Semiformal Description 2.128

Predicate "is an alternative character sets string" (5.9.3.2)

The value of the attribute 'alternative representation' is an atomic construct, i.e., the internal structure of the value is not considered in the formal specification of the document structure. The permitted values of this attribute depend on the value of the attribute 'alternative character sets' in the document profile.

Definition 2.128

```
1  ∀ v
2  (₀ IsAlternativeCharacterSetsString(v) iff
3    IsAtom(v)₀)
```

Semiformal Description 2.129

Predicate "is a coding attributes value" (5.9.4)

The value of the attribute 'coding attributes' is a collection where each element is either a character content coding attribute, a raster graphics content coding attribute or geometric graphics content coding attribute.

Definition 2.129

```
1  ∀ v
2  (₀ IsCodingAttributesValue(v) iff
3    IsCol(v) and
4    ∀ a ∈ v
5    (₁ IsCharacterContentCodingAttribute^2.130(a) or
6      IsRasterGraphicsContentCodingAttribute^2.131(a) or
7      IsGeometricGraphicsContentCodingAttribute^2.132(a) ₁)₀)
```

Semiformal Description 2.130

Predicate "is a character content coding attribute" (5.9.4)

A character content coding attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex C.

Definition 2.130

```
1   ∀ v
2   (₀ IsCharacterContentCodingAttribute(v) iff
3     IsAtom(v)₀)
```

Semiformal Description 2.131

Predicate "is a raster graphics content coding attribute" (5.9.4)

A raster graphics content coding attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex D.

Definition 2.131

```
1   ∀ v
2   (₀ IsRasterGraphicsContentCodingAttribute(v) iff
3     IsAtom(v)₀)
```

Semiformal Description 2.132

Predicate "is a geometric graphics content coding attribute" (5.9.4)

A geometric graphics content coding attribute is considered an atomic construct in the formal specification of the document structure. The permitted values are formally defined in annex E.

Definition 2.132

```
1   ∀ v
2   (₀ IsGeometricGraphicsContentCodingAttribute(v) iff
3     IsAtom(v)₀)
```

Semiformal Description 2.133

Predicate "is a character sequence"

A character sequence is a catenation of characters.

Definition 2.133

```
1   ∀ v
2   (₀ IsCharSeq(v) iff
3     IsCat(v) and ∀ m ∈ ˜v • (IsCharacter^2.134(C m))₀)
```

Semiformal Description 2.134

Predicate "is a character"

A character is an element of a defined character set and is considered an atomic construct in the formal specification.

Definition 2.134

1 $\forall z$
2 $(_0$ IsCharacter(z) *iff*
3 IsAtom(z)$_0$)

7.4 Subsidiary definitions

This is the list of all definitions for predicates and functions (operators) which have been factorized from 7.2–7.3 for the sake of readability.

7.4.1 Predicates

Semiformal Description 2.135

Predicate "is described by"

An entity *s* is described by an entity *g* if *s* is a specific description and *g* is a corresponding generic description (3-5) and if each object description *so* in *s* which has an attribute 'object class' (6,7) refers to an object class description *go* in *g* (8-10) with the same value of 'object type' if specified in *so* (11-12). If *so* is a composite object description (13) and the associated *go* has an attribute 'generator for subordinates' (14), *go* specifies through its attribute 'generator for subordinates' a sequence of or a set of sequences of (subordinate) object class identifiers. Among them is that sequence of object class identifiers which is obtained from the object descriptions immediately subordinate to *so* (15). If *so* is a basic logical object description which has no attribute 'content portions' (16) (and therefore refers to a basic logical object class description *go*) the content portions connected with *go* through 'content portions' or the content obtained from *go* through 'content generator' (17) are regarded as the content of *so*. If *so* is a basic layout object description which has no attribute 'content portions' (18) (and therefore refers to a basic layout object class description *go*) the content portions connected with *go* through 'content portions' (19) are regarded as the content of *so*. (If *so* is a basic object description which has an attribute 'content portions', any specific content portion descriptions are allowed.)

Definition 2.135

1 $\forall s, g$
2 $(_0(s)\text{IsDescribedBy}(g)$ *iff*
3 $(_1(_2\text{IsSpecificLogicalDescription}^{2.13}(s)$ *and* $\text{IsGenericLogicalDescription}^{2.10}(g)_2)$
4 *or*
5 $(_3\text{IsSpecificLayoutDescription}^{2.14}(s)$ *and* $\text{IsGenericLayoutDescription}^{2.11}(g)_3)_1)$ *and*
6 $\forall so \in s$
7 $(_4\text{IsObjectDescription}^{2.55}(so)$ *and* 'object class' $\in \text{NAMS}^{1.18}(so)$ *impl*
8 $\exists go \in g$
9 $(_5\text{IsObjectClassDescription}^{2.37}(go)$ *and*
10 $C\,\hat{}\,so\bullet\text{'object class'} = C\,\hat{}\,go\bullet\text{'object class identifier'}$ *and*
11 $(_6$ 'object type' $\in \text{NAMS}^{1.18}(so)$ *impl*
12 $C\,\hat{}\,so\bullet\text{'object type'} = C\,\hat{}\,go\bullet\text{'object type'}_6)$ *and*
13 $(_7\ \text{IsCompositeObjectDescription}^{2.51}(so)$ *and*
14 'generator for subordinates' $\in \text{NAMS}^{1.18}(go)$ *impl*
15 $(_8\text{SEQCLASSIDIMSUB}^{2.156}(s, so, C\,\hat{}\,so\bullet\text{'subordinates'}))\text{IsSeqClassIdImSubIn}^{2.145}(go)_8)$ *and*
16 $(_9\ (_{10}\text{IsBasicLogicalObjectDescription}^{2.41}(so)$ *and* 'content portions' $\notin \text{NAMS}^{1.18}(so)_{10})$ *impl*
17 $(_{11}$ 'content portions' $\in \text{NAMS}^{1.18}(go)$ *or* 'content generator' $\in \text{NAMS}^{1.18}(go)_{11})_9)$ *and*
18 $(_{12}{}_{13}\text{IsBasicLayoutObjectDescription}^{2.54}(so)$ *and* 'content portions' $\notin \text{NAMS}^{1.18}(so)_{13})$ *impl*
19 'content portions' $\in \text{NAMS}^{1.18}(go)_{12})_7)_5)_4)_0)$

Semiformal Description 2.136

Predicate "is completely described by"

An entity *s* is completely described by an entity *g* if *s* is described by *g* which is a complete generic description (4) and if each object description *so* in *s* (5,6) refers through its attribute 'object class' (7) to an object class description *go* in *g* (8-10), which has an attribute 'generator for subordinates' if it is a composite object class description (11,12). If *go* has the attribute 'content generator' specified, *go* and/or *so* have the attribute 'content portions' specified (14–16).

Definition 2.136

1 $\forall s, g$
2 $({}_0(s)\text{IsCompletelyDescribedBy}(g)$ *iff*
3 $(s)\text{IsDescribedBy}^{2.135}(g)$ *and*
4 $\text{IsCompleteGenericDescription}^{2.9}(g)$ *and*
5 $\forall so \in s$
6 $({}_1\text{IsObjectDescription}^{2.55}(so)$ *impl*
7 $({}_2$ *not* $\text{IsBlockDescription}^{2.49}(so)$ *impl* 'object class' $\in \text{NAMS}^{1.18}(so)_2)$ *and*
8 'object class' $\in \text{NAMS}^{1.18}(so)$ *impl*
9 $\exists go \in g$
10 $({}_3\text{IsObjectClassDescription}^{2.37}(go)$ *and*
11 $\text{C}\,\hat{}\,so \bullet$ 'object class' $= \text{C}\,\hat{}\,go \bullet$ 'object class identifier' *and*
12 $({}_4\text{IsCompositeObjectClassDescription}^{2.33}(go)$ *impl*
13 'generator for subordinates' $\in \text{NAMS}^{1.18}(go)_4)$ *and*
14 $({}_5$ 'content generator' $\in \text{NAMS}^{1.18}(go)$ *impl*
15 ('content portions' $\in \text{NAMS}^{1.18}(go)$ *or*
16 'content portions' $\in \text{NAMS}^{1.18}(so))_5)_3)_1)_0)$

Semiformal Description 2.137

Predicate "is content connected"

An entity *csts* is content connected if it is a collection (3) and if each content portion description *cont* in it (4,5) is referenced at least from one basic object description or basic object class description *cst* (6).

Definition 2.137

1 $\forall csts$
2 $({}_0\text{IsContentConnected}(csts)$ *iff*
3 $\text{IsCol}(csts)$ *and*
4 $\forall cont \in csts$
5 $({}_1\text{IsContentPortionDescription}^{2.56}(cont)$ *impl*
6 $\exists cst \in csts((cont)\text{DescribesContPortOf}^{2.153}(cst))_1)_0)$

Semiformal Description 2.138

Predicate "is content complete"

An entity *csts* is content complete if it is a collection (3) and if for all basic object descriptions or basic object class descriptions *cst* in it (4–6) each reference out of the attribute 'content portions' (6–8) is satisfied (9–15).

Definition 2.138

```
1   ∀ csts
2   (0 IsContentComplete(csts) iff
3     IsCol(csts) and
4     ∀ cst ∈ csts
5     (1 (2 (3 IsBasicObjectDescription^2.52(cst) or
6           IsBasicObjectClassDescription^2.36(cst)3) and
7       'content portions' ∈ NAMS^1.18(cst)2) impl
8       ∀ r ∈̂ ^cst • 'content portions' •
9       (4 ∃ cont ∈ csts
10        (5 IsContentPortionDescription^2.56(cont) and
11          (6 C ^cont • 'content identifier logical' =
12            C ^cst • <'object identifier', 'object class identifier'>//[ → C r → ]
13            or
14            C ^cont • 'content identifier layout' =
15            C ^cst • <'object identifier', 'object class identifier'>//[ → C r → ]6)5)4)1)0)
```

Semiformal Description 2.139

Predicate "is class-immediate-subordinates complete"

An entity *g* is complete with respect to the object class descriptions for immediate subordinates if it is a collection (3) and if for all composite object class descriptions *cst* in it (4,5) each reference *r* out of the value *e* of the attribute 'generator for subordinates' is satisfied (7–13).

Definition 2.139

```
1   ∀ g
2   (0 IsClassImSubComplete(g) iff
3     IsCol(g) and
4     ∀ cst ∈ g
5     (1 (2 IsCompositeObjectClassDescription^2.33(cst) and
6         'generator for subordinates' ∈ NAMS^1.18(cst)2) impl
7       (3 ∃ e
8         (4 e = C ^cst • 'generator for subordinates' and
9           IsConstructionExpression^2.85(e) and
10          ∀ r ∈ [id | (id)OccursIn^2.147(e)]
11          (5 ∃ sub ∈ g
12            (6 IsObjectClassDescription^2.37(sub) and
13              C ^sub • 'object class identifier' = r6)5)4)3)1)0)
```

Semiformal Description 2.140

Predicate "is immediate-subordinates complete"

An entity *s* is complete with respect to the object descriptions for immediate subordinates if it is a collection (3) and if for all composite object descriptions *cst* in it (4,5) each reference *r* out of the value of the attribute 'subordinates' (6) is satisfied (7-10).

Definition 2.140

1 $\forall s$
2 $(_0$IsImSubComplete(s) *iff*
3 IsCol(s) *and*
4 $\forall cst \in s$
5 $(_1$IsCompositeObjectDescription$^{2.51}(cst)$ *impl*
6 $\forall r \hat{\in}$ ^cst • 'subordinates' •
7 $(_2 \exists sub \in s$
8 $(_3$IsObjectDescription$^{2.55}(sub)$ *and*
9 C ^sub • 'object identifier' =
10 C ^cst • 'object identifier'//[→ C r →]$_3)_2)_1)_0)$

Semiformal Description 2.141

Predicate "is contained in"

An entity *slay* is contained in an entity *slog* if *slay* is a specific layout description, *slog* is a specific logical description and the set of all content portion descriptions of *slog* is a subset of all content portion descriptions of *slay*.

Definition 2.141

1 $\forall slay, slog$
2 $(_0(slog)$IsContainedIn$(slay)$ *iff*
3 IsSpecificLayoutDescription$^{2.14}(slay)$ *and*
4 IsSpecificLogicalDescription$^{2.13}(slog)$ *and*
5 ALLCONT$^{2.155}(slog) \subseteq$ ALLCONT$^{2.155}(slay)_0)$

Semiformal Description 2.142

Predicate "is an object class identifier for immediate subordinates"

An entity *id* is an object class identifier for immediate subordinates of the entity *gco*, if *gco* has an attribute 'generator for subordinates', whose value *e* is a "construction expression". The predicate OccursIn is True, if the object class identifier *id* occurs in the "construction expression" *e*.

Definition 2.142

1 $\forall id, gco$
2 $(_0(id)$IsClassIdImSubIn(gco) *iff*
3 IsAttributeSet$^{2.62}(gco)$ *and*
4 $\exists a \hat{\in}$ ^gco • $\exists e$
5 $(_1$ N a = 'generator for subordinates' *and* e = C a *and*
6 IsConstructionExpression$^{2.85}(e)$ *and* (id)OccursIn$^{2.147}(e)$ $_1)_0)$

Semiformal Description 2.143

Predicate "is an object identifier of an immediate subordinate"

An entity *id* is an object identifier of an immediate subordinate object of the entity *sco* if it is referred to (via 'subordinates') from the entity *sco*.

Definition 2.143

1 $\forall\, id, sco$
2 $(_0\,(id)\text{IsIdImSubIn}(sco)$ *iff*
3 $\quad\text{IsAttributeSet}^{2.62}(sco)$ *and*
4 $\quad\exists\, n \in \hat{}\,sco \bullet$ 'subordinates' $\bullet$
5 $\quad(id = \text{C}\ \hat{}\,sco \bullet$ 'object identifier' $//[\rightarrow \text{C}\ n \rightarrow])_0)$

Semiformal Description 2.144

Predicate "is an identifier of a content portion"

An entity *id* is an identifier of a content portion within the entity *b*, if it is referred to (via 'content portions') from the entity *b*, i.e., if *id* is the identifier of *b* extended with one of the numbers drawn from the value of 'content portions'.

Definition 2.144

1 $\forall\, id, b$
2 $(_0\,(id)\text{IsIdContIn}(b)$ *iff*
3 $\quad\text{IsAttributeSet}^{2.62}(b)$ *and*
4 $\quad\exists\, n \in \hat{}\,b \bullet$ 'content portions' $\bullet$
5 $\quad(_1\, id = \text{C}\ \hat{}\,b \bullet \langle$'object identifier', 'object class identifier'$\rangle // [\rightarrow \text{C}\ n \rightarrow]_1)_0)$

Semiformal Description 2.145

Predicate "is a sequence of object class identifiers for immediate subordinates"

An entity *idq* is a sequence of object class identifiers for immediate subordinates of the entity *gco*, if *gco* has an attribute 'generator for subordinates' (5.6), whose value *e* is a "construction expression". The sequence *idq* is one of the sequences defined by *e*, i.e., is covered by *e*.

Definition 2.145

1 $\forall\, idq, gco$
2 $(_0\,(idq)\text{IsSeqClassIdImSubIn}(gco)$ *iff*
3 $\quad\text{IsAttributeSet}^{2.62}(gco)$ *and*
4 $\quad\exists\, a \in \hat{}\,gco \bullet \exists\, e$
5 $\quad(_1\,(\text{N}\ a =$ 'generator for subordinates' *and* $e = \text{C}\ a)$ *and*
6 $\quad\text{IsConstructionExpression}^{2.85}(e)$ *and* $(idq)\text{CoveredBy}^{2.146}(e)\ _1)_0)$

Semiformal Description 2.146

Predicate "is covered by"

A sequence *idq* of object class identifiers is covered by the construction expression *e* (which defines a possibly infinite set of such sequences), if *idq* is one of the sequences of object class identifiers defined by *e*. In general, *e* is the value of the attribute **'generator for subordinates'** of an object class description, i.e., a "construction expression".

Definition 2.146

1 $\forall\, idq, e$
2 $(_0\,(idq)\text{CoveredBy}(e)$ *iff*
3 $\quad \text{IsSeqOfObjectClassId}^{2.84}(idq)$ *and*
4 $\quad \text{IsConstructionExpression}^{2.85}(e)_0)$

Semiformal Description 2.147

Predicate "occurs in"

An object class identifier *id* occurs in a construction expression *m* if it appears at least once in the construction expression *m*. In general, *m* is the value of the attribute **'generator for subordinates'** of an object class description.

Definition 2.147

1 $\forall\, id, m$
2 $(_0\,(id)\text{OccursIn}(m)$ *iff*
3 $\quad (_1\,\text{IsConstructionExpression}^{2.85}(m)$ *and*
4 $\quad\quad id \in \text{COLC}^{1.17}\ \hat{}m \downarrow <xs \parallel \text{IsNnInt}^{1.7}(\text{C}\ xs)>\,{}^{\bullet}{}_1)_0)$

Semiformal Description 2.148

Predicate "is initial generic constituent in"

An entity *cst* is an initial generic constituent in an entity *g* if *g* is a collection (3) and *cst* is an object class description in it (3,4) and if there is no entity *k* in *g* from which a reference via **'generator for subordinates'** is made to *cst* (5). Note that a self-reference ($k = cst$, i.e. from the object class description *cst* to itself) is not permitted for initial generic constituents.

Definition 2.148

1 $\forall\, cst, g$
2 $(_0\,(cst)\text{IsInitialGenericIn}(g)$ *iff*
3 $\quad \text{IsCol}(g)$ *and* $cst \in g$ *and*
4 $\quad \text{IsObjectClassDescription}^{2.37}(cst)$ *and*
5 $\quad$ *not* $\exists\, k \in g((cst)\text{DescribesClassImSubOf}^{2.150}(k))_0)$

Semiformal Description 2.149

Predicate "is initial specific constituent in"

An entity *cst* is an initial specific constituent in an entity *s* if *s* is a collection (3) and *cst* is an object description in it (3,4) and if there is no other entity *k* in *s* from which a reference via 'subordinates' is made to *cst* (5). Note that a self-reference (from the object description *cst* to itself) is not possible in a specific description.

Definition 2.149

1 $\forall\, cst, s$
2 $({}_0(cst)\text{IsInitialSpecificIn}(s)$ *iff*
3 $\text{IsCol}(s)$ *and* $cst \in s$ *and*
4 $\text{IsObjectDescription}^{2.55}(cst)$ *and*
5 *not* $\exists\, k \in s((cst)\text{DescribesImSubOf}^{2.152}(k))_0)$

Semiformal Description 2.150

Predicate "describes a class of immediate subordinates"

The "lower" object class description *lgo* describes a class of objects which are immediate subordinates of instances of the class described by the "higher" object class description *hgo*, if the 'object class identifier' of *lgo* is referred to from *hgo* via 'generator for subordinates'.

Definition 2.150

1 $\forall\, lgo, hgo$
2 $({}_0(lgo)\text{DescribesClassImSubOf}(hgo)$ *iff*
3 $(\text{C}\ \hat{}\ lgo \bullet \text{'object class identifier'})\text{IsClassIdImSubIn}^{2.142}(hgo)_0)$

Semiformal Description 2.151

Predicate "describes a class of subordinates"

An entity *lgo* ("lower" object class description) describes a class of objects which are (not necessarily immediate) subordinates of instances of the class described by the entity *hgo* ("higher" object class description) within the entity *g*, if *g* is a generic description containing *lgo* and *hgo* (3) and if *lgo* is referred to from *hgo* via 'generator for subordinates' (4) or has an "intermediate" object class description *i* immediately superior to it, which in turn describes a class of objects subordinate to instances of the object class described by *hgo* (5,6). The recursion ends when the intermediate object class description is immediately subordinate to *hgo* (4).

Definition 2.151

1 $\forall\, lgo, hgo, g$
2 $({}_0(lgo)\text{DescribesClassSubOf}(hgo)\text{In}(g)$ *iff*
3 $\text{IsGenericDescription}^{2.8}(g)$ *and* $lgo \in g$ *and* $hgo \in g$ *and*
4 $({}_1(lgo)\text{DescribesClassImSubOf}^{2.150}(hgo)$ *or*
5 $\exists\, i \in g$
6 $((lgo)\text{DescribesClassImSubOf}^{2.150}(i)$ *and* $(i)\text{DescribesClassSubOf}^{2.151}(hgo)\text{In}(g)_1)_0)$

Semiformal Description 2.152

Predicate "describes an immediate subordinate"

The "lower" object description *lso* describes an object which is an immediate subordinate of the object described by the "higher" object description *hso*, if the 'object identifier' of *lso* is referred to from *hso* via 'subordinates'.

Definition 2.152

1 $\forall lso, hso$
2 $(_0(lso)\text{DescribesImSubOf}(hso)$ *iff*
3 $(_1\text{IsObjectDescription}^{2.55}(lso)$ *and*
4 $\text{IsCompositeObjectDescription}^{2.51}(hso)$ *and*
5 $\exists n \in \hat{}hso \bullet \text{'subordinates'} \bullet$
6 $(\text{C } \hat{}lso \bullet \text{'object identifier'} = \text{C } \hat{}hso \bullet \text{'object identifier'}//[\rightarrow \text{C } n \rightarrow])_1)_0)$

Semiformal Description 2.153

Predicate "describes a content portion"

An entity *cont* describes a content portion of the basic object described by *b* or of a basic object, which is an instance of the object class described by *b*, if *cont* is a content portion description and either the 'content identifier logical' or the 'content identifier layout' is referred to from *b* via 'content portions'.

Definition 2.153

1 $\forall cont, b$
2 $(_0(cont)\text{DescribesContPortOf}(b)$ *iff*
3 $(_1\text{IsContentPortionDescription}^{2.56}(cont)$ *and*
4 $(\text{IsBasicObjectDescription}^{2.52}(b)$ *or* $\text{IsBasicObjectClassDescription}^{2.36}(b))$ *and*
5 $(_2(\text{C } \hat{}cont \bullet \text{'content identifier logical'})\text{IsIdContIn}^{2.144}(b)$ *or*
6 $(\text{C } \hat{}cont \bullet \text{'content identifier layout'})\text{IsIdContIn}^{2.144}(b)_2)_1)_0)$

7.4.2 Functions

Semiformal Description 2.154

Function "set of immediate subordinates"

The function IMSUB is defined, if *so* is an attribute set with at least the attributes 'object identifier' and 'subordinates', and if for each component spot *m* of the value of 'subordinates' the collection *sos* contains a nomination *cst* whose component 'object identifier' is equal to the object identifier of *so* with the component at *m* having been appended (i.e., if all subordinate object descriptions are within *sos*).

IMSUB returns the collection of all specific object descriptions within the collection *sos* whose identifiers, without the last number, are equal to the object identifier of the specific object description *so* and whose last number of their identifiers is among the numbers of 'subordinates' in the attribute set *so*.

Definition 2.154

```
1   ∀ so, sos
2   (0IMSUB(so, sos) =
3     IF IsAttributeSet^2.62(so) and
4       ['object identifier'; 'subordinates'] ⊆ NAMS^1.18(so) and
5       ∀ m ∈̂ (^so • 'subordinates' •) ∃ cst ∈ sos
6       (1C ^cst • 'object identifier' = C ^so • 'object identifier'//[ → C m ↪]1)
7     THEN
8     COLC^1.17 ^sos • <2xs ||F xs • 'object identifier' • <ys || PR^1.12 ys = 1> =
9                           C ^so • 'object identifier' and
10                          C xs • 'object identifier' • <ys || PR^1.12 ys = 1> ∈
11                          COLC^1.17 ^so • 'subordinates' • 2>
12    ELSE UNDEF0)
```

Semiformal Description 2.155

Function "all content portions"

ALLCONT returns the collection of all content portion descriptions contained in the collection *csts*.

Definition 2.155

```
1   ∀ csts
2   (0ALLCONT(csts) =
3     IF IsCol(csts)
4     THEN [cst | cst ∈ csts and IsContentPortionDescription^2.56(cst)]
5     ELSE UNDEF0)
```

Semiformal Description 2.156

Function "sequence of class identifiers of immediate subordinates"

The function SEQCLASSIDIMSUB is defined, if *so* is an attribute set (3) with at least the attributes 'object identifier' and 'subordinates' (4), if *sub* is equal to a catenation which extends from some component to the last component of the value of 'subordinates' (5), and if all subordinate object descriptions of *so* are in *sos* (6) and have the attribute 'object class' (7).

SEQCLASSIDIMSUB returns the sequence of object class identifiers drawn from those immediate subordinate object descriptions which are referred to using the numbers of *sub* along with the object identifier of *so*. The object class identifiers in the obtained sequence appear in the same order as their comprising object descriptions according to the catenation *sub*. The sequence is obtained through recursive application of the function: a sequence of one object class identifier is formed using the first component of *sub* (9-11) and is concatenated with the result of SEQCLASSIDIMSUB applied to the remaining rearpart of *sub* (14) or with the empty catenation (13) if the recursion is to end.

Definition 2.156

```
1   ∀ sos, so, sub
2   (₀SEQCLASSIDIMSUB(sos, so, sub) =
3     IF IsAttributeSet^2.62(so) and
4        ['object identifier'; 'subordinates'] ⊆ NAMS^1.18(so) and
5        ∃m ∈̃ (^so•'subordinates'•) (sub = [ → C m → ]//R m) and
6        IMSUB^2.154(so, sos) ⊆ sos and
7        ∀cst ∈ IMSUB^2.154(so, sos) ('object class' ∈ NAMS^1.18(cst))
8     THEN
9     [ →
10    C ^sos•<₁xs ||C xs•'object identifier' =
11                  C ^so•'object identifier'//[ → HEAD^1.18(sub) → ] ₁>•'object class' → ] //
12    IF LENGTH^1.16 sub = 1
13    THEN [ → ]
14    ELSE SEQCLASSIDIMSUB^2.156(sos, so, TAIL^1.15(sub))
15    ELSE UNDEF₀)
```

7.5 Additional terminological definitions

Semiformal Description 2.157

Predicate "is a constituent"

A constituent is either a document profile or an object class description or an object description or a content portion description or a layout style or a presentation style.

Definition 2.157

1 $\forall\, cst$
2 $({}_{o}\,\text{IsConstituent}(cst)$ *iff*
3 $\text{IsDocumentProfilePart2}^{2.20}(cst)$ *or*
4 $\text{IsObjectClassDescription}^{2.37}(cst)$ *or*
5 $\text{IsObjectDescription}^{2.55}(cst)$ *or*
6 $\text{IsContentPortionDescription}^{2.56}(cst)$ *or*
7 $\text{IsLayoutStyle}^{2.60}(cst)$ *or*
8 $\text{IsPresentationStyle}^{2.61}(cst)_{o})$

Semiformal Description 2.158

Predicate "is a page class description" (3.5.7)

A page class description is either a composite page class description or a basic page class description.

Definition 2.158

1 $\forall\, cst$
2 $({}_{o}\,\text{IsPageClassDescription}(cst)$ *iff*
3 $\text{IsCompositePageClassDescription}^{2.28}(cst)$ *or* $\text{IsBasicPageClassDescription}^{2.29}(cst)_{o})$

Semiformal Description 2.159

Predicate "is a logical content portion description"

A logical content portion description is a content portion description where the attribute 'content identifier logical' is specified.

Definition 2.159

1 $\forall\, logcont$
2 $({}_{o}\,\text{IsLogicalContentPortionDescription}(logcont)$ *iff*
3 $\text{IsContentPortionDescription}^{2.56}(logcont)$ *and* 'content identifier logical' $\in \text{NAMS}^{1.18}(logcont)_{o})$

Semiformal Description 2.160

Predicate "is a layout content portion description"

A layout content portion description is a content portion description where the attribute 'content identifier layout' is specified.

Definition 2.160

1 $\forall\, laycont$
2 $({}_{o}\,\text{IsLayoutContentPortionDescription}(laycont)$ *iff*
3 $\text{IsContentPortionDescription}^{2.56}(laycont)$ *and* 'content identifier layout' $\in \text{NAMS}^{1.18}(laycont)_{o})$

Semiformal Description 2.161

Predicate "is a logical identifier"

A logical identifier is a sequence of non-negative integers where the first integer is 2 or 3.

Definition 2.161

1 $\forall v$
2 $(_0$ IsLogicalId(v) *iff*
3 IsSeqOfNnInt$^{1.23}(v)$ *and*
4 $(_1$ HEAD$^{1.13}(v) = 2$ *or* HEAD$^{1.13}(v) = 3_1)_0)$

Semiformal Description 2.162

Predicate "is a layout identifier"

A layout identifier is a sequence of non-negative integers where the first integer is 0 or 1.

Definition 2.162

1 $\forall v$
2 $(_0$ IsLayoutId(v) *iff*
3 IsSeqOfNnInt$^{1.23}(v)$ *and*
4 $(_1$ HEAD$^{1.13}(v) = 0$ *or* HEAD$^{1.13}(v) = 1_1)_0)$

Semiformal Description 2.163

Predicate "is a specific identifier" (5.3.1.2)

A specific identifier is either a specific logical identifier or a specific layout identifier.

Definition 2.163

1 $\forall v$
2 $(_0$ IsSpecificId(v) *iff*
3 IsLogicalObjectId$^{2.78}(v)$ *or* IsLayoutObjectId$^{2.79}(v)_0)$

Semiformal Description 2.164

Predicate "is a generic identifier" (5.3.1.3)

A generic identifier is either a generic logical identifier or a generic layout identifier.

Definition 2.164

1 $\forall v$
2 $(_0$ IsGenericId(v) *iff*
3 IsLogicalObjectClassId$^{2.81}(v)$ *or* IsLayoutObjectClassId$^{2.82}(v)_0)$