INTERNATIONAL STANDARD

ISO 24610-1

FIrst edition
2006-04-15

Language resource management — Feature structures —

Part 1: Feature structure representation

Gestion des ressources linguistiques — Structures de traits —

Partie 1: Représentation de structures de traits

Reference number
ISO 24610-1:2006(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 24610-1 was prepared by Technical Committee ISO/TC 37, *Terminology and other language and content resources*, Subcommittee SC 4, *Language resource management*.

ISO 24610 consists of the following parts, under the general title *Language resource management — Feature structures*:

— *Part 1: Feature structure representation*

The following part is under preparation:

— *Part 2: Feature system declaration*

Introduction

This part of ISO 24610 results from the agreement between the Text Encoding Initiative Consortium (TEI) and the ISO TC 37/SC 4 that a joint activity should take place to revise the two existing chapters on feature structures and feature system declaration in *The TEI Guidelines* called *P4.*

It is foreseen that ISO 24610 will have the following two parts.

— Part 1, *Feature structure representation,* describes feature structures and their representation. It provides an informal but explicit overview of their basic characteristics and formal semantics. In addition, part 1 defines a standard XML (eXtended Markup Language) vocabulary for the representation of untyped feature structures, feature values, and feature libraries. It thus provides a reference format for the exchange of feature structure representations between different application systems.

— Part 2, *Feature system declaration*, discusses ways of validating typed feature structures which are conformant to part 1, and of enforcing application-specific constraints. It proposes an XML vocabulary for the representation of such constraints with reference to a set of features and the range of values appropriate for them, and thus facilitates representation and validation of a type hierarchy as well as other well-formedness conditions for particular applications, in particular those related to the goal of language resource management.

Language resource management — Feature structures —

Part 1:
Feature structure representation

1 Scope

Feature structures are an essential part of many linguistic formalisms as well as an underlying mechanism for representing the information consumed or produced by and for language engineering applications. This part of ISO 24610 provides a format for the representation, storage and exchange of feature structures in natural language applications concerned with the annotation, production or analysis of linguistic data. It also defines a computer format for the description of constraints that bear on a set of features, feature values, feature specifications and operations on feature structures, thus offering a means of checking the conformance of each feature structure with regards to a reference specification.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8879, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, as extended by TC 2 (ISO/IEC JTC 1/SC 34 N029: 1998-12-06).

ISO 19757-2, *Information technology — Document Schema Definition Language (DSDL) — Part 2: Regular-grammar-based validation — RELAX NG*

NOTE The first reference permits the use of XML and the second, RELAX NG,provides a specification for XML modules. RELAX NG is a schema language for XML, standing for REgular LAnguage for XML for Next Generation, and simplifies and extends the features of DTDs, Document Type Definitions.

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 8879 and ISO 19757-2 and the following apply. This list is provided to clarify the terminology relating to feature structures used throughout this part of ISO 24610. Terminology derived from XLM and other formal languages is not defined here.

3.1
alternation
operation on feature **values** (3.23) that returns one and only one of the values supplied as its argument

NOTE Given a feature specification $F : a|b$, where $a|b$ denotes the alternation of a and b, F has either the value a or the value b, but not both.

3.2
atomic value
value (3.23) without internal structure, i.e. value other than **feature structure** (3.10) and **collection** (3.4)

3.3
boxed label
label in box used in a matrix notation to denote a value shared by several **features** (3.8)

NOTE The label may be any alphanumeric symbol.

3.4
collection
list, set, or multiset of **values** (3.23)

NOTE A list is an ordered collection of entities some of which may be identical. A set is an unordered collection of unique entities. A multiset is an unordered collection of entities that may or may not be unique; it is sometimes referred to as a bag.

3.5
complex value
value (3.23) represented either as a **feature structure** (3.10) or as **collection** (3.4)

3.6
concatenation
operation of combining two lists of **values** (3.23) into a single list

3.7
empty feature structure
feature structure (3.10) containing no **feature specifications** (3.9)

3.8
feature
property of an entity

NOTE The combination of feature and feature-value constitutes a **feature specification** (3.9). For example, number is a feature, singular is a value, and a pair <number, singular> is a feature specification.

3.9
feature specification
assignment of a **value** (3.23) to a **feature** (3.8)

NOTE Formally, it is treated as a pair of a feature and its value.

3.10
feature structure
set of **feature specifications** (3.9)

NOTE The minimum feature structure is the **empty feature structure** (3.7).

3.11
graph notation
notation of **feature structure** (3.10) in a single rooted graph

3.12
incompatibility
relation between two **feature structures** (3.10) which have conflicting **types** (3.19) or at least one common **feature** (3.8) with incompatible **values** (3.23)

NOTE Two feature structures that are incompatible cannot be unified. The **empty feature structure** (3.7) is compatible with any other feature structure.

3.13
matrix notation
attribute-value matrix
AVM
notation that uses square brackets to represent **feature structures** (3.10)

NOTE In a matrix notation, each row represents a **feature specification** (3.9), with the feature name and the feature value separated by a colon (:), space ( ) or the equals sign (=).

3.14
merge
generic operation that includes **union** (3.22) of sets or multisets and **concatenation** (3.6) of lists

3.15
negation
(unary) operation on a **value** (3.23) denoting any other value incompatible with it

NOTE In this part of ISO 24610, negation applies to values only and is not understood as a truth function as in ordinary bivalent logics.

3.16
path
sequence of labeled arcs connecting nodes in a graph

3.17
structure sharing
re-entrancy
relation between two or more **features** (3.8) within a **feature structure** (3.10) that share a **value** (3.23)

3.18
subsumption
relationship between two **feature structures** (3.10) in which one is more specific than the other

NOTE A feature structure A is said to subsume a feature structure B if A is at least as informative as B. Subsumption is a reflexive, antisymmetric, and transitive relation between two feature structures.

3.19
type
name of a class of entities

NOTE **Feature structures** (3.10) may be characterized by grouping them into certain classes. Types are used to name such classes.

3.20
typed feature structure
feature structure (3.10) labelled by a **type** (3.19)

NOTE In the **graph notation** (3.11), each node is labelled with a type. In the **matrix notation** (3.13), a type is ordinarily placed at the upper left corner of the inside of the pair of square brackets that represents a typed feature structure. In XML notation, the type is supplied as the **value** (3.23) of a type attribute on the <fs> element.

3.21
unification
operation that combines two compatible **feature structures** (3.10) into the least informative feature structure that contains the information from the two

3.22
union
operation that combines two sets, or multisets, into one

NOTE The corresponding operation for lists is **concatenation** (3.6).

3.23
value
information about an entity

NOTE There are two kinds of feature values: **atomic value** (3.2) and **complex value** (3.5).

4 General characteristics of feature structure

4.1 Overview

A feature structure is a general purpose data structure that identifies and groups together individual features by assigning a particular value to each. Because of the generality of feature structures, they can be used to represent many different kinds of information. Interrelations among various pieces of information and their instantiation in markup provide a meta-language for representing analysis and interpretation of linguistic content. Moreover, this instantiation allows a specification of a set of features with values of specific types and restrictions, by means of feature system declarations, or other XML mechanisms discussed in ISO 24610-2[1)] .

4.2 Use of feature structures

Feature structures provide partial information about an object by specifying values for some or all of its features. For example, if a female employee named Sandy Jones who is 30 years old is of the present concern, then that person's sex, name and age can be specified in a succinct manner by assigning a value to each of these three features. These pieces of information can be put into a simple set notation, as in:

(1) Employee

{<SEX, *female*>, <NAME, *Sandy Jones*>, <AGE, *30*>}

Feature structures are generally used as a vehicle for linguistic descriptions. For example, the phoneme /p/ in English can be analysed in terms of its distinctive features: consonantal, anterior, voiceless, non-continuant or stop sound segment, etc. Each of these features may be combined with one or other of the binary values plus(+) and minus(-) to provide a feature specification. In a phonemic analysis such as the following, the value of a feature specifies the presence or absence of that feature:

(2) Sound segment /p/

{<CONSONANTAL, + >, <ANTERIOR, + >, <VOICED, ->, <CONTINUANT, ->}

In such an analysis, the sound segment /p/ is distinguished from other phonemes in terms of the presence or absence of specific features. For example, /p/ differs from the phoneme /b/ in VOICING, and from /k/ in articulatory position: one is articulated at the anterior, (the lip or alveolar area of the mouth), and the other at the nonanterior, namely the back of the oral cavity.

This feature analysis can be extended to other kinds of description. Consider a verb like "love". Its features include both syntactic and semantic properties: as a transitive verb, it takes an object as well as a subject as its arguments, expressing the semantic relation of loving between two persons or animate beings. The exact representation of these feature specifications requires a detailed elaboration of what feature structures are. For now, these grammatical features can be roughly represented in a set format like the following:

(3) Grammatical features of the verb "love"

{<POS, *verb*>, <VALENCE, *transitive*>, <SEMANTIC_RELATION, *loving*>},

where POS stands for part of speech.

1) Under preparation.

Since its first extensive use in generative phonology in mid-60s, the feature structure formalism has become an essential tool not only for phonology, but more generally in support of syntax and semantics as well as lexicon building, especially in computational work. Feature structures are used to describe and model linguistic entities and phenomena by specifying their properties. In the next clauses, some of the formal properties of feature structures are outlined together with means of representing them in a systematic manner.

4.3 Basic concepts

Feature structures may be viewed in a variety of ways. The most common and perhaps the most intuitive views are the following:

— a set of feature specifications that consists of pairs of features and their values;

— labelled directed graphs with a single root where each arc is labelled with the name of a feature and directed to its value.

In set-theoretic terms, a feature structure *FS* can be defined as a partial function from a set Feat of features to a set FeatVal of values, where FeatVal consists of a set AtomVal of atomic values and a set FS of feature structures.

(4) A feature structure as a set or partial function

$FS \subseteq \{<F_i, v_i> \mid F_i \in \mathbf{Feat}, v_i \in \mathbf{FeatVal}\}$

or

$FS : \mathbf{Feat} \longrightarrow \mathbf{FeatVal}$

where $\mathbf{FeatVal} = \mathbf{AtomVal} \cup \mathbf{FS}$ and where **FeatVal** stands for all possible values.

Values may be regarded as either atomic or complex. Atomic values are entities without internal structure, while complex values may be feature structures or collections of values (either complex or atomic).

NOTE These two definitions are not absolutely equivalent, for in a set there may be more than one value, v, for a feature, F.

For example, the part of speech (POS) feature can take the name of an atomic morpho-syntactic category such as verb as its value. Conversely, the agreement (AGR) feature in English takes a complex value in the form of a feature structure with features PERSON and NUMBER. The word “loves”, for instance, has a POS feature with the value “verb”, while the value of its AGR feature consists of a feature structure comprising two feature specifications: PERSON with the value “3rd”, and NUMBER with the value “singular”.

4.4 Notations

4.4.1 Overview

As a list of feature-value pairs, the overall form of a feature structure is simple. However, the internal structure of a feature structure may be complex when a feature structure contains either

a) a feature whose value is itself a feature structure, or

b) a multi-valued feature whose value is a list, set, or multiset.

Lists, sets and multisets may be made up of atomic values, or they may include complex values that are themselves feature structures. That is, feature structures allow limitless recursive embedding. It is therefore necessary to represent them in an understandable and mathematically precise notation.

There are two commonly used notations for the representation of feature structures: 1) graphs; and 2) matrices. This part of ISO 24610 proposes a third notation based on the XML standard. The following subclauses discuss how the same feature structure may be represented using each of these equivalent notations. Graphs are suitable for mathematical discourses, matrices for linguistic descriptions, and XML notations for computational implementation.

4.4.2 Graph notation

Feature structures are often represented as labelled directed graphs with a single root.

NOTE This graph can be either

a) acyclic, in which case it is referred to as a directed acyclic graph (DAG), or

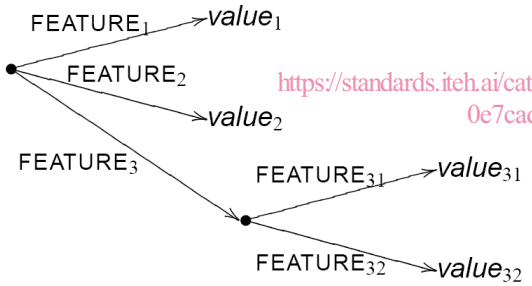
b) cyclic for handling cases like the Liar's paradox.

Feature structures are usually represented as DAGs, but it has been suggested that cyclical feature structures should be introduced to model some linguistic phenomena.

Each graph starts with a single particular node called the root. From this root, any number of arcs may branch out to other nodes and then some of them may terminate or extend to other nodes. The extension of directed arcs shall, however, stop at some terminal nodes. On such a graph, representing a feature structure, each arc is labelled with a feature name and its directed node is labelled with its value.

Here is a very simple example for a directed graph representing a feature structure.

(5) Feature structure in graph notation



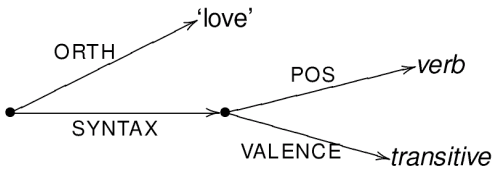
In this graph, the two features, FEATURE1 and FEATURE2, are atomic-valued, taking value1 and value2 on the terminal nodes respectively as their value. The feature FEATURE3 is, however, complex-valued, since it takes as its value the feature structure which is represented by the two arcs FEATURE31 and FEATURE32 with their respective values, value31 and value32.

A graph may just consist of the root node, that is, without any branching arcs. Such a graph represents the empty feature structure.

There may be one or more arcs branching out from the root, each of them bearing a single feature name. Some of these labelled arcs originating from the root may again stretch out to another node and then from this node to another, forming an indefinitely long sequence of feature names. Such a sequence of feature names, labelling the arcs from the unique root node to each of the terminal nodes on a graph, is called a path. For example, there are four paths in (5): FEATURE1, FEATURE2, FEATURE3.FEATURE31 and FEATURE3.FEATURE32.

Here is a linguistically more relevant example.

(6) Linguistic example in graph notation



This graph consists of three paths: orthography (ORTH), SYNTAX.POS and SYNTAX.VALENCE. The path consisting of a single feature name ORTH is directed to the terminal node "love", which is an atomic value. The path SYNTAX.POS terminates with the atomic value "verb" and the path SYNTAX.VALENCE with the atomic value "transitive". The nonterminal feature SYNTAX takes a complex value, namely a feature structure consisting of the two feature specifications, <POS, verb> and <VALENCE, transitive>.

4.4.3 Matrix notation

Despite their mathematical elegance, graphs are difficult to typeset or read, especially when complex. For this reason, feature structures are more often depicted using a matrix notation called attribute-value matrix, or simply AVM.

NOTE 1 The term "attribute" is an alternative term for the concept which is called "feature".

(7) Matrix notation

$$\begin{bmatrix} \text{FEATURE}_1 \ value_1 \\ \text{FEATURE}_2 \ value_2 \\ \text{FEATURE}_3 \begin{bmatrix} \text{FEATURE}_{31} \ value_{31} \\ \text{FEATURE}_{32} \ value_{32} \\ \ldots \\ \text{FEATURE}_{3k} \ value_{3k} \end{bmatrix} \\ \ldots \\ \text{FEATURE}_n \ value_n \end{bmatrix}$$

Each row in the square bracket with a FEATURE name followed by its value name represents a feature specification in a feature structure.

NOTE 2 A colon, an equals sign or a space separates a feature from its value on each row of an AVM.

Feature values can be either atomic or complex. Each row with an atomic value terminates at that value. But if the value is complex, then that row leads to another feature structure, as in the case of FEATURE3 above.

The notion of path is also important in several applications of the attribute-value matrix (AVM) notation, as discussed further below. A path in an AVM is a sequence of feature names, as is the case with feature structure graphs. An AVM with no rows and thus no occurrences of features, is represented as [ ]; this represents the empty feature structure. If an AVM has at least one row consisting of a feature name and its value, then there is a path of length 1 corresponding to each occurrence of a feature name in each row. Given a path of length i, if the last member of that path takes a nonempty feature structure as value, then that path forms a new path of length i +1 by taking each one of the features in that feature structure as its member.

For illustration, consider the following AVM which represents the same feature structure as is represented by the graph notation (6).

(8) Example of an AVM notation

$$\begin{bmatrix} \text{ORTH 'love'} & \\ \text{SYNTAX} & \begin{bmatrix} \text{POS } \textit{verb} \\ \text{VALENCE } \textit{transitive} \end{bmatrix} \end{bmatrix}$$

This AVM has three paths: ORTH, SYNTAX.POS and SYNTAX.VALENCE.

4.4.4 XML-based notation

Like any other formalism, XML has its own terminology. An XML document consists of typed elements, occurrences of which are marked by start- and end-tags which enclose the content of the element. Every XML element may be regarded as a linearization of a tree with a single root node. Each node contains either terminal text or other element nodes. Elements have a specific type. Element occurrences can also carry named attributes (or, more exactly, attribute-value pairs). For example:

(9) <word class="noun">love</word>

This is the XML way of representing an occurrence of the element type word, the content of which is the string “love”. The word element is defined as having an attribute called “class”, whose value in this case is the string “noun”.

The term “attribute” is thus used in a way which potentially clashes with its traditional usage in discussions of feature structures. In AVM, feature structures are represented in a square bracket form, with each row representing an attribute-value pair. In this usage, an attribute does not correspond necessarily to an XML attribute. To avoid this potential source of confusion, the reader should be aware that in the remainder of this clause, the term “attribute” will be used only in its technical XML sense.

To illustrate further the distinction, it should now be considered how a feature structure in AVM may be represented in XML. Consider the following AVM that represents a non-typed feature structure:

(10) Representation of a feature structure in AVM

$$\begin{bmatrix} \text{ORTH } \textit{love} \\ \text{POS } \textit{noun} \end{bmatrix}$$

This feature structure consists of two feature specifications: one combines the feature ORTH with the string value “love” and the other combines the feature POS with the string value “noun”.

NOTE This representation says nothing about the range of possible values for the two features: in particular, it does not indicate that the range of possible values for the ORTH feature is not constrained, whereas (at least in principle) the range of possible values for the POS feature is likely to be constrained to one of a small set of valid codes. In a constraint-based grammar formalism, possible values for the feature ORTH need to be strings consisting of a finite number of characters drawn from a well-defined character set, say the Roman Alphabet plus some special characters, for each particular language.

In the XML representation proposed here, a feature structure is represented by means of an XML element fs, and a feature-value specification by means of an XML element f:

(11)
```
<fs>
    <f> ...</f>
    <f> ...</f>

</fs>
```


This is a more generic approach than another, equally plausible, representation, in which each feature specification might be represented by a specifically-named element:

(12) <fs>
<orth>...</orth>
<pos>...</pos>
</fs>

Using this more generic approach means that systems can be developed which are independent of the particular feature set, at the expense of slightly complicating the representation in any particular case. By representing the specification of a feature as an f element, rather than regarding each specific feature as a different element type, the overall processing model is simplified considerably.

A feature specification, as has been seen, has two components: a name, and a value. Again, there are several equally valid ways of representing this combination in XML. Any of the following three, for example, could be chosen:

(13)

a) <f name="pos" value="noun"/> ;

b) <f name="pos">
<value>noun</value>
</f> ;

c) <f>
<name>pos</name>
<value>noun</value>
</f> .

The fact that all three of these are possible arises from a redundancy introduced in the design of the XML language largely for historical reasons. The present recommendation is to use a formulation like b) above, though c) may be preferable on the grounds of its greater simplicity.

Finally, the representation of the value part of a feature specification is discussed. A name is simply a name, but a value in the system discussed here may be of many different types: it might for example be an arbitrary string, one of a predefined set of codes, a Boolean value, or a reference to another (nested) feature structure. In the interests of greater expressivity, the present system proposes to distinguish amongst these kinds of value in the XML representation itself.

Once more, there are a number of more or less equivalent ways of doing this. For example:

(14)

a) <fs>
<f name="orth">
<value type="string">love</value></f>
<f name="pos">
<value type="symbol">noun</value></f>
</fs> ;