INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11

MPEG2001/Mxxxx

July 2001, Sydney

Source: **GETA, CPR, DFKI, AIST, Keio University, Kasetsart University, Université Montpellier II, IBM TRL, BPPT, NECTEC, New Mexico State University**

Title: **Expert Commitment on Linguistic/Semantic Descriptions**

Author: **Christian Boitet, Nicoletta Calzolari, Thierry Declerck, Kôiti Hasida, Shun Ishizaki, Asanee Kawtrakul, Mathieu Lafourcade, Katashi Nagao, Hammam Riza, Virach Sornlertlamvanich, Rémi Zajac, Antonio Zampolli**

Status: **for discussion**

1 Introduction

We are experts on natural language processing and artificial intelligence who are committed to the integration of the NLP/AI technology with the multimedia technology and hence to having MPEG-7 incorporate tools for describing linguistic/semantic structures of multimedia content. Described below are our basic disposition to what is happening in MPEG-7 in this respect and how we will contribute to the standardization process.

2 Disposition

People talk about multimedia content in terms of not only shapes and colors but also speeches, events, objects, and so on. The Linguistic DS, the Semantic DS, the Relation DS, the Classification Scheme, and so forth, are hence very important for access services on multimedia content.

The Linguistic DS is a simple enough tool to address the most important semantic aspect of linguistic structure, which normal people usually refer to when talking about multimedia content. So we recommend this DS as the tool for linguistic description, and thus will promote it as part of MPEG-7. However, there are possibilities of some extensions and probably some revisions for accommodating these extensions until the first amendment in the spring of 2002. For instance, multilinguality has not been explicitly considered in the design of this DS. In addition, more account is necessary on how to describe various linguistic phenomena.

We also support the motivation of the Semantic DS and the Relation DS. We thus believe that it is important that a more substantial group of experts participate in their design and evaluation. For instance, propositional attitude (*Tom believes that Sue is crazy*), quantifier scopes (distinction between the two readings of *every man loves a woman*), and so on, are not explicitly addressed in the current Semantic DS. In addition, the DS does not seem to be seriously considering the descriptive efficiency. We should begin with a minimal standard which could describe as much as possible. From this viewpoint, the SemanticState DS, the Semantic Place DS, the Semantic Time DS may not be??? necessary. Even the distinction between event and object is probably unnecessary at least in the first IS. %Could we say that all these issues need rethinking, or sth similar, without making too negative statements?? We also doubt that the AbstractionLevel Datatype makes any sense. The Relation DS is subcategorized into *XY*Relations for various *X* and *Y*, but this seems to make little sense in multimedia semantics, too.

Also for the Classification Scheme, since this tool should fit the other tools mentioned above, further discussions involving more expertise on the design of ontology in relation to linguistic and multimedia semantics is necessary until we decide what to standardize.

3 Recommendation

The tools discussed above are premature despite their importance. We suggest to avoid the inclusion of them in the first version of MEPG-7 IS. Our group of experts can work to improve them so that the coming amendment should incorporate them in better shapes. The Linguistic DS as of the below appendix should be included in the possible amendments so that we can effectively work on it.

Appendix: Revised Specification of the Linguistic DS

Linguistic Description

This section defines tools for describing the semantic structure of linguistic data associated with AV content, such as scenarios, transcriptions, critiques, and so forth.

The Linguistic DS

The Linguistic DS describes the semantic structure of the linguistic data associated with AV content, such as scenarios and transcriptions. Since such data have the same range of structures as linguistic data in general, the Linguistic DS provides a set of general tools for describing the semantic structure of any kind of linguistic data.

More precisely speaking, the Linguistic DS is to deal with linguistic data which the annotator cannot modify (for the sake of descriptive ease), whether or not such data are part of multimedia content. Scenarios and transcriptions are such data, because their modifications are not right scenarios and transcriptions any more%not very clear. In contrast, annotation texts can be arbitrarily rewritten so that they be easy to describe using Textual Annotation tools.

Concepts and Model

The Linguistic DS represents the semantic structure of linguistic data using textual mark-up. That is, the description of the semantic structure of the linguistic data is encoded by mixing MPEG-7 description directly into the text (of the linguistic data).

The Linguistic DS has been designed to reflect the semantic structure of linguistic data with XML tags. XML elements licensed by the Linguistic DS describe **linguistic entities.** Linguistic entities form a hierarchy. At the lowest level are syntactic constituents occurring within a sentence. At the higher levels are increasingly large units of linguistic structures, such as sentences, paragraphs, other document divisions and entire documents.

Linguistic entities are combined together to form larger linguistic entities in a recursive process called synthesis. **Dependency** is the most common way to synthesize linguistic entities. The second most common type of synthesis is **coordination**.

A **syntactic constituent** is a continuous linguistic entity inside a sentence that represents a semantic entity. For instance, in "Tom and Mary live in a small house", "Tom", "Tom and Mary", "in a small house", "a small house", "small house", "house", and so forth are syntactic constituents because each of them represents a semantic entity; "Tom and Mary" represents a group of two people, "small house" represents the notion of a small house, and so on. On the other hand, "Tom and", "Mary live", "in a", "a small", and so forth are not syntactic constituents because each of them fails to represent a semantic unit in this sentence.

Some linguistic accounts of agglutinative languages such as Japanese and Korean regard maximal morphological clusters (such as so-called *bunsetu* in Japanese) as syntactic constituents. However, the Linguistic DS does not regard them as such if they do not represent semantic entities. So□□□but not □□□ is a syntactic constituent in the following Japanese expression:

> □□ □ □□
> distant country from
> `from distant countries'

A syntactic constituent X **depends** on another syntactic constituent Y when the combination of X and Y represents a semantic unit that is equal to, a specialization of, or an instance of what Y represents. In "a small house", for instance, "a" depends on "small house" because "a small house" represents an instance of the concept of small houses. In "every man loves a woman", "every man" and "a woman" depend on "loves", because the sentence represents a state of affairs that is a specialization of the concept of loving. "Tom" depends on "for" in "for Tom" because "for Tom" represents a relationship with Tom that is a specialization (or an instance) of the general notion represented by "for".

Dependency is a way of synthesizing linguistic entities (typically syntactic constituents), in the sense that if a linguistic entity X depends on another linguistic entity Y that is next to it%not always next, then their concatenation Z is also a linguistic entity. Y is called the **governor** of X and the **head** of Z. In the case of "drive me crazy", for instance, "drive me" is the governor of "crazy"%NO?? and the head of "drive me crazy". "Drive" is also a head of "drive me crazy". A **phrase** is a syntactic constituent that is not a head. %avoid such an example difficult to agree on

Coordination is a second major way of combining linguistic entities. "Tom and Mary", "Tom or May", "not only Tom but also Mary", "Tom loves Mary and Bill loves Sue", and so forth are coordinate structures. Less straightforward coordinate structures include:

- "Tom loves Mary and Bill, Sue".
- "I gave this to Tom and that to Mary".

in which the second conjuncts ("Bill, Sue" and "that to Mary") lack their heads ("loves" and "gave", respectively). Such structures are called gapped structures.

Coordination may be regarded as a special case of dependency, in which case the coordinate conjunction is the head of the whole coordinate structure. For instance, "and" is the head of "Tom and Mary" and "or" is the head of "dead or alive".

Relation to the Dependency Structure Datatype

The Linguistic DS is an upward compatible extension of the DependencyStructure Datatype in that every instance of the DependencyStructure Datatype is an instance of the Linguistic DS. More precisely, the Linguistic extends the DependencyStructure in the following three respects:

1) The Linguistic DS addresses linguistic entities larger than sentences, such as paragraphs and divisions. This is needed for describing the structure of an entire document such as a scenario. Although the DependencyStructure Datatype addresses dependencies inside sentences (i.e., among syntactic constituents) and the notion of such dependency is common between DependencyStructure Datatype and the Linguistic DS, the Linguistic DS also addresses dependencies outside of sentences: i.e., among linguistic entities such as sentences, bunches of sentences, paragraphs, sections, chapters, and so on. When a sentence may represent the cause or reason of the event represented by another sentence, for instance, the former sentence is regarded as depending on the latter and having `cause` as its value for the `operator` attribute.

2) The Linguistic DS addresses **extraposition**, which is dependency on a constituent embedded within a phrase. For instance, an extraposition occurs in "Tom, I do not like". Here "Tom" depends on "like", which is embedded in phrase "I do not like". This is described by attaching a `depend` attribute to "Tom". The depend attribute may be used to describe dependencies outside of sentences as well, which sometimes simplifies the description by omitting several elements.

3) The Linguistic DS allows partial descriptions owing to the mixed content model and the synthesis attribute. The mixed content model simplifies the description by omitting tags. Elements may have mixed contents in partial descriptions, whereas they should have element-only or text-only contents in full descriptions. Every syntactic constituent must be an element in a full description.

Linguistic DS Syntax

```
<!-- ##################################################### -->
<!-- Definition of Linguistic DS                            -->
<!-- ##################################################### -->

<complexType name="LinguisticEntityType" abstract="true" mixed="true">
    <complexContent>
        <extension base="mpeg7:DSType">
            <choice minOccurs="1" maxOccurs="unbounded">
                <element type="mpeg7:MediaLocatorType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Relation" type="mpeg7:RelationType"
                    minOccurs="1" maxOccurs="1"/>
            </choice>
            <attribute ref="xml:lang"/>
            <attribute name="type" type="NMTOKENS" use="optional"/>
            <attribute name="depend" type="IDREF" use="optional"/>
            <attribute name="equal" type="IDREF" use="optional"/>
            <attribute name="operator" type="mpeg7:operatorType"/>
        </extension>
    </complexContent>
</complexType>

<!-- #####################################################   -->
<!-- Definition of LinguisticDocument DS                     -->
<!-- #####################################################   -->

<complexType name="LinguisticDocumentType">
    <complexContent>
        <extension base="mpeg7:LinguisticEntityType">
            <choice minOccurs="1" maxOccurs="unbounded">
                <element name="Heading" type="mpeg7:SentencesType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Division" type="mpeg7:LinguisticDocumentType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Paragraph" type="mpeg7:SentencesType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Sentences" type="mpeg7:SentencesType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Sentence" type="mpeg7:SyntacticConstituentType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Quotation" type="mpeg7:LinguisticDocumentType"
                    minOccurs="1" maxOccurs="1"/>
            </choice>
            <attribute name="synthesis" type="mpeg7:synthesisType"
                use="default" value="coordination"/>
        </extension>
    </complexContent>
</complexType>


<!-- #####################################################   -->
<!-- Definition of Sentences DS                              -->
<!-- #####################################################   -->

<complexType name="SentencesType">
    <complexContent>
        <extension base="mpeg7:LinguisticEntityType">
            <choice minOccurs="1" maxOccurs="unbounded">
                <element name="Sentence" type="mpeg7:SyntacticConstituentType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Quotation" type="mpeg7:LinguisticDocumentType"
                    minOccurs="1" maxOccurs="1"/>
            </choice>
            <attribute name="synthesis" type="mpeg7:synthesisType"
                use="default" value="coordination"/>
        </extension>
    </complexContent>
</complexType>


<!-- #####################################################   -->
<!-- Definition of Syntactic Consituent DS                   -->
<!-- #####################################################   -->

<complexType name="SyntacticConstituentType">
    <complexContent>
        <extension base="mpeg7:LinguisticEntityType">
            <choice minOccurs="0" maxOccurs="unbounded">
                <element name="Head" type="mpeg7:SyntacticConstituentType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Phrase" type="mpeg7:SyntacticConstituentType"
                    minOccurs="1" maxOccurs="1"/>
                <element name="Quotation" type="mpeg7:LinguisticDocumentType"
            </choice>
            <attribute name="term"
                type="mpeg7:controlledTermIdentifierType" use="optional"/>
            <attribute name="scheme"
                type="mpeg7:classificationSchemeIdentifierType" use="optional"/>
            <attribute name="schemeLocation"
                type="mpeg7:classificationSchemeLocatorType" use="optional"/>
            <attribute name="baseForm" type="string" use="optional"/>
            <attribute name="pronunciation" type="string" use="optional"/>
            <attribute name="edit" type="string" use="optional"/>
            <attribute name="synthesis"
                type="mpeg7:synthesisType" use="default" value="dependency"/>
            <attribute name="particle" type="string" use="optional"/>
        </extension>
    </complexContent>
</complexType>

<!-- ######################################################   -->
<!-- Definition of synthesis datatype                         -->
<!-- ######################################################   -->

<simpleType name="synthesisType">
    <restriction base="string">
        <enumeration value="none"/>
        <enumeration value="dependency"/>
        <enumeration value="forward"/>
        <enumeration value="backward"/>
        <enumeration value="coordination"/>
        <enumeration value="apposition"/>
        <enumeration value="repair"/>
        <enumeration value="error"/>
        <enumeration value="idiosyncratic"/>
    </restriction>
</simpleType>
```

Linguistic DS Semantics

Semantics for the LinguisticEntityType:

Name	Definition
LinguisticEntityType	An abstract DS for the various types of linguistic entities. Instances of this DS mix together the description and the text representation of the language data being described. In other words, this DS describes the data by "marking up" the text with a set of XML tags.
MediaLocator	Locates the portion of speech data, visual text data, or text (such as ASCII) data corresponding to the current linguistic entity. Instead of containing the linguistic data as text directly within the LinguisticEntity DS, an instance of the LinguisticEntity DS may choose only to locate the linguistic using a MediaLocator. For example, this can be used to locate an external speech-data file transcribed as the text embedded in the current LinguisticEntity element, or an external text file containing the transcript of an AV program.
xml:lang	Indicates the language of the data being described.
type	Indicates the type of the linguistic entity such as document part (chapter, section, embedded poem, letter, etc.), part of speech of a syntactic constituent, etc. For example,

Name Definition

```
<Head type="noun">salt</Head>
```

Editor's Note: do we need an enumeration list or a Classification Scheme for this attribute?%this will need to be elaborated further (as many other things)

depend

References the semantic governor (head of larger constituent). This encodes the linguistic entity that is dependent %NO viceversa the extraposed linguistic entity.

In the following example, Tom depends on hates, which is the head of embedded clause Mary hates.

```
<Sentence synthesis="backward">
<Phrase depend="H" operator="patient">
Tom
</Phrase>,
  <Phrase>I</Phrase>
  think that
  <Phrase>
    <Phrase>Mary</Phrase>
    <Head id="H">hates</Head>
  </Phrase>.
</Sentence>
```

Sentences and larger linguistic entities may have a value for `depend` too because they can be related through `operators` such as `cause` and `elaboration`.

equal

Indicates an element referring to the same thing (i.e. coreferent) as this linguistic entity. The coreferent may be an instance of the LinguisticEntity DS, the Segment DS, or the Semantic DS.

operator

Indicates the meaning of the %NO not only function words, see also the examples (preposition, conjunction, article, and so on) which is the head of the current element. Most of the case, it is the relationship of the element with the governor, which is the linguistic entity that this linguistic entity is dependent on.

Here are two examples:

```
<Sentence>
<Phrase operator="agent">Tom</Phrase>
  <Head>loves</Head>
  <Phrase operator="patient">Sue</Phrase>.
</sentence>
```

```
<Sentences synthesis="forward">
  <Sentence   operator="cause">   %always   of   the   'next'
sentence?
    Tom hit Mary.
  </Sentence>
  <Sentence>She cried.</Sentence>
</Sentences>
```

When the function word is a coordinate conjunction, its meaning is the relationship between the conjuncts, as in the following example. Note that the coordinate conjunction is the head of the coordinate structure.

```
<Sentence>
  <Phrase>
    <Phrase>Tom</Phrase>
    <Head operator="and">and</Head>
    <Phrase>Mary</Phrase>
  </Phrase>
  got married.
</Sentence>
```

&a list of values for operator will have to be elaborated

Semantics of LinguisticDocumentType:

Name	Definition
`LinguisticDocumentType`	DS describing an entire linguistic document, such as a scenario or transcript. The structure of a linguistic document is represented by a recursive hierarchy of entities: each linguistic entity in the document (section, paragraph, etc) can be broken down further into its component linguistic entities (other sections, sentences, syntactic constituents, etc.).
`Heading`	Describes one heading for a document or a division.
`Division`	Describes one textual division within a document, such as a chapter, a section, an embedded poem, etc.
`Paragraph`	Describes one paragraph within the document.
`Sentences`	Describes a series of sentences occurring within this document.
`Sentence`	Describes a phrase that is a complete proposition, question, request, etc., and does not participate in any syntactic dependency. Usually contains a period or a question mark at the end.
`Quotation`	Describes a direct narrative or citation. Namely, an utterance by someone other than the addressor of the surrounding content.
`synthesis`	Indicates the type of synthesis among the child entities and text contained within this entity.

Semantics of SentencesType:

Name	Definition
`SentencesType`	DS representing a sequence of sentences.
`Sentence`	Describes a phrase that addresses a proposition, a question, a request, etc., and does not participate in any syntactic dependency. Usually contains a period or a question mark at the end.
`Quotation`	Describes a direct narrative or citation.
`synthesis`	Indicates the type of synthesis among the child entities and text contained within this entity.

Semantics of SyntacticConstituentType:

Name	Definition
`SyntacticConstituentType`	DS representing a single syntactic constituent. Namely, a syntactic entity that represents a semantic entity. In a big apple, for instance, big apple is a syntactic constituent but a big is not.

`Head`	Describes a syntactic constituent that may, but need not, be the head (semantic representative) of a larger constituent.
	The following example accommodates two interpretations: planes which are quickly flying and to fly planes quickly:
	`<Phrase>`
	`  <Head>`
	`  <Phrase>quickly<Phrase>`
	`  flying`

Name	Definition

	`  </Head>`
	`  planes`
	`</Phrase>`

`Phrase`	Describes a syntactic constituent that is not the head of any larger constituent.

`Quotation`	Describes a direct narrative or citation.
	For example, the following describes the sentence "'I quit,' said Sue."
	`<Sentence>`
	`  <Quotation>I quit</Quotation>,`
	`  <Head>said</Head>`
	`  <Phrase operator="agent">Sue</Phrase>.`
	`</Sentence>`

`term`	Identifies a term in a classification scheme that represents the semantics of this syntactic constituent.
`scheme`	Identifies the classification scheme from which `term` is taken.
`schemeLocator`	Indicates a location where the classification definition for `term` can be located.

`baseForm`	Indicates the base or uninflected form of the syntactic constituent.
	For example,
	`<Head baseForm="grow">grew</Head>`
	`<Head baseForm="dogs">dog</Head>`

`pronunciation`	Indicates the pronunciation of the syntactic constituent; for example using the International Phonetic Alphabet (IPA).

`edit`	Indicates the original text string replaced by the annotator with the text in the element. Its value is a colon followed by this string. It begins with a comma to avoid empty value in the case of insertion.
	For example, the following means that the annotator replaced the comma in the original text with a white space followed by `loves'.
	`<Sentence>`
	`  <Phrase>Tom loves Mary</Phrase>`
	`  and`
	`  <Phrase>`
	`    Bill<Head edit=”:,”> loves</Head> Sue`
	`  </Phrase>`
	`</sentence>`

`synthesis`	Indicates the type of synthesis among the child entities and text contained within this element.
`particle`	Function word (or string of words) representing the `operator`. For example, the particle of `on the beach' is on, which represents `location`. The particle of in order to escape is in order to, which represents `purpose`.

Semantics of synthesisType:

Name	Definition
`synthesisType`	Datatype representing the type of synthesis (combination) used to combine a set of linguistic entities.
`none`	No semantic relation among child entities and texts.
`dependency`	The synthesis is dependency. Each child element except one depends on a Head child in a full description. (Note that elements in the full description may appear as non-

element texts in corresponding partial description.) The default value for synthesis in the SyntacticConstituent DS. %sth missing??

The interpretation of the example below may be to fly planes ("planes" depends on "flying") or planes which are flying ("flying" depends on "planes"), because the default value for synthesis is dependency for Phrase elements. Note that the head is not specified, and is therefore left open here:

```
<Phrase>flying planes</Phrase>
```

When the head is specified uniquely and explicitly, the dependency relationships among the children are uniquely determined (where both the and good depends on idea):

```
<Phrase>
   <Phrase>the</Phrase>
   <Phrase>good</Phrase>
   <Head>idea</Head>
</Phrase>
```

forward

The synthesis is a forward dependency chain. Each child except one depends on the closest sibling non-Phrase element in a full description. The dependency should be forward (i.e. the governor should be to the right) if possible (there is a non-Phrase sibling element to the right in a full description). The dependency relationships among the child elements are uniquely determined in an element-only content. %not clear

In the following example, "quickly" depends on "flying" and "flying" depends on "planes".

```
<Phrase synthesis="forward">
    <Head>very</Head> %phrase?
    <Head>quickly</Head>
    <Head>flying</Head>
    <Head>planes</Head>
</Phrase>
```

Note, that using `forward` simplifies the description. For instance, the above description is much simpler than the following, though they are equivalent.

```
<Phrase>
  <Phrase>
    <Phrase>
    <Phrase>very</Phrase>
        <Head>quickly</Phrase> %head?
    </Phrase>
    <Head>flying</Head>
  </Phrase>
  <Head>planes </Head>
</Phrase>
```

backward

The synthesis is a backward dependency chain. Each child except one depends on the nearest sibling non-Phrase element in a full description. Contrary to `forward`, the dependency should be backward if possible. As with `forward`, the dependency relationships among the child elements are uniquely determined in an elment-only content.

In the following example, "eat" depends on "to" and "to" depends on "want", because "to" is the nearest potential head before "eat", and "want" is the nearest potential head before "to".

```
<Head synthesis="backward">
    <Head>want </Head>
    <Head>to </Head>
    <Head>eat </Head>
</Head>
```

As with the use of `forward`, `backward` simplifies the description:

```
<Head>
    <Head>want</Head>
    <Phrase>
```

```
        <Head>to </Head>
        <Phrase>
            <Head>eat </Head>
        </Phrase>
    </Phrase>
</Head>
```

`coordination`

Coordination. Used to refrain from indicating the type of coordination (collective or distributive). Each child element must be either a coordinate conjunction or a coordinant.

`apposition`

The synthesis is a sequence of linguistic entities (syntactic constituents or larger linguistic entities) with the same meaning. When two linguistic entities form an apposition structure, the latter is a paraphrase, elaboration, etc. of the former.

```
<Sentence>
    <Phrase>I</Phrase>
    <Head synthesis="apposition">
        <Head>
            introduced Mary to Sue,
        </Head>
        that is,
        <Head>
            <Head edit=":">introduced </Head>
            my girlfriend to my wife
        </Head>
    </Head>
</Sentence>
```

`repair`

The synthesis is a sequence of multiple linguistic entities where the last entity repairs the preceding erroneous ones.

```
<Sentence>
  I
  <Head synthesis="repair">
    <Head>
      gave Mary to the dog,
    </Head>
    oh, I'm sorry,
    <Head>
       <Head edit=":">gave </Head>
      the dog to Mary
    </Head>
  </Head>.
</Sentence>
```

`error`

The synthesis type is an error, which is the same as repair, except that all the child linguistic entities, including the last one, are erroneous.

`idiosyncratic`

The synthesis type is an idiosyncratic construction. For example, an idiomatic expression as shown below:

```
<Phrase synthesis="idiosyncratic">
  four over seven
</Phrase>
<Phrase synthesis="idiosyncratic">
  France versus Germany
</Phrase>
```

Linguistic DS Example

The Linguistic DS allows partial description of linguistic structure. The following shows a description of the sentence "You might want to suppose that flying planes may be dangerous."

```
<Sentence>
    You might want to suppose that
    <Phrase><Phrase>flying</Phrase> planes</Phrase>
    may be dangerous.
</Sentence>
```

This example specifies that flying depends on planes. The relations among the child entity and pieces of child texts are left undescribed; here it is just assumed that some dependencies hold among them, without committing to any further interpretation. On the other hand, the following description describes the syntactic structure for the same sentence in more detail. The DependencyStructure Datatype forces a little more detailed description than this:

```
<Sentence>
    <Phrase>You</Phrase>
    <Head>might</Head>
    <Phrase>
        <Head>want</Head>
        <Phrase>
            <Head>to</Head>
            <Phrase>
                <Head>suppose</Head>
                <Phrase>
                    <Head>that</Head>
                    <Phrase>
                        <Phrase>
                            <Phrase><Head>flying</Head></Phrase>
                            <Head>planes</Head>
                        </Phrase>
                        <Head>may be dangerous</Head>
                    </Phrase>
                </Phrase>
            </Phrase>
        </Phrase>
    </Phrase>.
</Sentence>
```

In this connection, the `synthesis` attribute addresses accurate characterizations of the type of combination among the child elements and texts, and thus simplifies the description.

The following series of examples illustrate a minimal (in terms of the number of tags) tagging that uniquely determines the syntactic structures and coreferences. The examples assume that words may be heads.

The first example describes the sentence "This is Akashi Channel Bridge, which connects Kobe City and Awaji Island of Hyogo Prefecture", which contains a relative clause:

```
<Paragraph>
    <Sentence synthesis="backward">
        <Phrase id="THIS">This</Phrase>
        is
        <Head id="ACB" synthesis="forward">Akashi Channel Bridge</Head>,
        <Phrase synthesis="backward">
            <Phrase equal="ACB">which</Phrase>
            connects
            <Head>
                <Phrase synthesis="forward">Kobe City</Phrase>
                and
                <Phrase synthesis="forward">Awaji Island</Phrase>
            </Head>
            of
            <Phrase synthesis="forward">Hyogo Prefecture</Phrase>
        </Phrase>.
    </Sentence>
</Paragraph>
```

The following example describes the sentences "Look. It's so big. It's the world's longest suspension bridge, whose length is about 4,000 meters," which contain an ellipsis, which is the object of `look' and is coreferent with `this' in the previous sentence.

```
<Sentences>
    <Sentence>
        Look.
        <Phrase equal="THIS" operator="object"/>
    </Sentence>

    <Sentence>
        <Phrase equal="THIS">It</Phrase>'s
        <Phrase synthesis="forward">so big</Phrase>.
    </Sentence>

    <Sentence synthesis="backward">
        <Phrase equal="ACB">It</Phrase>'s
        <Head id="WLSB" synthesis="forward">
            the
            <Phrase synthesis="forward">world's longest</Phrase>
            <Phrase>suspension</Phrase>
            bridge
        </Head>,
        <Phrase synthesis="backward">
            <Phrase>
                <Phrase equal="WLSB">whose</Phrase>
                length
            </Phrase>
            is about
            <Phrase synthesis="forward">4,000 meters</Phrase>
        </Phrase>.
    </Sentence>
</Sentences>
```

The last example shows the description of a cleft sentence "It's two wires which support the weight of the bridge, which is as much as 150,000 tons."

```
<Sentence>
    <Phrase id="IT">It</Phrase>'s
    <Phrase synthesis="forward">two wires</Phrase>
    <Phrase synthesis="backward" depend="IT">
        <Phrase equal="IT">which</Phrase>
        support
        <Phrase id="W">
            <Phrase>the</Phrase>
            weight
            <Phrase>
                of
                <Phrase equal="ACB" synthesis="forward">
                    the bridge
                </Phrase>
            </Phrase>,
            <Phrase synthesis="backward">
                <Phrase equal="W">which</Phrase>
                is as much as
                <Phrase synthesis="forward">150,000 tons</Phrase>
            </Phrase>
        </Phrase>
    </Phrase>.
</Sentence>
```