Anatomy of Annotation Schemes: Mapping to GrAF

Abstract

In this paper, we apply the annotation scheme design methodology defined in (Bunt, 2010) and demonstrate its use for generating a mapping from an existing annotation scheme to a representation in GrAF format. By way of illustration, we apply the mapping strategy to annotations from ISO-TimeML (Mani et al., 2004), PropBank (Palmer et al., 2005), and FrameNet (Baker et al., 1998).

1 Introduction

The Linguistic Annotation Framework (LAF, (Ide and Romary, 2004); ISO 24612, 2009) defines an abstract model for annotations together with an XML serialization of the model, the Graph Annotation Format (GrAF, (Ide and Suderman, 2007)). GrAF is intended to be a pivot format capable of representing diverse annotation types of varying complexity, guaranteeing *syntactic* consistency among the different annotations. GrAF does not address the issue of *semantic* consistency among annotation labels and categories; this is assumed to be handled by other standardization efforts such as ISOCat (Kemps-Snijders et al., 2009). ISOCat provides a set of *data categories* at various levels of granularity, each accompanied by a precise definition of its linguistic meaning. Labels applied in a user-defined annotation scheme should be mapped to these categories in order to ensure semantic consistency among annotations of the same phenomenon.

While the mapping of annotation labels to a common definition, coupled with the syntactic consistency guaranteed by GrAF, takes a giant step towards the harmonization of linguistic annotations, this is still not enough to ensure that these annotations are sufficiently compatible to enable merging, comparison, and manipulation with common software. For this, the conceptual structure of the annotation, in terms of the structural relations among the defined annotation categories, must also be consistent. It is therefore necessary to consider this aspect of annotation scheme design in order to achieve a comprehensive treatment of the requirements for full harmonization of linguistic annotations.

In (Bunt, 2010), a design methodology for semantic annotation schemes is proposed, developed during the ISO project "Semantic annotation framework, Part 1: Time and events" ("SemAF/Time", for short), which is currently nearing completion (see ISO DIS 24617-1, 2009). The methodology includes a syntax that specifies both a class of *representation structures* and a class of more abstract *annotation structures*. These two components of the language specification are called its *concrete* and *abstract syntax*, respectively. A distinguishing feature of the proposed methodology is that the semantics is defined for the structures of the abstract syntax, rather than for the expressions that represent these structures.

In this paper, we generalize the design methodology defined in (Bunt, 2010) and demonstrate its use for generating a mapping from an existing annotation scheme to a representation in GrAF format. By way of illustration, we apply the mapping strategy to annotations from ISO-TimeML (Mani et al., 2004), PropBank (Palmer et al., 2005), and FrameNet (Baker et al., 1998).

2 Background

The process of corpus annotation may consist of attaching simple *labels* to textual elements, such as part of speech and syntactic designations and named entity tags. For more complex types of annotation, annotations include a variety of additional information about linguistic features and relations. This is especially true for the kinds of semantic annotation that have recently begun

to be undertaken in earnest, including semantic role labeling (e.g., FrameNet and PropBank) and time and event annotation (e.g., TimeML). However, these annotation schemes are not always designed based on formal principles, and as a result, comparing or merging information–even from two schemes annotating the same phenomenon–can be difficult or impossible without substantial human effort.

A major source of difficulties in interpreting annotation scheme content is that information in the annotation is implicit rather than explicit, making (especially) structural relations among parts of the linguistic information ambiguous. This often results from the use of an impoverished representation scheme, which provides only minimal mechanisms for bracketing and association. Consider, for example, the two annotation fragments below, expressed with parenthetic bracketing, taken from a computational lexicon:

```
(1) (SUBC ((NP-TO-INF-LOC) (NP-PP)))
(2) (FEATURES ((NHUMAN) (COUNTABLE)))
```

In (1), the bracketed information is a list of alternatives, whereas in (2), it is a set of properties, but there is no way to *automatically* distinguish the two in order to process them differently. Another example comes from PropBank:

```
wsj/00/wsj_0003.mrg 13 6 gold have.03
vn--a 0:2-ARG0 6:0-rel 7:1-ARG1
10:1-ARGM-ADV
```

Because of the "flat" representation[1], it is impossible to *automatically* determine if the morphosyntactic descriptor "vn–a" is associated with the element annotated as "rel", vs. the "gold" descriptor that is (assumedly) associated with the entire proposition. In both of these examples, linguistically-informed humans have little difficulty determining the structure because of the knowledge they bring to the interpretation. This knowledge is then embedded in the processing software so that the data are processed properly; however, because it is not a part of the representation itself, it is not available to others who may develop software for other kinds of processing.

To avoid these problems, annotation scheme design in ISO projects is split into two phases: the specification of (1) an abstract model consisting of *annotation categories and structures* and (2) specification of (possibly multiple) *representation structures*. An abstract model of annotation structures is typically implemented via development of a "metamodel", i.e. a listing of the categories of entities and relations to be considered, often visualized by a UML-like diagram–i.e., a graph. Schemes described via this method are trivially mappable to GrAF, ensuring that syntactic consistency among the different schemes, whatever their original representation structures may be, is achievable. It also ensures that these schemes are trivially mappable to different representation formats that are used in various software systems, e.g., GATE, UIMA, NLTK, GraphViz, etc.

3 Anatomy of an annotation scheme

As specified in (Bunt, 2010), an annotation scheme consists of a syntax that specifies a class of more abstract annotation structures (the *abstract syntax*) and a class of representation structures (the *concrete syntax*), plus a semantics associated with the abstract syntax.

3.1 Abstract syntax

The abstract syntax of an annotation scheme defines the set-theoretical structures which constitute the information that may be contained in annotations. It consists of (a) a specification of the elements from which these structures are built up, called a *conceptual inventory*; and (b) *annotation construction rules*, which describe the possible combinations of these elements into annotation structures. The semantics of the annotation scheme components is defined for the annotation structures of the abstract syntax.

For example, a fragment of the ISO-TimeML[2] conceptual inventory includes:[3]

- finite sets of elements called event types, tenses, aspects, signatures, cardinalities, and veracities.
- finite sets of elements called temporal relations, duration relations, event subordination relations, aspectual relations, etc.

The annotation construction rules for ISO-TimeML specify how to construct two types of annotation structures: *entity structures* and

[1] In PropBank annotation, this information appears on a single line.

[2] All references to ISO-TimeML are based on the state of the project as documented in ISO 264617-1:2009(E) from September 2009.

[3] See (Bunt, 2010) for the full specification for ISO-TimeML.

link structures. One type of entity structure, called an *event structure*, is defined as a 6-tuple $\langle e, t, a, s, k, v \rangle$ where e is a member of the set of event types; t and a are a tense and an aspect, respectively; s is a signature (a set-theoretical type that is used for handling quantification over events); k is a cardinality, used for expressing information about the size of a set of events involved in a quantified relation; and v is a veracity, which is used to represent whether an event is claimed to have occurred, or claimed not to have occurred (for dealing with positive and negative polarity, respectively), or to have yet another status such as 'possibly' or 'requested', for handling such cases as *Please come back later today.* A *time-amount structure* is a pair $\langle n, u \rangle$ or a triple $\langle R, n, u \rangle$, where n is a real number, R a numerical relation, and u a temporal unit. The rules also define a link structure called an *time anchoring structure* as a triple $\langle event\ structure, time\text{-}amount\ structure, duration\ relation \rangle$.

3.2 Concrete syntax

The concrete syntax provides the representation of annotation structures defined in the abstract syntax. A concrete syntax is said to be *ideal* for a given abstract syntax if there is a one-to-one correspondence between the structures defined by the abstract syntax and those defined by the concrete syntax. An ideal concrete syntax RF_1 defines a function F_1 from annotation structures to RF_i-representations, and an inverse function F_i^{-1} from RF_1-representations to annotation structures. In other words, the abstract and the concrete syntax are *isomorphic*. Since this holds for any ideal concrete syntax, it follows that any two ideal representation formats are isomorphic. Given two ideal representation formats RF_i and RF_j we can define a homomorphic mapping C_{ij} from RF_i-representations to RF_j-representations by

(1) $C_{ij} =_D F_j \circ F_i^{-1}$, i.e. $C_{ij}(r) = F_j(F_i^{-1}(r))$
for any RF_i-representation r

and conversely, we can define a homomorphic mapping C_{ji} from RF_j-representations to RF_i-representations by

(2) $C_{ji} =_D F_i \circ F_j^{-1}$, i.e. $C_{ji}(r) = F_i(F_j^{-1}(r))$
for any RF_j-representation r

These two mappings constitute *conversions* from one format to the other, that is, they constitute one-to-one meaning-preserving mappings: if $\mu(r)$ denotes the meaning of representation r, then $\mu(C_{ij}(r)) = \mu(r)$ for any F_i-representation r, and conversely, $\mu(C_{ji}(r')) = \mu(r')$ for any F_j-representation r'.

```
<isoTimeML-ICS1rep xml:id="a1">
 <EVENT xml:id="e1" anchor="t2"
    type ="FAST" tense=PAST
    signature="INDIVIDUAL"/>
 <TIME-AMOUNT xml:id="time1"
    anchor="t4" numeral="2" unit="day"/>
 <TIME-ANCHORING
    anchoredEvent="e1"
    anchorTime="time1" relType="FOR"/>
</isoTimeML-ICS1rep>
```

Tokens: $[I_{t1}][fasted_{t2}][for_{t3}][two_{t4}][days_{t5}]$.

Figure 1: ISO-TimeML_ICS1 annotation

Figure 1 shows a rendering of the sentence *I fasted for two days* using a concrete XML-based syntax for the annotation structures defined by the ISO-TimeML abstract syntax, called the ICS-1 format, as described in (Bunt, 2010).

4 GrAF overview

GrAF is an exchange or pivot format intended to simplify the processes of merging of annotations from different sources and using annotations with different software systems. The underlying data model is a directed acyclic graph, which is isomorphic to UML-like structures that may be used to define an abstract syntax for a given annotation scheme, as described in section 3.

GrAF is an XML serialization of a formal graph consisting of nodes and edges, either or both of which are decorated with feature structures. Nodes may have edges to one or more other nodes in the graph, or they may be linked directly to *regions* within the primary data that is being annotated. The feature structure attached to a node or edge provides the *content* of the annotation–that is, the associated linguistic information expressed as a set of attribute-value pairs. The feature structures in GrAF conform to formal feature structure specifications and may be subjected to operations defined over feature structures, including subsumption and unification. As a result, any representation of an annotation in GrAF must consist of a feature structure that provides all of the relevant linguistic information.

Figure 2 shows a fragment of a FrameNet frame element annotation, serialized in GrAF XML. It

```
<node xml:id="fn-n1"/>
<a label="FE" ref="fn-n1" as="FrameNet">
    <fs>
       <f name="FE" value="Recipient"/>
       <f name="GF" value="Obj"/>
       <f name="PT" value="NP"/>
   </fs>
</a>

<edge id="e1" from="fn-n1"
                 to="fntok-n5"/>
```

Figure 2: FrameNet frame element annotation in GrAF

consists of a graph node with id "fn-n1" and an annotation with the label "FE"[4]. The *ref* attribute on the <a> (annotation) element associates the annotation with node "fn-n1". The annotation contains a feature structure with three features: *FE* (Frame element), *GF* (Grammatical Function), and *PT* (Phrase Type). An edge connects the node to another node in the graph with the id "fntok-n5" (not shown here), which is associated with annotation information for a token that in turn references the span of text in primary data being annotated.

5 Mapping to GrAF

LAF specifies that an annotation representation R is *valid* if it is mappable to a meaning-preserving representation in GrAF, and that its GrAF representation is in turn mappable to R. In terms of the definitions in section 3, a *LAF-valid representation* R is one where $\mu(R) = \mu(C_{RG}(R))$ and $\mu(G) = \mu(C_{GR}(G))$, where G is a GrAF representation. We can also define a valid annotation scheme in terms of *conversion transitivity* through GrAF; that is, for two arbitrary annotation schemes R and S, the following holds:

$$\mu(R) = \mu(C_{RG}(R)) = \mu(C_{GS}(S))$$

Our goal here is to provide a formal specification for the mapping function C_{RG}, assuming the existence of a formal specification of an annotation scheme as outlined in section 3. To accomplish this, it is necessary to identify the two components of an abstract syntax for annotation scheme R: the conceptual inventory and the annotation construction rules that indicate how elements of the conceptual inventory are combined into *annotation structures*–specifically, *entity structures*, which describe annotation objects, and *link structures*, which describe relations among entity structures. Once these are available, a general procedure for establishing a GrAF representation of the annotation structures is as follows:

For each type of entity structure e:

- introduce a label L_e, where L_e is the entity structure type;
- define a set of features f corresponding one-to-one with the components of the n-tuple of elements from the conceptual inventory defining entity structure e.

A link structure is a triple $\langle E_1, E_2, r\rangle$ consisting of two sets of entity structures and a relational element defining a relation between them. For each type of link structure:

1. introduce a label L_r, where L_r is the type name of relation r.
2. If r is associated with a set of elements from the conceptual inventory, then features are created as in (2), above.

In GrAF, an annotation A consists of a label L and a feature structure containing a set of features f. Annotations may be associated with nodes or edges in the graph. Typically, entity structures are associated with nodes that have links into a region of primary data or one or more edges connecting it to other nodes in the graph. Link structures are associated with edges, identifying a relation among two or more entity structures. In the simplest case, a link structure consists of a relation between two entity structures, each of a given type; in the corresponding GrAF representation, the link structure label is associated with an edge d that connects nodes n_1, n_2, each of which is decorated with annotations labeled L_1, L_2, respectively.

For example, for the ISO-TimeML abstract syntax fragment provided in section 3, we define the labels EVENT and INSTANT corresponding to the two entity structures with names *event structure* and *time amount structure*, and a link structure TIME-ANCHORING. Because an event structure is defined as a 6-tuple $\langle e, t, a, s, k, v\rangle$, we define six features *event, tense, aspect signature, cardinality,* and *verac-*

[4]Note that the value of the *label* attribute is, for practical purposes, a convenience; it is used primarily when generating alternative representation formats.

ity.[5] A time-amount structure may be a pair $\langle n, u \rangle$ or a triple $\langle R, n, u \rangle$, where n is a real number, R a numerical relation, and u a temporal unit, so we introduce features *numeral, unit*, and *relType*. Finally, the time anchoring link structure is a triple $\langle event\ structure, time\text{-}amount structure, duration\ relation \rangle$. In this case, the first two elements of the triple are the entity structures being linked; these will be represented as nodes in the GrAF implementation. Therefore, the only attribute of TIME-ANCHORING is *relType*. The label and features associated with each entity and link structure provide the template for an annotation corresponding to that structure with appropriate values filled in, which may then be associated with a node or edge in the graph.

5.1 ISO-TimeML example

The GrAF representation of the ISO-TimeML annotation for the sentence *I fasted for two days* is shown in Figure 3, based on the abstract syntax given in section 3.1.

To create an annotation corresponding to an ISO-TimeML entity structure, a node `<node>` element) is created and assigned a unique identifier as the value of the XML attribute *xml:id*. An *annotation* (`<a>`) element is also created, with a *label* attribute whose value is the entity structure name, and which contains a feature structure providing the appropriate feature/value pairs for that entity structure. The annotation is associated with the node by using the node's unique identifier as the value of the *ref* attribute on the `<a>` element. An edge is then created from the node to another node in the graph that references the data to be annotated–in this case, one or more tokens defined over regions of the primary data.

ISO-TimeML link structures define a relation between two entity structures, and therefore is rendered in GrAF as a labeled edge between the nodes annotated with the entity structure information. In the ISOTimeML example, an annotation with label TIME-ANCHORING is created with a single feature *relType*. The *from* and *to* attributes on the `<edge>` element link the node with the EVENT entity structure annotation (node `tml-n1` in the example) to the node with the TIME-AMOUNT annotation (`tml-n2`). This edge is then associated with the TIME-ANCHORING annotation.

Figure 1 shows the rendering of the ISO-TimeML abstract syntax in the ICS-1 concrete syntax. Following Section 3.2, these two realizations of the abstract syntax for ISO-TimeML are isomorphic.

```
<node xml:id="tml-n1"/>
<a label="EVENT" ref="tml-n1"
  as="TimeML">
 <fs>
   <f name="event" value="fast"/>
   <f name="tense" value="Past"/>
   <f name="signature"
      value="individual"/>
 </fs>
</a>

<edge xml:id="tml-e1" from="tml-n1"
   to="r2"/>

<node xml:id="tml-n2"/>
<a label="TIME-AMOUNT" ref="tml-n2"
   as="TimeML">
 <fs>
   <f name="numeral" value="2"/>
   <f name="unit" value="day"/>
 </fs>
</a>

<edge xml:id="tml-e2" from="tml-n2"
   to="t4"/>
<edge xml:id="tml-e3" from="tml-n2"
   to="t5"/>

<edge xml:id="tml-e4" from="tml-n1"
   to="tml-n2"/>
<a label="TIME-ANCHORING" ref="tml-e4"
  as="TimeML">
  <fs>
    <f name="relType" value="FOR"/>
  </fs>
</a>
```

Tokens: $[I_{t1}][fasted_{t2}][for_{t3}][two_{t4}][days_{t5}]$.

Figure 3: ISO-TimeML annotation in GrAF

5.2 Reverse engineering the abstract syntax

The previous two sections show how schemes for which an abstract syntax is specified can be rendered in GrAF as well as other concrete syntax representations. However, as noted in section 2, many annotation formats–especially legacy formats–were not designed based on an underlying data model. Therefore, in order to achieve a mapping to GrAF, it is necessary to "reverse engineer" the annotation format to define its abstract

[5] The latter three attributes have the default values INDIVIDUAL, 1, and POSITIVE, respectively, and will be omitted in the examples to follow if they have these values.

syntax. Because of problems such as those outlined in Section 2, this exercise may require some extrapolation of information that is implicit, or not specified, in the original annotation format. We provide two examples below, one for PropBank and one for FrameNet.

5.2.1 An abstract syntax for PropBank

The PropBank format specifies an annotation for a sentence consisting of several columns, specifying the file path; the sentence number within the file; the number of the terminal in the sentence that is the location of the verb; a status indication; a frameset identifier (frame and sense number); an inflection field providing person, tense, aspect, voice, and form of the verb; and one or more "proplabels" representing an annotation associated with a particular argument or adjunct of the proposition. Proplabels are associated with primary data via reference to the Penn Treebank (PTB) node in the syntax tree of the sentence.

Based on this we can specify a portion of a PropBank conceptual Inventory:

- a special proposition type $verb$, designating the verb (replaces PropBank "rel");
- a finite set $PROP = \{ARGA, ARGM, ARG0, ARG1, ARG2\}$ of proposition labels;
- a finite set $FEAT = \{EXT, DIR, LOC, TMP, REC, PRD, NEG, MOD, ADV, MNR, CAU, PNC, DIS\}$, plus the set of prepositions and "null", comprising the set of features;
- a finite set of sets $INF = \{form, tense, aspect, person, voice\}$, where $form = \{infinitive, gerund, participle, finite\}$, $tense = \{future, past, present\}$, $aspect = \{perfect, progressive, both\}$, $person = \{3rd\}$, and $voice = \{active, passive\}$.
- a finite set $FrameSets = \{fs_1, fs_2, ...fs_n\}$ where each fs_i is a frame set defined in PropBank.

An abstract syntax for PropBank could specify the following annotation construction rules:

- a *proposition entity structure* is a pair $\langle f, A \rangle$ where f is a frameset and A is a set of *argument entity structures*.[6]
- an *argument entity structure* is an argument $a \in PROP \times FEAT$.
- a *verb entity structure* is a 5-tuple $\langle f, t, a, p, v \rangle$ where $f \in form$, $t \in tense$, $a \in aspect$, $p \in person$, and $v \in voice$.

Based on this, the PropBank annotation in Section 2 can be rendered into a concrete syntax; in this case, in GrAF as shown in Figure 4. Note that the *to* attribute on `<edge>` elements have as values the reference to PTB nodes from the original PropBank encoding; in GrAF, these values would be identifers on the appropriate nodes in a GrAF representation of PTB. We have also included role names (e.g., "owner") in the annotation, which are not present in the original; this was done for convenience and readability, and the values for the "role" feature could have been given as arg-0, arg-1, etc. instead.

The original PropBank encoding is close to an ideal concrete syntax, as it can be generated from the abstract syntax. However, the round trip back to the abstract syntax is not possible, because it is necessary to do some interpretation of associations among bits of annotation information in order to construct the abstract syntax and, subsequently, map the PropBank format to GrAF. Specifically, in the GrAF encoding the inflection information is associated with the node referencing the verb, but this association is not explicit in the original (and in fact may not be what the annotation scheme designers intended).

5.2.2 An abstract syntax for FrameNet

The FrameNet XML format is shown in Figure 5.[7] The structure and content of this encoding is highly oriented toward a presentation view, intended to support display of the sentence and frame elements in a browser.

A partial abstract syntax for FrameNet derived from this format includes the following conceptual inventory:

- a $Target$, designating the frame-evoking lexical unit;

[6] We do not include the bookkeeping information associated with a PropBank annotation in the abstract syntax.

[7] Some detail concerning the html display has been omitted for brevity.

```
<node xml:id="pb-n1"/>
<a label="Proposition" ref="pb-n1"
 as="PropBank">
 <fs>
  <f name="file"
     value="wsj/00/wsj_0003.mrg"/>
  <f name="sentenceNo" value="13"/>
  <f name="verbOffset" value="6"/>
  <f name="status" value="gold"/>
  <f name="frameSet"
     value="have.03"/>
 </fs>
</a>

<node xml:id="pb-n2"/>
<a label="VERB" ref="pb-n2"
 as="PropBank">
 <fs>
  <f name="role" value="rel"/>
  <f name="form" value="finite"/>
  <f name="tense" value="present"/>
  <f name="voice" value="active"/>
 </fs>
</a>

<edge xml:id="pb-e1" from="pb-n1"
   to="pb-n2"/>
<edge xml:id="pb-e2" from="pb-n2"
   to="ptb-6-0"/>

<node xml:id="pb-n3"/>
<a label="ARG0" ref="pb-n3"
 as="PropBank">
 <fs>
  <f name="role" value="owner"/>
 </fs>
</a>

<edge xml:id="pb-e3" from="pb-n1"
   to="pb-n3"/>
<edge xml:id="pb-e4" from="pb-n3"
   to="ptb-0-2"/>

<node xml:id="pb-n4"/>
<a label="ARG1" ref="pb-n4"
 as="PropBank">
 <fs>
  <f name="role" value="possession"/>
 </fs>
</a>

<edge xml:id="e5" from="pb-n1"
   to="pb-n4"/>
<edge xml:id="e6" from="pb-n4"
   to="ptb-7-1"/>

<node xml:id="pb-n5"/>
<a label="ARGM" ref="pb-n5"
 as="PropBank">
 <fs>
  <f name="role" value="adjunct"/>
  <f name="feature" value="adverbial"/>
 </fs>
</a>

<edge xml:id="e7" from="pb-n1"
   to="pb-n5"/>
<edge xml:id="e8" from="pb-n5"
   to="ptb-10-1"/>
```

Figure 4: PropBank annotation in GrAF

- a finite set $FE = \{Recipient, Supplier, Means, ...\}$ of frame element labels;
- a finite set $GF = \{Obj, Ext, Dep, ...\}$ of grammatical functions.
- a finite set $PT = \{NP, PP, ...\}$ of phrase types.
- a finite set $LU = \{u_1, u_2, ...u_n\}$ where each u_i is a lexical unit.
- a finite set $POS = \{n, v, a, r\}$ denoting parts of speech;
- a finite set $FrameNames = \{f_1, f_2, ...f_n\}$ where each f_i is a frame defined in FrameNet.

An abstract syntax for this partial inventory could specify the following annotation construction rules:

- a *frame entity structure* is a pair $\langle f, A \rangle$ where f is a frame name, u is a lexical unit, and F is a set of *frame element (FE) entity structures*.
- an *FE entity structure* is a triple $\{f, g, p\}, f \in FE, g \in GF, p \in PT$.

The GrAF rendering of the abstract syntax is given in Figure 6, which was generated from the FrameNet abstract syntax using the rules outlined in section 5. Both the FrameNet XML and the GrAF rendering provide an ideal concrete syntax because they are isomorphic[8] to the abstract syntax and, by the definition in section 3.2, are conversions of one another.

6 Conclusion

In this paper we outlined a methodology for annotation scheme design and development; demonstrated how schemes designed using this methodology may be easily mapped to GrAF; and demonstrated how "reverse engineering" an annotation format whose abstract syntax is unspecified can provide the information required to map that format to GrAF. This work was undertaken with two goals in mind: (1) to provide a formal method for mapping to GrAF; and (2) to demonstrate the advantages of a methodology for annotation scheme design that is based on an abstract model, as

[8] Obviously, in the FrameNet XML additional elements are introduced for display and bookkeeping purposes.

adopted in ISO TC37 SC4 projects and formalized in (Bunt, 2010). The ultimate goal is, of course, to achieve harmonization of annotation formats, so that they can be merged, enabling the study of interactions among information at different linguistic levels; compared, in order to both evaluate and improve automatic annotation accuracy; and to enable seamless transition from one software environment to another when creating and using linguistic annotations.

```
<annotationSet lexUnitRef="11673"
  luName="provide.v" frameRef="1346"
  frameName="Supply"
  status="MANUAL" ID="2022935">
 <layer rank="1" name="Target">
  <label end="109" start="103"
     name="Target"/>
 </layer>
  <layer rank="1" name="FE">
  <label bgColor="0000FF" ... end="138"
    start="111" name="Recipient"/>
  <label bgColor="FF0000"... end="84"
    start="83" name="Supplier"/>
  <label bgColor="FF00FF"... end="79"
    start="0" name="Means"/>
 </layer>
  <layer rank="1" name="GF">
  <label end="138" start="111"
    name="Obj"/>
  <label end="84" start="83"
    name="Ext"/>
  <label end="79" start="0"
    name="Dep"/>
 </layer>
 <layer rank="1" name="PT">
  <label end="138" start="111"
    name="NP"/>
  <label end="84" start="83"
    name="NP"/>
  <label end="79" start="0" name="PP"/>
 </layer>
...
</annotationSet>
```

Figure 5: FrameNet XML format

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA. Association for Computational Linguistics.

Harry Bunt. 2010. A methodology for designing semantic annotation languages exploiting semantic-syntactic isomorphisms. In *Proceedings of the Second International Conference on Global Interoperability for Language Resources (ICGL2010)*, pages


```
<node xml:id="fn-as1"/>
<a label="annotationSet" ref="fn-as1"
  as="FrameNet">
 <fs>
  <f name="lexUnitRef" value="11673"/>
  <f name="luName" value="provide.v"/>
  <f name="frameRef" value="1346"/>
  <f name="frameName" value="Supply"/>
  <f name="status" value="MANUAL"/>
  <f name="ID" value="2022935"/>
 </fs>
</a>

<node xml:id="fn-n1"/>
<a label="Target" ref="fn-n1"
  as="FrameNet">
 <fs>
  <f name="name" value="Target"/>
 </fs>
</a>
<edge xml:id="e69" from="fn-as1"
   to="fn-n1"/>
<edge xml:id="e90" from="fn-n1"
  to="fn-t1"/>

<node xml:id="fn-n2"/>
<a label="FE" ref="fn-n2"
  as="FrameNet">
 <fs>
  <f name="FE" value="Recipient"/>
  <f name="GF" value="Obj"/>
  <f name="PT" value="NP"/>
 </fs>
</a>
<edge xml:id="e67" from="fn-as1"
   to="fn-n2"/>
<edge xml:id="e91" from="fn-n2"
  to="fn-t2"/>

<node xml:id="fn-n3"/>
<a label="FE" ref="fn-n3"
  as="FrameNet">
<fs>
 <f name="FE" value="Supplier"/>
 <f name="GF" value="Ext"/>
 <f name="PT" value="NP"/>
</fs>
</a>
<edge xml:id="e46" from="fn-as1"
   to="fn-n3"/>
<edge xml:id="e92" from="fn-n3"
  to="fn-t3"/>

<node xml:id="fn-n4"/>
<a label="FE" ref="fn-n4"
  as="FrameNet">
<fs>
 <f name="FE" value="Means"/>
 <f name="GF" value="Dep"/>
 <f name="PT" value="PP"/>
</fs>
</a>
<edge xml:id="e10" from="fn-as1"
   to="fn-n4"/>
   <edge xml:id="e93" from="fn-n4"
  to="fn-t4"/>
```

Figure 6: FrameNet in GrAF format

29–46, Hong Kong SAR. City University of Hong Kong.

Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Journal of Natural Language Engineering*, 10(3–4):211–225.

Nancy Ide and Keith Suderman. 2007. GrAF: A graph-based format for linguistic annotations. In *Proceedings of the First Linguistic Annotation Workshop*, pages 1–8, Prague.

Marc Kemps-Snijders, Menzo Windhouwer, Peter Wittenburg, and Sue Ellen Wright. 2009. ISOcat : Remodelling metadata for language resources. *International Journal of Metadata and Semantic Ontologies*, 4(4):261–276.

Inderjeet Mani, James Pustejovsky, and Beth Sundheim. 2004. Introduction to the special issue on temporal information processing. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):1–10.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, March.