A note on the definition of semantic annotation languages

Harry Bunt and Chwhynny Overbeeke
harry.bunt@uvt.nl, info@chwhynny.com

1 Introduction

In the last few years, the international organization for standards ISO has started up various projects concerned with the definition of interoperable concepts for syntactic, morphosyntactic, and semantic annotation, with the ultimate aim to support the development of interoperable language resources. The Linguistic Annotation Framework (LAF, Ide & Romary, 2004) thereby serves as a meta-framework. LAF distinguishes between the concepts of *annotation* and *representation*: 'annotation' refers to the process of adding information to segments of language data, or to that information itself, independent of the format in which this information is represented. The term 'representation' refers to the format in which an annotation is rendered, for instance in XML. According to LAF, *annotations* are the proper level of standardization.

This distinction is reflected in the specification of ISO-TimeML, a proposed ISO standard for temporal annotation (ISO, 2008) which consists of an *abstract syntax*, a *concrete syntax*, and a semantics. The abstract syntax specifies the elements making up the information in annotations, and how these elements may be combined to form complex annotation structures; these combinations are defined as set-theoretical structures. The concrete syntax is a variant of the TimeML markup language (Pustejovsky et al., 2003). Any other representation that is a rendering of the abstract syntax can be converted into this representation. The ISO-TimeML semantics is associated with its *abstract* syntax, which explains why all concrete representations of ISO-TimeML annotations are semantically equivalent.

In this note we argue that the distinction of an abstract and a concrete syntax level is desirable not only from a standardization point of view, but also for designing annotation languages with a representation that is conceptually transparent for annotators and that allows a simple, systematic

Proceedings of the 8th International Conference on Computational Semantics, pages 268–271,
Tilburg, January 2009.

interpretation. We illustrate this for the annotation and interpretation of expressions denoting dates, times, and durations.

1.1 ISO-TimeML

The abstract syntax of ISO-TimeML consists of two parts: (a) a 'conceptual inventory', specifying the elements from which annotations are built up; and (b) a set of syntax rules which describe the possible combinations of these elements.

a. Conceptual inventory The concepts that can be used to build ISO-TimeML annotations fall into the following five categories, all formed by finite sets, plus the concepts of real and natural numbers.

- finite sets of elements called 'event classes'; 'tenses', 'aspects', 'polarities', and 'set-theoretic types' ;
- finite sets of elements called 'temporal relations','duration relations', 'numerical relations, 'event subordination relations', and 'aspectual relations';
- a finite set of elements called 'time zones';
- finite sets of elements called 'calendar years', 'calendar months', 'calendar day numbers'; 'clock times';
- a finite set of elements called 'temporal units'.

b. Syntax rules Annotation structures in ISO-TimeML come in two varieties, *entity structures* and *link structures*. Entity structures contain semantic information about a segment of source text; link structures describe semantic relations between segments of source text.

The simplest kind of ISO-TimeML structures are a single entity structure, which is a pair $< m, a >$ consisting of a markable[1] m and an annotation a, or a single link structure $< e_1, e_2, R >$ which relates two entity structures. More complex annotation structures consist of a set of entity structures and a set of link structures which link the entity structures together.

Entity structures come in 6 types, containing information about (1) events; (2) temporal intervals; (3) time points (or "instants"); (4) amounts of time; (5) frequencies of events; and (6) temporal relations. We focus here on the tree types of temporal concepts: intervals, instants, and amounts of time.

1. An *instant structure* is either a triple $<$ *time zone*, *date*, *clocktime* $>$, where a *date* is a triple consisting of a calendar year, a calendar month, and a calendar day number; or a triple $<$ *time-amount structure*, *instant structure*, *temporal relation*$>$ (*"an hour before midnight"*).

[1]The term *markable* is used to refer to the entities that the annotations are associated with. There are two kinds of markables in ISO-TimeML: *event markables* and *time markables*, corresponding to segments of primary data that describe events, and to those that describe temporal entities or relations, respectively.

2. An *interval structure* is either:
   (a) a pair $< t_1, t_2 >$ of two instant structures (beginning and end);
   (b) a calendar year, a pair consisting of a calendar year and a calendar month, or a triple $< cal.year, cal.month, cal.daynumber >$;
   (c) a triple $<$ *time-amount structure*, *interval structure*, *temporal relation* $>$ ("three weeks before Christmas");
   (d) a triple $< t_1, t_2, R >$ where t_1 and t_2 are either instant or interval structures, and where R is a duration relation (*"from '92 until '95"*).
3. A *time-amount structure* is a pair $< n, u >$, where n is a real number and u a temporal unit, or a triple $< R, n, u >$, where R is a numerical relation (like *greater than*) and n and u as before;

Link structures specify the temporal anchoring of events in time; the temporal ordering of events, intervals or instants; the length of an interval; subordination relations between events; and aspectual relations between events.

The semantics associated with this abstract syntax defines a mapping from the set-theoretical structures defined by the abstact syntax to the language of first-order predicate logic with lambda abstraction.

A *concrete syntax* in general consists of the specification of names for the various sets that make up the conceptual vocabulary, plus a listing of specific named elements of these sets, and for each rule of the abstract syntax a specification of how to represent the constructed annotation structure. The TimeML-based concrete syntax that is part of the ISO-TimeML specification makes use of a TIMEX3 tag to mark up explicit temporal expressions like dates, times and durations. Using this tag, the different types of temporal expressions are represented by means of the attribute `type`. An attribute called `value` has alphanumerical string values that follow a standard format to represent (combinations of) calendar days, weeks, months and years (`2007-03-16`); clock hours, minutes and seconds; (`T13:15:00`), as well as amounts of time (`P60D`) and frequencies. This representation does not have a transparent relation to the conceptual distinctions made in the abstract syntax.

A more transparent representation can be obtained by defining a concrete syntax where the categories (sets) of the conceptual inventory correspond to XML tags, and elements in these sets to attribute values. This gives annotation representations that wear there meaning on their sleeve, which is optimal both for human annotators and for computing the formal interpretation of the annotations. The following examples illustrate this, where we show, for three types of temporal expressions, (a) the conceptual annotation structure; (b) the TimeML-based representation; (c) an XML representation that directly instantiates the conceptual structure; (d) the

formal interpretation.[2] In all cases, the representations (c) are intuitively more transparent than the (b) ones, and have a more straightforward relation to the interpretations (d).

(1) March 2007 [$m_1 = w_1w_2$, w_1 ="March", w_2 ="2007"]
a. < m_1, <*interval*, <2007, *march*>>
b. `<TIMEX3 id="t1" type="DATE" value="207-03-XX"'>`
c. `<INTERVAL id="t1" calYear="2007" calMonth="MARCH">`
d. λt.INTERVAL(t) ∧ Calyear(t)=2007 ∧ Calmonth(t)=march

(2) Twelve-thirty tomorrow
[$m_1 = w_1w_2$, w_1 ="Twelve-thirty", w_2 ="tomorrow"]
a. < m_1, <*instant*, <2009, *january*, 8>,1230>>
b.`<TIMEX3 id="t1" type="TIME" value="T12:30">`
c. `<TIME id="t1" calYear="2009" calMonth="JANUARY" calDayNum="8" clockTime="1230" >`
d. λt.TIME(t) ∧ Calyear(t)=2009 ∧ Calmonth(t)=january ∧ Caldaynum(t)=8 ∧ Clocktime(t)=1230

(3) two-and-a-half minutes [$m_1 = w_1w_2$, w_1 ="Two-and-a-half", w_2 ="minutes"]
a. < m_1, <*time-amount*, <2.5, *minute*>>
b. `<TIMEX3 id="t1" type="DURATION" value="P2.5M">`
c. `<TIMEAMOUNT id="a1" num="2.5" unit="minute">`
d. λx.TIME-AMOUNT(x) ∧ Number(x)=2.5 ∧ Unit(x)=minute

References

[1] Bunt, H.C, Overbeeke, C. (2008) An Extensible Compositional Semantics for Temporal Annotation. In: *Proceedings of LAW II, the Second Workshop on Linguistic Annotation*, Satellite workshop at LREC 2008. Paris: ELRA.

[2] Bunt, H.C., Romary, L. (2002) Requirements on multimodal semantic representations. In *Proceedings of ISO TC37/SC4 Preliminary Meeting*, Seoul, 59-68.

[3] Ide, N., Romary, L. (2004) International Standard for a Linguistic Anootation Framework. *Natural language Engineering*, 10: 211-225.

[4] ISO (2008) *ISO Draft International Standard 24617-1 "Semantic annotation framework Part 1: Time and events"*. Geneva: ISO.

[5] Pustejovsky, J., Castano, J., Ingria, R., Gaizauskas, R., Katz, G., Sauri, R., Setzer, A. (2003) TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings IWCS-5*, Tilburg, pp. 337-353

[2]Depending on the semantic interpretation framework in which this interpretation is embedded, the semantic representations may be slightly different; e.g. Bunt & Overbeeke (2008) assign to the first example the representation $\lambda P.\exists t$.INTERVAL(t) ∧ Calyear(t)=2007 ∧ Calmonth(t)=march ∧ P(t).