A METHODOLOGY FOR DESIGNING SEMANTIC ANNOTATION LANGUAGES EXPLOITING SEMANTIC-SYNTACTIC ISO-MORPHISMS

Harry Bunt
Tilburg Center for Creative Computing
Dept. of Communication & Information Sciences,
Tilburg University, The Netherlands,
harry.bunt@uvt.nl

Abstract

This paper presents a methodology for the design of languages for semantic annotation. Central in this methodology is the specification of a representation format as a rendering of conceptual structures defined by an *abstract syntax*, which specifies in set-theoretical terms the possible information contents of annotations. The notion of an *ideal* representation format is introduced, which is related through an isomorphism to the set of conceptual structures, defined by the abstract syntax. This guarantees that every conceptual structure has a unique representation, and that every representation is the rendering of a unique conceptual structure. Moreover, the semantics of the annotation language is defined for the *abstract* syntax and is shared by all its representations. It is shown that this guarantees that every ideal representation format is convertible through a meaning-preserving mapping to any other ideal representation format.

The methodology is illustrated by its application to two ongoing ISO projects, concerned with the establishment of standards for the annotation of text with (a) information about time and events, and (b) information about communicative functions in dialogue.

1 Introduction

Language technology, computational linguistics, and present-day linguistics rely on large-scale annotated corpora. The process of making annotations is usually viewed as the attachment of certain labels to textual elements, such as part of speech labels or named entity tags, and is not commonly thought of as involving the use of an annotation *language*. But when it comes to semantic annotation, the situation is different. For example, Pustejovsky et al. (2003; 2005; 2007) have developed the XML-based language TimeML for the annotation of texts with information relating to time and events. The complexity of this kind of information is such that the annotations do not take the form of simple labels, but of expressions in a formal language.

The specification of a formal language usually consists of two parts: a syntax and a semantics. The syntax defines a class of expressions; the semantics describes what these expressions mean. This is commonly found in the definition of the languages of logic and computer science. An exception is the specification of XML, which consists of a syntax only and leaves its interpretation to the understander. For many applications of XML, such as syntactic or morphosyntactic annotation, this is good enough, but when it comes to semantic annotation, the lack of a semantics raises a serious question: since semantic annotations are meant to capture part of the meaning of the annotated text, if the annotations don't have a well-defined meaning, then why would they capture meaning in text better than the text itself? Bunt & Romary (2002) have therefore formulated the requirement of *semantic adequacy* for a semantic annotation language: it should have a well-defined semantics.

Another methodological requirement for the design of annotation languages comes from the ISO Linguistic Annotation Framework (LAF, Ide & Romary, 2004; ISO 24612, 2009). This framework draws a distinction between the concepts of *annotation* and *representation*. The term 'annotation' refers to the linguistic information that is added to segments of language data, independent of the format in which the information is represented. The term 'representation' refers to the format in which an annotation is rendered, for instance in XML, independent of its content. According to LAF, *annotations* are the proper level of standardization, rather than *representations*.

In order to deal with both requirements, we propose a design methodology for semantic annotation languages which includes a syntax that specifies besides a class of *representation structures* also a class of more abstract *annotation structures*. These two components of the language specification are called its *concrete* and *abstract syntax*, respectively. The concrete syntax defines a particular rendering of the annotation structures. Moreover, a semantics is specified which is defined for the *abstract* rather than for the concrete syntax.

More specifically, the proposed methodology consists of carrying out the following four steps, with as many feedback loops as may be desired until the end result is considered satisfactory:

1. establish a conceptual view of the information to be captured in annotations; such a view is sometimes called a 'metamodel' and often visualized as a UML diagram;
2. articulate the conceptual view in the form of a formal specification of its categories of entities, relations, and information structures that can be built up with them (an 'abstract syntax');
3. provide a formal semantics for the structures defined by the abstract syntax;
4. specify a representation format for the structures defined by the abstract syntax (a 'concrete syntax').

This methodology has been 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), and has played a part in this project without having been applied systematically. In this paper we will illustrate the methodology on the one hand by showing how its more systematic application can be used to improve the design of the resulting annotation language, and on the other hand how it is being applied systematically in the ongoing ISO project "Semantic annotation framework, Part 2: Dialogue acts" (ISO CD 25617-2, 2009) - "SemAF/Dialogue acts", for short.

The rest of this paper is organized as follows. Section 2 describes the components in the proposed design methodology in some more detail. Section 3 illustrates the methodology by showing the relation between a metamodel and an abstract syntax in the case of the SemAF/Dialogue acts project. Section 4 illustrates the methodology in more detail by describing the abstract syntax and an ideal representation format in the case of the SemAF/Time project, and by showing how the use of the methodology can improve the current state of the representation format proposed in the project. We end with some general conclusions in section 6.

2 Components of the proposed methodology

In this section we briefly explain the components of the 4-step methodology, as indicated above, before turning to illustrations in the design of languages for the annotation of dialogue recordings with dialogue act information, and for the annotation of documents with time- and event-related information.

2.1 Metamodel and conceptual view

The design of any annotation language should begin with a specification of the information to be captured in the annotations. Such a specification is initially informal, in the form of a conceptual analysis formulated in natural language, but should be made as precise as possible. In ISO projects it is customary to cast this analysis in the form of a 'metamodel', i.e. a listing of the categories of entities and relations to be considered, often visualized by a UML-like diagram. An example of such a conceptual analysis and its expression in a metamodel is provided in section 3.1.

2.2 Abstract syntax

The abstract syntax of a semantic annotation language 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) rules which describe the possible combinations of these elements into annotation structures. Two examples of an abstract syntax are provided in this paper. Section 3.3 contains the abstract syntax of the Dialogue Act Markup Language (DiAML) designed in the SemAF/Dialogue acts project; and section 4.1 summarizes the abstract syntax of the ISO-TimeML language under development in the SemAF/Time project.

2.3 Semantics

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 the

SemAF/Time project, a semantics is defined for the ISO-TimeML representation format by translating these representations (which are XML expressions) into a form of first-order logic.

The great advantage of attaching the semantics to the abstract rather than to a concrete syntax is that any representation format which forms a rendering of the abstract annotation structures inherits the same semantics – see the next subsection.

2.4 Concrete syntax and ideal representation formats

A representation of annotation structures can be said to be *ideal* if it gives an exact expression of the information in annotation structures. More precisely, we define a concrete syntax 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, i.e.:

- every annotation structure defined by the abstract syntax has a unique representation defined by the concrete syntax;
- every representation defined by the concrete syntax is the rendering of a unique annotation structure defined by the abstract syntax.

Note that an ideal concrete syntax F$_1$ defines a function F_1 from annotation structures to F$_i$-representations, and an inverse function F_1^{-1} from F$_1$-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 F$_i$ and F$_j$ we can define a homomorphic mapping C_{ij} from F$_i$-representations to F$_j$-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 F$_i$-representation r

and conversely, a a homomorphic mapping C_{ji} from F$_j$-representations to F$_i$-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 F$_j$-representation r

These two mappings constitute *conversions* from one format to the other, i.e. 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'. This meaning preservation is based on the fact that the meaning of a representation using an ideal format F$_k$ is by definition the meaning of the annotation structure which it represents:

(3) $\mu(r) =_D I_a(F_k^{-1}(r))$

where I_a is the interpretation function defining the semantics of the abstract syntax (see 4.2.1). Therefore, for any F$_1$-representation r:

(4)
$$\begin{array}{lll} \mu(C_{ij}(r)) & = & \mu(F_j(F_i^{-1}(r))) \\ & & I_a(F_j^{-1}(F_j(F_i^{-1}(r))) \\ & & I_a(F_i^{-1}(r)) \\ & & \mu(r) \end{array}$$

3 Illustration: dialogue act annotation

We illustrate here the formulation of a conceptual view of what a particular semantic annotation schema is about, and how this may be visualized in a metamodel, for the case of the SemAF/Dialogue acts project. We subsequently illustrate the notion of an abstract syntax for the case of DiAML, and discuss how it relates to the metamodel.

3.1 Conceptual view and metamodel for dialogue act annotation

A dialogue act[1] is conceived as a unit in the semantic description of communicative behaviour in dialogue, specifying how the behaviour is intended to change the information state of a dialogue participant who understands the behaviour correctly (i.e. as intended by the speaker). The specification of intended information state changes ('updates') requires two ingredients: (1) a specification of the information with which the information state is to be updated; (2) a specification of the way in which that information is to be used in updating the information state. These two ingredients are called the *semantic content* and the *communicative function* of the dialogue act, respectively. Formally, a dialogue act is an information-state update operator construed by applying a communicative function to a semantic content.

A dialogue act being a unit in the semantic description of communicative behaviour, the question arises what stretches of such behaviour are

[1]This subsection uses some material adapted from section 4 of ISO document N442 rev 05 (ISO CD 24617-2-2009-10-05).

considered as corresponding to dialogue acts. The identification of meaningful stretches of dialogue is called the segmentation of the dialogue. Dialogues are often segmented into turns, defined as the stretches of speech contributed by a single speaker, but turns can be quite lengthy and complicated, and are for most purposes too coarse as the stretches of behaviour to assign communicative functions to. These can be assigned more accurately to smaller units, which we call *functional segments*, and which we define as the functionally relevant minimal stretches of communicative behaviour.

According to the definition given in the first sentence of this subsection, a dialogue act has at least two participants: (1) an agent whose communicative behaviour is interpreted, usually called the “speaker”, or “sender”; and (2) a participant that he addresses and whose information state he wants to influence, called the “addressee” (also called “hearer” or ”recipient”). There may of course be more than one addressee. There may additionally be various types of side-participants who witness a dialogue without participating in it. The presence of side-participants may influence the communicative behaviour of the participants, if these are aware of their presence, as in a television interview or a talk show. Clark (1996) distinguishes between ‘overhearers’, ‘side-participants’ and ‘bystanders’, depending on the role they play in the communicative situation; we will use ‘overhearer’ as a cover term, allowing finer distinctions to be drawn when necessary.

Of the two most central aspects of a dialogue act, the communicative function and the semantic content, the former corresponds intuitively to the *type of action* that is performed, and as mentioned above, the term “dialogue act annotation” is commonly used to describe the assignment of communicative function labels to stretches of dialogue. A semantically more complete characterization of a functional segment also provides information about the *type of semantic content*. For example, the DAMSL annotation schema makes a coarse 3-way distinction of semantic content types into Task, Task Management, and Communication. These values indicate whether the semantic content of the dialogue act is concerned with performing the task that underlies the dialogue, or with discussing how to perform the task, or with the communication. The DIT^{++} annotation scheme (Bunt, 2009) makes a more fine-grained distinction of semantic content type by distinguishing communication-related information into a number of subtypes, such as information about the processing of something that was said before (feedback information), about the allocation of turns (turn management information), or about the structuring of the dialogue (topic and dialogue structure information). These types of semantic content are also called ‘*dimensions*’.

Many types of dialogue act have a responsive character, being semantically dependent on one or more dialogue acts that occurred earlier in the dialogue. This is for example the case for answers, whose meaning depends on which question is being answered; but also for the acceptance or rejection of offers, suggestions, invitations, and requests; and for accepting an assignment of the turn, or responding to a greeting. For these dialogue acts, an important aspect that may be marked up is the relation to the ‘antecedent’ on which their meaning depends. This relation is called a ‘functional dependence relation’.

Feedback-providing and eliciting acts are in a sense also responsive, as they relate to what happened earlier in the dialogue, but in a different way. Feedback acts are concerned with the processing of what was said before - such as its perception, interpretation, or evaluation. The difference is that feedback acts are about the processing of *what was said* earlier, rather than responding to the dialogue acts that were expressed. This relation is called a ‘feedback dependence relation’.

In the characterization of the notion of a dialogue act and its realization, as given so far, the following key elements occur, which form the backbone of the metamodel for dialogue act annotation shown in Figure 1.

- sender (or ‘speaker’)
- addressee(s)
- participants in other roles (‘overhearers’)
- functional segment
- dialogue act
- communicative function
- semantic content type (‘dimension’)
- functional dependence relation
- feedback dependence relation

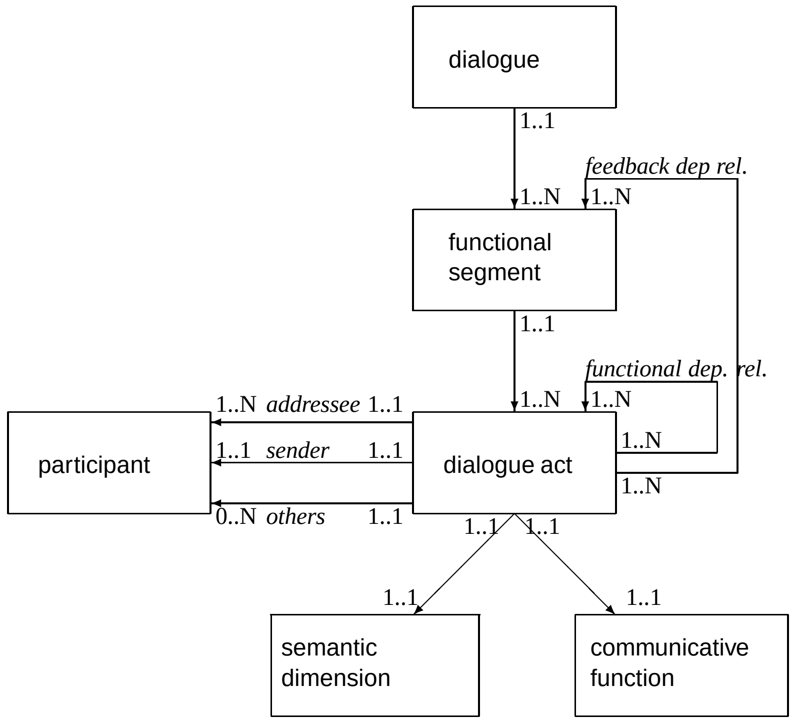

Figure 1: Metamodel for dialogue act annotation.

3.2 DiAML abstract syntax

The abstract syntax of DiAML defines certain set-theoretical structures ("DiAML annotation structures") which contain all and exactly those elements that constitute the annotation of functional segments in dialogue with communicative function information according to the metamodel of Figure 1.

Definition of DiaML abstract syntax.

1. Conceptual inventory:
   - a finite set $Parts$ = $\{P_1, P_2, ..., P_k\}$of elements called 'dialogue participants';
   - a finite set Dim = $\{D_1, D_2, ..., D_N\}$ of elements called 'dimensions';
   - a finite set of sets DSF = $\{DSF_1, DSF_2, ..., DSF_N\}$, where each element DSF_i is a finite set DSF_i = $\{F_{i1}, F_{i2}, ..., F_{in_k}\}$ of elements called 'dimension-specific communicative functions';
   - a finite set GPF = $\{F_{01}, F_{02}, ..., F_{0n}\}$ of elements called 'general-purpose communicative functions';
   - a finite set QA = $\{A_1...A_k\}$ of elements called 'qualification aspects', and a finte set QV = $\{Q_1, ..Q_k\}$ of finite sets of elements called 'qualifiers';
2. Annotation construction rules:
   - an annotation structure is a set of annotation structures (recursively), or a pair $<\sigma, \delta>$ where σ is functional segment and δ is a dialogue act structure, or a pair $<\sigma, \Delta>$ where σ is a functional segment and Δ is a set of dialogue act structures;
   - a dialogue act structure is one of the following:
     (a) a quadruple $<S, A, d, f>$ where $S \in Part$ (the sender/speaker of the dialogue act); $A \subset Parts$ (the set of addressees of the dialogue act); d is a dimension ($d \in Dim$); and f is a communicative function;
     (b) a quintuple $<S, A, d, f, \delta'>$ with S, A, d, and f as before, and where δ' is a dialogue act structure;
     (c) a quintuple $<S, A, d, f, \sigma'>$ with S, A, d, and f as before, and where σ'

is a functional segment.

- a communicative function is an element of the set of (core) communicative functions, i.e. $f \in DSF \cup GPF$; or a pair $<f, q>$ where $f \in DSF \cup GPF$ and q is a qualifier structure;
- a qualifier structure q is a list of pairs $<A_i, q_{ik}>$ with $A_i \in$ QA and $q_{ik} \in Q_i$, $Q_i \in$ QV, such that no qualification aspect occurs more than once.

It may be observed that all the ingredients in the metamodel occur in this specification, except that of a dialogue. The absence of 'dialogue' is to be expected, since a given dialogue as such does not directly turn up in an annotation, only the functional segments into which it is segmented. (In the representation of annotations, these segments contain information about how they relate to the original dialogue.) The elements that may occur in an annotation are the remaining three types of entity (participant - in the possible roles of speaker, addressee, or other; semantic dimension; communicative function); a functional dependency relation among dialogue acts; and a feedback relation between a dialogue act and a functional segment.

The abstract syntax is not just a set-theoretical formalization of the metamodel; it also refines the metamodel in two respects. First, a distinction is made in the SemAF/Dialogue acts project between communicative functions which can only be used to address a particular dimension (such as Turn Grabbing and Turn Giving, which are specific to the Turn Management dimension), and so-called 'general-purpose function' which can be used to address any dimension (such as Inform, Request, Answer..). This distinction is reflected in the abstract syntax in the split of the set of communicative functions, corresponding to an entity category in the metamodel, into two sets (DSF and GPF, respectively, in the conceptual inventory). Second, in order to capture the occurrence of emotional, partial, modal and conditional variants of dialogue acts, the SemAF/Dialogue project introduces 'qualifiers' that may be attached to communicative functions, rather than treating these variants as separate communicative functions. This is reflected in the abstract syntax by the occurrence of qualifiers in the conceptual inventory, and the possibility of attaching lists of qualifiers to communicative functions in the annotation construction rules.

3.3 DiAML ideal concrete syntax

The DiAML concrete syntax specifies an XML-based ideal representation format for the abstract syntax (see ISO CD 24617-2, 2009). Here we just give a (slightly simplified) example.[2]

In this example we see a stretch of dialogue, consisting of a question by participant P1 followed by a turn unit contributed by participant P2, in response P1's question. P2's utterance is segmented into two overlapping functional segments: one in the Auto-Feedback dimensions, with positive value, and one in the Task dimension, with value 'answer' qualified as 'uncertain'.

	P1:	*Do you know what time the next train to Utrecht leaves?*
	P2:	*The next train to Utrecht leaves at 8:32.*
(5)	AuFB	*The next train to Utrecht* [positiveAutoFeedback]
	TA	*The next train to Utrecht leaves I think at 8:32.* [answer, uncertain]

Dialogue act annotations may be attached to primary dialogue data in a variety of ways. They may be attached directly to stretches of speech, defined by temporal begin- and end points, but often they will be attached to structures at lower levels of analysis and annotation, such as the output of a tokenizer. Here we will assume that the relevant functional segments are identified at another level of XML representation, for instance in the way in which information is attached to digital documents according to TEI-ISO standard ISO 24610-1 (see TEI, 2009). Following ISO practice, we will use the term 'markable' to refer to the entities that annotations are attached to. For the example, we assume that P1's utterance is identified as the functional segment 'fs1', and the two functional segments in P2's turn as 'fs2' (in the Auto-Feedback dimension) and 'fs3' (in the Task dimension). The `target` attribute establishes the links with the primary text.

We further assume that the dialogue fragment considered here forms part of a digital document in which the metadata contain the relevant information that identifies the participants ('p1' and 'p2').

With these assumptions, the DiAML representation of the dialogue act annotation of (5) is as follows:

[2]Incidentally, ISO document N442 rev 05 specifies two alternative ideal representation formats, which are easily seen to be convertible from one to the other.

```
<diaml xmlns:"http://www.iso.org/diaml/">
   <dialogueAct xml:id="da1" target="#fs1"
      sender="#p1" addressee="#p2"
      communicativeFunction="setQuestion"
      dimension="task"
      conditionality="conditional"/>
   <dialogueAct xml:id="da2" target="#fs2"
      sender="#p2" addressee="#p1"
      communicativeFunction="autoPositive"
      dimension="autoFeedback"
      feedbackDependenceTo="fs1"/>
   <dialogueAct xml:id="da3" target="#fs2"
      sender="#p2" addressee="#p1"
      communicativeFunction="answer"
      dimension="task"
      functionalDependenceTo="da1"/>
</diaml>
```

The example shows that the representation is a straightforward, transparent rendering of its information content using the concepts of the abstract syntax.

4 Illustration: annotation of time and events

In this section we illustrate the use of the proposed methodology by applying it to the design of the ISO-TimeML language in the project SemAF/Time. In particular, we describe the abstract and concrete syntax and their relation, showing that the concrete syntax as specified so far defines a representation format that is not ideal, which causes some problems. We also show how these problems may be resolved by specifying an ideal representation format.

A prototypical example of an annotation representation in the ISO-TimeML format[3] is the following, for the sentence *John left on 31 December 2007*:

(6)
```
<isoTimeML xmlns:
"http://www.iso.org/isoTimeML
xml:id="a1">
<EVENT xml:id="e1" target=
"#token2" pred="LEAVE" type=
"TRANSITION" class= "OCCURRENCE"
tense="PAST" aspect="NONE"
pos="VERB" vform="NONE" mood="NONE"
polarity="POS"/>
<SIGNAL xml:id="s2"
target="#token3"/>
<TIMEX3 xml:id="t1"
target="#token4 #token5 #token6
type="DATE" value="2007-12-31"/>
<TLINK eventID="#e1"
relatedToTime="#t1" signalID="#s1"
relType="IS_INCLUDED"/>
</isoTimeML>
```

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

4.1 ISO-TimeML abstract syntax

The abstract syntax of ISO-TimeML defines the set-theoretical structures which constitute the information about time and events that may be contained in annotations.

a. Conceptual inventory
The concepts which can be used to build ISO-TimeML annotations fall into five categories, all formed by finite sets of temporal and event-related entities and relations, plus the concepts of real and natural numbers. The categories of temporal and event-related entities and relations are the following:

- finite sets of elements called 'event types'; 'tenses', 'aspects', 'veracities', and 'signatures';
- 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 weeks'; 'calendar day numbers'; (with 31 elements); 'week days'; and 'clock times';
- a finite set of elements called 'temporal units'.

b. Annotation construction rules
Annotation structures in ISO-TimeML consist of *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.

An entity structure is a pair <s, a> consisting of a stretch of source text s and an annotation a. A link structure is a triple $<e_1, e_2, r>$ consisting of two entity structures and a relational element.

There are four types of entity structures $<s, a>$, depending on the type of a component, and seven types of link structures. We describe here all four types of entity structures and three of the link structure types.

Entity structures:

1. An *event structure* is an 6-tuple $<e, t, a, \sigma, k, v>$ where e is a member

of the set of event types; t and a are a tense and an aspect, respectively; σ is a set-theoretical type, such as *individual object* or *set of individual objects*; k is a natural number or a numerical predicate (like *more than five*; and v is a veracity (including claimed truth or falsity);

2. An *instant structure* ('point in time') is either a triple $<time\ zone, date, clocktime>$, where a date is a time interval of one day, defined by a calendar year, a calendar month, and a calendar day number, or a triple $<time\text{-}amount\ structure, instant\ structure, temporal\ relation>$ (*"half an hour before midnight"*).
3. A *time interval structure* is a triple consisting of a calendar year, a calendar month, and a calendar day number;
4. The following set-theoretical structures are *interval structures*:
   (a) a calendar year; or a pair consisting of a calendar year and a calendar month (*May 2010*); or a triple: calendar year, calendar month, and calendar day number;
   (b) a pair $<t_1, t_2>$ of two instant structures, corresponding to the beginning and end points of the interval;
   (c) a triple $<time\text{-}amount\ structure, interval\ structure, temporal\ relation>$ (*"a week before Christmas"*);
   (d) a triple $<t_1, t_2, R>$ where t_1 and t_2 are either instant structures or interval structures, and where R is a duration relation (*"from nine to five"*).
5. A *time-amount structure* is a pair $<n, u>$ or a triple $<R, n, u>$, where n is a real number, R a numerical relation, and u a temporal unit.

Link structures:

1. A temporal anchoring structure is a triple $<event\ structure, interval\ structure, temporal\ anchoring\ relation>$, or a triple $<event\ structure, instant\ structure, temporal\ anchoring\ relation>$;
2. An event-duration structure is a triple $<event\ structure, time\text{-}amount\ structure, duration\ relation>$.
3. An aspectual structure is a triple $<event\ structure, event\ structure, aspectual\ relation>$.

4.2 ISO-TimeML semantics

4.2.1 Overview

The ISO-TimeML semantics as published in (ISO DIS 24617-1, 2009) specifies the meanings of the XML-expressions of its representation format, which is a variant of TimeML (Pustejovsky et al., 2005), through a translation into first-order logic.

By contrast, we provide here a semantics which specifies the meanings of the annotation structures defined by the abstract syntax, through a mapping from these structures into the language of discourse representation structures (DRSs). These structures form the semantic representation language of Discourse Representation Theory (DRT; Kamp & Reyle, 1993), and are known to be logically equivalent to first-order logic expressions, but to have advantages for compositional and incremental construction of its representations from natural language expressions.

Before presenting the details of this semantics, we illustrate the way it works with a simple example. Consider the sentence:

(7) *John started to read at midnight*

Using self-explanatory names for elements of the conceptual inventory, the annotation structure for this sentence would be as follows. The two events that the text refers to (a *start* event and a *read* event) give rise to two entity structures $e1$ and $e2$:

(8) $e1$: <m1, <start, past, individual, positive>>
$e2$: <m2, <read, individual, positive>>

where $m1$ is a markable that refers to the segment *started*, and $m2$ a markable referring to the segment *to read*. The annotation structure for the time of the start event is:

(9) $e3 :< m3, < \mathsf{GMT}, <>, 24:00 >>$

where the markable $m3$ refers to the source text segment *"at midnight"*, and the clock time is assumed to be taken relative to Greenwich Mean Time.

The two events are linked through an aspectual relation, and the start event is temporally anchored at the time that is mentioned. This is annotated by means of the link structures $L1$ and $L2$:

(10) $L1 :< e1, e2,$ `initiates`>
$L2 :< e1, e3,$ `at`>

The annotation structure as a whole is the pair $A_1 = <E_1, L_1>$ where $E_1 = \{e1, e2, e3\}$ and $L_1 = \{L1, L2\}$.

The semantics of this annotation structure can be computed by mapping the components of the annotation structure into small DRSs (see (11)) and merging these into one comprehensive DRS (12).

The DRSs interpreting the components of the annotation structure are the following (leaving out the default values `'individual'` and `'positive'` – see below for their treatment):

$$(11)\quad e1 \rightsquigarrow \begin{array}{|l|}\hline \text{e1}\\ \hline \text{START(e1)}\\ \text{PAST(e1)}\\ \hline \end{array}$$

$$e2 \rightsquigarrow \begin{array}{|l|}\hline \text{e2}\\ \hline \text{READ(e2)}\\ \hline \end{array}$$

$$e3 \rightsquigarrow \begin{array}{|l|}\hline \text{t1}\\ \hline \text{CLOCKTIME(GMT, t1) = 24:00}\\ \hline \end{array}$$

$$\text{L1} \rightsquigarrow \begin{array}{|l|}\hline \text{e3 e4 x1, t2}\\ \hline \text{THEME(e3, e4)}\\ \text{EVENT-TIME(e3) = t2}\\ \text{BEGIN(EVENT-TIME(e4)) = t2}\\ \text{AGENT(e3, x1)}\\ \text{AGENT(e4, x1)}\\ \hline \end{array}$$

$$\text{L2} \rightsquigarrow \begin{array}{|l|}\hline \text{e5 t3}\\ \hline \text{EVENT-TIME(e5) = t3}\\ \hline \end{array}$$

The information in the link structure $L1$, which describes how the two events are related to each other, is expressed in the corresponding DRS by means of a semantic relation ('THEME') between the two events, plus conditions expressing that the started event begins at the time of the start event, and that both events are performed by the same agent.

Merging the five DRSs in (11) results in the following DRS:

$$(12)\quad \begin{array}{|l|}\hline \text{e1 e2 x1 t1}\\ \hline \text{START(e1)}\\ \text{PAST(e1)}\\ \text{READ(e2)}\\ \text{THEME(e1, e2)}\\ \text{CLOCKTIME(GMT,t1) = 24:00}\\ \text{EVENT-TIME(e1) = t1}\\ \text{BEGIN(EVENT-TIME(e2)) = t1}\\ \text{AGENT(e1, x1)}\\ \text{AGENT(e2, x1)}\\ \hline \end{array}$$

This example might suggest that annotation structures can be interpreted simply by translating the component entity and link structures into DRSs and merging these. This is not true in general, however, as the following example shows.

(13) *John started to read at midnight. He enjoyed it.*

The annotation structure for this text is identical to that of (7), except for the additional entity structure (14a) for the *enjoy* event (with markable $m3$ referring to the segment *enjoyed*), with DRS-interpretation (14b):

(14) a. $e4$:<m3,<enjoy,past,individual,positive>>

$$\text{b.}\quad e4{:}\ \begin{array}{|l|}\hline \text{e5}\\ \hline \text{ENJOY(e5)}\\ \text{PAST(E5)}\\ \hline \end{array}$$

The DRSs (11) and (14b) can be merged in various ways, one being the possibility where the discourse referent e4 in the DRS for L1 is unified with the referent e5 in (14b), which leads to the following DRS for the entire annotation structure:

$$(15)\quad \begin{array}{|l|}\hline \text{e1 e2 e3 x1 t1}\\ \hline \text{START(e1)}\\ \text{PAST(e1)}\\ \text{READ(e2)}\\ \text{ENJOY(e3)}\\ \text{THEME(e1, e3)}\\ \text{CLOCKTIME(GMT, t1) = 24:00}\\ \text{EVENT-TIME(e1) = t1}\\ \text{BEGIN(EVENT-TIME(e3)) = t1}\\ \text{AGENT(e1, x1)}\\ \text{AGENT(e3, x1)}\\ \hline \end{array}$$

This is not a correct interpretation of the annotation structure, since it indicates an aspectual relation between the *start* event and the *enjoy event* rather than between the *start* event and the *read* event, and it is interesting to examine why this interpretation is technically speaking possible. The reason is that, different from the annotation structures, the DRSs corresponding to the annotation structure components do not contain information about the segments of source text that they apply to. In particular, the $L1$ link structure contains the entity structures $e1$ and $e2$, which in turn refer to particular markables, but the DRS for $L1$ just says that there are two events, one initiating the other, without referring to the *start*, *read* and *enjoy* events as mentioned in the source text, and as such the DRS does not really capture the linking information which the link structure L_1 contained. The same goes for the temporal anchoring structure L_2, so in fact the DRSs in (11) and (14b) could be merged in even weirder ways than (15).

The phenomenon that markable-related information gets lost when translating annotation structure components into logical representations caused Bunt (2007) to propose a way of keeping track of the identifiers of annotation structure components when interpreting annotation structures in a compositional way through their translation into first-order logic. Other attempts to provide a formal semantics for (ISO-) TimeML (Katz, 2007; Lee, 2008; Pratt-Hartman, 2007) have also encountered this problem. Lee (2008) adopted the solution proposed by Bunt (2007).

Here we propose a simpler solution, which consists of including the information about the relation to markables within the DRSs. For example, for the DRS-interpretation of the link structure L_1 this leads to (16), where the 'ANCHOR' function is used to specify the markable that the annotation structure applies to.

(16)
$$\begin{array}{|l|}
\hline
\text{e3 e4 x1, t2} \\
\hline
\text{ANCHOR(e3) = m2} \\
\text{ANCHOR(e4) = m4} \\
\text{AGENT(e3, x1)} \\
\text{AGENT(e4, x1)} \\
\text{THEME(e3, e4)} \\
\text{EVENT-TIME(e3) = t2} \\
\text{BEGIN(EVENT-TIME(e4)) =t2} \\
\hline
\end{array}$$

By including the markable information not only in link structure interpretations but also in the interpretations of entity structures, the merging of DRSs enforces the correct unification of the discourse referents and the generation of the intended interpretation.

4.2.2 Interpretation function

The following four clauses define an interpretation function I_a as a mapping from abstract annotation structures to DRSs. Clause 1 defines the interpretation of the elements of the conceptual inventory. Clauses 2 and 3 recursively define the interpretation of entity structures and link structures, respectively. Clause 4 simply says that the interpretation of an annotation structure $<E, L>$ is the merge of the interpretations of its components. The annotation structure may be 'singly connected', in the sense that the link structures in L relate all the entity structures in E to some particular member of E (so the total structure may be viewed as a connected graph). If this is the case, then the application of the interpretation to the link structures results in a single DRS. If the annotation structure does not have this property, as is the case for a text fragment with unrelated events, such as: *John called Mary at around 3 o'clock. Peter did not show up* then the interpretation of the link structures results in a set of semantically independent DRSs; in view of the conjunctive character of the components of an annotation structure, it is appropriate to merge these DRSs into a single DRS.

1. *Elements from the conceptual inventory*
The interpretation function I_a assigns an individual, predicate, or function name to the following elements of the conceptual inventory, where, in the interest of readability, we will indicate the interpretation $I_a(\alpha_i)$ of an element α_i of the conceptual inventory as α_i':

- event types (e_i), tenses (t_j), aspects (a_m), and cardinalities (k_n);
- temporal relations, time measurement functions, temporal units, duration relations, numerical relations, event subordination relations, and aspectual relations;
- time zones Z_i (interpreted as functions which map the time line onto pairs consisting of a date and a clock time);
- calendar years, calendar months, calendar day numbers;
- real numbers, where $I_a(r)$ will be its usual string name (like '95.743').

Veracities and set-theoretic types are not represented as such in DRS conditions, but are interpreted through the different interpretations of event annotations, depending on the values of these elements in the annotation. For instance, an entity structure with negative veracity will be interpreted as the negation of the representation which represents the same event structure with positive polarity.

2. *Entity structures*
In the clauses for entity structure interpretation we will use $m(e)$ to abbreviate "$\text{ANCHOR}(e) = m$".

2.1 Event annotations:
$I_a(< m, < e, t, a, indiv, pos >) =$
$= <\{e\}, \{m(e), <e'(e), t'(e), a'(e)\}>>$,

or
$$\begin{array}{|l|}
\hline
e \\
\hline
m(e) \\
e'(e) \\
t'(e) \\
a'(e) \\
\hline
\end{array}$$

$I_a(<< m, e, t, a, indiv, neg >>)$ =
= $\neg < \{e\}, \{m(e), <e'(e), t'(e), a'(e))\} >$,

or $\neg \begin{array}{|l|}\hline e \\ \hline m(e) \\ e'(e) \\ t'(e) \\ a'(e) \\ \hline \end{array}$

$I_a(< e, t, a, set, k, pos >)$ =
= $<< E >, \{m(E), k'(E)\}, << e_1 >, < e_1 \in E > \rightarrow$
$\{<<>, e'(e_1), t'(e_1), a'(e_1)\} >\} >$,

or $\begin{array}{|l|}\hline E \\ \hline m(e) \\ \text{k'(E)} \\ \begin{array}{|l|}\hline e_1 \\ \hline e_1 \in E \\ \hline \end{array} \Rightarrow \begin{array}{|l|}\hline \\ \hline e'(e_1) \\ t'(e) \\ a'(e) \\ \hline \end{array} \\ \hline \end{array}$

$I_a(< e, t, a, set, k, neg >)$ =
= $\neg << E >, \{m(E), << e_1 >, < e_1 \in E > \rightarrow$
$<<>, \{< e'(e_1), , t'(e_1), a'(e_1)\} >\} >$

2.2 Interval annotations:
$I_a(< t_1, t_2 >)$ =
= $<< t, t_1, t_2 >$,{begin$(t) = t_1$, end$(t) = t_2\} >$

$I_a(<< n, u >, T_1, R >$ =
= $<< t >, \{R'(t, T_1') \wedge$ distance$((t, T_1), u') = n'\} >$

$I_a(< z, y)$ = $<<, t >, \{\text{YEAR}(t) \wedge$ calyear$(t, z') = y'\} >$

$I_a(< z, y, m >)$ =
= $<<, t >, \{\text{MONTH}(t) \wedge$ calyear$(t, z') = y'$
$\wedge$ calmonth$(t, z') = m'\} >$

$I_a(< z, y, m, d >)$ =
= $<<, t >, \{\text{DAY}(t) \wedge$ calyear$(t, z') = y'$
$\wedge$ calmonth$(t, z') = m' \wedge$ caldaynum$(t, z') = d'\} >$

2.3 Instant annotations:
$I_a(< z, y, m, d, \tau >)$ =
= $<< t >$, {calyear$(t, z') = y' \wedge$ calmonth$(t, z') = m'$
$\wedge$ caldaynum$(t, z') = d' \wedge$ clocktime$(t, z') = \tau'\} >$

$I_a(<< n, u >, t_1, R >$ =
= $<< t >, \{R'(t, t_1) \wedge$ distance$((t, t_1), u') = n'\} >$

2.4 Time-amount annotations:
$I_a(< n, u >)$ = $<< x >$, {length$(x, u') = n'\} >$

$I_a(< R, n, u >)$ = $<<x>, \{R'($ length$(x, u'), n')\}>$

3. *Link structures:*

3.1 Temporal anchoring structures:

$I_a(< \epsilon, \tau, R >)$ = $I_a(\epsilon) \oplus I_a(\tau) \oplus$
$<< x, y >, \{R'($event-time$(x), y)\} >$

3.2 Event duration structures:

$I_a(< \epsilon, < nu >>)$ =
= $I_a(\epsilon) \oplus << e >$, {length(event-time$(e), u') = n'\} >$

3.3 Aspectual structures:
$I_a(< \epsilon1, \epsilon2, A >)$ = $I_a(\epsilon1) \oplus I_a(\epsilon1) \oplus$
$<< e_1.e_2 >, \{a'(e_1, e_2)\} >)$

4. Annotation structures ;
$I_a(<E, L>$ = $\oplus(\{e' | e' = I_a(e)$ for some $e \in E\} \cup$
$\{r' | r' = I_a(r)$ for some $r \in L\})$

Applied to example (13), we obtain the following two DRSs representing the meaning of the annotations in the link structures L1 and L2:

(17) L1: $I_a(< \epsilon1, \epsilon2,$ `initiates` $>)$ =
= $I_a(\epsilon1) \oplus I_a(\epsilon2) \oplus <<e_1, e_2>$,
{initiate$(e_1, e_2)\}>)$ =
= $<<e_1>$, {start(e_1), past(e_1) }> $\oplus$
$<<e_2>$, {read$(e_2)\}> \oplus$
$<<e_1, e_2>$, {initiate$(e_1, e_2)\}>$ =
= $<<e_1, e_2>$, {start(e_1), past(e_1), read(e_2),
initiate$(e_1, e_2)\}>$

(18) L2: $I_a(< \epsilon1, \tau1,$ `at`$>)$ =
= $I_a(\epsilon1) \oplus I_a(\tau1) \oplus$
$<<x, y >, \{R'($event-time$(x), y)\}>$ =
= $<<e_e>$, {start(e), past$(e_1)\}>$ *oplus*
$<<t_1>$, {clocktime(t_1) = 24:00}> $\oplus$
$<<x, y>, \{R'($event-time$(x), y)\}>$ =
= $<<e_1, t_1>$, {start(e_1), past(e_1),
clocktime(t_1) = 24:00, event-time$(e_1) = t_1)\}>$

These DRSs are logically equivalent to the first-order logic formulas in (19):

(19) a. $I_a(L1) = \exists e_1.\text{START}(e_1) \wedge \text{PAST}(e_1) \wedge \exists e_2.$ $\text{READ}(e_2) \wedge \text{THEME}(e_1, e_2) \wedge \text{INITIATE}(e_1, e_2)$

b. $I_a(L2) = \exists e.\ \text{START}(e) \wedge \text{PAST}(e) \wedge \exists t.$ clocktime$(t) = 24:0 \wedge \text{EVENT-TIME}(e) = t$

4.3 Concrete Syntax

The concrete syntax of ISO-TimeML (see ISO, 2009) specifies a format for representing annotation structures in XML, as illustrated in example (7). This format is a slight adaptation of that of TimeML, and has not been designed as an ideal representation format for the abstract syntax.

5 Representation problems

We consider two kinds of problems in the current definition of ISO-TimeML[4]: (1) those relating to inaccurate representation of annotation structures; (2) those relating to the difficulty of separating information related to time and events from general semantic information concerning e.g. quantification and modality, for which no general approach to their annotation is currently available.

[4]See footnote 3.

5.1 Recurring events

Reference to a recurring event, as in *John called twice* is represented in ISO-TimeML as follows (slightly simplified):

(20)
```
<EVENT id="e1" tense="PAST"/>
<TIMEX3 id="t1" freq="2X"/>
<TLINK eventID="#e1"
relatedToTime="#t1"
relType="DURING"/>
```

There are several problems with this representation. First, the `EVENT` part refers to an event `e1`, temporally linked to a temporal 'entity' "twice", while the source text refers to two events. Second, what kind of entity is "twice"? The ISO-TimeML concrete syntax uses the `TIMEX3` tag for all temporal entities, and distinguishes these entities by means of the `type` attribute into *periods*, *dates*, *times*, *measures*, and *sets*. The alleged entity "twice" fits none of these types. In fact, "twice" should not be considered as a temporal entity at all; it is rather a number, counting how many instances of a certain type of event occurred. A related problem is the use of the "DURING" relation between the event and the entity "twice". The relation between events and the number of their occurrence is not *temporal* in nature, and "DURING" is therefore not an appropriate kind of relation to apply.

The abstract syntax takes a different conceptual view. It does not include entities like 'twice'; instead, the interpretation of *twice* as a number is captured by the k (for 'cardinality') component in an event structure. Second, where the ISO-TimeML format has no way of encoding a reference in a source text to a *set of events* (as opposed to a single event), the abstract syntax has the 'signature' element in an event structure for indicating whether an individual event or a set of events is considered.

We see here that certain inadequacies in the representations occur due to an imperfect match between distinctions made in the abstract syntax and those expressed in the concrete representation format. We therefore propose the use of a representation format that does not suffer from such imperfections, an ideal representation format. In the next subsection we outline such a format for the abstract syntax given above.

5.2 An ideal representation format

We define a concrete XML-based syntax for the annotation structures defined by the ISO-TimeML abstract syntax, and call this representation format the ICS-1 format.

Recall that an annotation structure is a pair $<E, L>$ consisting of a set E of entity structures, and a set L of link structures that link (some of) the entity structures together. The building blocks of the various types of entity structures are the elements of temporal and event-related categories specified by the conceptual inventory.

5.2.1 Representation of conceptual inventory items

Event types:

- attribute `type`; values: `READ, TEACH, CALL, SLEEP, ENJOY,...` (any event type distinguished in a given inventory or ontology of events);
- tenses: attribute `tense`; values: `PRESENT, PAST, FUTURE`, `IMPERFECT, NONE`;
- aspects: attribute `aspect`; values: `PROGRESSIVE, PERFECTIVE, IMPERFECTIVE`,..;
- aspectual relations: attribute: `aspectRel`; values: `INITIATE TERMINATE`, `CONTINUE`, `CULMINATE, REINITIATE`;
- duration relations: attribute: `durationRel`; values: `THROUGH, WHILE, ..`;
- event subordination relations: attribute: `eventSubordRel`; values: `THEME`;

Temporal entities:

- time zones: attribute `timeZone`; values `CET, GMT, EST,...`;
- calendar years: attribute `calYear`; values `2010, 2009,...`;
- calendar months: attribute `calMonth`; values `JANUARY, FEBRUARY, MARCH,..., DECEMBER`;
- calendar day numbers: attribute `calDayNum`; values 1, 2, 3,..., 31
- clock times: attribute `clockTime`; values 00:00, 00:01, 00:02, 00:59, 01:00, 01:01, ..., 23:59;
- temporal relations: attribute: `tempRel`; values: `AFTER, BEFORE, DURING,...`;

- time measurement functions: attribute `length`; values: see temporal amounts;
- units of time: attribute: `unit`; values: `SECOND, MINUTE`, `HOUR`, `DAY, WEEK`, `MONTH`, `YEAR`, `DECADE`, `CENTURY`,..

Numerical relations:

- attribute: `numRel`; values: `"LESS_THAN"`, `"LESS_OR_EQUAL_THAN"`

Real numbers:

- attribute `numeral`; values: all real numbers, as represented by their usual string name (like '5.14').

5.2.2 Representation of annotation structures

The collection of entity structures and link structures which together form an annotation structure is represented in ICS-1 format as a list (in arbitrary order) of the representations of the entity structures and link structures.

a. Entity structures

An entity structure is a pair $<m, a>$ where m is a markable and a is an annotation which is either an event structure, a time interval structure, a time instant structure, or a time amount structure. For each type of entity structure we introduce an XML element, so this gives the element types `EVENT, PERIOD, INSTANT, DATE,` and `TIME_AMOUNT`. These annotations are all n-tuples of elements from the classes of the conceptual inventory. For example, an event structure is a 6-tuple consisting of an event type, a tense, an aspect, a signature, a cardinality, and a veracity. The ICS-1 format reflects this by defining 6 attributes for `EVENT` elements, whose values represent the elements of the 6-tuples. More generally, for each type of entity structure we define attributes for the corresponding XML element such that the components of each type of entity structures have a one-one correspondence with attribute values in the ICS-representation. For each XML element corresponding to an entity structure we add (1) a unique identifier, as the value of the special attribute `xml:id`, and (2) an attribute `anchor` which has a markable m as its value, indicating how the representation is anchored in the source text.

For example, the event structure in the sentence (21a), tokenized and with the markable 'm1' defined as in (21b), is represented as in (21c):

(21) a. *Mary laughed.*
b. m1= token2: 'laughed'
c.
```
<isoTmeML-ICS1rep xml:id="a1">
<EVENT xml:id="e1" anchor="#m1"
type="LAUGH"
tense="PAST" aspect="NONE"
veracity="POSITIVE"
signature="INDIVIDUAL"
cardinality="NONE" />
</isoTmeML-ICS1rep>
```

Compared to the current ISO-TimeML representation, the ICS-1 representation is simplified in certain respects, and has a different treatment of event quantification and of representing the length of temporal intervals. The main simplifications are the following:

- ISO-TimeML uses a double classification of kinds of events; on the one hand a 7-way classification inherited from TimeML, expressed by the values of the attribute `class`, and on the other hand a 3-way classification into processes, transitions, and states expressed by the three possible values of the attribute `type`. The value of the attribute `pred` is used to indicate the event-type of which a token is considered in the annotation of a certain event; this attribute is renamed `type` here. The use of an attribute with such values, is only useful if there is information available about the entities denoted by these values. For instance, the use of a `LAUGH` event-type, as in example (21), typically presupposes the information that laughing is an event of type `'PROCESS'`, and in the TimeML classification is an `'OCCURRENCE'`. It therefore seems redundant to specify these classifications in annotations. We therefore left out these attributes. This gives us the opportunity to rename the attribute `pred` to `type`, which seems a more appropriate name.

- (ISO-)TimeML uses an attribute `pos` to annotate the part of speech of the expression at which the annotation is anchored; an attribute `vform` to indicate in the case of a verbal expression whether it has infinitive, gerundive, or participle form; and an attribute `mood` to indicate whether a verbal expression has subjunctive mood. Since this is syntactic rather than semantic information, we leave it out.

The ideal concrete syntax uses the attributes `signature`, and `cardinality`, which do not correspond to attributes for event annotations in (ISO-)TimeML, for the representation of repeated events and number of repetitions, and of quantified relations between events or between events and time intervals or instants. This is discussed below.

b. Link structures

For each type of link structure defined by the abstract syntax we introduce an XML element, just as we did for the types of entity structure. Link structures consist of two entity structures and a relation, so for each type of link structure we introduce three attributes, two having pointer values for referring to the entity structure representations, and one whose value is the corresponding relation.

For the anchoring of an event in time, for isntance, we introduce the `TIME_ANCHORING` element with attributes for the anchored event and the anchor time, respectively, and an attribute `relType` specifying in what way the event is anchored (e.g. by the relation`"AT"` or by the relation `"DURING"`), as illustrated in example (22). In the abstract syntax there are no link structures for establishing relations between link structures (but only relations between entity structures), therefore link structure representations do not need a unique XML identifier, in contrast to entity structure representations.

The ICS-1 expression (22c) represents the annotation of example sentence (7), repeated here as (22a), tokenized and with markables defined in terms of tokens as in (22b). Attributes which have the value `"NONE"` or a (different) default value (see below) are not shown here.

(22) a. *John started to read at midnight*.

b. m1= token2: ‘started’, m2 = token3 token4: ’to read’, m3 = token5: ’at’, m = token6: ’midnight’

c.
```
<isoTimeML_ICS1rep xml:id="a1">
<EVENT xml:id="e1" anchor=’’#m1’’
type ="START" tense=PAST
signature="INDIVIDUAL">
<EVENT xml:id="e2" anchor="#m2"
type ="READ" tense="NONE"
signature="INDIVIDUAL">
<INSTANT xml:id="t1" anchor="#m4"
clockTime=’24:00’>
<TIME_ANCHORING anchor="#m3"
anchoredEvent="#e1"
anchorTime="#t1" relType=’AT’>
</isoTimeML_ICS1rep>
```

Using the ICS-1 format, the sentence *John called twice* is represented as follows:

(23)
```
<id="e1" type="CALL" tense="PAST"
signature="SET" cardinality="2"/>
```

This representation says simply that two *call*-events occurred in the past.

The ISO-TimeML representation (20) has yet another unsatisfactory feature, namely the use of the attribute `freq` with value `"2X"`. If the attribute `freq` is to capture the number of times a certain type of event occurs, then the value should be a numerical expression, and the use of a string containing the code ”X” (for ”times”) makes no sense. Second, the number of times an event is repeated is not a *frequency*; a frequency is the number of times something occurs within a period of a certain length, like *twice an hour*.

A sentence with a genuine frequency description, such as *John calls home twice a day*, in fact describes a quantified relation between a set of recurring events and the set of periods in which they occur. In ISO-TimeML, such a sentence is represented as shown in (24), leaving out attributes and values of no particular interest here.

(24)
```
<EVENT id="e1" pred="CALL"
tense="NONE"/>
<TIMEX3 id="t1" freq="2X"/>
<TIMEX3 id="t2" type="SET"
value="P1D" quant="EVERY">
<=TLINK eventID="#e1"
relatedToTime="#t2" />
relType="DURING"/>
```

The criticism of the representation (20) of *John called twice* also applies in this case; moreover, the temporal quantification is not represented correctly. Problematic is that the `EVENT` element does not correspond to a set of events; there is no way in ISO-TimeML to represent a set of events, and that the `TIMEX3` element refers to a set of entities which are periods of a length of 1 day, according to the value of the `value` attribute. (In `"P1D"`, ‘P’ stands for ‘period’ and ‘D’ for day; this is the ISO-TimeML way of describing a one-day period.) This is not correct, for suppose John has the habit of calling at 10:00 a.m. and at 7:00 p.m., then in the one-day period from 8:00 p.m. to 8:00 a.m. John doesn’t call at all.

Viewed as a quantification, the sentence describes a quantified relation between a set of events and a set of days (in the sense of periods starting at 0:00 and ending 24 hours later at midnight). This view underlies the ICS-1 representation of this sentence, shown in (25).

(25)
```
<EVENT id="e1" type="CALL"
tense="NONE"
signature="SET"/>
<PERIOD id="t1"/> type="DAY"
signature="SET"/>
<TIME_ANCHORING anchoredEvent="#e1"
anchorTime="#t1"
relType="INCLUDED_IN"/>
eventDistr="INDIVIDUAL"
timeDistr="INDIVIDUAL"
eventQuant="2" timeQuant=EVERY"/>
```

This representation can be read as saying that a set of *call* events is anchored timewise in a set of days, such that the individual events are anchored at individual days, where every day includes a time anchor for two of these events. This is exactly what we want.

6 Discussion and Conclusions

In this paper we have presented a new methodology for the design of languages for semantic annotation, bringing three innovations. First, we have added a third component to the usual two components of a language definition: a syntax that specifies the set of expressions of the language, and a semantics that specifies for each expression what it means. The third component, an *abstract syntax*, specifies the categories of information that the annotations expressed in the language may contain, and the ways in which elements of these categories may be combined into annotation structures. This specification is in set-theoretical terms, independent of any representation format. What is traditionally called a syntax, by contrast, is in terms of a particular representation format. This distinction is especially important in the context of defining annotation standards, since according to the Linguistic Annotation Framework, standards should be defined at an abstract level, independent of any representation format.

Second, we have shown the possibility to define the semantics of an annotation language as a specification of the meanings of the annotation structures defined by the abstract syntax, rather than as a description of the meanings of representations defined by a concrete syntax. This has the advantage that any representation format which defines a rendering of the structures defined by the abstract syntax inherits the semantics of the abstract syntax. For the case of ISO-TimeML we have shown that the meanings of the annotation structures can be derived in a straightforward manner by keeping track in the DRSs of the information in the annotations concerning their anchoring to the source text.

Third, we have introduced the notion of an *ideal representation format* where each annotation structure defined by the abstract syntax has a unique rendering in that format, and each representation is the rendering of a uniquely determined annotation structure. It is easy to see that any two ideal representation formats can be converted into each other through a strictly meaning-preserving mapping.

Is an ideal representation format 'ideal' in the sense of note being realistically achievable? We have shown that an ideal XML-representation format can be defined for a given abstract syntax by systematically introducing XML elements for entity types and relational types. In fact, being ideal should be a requirement of *any* satisfactory representation format, since a format which is not ideal is either unable to represent all the semantic distinctions that are made by the abstract syntax, or else it introduces irrelevant parts, which have no semantic basis.

Besides being theoretically important, ideal representation formats have the advantages of being maximally simple, only representing the conceptual distinctions made in the abstract syntax, and being optimally transparent, allowing a simple semantic interpretation.

We have illustrated this for the design of the DiAML language for annotating dialogue recordings with dialogue act information, and for the design of the ISO-TimeML language for annotating documents with information about time and events. In the latter case we have noted a number of representation problems in the ISO-TimeML in its current state, and pointed out how the language may be improved. This shows that the approach can be of practical value for improving the design of annotation languages.

The introduction of an abstract syntax as a information-theoretic layer in the definition of a semantic annotation language supports a principled view on the information that annotations are intended to capture, more than is encouraged by the traditional approach where a representation format is defined for which a semantics is defined (if a semantics is defined at all). Such a principled view gives us a handle on general issues in semantic annotation such as quantification and modification. This opens up interesting possibilities for developing satisfactory ways to annotate natural language expressions which display such omnipresent phenomena as quantification and modification .

Acknowledgments

I would like to thank James Pustejovsky and Kiyong Lee for many in-depth discussions of issues in the temporal and event-related semantics of natural language.

References

Bunt, H. (2007). The Semantics of Semantic Annotation. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation (PACLIC21)*, pages 13–29, Korean Society for Language and Information.

Bunt, H. (forthc.). Introducing Abstract Syntax and Semantics in Languages for Semantic Annotation.

Bunt, H. and Romary, L. (2002). Towards Multimodal Content Representation. In Choi, K. S., editor, *Proceedings of LREC 2002, Workshop on International Standards of Terminology and Language Resources Management*, pages 54–60, Las Palmas. Paris: ELRA.

Burnard, L. and Bauman, S. (2007). *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. TEI Consortium.

Clark, H. (1996). *Using language*. Cornell University Press, Cambridge, UK.

Ide, N. and Romary, L. (2004). International Standard for a Linguistic Annotation Framework. *Natural Language Engineering*, 10:211–225.

ISO (2009a). *Language Resource Management - Linguistic Annotation Framework (LAF)*. ISO, Geneva. ISO document ISO/DIS 24612:2009.

ISO (2009b). *Language Resource Management - Semantic Annotation Framework (SemAF) - Part 1: Time and Events*. Secretariat KATS. ISO International Standard 24617-1:2009(E)), 11 October 2009.

ISO (2009c). *Language Resource Management - Semantic Annotation Framework (SemAF) - Part 2: Dialogue Acts*. ISO, Geneva. ISO document ISO/CD 24617-2-2009-10-05.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht.

Katz, G. (2007). Towards a Denotatial Semantics for TimeML. In Schilder, F., Katz, G., and Pustejovsky, J., editors, *Annotation, Extraction, and Reasoning about Time and Events*. Springer, Dordrecht.

Lee, K. (2008). Against a Davidsonian Analysis of Copula Sentences. In Kadowaki, M. and Kawahara, S., editors, *NELS 33 Proceedings*.

Pratt-Hartmann, I. (2007). From TimeML to Interval Temporal Logic. In *Proceedings of the Seventh International Workshop on Computational Semantics (IWCS-7)*, pages 166–180, Tilburg, Netherlands.

Pustejovsky, J., Castano, J., Ingria, R., Gaizauskas, R., Katz, G., Saurí, R., and Setzer, A. (2003). TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 337–353, Tilburg, Netherlands.

Pustejovsky, J., Ingria, R., Saurí, R., Gastano, J., Littman, J., Gaizauskas, R., Setzer, A., Katz, G., and Habel, C. (2005). The Specification Language TimeML. In Mani, I., Pustejovsky, J., and Gaizauskas, R., editors, *The Language of Time*. Oxford University Press, Oxford.

Pustejovsky, J., Knippen, R., Littman, J., and Saurí, R. (2007). Temporal and Event Information in Natural Language Text. In Bunt, H. and Muskens, R., editors, *Computing Meaning*, volume 3, pages 301–346. Springer, Berlin.

TEI (2009). *TEI P5: Guidelines for Electronic Text Encoding and Interchange. Lou Burnard and Syd Bauman, editors*. Text Encoding Initiative. Also ISO standard 14610.