Abstract syntax and semantics in semantic annotation, applied to time and events

Harry Bunt

Tilburg Center for Cognition and Communication
Department of Communication and Information Sciences
Tilburg University, the Netherlands
harry.bunt@uvt.nl

Abstract. This paper describes an approach to the definition of semantic annotations using the notion of an abstract syntax, which characterizes the content of well-formed annotations in set-theoretical terms, independent of any particular representation format. Most importantly, a formal semantics is defined for the structures generated by the abstract syntax; this semantics is inherited by any representation format. This approach is applied in a redefinition of the ISO-TimeML language for annotating time and event-related information, developed by the International Organization for Standardization ISO. The semantics of the proposed abstract syntax of ISO-TimeML has the form of a compositional translation into discourse representation structures. The separation between an abstract syntax with a semantics, and a concrete syntax which defines a particular representation format, is shown to allow the design of semantically well-defined annotations which are optimally simple and transparent.

1 Introduction

The definition of standards for the annotation of language resources has in recent years become a focal area of interest in the International Organization for Standardization ISO. Researchers and developers in computational linguistics and language technology have realized that, while the availability of annotated corpora is more important for the development of the field and its applications than ever, the current state of affairs is far from ideal in that different researchers use different formats and concepts in their annotations, making the exploitation and effective sharing of annotated material from different sources very difficult. The ISO organization has therefore assembled expert groups in various areas of language technology and computational linguistics in order to develop international standards, e.g. for morphosyntactic annotation, for syntactic annotation, for the representation of lexical content, and for semantic annotation. Moreover, standards have been established for the definition of descriptive concepts ('data categories', as defined in ISO standard 12620); for the use of feature structures for expressing linguistic and other information (ISO standard 24610), and for linguistic annotation efforts in general in the Linguistic Annotation Framework (Ide & Romary, 2004; ISO 24612:2011).

In the area of semantic annotation, the project Semantic Annotation Framework (ISO 24617) has been launched, which consists of several parts dealing with standards for annotating different kinds of semantic information. Part 1: Time and events, deals with the annotation of information related to time and events, and has defined a standard (ISO 24617-1) which is based on TimeML (Pustejovsky et al., 2005; 2007). Part 2: Dialogue acts deals with the annotation of recorded dialogues with dialogue act information, and has defined standard ISO 24617-2. Other parts deal with the annotation of semantic roles, discourse relations, named entities, and spatial information.

For semantic annotation, standards have to comply with two methodological requirements which do not necessarily apply to other areas of annotation standardization. First, since semantic annotations are meant to capture some of the meaning of the annotated text, if the annotations don't have a well-defined meaning, then how can they capture any meaning in a better way than the text itself? Bunt & Romary (2002) have therefore formulated the requirement of semantic adequacy, which says that a language for semantic annotation is required to have a well-defined semantics.

A second requirement comes from the Linguistic Annotation Framework,which distinguishes 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 the Linguistic Annotation Framework, annotations are the proper level of standardization, rather than representations.

In order to deal with the first requirement, the annotation languages defined within the Semantic Annotation Framework have a formal semantics as part of their definition. This is not as simple and obvious as it may sound; for instance, the representation language for annotating text with time-and event-related information defined within this framework, called 'ISO-TimeML' because it descends from TimeML, is a particular form of XML, and as such does not come with a semantics.

To deal with the second requirement, we have proposed (Bunt, 2010) to introduce an additional component in the definition of an annotation language, besides the specification of (a) a representation format and (b) a semantics for that, which specifies the categories of information that can be used to build semantic annotations, and their possible combinations as annotation structures. This additional component is called an *abstract syntax*; the specification of a representation of annotation structures is called a concrete syntax. So a complete specification of a semantic annotation language consists of three components: an abstract syntax, a concrete syntax, and a semantics. This proposal has been adopted in the design of ISO-TimeML, the specification of which has these three components (see ISO 24617-1:2011).

Parallel to the submission of ISO-TimeML to the international community as a proposed annotation standard, we have continued to further develop the

three-component approach. First, we have designed a semantics for the abstract, rather than the concrete syntax. This has the great advantage that any concrete syntax which specifies a way of representing the annotation structures defined by the abstract syntax, inherits the same semantics. This provides a basis for proving that alternative representation formats are semantically equivalent, and hence convertible from one to another. Second, this development has brought to light that the concrete syntax defined in (ISO, 2009) is far from ideal, not being able to express all the distinctions that are made in the abstract syntax, and we developed the notion of an *ideal concrete syntax*.

In this paper we present this further development, and illustrate it for the annotation and representation of information relating to time and events. In Section 2 we characterize the information about time and events that ISO-TimeML aims to capture in annotations. In Section 3 we (re-)define the abstract syntax and formal semantics of ISO-TimeML. In Section 4 we introduce the notion of an 'ideal' representation format, and define an ideal XML representation for the structures generated by the abstract syntax. Section 5 describes the application of the ideal XML representations to ISO-TimeML annotations, and shows their interpretation defined by semantics of the abstract syntax. We show how this leads to semantically well-defined and well-motivated representations which are optimally simple and transparent. We end with concluding remarks in Section 6.

2 Information about Time and Events

2.1 Coverage

When designing an annotation language, the first question is what information the language should be able to capture. Using the terminology of the Linguistic Annotation Framework, mentioned above, this decision is crucial for defining the set of possible *annotations*, independent of any representational formats.

ISO-TimeML is meant to allow annotators (human or machine) to assign semantic mark-ups to the following kinds of expressions:

- expressions denoting an event, state, or a process, such as *"he woke up"; "Mary was asleep", "I will make breakfast"* but also *"the inauguration"; "the political situation"; "the melting of the ice caps"*;
- expressions relating events (and states, and processes) to aspectually related or subordinate events (and states, and processes), as in *"she started to laugh"; "she stopped crying"; "she wanted to eat something"*;
- temporal expressions, describing dates, times, particular periods, and so on, like *"February 29 2009"; "ten o'clock tomorrow morning"; "Christmas 1999"; "the twenty-first century"; "yesterday"*;
- expressions that indicate relations between temporal entities, such as *at, before, during, for* as in: *at midnight; two days before Christmas; during the weekend; for two years*;

– expressions denoting an extent of time, such as *"ten seconds"; "five minutes"; "two and a half years"*;
– expressions describing the frequency of a recurring event, such as *"four times last week"; "twice a day"*;
– expressions relating events (and states, and processes) to temporal objects, as in *"John woke up at six fifteen"; "this Tuesday's inauguration"; "he slept from nine to five"; "he slept eight hours"; "he called three times today"*.

2.2 Basic temporal and event-related concepts

In the literature a distinction is often made between 'states' and 'events', where the latter are characterized as occurring at a point in time or during a certain definite interval, while the former may obtain for any indefinite stretch of time (*"The Mediterranean Sea separates Europe from Africa"*). For linguistic and philosophical reasons, several classifications have been proposed of verbs describing various types of states or events, the Vendler classification being the best known (Vendler, 1967). For the annotation of temporal information in text, not only verbs with their tenses and temporal modifications should be considered, but also nouns, since nouns may also denote events and situations (*"the meeting at twelve"; "the six o'clock news"*). Terminological note: the term 'event' will henceforth be used as a generic term that also covers such notions as 'state', 'situation', 'action', 'process', etc.; this broad notion of event has also been termed 'eventuality' (Bach, 1986).

Temporal objects and relations have been studied from logical and ontological points of view; well-known studies include those by Allen (1984), Prior (1967), and more recently Hobbs and Pan (2004); see also the collection of papers in Mani et al. (2005). The most common view of time, that underlies most natural languages, is that time is an unbounded linear space running from a metaphorical 'beginning of time' at minus infinity to an equally metaphorical 'end of time' at plus infinity. This linear space can be represented as a straight line, the 'time line', of which the points correspond to moments in time. Following Hobbs & Pan (2004) we will also use the term 'instant' to refer to points of time.

In reality nothing happens in infinitesimally small time; every event or state that occurs in reality (or in someone's mind, for that matter) requires more than zero time, although natural languages offer speakers the possibility to express themselves as if something occurs at a precise instant (like *"I will call you at twelve o'clock sharp"*). Since instants are formally a special kind of interval, a consistent approach to modelling the time that an event occurs is to always use intervals, where it may happen that the interval associated with a particular event is regarded as having zero length, and thus being an instant.

Temporal intervals have a length which can also occur in natural language in relation to an event, as in *"I used twelve hours to read that book"*; *"It takes six and a half minutes to walk to the station"*; or *"The time difference between Amsterdam and Denver is eight hours"*. An expression like *"six and a half minutes"* denotes the length of a time interval or the accumulated length of a series of intervals. It is the temporal equivalent of spatial distance (6.5 kilometres).

To describe the length of a temporal interval one needs a unit of measurement, which may be combined with a numerical expression to obtain an amount of time.

Regarding the temporal anchoring of events in time, it may be noted that the association of a temporal interval with an event does not necessarily mean that the event took place during every moment within that interval. When someone says *"I've been working on this article from 8.30 to 12 o'clock"*, that presumably does not mean that the speaker has been working on his article every single moment between 8.30 and 12 o'clock; there must have been interruptions for having some coffee, going to the bathroom, and so on. Whether an event occurs during an interval with or without any interruptions can only be decided on a case by case basis, and is perhaps best modeled as a property of the temporal anchoring relation applied to a specific event token.

2.3 Anchoring annotations to primary text

The Linguistic Annotation Framework, mentioned above, insists on the use of *stand-off annotation*, i.e. the construction of annotations in separate files, separate from the document containing the primary language data, as contrasted with in-line (or 'embedded') annotation. Stand-off annotations refer to specific locations in the primary data by addressing byte offsets, linguistic elements such as words, or times associated with recorded data. Compared to in-line annotation, stand-off annotation has the advantages of respecting the integrity of the primary data and of allowing multiple annotations to be layered over a given primary document. Since semantic annotations typically occur at a relatively high level in a layered annotation structure, they do not necessarily refer directly to segments in the primary data, but rather to structures in other annotation layers. The generic term *markable* is used to refer to the entities that the annotations are associated with. Markables are for instance elements in a syntactic or in a morphosyntactic annotation layer; a requirement on markables is that they identify certain tokens in the source text to which they apply. In this paper we will assume that markables are identified by a `target` attribute whose values refer to source text tokens.

Markables are derived from documents, which will have certain metadata that are important for the interpretation of temporal annotations. For interpreting the tenses of verb forms, or adverbial temporal deixis (*"last night"; "next week"*), for instance, which occur in a text, one must know when the text was produced. This will often be defined by the document creation time, and more precisely by the combination of a creation time and a creation location, since the latter defines the time zone within which the creation time is precisely defined. In many documents the time and place of the document creation will apply to all the markables that may be derived from the document, but it may also happen that the text in a document introduces other times and places relative to which the annotations of certain markables should be understood. A time zone, like Greenwich Meat Time (GMT) can be seen as a way of mapping the time line to dates and clock times.

3 An abstract syntax and semantics for ISO-TimeML

3.1 Annotation and representation

The distinction between annotations and representations, made in the Linguistic Annotation Framework, is reflected in the specification of ISO-TimeML given below in the distinction between an abstract syntax and a concrete syntax. The abstract syntax specifies the types of elements and their possible combinations in the construction of annotations, defined as set-theoretical structures, independent of any particular representation format.

Annotations, as defined by the abstract syntax, can be represented in many ways. In line with other ISO standards for language resources, the ISO-TimeML proposal includes a concrete syntax defining a reference representation in XML.

In the rest of this section we present the abstract syntax of a revised specification of ISO-TimeML (Section 3.2), and the semantics of the annotation structures defined by the abstract syntax (Section 4.3).

3.2 Abstract syntax

The abstract syntax consists of two parts: (a) a specification of the elements from which the annotations are built up, called a *conceptual inventory*, and (b) a description of the possible combinations of these elements into annotation structures. What these combinations mean is specified by the semantics associated with the abstract syntax.

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 number and natural number. Natural numbers are needed for capturing the information expressed in English by *"twice"* and *"three times"*; real numbers are needed for dealing with expressions denoting amounts of time, such as *"two and a half hours"*.

The categories of temporal and event-related entities and relations are the following:

- finite sets of elements called 'event predicates'; 'tenses'; 'aspects'; 'polarities'; and 'set-theoretic types';
- finite sets of elements called 'unary temporal predicates', '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' (with 12 elements), 'calendar weeks' (52 elements),'calendar day numbers' (31 elements); 'day names' (7 elements); and 'clock times' (24×60 elements);
- a finite set of elements called 'temporal units'.

b. Annotation structures

Annotation structures in ISO-TimeML consist of two kinds of elements, which we will refer to as *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 by means of links between entity structures.

An entity structure is a pair $<m, a>$ consisting of a markable m and an annotation a. A link structure is a triple $<e_1, e_2, r>$ consisting of two entity structures and a relational element. Annotation structures consist of a set of entity structures and a set of link structures which link the entity structures together; in ISO-TimeML these links correspond to temporal and inter-event relations.

Entity structures:

Entity structures $<m, a>$ come in five varieties, depending on the kind of annotation that forms the second component. This component may contain information about (1) events; (2) dates; (3) time intervals; (4) time points (or 'instants', using the terminology of OWL-Time (Hobbs & Pan, 2004)); and (5) amounts of time).

(1)
1. An *event structure* is a n-tuple, with $1 \leq n \leq 6$; for $n = 6$ this is a tuple $< e, t, a, \sigma, k, v >$ where e is a member of the set of event predicates; 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 (e.g. the number 2 for dealing with such examples as *"John kissed Mary twice"*); and V is a polarity.
2. A *date structure* is any of the following structures:
   (a) a triple <*calendar year, calendar month, calendar day number*>;
   (b) a pair <*calendar year, calendar month*>;
   (c) a pair <*calendar month, calendar day number*>;
   (d) a pair <*calendar week, calendar day number or day name*>;
   (e) a single *calendar year, calendar month*, or *calendar week* (for things like *"week 35"*);
3. An *instant structure* is either (a) a triple <*time zone, date structure, clock time*>, or (b) a triple <*time-amount structure, instant structure, temporal relation*> (*"half an hour before midnight"*);
4. The following set-theoretical structures are *interval structures*:
   (a) a unary temporal predicate that names a particular period in history, like *"(the) Golden Age", "(the) Renaissance"*, or that characterizes a certain type of period, like *"decade", "century"* (and thus identifies a class of intervals rather than a single interval);
   (b) a triple consisting of a unary temporal predicate that names an annually recurring event, like *"Easter", "Carnaval", "Ramadan", "New Year's Eve"*, a year and a time zone;
   (c) a pair $<t_1, t_2>$ of two instant structures, corresponding to the beginning and end points of the interval;

(d) a triple <*time-amount structure, interval structure, temporal relation*> (*"three weeks before Christmas"; "two years from today"*);

5. 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 (*"less then two milliseconds"*).

Link structures:
There are seven types of link structures in ISO-TimeML: (1) for anchoring events in time; (2) for temporally relating one event to another; (3) for relating intervals and instants to each other; (4) for measuring the duration of an event; (5) for measuring the length of a temporal interval (not necessarily related to an event); (6) for subordination relations between events; and (7) for aspectual relations between events.

(2)
1. A temporal anchoring structure is a triple consisting of an event structure, a temporal anchoring relation, and an interval structure, instant structure, or date structure;
2. An event-temporal relation structure is a triple <*event structure, event structure, temporal relation*>;
3. A temporal relation structure is a triple <*interval* or *instant* or *date* structure, *interval* or *instant* or *date structure, temporal relation*>
4. An event-duration structure is a pair <*event structure, time-amount structure*>;
5. An interval length measurement structure is a pair <*interval structure, time-amount structure*>;
6. A subordination structure is a triple <*event structure, event structure, subordination relation*>;
7. An aspectual structure is a triple <*event structure, event structure, aspectual relation*>.

Annotation structures:
An ISO-TimeML annotation structure is a set of one or more entity structures and zero or more link structures.

3.3 Semantics

3.3.1 Outline
We define a semantics for ISO-TimeML which specifies the meaning of annotation structures through a mapping into the language of discourse representation structures (DRSs). This language is the semantic representation language of Discourse Representation Theory (DRT; Kamp & Reyle, 1993), and is known to be logically equivalent to first-order logic, but to have advantages for compositional and incremental construction of its representations from natural language expressions.

Before presenting the details of this semantics, we outline how it works for a simple example. Consider the sentence:

(3) John started to read at midnight.

Using self-explanatory names for elements from the conceptual inventory, the annotation structure for this sentence would be as follows. Annotation of the two events that the text refers to, a start event and a read event, gives rise to two entity structures $\epsilon 1$ and $\epsilon 2$ of the type of event structures, which according to (1.1) are n-tuples indicating an event type and, optionally, a tense, an aspect, a signature, a cardinality, and a polarity. In this case both events are considered as single, individual events with positive polarity. The read event is tenseless. The annotation structures are thus as follows, where $m1$ is a markable that refers to the segment *started*, and $m2$ a markable referring to the segment *to read*:

(4) $\epsilon 1$: <$m1$, <start, past, individual, 1, positive>>
$\epsilon 2$: <$m2$, <read, notense, individual, 1, positive>>

The annotation structure for the time of the start event is an entity structure of the type 'instant', which according to (1.2a) is a triple consisting of a time zone, a date, and a clock time. In this case let us assume that from the context the time zone is known to be Greenwich Mean Time and that the precise date is not known but is known to fall in January 2011. This leads to the entity structure (5), where lack of knowledge about the day of the month is indicated by omitting the corresponding element. An important point to note here is that annotations may in general be *incomplete*. According to definition (1) every entity structure is an n-tuple for some particular value of n (between 2 and 6); in all cases k-tuples with $0 \leq k \leq n$ are incomplete but well-formed entity structures.

(5) $\epsilon 3$: <$m3$, <GMT, <2011, january>, 24:00>>

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 ('initiates'), and the start event is temporally anchored at the time that is mentioned. This is annotated by means of the link structures $L1$ and $L2$:

(6) $L1$: <$\epsilon 1, e2$, `initiates`>
$L2$: <$\epsilon 1, e3$, `at`>

The annotation structure as a whole is the set

(7) A = {e1, e2, e3, L1, L2}

The semantics of this annotation structure can be computed by mapping its components into the small DRSs shown in (8), and merging these into one comprehensive DRS (9).[1]

[1] A DRS is formally a pair consisting of a set of variables, called 'discourse markers', and a set of propositions, called 'conditions'. It is common practice to represent DRSs in graphical form as 'boxes' of the kind shown in (8). For example, the box representing event structure $e1$ is formally the pair $\langle \{e1\}, \{start(e1), past(e1)\} \rangle$. In the rest of this paper we will occasionally use the box notation for its better readability, but we will also use the string notation in order to save space.

(8) a.

e1
start(e1) past(e1)

b.

e2
read(e2)

c.

t1
calyear(t1,GMT) = 2011 calmonth(t1,GMT) = january clocktime(t1,GMT) = 24:00

d.

e3 e4 x1, t2
theme(e3, e4) event-time(e3) = t2 begin(event-time(e4)) = t2 agent(e3, x1) agent(e4, x1)

e.

e5 t3
event-time(e5) = t3

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 these five DRSs results in the following DRS:

(9)

e1 e2 x1 t1
start(e1) past(e1) read(e2) theme(e1, e2) calyear(t1,GMT) = 2011 calmonth(t1,GMT) = january clocktime(t1) = 24:00 event-time(e1) = t2 begin(event-time(e2)) = t2 agent(e1, x1) agent(e2, x1)

This DRS does indeed express what the annotation means, and is logically equivalent to the first-order formula (10):

(10) $\exists e_1.\exists e_2.\exists x_1.\exists t_1.$ start$(e_1) \wedge$ past$(e_1) \wedge$ read$(e_2) \wedge$ theme $(e_1, e_2) \wedge$ calyear$(t_1$GMT$)$=2011 $\wedge$ calmonth$(t_1$,GMT$)$= january $\wedge$ event-time$(e_1) = t_2$ $\wedge$ begin(event-time$(e_2)) = t_2 \wedge$ agent$(e_1, x_1) \wedge$ agent(e_2, x_1)

3.3.2 Keeping track of source text anchoring

The example in the previous subsection might suggest that annotation structures can be interpreted simply by translating the component entity and link structures into DRSs and merging these. That's not true in general, however, as the following example shows.

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

The annotation structure for this text is identical to that of (3), except for the additional entity structure (12a) for the *enjoy*-event (with markable $m3$ referring to the segment *enjoyed*, with DRS-interpretation (12b).[2]

(12) a. $\epsilon 4$: <$m3$,<enjoy, past, individual, 1, positive>>

b.

e5
enjoy(e5) past(e5)

The DRSs (8) and (12b) can be merged with various possibilities for unifying the variables, one being the case where the discourse referent e4 in the DRS for $L1$ is unified with the referent e5 in (12b), which would lead to the following DRS for the entire annotation structure:

(13)

e1 e2 e3 x1 t1
start(e1) past(e1) read(e2) enjoy(e3) theme(e1, e3) clockttime(t1) = 24:00 event-time(e1) = t1 begin(event-time(e3)) = t1 agent(e1, x1) agent(e3, x1)

[2] The values 'individual', '1', and positive' are all default values, which do not require explicit representation in a DRS.

This DRS is an incorrect interpretation of the annotation structure consisting of (8) and (12b), since it indicates an aspectual relation between the *start* event and the *enjoy event*, rather than between the *start* event and the *read* event. This interpretation is technically possible because the DRSs (8) and (12b) do not contain information about the segments of source text that they apply to; in that respect they have less information than the annotation structures. In particular, the $L1$ link structure contains the entity structures $e1$ and $e2$, which refer to particular source text tokens, but the DRS for $L1$ just says that there are two events, one initiating the other, without referring to the specific *start, read* and *enjoy* events mentioned in the source text, and as such this DRS does not really capture the linking information which the annotation link structure contains. The same goes for the temporal anchoring link structure. (So in fact the DRSs in (8) and (12b) could be merged in even weirder ways than (13).)

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

In this paper we propose a simple solution, which consists of including the information about the relation to markables into the DRSs which interpret them. To this end we introduce a function 'anchor' which specifies the markable that the annotation structure applies to. For example, for the interpretation of the entity structures for the start and read events this leads to the following DRSs:

(14) a.

e1
anchor(e1) = m1
start(e1)
past(e1)

b.

e2
anchor(e2) = m2
read(e2)

For the link structure $L1$ this leads to the DRS (15):

(15)

e3 e4 x1, t2
anchor(e3) = m1 anchor(e4) = m2 agent(e3, x1) agent(e4, x1) theme(e3, e4) event-time(e3) = t2 begin(event-time(e4)) = t2

By including the markable information not only in link structure interpretations but also in the interpretations of entity structures, the merging of the DRSs enforces the correct unification of the discourse referents and the generation of the intended interpretation. In the next subsection we will see exactly how this is accomplished in a systematic way.

3.3.3. Interpretation function

An interpretation function I_a is defined as a mapping from abstract annotation structures to discourse representation structures. Clause (16) recursively defines the interpretation of the components of annotation structures: elements of the conceptual inventory, entity structures, and link structures. After that we will turn to the interpretation of complete annotation structures, consisting of multiple entity and link structures.

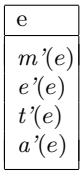
(16) 1. **Elements from the conceptual inventory:**
The interpretation function I_a assigns an individual, a predicate, a function, or a number to the following elements of the conceptual inventory, where, in the interest of readability, we will indicate the interpretation $I_a(\alpha_i)$ of an element α_i of the conceptual inventory as α_i':
- event predicates e_i, tenses t_j, and aspects a_m;
- temporal names (predicates like 'Golden Age', 'Independence Day'), temporal relations, duration relations, numerical relations, event subordination relations, and aspectual relations;
- temporal measurement functions; calendar functions (i.e. functions which for a given time zone assign to an instant its calendar month, calendar year, calendar week (number), calendar day number, or day name; and a clock time function, which for a given time zone assigns to an instant its time on the clock;
- time zones z_i;
- temporal units (milliseconds, hours,...);
- real numbers and integers, where $I_a(n)$ will be its usual string name (like '95.743').

Of the elements of the conceptual inventory, polarities and signatures are not translated into terms that occur in DRS interpretations, but into different DRS structures. For instance, an entity structure with negative polarity will be interpreted as the negation of the representation which represents the same event structure with positive polarity.

In what follows, we will abbreviate "anchor(e) = m" as $m'(e)$, and if τ is an n-tuple then we will use τ^1 to indicate the first element of that tuple. (In particular, if $\tau = <m, a>$, i.e. τ is an entity structure, then $\tau^1 = m$, so this is a way to grab the markable. This is used in the interpretation of link structures.)

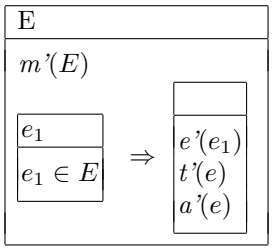
2. **Entity structures**
2.1 Event structures:

$I_a(<m,<e,t,a,$*individual, pos*$>>) = <\{e\}, \{m'(e), e'(e), t'(e), a'(e))\}>>$,
or in box notation:



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

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



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

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

2.2 Date structures:

$$
\begin{aligned}
I_a(<s,<y,m,n>>) &= <\{t,z\},\{s(t),\ \text{calyear}(t,z)=y',\\
&\qquad \text{calmonth}(t,z)=m',\ \text{caldaynum}(t,z)=n'\}>\\
I_a(<s,<y,m>>) &= <\{t,z\},\{s'(t),\ \text{calyear}(t,z)=y',\\
&\qquad \text{calmonth}(t,z)=m'\}>\\
I_a(<s,<m,n>>) &= <\{t,z\},\{s'(t),\ \text{calmonth}(t,z)=m',\\
&\qquad \text{caldaynum}(t,z)=n'\}>\\
I_a(<s,<w,n>>) &= <\{t,z\},\{s'(t),\ \text{calweek}(t,z)=w',\\
&\qquad \text{caldaynum}(t,z)=n'\}>\\
I_a(<s,<w,d>>) &= <\{t,z\},\{s'(t),\ \text{calweek}(t,z)=w',\\
&\qquad \text{dayname}(t,z)=d'\}>\\
I_a(<s,<y>>) &= <\{t,z\},\{s'(t),\ \text{calyear}(t,z)=y'\}>\\
I_a(<s,<m>>) &= <\{t,z\},\{s'(t),\ \text{calmonth}(t,z)=m'\}>\\
I_a(<s,<w>>) &= <\{t,z\},\{s'(t),\ \text{calweek}(t,z)=w'\}>\\
I_a(<s,<n>>) &= <\{t,z\},\{s'(t),\ \text{caldaynum}(t,z)=n'\}>\\
I_a(<s,<d>>) &= <\{t,z\},\{s'(t),\ \text{dayname}(t,z)=d'\}>
\end{aligned}
$$

2.3 Instant structures:

$$
\begin{aligned}
I_a(<m,\ <z,d,h>>) &= <\{t,z1\},\{m'(t),z1=z',\ \text{clocktime}(t,z')=h'\}>\\
&\qquad \oplus\ I_a(m,d)\\
I_a(<<m,a,t,R>> &= <\{t1,t2\},\ \{m'(t1),\ \text{anchor}(t2)=t^1,R'(t1,t2)\}>\\
&\qquad \oplus\ I_a(m,a)
\end{aligned}
$$

2.4 Interval structures:

$$
\begin{aligned}
I_a(<m,P>) &= <\{t\},\{m'(t),P(t)\}>\\
I_a(<P,y,z>) &= <\{t1,z1\},\{m'(t1),z1=z',\ \text{calyear}(t1,z)=y'\}>\\
I_a(<m,<t_1,t_2>>) &= <\{t,t1,t2\},\{m'(t),\ \text{anchor}(t1)=t1^1,\\
&\qquad \text{anchor}(t2)=t2^1,\ \text{begin}(t)=t1,\ \text{end}(t)=t2\}>\\
&\qquad \oplus\ I_a(<t_1^1,t1>)\oplus I_a(<t_2^1,t2>)\\
I_a(<<m,a,T,R>> &= <\{t1,t2,a\},\{m'(t1),\ \text{anchor}(t2)=T^1,\ R'(t1,t2)\}>\\
&\qquad \oplus\ I_a(<T^1,T>)\oplus I_a(m,a)
\end{aligned}
$$

2.4 Time-amount structures:

$$
\begin{aligned}
I_a(<m,\ <n,u>>) &= <\{x\},\ \{m'(x),\ \text{length}(x,u')=n'\}>\\
I_a(<\ m,\ <R,n,u>>) &= <\{x\},\ \{m'(x),\ R'(\text{length}(x,u'),n')\}>
\end{aligned}
$$

3. **Link structures**

3.1 Temporal anchoring structures:

$$
\begin{aligned}
I_a(<\epsilon,\tau,R>) = I_a(\epsilon)\oplus I_a(\tau)\oplus <\{x,y\},\ \{&\text{anchor}(x)=\epsilon^1,\\
&\text{anchor}(y)=\tau^1,R'(\text{event-time}(x),y)\}>
\end{aligned}
$$

3.2 Temporal relation structures:

$$I_a(<\epsilon 1, \epsilon 2, R>) = I_a(\epsilon 1) \oplus I_a(\epsilon 2) \oplus <\{x_1, x_2\}, \{\text{anchor}(x) = \epsilon 1^1, \text{anchor}(y) = \epsilon 2^1, R'(\text{event-time}(x), y)\}>$$

3.3 Relations between intervals and instants:

$$I_a(<\tau 1, \tau 2, R>) = I_a(\tau 1) \oplus (I_a(\tau 2) \oplus <\{x, y\}, \{\text{anchor}(x) = \tau 1^1, \text{anchor}(y) = \tau 2^1, R'(x, y)\}>$$

3.4. Event duration relations

$$I_a(<\epsilon,<n, u>>) = I_a(e) \oplus <\{e\}, \{\text{anchor}(e) = \epsilon^1, \text{length}(\text{event-time}(e), u') = n'\}>$$

3.5 Interval length measurement structures:

$$I_a(< \tau,< n, u >>) = <\{t\}, \{\text{length}(t, u') = n'\}> \oplus I_a(\tau)$$

3.6 Subordination structures:

$$I_a(< \epsilon 1, \epsilon 2, R >) = <\{e_1, e_2\}, \{\text{anchor}(x) = \epsilon^1, \text{anchor}(y) = \tau^1, R'(e_1, e_2)\}>$$

3.7 Aspectual structures:

$$I_a(< \epsilon 1, \epsilon 2, A >) = I_a(\epsilon 1) \oplus I_a(\epsilon 2) \oplus <\{x, y, z, t\}, \{\text{anchor}(x) = \epsilon 1^1, \text{anchor}(y) = \epsilon 2^1, \text{theme}(x, z), \text{theme}(y, z), A'(\text{event-time}(x), \text{event-time}(y))\}>$$

The above clauses define the interpretation of the building blocks of annotation structures. As an example of the use of clause 3.7, consider the interpretation of the link structure L1 in (6), for *"John started to read"*.

(17)
$$\begin{aligned}
&I_a(<\epsilon 1, \epsilon 2, \text{initiates}>) = \\
&= I_a(\epsilon 1) \oplus I_a(\epsilon 2) \oplus \\
&\quad <\{x, y, z, t\}, \{\text{anchor}(x) = \epsilon 1^1, \text{anchor}(y) = \epsilon 2^1, \text{theme}(x, z), \\
&\quad \text{theme}(y, z), I_a(\text{initiates})(\text{event-time}(x), \text{event-time}(y))\}> \\
&= I_a(\epsilon 1) \oplus I_a(\epsilon 2) \oplus \\
&\quad <\{x, y, z, t\}, \{\text{anchor}(x) = \epsilon 1^1, \text{anchor}(y) = \epsilon 2^1, \text{theme}(x, z), \\
&\quad \text{theme}(y, z), \text{event-time}(x) = t, \text{begin}(\text{event-time}(y)) = t\} > \\
&= < \{e_1\}, \{\text{anchor}(e_1) = \epsilon^1, \text{start}(e_1), \text{past}(e_1)\}> \oplus \\
&\quad < \{e_2\}, \{\text{anchor}(e_2) = \epsilon_2^1, \text{read}(e_2)\}> \oplus \\
&\quad <\{x, y, z, t\}, \{\text{anchor}(x) = \epsilon 1^1, \text{anchor}(y) = \epsilon 2^1, \text{theme}(x, z), \\
&\quad \text{theme}(y, z), \text{event-time}(x) = t, \text{begin}(\text{event-time}(y)) = t\} > \\
&= < \{e_1, e_2, z, t\}, \{\text{anchor}(e_1) = m1, \text{start}(e_1), \text{past}(e_1), \\
&\quad \text{anchor}(e_2) = m2, \text{read}(e_2), \text{theme}(e_1, e_2), \text{agent}(e_1, z), \text{agent}(e_2, z), \\
&\quad \text{event-time}(e_1) = t, \text{begin}(\text{event-time}(e_2)) = t\}>
\end{aligned}$$

This DRS is logically equivalent to the first-order logic formula in (18):

(18) $\exists e_1.\text{start}(e_1) \wedge \text{anchor}(e_1) = m1 \wedge \text{past}(e_1) \wedge \exists e_2.\ \text{read}(e_2) \wedge \text{anchor}(e_2) = m2 \wedge \text{theme}(e_1, e_2) \wedge \exists x.\ \text{agent}(e_1, x) \wedge \text{agent}(e_2, x) \wedge \text{event-time}(e_1) = \text{begin}(\text{event-time}(e_2)$

It would be possible to construct representations in first-order logic directly from annotations, rather than via DRSs, but this requires the use of rather complicated formula-manipulation operations.[3] We have seen that DRSs, by contrast, can be constructed in a simple fashion by exploiting the use of unification for merging the DRSs for the components of annotation structures, provided that we keep track of the anchoring of these components in the source text.

Where (16) defines the interpretation of individual entity structures and link structures, (19) defines the interpretation of an annotation structure as a whole as the merge of the interpretations of its components:

(19) $I_a(\{e_1, .., e_n, L_1, .., L_k\}) = I_a(e_1) \oplus .. \oplus I_a(e_n) \oplus I_a(L_1) \oplus .. \oplus I_a(L_k)$

Applied to the annotation structure (7) for the sentence *"John started to read at midnight"*, this gives the following result, which comes down to the merge of the interpretations of the two link structures, since all the entity structures are linked together by these link structures:

(20) $< I_a\{e1, e2, e3\}, \{L1, L2\} > =$
$= I_a(e1) \oplus I_a(e2) \oplus I_a(e3) \oplus I_a(L1) \oplus I_a(L2)$
$= I_a(L1) \oplus I_a(L2)$
$= <\{e_1, e_2, z, t\}$, $\{\text{anchor}(e_1) = m1$, $\text{start}(e_1)$, $\text{past}(e_1)$, $\text{anchor}(e_2) = m2$, $\text{read}(e_2)$, $\text{theme}(e_1, e_2)$, $\text{agent}(e_1, z)$, $\text{agent}(e_2, z)$, $\text{event-time}(e_1) = t$, $\text{begin}(\text{event-time}(e_2) = t\}> \oplus$
$<\{e_3, t_2\}$, $\{\text{anchor}(e_3) = m3$, $\text{start}(e_3)$, $\text{past}(e_3)$, $\text{midnight}(t_2)$, $\text{event-time}(e_3) = t\} >$
$= <\{e_1, e_2, z, t\}$, $\{\text{anchor}(e_1) = m1$, $\text{start}(e_1)$, $\text{past}(e_1)$, $\text{anchor}(e_2) = m2$, $\text{read}(e_2)$, $\text{theme}(e_1, e_2)$, $\text{agent}(e_1, z)$, $\text{agent}(e_2, z)$, $\text{event-time}(e_1) = t$, $\text{midnight}(t)$, $\text{begin}(\text{event-time}(e_2)) = t\}>$

or, in box notation:

[3] See Bunt (2007), where these operations are introduced in the context of combining translations into first-order logic of ISO-TimeML-representations in XML (rather than annotation structures generated by the abstract syntax).

(21)

e1, e2, z, t
anchor(e1)=m1 past(e1) start(e1) anchor(e2)=m2 read(e2) agent(e1, z) agent(e2, z) begin(event-time(e2))=t midnight(t) event-time(e1))=t

4 An Ideal Concrete Syntax

A representation of annotation structures can be said to be *ideal* if it expresses exactly the information in annotation structures. More precisely, we define a concrete syntax to be ideal for a given abstract syntax if the following conditions are satisfied:

- every annotation structure defined by the abstract syntax has a representation defined by the concrete syntax;
- every representation defined by the concrete syntax unambiguously represents an annotation structure defined by the abstract syntax.

In this section we define a concrete XML-based syntax that achieves this for the annotation structures defined by the ISO-TimeML abstract syntax; we call this syntax the ISO-TimeML *Ideal Concrete Syntax (ICS)*.[4]

4.1 Representation of conceptual inventory items

Event types:
- attribute `pred`; values: `READ, TEACH, CALL, SLEEP, ENJOY,...`;
- tenses: attribute `tense`; values: `PRESENT, PAST, FUTURE`;
- aspects: attribute `aspect`; values: `PROGRESSIVE, PERFECT, IMPERFECT`;

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, APRIL,,..., DECEMBER`;
- calendar day numbers: attribute `dayNumber`; values: `1, 2, 3,..., 31`;
- calendar day names: attribute `dayName`; values: `MONDAY, TUESDAY,...`;

[4] It is not the case that there exists just one ideal concrete syntax for the ISO-TimeML abstract syntax; for example, using typed feature structures another ideal representation format can be defined.

- clock times: attribute `clockTime`; values: 00:00, 00:01, 00:02,.., 00:59, 01:00, 01:0,.., 23:59;
- unary temporal predicates: attribute `pred`; values `CHRISTMAS, EASTER, GOLDEN_AGE, THANKSGIVING, INDEPENDENCE_DAY`,...;
- time measurement functions: attribute `tLength`; values: amounts of time;
- units of time: attribute `unit`; values: `SECOND, MILLISECOND, MINUTE, DAY, WEEK, MONTH, YEAR, DECADE, CENTURY`;
- aspectual relations: attribute `aspectRel` in aspectual links; values: `INITIATE, TERMINATE, CONTINUE`;
- duration relations: attribute: `durationRel` in duration likns; values: `THROUGH, WHILE...`;
- event subordination relations: attribute `eventSubordRel` in event subordination links; values: `THEME`;
- temporal relations: attribute `tempRel`; values: `AFTER, IMMED_AFTER, BEFORE, IMMED_BEFORE, DURING,`[5]...;

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 `5.107`).

4.2 Representation of annotation structures

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

a. Entity structures
Recall that 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 date structure, an instant structure, an interval structure, or a time amount structure. These annotations are all n-tuples of elements from the classes of the conceptual inventory. For each type of entity structure we introduce an XML element, so this gives the element types `EVENT, INSTANT, PERIOD, DATE` and `TIME_AMOUNT`. For example, an event structure is an n-tuple with $1 \leq n \leq 6$, consisting of (maximally) an event predicate, a tense, an aspect, a set-theoretical type, a cardinality, and a polarity. The Ideal Concrete Syntax reflects this by defining 6 attributes for `EVENT` elements, whose values represent the content of event structures.

More generally, for each type of entity structure we define a set of attributes for the corresponding XML element, such that its components have a one-to-one correspondence with attribute values in the ICS-representation. Moreover, for each of the XML elements corresponding to an entity structure we add (1) a unique identifier, as the value of the special attribute `xml:id`, and (2) an

[5] For the convenience of annotation, temporal relations are assumed to be polymorphic in the sense of being applicable to instants, time intervals, dates, and events.

attribute `target` whose value indicates how the representation is anchored in the source text. For example, the event structure in the sentence (22a), tokenized as in (22b), is represented as in (22c):

(22) a. *Mary laughed.*
b. token1='Mary', token2='laughed'
c.
```
<isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="LAUGH"
tense="PAST" aspect="NONE" polarity="POSITIVE"
signature="INDIVIDUAL" cardinality="NONE"/>
</isoTimeML_ICSrep>
```

Compared to the ISO-TimeML representation, defined in (ISO, 2009), the ICS representations differ in some respects, of which the following are the most important:

- ISO-TimeML uses a double classification 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`. Moreover, 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. The use of an attribute with such values only makes sense if there is information available about the entities denoted by these values. For instance, using a `LAUGH` event-type presupposes the availability in an ontology or other knowledge source of information such as laughing being an event of type `PROCESS` (rather than `STATE`, for instance) and in the TimeML classification being an `OCCURRENCE`. It therefore seems redundant to specify in annotations not only that an event is a `LAUGH` event (value of the `pred` attribute), but also that it is a `PROCESS` (value of the `type` attribute) and an `OCCURRENCE` (value of the `class` attribute). We therefore leave out both the `type` and `class` attributes and their values (as well as the corresponding elements in the structures defined by the abstract syntax).
- Following TimeML, 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, it is left out here (and from the abstract syntax as well).
- The attributes `signature` and `cardinality` in ICS-representations do not correspond to attributes in ISO-TimeML. We have introduced them for the representation of repetitions of events and quantified relations involving events and time; see below, Section 5.4.
- Following the original TimeML representation, ISO-TimeML uses the `TIMEX3` tag for all temporal entities, and distinguishes among them by means of the

type attribute into dates, times, periods, durations, and sets of these. In ICS representations, by contrast, we have the tags INSTANT, PERIOD, DATE, and TIME_AMOUNT for each of the different kinds of time-related concepts, and values of the signature attribute for differentiating between individual entities and sets of entities.

b. Link structures

Analogous to the way entity structures are represented, we introduce for each of the seven types of link structure defined by the abstract syntax an XML element type. 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 component entity structure representations, and one whose value is the corresponding relation. For link structures which anchor events in time this is the TIME_ANCHORING tag, with the attributes anchoredEvent and anchorTime, plus the attribute tempRel specifying in more detail in what way the event is anchored (e.g. by the relation AT or by the relation INCLUDED_IN). Link structures are usually not 'consumptive', i.e. the information which they contain is not expressed by a stretch of source text.

In the abstract syntax there are no link structures for relations between link structures, but only between entity structures, therefore link structure representations do not need an identifier, unlike entity structure representations. The ICS expression (23c) represents the annotation of example sentence (3), repeated here as (23a), tokenized as in (23b). Attributes which have the value NONE or another default value are not shown here for the sake of readability.

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

b. <token1='John', token2='started', token3='to', token4='read', token5='at', token6='midnight'>

c.
```
<isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="START"
tense="PAST"/>
<EVENT xml:id="e2" target="#token4" pred="READ"/>
<ASPECTUAL_LINK event="#e1" relatedToEvent="#e2"
aspectRel="INITIATE""/>
<INSTANT xml:id="t1" target="#token6" clockTime="24:00"/>
<TIME_ANCHORING anchoredEvent="#e1" anchorTime="#t1"
tempRel="AT"/>
</isoTimeML_ICSrep>
```

5 The representation of annotations in ideal format

In this section we consider the use of the ICS representation format and show its advantages over the use of formats which are not ideal. We will often simplify the representations by suppressing the anchoring to primary text in DRS interpretations.

We will first discuss the ICS-representation of annotations encoding dates and times, and subsequently consider the representation of annotations of periods and of amounts of time. We then turn to the representation of linking structures connecting annotations of events with those of time-related entities. Finally, we will discuss the annotation of phenomena relating to recurring events and to quantification in natural language involving time and events.

5.1 Representation of dates and times

ISO-TimeML follows ISO standard 8601 in using the format `yyyy-mm-dd` to encode a date, consisting of a year, a month and a day. This leads to representations like (24) for *"June 7, 2012"*:

(24) a. *June 7, 2012*
b. token1='June' token2='7' token3='2012'
c.
```
<isoTimeML xml:id="a1"/>
<TIMEX3 xml:id="t1" target="#range(token1,token3)"
type="DATE" value="2012-06-07/">
</isoTimeML>
```

The annotation of an incompletely specified date, like *"August"* in the sentence *"I will take some days off in August"*, is represented using variables in those positions where no information is available:

(25) a. *August*
b. token1='August'
c.
```
<isoTimeML xml:id="a1"/>
<TIMEX3 xml:id="t1" target="#token1" type="DATE"
value="xxxx-08-dd/">
</isoTimeML>
```

While we obviously encourage in general to follow existing ISO standards, in this case that does not seem to be a good idea. The ISO 8601 representation of dates in the form `"yyyy-xx-zz"` is meant to put an end to such confusions as whether "09-05-2010" refers to the 9th of May 2010 or to the 5th of September 2010. This makes sense for standardizing the specification of dates in a text (and in the metadata of a document), but this has little to do with the question of how to annotate natural language expressions referring to dates. Representations like (24) and (25) are unsatisfactory from a semantic point of view. Conceptually, the specification of a date has three parts: a calendar year, a calendar month, and a day within the month (and each of these should be considered for a given time zone), and each of these parts should be treated as semantic concepts rather than as substrings in a alphanumerical code. The representations (24) and (25) are not only conceptually but also technically unsatisfactory, as they do not make the components of the temporal information available for compositional interpretation and reasoning.

The ideal concrete syntax defined above, by contrast, follows the view incorporated in the abstract syntax that a date is a complex concept, and thus represents the examples (24) and (25) as follows.

(26) a. *June 7, 2012*

b.
```
<isoTimeML_ICSrep xml:id="a1">
<DATE xml:id="t1" target="#range(token1,token3)"
calYear="2012" calMonth="JUNE" dayNumber="7"/>
</isoTimeML_ICSrep>
```

(27) a. *August*

b.
```
<isoTimeML_ICSrep xml:id="a1">
<DATE xml:id="t1" target="#token1" calMonth="AUGUST"/>
</isoTimeML_ICSrep>
```

These representations are evidently more transparent as well as better suitable for semantic processing. They are straightforward renderings of the corresponding abstract annotation structures, from which they inherit their semantics. For example, the semantics of (26) is expressed by the following DRS, where here as in most examples in the rest of this paper we omit for brevity the anchoring to source text tokens, and where the variable z ranges over time zones (which are implicit in the examples here, but see example (29)):

(28)

t1 z
date(t1)
calyear(t1,z) = 2012
calmonth(t1,z) = June
caldaynum(t1,z) = 7

Times, as shown on a clock, are represented in ISO-TimeML in the same way as dates, using an alphanumerical code, for example:

(29) a. *4 p.m.*

b.
```
<isoTimeML xml:id="a1"/>
<TIMEX3 xml:id="t1" target="#token1 #token2"
type="TIME" value="T16:00/">
</isoTimeML>
```

Again, we find the use of a string encoding, including the character 'T' to distinguish time values from period lenghts and other temporal values, unsatisfactory from a semantic point of view. A clock time is conceptually similar to a date, in that it is a way of identifying a point in time. A clock time specification also has some internal structure, like a date, but the conceptual status of these components is different, since one can ask in what year, in what month, or on what day something occurs, but for an event occurring at 10:15 one cannot in a similar way ask in what hour or what minute the event occurs. We therefore do

not introduce separate attributes for the hour and minute parts of a clock time, but rather treat clock times as numerical values of an attribute clockTime that applies to instants. The attributes begin and end, which apply to intervals, also have these values. So the ICS-representation of (29) is:

(30) a. *4 p.m.*
b.
```
<isoTimeML_ICSrep xml:id="a1"/>
<INSTANT xml:id="t1" target="#token1 #token2"
clockTime="16:00"/>
</isoTimeML_ICSrep>
```

In practical situations, the temporal intervals and instants that may be referred to by specifying a date plus a clock time, are hardly ever fully specified. An example where they are is (32a), which would be annotated in ICS format as in (32c).

(31) a. *Monday March 16th 2007 at 10:15 a.m. CET*
b. token1 = 'Monday' token1='March' token3='16th' token4='2007' token5='at' token6='10:15' token7='a.m.' token8='CET'
c.
```
<isoTimeML_ICSrep xml:id="a1"/>
<INSTANT xml:id="t1" target="#range(token1,token8)"
timeZone="CET" calYear="2007" calMonth="MARCH"
dayName="MONDAY" dayNumber="16" clockTime="10:15"/>
</isoTimeML_ICSrep>
```

When describing a clock time, it is unusual to be explicit about a time zone, which is usually assumed to be clear from the context. This makes it attractive to have a separate attribute timeZone which can be left unspecified, rather then using an attribute which would have complex values consisting of a time on the clock *and* a time zone. Further support for factoring out time zones in a separate attribute comes from the consideration that dates are in fact also time zone-dependent, as we were reminded of when we watched the year 2000 millennium change on television occurring at different times in Hong Kong, Dubai, Paris, Rio de Janeiro, and Hawaii. Not only do both clock times and dates depend on the choice of a time zone, but in the precise specification of an 'instant', the date and the clock time should be taken according to the *same* time zone, hence it is best to consider an instant as defined by a triple <*time zone, date, clock time*>, as in the abstract syntax defined above.

Using this approach to the representation of the annotation of dates and times, the following example illustrates their use for an everyday example.

(32) a. *I will call you at two-thirty.*
b. token1='I' token2='will' token3='call' token4='you' token5='at' token6='two-thirty'
c.
```
<isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2 #token3" pred="CALL"
tense="FUTURE"/>
```

```
<INSTANT xml:id="t1" target="#token6" clockTime="14:30"/>
<TIME_ANCHORING anchoredEvent="#e1"
anchorTime="#t1" tempRel="AT"/>
</isoTimeML_ICSrep>
```

Besides the indication of a point in time by means of a date, a clock time, and a time zone, the abstract syntax also covers the annotation of cases like *"half an hour after midnight"*, where an instant is defined by its temporal distance and position relative to another instant or period. We will consider this case below, after we have considered the representation of temporal distances.

5.2 Representation of amounts of time

An amount of time measures the length of a time interval, or the accumulated length of a series of intervals (or, in other words, of an interval with holes). Like any measurement of length or size, temporal length measurement requires the choice of a unit and a numerical specification. Such units are for instance minute, second, and millisecond. Other units such as hour, day and year have a double life in natural language, in the sense that these words can be used to refer to certain intervals as well as to a unit of measurement (similar to *"cup"* and *"teaspoon"*, often used in cooking recipes for measuring volumes). This ambiguity, which is displayed by most of the words that are commonly used for measuring temporal length, often leads to confusion in annotation schemes, as we will see below.

A characteristic property of units of measurement is that the units that can be used in a given dimension are numerically interrelated. This can be expressed by means of a conversion function which specifies a numerical relation between the units, such as *Conversion(hour, minute) = 60*, thus explaining equivalences like 1.5 hour = 90 minutes and comparisons like 100 minutes is more than an hour. (See further Bunt, 1985 for a calculus of amounts.)

An example of the representation of amounts of time in ICS format is that of the expression *"eleven and a half hours"* in sentence (33a):

(33) a. *The flight to Hong Kong takes eleven and a half hours.*

c.
```
<ISOTimeML_ICSrep xml:id="s1">
<TIME-AMOUNT xml:id="a1" target="#token7 #token8 #token9
#token10 #token11" numeral="11.5" unit="HOUR"/>
</ISOTimeML_ICSrep>
```

We mentioned above that intervals and temporal distances are often confused; this is illustrated by the representation in ISO-TimeML of amounts of time, as in the following example:

(34) a. *no more than 60 days*

b.
```
<isoTimeML xml:id="a1">
<TIMEX3 xml:id="t1" target="#range(token1,token4)"
#token5" type="DURATION" value="P60D" mod="EQUAL_OR_LESS"/>
</isoTimeML>
```

This representation expresses that a temporal entity of type 'duration' is being annotated, which is correct, but then it says the 'value' of the duration is less than or equal to "P60D" which stands for 'period of 60 days'. But an amount of time *is not* a period; it is the *length of a period*.

In ICS format, by contrast, the annotation of this example is represented as shown in (35). This representation makes no reference to periods, and has its conceptual components marked as separate enitities – readily available for semantic processing.

(35) a. *no more than 60 days*
b.
```
<ISOTimeML_ICSrep xml:id="s1">
<TIME-AMOUNT xml:id="a1" target="#range(token1,token5)"
numRel="EQUAL_OR_LESS" numeral="60" unit="DAY"/>
</ISOTimeML_ICSrep>
```

The following example, from ISO 24617-1:2011, shows that the confusion between periods and their lengths is in fact more profound than in the value of the `value` attribute.

(36) a. *John taught for 20 minutes on Monday.*
b. token1='John' token2='taught' token3='for' token4='20' token5='minutes' token6='on' token7='Monday'
c.
```
<isoTimeML xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="TEACH"
type="PROCESS" class="OCCURRENCE" tense="PAST"/>
<SIGNAL xml:id="s1" target="#token3" pred="FOR"/>
<TIMEX3 xml:id="t1" target="#token4 #token5"
pred="20_MINUTES" type="DURATION" value="P20TM"/>
<SIGNAL xml:id="s2" target="#token6" pred="ON"/>
<TIMEX3 xml:id="t2" target="#token7" type="DATE"
value="xxxx-wxx-1"/>
<TLINK timeID="#t1" relatedToTime="#t1"
tempRel="IS_INCLUDED"/>
<TLINK eventID="#e1" relatedToTime="#t2"
tempRel="SIMULTANEOUS"/>
</isoTimeML>
```

This representation refers to a teaching event which is related to a duration of 20 minutes through a `TLINK` using the relation `SIMULTANEOUS`. Simultaneity, however, is a relation that may obtain between events, but not between an event and its duration. This problem is partly caused by the fact that ISO-TimeML, following TimeML, uses `TLINK` for several conceptually different relations: (1) for anchoring an event in time; (2) for expressing temporal relations between events; (3) for temporal relations between intervals and instants; (4) for relating an event to its duration; and (5) for relating a temporal interval to its length. By contrast, in the abstract syntax defined above and correspondingly also in the ideal concrete syntax we distinguish each of these types of relation, so that no such confusion can arise, as (37) illustrates.

(37)
```
<isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="TEACH" tense="PAST"/>
<TIME_AMOUNT xml:id="t1" target="#token4 #token5"
numeral="20" unit="MINUTES"/>
<DATE xml:id="d1" target="#token7" dayName="MONDAY"/>
<DURATION measuredEvent="#e1" tLength="#t1"/>
<TIME_ANCHORING anchoredEvent="#e1" anchorTime="#d1"
tempRel="INCLUDED_IN"/>
</isoTimeML_ICSrep>
```

5.3 Representation of time intervals

Time intervals can be described in natural language in a multitude of ways. The abstract syntax specified in Section 3.2 distinguishes four conceptually different types of interval identification: (1) by means of a predicate that names a particular period in history or that characterizes a certain type of period (and thus identifies a class of intervals rather than a single one); (2) by the name of an annually recurring event plus a calendar year and time zone; (3) by a begin- and end point; and (4) by means of a distance to a given instant or interval (like *"the two weeks up to Christmas"*. We consider each of these types in turn.

The identification of a unique temporal interval by means of a proper name is very simple, as illustrated by (38):

(38) a. *the Renaissance*

b.
```
<ISOTimeML_ICSrep xml:id="a1">
<PERIOD xml:id="t1" target="#token2" pred="RENAISSANCE"/>
</ISOTimeML_ICSrep>
```

The identification of a temporal interval by the name of an annually recurring event plus a year and a time zone is illustrated by (39), where the time zone is left implicit, as is commonly done in natural language, and with its interpretation in the form of DRS in (39c). Since the precise begin and end points of a year are different in different time zones, a time zone is required for the precise identification of an interval. While the specification of a time zone may be left out in natural language and hence also in the annotation, it must be assumed in the semantics; a time zone variable is therefore introduced in the DRS in (39c).

(39) a. *Easter 2011*

b.
```
<ISOTimeML_ICSrep xml:id="a1">
<PERIOD xml:id="t1" target="#token1 #token2"
pred="EASTER" year="2011"/>
</ISOTimeML_ICSrep>
```

c.

t1 z
xmas(t1) calyear(t1,z)=2011

The identification of a period by its begin-and end points is illustrated by (40). The ICS-representation of *"from nine to five"* in (40b) says in a very straightforward way that a period is considered which begins at nine o'clock a.m. and ends at five o'clock p.m.. The corresponding annotation structure has, according to the semantic rules laid out in Section 3.3, the DRS interpretation (40c), in which again a time zone variable is introduced for the precise interpretation of clock times.

(40) a. *from nine to five*

b.
```
<ISOTimeML_ICSrep xml:id="a1">
<PERIOD xml:id="t1" target="#range(token1,token4)"
begin="#t2" end="#t3"/>
<INSTANT xml:id="t2"target="#token2" clockTime="09:00"/>
<INSTANT xml:id="t3" target="#token4" clockTime="17:00"/>
</ISOTimeML_ICSrep>
```

c.

t1 t2 t3 z
clocktime(t2,z) = 09:00 begin(t1) = t2 clocktime(t3,z) = 17:00 end(t1) = t3

As mentioned above, an amount of time can be used to identify a time point relative to a given time point, or an time interval relative to a given date or time interval. Example (41) illustrates the ICS-treatment of such cases for the expression *"The three weeks before Christmas"*.

(41) a. *The three weeks before Christmas were very busy.*

b.
```
<ISOTimeML_ICSrep xml:id="a1">
<TIME_AMOUNT xml:id="a1" numeral="3" unit="WEEK"/>
<PERIOD xml:id="t1" target="#range(token1,token3)"
tLength="#a1" refTime="#t2" tempRel="IMMED_BEFORE"/>
<PERIOD xml:id="t2" pred="CHRISTMAS"/>
</ISOTimeML_ICSrep>
```

c.

t1 t2 z y
length(t1,week) = 3 immed-before(t1,t2) xmas(t2) calyear(t1,z) = y

A temporal distance can also be used in combination with a temporal relation in order to identify a time point; the ICS treatment of such cases is illustrated in (42). Note that an instant t1 is introduced which is located at a distance of 0.5 hours before midnight; the `CLOSE` event is temporally anchored at this instant.

(42) a. *They close half an hour after midnight*
b.
```
<isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="CLOSE"/>
<INSTANT xml:id="t1" target="#range(token3,token7)" distance="#a1"
anchorTime="24:00" tempRel="AFTER"/>
<TIME_AMOUNT xml:id="a1" target="#range(token3,token5)"
numeral="0.5" unit="HOUR"/>
<TIME_ANCHORING anchoredEvent="#e1" anchorTime="#t1"/>
</isoTimeML_ICSrep>
```

5.4 Representation of linking structures

The abstract syntax specified in Section 3.2 distinguishes seven kinds of links:

1. for anchoring events in time, as in *"John came home at midnight"* or in *"I have been teaching from nine to five today"*;
2. for temporally relating one event to another, as in *"He finalized the slides before going to bed"*;
3. for relating intervals and instants to each other, as in *"Chinese New Year is about four weeks after Christmas"*;
4. for measuring the duration of an event, as in *"I've been teaching eight hours"*;
5. for measuring the length of a temporal interval (not necessarily related to an event), as in *"Their summer vacation is six weeks long"*;
6. for subordination relations between events, as in *"You won't believe whom I ran into today"*;
7. for aspectual relations between events, as in *"starting to read"*.

For each of these seven types an XML-element has been introduced in the ideal concrete syntax. We have already seen examples of the `TIME_ANCHORING` element for anchoring an event in time, of the `ASPECTUAL_LINK` for relating events aspectually (see (23), and of the `DURATION` link for measuring the duration of an event (see (37). All the linking structures introduced in the abstract syntax are binary; therefore all these elements have two attributes which have pointer values, pointing to the representations of the entity structures which they relate, plus an attribute whose value specifies the relation that applies. The representation of the other types of linking structure is similar, all following the simple pattern:

(43)
```
<ELEMENT-NAME sourceAttribute=[pointer value]
targetAttribute=[pointer value] relationName=[relation]/>
```

5.5 Repetitions, frequencies, and quantification

All the examples considered so far concern the annotation of information about a single event. In natural language we also encounter cases of information relating to multiple events or to repetitions of events, as in *"John called twice"*. Using the ISO-TimeML representation format, the annotation of this sentence would be represented as follows:

(44)
```
<isoTimeML xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="CALL"
type="PROCESS" class="OCCURRENCE" tense="PAST"/>
<TIMEX3 xml:id="t1" target="#token3" freq="2X"/>
<TLINK eventID="#e1" relatedToTime="#t1" tempRel= "DURING"/>
</isoTimeML>
```

This representation is problematic in several respects. First, the `EVENT` part refers to a single event `e1`, which is temporally linked to the temporal entity *twice*, while it seems evident that two events are considered. Second, what kind of entity is 'twice'? As mentioned above, ISO-TimeML uses the `TIMEX3` tag for all temporal entities, and distinguishes among these entities by means of the `type` attribute into dates, times, periods, durations, and sets of these. But *twice* is neither of these types. In fact, the word *"twice"* does not denote a temporal entity at all; it is rather a name of a counter, which expresses how often a certain type of event occurred; it doesn't provide any *temporal* information.

The abstract syntax specifies as one of the components of an entity structure the set-theoretical type which captures whether the structure describes individual objects, sets of individual objects, or higher-order entities. In the ideal concrete syntax this is represented by the attribute `signature`, with such values as `INDIVIDUAL` and `SET` for events as well as for temporal objects, thus allowing the representation of sets of events, sets of temporal intervals, sets of instants, sets of amounts of time, and so on. Together with the cardinality component which event structures also have, represented in ICS format by the `cardinality` attribute and its values for indicating the number of elements in a set, this allows the temporal and event-related information in the sentence *"John called twice"* to be represented in a simple and transparent way as follows:

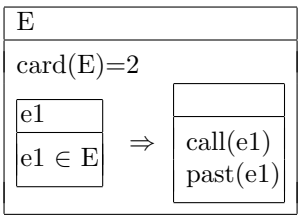
(45) a. *John called twice*

b. token1='John' token2='called' token3='twice'

c.
```
isoTimeML_ICSrep xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="CALL"
tense="PAST" signature="SET" cardinality="2"/>
</isoTimeML_ICSrep>
```

This representation says, simply, that two `CALL` events occurred. Because the `signature` attribute has the value `SET`, the annotation does not refer to a single event but to a set of events. This is also reflected in the semantics through the interpretation of the annotation structure as a DRS which contains a discourse referent denoting a set of events:

(46)



This interpretation says that there is a set E of events, containing two members, and that these are both 'call' events that happened in the past.

Expressions such as *"Only once", "twice", "thrice"*, and *"four times"*, denote in some contexts the number of times a certain type of event recurred, as in (44), and denote in other contexts a frequency, i.e. the number of times something occurs within a certain amount of time, like *"twice a day"*. In (ISO-)TimeML, the attribute `freq` is used for both, and this creates conceptual and formal problems in the interpretation of the annotations, because the number of times an event type recurs is just a number, while a frequency consists semantically of two components: a number and an amount of time. The (ISO-)TimeML representation in (44) is incorrect in this respect.

Repetitions and frequency of occurrence of events are closely related to event quantification. Quantification in natural language is the phenomenon that a predicate is applied to each or some of the members of a set, or collectively to the set as a whole, or to subsets of it. This predicate may be unary, as in (47a), in which case it applies to (members or subsets of) a single set, or it may have a higher arity, as in (47b-d), relating the members of one set to one or more other sets.

(47) a. These books are heavy.
b. The students have to read five papers.
c. The men moved the pianos.
d. The boys gave the girls some of the sweets.

In (47a), the predicate *heavy* can be understood as applying to the individual members of a certain set of books, or collectively to the set of books as a whole. Similar considerations apply yo the other examples. This is called the *distributivity* of the quantification (Bunt, 1985).

When verbs are viewed as referring to events, as in ISO-TimeML, then *every* sentence displays the phenomenon of quantification. Consider for instance the following sentence:

(48) Everybody will die.

This sentence can be read either as expressing that for each person there will be an event where this person dies, or that there will be an (apocalyptic) event in which everyone will die. So even an intransitive verb gives rise to quantification, and thus to issues such as collective versus individual involvement and relative scoping.[6]

ISO-TimeML does not consider the relations between events and their participants, therefore issues of relative scoping and collectiveness might seem not to arise. However, in principle *any* relation between two sets of entities is quantified in a certain way, and so are the relations between events and temporal entities, for instance by means of natural language temporal quantifiers such as *"always", "sometimes", "every Monday"*, so we do need some provisions to

[6] See also Bunt (2005) for an analysis of quantification in terms of feature structures.

represent time- and event-related quantification. ISO-TimeML has the attribute `quant` for this purpose, as one of the attributes of temporal entities, but this is not satisfactory, for similar reasons as why frequencies should not be attributes of temporal entities.[7]

Quantifications, or rather, the properties of a quantification such as the distributivities of the sets of participants involved, are aspects of *relations*, such as the temporal anchoring relation between a set of events and a set of time intervals. We therefore introduce a number of attributes in the ICS representation of link structures, allowing us to annotate quantificational properties. For instance, a `TIME_ANCHORING` element will have attributes `eventDistr` and `timeDistr`, with values like `INDIVIDUAL` and `COLLECTIVE` for indicating the distributivity on either side of the relation.

This is illustrated in the ICS representation (49b) of the quantification in *"John calls every day"*, with its representation in ISO-TimeML in (49c) for contrast. We invite the reader to inspect the two representations for their differences, and enjoy the transparency and conceptual clarity of the ICS representation.

(49) a. *John calls every day.*

b.
```
<isoTimeML_ICSrep>
<EVENT xml:id="e1" target="#token2" pred="CALL"
signature="SET"/>
<PERIOD xml:id="t1" target="#token4" pred="DAY"
signature="SET"/>
<TIME_ANCHORING anchoredEvent="#e1" anchorTime="#t1"
tempRel="INCLUDED_IN" eventDistr="INDIVIDUAL"
timeDistr="INDIVIDUAL" timeQuant=EVERY"/>
</ISOTimeML_iCSrep>
```

c.
```
<isoTimeML xml:id="a1">
<EVENT xml:id="e1" target="#token2" pred="CALL"
type="PROCESS" class="OCCURRENCE"/>
<TIMEX3 xml:id="t1" target="#range(token3,token4)"
type="SET" value="P1D" quant="EVERY"/>
<TLINK eventID="#e1" relatedToTime="#t1"
relType="IS_INCLUDED"/>
</ISOTimeML>
```

It was noted above that a description of recurring events with a certain frequency, as in *"John calls twice every day"*, forms in fact a case of event quantification. The ICS representation of such a sentence, viewed in this way, is shown in (50).

(50) a. *John calls twice every day*

b.
```
<isoTimeML_ICSrep>
<EVENT xml:id="e1" signature="SET"/>
```

[7] See also Bunt (2010) and Bunt & Pustejovsky (2010) for more discussion about event quantification and ISO-TimeML.

```
<PERIOD xml:id="t1" pred="DAY" signature="SET"/>
<TIME_ANCHORING anchoredEvent="#e1" anchorTime="#t1"
tempRel="INCLUDED_IN" eventDistr="INDIVIDUAL"
timeDistr="INDIVIDUAL" eventQuant="2" timeQuant=EVERY"/>
</isoTimeML_ICSrep>
```

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 Conclusions

This paper brings three innovations to the definition of semantic annotation languages. First, we have added a third component to the usual two components of a language definition: besides a syntax, that specifies the set of expressions of the language, and a semantics that specifies for each expression what it means, we have introduced an *abstract syntax* which 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 particularly important in the context of defining annotation standards, since according to the Linguistic Annotation Framework (Ide & Romary, 2004), standards should be defined at the level annotations, 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. The meanings of these annotation structures were shown to be derivable in a straightforward manner by keeping track in the DRSs of the information in the annotations concerning their anchoring (through markable structures) to the source text.

Third, we have introduced the notion of an ideal concrete syntax, being a specification of a 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 uniquely determined annotation structure. If F_1 is a representation format that has the property of being ideal, then the meaning of any representation ϵ_1 is the meaning defined by the semantics for the uniquely determined annotation structure ϵ', which is represented according to F_1 by ϵ_1. Therefore, if F_2 is another ideal representation format, then an expression ϵ_2 of F_2 is semantically equivalent to expression ϵ_1 of F_1 iff ϵ' is also the uniquely determined annotation structure represented by ϵ_2 in F_2. There is

then also provably a systematic procedure for converting F_1-representations into F_2-representations and vice versa. (See further Bunt, 2010.)

Is an ideal representation format ideal in the sense of not 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, or both.

Besides being theoretically important, ideal representation formats have the advantages of being maximally simple, accurately representing the conceptual distinctions made in the abstract syntax, and being optimally transparent, allowing a simple semantic interpretation. We have illustrated this for the annotation of documents with information about time and events, as when using the TimeML annotation language (Pustejovsky et al., 2005; 2007) and as undertaken in a project carried out by the International Organization for Standardization ISO. As a side-effect of applying our methodology to the latter, we have noted a number of points where the ISO-TimeMl proposal may be improved. This shows that the approach can be of practical value for the design of annotation languages.

We have also observed that 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 direct handle on general issues in semantic annotation such as quantification. We have indicated a direction for how to capture quantificational aspects in the annotation of time and events by viewing quantification as the description of properties of a relation between sets of entities, allowing us to also annotate descriptions in natural language of recurring events and frequencies of events in a more satisfactory way than has so far been proposed. This is potentially interesting not only for the annotation of timeand event-related information, but more generally for annotating expressions of quantification in natural language.

Acknowledgements

I would like to thank James Pustejovsky, project leader of the ISO-TimeML project, and Kiyong Lee, co-project leader, for many in-depth discussions of issues in the temporal and event-related semantics of natural languages.

References

1. Allen, J. (1984) A General Model of Action and Time. *Artificial Intelligence* 23 (2), 123–154.

2. Bach, E. (1986) The Algebra of Events. *Linguistics and Philosophy* 9, 5–16.
3. Bunt, H. (1985) *Mass terms and model-theoretic semantics.* Cambridge, UK: Cambridge University Press.
4. Bunt, H. (2005) Quantificationand modification represented as feature structures . In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6)*, Tilburg, January 2005, pp. 54-65.
5. Bunt, H. (2007) The Semantics of Semantic Annotation. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation (PACLIC-21)*, Seoul, November 2007, pp. 13-28.
6. Bunt, H. (2010) A methodology for designing semantic annotation languages exploiting syntacic-semantic iso-morprhisms. In *Proceedings of the Second Conference on Global Interoperability in Language Resources (ICGL 2010)*, Hong Kong, January 2010. Available on http://let.uvt.nl/general/people/bunt/
7. Bunt, H. and J. Pustejovsky (2010) Annotating temporal and event quantification. In *Proceedings of ISA-5, the Fifth Joint ISO-SIGSEM Workshop on Interoperable Semantic Annotation*, Hong Kong, January 2010. Available on http://let.uvt.nl/general/people/bunt/
8. Bunt, J. and C. Overbeeke (2008) An Extensible Compositional Semantics for Temporal Annotation. In N. Ide et al. (eds.) *Proceedings of LAW II, the Second Linguistic Annotation Workshop,* at the Sixth Conference on Language Resources and Evaluation (LREC 2008), Marrakech.
9. Bunt, H. and L. Romary (2002). Towards multimodal content representation. In Proceedings LREC 2002 Workshop on International Standards of Terminology and Language Resources, Las Palmas, May 2002. ELRA, Paris, pp. 54-60..
10. Hobbs, J. and F. Pan (2004) An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Processing (TALIP), Special Issue on Temporal Information Processing*, Vol. 3 No. 1, 66 – 85.
11. Ide, N. and L. Romary (2004). International standard for a linguistic annotation framework. *Journal of natural Language Engineering*, 10:3-4, 211-225.
12. ISO (2009) ISO DIS 24617::2009 *Language resources management - Semantic annotation framework, Part 1: time and events.* ISO, Geneva.
13. ISO (2011) ISO 24617-2:2011 Language resources management - Semantic annotation framework, Part 2: Dialogue acts. ISO, Geneva.
14. ISO (2011) ISO 24612:2011, *Language resources management - Linguistic annotation framework.* ISO, Geneva.
15. ISO (2011) ISO 24617-1:2011 Language resources management - Semantic annotation framework, Part 1: Time and events. ISO, Geneva.
16. Kamp, H. and U. Reyle (1993) *From Discourse to Logic.* Dordrecht: Kluwer.
17. Katz, G. (2007) Towards a denotational semantics for TimeML. In I. Mani, J. Pustejovsky, and R. Gaizauskas (eds.) *The Language of Time: A Reader.* Oxford: Oxford University Press, pp. 88–106.
18. Lee, K. (2008) Formal Semantics for Interpreting Temporal Annotation. In P. van Sterkenburg (ed.) *Unity and Diversity of Languages: Special Lectures for CIL18, the 18th International Congress of Linguists.* Amsterdam: John Benjamins, pp. 97–108.
19. Mani, I., J. Pustejovsky, and R. Gaizauskas (eds.) (2005) *The Language of Time: A Reader.* Oxford: Oxford University Press.
20. Pratt-Hartmann, I. (2007) From TimeML to Interval Temporal Logic. In H. Bunt, G. Geertzen, and V. Petukhova (eds.) *Proceedings of the 7th International Workshop on Computational Semantics IWCS-7*, 166–180.

21. Prior, A.N. (1967) *Past, Present, and Future*. Oxford: Clarendon Pres..
22. Pustejovsky, J. R. Ingria, R. Saurí, J. Gastano, J. Litman, R. Gaizauskas, A. Setzer, G. Katz, and Chr. Habel (2005) The Specification Language TimeML. In I. Mani, J. Pustejovsky, and R. Gaizauskas (eds.) *The Language of Time: A Reader*. Oxford: Oxford University Press.
23. Pustejovsky, J., R. Knippen, J. Litman, and R. Saurí (2007) Temporal and event information in natural language text. In H. Bunt and R. Muskens (eds) *Computing Meaning, Vol 3*. Dordrecht: Springer, pp. 301– 346.
24. Pustejovsky, J,, H. Bunt, K. Lee and L. Romary (2010) ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of LREC 2010*, Malta.
25. Vendler, Z. (1967) Verbs and times. In: Z. Vendler, *Linguistics in Philosophy*, Ithaca: Cornell University Press, pp. 97-121.